English

Fujitsu Software

BS2000 OS DX
DMS Macros

User Guide

Valid for:
BS2000 V21.0A

June 2021

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to: bs2000.info@fujitsu.com.

Certified documentation according to DIN EN ISO 9001:2015

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2015.

Copyright and Trademarks

Copyright © 2025 Fujitsu

All rights reserved.
Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

Table of Contents

BS2000 V21 DMS Macros (en) Final Home 7
1 DMS Macros 8
1.1 Preface 9
1.1.1 Objectives and target groups of this manual 10
1.1.2 Summary of contents 11
1.1.3 Changes since the last edition of the manual 12
1.1.4 Notational conventions 13
1.2 Overview of DMS macros 14
1.2.1 Table of DMS macros (in alphabetical order) 15
1.2.2 DMS macros in order of function 18
1.2.2.1 File maintenance 19
1.2.2.2 Controlling file processing 20
1.2.2.3 Data protection/security support 21
1.2.2.4 Device and volume management 23
1.2.2.5 Access to files 24
1.2.2.6 Generation of operand lists for control blocks, DMS tables, etc. 29
1.2.2.7 Output of information on files, volumes, devices, etc. 32
1.2.3 Comparison of macros and commands 33
1.3 Programming notes 36
1.3.1 BTAM - Basic Tape Access Method 37
1.3.1.1 OPEN modes 39
1.3.1.2 BTAM record and block formats 41
1.3.2 DIV - Data In Virtual 42
1.3.2.1 Opening a file 43
1.3.2.2 Defining windows 47
1.3.2.3 Writing data back to the disk file 48
1.3.2.4 Undoing modifications in a window 49
1.3.2.5 Disabling a window 50
1.3.2.6 Closing a file 51
1.3.3 EAM - Evanescent Access Method 52
1.3.3.1 MFCB (Mini File Control Block) 53
1.3.3.2 EAM processing 60
1.3.4 FASTPAM - Fast Primary Access Method 64
1.3.4.1 FASTPAM functions 65
1.3.4.2 Processing files with FASTPAM 68
1.3.5 ISAM - Indexed-Sequential Access Method 75
1.3.5.1 OPEN modes 77

1.3.5.2 ISAM pointers 80
1.3.6 SAM - Sequential Access Method 83
1.3.6.1 OPEN modes 84
1.3.7 UPAM - User Primary Access Method 90
1.3.7.1 OPEN modes 93
1.3.7.2 UPAM for disk files 97
1.3.7.3 UPAM processing of tape files 101
1.3.7.4 Chaining PAM macros in list form 103
1.3.7.5 TU eventing: event-driven processing 105
1.3.8 Files larger than 32 GB 108
1.4 Macros 111
1.4.1 ADDPLNK - Define pool link name 114
1.4.2 BTAM - Process tape files (type S) 117
1.4.3 CATAL - Process catalog entry 128
1.4.3.1 Version differences - VERSION=0/1/2/3/4 194
1.4.4 CHKFAR - Check file access rights 198
1.4.5 CHNGE - Change TFT entry 205
1.4.6 CLOSE - Close file 206
1.4.7 COMPFIL - Compare disk files 209
1.4.8 COPFILE - Copy file 217
1.4.9 CREAIX - Create secondary keys for ISAM file 231
1.4.10 CREPOOL - Create ISAM pool 240
1.4.11 DECFILE - Convert encrypted file into unencrypted file 246
1.4.12 DELAIX - Delete secondary key of ISAM file 249
1.4.13 DELPOOL - Delete/release ISAM pool 253
1.4.14 DIV - Access files via virtual address space 256
1.4.14.1 DIV function: OPEN 259
1.4.14.2 DIV function: MAP 265
1.4.14.3 DIV function: SAVE 270
1.4.14.4 DIV function: RESET 274
1.4.14.5 DIV function: UNMAP 278
1.4.14.6 DIV function: CLOSE 281
1.4.15 DROPTFT - Release TFT entry 294
1.4.16 EAM - Process EAM files 296
1.4.17 ELIM - Eliminate record 297
1.4.18 ENCFILE - Convert unencrypted file into encrypted filen 299
1.4.19 ERASE - Erase files 303
1.4.19.1 Variations in versions - VERSION=0/1/2 359
1.4.20 EXLST - Define exit address list 365
1.4.21 EXRTN - Return from error routine 381
1.4.22 FCB - Define file control block 383

1.4.23 FCBAD - Create FCB addresses 423
1.4.24 FEOV - Close tape 424
1.4.25 FILE - Define file attributes / control file processing 426
1.4.25.1 Variations in VERSION=0/1/2/3 486
1.4.26 FILELST - Create variable operand areas for FILE macros 491
1.4.27 FPAMACC - Access FASTPAM files 494
1.4.28 FPAMSRV - FASTPAM management function 511
1.4.28.1 FASTPAM function: ENABLE ENVIRONMENT 514
1.4.28.2 FASTPAM function: ENABLE IOAREA POOL 523
1.4.28.3 FASTPAM function: OPEN 529
1.4.28.4 FASTPAM function: CLOSE 537
1.4.28.5 FASTPAM function: DISABLE IOAREA POOL 540
1.4.28.6 FASTPAM function: DISABLE ENVIRONMENT 543
1.4.29 FSTAT - Request catalog information 556
1.4.29.1 Programming notes for VERSION=4 595
1.4.29.2 Programming notes (VERSION=2, 3 and 4) 596
1.4.29.3 Programming notes for VERSION=0 and VERSION=1 605
1.4.29.4 Version differences - VERSION=0/1/2/3/4/5 608
1.4.29.5 Version variations in the representation of the output area 615
1.4.30 GET - Read next record 619
1.4.31 GETFL - Read record by flag 623
1.4.32 GETKY - Get record with specified key 630
1.4.33 GETR - Get record "reverse" 632
1.4.34 IDBPL - Provide BTAM operand list with symbolic names 635
1.4.35 IDFCB - Provide FCB with symbolic names 636
1.4.36 IDFCBE - Provide FCBE with symbolic names 637
1.4.37 IDPPL - Provide PAM operand list with symbolic names 638
1.4.38 IMPNFIL - Create (import) catalog entries for node files 639
1.4.39 IMPORT - Create catalog entry for files 647
1.4.40 INSRT - Insert record 653
1.4.41 ISREQ - Unlock data block 655
1.4.42 LBRET - Return from user label routine 657
1.4.43 LFFSNAP - List files from a Snapset 659
1.4.44 LJFSNAP - List job variables from a Snapset 667
1.4.45 MAILFIL - Send file by email 675
1.4.46 NDWERINF - Evaluate status bytes 683
1.4.47 OPEN - Open file 684
1.4.48 OSTAT - Request information on open files 688
1.4.49 PAM - Perform UPAM actions 690
1.4.50 PUT - Write record 700
1.4.51 PUTX - Replace record 703

1.4.52 RDTFT - Read TFT and TST information 705
1.4.53 RELSE - Close block 712
1.4.54 RELTFT - Delete TFT entry 714
1.4.55 REMPLNK - Delete pool link name 718
1.4.56 RETRY - Repeat macro 720
1.4.57 RFFSNAP- Restore files from Snapset 722
1.4.58 RJFSNAP- Restore job variables from a Snapset 733
1.4.59 SETL - Position file pointer 742
1.4.60 SHOPLNK - Return information on ISAM pool link names 747
1.4.61 SHOPOOL - Return information on ISAM pools 757
1.4.62 SHOWAIX - Request information on secondary keys 771
1.4.63 STORE - Store record 776
1.4.64 VERIF - Recover file 778
1.5 Appendix 783
1.5.1 Syntax presentation 784
1.5.1.1 Macro format 785
1.5.1.2 Metasyntax used for the macros 786
1.5.1.3 Wildcards 789
1.5.1.4 Format of date specifications 790
1.5.1.5 Macro types 791
1.5.1.6 Standard header 795
1.5.2 DMS error codes 797
1.5.3 CALL interface for DIV 802
1.5.4 Labels 805
1.5.4.1 Volume header labels 806
1.5.4.2 User volume header labels (UVL1 through UVL9) 808
1.5.4.3 File header labels (HDR1 through HDR9) 809
1.5.4.4 User file header labels (UHL) 814
1.5.4.5 End-of-volume labels (EOV1 through EOV9) 815
1.5.4.6 End-of-file labels (EOF1 through EOF9) 817
1.5.4.7 User file trailer labels (UTL) 819
1.5.4.8 Processing of label fields 820
1.5.5 DMS dummy sections (DSECTs) 823
1.5.6 Formats of replaced macros 825
1.5.6.1 COPY - Copy file 826
1.5.6.2 REL - Delete TFT entry 827
1.6 Glossary 828
1.7 Related publications 835

BS2000 V21 DMS Macros (en) Final Home

Welcome to your new space!

Confluence spaces are great for sharing content and news with your team. This is your home page. Right now it shows recent space activity, but you can customize this page in any way you like.

Complete these tasks to get started

- **Edit this home page** - Click *Edit* in the top right of this screen to customize your Space home page
- **Create your first page** - Click the *Create* button in the header to get started
- **Brand your Space** - Click *Configure Sidebar* in the left panel to update space details and logo
- **Set permissions** - Click *Space Tools* in the left sidebar to update permissions and give others access

Recent space activity

Ohme, Axel

DMS Macros (BS2000 V21.0A DMS Macros 2021-06, #1) updated less than a minute ago view change

Related publications (BS2000 V21.0A DMS Macros 2021-06, #159) updated Jun 18, 2021 view change

Changes since the last edition of the manual (BS2000 V21.0A DMS Macros 2021-06, #5) updated Jun 18, 2021 view change

Anonymous

Related publications (BS2000 V21.0A DMS Macros 2021-06, #159) updated Jun 11, 2021 view change

Glossary (BS2000 V21.0A DMS Macros 2021-06, #158) updated Jun 11, 2021 view change

Space contributors

- Ohme, Axel (less than a minute ago)

DMS Macros

Preface

The Data Management System (DMS) of BS2000 and its functions are described in the manual “Introductory Guide to DMS“ [1].

This manual describes the macros of the BS2000 Data Management System.

Objectives and target groups of this manual

This manual is intended for users who wish to manage or process their files with the aid of the Assembler programming interface or to control data and file protection, data security, etc. with the aid of the DMS macros.

Summary of contents

Chapter 2: Overview of DMS macros

This part of the manual lists all DMS macros in alphabetical order and by function.

Chapter 3: Programming notes

This chapter lists the individual access methods used by DMS. It also lists the macros and operands available for the relevant access methods. This section focuses on features of individual access methods which are relevant to programming.

Chapter 4: Macros

This part of the manual is intended for use as a reference section. It contains descriptions of functions, syntax and operands for DMS macros. All macros are listed in alphabetical order.

The appendix contains several tables and lists referred to at various places in this manual. It includes metasyntax and tables showing label formats for tape files, DMS error messages, and a list of device and volume types.

The manual concludes with a glossary, a list of related publications, and an index.

Notes on using this manual

References to other manuals in the text are generally in the form of abbreviated titles together with a reference number. The Related publications section contains the full titles.

A general description of the DMS functions can be found in the manual “Introductory Guide to DMS” [1].

Additional product information

Current information, version and hardware dependencies, and instructions for installing and using a product version are contained in the associated Release Notice. These Release Notices are available online at *http://bs2manuals.ts.fujitsu.com*.

Changes since the last edition of the manual

The manual “DMS Macros” was last published for BS2000 OSD/BC V11.0. The following major changes have been made since the last edition of the manual:

A new Net-Storage volume type NETVOL is introduced with BS2000 V21.0A.

Fundamental information on the use of Net-Storage in BS2000 is provided in the “Introduction to System Administration” [7]. How to work with BS2000 files and node files on Net-Storage is described in the “Introductory Guide to DMS” [1].

The device and volume type tables are contained in the "System Installation" manual [16]. Information on the volume types of DMS (Net-Storage and tape processing is provided in the “Commands” manual [3].

USER-PFA is no longer supported.

Notational conventions

The following typographical elements are used in this manual:

i	For notes on particularly important information
!	For special information that points out the possibility that data can be lost or that other serious damage may occur.
[ ]	References to publications are shown in the text by abbreviated titles. The complete title of each document which is referred to by a number is listed in the Related publications section after the corresponding number.
`input`	Inputs and system outputs in examples are shown in `typewriter font`.

Information on syntax display of the macros can be found in the appendix (chapter "Syntax presentation").

Overview of DMS macros

- Table of DMS macros (in alphabetical order)
- DMS macros in order of function
  - File maintenance
  - Controlling file processing
  - Data protection/security support
  - Device and volume management
  - Access to files
  - Generation of operand lists for control blocks, DMS tables, etc.
  - Output of information on files, volumes, devices, etc.
- Comparison of macros and commands

Table of DMS macros (in alphabetical order)

Macro	Brief description
ADDPLNK	*ISAM:* define pool link name
BTAM	controls all BTAM actions
CATAL	process catalog entry
CHKFAR	check access rights to file
CHNGE	change TFT entry
CLOSE	close file
COMPFIL	compare two disk files
COPFILE	copy file
CREAIX	*ISAM:* create secondary key for ISAM file
CREPOOL	*ISAM:* create ISAM pool
DECFILE	convert encrypted file into unencrypted file
DELAIX	*ISAM:* delete secondary key of ISAM file
DELPOOL	*ISAM:* delete/release ISAM pool
DIV	access file via virtual address space
DROPTFT	release TFT entry
EAM	macro (type R)
ELIM	*ISAM:* delete record
ENCFILE	convert unencrypted file into encrypted file
ERASE	delete files
EXLST	specify exit address list (type O)
EXRTN	return from error routines (type R)
FCB	define file control block (type O)
FCBAD	create FCB addresses (type O)
FEOV	*BTAM/SAM:* close volume
FILE	define file characteristics / control file processing
FILELST	create variable operand areas for FILE macro

FPAMACC	*FASTPAM:* formulate FASTPAM file accesses
FPAMSRV	*FASTPAM:* formulate FASTPAM management calls
FSTAT	request catalog information
GET	*ISAM/SAM:* read next record
GETFL	*ISAM:* read record after flag
GETKY	*ISAM:* read record with specified key
GETR	*ISAM:* sequential “reverse” read
IDBPL	*BTAM:* BTAM operand list (type O)
IDFCB	provide FCB with symbolic names (type O)
IDFCBE	provide FCBE with symbolic names (type O)
IDMCB	provide MFCB (EAM control block with symbolic name)
IDPPL	*UPAM:* PAM operand list
IMPNFIL	create (import) catalog entries for node files
IMPORT	create (import) catalog entries for files
INSRT	*ISAM:* insert record (type R)
ISREQ	*ISAM:* clear lock (type O)
LBRET	return from user label routine (type R)
LFFSNAP	list files from a Snapset
LJFSNAP	list job variables from a Snapset
MAILFIL	send file to a user ID by email
NDWERINF	*BTAM:* interrogate status bytes
OPEN	open file (type R)
OSTAT	*ISAM:* information about opened files (type R)
PAM	*UPAM:* perform UPAM actions
PUT	*ISAM:* write record
PUTX	*ISAM/SAM:* replace record
RDTFT	information from TFT and TST
RELTFT	delete TFT entry
REMPLNK	*ISAM:* delete pool link name

RETRY	*ISAM:* repeat macro
RELSE	close block
RFFSNAP	restore files from a Snapset
RJFSNAP	restore job variables from a Snapset
SETL	*ISAM/SAM:* position in file
SHOPLNK	*ISAM:* output information on ISAM pool link name
SHOPOOL	*ISAM:* output information on ISAM pool
SHOWAIX	*ISAM:* output information on secondary key
STORE	*ISAM:* store record
VERIF	restore file

DMS macros in order of function

The following tables provide an overview of the functions of the macros described in this manual.

Further details can be found in the descriptions of the macros and operands in this manual or in the appropriate introductory chapters in the “Introductory Guide to DMS” [1].

File maintenance

File maintenance includes not only creating, copying, deleting and restoring files, but also maintenance of the file catalog by the user.

Macro	Operands	Brief description
CATAL		Creates or updates catalog entries.
COMPFIL		Compares two disk files.
COPFILE		Copies files.
ERASE		Erases/exports files.
FILE		Creates a catalog entry and reserves storage space for noncataloged files.
	SPACE	Reserves or releases storage space.
	STATE	Creates catalog entries for files on private disk.
FSTAT		Displays information from the file catalog.
IMPNFIL		Creates (imports) catalog entries for node files
IMPORT		Creates (imports) catalog entries for files
LFFSNAP LJFSNAP		Lists files from a Snapset Lists job variables from a Snapset
MAILFIL		Sends a file to a user ID by email
RFFSNAP RJFSNAP		Restores files from a Snapset Restores job variables from a Snapset
VERIF		Restores corrupted files.

Controlling file processing

To enable a file to be processed by a program, it must be possible to set up a connection between the two. This connection can either be defined in the FCB or, if the program uses an internal file name (the file link name), be established using the FILE macro. The connection is stored together with other information in the task file table (TFT). File processing is thus controlled via the TFT.

For NK-ISAM files, file processing control also incorporates the management of user ISAM pools, in which these files are processed. The user can also use standard ISAM pools of the system, but then has no influence on pool size or reservation.

Macro	Operands	Brief description
ADDPLNK		assigns a pool link name to a user ISAM pool.
CHNGE		assigns a new file link name to a file.
CREAIX		creates a secondary key for an ISAM file.
CREPOOL		creates a user ISAM pool.
DELAIX		deletes secondary indices of an ISAM file.
DELPOOL		deletes a user ISAM pool.
FCB	FILE	defines a fixed link between file and program.
	LINK	defines a file link name in the program.
	POOLLNK	sets up a connection to a user ISAM pool.
FILE	LINK	creates a TFT entry; further operands describe the file and processing attributes.
	BLKCTRL	defines the file format.
	NFTYPE	defines the file type of a file on Net-Storage (BS2000 file or node file).
	POOLLNK	sets up the link to the ISAM pool.
RDTFT		displays TFT information.
RELTFT		deletes a TFT entry.
REMPLINK		deletes a pool link name.
SHOPLNK		shows assignments of pool link names to ISAM pools.
SHOPOOL		returns information on the attributes and occupancy of ISAM pools.

Data protection/security support

The mechanisms for file and data protection automatically supported by DMS (access authorization checks, etc.) can be extended by the user, e.g. with the aid of passwords. Data security is assured by various mechanisms for recovering files or programs, etc.

File protection

Macro	Operands	Brief description
CATAL	SHARE	controls shareability.
	ACCESS	controls the type of access.
	OWNERAR GROUPAR OTHERAR	defines, in the BASIC-ACL,the access rights for user groups (see the “Introductory Guide to DMS” [1]).
	GUARDS	when SECOS is used: provides enhanced access protection for files.
	EXPASS RDPASS WRPASS	define passwords for the various access levels.
	RETPD	specifies a retention period.
	PROTECT	transfers protection attributes
CHKFAR		checks the caller's file access rights.
COPFILE	PROTECT	transfers the protection attributes when a file is copied.
DECFILE		converts an encrypted file into an unencrypted file.
ENCFILE		converts an unencrypted file into an encrypted file.
FILE	RETPD	defines a retention period (valid only if specified when the file is opened)
FCB	PASS	permits access to password-protected files.
	RETPD	defines a retention period for a file.

Data protection

Macro	Operands	Brief description
CATAL	DESTROY	specifies in the catalog entry that disk files or superfluous foreign data on tape files are to be overwritten during deletion (cf. FILE: DESTOC).
ERASE	DESTROY	specifies data destruction in conjunction with deletion.
DECFILE		converts an encrypted file into an unencrypted file.
ENCFILE		converts an unencrypted file into an encrypted file.
FILE	DESTOC	specifies that any data remaining on the tape in the case of a tape swap or after closing a tape file is overwritten.

Data security

Macro	Operands	Brief description
CATAL	BACKUP	specifies the frequency of automatic saving.
	NUM_OF_BACKUP_VERS	defines, if and how many versions of a file can be saved in an version backup archive.
COPFILE	REPLACE	specifies whether an existing file is to be overwritten during copying.
CREPOOL	WROUT	writes updated blocks in ISAM files back to disk immediately.
EXLST		defines exit routines for errors and other events.
FCB	EXIT	the address of an exit routine or of an EXLST macro.
	WRCHK	specifies a read-after-write check as a safeguard against recording errors.
FILE	WRCHK	performs a read-after-write check as a safeguard against recording errors.
	WROUT	writes updated records in ISAM files back to disk immediately.
VERIF		restores file structures, unlocks files.
WRCPT		writes a checkpoint / creates a checkpoint file for restart with the RESTART-PROGRAM command.

Device and volume management

DMS supports users in the processing of files on private volumes by making it possible for them to reserve volumes and devices for their jobs.

Macro	Operands	Brief description
FILE	DEVICE VOLUME	Defines the device type and volumes for a file on private disks or on Net-Storage volumes.
	MOUNT	Issues a mount request at the console for private disks.
IMPORT		Creates (imports) catalog entries for files
RELTFT		Deletes TFT entries and implicitly releases devices

Access to files

Access to files is executed by calling action macros for the various access methods. DMS also handles the opening and closing of files (OPEN/CLOSE processing) as a function of the access method involved.

DMS macros for file processing (“service macros”)

The DMS macros for file processing (i.e. the “service macros”) are macro calls which are valid for all access methods.

Macro	Brief description
CLOSE	closes one or more files.
EXLST	defines error exits.
EXRTN	implements a return from EXLST routines.
FCB	creates a file control block (FCB).
FCBAD	creates the FCB in the literal pool of a program.
LBRET	implements a return from user label handling routines (tape processing).
OPEN	opens a file.

Macros specific to the access methods

A distinction is made between the following access methods:

- BTAM
- DIV
- EAM
- FASTPAM
- SAM
- ISAM
- UPAM

Access takes place in the familar way for encrypted files (with plain text content). Decryption when reading and encryption when writing are performed internally and automatically. The macros for file access in a program do not need to be modified for this purpose. Before a file is opened it is only necessary to enter the associated crypto password in the crypto password table of the accessing task. This can be done before the program is started.

In the tables below, the macros used for file access are assigned to the various access methods.

BTAM (Basic Tape Access Method)

BTAM is an access method for block-oriented tape processing; it can also be used to process tape files which were not created with BTAM. During processing of a tape file, the direction in which the file is processed can be changed as desired within the file, and tapes can be positioned to any desired block or section. BTAM processes files with or without standard blocks.

Macro	Brief description
BTAM	controls all BTAM actions.
FEOV	initiates a tape swap.
NDWERINF	interrogates the status bytes.

DIV (Data In Virtual)

The basis for file processing for a user is 4KB blocks. DIV can also be used to process files that were not created with DIV.

Macro	Brief description
DIV	Process files with the DIV access method • open a file • define a window (i.e. a work area in virtual address space) • write modified pages from the window back to the file on disk • undo changes in the window • release windows in virtual address space • close a file, releasing any existing windows with default values, if applicable

EAM (Evanescent Access Method)

EAM is used to process task-specific temporary files in the SYSEAM area. It is a block-oriented access method and is particularly suitable for rapid processing of task-specific work files.

Macro	Brief description
EAM	Controls all EAM accesses

FASTPAM (Fast Primary Access Method)

FASTPAM is a block-oriented access method which always works with 4KB blocks. FASTPAM can also be used to process files that were not created with FASTPAM.

Macro	Brief description
FPAMACC	File access functions
	• read and write blocks synchronously • read and write blocks asynchronously • wait for the end of asynchronous I/O jobs • report the end of asynchronous I/O jobs
FPAMSRV	Management functions
	• enable the system environment (FASTPAM environment) • enable I/O areas (FASTPAM I/O area pool) • open a file for processing • close a file opened with FPAMSRV • disable the system environment (FASTPAM environment) • disable I/O areas (FASTPAM I/O area pool)

SAM (Sequential Access Method)

SAM is a record-oriented access method. The records are stored sequentially in the file. SAM lets users process records sequentially in either direction (beginning-of-file to end-of-file or vice versa). For tape processing, SAM complies with all requirements of DIN 66029 up to exchange level 3. Files with either standard or nonstandard blocks can be processed.

Macro	Brief description
FEOV	initiates a tape swap.
GET	retrieves the next record.
PUT	writes the next record.
PUTX	(locate mode only) replaces a record read by means of GET.
RELSE	terminates a data block.
SETL	positions to beginning-of-file, to end-of-file, or to a record.

ISAM (Indexed-Sequential Access Method)

The basis for file processing is the structure of an ISAM file with its index and data sections. Each record contains a key and the keys are the criterion for sorting the record. For index and data blocks, see the “Introductory Guide to DMS” [1].

Macro	**Brief description**
ADDPLNK	assigns a pool link name to a user ISAM pool.
CREAIX	creates a secondary index for an ISAM file.
CREPOOL	sets up a user ISAM pool.
DELAIX	deletes secondary indices of an ISAM file.
DELPOOL	deletes a user ISAM pool.
ELIM	deletes a record from the file.
GET	reads the records from the file sequentially.
GETFL	if flagged ISAM keys are used: reads the next record within the flag range (sequentially).
GETKY	reads the first record with the specified key.
GETR	reads the records sequentially in reverse order.
INSRT	inserts a record into the file with a new ISAM key.
ISREQ	clears an ISAM lock.
OSTAT	informs the caller about the number and type of concurrent file accesses.
PUT	sequentially writes records to the end of the file (and also checks that the keys are in the right order).
PUTX	replaces a record read previously.
REMPLNK	deletes the pool link name.
RETRY	after execution of the EXLST PGLOCK exit, resets the ISAM pointer and repeats the last macro.
SETL	positions the ISAM pointer to the beginning of the file, to the end of the file or to a specific record for subsequent processing.
SHOPLNK	provides information about the assignment of ISAM pools to pool link names.
SHOPOOL	provides information about attributes and assignment status of ISAM pools.
SHOWAIX	provides information about secondary indices of an ISAM file.

STORE	• inserts a record with a new ISAM key into the file, or • overwrites a record with an existing ISAM key if duplicate ISAM keys are not permitted, or • inserts into the file a record with an existing ISAM key as the last record with this key.

UPAM (User Primary Access Method)

UPAM is a block-oriented access method. The basis of UPAM is the standard block (= PAM page). UPAM can also be used to process files that were not created with UPAM.

Macro	**Brief description**
PAM	Controls all UPAM accesses

For event-driven processing, the following macros are also significant (for detailed descriptions, see the “Executive Macros” [2] manual).

Macro	**Brief description**
CHKEI	checks the queue status for an event item.
CONTXT	accesses the register set of the interrupted task/process.
DISCO	closes the routine for the contingency process.
DISEI	disconnects the user program from the event item.
ENACO	opens a routine as a contingency process and assigns it a name and a priority.
ENAEI	creates an event item and/or establishes the link between the calling process and the event item.
FECB	creates a file event control block.
LEVCO	changes the priority of the called process.
POSSIG	signals an event.
RETCO	terminates the calling contingency process.
SOLSIG	requests a signal from the event item.
SUSPEND	places the calling process in an interruptible wait state.

Generation of operand lists for control blocks, DMS tables, etc.

The user can generate program areas or DSECTs (Dummy SECTions) which permit him/her to access the contents of DMS tables, file control blocks, etc. or the operand lists of DMS macros with the aid of symbolic addresses. For most macros this is possible with the aid of the MF operand, alternatively there are special DSECT macros. In the case of "older" macros (e.g. CATAL) which were only converted in a later version, the VERSION operand decides whether the special DSECT macro must still be used or whether specification using the MF operand is possible.

Operand lists for DMS macros with command functions

Macro	**Required VERSION specification**	**MF operand**	**“DSECT macro”**
ADDPLNK		x	-
CATAL	Without VERSION operand	-	IDCAT
	VERSION>=1	x	-
CHNGE		-	IDCHA
COMPFIL		x	-
COPFILE		x	-
COPY	*(Macro replaced by COPFILE)*	-	IDCOP
CREAIX		x	-
CREPOOL		x	-
DECFILE		x	-
DELPOOL		x	-
DELAIX		x	-
DROPTFT		x	-
ENCFILE		x	-
ERASE	VERSION=0	-	IDERS
	VERSION>=1	x	-
FILE	VERSION=0	-	IDPFL/IDPFX
	VERSION>=1	x	-
FSTAT	VERSION=0 (corresponds to 710)	-	IDFST
	VERSION>=1 (1 corresponds to 800)	x	
IMPNFIL		x	-
IMPORT	Without VERSION operand	-	DMAIMP

	VERSION=1	x	-
LFFSNAP		x	-
LJFSNAP		x	-
MAILFIL		x	-
RDTFT	Without VERSION operand	-	DMARD (PLIST=INPUT) DMADR (PLIST=OUTPUT)
	VERSION>=2	x	-
REL	*(Macro replaced by RELTFT)*	-	IDREL
RELTFT		x	-
REMPLNK		x	-
RFFSNAP		x	-
RJFSNAP		x	-
SHOPLNK		x	-
SHOPOOL		x	-
SHOWAIX		x	-
VERIF		-	IDVRF

where:

- x DSECT can be generated via the MF operand
- \- no MF operand/no DSECT macro

Control blocks and macros specific to access methods

Macro	**Brief description**
IDECB	for the UPAM file event control block (FECB).
IDFCB	for the file control block (FCB) of the user program at the TU level.
IDFCBE	extension of the 24-bit TU FCB.
IDMCB	EAM control block for EAM macros.
IDPPL	operand list for the PAM macro.
IDOST	operand list for the OSTAT macro.

DMS tables, file catalog, etc.

Macro	Brief description
IDCE	catalog entry.
IDCEG	catalog entry (extension for file generation groups).
IDCEX	catalog entry (extension).
IDEE	catalog entry (extent list).
IDEMS	DMS error messages.
IDTFT	TFT entry
IDVT	volume label entry (in the volume table).

Output of information on files, volumes, devices, etc.

Various DMS macros are provided enabling information on catalog entries, file status, the task file table, device and volume allocation, etc. to be requested in the program at any time and utilized for further processing.

Macro	Brief description
FSTAT	information from file catalog or about catalog entries for files.
OSTAT	information about the number and type of ISAM file accesses by different jobs.
RDTFT	information about TFT entries.
SHOWAIX	information about secondary keys of an ISAM file.
SHOPLNK	information about ISAM pool link names.
SHOPOOL	information about ISAM pools.

Comparison of macros and commands

Macro	Command	Function
ADDPLNK	ADD-ISAM-POOL-LINK	Define a pool link name for an ISAM pool
CATAL	CREATE-FILE	Create catalog entry
	CREATE-FILE-GROUP	Define file generation groups
	MODIFY-FILE-ATTRIBUTES	Define protection mechanisms
	MODIFY-FILE-GROUP-ATTRIBUTES	Define protection mechanisms
CHNGE	CHANGE-FILE-LINK	Change a file link name in the TFT
COMPFIL	COMPARE-DISK-FILES	Compare two disk files
COPFILE	COPY-FILE	Copy a file
CREAIX	CREATE-ALTERNATE-INDEX	Create an alternate index (secondary key) for an ISAM file
CREPOOL	CREATE-ISAM-POOL	Create an ISAM pool
DECFILE	DECRYPT-FILE	Convert an encrypted file into an unencrypted file
DELAIX	DELETE-ALTERNATE-INDEX	Delete alternate indices (secondary keys) of an ISAM file
DELPOOL	DELETE-ISAM-POOL	Delete/release an ISAM pool
DROPTFT	UNLOCK-FILE-LINK	Release lock on TFT entry
ENCFILE	ENCRYPT-FILE	Convert an unencrypted file into an encrypted file
ERASE	DELETE-FILE	Delete or export one or more files
	DELETE-FILE-GENERATION	Delete file generation group(s)
	DELETE-FILE-GROUP	Delete file group(s)
	DELETE-SYSTEM-FILES	Delete system files
	EXPORT-FILE	Export file(s)
	EXPORT-NODE-FILE	Export node file(s)
FILE	ADD-FILE-LINK	Create a TFT entry
	CREATE-FILE	Create file
	CREATE-FILE-GENERATION	Create file generation group
	CREATE-TAPE-SET	Create TST entry

	EXTEND-TAPE-SET	Extend TST entry
	IMPORT-FILE	Import file
	MODIFY-FILE-ATTRIBUTES	Modify attributes of a file
	MODIFY-FILE-GENERATION-SUPPORT	Modify attributes of a file generation group
FSTAT	IMPORT-FILE SHOW-FILE-ATTRIBUTES	Retrieve information from the file catalog
IMPNFIL	IMPORT-NODE-FILE	Create (import) catalog entries for node files
IMPORT	CHECK-IMPORT-DISK-FILE	Check file import in advance
	IMPORT-FILE	Create (import) catalog entries for files
LFFSNAP	LIST-FILE-FROM-SNAPSET	List files from a Snapset
LJFSNAP	LIST-JV-FROM-SNAPSET	List job variables from a Snapset
MAILFIL	MAIL-FILE	Send file or library member to a user ID by email
RDTFT	SHOW-FILE-LINK	Retrieve information from the task file table
RELTFT	DELETE-TAPE-SET	Delete a TST entry
	REMOVE-FILE-LINK	Delete a TFT entry
REMPLNK	REMOVE-ISAM-POOL-LINK	Delete a pool link name
RFFSNAP	RESTORE-FILE-FROM-SNAPSET	Restore files from a Snapset
RJFSNAP	RESTORE-JV-FROM-SNAPSET	Restore job variables from a Snapset
SHOWAIX	SHOW-INDEX-ATTRIBUTES	Output information on the alternate indices (secondary keys) of an ISAM file
SHOPLNK	SHOW-ISAM-POOL-LINK	Show assignments of ISAM pools to ISAM pool link names
SHOPOOL	SHOW-ISAM-POOL-ATTRIBUTES	Return information on an ISAM pool
VERIF	CHECK-FILE-CONSISTENCY	Restore file consistency
	REMOVE-FILE-ALLOCATION-LOCKS	
	REPAIR-DISK-FILES	

DMS commands without corresponding macros

Command	Function
ADD-CRYPTO-PASSWORD	Store the crypto password for decrypting encrypted file contents in the task's password table
ADD-PASSWORD	Enter a password in the password table of the job
CONCATENATE-DISK-FILES	Concatenate SAM files
EDIT-FILE-ATTRIBUTES EDIT-FILE-GROUP-ATTRIBUTES EDIT-FILE-GENERATION-SUPPORT	Start the guided dialog of the corresponding MODIFY command and enable an existing catalog entry to be "edited"
EDIT-FILE-LINK	Start the guided dialog of the ADD-FILE-LINK command and enable an existing TFT entry to be "edited"
LIST-NODE-FILES	Provide information about node files on the Net-Storage
EDIT-FILE-GROUP-ATTRIBUTES	Only possible if the chargeable product SDF-P is in use: activates the guided dialog of the MODIFY-FILE-GROUP-ATTRIBUTES command
EDIT-FILE-LINK	Only possible if the chargeable product SDF-P is in use: activates the guided dialog of the ADD-FILE-LINK command
LOCK-FILE-LINK	Lock a TFT entry, thus preventing it from being released until after a UNLOCK-FILE-LINK command
REMOVE-CRYPTO-PASSWORD	Remove the crypto password from the password table of the ongoing task
REMOVE-PASSWORD	Delete a password from the password table of the job
RESTART-PROGRAM	Restart a program at a checkpoint which was set by means of a WRCPT (or CHKPT) macro
SHOW-BLOCK-TO-FILE-ASSIGNMENT	Privileged command: shows files in which the requested blocks are located
SHOW-FILE-LOCKS	Show the locks on a file

Programming notes

This chapter describes features of the various access methods which are relevant to programming.

BTAM - Basic Tape Access Method

BTAM is an access method for block-oriented tape processing; it can even be used to process tape files which were not created with BTAM. While a tape file is being processed, the processing direction within the file can be changed as desired. The tapes can be positioned by block or by section as desired. BTAM processes files with and without standard blocking.

Macros for the BTAM access method

The following macros can be used by the BTAM access method:

Macro	Operation	Function
CLOSE		Service macro
EXLST		Service macro
EXRTN		Service macro
FCB		Service macro
FCBAD		Service macro
IDFCB		Service macro
IDFCBE		Service macro
LBRET		Service macro
OPEN		Service macro
BTAM	CHK	check processing status of an I/O operation.
	ERG	generate interblock gap.
	MINF	fetch medium information (only useful for volume types which contain optical disks)
	POS	position tape.
	RBID	determine tape position.
	RD/RDWT	read data info main memory and wait for completion of I/O operation.
	RDBF	read data from save area of tape cartridge buffer.
	REV/REVWT	read tape in reverse direction and wait for completion of I/O operation.
	RT/RTL	read with data transfer; with/without message if the length is less than anticipated.
	RNT/RNTL	read without data transfer; with/without message if the length is less than anticipated.
	SYNC	synchronize and determine tape position.
	WRT/WRTWT	write data from main memory and optionally wait for completion of I/O operation.

	WT	wait for completion of the I/O operation.
	BSF	synchronize and determinate tape position.
	BSR	Control code for positioning and for writing tape marks
	FSF	Control code for positioning and for writing tape marks
	FSR	Control code for positioning and for writing tape marks
	REW	Control code for positioning and for writing tape marks
	RUN	Control code for positioning and for writing tape marks
	WTM	Control code for positioning and for writing tape marks

OPEN modes

The following open modes are available with the BTAM access method:

INOUT	Retrieve records from an existing file and add new records; no header labels are written, since the file must already exist.
INPUT	Retrieve files from an existing file in the forward direction.
OUTIN	Create a new file and/or retrieve records; labels are written since a new file is being created.
OUTPUT	Create a new file.
REVERSE	As for INPUT, but the tape is positioned to EOF at OPEN time and the file is read backwards. Files which extend over several volumes can only be processed individually (with the aid of the VSEQ operand).
SINOUT	As for INOUT, but the tape is not positioned; this is not permitted if the tape is positioned at the beginning of the tape.

The OPEN modes INPUT and REVERSE differ, at OPEN time, only in how the tape is positioned. An OPEN REVERSE followed by an RD or RDWT operation will not result in a reverse read.

BTAM operations and OPEN modes

BTAM operation	INPUT	REVERSE	OUTPUT	INOUT / OUTIN / SINOUT
CHK	x	x	x	x
Control *	n	n	x	x
ERG			x	x
RD/RDWT	x	x		x
REV/REVWT	x	x		x
RT/RTL	x	x		x
RNT/RNTL	x	x		x
WRT/WRTWT			x	x
WT	x	x	x	x
MINF	x	x	x	x
POS	x	x	x	x
RBID	x	x	x	x
RDBF			x	x
SYNC	x	x	x	x

where:

x = permissible.

n = output control functions are not permissible.

* “Control” stands for the operands FSF, BSF, WTM, RUN, ERG, FSR, BSR, REW. These are described in the chapter "BTAM - Process tape files (type S)".

BTAM record and block formats

BTAM is a block-oriented access method for tape files of format BLKCTRL=NO. The following applies for block length: 18 bytes <= block length <= 32768 bytes (see description of the LEN operand in the BTAM action macro).

BTAM evaluates the FCB operand RECFORM. The specification of the record format is related to the block length.

- RECFORM=F fixed length for blocks
  The length is defined with BLKSIZE=length or is specified as LEN in the BTAM macro. The values specified above apply for the minimum and maximum length.
- RECFORM=U for blocks of undefined length
  BTAM takes the block length either from the LEN operand in the macro or from the register specified in FILE /FCB with RECSIZE=reg.
- RECFORM=V for blocks of variable length
  (record format V is treated as record format U)

Tape files created with SAM can be read block-by-block with BTAM. Since BTAM does not know the record structure, the user is responsible for deblocking the records.

DIV - Data In Virtual

Data in Virtual (DIV) is an access method that differs from the traditional access methods such as ISAM, SAM, and UPAM because it does not require a file to be structured into records and blocks and works without I/O buffers, and operations such as GET or PUT.

DIV works with a special DIV-OPEN.

DIV is an object-oriented access method that is particularly suitable for processing unstructured data (Binary Large Objects (BLOBS)).

The DIV interface is an SVC interface. Jobs are formulated by a parameter list; acknowledgments are made by means of a return code in the parameter list (not via exits).

DIV is not supported on SPARC systems.

Macro for the DIV access method

The DIV access method works with the DIV macro, which covers the various functions for data processing:

Macro	Function	
DIV	CLOSE	Close a file, releasing any existing windows with default values, if applicable
	MAP	Define a window (i.e. a work area) in the virtual address space
	OPEN	Open a file
	RESET	Undo changes in the window
	SAVE	Write modified pages from the window back to the file on disk
	UNMAP	Release the window in the virtual address space

Opening a file

A program can process a file only after calling an OPEN function. Among other things, the OPEN function verifies whether a user has the necessary access rights, whether the file is already open and whether the file open modes are compatible with each other. The file open modes and file access by many users are to a great extent similar to the OPEN function of other DMS access methods (see the “Introductory Guide to DMS” [1])
The file can be set for read-only or read and write access. Following a write access, the file can be modified, extended or rewritten from the beginning of the file.
The file can be simultaneously accessed by a single user with write authorization and/or many users with read authorization as well as many users with read and write authorization.The user can specify whether a given data space or the entire file should be read immediately into a window when it is defined, or whether a given page should be read into a window only when the page is first accessed.

List of most important macro operands for opening a file

Operand	Operand values	Meaning
FCT	*OPEN	Opening a file
LARGE_FILE		The LARGE_FILE function operand specifies whether the file that is to be opened may grow to become a “large file” with a file size that may exceed 32 GB.
	*ALLOWED	The file may become a large file.
	*FORBIDDEN	The file may not become a large file.
MODE		The function operand MODE specifies which operation (read, write) should be performed on the file (write means executing the DIV function SAVE).
	*INPUT	The file is opened as the input file; the SAVE function is not permitted.
	*INOUT	The file must exist; both read and write operations are permitted.
	*OUTIN	A new file will be created; both read and write operations are permitted.
SHARUPD		Depending on the MODE setting, the function SHARPUPD is used to indicate which multiuser operation modes should be allowed (see the tables under “Multiuser operation” below):
	*NO	One write-authorized user **or** many read-authorized users can open the file simultaneously.
	*WEAK	One write-authorized user **and** many read-authorized users can open the file simultaneously.
	*YES	**Multiple** write-authorized users can open the file simultaneously.
LOCVIEW		The LOCVIEW operand is used to specify when a page should be read into a window.
	*MAP	All pages of the specified file region are read into a window as soon as it is defined (FCT=*MAP).
	*NONE	A page is not read into a window from the disk file until a page is accessed.

Multiuser operation

A UPAM file can be created and processed with the access methods UPAM (see "UPAM - User Primary Access Method"), FASTPAM (see "FASTPAM - Fast Primary Access Method") or DIV. FASTPAM and DIV can, however, only process UPAM files with the attribute BLKCTRL=NO.

Authorization for parallel file processing is dependent on the operand open values specified for SHARUPD, MODE, LOCKENV and LOCVIEW.

The permissible parallel opens are shown in the following table:

Compatibility matrix for DIV-OPEN

			USER B	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**
	SHARUPD=		*YES	*YES	*YES	*NO	*NO	*NO	*WEAK	*WEAK	*WEAK	*WEAK
		OPEN mode	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN	INPUT	LMAP	INOUT	OUTIN
USER A	*YES	INPUT INOUT OUTIN	X O O	O O O		X			X X X	X		
	*NO	INPUT INOUT OUTIN	X			X			X X X	X		
	*WEAK	INPUT LMAP INOUT OUTIN	X X	X		X X	X		X X X X	X X O	X O	

Table 1: DIV: Permissible SHARUPD/OPEN combinations

LMAP: INPUT LOCVIEW=MAP (only with DIV))

X: OPEN permitted

O: OPEN permitted only

- if the same block-oriented access method is used by all (either UPAM/FASTPAM or DIV)
- and the same value is used by all for the LOCKENV operand (either LOCKENV=*HOST or LOCKENV=*XCS)
- and all are running on the same host or, with LOCKENV=*XCS, in an XCS network

Comments

- Read operations with SHARUPD=*WEAK may have opened a file simultaneously with any write operation.

  All read operation window pages with DIV-SHARUPD=*WEAK which specified LOCVIEW=*MAP with OPEN are read from the file into the window when MAP is reached. DIV thereby ensures that no file pages can be modified during reading by the parallel SAVE of a write operation with DIV-SHARUPD=*WEAK.
  This protection against parallel writing does not exist for a UPAM/FASTPAM write operation.

  For this reason, read operations with DIV-SHARUPD=*WEAK for which LOCVIEW= *MAP was specified, are compatible to write operations with DIV-SHARUPD=*WEAK, but not, however, to other write operations.

  All other conditions formulated for entry 'O' above must also be fulfilled (all openers with the same value for the LOCKENV operand and all openers in the same host or, if in different hosts, with the entry LOCKENV=*XCS).

  Read operations with DIV-SHARUPD=*WEAK which specified LOCVIEW=*NONE with OPEN possess the same compatibility as read operations with UPAM/FASTPAM-SHARUPD=*WEAK.
- Openers with DIV-SHARUPD=*YES are not compatible with openers with UPAM/FASTPAM-SHARUPD=*YES.
- Read operations are always compatible with each other (regardless of access method, SHARUPD specification, LOCKENV specification and host).
- SHARUPD=*YES:
  The file size is checked whenever the allocator is called.
  If this check indicates a file size >= 32 GB and the attribute LARGE_FILE=*FORBIDDEN is set in the associated FCB or the attribute EXCEED-32GB=*FORBIDDEN is set in the TFT then processing is canceled.
  In this case, DIV returns the code `X'00400030'` in its local parameter list DIV(I).

Defining windows

The MAP function is used to define a window, i.e. a region in an address area (program or data space) that is assigned to a file area or an entire file.
At the time of calling OPEN, users can specify whether a given page should be read into a window only when it is first accessed, or as soon as MAP is called. The individual operands of the MAP function can be used to specify the type (program or data space), position and size of the virtual address area.

The FCT=*MAP and FCT=*UNMAP functions of the DIV macro are used to open and close windows for a program call. SPID=0 must always be specified for the SPID operand (ID of the data space).

The following rules must be observed:

- One page of an address space can be assigned to **one** window only.
- Within an OPEN, a file page can be assigned to **one** window only.
- A file page can be assigned to more than one window only if these windows belong to different OPEN operations.

List of the most important macro operands for defining windows

Operand	Operand values	Meaning
FCT	*MAP	Defines a window
SPID		SPID specifies the ID of the data space in which the window is to be created.
AREA		AREA specifies the starting address of the window in a virtual address space (aligned on a 4KB page boundary). A page of an address space must be assigned to **one** window only.
OFFSET SPAN		The OFFSET and SPAN operands specify the (beginning and length) of the file area in units of 4KB pages and thus define the size of the window. The MAP function can be used to extend the physical length of a file.
DISPOS		DISPOS can be used to define the contents of the window: *Note* After a call to the REQM or DSPSRV macro, the specified virtual address space is initialized with X' 00' (see the “Introductory Guide to DMS” [1])
	*OBJECT	A page of the file will be read into the window when the page is first accessed.
	*UNCHNG	At the time of the first access, a window page retains its contents (i.e. the contents of the virtual address space) and is not replaced by the corresponding file page).

Writing data back to the disk file

The SAVE function is used to write (save) modified window pages back to the disk file.

A file area for which the SAVE function is to be executed is specified, and all modified window pages of this area are written back to the disk file. In the process, the file length can be **logically** extended or reduced. (The file can be physically extended only by using the MAP function, see the “Introductory Guide to DMS” [1]). The new file length is returned by the SAVE function in a field (DIVPSIZE) of the parameter list.

List of the most important macro operands for writing back to the disk file

Operand	Operand values	Meaning
FCT	*SAVE	Writes data back to the disk file
OFFSET SPAN		The OFFSET and SPAN operands define the file area for which the SAVE function is to be executed (i.e. the beginning of the area and its length in 4KB blocks. The file length can be **logically** extended or reduced. The new file length is returned by SAVE. Physical extensions of the file length are only possible with MAP; see extending of a file physically as described in the “Introductory Guide to DMS” [1].

Undoing modifications in a window

The RESET function can be used to undo (“erase”) changes that have been made in windows since MAP or the most recent SAVE. A file region for which the RESET function is to be executed must be defined, taking the effects of the DISPOS operand of the MAP function into account:

- If the window is defined with DISPOS=*OBJECT, an access to a page following a RESET will cause that page of the file to be displayed on the screen.
- If the window is defined with DISPOS=*UNCHNG, an access to a page following a RESET will cause that page to be initialized with X'00', unless that page has already been written to the file using SAVE. In the latter case, the page is read again from the file after RESET.

List of the most important macro operands for undoing modifications in a window

Operand	Operand values	Meaning
FCT	*RESET	Undoes modifications in a window
OFFSET SPAN		The OFFSET and SPAN operands define the file area for which the RESET function is to be executed (i.e. the beginning of the area and its length in 4KB blocks). All modifications in the window pages of the RESET region are undone.
RELEASE		The RELEASE=*YES operand extends the RESET function to **all** pages of the specified region, not just to modified pages.

Disabling a window

UNMAP is used to disable a window. The user can define what the contents of the pages of the (disabled) window should be after the execution of the UNMAP function. The window pages can be either initialized with X'00' or set to appear to the user as they were before UNMAP.

List of the most important macro operands for closing windows

Operand	Operand values	Meaning
FCT	*UNMAP	Disables windows
SPID		Specifies the address space in which the window is located.
AREA		Specifies the starting address of the window that lies in the address space defined by SPID.
DISPOS		Specifies the state in which the pages of the (disabled) window should remain.
	*FRESH	The window pages are initialized (X' 00').
	*UNCHNG	As far as the program is concerned, the windows retain their original contents prior to UNMAP.

Closing a file

The CLOSE function can be used to closed a specified file. Existing windows, if any, are shut down using default values.

List of the most important macro operands for closing a file

Operand	Operand values	Meaning
FCT	*CLOSE	Closes a file.
ID		ID of the file to be closed.

EAM - Evanescent Access Method

EAM is used for processing task-specific files in the SYSEAM area. EAM is a block-oriented access method and is particularly suitable for the rapid processing of job-dependent work files.

Macro for the EAM access method

All EAM functions are controlled by the EAM macro. The EAM macro covers the following functions:

Macro	Function
EAM	• sets up and opens a new file • opens an existing file • reads (blockwise, sequential or direct) • writes (blockwise, sequential or direct) • checks and waits for I/O termination • closes a file • deletes a file

The desired operation is selected by specifying a hexadecimal operation code in the MFCB, and initiated by the EAM macro. The effect is determined by the MFCB fields which EAM additionally evaluates after analyzing the operation code (see table in chapter "MFCB (Mini File Control Block)", section "Structure of the MFCB").

The EAM macro controls all EAM accesses. EAM has the following characteristics:

- EAM files are not cataloged. As a result of this, no disk access is necessary when an EAM file is opened.
- Each EAM file is automatically deleted when the job which opened it is terminated (temporary file).
- Communication between EAM and the user takes place only via the EAM control block (MFCB = Mini File Control Block). No facilities exist for modifying this control block at OPEN time.
- EAM works exclusively with public volumes (pubsets). No distinction is made between disks with and without PAM keys (K and NK disks).
- The space requirements for the EAM routines and the runtimes for read and write accesses are less than for the standard access methods for cataloged files.
- An EAM file can be processed only by the job which created and opened it. One job may open and process several EAM files.
- EAM is block-oriented and is based on blocks of 2048 bytes each (= a PAM page). For chained I/O, up to 16 sequential blocks may be transferred at one time.
- EAM files may not exceed 32 GB.
- If a program is restarted using RESTART-PROGRAM, all existing EAM files belonging to the job are erased.

Note

Where the EAM file is created depends on whether or not shared pubsets are used and on the definitions set by systems support. For more information, see the “Introduction to System Administration” [7].

MFCB (Mini File Control Block)

Structure of the MFCB

The table below shows the fields within the MFCB and the way in which they are interpreted according to the operation selected.

IDMFOPC (Operation code)	**IDM FUNIT*** Functional unit	**IDM VERS*** Version ID	**IDM RETCO*** Return code	**IDM FOC** Option byte	**IDMFLBN** Logical block number	**IDM FFN** File name	**IDMFEB** Sense byte	**IDM FSB** Status byte	**IDMFIO A1*** Address IOAREA1	**IDMFNHP** No. of blocks to be transf.	**IDMFIO A2*** Address IOAREA2
Open new file (IDMFO)	A	A	S	A	(+)	S	S	-	-	(A)	-
Reopen (IDMFRO)	A	A	S	A	S	A(+)	S	-	-	(A)	-
Read (IDMFRD)	A	A	S	A	A	A	S	S	(A)	-	(A)
Write (IDMFWR)	A	A	S	A	A	A	S	S	(A)	-	(A)
Check (IDMFCK)	A	A	S	-	-	A	S	S	-	-	-
Check and wait (IDMFCW)	A	A	S	-	-	A	S	S	-	-	-
Close file (IDMFCL)	A	A	S	-	S	A	S	S	-	-	-
Erase file (IDMFER)	A	A	S	-	-	A	S	-	-	-	-

where:

* applies only with PARMOD=31

E field contents are evaluated

S field contents are set by the system

\+ exceptions in the case of object module files (see "Handling object module files with EAM")

Description of the MFCB

An overview of the fields contained in the MFCB and the way in which they are interpreted according to the operation selected is given in the table.

The MFCB is the communication area between EAM and the user. It must be aligned on a word boundary. The fields required for the selected operation must be supplied with the appropriate values before the EAM macro is called.

The IDMCB macro can be used to assign symbolic names to the MFCB.

Functional unit (IDMFUNIT)

If PARMOD=31 applies either explicitly or implicitly, the value 'DMFEAM' must be placed in IDMFUNIT.

Operation code (IDMFOPC)

Code	Meaning
IDMFO	Create and open a new file (OPEN). EAM evaluates bit IDMFOO of the option byte and then opens either the object module file or a new job-specific file. A new object module file has its name recorded in the TCB (Task Control Block); an existing object module file is reopened (i.e. OPEN=REOPEN applies; see also section "Handling object module files with EAM"). A new EAM file is assigned a binary file name of between 1 and 14000, and this is written into the “file name” field (IDMFFN). In addition, bit IDMFCI of the option byte is evaluated. If this bit is set (i.e. indicating chained I/O mode), a check is made to test whether field IDMFNHP (= number of blocks to be transferred) contains a number between 1 and 16. The addresses of the I/O areas are not checked.
IDMFRO	Reopen an existing file (REOPEN). Bits IDMFOO, IDMFSBR and IDMFCI of the option byte are evaluated. For IDMFCI and hence for field IDMFNHP, the same applies as in the case of the OPEN operation. According to the result of the IDMFOO evaluation, either the job-specific object module file or the file named in the IDMFFN field is opened (see "Handling object module files with EAM").
IDMFRD	Read (READ). EAM evaluates bit IDMFI1 of the option byte and checks the address of an input area (field IDMFIO1 or IDMFIO2) selected via this bit. Input takes place at the corresponding address, even if the contents of IDMFIO1/2 were modified immediately after the operation was invoked. If the end-of-file condition is recognized during a read operation, bit IDMFEF of the sense byte is set. If bit IDMFCI of the option byte was set at OPEN/REOPEN time, chained input is operative. The number of blocks to be transferred is also taken from field IDMFNHP at OPEN/REOPEN time (see "Number of blocks to be transferred (IDMFNHP)"). The “logical block number” field (IDMFLBN) contains the number of the next block to be read, or a value of zero in the case of sequential reading. The read/write operation is asynchronous in EAM, i.e. control is returned to the user immediately after the EAM macro is called, unless a previous I/O operation has not yet terminated. Any error information entered in the MFCB (fields IDMFERR, IDMFSB) always relates to the preceding I/O operation. Consequently, when there are two I/O operations in immediate succession, the second has to wait. Overlapping I/O or double buffering can be implemented by specifying two different I/O area addresses (see section "Overlapping input/output" in chapter "EAM processing").

IDMFWR	Write (WRITE). As in the case of READ, bit IDMFI1 is evaluated and, depending on the result, either field IDMFIO1 or field IDMFIO2 is checked for the validity of the output area address it contains. If the “logical block number” field (IDMFLBN) contains a value of zero, the block is written sequentially at the end of the file. If its value is not equal 0, the block to be transferred is inserted at the position in the file designated by this value. In all other respects, the execution sequence for write operations is the same as for read operations (see above).
IDMFCK	Check for termination of an I/O operation (CHECK). A check is made to determine whether an outstanding I/O operation has terminated. Whatever the case, control is immediately returned to the user. If the operation is not yet completed, a value of 8 is placed in register 15. If the operation is completed, the status bytes are transferred to the MFCB (IDMFSB) and register 15 is set to the value 0.
IDMFCW	Check an I/O operation and wait (CHECK WAIT). EAM is instructed to wait for the last I/O operation to terminate. Following this, the status bytes are transferred. If another operation has already initiated the transfer of the status bytes, this EAM macro has no effect.
IDMFCL	Close file (CLOSE). Following termination of the final outstanding I/O operation, the file is marked as closed. The block number of the last block in the file is transferred to the “logical block number” field (IDMFLBN).
IDMFER	Erase file (ERASE). The file is erased, irrespective of whether or not it is open.

Version number (IDMVERS)

If PARMOD=31 applies either explicitly or implicitly, the value DMEAMV must be placed in IDMVERS. This is important with regard to future versions of BS2000, since it means that different versions of this interface can be supported without the need for recompilation.

Return code (IDMRETCO)

In the 31-bit version the return code is placed in the IDMRETCO field in the MFCB. This return code corresponds to that in register 15.

Return code	**Meaning**
0	Operation completed successfully
4	Operation not completed successfully; check sense byte (IDMFEB)
8	After check operation: checked I/O operation not yet terminated

Option byte (IDMFOC)

IDMFOO	Open object module file. This bit is evaluated in all operations. If the bit is set, the job-specific object module file is processed (see "Handling object module files with EAM"). If the bit is not set, a new file is opened in the OPEN function and its name is moved to the “file name” field (IDMFFN). In the REOPEN function, the file named in the IDMFFN field is opened.
IDMFCI	Chained I/O. This bit is evaluated at OPEN/REOPEN time and its contents are saved. Thus, when a file is (re)opened, if this bit is set, subsequent I/O operations will be chained. The length of the chain is specified by the contents of field IDMFNHP (= number of blocks to be transferred), which is also evaluated when the file is opened. If the bit is not set, no chained I/O will be performed.
IDMFSBR	Starting point for I/O in the file. This bit controls where I/O operations are to start in a file. It is evaluated when the file is reopened. If it is set, a value of 0 is moved to the “logical block number” field. If it is not set, the highest block number so far assigned to the file is placed in the field IDMFLBN (see the description of the IDMFLBN field).
IDMFI1	Control of the I/O area. This bit is evaluated during current I/O operations. If it is set, the I/O area 2 address specified in field IDMFIO2 is checked for validity, and this area is used for the pending I/O operation. If the bit is not set, the I/O area 1 address contained in field IDMFIO1 is checked. This area is then used for I/O operations (see section "Overlapping input/output" in chapter EAM processing).

Logical block number (IDMFLBN)

The logical block number is a 2-byte binary number (0 <= n <= 65535). If its value is 0, processing is performed sequentially. In a read operation, the block immediately following the last block accessed in a read/write operation is transferred. In a write operation, a block is added to the end of the file.

If the logical block number is not equal 0, it points directly to the block of the file that is to be read or written.

In chained I/O, the value specified applies to the first block in the chain.

File name (IDMFFN)

The file name is a 2-byte binary number (decimal: 1 <= n <= 14000). When a new file is opened by EAM, this number is placed in the “file name” field and must be specified whenever the file is subsequently referenced.

Sense byte (IDMFEB)

If an error occurs during an operation initiated by the EAM macro, bits are set in the sense byte according to the type of error (the bits can be addressed by their symbolic names). At the same time a value of X'00000004' is placed in register 15.

IDMFIC	Invalid operation. Invalid operations include, for example, illegal operation code, attempt to access a file which is not open, in chained I/O the value of field IDMFHNP is not between 1 and 16, and MFCB is not aligned on a word boundary.
IDMFIF	Invalid file name. The number specified in the “file name” field (IDMFFN) does not identify any EAM file associated with this job.
IDMFIB	Invalid block number. The number specified in the “logical block number” field (IDMFLBN) refers to a block outside the file (read access) or is greater than the number of the last block written + 1 (write access).
IDMFIA	Invalid I/O area address. The address contained in field IDMFIO1 or IDMFIO2 for I/O area 1 or I/O area 2 respectively is invalid.
IDMFNS	No more EAM space available. For example, the user has reached the maximum number of EAM files allowed (14000) or the total amount of space available on the system for all EAM files has been exhausted.
IDMFNP	Illegal access to a privileged file. A non-privileged user attempted to access a privileged file.
IDMFEF	End-of-file. End-of-file was reached during a read access: if a block chain encounters the end of the file during chained input, as much of the read operation as possible is completed and the end-of-file bit is set.
IDMFERR	Check status bytes. The preceding read or write operation was not completed successfully. The status bytes should be checked to determine the cause of the error.

Status field (IDMFSB)

The value of this field is set by the system if the following conditions are true simultaneously:

- the preceding operation was a read or write operation;
- the current operation is a read, write, check, check-and-wait, or close operation.

The following bytes are transferred from the channel control block (CCB): the standard device byte, 3 sense bytes, the Executive flag byte.

Address of I/O area 1 (IDMFIO1/IDMFIOA1)

This field contains the virtual address of the first byte of I/O area 1.
In write operations, a block or block chain is transferred from this address. In read operations, the block/block chain is transferred to this address.

If the I/O area is the same size as a block (2048 bytes), it should be located within a page (4096 bytes) and commence on a word boundary. In the case of chained I/O the area should commence on a page boundary; it must be able to hold as many blocks as can be transferred with a single I/O request.

If the area is not aligned as specified above, buffering may be required in conjunction with certain hardware, which in turn leads to a drop in performance.

Number of blocks to be transferred (IDMFNHP)

This field is evaluated at OPEN/REOPEN time if chained I/O (IDMFCI) is specified in the option byte. It contains a 1-byte binary number <= 16.

If end-of-file is reached during chained I/O in read mode, the system sets the value of this field equal to the number of blocks transferred.

Address of I/O area 2 (IDMFIO2/IDMFIOA2)

This field contains the virtual address of the first byte of I/O area 2. The same address can be used as for I/O area 1, but for asynchronous processing with overlapping input/output, the address specified in this field must refer to an area which does not overlap I/O area 1.

The same conditions apply here as in the case of I/O area 1.

Chained input/output

If bit IDMFCI for chained processing is set in the MFCB, I/O operations are performed at a faster rate. The blocks to be written as a chain need not be adjacent to each other. In write operations, block chains are always written in groups of 3 PAM pages. In read operations, adjacent pages are read as a chain. Field IDMFNHP (= number of blocks to be transferred) must therefore always contain a value that is a multiple of 3. In addition, I/O operations should always start at block numbers that can be represented in the form (3 x n) + 1, i.e. 1, 4, 7, ...

Note

> In order to support NK4 pubsets (in later operating system versions), EAM users are requested to convert chained processing to the blocking factor 2 or a multiple of 2. In this case, odd block numbers (BLOCK#) should be specified for direct I/O operations, i.e. 1, 3, 5, ...
>
> Increasing the blocking factor at the expense of main memory space (I/O buffer) leads to savings with regard to CPU time (initiation and termination of the I/O request) and channel and device times (seek and search times); this is because several blocks can be read or written in one physical I/O operation. The system has no influence on this optimization possibility, which lies effectively in the hands of the EAM user.

EAM processing

Using check operations

After a read or write call, control is returned to the user as soon as the requested operation has been accepted. In other words, there is no need to wait for this operation to be completed.

Before a read or write operation is initiated, however, the system waits for the preceding read/write operation (if any) to terminate (i.e. implicit check-and-wait operation). Similarly, when a close operation is requested, the system waits for the last read/write operation to terminate.

Thus, after the last in a series of read/write operations, a check operation is necessary only if the file is not immediately closed again or if, in the case of chained I/O, reading is continued until the end-of-file condition is reached (bit IDMFIB or IDMFEF of sense byte=1) and the number of blocks transferred is not equal to 0.

Example

In an EAM file, 3 read operations are performed. The file is not closed after this, because it is still required for later I /O operations. However, these I/O operations are not requested until after processing of the blocks that were read has been completed:

```
READ
CHECK/WAIT                Wait for termination of the last I/O operation.
  .
  .
Processing of the blocks that were read
  .
Further I/O operations
```

Changing processing characteristics

The following specifications are observed when opening or reopening a file:

- chained/non-chained input/output
- number of blocks to be transferred.

If one of these values is to be changed during processing of the file, the following actions are required:

1. Close file
2. Modify fields in MFCB
3. Reopen file

The value for the number of blocks to be transferred should be changed if, for example, fewer blocks are to be transferred in the last write operation than was specified when the file was opened.

Example

99 blocks are to be written to an EAM file. Chaining is to be used, with 15 blocks being transferred per write call (byte IDMFNHP in the MFCB). The following operations are then requested:

```
OPEN (new file) -> WRITE -> WRITE -> WRITE -> WRITE -> WRITE -> CLOSE *) ->
REOPEN -> WRITE -> CLOSE
```

*) After 6 write operations, there are still 9 blocks to be written. The file must therefore be closed and then reopened, this time with a value of 9 set for the number of blocks to be transferred.

Overlapping input/output

A reduction in processing time can be achieved by means of asynchronous I/O operations:Once an I/O operation has been initiated, control is immediately returned to the program in order to enable other processing to take place in parallel with the physical I/O. The next I/O operation is then initiated using a second I/O area that does not overlap the first, and so on. Figure 1 illustrates the overlapping of processing and input operations.

Sequential read with chained input/output

Chained I/O is used most effectively for sequential reading if

- a multiple of 3 is selected as the number of blocks to be transferred (3 x n);
- the block number selected is in the form: (3 x n) + 1 (e.g. 1, 4, 7 ...)

Note

In order to support NK4 pubsets (in later operating system versions), EAM users are requested to convert chained processing to the blocking factor 2 or a multiple of 2. In this case, odd block numbers (BLOCK#) should be specified for direct I/O operations, i.e. 1, 3, 5, ...

A program converted in this way is downward compatible, which means it can run in earlier versions.

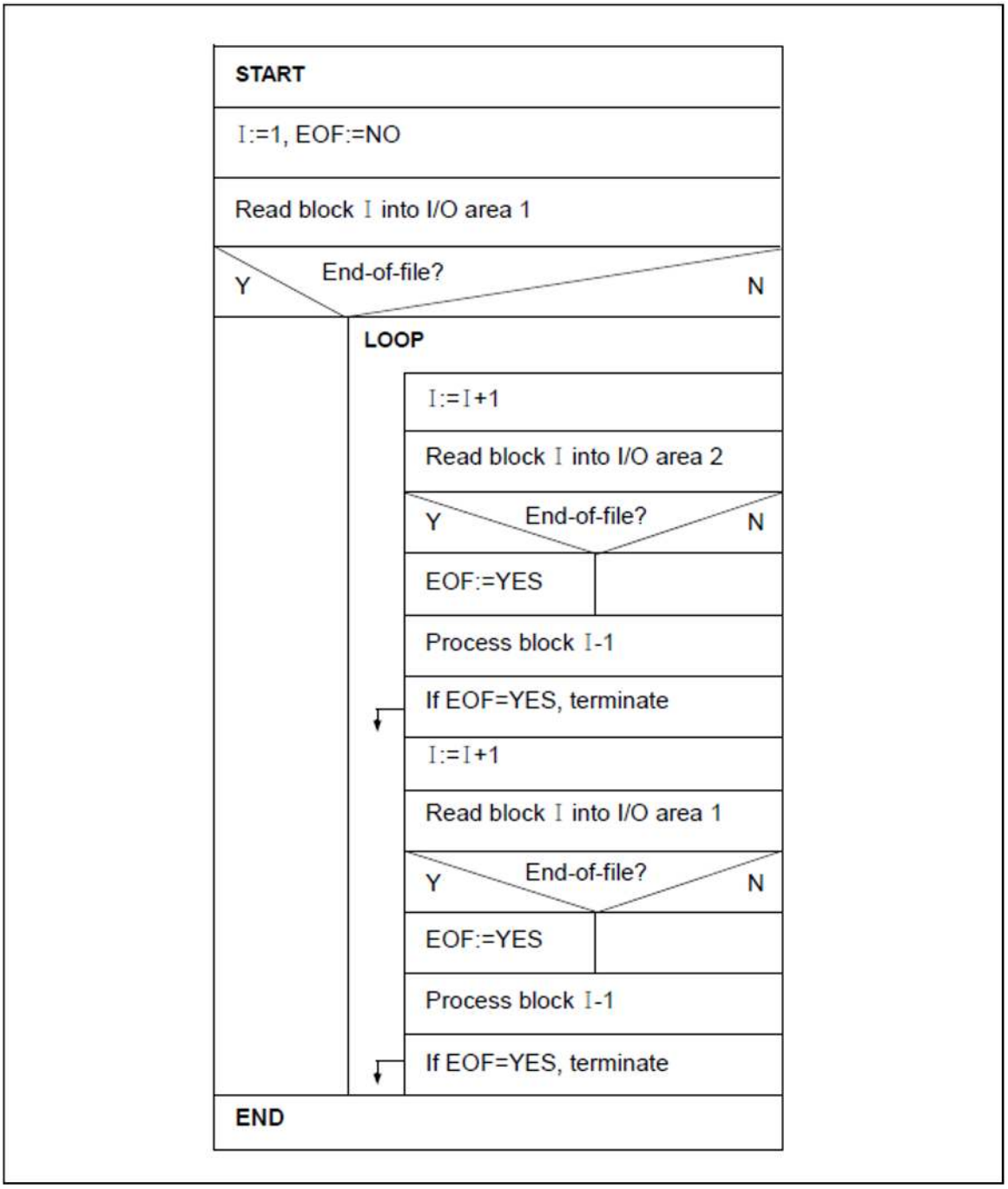


Figure 1: EAM – overlapping input/output

Handling object module files with EAM

Each job can process exactly one object module file. If bit IDMFOO of the option byte is set, all operations relate to the object module file. The actions involved in opening or reopening a file are illustrated in the following diagram:

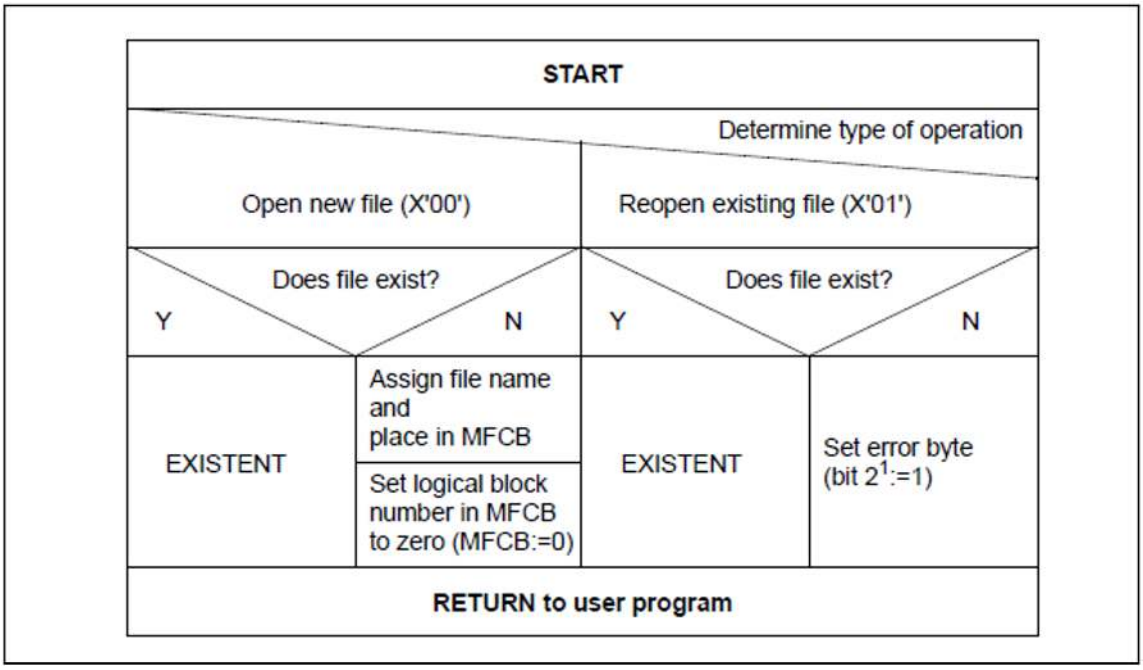


Figure 2: Actions when an EAM file is opened

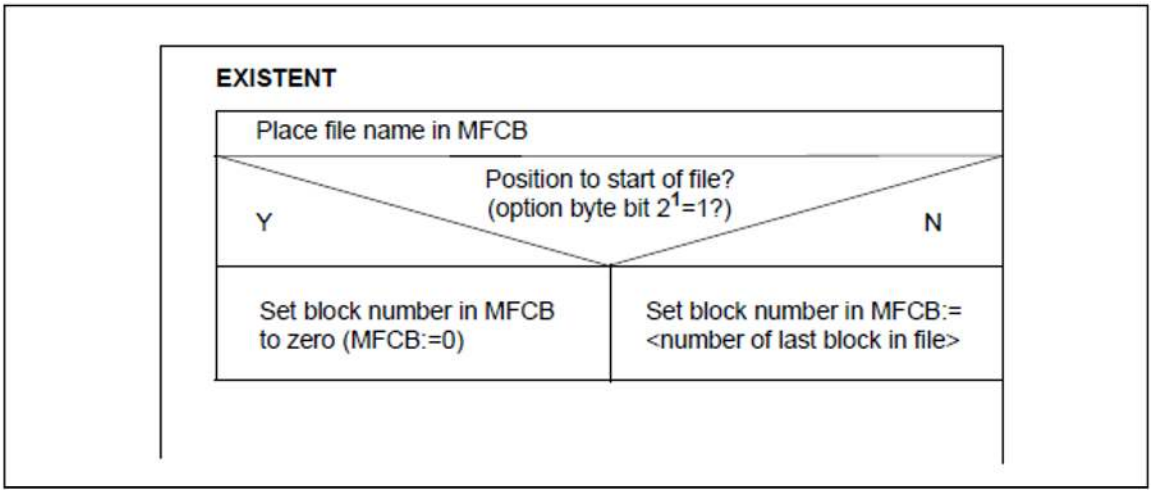


Figure 3: EAM – sequence when opening an object module file

FASTPAM - Fast Primary Access Method

FASTPAM (Fast Primary Access Method) is a block-access method for NK4 disk files. It is comparable with UPAM in terms of functionality, but is far superior to it in terms of performance, especially with multiprocessor systems.

With FASTPAM, I/O operations can be directly performed in data spaces. I/O area pools are placed in data spaces for this purpose, but these I/O area pools can only be created in non-resident memory.

The FASTPAM access method uses a special OPEN.

The FASTPAM interface is an SVC interface. Jobs are formulated by means of a parameter list, and return messages for results are supplied via a return code in the parameter list (not via exits).

Macros for the FASTPAM access method

The FASTPAM access method uses the following two macros:

Macro	Function
FPAMSRV	Management functions: • prepare system environment (FASTPAM environment) • prepare I/O areas (FASTPAM IO area pool) • open file for processing • close file opened with FPAMSRV • disable system environment (FASTPAM environment) • disable I/O areas (FASTPAM IO area pool)
FPAMACC	Access functions (formulate file accesses): • synchronous reading and writing of logical blocks • asynchronous reading and writing of logical blocks • waiting for the end of asynchronous I/O jobs • reporting the end of asynchronous I/O jobs

FASTPAM functions

The functions of the FASTPAM access method are implemented in the macros FPAMSRV and FPAMACC. These functions are as follows:

Function	Meaning
ENABLE ENVIRONMENT	Enable system environment for FASTPAM processing
ENABLE IOAREA POOL	Enable I/O area for FASTPAM processing
OPEN FILE	Open file for processing with FASTPAM
ACCESS FILE	Process a file (opened with FPAMSRV)
CLOSE FILE	Close a file (opened with FASTPAM), optionally specifying the last-page pointer.
DISABLE ENVIRONMENT	Disable system environment for FASTPAM processing
DISABLE IOAREA POOL	Disable I/O area for FASTPAM processing

*Enabling the system environment for FASTPAM processing (macro function FCT=*ENAENV)*

The ENABLE ENVIRONMENT function (macro FPAMSRV, operand FCT=*ENAENV) enables a user to create a FASTPAM environment or join an existing one. The caller is returned a task-specific environment short ID which can be used to refer to the environment in subsequent OPEN calls.

Since a FASTPAM environment is uniquely identified by its name and scope, and the scope is implicitly derived from the address of the FPAMACC parameter lists, the name as well as the address of the parameter lists must be specified in each ENABLE-ENVIRONMENT call. The other attributes need not be specified when joining an existing environment. If they are specified, however, they must match the corresponding values for the existing environment.

If the user has FASTPAM authorization, the entire class 3 memory area required for disk access is generated in advance, and the area of the FPAMACC parameter list is fixed.In order to do this, the address of the parameter list area, the number of parameter lists, and the maximum transfer length used for later file access are required.

The only values permitted for the transfer length are 4 KB and 32KB. A value of 32KB should be used only if the number of parallel access operations is not too high, since 2KB of resident system memory is used for each I/O path. The logical block size of files subsequently opened with this environment and the I/O length of the following file accesses must not exceed this maximum value.

Users who wish to work with eventing must specify the short ID of the event item at the time of creating the environment.

The parameter list area must be requested in advance and must allow write access. When the FASTPAM environment is being created by the first environment user (i.e. when the ENABLE ENVIRONMENT call is being processed), no I/O is permitted on any page that overlaps the parameter list area.

*Enabling the I/O area for FASTPAM processing (macro function FCT=*ENAIPO)*

The ENABLE IOAREA POOL function (macro FPAMSRV, operand FCT=*ENAIPO) enables a user to create a FASTPAM I/O area pool or join an existing one. Given the appropriate FASTPAM authorization, the operating system “fixes” the specified memory area and returns to the caller a task-specific I/O area pool short ID which can be used to refer to the pool in subsequent OPEN FASTPAM calls.

Like the FASTPAM environment, an I/O area pool is also identified uniquely by its name and scope. The attributes of the I/O area pool are fixed at the time it is created and cannot be changed, so no deviating attributes may be specified by other users of the same I/O area. The I/O area pool is typically joined by specifying just the name and the address.

The memory area must be requested in advance and must allow write access. When the FASTPAM I/O area pool is being created by the first I/O area user (i.e. when the ENABLE-IOAREA-POOL call is being processed), no I/O is permitted on any page of the I/O area pool.

*Opening files for processing with FASTPAM (macro function FCT=*OPEN)*

The short IDs obtained from ENABLE ENVIRONMENT and ENABLE IOAREA POOL can be used by any task to open any number of files.
If the environment is associated with an event item, each file can be opened by using the FPAMSRV macro with the operand FCT=*OPEN, EVENTNG=*YES. Every asynchronous ACCESS job is then acknowledged via the event item. Otherwise, each asynchronous job must be explicitly terminated by the user with a WAIT operation. Synchronous jobs are treated identically in both cases.
The access mode (read or write) can be defined with the MODE operand, and the SHARUPD operand can be used to define multiuser mode. The BLKSIZE operand determines the granularity of subsequent file access operations. The LARGE_FILE operand specifies whether the file that is to be opened may grow to become a “large file” with a file size >= 32 GB.
For each OPEN that is completed without error, the user is returned an OPEN short ID, which must be specified for following ACCESS FILE jobs.

As in the case of UPAM, parameter values specified in a previously issued ADD-FILE-LINK call override the values specified in the *OPEN call. If these values are not permitted for the FASTPAM-OPEN, the *OPEN call is rejected.

Processing a file opened with FPAMSRV (macro FPAMACC)

The FPAMACC macro can be used to write to the file and read from it. The file, its associated environment, and the I /O pool are identified by the OPEN short ID. I/O requests can be submitted both synchronously and asynchronously.

Synchronous operations are:

- READ AND WAIT
- WRITE AND WAIT
- READ AND EQUALIZE

Asynchronous operations are:

- READ
- WRITE

The WAIT operation is used to wait for the end of asynchronous jobs, i.e. jobs which are not executed synchronously.

In order to enable users to make efficient use of cached files, FASTPAM closes synchronously executed asynchronous jobs completely and does not send any signal to the event item when eventing is used. No WAIT macro (for eventing: no SOLSIG call) may be issued following a synchronously executed job.

The I/O length must be a multiple of BLKSIZE and must not exceed the value specified for MAXIOLN. In addition, specifications for the logical block within the file (BLOCK) and the address of the I/O buffer (IOAREA) are also required.

In order to avoid SVCs, job chaining is also supported. The CHAIN operand may be used to concatenate up to 5000 FPAMACC lists.

Eventing

Like UPAM, FASTPAM supports event-driven processing of I/O requests (see also section "TU eventing: event-driven processing" and the “Executive Macros” manual [2]). If a job is not terminated synchronously, FASTPAM sends a message to the associated event item on completion of an I/O operation. This message can be retrieved by the user with the SOLSIG macro. The message is sent if EVENTNG=YES is specified; no message is sent if EVENTNG=NO. A WAIT macro must be issued in any case.

*Closing a file opened with FASTPAM (macro function FCT=*CLOSE)*

The CLOSE function (macro FPAMSRV, operand FCT=*CLOSE) is used to close a file that was open. The file is identified by the OPEN short ID in this case as well.

*Disabling the system environment for FASTPAM processing (macro function FCT=*DISENV)*

The DISABLE ENVIRONMENT function (macro FPAMSRV, operand FCT=*DISENV) is used to disconnect a task from a FASTPAM environment that is specified by means of a short ID. When the last task is disconnected, the environment is disabled, but no user memory is released.

*Disabling the IOAREA for FASTPAM processing (macro function FCT=*DISIPO)*

The DISABLE IOAREA POOL function (macro FPAMSRV, operand FCT=*DISIPO) is used to disconnect a task from an I/O area pool specified by a short ID. When the last task is disconnected, the I/O area pool is disabled, but no user memory is released.

Processing files with FASTPAM

File format

FASTPAM will only process PAM files with BLKCTRL=NO/DATA and BLKSIZE=(STD,2n), where n=1,2,3...8. Files which do not have this format must first be converted.

FASTPAM authorization

In order to receive resident memory via FASTPAM calls, the user must be authorized for this purpose, which means that there must be an entry for the user in the user catalog (field `DMS-TUNING-RESOURCES=*EXCLUSIVE` in the SHOW-USER-ATTRIBUTES command or field `DMS-TUNING-RESOURCES=*EXCLUSIVE-USE` in the MODIFY-USER-ATTRIBUTES command). Users who do not have such authorization may also use the FASTPAM access method, but no resident areas are maintained. FASTPAM behaves like UPAM in such cases, i.e. only a small, non-resident part of the I/O path is created by the system, and the area of the parameter lists and the I/O area pool are not fixed. In other words, the path must be recreated, and user areas must be validated and fixed for each I/O. Performance gains typically achieved with FASTPAM are lost as a result.

If no memory can be made resident, FASTPAM behaves as if the required FASTPAM authorization were missing, thus offering a performance level equivalent to or better than that of UPAM.

Making memory areas resident

One of the primary purposes of FASTPAM is to enable rapid file access. This is done by making the required system environment available in resident memory before the first file is accessed.

In order to do this, the memory areas containing the user parameter lists and the I/O areas (both of which must be supplied by the user) are made memory resident by the “FASTPAM page fixing” mechanism.

This is essentially the same procedure that is performed by PPAM for the I/O area for every I/O operation when other access methods are used. The only difference with the other access methods is that the I/O area is released on completion of each I/O operation.

With FASTPAM, the user can define how long the parameter lists and the I/O area are (with ENABLE/DISABLE ENVIRONMENT and ENABLE/DISABLE IOAREA POOL) and can use them during that period. Validation is only required once at the beginning, since fixed areas cannot be released.

The ENABLE ENVIRONMENT function is also used to request the system memory that is required for I/O operations (once for each I/O operation that can be concurrently executed). A major part of this memory, i.e. the area used by IOCTRL, is always resident. This is also true for the other access methods, but the area is reallocated for each I/O and is not permanently reserved.

The rest of the system memory consists of a FASTPAM work area, which primarily contains the parameter list to call PPAM. In contrast to UPAM, if TU eventing is used, this area must also be in resident memory (since the I/O is then terminated in SIH).

The fixing of memory areas as described above is performed only if the user ID has the required FASTPAM authorization (entry in the user catalog, field `DMS-TUNING-RESOURCES=*EXCLUSIVE`).

If such an entry exists and if the appropriate memory areas are fixed, the resulting environment or I/O area pool is said to be “resident”.

A “resident environment” thus refers to:

- prevalidated parameter lists in resident memory
- system memory that is reserved in advance, and

  if eventing is used:

- a resident FASTPAM work area

Similarly, a “resident IO area pool” implies:

- prevalidated I/O areas in resident memory

Prerequisites for resident FASTPAM areas

- The user has specified the appropriate parameters
  (macro FPAMSRV, FCT=*ENAENV/*ENAIPO, operand RES=YES)
- The user has the required FASTPAM authorization
- No data spaces are being used
- An adequate amount of main memory is available
- A sufficient number of resident pages were allocated on calling the program (command START-PROGRAM /LOAD-PROGRAM, operand RESIDENT-PAGES); when resident pages are allocated in the program call, the maximum value defined in the user catalog and the system-global limit for resident memory pages must not be exceeded.

FASTPAM macros and their functions

Two macros are available to the user for processing files: FPAMSRV and FPAMACC. These macros can be used to execute various functions and operations (see the section "FASTPAM functions", and the macro and operand descriptions in the chapter "FPAMSRV - FASTPAM management function").

The FPAMSRV macro has the following functions:

ENABLE ENVIRONMENT	Enable system environment for FASTPAM processing
ENABLE IOAREA POOL	Enable I/O area for FASTPAM processing
OPEN FILE	Open file for processing with FASTPAM
ACCESS FILE	Process a file (opened with FPAMSRV)
CLOSE FILE	Close a file (opened with FASTPAM), optionally specifying the last-page pointer
DISABLE ENVIRONMENT	Disable system environment for FASTPAM processing
DISABLE IOAREA POOL	Disable I/O area for FASTPAM processing

The following function is implemented in the FPAMACC macro:

ACCESS FILE: Process a file (opened with FPAMSRV)

Multiuser mode on one computer

A PAM file can be created and/or processed with the UPAM access method (see "UPAM - User Primary Access Method"), FASTPAM or DIV (see "DIV - Data In Virtual").

The first user (User A) can select any combination of values for OPEN and SHARUPD (in the FCB macro) when opening his PAM file. The table 2 shows which OPEN and SHARUPD combinations a second user (User B) may use to open the same (already opened) file. If a file has been opened by more than one user, the OPEN/SHARUPD combination specified by each subsequent user (User B) is compared with all the existing opens (User A). each of these comparisons must yield a positive result before the file can be opened by the next user. Illegal combinations result in an OPEN error.

The following points apply to the FASTPAM access method:

- A file can be concurrently processed with FASTPAM by multiple tasks (multiple SHARUPD=*YES and MODE=*OUTIN/*INOUT opens).

  Note

  When a file is accessed in shared-update mode, appropriate synchronization routines must be supplied by the user if no such routines are built into the software product being used. In contrast to UPAM, FASTPAM does not provide any locking mechanism for this purpose.

- FASTPAM and UPAM openers

  A file can be opened in parallel by multiple tasks with FASTPAM as well as UPAM. Processing of the file is controlled by the operands MODE and SHARUPD (see below) of the OPEN function. Although FASTPAM does not support SHARUPD=WEAK, it otherwise behaves exactly like UPAM: both for FASTPAM openers exclusively and also when UPAM and FASTPAM openers are mixed.

  When a file is concurrently accessed with UPAM and FASTPAM, the UPAM user must also synchronize operations with the FASTPAM user, since UPAM page locks are only effective when used by both sides, and since FASTPAM has no page-locking mechanism.

- FASTPAM and DIV openers
  FASTPAM interacts with DIV exactly like UPAM. Parallel processing is permitted only if the file is opened with INPUT by all users.

Compatibility matrix: FASTPAM with UPAM/FASTPAM/DIV

FASTPAM does not support SHARUPD=*WEAK

			USER B	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**
	SHARUPD=		*YES	*YES	*YES	*NO	*NO	*NO	*WEAK	*WEAK	*WEAK
		OPEN mode	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN
USER A	*YES	INPUT INOUT OUTIN	X O O	O O O		X			X X X		
	*NO	INPUT INOUT OUTIN	X			X			X X X		
	*WEAK	INPUT INOUT OUTIN	X	X		X	X		X X X	X	

Table 2: FASTPAM: Permissible SHARUPD/OPEN combinations

X: OPEN permitted

O: OPEN permitted only

- if the same block-oriented access method is used by all (either UPAM/FASTPAM or DIV)
- and the same value is used by all for the LOCKENV operand
  (either LOCKENV=*HOST or LOCKENV=*XCS)
- and all are running on the same host or, with LOCKENV=*XCS, in an XCS network

Comments

- Read operations with SHARUPD=*WEAK may have opened a file simultaneously with any write operation (SHARUPD=*WEAK is only possible with UPAM and DIV).

  Exception:
  Parallel opening with a UPAM/FASTPAM write operation is not allowed for read operations with DIV-SHARUPD=*WEAK and which specified LOCVIEW=*MAP with OPEN.

  Read operations with DIV-SHARUPD=*WEAK and which specified LOCVIEW=*NONE with OPEN possess the same compatibility as read operations with UPAM/FASTPAM-SHARUPD=*WEAK.
- Openers with DIV-SHARUPD=*YES are not compatible to openers with UPAM/FASTPAM-SHARUPD=*YES.
- Read operations are always compatible to each other (regardless of access method, SHARUPD specification, LOCKENV specification and host).

- Illegal combinations lead to an OPEN error.

- SHARUPD=*YES:
  The file size is checked whenever the allocator is called.
  If this check indicates a file size >= 32 GB and the attribute LARGE_FILE=*FORBIDDEN is set in the associated FCB or the attribute EXCEED-32GB=*FORBIDDEN is set in the TFT then processing is canceled.
  In this case, FASTPAM returns the code `X'00400145'` in its local parameter list FPAMACC(I).

Multiuser mode with multiple systems

A multisystem environment is a configuration in which several systems are interlinked by means of shareable private disks (see the chapter “Volumes” in the “Introductory Guide to DMS” [1]) or shared pubsets. The table 2 shows the permissible combinations for access from two different systems.

The following points are relevant for the FASTPAM access method: A multisystem environment is a configuration in which several systems are interlinked by means of shared pubsets.

- Files on shared pubsets (shared PVS) are supported by FASTPAM: FASTPAM openers can access a shared PVS from different systems and read from it concurrently and can also read in parallel with UPAM and DIV openers.
- Remote file access (RFA) is not supported.

Data consistency

- Data consistency in multiuser mode

  The FASTPAM access method **does not** provide a synchronization mechanism for shared access to a file (shared-update mode). Appropriate synchronization routines must therefore be supplied by the **user** if no such routines are included in the software product being used.
  If FASTPAM, UPAM and DIV applications are all operating in shared-update mode, a common synchronization mechanism must be used for all accesses
- Data consistency following a system crash

  If an error occurs during an ACCESS FILE job, it is not possible to specify whether and how much data has been transferred. The writing of a block cannot be treated as an atomic operation. The contents of the file may be inconsistent in such cases.

Summary of functional differences between UPAM and FASTPAM

- FASTPAM can only be used to process PAM files with the following file attributes:
  - BLOCK-CONTROL-INFO=NO
  - BUF-LEN = an even number
- FASTPAM supports I/Os in data spaces.
- The following functions are not supported by FASTPAM:
  - DUMMY files
  - tape processing
  - RFA
- FASTPAM supports synchronous and asynchronous read and write operations. The following operations are offered by UPAM, but are not supported by FASTPAM:
  - CHK
  - LOCK / UNLOCK
  - LRD / LRDWT / WRTWU
  - SETL
  - SYNC
- The functionality of the UPAM operation SETLPP is included in the framework of FASTPAM-CLOSE processing.
- The function SHARUPD=*WEAK is not supported (see also "Compatibility matrix: FASTPAM with UPAM /FASTPAM/DIV").
- An implicit WAIT is not possible.
- Within an OPEN/CLOSE bracket asynchronous I/Os can either be terminated **only** by WAIT, or their termination can be reported **only** via the eventing mechanism.
- It is not possible to specify a relative page number.

ISAM - Indexed-Sequential Access Method

ISAM, like SAM, is a record-oriented access method for disk files. Processing is based on a file composed or index and data blocks. Each data record contains a key, and these act as a sort criterion (for index and data blocks, see the “Introductory Guide to DMS” [1]).

There are two versions of the ISAM access method, capable of processing files with two different block formats (see chapter “Access Methods” in the “Introductory Guide to DMS” manual [1).

- NK-ISAM (Non-Key ISAM) processes files with the block format “DATA”: These do not contain a separate PAM key. The DMS management information is kept in a block control field within the PAM page.
- K-ISAM (Key ISAM) processes files with the block format “PAMKEY”: These are characterized by the fact that DMS management information for each PAM page is kept in a separate PAM key located outside the page.

By means of the BLKCTRL operand in the macros FILE and FCB, the user can select whether a K file or an NK file is to be processed:

- BLKCTRL=DATA/DATA2K/DATA4K declares an NK-ISAM file
- BLKCTRL=PAMKEY declares a K-ISAM file

> As K-ISAM is becoming less and less relevant in practice, from BS2000 OSD/BC V11.0 onwards ISAM files are created as NK-ISAM on K disks by default as well.
>
> If the user makes no explicit specification for BLKCTRL, the BLKCTRL system parameter controls in general whether the file is created on K disks with the BLKCTRL=PAMKEY or DATA/NO attribute.
> If the file is to be created as ISAM file, the ISBLKCTL system parameter controls whether the file is created as an NK-ISAM file on K disks as well. The default setting of ISBLKCTL is c'NONKEY', i.e. an ISAM file is created as NK-ISAM file on K disks by default as well.

Macros for the ISAM access method

The macros for the ISAM access method can be split into function classes:

Administration: Macros with administration functions that support file processing.

Access: Macros that access the data on a file.

Macro call	Function class	Brief description
ADDPLNK	Administration	Assigns a pool link name to a user ISAM pool
CREAIX	Administration	Creates a secondary index for an ISAM file
DELAIX	Administration	Deletes secondary indices for an ISAM file
DELPOOL	Administration	Deletes a user ISAM pool
ELIM	Access	Erases a record from the file
GET	Access	Reads the following record in the file (sequential reading)

GETFL	Access	When using marked ISAM keys: reads the following record in the marking area (sequentially)
GETKY	Access	Reads the first record with the specified key
GETR	Access	Reads the previous record (sequential reading, in reverse)
INSRT	Access	Adds a record with a new ISAM key to the file
ISREQ	Access	Removes an ISAM lock
OSTAT	Access	Gives details of the number and type of synchronous file accesses
PUT	Access	Writes records sequentially to the end of file (including checking the key for valid sequence)
PUTX	Access	Replaces a record provided by GET, etc.
REMPLNK	Administration	Deletes the pool link name
RETRY	Access	Resets the ISAM pointer after execution of the EXLST-PGLOCK and repeats the last macro
SETL	Access	Positions the ISAM pointer for subsequent sequential processing at start or end of file or a particular record
SHOPLNK	Administration	Gives details of assignment of ISAM pools to pool link names
SHOPOOL	Administration	Gives details of attributes and allocation states of ISAM pools
STORE	Access	• Adds a record with a new ISAM key to the file • Overwrites a record with an existing ISAM key, of multiple assignment of ISAM keys is not permitted • Adds a record with an existing ISAM key to the file as the last record with this key

OPEN modes

Before a file can be processed, it must be opened using an OPEN macro. The following OPEN modes are permitted for ISAM files: OUTPUT, OUTIN, EXTEND, INOUT, INPUT. At the same time, it is necessary to check whether the file has already been opened by another job, in which ISAM pool it is to be processed, whether it is to be processed in locate or move mode, etc.

OUTPUT	A new file is created sequentially and only the PUT macro may be used. If an ISAM file with the specified name already exists, it is overwritten and the catalog entry is updated.
OUTIN	As for OPEN OUTPUT, a new file is created and any existing file with the specified name is overwritten, but all ISAM actions are permitted.
EXTEND	An existing file is extended sequentially; as for OPEN OUTPUT only write operations with PUT are permitted.
INOUT	An existing file is to be updated: as for OUTIN, all ISAM actions (such as finding, reading, updating, inserting and deleting records) are permitted.
INPUT	An existing file is to be read, i.e. only read operations are permitted.

Operating modes

ISAM files are normally processed in move mode; locate mode can be used, but in NK-ISAM this is only still supported for reasons of compatibility.

Action macro	**OPEN type INPUT**	**OPEN type OUTPUT**	**OPEN type EXTEND**	**OPEN type INOUT**	**OPEN type OUTIN**
GET	B	-	-	B	B
GETR	B	-	-	B	B
GETFL	B	-	-	B	B
GETKY	B	-	-	B	B
PUT	-	B	B	B	B
PUTX	-	-	-	B	B
INSRT	-	-	-	M	M
STORE	-	-	-	M	M
ELIM	-	-	-	x	x
SETL	x	-	-	x	x

where:

M Move mode (locate mode only if a work area is supplied).

B Move mode or locate mode permitted.

x Action macro may be used.

- Action macro must not be used.

OPEN errors when processing NK-ISAM files

NK-ISAM file on tape: an invalid data format was specified in the FILE or FCB macro when importing the tape file or writing it back to tape.

The ISAM pool in which an NK-ISAM file is to be opened is overloaded, OPEN processing is rejected with error message DMS0D9B.

On each SVC entry, the size of the NK-ISAM file in question is checked on the basis of the extent list present in the File Entry Table. If this check indicates that the file size is greater than 32 GB and if the caller has set the attribute LARGE_FILE=*FORBIDDEN in the FCB then processing is canceled. In this case, NK-ISAM issues the return code: `X'00000A23'` (`FILE SIZE GREATER 32 GIGABYTES IS NOT ALLOWED`).

Transferring file attributes to the FCB

The contents of FCB fields for file attributes may differ from the values specified for the corresponding operands in the FILE macro. This is true, for example, for KEYPOS, KEYLEN, PAD and BLKSIZE. If “n” is the value in field KEYPOS and “m” is the value of KEYLEN in the catalog entry or in the TFT or the FCB macro, the following applies to files which are open:

- field KEYLEN in the TU FCB contains m-1
- field KEYPOS in the TU FCB contains, for RECFORM=F, (n+4)-1 = n+3

The PAD value is taken into account only when a file is created sequentially by means of the PUT macro. The macros INSRT and STORE use all the available space of each data block. Block splitting may occur during creation of the file in both functions.

The contents of field BLKSIZE in the TU FCB may also differ from the value of the BLKSIZE operand in the FILE or FCB macro or the catalog entry: except for input files, the BLKSIZE is recalculated, taking the PAD value into account (BLKSIZE minus PAD) and placed in FCB field BLKSIZE.

Example

> Catalog entry: BLKSIZE=(STD,3); PAD=15 (default value). The contents of BLKSIZE in the TU FCB are calculated as follows: (3 * 2048) * (1.0 – 0.15); during file processing, the field BLKSIZE contains the value X'1467'.

Dummy files

Dummy files can be used for test purposes, particularly for testing error routines in user programs. These dummy files are defined by *DUMMY in the FILE macro. No actual I/O operations are executed when these files are processed. Any attempt to read such a file causes control to be passed to an error routine and write attempts are ignored. In each case, a no-op (no operation) is executed.

The following table shows which events occur for the various ISAM read operations.

Read operation	Macro	Event	EXLST exit	Message

Sequential read	GET/GETR	End of file	EOFADDR	DMS0AAE
Read with key	GETKY	Key not found	NOFIND	DMS0AA8
Read with flags	GETFL LIMIT=KEY	Key not found	NOFIND	DMS0AA8
	LIMIT=END	End of file	EOFADDR	DMS0AAE

Block usage: PAD value

During sequential creation of a file with PUT, the user can specify via the PAD operand in the FILE/FCB macro how much space is to be left free in each data block. This space is needed if the records in the file are subsequently updated and extended. The default value is PAD=15, i.e. 15% of the space in each data block is left free. If a PUT macro would result in less free space in an existing block, a new data block is created for the next record.

For NK-ISAM files, a new data block is requested as soon as this limit is exceeded; K-ISAM requests a new data block before the limit is exceeded.

For sequential creation with PUT, the size of a file increases if the PAD factor is increased. However, subsequent processing of the file (STORE/INSRT) can be optimized by selecting a suitable value for PAD: the free space left in the data blocks should be so large that block splitting does not occur when the file is subsequently extended. In order to select the correct PAD value, it is thus necessary to estimate how the file will grow.

If ISAM files are created using STORE, the PAD value has no effect on how the blocks are filled. STORE writes records into a block until it is full. If a further record is then written, block splitting occurs. The blocks are usually only 50% full.

Even if the file is created by means of PUT, but not via an I/O area in the program, the PAD has no effect (the I/O area is defined in the FCB with IOAREAn; see “Program buffer”, below). Each PUT macro initiates a write operation and DMS attempts to place the new record in the current data block. A new data block is created only when the current block is full.

Program buffer = I/O area in the user program

If a program uses its own I/O area, this must be at least as large as a data block (= n * 2048 bytes, where 1 <= n <= 16), as specified in BLKSIZE=(STD,n). By default, the system generates an I/O area in class 5 memory.

The existence of an I/O area defined by the user via IOAREA1/2 in his/her program is particularly advantageous for the sequential processing of ISAM files, since it reduces the number of SVCs:

- sequential read operations (GET/GETR): in the first read operation, as many records as possible are transferred to the I/O area before the first record is returned to the program. For subsequent read operations, the records already in the I/O area are returned to the program and a new I/O operation is executed only when all of these records have been passed to the program.
- sequential write operations (PUT): during the sequential creation or extension of a file, the records are collected in the I/O area until it is full (taking the PAD value into account) or until sequential processing is terminated by a call for another operation. Care should be taken that sequences of PUT operations are not interrupted by other actions, since each such interruption causes the contents of the I/O area to be written as a new data block and this can lead to blocks with relatively small amounts of data.

For NK-ISAM in move mode, the user can dispense with an I/O area (IOAREA1=NO in the FCB). In this case, each action macro call leads to an SVC.

ISAM pointers

Macros which refer to a record that was processed by a preceding macro use internal “pointers” to determine the current position within the file.

A separate pointer is maintained for the primary key and for each secondary key defined for the file. A successful positioning or read operation via a secondary key (using SETL, GET, GETKY or GETR) first modifies the pointer for the secondary key and then, with the aid of the updated secondary key value, sets the pointer for the primary key to the appropriate record.

Macros which evaluate the ISAM pointers are all sequential macros or, for example, PUTX, which writes a previously read record back to the file.

If a macro cannot be executed completely, e.g. because an error or a PGLOCK condition occurs, then the pointer is reset, for NK-ISAM, to the value it had before the macro was called. The only exception to this is the error “NOFIND”: since the desired record could not be found, it is not possible to position the pointer to this key.

ISAM pointers are generally updated before the macro is executed. However, for sequential read operations (GET, GETR), the preceding macro must be taken into account: if a sequential read operation is preceded by a SETL macro, then this SETL acts as a “positioning macro”.

The ISAM actions OSTAT (Open STATus) and ISREQ have no effect on the pointer position since they do not initiate any actions in the file.

When a file is opened, a “SETL B” is initiated internally as the first action for an ISAM file, i.e. the pointer is positioned to the first record in the file.

Rules for ISAM pointers

Action macro	Pointer	Action	Comments
ELIM	KEY not specified: Pointer is not moved. KEY specified: Pointer is moved to the first record with the specified key.	Eliminates a record from the file.	ELIM can be regarded as a “left-shift” operation on that portion of the file to the right of the defined record. Thus, if ELIM is successful, the pointer does not have to be updated.
GET	The pointer for the primary or secondary key specified in the macro is moved one record towards the end of the file.	Retrieves the record referenced by the pointer.	If the pointer points to a record outside the file, the user is given control at the EOFADDR exit. If the preceding macro was a SETL or ELIM (with KEY), the pointer is not updated until the record is retrieved.
GETFL	The pointer is moved to the record retrieved, or to the last record in the defined area.	Retrieves the next sequential record which satisfies the flag criteria.	The record which would have been retrieved by a corresponding GET or GETR macro is the first record examined for the flag criteria.

GETKY	The pointer is set to the record with the specified primary or secondary key or to the position in the file where this record should be.	Retrieves the record pointed to by the pointer, assuming the record with the specified key exists.	A GETKY for a key without a corresponding record is equivalent to a GETKY for a record of zero length that is assumed to be located between two existing records. Thus, a subsequent GET or GETR will work correctly (see GET).
GETR	The pointer for the primary or secondary key specified in the macro is moved one record towards the beginning of the file.	Retrieves the record referenced by the pointer (see reverse read)	If the pointer points to a record situated before the beginning of the file, the user is given control at the EOFADDR exit (see GET).
INSRT	The pointer is moved to the position specified by the record key.	Inserts the record at the location indicated by the pointer.	The record is not inserted if a record with the same key already exists. However, the pointer still points to the position in the file at which the record was to be inserted. A GET macro issued after an unsuccessful INSRT retrieves the duplicate record.
OPEN	The pointer is positioned before the first record in the file.		See "OPEN - Open file"
PUT	The pointer is positioned immediately after the current end of the file.	Places the record in the file at the position indicated by the pointer.	
PUTX	The pointer does not change its position.	Places the record in the file at the position indicated by the pointer.	A check is made to verify that a GET, GETR, GETFL or GETKY macro was called immediately prior to PUTX. This ensures that no errors occur even though the pointer was not updated.
RETRY	Contingent on the operation to be repeated, or on the positioning operation.	Repeats the last action macro for the file or positions the pointer to the origin or places the program in a queue.	RETRY itself does not affect the pointer except when it results in the execution of an action macro which does move the pointer.

SETL	*SETL B:* The pointer is positioned in front of the first record in the file. *SETL E:* The pointer is positioned after the last record in the file. *SETL KEY:* The pointer is set to the first record with the specified primary or secondary key or to the record with the next-higher key value.	No action	Inhibits the pointer update by a following GET or GETR macro. Under the pointer concept, a SETL for a non-existent record can, when followed by a GET or GETR macro, be interpreted as a SETL for a record of zero length at the correct position. In this case the SETL does not cancel updating of the pointer by the following GET or GETR macro.
STORE	The pointer is moved to the position specified by the key. DUPEKY=YES: If a duplicate key is encountered, the pointer is positioned after the existing record.	Writes the record at the desired location.	If a record with the same key already exists in the file, it is overwritten (unless DUPEKY=YES was specified). If DUPEKY=YES applies, the new record is written after the “old” record.

SAM - Sequential Access Method

SAM is a record-oriented access method used to process files sequentially. SAM can be used to write to, update and read records. SAM also features a positioning function which permits positioning at the logical start or end of a file or at any existing record.

As SAM is a record-oriented access method, it carries out the blocking, deblocking and buffering of the records for the user. If two I/O areas are available in the user program, exchange buffer operation can be used. If only one I/O area is provided, no overlapped processing is possible.

SAM is virtually device-independent and permits the processing of files on disks and tapes; tape cartridges are essentially handled like normal magnetic tapes.

The access method SAM is capable of processing files with different block formats (see the section dealing with access methods in the “Introductory Guide to DMS” [1]).

- K-SAM files, i.e. files with a PAM key, have the conventional block format “PAMKEY”. They are characterized by the fact that DMS management information for each PAM page is held in a separate PAM key located outside the page.
- NK-SAM files, i.e. PAM files without a PAM key (non-key PAM files), have the block format “DATA” or “NO”. They do not contain separate PAM keys. With the block format “DATA”, the DMS management information is kept in a block control field within the PAM page.
  No such block-specific management information is supported for the block format “NO”.

By way of the BLKCTRL operand in the macros FILE and FCB, the user can select whether a K file or an NK file is to be processed: BLKCTRL=PAMKEY defines a K-SAM file, BLKCTRL=DATA or BLKCTRL=NO specifies an NK-SAM file.

Macros for the SAM access method

The following action macros are available for processing files with SAM:

Macro	Function
FCB	set up file control block.
FEOV	for tape files: initiate tape swap.
GET	read sequentially; records are retrieved one after the other.
PUT	write sequentially: in move mode, record blocking is handled by the logical routines of the access methods i.e. output to the data volume is delayed until the output buffer is full; the buffers are served automatically by the system; in locate mode, blocking must be handled by the user.
PUTX	a record that has been read is replaced (for disk files in locate mode only).
RELSE	terminate a data block i.e. for input files:the next GET causes the next data block to be read; for output files: the next PUT causes the buffer contents to be written as a data block, the next record becoming the first record in the new data block (this is necessary in locate mode if the next record no longer fits in the current buffer).
SETL	position to a certain record in the file, to beginning-of-file or to end-of-file.

OPEN modes

INPUT	read sequentially towards the end of the file; the file must exist.
OUTPUT	create a new file sequentially or overwrite an existing file.
EXTEND	extend file.
UPDATE	for disk files in locate mode only: update records; the record to be updated must be retrieved by means of GET; the length of the record must not be changed during processing; the updated record is replaced by means of PUTX.
REVERSE	read sequentially towards the beginning of the file; the file must exist; tape files which extend over more than one volume can only be read volume by volume with the aid of the VSEQ operand (see "FILE - Define file attributes / control file processing"); no automatic tape swap.

The following table shows which OPEN modes are possible for which SAM action macros.

Action macro (SAM access method)	**OPEN type INPUT**	**OPEN type OUTPUT**	**OPEN type EXTEND**	**OPEN type UPDATE**	**OPEN type REVERSE**
GET	x			x	x
PUT		x	x		
PUTX				x	
RELSE	x	x	x	x	x
SETL	x	x	x	x	x

If files are opened as output files (OPEN OUTPUT/EXTEND), SAM interprets each PUT or SETL macro as an end-of-file (EOF) indicator. The last PUT or SETL macro prior to a CLOSE macro for a file thus automatically indicates EOF to the system. If the user wishes to delete all records after a specific record in a file, he/she can use the SETL macro to position to the desired point in the file, and then issue a CLOSE macro to close the file.

For direct access, a retrieval address is made available to the user. The format of this retrieval address is described in detail in the “SAM” chapter of the “Introductory Guide to DMS” [1]. When a record is written, its retrieval address is made available in the FCB. If the user wishes, he/she can create a new file from this retrieval address data to serve as a basis for subsequent non-sequential processing of this file. The retrieval address is also made available in the FCB after execution of a GET macro. In this case too, therefore, even if the user did not create the file, he /she can still create a secondary file from the retrieval addresses in order to perform subsequent non-sequential processing of the file.

If, in move mode with two output buffers, physical end-of-volume is detected when a data block is written to a tape file, the other buffer (which contains only one record) is written to the other tape. A tape swap is not executed until this has been done.

If a file is to be created in locate mode (OPEN OUTPUT/EXTEND), the start address of the first record to be written is returned to the user, after OPEN, in the register specified in the operand IOREG in the FCB macro. After execution of the PUT macro, this register contains the address for the next higher record.
In the case of a file with variable-length records (RECFORM=V), the user also always receives the number of free bytes in the current block. This information is passed in the register he/she specified in the VARBLD operand of the FCB macro.

Primary and secondary allocations (disk files)

When a SAM file is created or overwritten (OPEN OUTPUT) or extended (OPEN EXTEND), both the primary and secondary allocations must be at least the same as the block size.

If a file for which RECFORM=F or RECFORM=V is specified is to be created or extended in move mode (OPEN OUTPUT/EXTEND) and is to have more than one record per data block, then the following conditions apply:

- The primary allocation must be at least twice as long as the data block (primary allocation >= 2 * BLKSIZE). Otherwise control will be passed to the EXLST exit NOSPACE (insufficient storage space).
- The secondary allocation defined by SAM is implemented as of the first record of the last block that can still be written in the assigned area. If the secondary allocation cannot be implemented, control passes to the EXLST exit NOSPACE (provided it is available in the program), thereby giving the user the opportunity to write the remaining records of the last block.

Effects of the LABEL operand (tape files)

NO / NSTD	Specification of the BUFOFF operand will lead to an OPEN error with CODE=ISO/OWN and BLKSIZE=STD.
(STD,0)	Specification of the BUFOFF operand and CODE=ISO/OWN will lead to an OPEN error.
(STD,1)	Specification of the BUFOFF operand and the entry CODE=ISO/OWN will lead to an OPEN error.
(STD,2)	Standard blocks are converted into nonstandard blocks (BLKCTRL=DATA/NO) and V-records are converted to D-records; RECSIZE > 9999 together with RECFORM=V will lead to an OPEN error.
(STD,3)	Standard blocks are converted into nonstandard blocks (BLKCTRL=DATA/NO) and V-records are converted to D-records; RECSIZE > 9999 together with RECFORM=V will lead to an OPEN error.
STD / no LABEL operand specified	Specification of the BUFOFF operand will lead to an OPEN error.

[1)] Length specified in decimal form in the case of D-records.

The combination of the specifications for the operands CODE, RECSIZE and RECFORM result in the following values for LABEL if the file to be created is the first file on a tape:

CODE	Block format	RECFORM	LABEL value (implicit)
EBCDIC	PAMKEY	-	(STD,1)

EBCDIC	DATA/NO	U	(STD,2)
EBCDIC	DATA/NO	F/V	(STD,3), V-records -> D-records
ISO/OWN	PAMKEY	-	OPEN error

ISO/OWN	DATA/NO	U	(STD,2)
ISO/OWN	DATA/NO	F	(STD,3)
ISO/OWN	DATA/NO	V	V-records -> D-records (STD,3) OPEN error if BLKSIZE > 9999

Table 3: Effeccts of the CODE BLKSIZE, RECFORM operands on the LABLE operand

The combination BUFOFF and RECFORM=U or BLKCTRL=DATA is not permitted.

In the case of LABEL=(STD,1), it is not possible to write a file with CODE=EBCDIC, with nonstandard blocks and D-records. However, it is possible to read such a file.

An OPEN error will occur if the standard label level requested via the LABEL operand does not match that specified in the VOL1 label.

If LABEL=STD is specified or the LABEL operand is omitted, the label level is determined as shown in Table 55. If the resulting label level is higher than that in the VOL1 label, the label level defined in the VOL1 label is used. If the resulting label level is lower than that in the VOL1 label, or if the VOL1 label specifies LABEL=(STD,0), and if CODE=ISO/OWN is specified, an OPEN error will result.

SAM file record formats

The SAM access method permits the record formats: F (fixed record length), V (variable record length) and U (undefined record length).
When format U is used, SAM reads or writes only one record per data block (buffer). For example, the definition RECFORM=U in combination with BLKSIZE=STD, BLKCTRL=PAMKEY and a current record length of 48 bytes would mean that 2000 bytes in each PAM page would be “wasted”.

If the full capacity of a standard block is not used (e.g. after the action macro RELSE, SETL, FEOV or CLOSE), the remaining bytes in the block are unchanged, which means that their contents are undefined.
The size of the record must not exceed the block size (see the BLKSIZE operand in the FILE/FCB macro).

Further information on record format can be found in the section dealing with access methods in the “Introductory Guide to DMS” [1].

FCB retrieval address

When creating a SAM file, DMS places a retrieval address in the FCB. This address can be used for positioning by means of SETL. It consists of block numbers and record numbers. The block number always refers to a logical data block (not to a PAM page), while the record number indicates the position of the record within the indicated block. For multivolume files, it should be noted that the block number is maintained only within each volume.

In the 31-bit TU FCB, i.e. for XS programming, the retrieval address is split into two fields, each one word long: the field ID1BLK# contains the block number within the file and the field ID1REC# contains the record number within this block. Both fields are incremented automatically by the system. Whenever a data transfer operation is initiated, the record counter is automatically reset.

For non-XS programming, the retrieval address is kept in field ID1RPTR of the 24-bit TU FCB in the form “bbbbbbrr”, where “bbbbbb” is the number of the data block in the file and “rr” is the number of the record within the block. The record counter is not automatically incremented by the system; the user must do this in his program if he /she wants to use the retrieval address. However, the record counter is automatically reset whenever a data transfer operation is initiated.
For tape files, it should be noted that the retrieval address is returned only for files with standard blocks, which means that SETL R (see the SETL macro, "SETL - Position file pointer") cannot be used for files with nonstandard blocks.

Structure of the retrieval address:

ID1BLK# (1 word)	ID1REC# (1 word)
Block number	Record number

Table 4: Structure of the retrieval address XS interface

ID1RPTR[1] byte 1-3	ID1RPTR[1] byte 4
Block number	Record number

Table 5: Structure of the retrieval address Non-XS interface

[1] 1 word

The first record in a file thus has the following retrieval address:

in the 31-bit TU FCB	00000001 in field ID1BLK# and 00000001 in field ID1REC#
in the 24-bit TU FCB	00000101 in field ID1RPTR

The following has to be noted for processing SAM node files:

If a data block is closed with RELSE during a write process on a SAM node file before it has been filled completely, the retrieval addresses are only valid from that moment on for as long as the file is open.
CLOSE and re-OPEN results in a different distribution of the data sets of the node file on the SAM blocks, which are transferred to the access method for processing; this means the previous retrieval addresses do not match the file anymore!

This behavior of SAM node files (Net-Storage) is incompatible with the processing of SAM files on conventional public space, where the block and record structure remains unchanged even after CLOSE and OPEN. Because of this, the application must set the indicator SAM_NODE_FILE_ENABLE in the file control block (FCB) before it opens the file, to indicate that it is capable of processing SAM node files correctly.

Retrieval address values for SETL B and SETL E, depending on the OPEN type:

OPEN type	**SETL B ID1BLK#**	**SETL B ID1REC#**	**SETL B ID1RPTR**	**SETL E ID1BLK#**	**SETL E ID1REC#**	**SETL E ID1RPTR**
INPUT, UPDATE	-	-	-	max.	1	max. 1
OUTPUT	1	0	1 0	error	error	error
EXTEND	1	0	1 0	error	error	error

REVERSE	-	-	-	-	-	-

where:

- - Field contents unchanged.
- max Highest block number.

Field IDRPTR contains both the block and record counters.

The action macros return the retrieval address as follows:

GET

if a data transfer is necessary for the requested record, the block counter contains the logical block number and the record counter is reset. For 31-bit FCB, the record counter is updated by each action macro.

PUT

if the PUT macro results in a data transfer, the block counter is updated and the record counter is reset. The block counter thus contains the number of the new data block which will receive the record. For 31-bit FCB, the record counter is updated by each action macro.

RELSE

if a file is being created or extended (OPEN OUTPUT/EXTEND), the block counter contains the number of the block which will receive the following record and the record counter is set to zero.

FEOV

after a tape swap, the block and record counters are reset for the new tape by the system.

Example

File attributes: BLKCTRL=PAMKEY, BLKSIZE=(STD,2), RECFORM=F, RECSIZE=512

	Retrieval address 31-bit TU FCB ID1BLK#	**Retrieval address** 31-bit TU FCB ID1REC#	**Retrieval address** 24-bit TU FCB IDRPTR
Record 10	00000002	00000002	00000202
Record 20	00000003	00000004	00000304

UPAM - User Primary Access Method

UPAM is the primary, block-oriented access method in BS2000 for random access to disk files. Read or write access to any block of a file is possible at any time.

Tape files may likewise be processed using UPAM (see below).

By way of the BLKCTRL operand in the macros FILE and FCB, the user can select whether a K file or an NK file is to be processed: BLKCTRL=PAMKEY defines a K-PAM file, BLKCTRL=DATA or BLKCTRL=NO specifies an NK-PAM file.

The format of the file created on an NK2 disk depends on how the block size is specified. If the block size is specified as BLKSIZE=(STD,n), and n is even, an NK4-PAM file is created; if n is odd, an NK2-PAM file is created. An NK4 disk can only be used for an NK4-PAM file (see also the section dealing with access methods in the “Introductory Guide to DMS” [1]).

Multiple 2048-byte standard blocks can be combined to form a single data block (logical block) by specifying BLKSIZE=(STD,n) (with n > 1) in the FCB or FILE macro. In the case of a K-PAM file, each standard block within the logical data block can be addressed in the program.

For an NK-PAM file, it is only possible to address a complete data block from within a program; separate processing of the individual 2048-byte blocks which make up the data block is not possible.

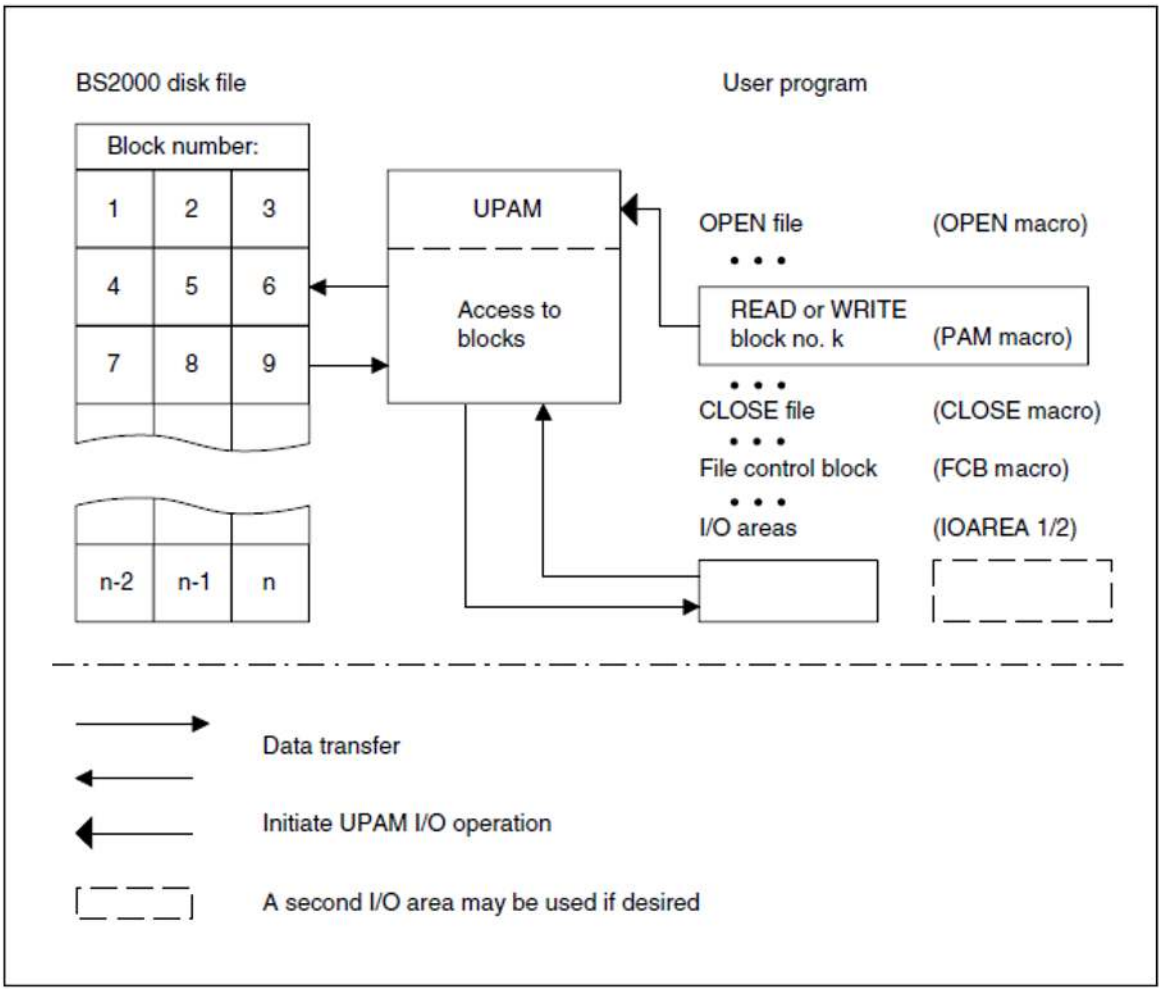


Figure 4: Principles of UPAM operation

Macros for UPAM access methods

Service macros

Macro	Operation	Function
OPEN		open file.
CLOSE		close file.
FCB		define file control block.
EXLST		define error exits.
PAM	CHK	check status of I/O processing.
	LOCK	lock a PAM page.
	LRD	lock a PAM page and read its contents into main memory.
	LRDWT	lock a PAM page, read its contents into main memory and wait for completion of I/O.
	RD	read PAM page into main memory.
	RDEQU	same as RDWT (see below), but the copy is also updated in the case of files with DRV (Dual Recording by Volume; see (the “DRV” manual [15]).
	RDWT	read PAM page into main memory and wait for completion of I/O.
	SETL	set file pointer.
	SETLPP	change last-page pointer (EOF pointer); this enables files to be reduced in length, i.e. after the CLOSE call has been issued, PAM pages which are no longer required can be released. This operation is not permitted for files which have been opened with SHARUPD=WEAK/YES or OPEN=INPUT. SETLPP is ignored in conjunction with tape files.
	SYNC	wait for completion of I/O operation and write the contents of the tape cartridge buffer to tape.
	UNLOCK	unlock PAM page.
	WRT	write data from main memory to a PAM page.
	WRTWT	write data from main memory to a PAM page and wait for completion of I/O.
	WRTWU	write from main memory to a PAM page, wait for completion of I/O operation and unlock the PAM page that has been written.
	WT	wait for completion of I/O operation.

Macros used for eventing

The ENAEI, DISEI and SOLSIG macros must be used in all types of eventing. For further information on the macros listed below, refer to the “Executive Macros” manual [2].

CHKEI	checks the status of an event item.
CONTXT	enables read or write access to the register set and program counter (the “context”) of an interrupted contingency process or of the basic task.
DISEI	detaches a job from an event item.
DISCO	prevents a contingency definition from controlling contingency processes.
ENACO	allows a contingency definition to control contingency processes.
ENAEI	assigns an event item to a job.
LEVCO	changes the priority level of a contingency process or of the basic task.
RETCO	terminates a contingency process.
SOLSIG	sends a request to an event item.

OPEN modes

INPUT	read blocks from an existing file.
OUTIN	create a new file and, if required, read blocks from this file.
INOUT	read blocks from an existing file and, if required, add and/or exchange blocks.

PAM operations and OPEN modes

PAM macro functions	OPEN mode INPUT	OPEN mode OUTIN	OPEN mode INOUT
RD, RDWT, RDEQU, LRD, LRDWT	X	X	X
WRT, WRTWT, WRTWU	-	X	X
WT, CHK, SYNC	X	X	X
LOCK, UNLOCK, SETL	X	X	X
SETLPP	-	X	X

Multiuser operation

A UPAM file can be created and processed with the UPAM, FASTPAM (see "FASTPAM - Fast Primary Access Method") or DIV (see "DIV - Data In Virtual") access methods. FASTPAM and DIV can, however, only process UPAM files with the attribute BLKCTRL=NO.

Authorization for parallel file processing is dependent on the FCB operand open values specified for SHARUPD, MODE, LOCKENV and LOCVIEW (the FCB OPEN operand corresponds to the MODE operand of macros DIV and FPAMSRV).

The allowed parallel opens are shown in the following table:

Compatibility matrix with UPAM-OPEN

			USER B	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**	**USER B**	**USE**
	SHARUPD=		*YES	*YES	*YES	*NO	*NO	*NO	*WEAK	*WEAK	*WE
		OPEN mode	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN
USER A	*YES	INPUT INOUT OUTIN	X O O	O O O		X			X X X		
	*NO	INPUT INOUT OUTIN	X			X			X X X		
	*WEAK	INPUT INOUT OUTIN	X	X		X	X		X X X	X	

Table 6: UPAM: Permissible SHARUPD/OPEN combinations

X: OPEN permitted

O: OPEN permitted only

- if the same block-oriented access method is used by all (either UPAM/FASTPAM or DIV)
- and the same value is used by all for the LOCKENV operand
  (either LOCKENV=*HOST or LOCKENV=*XCS)
- and all are running on the same host or, with LOCKENV=*XCS, in an XCS network

Read operations with SHARUPD=*WEAK may have opened a file simultaneously with any write operation.

> Exception:
> Read operations with DIV-SHARUPD=*WEAK which specified LOCVIEW=*NONE with OPEN possess the same compatibility as read operations with UPAM/FASTPAM-SHARUPD=*WEAK.

- Openers with DIV-SHARUPD=*YES are not compatible with openers with UPAM/FASTPAM-SHARUPD=*YES.
- Read operations are always compatible with each other (regardless of access method, SHARUPD specification, LOCKENV specification and host).
- Illegal combinations lead to an OPEN error.
- An attempt to open a tape file with SHARUPD=YES or WEAK also leads to an OPEN error.
- If the block size of a tape file without PAM key is not a multiple of 2048, any attempt to open it with FCBTYPE=PAM is also rejected by UPAM with an OPEN error.

- SHARUPD=*YES:
  The file size is checked whenever the allocator is called. If this check indicates a file size >= 32 GB and the attribute LARGE_FILE=*FORBIDDEN is set in the associated FCB then processing is canceled.
  In this case, UPAM issues the return code `X'000009AD'` (`FILE SIZE GREATER 32 GIGABYTES IS NOT ALLOWED`).

UPAM formats

UPAM works block-oriented, the basic processing unit is the 2-Kbyte standard block for K-PAM files and the logical block for NK-PAM files, its size being determined by the BLKSIZE operand in the FCB or FILE macro. UPAM is able to read or output up to 16 2 KB standard blocks at the same time (LEN=(STD,n) or LEN=n*2048 (n <= 16)).

The following point applies to K-PAM files:

> If the value specified for the LEN operand in the PAM macro is not an integer multiple of 2048, it is rounded up to the next higher integer multiple of 2048. In the case of a write operation, the remainder of the last PAM page to be written in the file is undefined. If this write operation took place at the end of the file and this file is then closed, the position of the last valid byte on this PAM page is stored in the catalog entry's last-byte pointer (see "Prerequisite for using the last-byte pointer"). For a read operation, the remainder of the last PAM page to be read is not passed to the buffer.

The following point applies to NK-PAM files:

> If the value specified for the LEN operand in the PAM macro is not an integer multiple of the size of a logical block, it is rounded up to the next higher integer multiple of the logical block size. In the case of a write operation, the remainder of the logical block in the file is undefined. If this write operation took place at the end of the file and this file is then closed, the position of the last valid byte of the last logical block is stored in the catalog entry's last-byte pointer (see "Prerequisite for using the last-byte pointer"). For a read operation, the remainder of the last logical block to be read is not passed to the buffer and the remainder of the buffer contents is undefined.

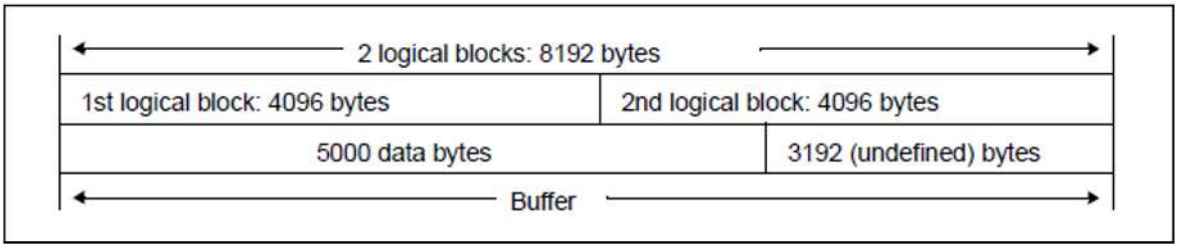
Prerequisite for using the last-byte pointer

Prerequisite for the usage of the last-byte pointers for PAM files on public space is the setting of the LBP_REQUIRED indicator in the file control block (FCB) for OPEN. LBP_REQUIRED makes the LBP in the FCP accessible during OPEN and saves the updated value in the catalog entry during CLOSE.

With PAM node files, a last-byte pointer is always supplied, irrespective of the LBP_REQUIRED flag. However, it is to be noted that the LBP in the CE may be outdated, as the file might have been changed by another system. Therefore, the current valid last-byte pointer is identified and made accessible to the application in the file control block (FCB) during OPEN.

Example

The operands WRT and LEN=5000 are specified in a PAM call for a file with BLKCTRL=NO and BLKSIZE=(STD,2). 5000 bytes are transferred from the buffer, and the remainder (up to the next higher integer multiple of the logical block size (8192 bytes)) is undefined.


2 logical blocks: 8192 bytes
1st logical block: 4096 bytes
2nd logical block: 4096 bytes
5000 data bytes
3192 (undefined) bytes
Buffer

UPAM for disk files

UPAM offers the following functions for disk files:

Creation of disk files; users themselves must program access to records (e.g. sequential access or associative access using hashing techniques).

Reading of SAM and ISAM files (OPEN=INPUT) and their transfer to other volumes (e.g. from disk to tape); the file attributes, e.g. BLKSIZE, RECSIZE, RECFORM, are stored in the associated FCB. This enables the user to program access to records.

UPAM cannot open SAM or ISAM files in UPDATE mode.

Because of the complex relationships between index blocks and data blocks, ISAM files cannot be effectively processed using UPAM. UPAM can, however, be used for the block-oriented transfer of ISAM files to tape.

Shared-update processing: a number of parallel jobs can process a PAM file concurrently. The OPEN modes permitted in the shared-update processing of a PAM FILE (mono system) are described in detail in the “UPAM” chapter of the “Introductory Guide to DMS” [1].

PAM macros in list form (up to 255, not necessarily all referring to the same file) can be handled with a single UPAM I/O request, i.e. only one SVC is required. The chaining of PAM macros (just like chained I/O) serves to optimize the runtime performance of user programs.

Eventing mechanism: the user job is notified when a UPAM I/O operation terminates and a contingency process starts.

For files with DRV (Dual Recording by Volume; see the “DRV” manual [15]): user information is output on the current status (e.g. loss of a copy of a disk). This information is requested by UPAM when an I/O operation is performed, and stored in the FCB (field ID1DRVST). However, this field is not updated unless the DRV status is modified.

When using UPAM for a disk file without a PAM key (BLKCTRL=DATA or BLKCTRL=NO), the following points should be borne in mind:

- the file must have standard blocks (BLKSIZE=(STD,n))
- if the file is not an ISAM file (FCBTYPE=SAM or FCBTYPE=PAM), the secondary allocation must be at least as large as the defined block size (BLKSIZE).

Chained I/O

Chained I/O permits the simultaneous input/output of up to 255 logically consecutive PAM pages in one file using a single PAM macro (not to be confused with the chaining of PAM macros in list form with the CHAIN= operand). This reduces the number of I/O operations (and interrupts) and thus reduces processing time. On the downside, however, it increases main memory requirements and the paging rate.

UPAM uses chained I/O if the LEN operand of the PAM macro contains a value greater than STD or greater than 2048.

End-of-file (EOF) processing

If the end-of-file condition is encountered during a write operation, a secondary allocation is performed and the specified number of PAM pages are appended to the file.

If the end-of-file condition is detected during a read operation, UPAM merely transfers the PAM pages belonging to the file into the buffer.

UPAM notifies the calling job of end-of-file processing as follows:

- *If eventing is not used:*
  the user job receives control at the EXLST exit USERERR with error code X'0922' in field ID1ECB of the FCB. The number of PAM pages transferred is contained in field ID1NBPP of the FCB. The value X'00' in this field means that all the PAM pages to be read are located outside the file. A value greater than X'00' means that the user job must perform a wait operation, unless this was implicitly included in the read operation (i.e. in a RDWT operation).
- *If eventing is used:*
  if all the PAM pages to be read are located outside the file, UPAM passes control to the user job at the EXLST exit USERERR with error code X'0922' in field ID1ECB of the FCB. If at least one of the PAM pages to be read belongs to the file, UPAM either resumes the basic task or initiates a contingency process (see "TU eventing: eventdriven processing"). Field IDECBNPA of the FECB (File Event Control Block, see "TU eventing: event-driven processing") now contains a value indicating the number of PAM pages transferred.

  X'00' All PAM pages to be read were transferred to the buffer

  X'0n' n = number of PAM pages belonging to the file that were transferred to the buffer.

Locking and unlocking of PAM pages

Only the first in a series of PAM pages to be locked/unlocked needs to be specified in a PAM macro; the number of pages to be locked/unlocked is derived from the value of the LEN operand. It should, however, be noted that after a lock or unlock operation, the file pointer points to the last PAM page that was locked/unlocked. This page may lie outside the file (see “End-of-file (EOF) processing” above).

A LOCK or UNLOCK operation applied to a file opened as SHARUPD=NO or SHARUPD= WEAK is treated as a no-op, i.e. the only action taken is to update the pointer to indicate the last page processed. The LEN operand is interpreted in this case.

Processing of PAM keys

There are two possible ways of processing PAM keys:

- The user reads/writes each individual key of a series of PAM pages: MKEY=YES operand in the PAM macro; the KEYFLD operand must contain the address of a sufficiently large area.
- The user reads/writes only the first key of a series of PAM pages. During writing, succeeding blocks are assigned the same key as the first, with just the logical block number being incremented by 1 each time.

Notes on the processing of disk files with UPAM

Blocks are not transferred to the user buffer until an explicit action macro is issued. As this causes a delay, asynchronous I/O operations have to be terminated by means of the WTaction macro. For TU eventing, the SOLSIG macro should be used (synchronously or asynchronously).

Any unsuccessful branch to the UPAM routines causes control to be passed to the routine specified in the EXIT operand of the FCB macro, or to the corresponding EXLST routine. An indicator is stored in the FCB.

Whenever UPAM induces a program termination, it supplies registers 0, 1 and 15 with the following contents, which can be easily identified and evaluated in memory dumps:

Register 0	Address at which the termination occurred.
Register 1	Address of the element in the UPAM operand list chain in which the error was detected.
Register 15	UPAM error code.

If register 1 contains an invalid address when the PAM macro is issued for the first time, this address will be contained in register 0 when the dump is taken, and register 1 will contain zeros (i.e. the error occurred before the first element in the operand list chain was located).

UPAM uses the following EXLST exits:

ERRADDR	Hardware fault or abnormal I/O termination.
USERERR	Invalid use of a macro in the program or attempt to read a PAM page not belonging to the file (EOF).
EOFADDR	Attempt to read a dummy file.
PGLOCK	Not all of the requested locks become available within the specified time, and the job currently has no blocks locked.
DLOCK	The request to set up a lock is rejected and the job already has locks set.

PAM pages which have been allocated to a file but have not yet been written by the file owner are identified by an internal file name code assigned by the system (CFID = Coded File ID, bytes 0-3 of the PAM key or block control field), which does not correspond to the current file name. The name comparison must be performed by the user, taking the following points into account (see also the KEYFLD operand of the PAM macro, "PAM - Perform UPAM actions"):

- At OPEN time, the current CFID is written to the first word of field ID1KEY1 in the FCB.

The following points apply solely to the processing of K-PAM files (BLKCTRL=PAMKEY):

- After execution of a RDWT, LRDWT or RDEQU operation, the CFID of the block just read is located in the first word of FCB field ID1KEY2.
- After execution of one of the operations WRT, WRTWT, WRTWU, or WT, the CFID of the PAM page concerned is located in the first word of FCB field ID1KEY1. As the entry generated by OPEN will be overwritten, it should be saved prior to processing so that subsequent comparisons can be made.
- After execution of an LRD or RD operation, the contents of fields ID1KEY1 and ID1KEY2 are not changed.
- With event-driven processing, the CFID of the relevant PAM page is in the first word of FCB field ID1KEY1 after completion of an I/O operation.

The fields ID1LWB (PARMOD=24) and ID1LWBPT (PARMOD=31) in the FCB are used to store the address of the last block on which UPAM successfully performed an I/O operation. The leftmost byte of field ID1LWB is used as an indicator for an outstanding WT operation.

If the last UPAM operation on the file was a successful WT, then the indicator byte in ID1LWB is set to X'00' and the three least significant bytes of ID1LWB or ID1WBPT contain the address of the block affected by the WT operation. This occurs regardless of whether or not the operation which initiated the WT was successfully completed.

If the last UPAM operation on that file did not initiate a WT operation, the indicator byte in ID1LWB is set to X'FF'. The contents of the remaining three bytes in ID1LWB or ID1LWBPT are of no significance.

UPAM processing of tape files

UPAM offers the following functions for tape files:

Creation of tape files not extending over more than one tape; the user is responsible for programming access to logical records in these files.

Reading of SAM files with standard blocks; the file attributes are stored in the FCB by OPEN processing (see the chapter “OPEN processing” in the “Introductory Guide to DMS” [1]), e.g. BLKSIZE, RECSIZE, RECFORM. This enables the user to program access to logical records.

Chained I/O is not possible for tape files.

Informing the user job upon termination of a UPAM I/O operation and start of a contingency process (eventing mechanism).

When using UPAM, the following basic points must be noted:

- UPAM is a block-oriented access method, i.e. the system is unaware of any logical structure within the blocks. The user must program his own record handling.

The following points apply to the use of UPAM with tape files:

- The file must reside on a single tape.
- Every read/write operation processes exactly one physical block.
- In the case of a file with the format BLKCTRL=PAMKEY (explicit or implicit), this must be a 2064-byte standard block. The first 16 bytes contain the PAM key and the remaining 2048 bytes contain the user data. Only STD or 2048 may be specified for the LEN operand of the PAM macro.
- In the case of a file with the format BLKCTRL=DATA or BLKCTRL=NO, BLKSIZE can be specified in bytes, in which case the value must be a multiple of 2048. The LEN value must not be greater than the BLKSIZE value. Blocks of the length specified by LEN and containing either only user data (with BLKCTRL=NO) or a 12-byte block control field plus the LEN value minus 12 bytes of user data (with BLKCTRL=DATA) are written to the tape. These blocks can be read by means of the UPAM function RD, in which case the value for LEN_{RD} in the read call must lie within the range:
  LEN_{WR} <= LEN_{RD} <= BLKSIZE. (LEN_{WR} = value for LEN when writing to the file.)

  In this context, the following special feature applies to files with FCBTYPE=ISAM: regardless of the value specified for BLKSIZE, UPAM always works with a block size of 2048 bytes. This is because, for ISAM, each 2K block has a block control field and thus constitutes a separate entity. In this case, the LEN value must not be more than 2048.
- The following errors may arise in the context of a UPAM access to a tape file with BLKCTRL=PAMKEY and nonstandard blocks (not equal 2064 bytes):
  - hardware error (error code = 927)
  - the data may be stored in the buffer incorrectly
  - the PAM key may be corrupted.

- Tape files must not be accessed by several jobs at once, or by one job several times (this is due directly to the properties of magnetic tape). This means that:
  - a tape file must not be opened with SHARUPD=YES/WEAK
  - a tape file which is already open cannot be opened again, even when both OPEN operations are in INPUT mode
  - a tape file must not be opened with an FCB operand whose PAMREQS value is greater than one.
- It is possible to read a tape file with UPAM using random access. However, the time outlay involved can be considerable.
- It is possible to write PAM blocks as of a specific point in an existing tape file. The last newly written block automatically becomes the last block in the file, even if the existing file contained more blocks. The file can then only be read up to this block. If a block nearer the start of the file is then read and the file is then closed, the most recently read block becomes the last block in the tape file.

Programming notes

Magnetic tape cartridges are treated just like normal magnetic tapes.

Whenever UPAM causes program termination, it places the address which was responsible for the abortion in register 0, while the address of the element in the UPAM operand list chain at which the error was detected is placed in register 1, and the UPAM error code is placed in register 15. This facilitates location of this information in memory dumps.

If register 1 contains an invalid address after the first PAM macro is issued, register 0 in the memory dump will also contain this address; register 1 will then contain the value 0, i.e. the error occurred before the first element of the operand list chain could be located.

UPAM uses the following EXLST exits:

ERRADDR	Hardware error or abnormal I/O termination.
USERERR	Invalid operation, such as an attempt to read a PAM page which does not belong to the file (end-of-file).

Fields ID1LWB (PARMOD=24) and ID1LWBPT (PARMOD=31) in the FCB are used to store the address of the last block on which a WT operation was carried out successfully by UPAM. The leftmost byte of field ID1LWB is used as an indicator byte.

If the last UPAM operation on the file was a successful WT, then the indicator byte in ID1LWB is set to X'00' and the three least significant bytes of ID1LWB/ID1LWBPT contain the address of the block affected by the WT operation. This occurs regardless of whether or not the operation which initiated the WT was successfully completed.

If the last UPAM operation on that file did not initiate a WT, the indicator byte in ID1LWB is set to X'FF'. The contents of the rightmost three bytes in ID1LWB/ID1LWBPT are of no significance.

Chaining PAM macros in list form

PAM macros which are to be chained, and which do not necessarily have to refer to the same file, must be generated in list form by means of the operand MF=L and then stored in a constant area; chaining is implemented by specifying the CHAIN= operand.

All the macros (except the last) have the following format:

	Operation	Operands
$element_n$	PAM	fcbaddr,operation,...,MF=L,CHAIN=$element_{n+1}$

In the last element of the chain, the CHAIN operand is omitted.

A chain of PAM macros in list form is called by means of a PAM macro with the following format:

	Operation	Operands
	PAM	MF=(E,$element_1$)

Only one SVC instruction is required for each chain, i.e. by chaining UPAM requests the user avoids the overhead of multiple SVC processing.

Example of coding

```
        START
        LDBASE 10
        USING *,10
        .
        .
        PAM MF=(E,ELEM1)
        .
        .
        .
        TERM
*CONSTANT-AREA
ELEM1   PAM     ......,MF=L,CHAIN=ELEM2
ELEM2   PAM     ......,MF=L,CHAIN=ELEM3
ELEM3   PAM     ......,MF=L
        .
        .
        .
        END
```

If execution is successful, the user regains control at the statement following the PAM macro which requested processing of an operand list chain.

All operations are carried out in exactly the same order in which the PAM macro lists appear within the chain – with one exception: when the first operation to request a lock is encountered, the rest of the chain is examined and all operations requesting locks are registered. If all the requested locks cannot be imposed within the specified time, the chain is terminated at the operation which requested the first lock.
If an action requested in the chain is not performed successfully, none of the following actions in the chain is executed (including locks).
Control is passed to the appropriate EXLST exit; register 1 points to the FCB of the file in which the error occurred. The error code (ID1ECB) and the sense byte (ID1XITB) are set in this FCB in the normal way, and field ID1CHERR in the FCB is set to the address of the element in the operand list chain that caused the error.
A check on an unfinished I/O operation also causes control to be transferred from the operand list chain to the user program at the specified address. The remaining requests in the chain (including locks) are not executed, and field ID1CHERR in the FCB is set to the address of the operand list element which contains the CHK operation. It is therefore not advisable to use check operations in operand list chains.
The user must ensure that the lock, read, write, wait, check and unlock operations are applied appropriately within a chain. Buffers and key fields are utilized by UPAM according to the request. A check is made to verify that a buffer exists and that access is authorized, but there is no guarantee that a buffer or a key field filled by an operation in a chain will not be overwritten by a later operation involving this chain.

The format of the PAM operand list can be described by means of a dummy program section (DSECT) generated with the IDPPL macro.
In all cases where a chain of UPAM operand lists cannot be processed in full (e.g. I/O operation failed, error detected, lock not accepted, EOF condition encountered, or CHK operation attempted on currently executing I/O request), the address of the first unexecuted chain entry is moved to field ID1CHERR of the FCB. The user can assume that all preceding entries were executed correctly.
UPAM does not report errors via FCB-EXIT and EXLST

- if the UPAM SVC is executed and neither register 1 nor any chain operand contains a valid address (e.g. the address is not aligned on a word boundary or refers to a field which is not entirely within the user's virtual address space and is not large enough to accommodate a UPAM macro operand list, etc.);
- if the FCB address in the UPAM macro operand list is either missing or invalid.

In both these cases, there is no FCB to which the error could be reported, and therefore UPAM aborts the program.
A UPAM operand list chain is validated in full before any function is initiated. If a CHAIN address or FCB address is found to be invalid, the job is aborted before any chain element is executed.

Where the eventing mechanism is used, the user program is likewise aborted if the I/O operation is completed and there is no event item available to which the event can be reported.

TU eventing: event-driven processing

This section describes event-driven processing using certain macros. For a more detailed explanation of “eventing”, see the “Executive Macros” manual [2].

Eventing is used by UPAM to report the completion of an I/O request to a job. The job can

- be continued in parallel with the UPAM I/O operation and, when the expected event occurs (in this case, termination of the requested I/O operation), proceed with a contingency process (asynchronous processing).
- wait for termination of the requested I/O operation and then proceed (synchronous processing; this is, of course, equally feasible without eventing).

Upon completion of an I/O operation, UPAM sends a message to the associated event item (using the POSSIG macro). Sooner or later this message encounters the request issued by the user (by means of the SOLSIG macro). When both request and message are present (for the same event item), a contingency process is started or the basic task resumed.

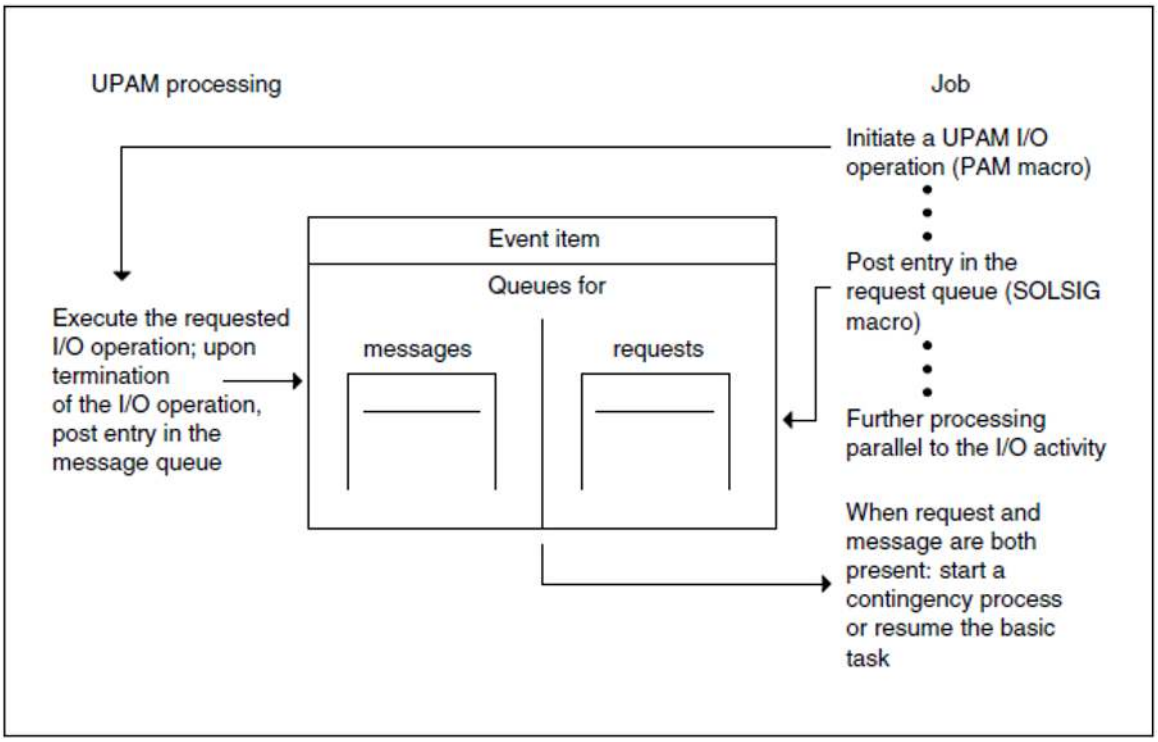


Figure 5: Coordination of user job and UPAM processing

Basic task

The system must be informed of the event items and contingency definitions that are to be used (ENAEI, ENACO macros).

For each I/O operation, the system must be supplied with the address of a file event control block (FECB). I/O operations running in parallel must refer to different FECBs. The maximum number of parallel I/O operations is defined via the FCB operand PAMREQS; for tape files, only PAMREQS=1 may be specified.

The number of contingency definitions depends on whether different procedures are required after execution of I/O operations. If, for instance, the same procedure is to be used in all cases, the contingency definition needs to be coded once only.

A 14-byte FECB (File Event Control Block) must be set up for each event item.

The FCB macro for each file must specify a valid PAMREQS operand; PAMREQS defines the maximum number of concurrent I/O operations which may be requested for that file.

Until the first I/O operation with an FECB is terminated, the FECB must not be used for other I/O operations.

The address of the associated FECB must be specified for each UPAM I/O request (FECB operand in the PAM macro). No wait operations may be requested either explicitly or implicitly by the PAM macro. The instruction sequence read (RD) -> write (WRT) -> wait (WT) for the same block would thus yield an undefined result. The user must wait for the event (I/O termination) before issuing the WRITE call.

After each UPAM I/O request, precisely one request must be issued to the associated event item (SOLSIG macro). The request may also be used to specify whether the basic task is to continue in parallel with the I/O operation or is to wait for it to terminate.

At the start of a contingency process, it is passed the following information via registers 2 and 3:

For PARMOD=31: the I/O operation is initiated using the 31-bit operand list; the required information is transferred in registers 2, 3 and 4:

Register	Information
2	contains the event information code.
3	the two rightmost bytes contain a post code supplied by the user at the start of the I/O operation, the leftmost byte contains an identifier indicating a UPAM event (X'10').
4	contains the address of the operand list for the operation which has just been completed.

For PARMOD=24: the I/O operation is initiated using the 24-bit operand list; the required information is held in registers 2 and 3 (as in earlier versions of BS2000):

Register	Information
2	contains the event information code.
3	the three rightmost bytes contain the address of the operand list for the operation just terminated, the leftmost byte contains the value X'10'.

If a SOLSIG macro generated with PARMOD=24 or a 24-bit contingency definition is addressed via an operand list created with PARMOD=31, the secondary return code (leftmost byte in register 15) indicates that the sending and receiving lengths are not consistent.

Format of the file event control block (FECB)

The FECB must be aligned on a word boundary. It can be given a symbolic name by means of the IDECB macro.

Executive flag byte

Meaning of field	**Field length (in bytes)**	**Field name**
Internal ID of event item	4	CBEVID
Address of FCB	4	CBP1LNK
Standard device byte	1	CBSDB
Sense bytes	3x1	CBSB1, CBSB2, CBSB3
Executive flag byte	1	CBEFB
Number of PAM pages transferred	1	CBNPA

A UPAM I/O operation can be terminated in a number of different ways (EFB = Executive flag byte; see FECB):

Normal I/O termination	EFB=X'80'
I/O operation led to exception condition	EFB=X'C0'
Unrecoverable error (e.g. hardware fault)	EFB=X'A0'

In a contingency process, the user can program appropriate responses to the various ways in which an I/O operation can terminate.

Files larger than 32 GB

BS2000 supports files and pubsets with a capacity of up to 4 TB and volumes of up to 2 TB. These are called “large files”, “large pubsets” and “large volumes”.

The thresholds supported by BS2000 are as follows:

- The maximum capacity of a pubset or volume set is about 4 TB.
- The maximum capacity of a single disk is about 2 TB.
- The maximum file size is about 4 TB (which is the maximum size of a pubset or volume set minus SVL, F5 label and system files).

Large files and large volumes are only supported in special pubsets whose attributes must have been set up for the use of these large objects by systems support.

Expanding catalog entries

The introduction of 4-byte fields for the following data stored in the catalog entry is a key aspect in the lifting of the 32-GB limit for volume and data sizes:

- FILE-SIZE – the storage space allocated for the file
- HIGHEST-USED-PAGE – the amount of storage that currently contains data
- LHP (Logical halfpage number) and PHP (physical halfpage number) of the individual extents in the extent list, that allocate “physical” halfpages of volumes to the logical halfpages

3-byte and 4-byte fields

Block numbers and block counters are visible at various BS2000 user interfaces. Although 4-byte fields have been used exclusively in all new versions of these interfaces, some old interface versions may still use 3-byte fields. If files >= 32 GB exist, you may experience compatibility problems. This may also occur in rare cases when only volumes >= 32 GB exist.

New format for the extent list

With the introduction of 4-byte fields for LHP and PHP, a new (additional) format has been introduced for the extent list.

The correlation between the maximum size of volumes and files and the field width of LHP and PHP is depicted in figure 6:

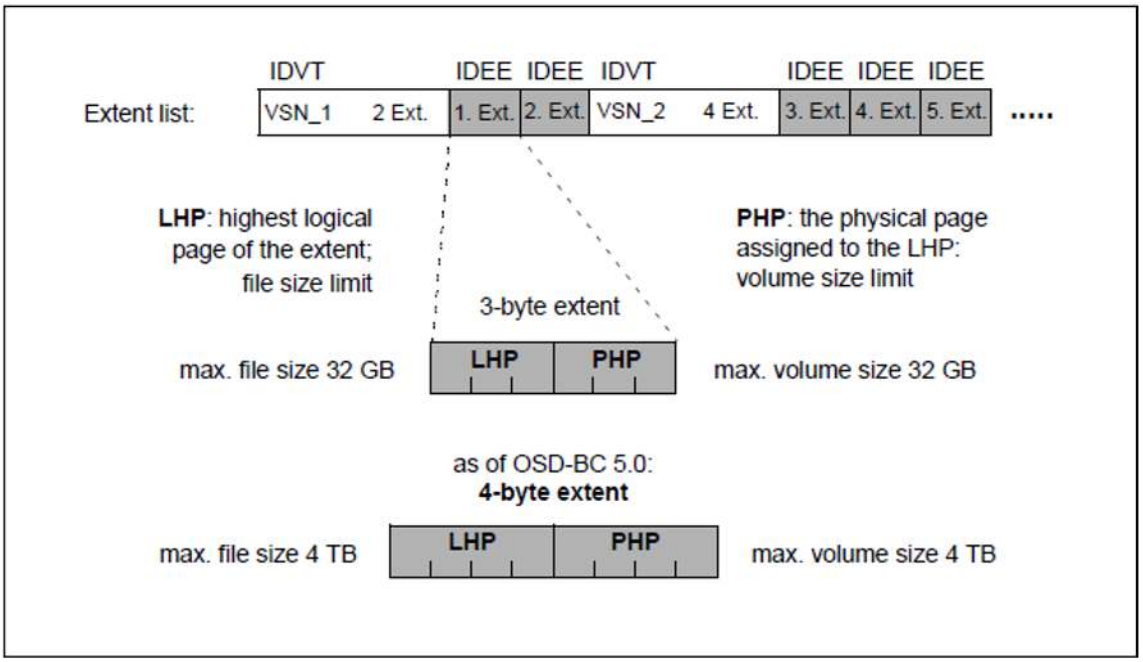


Figure 6: Correlation between file and volume sizes and the field width of LHP and PHP

For compatibility reasons BS2000 supports both formats of the extent list as follows:

- In general, the “old” format with 3-byte block numbers will be used.
- Only in the case of large files or of files that can only be addressed using PHPs > X'FFFFFF' will the new format with 4-byte block numbers be used.

Extent lists therefore contain either extents with 3-byte block numbers or extents with 4-byte block numbers.

Restrictions for large files

- A SYSEAM file cannot be >= 32 GB.
- Files with BLKCTRL=PAMKEY:
  The logical page number is stored as a 3-byte field in the system section of the PAMKEY.

Summary of the DMS macro interfaces affected by 32-byte objects

Interface	Change
FCB	New operand for large files
FILE	New operand for large files
FSTAT	Effort involved in check and conversion if VERSION=0/1
OPEN	Takes account of semantic problems
RDTFT	Attribute of the “large file” file attribute
DIV	New operand for large files; extended range of values for BLOCK and SPAN

FPAMACC	Extended range of values for BLOCK
FPAMSRV	New operand for large files

User programs

As stated above, it cannot be assumed that all programs have been prepared for accessing large objects, i.e. that they can address 4-byte block numbers and block counters and it must be borne in mind that the interfaces available to user applications only relate to accessing and processing files and their metadata.

Program behavior can thus be classified as follows:

Class A: A program is able to process large files without restrictions. This behavior is defined as LARGE_FILES-capable.

Class B: A program has not been prepared for processing large files and/or their metadata. It is, however, able to perform defined rejections of corresponding access attempts that it regards as illegal. Alternatively, there are no access files or their metadata within the program. This behavior is defined as LARGE_FILES-capable.

Class C: A program has not been prepared for processing large files and cannot perform a defined rejection of corresponding access attempts. This behavior is defined as LARGE_FILES-incompatible.

In configurations that contain large files, programs that are compatible with or capable of LARGE_FILES are required. LARGE_FILES compatibility is to be regarded as the norm. Growth over 32 GB will initially be limited to a relatively small number of files. Only the programs that access these require LARGE_FILES capability.

For concrete examples of this classification for the relevant DMS interfaces in BS2000 and further, detailed information on this subject, refer to the manual “Files and Volumes larger than 32 GB” [19].

Macros

- ADDPLNK - Define pool link name
- BTAM - Process tape files (type S)
- CATAL - Process catalog entry
  - Version differences - VERSION=0/1/2/3/4
- CHKFAR - Check file access rights
- CHNGE - Change TFT entry
- CLOSE - Close file
- COMPFIL - Compare disk files
- COPFILE - Copy file
- CREAIX - Create secondary keys for ISAM file
- CREPOOL - Create ISAM pool
- DECFILE - Convert encrypted file into unencrypted file
- DELAIX - Delete secondary key of ISAM file
- DELPOOL - Delete/release ISAM pool
- DIV - Access files via virtual address space
  - DIV function: OPEN
  - DIV function: MAP
  - DIV function: SAVE
  - DIV function: RESET
  - DIV function: UNMAP
  - DIV function: CLOSE
- DROPTFT - Release TFT entry
- EAM - Process EAM files
- ELIM - Eliminate record
- ENCFILE - Convert unencrypted file into encrypted filen
- ERASE - Erase files
  - Variations in versions - VERSION=0/1/2
- EXLST - Define exit address list
- EXRTN - Return from error routine
- FCB - Define file control block
- FCBAD - Create FCB addresses
- FEOV - Close tape
- FILE - Define file attributes / control file processing
  - Variations in VERSION=0/1/2/3
- FILELST - Create variable operand areas for FILE macros
- FPAMACC - Access FASTPAM files

- FPAMSRV - FASTPAM management function
  - FASTPAM function: ENABLE ENVIRONMENT
  - FASTPAM function: ENABLE IOAREA POOL
  - FASTPAM function: OPEN
  - FASTPAM function: CLOSE
  - FASTPAM function: DISABLE IOAREA POOL
  - FASTPAM function: DISABLE ENVIRONMENT
- FSTAT - Request catalog information
  - Programming notes for VERSION=4
  - Programming notes (VERSION=2, 3 and 4)
  - Programming notes for VERSION=0 and VERSION=1
  - Version differences - VERSION=0/1/2/3/4/5
  - Version variations in the representation of the output area
- GET - Read next record
- GETFL - Read record by flag
- GETKY - Get record with specified key
- GETR - Get record "reverse"
- IDBPL - Provide BTAM operand list with symbolic names
- IDFCB - Provide FCB with symbolic names
- IDFCBE - Provide FCBE with symbolic names
- IDPPL - Provide PAM operand list with symbolic names
- IMPNFIL - Create (import) catalog entries for node files
- IMPORT - Create catalog entry for files
- INSRT - Insert record
- ISREQ - Unlock data block
- LBRET - Return from user label routine
- LFFSNAP - List files from a Snapset
- LJFSNAP - List job variables from a Snapset
- MAILFIL - Send file by email
- NDWERINF - Evaluate status bytes
- OPEN - Open file
- OSTAT - Request information on open files
- PAM - Perform UPAM actions
- PUT - Write record
- PUTX - Replace record
- RDTFT - Read TFT and TST information
- RELSE - Close block
- RELTFT - Delete TFT entry
- REMPLNK - Delete pool link name

- RETRY - Repeat macro
- RFFSNAP- Restore files from Snapset
- RJFSNAP- Restore job variables from a Snapset
- SETL - Position file pointer
- SHOPLNK - Return information on ISAM pool link names
- SHOPOOL - Return information on ISAM pools
- SHOWAIX - Request information on secondary keys
- STORE - Store record
- VERIF - Recover file

ADDPLNK - Define pool link name

Macro type: type S (E form/L form/D form/C form); see "Macro types"

The ADDPLNK macro is used to assign a pool link name to an ISAM pool for a user job and to enter this name in a pool table. This pool

link name must have been entered in the task file table by means of the FILE command (operands LINK and POOLLNK) or specified in the FCB (field POOLLNK). During OPEN processing, the system checks whether a pool link name exists for the file and whether an ISAM pool exists for the name.

Format

Operation	Operands
ADDPLNK	**POOLNME = poolname** **,LINKNME = name** **[,CATID = catid]** **[,SCOPE = TASK / USERID / USERGROUP / HOST]** **MF = L,POOLNME = poolname**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX=pre]
	MF = C[,PREFIX = pre][,MACID = macid]

Operand descriptions

CATID = catid

Specifies the pubset to which the ISAM pool belongs. This must match the “catid” specification in the CREPOOL macro.

Default value: the default catalog ID of the task.

LINKNME = name

allocates the pool link name “name” to the ISAM pool “poolname”. “name” can consist of 1-8 characters; character set allowed: all letters and digits as well as the special characters $, # and @, with the first character not being a digit or $.

MACID

Evaluated only in conjunction with MF=C; defines the second through fourth characters of the field names and equates generated in the data area when the macro is expanded.

Default value: MACID = ISA

= macid
Three-character string defining the second through fourth characters of each field name and equate generated.

MF

The forms of the MF operand are described in detail in the appendix ("Macro types").

PARAM

Specifies the address of the operand list; evaluated only in conjunction with MF=E (see "Macro types").

= addr
Symbolic address (name) of the operand list.

= (r)
Number of the register containing the address of the operand list. The register must be loaded with this address value before the macro is called.

POOLNME = poolname

Specifies the ISAM pool to be used for file processing. “poolname” is the name with which the ISAM pool was created (see the CREPOOL macro, "CREPOOL - Create ISAM pool").

PREFIX

evaluated only in conjunction with MF=C or MF=D; defines the first character of each field name and equate generated in the data area when the macro is expanded.

Default value: PREFIX = D

= pre
One-character prefix with which the field names and equates generated by the assembler are to begin.

SCOPE

Specifies the scope of the ISAM pool. This must have the same value as in the CREPOOL macro.

= TASK
The pool link name is assigned to the task-local ISAM pool “poolname”.

= USERID
This scope is still supported only for reasons of compatibility (see the CREPOOL macro, "CREPOOL - Create ISAM pool").

= USERGROUP
This scope is still supported only for reasons of compatibility (see the CREPOOL macro, "CREPOOL - Create ISAM pool").

= HOST
The pool link name is assigned to the cross-task ISAM pool “poolname”.

Return codes

Unless otherwise specified, the field names and EQU statements for return codes generated by the C and D forms of the macro begin with the string DISA; this string can be modified by means of PREFIX and MACID.

The return codes are kept in the default header of the operand list.

Main return code	Meaning
DISAOK X'0000'	The macro call was successful
DISANPAR X'0001'	Access to the operand list was not possible
DISANCAT X'0003'	The catalog ID “catid” is unknown in the system
DISANACC X'0004'	There is no connection to pubset “catid”
DISAINVN X'0005'	The pool name or pool link name is invalid
DISANCLA X'000A'	The pool name already exists and the existing assignment cannot be cleared
DISASYSE X'000B'	An internal error occurred during macro processing
DISANOPL X'000D'	There is no pool with the specified name
DISARLNK X'FFFF'	The macro call could not be executed: evaluate subsidiary return code 1 (linkage error)

BTAM - Process tape files (type S)

Macro type: type S (E form/L form); see "Macro types"

All user requests for BTAM are handled using this macro. In the operand descriptions, the abbreviation “MTC” is used for “magnetic tape cartridge”.

Format

Operation	Operands
BTAM	**fcbadr** **[,RDWT / RBID / RD / RDBF / CHK / MINF / POS / REV / REVWT / RNT / RNTL / RT / RTL / SYNC / WRT / WRTWT / WT / ERG / BSF / BSR / FSF / FSR / REW / RUN / WTM]** **[,LEN = length]** **[,LOC = 1 / 2 / relexp]** **[,PARMOD = 24 / 31]** **[,REQNO = number]** **MF = L**
	MF = (E,adr / E,(r))

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed.

RDWT

Reads the tape forwards and waits until the operation is completed before returning control to the user program (default function).

CHK

Checks whether the previous I/O operation has been completed. If not, control is transferred to the address specified in LOC. Otherwise, the operation is equivalent to WT.

MINF

Fetches information about the medium during the processing of optical disks. In BS2000, optical disks are operated via an MTC emulation. The area to which the information is to be written and its length (currently 128 bytes) must be specified via the LOC and LEN operands. The layout of the output information is described with the NDWMINF macro.

POS

Permissible only with PARMOD=31; positions the tape (see "Operation codes").

RBID

Permissible only with PARMOD=31; determines the tape position (see "Operation codes").

RD

Reads the tape in a forward direction.

RDBF

For tape cartridges, permissible only with PARMOD=31; transfers data block-by-block from the save area of the MTC buffer to the user area (see "Operation codes").

REV

Reads the tape backwards (towards BOT).

REVWT

Reads the tape backwards and waits until the operation has been completed before returning control to the user program.

RNT

Reads without data transfer; a message is issued if the length is shorter than expected.

RNTL

Reads without data transfer; no message is issued if the length is shorter than expected.

RT

Reads with data transfer; a message is issued if the length is shorter than expected.

RTL

Reads with data transfer; no message is issued if the length is shorter than expected.

SYNC

Permissible only with PARMOD=31; determines the tape position and synchronizes (see "Operation codes").

WRT

Writes to a magnetic tape.

WRTWT

Writes to a magnetic tape, waiting until the operation has been completed before returning control to the user program.

WT

Waits until the previous I/O operation has been completed. Control is not returned to the user program until the operation has been completed or the necessary error recovery functions have been performed.

ERG

Generates an interblock gap; if repeated, the tape is erased.
The operation triggered by ERG is physically a write operation. However, instead of a bit pattern, an “interblock gap” pattern is generated. The length of this pattern is defined by the user (although for some magnetic tape types, the length is preset).

BSF

Rewinds (backspaces) the tape by one tape mark.

BSR

Rewinds (backspaces) the tape by one block.

FSF

Forward-spaces the tape by one tape mark.

FSR

Forward-spaces the tape by one block.

REW

Rewinds the tape to the BOT marker.

RUN

Rewinds and unloads the tape; subsequently, only CLOSE is possible.

WTM

Writes a tape mark.

LEN = length

Specifies the length of the individual blocks or, for chained I/O (CHAINIO operand in FILE/FCB), the length of the transport unit.
If LEN is not specified, the length of a transport unit is the product of the block size and the chaining factor, where the block size is defined by BLKSIZE if RECFORM=F is specified, and by the contents of the register specified for RECSIZE if RECFORM=U applies.

Without chaining, the system takes the specifications only from the RECSIZE operand when writing with RECFORM=U; otherwise, they are taken from the BLKSIZE operand. If RECFORM=U is specified, a length specification entered via the LEN operand may be such that the last block within a transport unit is shorter than the previous blocks. If RECFORM=F is specified, the given length should be a multiple of the BLKSIZE specified (but the job will not be rejected if this is not the case).
The length to be read is always taken from the current length specification and only from this. After a successful READ, the actual block size is returned in the RECSIZE register.If specified in conjunction with operation code RDBF, LEN specifies the length of the blocks to be saved from the buffer. If RECFORM=U/V applies, the current block length is stored in the RECSIZE register.

LOC

Specifies the area from which data is to be read or to which data is to be written.

Default value:

- IOAREA1
  - for the first input or output
  - when switching from LOC=relexp to an IOAREA
  - if IOAREA2 was used last
  - if IOAREA2 is not defined
- IOAREA2, if IOAREA1 was used last

= relexp
Address of an area in the macro.The LOC operand must be specified in the form LOC=relexp if a CHK operation is required. It specifies the address to which control is to be passed if the operation being checked has not yet been completed. The addressed area must not coincide with IOAREA1/2.

= 1
Points to the address IOAREA1 in the FCB.

= 2
Points to the address IOAREA2 in the FCB. The used I/O area is indicated in the TU FCB:

- in field ID1BLWB of the 31-bit TU FCB, or
- in field ID1LWB of the 24-bit TU FCB.

When used with the operation codes POS, RBID, RDBF, and SYNC, LOC has the following functions:

- POS:address of the specified tape position (9 bytes)
- RBID:address at which the specified tape position (9 bytes) is output
- RDBF:address of the area into which the saved block is to be placed
- SYNC:address at which the specified tape position (9 byes) is output

The pointer in the TU FCB (ID1BLWB or ID1LWB) is set to the last used area for these operation codes as well.

MF

The formats of the MF operand are described in detail in the appendix, "Macro types".

PARMOD

Specifies the generation mode for the macro.

Default value: the value preset for the generation mode by means of the GPARMOD macro or by the assembler.

= 24
The macro is expanded in accordance with the format for the 24-bit interface. The object code is thus executable only in 24-bit addressing mode.

= 31
The macro is generated as addressing mode-independent.

REQNO = number

Number <= 8; the number of the input/output request or of the associated macro. Several asynchronous read or write operations, identified by different numbers, can be started. Each of these read/write operations must be terminated by the WT operand (together with the relevant number). The maximum number of concurrent I/O operations is specified in the BTAMRQS operand of the FCB.

Operation codes

POS – Position the tape

This is useful for restarting, e.g. after a write error with loss of data: the user can specify the block number (2 bytes) which he/she obtained from an earlier RBID operation at the address defined by LOC and the tape is then positioned to the corresponding block.

RBID – Determine current tape position (block number)

Each block on a tape, including the tape marks, can be identified by means of its position on the tape.
In the case of conventional magnetic tapes, the tape position is returned to the user as a pair of values (TM and record counters). This information is returned to the user after an RBID instruction (or SYNC instruction, see "BTAM - Process tape files (type S)") at the address defined by means of the LOC operand (see the operand description) or, if LOC is not specified, in one of the I/O areas defined in the FCB. The position information (first 8 bytes) depends on the processing status and returns the information listed in the table on "BTAM - Process tape files (type S)" in field ID1ECB of the FCB in the case of an error.

The 9th byte specifies how the position information is to be interpreted.

2**7 = 1:	TM and record counters
2**6 = 1:	block ID
2**0 = 1:	no valid position specified

Before an RBID instruction is issued, all outstanding asynchronous read/write operations (in MAV mode) must be terminated with a WAIT. In the case of synchronous file processing, BTAM executes the WAIT automatically.

For magnetic tape cartridges, the tape position is defined as a “block number” (or block ID).

Information returned

Event	Information	Meaning	Action
Successful execution	Block number	Indicates the position of the next block to be written to or read out of the MTC buffer	----
Unsuccessful execution	Block number + error DC7C	The tape position has been saved; the block number shows which block was written last without error	Reposition using POS
	Block number undefined + error DC7B	The tape position could not be saved	Reposition to checkpoint or program is aborted
	Block number undefined + error DC79	The tape position could not be saved	Program is aborted
	Block number + error DC77	I/O error; the block number is the one that existed after error handling	----
	Block number undefined + error DC77	I/O error with “loss of position”	Reposition to checkpoint or program is aborted

RDBF – for MTC only – after an unrecoverable write error, read data from the save area of the MTC buffer into a user area.

When writing is buffered, it is possible for errors to occur for data blocks for which the user has already received a positive acknowledgment. DMS attempts to save the data already passed to the MTC buffer and the tape position so that the user can carry out normal error handling. (S)he can retrieve the block causing the error and subsequent blocks (which have been acknowledged but not yet written to tape) from the save area of the buffer and process them again, for example by sending them to a different volume.

Before the file is closed (CLOSE) or a tape swap is initiated by FEOV, the user must use the POS function to position the tape after the last block which was written successfully. The blocks are read on the principle “last in, first out” and the number of blocks stored in the buffer is kept in field ID1BLANZ in the TU FCB. A separate RDBF instruction must be issued for each block, in which case the input area is specified as for normal read operations: the address is given by the LOC operand in the BTAM macro or by IOAREA in the FCB and the length in the LEN operand or in the FCB, in the same way as for reading from tape. The blocks are transferred with the specified length. If RECFORM=V/U is specified for the file, the actual length of the saved block is indicated in the register specified by means of RECSIZE in the FILE or FCB macro.

SYNC – Synchronize and set marker points

The data contained in the MTC buffer is written to tape. For synchronous processing (not MAV mode), BTAM initiates any outstanding WAIT. For asynchronous processing (MAV mode), the user must ensure that any outstanding WAITs are executed before the SYNC instruction is issued. The SYNC call implicitly includes an RBID call, i.e. the current tape position is returned to the user in the I/O area. The user can utilize this fact to set checkpoints, e.g. for handling subsequent errors: he/she can then restart at one of these points and continue processing.

Programming notes

1. The BTAM macro destroys the contents of registers 0, 1, 14 and 15.
2. Misuse of FSF or FSR can cause the tape to run to end-of-volume, in which case it must be rewound by the operator (offline).
3. Whenever a branch to BTAM is unsuccessful, control is passed to the address specified in the EXIT operand of the FCB. An appropriate hexadecimal error code is stored in the FCB.
4. If the user specifies fixed-length blocks (format F) but reads blocks of a different length, control is passed to the ERRADDR exit in the EXLST macro.
   If the record length exceeds the specified length, an “abnormal termination” bit is set in the Executive flag byte, and the “record-length error” bit is set in the sense byte, provided no RTL or RNTL operation with the simultaneous use of chaining and/or MAV mode is involved (in which case the user is not informed). If the block is shorter than the specified length, however, the “abnormal termination” bit is set, but no error byte is returned (the residual count is stored in sense bytes 2 and 3; see the NDWERINF macro, "NDWERINF - Evaluate status bytes"). No notification is provided to the user if an RTL or RNTL operation is involved.
5. If the user specifies variable-length blocks (format U or V) or uses the operation codes RTL or RNTL, but reads blocks of a greater length than specified, control is passed to the ERRADDR exit in the EXLST macro, provided chaining and MAV mode are not used. The ERRADR exit is always activated for RT and RNT operations. In all other cases, no information is returned to the user for the U/V format.

   The various cases and situations in which the user is notified are summarized in the table below:

Operation	RECFORM=F Block < spec. length	RECFORM=F Block > spec. length	RECFORM=U/V Block < spec. length	RECFORM=U/V Block > spec. length
RD(WT)	yes	yes	no	conditional *)
R(N)T	yes	yes	yes	yes
R(W)TL	no	conditional *)	no	conditional *)

   *) yes if chaining and MAV mode are not used
6. The user need not specify wait operations explicitly, provided that he/she is not working in MAV mode. BTAM automatically outputs a WAIT before each new operation. However, if an error occurs during this WAIT operation, error code X'0C77' is output. The new operation is not executed.
7. Note that this implicit WAIT is only meaningful when working with multiple I/O areas. Since the data in the output area must essentially be retained when writing until the completion of a WAIT, and since the input area is correctly filled when reading only after completion of a WAIT, users who work with only one I/O area must ensure that a WAIT is issued as soon as they wish to reuse this single area.

8. BTAM does not use the EOFADDR exit of the EXLST macro (end of file). If a tape mark is read by a RD, RDWT, RNT, RNTL, RT, RTL, REV or REVWT, control is transferred to the ERRADDR address.The program can evaluate the 5 status bytes in the FCB (SDB, FB1, FB2, FB3, AMB) after the WAIT. Control is also transferred to the ERRADDR address if an attempt is made to reverse-read a tape that is positioned at the beginning of the tape.In the case of the REV and REVWT operations, the address for the first byte into which data is read is defined by:

   LOC + LEN - 1

9. The SAM macro FEOV can be used for a file opened by BTAM. If end-of-file (on the current tape) is encountered, BTAM branches to the otherwise unused exit EOFADDR.
10. CHAINIO with tape cartridges: the transport unit should not be larger than the buffer size, and the buffer size should be an integral multiple of BLKSIZE. In the case of an error, the indicated number of blocks in the buffer (TU FCB: ID1BLANZ) always refers to individual blocks, not to transport units. The RDBF operations also process only individual blocks.
11. MAV mode with tape cartridges: the value specified for BTAMREQS in the FCB can be smaller than for tapes, since the user I/O areas become free again more rapidly in the case of tape cartridges.
12. In the event of an error, it is possible that the entire contents of the MTC buffer cannot be written to the cartridge. The user can access the blocks transferred by the operating system from the MTC buffer to a save area, which is equivalent to reading a tape in the reverse direction. After this, the user should attempt to position the MTC after the last block which was written correctly.

    The user must ensure that the tape is positioned correctly (by means of POS) before further write operations are started (particularly before writing the EOV/EOF labels). Otherwise, data on the tape may be overwritten. An RBID operation after error DC7C returns the tape position after the last block which was written correctly.

    The tape can be closed correctly by writing the end labels if it is first read backwards for a few blocks or positioned backwards (if the user does not need the data already written).
13. If, due to a write error, some blocks were not written from the MTC buffer to tape, but are still saved in the save area of the buffer, the user must first execute an RDBF or RBID operation in order to tell the system that he/she does not want to discard this data; otherwise, it will not be available later. RDBF and RBID may be issued in any sequence; the contents of the buffer are retained as long as the user starts no other operations, with the exception of any outstanding WAITs. The user may issue these before issuing the RBID and/or RDBF instructions (and **must** issue them before the RBID); naturally, all of these WAITs will result in error code DC7C.

    If the user does not position the tape after the DC7C error message, the tape position becomes “UNDEFINED” and this state can be cleared only by positioning the tape (or with REW and UNL).
14. Before checkpoints are written by means of the SYNC instruction, all previously initiated I/O operations must be terminated by means of WAIT.
15. If a calling sequence in which a “write” operation follows a “reverse read” is used when in MAV mode, all outstanding WAITS must be issued before the “write” operation.
16. A characteristic feature of tape cartridge devices is that errors which occur during output operations are reported only when the buffer contents are transferred to tape. The user may thus be informed that an error has occurred only when (s)he starts another request. Consequently, the user may receive the message “end of tape” before the message indicating that an error has occurred. The “end-of-tape” message means that the user should not start any more write operations, in order to ensure that all data still in the buffer and the end labels will fit on the current tape. In the case of tape cartridges, this “end-of-tape” message does not result from detection of an end-of-volume label on the tape; rather it is generated during the transfer of data into the buffer, contingent on the amount of data already in the buffer and other criteria.

17. When the FEOV or CLOSE macro is issued for a BTAM file which was opened in INOUT mode, the file is treated as an output file if a WRITE-and-WAIT operation was requested for it during processing (WRT, WRTWT, WTM). If no WRITE-and-WAIT operation was requested, the file is treated as an input file.
18. BTAM issues the error code X'0C95' if:
    - an invalid device type is specified
    - the operation code is invalid
    - one of the registers 0, 1, 13, 14 or 15 is specified in the FCB RECSIZE operand
    - the value of the LEN operand in the BTAM macro is less than or equal to 0 or, if LOC=relexp is not specified, greater than BLKSIZE (in which case IOAREA1/2 will be used, which is the same size as BLKSIZE)
    - a read operation is requested for a file opened in OUTPUT mode
    - the REQNO specification in MAV mode exceeds the maximum permissible value
    - a WAIT macro has not been issued before a read/write call for the same REQNO, when in MAV mode
    - a BTAM call is specified after RUN (only CLOSE is possible) or after an unsuccessful FEOV operation
    - BTAM calls are issued for a file which has not been opened for BTAM (except in EXLST routines for label handling)
19. When extending a tape file, a direct switch from “write” to “read” should be avoided. Discrepancies in deletion overlaps can lead to problems on certain magnetic tape devices.
    The following sequence of operations always guarantees error-free execution:

BTAM macro	Meaning
BSR, FSR, ...	Position tape before block n
RD	Read (save) block n
BSR	Reposition tape before block n
WRT	Write block n back to the tape
WRT/WRTM	Write block n+1 (or tape mark)

20. An incorrect block length is not indicated in some cases, e.g. when reading data with chaining specified (see "BTAM - Process tape files (type S)").
    The chain for the input of CCWs continues to run. However, the areas to which the individual blocks are transferred are incremented at the customary rate for blocks of normal size; in other words there will be gaps in the input/output areas in those cases where blocks of shorter length occur.
21. Information relating to the buffers used is provided in the TUFCB field ID1LWB when PARMOD=24, and in the field ID1BLWB when PARMOD=31, when a WAIT is executed in the following manner (asynchronous processing): for an input operation, a note is made of the buffer affected by the last WAIT operation without an error message. For output operations, the buffer indicated is the one which was used for the last output (which may not necessarily have terminated).
    For synchronous processing, the following applies: for PARMOD=24, the value specified in the FCB field ID1LWB refers to the buffer which was last used, and for which a successful WAIT was issued; if PARMOD=31 applies, the FCB field ID1BLWB is used.
22. When the block and transport unit sizes are defined by the user, the following point should be borne in mind: Each I/O operation initiated by the device driver involves defining block and transport unit sizes for the pages affected. In the case of chained input/output, these definitions are effected for every CCW of the chain. For

memory management reasons, however, this is only possible for up to 63 definitions per page; that is, if a job is issued in such a way that more than 63 CCWs referring to the same page may be created (e.g. when using small block sizes and large transport unit lengths), a CSTAT error may be the result.

23. In the case of tapes with standard labels, BTAM checks whether a planned write operation is actually allowed by the current tape position. If not, the request is rejected with USERERR and error code 0C9D.

Return codes

Five FCB sense bytes (SDB, FB1, FB2, FB3 and AMB) are loaded in the TU FCB. In addition, the following information is issued in field ID1LRCRB:

Byte 1:	X'04' End-of-tape/beginning-of-tape encountered X'02' Block shorter than value from BLKSIZE X'05' Block longer than value from BLKSIZE X'01' Tape mark detected X'08' Undefined error (unrecoverable) X'09' Parity error (inoperable) X'0C' Device defective X'0D' Device in operation X'0E' Continuation error on tape cartridge; position to defective block X'0F' Tape format not compatible with device type
Byte 2:	Request number (required, since EXIT ERRADR can be issued only once per FCB).
Bytes 3/4:	Block number within a chained input/output job during which an error occurred, or number of the block at which the end of the tape was reached (this block will still be written).

A DSECT (DLRC macro) is supplied to interrogate the values set in the first byte of the ID1LRCLB field.

With operations in asynchronous mode (REQNO operand), any further input/output jobs which have already been accepted are no longer started if an error occurs during input/output. Instead, these jobs are terminated logically after a WAIT command has been issued by the user. When this is the case, the block number in bytes 3 and 4 of field ID1LRCLB and the error information for the errored I/O operation both have a value of 1, and the value in the RECSIZE register (for RECFORM=U) is 0.

The user can issue a CLOSE macro at any time, even before all outstanding WAITS have been issued. All outstanding input/output requests which may not yet have been completed at this point in time are then terminated logically. However, this does not necessarily mean that all these requests have been honored.

CATAL - Process catalog entry

Macro type: type S (E form/L form/D form/C form/M form); see "Macro types"

The CATAL macro creates or modifies catalog entries. It can be used to define attributes for file and data protection, to specify the coded character set and performance attributes, and to convert temporary files to permanent files and vice versa.

If attributes in existing catalog entries are to be modified, the operand STATE=*UPDATE must be specified. Only those file attributes, i.e. fields in the catalog entry, whose associated operands are specified with valid operand values are updated.

A catalog entry can be updated only if write access is not prevented by means of a password. Otherwise, the password must be entered in the password table of the job by means of the PASSWORD command (see the “Introductory Guide to DMS” [1]).

CATAL can be used to catalog files, file generations and file generation groups. The protection attributes for files and file generation groups can be modified; the protection attributes for file generations are defined by the related group entry.

The CATAL macro supports the “Default-Protection” function.

The encryption attributes of a file cannot be modified using the CATAL macro.

Temporary files

Since temporary files are job-specific, it is not possible to define file protection for them, i.e. the applicable default attributes cannot be modified. The following points must be noted when setting up a temporary file or if a permanent file is recataloged as a temporary file (or vice versa).

Nonprivileged users can only create temporary files on the default pubsets relating to their user IDs.

- Setting up a temporary file:

  The temporary file is assigned the following values (explicit specification of other values is generally not permitted):

EXPIRATION-DATE	= <date>	BACKUP-CLASS	= *E
USER-ACCESS	= *OWNER-ONLY	READ-PASSWORD	= *NONE
WRITE-PASSWORD	= *NONE	EXEC-PASSWORD	= *NONE
BASIC-ACL	= *NONE	GUARDS	= *NONE
FREE-FOR-DELETION	= *NONE	ACCESS	= *WRITE
MANAGEMENT-CLASS	= *NONE	AVAILABILITY	= *STD
NUM-OF-BACKUP-VERS	= 0		

  The attribute DISK-WRITE is set as default to *BY-CLOSE but may, however, be explicitly set to *IMMEDIATE. The attribute MIGRATE, which is set to the default value *INHIBITED, may also be set explicitly to *FORBIDDEN.

- Recataloging from temporary to permanent:

  The permanent file is assigned the following values if no explicit entries are made:

BACKUP-CLASS	= value of the class 2 option BACKUP
NUM-OF-BACKUP-VERS	= value of the NUMBACK Class2 option
DISK-WRITE	= *IMMEDIATE
MIGRATE	*ALLOWED is set for the permanent file if MIGRATE=*INHIBITED was set for the temporary file (MIGRATE=*FORBIDDEN remains unchanged).

  The remaining attributes are taken over unchanged from the temporary file.

- Recataloging from permanent to temporary:
  The temporary file is assigned the same values as when a temporary file is set up.The only difference is the value for MIGRATE:If MIGRATE=*ALLOWED was set for the permanent file, *INHIBITED is set for the temporary file (MIGRATE=*FORBIDDEN or *INHIBITED remains unchanged).
  Recataloging is rejected if the specified value for NUM-OF-BACKUP-VERS is > 0, irrespective of whether the permanent file is part of the version backup.
- Renaming from temporary to permanent and vice versa is rejected in an SM pubset if simultaneous changing of the file attributes requires reallocation to another volume set (S0 migration).
- Recataloging a work file or a file on a Net-Storage volume to a temporary file or vice versa is not allowed.

File generation groups (FGG)

The following points must be noted when creating or accessing file generation group catalog entries:

- If the user wishes to work with a file generation group (FGG), he must create the group entry before he catalogs the first generation. In contrast to files and file generations, which can be cataloged by means of FILE, the group entry can be created at the program level only by means of the CATAL macro.
- If the file generation group is indexed on public volumes (no VOLUME and/or DEVICE specification), the generations can be created on both public volumes and on tapes (FILE program interface).
  If the file generation group is indexed on a private disk (VOLUME/DEVICE specification), the generations can then also only be created on private disks (FILE program interface).
- Files can be recataloged as file generations if the file generations do not already exist. A file on a Net-Storage volume cannot be renamed as a file generation. However, file generations cannot be recataloged as files.
- The attributes (operands) STOCLAS, IOPERF, IOUSAGE, DISKWR and SOMIGR can be assigned or modified for the separate file generations of a file generation group.
  The entries of the user or system administration metainformation (USRINFO and ADMINFO operands) can be defined separately for the index of the file generation group and each individual file generation.
  The remaining attributes can only be defined for the complete file generation group. They are inherited automatically from the index by all cataloged generations.
- The USER-ACCESS attribute must not be set to SPECIAL for file generation groups.
- It is not possible or meaningful to assign execution rights for file generation groups since generations cannot be executed (/CALL-PROCEDURE or /START-PROGRAM is rejected).

- The protection attributes READ-PASSWORD, WRITE-PASSWORD and EXPIRATION-DATE do not protect the index of a file generation group against new generations being created using CATAL. This means that new generations can be cataloged and, dependent on the selected OVERFLOW-OPTION, old generations deleted regardless of the protection attributes.
- A new generation cannot be created in a generation group with the attribute ACCESS=READ using CATAL <generation name>,STATE=*NEW;
  CATAL <file><generation name>,STATE=*UPDATE must be used instead.
- The protection attributes BASIC-ACL and GUARDS protect the file generation group (index) as well as each individual file generation. This means that a caller who does not posses write authorization cannot create a new file generation in a file generation group which is write-protected with these attributes.
- File generation groups which are stored on private volumes and for which no catalog entries exist are called foreign file generation groups. If such FGGs are to be cataloged again, the group entry must first be created. For file generation groups on private disks, the operand STATE=*FOREIGN can be used for this purpose if the F1 label on the disk contains the group entry. The system then creates the catalog entry from the information in the F1 label of the private disk specified via the DEVICE and VOLUME operands. The associated file generations must then be imported (e.g. FILE macro, operand STATE=*FOREIGN).
- If a file generation group whose generations are stored on a tape or private disk is to be imported, and if the F1 label of the disk does not contain the group entry, the operand FIRST and at least one of the operands BASE or LAST must be specified in the CATAL macro to enable reconstruction of the group entry.
- When a file generation group is set up in the SM pubset, this must be defined as either a group of permanent generations or a working generation group (WORK-FILE attribute) by either implicitly assigning a default storage class or explicitly specifying the WORKGRP operand. It is not possible to subsequently change the attribute.
  If the generation concerned is assigned a storage class (during initial allocation via the FILE program interface) or the storage class is exchanged, the WORK-FILE attribute in the storage class must match the attribute of the group.

Files on tapes and tape cartridges

When creating or updating the catalog entries for tape files, some special features which result from the use of tapes must be observed.

- Details of shareability (SHARE), access type (ACCESS) and passwords are transferred, for files with standard labels, from the catalog entry to the file labels when the file is created. For foreign files the details of the access rights are transferred from the file labels into the catalog entry when the file is opened.
- Since file labels on a tape cannot be modified without destroying the file (this is a hardware restriction), and the contents of the catalog entry for a file must match the contents of the file labels, the access rights and the expiration date of a tape file cannot be modified by means of the CATAL macro once the file has been opened and closed correctly.
- If the tape file was cataloged by means of a FILE macro, the file protection attributes can be modified by means of the CATAL macro before the file is opened for the first time. These attributes are then transferred without further checking to the file labels when the file is created. In this way, it is possible to define write protection (ACCESS=READ) for a file which is still to be created. The file can then be opened as an output file and created; the write-protection then becomes effective.
  Note

  If a tape file is cataloged using FILE, it is shareable unless SHARE=NO is set by means of a CATAL macro before it is opened for the first time.

- If password protection is specified for a tape file, the label processing routines transfer the passwords to the HDR3 label from the catalog entry when the file is created, without checking them (the reverse applies when a file is imported, i.e. passwords are transferred from the HDR3 label into the catalog entry).
  The passwords are not checked for a file for which SECLEV=LOW is specified. If the system administrator selected password encryption when the system was generated, the encryption indicator in the HDR3 label is set to “1” when the file is opened.
- If a file (FILE=...) is to be renamed (NEWNAME=...), the new name may only consist of the old name plus a version designation enclosed in parentheses. The version designation must differ from any other version designation that may already be present.
  This restriction results from the tape label processing: for hardware reasons, the separate blocks of a tape file cannot be overwritten and the file name in the label is compared with the file name in the catalog entry when the file is opened.
- ACCESS types for tape files:
  - All OPEN modes are permitted with ACCESS=*WRITE.
  - Only the OPEN modes INPUT and REVERSE are permitted with ACCESS=*READ.
  - The access type is entered in the HDR1 label according to the entries in the ACCESS operand, as follows:
    ACCESS=*READ -> access type 1
    ACCESS=*WRITE -> access type 2
  - The ACCESS operand is used mainly for securing a file against destruction (ACCESS=*READ). Only the owner of a tape file can bypass checking of the access authorization by specifying SECLEV=LOW in the FCB macro.
- Shareability (USER-ACCESS/SHARE) for tape files:
  - The access type is entered in the HDR1 label according to the entries in the SHARE operand, as follows:
    SHARE=*NO (USER-ACCESS=*OWNER-ONLY) -> access type 1
    SHARE=*YES (USER-ACCESS=*ALL-USERS) -> access type 2
  - USER-ACCESS=*ALL-USERS is assumed as default if the catalog entry was created with the FILE call.
- DESTROY operand:
  If DESTROY=*YES was specified, the remaining part of the tape is deleted after the file is closed (CLOSE).

Using wildcard file names

- If a wildcard file name (selection and/or construction specification) is to begin with an asterisk and contains no further wildcards, the leading asterisk must be entered twice. Otherwise, the request is rejected.
  Examples: '**A' and '*A/Z' are valid, '*ABC' and 'P(A)' are invalid.
- If a wildcard file name (selection or construction) is to begin with two asterisks, these are handled with respect to the construction as a single asterisk.
  Example: a CATAL call with the parameters FILE='A.*.* and NEWNAME='**.OLD.*' renames an existing file 'A.TEST.1' to 'TEST.OLD.1'
- The contents of the class 2 option TEMPFILE do not represent a partially qualified file name in this case (in contrast to FSTAT). '.*' can or must be used instead.

Note on SM pubsets

The following specifications are ignored for files/generations/FGGs on volume sets with permanent data storage if the file identifier in the SM pubset in question has a default storage class and physical allocation is forbidden:
AVAIL, DISKWR, IOPERF, IOUSAGE, S0MIGR=*ALLOWED, STOCLAS=*NONE.

Overview of the macro functions

Function	FGG (Index) PUB	FGG (Index) PRV	Gene-ration PUB	Gene-ration PRV	Gene-ration TAP	perm. file PUB	perm. file PRV	perm. file TAP	temp. file PUB	temp. file TAP	Operand
Identify catalog entry	x	x	x	x	x	x	x	x	x	x	pathname
Rename file or FGG	x	x				x	x	x	x	x	$pathname_2$
Catalog entry - create - change - import	 x x	 x x x	 x x			 x x	x	x	 x x	x	STATE =NEW =UPDATE =FOREIGN
Permit read or write access	x	x				x	x	x			ACCESS
Access control withBASIC-ACL	x					x					BASACL
	x x x					x x x x					OWNERAR READ WRITE EXEC
	x x x					x x x x					GROUPAR READ WRITE EXEC
	x x x					x x x x					OTHERAR READ WRITE EXEC
Access control with GUARDS	x x x					X X X X					GUARDS READ WRITE EXEC
Transfer of protection attributes	x	x				x	x	x	x	x	PROTECT
Shareability	x	x				x	x	x			SHARE
Password protection: - write - read - execute	 x x	 x x				 x x x	 x x x	 x x x			 WRPASS RDPASS EXPASS
Expiration date (retention period)	x	x				x	x				EXDATE (RETPD)
Release for deleting	x	x				x	x				DELDATE
Automatic data destruction	x	x				x	x	x	x	x	DESTROY
Audit monitoring	x	x				x	x	x			AUDIT
Frequency of ARCHIVE backups	x	x				x	x				BACKUP
Scope of ARCHIVE backups	x	x				x	x				LARGE

HSMS migration permitted/not permitted	x	x				x	x	x			MIGRATE
Maximum number of file versions to be saved in the version backup archive						x					NUM_OF_BACKUP_VERS
SM pubset migration permitted/not permitted			x			x			x		S0MIGR
Availability			x			x					AVAIL
HSMS management class	x					x					MANCLAS
Storage class			x			x					STOCLAS
Enable/disable character set	x					x		x	x	x	CCS

Assign character set for node file on Net-Storage						x				NETCCS
Define performance attributes			x			x			x	IOPERF
Performance required for I/O operations			x			x			x	IOUSAGE
Suitability for processing in cache (DAB)			x			x				DISKWR
User metainformation	x		x			x			x	USRINFO
System administration metainformation	x		x			x			x	ADMINFO
Define FGG: Oldest generation First generation Base generation Overflow handling Define volume - volume - device Work group	 x x x x x x	 x x x x x x x								 GEN FIRST LAST BASE DISP VOLUME DEVICE WORK

Key

PUB : public volume

PRV : private volume (private disk)

TAP : tape

x: the attribute can be set or modified with CATAL

Format

Operation	Operands

CATAL	VERSION = 1 / 2 / 3 / 4
	,MF = C / D / E / L / M
	,PARAM = <name 1..8>
	,PREFIX = I/ <pre>
	,MACID = DK/ <macid>
	,ACCESS = *WRITE / *READ / *UNCHANGED
	,ADMINFO = *NONE / <c-string 1..8> / (<reg: A(char:8)>) /
	<var: char:8>
	,AUDIT = *NONE / *FAILURE / *SUCCESS / *ALL
	,AVAIL = *STD / *HIGH
	,BACKUP = *A / *B / *C / *D / *E

```
,BASACL = *NONE / *STD / *UNCHANGED

,BASE = <integer -99..9999> / (<reg: int:2>) / <var: int:2>

,CCS = *NONE / *STD / <c-string 1..8> / (<reg: A(char:8)>) /
       <var: char:8>

,CHECK = *NO / *MULTIPLE / *ERROR / *SINGLE / *CATALOG / *USERID

,DELDATE = *NONE / *UNCHANGED / <c-string 1..10> /
         (<reg: A(char:10)>) / <var: char:10> /
         [(<c-string 8..8> / (<reg: A(char:8)>) / <var: char:8>)]

,DESTROY = *NO / *YES / *UNCHANGED

,DEVICE = <c-string: device> / (<reg: A(char:8)>) / <var: char:8>

,DISKWR = *IMMEDIATE / *BY-CLOSE

,DISP = *CYCLE / *REUSE / *DELETE / *KEEP

,EXDATE = *UNCHANGED /
        <c-string 1..10> / (<reg: A(char:10)>) / <var: char:10>
       [(<c-string 8..8> / (<reg: A(char:8)>) / <var: char:8>)]

,EXPASS = *NONE / *UNCHANGED / <c-string 1..4> / <x-string 1..8>/
              <integer -2147483648..2147483647>
          (<reg: A(char:4)>) / <var: char:4>

,FILE = <c-string 1..80: filename 1..54 with-wild(80)> /
       (<reg: A(char:80)>) / <var: char:80>

,FIRST = <integer 1..9999> / (<reg: int:2>) / <var: int:2>

,GEN = <integer 0..255> / (<reg: int:2>) / <var: int:2>

,GROUPAR = *NO-ACCESS / (
           [READ = *NO / READ = *YES / R = *N / R = *Y]
           [,WRITE = *NO / WRITE = *YES / W = *N / W = *Y]
           [,EXEC = *NO / EXEC = *YES / X= *N / X = *Y] )

,GUARDS = *NONE / (
          [ READ = *NONE /
                   <c-string: filename 1..18 without cat-gen-vers>/
                   <var: char:18> / (<reg: A(char:18)>) ]
          [ ,WRITE = *NONE /
                   <c-string: filename 1..18 without cat-gen-vers>/
                   <var: char:18> / (<reg: A(char:18)>) ]
          [ ,EXEC = *NONE /
                   <c-string: filename 1..18 without cat-gen-vers>/
                   <var: char:18> / (<reg: A(char:18)>) ] ) /
         *UNCHANGED

,IOPERF = *STD / *HIGH / *VERY-HIGH / *USER-MAX

,IOUSAGE = *READ-WRITE / *WRITE / *READ
```

```
,LARGE = *NO / *YES

,LAST = <integer 1..9999> / (<reg: int:2>) / <var: int:2>

,LIST = *NO / *SYSOUT / *ERRORS-TO-SYSOUT

,MANCLAS = *NONE / <c-string: struct-name 1..8> /
           (<reg: A(char:8)>) / <var: char:8>

,MIGRATE = *ALLOWED / *INHIBITED / *FORBIDDEN

,NETCCS = *USER_DEF / *ISO / *NO-CONV / <c-string 1..8> /
          (<reg: A(char:8)>) / <var: char:8>

,NEWNAME = c-string 1..80: filename 1..54 with-constr-wild(80)> /
           (<reg: A(char:80)>) / <var: char:80>

,NUM_OF_BACKUP_VERS = <integer 0..32> /
           (<reg: A(int:1)>) / <var: int:1>

,OPNBACK = *NO / *YES

,OTHERAR = *NO-ACCESS / (
           [READ = *NO / READ = *YES / R = *N / R = *Y]
           [,WRITE = *NO / WRITE = *YES / W = *N / W = *Y]
           [,EXEC = *NO / EXEC = *YES / X= *N / X = *Y] )

,OWNERAR = *NO-ACCESS / (
           [READ = *NO / READ = *YES / R = *N / R = *Y]
           [,WRITE = *NO / WRITE = *YES / W = *N / W = *Y]
            [,EXEC = *NO / EXEC = *YES / X= *N / X = *Y] )

,PROTECT = *STD / *BY_DEF_PROT_OR_STD /
           (*FROM_FILE,<c-string: filename 1..54>) /
           (*FROM_FILE,(<reg: A(char:54)>)) /
            (*FROM_FILE,<var: char:54>)

,RDPASS = *NONE / *UNCHANGED / <c-string 1..4> /
          <x-string 1..8> / <integer -2147483648..2147483647>
          (<reg: A(char:4)>) / <var: char:4>

,RELSPAC = *ALLOWED / *IGNORED / *UNCHANGED

,RETPD = <integer 0..32767> / (<reg: int:2>) / <var: int:2>

,S0MIGR = *ALLOWED / *FORBIDDEN

,SHARE = *NO / *YES / *SPECIAL / *UNCHANGED

,STATE = *NEW / *UPDATE / *FOREIGN

,STOCLAS = *STD / *NONE / *UPDATE / <c-string: struct-name 1..8>/
           (<reg: A(char:8)>) / <var: char:8>

,TIMBASE = *UTC / *LTI

,USRINFO = *NONE / <c-string 1..8> / (<reg: A(char:8)>) /
           <var: char:8>
```

```
,VOLUME = <c-string: vsn 1..6> / (<reg: A(char:6)>) /
          <var: char:6>

,WORKGRP = *YES

,WRPASS = *NONE / *UNCHANGED / <c-string 1..4> /
          <x-string 1..8> / <integer -2147483648..2147483647>
            (<reg: A(char:4)>) / <var: char:4>
```

Operand descriptions

ACCESS

The ACCESS operand can be used to protect a file against overwriting. It specifies whether write/read or only read access is permitted for the file or file generation. This protection attribute is only relevant if no BASIC-ACL or GUARDS protection is activated.

Tape files:
when the file is opened for the first time, DMS places the ACCESS indicator in its HDR3 label. For subsequent accesses, the file owner can bypass access type checking by specifying SECLEV=LOW (see FCB - Define file control block, section "SECLEV" and FILE - Define file attributes, section "SECLEV").

Default setting (only in conjunction with STATE=*NEW): ACCESS=*WRITE

= *WRITE
All access types are permitted for the file or file generations.
Tape files, HDR3 label: access type = 0.

= *READ
Only read access is permitted for the file or the file generations, i.e. only the OPEN modes INPUT and REVERSE are permitted.
Temporary files: write access cannot be prevented; ACCESS=READ is rejected.
Tape files, HDR3 label: access type = 1.

= *UNCHANGED
Only relevant if PROTECT is specified:
If STATE=*UPDATE is specified at the same time, the value of ACCESS remains unchanged. If STATE=*NEW is specified at the same time, the value ACCESS=*WRITE is entered.

Specification of *UNCHANGED has the following effects:

- if PROTECT=*FROM-FILE is specified: the value *UNCHANGED prevents the corresponding value from being taken over from the reference file
- if PROTECT=*BY_DEF_PROT_OR_STD is specified and if STATE=*NEW is specified without a value for PROTECT:
  the value *UNCHANGED prevents the corresponding value supplied by default protection from being taken over
- if PROTECT=*STD and STATE=*UPDATE are specified together: the value *UNCHANGED prevents the value in the catalog entry from being reset to the value ACCESS=*WRITE

If STATE=*UPDATE is specified, then: if PROTECT is not specified, *UNCHANGED has the same effect as no specification.

If STATE=*NEW is specified, then: *UNCHANGED has the same effect as ACCESS=*WRITE (irrespective of what is specified for PROTECT).

ADMINFO

Reserved for system administrators (called under the TSOS ID):
Enters system administration metainformation into the file catalog entry. The entry can have a maximum of 8 bytes, with any contents; the system administrator defines the meaning.The operand is ignored for files on private volumes.

= *NONE
The entry is deleted.

= <c-string 1..8>
The specified characters are entered.

= (<reg: A(char:8)>)
Only possible with MF=M:
The specified register contains the address of an 8-byte memory area containing the metainformation to be entered.

= <var: char:8>
Only possible with MF=M:
Symbolic address of an 8-byte memory area containing the metainformation to be entered.

AUDIT

Reserved for user IDs with AUDIT=YES rights:
Defines whether DMS accesses to files or file generations are to be monitored with the aid of system exit routines. Monitoring applies to the operations CATAL, FILE, OPEN and ERASE.
If the user does not have the authorization AUDIT=YES, a CATAL macro specifying AUDIT will be rejected.

Default setting (only in conjunction with STATE=*NEW): AUDIT=*NONE

= *NONE
No monitoring.

= *All
All DMS operations for the file/generation are monitored.

= *SUCC
All successful DMS operations for the file/generation are monitored.

= *FAIL
All unsuccessful DMS operations for the file/generation are monitored.

AVAIL

*Only relevant with STATE=*UPDATE for files on public volumes or on a Net-Storage volume:*
The file availability requirements are modified. Files that are to have an increased availability are reallocated to an appropriate volume set (e.g. DRV – Dual Recording by Volume).
An existing storage class entry in the SM pubset file catalog entry is removed by specifying the operand.

The operand must not be specified simultaneously with the STOCLASnot equal to *NONE operand.

The operand is ignored for files and generations on volume sets with permanent data storage if the file identifier on the SM pubset concerned has a default storage class and physical allocation is forbidden.

= *STD
No special availability requirements are set.

= *HIGH
The file is to have increased availability. It is ensured that the file is allocated to a corresponding volume set. If the current storage location does not provide the requirements of increased availability, the request is rejected for SF pubsets. The request is only rejected in an SM pubset if no suitable volume set is available or if the permitted user ID allocations are exceeded. Otherwise, the storage space is reallocated to a suitable volume set.

If the file is stored on a private volume, if it is a work file, or if it has been migrated to a background level, the request is rejected with a return code. The request is also rejected for temporary files, even if a temporary file is renamed to a permanent file. In this case, the file must be renamed first and then increased availability can be assigned with an additional CATAL call.

Files in SM pubsets:
If the current storage location (volume set) does not provide the requirements of high availability and a suitable volume set is in the SM pubset, the data or storage space is automatically reallocated. The file is locked (opened) during this reallocation, i.e. all accesses to the file or its catalog entry are rejected instead of being put into a wait state.

BACKUP

Valid only for files or FGGs on disks:
controls automatic file backup with the archiving system ARCHIVE or HSMS; specifies in which backup runs the files or the generations of the FGG are to be saved.

Default setting:	for permanent files:	according to class 2 option BACKUP
	for temporary files:	BACKUP=*E

= *A
The files/generations are to be saved in each backup run.

= *B
The files/generations are to be saved when a backup run occurs for files with BACKUP=*B, *C or *D.

= *C
The files/generations are to be saved in backup runs with BACKUP=*C or *D.

= *D
The files/generations are to be saved only in backup runs with BACKUP=*D.

= *E
No automatic backup via ARCHIVE.

BASACL

Activates or deactivates access control via the BASIC-ACL with standard access rights. Protection is only effective if no GUARDS protection is activated.

The operand must not be specified together with the OWNERAR, GROUPAR or OTHERAR operand.

Default setting (only in conjunction with STATE=*NEW): BASACL=*NONE

= *NONE
Deactivates access control via BASIC-ACL:

= *STD
Enters the following standard access control rights in the basic ACL:

- If STATE=*NEW is specified or if no STATE value is specified, the following access rights are entered:

  ```
  OWNER  GROUP  OTHERS
  R W X  - - -  - - -
  ```

  This corresponds to the following operand entries:
  OWNERAR=(READ=*YES,WRITE=*YES,EXEC=*YES),GROUPAR=*NO-ACCESS,OTHERAR=*NO-ACCESS.
- If STATE=*UPDATE is specified (i.e. if the file involved is already cataloged), the valid values for SHARE and ACCESS are converted to BASIC-ACL values if access control via the basic access control list was not activated before.

The values are converted in accordance with the following table:

SHARE	ACCESS	OWNER	GROUP	OTHERS
NO	READ	R - X	- - -	- - -
NO	WRITE	R W X	- - -	- - -
YES / SPECIAL	READ	R - X	R - X	R - X
YES / SPECIAL	WRITE	R W X	R W X	R W X

Notes

If the operands SHARE, ACCESS and/or PROTECT are specified together with BASACL=*STD, conversion is carried out according to these entries.
Nothing (no access) is entered for EXEC with file generation groups.

= *UNCHANGED

Only relevant in conjunction with specification of PROTECT:
If STATE=*UPDATE is specified at the same time, the value of BASIC-ACL remains unchanged. If STATE=*NEW is specified at the same time, no BASIC-ACL is entered.

Specification of *UNCHANGED has the following effects:

- if PROTECT=*FROM-FILE is specified:the value *UNCHANGED prevents the corresponding value from being taken over from the reference file
- if PROTECT=*BY_DEF_PROT_OR_STD is specified and if STATE=*NEW is specified without a value for PROTECT:
  the value *UNCHANGED prevents the corresponding value supplied by default protection from being taken over
- if PROTECT=*STD and STATE=*UPDATE are specified at the same time:the value *UNCHANGED prevents the value in the catalog entry from being reset (no BASIC-ACL)

If STATE=*UPDATE is specified, then: if PROTECT is not specified, *UNCHANGED has the same effect as no specification.

If STATE=*NEW is specified, then: *UNCHANGED has the same effect as no BASIC-ACL in the catalog entry (irrespective of what is specified for PROTECT).

BASE

For file generation groups only:
defines a reference point (a base generation) to which all relative generation numbers are related and also enables the reconstruction of an index for file generations on private volumes.
If the BASE operand is not specified when creating an FGG (group index), it is assigned the value of the FIRST operand if that operand is specified; otherwise, it is assigned the value 0.

Use of the BASE operand for index reconstruction:
If generations of an FGG are to be imported from a private disk, and if the group entry is neither in the catalog nor in the F1 label of the disk, the group index must first be reconstructed. A CATAL call with the operands STATE=*NEW, GEN=<num>, FIRST=<num>, and at least one of the operands BASE or LAST must be executed for this purpose. If only the BASE operand is specified, it also defines the most recent file generation, i.e. the value for LAST.

= <integer -99..9999>
The new base generation can be defined in relation to the specified “number” in absolute or relative form.

Absolute form: value range: 1 <= number <= 9999.

- STATE=*UPDATE: “number” is assigned as the new base value; it must designate an existing generation according to the index entry.
- STATE=*NEW: “number” is added to the catalog as the base value.If BASE is specified in conjunction with FIRST and/or LAST, the specified number must fulfill the following condition: FIRST <= number <= LAST.

Relative form:

- *Only possible with STATE=*UPDATE:*
  value range: -99 <= number <= 0.
  Defines the base generation relative to the most recently cataloged generation (catalog field `LAST-GEN`) according to the index.
  The new base value must designate an existing file generation in accordance with the index, i.e. it must be >= the value in the output field `FIRST-GEN`.

= (<reg: int:2>)
Only possible with MF=M:
The base value (see above for value range and meaning) is stored in the lower half-word of the specified register.

= <var: int:2>
Only possible with MF=M:
Symbolic address of a half-word in which the base value is stored (see above for value range and meaning).

CCS

Only valid for files/FGGs on public volumes, for files on Net-Storage and for tape files:
Character set to be used for the file.
The coded character set (CCS) defines how the characters of a national character set are to be stored in binary form. The specified character set has an effect on the representation of characters on the screen, the collating sequence, etc. (see the “XHCS” manual [22]).

The operand is ignored for files on private volumes; as a result, no return code is supplied.

Default setting (only in conjunction with STATE=*NEW): CCS=*NONE

= *NONE
No character set is to be specified for the file.

= *STD
The character set is taken over from the file owner’s user catalog entry provided a character set which is not `EDF03IRV` is entered. Otherwise *NONE applies.

= <c-string 1..8>
Name of the coded character set with which the file is to be processed (e.g.: `EDF03IRV` for the international version of `EBCDIC.DF.03`).
The specified string is not checked, in particular not whether it is the name of a defined coded character set.

= (<reg: A(char:8)>)
Only possible with MF=M:
The specified register contains the address of an 8-byte memory area containing the name of the coded character set.

= <var: char:8>
Only possible with MF=M:
Symbolic address of an 8-byte memory area containing the name of the coded character set.

CHECK

Defines the conditions under which a user dialog is to be started.

When the dialog is started, the user can decide whether the displayed files are processed or not. He can also call up help text on the reply options and define a new value for LIST and/or CHECK when processing is resumed.

The value *NO always applies in batch mode.

Default: CHECK=*NO

= *NO
All selected files are processed without a check dialog, i.e. the user cannot intervene.

= *MULTIPLE
A check dialog is only started if more than one file is selected.If the catalog and/or user ID contain wildcards, a check dialog is executed for each catalog and/or user ID.
CHECK=*ERROR is also implied.

= *ERROR
An error check dialog is started if an error occurs while a selected file name is being processed. A file set check dialog is started if the selection entry selects more files than can be processed in available memory.
CHECK=*ERROR is also always implied for CHECK not equal to *NO.

= *SINGLE
A check dialog is executed for each selected file name. CHECK=*ERROR is also implied.

= *CATALOG
The user must decide in a check dialog for each catalog whether the files selected on it are to be processed.
CHECK=*ERROR is also implied.

= *USERID
Reserved for system administrators.
The system administrator must decide in a check dialog for each user ID on each catalog whether the selected files are to be processed.CHECK=*ERROR is also implied.

DELDATE

Only for files on public volumes and for files on Net-Storage:
Determines the time after which the file may be deleted regardless of the protection attributes ACCESS, BASACL, EXDATE, GUARDS, RDPASS, WRPASS and EXPASS or after which its storage place may be released.

An absolute date is interpreted according to the TIMEBASE operand, either based on local time (LTI) or the universal time coordinate (UTC), while a relative date is always based on local time.

= *NONE
The file should not be deleted without taking the protection attribute into account (corresponds to DELDATE='+0').

= *UNCHANGED
If STATE=*UPDATE is specified at the same time, the value of DELDATE remains unchanged. If STATE=*NEW is specified at the same time, the value DELDATE=*NONE is entered.

If STATE=*UPDATE is specified, then: if PROTECT=*BY_DEF_PROT_OR_STD is specified, the value *UNCHANGED prevents the transfer of the corresponding value supplied by the default protection function; otherwise *UNCHANGED has the same effect as no specification.

If STATE=*NEW is specified, then: *UNCHANGED has the same effect as the specification *NONE (irrespective of what is specified for PROTECT).

= <c-string 1..10> / (<reg: A(char:10)>) / <var: char:10>
[<c-string 8..8> / (<reg: A(char:8)>) / <var: char:8>]
Defines the time from which the file may be deleted regardless of the protection attributes, where the variable “...10” stands for the date and “...8” for the time.

The following formats are allowed:

- '+<integer 0..99999>'[('<time 8..8>')]
- '<date 8..10>'[('<time 8..8>')]
- 'yymmdd'[('<time 8..8>')]

It is imperative that the '+' is specified for relative date entries to discriminate them from absolute ones.

Date entries of less than 10 characters must be terminated with blanks.

Two-digit year specifications from 00 to 59 are complemented with 20, from 60 to 99 with 19.

Examples

```
          ...
          LA    7,DELTIME
BEISP1    CATAL MF=M,VERSION=4,...,DELDATE='+1'((7))
          LA    6,DELDATE
BEISP2    CATAL MF=M,VERSION=4,...,DELDATE=(6)(DELTIME)
BEISP3    CATAL MF=M,VERSION=4,...,DELDATE=DELDATE('00:00:00')
          ...
DELDATE   DC    CL10'2016-11-11'    11.11.2016
DELTIME   DC    CL8'11:11'          11:11:00
```

DESTROY

In order to improve data protection, the user can specify in the catalog entry that data which is no longer needed is to be overwritten with X'00' (binary zeros). For disk files, this applies to erase operations (see the ERASE command); for tape files, it applies to the overwriting of data remaining on the tape during EOF or EOV processing (see the DESTOC operand of the FILE macro, "FILE - Define file attributes / control file processing").

Default setting (only in conjunction with STATE=*NEW): DESTROY=*NO

= NO
Disk files:
The storage space is simply released unless the operand DESTROY=*YES is specified in the ERASE macro.

Tape files:
Any further data on the tape is not overwritten unless the operand DESTOC=YES is specified in the FILE macro for the current processing run.

= YES
Disk files:
The storage space which is released is automatically overwritten with binary zeros (X'00').

Tape files:
Any data remaining on the tape is erased; this can also be done using the FILE macro for the current processing run by specifying the operand DESTOC=YES.

= *UNCHANGED
Only relevant in conjunction with specification of PROTECT:
If STATE=*UPDATE is specified at the same time, the value of DESTROY remains unchanged. If STATE=*NEW is specified at the same time, the value DESTROY=*NO is entered.

Specification of *UNCHANGED has the following effects:

- if PROTECT=*FROM-FILE is specified:
  prevents the corresponding value from being taken over from the reference file
- if PROTECT=*BY_DEF_PROT_OR_STD is specified and if STATE=*NEW is specified without a value for PROTECT:
  prevents the corresponding value supplied by default protection from being taken over
- if PROTECT=*STD and STATE=*UPDATE are specified at the same time:
  prevents the value in the catalog entry from being reset to the value DESTROY=*NO

If STATE=*UPDATE is specified, then: if PROTECT is not specified, *UNCHANGED has the same effect as no specification.

If STATE=*NEW is specified, then: *UNCHANGED has the same effect as DESTROY=*NO (irrespective of what is specified for PROTECT).

DEVICE

Only for file generation groups on private disks and in conjunction with the VOLUME operand:
Specifies the device type on which the file generation group is to be stored or the device type from which it is to be imported (see also the DEVICE operand of the FILE macro).

DEVICE does not have to be specified if MAREN is available.

= <c-string: device>
Device type: permissible entries are listed in the device table in „System installation“ manual [16].
Every disk device type specification is handled like the STDDISK specification.

= (<reg: A(char:8)>)
Only possible with MF=M:
The specified register contains the address of an 8-byte memory area containing the device type.

= <var: char:8>
Only possible with MF=M:
Symbolic address of an 8-byte memory area containing the device type.

DISKWR

Only for files and file generations on public volumes and files on Net-Storage volumes:
Specifies the time at which data consistency is required after a write operation.If the file is processed via a temporary cache for writing, data in the file will not be in a consistent state until CLOSE processing has been completed. This means that system errors during the processing phase could lead to inconsistencies. Immediate data consistency after each write operation should therefore be requested for files which contain important data.

Default setting:

DISKWR=*IMMEDIATE	for permanent files
DISKWR=*BY-CLOSE	for temporary files

If DISKWR is not specified when recataloging from temporary to permanent or vice versa, the entry is set automatically to the respective default/permitted value.

The operand is ignored (no return code!) if:

- it is specified for files which are not located on a public volume or which are to be created.
- the file identifier on the SM pubset concerned has a default storage class and physical allocation is forbidden.

A storage class entered in the file catalog entry in an SM pubset is removed if this operand is specified. The operand must not be specified simultaneously with the operand STOCLASnot equal to *NONE.

= *IMMEDIATE
Data contained in the file must be in a consistent state immediately after a write operation, so a volatile write cache should not be used when processing the file (default value for permanent files).
This entry is ignored for temporary files.

= *BY-CLOSE

Data in the file need not be in a consistent state (i.e. written to disk) until CLOSE processing has been completed. This is the default value for temporary files.

DISP

Only for file generation groups:
Specifies whether the oldest generations are to be erased and, possibly, their storage space reused when the maximum number of simultaneously existing generations specified via GEN= is exceeded. In the case of generations on tape, only the catalog entry is deleted.Existing expiration dates for the oldest generations, if any, are ignored.

Default setting (only in conjunction with STATE=*NEW): DISP=*CYCLE

= *CYCLE
The oldest existing generation is erased and its storage space on disk or the tapes it occupies is/are released. The fields LAST-GEN and FIRST-GEN in the group entry (youngest and oldest existing generations) are updated accordingly.

= *REUSE
The effects of DISP=*REUSE depend on the type of volume:

For FGGs on public disks:
The oldest generation is erased, its storage space is returned to the system, and the group entry is updated (see DISP=*CYCLE).

For FGGs on private disks:
The new generation is created, the oldest generation is erased, and the volume of the oldest generation is used for storing the new generation. If the generation which was deleted extended over several volumes, the new generation is cataloged only on the first of these volumes. The catalog entry is updated accordingly. Since the old generation is erased only after the new generation has been created, insufficient space on the volume can mean that the new generation cannot be created although DISP=*REUSE is specified.

For FGGs on tape:
The oldest generation is deleted from the catalog and the new generation is created on the tapes which become free. The group entry is updated accordingly. DISP=*REUSE must not be specified for FGGs in MF /MV sets.

= *DELETE
All generations of the FGG are erased and the new generation becomes the oldest generation of a new series. The group entry is updated accordingly.

= *KEEP
The “superfluous” oldest generations are not erased automatically, but only when the user, in a further CATAL macro with the operands FIRST and BASE, defines a new “oldest” and a new base generation, or when the user specifies a new value for DISP=. As each new generation is created, only the field LAST-GEN in the group entry is updated.

EXDATE

Specifies the period (EXPIRATION-DATE) during which the file cannot be modified or deleted, i.e. when it is only available for read access (“read only”).

An expiration date can only be set for existing files, i.e. the catalog fields CRE-DATE and FILE-STRUC must have a value not equal to NONE. This also means that the CATAL operands EXDATE and STATE=*NEW or STATE=*FOREIGN cannot be combined (EXDATE is ignored).

An absolute date is interpreted according to the TIMEBASE operand, either based on local time (LTI) or the universal time coordinate (UTC), and a relative date is always based on local time.

A time of 00:00:00 is always assumed in conjunction with TIMBASE=*LTI or a relative time entry. An explicit time specification is only accepted in conjunction with TIMBASE=*UTC. The minutes and seconds are, however, always set to zero.

Note

If the specified expiration date is earlier than the current date, it is not entered. Instead the current date is entered with 0.00 hours local time.

Simultaneous use of the EXDATE and RETPD operands is not permitted.

= <c-string 1..10> / (<reg: A(char:10)>) / <var: char:10>
[(<c-string 8..8> / (<reg: A(char:8)>) / <var: char:8>)]
Defines the time from which the file may be modified, where the variable “...10” stands for the date and “...8” for the time.

The following formats are permitted:

- '+<integer 0..99999>'
- '<date 8..10>'[('<time 8..8>')]
- 'yymmdd'[('<time 8..8>')]

It is imperative that the '+' is specified for relative date entries to distinguish them from absolute ones.

Date entries of less than 10 characters must be terminated with blanks.

Two-digit year specifications from 00 to 59 are complemented with 20, from 60 to 99 with 19.

Examples

```
         ...
         LA    6,EXPDATE
         LA    7,EXPTIME
BEISP1   CATAL MF=M,VERSION=4,...,EXDATE=(6)((7))
BEISP2   CATAL MF=M,VERSION=4,...,EXDATE=(6)('23:00:00')
BEISP3   CATAL MF=M,VERSION=4,...,EXDATE=EXPDATE(EXPTIME)
         ...
EXPDATE  DC    CL10'161231'        31.12.2016
EXPTIME  DC    CL8'00:00:00'       00:00:00
```

= *UNCHANGED

Only relevant in conjunction with specification of PROTECT:
If STATE=*UPDATE is specified at the same time, the value of EXDATE remains unchanged.

The value *UNCHANGED has the following effects for permanent files with a creation date and for file generation groups with a cataloged group entry:

- if PROTECT=*FROM-FILE is specified:
  prevents the corresponding value from being taken over from the reference file
- if PROTECT=*BY_DEF_PROT_OR_STD is specified:
  prevents the corresponding value supplied by default protection from being taken over
- if PROTECT=*STD is specified:
  prevents the value in the catalog entry from being reset to the current date

If STATE=*UPDATE is specified, then: if PROTECT is not specified, *UNCHANGED has the same effect as no specification.

EXPASS

Only for files; not for FGGs or file generations:
This operand is used to define or delete the execute password. Execute protection is provided for the call of a program or a procedure file by means of the command START-PROGRAM, LOAD-PROGRAM, CALL-PROCEDURE or ENTER-PROCEDURE.

Tape files:
The password protection is recorded in the HDR3 label.

Encrypted files:
All EXPASS specifications are handled like *UNCHANGED.

Default setting (only in conjunction with STATE=*NEW:) EXPASS=*NONE

= *NONE
No execute password is defined or an existing password is deleted.

= *UNCHANGED
If STATE=*UPDATE is specified at the same time, the value of EXPASS remains unchanged. If STATE=*NEW is specified at the same time, the value EXPASS=*NONE is entered.
If PROTECT=*BY_DEF_PROT_OR_STD or STATE=*NEW is specified without a value for PROTECT, *UNCHANGED prevents the corresponding value supplied by default protection from being taken over.
If STATE=*UPDATE is specified, then: if PROTECT=*BY_DEF_PROT_OR_STD is not specified, *UNCHANGED has the same effect as no specification.
If STATE=*NEW is specified, then: *UNCHANGED has the same effect as the specification *NONE (irrespective of what is specified for PROTECT).

= <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
Defines a password required for calling the program or procedure.

The specification EXPASS=X'00000000' is treated in the same way as *NONE.

Before writing to the file, “password” must be entered in the password table of the job by means of the ADD-PASSWORD command (see the corresponding description in the “Commands” [3] manual).
If password assignment is logged, the passwords are not shown in plaintext.

= (<reg: A(char:4)>)
Only possible with MF=M:
The specified register contains the address of a 4-byte memory area containing the execution password.

= <var: char:4>
Only possible with MF=M:
Symbolic address of a 4-byte memory area containing the execution password.

FILE

Path name of the file, file generation or file generation group catalog entry which is referenced by the entries of the remaining macro operands.

- With STATE=*NEW (default), a catalog entry is created under the specified name.
- With STATE=*FOREIGN, a file generation group catalog entry is imported which exists on the private volume specified with the VOLUME and DEVICE operands.
- With STATE=*UPDATE, a catalog entry can be modified which exists under the specified path name. If wildcards are specified, all catalog entries selected can be modified.

= <c-string 1..80: filename 1..54 with-wild(80)>
The path name consists of [:catid$_1$:][$userid$_1$.]<filename$_1$>.

catid $_1$
ID of the catalog which contains or is to contain the file catalog entry. Wildcards may be used if STATE=*UPDATE is specified. However, only those files in the catalog are then selected that are locally available.
The default catalog ID of the caller (DEFAULT-PUBSET) is assumed if no catalog ID is specified.

userid $_1$
User ID under which the file is stored or is to be created. Only the system administrator may also specify wildcards with STATE=*UPDATE.

- The logon ID is assumed if no user ID is specified.
- The system administrator may specify a foreign user ID if no TSOS restriction has been declared for the file (see the “SECOS” manual [8]) or if STATE=*UPDATE is not specified.
- Nonprivileged users may specify a foreign user ID if they are co-owners of the file.

filename $_1$
Name of the temporary or permanent file, the file generation or file generation group.
The file may contain wildcards or be partially qualified (i.e. end with a period) if STATE=*UPDATE is specified.

= (<reg: A(char:80)>)
Only possible with MF=M:
The specified register contains the address of an 80-byte memory area containing the path name. If the path name is shorter than the maximum length of 80 bytes, it must be terminated with at least one blank (X'40').

= <var: char:80>
Only possible with MF=M:
Symbolic address of an 80-byte memory area containing the path name. If the path name is shorter than the maximum length of 80 bytes, it must be terminated with at least one blank (X'40').

FIRST = num

Only for file generation groups:
This operand may only be specified with STATE=*NEW.
It defines the absolute generation number of the oldest *cataloged* file generation. It is needed in order to reconstruct the index entry of a file generation group on private volumes (foreign file generation group) and should be used only for this purpose. The FIRST operand specifies the number of the oldest file generation to be imported.
File generations stored on tape must be cataloged individually using FILE (operand STATE=FOREIGN).
Generations stored on private disk can be imported by using IMPORT or individually by means of FILE (operand STATE=FOREIGN).

= <integer 1..9999>
The file generation specified by FIRST can only be imported, not recataloged. This means that if the created index entry is not to be used for the reconstruction of a file generation group, there is no way of actually creating the file generation group specified here (or any other file generations <= LAST or <= BASE; see also the description of those operands).

= (<reg: int:2>)
Only possible with MF=M:
The specified register contains the generation number of the oldest cataloged file generation in the lower half-word.

= <var: char:2>
Only possible with MF=M:
Symbolic address of a half-word containing the generation number of the oldest cataloged file generation.

GEN = num

Only for file generation groups:
Specifies how many generations of a file generation group may be cataloged concurrently (see also the DISP operand).

GEN may be specified for a new (STATE=*NEW) or an existing file generation group (STATE=*UPDATE).

Default value: GEN = 0

= <integer 0..255>
Maximum number of concurrently cataloged file generations.If GEN=0 is specified together with STATE=*NEW, a “normal” file rather than a file generation group is created; if it is specified together with STATE=*UPDATE, GEN=0 is ignored.

= (<reg: int:2>)
Only possible with MF=M:
The specified register contains the maximum number of concurrently cataloged file generations in the lower half-word.

= <var: char:2>
Only possible with MF=M:
Symbolic address of a half-word containing the maximum number of concurrently cataloged file generations.

GROUPAR

Only for files on public volumes and on Net-Storage:
Activates access control via the BASIC-ACL and specifies how a user who is not the file owner but who belongs to the same user group as the file owner may access the file when no GUARDS protection is active.
User groups can be defined in a system only if the software product SECOS is installed (see the “SECOS” manual [8]). In a system without user groups and in which SECOS has not been installed, the value for GROUPAR applies to all users except the file owner (and the system administrator). When user groups are defined in the system, this value is evaluated for the members of the file owner's user group.

The operand must not be specified together with the BASACL operand.

= *NO-ACCESS
No access to the file is permitted for the user group.

= ([READ = *NO / READ = *YES / R = *N / R = *Y]
[,WRITE = *NO / WRITE = *YES / W = *N / W = *Y]
[,EXEC = *NO / EXEC = *YES / X= *N / X = *Y])
The access types for which *YES or *Y is specified in the list are permitted. The parentheses are component parts of the access list and must be specified.

The various elements of the access list have the following meanings:

READ=NO or R=N	Read access is forbidden (default value).
READ=YES or R=Y	Read access is permitted. In contrast to access control via the ACCESS operand, this does *not* automatically imply the right to execute the file.
WRITE=NO or W=N	Write access is forbidden (default value).
WRITE=YES or W=Y	Write access is permitted. In contrast to access control via the ACCESS operand, this does *not* automatically imply the right to read or execute the file.
EXEC=NO or X=N	Execution of the file is forbidden (default value).
EXEC=YES or X=Y	Execution of the file is permitted (not for file generation groups).

GUARDS

Activates/deactivates access control using GUARDS (Generally Usable Access contRol aDministration System). GUARDS protects the file by means of a special access profile. This access protection will be effective only if the GUARDS function unit of the software product SECOS is loaded (see the “SECOS” manual [8]).

File protection with GUARDS is activated if at least one access mode (READ/WRITE/EXEC) is linked with a “guard” entry in the “guard” catalog. The specification of READ/WRITE/EXEC=*NONE is also considered to be a “guard” entry and activates the GUARDS file protection mechanism (thus preventing read, write, or execute access to the file).This specification can be given even if the access profile has not yet been defined and even if the GUARDS function unit is not being used. In both cases, all attempts to access the file will be rejected.

It is only at the time of accessing a file protected with GUARDS that a check is performed to determine whether the specified guard entry (guard name) exists, whether it may be used, and whether the corresponding access profile permits the user to access the file in the desired access mode.

Notes

- If GUARDS protection is entered for a file in the file catalog but no access profile has been defined for the specified guard name in the guard catalog, the file in question cannot be accessed.
- If GUARDS protection is enabled, the access protection defined previously using BASIC-ACL or USER-ACCESS and ACCESS is retained.
- For more information on access protection using the GUARDS function unit, see the section on “File protection” in the “Introductory Guide to DMS” [1].

= *NONE
Deactivates GUARDS protection, thus disabling any existing access protection provided by a guard. In other words, the file will no longer be protected by the GUARDS protection mechanism.

= ([READ...] [,WRITE...] [,EXEC...])
Each of the three access modes (read, write, execute) can be protected by means of a separate guard entry. When GUARDS protection is activated for a file, all access modes not explicitly specified are set to *NONE, which means that they are **not** permitted.

[READ = *NONE / <c-string: filename 1..18 without cat-gen-vers> /
<var: char:18> / (<reg: A(char:18)>)]
Activates read access control using GUARDS.

Default value: READ = *NONE, if GUARDS protection was activated via another access mode.

= *NONE
Disables GUARDS protection for read access (i.e. cancels the link between read access control and the access profile). File protection via GUARDS remains active, but the file cannot be read.

= <c-string: filename 1..18 without cat-gen-vers>
Name of the access profile (guard entry in the guard catalog) that provides read protection via GUARDS. Read access to the file is granted only if the conditions specified in the access profile are fulfilled.
The name must not exceed a maximum length of 8 characters (18 characters if specified with the user ID). It is not possible to specify a catalog ID.

= (<reg: A(char:18)>)
*Only possible with MF=M:*The specified register contains the address of an 18-byte memory area containing the name of the READ-GUARD.

= <var: char:18>
Only possible with MF=M:
Symbolic address of an 18-byte memory area containing the name of the READ-GUARD.

[,WRITE = *NONE / <c-string: filename 1..18 without cat-gen-vers> /
<var: char:18> / (<reg: A(char:18)>)]
Activates write protection using GUARDS.

Note

Unlike password protection, write access does not automatically imply read access.

Default value: WRITE = *NONE, if GUARDS protection was activated via another access mode.

= *NONE
Disables GUARDS protection for write access (i.e. cancels the link between write access control and the access profile). File protection via GUARDS remains active, but no write access to the file is possible.

= <c-string: filename 1..18 without cat-gen-vers>
Name of the access profile (guard entry in the guard catalog) that provides write protection via GUARDS.
Write access to the file is granted only if the conditions specified in the access profile are fulfilled.
The name must not exceed a maximum length of 8 characters (18 characters if specified with the user ID). It is not possible to specify a catalog ID.

= (<reg: A(char:18)>)
Only possible with MF=M:
The specified register contains the address of an 18-byte memory area containing the name of the WRITE-GUARD.

= <var: char:18>
Only possible with MF=M:
Symbolic address of an 18-byte memory area containing the name of the WRITE-GUARD.

[,EXEC = *NONE / <c-string: filename 1..18 without cat-gen-vers> /
<var: char:18> / (<reg: A(char:18)>) / *UNCHANGED])
Activates execute protection using GUARDS.

Default value: EXEC = *NONE, if GUARDS protection was activated via another access mode.

= *NONE
Disables GUARDS protection for execute access (i.e. cancels the link between execute access control and the access profile). File protection via GUARDS remains active, but no execute access to the file is possible.

= <c-string: filename 1..18 without cat-gen-vers>
Name of the access profile (guard entry in the guard catalog) that provides execute protection via GUARDS.
Write access to the file is granted only if the conditions specified in the access profile are fulfilled.
The name must not exceed a maximum length of 8 characters (18 characters if specified with the user ID). It is not possible to specify a catalog ID.

= (<reg: A(char:18)>)
Only possible with MF=M:
The specified register contains the address of an 18-byte memory area containing the name of the EXEC-GUARD.

= <var: char:18>
Only possible with MF=M:
Symbolic address of an 18-byte memory area containing the name of the EXEC-GUARD.

= *UNCHANGED
Only relevant in conjunction with specification of PROTECT:
If STATE=*UPDATE is specified at the same time, the value of GUARDS remains unchanged. If STATE=*NEW is specified at the same time, the value GUARDS=*NONE is entered.

The value *UNCHANGED has the following effects:

- if PROTECT=*FROM-FILE is specified:
  prevents the corresponding value from being taken over from the reference file
- if PROTECT=*BY_DEF_PROT_OR_STD is specified and if STATE=*NEW is specified without a value for PROTECT:
  prevents the corresponding value supplied by default protection from being taken over
- if PROTECT=*STD and STATE=*UPDATE are specified at the same time:
  prevents the value in the catalog entry from being reset to the value GUARDS=*NONE

If STATE=*UPDATE is specified, then: if PROTECT is not specified, *UNCHANGED has the same effect as no specification.

If STATE=*NEW is specified, then: *UNCHANGED has the same effect as GUARDS=*NONE (irrespective of what is specified for PROTECT).

IOPERF

Only for files and file generations on public volumes and files on Net-Storage volumes:
Requested performance attribute of the file for I/O processing. There are three performance attributes: “*VERY-HIGH”, “*STD”, and “*HIGH”. The highest permissible value depends on the user ID.

If a performance higher than the maximum permitted value is required, the maximum value from the catalog entry is taken over (no return code).

The operand is ignored (no return code!) if:

- it is specified for files which are not located on a public volume or which are to be created.
- the file identifier on the SM pubset concerned has a default storage class and physical allocation is forbidden.

A storage class entered in the file catalog entry in an SM pubset is removed if this operand is specified. The operand must not be specified simultaneously with the operand STOCLASnot equal to *NONE.

Default setting (only in conjunction with STATE=*NEW): IOPERF=*STD

= *STD
The file should not be processed via a cache.

= *HIGH
The file has a high performance priority and should therefore be processed via a cache if possible.

= *VERY-HIGH
The file has a very high performance priority, so all pages of the file should be maintained in cache if possible.

= *USER-MAX
The file is processed in accordance with the highest performance attribute permitted for the user ID.
This entry ensures that the maximum value is always used without program changes even if the value range is extended above *VERY-HIGH.

IOUSAGE

Only for files and file generations on public volumes and files on Net-Storage volumes:
Specifies the I/O operations to which the performance attribute of the file (operand IOPERF) applies.

The operand is ignored (no return code!) if:

- it is specified for files which are not located on a public volume or which are to be created.
- the file identifier on the SM pubset concerned has a default storage class and physical allocation is forbidden.

A storage class entered in the file catalog entry in an SM pubset is removed if this operand is specified. The operand must not be specified simultaneously with the operand STOCLASnot equal to *NONE.

Default setting (only in conjunction with STATE=*NEW): IOUSAGE=*RDWRT

= *RDWRT
The performance attribute applies to read and write operations.

= *WRITE
The performance attribute applies to write operations only.

= *READ
The performance attribute applies to read operations only.

LARGE

Only for files and FGGs on disk:
similar to BACKUP, LARGE refers to the saving of files with ARCHIVE or HSMS. It specifies whether the complete file (or generations) is to be saved or only the blocks which have been changed since the last save operation.

Default setting (only in conjunction with STATE=*NEW): LARGE=*NO

= *NO
A complete backup is to be made.

= *YES
A partial backup is to be made (only blocks which have been changed); this setting is useful for large files.

LAST

Only for file generation groups:
This operand may only be specified in conjunction with STATE=*NEW and the FIRST operand. It defines the absolute generation number of the most-recently cataloged (i.e. youngest) file generation and is required when reconstructing the index entry of a file generation group on private volumes. The LAST operand determines the number of the most recent file generation to be imported.

= <integer 1..9999>
The file generation specified by LAST can only be imported, not recataloged. This means that if the created index entry is not to be used for the reconstruction of a file generation group, there is no way of actually creating the file generation group specified here (or any other file generations >= FIRST; see also the description of the FIRST operand).

= (<reg: int:2>)
Only possible with MF=M:
The specified register contains the generation number of the latest cataloged file generation in the lower half-word.

= <var: char:2>
Only possible with MF=M:
Symbolic address of a half-word containing the generation number of the latest cataloged file generation

LIST

Defines whether a log is to be written to SYSOUT for the processed file name.

= *NO
The file is not to be logged.

= *SYSOUT
Each processed file name and any errors are logged in a report.

= *ERRORS-TO-SYSOUT
Only file names whose processing leads to errors are logged in a report.

MACID

Only evaluated in conjunction with MF=C/D/M; defines the second and third characters of the field names and equates which are generated during macro expansion in the data area.

Default: MACID = DK

= <macid>
One- or two-character string defining the second and third characters of the generated field names and equates.

MANCLAS

Only for permanent files and file generation groups on public volumes in SM pubsets and for files on the Net-Storage of an SM pubset if the software product HSMS is loaded:
Specifies whether data backup and migration are to be controlled via a management class (see the manual “HSMS” [10] for further details).

= *NONE
A management class is not assigned. Only the corresponding operand specifications are relevant for file backup and migration.

= <c-string: structured-name 1..8>
File backup and migration are controlled via the specified management class.The management class must exist and the user must posses the right to use it.The entry is ignored for files in SM pubsets or on private volumes and rejected for temporary files.

= (<reg: A(char:8)>)
Only possible with MF=M:
The specified register contains the address of an 8-byte memory area containing the name of the management class.

= <var: char:8>
Only possible with MF=M:
Symbolic address of an 8-byte memory area containing the name of the management class.

MF

The forms of the MF operand are described in detail in the appendix ("Macro types"). In all macros differentiated by MF operands (MF=L/E/D/C), the version operand must contain the same value.

MIGRATE

Only for files on public disks and for files on Net-Storage:
is interpreted by means of the software product HSMS (Hierarchical Storage Management System).
Using MIGRATE, the user can specify whether files which he/she has not accessed for some time may be migrated to a storage level with slower access. The files are migrated from online processing level S0 to online background level S1 or offline background level S2 (e.g. tape). For further details see the “HSMS” manual [10].

File generation groups:
the MIGRATE value specified for an FGG index is representative of the whole group.

Default value:
MIGRATE = *ALLOWED for permanent files
MIGRATE = *INHIBITED for temporary files

= *ALLOWED
The file may be migrated from S0 to storage level S1 or S2.

= *INHIBITED
The file is not to be migrated but may, however, be temporarily stored to a background level, e.g. for reorganization purposes.

= *FORBIDDEN
Only for users with the right for physical allocation:
The file must not be migrated to a background level.
This entry is rejected if the file is migrated to a background level or if no authorization for physical allocation exists.

NETCCS

Only for files of the node file type on Net-Storage. The specification is ignored for other files.
Specifies which Net-Storage coded character set (NETCCS) is to be used for a node file on Net-Storage.
The NETCCS is the actual character set with which the node file is created on the Net-Storage. All SAM node files and PAM node files created with BS2000 OSD/BC V11.0 or later have a defined NETCCS. PAM node files created with an older version than V11.0 are treated as if they were created with *NO-CONV.

= *USER_DEF
This is set according to the definition in the user entry. The resulting NETCCS of the file is identified based on the following table:

CCS entry [1]	NETCCS entry [1]	Resulting NETCCS in the catalog entry of the node file
EDF03IRV/*NONE	*ISO	ISO88591; during code conversion, EDF041 is assumed for CCS
EDF03DRV	*ISO	ISO88591; during code conversion, EDF04DRV is assumed for CCS
EDF04DRV	*ISO	ISO88591
EDF04x	*ISO	ISO8859x with x=1,2,..F
ISO8859x	*ISO or *NO-CONV	ISOx
UTFx	*ISO or *NO-CONV	UTFx
<name_a 1..8>	<name_b 1..8>	<name_b 1..8>
<name_a 1..8>	*NO-CONV	<name_a 1..8>

[1] User entry (SYSSRPM) or CATALOG or CREATE-FILE or MODIFY-FILE-ATTRIBUTES entry

= *ISO
For SAM node files, EBCDIC character sets are converted into character sets common in the open world. During this process, NETCCS is mirrored to an ISO variant that corresponds to the coded character set (CSSN). If an ISO or UTF character set has been specified under CODED-CHARACTER-SET, no conversion takes place. In this case, *ISO acts like *NO-CONV. Examples see table under *USER-DEF.

= *NO-CONVERSION
The character set is not converted.

= <name 1..8>
Name of the NETCCS with which the node file is created on the Net-Storage.

> There is no check whether the name composes of a valid character set!

= (<reg: A(char:8)>)

Only possible with MF=M:
The specified register contains the address of a storage area of 8 bytes, in which the name of the NETCCS is stored.

= <var: char:8>
Only possible with MF=M: Symbolic address of a storage area of 8 bytes, in which the name of the NETCCS is stored.

NEWNAME

*Only permitted with STATE=*UPDATE:*
The file specified in the FILE operand is renamed to this name. It is thereby not possible to change either the pubset (catalog) or the user ID. A file on a Net-Storage volume cannot be renamed as a temporary file or as a file generation. See also the sections "Temporary files" and "File generation groups (FGG)".

= <c-string 1..80: filename 1..54 with-wild(80)>
The path name consists of [:catid$_2$:][$userid$_2$.]<filename$_2$>.

catid $_2$
ID of the catalog containing the file catalog entry. If an entry is made here, it must beidentical to the catid$_1$ specified in FILE.

userid $_2$
User ID under which the file is stored. If an entry is made here, it must be identical to the userid$_1$ specified in FILE.

filename $_2$
File name to which filename$_1$, which is specified in the FILE operand, is to be renamed. A suitable construction can be specified if filename$_1$ contains wildcards. If filename$_1$ ends with a period.
filename$_2$ may also end with a period.filename$_2$ must be specified if a file/FGG is to be renamed. filename$_1$ and filename$_2$ must be different.

In the case of tape files, filename$_1$ must differ from filename$_2$ by the added or modified version designation.

If HSMS is used then files that were migrated to storage level S1 or S2 cannot be renamed.

= (<reg: A(char:80)>)
Only possible with MF=M:
The specified register contains the address of an 80-byte memory area containing the new path name. If the path name is shorter than the maximum length of 80 bytes, it must be terminated with at least one blank (X'40').

= <var: char:80>
Only possible with MF=M:
Symbolic address of an 80-byte memory area containing the new path name. If the path name is shorter than the maximum length of 80 bytes, it must be terminated with at least one blank (X'40').

NUM_OF_BACKUP_VERS

Maximum number of file versions that the file takes part in the version backup with.

For files on public volumes and for files on Net-Storage:
If the file is being newly created with STATE=*NEW, the value of the NUMBACK system parameter is taken over. If the attributes of an existing file are being changed (STATE=*UPDATE) and the operand is not specified, the value of the file attribute NUM-OF-BACKUP-VERS remains unchanged.

For files on private disk, for FGG in general and for temporary files:
NUM_OF_BACKUP_VERS may not have a value > 0. If the operand for these objects is not specified, the NUM-OF-BACKUP-VERS file attribute is set to the value 0.

For tape files:
If the file is being newly created with STATE=*NEW, the file attribute NUM-OF-BACKUP-VERS is alway set to the value 0. This is independent of the value of the operand NUM_OF_BACKUP_VERS or the value of the system parameter NUMBACK. If the attributes of an existing file are being changed (STATE=*UPDATE), the file attribute NUM-OF-BACKUP-VERS remains set to the value 0, regardless of whether the operand NUM_OF_BACKUP_VERS is specified or not.

= <integer 0..32>
Maximum number of file versions that the file takes part in the version backup with. If the value is 0, the file /FGG is not part of the version backup.

= (<reg: int:1>)
Only possible with MF=M:
In the lower half-word, the specified register includes the maximum number of file versions of the file/FGG to be stored in the version backup archive.

= <var: int:1>
Only possible with MF=M:
Symbolic address of a half-word in which the maximum number of file versions of the file to be saved in the version backup archive is stored.

OPNBACK

Is provided specially for database files (UDS files) and permits the user to back up the file with ARCHIVE (see the “ARCHIVE” manual [9]) while it is still open. This may lead to inconsistencies in the file; it is the user's responsibility to avoid this.

Default setting (only in conjunction with STATE=*NEW): OPNBACK=*NO

= *NO
Only the closed file is saved.

= *YES
The file may be backed up while it is open.

OTHERAR

Only for files on public volumes and for files on Net-Storage:
Activates access control via the BASIC-ACL and specifies how a user who is neither the file owner nor belongs to the same user group as the file owner may access the file if no GUARDS protection is active.
User groups can be defined in a system only if the software product SECOS is installed (see the “SECOS” manual [8]).
In a system without user groups and in which SECOS is not installed, the value for OTHERAR applies to all user IDs except the file owner.

The operand must not be specified together with the BASACL operand.

= *NO-ACCESS
No access to the file is permitted for the user group.

= ([READ = *NO / READ = *YES / R = *N / R = *Y]
[,WRITE = *NO / WRITE = *YES / W = *N / W = *Y]
[,EXEC = *NO / EXEC = *YES / X= *N / X = *Y])
The access types for which *YES or *Y is specified in the list are permitted. The parentheses are component parts of the access list and must be specified. The various elements of the access list have the following meanings:

READ=NO or R=N	Read access is forbidden (default value).
READ=YES or R=Y	Read access is permitted. In contrast to access control via the ACCESS operand, this does *not* automatically imply the right to execute the file.
WRITE=NO or W=N	Write access is forbidden (default value).
WRITE=YES or W=Y	Write access is permitted. In contrast to access control via the ACCESS operand, this does *not* automatically imply the right to read or execute the file.
EXEC=NO or X=N	Execution of the file is forbidden (default value).
EXEC=YES or X=Y	Execution of the file is permitted (not for file generation groups).

OWNERAR

Only for files on public volumes and for files on Net-Storage:
Activates access control via the BASIC-ACL and specifies how the file owner (and the system administrator) may access the file if no GUARDS protection is active.

The operand must not be specified together with the BASACL operand.

= *NO-ACCESS
No access to the file is permitted for the user group.

= ([READ = *NO / READ = *YES / R = *N / R = *Y]
[,WRITE = *NO / WRITE = *YES / W = *N / W = *Y]
[,EXEC = *NO / EXEC = *YES / X= *N / X = *Y])
The access types specified with *YES or *Y in the list are permitted. The parentheses are component parts of the access list and must be specified.

The various elements of the access list have the following meanings:

READ=NO or R=N	Read access is forbidden (default value).
READ=YES or R=Y	Read access is permitted. In contrast to access control via the ACCESS operand, this does *not* automatically imply the right to execute the file.
WRITE=NO or W=N	Write access is forbidden (default value).
WRITE=YES or W=Y	Write access is permitted. In contrast to access control via the ACCESS operand, this does *not* automatically imply the right to read or execute the file.
EXEC=NO or X=N	Execution of the file is forbidden (default value).
EXEC=YES or X=Y	Execution of the file is permitted (not for file generation groups).

PARAM

Designates the address of the operand list and is only evaluated in conjunction with MF=E (see also "Macro types").

= <name 1..8>
Symbolic address (name) of the operand list.

PREFIX

Evaluated only in conjunction with MF=C/D/M. Defines the first character of each field name and equate generated in the data area when the macro is expanded.

= I
Default prefix with which the field names and equates generated by the assembler begin.

= pre
Single-character prefix with which the field names and equates generated by the assembler are to begin.

= *
No prefix is generated.

PROTECT

Defines where the protection attribute is to be taken over from, that cannot be explicitly defined with the respective operand.
The following protection attributes (operands) can be assigned with PROTECT (depending on the operand values):

Access protection	Protection attribute	CATAL operand
Standard access control (access type)	ACCESS	ACCESS
Standard access control (access by other users)	USER-ACCESS	SHARE
Basic access control list	BASIC-ACL	BASACL, OWNERAR, GROUPAR, OTHERAR
Access control via GUARDS	GUARDS	GUARDS
Passwords	READ-PASSWORD, WRITE-PASSWORD, EXEC-PASSWORD	RDPASS, WRPASS, EXPASS
Binary deletion	DESTROY-BY-DELETE	DESTROY
Memory space lock	SPACE-RELEASE-LOCK	RELSPAC
Release date for deletion	FREE-FOR-DELETION	DELDATE
Expiration date	EXPIRATION-DATE	EXDATE or RETPD

The values of these protection attributes may be preset differently depending on the value of the STATE operand (NEW or UPDATE); (see tables).

Protection attributes when cataloging new files

PROTECTION-ATTR= (Protection attribute)	***FROM-FILE**	***STD** [1]	***BY-DEF-PROT-OR-STD** [1] **(Def Prot. not active)**	***BY-DEF-PROT-OR-STD** [1] **(Default Protection active)**
ACCESS	Value transferred from reference file	WRITE	WRITE	Value supplied by default protection
USER-ACCESS		OWNER-ONLY	OWNER-ONLY	
BASIC-ACL		NONE	NONE	
DESTROY-BY-DELETE		NO	NO	
GUARDS		NONE	NONE	
SPACE-RELEASE-LOCK		NO	NO	
READ-PASSWORD	NONE	NONE	NONE	
WRITE-PASSWORD	NONE	NONE	NONE	
EXEC-PASSWORD	NONE	NONE	NONE	
FREE-FOR-DELETION	NONE	NONE	NONE	NONE
AUDIT	NONE	NONE	NONE	NONE

1) System default values

No expiration date (EXPIRATION-DATE) can be defined for the first entry. In the case of files, it is implicitly preset to *NONE, and in the case of file generation groups to *TODAY.

Protection attributes when changing file attributes

PROTECTION-ATTR= (Protection attribute)	***UNCH**	***FROM-FILE**	***STD** [1]	***BY-DEF-PROT-OR-STD** [1] **(Def Prot. not active)**	***BY-DEF-PROT-OR-STD** [1] **(Default Protection active)**
ACCESS	UNCHANGED	Value transferred from reference file	WRITE	WRITE	Value supplied by default protection
USER-ACCESS	UNCHANGED		OWNER-ONLY	OWNER-ONLY	
BASIC-ACL	UNCHANGED		NONE	NONE	
DESTROY-BY-DELETE	UNCHANGED		NO	NO	
GUARDS	UNCHANGED		NONE	NONE	
SPACE-RELEASE-LOCK	UNCHANGED		NO	NO	
EXPIRATION-DATE [2]	UNCHANGED		TODAY	TODAY	
READ-PASSWORD	UNCHANGED	UNCHANGED	UNCHANGED	NONE	
WRITE-PASSWORD	UNCHANGED	UNCHANGED	UNCHANGED	NONE	
EXEC-PASSWORD	UNCHANGED	UNCHANGED	UNCHANGED	NONE	
FREE-FOR-DELETION	UNCHANGED	UNCHANGED	UNCHANGED	NONE	
AUDIT	UNCHANGED	UNCHANGED	UNCHANGED	UNCHANGED	UNCHANGED

1) System default values

2) The expiration date is only entered for permanent files with creation dates or for file generation groups. If the referende file has no expiration date, *TODAY is entered.

= *STD

The following system defaults are set:

ACCESS	= *WRITE
BASIC-ACL	= *NONE
USER-ACCESS	= *OWNER-ONLY (also for tape files)
DESTROY	= *NO
SPACE-RELEASE-LOCK	= *NO
GUARDS	= *NONE
EXPIRATION-DATE	= *TODAY (only for permanent files with a creation date and for file generation groups)

PROTECT=*STD is rejected for individual file generations.

= *BY_DEF_PROT_OR_STD

The protection attributes are assigned depending on the use of the “default protection” function.
If default protection is activated, it provides values for all the protection attributes listed above, unless these have been specified explicitly. If default protection has not been activated, the protection attributes are entered as for PROTECT=*STD. In addition, the following system default values apply:

FREE-FOR-DELETION	= *NONE
READ-PASSWORD	= *NONE
WRITE-PASSWORD	= *NONE
EXEC-PASSWORD	= *NONE

If STATE=*NEW is specified, PROTECT=*BY_DEF_PROT_OR_STD has the same effect as no specification.

= (*FROM_FILE,<c-string: filename 1..54>)

All the protection attributes listed for *STD that are not explicitly specified by the caller are imported from the reference file. The protection attributes taken over from the reference file are treated as if they had been specified explicitly.

Exceptions:

- In the case of a file generation group, the execution rights are ignored rather than rejected.
- In the case of temporary files, EXDATE is ignored rather than rejected.

If the reference file does not have an expiration date, EXDATE=*TODAY is used for file generation groups and for permanent files with a creation date.

Passwords and the release date of the reference file are not copied: if CATAL STATE=*NEW applies, they have the system default value *NONE (see also "Protection attributes when cataloging new files"), if STATE=*UPDATE applies, they have the value *UNCHANGED (see the table "Protection attributes when changing file attributes").

The reference file must be located on the same pubset as the file specified by FILE. If no catalog ID is specified then the user ID's default catalog is assumed. For this reason it is always necessary to specify the catalog ID if file does not refer to the default ID.

= (*FROM_FILE,(<reg: A(char:54)>))
Only possible with MF=M:
The specified register contains the address of a 54-byte memory area containing the path name. If the path name is less than the maximum of 54 bytes long, it must be terminated with at least one blank (X'40').

= (*FROM_FILE,<var: char:54>)
Only possible with MF=M:
Symbolic address of a 54-byte memory area containing the path name. If the path name is less than the maximum of 54 bytes long, it must be terminated with at least one blank (X'40').

RDPASS

This operand is used to define, modify or delete a read password.

Temporary files: Password protection is not possible.
Tape files: The password is recorded in the HDR3 label.
For encrypted files: All RDPASS specifications are handled like *UNCHANGED

Default setting (only in conjunction with STATE=*NEW): RDPASS=*NONE

= *NONE
No read password is assigned or an existing read password is deleted.

= *UNCHANGED
If STATE=*UPDATE is specified at the same time, the value of RDPASS remains unchanged. If STATE=*NEW is specified at the same time, the value RDPASS=*NONE is entered.
If PROTECT=*BY_DEF_PROT_OR_STD or STATE=*NEW is specified without a value for PROTECT, *UNCHANGED prevents the corresponding value supplied by default protection from being taken over.

If STATE=*UPDATE is specified, then: if PROTECT=*BY_DEF_PROT_OR_STD is not specified, *UNCHANGED has the same effect as no specification.

if STATE=*NEW is specified, then: *UNCHANGED has the same effect as the specification *NONE (irrespective of what is specified for PROTECT).

= <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
Defines a password required for read access.

If a program is protected by a read password, this also applies to the load module in main memory. The LOAD-PROGRAM command is rejected, as are the IDA commands DISPLAY and AT. If a source program is protected by a read password, it cannot be assembled or compiled.

= (<reg: A(char:4)>)
Only possible with MF=M:
The specified register contains the address of a 4-byte memory area containing the read password.

= <var: char:4>

Only possible with MF=M:

Symbolic address of a 4-byte memory area containing the read password.

RELSPAC

Specifies whether or not storage space may be released by using the MODIFY-FILE-ATTRIBUTES command or the FILE macro.

Default setting (only in conjunction with STATE=*NEW:) RELSPAC=*ALLOWED

= *ALLOWED
The storage space may be released.

= *IGNORED
The request to release space is ignored.

= *UNCHANGED
Only relevant in conjunction with specification of PROTECT:
If STATE=*UPDATE is specified at the same time, the value of RELSPAC remains unchanged. If STATE=*NEW is specified at the same time, the value RELSPAC=*ALLOWED is entered.

The value *UNCHANGED has the following effects:

- if PROTECT=*FROM-FILE is specified: prevents the corresponding value from being taken over from the reference file
- if PROTECT=*BY_DEF_PROT_OR_STD is specified and STATE=*NEW is specified without a value for PROTECT:prevents the corresponding value supplied by default protection from being taken over
- if PROTECT=*STD and STATE=*UPDATE are specified together:prevents the value in the catalog entry from being reset to the value RELSPAC=*ALLOWED

If STATE=*UPDATE is specified, then: if PROTECT is not specified, *UNCHANGED has the same effect as no specification.

If STATE=*NEW is specified, then: *UNCHANGED has the same effect as RELSPAC=*ALLOWED (irrespective of what is specified for PROTECT).

RETPD

Defines an EXPIRATION DATE before which the file must not be updated or erased, i.e. during which it can only be read.

Default value (only if STATE=*NEW): RETPD = 0, i.e. the file or generation can be updated or erased at any time.

An expiration date can only be defined for an existing file, i.e. the catalog fields CRE-DATE and FILE-STRUC must contain a value not equal to NONE. This also means that RETPD cannot be specified together with the CATAL operand STATE=*NEW or STATE=*FOREIGN (RETPD is ignored).

The expiration date is calculated from the number of specified days and is always based on local time with the date and a time of 00:00:00.

The expiration date can be canceled or modified by means of a further CATAL macro containing the RETPD operand. Once the expiration date has elapsed, write access is again possible.

An existing generation of an FGG can be erased when creating a new generation even if the expiration date has not expired (see the DISP operand for details).

Simultaneous use of the EXDATE and RETPD operands is not possible.

If the RETPD operand is specified, the EXDATE value returned by the default protection function is ignored.

= <integer 0..32767>
Number of days for which the file is to be protected.

= (<reg: int:2>)
Only possible with MF=M:
The specified register contains the number days for which the file is to be protected in the lower half-word.

= <var: char:2>
Only possible with MF=M:
Symbolic address of a half-word containing the number days for which the file is to be protected.

S0MIGR

*Only relevant with STATE=*UPDATE for files in SM pubsets that already occupy storage:*
Defines whether the file within the SM pubset (storage hierarchy level S0) may be reallocated to another volume set.

A storage class entered in the file catalog entry in an SM pubset is removed if this operand is specified. The operand must not be specified simultaneously with the operand STOCLAS not equal to *NONE.

=*ALLOWED

The file may be reallocated within the SM pubset.

The value is ignored for files and generation on volume sets with permanent data storage if the file identifier on the SM pubset concerned has a default storage class and physical allocation is forbidden.

= *FORBIDDEN
Only for users with the right for physical allocation:
Automatic reallocation is not permitted. The file is to remain on the volume set to which it is currently allocated. This entry is rejected in the following cases:

- No authorization for physical allocation exists.
- The file is on an SM pubset but does not occupy storage.
- The file is cataloged on an SM pubset, but resides on a Net-Storage volume.

SHARE

Specifies whether the file or file generation may be processed under a user ID other than that of the owner if no BASIC-ACL or GUARDS protection is active. The type of access which is permitted is determined by the other file protection attributes (see the operands ACCESS, WRPASS etc.).

Default setting (only in conjunction with STATE=*NEW): SHARE=*NO for disk files, SHARE=*YES for tape files

Tape files:
when the file is opened for the first time, DMS writes the SHARE indicator in the HDR1 label (“access indicator”).

= *NO
The file is not shareable.
Tape files, HDR1 label: access indicator = 1

= *YES
File access is permitted for any user ID, i.e. the file or generation is shareable.
Temporary files, SHARE=*YES is not permitted
Tape files, HDR1 label: access indicator = X'40'

= *SPECIAL
File access is permitted for the user ID with `HW-MAINTENANCE` privileges.
SHARE=*YES is implied.
This value must not be specified for file generation groups.

= *UNCHANGED
Only relevant in conjunction with specification of PROTECT:
If STATE=*UPDATE is specified at the same time, the value of SHARE remains unchanged. If STATE=*NEW is specified at the same time, the value SHARE=*NO is entered.

The value *UNCHANGED has the following effects:

- if PROTECT=*FROM-FILE is specified:
  prevents the corresponding value from being taken over from the reference file
- if PROTECT=*BY_DEF_PROT_OR_STD is specified and STATE=*NEW is specified without a value for PROTECT:
  prevents the corresponding value supplied by default protection from being taken over
- if PROTECT=*STD and STATE=*UPDATE are specified at the same time:prevents the value in the catalog entry from being reset to the value SHARE=*NO

If STATE=*UPDATE is specified, then: if PROTECT is not specified, *UNCHANGED has the same effect as no specification.

If STATE=*NEW is specified, then: *UNCHANGED has the same effect as SHARE=*NO (irrespective of what is specified for PROTECT).

STATE

Specifies whether a new catalog entry is to be created, an existing catalog entry is to be updated or a catalog entry is to be imported.

= *NEW
A new catalog entry is to be created.

= *UPDATE
An existing catalog entry (FILE=...) is to be updated. STATE=*UPDATE must be specified for each access to an existing catalog entry. The attributes whose associated operands are specified in this CATAL macro are updated. If password protection exists for the file, the write password must be entered in the password table of the job by means of a PASSWORD command before the catalog entry can be updated.

= *FOREIGN
Only for exported FGGs on private disks:
A group entry (that is only in the F1 label of a private disk) of a file generation group stored on private disks is to be imported. The VOLUME and DEVICE operands must be specified with this operand; all other operands are ignored or rejected. The generations to be transferred must then be imported individually or collectively by using the FILE macro (operands STATE=*FOREIGN, DEVICE and VOLUME) or IMPORT, respectively.

STOCLAS

The operand is only relevant for files, file generations and file generation groups on public volumes in SM pubsets and files on a Net-Storage volume which are cataloged on an SM pubset.

It defines whether the data storage location selection (volume set) within the SM pubset is to be controlled via a storage class for files and file generations with STATE=*UPDATE for which storage has been allocated.
A storage class must not be assigned (i.e. by specifying the operand STOCLAS not equal to *NONE) simultaneously with assignment of the separate attributes it contains (AVAIL, DISKWR, IOPERF or IOUSAGE operand) or together with the S0MIGR operand.
The request is also rejected if the file does not occupy storage space or if the storage class contains the attribute AVAILABILITY=*HIGH and the file is currently migrated to a background level (S1 or S2 migration with HSMS).

The operand defines the default storage class for file generation groups, which is used for the first storage assignment to a generation if no explicit storage class or one of the separate attributes is specified.

With STATE=*NEW, a storage class must not be assigned (i.e. by specifying the STOCLAS*NONE operand) together with the WORKGRP operand.
With STATE=*UPDATE, it is only possible to assign a storage class whose WORK-FILE attribute matches the WORK-FILE attribute of the file generation group.
A file on a Net-Storage volume can be assigned a storage class. In this case no work file and no file with the preliminary file format K can be created.

Note

Specifying a value which is not equal to *NONE for the STOCLAS operand can result in the file being displaced (reallocated) from its current volume set to another volume set which suits the storage class better. The following cases can occur here:

- If the storage class contains AVAILABILITY=*HIGH and AVAILABILITY=*STD applies for the current volume set, the file must be reallocated to a colume set with the attribute AVAILABILITY=*HIGH. If reallocation is not possible, the CATAL call is rejected.
- If the storage class contains a volume set list and the file is not located on a volume set in the volume set list, the file will, if possible, be reallocated to a volume set from the list. If reallocation is not possible, the CATAL call is executed without reallocation taking place.

The file is locked (opened) during this reallocation, i.e. all accesses to the file or its catalog entry are rejected instead of being put into a wait state.

= *NONE
No storage class is assigned. The corresponding separate attributes are evaluated for selecting the storage location.

The value is ignored for files and generation on volume sets with permanent data storage if the file identifier on the SM pubset concerned has a default storage class and physical allocation is forbidden.

= *UPDATE
Only relevant for files that have been assigned a storage class whose attributes have been modified
The file attributes are modified according to the assigned storage class. If the entry is made for file generation groups, the only check made is whether the WORK-FILE attribute of the storage class still matches the WORK-FILE attribute of the file generation group. The request is rejected with a return code if this is not the case.

= *STD
The default storage class of the user ID for the respective pubset is used for files and file generation groups. The default storage class of the file generation group is used for file generations, in other words the storage class that was assigned to the file generation group index.

= <c-string: structured-name 1..8>
The specified storage class defines selection of the file storage location.
The storage class must exist and the user must have the right to use it.
The file attributes are not updated if the file was not already assigned the specified storage class, i.e. intermediate storage class modifications are not effective, see *UPDATE.
The entry is ignored for files in SF pubsets or on private volumes. The entry is rejected with STATE=*NEW and for cataloged files without allocated storage space.

= (<reg: A(char:8)>)
Only possible with MF=M:
The specified register contains the address of an 8-byte memory area containing the name of the storage class.

= <var: char:8>
Only possible with MF=M:
Symbolic address of an 8-byte memory area containing the name of the storage class.

TIMBASE

Defines the basis on which absolute dates specified with the EXDATE and DELDATE operands are to be interpreted (relative dates always refer to local time).

The TIMBASE operand has no effect on dates supplied by the default protection function. These always refer to the local time.

= *UTC
Absolute dates are interpreted based on UTC world time (universal time coordinate).

= *LTI
All dates are interpreted based on LTI (local time).

USRINFO

Enters user metainformation into the file catalog entry. The entry can be a maximum of 8 bytes long, with any contents, whose meaning is defined by the user. The operand is ignored for files on private volumes.

= *NONE
No entry or the entry will be deleted.

= <c-string 1..8>
The specified characters are entered.

= (<reg: A(char:8)>)
Only possible with MF=M:
The specified register contains the address of an 8-byte memory area containing the metainformation to be entered.

= <var: char:8>
Only possible with MF=M:
Symbolic address of an 8-byte memory area containing the metainformation to be entered.

VERSION

Specifies which version of the parameter list is to be generated. The latest version should always be used.
The default setting cannot be specified explicitly!

Implicit default setting: VERSION=0

The parameter list format that was supported prior to BS2000 V9.5A is generated, but only for the parameters recognized at the time.
The supported operands and operand values are listed in table "Version differences - VERSION=0/1/2/3/4".

= 1
Generates the parameter list format that was supported in BS2000 V9.5 and V10.0, but only for the parameters recognized at the time.
The supported operands and operand values are listed in table "Version differences - VERSION=0/1/2/3/4".

= 2
Generates the parameter list format for versions BS2000/OSD-BC V1.0 and V2.0.

= 3
The parameter list format for versions BS2000/OSD-BC V3.0 and higher is generated.

= 4
The parameter list format for versions BS2000 OSD-BC V11.0 and higher is generated.

Note

If existing software which manipulates the generated parameter list is to be reassembled, the old format (0, 1 or 2) must be requested. In all other respects, source code compatibility is ensured.

VOLUME

Only for FGGs on private disks:
Specifies the volume serial number (“vsn”) of a private volume (private disk).

The operands VOLUME and DEVICE must be specified when an FGG is created or reconstructed on private disks (STATE=*NEW) or when an FGG kept on private disks is to be imported (STATE=*FOREIGN).

If the software product MAREN is being used, a VOLUME may be specified without a DEVICE.

= <c-string 1..6>
Archive number

= (<reg: A(char:6)>)
Only possible with MF=M:
The specified register contains the address of a 6-byte memory area containing the volume serial number.

= <var: char:6>
Only possible with MF=M:
Symbolic address of a 6-byte memory area containing the volume serial number.

WORKGRP

Only relevant when setting up a file generation group in SM pubsets:
Defines whether the file generation group is to be a permanent or work file generation group.
Work file generation groups can be deleted by system administration at a time specified by system administration.

= *YES
The file generation group is set up as a work file generation group.

WRPASS

The user can define, update or delete a write password with this operand.

Temporary files:
Password protection is not permitted.

Tape files:
The password is stored in the HDR3 label.

Default setting – only in conjunction with STATE=*NEW: WRPASS=*NONE

= *NONE
No password is defined or an existing password is deleted.

= *UNCHANGED
If STATE=*UPDATE is specified at the same time, the value of WRPASS remains unchanged. If STATE=*NEW is specified at the same time, the value WRPASS=*NONE is entered.

If PROTECT=*BY_DEF_PROT_OR_STD or STATE=*NEW is specified without a value for PROTECT, *UNCHANGED prevents the corresponding value supplied by default protection from being taken over.

If STATE=*UPDATE is specified, then: if PROTECT=*BY_DEF_PROT_OR_STD is not specified, *UNCHANGED has the same effect as no specification.

If STATE=*NEW is specified, then: *UNCHANGED has the same effect as the specification *NONE (irrespective of what is specified for PROTECT).

= <c-string 1..4> / <x-string 1..8> / <integer -2147483648..2147483647>
Defines the password needed for write access.

= (<reg: A(char:4)>)
Only possible with MF=M:
The specified register contains the address of a 4-byte memory area containing the write password.

= <var: char:4>
Only possible with MF=M:
Symbolic address of a 4-byte memory area containing the write password.

Programming notes

1. Calling the CATAL macro with the new operand list:
   label CATAL <operands,...>,VERSION=3
2. Register 1 Address of the operand list.
3. The error code is only returned in the standard header of the parameter list (IDKRET field) and no longer in general-purpose register 15 as in version 2.

Return codes

Standard header: `ccbbaaaa`

The following code relating to execution of the CATAL macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	No error
X'01'	X'00'	X'0000'	Only with check dialogs: Request was fully or partially retracted in dialog, i.e. at least one check dialog was answered with *NO.
X'02'	X'00'	X'0000'	Only in conjunction with CHECK*NO: An error has occurred but continuation of the function was requested in an error dialog.
	X'40'	X'0501'	Requested catalog not available
	X'82'	X'0502'	Requested catalog in the wait state
	X'40'	X'0503'	Incorrect information in MRSCAT
	X'82'	X'0504'	Error in catalog management system
	X'40'	X'0505'	Error during computer communication (MRS)
	X'80'	X'0506'	Operation canceled because of master switch
	X'01'	X'0509'	Specified number of versions for version backup invalid
	X'40'	X'0510'	Error when calling an internal function
	X'40'	X'0512'	Requested catalog unknown
	X'40'	X'0513'	Call rejected by system exit routine
	X'40'	X'051B'	User ID not known in specified pubset
	X'40'	X'051C'	No access right to specified pubset
	X'40'	X'051D'	LOGON password to specified pubset is different
	X'20'	X'0527'	I/O error while reallocating the data in an SM pubset

X'20'	X'0530'	CMS reports error during a request for storage space
X'20'	X'0531'	Unexpected error during catalog access
X'82'	X'0532'	File locked because it is in use
X'82'	X'0534'	Private volume cannot be assigned
X'40'	X'0535'	No access right to the file catalog entry (only in conjunction with CCS or NETCSS assignment to a foreign user ID)
X'20'	X'0536'	Error in file management system
X'40'	X'053A'	Error while modifying the F1 label on a private disk
X'20'	X'053B'	System error during file access
X'82'	X'053C'	Catalog file of the pubset is full
X'40'	X'053D'	Catalog of F1 label block is destroyed
X'40'	X'053E'	File on private volume is already cataloged
X'82'	X'053F'	File is reserved by another task
X'40'	X'0540'	No volume set that matches the required file attributes is available in the specified pubset
X'82'	X'0541'	Data reallocation is not possible because there is no suitable volume set with enough free storage space
X'40'	X'0546'	File catalog entry is full
X'82'	X'054D'	Storage allocation exceeded
X'20'	X'054F'	Unexpected error while accessing JOIN file
X'40'	X'0555'	STATE=*FOREIGN: specified file already exists in the catalog of the user
X'82'	X'055A'	Device currently reserved
X'40'	X'055C'	Catalog entry on private disk not found
X'40'	X'055D'	User has no right for physical allocation
X'40'	X'055F'	Volume could not be reserved
X'01'	X'0576'	Contradictory operand combination or reserved parameter area fields used
X'20'	X'0577'	Internal error while accessing job environment
X'20'	X'0578'	Internal error while checking access rights
X'01'	X'0579'	Invalid operand specified for temporary file
X'40'	X'057A'	Attribute cannot be assigned for work file

X'40'	X'057E'	HSMS not available
X'40'	X'057F'	File is migrated, renaming not possible
X'01'	X'0590'	Volume specification not permitted without device specification
X'82'	X'0594'	Insufficient virtual memory available (also if wildcards are used and too many files are selected)
X'01'	X'0599'	Operand is not supported in the RFA-BS version
X'40'	X'05A0'	Updating the performance attributes (DISKWR, IOPERF, IOUSAGE) is not permitted if data in the write cache has not been written yet
X'01'	X'05A8'	Requested device type not found in system
X'40'	X'05AD'	Only when renaming with simultaneous S0 migration: File attributes were modified but the file could not be renamed because of CMS problems
X'82'	X'05B0'	No suitable device is currently available
X'40'	X'05B4'	Only in conjunction with VOLUME/DEVICE: A MOUNT message for the requested volume was answered with 'NO' by the operator
X'40'	X'05B5'	Guard not available
X'40'	X'05BD'	Illegal combination of file and volume set attributes
X'20'	X'05C7'	Internal error in DMS
X'82'	X'05C8'	Maximum number of files reached for user ID
X'20'	X'05CA'	Internal error while modifying the CE allocation
X'01'	X'05CB'	Incorrect or illegal first file name
X'40'	X'05CC'	File name already cataloged
X'01'	X'05CD'	Incorrect or illegal new file name
X'40'	X'05CE'	First file name not cataloged
X'40'	X'05CF'	File is protected with a password
X'82'	X'05D0'	File locked because it is in use
X'40'	X'05D1'	Error while requesting a device
X'40'	X'05D2'	EXPIRATION-DATE was specified for an empty file
X'01'	X'05D3'	GUARDS name incorrect
X'40'	X'05D4'	GUARDS catalog must not be protected by a guard
X'01'	X'05E8'	File name illegal for disk file

X'01'	X'05EE'	File name too long
X'01'	X'05EF'	BASIC-ACL or guard cannot be assigned
X'01'	X'05FA'	Access to REMOTE-IMPORTED pubset not possible
X'40'	X'05FC'	Specified user ID not in home pubset
X'40'	X'05FD'	File is write-protected with USER-ACCESS or EXPIRATION-DATE (only for CCS or NETCSS assignment to foreign user ID)
X'40'	X'0606'	Volume request rejected by MAREN
X'40'	X'0609'	Action not permitted for system file
X'40'	X'060D'	Error while reading reference file attributes (PROTECT operand) • Syntax error in file name, possibly also in conjunction with an ACS replacement • Specified reference file not accessible
X'40'	X'0610'	Function execution supplies a return code for at least one of the selected file names
X'01'	X'0611'	Incorrectly specified construction (NEWNAME operand with wildcards)
X'40'	X'0613'	Unknown management class
X'40'	X'0614'	No access right for management class
X'40'	X'0616'	Specified attributes require an S0 migration, but the file is locked against reallocation
X'40'	X'0618'	Unknown storage class
X'40'	X'0619'	No access right for storage class
X'40'	X'0640	Access to Net-Storage is rejected by the ONETSTOR subsystem because of communication problems with the net client
X'40'	X'0643'	Net client reports access error
X'40'	X'0644'	Net client reports internal error
X'40'	X'0645'	File does not exist on Net-Storage
X'40'	X'0646'	FGG not permitted on Net-Storage volume
X'40'	X'0649'	Net server reports POSIX ACL error
X'40'	X'064A'	Net client reports that access to files on the Net-Storage volume is forbidden
X'40'	X'064B'	Access to node files from the net client not supported
X'40'	X'0666'	File is write-protected by GUARDS (only with CCS or NETCSS assignment to foreign user ID)

X'40'	X'0685'	File occupies no storage space and AVAIL=*HIGH, a storage class or an S0 migration lock is to be set
X'20'	X'069D'	Incorrect catalog entry structure
X'40'	X'06A6'	AUDIT specification not permitted for user ID
X'00'	X'06A9'	Generations missing from the file generation group
X'40'	X'06B6'	File attributes unsuitable for the file generation group
X'01'	X'06C1'	More than 255 generations requested or conflict with BASE, LAST or FIRST operand
X'01'	X'06C3'	Illegal name for a file generation group
X'40'	X'06C4'	File generation group not cataloged
X'01'	X'06C5'	File generation group name too long
X'01'	X'06C6'	Tape file name or attribute cannot be modified
X'01'	X'06C7'	Invalid generation number specified
X'01'	X'06C8'	Attribute can only be modified for the complete file generation group
X'01'	X'06C9'	Generation-specific operand in incorrect context
X'00'	X'06CA'	Command executed, apart from incorrect BASE specification

X'40'	X'06CC'	Only with wildcard selection: No file matches the specified selection entry
X'40'	X'06CD'	Specified file generation group locked with write protection against extensions
X'01'	X'06CE'	Retention date (RETPD, EXDATE) or delete date (DELDATE) incorrectly specified
X'40'	X'06D5'	Deleting of superfluous file generations is prevented by write protection
X'01'	X'06DA'	Illegal combination of private and public volumes for a file generation group
X'01'	X'06DB'	Incorrect VOLUME and DEVICE specification
X'01'	X'06FA'	New file name only permitted with STATE=*UPDATE
X'01'	X'06FB'	Granting of execution rights not possible for file generation groups
X'01'	X'06FD'	Parameter range invalid or not accessible
X'40'	X'06FF'	BCAM connection interrupted
X'01'	X'FFFF'	Incorrect function number in parameter range header
X'03'	X'FFFF'	Incorrect version number in parameter range header

Version differences - VERSION=0/1/2/3/4

MF=	Operands	without version	Vers=1	Vers=2	Vers=3	Vers=4	Comments
MF=E		(1)	(1)	(1)	x	x	
	VERSION	-	x	x	x	x	
	PARAM	-	-	-	x	x	
MF=D/ MF=C		x	x	x	x	x	
	PREFIX	x	x	x	x	x	
	MACID	-	-	-	x	x	
	VERSION	-	x	x	x	x	
MF=M		-	-	-	x	x	
	PREFIX	-	-	-	x	x	
	MACID	-	-	-	x	x	
	all operands of MF=I/L	-	-	-	x	x	
MF=I/ MF=L		x	x	x	x	x	
	pathname1	(2)	(2)	(2)	-	-	
	pathname2	(3)	(3)	(2)	-	-	
	ACCESS	x	x	x	x		
	ADMINFO	-	-	-	x	x	
	AUDIT	x	x	x	x	x	
	BACKUP	x	x	x	x	x	
	BASACL	-	(x)	x	x	x	
	BASE	x	x	x	x	x	
	CSS	-	-	x	x	x	
	CHECK	-	-	-	x	x	
	DELDATE	-	-	-	x	x	
	DESTROY	x	x	x	x	x	
	DEVICE	x	x	x	x	x	(5)
	DISKWR	-	-	x	x	x	
	DISP	x	x	x	x	x	

EXDATE	-	-	-	x	x	
EXPASS	x	x	x	x	x	
FILE	(2)	(2)	(2)	(4)	(4)	
FIRST	x	x	x	x	x	

GEN	x	x	x	x	x	
GROUPAR	-	(x)	x	x	x	
GUARDS	-	-	x	x	x	
IOPERF	-	-	x	x	x	
IOUSAGE	-	-	x	x	x	
LARGE	x	x	x	x	x	
LAST	-	-	x	x	x	
LIST	-	-	-	x	x	
MANCLAS	-	-	-	x	x	
MIGRATE	-	x	x	x	x	(6)
NETCCS	-	-	-	-	x	
NEWNAME	3	2	2	4	4	
NUM_OF_BACKUP_VERS	-	-	-	-	x	
OPNBACK	-	(x)	x	x	x	
OTHERAR	-	(x)	x	x	x	
OWNERAR	-	(x)	x	x	x	
PROTECT	-	-	-	x	x	
RDPASS	x	x	x	x	x	
RELSPAC	-	-	x	x	x	
RETPD	x	x	x	x	x	
SHARE	x	x	x	x	x	
STATE	x	x	x	x	x	
STOCLAS	-	-	-	x	x	
S0MIGR	-	-	-	x	x	
TIMBASE	-	-	-	x	x	
USRINFO	-	-	-	x	x	
VERSION	-	x	x	x	x	
VOLUME	x	x	x	x	x	
WORKGRP	-	-	-	x	x	
WRPASS	x	x	x	x	x	

Table 9: CATAL - Version differences

Key

x	The operand is available in the macro version.
(x)	Operand is available in the macro version, but not as of the first release.
-	The operand is not available in the macro version.
Vers	Version
(1)	Format MF=(E,<addr>)
(2)	Path name with a maximum of 54 characters , format: [:catid:][$userid.]filename
(3)	File name without catalog ID and user ID (44 characters maximum)
(4)	Selection or construction specifications (analogous to 2), 80 characters maximum
(5)	Only the respectively supported device types
(6)	Vers=1: operand value is called INHIBIT instead of INHIBITED (Vers=2)

In the above table, positional operands are listed before keyword operands under MF=L.

CHKFAR - Check file access rights

Macro type: type S (E form/L form/D form/C form/M form); see "Macro types"

The CHKFAR macro checks the access rights for the file specified in the call and informs the caller which access rights he/she has for this file. The user may select

- whether all access facilities he/she has for the file are to be shown (ignoring any passwords or retention period which may exist) or
- whether he/she desires information on a specific access right (including any password protection or retention period defined for the file).

The information is returned to the caller in an output area of the operand list.

For private files: the CHKFAR macro evaluates information from the user catalog only, not from the F1 label.

For tape files: the CHKFAR macro evaluates information from the user catalog, but not from the header record on the tape.

Format

Operation	Operanden
CHKFAR	**,FILE = 'pathname' / adr1 / (r)** **,ACCESS = *ANY / *READ / *WRITE / *UPDATE / *DELETE / *EXEC / adr2** **MF = L / M**
	MF = E,PARAM = adr / (r)
	MF = D,[,PREFIX = pre]
	MF = C,[,PREFIX = pre][,MACID = macid]

Operand descriptions

FILE

Specifies the file for which the user wishes to determine or check his/her access rights.

= pathname
Name of the file whose access rights are to be checked, where
<c-string 1..54: filename 1..54>

Pathname means [:catid:][$userid.]filename

catid
Catalog ID: if omitted, the default catalog ID for the current user ID is assumed.

userid
User ID: if omitted, the user ID in the SET-LOGON-PARAMETERS or LOGON command is assumed.

filename
A fully qualified file name.

= addr1
Symbolic address (i.e. the name) of a 54-byte field in the user program which contains the path name of the file to be checked.

= (r)
Number of a register which contains the address of the “addr1” field. The register must be loaded with this address value before the macro is called.

ACCESS

Specifies whether all access rights enjoyed by the user for the specified file are to be returned or whether the file is to be checked for a specific access right for the user. The information is returned in an output area of the operand list.

= *ANY
Information on all access rights which the user possesses for the specified file is placed in an output area of the operand list. Any other possible protection attributes of the file, such as passwords or a retention period, are not evaluated.

= *READ
The system checks whether the caller may read the specified file. Any existing password for the file is also taken into account.

= *WRITE
The system checks whether the caller may write to the specified file. Any existing password or retention period for the file is also taken into account.

= *UPDATE
The system checks whether the caller may read from and write to the specified file. Any existing password or retention period for the file is also taken into account.

= *DELETE
The system checks whether the caller may delete the specified file. Any existing password or retention period for the file is also taken into account.

= *EXEC
The system checks whether the caller may execute the specified file. Any existing password for the file is also taken into account.

MACID

Defines the second through fourth characters of each field name and equate generated when the macro is expanded.

Default value: MACID = RMZ

= macid
Three-character string defining the second through fourth characters of the generated field names and equates.

PARAM

Specifies the address of the operand list; it is evaluated only if MF=E applies (see "Macro types").

= addr
Symbolic address (name) of the operand list.

= (r)
Number of the register which contains the address of the operand list. The register must be loaded with this address value before the macro is called.

PREFIX

Specifies the first character of each field name or equate which the assembler generates in the data area when expanding the macro.

Default value: PREFIX=S.

= pre
Single-character prefix with which the field names and equates generated by the assembler are to begin.

Return codes

Standard header: `ccbbaaaa`

The following code relating to execution of the CHKFAR macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'bb'	X'aaaa'	Meaning
X'00'	X'0000'	The function was executed successfully.
X'01'	X'6000'	The function could not be executed: the operand list contains an invalid value.
X'40'	X'6001'	The function could not be executed: the specified file was not found in the catalog.
X'40'	X'6008'	The function could not be executed: the specified catalog is unknown or was not available.
X'20'	X'6014'	The function could not be executed: system error.
X'40'	X'6021'	BCAM connection error
X'40'	X'6022'	BCAM connection interrupted
X'01'	X'6040'	The function could not be executed: The operand list was not available or assigned with the necessary length.

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table in chapter "Standard header".

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller

- the list is not aligned on a word boundary
- the list is write-protected.

Description of the output fields

- The following information is returned for ACCESS = *ANY:
  Access rights: the caller's access rights for the file. Passwords and retention periods are not taken into account.
  - If access control using GUARDS is defined, the guards for the caller are evaluated.
  - If a BASIC-ACL is in effect, the values from the entry applicable to the caller are used.
  - If SHARE/ACCESS is in effect, the values are set as follows:

ACCESS	SHARE	Owner R	Owner W	Owner X	Others R	Others W	Others X
WRITE	NO	Y	Y	Y	N	N	N
WRITE	YES	Y	Y	Y	Y	Y	Y
READ	NO	Y	N	Y	N	N	N
READ	YES	Y	N	Y	Y	N	Y

Key: R: READ, W: WRITE, X: EXECUTE, Y: YES, N: NO

If the file to be checked exists, but the caller has no access rights for it, the CHKFAR macro returns the access rights in the ACCESS-RIGHTS field and the return code null.

- Where ACCESS = *READ/*WRITE/*UPDATE/*DELETE/*EXEC:
  CHECK-RESULT: specifies whether or not the desired access is permitted.

Layout of the operand list

(macro expansion with MF=D and default values for PREFIX and MACID)

```
          CHKFAR MF=D
1          MFCHK MF=D,PREFIX=S,MACID=RMZ,PARAM=,                          C
1                SUPPORT=(C,D,E,L,M),DMACID=RMZ,SVC=8
2 SRMZ     DSECT ,
2                *,##### PREFIX=S, MACID=RMZ #####
1 ************************************************************
1 *               CHKFAR  - PARAMETER AREA                    *
1 ************************************************************
1          #INTF REFTYPE=REQUEST,INTNAME=CHKFAR,INTCOMP=001
1 *
1 SRMZPA     DS    0F          BEGIN of PARAMETER AREA          _INOUT
1 *
1          FHDR  MF=(C,SRMZ),EQUATES=NO
2          DS    0A
2 SRMZFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 SRMZIFID DS    0A              0   INTERFACE IDENTIFIER
2 SRMZFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 SRMZFCT  DS    AL1             2   FUNCTION NUMBER
2 SRMZFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 SRMZRET  DS    0A              4   GENERAL RETURN CODE
2 SRMZSRET DS    0AL2            4   SUB RETURN CODE
2 SRMZSR2  DS    AL1             4   SUB RETURN CODE 2
2 SRMZSR1  DS    AL1             5   SUB RETURN CODE 1
2 SRMZMRET DS    0AL2            6   MAIN RETURN CODE
2 SRMZMR2  DS    AL1             6   MAIN RETURN CODE 2
2 SRMZMR1  DS    AL1             7   MAIN RETURN CODE 1
2 SRMZFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *
1 SRMZACC  DS    XL1           ACCESS                                 001
1 SRMZANY  EQU   0               = *ANY                               001
1 SRMZREA  EQU   1               = *READ                              001
1 SRMZWRI  EQU   2               = *WRITE                             001
1 SRMZUPD  EQU   3               = *UPDATE                            001
1 SRMZDEL  EQU   4               = *DELETE                            001
1 SRMZEXE  EQU   5               = *EXEC                              001
1 *
1 SRMZFILE DS    CL54          FILE   = pathname                      001
1 *
1 SRMZRIF  DS    XL1           RETURN_INFO                            001
1 * Here after: ACCESS_RIGHTS bits returned when ACCESS = *ANY
1 SRMZARR  EQU   X'80'           READ                                 001
1 SRMZARW  EQU   X'40'           WRITE                                001
1 SRMZARE  EQU   X'20'           EXEC                                 001
1 SRMZARU  EQU   X'1F'           UNUSED                               001
1 * Here after: AUTHORIZATION bin-value returned when ACCESS NE *ANY
1 SRMZALW  EQU   0               ALLOWED                              001
1 SRMZFBD  EQU   1               FORBIDDEN                            001
1 *
1 SRMZUNU  DS    XL4           -- MUST BE ZERO --                     001
1 SRMZPA#  EQU   *-SRMZPA      LENGTH OF PARAMETER AREA               001
```

CHNGE - Change TFT entry

Macro type: type S (E form/L form); see "Macro types"

The CHNGE macro changes the file link name in an entry in the task file table (TFT), i.e. a new file link name is assigned to the file. All other values in the TFT entry remain unchanged.

CHNGE cannot be used on the TFT entry of a file which is currently open.

Format

Operation	Operands
CHNGE	**[name$_1$],name$_2$[,MF = L]**
	MF = (E,adr / E,(r))

Operand descriptions

The forms of the MF operand are described in detail in the appendix, "Macro types".

name$_1$
The file link name (1-8 characters long) which is to be replaced by “name$_2$”.

Default setting: the first TFT entry with file link name C"BLANK"BLANK"BLANK"BLANK"BLANK"BLANK"BLANK"BLANK" is processed (e.g. created via the LOCK-FILE-LINK function).

name$_2$
The new file link name (1-8 characters long) which is to replace the old name “name$_1$”.

Programming note

The following return codes are placed in register 15:

X'00' -	call was executed successfully
X'05A6' -	second operand errored
X'05C2' -	file link name contains illegal binary zeros
X'05D5' -	file link name not found
X'05D6' -	file with specified file link name is currently open
X'05DD' -	second file link name already exists

CLOSE - Close file

The CLOSE macro closes files, i.e. it disconnects them from the user program in which they were opened. All input /output buffers which the system generated automatically when the file was opened are now released. The FCB is restored to the state it was in before the file was opened.

During CLOSE processing, the user program can make use of EXLST exits, as with OPEN processing, in order to position the tape (CLOSPOS) or write user labels (LABEND).

A CLOSE macro issued for a file which is not open is ignored.

Format

Operation	Operands
CLOSE	**ALL / fcbadr / (1)** **[,RWD / REPOS / DISCON / LEAVE / INVAL / KEEP-DATA-IN-CACHE / (0)]** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB for the file to be closed.

(0)

Register 0 contains the positioning key and CLOSE mode in the right-hand byte:

X'00'	LEAVE
X'01'	DISCON
X'02'	REPOS
X'03'	RWD
X'05'	INVAL (only with PARMOD=31)
X'06'	KEEP-DATA-IN-CACHE

(1)

Register 1 contains the FCB address.

ALL

Closes all files which were opened in the current program and have not yet been closed. System files and EAM files are not affected. If a file is not closed normally, a warning is issued.

DISCON

For tape files:
The tape is positioned to the start and unloaded/released. A device which may have been reserved using FILE remains assigned to the task; it is not released until a subsequent RELEASE command (REL macro) is issued.

INVAL

For disk files:
The cached pages of the file are to be invalidated, but not written back to the disk, i.e. the data is lost after CLOSE. INVAL can only be specified if PARMOD=31 applies.

KEEP-DATA-IN-CACHE

For disk files:
The data that was buffered in a cache is not saved to the disk at CLOSE. A subsequent OPEN on the same file can then use this data immediately.

Note

Files that have been closed in this way can be displayed using the SHOW-FILE-ATTRIBUTES command (CACHE-NOT-SAVED operand). A backup of the data from the cache onto the disk can be forced either by means of a further OPEN/CLOSE cycle without this function or implicitly when the cache is dissolved with the system administrator command STOP-PUBSET-CACHING or EXPORT-PUBSET. A file-specific cache backup for closed files is not possible.

LEAVE

For tape files:
The tape is positioned to the logical end of the file, depending on the LABEL specification in FILE or FCB.

If the BYPASS operand was specified in the FILE command, the tape position is not changed and the CLOSPOS routine is not activated. Otherwise the LEAVE functions are as indicated in the “REPOS” table: REPOS for OPENREVERSE corresponds to LEAVE for OPEN=REVERSE, and vice versa.

In the case of LEAVE for OPEN OUTPUT, it should be noted that no CLOSPOS routine is activated, and the tape position is not changed.

For multifile tapes, the tape is rewound to the start by CLOSE unless LEAVE is specified.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The macro is expanded in accordance with the format for the 24-bit interface. The object code is thus executable only in 24-bit addressing mode.

= 31
The macro is generated as addressing mode-independent.

REPOS

For tape files:
positions a tape to the logical beginning of the file, depending on the LABEL specification in FILE or FCB.

If the BYPASS operand was specified in the FILE macro, the tape is rewound and the CLOSPOS routine is not activated. In all other cases, the following table applies (position tape):

LABEL spec.	**OPEN != REVERSE**	**OPEN = REVERSE**
LABEL=(STD,n)	The tape is automatically positioned to HDR1; FSEQ is not changed.	The tape is positioned to the tape mark after the last EOF label of the file; FSEQ is incremented by 1.
LABEL=NSTD	EXLST: CLOSPOS=NO, the tape is automatically positioned to the start of tape mark; FSEQ is not changed.	EXLST: CLOSPOS=NO, the tape is positioned to the start of tape mark; FSEQ=0
	EXLST: CLOSPOS!=NO, the user must program a routine to position the tape; FSEQ is not changed	EXLST: CLOSPOS!=NO the user himself positions the tape in the CLOSPOS routine; FSEQ is not changed.
LABEL=NO		EXLST: CLOSPOS=NO, the tape is automatically positioned to the tape mark after the last block; FSEQ is incremented by 1.

For OPEN OUTPUT: no CLOSPOS routine, no positioning.

RWD

Default setting for tape files:
the tape is rewound and positioned to the start; FSEQ is set to zero (this also applies to files with NSTD labels), i.e. FSEQ points to the first file of the file set or tape volume.

Programming note

The CLOSE macro destroys the contents of registers 0, 1, 14 and 15.

COMPFIL - Compare disk files

Macro type: type S (E form/L form/D form/C form/M form) (see "Macro types")

The COMPFIL macro, like the COMPARE-DISK-FILES command, compares two disk files block by block (UPAM) or record by record (SAM, ISAM) and informs the user of the result of the comparison.

Temporary or work files can also be compared. The files can reside on public volumes, Net-Storage or private disks.

The files to be compared must be identical with respect to the following properties:

- Access method (FILE-STRUCTURE or FCBTYPE)
- Block format (BLOCK-CONTROL-INFO)
  When BLKCTRL=*IGNORE, different block formats are also permissible.
- Coded character set (CODED-CHARACTER-SET, EXTENDED_HOST_CODE)
- For SAM files:
  - RECORD-SIZE (RECSIZE) when RECORD-FORM=F (RECFORM=F)
- For ISAM files:
  - RECORD-SIZE (RECSIZE) when RECORD-FORM=F (RECFORM=F)
  - Structure of the ISAM key (KEY-LEN, KEY-POS, LOG-LEN and VAL-FL-LEN)
  - Structure of the secondary key (KEY-LEN, KEY-POS, DUPKEY) for NK-ISAM
- For UPAM files:
  - BUFFER-LENGTH (BLKSIZE)
  - HIGHEST-USED-PAGE (LPP)

Files with the following properties **cannot** be compared:

- empty files
- opened files
- locked files (e.g. SECURE lock)
- REPAIR-NEEDED label set
- NO-DMS-ACCESS label set

Entire file generation groups cannot be compared, although individual file generations can.

PLAM libraries can be compared block by block. The members they contain cannot be compared.

The TPCOMP2 utility routine is available to compare tape files, see the “Utility Routines” manual [14].

Privileged functions

Systems support (TSOS privilege) can compare files of all user IDs. Wildcards are not permissible in the user ID here.

Format

Operation	Operands
COMPFIL	,PATHNM1=<c-string 1..54: filename 1..54> / <var: char:54> ,PATHNM2=<c-string 1..54: filename 1..54> / <var: char:54> ,BLKCTRL=*IGNORE / *INCLUDE ,PAMINFO=*INCLUDE / *IGNORE ,OUTAREA=(NULL / <var: pointer>, 0 / <integer 0..32767> / <var: int:4>) ,CALLER=USER / SYSTEM ,EQUATES=YES / NO ,XPAND=PARAM / OUTPUT MF=L
	MF=D,PREFIX=D / <pre>
	MF=E,PARAM=<name 1..27>

	MF=C / M
	,PREFIX=D / <pre>
	,MACID=MAM / <macid>

Operand descriptions

PATHNM1

Selects the first file to be compared.

=<c-string 1..54: filename 1..54>
Name of the first file.

=<var: char:54>
Only possible with MF=M:
Symbolic address of a memory area of 54 bytes in which the name of the first file is stored.

PATHNM2

Selects the second file which is to be compared.

=<c-string 1..54: filename 1..54>
Name of the second file.

=<var: char:54>
Only possible with MF=M:
Symbolic address of a memory area of 54 bytes in which the name of the second file is stored.

BLKCTRL=*IGNORE / *INCLUDE

Specifies whether the files' block format is included in the comparison (*INCLUDE) or ignored (*IGNORE).

PAMINFO=*INCLUDE / *IGNORE

Specifies whether the user information in the PAM key of UPAM files is included in the comparison (*INCLUDE) with BLOCK-CONTROL-INFO=*PAMKEY or ignored (*IGNORE).

OUTAREA=(<address>,<length>)

Determines the address and length of the output area in which the information concerning the file comparison is to be stored.

<address>=NULL / <var: pointer>
Specifies the address of the output area.

<length>=0 / <integer 0..32767>
Specifies the length of the output area.

<length>=<var: int:4>)
Only possible with MF=M:
Symbolic address of a memory area of 4 bytes in which the length of the output area is stored.

CALLER

Control operand; for MF=E and MF=M only:
Specifies whether an SVC or a direct call is to be generated when the function is called.

= USER
The function call is generated via SVC 144.

= SYSTEM
Control operand; for callers from TPR only:
A direct call is generated with BASR. The DSL conventions apply for the interface. When TU programs are linked, the entry generated cannot be satisfied.

EQUATES

Control operand; for MF=C and MF=D only:
Specifies whether equates are also to be generated for the values in the fields of the parameter or output area when the parameter or output area is expanded.

= *YES
Equates are also generated for the values in the fields of the parameter or output area when the parameter or output area is expanded.

= *NO
No equates are generated for the values in the fields of the parameter or output area when the parameter or output area is expanded.

XPAND

Control operand; for MF=C and MF=D only:
Determines which structure is to be expanded (generated). Specifications for this operand are ignored for other MF values.

= PARAM
The layout of the parameter list is expanded.

= OUTPUT
The layout of the output area is expanded.

Programming notes

1. All RESERVED fields of the parameter area must have been deleted with binary zeros.
2. The caller is responsible for the consistency of the parameter area whenever modifications are made to the parameter area without the help of GCs.
3. For all changes at parameter level that are not called with macros, the caller is responsible for the consistency of the parameter area.
4. The caller is responsible for deleting the output area.
5. In the event of a nonprivileged call (function status TU), register 1 points to the parameter area. In the event of a privileged call (function status TPR), the register assignment complies with the DSL convention.

Return codes

The return code is returned in the standard header of the parameter area. The parameter area may then not be located in the read-only area, otherwise the program terminates.

The following return codes are generated by COMPFIL:

X'cc'	X'bb'	X'aaaa'	Explanation
X'00'	X'00'	X'0630'	No error. The files are identical.
X'00'	X'00'	X'0631'	No error. The files are not identical.
X'00'	X'40'	X'0501'	File catalog not available
X'00'	X'40'	X'0505'	Error in host communication
X'00'	X'40'	X'0512'	File catalog not found
X'00'	X'40'	X'051B'	User ID not on the pubset
X'00'	X'20'	X'0531'	Unexpected error during file catalog access
X'00'	X'01'	X'0554'	Format of file name invalid
X'00'	X'01'	X'0576'	Incorrect operand combination or UNUSED fields not deleted
X'00'	X'82'	X'0594'	Insufficient virtual memory available
X'00'	X'20'	X'05AB'	Address of output area incorrect or not specified
X'00'	X'20'	X'05C7'	Internal error in DMS
X'00'	X'40'	X'05F4'	Specified file names are identical
X'00'	X'40'	X'05FC'	User ID not in home pubset
X'00'	X'01'	X'0624'	Invalid file name
X'00'	X'40'	X'0636'	File attributes are incompatible

X'02'	X'00'	X'06CB'	Output information not completely transferred

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table in chapter "Standard header".

Layout of the parameter area

The parameter area must be aligned on a word boundary. It begins with a standard header which COMPFIL initializes as follows:

Function Unit Number	22
Function Number	33
Interface Version Number	1
Return Code	-1

Macro expansion with MF=D and XPAND=PARAM and default values for EQUATES, PREFIX and MACID:

```
         COMPFIL MF=D,XPAND=PARAM
DMAVGLPL DSECT ,
DMAVHDR  DS    0A
DMAVFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
DMAVIFID DS    0A              0   INTERFACE IDENTIFIER
DMAVFCTU DS    AL2             0   FUNCTION UNIT NUMBER
DMAVFCT  DS    AL1             2   FUNCTION NUMBER
DMAVFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
DMAVRET  DS    0A              4   GENERAL RETURN CODE
DMAVSRET DS    0AL2            4   SUB RETURN CODE
DMAVSR2  DS    AL1             4   SUB RETURN CODE 2
DMAVSR1  DS    AL1             5   SUB RETURN CODE 1
DMAVMRET DS    0AL2            6   MAIN RETURN CODE
DMAVMR2  DS    AL1             6   MAIN RETURN CODE 2
DMAVMR1  DS    AL1             7   MAIN RETURN CODE 1
DMAVFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
*
DMAVPNAM1       DS    CL54               Pathname1
DMAVPNAM2       DS    CL54               Pathname2
DMAVBCTRL       DS    FL1                blockctrl operand
*   values of operand blckctrl_info
DMAVBLKIG       EQU   0                  ignore
DMAVBLKIN       EQU   1                  include
*
DMAVPINFO       DS    FL1                paminfo operand
*  values of operand paminfo
DMAVPIIG        EQU   0                  ignore
DMAVPIIN        EQU   1                  include
*
DMAVRES1        DS    XL6                RESERVED
DMAVARAD        DS    A                  Outarea=(<addr>,...)
DMAVARLN        DS    F                  Outarea=(...,<length>)
DMAV#           EQU   *-DMAVHDR
```

Layout of the output area

The output area must be aligned on a word boundary.

Macro resolution with MF=D and EXPAND=OUTPUT, as well as default values for EQUATES, PREFIX and MACID:

```
         COMPFIL MF=D,XPAND=OUTPUT
DMAVOUTP DSECT ,
*   Compfile Output
DMAVMSGNR      DS    F                   MESSAGENUMBER
*
DMAVMSGINF     DS    0XL80               MESSAGEINFO
DMAVMSG_DETAILS DS   0XL80               MESSAGEDETAILS
DMAVPNAM       DS    CL80                PATHNAME
           ORG   DMAVMSG_DETAILS
*
DMAVCMPINF     DS    0XL12               COMPAREINFO
DMAVPAGNR      DS    F                   PAGENUMBER
DMAVRECNR      DS    F                   RECORDNUMBER
DMAVBYTENR     DS    F                   BYTENUMBER
DMAVABSRECNR   DS    F                   ABSOLUT RECORDNUMBER FOR SAM
*                                        FILES
DMAVERRNR      DS    FL1                 ATTRIBUT ERROR
*   type of file attribut error
DMAVBKCTR      EQU   1                   blk-contr
DMAVFISTR      EQU   2                   file-struc
DMAVFITYP      EQU   3                   file-type
DMAVHIUSP      EQU   4                   high-us-pa
DMAVRECFR      EQU   5                   rec-form
DMAVRECSZ      EQU   6                   rec-size
DMAVKEYPO      EQU   7                   key-pos
DMAVKEYLN      EQU   8                   key-len
DMAVVALLN      EQU   9                   val-len
DMAVLOGLN      EQU   10                  log-len
DMAVALTIX      EQU   11                  alternate-index
DMAVLBP        EQU   12                  last-byte-pointer
DMAVBKSZ       EQU   13                  block-size
DMAVPKUI       EQU   14                  pamkey-user-info
DMAVLASTP      EQU   14                  last-position
DMAVABSRECNR63 DS    0XL8                ABSOLUT RECORDNUMBER FOR SAM
*                                                    FILES big counter
DMAVABSRECNRH   DS    F                       high cnt
DMAVABSRECNRL    DS    F                       low cnt
*
*
         ORG   DMAVMSG_DETAILS+80
*
DMAVUNUS       DS    XL8                 UNUSED
DMAVOUTPUT#    EQU   *-DMAVMSGNR
```

The following cases are distinguished when the COMPFIL information is output to the user's output area:

- No output was possible
  No output area was made available by the caller or the output area was write-protected. In the standard header of the output area the user receives the return code X'05AB' following validation of the output area or of the address. If the output area was too small to transfer the output information, the caller receives the return code X'06CB'.
- Error when accessing one or both of the files to be compared
  The first file in which an access error occurs is output in the DMAVPNAM field in the format :<cat-id>:$<user-id>.<filename>. In addition, message number X'0681', indicating a general file access error, is output in the DMAVMSGNR field. The precise reason for this access error can be found in the return code placed in the parameter area's standard header.
- The comparison of the two files is not possible because of the incompatibility of their file attributes
  Message number X'0636' is output in the standard header of the parameter area and in the DMAVMSGNR field of the output area. In addition, the DMAVERRNR field indicates which file attribute led to the comparison aborting.
- The same name was specified for both files
  The same return code is displayed in the standard header of the parameter area and in the DMAVMSGNR field of the output area.
- The two files are identical
  Return code X'0630' is displayed in the standard header of the parameter area. The DMAVMSGNR field of the output area contains X'0000'.
- The two files are not identical
  Return code X'0631' is displayed in the standard header of the parameter area. Depending on the access method, the following correlation exists between MESSAGENUMBER in the DMAVMSGNR field and the COMPARINFO in the DMAVPAGNR (PAGENUMBER), DMAVRECNR (RECORDNUMBER), DMAVBYTENR (BYTENUMBER) and DMAVABSRECNR (ABSOLUT RECORD NUMBER FOR SAM FILES) fields.

Access method	MESSAGE NUMBER	PAGE NUMBER	RECORD NUMBER	BYTE NUMBER	ABSOLUTE RECORD NUMBER	Meaning
SAM	X'0632'	<p>	<r>		<a>	[1]
ISAM	X'0633'		<r>			[2]
UPAM	X'0634'	<p>		<b>		[3]
UPAM	X'0635'	<p>				[4]

[1] The two SAM files differ as of record <a>. That is record <r> within the 2k data block <p>.

[2] The two SAM files differ as of record <r>.

[3] The two UPAM files differ as of data byte <b> within the 2k data block <p>.

[4] The two UPAM files differ in the user information in the PAM key of the 2K data block <p>, but their contents are identical (specification PAMINFO=*INCLUDE).

Sample calling sequence

```
         MVC     COMPMFC(DMAV#),COMPMFL
         COMPFIL MF=M,OUTAREA=(A(COMPOAC),OUTLEN),PARAM=COMPMFC,       -
                      PATHNM1=':X:SAM.1',PATHNM2=':X:SAM.2'
         COMPFIL MF=E,PARAM=COMPMFC
            .
            .
COMPMFC  COMPFIL MF=C,XPAND=PARAM
COMPOAC  COMPFIL MF=C,XPAND=OUTPUT
COMPMFL  COMPFIL MF=L,PATHNM1='AAA',PATHNM2='BBB'
OUTLEN   DC      A(DMAVOUTPUT#)
```

COPFILE - Copy file

Macro type: type S (C form/D form/E form/L form/M form); see "Macro types"

The COPFILE macro copies files, file generations and file generation groups in blocks from disk to disk, from disk to tape and from tape to disk without modifying them. Consequently it cannot normally be used to modify file attributes. The only exception concerns the block control attribute, which can be modified during copying in certain cases (see "File link names" and the description of the operand "BLKCTRL").

If the output (or target) file is not yet cataloged, it is automatically created on a public volume (as with a FILE with default values for the specified output file) when COPFILE is executed.

If the target file is to be on a different Speichertyp (Public-Storage oder Net-Storage) oder volume (private disk, Net-Storage or tape), it must be set up using FILE (operands DEVICE, VOLUME) before the COPFILE macro is called.

If the target file is a disk file which has not yet been cataloged, the primary and secondary allocations are taken from the original disk file.

If the target file is a disk file, its primary and secondary allocations are not modified unless they are smaller than those of the original file.

If the original file is on tape, a default value is used for the target file.

Note

> The COPFILE macro is the earlier COPY macro extended by the use of wildcards in pathname1 (selection) and pathname2 (construction). The additional operands CHECK and LIST have also been provided. The function of the earlier COPY is still supported. The format of the COPY macro is therefore still included in the appendix (see "Formats of replaced macros"). Its operands, however, are the same as those of the COPFILE macro and are therefore only described here.

File generation groups

A file generation group can be copied into another file generation group only if one of the following conditions is fulfilled:

- The group entries for the two file generation groups match (i.e. the values for GEN, FIRST, LASTGN and BASE are the same). In the file generation group into which DMS is to write the copy, the generations from FIRST to LASTGN must already be cataloged and have storage space allocated.
- The value for GEN is the same for both file generation groups, and the file generation group into which DMS is to write the copy contains no generations (i.e. FIRST, LASTGN and BASE have the value zero).

The file generation group to be copied may not contain tape file generations (COPFILE does not support copying of tapes).

A file generation group can be copied into a single file or file generation only if the following conditions are satisfied:

- The file generation group consists of SAM file generations with identical attributes (e.g. the same record and block lengths, the same record format, the same block control attribute).
- The file generation to which the copy is to be made does not belong to the file generation group to be copied.

A single file or file generation can only be copied into a file generation group if the following condition is fulfilled:

- The file or file generation must possess the same CODED-CHARACTER-SET as the file generation group.

Files on private disks

If a file on private disk only has an entry in the system catalog but no F1 label, the catalog entry is deleted. If the file is the input file, COPFILE is rejected.

A COPFILE call for an ISAM file on private disk with index and data sections on different disks is rejected.

Tape files

Internally, COPFILE uses the UPAM access method, which does not support continuation tape processing. This means that it is possible to copy several files to the same tape (FILE macro, FSEQ operand), but not files that extend over more than one tape.

- **K tape files** (BLKCTRL=PAMKEY) must have standard block format (BLKSIZE=(STD,n)) if they are to be processed by the COPFILE macro.
- **NK tape files** (BLKCTRL=DATA/NO) can be processed by COPFILE if their BLKSIZE value is a multiple of 2048 bytes.
  If NK files are copied to tape, the BLKCTRL information is lost when the catalog entry is deleted. If the file is to be copied back again, the COPFILE macro must be preceded by a FILE macro with the operands LINK and STATE=FOREIGN and with the correct value for the BLKCTRL operand, i.e. either NO or DATA, to match the actual data format of the file.

If a K file (BLKCTRL=PAMKEY) is inadvertently copied in this manner into an NK file (BLKCTRL=DATA), the resulting disk file cannot be read, because the first 16 bytes of each logical block, which contain data when BLKCTRL=PAMKEY applies, are overwritten with management information.

- **Foreign files on tape**: if an uncataloged tape file is to be copied, a TFT entry with the file link name valid for COPFILE must be created before copying in order to define the file attributes (see "File link names").
  `FILE pathname`$_1$`,LINK=DMCOPY11,STATE=FOREIGN,BLKCTRL=...`

File link names

Internally, COPFILE uses the file link names DMCOPY11 (for the original file pathname1) and DMCOPY22 (for the target file pathname2). On completion of processing, the file link names are implicitly released (implicit REL macro).

The option of selecting an original and a target file for COPFILE processing by means of a FILE macro with appropriate file link names can be used, for example, in order to modify the file's block control attribute during copying. Specifying the BLKCTRL operand in the FILE macro together with BLKCTRL=*IGNORE/*CHECK in the COPFILE macro enables the definition of different BLKCTRL attributes for original and target file in the course of copying (see the description of the operand "BLKCTRL").

When there are wildcards in the file name, an existing TFT entry (DMCOPY11/DMCOPY22) only becomes effective when the first file to be processed is copied.

Remote file access (see also the “RFA” manual [6])

Copying from one remote system to another, with input and output on different systems, is supported by a higher-level execution routine. In this case, the local system acts only as an intermediate station for data transfer. A SET-RFA-CONNECTION command must be issued for each of the remote systems before copying is started.

If a remote file is copied to a local file with the PROTECT=*SAME operand, the passwords are not copied with the file.

SM pubsets

If the target file does not yet exist, an attempt is made to create it on a suitable volume set using the source file attributes for selecting the volume set (performance, availability).

File encryption

Normally no crypto password is required to copy an encrypted file. COPFILE transfers the contents of an encrypted file without decrypting the file, and the target file is assigned the same encryption attributes as the source file, in particular the crypto password.

The exceptions here are copy operations which require file decryption:

- An encrypted file is to be copied to tape or private disk.
- An encrypted file is to be copied to a file generation.
- A shared update was declared via the TFT entry DMCOPY11 or DMCOPY22.

Macro format

Operation	Operands

COPFILE	**BLKCTRL = *IGNORE / *CHECK / <var: blkctrl>**
	,CHDATE = *STD / *SAME / <var: bit: 1>
	,CHECK = *MULTIPLE / *NO / *ERROR / *SINGLE / *CATALOG / *USERID / <var: check>
	***,IGNORE* = *SOURCE / *TARGET / (*SOURCE,*TARGET)**
	,LIST = *NO / *SYSOUT / *ERRORS_TO_SYSOUT / <var: list>
	,PATHNM1 = <c-string 1..80: filename 1..54 with-wild(80)> / <var: char: 80>
	,PATHNM2 = <c-string 1..80: filename 1..54 with-constr-wild(80)>/ <var:char: 80>
	,PROTECT = *STD / *SAME / *SAME-AND-CHANGE-DATE / <var: prot>
	,REPLACE = *YES / *NO / <var: replace>
	,MF = C / D / E / L / M
	,PARAM = DMACOPPL / <addr> / <(r)>
	,PREFIX = D / <pre>
	,MACID = MAC / <macid>

Operand descriptions

BLKCTRL

Specifies whether the target file (or the TFT entry DMCOPY22) may have a different BLKCTRL attribute than the source file pathname$_1$.

TFT entry or target file (with null operands in the TFT) must have the same BLKCTRL attribute as the source file.

Default value: pathname$_1$ and the TFT entry for DMSCOPY22 must have the same BLKCTRL attribute.

= *IGNORE
Even if the BLKCTRL attributes of pathname$_1$ and the TFT entry for DMCOPY22 do not match, pathname$_1$ can be copied to pathname$_2$ in the following cases:

BLKCTRL attribute of the file pathname $_1$	**BLKCTRL attribute of the file pathname $_2$**
PAMKEY	DATA (disk files only)
PAMKEY	NO
DATA (disk files only)	PAMKEY
NO	PAMKEY

Note

It is the user's responsibility to ensure that no data is lost in the course of copying. This danger exists when copying a file with BLKCTRL=PAMKEY to a file with BLKCTRL=DATA or BLKCTRL=NO: in both cases, the information in the user section of the PAM key is lost. Furthermore, if the target file has the attribute BLKCTRL=DATA, the first 12 bytes of each logical block (in the case of ISAM files, the first 16 bytes) are overwritten by the block control field.

= *CHECK
Even if the BLKCTRL attributes of pathname$_1$ and the TFT entry for DMCOPY22 do not match, it is possible to copy pathname$_1$ to pathname$_2$ whenever this can be done without losing any user information in the user section of the PAM key. If the user part of the PAM key does not contain any user information (this is checked here), pathname$_1$ can be copied to pathname$_2$ when the following BLKCTRL attributes apply; otherwise, the command is rejected.

BLKCTRL attribute of the file pathname $_1$	**BLKCTRL attribute of the file pathname $_2$**
PAMKEY	DATA (disk files only)
PAMKEY	NO

CHDATE

Specifies whether the target file will be given the same change date (CHANGE-DATE) as the source file.

= *STD
*Only for PROTECT=*STD or *SAME:*
The change date of the target file is updated.

> The specification PROTECT=*SAME-AND-CHANGE-DATE is still supported for reasons of compatibility and causes the source file's change date to be transferred to the target file.

= *SAME
The source file's change date is transferred to the target file. The specification CHDATE=*SAME also applies in the following cases:

- The target file is located under a foreign user ID.
- The target file is a file generation.

CHECK

Only for wildcard entries:
Defines the conditions in interactive mode under which a user dialog is to be started if multiple files are selected using wildcards.

If the dialog is started, the user can decide whether or not processing is to be executed on the displayed file(s). He can also call up help text on the reply options and define a new value for CHECK and/or LIST when processing is resumed.

The value 'NO' always applies in batch mode.

The operand has no effect if $pathname_1$ contains no wildcards or is not partially qualified.

= *MULTIPLE
A check dialog is only started if multiple files are selected.
If the catalog and/or user ID contain wildcards, a check dialog is executed for each catalog and/or user ID.
CHECK=*ERROR is also implied.

= *NO
All selected files are processed without a check dialog, i.e. without any possible user intervention.

= *ERROR
An error check dialog is started if an error occurs during processing of a selected file name. A file set check dialog is started if the selection entry selects more files than can be processed in available memory.
CHECK=*ERROR is also always implied for all entries where CHECK!=*NO.

= *SINGLE
A check dialog is executed for each selected file name. CHECK=*ERROR is also implied.

= *CATALOG

The user must decide in a check dialog for each catalog whether the files selected in them are to be processed. CHECK=*ERROR is also implied.

= *USERID
Reserved for system administrators:
The system administrator must decide in a check dialog for each user ID and each catalog whether the selected files are to be processed.
CHECK=*ERROR is also implied.

IGNORE

For the system administrator only:
Allows the system administrator to ignore file protection for the source and/or target file.This operand has no effect on files located on a remote computer (RFA). If a TSOS restriction exists for a file under a foreign user ID then the ACCESS protection attribute is ignored.

= *SOURCE
The protection attributes READ-PASSWORD and EXEC-PASSWORD of the source file are ignored when copying (also applies to BASIC-ACL and GUARDS protection).

= *TARGET
The protection attributes ACCESS and EXPIRATION-DATE and the READ-/WRITE-/ EXEC-PASSWORD attributes of the target file are ignored when copying (also applies to BASIC-ACL and GUARDS protection).

LIST

Defines whether a log is to be written to SYSOUT for all file names selected with wildcards after their processing.

The operand has no effect if $pathname_1$ contains no wildcards or is not partially qualified.

= *NO
No log is kept.

= *SYSOUT
Each processed file and any errors that occur are logged in a report.

= *ERRORS_TO_SYSOUT
Only those files whose processing led to errors are logged in a report.

MACID

Only evaluated in conjunction with MF=C; defines the second through fourth characters of the field names and equates which are generated during macro execution in the data area.

Default: MACID = MAC

= macid
“macid” is a three-character string which defines the second through fourth characters of the generated field names and equates.

MF

The forms of the MF operand are described in detail in the appendix (''Macro types'').

PARAM

Defines the address of the operand list and is only evaluated in conjunction with MF=E (see also "Macro types").

= addr
The symbolic address (the name) of the operand list.

= (r)
The number of the register containing the address of the operand list. This register must be loaded with the appropriate address value before calling the macro.

PATHNM1 =

Pathname of the original file

= <c-string 1..80: filename 1..54 with-wild(80) without-gen>
$pathname_1$ (enclosed in single quotes)

= <var: char: 80: filename 1..54 with-wild(80) without-gen>
Name of a variable that contains $pathname_1$

$Pathname_1$ means [:$catid_1$:][$userid_1$.]$filename_1$

catid $_1$
Catalog ID of the original file; default value: the catalog ID belonging to the user ID.

userid $_1$
User ID of the original file; default value: the user ID specified in the SET-LOGON-PARAMETERS/LOGON command.

filename $_1$
Name of the original file, file generation or file generation group.

Read permission must exist for the original file.

If $pathname_1$ is an FGG, $pathname_2$ must also be an FGG, unless the FGG $pathname_1$ consists of SAM file generations with the same attributes with respect to record format, record length, block size, and block control information. In this case, it is possible to copy into a single file or into a file generation, but this file generation must not belong to the FGG which is to be copied.

Wildcard use
Selection criteria for the files to be copied. The nonprivileged user may only use wildcards in the catalog ID and file name.

PATHNM2

Pathname of the output/target file.

= <c-string 1..80: filename 1..54 with-wild(80) without-gen>
$pathname_2$ (enclosed in single quotes)

= <var: char: 80: filename 1..54 with-wild(80) without-gen>
Name of a variable that contains $pathname_2$

pathname$_2$ means [:catid$_2$:][$userid$_2$.]filename$_2$

catid $_2$
Catalog ID of the output file; default value: the catalog ID belonging to the user ID.

userid $_2$
User ID of the output file; default value: the user ID specified in the SET-LOGON-PARAMATERS or LOGON command.

filename $_2$
Fully qualified name of the output file, file generation or file generation group.

pathname$_1$ and pathname$_2$ must not be identical.

If pathname$_2$ is not yet cataloged, only the user's own user ID may be specified, i.e. the user ID of the SET-LOGON-PARAMETERS/LOGON command or a user ID of which the user is co-owner.

If pathname$_2$ is already cataloged, **write access** must be permitted.

The COPFILE macro call is rejected if pathname$_2$ is read-only (e.g. ACCESS=READ or EXDATE > current date) or if the secondary allocation for disk file pathname$_2$ is 0 and the primary allocation is too small to accommodate the file to be copied.

If pathname$_2$ is cataloged under a foreign user ID, this user ID must also be specified.

If pathname$_2$ is a file generation group, pathname$_1$ must also be a file generation group.

Wildcard use
Construction entry for the files to be copied to as a result of the selection criteria (pathname$_1$).

SM pubsets
If the output/target file is not yet cataloged, an attempt is made to create it on a suitable volume set using the attributes of the original file.

PREFIX

Only evaluated in conjunction with MF=C or MF=D; this defines the first character of field names and equates which are generated in the data area with macro execution.

Default: PREFIX = D

= pre
A single-character prefix with which field names and equates generated by the assembler are to begin.

PROTECT

Defines whether the copy pathname$_2$ receives the same file backup and protection attributes as pathname$_1$.
As far as is possible and permitted, the encryption attributes are taken over into the target file when copying takes place regardless of the PROTECT specifications (see also "File encryption").

= *STD
If pathname$_2$ is not yet cataloged, the new file is set up with the default attributes (see operand defaults in the CATAL macro, "CATAL - Process catalog entry", e.g. SHARE=NO, ACCESS=WRITE for disk files etc.).

= *SAME
The copy pathname$_2$ receives the same file backup and file protection attributes as pathname$_1$ (identical values for ACCESS, BACKUP, DELDATE, DESTROY, LARGE, MANCLAS, MIGRATE (FORBIDDEN is set to INHIBIT), NUM-OF-BACKUP-VERS, OPNBACK, RETPD, SHARE and the same passwords). The following are not transferred: AUDIT, AVAIL, PREFORM, S0MIGR, STOCLAS, VOLSET and WORKFIL.

The entry PROTECT=*SAME is ignored if pathname$_2$ is cataloged under a foreign user ID or is a file generation (its file attributes are then defined in the group entry).

If a temporary file is copied into a permanent file, only the attribute BACKUP=E is taken over for the specification PROTECT=*SAME. The new file is ignored for ARCHIVE save runs. If the new file is to be saved automatically with ARCHIVE, the BACKUP value must be changed by means of CATAL.

When pathname$_1$ is protected by a BASIC-ACL entry (see the “Introductory Guide to DMS” [1]), or GUARDS entry, the following points apply to copying with PROTECT=*SAME:

- If the target file was created on a public disk, the access rights of BASIC-ACL or GUARDS are copied.
  If the target file pathname$_2$ is created on a private disk, and if pathname$_1$ is protected by BASIC-ACL, the protection attributes from the BASIC-ACL are used for pathname$_2$. If a GUARDS entry has been created for pathname$_1$, pathname$_2$ is assigned the default protection attributes SHARE=NO and ACCESS=WRITE.
- If the target file pathname$_2$ is created on a magnetic tape, it is assigned the default protection attributes SHARE=YES and ACCESS=WRITE, regardless of the protection attributes defined for pathname$_1$ by the BASIC-ACL or GUARDS.
- If the source file pathname$_1$ is not cataloged under the user ID under which COPFILE was called, pathname$_2$ is assigned default protection attributes, regardless of the protection attributes defined by the BASIC-ACL or GUARDS for pathname$_1$. These default protection attributes are USER-ACCESS=OWNER-ONLY and ACCESS=WRITE for disk files, USER-ACCESS=ALL-USERS and ACCESS=WRITE for tape files.

When copying to tape, the retention period (EXDATE) can only accept values up to a difference of 32767 (for larger values the maximum value is assumed).

= *SAME-AND-CHANGE-DATE
This specification has the same effect as PROTECT=*SAME. In addition, the source file’s change date (CHANGE-DATE) is transferred to the target file.

> The specification PROTECT=*SAME-AND-CHANGE-DATE is only still supported for reasons of compatibility. The CHDATE=*SAME operand should be used to transfer the change date of the source file.

REPLACE

The user can specify whether an existing output file pathname$_2$ is to be overwritten.If pathname$_2$ is a tape file or if it is empty, the operand is ignored and the “old” file is overwritten without output of a message.

= *YES

“pathname$_2$” is overwritten without output of a message.

= *NO

“pathname$_2$” is not overwritten. The call is rejected with the error code X'051A'.

Return codes

The error code is only returned in the standard header and no longer in general-purpose register 15 as with the COPY macro. If the parameter area is not accessible or shorter than the length of the standard header or if a setup error occurs, program termination is initiated via STXIT. The error codes are described in the DMAIDEM/DCOIDEM macros.

Standard header: `ccbbaaaa`

The following code relating to execution of the COPFILE macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	No error
X'01'	X'00'	X'0000'	Only in conjunction with check dialogs: the job was completely or partially withdrawn in interactive mode, i.e. at least one check dialog was answered with *NO.
X'02'	X'00'	X'0000'	Only in conjunction with CHECK!=NO: an error has occurred, but continuation of the function was requested in an error dialog
	X'40'	X'0501'	Requested catalog not available
	X'82'	X'0502'	Requested catalog in the rest state
	X'40'	X'0503'	Incorrect information in the MRSCAT
	X'82'	X'0504'	Error in catalog management system
	X'40'	X'0505'	Computer communication error (MRS)
	X'80'	X'0506'	Operation canceled because of master change
	X'40'	X'0510'	Error while calling an internal function
	X'40'	X'0512'	Requested catalog unknown
	X'40'	X'051A'	File already exists
	X'40'	X'051B'	User ID not known in specified pubset
	X'40'	X'051C'	No access right to specified pubset
	X'40'	X'051D'	LOGON password different on specified pubset
	X'20'	X'0530'	Error in storage space request
	X'20'	X'0531'	Unexpected catalog access error
	X'40'	X'0533'	File not found
	X'82'	X'0534'	Private volume cannot be allocated

X'40'	X'0535'	No access right to the file catalog entry (only in conjunction with CCS or NETCSS assignment on foreign user ID)
X'20'	X'053B'	System error during file access
X'82'	X'053C'	Catalog file of the pubset is full
X'40'	X'053D'	Catalog or F1 label block is destroyed
X'40'	X'053E'	File on private volume already cataloged
X'82'	X'053F'	File reserved by another task
X'01'	X'0576'	Contradictory operand combination or reserved fields of the parameter area used
X'20'	X'0577'	Internal error during access to job environment
X'82'	X'0594'	Not enough virtual memory available. This return code can also occur in particular in conjunction with a selection specification (wildcard) if too many files are selected
X'01'	X'0599'	Operand is not supported in the RFA-BS version
X'01'	X'05A7'	First file name incorrect
X'01'	X'05A9'	Second file name incorrect
X'20'	X'05C7'	Internal error in DMS
X'01'	X'05EE'	File name too long
X'01'	X'05F0'	Foreign user ID not permitted for file2
X'01'	X'05F1'	Copying to the specified file not possible
X'01'	X'05F2'	Illegal specification of *DUMMY
X'40'	X'05F3'	First or second file protected
X'01'	X'05F4'	First and second file name are identical
X'20'	X'05F5'	Some blocks could not be copied
X'01'	X'05F6'	File cannot be copied
X'40'	X'05F9'	Incompatible attributes of source and target file
X'40'	X'05FC'	Specified user ID not in home pubset
X'40'	X'0610'	The function execution sent a return code for at least one of the selected file names
X'01'	X'0611'	Incorrect constructor specification (PATHNM2 operand in conjunction with wildcards)
X'40'	X'0666'	The file is write-protected by GUARDS (only in conjunction with CCS assignment on foreign user ID)
X'40'	X'0698'	File generation groups do not have the same attributes

	X'40'	X'06B5'	File is not properly closed
	X'40'	X'06B6'	Attributes of the file are not compatible with the file generation group
	X'40'	X'06C4'	File generation group not yet cataloged
	X'01'	X'06C7'	Invalid generation number specified
	X'40'	X'06CC'	only with selection specification (wildcard): no file matches the selection specification
	X'01'	X'06D7'	Generation group cannot be copied to an individual generation of this group
	X'01'	X'06D8'	Generations of the specified group have different file characteristics
	X'01'	X'06DE'	File or generation cannot be copied to a group
	X'01'	X'06FD'	Parameter area invalid or not accessible
	X'40'	X'06FF'	BCAM connection aborted
	X'01'	X'FFFF'	Wrong function number in parameter area header
	X'03'	X'FFFF'	Wrong version number in parameter area header

CREAIX - Create secondary keys for ISAM file

Macro type: type S (E form/L form/D form/C form); see "Macro types"

The CREAIX macro defines one or more secondary keys for an NK-ISAM file. Up to 30 secondary keys may be declared for one file; each must be identified by a name defined in the CREAIX macro. Each of these secondary keys can then be addressed via its name in the macros GET, GETR, GETKY and SETL, thus permitting the user to access records on the basis of secondary key values.

In order to create a secondary key, all the records in the file are first read sequentially. For each record, a triplet is formed from the index in the list of secondary keys to be created, the current secondary key itself and the current primary key. These triplets are then sorted for each secondary key, in ascending order of the secondary key values, a time stamp is added to each and they are transferred to the secondary index blocks created for this secondary key.

Secondary keys can be created only for existing NK-ISAM files, i.e. for files which have already been opened at least once with OUTPUT or OUTIN. Furthermore, there must be no duplicate primary keys in a file for which a secondary key is to be created (no DUPKEYs) and neither logical nor value flags may be defined for the file. When the macro is called, neither SHARED-UPDATE=*YES (via an ADD-FILE-LINK command), nor SHARUPD=YES (by a macro) must have been set for the NK-ISAM file, and no other user can access it while the secondary key is being set.

If the program is aborted during creation of the secondary key, the secondary key is flagged as incomplete in the control block of the file. If the user then attempts to open the file, control branches to the OPENER exit (assuming it has been defined in the program) and error code 0D84 is placed in the FCB. The file cannot be opened again until the incomplete secondary key has been deleted (and, if applicable, defined again by means of CREAIX).

For performance reasons, it is advisable not to define secondary keys for a file until it has been filled with records.

Format

Operation	Operands
CREAIX	[,DUPKEY = (YES / NO [,YES / NO,...])] ,KEYLEN = (keylen1 [,keylen2,...]) ,KEYNAME = (keyname1 [,keyname2,...]) ,KEYPOS = (keypos1 [,keypos2,...]) ,LINK = linkname1 / FILE = pathname [,SORTLNK = linkname2] [,VERSION = 1 / 2] MF = L
	MF = E,PARAM = adr / (r)
	MF = D [,PREFIX = D / pre] [,VERSION = 1 / 2]
	MF = C [,PREFIX = D / pre] [,MACID = ISS / macid] [,VERSION = 1 / 2]

Operand descriptions

DUPKEY

Specifies whether duplicate values may exist in different records for each secondary key to be created (KEYNAME operand).

The parentheses in this entry can be omitted if the list only contains one specification for DUPKEY. A list can only be specified for VERSION=2; for VERSION=1, only an entry without parentheses is allowed.

= YES
Default value; the same value of the secondary key may occur in more than one record in the file.

= NO
Different records in the file must not have the same value for the secondary key.

FILE = pathname

Denotes the NK-ISAM file for which a secondary key is to be created, with:
<c-string 1..54: filename 1..54>.

When the CREAIX macro is called, the file must have been opened at least once with OUTPUT or OUTIN and it must not contain duplicate primary key values, logical or value flags. The value specified for the FILE operand is ignored if the LINK operand is also specified.

Pathname means [:catid:][$userid.]filename

catid
Catalog ID: if omitted, the default catalog ID for the user ID is assumed.

userid
User ID: if omitted, the user ID in the SET-LOGON-PARAMETERS or LOGON command is assumed.

filename
Fully qualified file name.

KEYLEN = (keylen1 [,keylen2,...])

Specifies the length of each secondary key (KEYNAME operand) to be created (in bytes). “keylen” is any whole number where 1 <= keylen <= 127.
KEYPOS and KEYLEN must be selected such that the secondary key

- is completely contained in even the shortest record of the file and
- lies entirely within a data block and does not extend into an overflow block.

The parentheses in this entry can be omitted if the list only contains one specification for KEYLEN. A list can only be specified for VERSION=2; for VERSION=1, only an entry without parentheses is allowed.

KEYNAME = (keyname1 [,keyname2,...])

Specifies the name of the secondary key to be created. A maximum of 30 names may be specified in the list and it must be noted that a maximum total of 30 secondary keys can be created for an NK-ISAM file.

The name must not already have been defined for another secondary key. “keyname” may be up to eight characters long and may contain any letters or digits and the special characters “$”, “#” and “@”; it must begin with a letter or special character.

The parentheses in this entry can be omitted if the list only contains one name. A list can only be specified for VERSION=2; for VERSION=1, only an entry without parentheses is allowed.

KEYPOS = (keypos1 [,keypos2,...])

Specifies the position within a record, of the first character of each secondary key (KEYNAME operand) to be created.
“keypos” may be any integer in the range 1 <= keypos <= 32496.
In variable-length records, the four bytes used for the record length and control field must be taken into account.
KEYPOS and KEYLEN must be selected such that the secondary key

- is completely contained in even the shortest record of the file and
- lies entirely within a data block and does not extend into an overflow block.

The parentheses in this entry can be omitted if the list only contains one specification for KEYPOS. A list can only be specified for VERSION=2; for VERSION=1, only an entry without parentheses is allowed.

LINK = linkname1

Specifies the link name for the file for which a secondary key is to be created. When the program is executed, an NK-ISAM file must be assigned to this link name. When CREAIX is called, this file must have been opened at least once with OUTPUT or OUTIN and it must not contain duplicate primary key values, logical flags or value flags. “linkname1” may be up to eight characters long. If the file link name is to be accessible via the command interface, it must comply with the data type <structured_name 1..8> (see the “Commands” manual [3]).

MACID

Defines the second through fourth characters of each field name and equate generated when the macro is expanded.

Default value: MACID = ISS

= macid

Three-character string defining the second through fourth characters of the generated field names and equates.

PARAM

Specifies the address of the operand list; it is evaluated only if MF=E applies (see "Macro types").

= addr
Symbolic address (name) of the operand list.

= (r)
Number of the register which contains the address of the operand list. The register must be loaded with this address value before the macro is called.

PREFIX

Defines the first character of each field name and equate generated when the macro is expanded.

Default value: PREFIX = D

= pre

One-character prefix with which the generated field names and equates are to begin.

SORTLNK = linkname2

Specifies the file link name of a work file for the sort program. This work file is used only if there is not enough virtual address space for sorting the entries for the secondary index block.
If a work file is needed for sorting and SORTLNK was not specified, or if no file is assigned to the file link name when the program is executed, the macro creates a work file with the name DISWORK.tsn (where “tsn” is the task sequence number of the task which called the macro).
“linkname2” may be up to eight characters long and must be formed from letters, digits and special characters in accordance with the rules governing the format of file names.

VERSION

Defines the version of the generated CREAIX macro.

= 1
The “old” macro version is generated.

= 2
The version of the CREAIX macro, which is valid as of BS2000/OSD-BC V3.0, is generated.

Programming notes

1. The C and D forms of the macro generate field names and equates for return codes. They begin with the string DISS..., which can be modified with the PREFIX and MACID operands.
2. If no symbolic address is specified with the D form, the DSECT name DISCRAIX is generated, where the first character is modified by a PREFIX entry.
3. When the CREAIX macro is expanded, a field is created in the parameter list with the name KEY# (with the default prefix DISS or correspondingly modified by the PREFIX and MACID operands). This field contains the number of secondary indices to be created (maximum 30), supplied by the macro expansion. If, however, the parameter list is built up dynamically at program runtime, the KEY# field must be supplied explicitly by the program.
4. If an error occurs, the parameter list contains the index of the secondary index in the specified list with which the error occurred. In addition to this, any DMS error that occurs is also stored in the parameter list (the name of the field containing the index of the secondary index with which the error occurred is DISAERR or <xxxy>AERR depending on PREFIX and MACID)
5. A total of up to 30 secondary indices can be created for an NK-ISAM file. It must therefore be noted that, on the one hand, the list of names in specified in the macro for the secondary indices to be defined may not contain more than 30 elements. On the other hand, the sum of existing secondary indices and those to be created may also not exceed 30 (both cases lead to a corresponding return code).

6. It must be ensured that for a parameter list VERSION is consistently given a value for calls with different MF formats.

Return codes

The return codes output by the CREAIX macro are stored in the standard header of the operand list. The standard header must be defined for the CREAIX parameter list before generating the DSECT.

By default, the return codes generated with the C or D form of the macro start with the string DISS, which can be modified by specifying PREFIX (first character) and/or MACID (second through fourth characters).

Standard header: `ccbbaaaa`

The following code relating to execution of the CREAIX macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'bb'	X'aaaa'	Meaning
X'00'	X'0000'	The function was executed successfully.
X'01'	X'0001'	The operand list is not available.
X'40'	X'0003'	The specified catalog ID does not exist.
X'40'	X'0004'	The catalog cannot be accessed.
X'01'	X'0005'	The operand list contains an invalid name.
X40'	X'0006'	The specified file contains duplicate keys.
X'40'	X'0007'	The secondary key to be created already exists.
X'01'	X'0009'	The value specified for KEYLEN is invalid.
X'40'	X'000A'	The specified file contains logical or value flags.
X'20'	X'000B'	System error.
X'40'	X'000C'	The user address space is too small.
X'01'	X'000D'	The value specified for KEYPOS is invalid.
X'40'	X'000E'	The control block of the file is incorrect.
X'40'	X'000F'	A record in the specified file is too short for the secondary key to be defined
X'40'	X'0010'	There are already 30 secondary keys defined for the file.
X'40'	X'0011'	The file contains incomplete secondary index blocks.
X'40'	X'0012'	The ISAM pool is overloaded.
X'40'	X'0013'	The secondary key has already been defined with other attributes.
X'40'	X'0014'	Interruption via CANCEL.
X'40'	X'0015'	Interruption via BREAK.

X'01'	X'0017'	There was no file specified in the operand list.
X'40'	X'0018'	The file was set to SHARUPD=YES when the macro was called.
X'40'	X0019'	The file link name is invalid.
X'40'	X'001A'	Although DUPKEY=NO was specified, duplicate secondary key values exist in different records.
X'40'	X'001B'	Invalid list element.
X'40'	X'001C'	Invalid number of secondary indices in the list.
X'40'	X'0040'	OPEN error.
X'40'	X'0041'	CLOSE error.
X'40'	X'0042'	An error occurred when writing the secondary index blocks.
X'40'	X'0043'	An error occurred when reading the file.
X'40'	X'0044'	The file is not an NK-ISAM file.
X'40'	X'0081'	A DMS special status occurred when sorting the secondary index entries.
X'40'	X'0082'	An internal error occurred when sorting the secondary index entries.
X'01'	X'FFFF'	Linkage error (function not supported).
X'02'	X'FFFF'	Linkage error (function not available).
X'03'	X'FFFF'	Linkage error (version not supported).

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table in chapter "Standard header".

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the called
- the list is not aligned on a word boundary
- the list is write-protected.

CREPOOL - Create ISAM pool

Macro type: type S (E form/L form/D form/C form); see "Macro types"

The CREPOOL macro creates a task-specific, host-specific, or user ID-specific ISAM pool or links a job to an existing ISAM pool. The ISAM pool is unambiguously identified by the following characteristics:

- its pool name: operand NAME
- its catalog ID: operand CATID
- its scope: operand SCOPE
- its size: operand SIZE
- type of buffering: operand WROUT
- performance characteristic of the ISAM pool: operand RESDNT

The CREPOOL macro may be used only for XS programming (31-bit interface).

Note

Cross-task ISAM pools are created automatically in a data space on a file-specific basis when the file is opened.
SCOPE=USERID and SCOPE=USERGROUP, which were available up to BS2000/OSD V6.0A, are still accepted for reasons of compatibility, but are mapped internally to SCOPE=HOST (cross-task ISAM pool).
For further information on ISAM pools in data spaces please refer to the “Introductory Guide to DMS” [1].

Format

Operation	Operands
CREPOOL	**[,CATID = catid]** **[,MODE = ANY / NEW** **NAME = poolname** **[,SCOPE = TASK / USERID / USERGROUP / HOST]** **[,SIZE = STD / number]** **[,RESDNT = NO / YES]** **[,WROUT = YES/ NO / UNCOND-NO]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = pre]
	MF = C[,PREFIX = pre][,MACID = macid]

Operand descriptions

CATID = catid

Specifies the catalog ID of the pubset to which the ISAM pool is to be assigned. The ISAM pool is created on the host computer to which this pubset belongs. The catalog ID can – as in the file name – be regarded as part of the name, i.e. different catalog IDs identify different ISAM pools.

Default value: the default catalog ID of the calling job.

MACID

Evaluated only in conjunction with MF=C; defines the second through fourth characters of each field name and equate generated in the data area when the macro is expanded.

Default setting: MACID = ISC

= macid

Three-character string defining the second through fourth characters of each field name and equate generated.

MF

The forms of the MF operand are described in detail in the appendix (see "Macro types").

MODE

Specifies, for cross-task ISAM pools, whether the user wants to create a new ISAM pool or whether a link to any existing ISAM pool with the same name and the same catalog ID may be established.

By default, DMS always links the job to an existing ISAM pool with the specified name.

= ANY
If a cross-task ISAM pool with the same name and the same catalog ID has already been created by another task, the job is linked to this pool, even if the value specified for SIZE does not match the actual pool size.
If no such ISAM pool exists, a new pool with the size specified in SIZE is created.

The parameter CREATION-MODE=ANY can be used to set up an **exclusive** connection to an ISAM pool that was created by some other task. This means that if a task issues a CREPOOL with CREATION-MODE=ANY more than once in succession for **one** ISAM pool, the second call (and all others) will be rejected with an error message even if the ISAM pool already exists as a result of the first call. This in turn implies that an **exclusive** ISAM pool can be created within a task only if the pool does not exist for that task.

= NEW
A new cross-task ISAM pool is to be created. If, in this case, a host-specific ISAM pool with the same name and the same catalog ID already exists, the command is rejected with an error message.

NAME = poolname

Assigns a name to the ISAM pool. This, together with the catalog ID and the scope, uniquely identifies the pool.

“poolname” may be 1-8 characters long and may contain all letters and digits and the special characters $, # and @; the first character of “poolname” must be a letter or the special character # or @.

PARAM

Specifies the address of the operand list; evaluated only in conjunction with MF=E (see "Macro types").

= addr
Symbolic address (name) of the operand list.

= (r)
Number of the register containing the address of the operand list. The register must be loaded with this address value before the macro is called.

PREFIX

Evaluated only in conjunction with MF=C or MF=D; defines the first character of each field name and equate generated in the data area when the macro is expanded.

Default setting: PREFIX = D

= pre

Single-character prefix with which the field names and equates generated by the assembler are to begin.

RESDNT

Specifies whether the pages of an ISAM pool are to be maintained in resident working memory (as defined in the function $CSTAT):

= NO
Specifies that the pages of the ISAM pool to be created are not to be maintained in resident memory.

= YES
Specifies that the pages of an ISAM pool are to be maintained in resident memory. A PFA (Performant File Access) privilege is required in order to execute this function.

The call is rejected for existing ISAM pools if there is a conflict between the RESDNT attribute of the pool and the requested RESDNT attribute.

SCOPE

Specifies the scope of the ISAM pool.

All the operand values except TASK are only still supported for reasons of compatibility (see the "Note").

= TASK
The ISAM pool can be used only by the calling job: it is task-local.

= USERID
= USERGROUP
SCOPE=USERID and SCOPE=USERGROUP, which were available up to BS2000/OSD V6.0A, are still accepted for reasons of compatibility, but are mapped internally to SCOPE=HOST (cross-task ISAM pool).

= HOST
The ISAM pool is cross-task and may be used by all jobs.

In the case of SCOPE=HOST, the MODE operand is evaluated. At the same time, SCOPE=HOST affects the WROUT operand: the default value WROUT=YES applies to all files in the ISAM pool and cannot be changed by the user.

SIZE

defines the size of the new ISAM pool to be created.

= STD
The ISAM pool is to be created with the default size specified in the ISAM parameter service.

If the CREATION-MODE=ANY parameter has been specified, the following applies: if the pool is newly created, the SIZE specification is analyzed as described above. If the ISAM pool already exists, the size of the existing ISAM pool is taken over. The specifications for RESDNT and WROUT, however, have to match the attributes of the existing ISAM pools.

= number
defines the size of the ISAM pool to be created in PAM pages:
32 <= num <= 32767 for cross-task ISAM pools
32 <= num <= 8192 for task-local ISAM pools
It is possible that the maximum size set by the system manager for the user address space is the upper limit.
An ISAM pool that is used to buffer files which were created with both BLKCTRL=DATA2K and =DATA4K is dynamically allocated a second extent and thus consists of a 2K extent as well as a 4K extent of the size defined by “num”.

It should be noted that with the minimum size of 32 PAM pages only files of a block size of up to (STD,6) can be processed. For processing files with logical blocks of the size (STD,16), an ISAM pool with 96 PAM pages is required.

WROUT

specifies for the pool whether the changed blocks of a file are to be written to disk immediately:

Default setting:

- WROUT=NO for a task-local ISAM pool (SCOPE=TASK)
- WROUT=YES for a cross-task ISAM pool (SCOPE=USERID/USERGROUP/HOST)
  = YES
  Changed blocks are written to disk immediately, irrespective of the value of the WROUT operand in the FILE or FCB call for the associated file.
  = NO
  specifies that changed blocks do not need to be written to disk immediately. In spite of specifying WROUT=NO at CREPOOL, a changed block is written to disk immediately if
  - WROUT=YES (by means of FILE or FCB or ADD-FILE-LINK) has been specified for the associated file or if
  - SCOPETASK is specified for the pool.

  = UNCOND-NO
  Specifies that updated blocks need not be written back to disk immediately; the restrictions of WROUT=NO do not apply in the following cases:
  - For a cross-task pool (SCOPE=USERID/USERGROUP/HOST), too, updated blocks are not written back to disk immediately;

- OPEN is only carried out for files that are to be opened with SHARUPD=YES in case WRITE-IMMEDIATE=NO or WROUT=NO has been explicitly specified in the associated ADD-FILE-LINK command or FILE macro.

Return codes

The field names and the EQU statements for return codes generated with the C and D forms of the macro start with the string DISC. The first character of this string can be changed by PREFIX, characters 2-4 by MACID.

The return codes are placed in the standard header of the operand list.

Main return code	Meaning
DISCOK X'0000'	The macro was executed successfully.
DISCNPAR X'0001'	Access to the operand list is not possible.
DISCNCAT X'0003'	The catalog ID “catid” is unknown in the system.
DISCNACC X'0004'	There is no link to the pubset “catid”.
DISCINVN X'0005'	The pool name is invalid.
DISCSPAC X'0007'	There is not enough free address space to create a pool (SIZE specification is too large).
DISCPLEX X'0008'	The specified ISAM pool already exists; MODE=NEW was already used for creation by another task; MODE-ANY was already used by the same task
DISCSYSE X'000B'	A system error occurred during macro processing.
DISCSIZE X'000C'	The SIZE specification is invalid.
DISCINVW X'000E'	The WROUT specification is invalid.
DISCINVS X'000F'	The SCOPE specification is invalid.
DISCINVM X'0010'	The MODE specification is invalid.
DISCPRIV X'0011'	Missing privilege with a RESDNT=YES specification
DISCPRES X'0012'	The RESDNT specification in the parameter list and the one for the existing pool are in conflict.
DISCPERR X'0013'	Parameter error.
DISCSPEX X'0014'	Contingent for ISAM pools exceeded.
DISCRLNK X'FFFF'	Macro could not be executed (linkage error): evaluate sub return code 1.

DECFILE - Convert encrypted file into unencrypted file

Macro type: type S (E Form/M Form/L Form/C Form/D Form) (see "Macro types")

The DECFILE macro converts an encrypted file into an unencrypted file (see the ENCFILE macro on "ENCFILE - Convert unencrypted file into encrypted filen").

After DECFILE has run all encryption attributes (procedure and check string for crypto password) are deleted in the catalog entry.

File generations

DECFILE cannot be used for individual file generations but only for complete file generation groups. Within a file generation group, all generations with the exception of tape generations have the same encryption attributes as the group entry.

Format

Operation	Operands
DECFILE	**,PATHNAM=<c-string 1..54: filename 1..54> / <var: char:54>** **,EQUATES=YES / NO** **MF=L**
	MF=D,PREFIX=D / <pre>
	MF=E,PARAM=<name 1..27>

	MF=C / M
	,PREFIX=D / <pre>
	,MACID=MAE / <macid>

Operand descriptions

PATHNAM

Specifies the file which is to be decrypted. The file's crypto password must be contained in the crypto password table of the calling task.

=<c-string 1..54: filename 1..54>
Path name of the file.

=<var: char:54>
Only possible with MF=M:
Symbolic address of a memory area of 54 bytes in which the file's path name is stored.

EQUATES

Specifies whether equates are also to be generated for the values of the parameter area fields when the parameter area is expanded.

= YES
Equates are also generated for the values of the parameter area fields when the parameter area is expanded.

= NO
No equates are generated for the values of the parameter area fields when the parameter area is expanded.

Example

```
:
MVC DECFMFC(YMAD#),DECFMFL
DECFILE MF=M,PREFIX=Y,PATHNAM='UMSATZ.3.QUARTAL.2004'
DECFILE MF=E,PARAM=DECFMFC
:
DECFMFC DECFILE MF=C,PREFIX=Y
DECFMFL DECFILE MF=L
:
```

Programming notes

1. All RESERVED fields of the parameter area have to be deleted with binary zeros.
2. For all changes at parameter level that are not called with macros, the caller is responsible for the consistency of the parameter area.
3. For non-privileged calls (function state TU), register 1 points to the parameter area.
4. For the names listed in the parameter area, no conversion from small to capital letters is performed during the function execution. With the macro expansion, however, a conversion from small to capital letters is possible, depending on the compiler settings.

Notes on function execution

- File locks and file protection attributes which forbid write access to the catalog entry or the content of a file thus also prevent conversion of the file using DECFILE.
- Conversion of a file with DECFILE requires the calling task to have ownership rights for the file. Conversion therefore takes place when:
  - the file is under the user ID of the calling task.
  - the calling task is running under a user ID with TSOS privilege.
  - the user ID of the calling task is co-owner of the file and the file is not temporary.
- Additional functions for tasks with TSOS privilege: If the calling task has TSOS privilege, the following additional functions are possible:
  - Temporary files which do not belong to the calling task but to another task can also be specified.
  - Temporary files can also be specified on a pubset other than the default pubset of the user ID. (These are not deleted automatically when the calling task terminates.)
- Conversion of the encrypted file is logged with SAT.The AUDIT attribute output here is taken from the catalog entry of the file to be converted (see the CREATE-FILE command, AUDIT operand, in the “Commands” manual [3]).
- RFA: DECFILE is rejected if the file to be converted can only be accessed via RFA.

- Help file: When converting with DECFILE a help file is created and then automatically deleted when the function has been completed. The converted file content is written to the help file. The help file needs as much disk storage space as the file to be converted.The file name of the help file has the following format: `S.DMS.<tsn>.<date><time>.CRYPTO`

Return codes

The return code is returned in the standard header of the parameter list. The standard header may not be located in the read-only area, otherwise the program is terminated.

Standard header: `ccbbaaaa`

The following code relating to execution of the DECFILE macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Explanation
X'00'	X'00'	X'0000'	No error
	X'01'	X'0554'	Format of the file name not permitted
	X'01'	X'0576'	a) Incorrect operand combination b) Undeleted UNUSED fields
	X'20'	X'0578'	Internal error when checking the access rights
	X'82'	X'0594'	Not enough virtual memory available
	X'20'	X'05C7'	Internal error in DMS
	X'01'	X'05CB'	Incorrect/inadmissible first file name
	X'40'	X'05CF'	Password not in password table
	X'40'	X'05FD'	File is write-protected
	X'40'	X'0609'	Action not permitted for system file
	X'40'	X'0666'	File protection prevents access
X'01'	X'00'	X'066B'	File is already decrypted
X'00'	X'00'	X'066E'	Use help file
	X'01'	X'FFFF'	Wrong function number in standard header
	X'03'	X'FFFF'	Wrong version number in standard header

DELAIX - Delete secondary key of ISAM file

Macro type: type S (E form/L form/D form/C form); see "Macro types"

The DELAIX macro deletes a selected secondary key or all secondary keys in an NK-ISAM file.

Deleting a secondary key does *not* mean that the values of this key are deleted from the records. Instead, the secondary index blocks belonging to the secondary key(s) are deleted, which means that access to the records via the secondary key(s) is no longer possible.

Format

Operation	Operands
DELAIX	**,KEYNAME = (keyname1[,keyname2,...]) / *ALL** **,FILE = pathname / LINK = linkname** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = pre]
	MF = C[,PREFIX = pre][,MACID = macid]

Operand descriptions

FILE = pathname

Specifies the NK-ISAM file from which the secondary key(s) specified for KEYNAME is/are to be deleted, with: <c-string 1..54: filename 1..54>.

The value specified for the FILE operand is ignored if the LINK operand is also specified.

pathname means [:catid:][$userid.]filename

catid
Catalog ID: if omitted, the default catalog ID for the current user ID is assumed.

userid
User ID: if omitted, the user ID in the SET-LOGON-PARAMETERS or LOGON command is assumed.

filename
Fully qualified file name.

KEYNAME

Specifies which secondary key(s) is/are to be deleted.

= (keyname1[,keyname2,...])
All secondary keys whose names are included in the list are deleted. The secondary keys with the names “keyname1”, “keyname2”, etc. must have been defined for the file specified in the FILE or LINK operand. The user can determine the names and attributes of all secondary keys defined for a file with the aid of the SHOWAIX macro or of the SHOW-INDEX-ATTRIBUTES command.
The parentheses in the list format may be omitted if the list contains only one name.

= *ALL
All secondary keys defined for the file specified in the FILE or LINK operand are deleted.

LINK = linkname

Specifies the link name for the file from which the secondary key(s) specified in the KEYNAME operand is/are to be deleted.
“linkname” may be up to eight characters long. If the file link name is to be addressed via the command interface, it must correspond to the data type <structured_name 1..8> (see the “Commands” manual [3]).

MACID

Defines the second through fourth characters of each field name and equate generated when the macro is expanded.

Default value: MACID = IST

= macid

Three-character string defining the second through fourth characters of the generated field names and equates.

PARAM

Specifies the address of the operand list; it is evaluated only if MF=E applies (see "Macro types").

= addr
Symbolic address (name) of the operand list.

= (r)
Number of the register which contains the address of the operand list. The register must be loaded with this address value before the macro is called.

PREFIX

Defines the first character of each field name and equate generated in the data area when the macro is expanded.

Default value: PREFIX = D

= pre

One-character prefix with which the generated field names and equates are to begin.

Return codes

Standard header: `ccbbaaaa`

The following code relating to execution of the DELAIX macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'bb'	X'aaaa'	Meaning
X'00'	X'0000'	The function was executed successfully.
X'01'	X'0001'	The function could not be executed: the operand list is not available.
X'40'	X'0002'	The function could not be executed: secondary keys are not supported in the remote system (if the macro is called via RFA).
X'40'	X'0003'	The function could not be executed: the specified catalog ID does not exist.
X'40'	X'0004'	The function could not be executed: the catalog cannot be accessed.
X'01'	X'0005'	The function could not be executed: the operand list contains an invalid name.
X'40'	X'0008'	The function could not be executed: the specified secondary key does not exist.
X'20'	X'000B'	The function could not be executed: system error.
X'40'	X'000C'	The function could not be executed: the user address space is too small.
X'40'	X'000E'	The function could not be executed: the control block of the file is errored.
X'40'	X'0012'	The function could not be executed: the ISAM pool is overloaded.
X'40'	X'0016'	The function could not be executed: an invalid number of key names was specified for KEYNAM.
X'01'	X'0017'	The function could not be executed: there was no file specified in the operand list.
X'40'	X'0018'	The function could not be executed: the file was set to SHARUPD=YES when the macro was called.
X'40'	X'0019'	The function could not be executed: the file link name is invalid.
X'40'	X'0040'	The function could not be executed: OPEN error.
X'40'	X'0041'	The function could not be executed: CLOSE error.
X'40'	X'0044'	The function could not be executed: the file is not an NK-ISAM file.

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller

- the list is not aligned on a word boundary
- the list is write-protected.

DELPOOL - Delete/release ISAM pool

Macro type: type S (E form/L form/D form/C form); see "Macro types"

With the aid of the DELPOOL macro, the user can delete ISAM pools he/she has created or clear the link between his/her job and ISAM pools. When the link between an ISAM pool and the last job connected to it is cleared, the ISAM pool is automatically deleted.

If the specified ISAM pool does not exist, the macro is rejected with an error message.

Before an ISAM pool or the link between a job and an ISAM pool can be deleted, all entries in the pool table for the affected ISAM pools must be deleted by means of REMPLNK. If a pool is still linked to a pool table via its link name, DELPOOL is rejected with an error message.

Note

Cross-task ISAM pools which are not task-specific are created automatically in a data space on a file-specific basis when the file is opened and released again when the file is closed.
COPE=USERID and SCOPE=USERGROUP, which were available up to BS2000/OSD V6.0A, are still accepted for reasons of compatibility, but are mapped internally to SCOPE=HOST (cross-task ISAM pool). For further information on ISAM pools in data spaces please refer to the “Introductory Guide to DMS” [1].

Format

Operation	Operands
DELPOOL	**MF = L** **,MODE = SINGLE,NAME = poolname[,CATID = catid]** **[,SCOPE = TASK / USERID / USERGROUP / HOST]** **/** **ALL**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = pre]
	MF = C[,PREFIX = pre][,MACID = macid]

Operand descriptions

MACID

Evaluated only in conjunction with MF=C; defines the second through fourth characters of each field name and equate generated when the macro is expanded.

Default value: MACID = ISD

= macid
Three-character string defining the second through fourth characters of the generated field names and equates.

MF

The forms of the MF operand are described in detail in the appendix ("Macro types").

MODE

Specifies whether only a specific pool or all pools linked to the job are to be released.

MODE = SINGLE
At least one pool name must be specified. Only the ISAM pool identified by NAME, CATID and SCOPE is to be deleted/released.

NAME=poolname
Specifies the name with which the pool was created by means of the CREPOOL macro.

CATID=catid
Specifies the pubset to which the pool was assigned by means of the CREPOOL macro.

Default value: the default catalog ID of the job.

SCOPE
Specifies the scope of the ISAM pool as defined in the CREPOOL macro.

All the operand values except TASK are only supported still for reasons of compatibility (see the note on "DELPOOL - Delete/release ISAM pool").

= TASK
The task-local ISAM pool is deleted or released if there is no pool link name still active for it; otherwise, the macro is aborted with an error message.

= USERID
= USERGROUP
SCOPE=USERID and SCOPE=USERGROUP, which were available up to BS2000/OSD V6.0A, are still accepted for reasons of compatibility, but are mapped internally to SCOPE=HOST (cross-task ISAM pool).

= HOST
The cross-task ISAM pool “poolname” is deleted or released if no further pool link names exist for the job; otherwise, the macro is aborted with an error message.

MODE = ALL
Specifies that all (task-local and cross-task) ISAM pools linked to this job are to be deleted.

PARAM

Specifies the address of the operand list; evaluated only in conjunction with MF=E (see "Macro types").

= addr
Symbolic address (name) of the operand list.

= (r)
Number of the register containing the address of the operand list. The register must be loaded with this address value before the macro is called.

PREFIX

Evaluated only in conjunction with MF=C or MF=D; defines the first character of each field name and equate generated in the data area when the macro is expanded.

Default setting: PREFIX = D

= pre
Single-character prefix with which the field names and equates generated by the assembler are to begin.

Return codes

Unless otherwise specified, the field names and the EQU statements for the return codes generated by the C and D forms of the macro begin with the string DISD. This string can be modified by PREFIX and MACID.

The return codes are placed in the standard header of the operand list.

Main return code	Meaning
DISDOK X'0000'	The macro was executed successfully.
DISDNPAR X'0001'	Access to the operand list is not possible.
DISDNCAT X'0003'	The catalog ID “catid” is unknown in the system.
DISDNACC X'0004'	There is no link to the pubset “catid”.
DISDINVN X'0005'	The pool name is invalid.
DISDNANF X'0006'	There is no ISAM pool with the specified name, catalog ID and scope.
DISDPUSE X'0009'	The task pool table still contains entries with pool link names for this ISAM pool.
DISDSYSE X'000B'	A system error occurred during macro processing.
DISDSPEX X'0014'	Allocation for ISAM pools exceeded.
DISDRNLK X'FFFF'	Macro could not be executed (linkage error): evaluate sub return code 1.

DIV - Access files via virtual address space

Macro type: type S (E form /L form /D form /C form /M form); see "Macro types"

General

All operands can be specified in a DIV macro, regardless of the DIV function (operand FCT). The evaluation of the operands depends upon the selected DIV function.First, all operands are presented in an overview. The format and the operands evaluated for the respective functions are described for each function unit. The format overview is follwed by a summary description of the DIV macro funtions.

Operand values which are neither address nor register entries are referred to as “direct specification” in the operand description.

The operand value “addr” defines a symbolic address that can be stored in an A constant, i.e. the symbolic address must be evaluatable at compile time and must not be included in a DSECT.

The various forms of the MF operand are described in detail in the appendix ("Macro types").

Parameter list

The parameter list of the macro contains a header with fields that can be automatically supplied with values by means of the L form when the parameter list is generated.

If the parameter list is dynamically generated with the D or C forms, it must first be initialized with a parameter list that has been generated with the L form. This is the only way of ensuring that the header of a parameter list is correctly supplied with values.

Wherever fields of a parameter list are referred to in the following description, the names of the parameter list have been indicated as they are generated by MF=D (without a PREFIX specification).

Special parameters in the parameter list

The following parameters are return parameters which can only be accessed directly via the parameter list.

DIVPID

The ID of the OPEN is returned in DIVPID. It must be contained in the parameter list in order to call other DIV functions that belong to the same OPEN. If the same parameter list is used, the DIVPID will have already been entered.

DIVPSIZE

DIVPSIZE returns the logical file size of OPEN in 4096-byte page units. DIVPSIZE contains the number of the last logical 4K page (1 means that the first file page is the last logical page, and 0 means that the file is empty). DIVPSIZE can be evaluated to request memory for a window.

If a file has already been accessed using the UPAM access method, it is possible that the logical end-of-file may not lie on a 4K page boundary. In such cases, DIVPSIZE returns the rounded value. The last half-page before the logical EOF will then appear in the window initialized with X'00'.

Since the file can be logically extended or truncated by SAVE, DIVPSIZE is updated after every successful call to SAVE. DIVPSIZE will then contain the number of the last logical 4K page of the file (1 means that the first page of the file is the last logical page; 0 means that it is empty).

Modifying file characteristics via a command or macro

SHARUPD mode (NO | WEAK | YES) can be changed via the ADD-FILE-LINK command (or via the FILE macro).

OPEN mode (INPUT | INOUT | OUTIN) **cannot** be changed via the ADD-FILE-LINK command or via the FILE macro.

With the ADD-FILE-LINK command, the operands ACCESS-METHOD and BLOCK-CONTROL-INFO must not be specified in a way which contradicts the file structure attributes FILE_STRUC=PAM or BLK-CONTR=PAM. The same applies for the operands FCBTYPE and BLKCRTL of the FILE macro.

Format overview

Operation	Operands
DIV	**,FCT = *OPEN / *CLOSE / *MAP / *UNMAP / *SAVE /** ***RESET / adr / (r)** **[,ID = adr / (r)]** **[,LINK = 'name' / adr / (r)]** **[,FILE = 'name' / adr / (r)]** **[,MODE = *INPUT / *INOUT / *OUTIN / adr / (r)]** **[,SHARUPD = *NO / *WEAK / *YES / adr / (r)]** **[,LOCVIEW = *NONE / *MAP / adr / (r)]** **[,SPID = adr / (r)]** **[,AREA = adr / (r)]** **[,OFFSET = number / adr / *equ / (r)]** **[,SPAN = number / adr / *equ / (r)]** **[,DISPOS = *OBJECT / *UNCHNG / *FRESH / adr / (r)]** **[,PFCOUNT = number / adr* / equ / (r)]** **[,RELEASE = *NO / *YES / adr / (r)]** **[,ENV = *HOST / *XCS / adr / (r)]** **[,LARGE_FILE = *FORBIDDEN / *ALLOWED / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = D / pre]

```
MF = C / M

[,PREFIX = D / pre]

[,MACID = IVP / macid]
```

Function overview

Function	Brief description	See
FCT = *OPEN	Open DIV / PAM file	"DIV function: OPEN"
FCT = *MAP	Create window in address space	"DIV function: MAP"
FCT = *SAVE	Write back modified window pages to the disk file	"DIV function: SAVE"
FCT = *RESET	Undo changes to window pages	"DIV function: RESET"
FCT = *UNMAP	Delete window	"DIV function: UNMAP"
FCT = *CLOSE	Close disk file	"DIV function: CLOSE"

DIV function: OPEN

A file is opened, and an ID is entered in the parameter list. This ID must be used in subsequent calls in order to identify the OPEN association of these calls.

If the same parameter list is used for every DIV call that is associated with a DIV OPEN, the ID will have already been entered into the parameter list and need not be taken into account.
The size of the file is returned in the parameter list after the call.

The OPEN function evaluates only the function operands described below. However, additional operands can be used even at this stage, to prepare the parameter list for other DIV function calls.

Format FCT=*OPEN

Operation	Operands
DIV	**[,FCT = *OPEN / adr / (r)]** **[,LINK = 'name' / adr / (r)]** **[,FILE = 'name' / adr / (r)]** **[,MODE = *INPUT / *INOUT / *OUTIN / adr / (r)]** **[,SHARUPD = *NO / *WEAK / *YES / adr / (r)]** **[,LOCVIEW = *NONE / *MAP / adr / (r)]** **[,ENV = *HOST / *XCS / adr / (r)]** **[,LARGE_FILE = *FORBIDDEN / *ALLOWED / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = D / pre]
	MF = C / M **[,PREFIX = D / pre]** **[,MACID = IVP / macid]**

Operand descriptions

ENV

Influences the compatibility of parallel openers dependent on their execution location (see DIV - Data In Virtual section "Compatibility matrix for DIV-OPEN").

Only direct specification is permitted with the MF=L form.

= *HOST
The maximum possible parallelism is restricted to openers running in the same host.

= *XCS
The openers can run in different hosts in an XCS network without restricting the compatibility (e.g. write operations with SHARUPD=*YES can run in parallel).

= addr
The symbolic address of a one-byte field containing the value of ENV.

= (r)
A register containing the value of ENV.

FCT

Specifies the DIV function to be executed.

Only a direct specification is allowed for the MF=L form.

= *OPEN
A file is opened, and an ID is entered into the parameter list. This ID must be used in subsequent calls in order to indicate the OPEN association of these subsequent calls.

If the same parameter list is used for every DIV call that is associated with a DIV OPEN, this ID will have already been entered into the parameter list and need not be taken into account.

Following the call to OPEN, the OPEN ID and the size of the file will be contained in the `DIVPSIZE` field of the parameter list, respectively.

= addr
Symbolic address of a 1-byte field containing the value for the OPEN function (for more information on the `DIVPOPEN` value, see the layout of the parameter list on "DIV function: CLOSE").

= (r)
Register containing the value for the OPEN function.

FILE

Specifies the name of the file. The FILE specification is not evaluated if a value has been specified for the LINK operand.

Only a direct specification is allowed for the MF=L form.

= 'name'
The name must be enclosed within single quotes.
Length of file name: 1-54 characters (including the catalog ID).

= addr
Symbolic address of a 54-byte field containing the file name.

= (r)
Register containing the address of a 54-byte field with the file name.

LARGE_FILE

The LARGE_FILE operand determines whether or not the file size may grow beyond 32 GB during data processing (see "Files larger than 32 GB"). The operand is entered in the TFT (Task File Table) and is not evaluated until the file is opened with OPEN.

Default value: LARGE_FILE = *FORBIDDEN

In the case of MF=L, only direct specification is permitted.

= FORBIDDEN
The default value means that the specifications in the TU-FCB are to be used.

= ALLOWED
The file is created as a “large file”: the file size may exceed 32 GB.

= addr
The address of an 8-byte field that contains the value for LARGE_FILE.

= (r)
Register containing the address of an 8-byte field with the value for LARGE_FILE.

LINK

Specifies the link name of the file.

The file link name/TFT links the program and file (for information on the file link name/TFT, see the “Introductory Guide to DMS” [1]).

Only a direct specification is allowed for the MF=L form.

= 'name'
If the file link name is enclosed in single quotes “name” may be up to 8 characters long. If the file link name can be addressed via the command interface, it must correspond to the data type <structured_name 1..8> (see the “Commands” manual [3]).

= addr
Address of an 8-byte field containing the file link name.

= (r)
Register which contains the address of an 8-byte field containing the file link name.

LOCVIEW

Operand that specifies whether pages are to be read into a window as soon as MAP is called, or only when the page is accessed.

Default value: LOCVIEW = *NONE

The LOCVIEW operand is only effective for windows created with DISPOS=*OBJECT

Only a direct specification is allowed for the MF=L form.

= *MAP
When a window is defined (function FCT=*MAP), all file pages are read into the window as soon as MAP is called. When the file is being read by a SHARUPD=*WEAK user, DIV prevents the file pages from being updated by a parallel write operation from a SHARUPD=*WEAK user who calls SAVE.

= *NONE
A page will be read from the file into the window when first accessed (default setting).

= addr
Symbolic address of a 1-byte field containing a value for LOCVIEW (`DIVPLNON | DIVPLMAP`; see the layout of the parameter list on "DIV function: CLOSE").

= (r)
Register containing a value for LOCVIEW.

MACID

Defines the second through fourth characters of each field name and equate generated when the macro is expanded.

= IVP
Default value: MACID=IVP

= macid
“macid” is three-character string that defines the second to the fourth character (inclusive) of the generated field names and equates.

MF

The forms of the MF operands are described in detail in the appendix (see "Macro types").

MODE

Specifies OPEN mode (see section "Multiuser operation" in chapter Opening a file):
OPEN mode **cannot** be changed by an ADD-FILE-LINK command (return code DIVPICFS (INCOMPATIBLE_FILE_SPEC)).

Default value: MODE=*INPUT

Only a direct specification is allowed for the MF=L form.

= *INPUT
The file can only be read, so the SAVE function cannot be executed in this OPEN mode.

Parallel INPUT opens are possible, even with the UPAM access method, regardless of the SHARUPD mode.

The file must exist, i.e. must have been opened once with OUTIN.

= *INOUT
The file can be modified, i.e. can be saved with the DIV function SAVE.

The file must exist, i.e. must have been opened once with OUTIN.

Parallel OPEN calls are possible, depending upon the SHARUPD mode.

= *OUTIN
A new file will be created, i.e. the file will be “empty” after the OPEN, and may then be written to. As with MODE=*INOUT, the file can be written to by the SAVE function.

Parallel OPEN calls are possible, depending upon the SHARUPD mode.

In multiuser mode (SHARUPD=*WEAK|*YES), a MODE=*OUTIN user must always be the first to open the file; otherwise, the OPEN call will be rejected.

= addr
Symbolic address of a 1-byte field containing a value for MODE (`DIVPINPT | DIVPINOT | DIVPOUTI`; see layout of the parameter list, "DIV function: CLOSE").

= (r)
Register containing a value for MODE.

PARAM

Indicates the address of the operand list. This operand is only evaluated in conjunction with MF=E (see also "Macro types").

PREFIX

Specifies the first character of each field name and equate generated when the macro is expanded.

= D
Default value: PREFIX=D

= pre
“pre” is a one-character prefix with which the generated field names and equates are to begin.

SHARUPD

Controls parallel access by a number of users (see the section “Multi-user mode on a single computer” in the “Introductory Guide to DMS” [1].

Default value: SHARUPD = *NO

Only a direct specification is allowed for the MF=L form.

= *NO
Allows multiple concurrent reads with MODE=*INPUT **or** one write with MODE=*INOUT | *OUTIN.

= *WEAK
Allows multiple concurrent reads with MODE=*INPUT **and** one write with MODE=*INOUT | *OUTIN.
The data in the window of a read-authorized user (MODE=*INPUT) will therefore depend on the changes that are made by a parallel write-authorized user (MODE=*INOUT | *OUTIN) and on the time at which a page is read into the window. Users with read and write authorization must therefore coordinate their activities. If LOCVIEW=*MAP is specified, the consistency of data in a window is not affected by parallel write operations even if the tasks are not coordinated.

= *YES
Multiple write-authorized users may open a file.

Notes

In this case, data consistency is not maintained by DIV, but must be ensured by the users themselves, e. g. by sequential calls to SAVE.

The file size is checked whenever the allocator is called.
If this check indicates a file size >= 32 GB and the attribute LARGE_FILE=*FORBIDDEN is set in the associated FCB or the attribute EXCEED-32GB=*FORBIDDEN is set in the TFT then processing is canceled. In this case, DIV returns the code **`X'00400030'`** in its local parameter list DIV(I).

The SHARUPD mode can be changed by means of an ADD-FILE-LINK command.

= addr
Symbolic address of a 1-byte field containing a value for SHARUPD (value `DIVPSNO | DIVPSWEA | DIVPSYES`; see layout of the parameter list, "DIV function: CLOSE").

= (r)
Register containing a value for SHARUPD.

Note

DIVPID and DIVPSIZE are return parameters that can only be accessed directly via the parameter list. See subsection "Parameter list" for details.

DIV function: MAP

The MAP function creates a window in an address space (program or data space). A window is assigned to a file region or an entire file.

The address space must be allocated before calling the MAP function (explicitly by REQM, implicitly by the linking loader). The address space in which a window is located cannot be released until the window has been disabled (using UNMAP).

The window region should not include a READ-ONLY page, and at the time of MAP, no I/O fixed page, i.e. no page on which I/O is enabled (e.g. asynchronous I/O by UPAM during the MAP function).

The address space must not be shareable.

When the MAP function is executed, DIV ensures that all pages of the file which are represented by the window are allocated for. If part or all of a window lies beyond the physical end-of-file, the required additional pages are allocated. No allocation is made for a user opening the file in INPUT mode.

The DISPOS operand can be used to specify whether pages of the file are to be displayed in the window, or whether the data in the address space should be retained.

MAP only evaluates the function operands described below.

Format FCT=*MAP

Operation	Operands
DIV	**[,FCT = *MAP / adr / (r)]** **[,ID = adr / (r)]** **[,SPID = adr / (r)]** **[,AREA = adr / (r)]** **[,OFFSET = number / adr / *equ / (r)]** **[,SPAN = number / adr / *equ / (r)]** **[,DISPOS = *OBJECT / *UNCHNG / adr / (r)]** **[,PFCOUNT = number / adr / *equ / (r)]**
	MF = L
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = D / pre]
	MF = C / M **[,PREFIX = D / pre]** **[,MACID = IVP / macid]**

Operand descriptions

AREA

Specifies the starting address of the window within the address space defined by the SPID operand (data or program space).

The address space must be allocated before calling the MAP function (macro REQM, DSPSRV, linking loader) and cannot be released until the window is disabled (UNMAP).

The window must lie on a 4K page boundary. Its length is specified by the SPAN operand.

Each page in the virtual address space may only belong to a **single** window. Once a page is assigned to a window, any request to use it for another window is rejected. In the MF=L form, the starting address of the window can only be specified by a symbolic address.

= addr
Symbolic address of a 4-byte field containing the starting address of the window.

= (r)
Register containing the starting address of the window.

DISPOS

Determines what data should be visible in the window after MAP: unchanged data in the address space (as before MAP), or the data of the corresponding pages of the file.

Default value: DISPOS = *OBJECT

Only a direct specification is allowed for the MF=L form.

= *OBJECT
The file pages appear in the window. Pages behind the last logical page appear filled with X'00'.

= *UNCHNG
The window pages retain their contents and are not replaced by pages from the file.

A window defined with DISPOS=*UNCHNG can be used to initialize the corresponding file pages with the page contents of the virtual address space when SAVE is called (see the SAVE function and the logical extension of filesas described in the “Introductory Guide to DMS” [1]).

DISPOS=*FRESH must not be specified with FCT=*MAP.

= addr
Symbolic address of a 1-byte field containing a value for DISPOS (`DIVPOBJ | DIVPUNCH`; see layout of the parameter list, "DIV function: CLOSE")).

= (r)
Register containing a value for DISPOS.

FCT

Specifies the DIV function to be executed.
Only a direct specification is allowed for the MF=L form.

= *MAP
The MAP function is used to define a window in an address space (program or data space). See "DIV function: MAP" for details.

= addr
Symbolic address of a 1-byte field containing the value for the MAP function (value `DIVMAP`, see the layout of the parameter list on "DIV function: CLOSE").

= (r)
Register containing a value for the MAP function.

ID

Specifies the OPEN for which the DIV function is to be executed.

If the same parameter list is used as in OPEN, the ID need not be specified, since the ID of the OPEN will already be in the parameter list. The ID is contained in the `DIVPID` field of the parameter list.

If a different parameter list is used than the one for OPEN, the ID can be specified here and be transferred to the new parameter list by using the MF=M form of the DIV macro.

ID cannot be specified with the MF=L form.

= addr
Symbolic address of an 8-byte field containing the identification.

= (r)
Register containing the address of the 8-byte field.

MACID

See the description under the format FCT=*OPEN on "DIV function: OPEN".

MF

The forms of the MF operands are described in detail in the appendix ("Macro types").

OFFSET

The operands OFFSET and SPAN specify the file region for which the window is created.

- OFFSET specifies the beginning of the file region. It indicates from which block (i.e. which 4-Kbyte page) the file region begins.
- SPAN defines the number of 4-Kbyte blocks in the file region (i.e. the length of the region).

The file region defined by OFFSET and SPAN is assigned to the window in virtual address space.

Default value: OFFSET = 0

If OFFSET = 0, the first window page corresponds to the first page of the file. The file is read into the window from the beginning of the file up to the length defined by SPAN.

If SPAN is not specified (or SPAN=0), the window size is selected such that the last window page corresponds to the logical last page of the file. If neither OFFSET nor SPAN is defined, an appropriate window size that can accommodate the entire file (until the logical last page) in the window is selected.

If the file is empty and SPAN has been assigned default values, the call will be rejected.

SPAN and OFFSET can be selected such that pages lying beyond the **logical** EOF appear in the window. Pages which follow the logical EOF are displayed in the window filled with X'00'.

SPAN and OFFSET can also be selected such that pages which follow the **physical** EOF appear in the window. If OPEN OUTIN | INOUT is then called, MAP will allocate additional blocks for the file, ending with the file page that corresponds to the last window page.

Note

A file can be physically extended with MAP, and SAVE can be used to extend (or reduce) it logically. The logical EOF is not changed by MAP (only by SAVE).

A file page can be assigned to only **one** window for the same OPEN, but may be assigned to multiple windows if they are part of different OPEN calls.

Only a direct specification is allowed for the MF=L form.

= number
Specifies the first block of the file region to be mapped in virtual address space. The value of OFFSET is limited by the maximum size of a file in 4-KB pages minus 1:

0 <= number <= 8388606 for LARGE_FILE=*FORBIDDEN

0 <= number <= 1073741823 for LARGE_FILE=*ALLOWED

= addr
Symbolic address of a 4-byte field containing the numeric value (binary) of the specification for the first block of the file region to be mapped in virtual address space.

= *equ
Equate representing the numeric value of the specification for the first block of the file region to be mapped in virtual address space. The '*' character must precede the name of the equate.

= (r)
Register containing the numeric value of the specification for the first block of the file region to be mapped in virtual address space.

PARAM

See the description under the format FCT=*OPEN on "DIV function: OPEN".

PREFIX

See the description under the format FCT=*OPEN on "DIV function: OPEN".

PFCOUNT

If pages of a window are accessed sequentially (in ascending order), the number of page fault interrupts can be reduced by specifying a PFCOUNT. If a file page is read into a window as a consequence of a page fault interrupt, and if PFCOUNT has been specified for the window, all following pages are read in a single read operation until the number of pages specified in PFCOUNT, the end of the window, or a previously read page is reached.

Only a direct specification is allowed for the MF=L form.

= number
Specifies how many additional pages are to be read.
0 <= number <= 15

= addr
Symbolic address of a 4-byte field containing a numeric value (binary) for the number of pages.

= *equ
Equate, representing the numeric value. The '*' character must precede the name of the equate.

= (r)
Specifies a register containing the numeric value.

SPAN

The operands OFFSET and SPAN define the file region for which the window is created. The file region specified by SPAN and OFFSET is assigned to the window in virtual address space.

Default value: SPAN = 0

For a description of SPAN see the OFFSET operand.

Only a direct specification is allowed for the MF=L form.

= number
Specifies the length of the file region in 4K blocks. The value for SPAN is restricted to the maximum address space (2 Gbytes) in units of 4K pages.
0 <= number <= 524287

= addr
Symbolic address of a 4-byte field which specifies the length of the data area in 4KB blocks (binary).

= *equ
Equate to define the length of the file region in 4K blocks (binary). The '*' character must precede the name of the equate.

= (r)
Register containing the length of the file region in 4K blocks (binary).

SPID

Specifies the address space (program or data space) in which the window is to be created.

Default value: SPID = 0

If SPID is omitted or SPID = 0 is specified, the window is created in program space; otherwise, SPID defines a data space.

SPID cannot be specified with the MF=L form.

= addr
Symbolic address of an 8-byte field containing the identification of the data space.

= (r)
Register with the address of an 8-byte field containing the identification of the data space.

DIV function: SAVE

The SAVE function writes modified window pages to a file, if they lie in a file region defined by SPAN and OFFSET.

If modified window pages exist in the defined region and these pages lie beyond the logical end-of-file, the file is extended: the last modified page defines the new logical EOF. When a file is extended, unmodified window pages that lie between the previous and the new logical EOF are also written to the file. If no window has been defined for file segments that need to be added, no pages are written to the file for these segments, and the contents of the corresponding file pages are undefined (see also the logical extension of files as described in the “Introductory Guide to DMS” [1]).

If the logical last page of the file is in a window that was defined with DISPOS=*UNCHNG, and if no conditions exist for logical file extension, then the file is truncated if the logical last page was not accessed, i.e. if the logical last page is still in its initial state. Truncation ends when a page which is no longer in the initial state, a previously saved page (with SAVE), or a page which does not belong to a DISPOS=*UNCHNG window is encountered (see also the logical truncation of files as described in the “Introductory Guide to DMS” [1]).

A page that is written to file with SAVE is no longer considered modified, i.e. will not be written to the file by a subsequent SAVE unless it is modified again after the last SAVE.

The SAVE function cannot be used if the file is opened with MODE=*INPUT.

The SAVE function evaluates only the function operands described below.

Format FCT=*SAVE

Operation	Operands
DIV	**[,FCT = *SAVE / adr / (r)]** **[,ID = adr / (r)]** **[,OFFSET = number / adr / *equ / (r)]** **[,SPAN = number / adr / *equ / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = D / pre]
	MF = C / M **[,PREFIX = D / pre]** **[,MACID = IVP / macid]**

Operand descriptions

FCT

Specifies the DIV function to be executed.

Only a direct specification is allowed for the MF=L form.

= *SAVE
The DIV function SAVE writes modified window pages in the file back to disk (see also "DIV function: SAVE").

= addr
Symbolic address of a 1-byte field containing the value for the SAVE function (`DIVPSAVE`; see the layout of the parameter list on "DIV function: CLOSE").

= (r)
Register containing the value for the SAVE function.

ID

Specifies the OPEN for which the DIV function is to be executed.

If the same parameter list is used as in OPEN, the ID need not be specified, since the ID of the OPEN will already be in the parameter list. The ID is contained in the `DIVPID` field of the parameter list.

If a different parameter list is used than the one for OPEN, the ID can be specified here and be transferred to the new parameter list by using the MF=M form of the DIV macro.

ID cannot be specified with the MF=L form.

= addr
Symbolic address of an 8-byte field containing the identification.

= (r)
Register containing the address of the 8-byte field.

MACID

See the description under the format FCT=*OPEN on "DIV function: OPEN".

MF

The forms of the MF operands are described in detail in the appendix ("Macro types").

OFFSET

The operands OFFSET and SPAN specify the file region to which the SAVE function applies.

- OFFSET specifies the beginning of the file region. It indicates from which block (i.e. which 4-Kbyte page) the file region begins.
- SPAN defines the number of 4-Kbyte blocks in the file region (i.e. the length of the region).

The SAVE function applies to all window pages of the file region defined by OFFSET and SPAN.

Default value: OFFSET = 0

If SPAN is omitted or SPAN = 0 is specified, the file region is selected so as to enable the last page of the last window defined for the OPEN to be included in the region. If neither OFFSET nor SPAN is specified, all pages of all windows defined for the OPEN are taken into account.

Only a direct specification is allowed for the MF=L form.

= number
Specifies the first block of the file region to be mapped in virtual address space. The value of OFFSET is limited by the maximum size of a file in 4-KB pages minus 1:

0 <= number <= 8388606 for LARGE_FILE=*FORBIDDEN

0 <= number <= 1073741823 for LARGE_FILE=*ALLOWED

= addr
Symbolic address of a 4-byte field containing the numeric value (binary) of the specification for the first block of the file region to be mapped in virtual address space.

= *equ
Equate representing the numeric value of the specification for the first block of the file region to be mapped in virtual address space. The '*' character must precede the name of the equate.

= (r)
Register containing the numeric value of the specification for the first block of the file region to be mapped in virtual address space.

PARAM

See the description under the format FCT=*OPEN on "DIV function: OPEN".

PREFIX

See the description under the format FCT=*OPEN on "DIV function: OPEN".

SPAN

SPAN and OFFSET define the file region to which the SAVE function applies.

Default value: SPAN = 0

For a description of SPAN see the OFFSET operand.

Only a direct specification is allowed for the MF=L form.

= number
Specifies the first block of the file region to be mapped in virtual address space. The value of SPAN is limited by the maximum size of a file in 4-KB pages:

0 <= number <= 8388607 for LARGE_FILE=*FORBIDDEN

0 <= number <= 1073741824 for LARGE_FILE=*ALLOWED

= addr
Symbolic address of a 4-byte field containing the length of the file region in 4-Kbyte blocks (binary).

= *equ
Equate that specifies the length of the file region in 4-Kbyte blocks (binary). The '*' character must precede the name of the equate.

= (r)
Register containing the length of the file region in 4-Kbyte blocks (binary).

Special parameters in the parameter list

The following parameter is a return parameter that can only be accessed directly via the parameter list. The name of the parameter list generated by means of MF=D (without a PREFIX specification) is specified:

DIVPSIZE

Since the file may be logically extended or truncated by SAVE, DIVPSIZE is updated after every successful call to SAVE and contains the number of the last logical 4K page of the file (1 means that the first file page is the last logical page; 0 means that the file is empty).

DIV function: RESET

The RESET function can be used to undo changes in window pages which belong to a region defined by SPAN and OFFSET.

This is done by setting each modified page to the initial state so that the page will be read from the file when accessed. If the page belongs to a window that is defined with DISPOS=*UNCHNG, it will be read from the file upon access only if it has already been written to the file with SAVE; otherwise, it is initialized with X'00'.

Specifying RELEASE=*YES resets not only unmodified pages, but all pages of the defined region to the initial state. This allows file pages that have been updated by a parallel write-authorized user to be displayed in the window.

The RESET function evaluates only the function operands described below.

Format FCT=*RESET

Operation	Operands
DIV	**[,FCT = *RESET / adr / (r)]** **[,ID = adr / (r)]** **[,OFFSET = number / adr / *equ / (r)]** **[,SPAN = number / adr / *equ / (r)]** **[,RELEASE = *NO / *YES / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = D / pre]
	MF = C / M **[,PREFIX = D / pre]** **[,MACID = IVP / macid]**

Operand descriptions

FCT

Specifies the DIV function to be executed.

Only a direct specification is allowed for the MF=L form.

= *RESET
The DIV function RESET is used to undo changes in window pages that belong to a file region defined by SPAN and OFFSET (for further details, see "DIV function: RESET").

= addr
Symbolic address of a 1-byte field containing the value for the RESET function (field `DIVPRES`, see the layout of the parameter list on "DIV function: CLOSE").

= (r)
Register containing the value for the RESET function.

ID

Specifies the OPEN for which the RESET function is to be executed.

If the same parameter list is used as in the OPEN, the ID need not be specified, since the ID of OPEN will already be in the parameter list.

If a different parameter list is used than the one for OPEN, the ID can be specified here and be transferred to the new parameter list by using the MF=M form of the DIV macro.

ID cannot be specified with the MF=L form.

= addr
Symbolic address of an 8-byte field containing the identification.

= (r)
Register containing the address of the 8-byte field.

MACID

See the description under the format FCT=*OPEN on "DIV function: OPEN".

MF

The forms of the MF operands are described in detail in the appendix ("Macro types").

OFFSET

OFFSET and SPAN specify the file region (in the window) for which window pages are to be reset to their initial state.

- OFFSET specifies the first 4K page of the file region; SPAN specifies the length of that region in 4K pages.
- SPAN defines the number of 4-Kbyte blocks in the file region (i.e. the length of the region).

The RESET function applies to all window pages in the file region defined by SPAN and OFFSET.

Default value: OFFSET = 0

If no value is specified for SPAN (or SPAN = 0), the file region is selected so as to enable the last page of the last window to be included in the region. If neither OFFSET or SPAN is specified, all pages of all windows defined for the OPEN are taken into account.

Only a direct specification is allowed for the MF=L form.

= number
Specifies the first block of the file region to be mapped in 4KB blocks. The value of OFFSET is limited by the maximum size of a file in 4-KB pages minus 1:

0 <= number <= 8388606 for LARGE_FILE=*FORBIDDEN

0 <= number <= 1073741823 for LARGE_FILE=*ALLOWED

= addr
Symbolic address of a 4-byte field containing the length of the file region in 4K blocks (binary).

= *equ
Equate that specifies the length of the file region in 4K blocks (binary). The '*' character must precede the name of the equate.

= (r)
Register containing the length of the file region in 4K blocks (binary).

PARAM

See the description under the format FCT=*OPEN on "DIV function: OPEN".

PREFIX

See the description under the format FCT=*OPEN on "DIV function: OPEN".

RELEASE

Defines whether only modified pages are to be reset to the initial state.

Default value: RELEASE=*NO

Only a direct specification is allowed for the MF=L form.

= *NO
All modified pages are returned to the initial state.

As a result, when such a page is accessed, the corresponding page is read in from the file if the page lies in a DISPOS=*OBJECT window. If the page lies in a window that is defined with DISPOS=*UNCHNG, it is initialized with X'00' upon access if it has not yet been written to file with SAVE. Otherwise, it is read from the file when accessed.

Pages in the initial state that follow the logical last page of the file are always initialized with X'00' when accessed.

= *YES
All window pages – both modified and unmodified – in the specified region are reset to the initial state, with the consequences described above.

= addr
Symbolic address of a 1-byte field containing a value for RELEASE (`DIVPRNO | DIVPRYES`; see the layout of the parameter list on "DIV function: CLOSE").

= (r)
Register containing a value for RELEASE.

SPAN

SPAN and OFFSET define the file region to which the RESET function applies.

Default value: SPAN = 0

For a description of SPAN see the description of the OFFSET operand.

Only a direct specification is allowed for the MF=L form.

= number
Specifies the first block of the file region to be mapped in virtual address space. The value of SPAN is limited by the maximum size of a file in 4-KB pages:

0 <= number <= 8388607 for LARGE_FILE=*FORBIDDEN

0 <= number <= 1073741824 for LARGE_FILE=*ALLOWED

= addr
Symbolic address of a 4-byte field which defines the length of the file region in 4K blocks (binary).

= *equ
Equate to define the length of the file region in 4K blocks (binary). The '*' character must precede the name of the equate.

= (r)
Register containing the length of the file region in 4K blocks (binary).

DIV function: UNMAP

The UNMAP function disables a window created by MAP.

UNMAP does not result in any changes to file pages.

The state of the unmapped window pages is determined by the DISPOS operand:

- If DISPOS=*UNCHNG is specified, the pages in the window will have the same contents before and after UNMAP, from the program's point of view. This means that all pages that would be read from the file if accessed prior to UNMAP must be read in from the file at the time of UNMAP, since no data can be read from the file after the window is disabled.
- If DISPOS=*FRESH is specified, all pages of the window will be reset to the initial state. If accessed after UNMAP, they will appear in the window initialized with X'00'.

UNMAP only evaluates the function operands described below.

Format FCT=*UNMAP

Operation	Operands
DIV	**[,FCT = *UNMAP / adr / (r)]** **[,SPID = adr / (r)]** **[,AREA = adr(/ r)]** **[,DISPOS = *FRESH / *UNCHNG / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = D / pre]
	MF = C / M **[,PREFIX = D / pre]** **[,MACID = IVP / macid]**

Operand descriptions

AREA

Specifies the starting address of the window within the address space defined by SPID (data space or program space if no data space is defined).

In the MF=L form, the starting address of the window can only be specified by a symbolic address.

= addr
4-byte (symbolic) starting address of the window.

= (r)
Register containing the starting address of the window.

DISPOS

Determines the contents (state) of the window pages on disabling the window:

Default value: DISPOS = *FRESH

Only a direct specification is allowed for the MF=L form.

= *UNCHNG
The pages appear as they were before the window was disabled.

This variant could have an adverse effect on performance if pages which have never been accessed since UNMAP are read from the file.

= *FRESH
All pages of the window are reset to their initial state. They appear initialized with X'00'.

If a parameter list that was used by a preceding *MAP function is used for FCT=*UNMAP, it should be noted that a value will have already been entered for the DISPOS operand. This value may need to be redefined, since DISPOS=*OBJECT is an illegal value for UNMAP.

= addr
Symbolic address of a 1-byte field containing a value for DISPOS (`DIVPFRSH | DIVPUNCH`; see parameter list, "DIV function: CLOSE").

= (r)
Register containing a value for DISPOS.

FCT

Specifies the DIV function to be executed.

= *UNMAP
The UNMAP function disables a window created by MAP (for a detailed description of the function, see "DIV function: UNMAP").

Only a direct specification is allowed for the MF=L form.

= addr
Symbolic address of a 1-byte field containing the value for the UNMAP function (field `DIVUNMP`, see the layout of the parameter list on "DIV function: CLOSE").

= (r)
Register containing the value for the UNMAP function.

MACID

See the description under the format FCT=*OPEN on "DIV function: OPEN".

MF

The forms of the MF operands are described in detail in the appendix ("Macro types").

PARAM

See the description under the format FCT=*OPEN on "DIV function: OPEN".

PREFIX

See the description under the format FCT=*OPEN on "DIV function: OPEN".

SPID

Specifies the address space (program or data space) in which the window is located.

SPID cannot be specified with the MF=L form.

= addr
Symbolic address of an 8-byte field containing the identification of the address space.

= (r)
Register with the address of an 8-byte field containing the identification of the address space (program or data space).

DIV function: CLOSE

The CLOSE function closes a file.

If windows that were defined for the OPEN still exist, they are disabled using default values for the operands.

The CLOSE function only evaluates the function operands described below.

Format FCT=*CLOSE

Operation	Operands
DIV	**[,FCT = *CLOSE / adr / (r)]** **[,ID = adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = D / pre]
	MF = C / M **[,PREFIX = D / pre]** **[,MACID = IVP / macid]**

Operand descriptions

FCT

Specifies the DIV function to be executed.

= *CLOSE
The CLOSE function closes the file.

Only a direct specification is allowed for the MF=L form.

= addr
Symbolic address of a 1-byte field containing the value for the CLOSE function (field `DIVPCLS`; see the layout of the parameter list on "DIV function: CLOSE").

= (r)
Register containing the value for the CLOSE function.

ID

Specifies the OPEN for which the CLOSE function is to be executed.

If the same parameter list is used as in OPEN, the ID need not be specified, since the ID of the OPEN will already be in the parameter list. The ID is contained in the `DIVPID` field of the parameter list.

If a different parameter list is used than the one for OPEN, the ID can be specified here and be transferred to the new parameter list by using the MF=M form of the DIV macro.

ID cannot be specified with the MF=L form.

= addr
Symbolic address of an 8-byte field containing the identification.

= (r)
Register with the address of an 8-byte field containing the identification.

MACID

See the description under the format FCT=*OPEN on "DIV function: OPEN".

MF

The forms of the MF operands are described in detail in the appendix ("Macro types").

PARAM

See the description under the format FCT=*OPEN on "DIV function: OPEN".

PREFIX

See the description under the format FCT=*OPEN on "DIV function: OPEN".

Return codes

Return codes are placed in the header of the parameter list. All DIV-specific return codes are explained in the table below. Other return codes and their meanings as well as the structure of the default header are defined by conventions applicable to all macros and are described on "Standard header".

Standard header: `ccbbaaaa`

The following code relating to execution of the DIV macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	Function executed successfully.
	X'01'	X'0001'	At least one part of the parameter list is not accessible. If the header of the parameter list (or a portion thereof) cannot be accessed, or if the parameter list is not aligned on a word boundary, the program will be aborted with an error message.
	X'01'	X'0002'	The window address specified by AREA (in MAP or UNMAP) is not aligned on a 4K page boundary.
	X'01'	X'0003'	For MAP, SAVE and RESET: the size of the region (SPAN, possibly in conjunction with the specified OFFSET) results in a disk address that is too large. Possible cause with MAP: the window size (SPAN) is greater than 2 Gbytes.
	X'01'	X'0004'	For MAP, SAVE and RESET: the OFFSET value corresponds to a disk address that is too large.
	X'01'	X'0005'	For MAP: the DISPOS value is neither *OBJECT nor *UNCHNG. For UNMAP: the DISPOS value is neither *FRESH nor *UNCHNG.

X'01'	X'0006'	The value for PFCOUNT is not in the range of 0 to 15 (MAP).
X'01'	X'0007'	The value for RELEASE is neither *NO nor *YES (RESET).

X'01'	X'0008'	The value for MODE (OPEN) is neither *INPUT, nor *INOUT or *OUTIN.
X'01'	X'0009'	The value for SHARUPD (OPEN) is neither *NO, nor *WEAK or *YES.
X'01'	X'000A'	The value for LOCVIEW (OPEN) is neither *NONE nor *MAP.
X'01'	X'000B'	The value for ENV (OPEN) is neither *HOST nor *XCS.
X'01'	X'000C'	The value for LARGE_FILE (OPEN) is neither *ALLOWED nor *FORBIDDEN.
X'01'	X'000D'	DIV is not supported on SPARC-HSI.
X'40'	X'0014'	DIV calls from TPR are not permitted.
X'40'	X'0015'	An error was detected during (general) OPEN handling by DMS. The DIVPDMSC field contains the DMS error code.
X'40'	X'0016'	An error was detected during (general) CLOSE handling by DMS. The DIVPDMSC field contains the DMS error code.
X'40'	X'0017'	The file is already open. The OPEN request is rejected because the MODE value and/or the SHARUPD value of the current OPEN request, and the existing values under which the file is opened do not permit parallel processing.
X'40'	X'0018'	The OPEN request is rejected due to one of the following situations: • An OPEN mode specified in the ADD-FILE-LINK command is not supported by DIV (e.g. OUTPUT). • The OPEN mode specified in the ADD-FILE-LINK command differs from the one specified by the program. • ACCESS-METHOD=*UPAM was not specified in the ADD-FILE-LINK command. • An impermissible value for ACCESS-METHOD was specified in the ADD-FILE-LINK command. • The BUFFER-LENGTH operand in the ADD-FILE-LINK command was explicitly set to a value not equal to 2 in the case of a new file (MODE=*OUTIN). • The operand BLOCK-CONTROL-INFO was not specified correctly in the ADD-FILE-LINK command, or an attempt was made to open an existing file that does not have the attribute BLKCTRL=NO.
X'40'	X'0019'	Neither a link name nor a file name is specified (OPEN).
X'40'	X'001A'	The file to be opened is located on a shared private disk (SPD). Files on shared private disks are not processed by DIV.
X'40'	X'001B'	A window with the DISPOS=*UNCHNG attribute is not permitted for a file opened with MODE=*INPUT (MAP).
X'40'	X'001C'	The privileges of the user (USER or SYSTEM) who opened the file are not the same as those of the user calling the MAP, UNMAP, SAVE, RESET or CLOSE function.

X'40'	X'001D'	The ID supplied with one of the functions MAP, UNMAP, RESET or CLOSE to identify the OPEN is not known (any longer) by DIV. The parameter list is possibly different from that of the OPEN and the ID was not indicated, or the file has already been closed.
X'40'	X'001E'	SPID was specified but does not indicate a data space for the window, or indicates a data space that the caller is not permitted to access (MAP, UNMAP).
X'40'	X'001F'	At least part of the address space specified for a window is already being used by an existing window (MAP).
X'40'	X'0020'	At least part of the file region specified for a window is already mapped in another window of the opener (MAP).
X'40'	X'0021'	SPAN is not specified (or SPAN=0), and cannot be determined by DIV for window definition (MAP) because either • the file is empty or • the OFFSET points beyond the logical last page.
X'40'	X'0022'	The window area contains multiple memory classes (e.g. class 5 and class 6 memory) (MAP).
X'40'	X'0023'	A page in the virtual address space has been requested as a window page, but that page has been fixed for an I/O (MAP).
X'40'	X'0024'	A page in virtual address space that is intended for a window is resident (MAP).
X'40'	X'0025'	A page in virtual address space that is intended for a window is marked READ-ONLY (MAP).
X'40'	X'0026'	At least part of the address space that is defined for a window is not allocated (e.g. a REQM was not executed) (MAP).
X'40'	X'0027'	The address space specified for a window is shareable (MAP).
X'40'	X'0028'	The address space specified for a window is not accessible to nonprivileged users and the caller is not privileged (MAP).
X'40'	X'0029'	An internal DIV table cannot be created due to insufficient user address space (MAP).
X'40'	X'002A'	The file cannot be physically extended (MAP) because • either no more disk space is available to the user or • a secondary allocation has not been defined for the file (operand SPACE... SEC-ALLOC in the CREATE-FILE or MODIFY-FILE-ATTRIBUTES command).
X'40'	X'002B'	An error occurred when reading a block (MAP, UNMAP).
X'40'	X'002C'	An error occurred when writing a block (SAVE).

X'40'	X'002D'	The window defined by AREA (and SPID) does not exist for the current OPEN (UNMAP).
X'40'	X'002E'	No window exists for the file region defined by OFFSET and SPAN (SAVE, RESET).
X'40'	X'002F'	Read-authorized users are not allowed to call SAVE.

X'40'	X'0030'	On file access in mode SHARUPD=YES, it was detected that the file size exceeds the value of 32 GB even though there is no permission to exceed this value when using OPEN together with this file.
X'80'	X'003C'	The DIV subsystem was stopped by a command. Subsequent OPEN requests will be rejected.
X'80'	X'003E'	The system address space required to create internal DIV tables is not available (OPEN, MAP).
X'20'	X'0046'	Internal DIV error.
X'20'	X'0047'	Possibly due to an internal DIV error, a wait for the release of a locked system resource was unsuccessful.

Explanations for the return codes

Return codes are placed in the header of the parameter list:

- The main return code is stored in a half-word with the name DIVPMRET.
- Subcode1 is stored in a byte with the name DIVPSR1.Subcode1 describes error classes, which allow the caller to respond to them (see table on "Standard header"). The caller can refer both to the main code as well as to subcode1.
- Subcode2 is currently not used. It is always zero (X'00').

Return codes cannot be placed in the header if:

- the list is not assigned to the user
- the list is not aligned on a word boundary
- the list is write-protected.

The calling program is aborted with an error message (see section "DMS error codes") and the STXIT event for a “non-recoverable program error” is generated in such cases.

The field names generated by the C or D forms of the macro and EQU instructions for the return codes begin with the string DIVP by default, and can be changed using PREFIX and MACID.

Layout of the parameter list

The following parameter list is returned by a DIV macro with MF=D:

```
           DIV   MF=D
1          MFCHK MF=D,PREFIX=D,MACID=IVP,PARAM=,
1                SVC=126,DMACID=IVP,DNAME=IVPLIST,SUPPORT=(C,D,E,L,M)
2 DIVPLIST DSECT ,
2                *,##### PREFIX=D, MACID=IVP #####
1          #INTF REFTYPE=REQUEST,INTNAME=DIV,INTCOMP=002
1 *
1 DIVPPA   DS    0F          BEGIN of PARAMETERAREA
1          FHDR  MF=(C,DIVP),EQUATES=YES
2          DS    0A
2 DIVPFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
```

```
2 DIVPIFID DS    0A              0   INTERFACE IDENTIFIER
2 DIVPFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 DIVPFCT  DS    AL1             2   FUNCTION NUMBER
2 DIVPFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 DIVPRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 DIVPSRET DS    0AL2            4   SUB RETURN CODE
2 DIVPSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 DIVPR2OK EQU   X'00'               All correct, no additional info
2 DIVPR2NA EQU   X'01'               Successful, no action was necessary
2 DIVPR2WA EQU   X'02'               Warning, particular situation
2 DIVPSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 DIVPRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
2 DIVPRPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 DIVPRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
2 DIVPRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 DIVPRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 DIVPRAER EQU   X'04'               ALIGNMENT ERROR
2 DIVPRIER EQU   X'20'               INTERNAL ERROR
2 DIVPRCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 DIVPRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 DIVPRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 DIVPRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 DIVPRWLR EQU   X'81'                    "     LONG       "
2 DIVPRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 DIVPRTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
2 DIVPRDH  EQU   X'82'               SS IN DELETE / HOLD
2 *
2 DIVPMRET DS    0AL2            6   MAIN RETURN CODE
2 DIVPMR2  DS    AL1             6   MAIN RETURN CODE 2
2 DIVPMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
```

```
2 *
2 DIVPRLNK EQU   X'FFFF'             LINKAGE ERROR / REQ. NOT PROCESSED
2 DIVPFHL  EQU   8                8  GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *                          DIV-FUNCTIONS:
1 DIVPOPEN EQU   1           OPEN
1 DIVPCLS  EQU   2           CLOSE
1 DIVPMAP  EQU   3           MAP
1 DIVPUNMP EQU   4           UNMAP
1 DIVPSAVE EQU   5           SAVE
1 DIVPRES  EQU   6           RESET
1 *                          DIV-MAINCODES:
1 DIVPPNAC EQU   1           PARLIST NOT ACCESSIBLE
1 DIVPIWAD EQU   2           INVALID WINDOW ADDRESS
1 DIVPISPA EQU   3           INVALID SPAN
1 DIVPIOFF EQU   4           INVALID OFFSET
1 DIVPIDSP EQU   5           INVALID DISPOS
1 DIVPIPFC EQU   6           INVALID PFCOUNT
1 DIVPIREL EQU   7           INVALID RELEASE
1 DIVPIOM  EQU   8           INVALID OPEN MODE
1 DIVPISUM EQU   9           INVALID SHARUPD MODE
1 DIVPILVM EQU   10          INVALID LOCVIEW MODE
1 DIVPIENV EQU   11          INVALID LOCKENV
1 DIVPILRF EQU   12          INVALID LARGE_FILE
1 DIVPNSPA EQU   13          DIV NOT SUPPORTED ON SPARC-HSI
1 DIVPPRVC EQU   20          PRIVILEGED DIV CALL
1 DIVPDOER EQU   21          DMS OPEN ERROR
1 DIVPDCER EQU   22          DMS CLOSE ERROR
1 DIVPICOM EQU   23          INCOMPATIBLE OPEN MODE
1 DIVPICFS EQU   24          INCOMPATIBLE FILE SPEC
1 DIVPNLNF EQU   25          NEITHER LINK NOR FILE
1 DIVPPD   EQU   26          PRIVATE DISK
1 DIVPDOI  EQU   27          DISPOS OPEN INCONSISTENCY
1 DIVPPRVI EQU   28          PRIV INCONSISTENCY
1 DIVPWRID EQU   29          WRONG ID
1 DIVPSIDU EQU   30          SPID UNDEFINED
1 DIVPSPOV EQU   31          SPACE OVERLAP
1 DIVPFOV  EQU   32          FILE OVERLAP
1 DIVPUSNP EQU   33          UNDEF SPAN NOT POSSIBLE
1 DIVPINHM EQU   34          INHOMOG MEM
1 DIVPPFIX EQU   35          PAGE FIXED
1 DIVPRESP EQU   36          RESIDENT PAGE
1 DIVPROP  EQU   37          READ ONLY PAGE
1 DIVPMNA  EQU   38          MEM NOT ALLOC
1 DIVPSHRM EQU   39          SHARABLE MEMORY
1 DIVPPRSP EQU   40          PRIVILEGED SPACE
1 DIVPNOAS EQU   41          NO ADDRESS SPACE
1 DIVPALER EQU   42          ALLOC ERROR
1 DIVPRDER EQU   43          READ ERROR
1 DIVPWRER EQU   44          WRITE ERROR
1 DIVPWNF  EQU   45          WINDOW NOT FOUND
1 DIVPNWIR EQU   46          NO WINDOW IN RANGE
1 DIVPSAVN EQU   47          SAVE NOT ALLOWED
1 DIVPLFNS EQU   48          LARGE_FILE NOT SPECIFIED
1 DIVPSSS  EQU   60          SUBSYSTEM STOPPED
1 DIVPSOR  EQU   61          SHORTAGE OF RESOURCES
1 DIVPIERR EQU   70          INTERNAL ERROR
1 DIVPTOUT EQU   71          TIME RUNOUT
1 *
```

```
1 DIVPID   DS    XL8         ID
1 DIVPLARF DS    AL1         LARGE_FILE
1 DIVPFRBD EQU   0           LARGE_FILE=FORBIDDEN
1 DIVPALWD EQU   1           LARGE_FILE=ALLOWED
1 DIVPUNUS DS    AL1         UNUSED BYTE
1 DIVPDMSC DS    H           DMS-CODE
1 DIVPLINK DS    CL8         LINK
1 DIVPFILE DS    CL54        FILE
1 DIVPOMOD DS    AL1         MODE
1 DIVPINPT EQU   1           MODE=INPUT
1 DIVPINOT EQU   2           MODE=INOUT
1 DIVPOUTI EQU   3           MODE=OUTIN
1 DIVPSUPD DS    AL1         SHARUPD
1 DIVPSNO  EQU   1           SHARUPD=NO
1 DIVPSWEA EQU   2           SHARUPD=WEAK
1 DIVPSYES EQU   3           SHARUPD=YES
1 DIVPLOCV DS    AL1         LOCVIEW
1 DIVPLNON EQU   1           LOCVIEW=NONE
1 DIVPLMAP EQU   2           LOCVIEW=MAP
1 DIVPDISP DS    AL1         DISPOS
1 DIVPOBJ  EQU   1           DISPOS=OBJECT
1 DIVPUNCH EQU   2           DISPOS=UNCHNG
1 DIVPFRSH EQU   3           DISPOS=FRESH
1 DIVPREL  DS    AL1         RELEASE
1 DIVPRNO  EQU   1           RELEASE=NO
1 DIVPRYES EQU   2           RELEASE=YES
1 DIVPLENV DS    AL1         LOCKENV
1 DIVPHOST EQU   1           LOCKENV=HOST
1 DIVPXCS  EQU   2           LOCKENV=XCS
1 DIVPSIZE DS    F           SIZE
1 DIVPSPID DS    XL8         SPID
1 DIVPAREA DS    A           AREA
1 DIVPOFFS DS    F           OFFSET
1 DIVPSPAN DS    F           SPAN
1 DIVPPFC  DS    F           PFCOUNT
1 DIVP#    EQU   (*-DIVPPA)  LENGTH OF STRUCTURE
```

Examples

Example 1: Reading and updating an existing file

```
*-------------------------------------------------------------------*
* Example 1: Reading and updating an existing file                  *
*-------------------------------------------------------------------*
D1       DIV     MF=D
*
BSP001   START
         BALR  R10,0
         USING *,R10
         USING D1,R9
         LA    R9,PA
*-------------------------------------------------------------------*
* Open a file with INOUT                                            *
*-------------------------------------------------------------------*
DIV   MF=E,PARAM=PA
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
```

```
*--------------------------------------------------------------------*
* Determine the file size and request pages for a window.            *
* REQM supplies the starting address in R1.                          *
*--------------------------------------------------------------------*
L     R3,DIVPSIZE        FILE SIZE IN 4K PAGES -> R3
         REQM  (R3),PARMOD=31
         LTR   R15,R15
         BNZ   ERROR
         LR    R8,R1
*--------------------------------------------------------------------*
* Create a window in which the pages of the file appear only after   *
* an access (DISPOS=*OBJECT and LOCVIEW=*NONE are default settings). *
* A window of the same size as the file is created (OFFSET and SPAN  *
* are not specified).                                                *
*--------------------------------------------------------------------*
DIV   MF=M,PARAM=PA,FCT=*MAP,AREA=(R8)
         DIV   MF=E,PARAM=PA
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*--------------------------------------------------------------------*
* Write modified window pages to file.                               *
*--------------------------------------------------------------------*
DIV   MF=M,PARAM=PA,FCT=*SAVE
         DIV   MF=E,PARAM=PA
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
* Disable a window and close the file.                               *
* After the window is disabled, the pages are in the same state      *
* as just before REQM.                                               *
*--------------------------------------------------------------------*
*
         DIV   MF=M,PARAM=PA,FCT=*UNMAP,DISPOS=*FRESH
         DIV   MF=E,PARAM=PA
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*
         DIV   MF=M,PARAM=PA,FCT=*CLOSE
         DIV   MF=E,PARAM=PA
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*
*
ERROR    DS   0Y
*
*
PA       DIV   MF=L,FCT=*OPEN,LINK='TST001',MODE=*INOUT
         END
```

Example 2: Copying and modifying a file

```
*--------------------------------------------------------------------*
* Example 2: Copying and modifying a file.                           *
*--------------------------------------------------------------------*
*
D1       DIV     MF=D
*
BSP002   START
```

```
         BALR  R10,0
         USING *,R10
         USING D1,R9
         LA    R9,PA1
*------------------------------------------------------------------*
* Open an existing file.                                           *
*------------------------------------------------------------------*
DIV   MF=E,PARAM=PA1
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*------------------------------------------------------------------*
* Determine the file size and request pages for a window           *
* REQM supplies the starting address in R1.                        *
*------------------------------------------------------------------*
L     R3,DIVPSIZE        FILE SIZE IN 4K PAGES -> R3
         REQM  (R3),PARMOD=31
         LTR   R15,R15
         BNZ   ERROR
         LR    R8,R1
*------------------------------------------------------------------*
* Create a window. The pages of the file are immediately read into *
* the window (file pages appear in the window because DISPOS=*OBJECT *
* (default); the pages are read immediately because LOCVIEW=*MAP).  *
* A window of the same size as the file is created (OFFSET and SPAN *
* are not specified).                                              *
*------------------------------------------------------------------*
DIV   MF=M,PARAM=PA1,FCT=*MAP,AREA=(R8)
         DIV   MF=E,PARAM=PA1
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*------------------------------------------------------------------*
* Modify window pages.                                             *
*------------------------------------------------------------------*
*
*
*------------------------------------------------------------------*
* Disable the window and close the file.                           *
* The contents of the window pages are retained (DISPOS=*UNCHNG).  *
*------------------------------------------------------------------*
*
         DIV   MF=M,PARAM=PA1,FCT=*UNMAP,DISPOS=*UNCHNG
         DIV   MF=E,PARAM=PA1
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*
         DIV   MF=M,PARAM=PA1,FCT=*CLOSE
         DIV   MF=E,PARAM=PA1
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*
*------------------------------------------------------------------*
* Open the new file.                                               *
*------------------------------------------------------------------*
LA    R9,PA2
         DIV   MF=E,PARAM=PA2
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
* Create a window. Data remains unchanged (because DISPOS=*UNCHNG).  *
* The address of the region is still in R8; the window size in R3.   *
```

```
*-------------------------------------------------------------------*
DIV   MF=M,PARAM=PA2,FCT=*MAP,AREA=(R8),DISPOS=*UNCHNG,SPAN=(R3)
         DIV   MF=E,PARAM=PA2
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*-------------------------------------------------------------------*
* Write window pages to the new file.                               *
*-------------------------------------------------------------------*
DIV   MF=M,PARAM=PA2,FCT=*SAVE
         DIV   MF=E,PARAM=PA2
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*-------------------------------------------------------------------*
* Delete the window and close the file.                             *
*-------------------------------------------------------------------*
DIV   MF=M,PARAM=PA2,FCT=*UNMAP,DISPOS=*FRESH
         DIV   MF=E,PARAM=PA2
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*
         DIV   MF=M,PARAM=PA2,FCT=*CLOSE
         DIV   MF=E,PARAM=PA2
         CLI   DIVPSR1,DIVPRFSP
         BNE   ERROR
*
*
ERROR    DS   0Y
*
*
PA1      DIV   MF=L,FCT=*OPEN,LINK='TST001',LOCVIEW=*MAP
PA2      DIV   MF=L,FCT=*OPEN,LINK='TST002',MODE=*OUTIN
         END
```

DROPTFT - Release TFT entry

Macro type: type S (C form/D form/E form/L form/M form); see "Macro types"

The DROPTFT macro releases a LOCK-FILE-LINK lock for an entry in the task file table (TFT). If a REMOVE-FILE-LINK command or RELTFT macro call is still pending for this entry, it is processed now, i.e. the TFT entry is deleted according to the command/macro parameters and the private devices connected to it are released.

Format

Operation	Operands

DROPTFT	**,LINK = <c-string 1..8> / <var: char:8>**
	,VERSION = <integer 1..1>
	,MF = C / D / E / L / M
	,PARAM = <addr> / <(r)>
	,PREFIX = D / <pre>
	,MACID = MAD / <macid>

Operand descriptions

LINK

File link name of the TFT entry to be released.

Default value: The first TFT entry with the link name *BLANK is released.

= <c-string 1..8>

File link name (specified in quotes).

= <var: char: 8>
Name of a variable that contains the file link name.

MACID
Only evaluated with MF=C/D/M; this defines the second, third and fourth characters of the field names and equates that are generated in the data area when the macro is expanded.

Default value: MACID = MAD

= macid

“macid” is a three-character string that defines the second, third and fourth characters of the generated field names and equates.

MF
The forms of the MF operand are described in detail in the appendix ("Macro types").

Default value: Operand list and SVC as previously

PARAM

Designates the address of the operand list and is only evaluated in conjunction with MF=E (see also "Macro types").

= addr
The symbolic address (name) of the operand list.

= (r)
The number of the register containing the address of the operand list. The register must be loaded with this address value before calling the macro.

PREFIX

Only evaluated with MF=C/D/M; this defines the first character of the field names and equates that are generated in the data area when the macro is expanded.

Default value: PREFIX = D

= pre

Single-character prefix with which the field names and equates generated by the assembler should begin.

VERSION = <integer 1..1>
Control operand; controls generation.

Programming notes

The error is returned in the standard header of the parameter area. Program termination with STXIT can be initiated in the following cases:

- parameter address incorrect (e.g. shorter than the standard header)
- parameter address not aligned on a word boundary
- UNIT or FUNCTION in header incorrect
- header is write-protected

Return codes

Standard header: `ccbbaaaa`

The following code relating to execution of the DROPTFT macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
X'02'	X'00'	X'0662'	Link name missing or invalid. No action necessary
	X'40'	X'06FF'	BCAM connection not operational or is closed
	X'01'	X'xxxx'	RFA not supported for version < 12

EAM - Process EAM files

Macro type: R type

All processing requests addressed to the EAM access method are handled via the EAM macro. The operation to be performed is determined by the contents of the MFCB.

Format

Operation	Operands
EAM	**mfcbadr / (1)** **[,PARMOD=24 / 31]**

Operand descriptions

mfcbaddr

Address of the MFCB.

(1)
The address of the MFCB is stored in register 1.

PARMOD
Specifies the generation mode for the macro.

Default value: The value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The macro is expanded in accordance with the 24-bit interface format. The object code can thus run only in 24-bit addressing mode.

= 31
The macro is generated as addressing mode-independent.

Programming note

The EAM macro overwrites the contents of registers 0, 1, 14 and 15.

Return codes

If PARMOD=24, the return code is placed in register 15; for PARMOD=31, it is placed in the IDMRETCO field in the MFCB.

Return code	Meaning
0	Operation completed successfully
4	Operation not completed successfully; check sense byte (IDMFEB)
8	After check operation: checked I/O operation not yet terminated

ELIM - Eliminate record

Macro type: R for PARMOD=24
O for PARMOD=31

The ELIM macro eliminates (deletes) a record from an ISAM file. The second operand (LAST/KEY/(0)) indicates which record is to be eliminated.

Format

Operation	Operands
ELIM	**fcbadr / (r)** **[,LAST / KEY / (0)]** **[,PARMOD=24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed.

(1)
The FCB address is stored in register 1.

LAST

The last record made available by one of the macros GET, GETFL, GETKY and GETR is processed (default value).

With the exception of OSTAT, no other ISAM macro which refers to the same FCB may be executed between the macro GET, GETFL, GETR or GETKY and the ELIM macro with function LAST. If SHARUPD=YES is specified, the preceding read macro must also set a lock and this lock must still be active when the ELIM macro is issued, i.e. no action, even for another FCB, which would cancel the lock may be executed.

KEY

The key of the record to be deleted is located at the address defined by KEYARG in the FCB.

If the specified key does not exist, the user program is continued at the address NOFIND (see "EXLST - Define exit address list"). If the file contains several records with the same key, the first of these records is eliminated.

(0)
The contents of register 0 indicate which record is to be processed:

- If the address in register 0 is not the FCB address, the function “KEY” is executed.
- If the address in register 0 is the FCB address, the “LAST” function is initiated, i.e. the last record made available is deleted.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the
GPARMOD macro or preset by the assembler.

= 24
The object code generated can run only in the 16-Mb address space (24-bit addressing only).

= 31
The object code generated can run in the 2-Gb address space (24-bit or 31-bit addressing).

Programming note

The ELIM macro overwrites the contents of registers 0, 1, 14 and 15.

ENCFILE - Convert unencrypted file into encrypted filen

Macro type: type S (E Form/M Form/L Form/C Form/D Form) (see "Macro types")

The ENCFILE macro converts an unencrypted file into an encrypted file.

File encryption with a crypto password enables the contents of a file to be protected against unauthorized access – even against people with TSOS privilege. However, file encryption does not protect against deletion, overwriting or destruction of the file contents and cannot replace file protection (e.g. by a write password).

The encryption method used is taken from the system parameter FILECRYP.

After ENCFILE has run all encryption attributes (procedure and check string for crypto password) are entered in the catalog entry and the read and execute passwords are deleted.

Only disk files on pubsets can be encrypted.

When PAM files are encrypted, the last-byte pointer is incremented to the block boundary.

File generations

ENCFILE cannot be used for individual file generations but only for complete file generation groups. Within a file generation group all generations with the exception of tape generations have the same encryption attributes as the group entry.

Format

Operation	Operands
ENCFILE	**,PATHNAM=<c-string 1..54: filename 1..54> / <var: char:54>** **.CRYPASS=<c-string 1..8: filename 1..8> / <var: char:8>** **,REFFILE=<c-string 1..54: filename 1..54> / <var: char:54>** **,EQUATES=YES / NO** **MF=L**
	MF=D,PREFIX= D / <pre>
	MF=E,PARAM=<name 1..27>

	MF=C / M
	,PREFIX= D / <pre>
	,MACID= MAE / <macid>

Operand descriptions

PATHNAM

Specifies the name of the file which is to be encrypted.

The file to be encrypted must satisfy the following requirements:

- It must be unencrypted.

- It must already have a catalog entry.
- The pubset on which it is cataloged must be available locally.
- It may not be located on a private disk.
- It may not have a tape type.
- It may not be located under the TSOS user ID on the home pubset.

=<c-string 1..54: filename 1..54>
Fully qualified path name of the file.

=<var: char:54>
Only possible with MF=M:
Symbolic address of a memory area of 54 bytes in which the file's path name is stored.

CRYPASS

This operand must not be specified together with the REFFILE operand.
Explicit specification of the crypto password which is needed for all accesses to the decrypted file contents. Instead of being specified here, the crypto password can also be taken over from an encrypted reference file (see the REFFILE operand).

If a user ID is entered for the FREFCRYP system parameter, CRYPASS can only be specified if the file specified under PATHNAM resides on this user ID.

=<c-string 1..8: filename 1..8>
Crypto password.

=<var: char:8>
Only possible with MF=M:
Symbolic address of a memory area of 54 bytes in which the crypto password is stored.

REFFILE = filename

Encrypted reference file from which the crypto password is taken over.

The reference file's crypto password must be entered in the calling task's crypto password table. If REFFILE is specified, CRYPASS may not be specified. If REFFILE is not specified, CRYPASS must be specified.

=<c-string 1..54: filename 1..54>
Fully qualified path name of the reference file.

=<var: char:54>
Only possible with MF=M:
Symbolic address of a memory area of 54 bytes in which the reference file's path name is stored.

EQUATES

Specifies whether equates are also to be generated for the values of the parameter area fields when the parameter area is expanded.

= YES
Equates are also generated for the values of the parameter area fields when the parameter area is expanded.

= NO
No equates are generated for the values of the parameter area fields when the parameter area is expanded.

Example

```
:
MVC ENCFMFC(XMAE#),ENCFMFL
ENCFILE MF=M,PREFIX=X,PATHNAM='UMSATZ.3.QUARTAL.2004'
ENCFILE MF=E,PARAM=ENCFMFC
:
ENCFMFC ENCFILE MF=C,PREFIX=X
ENCFMFL ENCFILE MF=L,CRYPASS='KROKODIL'
:
```

Programming notes

1. All RESERVED fields of the parameter area have to be deleted with binary zeros.
2. For all changes at parameter level that are not called with macros, the caller is responsible for the consistency of the parameter area.
3. For non-privileged calls (function state TU), register 1 points to the parameter area.
4. For the names listed in the parameter area, no conversion from small to capital letters is performed during the function execution. With the macro expansion, however, a conversion from small to capital letters is possible, depending on the compiler settings.

Notes on function execution

- File locks and file protection attributes which forbid write access to the catalog entry or the content of a file thus also prevent conversion of the file using ENCFILE.
- Conversion of a file with ENCFILE requires the calling task to have ownership rights for the file. Conversion therefore takes place when:
  - the file is under the user ID of the calling task.
  - the calling task is running under a user ID with TSOS privilege.
  - the user ID of the calling task is co-owner of the file and the file is not temporary.
- Conversion of the encrypted file is logged with SAT.
  The AUDIT attribute output here is taken from the catalog entry of the file to be converted (see the CREATE-FILE command, AUDIT operand, in the “Commands” manual [3]).
- Additional functions for tasks with TSOS privilege:
  If the calling task has TSOS privilege, the following additional functions are possible:
  - Temporary files which do not belong to the calling task but to another task can also be specified.
  - Temporary files can also be created on a pubset other than the default pubset of the user ID. (These are not deleted automatically when the calling task terminates.)
- RFA:
  ENCFILE is rejected if the file to be converted can only be accessed via RFA.
- Help file:
  When converting with ENCFILE a help file is created and then automatically deleted when the function has been completed. The converted file content is written to the help file. The help file needs as much disk storage space as the file to be converted.The file name of the help file has the following format:
  `S.DMS.<tsn>.<date><time>.CRYPTO`

Return codes

The return code is returned in the standard header of the parameter list. The standard header may not be located in the read-only area, otherwise the program is terminated.

Standard header: `ccbbaaaa`

The following code relating to execution of the ENCFILE macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	No error
	X'01'	X'0554'	Format of the file name not permitted
	X'01'	X'0576'	a) Incorrect operand combination b) Undeleted UNUSED fields
	X'20'	X'0578'	Internal error when checking the access rights
	X'82'	X'0594'	Not enough virtual memory available
	X'20'	X'05C7'	Internal error in DMS
	X'01'	X'05CB'	Incorrect/inadmissible first file name
	X'40'	X'05CF'	Password not in password table
	X'20'	X'05EC'	Internal error in crypto password handling
	X'40'	X'05FD'	File is write-proteected
	X'40'	X'0609'	Action not permitted for system file
	X'40'	X'060D'	Incorrect file name specified for reference file
	X'40'	X'0663'	Encryption of the file not permitted
	X'40'	X'0666'	File protection prevents access
	X'40'	X'0667'	File cannot be used as reference file
X'02'	X'00'	X'0669'	Protection attribute changed implicitly
X'00'	X'40'	X'066A'	Crypto password cannot be used
	X'40'	X'066D'	Crypto password specification has been restricted
	X'00'	X'066E'	Use help file
	X'01'	X'06C8'	Attribute not permissible for file generation
	X'01'	X'FFFF'	Wrong function number in standard header
	X'03'	X'FFFF'	Wrong version number in standard header

ERASE - Erase files

Macro type: type S (E form/L form/D form); see "Macro types"

Using the ERASE macro, the user can erase his/her own temporary or permanent files, file generation groups or file generations, depending on the selection criteria specified in the command. Furthermore, the user can release storage space and export files (= delete their catalog entries). The operands of the ERASE macro can be divided into five groups which correspond to the various functional levels (see table).

Selection operands

By means of the selection operands, the user specifies which files or catalog entries are to be processed, using attributes stored in the catalog entries as selection criteria. For this purpose, some operands of the FSTAT macro are integrated in the ERASE macro.If a selection operand is omitted, the files/catalog entries to be processed are selected without taking that selection criterion into account.

File protection operands

File protection operands make it unnecessary for the user to reset the protection attributes in order to erase files for which file protection attributes such as passwords, retention period, etc. are defined.

Macro execution

Action operands control the internal execution of the ERASE macro. The user can specify the scope for erasure and also define conditions for erasure.

Control operands permit the user to define, to a certain degree, his/her own user interface. (S)he can, for example, have a log printed or respond to control questions in a dialog.

Macro generation

Assembler operands control how the macro is generated. The VERSION operand, for example, controls the generation of the operand list.

Source compatibility is ensured for existing programs since the new format VERSION=3 fully covers the functions of the old formats VERSION=0/1/2. The only exception to this is: if the generated operand list is modified in the program, the program must be re-assembled.

Functional overview

Operand	Operand value	Selection criterion
Selection – file name		
*DUMMY		Dummy file
pathname		Path name (fully qualified, partially qualified, wildcards)
prefix		Temporary user files
*SYSid		System files, wildcards permitted
*		EAM object module file

Selection – file type		
BLKCTRL	NONE PAMKEY/DATA/DATA2K/ DATA4K/NO/NK/NK2/NK4	Catalog entries of unopened files File format
FCBTYPE	NONE ISAM/SAM/BTAM/PAM	Catalog entries of unopened files Access method
FILTYPE	*ANY/*BS2000/*NODE	File type on Net-Storage: BS2000 file or node file
POS	AFTER/BEFORE	In conjunction with TYPE=FGG; specifies the file generations to be processed
TYPE	FILE FGG PLAM	Files, not FGGs or file generations File generations or FGGs PLAM libraries
WORKFIL	*NO/*YES	Work files
Selection – volume		
STOTYPE	*PUBSPACE/*NETSTOR	Storage type
SUPPORT	PUBLIC PRDISC TAPE	Files on public disks Files on private disks Tape files
VOLSET		Volume set
VOLUME		VSN of the volume
Selection – data security and protection		
ACCESS	READ/WRITE	Write protection
AVAIL	*STD/*HIGH	Availability
BACKUP	A/B/C/D/E	Backup level
BASACL	NONE/YES	BASIC-ACL protection
ENCRYPT	ANY/NONE/AES/DES	Encrypted files
GROUPAR	NO-ACCESS/access-list	Access rights of the user group
GUARDS	(READ...,WRITE..., EXEC...)	GUARDS protection
OTHERAR	NO-ACCESS/access-list	Access rights of the others group

OWNERAR	NO-ACCESS/access-list	Access rights of the owner

PASS	NONE EXPASS RDPASS WRPASS	Password protection
PROTACT	ANY/LEVEL-0/ LEVEL-1/LEVEL-2	Protection level of the activated access control method
SHARE	NO/YES/SPECIAL	Shareability
Selection – storage space (disk files)		
EXTENTS		Number of extents
FSIZE		Size of reserved but unused storage space
LASTPAG		Number of PAM pages used
RELSPAC		Lock preventing release of storage space
SIZE		Size of reserved storage space
Selection – storage space (tape files)		
BLKCNT		Number of blocks in the file on tape
Selection – date and time entries		
CRDATE		Date and time of creation
DELDATE		DELETION date and time (implicit: retention period)
EXDATE		Expiration date and time (implicit: retention period)
LADATE		Date and time of last access
LCDATE		Date and time of last write access
TIMBASE	*UTC/*LTI	Time base of date entries
Selection – HSMS		
MIGRATE	ALLOWED/ INHIBIT/ FORBIDDEN	Migration allowed/ briefly allowed/ not allowed
SLEVEL	S0/S1/S2	Storage level
S0MIGR	*ALLOWED/*FORBIDDEN	Migration allowance

Selection – performance and I/O attributes		
DISKWR	IMMEDIATE/BY-CLOSE	Time at which data is written back to disk
IOPERF	STD/HIGH/VERY-HIGH	Performance attribute
IOUSAGE	RDWRT/WRITE/READ	Type of I/O operation
Selection – file status		
STATE	NOCLOS CLOSED CACHED NOT-CACHED CACHE-NOT-SAVED REPAIR-NEEDED DEFECT-REPORTED OPEN-ALLOWED NO-OPEN-ALLOWED	Current status of the file
Selection – coding		
CCS		Code table (coded character set)
Selection – metainformation		
ADMINFO	*NONE/<c-string 1..8>	System administrator metainformation
USRINFO	*NONE/<c-string 1..8>	User metainformation
Selection – SM pubset		
MANCLAS	*NONE/<c-string 1..8>	Management class
PREFORM		Intended file format on SM pubsets
STOCLAS	*NONE/<c-string 1..8>	Storage class
File protection – file protection operands		
IGNORE	omitted	Defined protection attributes are evaluated
	ACCESS	The protection attribute ACCESS=READ, BASIC-ACL or GUARDS is ignored
	EXDATE	Retention periods are ignored
	RDPASS WRPASS EXPASS	*For system administration only:* Dialog for change of user ID

PASSWORD	omitted	Password protection is evaluated
	password	The password protection defined by the specified password is ignored

Macro execution – action operands		
CATALOG		Files on private volumes are exported
DATA		Logical erasure: the data-specific attributes of the file are deleted, the catalog entry updated accordingly and the allocated storage space is retained
DATA-KEEP-ATTR		Logical erasure as with DATA, but the data-specific attributes are retained
DELETE-OR-EXPORT		Files on private volumes and node files on Net-Storage volumes are exported, files on public disks or on Net-Storage volumes are deleted
DESTROY		The catalog entry is deleted, the storage space is released and overwritten
MOUNT		Specifies, for files on private disks, whether all affected disks must be online
SPACE		Only storage space is released, the catalog entry is retained
SPACE-CATALOG		The catalog entry is deleted and storage space released
Macro execution – control operands		
CHECK	NO	No user intervention permitted (default value for procedures and batch jobs)
	MULTIPLE	Dialog when the catalog or user ID is changed if “pathname” was not fully qualified (default value for interactive mode)
	ERROR	Dialog if a user-correctable error occurs
	PVS	Dialog when the catalog ID is changed
	SINGLE	The user decides, in a dialog, whether each selected file is to be processed by the current ERASE macro
	USERID	*For system administration only:* Dialog for change of user ID
LIST	NO/YES	[Do not] log erasure on SYSOUT
NOSTEP	errcode	Via the DMS error code, the user can specify which errors are not to trigger the spin-off mechanism
Macro generation – assembler operands		
MF		Macro generation (operand list/SVC/DSECT)

PREFIX	prefix	Call-specific prefix

VERSION	0	Macro format for BS2000 versions < V9.5A (see table "Variations in versions - VERSION=0/1/2")
VERSION	1	Macro format for BS2000 Versions V9.5A and V10.0A (see table "Variations in versions - VERSION=0/1/2")
VERSION	2	Macro format as of BS2000/OSD-BC V1.0 (see “Macro format and operand descriptions” below)
VERSION	3	Macro format as of BS2000/OSD-BC V3.0 (see “Macro format and operand descriptions” below)

Operand overview

Operand	Operand value	Function
*DUMMY		Selection operand dummy file
pathname		Selection operand - path name (fully qualified, partially qualified, wildcards)
prefix		Selection operand - temporary user files
*SYSid		Selection operand - system files, wildcards permitted
*		Selection operand - EAM object module file
CATALOG		Files on private volumes are exported
DATA		Logical erasure: the data-specific attributes of the file are deleted, the catalog entry updated accordingly and the allocated storage space is retained
DATA-KEEP-ATTR		Logical erasure as with DATA, but the data-specific attributes are retained
DELETE-OR-EXPORT		Files on private volumes and node files on Net-Storage volumes are exported, files on public disks or on Net-Storage volumes are deleted
DESTROY		The catalog entry is deleted, the storage space is released and overwritten
MOUNT		Specifies, for files on private disks, whether all affected disks must be online
SPACE		Only storage space is released, catalog entry is retained
SPACE-CATALOG		The catalog entry is deleted and storage space released

ACCCNT		Selection operand - access counter
ACCESS	READ/WRITE	Selection operand - write protection

ADMINFO	*NONE/ <c-string 1..8>	Selection operand - system administrator metainformation
AVAIL	*STD/*HIGH	Selection operand - availability
BACKUP	A/B/C/D/E	Selection operand - backup level
BASACL	NONE/YES	Selection operand - BASIC-ACL protection
BLKCNT		Selection operand - number of blocks in the file on tape
BLKCTRL	NONE PAMKEY/DATA/ DATA2K/DATA4K/ NO/NK/NK2/NK4	Selection operand - catalog entries of unopened files File format
CCS		Selection operand - code table (coded character set)
CHECK	NO	No user intervention permitted (default value for procedures and batch jobs)
	MULTIPLE	Dialog when the catalog or user ID is changed if “pathname” was not fully qualified (default value for interactive mode)
	ERROR	Dialog if a user-correctable error occurs
	PVS	Dialog when the catalog ID is changed
	SINGLE	The user decides, in a dialog, whether each selected file is to be processed by the current ERASE macro
	USERID	*For system administration only:* dialog for change of user ID
CRDATE		Selection operand - date and time of creation
DELDATE		Selection operand - DELETION date and time (implicit: retention period)
DISKWR	IMMEDIATE/ BY-CLOSE	Selection operand - time at which data is written back to disk
EXDATE		Selection operand - expiration date and time (implicit: retention period)
EXTENTS		Selection operand - number of extents
FCBTYPE	NONE ISAM/SAM/ BTAM/PAM	Selection operand - catalog entries of unopened files Access method

FILTYPE	*ANY/*BS2000/ *NODE	Selection operand – file type on Net-Storage (BS2000 file or node file)
FSIZE		Selection operand - size of reserved but unused storage space

GROUPAR	NO-ACCESS/ access-list	Selection operand - access rights of the user group
GUARDS	(READ..., WRITE..., EXEC...)	Selection operand - GUARDS protection
IGNORE	omitted	Defined protection attributes are evaluated
	ACCESS	The protection attribute ACCESS=READ, BASIC-ACL or GUARDS is ignored
	EXDATE	Retention periods are ignored
	RDPASS WRPASS EXPASS	*For system administration only:* the defined password is ignored
IOPERF	STD/HIGH/ VERY-HIGH	Selection operand - performance attribute
IOUSAGE	RDWRT/WRITE/ READ	Selection operand - type of I/O operation
LADATE		Selection operand - date and time of last access
LASTPAG		Selection operand - number of PAM pages used
LCDATE		Selection operand - date and time of last write access
LIST	NO/YES	[Do not] log erasure on SYSOUT
MANCLAS	*NONE/ <c-string 1..8>	Selection operand - management class
MF		Macro generation (operand list/SVC/DSECT)
MIGRATE	ALLOWED/ INHIBIT/ FORBIDDEN	Selection operand - migration allowed/ briefly allowed/ not allowed
NOSTEP	errcode	Via the DMS error code, the user can specify which errors are not to trigger the spin-off mechanism
OTHERAR	NO-ACCESS/ access-list	Selection operand - access rights of the others group
OWNERAR	NO-ACCESS/ access-list	Selection operand - access rights of the owner

PASS	NONE EXPASS RDPASS WRPASS	Selection operand - password protection

PASSWORD	omitted password	Password protection is evaluated The password protection defined by the specified password is ignored
POS	AFTER/BEFORE	Selection operand - in conjunction with TYPE=FGG; specifies the file generations to be processed
PREFIX	prefix	Call-specific prefix
PREFORM		Selection operand - intended file format on SM pubsets
PROTACT	ANY/LEVEL-0/ LEVEL-1/LEVEL-2	Selection operand - protection level of the activated access control method
RELSPAC		Selection operand - lock preventing release of storage space
SHARE	NO/YES/SPECIAL	Selection operand - shareability
SIZE		Selection operand - size of reserved storage space
SLEVEL	S0/S1/S2	Selection operand - storage level
STATE	NOCLOS CLOSED CACHED NOT-CACHED CACHE-NOT-SAVED REPAIR-NEEDED DEFECT-REPORTED OPEN-ALLOWED NO-OPEN-ALLOWED	Selection operand - current status of the file
STOCLAS	*NONE/ <c-string 1..8>	Selection operand - storage class
STOTYPE	*PUBSPACE/ *NETSTOR	Selection operand - storage type
SUPPORT	PUBLIC PRDISC TAPE	Selection operand - files on public disk Files on private disk Tape volume
S0MIGR	*ALLOWED/ *FORBIDDEN	Selection operand - migration allowance
TIMBASE	*UTC/*LTI	Selection operand - time base of date entries

TYPE	FILE	Selection operand - files, not FGGs or file generations
	FGG	File generations or FGGs
	PLAM	PLAM libraries

USRINFO	*NONE/ <c-string 1..8>	Selection operand - user metainformation
VERSION	0	Macro format for BS2000 versions < V9.5A (see table "Variations in versions - VERSION=0/1/2")
	1	Macro format for BS2000 versions V9.5A and V10.0A (see table "Variations in versions - VERSION=0/1/2")
	2	Macro format as of BS2000/OSD-BC V1.0 (see Macro format and operand descriptions)
	3	Macro format as of BS2000/OSD-BC V3.0 (see Macro format and operand descriptions)
VOLSET		Selection operand - volume set
VOLUME		Selection operand - VSN of the volume
WORKFIL	*NO/*YES	Selection operand - work files

Format

The macro format represented below contains all the operands that are supported as of BS2000/OSD-BC V3.0. In order to generate this format, **VERSION=3** must be specified.

Source code compatibility for existing programs is ensured, since the new format VERSION=3 completely covers the functions of the old formats VERSION=0/1/2. However, an operand list generated in the program can only be changed if the program is reassembled with the updated macro.

- All operands/operand values that were supported up to and including BS2000 Version V2.0A may be used in the macro format with VERSION=2.
- All operands/operand values that were supported up to and including BS2000 Version V10.0A may be used in the macro format with VERSION=1.
- Operands/operand values that were supported up to and including BS2000 Version V9.0A may be used in the macro format with VERSION=0.

The table "Variations in versions - VERSION=0/1/2" shows which operands/operand values are supported with VERSION=2/1/0.

In the format, the representation of operands for which a list of operand values can be specified has been simplified. The list is shown as an additional value (list-of-operand) in the format and should be interpreted as follows:

- several operand values can be specified in the form of a list:
  `(element1, element2, ...)`
- if only one operand value is specified, i.e. the list consists of only one element, the parentheses may be omitted:
  `element` or `(element)`.

Operation	Operands
ERASE	**[pathname / prefix / * / *SYSid / *DUMMY]** **[,SPACE-CATALOG / SPACE / DATA / DATA-KEEP-ATTR / CATALOG /** **DELETE-OR-EXPORT / DESTROY]** **[,ACCCNT = ANY / nmbr / (nmbr[,]) / (,nmbr) / (nmbr1,nmbr2)** **[,ACCESS = ANY / READ / WRITE]** **[,ADMINFO = *ANY / *NONE / <c-string 1..8>]** **[,AVAIL = *ANY / *STD / *HIGH]** **[,BACKUP = ANY / A / B / C / D / E / (list-of-backup)]** **[,BASACL = ANY / NONE / YES]** **[,BLKCNT = ANY / nmbr / (nmbr[,]) / (,nmbr) / (nmbr1,nmbr2)]** **[,BLKCTRL = ANY / PAMKEY / DATA4K / DATA2K / DATA /** **NO / NONE / NK4 / NK2 / (list-of-blkctrl)]** **[,CCS = *ANY / *NONE / ccs-name]** **[,CHECK = NO / STD / MULTIPLE / ERROR / PVS / SINGLE / USERID]** **[,CRDATE = *ANY / *NONE / date / date(time[,]) / date(time1,time2) /** **(date[,]) / (date(time)[,]) / (,date) / (,date(time)) /** **(date1,date2) / (date1(time),date2) /** **(date1(time),date2(time))]** **[,DELDATE = *ANY / *NONE / date / date(time[,]) / date(time1,time2) /** **(date[,]) / (date(time)[,]) / (,date) / (,date(time)) /** **(date1,date2) / (date1(time),date2) /** **(date1(time),date2(time))]** **[,DISKWR = ANY / IMMEDIATE / BY-CLOSE]** **[,ENCRYPT = *ANY / *NONE / *AES / *DES / (list-of-encrypt)]** **[,EXDATE = *ANY / *NONE / date / date(time[,]) / date(time1,time2) /** **(date[,]) / (date(time)[,]) / (,date) / (,date(time)) /** **(date1,date2) / (date1(time),date2) /** **(date1(time),date2(time))]** **[,EXTENTS = ANY / nmbr / (nmbr[,]) / (nmbr1,nmbr2)]** **[,FCBTYPE = ANY / ISAM / BTAM / SAM / PAM / NONE / (list-of-fcbtype)]** **[,FILTYPE = *ANY / *BS2000 / *NODE]** **[,FSIZE = ANY / SIZE / nmbr / (nmbr[,]) / (nmbr) / (nmbr1,nmbr2)]** **[,GROUPAR = ANY / NO-ACCESS / list-of-ownerar]**

```
[,GUARDS = *ANY /
           *NONE /
           *YES /
           ([READ = *ANY / *NONE / fname]
            [,WRITE = *ANY / *NONE / fname]
            [,EXEC = *ANY / *NONE / fname])]

[,IGNORE = ANY / ACCESS / EXDATE / RDPASS / WRPASS / EXPASS /
           (list-of-ignore)]

[,IOPERF = ANY / STD / HIGH / VERY-HIGH / (list-of-ioperf)]

[,IOUSAGE = ANY / RDWRT / WRITE / READ / (list-of-iousage)]

[,LADATE = *ANY / *NONE / date / date(time[,]) / date(time1,time2) /
           (date[,]) / (date(time)[,]) / (,date) / (,date(time)) /
           (date1,date2) / (date1(time),date2) /
           (date1(time),date2(time))]

[,LASTPAG = ANY / nmbr / (nmbr[,]) / (,nmbr) / (nmbr1,nmbr2)]

[,LCDATE = *ANY / *NONE / date / date(time[,]) / date(time1,time2) /
           (date[,]) / (date(time)[,]) / (,date) / (,date(time)) /
           (date1,date2) / (date1(time),date2) /
           (date1(time),date2(time))]

[,LIST = ERRORS-TO-SYSOUT / (area,len)]

[,MANCLAS = *ANY / *NONE / <c-string 1..8>]

[,MIGRATE = ANY / ALLOWED / INHIBIT / FORBIDDEN / (list-of-migrate)]

[,MOUNT = FIRST-DISK / ALL-DISKS]

[,NOSTEP = errcode / (list-of-nostep)]

[,OTHERAR = ANY / NO-ACCESS / list-of-ownerar]

[,OWNERAR = ANY / NO-ACCESS / list-of-ownerar]

[,PASS = ANY / NONE / EXPASS / RDPASS / WRPASS / (list-of-pass)]

[,PASSWD = kennwort / (list-of-passwd)]

[,POS = AFTER / BEFORE]

[,PREFORM = *ANY / *NONE / *K / *NK2 / *NK4 / (list-of-preform)]

[,PROTACT = ANY / LEVEL-0 / LEVEL-1 / LEVEL-2 / (list-of-protact)]

[,RELSPAC = ANY / ALLOWED / IGNORED / (list-of-relspac)]

[,SHARE = ANY / YES / NO / SPECIAL / (list-of-share)]

[,SIZE = ANY / FSIZE / nmbr / (nmbr[,]) / (,nmbr) / (nmbr1,nmbr2)]

[,SLEVEL = ANY / S0 / S1 / S2 / (list-of-slevel)]
```

[,STATE = ANY / NOCLOS / CLOSED / CACHED / NOT-CACHED / CACHE-NOT-SAVED / OPEN-ALLOWED / NO-OPEN-ALLOWED / REPAIR-NEEDED / DEFECT-REPORTED / (list-of-state)] **[,STOCLAS = *ANY / *NONE / <c-string 1..8>]** **[,STOTYPE = *ANY / *PUBSPACE / *NETSTOR]** **[,SUPPORT = ANY / PUBLIC / PRDISC / TAPE / (list-of-support)]** **[,S0MIGR = *ANY / *ALLOWED / *FORBIDDEN / (list-of-s0migr)]** **[,TIMBASE = *UTC / *LTI]** **[,TYPE = ANY / FILE / FGG / PLAM / (list-of-type)]** **[,USRINFO = *ANY / *NONE / <c-string 1..8>]** **[,VOLSET = *ANY / <c-string 1..4>]** **[,VOLUME = *ANY / vsn]** **[,WORKFIL = *ANY / *NO / *YES]**
[,MF = L],VERSION = 0 / 1 / 2 / 3[,PREFIX = pre]
MF = (E,adr / E,(r)),VERSION = 0 / 1 / 2 / 3
MF = D,VERSION = 1 / 2 / 3[,PREFIX = pre]

Operand descriptions

pathname

Designates the pathname of the files that are to be erased, with:
<c-string 1..80: filename 1..54 with-wild(80) without-gen>

Only the user's own files or for which the user possesses co-owner rights may be erased.

pathname means [:catid:][$userid.][filename]

catid
Catalog ID of the files that are to be erased; if wildcards are used for “catid”, then these are evaluated only for catalogs in an MPVS environment. Catalogs in an MSCF environment can be addressed only via their explicit “catid” (for information on MSCF see the “HIPLEX MSCF” User Guide [11]).
The default value is the “catid” assigned to the user ID.

userid
User ID: non-privileged users may only delete their own files or files under user IDs for which they are entered as co-owner. The system administrator can also specify wildcards.
Default value: the user ID of the current job (i.e. of the SET-LOGON-PARAMETERS or LOGON command).

filename
Designates the files, file generations, FGGs or temporary files to be erased. The user may specify a fully or partially qualified file name or use wildcards. The prefix must be included in the names of temporary files; otherwise, the temporary files are ignored.

Wildcard specifications
Nonprivileged users may only use wildcards in the “catid” and “filename” specifications whereas the system administrator may also use them in the “userid” (as in the FSTAT macro; see section "Wildcards"). If the wildcard '*' is used, it must be entered twice (**) if it is to include the beginning of the file name (for example: ERASE *SYSLST erases the system file SYSLST, while ERASE **SYSLST erases all files whose names end with the string SYSLST).

prefix

With the prefix defined for temporary files, all temporary files of the job can be erased. If the erase operation is logged, the internal names of the temporary files being processed are output.

*****
The ERASE macro is for the EAM object module file (* file) which is created and used by compilers. All operands except the control and Assembler operands (CHECK, LIST, NOSTEP) are checked for syntax errors, but are otherwise ignored. Errors which occur when erasing the * file are ignored.

***SYSid**
Designates the logical system files SYSLST, SYSLSTnn and SYSOUT (00 <= nn <= 99). Wildcards may be specified for “id”, which means that one ERASE command may be used for several system files (see section "Wildcards" for details of wildcards). All operands except the control operands (CHECK, LIST, NOSTEP) are checked for syntax errors, but are otherwise ignored.

The system file SYSOUT can also be erased in dialog mode.

If SYSLST is assigned to a file and has been printed out with PRINT *SYSLST, a subsequent ERASE *SYSLST macro logically erases only those pages which have been created since the printout.

If a LOGOFF command immediately follows an ERASE *SYSOUT or ERASE *SYSLST and no log is requested via LIST=YES or /OPTION MSG=H, no new SYSOUT or SYSLST file is created.

***DUMMY**
Designates the dummy file *DUMMY, which “always exists” and fulfills all selection criteria. All operands except the control operands (CHECK, LIST, NOSTEP) are checked for syntax errors, but are otherwise ignored. If *DUMMY is specified, no catalog or data access is necessary. *DUMMY is particularly useful for test runs.

CATALOG

Only for files, file generation groups and file generations on private volumes and for files on Net-Storage of type NETSTOR:
The catalog entries of the specified or selected files are deleted, but their storage space is retained. Password protection is observed, but write protection defined with ACCESS=READ or implied by RETPD (see the CATAL macro) is ignored. Any definition for “binary deletion” (see DESTROY, CATAL macro) in the catalog entry is also ignored.

For tape files, “CATALOG” is the default value for the execution of ERASE.

The action ERASE ...,CATALOG is equivalent to exporting the file(s) (see the VOLUME operand). These files can be imported again later, either individually by means of FILE (with STATE=FOREIGN) or using IMPORT, which can import one or more files on private disks or on Net-Storage of type NETSTOR at the same time. Exclusively reserved files cannot be exported.

Files on a Net-Storage volume of type NETVOL cannot be exported, because there is no file catalog (BS2FSCAT) on the Net-Storage volume.

DATA

Only for disk files; the default value CATALOG applies to tape files:
The data of the affected files is “logically erased” (see “Logically erasing a file” in the ”Introductory Guide to DMS” [1]). After this, the user can no longer access the file’s data since physical access to the relevant volume is not permitted. The catalog entry and the space allocation still exist. The catalog entry is identical to that for a file which has been created by means of FILE but not yet opened (FCBTYPE=NONE, CRDATE=NONE).

DATA-KEEP-ATTR

Only for disk files; the default CATALOG applies for tape files:
The files are logically erased as with DATA, but the data-specific attributes are retained. The data itself can, however, no longer be addressed by the user.

DELETE-OR-EXPORT

Selects files for processing by ERASE on the basis of the type of volume on which the files are stored:

- Files, FGGs, etc. on public volumes and files are erased, i.e. the catalog entry is deleted and the storage space is released (corresponds to the specification “SPACE-CATALOG”).
- For files on Net-Storage the following applies depending on the file type:
  - For BS2000 files the catalog entry is deleted and the storage space is released, regardless of the Net-Storage volume type (NETSTOR or NETVOL) on which they reside.
  - For node files the catalog entry is deleted. The files are retained on the Net-Storage (corresponds to the EXPORT-NODE-FILE command),regardless of the Net-Storage volume type (NETSTOR or NETVOL) on which they reside.
- For files, FGGs, etc. on private volumes, the catalog entry is simply deleted (corresponds to the specification “CATALOG”).

DESTROY

Only for disk files; the default value CATALOG is valid for tape files:
The storage space for the selected files is released, the catalog entry is deleted, and the storage space being released is overwritten by binary zeros so that if the space is allocated again, nobody can read the old data (data protection). In the case of files on private disks, all volumes on which the file was stored must be mounted when erasure takes place.

“Data destruction” during erasure can also be set in the catalog entry (DESTROY=YES) by means of the CATAL macro. In this case, the storage space being released is automatically overwritten. When a file is erased, the action operands are evaluated first: if the file is to be exported (specification CATALOG or DELETE-OR-EXPORT), the data is not overwritten, since the storage space is not released.

SPACE

Only for files on public disks and on Net-Storage; the default value CATALOG is valid for tape files:
The storage space for the files affected by ERASE is released. The catalog entry is retained but updated: it is then identical to one created via CATAL. The SPACE operand is rejected for files on private disks.

SPACE-CATALOG

is the default value for disk files; the catalog entries for the affected files are deleted, and the storage space used by these files is released.

ACCCNT

Allows the user to select files to be processed on the basis of the access count, which indicates how often a file has been accessed. The access counter can be assigned values from 0 to 2147483647.

= ANY
The access counter is not a selection criterion.

= nmbr
Processes files for which the access count exactly matches the specified number of accesses.

= (nmbr[,])
Processes files for which the access count is greater than or equal to the specified value.

= (,nmbr)
Processes files for which the access count is less than or equal to the specified value.

= (nmbr1,nmbr2)
Processes files for which the access count lies in the specified interval:(nmbr1 <= access-count <= nmbr2).

ACCESS

The user can select files for processing depending on his or her access authorization.

= ANY
The access type is not a selection criterion.

= READ
Only files for which write access is forbidden by ACCESS=READ, i.e. read-only files, are processed.

= WRITE
Selects files for which read and write access is permitted.

ADMINFO

The user can select files/file generations for processing dependent on the system administrator metainformation.

= *ANY
The system administrator metainformation is not a selection criterion.

= *NONE
Only those files are processed that possess no system administrator metainformation.

= <c-string 1..8>
Only those files possessing the specified system administrator metainformation are processed.

AVAIL

The user can select files/file generations for processing dependent on their availability.

= *ANY
The availability is not a selection criterion.

= *STD
Only those files not on a volume set with high availability are processed.

= *HIGH
Only those files on a volume set with high availability are processed (DRV pubset).

BACKUP

The user can select the files for processing on the basis of the BACKUP level. The backup defines in which backup runs the file is to be saved.

= ANY
The backup level is not a selection criterion.

= A
Only the files with backup level A are processed.

= B
Only the files with backup level B are processed.

= C
Only the files with backup level C are processed.

= D
Only the files with backup level D are processed.

= E
Only the files with backup level E are processed.

= (list-of-backup)
All files that have one of the specified backup levels are processed. Up to 5 backup different levels may be specified in a list.

BASACL

Allows the user to select files for processing on the basis of whether they are protected by a basic access control list (BASIC-ACL).

= ANY
The BASIC-ACL is not a selection criterion.

= NONE
Processes all files for which no BASIC-ACL entry is defined.

= YES
Processes only those files for which a BASIC-ACL entry is defined. The selection operands OWNERAR, GROUPAR, and OTHERAR can be used to restrict the selection to specific BASIC-ACL entries.

BLKCNT

For tape files only:
Selects files for processing on the basis of the number of blocks on tape.

= ANY
The number of blocks on tape is not a selection criterion.

= nmbr
Processes all tape files with exactly the specified number of blocks.

= (nmbr[,])
Processes all tape files for which the number of blocks is greater than or equal to the specified value.

= (,nmbr)
Processes all tape files for which the number of blocks is less than or equal to the specified value.

= (nmbr1,nmbr2)
Processes all tape files for which the number of blocks lies within the specified interval.

Any integers from the range 0 <= value <= 2147483647 may be specified.

BLKCTRL

Allows the user to select files for processing on the basis of the file format. The file format is defined when creating the file and is based on the existence and position of the block control field that contains management information for the PAM page.

= ANY
The file format is not a selection criterion.

= PAMKEY
Processes all files which use a separate PAM key for the block control field, i.e. files for which the block control information is stored in a special key field outside the PAM block. Such files are created with BLKCTRL=PAMKEY (see the FILE macro).

= DATA
Processes all files for which the block control information is located at the start of the data block. Such files are created with BLKCTRL=DATA (see the FILE macro).

= NO
Processes all files which contain no block control field. Such files are created with BLKCTRL=NO (see the FILE macro).

= NONE
Processes all files for which no BLKCTRL value was defined, i.e. files which have not yet been opened.

= DATA2K
Processes all files which were created with BLKCTRL=DATA2K (see the FILE macro).

= DATA4K
Processes all files that were created with BLKCTRL=DATA4K (see the FILE macro).

= NK2
Processes all NK2 files (files which can be stored on NK2 volumes).

= NK4
Processes NK4 files only (files which can be stored on NK4 volumes).

= (list-of-blkctrl)
Processes all files that match one of the specified file formats. All values except ANY may be specified in a list.

CCS

Allows the user to select files for processing on the basis of the specified **c**oded **c**haracter **s**et.
The coded character set (CCS) defines how the characters of a national character set are to be stored in binary form. The specified character set has an effect on the representation of characters on the screen, the collating sequence, etc. (see the “XHCS” manual [22]).

= *ANY
The code table is not a selection criterion for files to be processed with ERASE.

= *NONE
Only files for which no character set is defined are erased.

= ccs-name
Only the files for which the specified code table was defined are processed. The name of the code table may consist of up to 8 alphanumeric characters.

CHECK

As in interactive mode, the user can specify that a control check be performed by issuing a prompt to SYSOUT before a file set is processed. The file set for which the prompted dialog is to be executed can be specified by the user (e.g. for all files to be processed). The issued prompt must be answered by the user as follows:

- “Y” confirms that the specified file set is to be processed.
- “N” excludes the specified file set from the operation.
- “T” terminates the entire ERASE processing operation.

Any response that consists of only blanks or the “null string” will be interpreted as “N”.

In batch mode, CHECK=NO always applies.

= STD
The default setting depends on the operating mode:

- In an interactive dialog (SYSCMD is assigned to the terminal), CHECK=MULTIPLE is the default.
- In procedures and in batch mode, the default value is CHECK=NO.

= NO
The user cannot intervene in the ERASE processing; all specified or selected files are erased.

= MULTIPLE
If “pathname” is partially qualified, which means that more than one file is selected, or if “pathname” contains wildcards, the user can decide, each time the catalog ID changes, whether or not files from the new catalog are to be erased. He/she must respond with “YES” or “NO” to the question issued by the system.
CHECK=MULTIPLE

is useful if wildcards are specified for “catid” in the “pathname”. In the dialog, ERASE processing can be terminated by responding with “TERMINATE” to the question, or the CHECK mode can be changed (-> NO/ERROR/SINGLE/PVS).

= ERROR
By means of CHECK=ERROR, the user specifies that a dialog as for CHECK=SINGLE is to be started if user-correctable errors occur. As long as no errors occur, CHECK=ERROR is equivalent to CHECK=NO (i.e. no dialog). CHECK=ERROR is set implicitly if CHECK=SINGLE is selected.

In the case of an error, the user can acknowledge the error message, abort ERASE processing or attempt to recover the error. If desired, he/she can also change the CHECK mode.

= PVS
As for CHECK=MULTIPLE, ERASE processing starts a dialog if files in different catalogs are affected by the ERASE macro. The user can respond with “YES” or “NO” to the system question, abort ERASE processing (“TERMINATE”) or change the CHECK mode.

= SINGLE
For each file which is processed, the user can decide in a dialog whether or not it is to be erased (response YES/NO). If, in the dialog, (s)he specifies protection attributes or one or more passwords together with “IGNORE”, these specifications are evaluated and any file which fulfills them is erased without further questions (“YES” must also be specified). The user can also abort ERASE processing or change the CHECK mode.

The affected files are listed in alphabetical order. If file generation groups are affected, the generations are listed separately in the order of their generation numbers. If the user elects not to erase a file generation, processing of the file generation group is terminated and the current status is saved (there must be no gaps in the sequence of file generations).

If only parts of a generation group are to be erased, the order of the generations depends on the value of the POS operand: with POS=AFTER, the generations are listed in descending order of their generation numbers, starting with the youngest generation; with POS=BEFORE, they are listed in ascending order of their generation numbers, starting with the oldest generation.

= *USERID*
For system administration only:
ERASE processing branches to an interactive dialog if files of various user IDs are involved. Whenever the user ID changes, a prompt is issued to determine whether the next user ID is to be processed.
The system administrator can accept (“YES”), deny (“NO”), or end (“TERMINATE”) the ERASE operation, or switch to CHECK mode.

CRDATE

Allows the user to select files for processing on the basis of their creation dates. File generation groups and file generations are not taken into account.

Date values may be supplemented by specifying a time. The rules for date and time specifications are described on "Format of date specifications". Range specifications are inclusive of both specified limits.

= ANY
The creation date is not a selection criterion.

= NONE
Processes all files for which no creation date has been entered in the catalog, i.e. files which have not yet been opened.

= date
Processes all files that were created on the specified date.

= (date[,])
Processes all files that were created on or after the specified date
(creation date >= current date).

= (,date)
Processes all files that were created on or before the specified date
(creation date <= current date).

= (date1,date2)
Processes all files that were created within the specified period
(date1 <= creation date <= date2).

= date(time[,])
Processes all files that were created on the specified date on or after the specified time.

= date(time1,time2)
Processes all files that were created on the specified date within the specified period.

= (date(time)[,])
Processes all files that were created on or after the specified date and time.

= (,date(time))
Processes all files that were created before the specified date and time.

(date1(time),date2(time))
Processes all files that were created within the specified period. The upper and lower limits of the specified period are defined more precisely by a time specification in both cases.

DELDATE

Allows the user to select files on the basis of the DELETION-DATE (the time from which the file may be deleted irrespective of the protection attributes).

The user can supplement the date values by specifying a time. It must be noted in this respect that the deletion time of 00:00:00 is currently always entered in the file catalog.

The rules for date and time specifications are described on "Format of date specifications". Range specifications are inclusive of both specified limits.

= *ANY
The DELETION-DATE is not a selection criterion.

= *NONE
Processes all files for which no DELETION-DATE is entered in the catalog.

= date
Processes all files for which the specified DELETION-DATE is defined.

= (date[,])
Processes all files whose DELETION-DATE is later or equal to the specified date.

= (,date)
Processes all files whose DELETION-DATE is earlier or equal to the specified date.

= (date1,date2)
Processes all files whose DELETION-DATE lies within the specified time period (date1 <= release date <= date2).

= date(time[,])
Processes all files for which the specified DELETION-DATE is defined and for which the release time is later or equal to the specified time. The release time (referred to the DELETION-DATE) is always entered in the catalog as 00:00:00.

= date(time1,time2)
Processes all files for which the specified DELETION-DATE is defined and for which the release time is within the specified time period. The release time (referred to the DELETION-DATE) is always entered in the catalog as 00:00:00.

= (date(time)[,])
Processes all files whose DELETION-DATE and time is later than or equal to the specified time. The release time (referred to the DELETION-DATE) is always entered in the catalog as 00:00:00.

= (,date(time))
Processes all files whose DELETION-DATE and time is earlier than or equal to the specified time. The release time (referred to the DELETION-DATE) is always entered in the catalog as 00:00:00.

(date1(time),date2(time))
Processes all files whose DELETION-DATE lies within the specified time period (date1 <= DELETION-DATE <= date2). The upper and lower limits of the specified time period are defined more exactly by specifying a time.

DISKWR

Enables the user to select files for processing based on the time at which data consistency is required for them, as defined in the catalog entry.

= ANY
The time at which data consistency is required, as defined in the catalog, is not a selection criterion.

= IMMEDIATE
Processes all files for which data consistency is required immediately after a write operation is completed. Such files are not suitable for processing in a write cache.

= BY-CLOSE
Processes all files for which data consistency is not required until CLOSE processing. These files are suitable for processing in a write cache.

ENCRYPT

Enables the user to select files based on whether or with which encryption method they are encrypted.

= *ANY
Processes all files regardless of whether or with which encryption method they are encrypted.

= *NONE
Processes only those files which are not encrypted.

= *AES
Processes only those files which are encrypted with the AES encryption method.

= *DES
Processes only those files which are encrypted with the DES encryption method.

EXDATE

The user can select files to be processed on the basis of their expiration date. The expiration date of a file is defined in the catalog and specifies when the file may be updated again or deleted. If no expiration date is defined when creating the file, the expiration date is set to the creation date.
File generation groups and file generations are not taken into account.

The user may supplement date specifications by a time value; however, it should be noted that the time stamp for the expiration date is always set to 00:00:00 in the file catalog at present.
The rules for date and time specifications are described on "Format of date specifications". Ranges defined in intervals include both specified limits.

= ANY
The expiration date is not a selection criterion.

= NONE
Processes all files for which no expiration date has been entered in the catalog, i.e. files that have not yet been opened.

= date
Processes all files for which the specified expiration date is defined.

= (date[,])
Processes all files for which the expiration date is greater than or equal to the specified date.

= (,date)
Processes all files for which the expiration date is less than or equal to the specified date.

= (date1,date2)
Processes all files for which the expiration date lies within the specified period
(date1 <= expiration date <= date2).

= date(time[,])
Processes all files with the specified expiration date and with an expiration time that is greater than or equal to the specified time.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= date(time1,time2)
Processes all files with the specified expiration date and with a time of expiration that lies within the specified time interval.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= (date(time)[,])
Processes all files for which the expiration date and time is greater than or equal to the specified time.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= (,date(time))
Processes all files for which the expiration date and time is less than or equal to the specified time.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= (date1(time),date2(time))
Processes all files for which the expiration date lies within the specified period (date1 <= expiration date <= date2). The upper and lower limits of the specified period are defined more precisely by time values in both cases.

EXTENTS

Only for files on disks and on Net-Storage:
Allows the user to select files for processing on the basis of the specified number of extents. An extent is a contiguous area occupied by a file on a disk. The number of extents which make up a file is stored in the catalog. A file on Net-Storage has precisely one extent.
File generation groups and file generations are not taken into account.

The possible values for “nmbr” are: 0 <= nmbr <= 65535. Range specifications include both upper and lower limits.

= ANY
The number of extents is not a selection criterion.

= nmbr
Only the files with exactly the specified number of extents are processed.

= (nmbr [,])
Only the files with at least the specified number of extents (number of extents >= nmbr) are processed.

= (,nmbr)
Only the files that have at most the specified number of extents (number of extents <= nmbr) are processed.

= (nmbr1, nmbr2)
Only the files that have at least as many extents as “nmbr1” and at most as many extents as “nmbr2” (i.e. nmbr1 <= number of extents <= nmbr2) are processed.

FCBTYPE

Allows the user to select files for processing on the basis of the access method with which they were created. The access method is entered into the file catalog when the file is created. It corresponds to the specification for FCBTYPE in the FILE macro.
File generation groups and file generations are not taken into account.

= ANY
The access method is not a selection criterion.

= NONE
Processes only those files for which no access method has been entered in the catalog, i.e. files which have not yet been opened.

= ISAM
Processes only those files which were created with the ISAM access method.

= BTAM
Processes files for that were created with the BTAM access method (BTAM files). BTAM files are tape files, and can therefore only be exported (see the action operand CATALOG).

= SAM
Processes files that were created with the SAM access method (SAM files).

= PAM
Processes files that were created with the UPAM access method (PAM files).

= (list-of-fcbtype)
More than one access method may be specified in a list. All files that were created with one of the specified access methods will be processed.

FILTYPE

The user can select the files to be processed according to the file type.

= *ANY
The file type is not a selection criterion.

= *BS2000
BS2000 files are processed.

= *NODE
Files are processed which are created as node files.

FSIZE

Only for disk files on disk and on Net-Storage:
Allows the user to select files for processing on the basis of the number of free PAM pages. The free PAM pages of a file indicate the amount of reserved but unused storage space.File generation groups and file generations are not taken into account.

“number” must be: 0 <= number <= 2147483647; if ranges are specified, the limit values are included in the range.

= ANY
The size of the free (= reserved but unused) storage space is not a selection criterion.

= SIZE
Only the files for which none of the reserved pages are used (i.e. for which no PAM page has been written) are processed.

= nmbr
Processes files with exactly the specified number of reserved but unused PAM pages.

= (nmbr[,])
Processes files with at least the specified number of reserved but unused PAM pages (free PAM pages >= nmbr).

= (,nmbr)
Processes files with no more than the specified number of reserved but unused PAM pages (free PAM pages <= nmbr).

= (nmbr1,nmbr2)
Processes files for which the number of free pages lies within the specified range (nmbr1 <= free PAM pages <= nmbr2).

GROUPAR

Selects and processes files on the basis of the access rights that are defined for members of the file owner's user group in BASIC-ACL entries.

= ANY
The BASIC-ACL entries for members of the file owner's user group are not a selection criterion.

= NO-ACCESS
Processes all files that cannot be accessed by the user group of the owner.

= access-list
Only processes files for which at least one of the access rights specified in the list has been entered for the file owner's user group in the BASIC-ACL entry.

The access list has the following format:

- Long format:
  ([READ=YES / READ=NO][,WRITE=YES / WRITE=NO][,EXEC=YES / EXEC=NO])
- Short format:
  ([R=Y / R=N][,W=Y / W=N][,X=Y / X=N])

The parentheses constitute part of the operand value and must be specified.

The individual elements of the access list mean the following:

READ=YES or R=Y	Processes all files to which the user group of the owner has read access.
READ=NO or R=N	Processes all files to which the user group of the owner does not have read access.
WRITE=YES or W=Y	Processes all files to which the user group of the owner has write access.
WRITE=NO or W=N	Processes all files to which the user group of the owner does not have write access.
EXEC=YES or X=Y	Processes all files which the user group of the owner may execute.
EXEC=NO or X=N	Processes all files which the user group of the owner may not execute.

GUARDS

The user can select files to be processed on the basis of the access protection defined by GUARDS (see the “SECOS” [8] manual).

= *ANY
The access protection defined by GUARDS is not a selection criterion.

= *NONE
Processes all files which have no access protection defined by GUARDS.

= *YES
Processes all files which have access protection defined by GUARDS.

= (READ=...,WRITE=...,EXEC=...)
The type of access protection provided by GUARDS that is to be used as a selection criterion can be defined by the user in a list. For each access mode (read, write, and execute), the defined protection can be specified precisely. If no entry is made for an access mode, the protection defined for that access mode has no effect on the selection.

For each access mode, one of the following values may be specified:

*ANY	The defined GUARDS protection is not a selection criterion.
*NONE	No guard has been defined for the specified access mode, i.e. the corresponding access is denied.
fname	All conditions for granting access in the specified access mode are defined in the guard fname.

IGNORE

The user can define whether defined protection against write access or a defined retention period is to be ignored. The IGNORE operand replaces the call to the CATAL macro which would be otherwise required to reset the protection attributes for the files to be processed before calling the ERASE macro.

= ANY
If no operand is specified here, all specified protection attributes are taken into account for ERASE processing.

= ACCESS
Enables files which are protected against write access by the owner to be selected for deletion with ERASE. Existing write protection is ignored. This specification is ignored if there is a TSOS restriction for a file under a foreign user ID.

= EXDATE
Allows files for which a retention period exists (expiration date > current date) to be erased. Existing retention periods are ignored.

= *RDPASS*
For system administration only:
Allows files which are protected by a read password to be erased during ERASE processing. Existing access protection by a read password is ignored.

= *WRPASS*
For system administration only:
Allows files which are protected by a write password to be erased during ERASE processing. Existing access protection by a write password is ignored.

= *EXPASS*
For system administration only:
Allows files which are protected by an execute password to be erased during ERASE processing. Existing access protection by an execute password is ignored.

= (list-of-ignore)
The operand values ACCESS and IGNORE can be specified in a list, i.e. both protection attributes are ignored.

IOPERF

Processes all files based on the performance attribute that is defined for them in the catalog (see the IOPERF operand in the CATAL macro).

= ANY
The performance attribute is not a selection criterion.

= STD
Processes all files for which the performance attribute was defined as STD.

= HIGH
Processes all files for which the performance attribute was defined as HIGH (high performance priority).

= VERY-HIGH
Processes all files for which the performance attribute was defined as VERY-HIGH (highest performance priority).

= (list-of-ioperf)
Up to three performance attributes may be specified in a list. All files that have one of these specified attributes will be shown.

IOUSAGE

The user can select the files to be processed, depending on the type of I/O operations to which the performance attribute applies (see the IOUSAGE operand in the CATAL macro).

= ANY
The performance attribute is not a selection criterion.

= RDWRT
Processes all files for which the performance attribute applies to read and write operations.

= WRITE
Processes all files for which the performance attribute applies to write operations only.

= READ
Processes all files for which the performance attribute applies to read operations only.

= (list-of-iousage)
More than one type of I/O operation may be specified in a list. All files for which the performance attribute applies to at least one of the specified I/O operations will be shown.

LADATE

Processes all files with the corresponding date of last access. File generation groups and file generations are not taken into account.

The user can supplement date specifications by specifying time values. The rules for date and time specifications are described on "Format of date specifications".
Ranges specified in intervals include both upper and lower limits.

= ANY
The last access date is not a selection criterion.

= NONE
Processes all files for which no last access date has been entered in the catalog, i.e. files that have not yet been opened.

= date
Selects files which were last accessed on the specified date.

= (date[,])
Processes all files that were last accessed on or after the specified date
(last access date >= date).

= (,date)
Selects all files that were accessed on or before the specified date
(last access date <= date).

= (date1,date2)
Processes all files that were last accessed during the specified period
(date1 <= last access date <= date2).

= date(time[,])
Processes all files that were last accessed on the specified date on or after the specified time.

= date(time1,time2)
Selects all files that were last accessed on the specified date and within the specified period.

= (date(time)[,])
Selects all files that were last accessed on or after the specified date and time.

= (,date(time))
Selects all files that were last accessed before the specified date and time.

= (date1(time),date2(time))
Selects all files that were last accessed within the specified period. The upper and lower limits of the specified period are defined more accurately by means of a time specification.

LASTPAG

Selects and processes files based on the amount of storage space used (i.e. the number of PAM pages written). The last-page pointer is set to the highest used PAM page.

= ANY
The storage space used is not a selection criterion.

= nmbr
Processes all files for which exactly the specified number of PAM pages have been written.

= (nmbr[,])
Processes all files for which the number of used PAM pages is greater than or equal to the specified value.

= (,nmbr)
Processes all files for which the number of used PAM pages is less than or equal to the specified value.

= (nmbr1,nmbr2)
Processes all files for which the number of written PAM pages lies in the interval defined by nmbr1 and nmbr2 (nmbr1 <= used PAM pages <= nmbr2).

Any integers from the range 0 <= nmbr <= 2147483647 may be specified.

LCDATE

Selects and processes files based on the date on which they were last accessed for writing (last change date). File generation groups and file generations are not taken into account.

The user can supplement date specifications by means of time values. The rules for date and time specifications are described on "Format of date specifications". Ranges defined by intervals include both the specified limits.

= ANY
The date of last write access is not a selection criterion.

= NONE
Processes all files for which no date of last write access is entered in the catalog, i.e. files that have not yet been opened.

= date
Processes all files that were last written to (i.e. changed) on specified date.

= (date[,])
Processes all files that were last changed on or after the specified date
(last change date >= date).

= (,date)
Processes all files that were last changed on or before the specified date
(last change date <= date).

= (date1,date2)
Processes all files that were last changed during the specified period
(date1 <= last change date <= date2).

= date(time[,])
Processes all files that were last changed on the specified date on or after the specified time.

= date(time1,time2)
Processes all files that were last changed on the specified date and within the specified period.

= (date(time)[,])
Processes all files that were last changed on or after the specified date and time.

= (,date(time))
Processes all files that were last changed before the specified date and time.

= (date1(time),date2(time))
Processes all files that were last changed within the specified period. The upper and lower limits of the specified period are defined more precisely by time values.

LIST

The user can specify whether successful execution of ERASE processing is to be logged and whether errors which occur during processing are to be reported on SYSOUT. Otherwise, errors are reported only via the return code field.

Default value: no additional log is created.

= ERRORS-TO-SYSOUT
Errors (with the exception of errors ignored via NOSTEP) are logged on SYSOUT.

= (area,len)
The names of all files affected by ERASE are written into a user output area.

`(area,len)` means: **`(adr / (r1),length / equ / (r2))`**

addr	Symbolic address of the output area.
r1	Register r1 contains the address of the output area.
length	Length of the output area, specified as a constant.
equ	Length of the output area, specified as an equate.
r2	Register r2 contains the length of the output area.

The entries are written sequentially into the output area. Each entry has the following format:

```
RL D RC pathname EC
```

RL	Record length (field length: 2 bytes); shows the length of an information unit.
D	2 bytes, reserved.
RC	Return code (field length: 4 bytes); this contains the DMS error code, which can be evaluated with the aid of the IDEMS macro.
pathname	Path name of the file (field length: variable).
EC	End criterion (field length: 2 bytes)

The end of the output in the output area is indicated with X'00' in EC field. In the event of an output area overflow, subcode 2 in the error code is set to '1' and processing continues without further logging.

Since the internal buffer areas are limited, an overflow may be indicated during RFA access, even though the user has provided a sufficiently large output area. In this case, output has been terminated prematurely.

MANCLAS

The user can select the files to be processed according to the HSMS management class for data backup to SM pubsets.

= *ANY
The HSMS management class is not a selection criterion.

= *NONE
Only files for which no HSMS management class is defined are selected.

= <c-string 1..8>
Only files with the specified HSMS management class are selected.

MF

The forms of the MF operand are described in detail in the appendix ("Macro types"). In all macros differentiated by the MF operand (MF=L/E/D), the version operand must contain the same value.
This description covers all operands supported in BS2000/OSD-BC >= V3.0.
VERSION=3 must be specified for generating this format.

MIGRATE

The user can use the migration attribute defined in the catalog entry (see the MIGRATE operand of the CATAL macro, "CATAL - Process catalog entry") to select files for processing with ERASE.

= ANY
The specified files are processed, regardless of what value is defined for MIGRATE in the catalog entry.

= ALLOWED
Only those files are processed for which the corresponding operand value has been specified in the catalog entry, i.e. files which may be migrated to storage levels S1 and S2.

= INHIBIT
Only those files are processed for which MIGRATE=INHIBIT has been specified in the catalog entry, i.e. files which may be briefly migrated (e.g. for reorganization purposes).

= FORBIDDEN
Only those files are processed for which the corresponding operand value has been specified in the catalog entry, i.e. files which must not be migrated.

= (list-of-migrate)
The ALLOWED and INHIBIT values may be specified by the user in a list. All files for which one of the specified values was defined in the catalog will be processed.

MOUNT

Only for files on private disks:
The user can specify if when erasing a file on private disk, only the first or all private disks involved are to be available online.
The MOUNT operand should be specified together with SPACE-CATALOG or DESTROY; it must be specified if DATA is specified.

Any MOUNT operand specified for tape files or files on public volumes is ignored.

Default value: MOUNT = FIRST-DISK

= FIRST-DISK

Only the private disk containing the beginning of the file and the catalog entry must be online.

= ALL-DISK
All private disks on which sections of the file are stored must be online. If any disk is missing, the file is not erased.

NOSTEP

The user can specify which errors are not to be reported via a return code in register 15.

= NONE
All errors are to be reported via return code.

= errcode
The user can specify the DMS error codes of the errors which are to be ignored, i.e. not reported via the return code field in the standard header.

`"error code"` means: `const / equ`

const	The error code is specified as a decimal or hexadecimal constant.
equ	The error code is specified as an equate.

It is advisable to use the equates generated by the IDEMS macro.

The list (errcode,...) may contain up to three elements.

OTHERAR

Selects and processes files based on the access rights that are defined via BASIC-ACL entries for all users other than the file owners's user group.

= ANY
BASIC-ACL entries for all users other than the file owner's user group are not a selection criterion.

= NO-ACCESS
Processes all files that may be accessed by users not belonging to the file owner's user group.

= access-list
Only processes files for which at least one of the listed access rights has been entered for users not in the file owner's user group in the BASIC-ACL entry.

“access list” has the following format:

- Long format:
  `([READ=YES / READ=NO][,WRITE=YES / WRITE=NO][,EXEC=YES / EXEC=NO])`
- Short format:
  `([R=Y / R=N][,W=Y / W=N][,X=Y / X=N])`

The parentheses form part of the operand values and are mandatory. The individual elements of the access list mean the following:

READ=YES or R=Y	Processes all files to which users not in the owner's user group have read access.
READ=NO or R=N	Processes all files to which users not in the owner's user group do not have read access.

WRITE=YES or W=Y	Processes all files to which users not in the owner's user group have write access.
WRITE=NO or W=N	Processes all files to which users not in the owner's user group do not have write access.
EXEC=YES or X=Y	Processes all files which users not in the owner's user group may execute.
EXEC=NO or X=N	Processes all files which users not in the owner's user group may not execute.

OWNERAR

Processes files selected on the basis of the access rights that are defined for the file owner in the BASIC-ACL entries.

= ANY
BASIC-ACL entries for file owners are not a selection criterion.

= NO-ACCESS
Processes all files that the owner is not allowed to access.

= access-list
Only processes files for which at least one of the listed access rights has been entered for the file owner in the BASIC-ACL entry.

“access list” has the following format:

- Long format:
  ([READ=YES / READ=NO][,WRITE=YES / WRITE=NO][,EXEC=YES / EXEC=NO])
- Short format:
  ([R=Y / R=N][,W=Y / W=N][,X=Y / X=N])

The parentheses form part of the operand value and must be specified. The individual elements of the access list mean the following:

READ=YES or R=Y	Selects all files that may be accessed by the owner for reading.
READ=NO or R=N	Selects all files that cannot be accessed by the owner for reading.
WRITE=YES or W=Y	Selects all files that can be accessed by the owner for writing.
WRITE=NO or W=N	Selects all files that cannot be accessed by the owner for writing.
EXEC=YES or X=Y	Selects all files that may be executed by the owner.
EXEC=NO or X=N	Selects all files that the owner is not allowed to execute.

PASS

The user can use the password type to select the files to be processed by ERASE.

= ANY
Password protection is not a selection criterion.

= NONE
Only files for which no passwords are defined are processed.

= EXPASS
Only files protected by an execute password are processed.

= RDPASS
Only files protected by a read password are processed.

= WRPASS
Only files protected by a write password are processed.

= (list-of-pass)
The user may specify more than one type of password in the form of a list. All files protected by one of the specified password types will be processed.

PASSWD

The user can specify one or more passwords to permit files protected by these passwords to be erased. The specified password must not be entered in the password table of the job. Note that the passwords entered here are valid only for the current ERASE macro.

The passwords must comply with the rules for password definition. The passwords are not reproduced in plaintext in any logs created by ERASE.

If no specification is entered, none of the files protected by a password will be processed by ERASE.

= ANY
No password is passed to ERASE.

= password
Protection by this password is to be cancelled.

= (list-of-passwd)
Up to three passwords may be specified in the form of a list.

POS

Only for file generations:
Wildcards may be used in “pathname” everywhere except in the generation number, which must be entered as an absolute or relative generation number. The generation identified by “pathname” must exist and is not erased.

Depending on the operand value AFTER/BEFORE, all younger or all older file generations are erased, and the catalog entry is updated as follows:

- If the oldest generation is erased, the generation specified in “pathname” becomes the oldest generation.
- If the youngest generation is erased, the generation specified in “pathname” becomes the youngest generation.
- If the generation with relative generation number 0 is erased, the generation specified in “pathname” becomes the base generation.

= AFTER
All generations selected by “pathname” and with a generation number greater than that specified in “pathname” are erased.
= BEFORE
All generations selected by “pathname” and with a generation number less than that specified in “pathname” are erased.

PREFIX = pre

Only in conjunction with MF=D:
“pre” is a 1-3 character string which replaces the corresponding string at the beginning of the generated names and thus creates macro-specific names. The first character of “pre” must be a letter.

PREFORM

Erases files depending on their (intended) file format on SM pubsets.

= *ANY
The file format is not a selection criterion.

= *NONE
Erases all files for which no PREFORM value was defined.

= *K
Erases all files with the intended file format *K.

= *NK2
Erases all files with the intended file format *NK2.

= *NK4
Erases all files with the intended file format *NK4.

= (list-of-preform)
Erases all files which have one of the specified file formats. The list may contain any values except for ANY.

PROTACT

The user can select files to be erased on the basis of the protection level provided by the highest activated access control.

When the file is accessed, the highest activated protection level applies. The following table shows the method used for access control, the protection attribute to be specified in the CATAL macro and the file protection hierarchy (protection levels):

Access control method	Protection attribute	Protection level
Standard access control	ACCESS and SHARE	0
Basic access control list	BASACL, OWNERAR, GROUPAR, OTHERAR	1
Access control using GUARDS	GUARDS	2

All other protection attributes of the file (e.g. passwords) are evaluated independently, without regard to the implemented protection level.

= ANY
The files to be processed are selected without regard to the protection level of the highest activated access control.

= LEVEL-0
Processes files for which access is controlled via standard access control.

= LEVEL-1
Processes files for which access is controlled via a basic access control list (BASIC-ACL protection).

= LEVEL-2
Processes files for which access is controlled by GUARDS.

= (list-of-protact)
The user may specify up to a maximum of three protection levels in a list. All files for which the protection level of the access control method matches one of those specified are selected.

RELSPAC

Selects files for processing on the basis of the lock to prevent the release of unused memory space (defined in the FILE macro or the MODIFY-FILE-ATTRIBUTES command). The lock can be defined in the catalog by using the CATAL macro.

= ANY
The lock preventing release of unused memory space is not a selection criterion.

= ALLOWED
Selects all files for which unused memory space may be released.

= IGNORED
Selects all files for which the release of unused memory space is not permitted.

SHARE

Selects files to be erased on the basis of whether or not they are shareable (see the SHARE operand in the CATAL macro).

= ANY
Shareability is not a selection criterion.

= YES
Processes all files that are shareable, i.e. which are also accessible to other user IDs under active standard access control.

= NO
Processes all files that are not shareable, i.e. only accessible to the owner under active standard access control.

= SPECIAL
Processes all shareable files (see YES) that can also be accessed by user IDs with hardware maintenance privileges.

= (list-of-share)
One or more operand values may be specified in a list.

SIZE

Only for disk files:
The user can use the file size or the size of the reserved space (= number of PAM pages) to select the files to be processed by ERASE. File generation groups and file generations are not included.

“number” specifies a number of PAM pages, where 0 <= number <= 2147483647.
Range specifications are inclusive of the limit values.

= ANY
The size of the reserved storage space is not a selection criterion.

= FSIZE
Only files for which space has been reserved but not actually used (LASTPG = 0) are processed, i.e. files which have not yet been opened.

= nmbr
Only files for which precisely the specified number of PAM pages have been reserved are processed.

= (nmbr[,])
Only file for which at least the specified number of PAM pages have been reserved are processed (SIZE >= nmbr).

= (,nmbr)
Only files for which not more than the specified number of PAM pages have been reserved are processed (SIZE <= nmbr).

= (nmbr1,nmbr2)
Only files for which a number of PAM pages in the range “nmbr1” to “nmbr2” have been reserved are processed.

SLEVEL

The user can use the storage level to select the files which are to be processed by ERASE. HSMS supports the following storage levels:

S0: implemented via disk storage devices with fast access (online processing).

S1: implemented via high capacity disk storage devices (online background level).

S2: implemented via magnetic tape or magnetic tape cartridge archives (offline background level).

= ANY
The specified files are processed regardless of the storage level on which they are stored.

= S0
Only those files stored on level S0 are processed.

= S1
Only those files stored on level S1 are processed.

= S2
Only those files stored on level S2 are processed.

= (list-of-slevel)
The user may specify up to 3 storage levels in the form of a list. Only the files on one of the specified storage levels will be processed.

STATE

The user can select files to be processed on the basis of their current processing state.

= ANY
The storage level is not a selection criterion.

= NOCLOS
Processes all files which have been opened for writing and are not closed. Such files include:

- normally open files (OPEN mode OUTIN, INOUT, OUTPUT)
- files not closed in a previous session
- files not closed in the current session because of job abortion.

= CLOSED
Processes all files that have already been closed, i.e. files not selected by NOCLOS.

= CACHED
Processes the files which are currently cached.

= NOT-CACHED
Processes all files which are not being currently processed in a cache.

= CACHE-NOT-SAVED
Processes all files for which it was not possible to save all data from the cache to a disk during closing.

= REPAIR-NEEDED
Processes all files which were not closed in an earlier session and which have not yet been verified (see VERIFY macro).

= DEFECT-REPORTED
Processes all files which may contain defective disk blocks.

= NO-OPEN-ALLOWED
Processes all files which cannot be opened due to data inconsistency.

= OPEN-ALLOWED
Processes all files that can be opened.

= (list-of-state)
A list of values may be specified (with a maximum of 4 file states). All files which are in one of the specified states are processed.

STOCLAS

The user can select the files to be processed according to the storage class for file storage on SM pubsets.

= *ANY
The storage class is not a selection criterion.

= *NONE
Selects all files for which no storage class is defined.

= <c-string 1..8>
Selects all files with the specified storage class.

STOTYPE

The user can select the files to be processed according to the storage type.

= *ANY
The storage type is not a selection criterion.

= *PUBSPACE
Only files on public volumes are selected.

= *NETSTOR
Only files on Net-Storage volumes are selected.

SUPPORT

The user can select the files to be processed by ERASE on the basis of the volume type. File generation groups and file generations are not included.

= ANY
Volume type is not a selection criterion.

= PUBLIC
Only files on public disks and on Net-Storage are processed.

= PRDISC
Only files on private disks are processed.

= TAPE
Only files on tapes or tape cartridges are processed.

= (list-of-support)
A list of values may be specified (with up to 3 volume types). All files which are stored on one of the specified volume types will be processed.

S0MIGR

Files are processed dependent on whether reallocation (migration) to the S0 level is allowed.

= *ANY
The migration allowance is not a selection criterion.

= *ALLOWED
All files for which migration within the S0 level is allowed are processed.

= *FORBIDDEN
All files for which migration within the S0 level is not allowed are processed.

= (list-of-s0migr)
The user can specify the desired values in a list. All files for which one of the specified values is defined in the catalog are processed.

TIMBASE

Defines whether the absolute date definitions are in UTC or local time. This affects the CRDATE, DELDATE, EXDATE, LADATE and LCDATE operands. Relative dates are always based on local time.

= *UTC
Absolute dates are specified in UTC time.

= *LTI
Absolute dates are specified in local time.

TYPE

Processes files selected on the basis of their file type. The TYPE operand also determines which selection criteria are evaluated for ERASE processing. File generation groups and file generations are not taken into account by any of the selection parameters.

= ANY
ERASE processes “normal” files, file generation groups and file generations. However, FGGs and file generations are ignored by some of the selection operands in order to avoid the creation of gaps in the sequence of generations.

= FILE
File generation groups and file generations are not processed by ERASE; all other selection operands are evaluated.

= FGG
Processes file generation groups and file generations only. The only sensible selection operands for TYPE=FGG are those which refer to attributes that are identical for all generations of an FGG (ACCESS, BACKUP, CCS, DELDATE, EXDATE, MANCLAS, MIGRATE, PASS, RELSPAC, SHARE, SUPPORT=PRDISC and WORKFIL).

File generation groups or file generations are not selected if:

- the selection operand VOLUME is not specified together with CATALOG or does not designate a private disk,
- a selection operand that is not the same for all generations/FGGs is specified.

= PLAM
Processes PLAM libraries. This is a subset of the files which are selected by the TYPE=FILE specification.

= (list-of-type)
A maximum of three file types can be specified by the user in the form of a list. Only the files which match one of the specified file types are selected.

USRINFO

The user can select files/file generations for processing dependent on the user-specific metainformation.

= *ANY
The user-specific metainformation is not a selection criterion.

= *NONE
All files possessing no user-specific metainformation are processed.

= <c-string 1..8>
All files with the specified user-specific metainformation are processed.

VERSION

Specifies which version of the parameter list is to be generated.

= 0
Default value: generates the parameter list format that was supported prior to BS2000 V9.5A.
This format will, however, only support parameters which were known at that time. For example, the path name can only be specified without wildcards, and only VOLUME and POS are permitted as selection parameters. The supported operands and operand values can be found in section "Variations in versions - VERSION=0/1/2".

= 1
Generates the parameter list format that was supported in BS2000 V9.5 and V10.0. This format will support only parameters which were known at the time. The supported operands and operand values can be found in section "Variations in versions - VERSION=0/1/2".

= 2
Generates the parameter list format for versions as of BS2000/OSD V1.0.

= 3
Generates the parameter list format for versions as of BS2000/OSD V3.0.

Note

If existing software which manipulates the generated parameter list is to be recompiled or reassembled, the old format must be requested. Otherwise, source compatibility is ensured.

VOLSET

The user can select the files to be processed via the volume set on which they reside.

= *ANY
The volume set is not a selection criterion.

= <c-string 1..4>
All files on the specified volume set are selected.

VOLUME

The user can select files to be processed on the basis of the VSN (volume serial number) of the disks on which they are stored.

= *ANY
All files are processed regardless of the VSNs of their volumes.

= vsn
Processes all files that occupy storage space on the specified volume. If any of the action operands SPACE, SPACE-CATALOG, DATA and DESTROY is specified at the same time, no file generations and file generation groups are selected. If the action operand CATALOG is specified with the VSN, no file generations on tape are selected.

WORKFIL

The user can select the files on SM pubsets to be processed dependent on whether they can be deleted by the system administrator (work files).

= *ANY
Whether or not the files are work files is not a selection criterion.

= *NO
All files that are not work files are processed.

= *YES
All work files are processed.

Programming notes

1. The error code is now only stored in the standard header of the parameter list and no longer in register 15 as was the case up to VERSION=2.
2. Error code 06D6 – “filename” was partially qualified and the system could not erase all matching files.
3. Error code 05DF – *SYSOUT not permitted in interactive mode.
4. In specific error cases (parameter range not accessible or not set up), program termination with STXIT connection is initiated.

Return codes

Standard header: `ccbbaaaa`

The following code relating to execution of the ERASE macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	No error
	X'40'	X'0501'	Requested catalog not available
	X'82'	X'0502'	Requested catalog in the rest state
	X'40'	X'0503'	Incorrect information in the MRSCAT
	X'82'	X'0504'	Error in catalog management system
	X'40'	X'0505'	Computer communication error (MRS)
	X'80'	X'0506'	Operation canceled because of master change
	X'40'	X'0510'	Error while calling an internal function
	X'40'	X'0512'	Requested catalog unknown
	X'40'	X'051A'	File already exists
	X'40'	X'051B'	User ID not known in specified pubset

X'40'	X'051C'	No access right to specified pubset
X'40'	X'051D'	LOGON password different on specified pubset
X'20'	X'0530'	Error in storage space request
X'20'	X'0531'	Unexpected catalog access error
X'82'	X'0532'	File in use, therefore locked
X'40'	X'0533'	File not found
X'82'	X'0534'	Private volume cannot be allocated
X'40'	X'0535'	No access right to the file catalog entry
X'20'	X'053B'	System error during file access
X'40'	X'053D'	Catalog or F1 label block is destroyed
X'82'	X'053F'	File reserved by another task
X'20'	X'054F'	Unexpected error during access to JOIN file
X'40'	X'055C'	Catalog entry on private disk not found
X'01'	X'0571'	System file declared as *DUMMY
X'40'	X'0572'	System file not assigned to a DMS file
X'40'	X'0574'	DMS error during deletion of a system file
X'82'	X'0575'	System command active for this system file
X'01'	X'0576'	Contradictory operand combination or reserved fields of the parameter area used
X'20'	X'0577'	Internal error during access to job environment
X'20'	X'0578'	Internal error during checking of access rights
X'40'	X'057C'	HSMS has rejected recall
X'40'	X'057D'	HSMS file displaced. Cannot be retrieved without delay.
X'40'	X'057E'	HSMS not available
X'82'	X'0594'	Not enough virtual memory available. This return code can also occur in particular in conjunction with a selection specification (wildcard) if too many files are selected
X'01'	X'0599'	Operand is not supported in the RFA-BS version
X'01'	X'05AB'	Address of output area incorrect or not specified
X'01'	X'05AC'	Incorrect second operand
X'40'	X'05B3'	Only the system administrator is allowed to specify a foreign user ID

	X'40'	X'05BF'	File protected with password
	X'82'	X'05C3'	File generation to be erased is locked
	X'01'	X'05C5'	SPACE specification for files on private disks is not allowed
	X'40'	X'05C6'	Release data does not permit deletion of file
	X'20'	X'05C7'	Internal error in DMS
	X'01'	X'05C9'	Only files on private volumes can be exported
	X'82'	X'05D0'	File in use, therefore locked
	X'01'	X'05DE'	File name not found or not allowed
	X'01'	X'05EE'	File name too long
X'02'	X'00'	X'05F7'	File generation does not exist but group entry will be changed
	X'01'	X'05FA'	Access to remote imported pubset not possible
	X'40'	X'05FC'	Specified user ID not in home pubset
	X'40'	X'0609'	Action for system file not permitted
	X'40'	X'0640'	Access to Net-Storage is rejected by the ONETSTOR subsystem because of communication problems with the net client
	X'40'	X'0643'	Net client reports access error
	X'40'	X'0644'	Net client reports internal error
	X'40'	X'0645'	File does not exist on Net-Storage
	X'40'	X'0649'	Net server reports POSIX ACL error
	X'40'	X'064A'	Net client reports that access to files on the Net-Storage volume is forbidden
	X'40'	X'064B'	Access to node files from the net client not supported
	X'40'	X'064C'	Directory of the specified user ID does not exist on the net server
	X'40'	X'0666'	File is write-protected by GUARDS
	X'20'	X'069D'	Incorrectly structured catalog entry
X'01'	X'00'	X'06B4'	No generation to be erased exists before or after the specified generation
	X'01'	X'06C7'	Invalid generation number specified
	X'40'	X'06CC'	Only with selection specification (wildcard): no file matches the selection specification
	X'01'	X'06D4'	Invalid generation specification
	X'40'	X'06D5'	File protected

X'02'	X'00'	X'06D6'	Error during deletion of some files
	X'01'	X'06F5'	No authorization for use of specified operands (TPR or TSOS required)
	X'01'	X'06F9'	Either the file name or the volume must specified
	X'01'	X'06FD'	Parameter area invalid or not accessible
	X'40'	X'06FF'	BCAM connection aborted
	X'01'	X'FFFF'	Wrong function number in parameter area header
	X'03'	X'FFFF'	Wrong version number in parameter area header

Variations in versions - VERSION=0/1/2

The “version overview” in the table below shows which operands and operand values are supported with VERSION=2/1/0.

All operands and operand values that were supported up to and including BS2000/OSD-BC V2.0A may be used in the macro format with VERSION=2.

All operands and operand values that were supported up to and including BS2000 V10.0A may be used in the macro format with VERSION=1.

All operands and operand values that were supported in BS2000 versions <= V9.0A can be used in the format with VERSION=0..

Operand	Vers=0	Vers=1	Vers=2	Comments
MF=E	x	x	x	
VERSION	x	x	x	
MF=D	-	x	x	
PREFIX	-	x	x	
VERSION	-	x	x	
MF=L	x	x	x	
*	x	x	x	
*SYSid	x	x	x	
*DUMMY	-	x	x	
CATALOG	x	x	x	
DATA	x	x	x	
DATA-KEEP-ATTR	-	-	-	
-DELETE-OR-EXPORT	-	x	x	
DESTROY	x	x	x	
pathname	x	x	x	*Vers=0:* No wildcards are permitted
prefix	-	x	x	
SPACE	x	x	x	

SPACE-CATALOG	-	x	x	
ACCCNT	-	-	x	
ACCESS	-	x	x	

ADMINFO	-	-	-	
AVAIL	-	-	-	
BACKUP	-	x	x	*Vers=1:* Operand value (list-of-backup) not permitted
BASACL	-	-	x	
BLKCNT	-	-	x	
BLKCTRL	-	x	x	*Vers=1:* Operand values ANY, DATA4K, DATA2K, NK4, NK2, NK and (list-of-blkctrl) not permitted
CCS	-	-	x	
CHECK	-	x	x	
CRDATE	-	x	x	*Vers=1:* Operand values cannot be entered with time specifications, see[1)]
DELDATE	-	-	-	
DISKWR	-	-	x	
ENCRYPT	-	-	-	
EXDATE	-	x	x	*Vers=1:* Operand values cannot be entered with time specifications, see[1)]
EXTENTS	-	x	x	
FCBTYPE	-	x	x	*Vers=1:* Operand value (list-of-fcbtype) not permitted
FILTYPE	-	-	-	
FSIZE	-	x	x	
GROUPAR	-	-	x	
GUARDS	-	-	x	
IGNORE	-	x	x	
IOPERF	-	-	x	
IOUSAGE	-	-	x	
LADATE	-	x	x	*Vers=1:* Operand values cannot be entered with time specifications, see[1)]
LASTPAG	-	-	x	
LCDATE	-	-	x	
LIST	-	x	x	

MANCLAS	-	-	-	
MIGRATE	-	x	x	*Vers=1:* Operand value (list-of-migrate) not permitted *0/1/2:* Operand value FORBIDDEN not permitted
MOUNT	-	x	x	

NOSTEP OTHERAR OWNERAR	- - -	x - -	x x x	
PASS	-	x	x	*Vers=1:* Operand value (list-of-pass) not permitted
PASSWD	-	x	x	
POS	x	x	x	
PREFIX	-	x	x	
PROTACT	-	-	x	
RELSPAC	-	-	x	
SHARE	-	x	x	*Vers=1:* Operand value (list-of-share) not permitte
SIZE	-	x	x	
SLEVEL	-	x	x	
STATE	-	-	x	*Vers=2*: Operand values CACHE-NOT-SAVED and DEFECT-REPORTED not permitted
STOCLAS	-	-	-	
SUPPORT	-	x	x	*Vers=1:* Operand value (list-of-support) not permitted
S0MIGR	-	-	-	
TIMBASE	-	-	-	
TYPE	-	-	x	
USRINFO	-	-	-	
VERSION	x	x	x	
VOLSET	-	-	-	
VOLUME	x	x	x	*Vers=0:* Only possible value: vsn *Vers=1:* Only possible value: vsn
WORKFIL	-	-	-	

Key

x The operand is available in the macro version.

- The operand is not available in the macro version.

Vers Version

In the above table, positional operands are indicated before keyword operands under MF=L.

Note

1) The format for the CRDATE, EXDATE and LADATE operands in macro version 1 is as follows:
   `CRDATE / EXDATE / LADATE = NONE / date / (date[,]) / (,date) / (date1,date2)`

EXLST - Define exit address list

Macro type: type O

By means of the EXLST macro, the user defines a list of symbolic addresses which point to program routines for evaluating and handling events which cause normal processing to be interrupted.

There is a separate operand in the EXLST macro for each event. If a symbolic address is specified in the macro for the related operand, DMS can branch to the appropriate program routine if a given event occurs. If the operand is omitted or contains only a null string, the occurrence of the event will result in abortion of the program unless a common error handling routine is addressed via the operand COMMON.

However, this routine is not called for the operands with which the FCB can be modified during OPEN processing or with which the user writes his own tape labels during OPEN or CLOSE processing. If these operands are specified in the EXLST macro, DMS branches to the program routines which they address.

Format

Operation	Operands
EXLST	**[PARMOD = 24 / 31]** **[,CLOSER = NO / relexp]** **[,CLOSPOS = NO / relexp]** **[,COMMON = relexp]** **[,DLOCK = NO / relexp]** **[,DUPEKY = NO / relexp]** **[,EOFADDR = NO / relexp]** **[,ERRADDR = NO / relexp]** **[,ERROPT = NO / SKIP / IGNORE / relexp]** **[,ISPERR = NO / relexp]** **[,LOCK = NO / relexp]** **[,NODEV = NO / relexp]** **[,NOFIND = NO / relexp]** **[,NOSPACE = NO / relexp]** **[,OPENC = NO / relexp]** **[,OPENER = NO / relexp]** **[,OPENX = NO / relexp]** **[,OPENZ = NO / relexp]** **[,PASSER = NO / relexp]** **[,PGLOCK = NO / relexp]** **[,SEQCHK = NO / relexp]** **[,USERERR = NO / relexp]** **[,WLRERR = NO / relexp]** **[,EOVCTRL = NO / relexp]** **[,LABEND = NO / relexp1]** **[,LABEOV = NO / relexp]** **[,LABERR = NO / relexp]** **[,LABGN = NO / relexp]** **[,OPENV = NO / relexp]**

Operand descriptions

relexp

Symbolic address in the Assembler program.

PARMOD

Specifies the generation mode for the macro.

Default value: the value preset for the generation mode by means of the GPARMOD macro or by the assembler.

= 24
The macro is expanded in accordance with the format for the 24-bit interface. The object code is thus executable only in 24-bit addressing mode, i.e. in the 16-Mb address space.

= 31
The macro is generated as addressing mode-independent; the object code can run in the 2-Gb address space.

CLOSER

An error has occurred during CLOSE processing. An error code describing the condition in more detail is stored in FCB field ID1ECB.

CLOSPOS

For tape input files with nonstandard labels:
The user can use BTAM macros to position a tape for which a CLOSE macro with either REPOS or LEAVE was called. CLOSE processing continues after execution of the CLOSPOS routine. If this exit is not defined, the system handles tape positioning.

COMMON

Control is passed to this address if there is no special exit for the type of error encountered.

Exceptions:

CLOSPOS	LABEOV	OPENC	OPENZ
EOVCTRL	LABERR	OPENV	WLRERR
LABEND	LABGN	OPENX	

If processing is interrupted by an event for which no EXLST operand is specified, control is passed to the routine addressed via COMMON, except for the operands listed above, which “expect” an intervention in the processing.

For further details see the tables at the end of the EXLST description.

DLOCK

For UPAM processing of disk files only:
Deadlock: further locks, which are not available, have been requested for a job which already has locks active.

Control is passed to the deadlock exit after the waiting time specified in the PAMTOUT operand. The DLOCK routine must first release the “old” locks before new locks can be requested; otherwise, the program will be aborted.

DUPEKY

For ISAM only:
An attempt has been made to write a record into an ISAM file, although the key of this record already exists in the file. This attempt was made:

- using the INSRT macro, which must not be used for writing records with duplicate keys, even if DUPEKY=YES is specified in the FILE or FCB macro, or
- using the PUT macro, but without DUPEKY=YES specified in the FILE or FCB macro.

EOFADDR

For input files only:
The end of the file was encountered when trying to read a record.

After checking all the labels, DMS activates the EOFADDR routine, in which the user can close the file.

EOVCTRL

This exit must be specified in order to transfer control from the system to the user after a new tape volume has been mounted.

For input files:
This exit is activated only for multivolume files. It is called for every continuation tape only after the system has checked the standard header labels on the continuation tape and the user has (if necessary) checked his own labels in the LABGN exit.

For output files:
This exit is activated only in the case of multivolume files, and only for the continuation tapes, after the system has finished checking and writing the standard header labels (VOL, HDR).

This exit is left by specifying the EXRTN macro.

ERRADDR

During file processing, a hardware error or abnormal I/O termination has occurred. The status byte, the standard device bytes, the Executive flag byte (EFB) and the three sense bytes are stored in the FCB. For ISAM processing, this exit may indicate that ISAM blocks are inconsistent (SAM uses the exits EOFADDR, ERROPT, USERERR).

ERROPT

For SAM input files:
A parity error occured while reading a block: DMS makes several attempts to read the block again before flagging it as errored and attempting to branch to the ERROPT routine. If no ERROPT error exit is provided, the program is terminated. In the EXLST macro, the user can choose to ignore the error, to skip the block or to handle the error in an ERROPT routine.

For SAM output files:
ERROPT is relevant only if no WLRERR exit has been defined and records of incorrect length occur.

= SKIP
The faulty block is skipped, i.e. no records from this block are made available to the program for processing. The next block is read and processing continues with the first record of this block.

= IGNORE
The error condition is ignored and the records from the block are made available to the user program for further processing.

= relexp
Symbolic address of a user routine. This routine must not contain any GET macros for the file containing the faulty block. The routine may evaluate register 0, which contains the address of the errored block, and set a flag for further processing (also in register 0). Normal processing is resumed by means of the EXRTN macro.

If general-purpose register 0 contains the value X'00000001', the current block is skipped and processing continues with the next block. Any other code signifies that the current block is to be processed as if no error had occurred.

ISPERR

For ISAM files with index and data sections on different private disks:
Not enough space is available for extending the index section (for this type of ISAM file, the space can be extended separately for the index and data sections (see the FILE macro)).

LABEND

For tape files:
For checking/writing user file trailer labels; if this exit is not used, the system ignores all user labels.

For input files:
After encountering the end of the file, the system activates the LABEND exit before activating the EOFADDR exit. In the LABEND routine, the user can check his end-of-file labels (UTL). The system provides him/her with the address of a UTL in register 0.

In the case of input files with nonstandard labels, the LABEND exit provides the user with a means of reading and checking his/her labels, if present, when the EOF condition is encountered.

For output files:
The LABEND exit is activated for the purpose of writing user labels after the file has been closed by the user and the end-of-file labels (EOF) have been written. The system supplies the user, in register 0, with an address at which the user trailer labels (UTL) must be made available.
In the case of output files with nonstandard labels, the user, after issuing the CLOSE macro, receives control at this exit for the purpose of writing the nonstandard labels.

Once control has been returned to the system (via the LBRET macro), CLOSE processing is completed.

LABEOV

For tape files:
For checking/writing user trailer labels.

For input files:
This exit is activated after EOF has been encountered and the trailer labels have been written. The user can check the user labels (UTL) here. The system provides the user with the address of a UTL in register 0.
In the case of input files with nonstandard labels, this exit enables the user to read and check the user labels following his/her file, if there are any such labels.

For output files:
This exit is activated when EOV is encountered (or is explicitly requested by the user via the FEOV macro) and after the EOV labels have been written. The system provides an address in register 0 at which the user must make his/her trailer labels (UTL) available.In the case of output files with nonstandard labels, the user receives control from the system via this exit either after EOV is encountered or when a FEOV macro is issued, in order to write his /her labels.

Users working with the BTAM access method must issue an FEOV macro after encountering the end of the tape in the ERRBYTE field. SAM initiates EOV processing automatically.
After control has been returned to the system (by means of the LBRET macro), a tape swap can be executed.

LABERR

For tape files:
For files with standard labels, the system branches to this exit if an error occurs during end-of-tape processing. One of the following error codes is placed in the ID1ECB field of the FCB:

X'0DE9' A tape mark was read instead of the expected EOV/EOF labels.

X'0DEA' A record other than the expected EOV/EOF label was read.

X'0DEB' End-of-tape (double tape mark) was encountered without the expected EOV/EOF labels being read.

X'0DEC' The check of the block count was negative.

The user must return control to the system by means of the EXRTN macro.

Register 0 must contain one of the following values in order to notify the system of how it is to proceed:

X'00' Perform a tape swap as if the correct EOV/EOF labels had been read.

X'01' Terminate the task with an error.

X'02' Perform EOF processing.

If the LABERR exit is not used and one of the conditions described above occurs, the system sends the user an error message. (S)he can then either continue the program or activate a CLOSE routine.

If this exit is not specified, the system ignores user labels for input files, and no user labels can be created for output files.

LABGN

For tape files:

For input files:
The LABGN exit is activated after the standard header labels (VOL, HDR) have been checked; the user header labels (UHL) can be checked in the LABGN routine. The address of the label in the buffer is passed to the user program in register 0. If the file contains nonstandard labels, the user can read and check his/her labels in the LABGN routine.

For output files after OPEN processing and after checking and writing the standard header label:
The system provides the user, in register 0, with the address at which he must make his labels (UHL) available. If the file is to have nonstandard labels, these can be written in the LABGN routine.

When control is returned to the system (by means of the LBRET macro), the tape must be positioned either before the first data block (for OPEN=REVERSE) or after the last data block (for OPEN=EXTEND). If TPMARK=YES was specified, the tape is rewound by one tape mark for OPEN INPUT or wound forward by one tape mark for OPEN REVERSE. If TPMARK=NO was specified, the system assumes that the tape is positioned before the first data block.

LOCK

The file is locked: it cannot be opened because it has already been opened by another job and the OPEN modes are incompatible, i.e. at least one of the jobs has requested an OPEN mode other than INPUT.

NODEV

Either no device on which the private volume could be mounted is free, or the private volume is already being used by another user (it is advisable to reserve devices beforehand).

NOFIND

For ISAM:
The action macro GETKY, ELIM (with KEY specified) or GETFL could not be executed successfully:

- GETKY / ELIM: there is no record with the specified key in the file
- GETFL: the specified file range contains no records which match the flag conditions.

NOSPACE

For disk files:
The required storage space cannot be provided.

OPENC

The file was opened earlier as an output file but was not closed properly. The VERIF macro can be used in the OPENC routine to close the file and restore its consistency. Field ID1ECB in the FCB contains DMS error code X'0DD1'. If no OPENC exit is specified, the system continues with OPEN processing.

OPENER

An error occurred when opening a file (e.g. inconsistent FCB). An error code describing this condition more precisely is stored in the FCB.

OPENV

For tape files:

For files with standard labels:
This exit is used to check (for input files) or write (for output files) the UVL labels.
DMS provides, in register 0, the address where the label can be found for input files or where the label must be made available by the user for output files. For output files, OPENV can be used for positioning or for writing up to 9 user labels (UVL). The user can issue any BTAM macros which may be required.

For input files with nonstandard labels:
The OPENV exit enables the user to read and check his/her volume header labels (VOL), if any.

Before returning control to the system (via the EXRTN macro), the user must position the tape before the file header labels (HDR), if any. Positioning can also be carried out with the FSEQ operand (see the FILE or FCB macro).
The user is responsible for correct positioning. The user program cannot perform tape swaps during positioning.

OPENX

The FCB has already been updated, during OPEN processing, on the basis of values in the TFT or the catalog. The program can now check that all parameters are set such that OPEN processing can be completed without errors. If OPEN=OUTPUT/OUTIN, the FCB is updated on the basis of the TFT, but the catalog entry is not updated. Processing can be continued after calling the EXRTN macro.

OPENZ

For files being opened in OUTPUT or OUTIN mode, catalog processing has already been completed when this exit is activated, but the remaining OPEN processing must still be performed. In the OPENZ routine, the program can modify the FCB so that processing is possible: the user can, for example, process the file with an access method other than that specified in the catalog.

Processing is continued after the EXRTN macro has been called.

PASSER

An invalid password has been specified for a protected file.

PGLOCK

Only together with SHARUPD=YES for UPAM or ISAM:
Locks requested by the calling job cannot be set, because they are already set by another job. However, there is no danger of a deadlock situation.

UPAM:
The waiting time specified in the PAMTOUT operand of the FCB has elapsed when control is passed to this exit.

ISAM:
If a PGLOCK routine exists, ISAM does not wait for a record lock. If PGLOCK=NO applies, the job is entered in a queue; the user is not informed, in this case, that the record is already locked by another job.

If a file was opened with SHARUPD=YES, control can be passed to this exit during processing of any ISAM macro (except OSTAT). If the PGLOCK exit is taken, the “internal pointer” will be wrong unless the condition is due to a PUTX or ELIM macro (without KEY).

It is therefore imperative that this internal pointer is repositioned before any macro is issued which requires it to be set correctly (e.g. GET, GETR and GETFL). The pointer can be repositioned using the RETRY macro or with one of the ISAM action macros GETKY, SETL, PUT, STORE, INSRT or ELIM (with KEY). If GET, GETR or GETFL is called before the pointer is repositioned, control is passed to the USERERR exit.

If the condition which caused the PGLOCK exit to be taken was due to a PUTX or ELIM macro (without KEY), the data block remains locked and repositioning is not necessary.

SEQCHK

For ISAM:
A record to be added to an ISAM file by means of the PUT macro has a key less than the highest key already present in the ISAM file.

USERERR

The program attempted to execute an illegal or invalid action, such as trying to write to a file opened in INPUT mode, calling a PUTX or ELIM macro (without KEY) for a file opened with SHARUPD=YES, or an invalid PAM operation code.

WLRERR

A record of invalid length was read. In the case of blocked records of fixed length, the record length is regarded as invalid if the block size is not a multiple of the record length defined in the FCB entry RECSIZE (up to the maximum block size defined in the FCB entry BLKSIZE). This permits short blocks of logical records to be read without “invalid record length” being flagged. In the case of variable-length records, the record length is invalid if it no longer matches the record length specified in the block count control field.

For files with record format U, no WLRERR exit needs to be provided, because the record length is not checked in this case.

If a program is given control at this exit, general-purpose register 0 contains the address of the errored block. An EXRTN macro is required if processing is to continue. If register 0 contains X'00000001', the current block is skipped and processing continues with the next block. Any other code signifies that the current block is to be processed as if no error had occurred.

If no special form of error handling is defined for WLRERR and an “invalid record length” event occurs, DMS checks the value of the ERROPT operand:

- ERROPT NO: the “invalid length” record is treated as an errored block and control is passed to the address specified in ERROPT.
- ERROPT = NO: the job is aborted.

Programming notes

1. Registers 14, 15, 0 and 1 are DMS parameter registers. Therefore, unless explicitly stated otherwise, it cannot be assumed that these registers have a defined value when the user receives control at the EXLST exit.
2. The following tables show which EXLST exits are used, and when:

Error exit	STD SAM	NSTD SAM	ISAM	PAM	BTAM	Spec. CLOSE	Spec. EXRTN	Spec. Action macro
CLOSER	A	A	A	A	A	N	N	N
CLOSPOS	N	A	N	A	A	N	A	A
DLOCK	N	N	N	A	N	A	N	A
DUPEKY	N	N	A	N	N	A	N	A
EOFADDR	A	A	A	A(1)	A(2)	A	N	A
EOVCTRL	A	A	N	N	A	A	A	N
ERRADDR	A	A	A	A	A	A	N	A
ERROPT	A	A	N	N	N	A	A	N
ISPERR	N	N	A	N	N	A	N	A
LABEND	A	A	N	A	A	X	A	A(3)
LABEOV	A	A	N	A	A	X	A	A(3)
LABERR	A	X	N	A	A	X	A	A(3)
LABGN	A	A	N	A	A	X	A	A(3)
LOCK	A	A	A	A	N	N	N	A
NODEV	A	A	A	A	A	N	N	N
NOFIND	N	N	A	N	N	A	N	A
NOSPACE	A	N	A	N	N	A	N	A/N(SAM)
OPENC	A	N	A	A	N	N	N	Try VERIF
OPENER	A	A	A	A	A	N	N	N
OPENV	A	A	N	A	A	X	A	A(3)
OPENX	A	A	A	A	A	A	A	N
OPENZ	A	A	A	A	A	A	A	N
PASSER	A	A	A	A	A	N	N	N
PGLOCK	N	N	A	A	N	A	N	A

SEQCHK	N	N	A	N	N	A	N	A
USERERR	A	A	A	A	A	A	N	A
WLRERR	A	A	N	N	N	A	A	N

Meanings of the entries

EXLST operand	COMMON exit	FCB code (ID1XITB field) X' '	1) Program abortion	Meaning
CLOSER	Z	2C	A	File regarded as closed
CLOSPOS	N	70	N	Only for files with standard labels
DLOCK	Z	3C	A	No further locks until all existing locks are cleared
DUPEKY	Z	54	A	
EOFADDR	Z	40	A	
EOVCTRL	N	34	N	
ERRADDR	Z	44	A	Error code and/or end byte (PAM and BTAM)
ERROPT	N	48	A	End byte stored in FCB
ISPERR	Z	50	A	
LABEND	N	30	N	File regarded as closed
LABEOV	N	28	N	-
LABERR	N	6C	N	Only for files with standard labels
LABGN	N	24	N	-
LOCK	Z	10	A	
NODEV	Z	14	A	
NOFIND	Z	58	A	
NOSPACE	Z	4C	A	
OPENC	N	68	N	
OPENER	Z	08	A	Error code stored in FCB
OPENV	N	1C	N	
OPENX	N	04	N	
OPENZ	N	18	N	
PASSER	Z	0C	A	
PGLOCK	Z	38	A	

SEQCHK	Z	60	A	
USERERR	Z	5C	A	Error code stored in FCB
WLRERR	N	64	N	End byte stored in FCB

where:	
A	Allowed
X	Not allowed
Z	Applicable
N	Not applicable
(1)	Only for *DUMMY files
(2)	Only used with FEOV
(3)	Only if LABEL=NSTD is specified.
1)	The program is terminated if the exit is not defined and the event occurs.

EXRTN - Return from error routine

Macro type: type R

The EXRTN macro is required in some user routines which are addressed via EXLST exits. It returns control to DMS, which evaluates the function code and continues processing accordingly.

The EXRTN macro must be specified in routines which use the following EXLST exits: CLOSPOS, EOVCTRL, ERROPT, LABEND, LABEOV, LABERR, LABGN, OPENV, OPENX, OPENZ, WRLERR.
If UHL, UTL or UVL labels are being processed, the LBRET macro must be used.

Format

Operation	Operands
EXRTN	**fcbadr / (1)** **0 / 1 / 2 / (0)** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Symbolic address of the FCB for the file being processed when a branch was made in the program to an EXLST exit.

(1)

Register 1 contains the address of the FCB.

The second operand specifies a function code; this is significant only for the EXLST exits ERROPT and WRLERR.

0

The error is to be ignored.

1

Disk files: the current block is to be skipped and the next one processed.

Tape files: the user program is to be terminated with an OPEN or end-of-tape error.

2

Tape files: end-of-tape processing is to be continued (for the LABERR exit only).

(0)

The rightmost byte of register 0 contains the function code.

PARMOD

Specifies the generation mode for the macro.

Default value: the value preset for the generation mode by means of the GPARMOD macro or by the assembler.

= 24
The macro is expanded in accordance with the format for the 24-bit interface. The object code is thus executable only in 24-bit addressing mode.

= 31
The macro is generated as addressing mode-independent.

Programming note

The EXRTN macro overwrites registers 0, 1, 14 and 15.

FCB - Define file control block

Macro type: type O

The file control block is the central source of information for the access methods BTAM, ISAM, SAM and UPAM. A file control block is required whenever a file is processed.

DMS takes the necessary information from various sources:

- the user can specify values for FCB fields in the FCB macro;
- in user programs, values can be placed in FCB fields during program execution, but before the file is opened;
- specifications in the FILE macro with the appropriate link name are placed in the TFT and are given precedence over the corresponding FCB fields when the file is opened;
- the FCB can also be modified while the file is being opened using routines in the application program (see the EXLST macro, "EXLST - Define exit address list", OPENX and OPENZ routines).
- further information is taken from the catalog entry for the file during OPEN processing.

The events which lead to the construction of a complete FCB are listed below in chronological order:

- FCB macro during assembly
- modification during program execution, before the OPEN macro is called
- at OPEN time, construction of the FCB
  - from the TFT (FILE call with appropriate link name)
  - from the catalog entry (regardless of the OPEN mode)
  - in OPEN routines (see the EXLST macro, "EXLST - Define exit address list", OPENX and OPENZ routines) by the user program

The complete FCB also sets up the connection to the logical routines which handle the blocking and unblocking of records for the record-oriented access methods (SAM, ISAM).

DSECTs

Using the IDFCB macro, a DSECT can be generated for the FCB so that the user can address its fields symbolically. If files are processed via the 24-bit interface, a DSECT for the FCB extension can be generated by means of the IDFCBE macro.

NULL operands

If the file/file generation group specified directly in the FCB macro or indirectly via the LINK operands already exists (i.e. if the file has already been opened using OPEN modes OUTPUT or OUTIN), certain FCB macro operands can be specified as “NULL operands”. This means that the operand is specified but no operand value (i.e. the operand value is an empty character string).

FCB LINK=name, FCBTYPE=, BLKSIZE=, RECFORM=, ...

For a list of the operands that can be specified as NULL operands, see table "Operands for disk files".

FCB and access methods

Various access methods use the FCB, but each only evaluates certain operands. The following table provides an overview of which access methods use which operands. Operands which an access method cannot evaluate are ignored. No error message is issued.

Operand in FCB	Meaning	BTAM	ISAM	PAM	SAM
BLIM	Tape only: number of data blocks per tape				x
BLKCTRL	Block format (UPAM: tape)		x	x	x
BLKSIZE	Length of data block (UPAM: tape)	x	x	x	x
BTAMRQS	Number of I/O requests	x			
BUFOFF	Tape only: buffer offset				x
CHAINIO	Chaining factor	x			
CHKPT	Tape only: checkpoint				x
CODE	Translation table (SAM: tape)	x			x
DUPEKY	Duplicate key		x		
EXIT	Error exit	x	x	x	x
FCBTYPE	Access method	x	x	x	x
FILE	Designates the file to be processed	x	x	x	x
FORM	Memory space reservation				x
FSEQ	Tape only: number of a file within a file set	x			x
IOAREA1	Program buffer	x	x	x	x
IOAREA2	Second program buffer	x	x	x	x
IOPERF	Performance attribute (pubset only)		x	x	x
IOREG	Register for file processing in locate mode		x		x
IOUSAGE	Usage of cache (pubset only)		x	x	x
KEYARG	Address of field containing ISAM key		x		
KEYLEN	Length of ISAM key		x		
KEYPOS	Beginning of ISAM key		x		
LABEL	Tape only: label attributes	x			x
LARGE_ FILE	Disks only: file size allocation over 32 GB		x	x	x
LBP_ REQUIRED	last-byte pointer (LBP) processing requested			x	

LINK	File link name	x	x	x	x
LOCKENV	Lock log for synchronization (shared-update processing)		x	x	
LOGLEN	Length of logical flag		x		

OPEN	OPEN mode	x	x	x	x
OPTION	Options	x	x	x	x
OVERLAP	Overlapping		x		
PAD	Block padding for files created sequentially		x		
PAMREQS	Asynchronous I/O operations			x	
PAMTOUT	Waiting time			x	
PARMOD	Generation mode	x	x	x	x
PASS	Password	x	x	x	x
POOLLNK	Pool link name		x		
RECFORM	Record format	x	x		x
RECSIZE	Record length	x	x		x
RETPD	Retention period	x	x	x	x
SAM_NODE_FILE_ENABLE	SAM node file processing enabled				x
SECLEV	Tape only: security level	x			x
SHARUPD	Multi-user mode		x	x	
STREAM	Streaming mode	x			
TAPEWR	Tape only: buffered output	x			x
TPMARK	Tape only: tape marks	x			x
TRANS	Tape only: code translation	x			x
TRTADR	Tape only: user' s own translation table (for reading)	x			x
TRTADW	Tape only: user' s own translation table (for writing)	x			x
UPAM_RAW_ACCESS	UPAM RAW access on SAM node files			x	
VALLEN	Length of value flag		x		
VALPROP	Value flag		x		
VARBLD	Register for the free space in the block to be written				x
WRCHK	Read-after-write check when writing blocks		x	x	x

WROUT	Immediate writing back		x		

Format

Operation	Operands
FCB	[BLIM = number] [,BLKCTRL = PAMKEY / DATA / DATA2K / DATA4K / NO] [,BLKSIZE = STD / (STD,n) / length] [,BTAMRQS = number] [,BUFOFF = L / length] [,CHAINIO = number] [,CHKPT = NO / ANY / BLIM / FEOV,ACTIVE / DUMMY] [,CODE = EBCDIC / ISO7 / OWN] [,DUPEKY = YES] [,EXIT = (relexp) / number] [,FCBTYPE = ISAM / BTAM / PAM / SAM] [,FILE = pathname] [,FSEQ = UNK / NEW / number] [,IOAREA1 = NO / SECRET / number] [,IOAREA2 = NO / SECRET / number] [,IOPERF = VHIGH / HIGH / STD] [,IOREG = reg] [,IOUSAGE = RDWRT / WRITE / READ] [,KEYARG = relexp] [,KEYLEN = length] [,KEYPOS = number] [,LABEL = (STD,3) / STD / (STD,number) / NONSTD] [,LARGE_FILE = *FORBIDDEN / *ALLOWED] [,LBP_REQUIRED = NO / YES] [,LINK = name] [,LOCKENV = *HOST / *XCS] [,LOGLEN = length] [,OPEN = INPUT / EXTEND / INOUT / OUTIN / OUTPUT / REVERSE / SINOUT / UPDATE]

```
[,OPTION = code / (code1,code2)]
[,OVERLAP = YES]
[,PAD = number]
[,PAMREQS = number]
[,PAMTOUT = number]
[,PARMOD = 24 / 31]
[,PASS = password]
[,POOLLNK = name]
[,RETPD = days]
[,SAM_NODE_FILE_ENABLE = NO / YES]
[,SHARUPD = NO / YES / WEAK]
[,RECFORM = V / F / U / (V / F / U [,N / M / A ])]
[,RECSIZE = length / reg]
[,SECLEV = HIGH / LOW / (HIGH / LOW,OPR)]
[,STREAM = NO / YES]
[,TAPEWR = DEVICE-BUFFER / IMMEDIATE]
[,TPMARK = YES / NO]
[,TRANS = YES / NOv
[,TRTADR = relexp]
[,TRTADW = relexp]
[,UPAM_RAW_ACCESS = NO / YES]
[,VALLEN = length]
[,VALPROP = MIN / MAX]
[,VARBLD = reg]
[,WRCHK = NO / YES]
[,WROUT = NO / YES]
```

Operand descriptions

BLIM = number

Only for creating files with standard labels which are to be processed with the SAM access method and extend over several tapes. The following operands must be specified along with BLIM in the FCB macro: FCBTYPE=SAM, OPEN=OUTPUT, LABEL=(STD,n).
“number” specifies the maximum number of data blocks, that are allowed to be written on a single tape, 1 <= number <= 999999.
When the block limit value is reached, a tape swap is initiated (EOV processing).

If requested via the CHKPT operand, a checkpoint is written at the end of the tape before EOV processing is started. If the end of the tape is reached before the number of blocks specified via BLIM has been written, an error message is entered in the FCB.

BLKCTRL

Only for 31-bit processing (PARMOD=31); the operand is ignored in 24-bit processing (PARMOD=24):
Specifies whether a file with the K format (with PAM keys) or NK format (without PAM keys) is to be processed. The same functions, with identical user interfaces, as for the corresponding K files are available for the processing of NK-SAM and NK-PAM files. The access method NK-ISAM provides some functions over and above those of K-ISAM (ISAM pools, secondary keys; for details see the “Introductory Guide to DMS” [1].

When an existing file is opened, no value should be specified for BLKCTRL (i.e. a NULL operand) (for more information on the NULL operand, see "FCB - Define file control block"). The value is then transferred to the FCB from the catalog entry when the OPEN function is executed (for more information on “existing files” see also the note in the section “Sequence of OPEN processing” in the “Introductory Guide to DMS” [1]).
When a new file is created (with OPEN mode-OUTPUT or OUTIN), the BLKCTRL operand should be omitted. Depending on the file structure and the disk format, the following **default setting** will be assumed for BLKCTRL during OPEN processing.

BLKCTRL	File structure and disk format
PAMKEY	for files (PAM, SAM, ISAM) on K disks and tape files, unless FCBTYPE=BTAM.
DATA	for SAM files on NK2 and NK4 disks
DATA2K	for ISAM files on NK2 disks
DATA4K	for ISAM files on NK4 disks
NO	for PAM files on NK2 and NK4 disks and for BTAM files

= PAMKEY
The file has the K format: the block control information is kept in a PAM key outside the data block. Such a file cannot be created on an NK disk (FBA disk without simulation of PAM keys).

= DATA
The file has NK format: the block control information is kept at the beginning of each logical block (see also BLKSIZE operand description; for ISAM files – at the beginning of each 2-Kbyte or 4-Kbyte block). An NK file may be located on K disks, NK2 disks, and – if the appropriate block size is selected – on NK4 disks as well.
When a file is created for the first time (OPEN OUTPUT/OUTIN), an NK2 or NK4 file is created:
for files created with an access method other than ISAM, the format of the file depends on the blocking factor “n” in the BLKSIZE specification for the size of a logical block.

- If the blocking factor n is an odd number, an NK2 file is created.
- If the blocking factor n is an even number, an NK4 file is created.

When an NK-ISAM file (OPEN OUTPUT/OUTIN) is created, the format of the file is selected on the basis of the disk format. A file that has already been opened can be opened without regard to the block format.

= DATA2K
For ISAM files only:
explicitly creates or processes an NK2 file. This value cannot be used to create a file on an NK4 disk or to open a file located on one.
The block-specific management information is stored in the first 16 bytes of each 2-Kbyte block.

= DATA4K
For ISAM files only:
explicitly creates or processes an NK4 file. The block-specific management information is stored in the first 16 bytes of each 4-Kbyte block. If a blocking factor “n” is specified, “n” must be an even number, i.e. the logical block size must be a multiple of 4-Kbyte (BLKSIZE=(STD,n) where “n” is even).
The file can be created and/or opened on K disks, NK2 disks, and NK4 disks.

= NO
This value is only meaningful for PAM files and SAM tape files: it is converted into BLKCTRL=DATA for SAM disk files, and into BLKCTRL=DATA2K or BLKCTRL=DATA4K for ISAM files.
If FCBTYPE=PAM is specified, an NK-PAM file containing no block-specific management information is created.
This file can be created on both K disks as well as NK2 disks, without regard to the selected logical block size (BLKSIZE). If the specified logical block size (BLKSIZE) is a multiple of 4K (with the blocking factor “n” even), the file can also be created on an NK4 disk.

BLKSIZE

Defines the length of the logical block (data block), i.e. the length of the unit of data transfer from and to I/O devices, and thus the length of the program's I/O area.

If there is **no** specification for BLKSIZE, the value from the catalog entry is used for existing files (for a definition, see also the note in the section “Sequence of OPEN processing” in the “Introductory Guide to DMS” [1]). With new files, (STD,2) files are created on NK4 volumes. (STD,1) files are created on other volumes.
If (STD,2) is specified, the user cannot user his own IOAREA's. He should have these created by the system in class 5 memory when the OPEN function is executed.

For processing, see the note under BLKCTRL on "FCB - Define file control block" and “Programming notes” on "FCB - Define file control block".

Disk files/tape files with standard blocks:
A logical block may consist of several PAM pages. The system automatically links the PAM pages belonging to one transfer unit.

For disk files, there are interactions between this operand and the RECSIZE operand, for tape files between this operand and the LABEL operand.

Tape files with nonstandard blocks:
The data block is defined as the number of bytes which are written/read per write or read operation.

= STD
Equivalent to (STD,1); see below.
Data is transferred from and to devices in units of 2048 bytes; the usable length of the transfer unit (for user data) depends on the BLKCTRL specification (or the disk type).

= (STD,n)
“STD” is a standard block with a block size of 2048 bytes; “n” is the blocking factor (1 <= n <= 16)

Each logical block consists of “n” PAM blocks (1 PAM block or PAM page = 2048 bytes), so the maximum length of a logical block is 16 PAM pages or 32768 bytes. For NK files:“n” defines the length of the logical block as a multiple of 2048 bytes: the length of each such block = n * 2048 bytes.For NK4 files: the value of “n” must be even; the length of the logical block is a multiple of 4 Kbytes. In the case of NK-ISAM files, the operand value DATA4K must be specified for the BLKCTRL operand.

For SAM files with SETL processing: up to 255 records may be held in each logical block, since the positioning information is held in only one byte. This restriction is not applicable to a 31-bit FCB.

BLKCTRL	**Block size available for user data (bytes)**
= PAMKEY	n * 2048
= DATA	for ISAM: n * (2048 - 16) - 16 for SAM: (n * 2048) - 16 for PAM: (n * 2048) - 12
= DATA2K / DATA4K (only possible for ISAM)	n * (2048 - 16) - 16
= NO	n * 2048

Note

The following points should also be taken into account when determining whether a record will fit in a block or how many records will fit in a block:

- for NK-ISAM files with duplicate keys, the length of the time stamp;
- for NK-ISAM files with RECFORM=F, the length of the record format field;
- for NK-SAM files, the length of the length field (fill rate information).

= length
Only for tape files:
specifies the maximum block length in bytes and, at the same time, specifies that the file consists of nonstandard blocks, i.e. no PAM keys are kept.

When specifying “length”, the user must consider, on the one hand, the settings of BUFOFF and RECFORM and, on the other hand, the settings of FCBTYPE and CHAINIO.

Operand RECFORM	Effects
RECFORM=F	“length” specifies the block size including the length of any buffer offset (see the operand BUFOFF). All blocks are the same size.
RECFORM=V/U	“length” specifies the maximum block size including the length of any buffer offset (see the BUFOFF operand). The block size, just like the record length, is variable. If RECFORM=V is used together with CODE=EBCDIC or LABEL=(STD,n) where n > 1 <, “length” must be less than 10000 (converted internally into record format D).

Operand FCBTYPE	Permissible values for “length”
SAM, BTAM	1 <= n <= 32768
PAM	-------

Note

If an existing file is to be opened, it is advisable to use the null operand, so the appropriate value is taken from the catalog entry at OPEN time.
If a file is being created for the first time, BLKSIZE=(STD,n) must be specified for NK4 volumes, where 1 <= n <= 16 and n is even; otherwise, the OPEN will be rejected.

BTAMRQS = number

Only for BTAM:
specifies the number of BTAM I/O requests which can be issued directly one after the other (without WAITs) to the system (MAV mode). All accepted requests are simultaneously present in the system for processing. The system ensures that they are processed sequentially, i.e. processing is asynchronous.

1 <= number <= 8.

Default value BTAMRQS = 1

The BTAM access method is described in the “Introductory Guide to DMS” [1].

BUFOFF

Only for SAM tape files without standard blocking or tape files with BLKCTRL=DATA:
defines the buffer offset, i.e. the length of a field which is inserted at the beginning of each data block.

Default value: If the BUFOFF operand is not specified (neither in the TFT nor in the FCB), the following value will be assigned to the file after it is opened (assuming the value from the catalog entry has not been taken):

- for tape files with BLKCTRL=DATA
  - if FCBTYPE=SAM: BUFOFF=16
  - if FCBTYPE=PAM: BUFOFF=12
  - for SAM tape files without standard blocking
    - if RECFORM=V: BUFOFF=4
    - if RECFORM=F/U: BUFOFF=0

= L
The BUFOFF value is taken from the HDR2 label of the file. If there is no HDR2 label, or if the field “buffer offset” in the label contains blanks (X'4040'), the default value is used.

= length
Specifies the length of the buffer offset.

For files with RECFORM=V: 0 <= length <= 4; if BUFOFF=4 applies, this field contains the length of the current block.

CHAINIO = number

Only for BTAM files with chained I/O:
1 <= number <= 16; “number” is the chaining factor and defines the length of the transport/transfer unit for input and output. “number” is a number of physical blocks, which means that the length of the transport unit is n * BLKSIZE.

A value specified for the LEN operand in the BTAM action macro has priority over the result of “BLKSIZE times number”; nevertheless, CHAINIO must still be specified if chaining is to be used.

CHKPT

For tape files only:
Controls whether and when a checkpoint is to be written automatically to the end of the tape or how file processing is to continue after a restart (RESTART-PROGRAM command; see the “Commands” manual [3]).

Default value: CHKPT=(NO,ACTIVE)

= (NO,...)
Checkpoints are not written automatically.

= (BLIM,...)
When the block limit specified by the BLIM operand is reached, a checkpoint is written automatically; the operand BLIM must be specified.

= (FEOV,...)
A checkpoint is written automatically each time the FEOV macro is called.

= (ANY,...)
A checkpoint is written automatically when the BLIM limit is reached or when the FEOV macro is called. Specification of the BLIM operand is mandatory.

= (...,DUMMY)
“pathname” is treated like a DUMMY file during a restart by means of the RESTART-PROGRAM command.

= (...,ACTIVE)
The file “pathname” is processed further in the case of a restart using the RESTART-PROGRAM command.

CODE

For tape processing:
specifies whether code translation tables are to be used during input and output and, if so, which tables.

For CODE=EBCDIC and CODE=ISO7 the German and international character sets are encoded in the same manner.

For CODE=ISO7 and CODE=OWN, the following points should be noted:

- the block size must be specified by BLKSIZE=length, so that no PAM keys are written;
- for output in locate mode with variable-length records (RECFORM=V), the contents of the record length field change.

= EBCDIC
No code conversion is necessary during processing.
= ISO7
The tape file is written in ISO 7-bit code, which means that EBCDIC code is converted to ISO 7-bit code during output and ISO 7-bit code is converted to EBCDIC code during input.
= OWN
Conversion is carried out with code tables provided by the user. The addresses of these tables must be specified in an FCB macro. At the same time, label processing must either be deactivated via the LABEL operand (LABEL=NO) or be carried out in the user program (LABEL=NSTD).

DUPEKY = YES

For ISAM files:
if several records have the same primary key value, they do not overwrite each other, but are written sequentially in the order in which they are created. The operand DUPEKY=YES is significant only if an ISAM file is created sequentially using PUT macros or extended non-sequentially using STORE macros. The INSRT macro cannot be used to write records with duplicate primary keys.

Default value: the file must not contain duplicate primary keys.

In NK-ISAM, an 8-byte time stamp is appended internally to records with duplicate primary keys. This must be taken into account when defining the record length.

The ISAM macros PUT, STORE and INSRT have different effects if there is a duplicate primary key.

Macro	**Duplicate keys not permitted**	**Duplicate keys permitted (DUPEKY=YES)**
PUT	A record with a duplicate key is not written; EXLST exit: DUPEKY	The records are written sequentially to the file
STORE	The “new” record overwrites the record already stored with this ISAM key	The new record is stored after the old record
INSRT	A record with a duplicate key is not written; EXLST exit: DUPEKY	A record with a duplicate key is not written; EXLST exit: DUPEKY

Note

No secondary keys may be defined in a file containing duplicate primary keys.

EXIT

Specifies the address to which the program is to branch in the case of an error. If the operand is not specified, exception conditions during accessing of the file lead to abnormal program termination.
If the exit is taken, a flag is set in the FCB. The user routine can determine the relevant exit condition. If a hardware error occurs during input or output, a 5-byte field in the FCB is filled with information from the CCB (standard device byte, sense bytes 1-3, Executive flag byte; see the NDWERINF macro, "NDWERINF - Evaluate status bytes"). The meanings of these bytes are described in the appendix, "DMS error codes"). The Executive flag byte and the standard device byte are defined in the DSECT for the CCB (IDCCB).

= (relexp)
Address of a user routine in the user program which executes error handling.

= relexp
Address of the EXLST macro via whose exits various user routines are addressed for specific handling of the various error types.

FCBTYPE

Specifies the access method to be used for file processing.

= ISAM
Depending on the value specified for the BLKCTRL operand, it is processed as an NK-ISAM file (BLKCTRL=DATA /DATA2K/DATA4K) or as a K-ISAM file (BLKCTRL=PAMKEY). The ISAM access method is described in the “Introductory Guide to DMS” [1].

ISAM-specific operands: DUPEKY, KEYLEN, KEYPOS, LOGLEN, POOLLNK, VALLEN, WROUT and VALPROP.

= BTAM
A tape file is processed with the access method BTAM (The BTAM access method is described in the “Introductory Guide to DMS” [1].)

BTAM-specific operands: CHAINIO, OPEN=SINOUT, BTAMRQS.

= PAM
The file is processed with the access method UPAM (see description of corresponding access method). PAM files may be stored on tape or disk.

= SAM
The file is processed with the access method SAM and may be located on disk or tape. SAM files are generally processed sequentially with the access method SAM. They may also be processed with UPAM. (The SAM access method is described in the “Introductory Guide to DMS” [1].)

SAM-specific operands: BUFOFF, CLOSMSG, OPEN=UPDATE.

FILE = pathname

Designates the permanent or temporary file or the file generation to be processed with:<c-string 1..54: filename 1..54>

File generations can be addressed via their absolute or relative generation numbers.

Default value: FILE=fcbaddr (= symbolic address of the FCB); if not present: blanks

Pathname means [:catid:][$userid.]filename

catid
Catalog ID; if omitted, the default catalog ID for the user ID is used.

userid
User ID; if omitted, the user ID specified in the LOGON command is used.

filename
Fully qualified file name.

FORM = SHORT

For the 24-bit interface only (non-XS processing):
specifies that no memory space is to be reserved for logical routines.

Default value: space is reserved in the 24-bit FCB for the logical routines.

The logical routines handle the blocking and unblocking of records for SAM and ISAM; this means, for normal SAM /ISAM processing, that the file cannot be opened if FORM=SHORT is specified in the FCB macro. The logical routines are not needed for PAM files and the operand FORM=SHORT is ignored in this case.

For XS processing (PARMOD=31), FORM=SHORT is ignored since the 31-bit TU FCB contains only the addresses of the logical routines.

FSEQ

For tape files which belong to a file set:
specifies the (sequence) number of a file within the file set. For example, if several files with the same name are stored on one tape, access to a specific file is controlled by FSEQ. This also applies to MF/MV sets.

Default value: FSEQ = 0 (tape processing begins with the first file)

= UNK
Permissible only for files with standard labels:
the start position of the file is unknown. The tape is rewound for file processing.

= NEW
Permissible only for files which are not yet cataloged:
a new file is to be added to a file set. The tape is positioned to the end of the file set and the new file is written behind the currently last file of the file set. The new file receives a file sequence number 1 higher than that of the “old” last file.

= number
Specifies the file sequence number of “pathname” within the file set;
0 <= number <= 9999.
FSEQ=0, just like FSEQ=1, denotes the first file of the file set.

If “pathname” is already cataloged, the FSEQ value must be the same as the file sequence number in the catalog entry. If a new file is to be created, it is written at the end of the file set, which means that its file sequence number must be 1 higher than that of the “old” last file in the file set.

The tape is not rewound on opening the file if it is already at the position specified with FSEQ.

IOAREA1

Specifies whether a buffer area is to be allocated at OPEN time, and if so, at what address.

Default value: DMS automatically requests a buffer area (in class 5 memory) at OPEN time – either above or below the 16-Mb boundary, dependingon the addressing and generation mode (see "Operands IOAREA1/2").

= NO
No buffer area is to be allocated at OPEN time (not permitted for SAM and K-ISAM processing).

= SECRET
At OPEN time, DMS requests an area in nonprivileged class 5 memory for IOAREA1;
in the event of a dump analysis, the pages in this area are not output.

= relexp
Address of a buffer area; if this area is less than or equal to one page in size (4096 bytes) it must be fully contained within one page and be aligned on a word boundary; if it is larger than a page, it must be aligned on a page boundary.

IOAREA2

Specifies whether a second buffer area is to be allocated at file opening time.

Default value: DMS automatically requests a buffer area (in class 5 memory) at OPEN time – either above or below the 16-Mb boundary, depending on the addressing and generation mode
(see "Operands IOAREA1/2").

= NO
Only one buffer area is allocated for the file, i.e. overlapped processing is not possible (see also the operand OVERLAP=YES); IOAREA2=NO cannot be specified for K-ISAM processing.

= SECRET
At OPEN time, DMS requests an area in nonprivileged class 5 memory for IOAREA2;
in the event of a dump analysis, the pages in this area are not output.

= relexp
Address of the second buffer area; if it is less than or equal to one page (4096 bytes), it must be contained within one page and aligned on a word boundary; if larger than one page, it must be aligned on a page boundary.

IOPERF

Only applicable to the 31-bit interface:
sets the desired performance attribute for I/O processing with regard to the use of a cache.

= VHIGH
Data should be permanently maintained in the cache.

= HIGH
If possible, the file should be processed via a cache.

= STD
There are no specific performance requirements. The file is not processed via a cache (for more information on cache processing, see the “Introductory Guide to DMS” [1]).

IOREG = reg

For SAM and ISAM only:
specifies that the file is to be processed in locate mode.
“reg” specifies the register containing the address of the current record (2 <= reg <= 12).

Default value: The file is processed in move mode

In locate mode, the user is responsible for correctly addressing records in the buffer; no automatic blocking /deblocking of records is executed.

IOUSAGE

Only applies to the 31-bit interface:
This parameter specifies, how the cache should be used for the file.

= RDWRT
The performance attribute refers to read and write operations.

= WRITE
The performance attribute refers to write operations.

= READ
The performance attribute refers to read operations.

KEYARG = relexp

For ISAM only:
specifies the address of a field containing the ISAM key for the current record. This field is evaluated for the ISAM macros GETKY, GETFL, ELIM and SETL.

KEYLEN = length

For ISAM files:
specifies the length of the ISAM key in bytes,
where 1 <= length <= 255 - VALLEN - LOGLEN

Default value: KEYLEN = 8

KEYPOS = number

For ISAM files :
specifies the position of the ISAM key in the record. In variable-length records, 4 bytes for the record length and control field must be taken into account. The ISAM key may be anywhere in the record, but must be in the same position in each record of one file.

Default value: for files with RECFORM = V: KEYPOS = 5;
for files with RECFORM = F: KEYPOS = 1.

LABEL

Only for tape files :
specifies the label type for files on tape or tape cartridge; the SECLEV operand determines how the labels are processed.

Default value: LABEL = (STD,1)

For existing tape files, the standard identifier in the VOL1 label always applies. The LABEL operand is evaluated for output files (OPEN OUTIN/OUTPUT). If the tape already contains files or file sections, the standard indicator in the VOL1 label is set or updated as specified in the LABEL operand (see also section "Programming notes").

= STD
File and volume either already have or will receive standard labels in accordance with DIN 66029, exchange level 1.

= (STD,number)
File and volume either already have or will receive standard labels in accordance with the exchange level of DIN 66029 designated by “number”; 0 <= number <= 3.

Effects of the LABEL operand

	(STD,0)	**(STD,1)**	**(STD,2)**	**(STD,3)**
DIN 66029 exchange level Date	-	1 8/1972	2 6/1976	3 3/1978
Label standard version in VOL1 label	_ (blank)	1	2	3
CODE=ISO-7/ OWN	not permitted	STD blocks converted to nonstandard blocks	STD blocks converted to nonstandard blocks	STD blocks converted to nonstandard blocks
		RECFORM=V: conversion to D format (not for BTAM)	RECFORM=V: conversion to D format	RECFORM=V: conversion to D format
		RECSIZE > 9999 or BLKSIZE > 9999 OPEN error	RECSIZE > 9999 or BLKSIZE > 9999 OPEN error	RECSIZE > 9999 or BLKSIZE > 9999 OPEN error
CODE=EBCDIC			STD blocks converted to nonstandard blocks	STD blocks converted to nonstandard blocks
Access method			SAM only	SAM only

RECFORM=U				Invalid for output files; converted to (STD,2)

(STD,1) is assumed for:

- RECFORM=V and CODE=EBCDIC
- BLKSIZE=STD
- FCBTYPE=PAM or FCBTYPE=BTAM

For (STD,0), CODE=EBCDIC must be used.

If the value in the label standard version in the VOL1 label is less than (STD,number), the value from the label standard version is assumed for “number”.

= NO
Labels are neither read nor written (no file label processing). If the tape has standard labels, the system processes the volume labels and checks the access authorization.

= NSTD The tape file already has or is to receive nonstandard labels and file label processing is performed in the user program. If the volume has standard labels, the system processes them and checks the access authorization.

LARGE_FILE

Only for disk files (access methods ISAM, SAM and UPAM):
The LARGE_FILE operand determines whether or not the file size may grow beyond 32 GB (see "Files larger than 32 GB").

= *FORBIDDEN
The file size may not exceed 32 GB.

= *ALLOWED
Only for 31-bit interfaces and files with BLKCTRL PAMKEY:
The file size may exceed 32 GB.

LBP_REQUIRED

specifies, if a last-byte pointer (LBP) processing is requested. The LBP then specifies the exact logic file end of a PAM file in byte limitation. For non-PAM files, LBP_REQUIRED is ignored.

=*NO
No last-byte pointer processing is requested.

=*YES
It is requested that the PAM file on public space has a last-byte pointer (LBP). This request must be carried out for a new file at the time of OPEN OUTIN. Then, a valid last-byte pointer is generated for this file, in which the LBP_valid bit and the LBP value for UPAM processing to OPEN OUTIN or INOUT in CLOSE is set when the last page of the file has been written.

For files on private disk or tape, for encrypted files and for OPEN with SHARUP=YES, a last-byte pointer processing for PAM files is rejected with DMS0D09.

LINK = name

Via the file link name (“name”) specified here, DMS establishes a link to the TFT and thus to the file and processing attributes defined by means of the FILE macro.

“name” may be up to eight characters in length. If the file link name is to be addressed via the command interface, it must correspond to the data type <structured_name 1..8> (see the “Commands” manual [3]).

LOCKENV

For UPAM access method only:
specifies which lock log the user uses for synchronization.

= *HOST
Shared-update processing is only permitted on the user's own host. The synchronization takes place via the Task Lock Manager.

= *XCS
Shared-update processing is permitted within an XCS network; the synchronization must take place via the Distributed Lock Manager.

LOGLEN = length

For ISAM files:
specifies the length (in bytes) of the logical flag in the ISAM index; the maximum length is determined by the length of the ISAM key and the length of any existing value flag (see the VALLEN operand, "FCB - Define file control block"), since the entire ISAM index must not be longer than 255 bytes. The rule is thus:

length <= 255 - KEYLEN - VALLEN

Default value: LOGLEN=0, i.e. there is no logical flag in the ISAM index.

In the ISAM index, the ISAM key may be followed by a logical flag in which selection criteria are defined bit-by-bit and encoded in binary code. In K-ISAM files, all logical flags of a block are evaluated and the result is placed in the next-higher index entry. NK-ISAM supports logical flags only compatibly, and does not place the flags in the index entry.

OPEN

Specifies the OPEN mode for the file. This setting may be overwritten by the OPEN mode specified in the OPEN macro.

Default value: OPEN = INPUT

The following table shows which OPEN modes are permissible for the various access methods.

OPEN modes with the FCB macro

OPEN mode	ISAM	BTAM	SAM	UPAM
INPUT	x	x	x	x
EXTEND	x	-	x	-

INOUT	x	x	-	x
OUTIN	x	x	-	x
OUTPUT	x	x	x	-
REVERSE	-	x	x	-
SINOUT	-	x	-	-
UPDATE	-	-	x	-

where:

x OPEN mode is permitted

\- OPEN mode is not permitted

The various OPEN modes are described in detail in the descriptions of the access methods.

= INPUT
An existing file is read (for a definition of an existing file, see also the note in the section “Sequence of OPEN processing” in the “Introductory Guide to DMS” [1]).

= EXTEND
A file is extended, i.e. further data blocks are added to the end of the file or the file is overwritten from a certain position onwards; only sequential write operations are permitted.

= INOUT
An existing file is opened for non-sequential processing; write and read operations are permitted.

= OUTIN
A file is created or, if it already exists, overwritten from the beginning. Both read and write operations are permitted (non-sequential).

= OUTPUT
A file is created or, if it already exists, overwritten from the beginning.

= REVERSE
An existing file is opened as an input file for sequential reading from end-of-file -> beginning-of-file.The file section number of the file section to be processed can be specified via the VSEQ operand in the FILE macro. Tape files are positioned to the end of the file section on completion of OPEN processing. If no VSEQ is specified, the last file section is processed. Automatic tape switching is not supported.

= SINOUT
Only for BTAM tape files:
the file must exist and the tape must not be positioned to the beginning of the tape. Data blocks can be read or written; labels are not processed.
Files that extend over multiple reels cannot be processed with SINOUT.

= UPDATE
Only for SAM disk files:
the file is to be processed in locate mode.

OPTION = code

A list of options can be specified for this operand. By default, no codes are stored. Either GLODEF or NOWAIT may be specified for “code”.

= GLODEF
If the file name is specified without an explicit user ID, and the file is not found under the caller's user ID, a second read attempt is made under the system default user ID (see “Access via the system default user ID” in the “Introductory Guide to DMS” [1]). This applies only if no TFT entry for the file exists at OPEN time, since the TFT already contains the path name.

= NOWAIT
If an I/O operation encounters an error caused by a device (such as device INOP), the program does not wait for an operator reaction, but branches immediately to the EXLST exit ERRADDR. The code NOWAIT is accepted only together with PARMOD=31 and for the access methods PAM and ISAM.

OVERLAP = YES

Only for ISAM files:
if this is specified and a second I/O area is defined in the program (IOAREA2 in the FCB), read operations (GET /GETR) can be executed in overlapped mode.

Default value: OVERLAP = NO

For NK-ISAM, “overlapped processing” means that neighboring blocks are also read into the ISAM pool. OVERLAP=YES should be used only when reading is primarily sequential.

PAD = number

For ISAM files:created sequentially (using the ISAM macro PUT); the “padding factor” PAD specifies how much free space is to be left in each data block for subsequent extension of the file (specified as a percentage of the block size defined by BLKSIZE). PAD thus has an effect on the block splitting rate when a file is extended non-sequentially.

Default value: PAD = 15

The PAD specification has different effects for NK-ISAM and K-ISAM. For NK-ISAM, the block is filled at least up to the PAD limit; for K-ISAM, it is never filled above the PAD limit.

PAMREQS = number

For UPAM processing:
specifies how many asynchronous I/O operations can be requested simultaneously (in one PAM macro; see the PAM macro, operand REQNO, "PAM - Perform UPAM actions"; 0 <= number <= 100).

Default value: PAMREQS = 1

PAMTOUT = number

For UPAM processing:
specifies how long a job is to wait for requested locks (in seconds).

0 <= number <= 43200

Default value: PAMTOUT = 0

If the locks are still not available after the time specified here, control is passed to EXLST exit DLOCK or PGLOCK. PAMTOUT=0 means that control is returned immediately, regardless of whether or not the requested locks are available.

PARMOD

Specifies the generation mode for the macro.

Default value: the value defined with the GPARMOD macro or preset by the Assembler.

The generation mode can be set globally for all macros in a program by means of the GPARMOD macro.

The PARMOD operand in the DMS macros overrides the default value set by the GPARMOD macro or (if GPARMOD is not specified) preset in the Assembler.

All PARMOD specifications for one file must have the same value.

= 24
The macro is expanded in accordance with the format for the 24-bit interface. The object code is thus executable only in 24-bit addressing mode.

= 31
The macro is generated as addressing mode-independent.

PASS = password

If the file is password-protected, the password necessary for the desired access must either be stored in the password table of the job or specified by the PASS operand in the FCB macro.

The maximum length of the password is four bytes. The password field ID1PASS is padded on the left with zeros or, if the specified password is longer, truncated on the left (in accordance with the rules for the processing of address constants in Assembler programs).

The rules for specifying passwords and the password hierarchy are described in various parts of the manual, for example in the description of the CATAL macro.

POOLLNK = name

Only for ISAM files:
processed in user ISAM pools (NK-ISAM); “name” is the “pool link name” (up to 8 characters long) which is placed in the TFT. This pool link name must be assigned to an ISAM pool (see the macros ADDPLNK, "ADDPLNK - Define pool link name", and CREPOOL, "CREPOOL - Create ISAM pool").

RECFORM

Specifies the record format of the file “pathname” and also specifies which control characters are to be interpreted if the file is sent to a printer.

Default value: RECFORM = (V,N)

The record format specification is evaluated only for the access methods SAM and ISAM. UPAM processes files only on a block basis and any RECFORM specification is ignored. BTAM is also a block-oriented access method, but accepts a RECFORM specification.

The record formats are described in detail under the section dealing with access methods in the “Introductory Guide to DMS” [1]. For information on the relationship between the RECFORM and RECSIZE specifications, see the RECSIZE operand. For details of the evaluation of print control characters, see the PRINT-DOCUMENT command (CONTROL-MODE and LINE-SPACING operands) in the manuals “Commands” [3] and “SPOOL” [4].

For tape files with RECFORM=V and CODE=EBCDIC or LABEL=(STD,n) – where n > 1 – the contents of the record length and block size fields are converted internally into the D-format: the value for the record/block size is represented as a decimal number. For such files, the block size must not exceed 10000 bytes. During input, format-D records are converted back to hexadecimal form before being transferred to the user's area.

= V
“pathname” consists of variable-length records, which means that the user must remember, when programming, that each record is preceded by a 4-byte field whose first two bytes contain the record length in binary form. Bytes 3 and 4 of this field are used by the system. For input files, the record length field is set by the system; for output files, this must be done by the user. The value specified for RECSIZE is the maximum permissible record length. For BTAM files, the specification RECFORM=V is treated like RECFORM=U.

= F
“pathname” consists of fixed-length records, i.e. the user does not need to worry about the record length and control fields. All records in the file have the same length, which is defined via the RECSIZE operand. (The decisive factor here is the BLKSIZE value, not the RECSIZE value.)

= U
“pathname” consists of records with “undefined” length. Each data block contains only one record, whose length is passed in a register. The system sets this register for input and the user must set it for output (see the RECSIZE operand). RECFORM=U converts the specification LABEL=(STD,3) into (STD,2).

RECFORM=U is not permitted for ISAM files.

= (...,N)
“pathname” is not a print file and therefore contains no printer control characters. It should not be printed with control character evaluation.

= (...,M)
The first data byte in each record is interpreted as a control character in EBCDIC code. The file can be printed out by specifying the PRINT-DOCUMENT command with the operand LINE-SPACING=*BY-EBCDIC-CONTROL. For ISAM files, the ISAM index is taken into account.

= (...,A)
The first data byte in each record is interpreted as an ASA control character. The file can be printed out by specifying the PRINT-DOCUMENT command with the operand LINE-SPACING=*BY-ASA-CONTROL.

RECSIZE

Specifies the record length, depending on the specification in the RECFORM operand.

Default value: for RECFORM = V: RECSIZE = BLKSIZE;
for RECFORM = F: no default value.

= length
the maximum record length in bytes

For RECFORM=V:
For NK-ISAM files, it should be noted that overflow blocks may result if the maximum record length is fully utilized. For ISAM files, the maximum record length is BLKSIZE. For SAM files, the maximum record length is BLKSIZE-4, but with BLKCTRL=DATA only BLKSIZE-16.
For tape files, the interaction with the operands CODE and LABEL should be noted, see table "Effects of the LABEL operand".

For RECFORM=F (all records in the file are the same length):
For ISAM files, the maximum record length is BLKSIZE-4.
For SAM files, the maximum record length is BLKSIZE, but with BLKCTRL=DATA only BLKSIZE-16.

If a GET is used (to read) in move mode for files with RECFORM=V and RECSIZE=length in the FCB macro, and if the record to be read is longer than the specified RECSIZE, the following applies:

- ISAM files: the operation is aborted with error message DMS0AAD.
- SAM files: the entire record is read into the program area, regardless of the RECSIZE specification.

If a PUT is used (to write) in move mode for files with RECFORM=V and RECSIZE=length in the FCB macro, and if the record to be written is longer than the specified RECSIZE, the following applies:

- ISAM files: the entire record is written to the file, regardless of the RECSIZE specification.
- SAM files: the entire record is written to the file, regardless of the RECSIZE specification.

= reg
For RECFORM=U: the RECSIZE operand must specify a general register (2 <= reg <= 12) which contains the current record length for input and output. The system sets this register for input and the user must set it for output.

RETPD = days

With “RETPD”, the user can define a retention period during which no write access (update, delete) is possible.

Default value: RETPD=0, i.e. the file can be updated or erased at any time.

“days” is an integer (not greater than 32767) which specifies the length of the retention period in days.
Once the retention period has elapsed, the file is not automatically erased; this simply means that write access is permitted again.

The retention period can also be controlled by means of the MODIFY-FILE-ATTRIBUTES command or the CATAL macro (see "CATAL - Process catalog entry"): any RETPD specification in CATAL is immediately placed in the catalog entry. For tape files, the CATAL macro can be used only before the file is opened for the first time.

SAM_NODE_FILE_ENABLE

specifies if the processing of SAM node files is allowed or not.

=*NO
The processing of SAM node files is rejected with the return code DMS0D1A.

=*YES
The processing of SAM node files is requested by the user. As the processing of SAM node files is different from the processing of SAM files on public volumes, the SAM_NODE_FILE_ENABLE parameter prevents older applications from accidentally accessing the SAM node files.

When processing SAM node files, the retrieval addresses are currently calculated the same way as for SAM files on public volumes. The retrieval addresses that are given back to the application are only valid between OPEN and CLOSE.

These retrieval addresses are no longer valid after a CLOSE with subsequent OPEN.

In future versions of BS2000, the user will be able to use the PROCESSING_MODE parameter during the OPEN call to define, whether the retrieval addresses are to be calculated the same as now, whether only retrieval addresses still usable after CLOSE and OPEN will be given back or whether no retrieval addresses should be used at all and the files are only to be processed sequentially.

SECLEV

Only for tape files:
the operand SECLEV (security level) refers to the TPIGNORE entry in the JOIN file (cf. the SHOW-USER-ATTRIBUTES command). A SECLEV specification is ignored in interactive mode. In batch mode, users with the appropriate authorization can use the SECLEV operand to specify whether error messages are to be suppressed and/or whether additional label checking is to be executed.

= HIGH
In batch mode, error messages are sent to the console. If the job is running under a user ID with TPIGNORE=YES in its JOIN file, the operator can ignore the error messages.

= LOW
Permissible only for the tape/file owner if TPIGNORE=YES is defined in the JOIN file of the user ID: certain error messages are suppressed in batch mode.

= (...,OPR)
The entry OPR (= Overwrite PRotection) causes the system to execute additional label checking:

- if a file is written on a tape behind an existing file, the labels of the preceding file are checked;
- the expiration date of the new file must not be greater than that of the preceding file.

SHARUPD

Only for ISAM or UPAM disk files:
specifies whether several jobs may concurrently open the file with an OPEN mode other than OPEN INPUT.

= NO
As soon as the file is opened by a job with OPEN INPUT, it is locked for all other jobs. Concurrent access to the file by several jobs is possible only if the file is used as an input file by all of these jobs, i.e. it is opened with OPEN INPUT. If the file has been opened with OPEN INPUT, any attempt to open it with another OPEN mode is rejected.

= YES
Only for ISAM and PAM files:
the file can be processed concurrently by several jobs, but SHARUPD=YES must be specified in all these jobs (if writing is allowed). In the case of UPAM, the user can protect data blocks from access by other jobs as long as he is processing them. In the case of ISAM, these locks – whenever necessary – are set automatically by the system. With NK-ISAM, files which are opened for shared-update processing must be processed in host-specific ISAM pools. SHARUPD=YES for ISAM files simultaneously activates the WROUT function (see the WROUT operand, below).

= WEAK
For UPAM processing only:
ensures write protection but not read protection, i.e. only one job can open the file for updating, but other jobs may use it simultaneously as an input file. The user must take into account in his program that the contents of the file may change while he is using it as an input file.

The following table shows the various contending levels, with the type of protection offered in each case:

SHARUPD options	User actions	Protection type READ	Protection type WRITE
YES	n users are reading and m users are writing	*	*
WEAK	n users are reading and 1 user is writing	-	*
NO	n users are reading or a user is writing	*	*

The WEAK operand is supported for PAM files only.
For information on permitted SHARUPD combinations see the section “UPAM” in the “Introductory Guide to DMS” [1].
If FCBTYPEPAM, SHARUPD=WEAK is processed as if it were SHARUPD=NO.

STREAM

Only for BTAM tape files:
The STREAM operand enables users to specify whether they wish to use “streaming mode”. This implies that the user is working with chained data blocks as well as MAV mode (for which appropriate specifications must be made by the user!) and that the individual jobs are to be internally concatenated. It also implies that if a streamer is being used, this mode is to be set in terms of hardware.

= NO
Streaming mode is not set.

= YES
Streaming mode is set.

TAPEWR

Only for files on tape cartridges:
the user can specify whether or not input and output are to be buffered.

= DEVICE-BUFFER
Input and output are buffered in the tape controller, resulting in a high data transfer rate.

= IMMEDIATE
Input and output are not buffered.

TPMARK

Only for tape files without standard labels:
specifies whether tape marks are to be written. The TPMARK operand is evaluated during OPEN only for tape files with LABEL=NO/NSTD. Tape files with LABEL=(STD,n) automatically receive tape marks after the labels.

= NO
No tape mark is written.

= YES
Tape files with NSTD labels: the tape mark follows the label.

Tape files without labels: the tape mark is written at the beginning of the tape.

TRANS

Only for tape files:
used as input files and not created with CODE=EBCDIC; specifies how the code of the file is to be converted during reading.

= YES
ISO 7-bit code or OWN code is converted into EBCDIC code.

= NO
ISO 7-bit code is converted into 8-bit format by inserting a leading zero.

TRTADR = relexp

Specifies the address of the user's own translation table for reading a tape file. This may be specified only together with CODE=OWN or if no value is specified for CODE.

TRTADW = relexp

Specifies the address of the user's own translation table for writing a tape file. This may be specified only together with CODE=OWN or if no value is specified for CODE.

UPAM_RAW_ACCESS

specifies, how an application accesses SAM node files using UPAM. If no SAM node file is accessed, the specification is ignored.

=*NO
At UPAM OPEN for a SAM node file, logical SAM blocks are accessed. They are created from UNIX data during the read process by using the SAM converter or they are converted to a UNIX byte flow during the write process and written to the UNIX file.

For OPEN OUTIN or INOUT with UPAM_RAW_ACCESS=*NO, SAM_NODE_FILE_ENABLE=*YES has to be specified in order to gain write access to the file. Otherwise, the processing is rejected with DMS0D1A. For OPEN INPUT, the specification of SAM_NODE_FILE_ENABLE=*YES is unnecessary.

=*YES
In case of a UPAM OPEN on a SAM node file, the SAM converter gets disabled, so that the application receives unstructured data or writes into the UNIX file. The value of the SAM_NODE-FILE_ENABLE parameter is not relevant.

VALLEN = length

Only for ISAM files:
specifies the length of the value flag in the ISAM index.

Default value: VALLEN = 0, i.e. the ISAM index does not contain a value flag.

length <= 255 - KEYLEN - LOGLEN

Value flags are treated differently by NK-ISAM (BLKCTRL=DATA) and K-ISAM (BLKCTRL=PAMKEY). In K-ISAM, they are evaluated block-by-block and transferred to the next higher index entry as specified in the VALPROP operand. In NK-ISAM, no flags are evaluated for the index entry.

VALPROP

Only for K-ISAM files:
in conjunction with BLKCTRL=PAMKEY, i.e. with K-ISAM files (NK-ISAM will ignore a VALPROP specification); specifies how the value flag is to be included in the index entries (VALPROP = VALue PROPagation).

= MIN
The lowest value of the value flag within one data or index block is included in the index entry at the next higher level.

= MAX
The highest value of the value flag in a data or index block is included.

VARBLD=reg

Only for SAM files with RECFORM=V, which are processed in locate mode (see also the IOREG operand); specifies the register (2 <= reg <= 12) in which DMS is to show the free space (in bytes) in the block to be written.

WRCHK

Only for the processing of disk files:
specifies whether a read-after-write check is to be executed. “WRCHK” is not placed in the catalog entry and must therefore be repeated each time before the file is opened or processed.
A read-after-write check is designed to detect recording errors (-> error recovery measures). If the error cannot be rectified, control is passed to the EXLST exit ERRADR. Due to the additional disk revolutions involved, the read-after-write function has a decidedly negative effect on system performance.

= NO
No read-after-write check is executed.

= YES
A read-after-write check is executed.

WROUT

For ISAM processing:
WROUT controls how often updated blocks are written back to disk. For shared-update processing or in cross-task ISAM pools, WROUT=YES is set implicitly: updated blocks are written back to disk immediately.

Default value:

- for “normal” file processing: WROUT = NO
- for shared-update processing: WROUT = YES
- in cross-task ISAM pools: WROUT = YES
- in task-local ISAM pools for which WROUT=YES is valid: WROUT = YES

= NO
An updated block is written back to disk only when the contents of the related buffer area need to be overwritten or, at the latest, when the file is closed.
= YES
Each updated block is written back to disk immediately, thus always ensuring the consistency of the data on the disk and in virtual memory. However, this also increases the I/O rate.

Return information (example)

The return code 0D33, which was kept in the FCB field ID1ECB by the OPEN has the following meaning, depending on the specifications made by the caller in the file name and in the OPTION parameter.

File name	OPTION	Meaning of 0D33
:cat:$user.file		The file is not present under the $user label on the cat pubset.
$user.file		The file is not present under the $user label on the default pubset of $user.
$.file		The file is not present on the pubset under the label specified with the DEFLUID system parameter.
:cat:$.file		The file is not present on the cat pubset under the label specified with the DEFLUID system parameter.
file	not GLODEF	The file is not present under the caller label on its default pubset.
file	GLODEF	The file is neither present under the caller label on its default pubset nor under the label specified with the DEFLUID system parameter on the pubset specified there.
:cat:file	not GLODEF	The file is not present under the caller label on the cat pubset.
:cat:file	GLODEF	The file is neither present on the cat pubset under the caller label nor the cat pubset under the label specified with the DEFLUID system parameter.

Any modification to the file name in field ID1FILE within an OPEN-EXIT routine is ignored.

Programming notes

FCB structure

The structure of a 31-bit FCB differs considerably from that of a 24-bit FCB:

- The FCB macro expands in 31-bit mode into a CSECT statement if no other DMS action macro has been called previously in 31-bit mode.
- There is no longer any FCB extension, i.e. all data is accommodated within the FCB itself.
- The logical routines of the SAM and ISAM access methods are no longer included in the FCB; the FCB now contains only the addresses of these routines.
- All 3-byte addresses have been eliminated and corresponding new 4-byte addresses have been introduced.
- The FCB has a fixed, uniform size.

FCB modification

When a file is opened, DMS checks the FCB entries and uses them to set up a privileged file control block (TPR FCB); any subsequent changes to the FCB values are ignored. The file control block can be modified only by closing the file (CLOSE) and then reopening it (OPEN).

If an operand is not specified in the FCB macro, the default value is assumed. If a null string operand is specified, it is assumed that the value of the operand is supplied by a FILE macro or by the catalog entry of the file (exception: LINK= operand).

Operands for disk files

FCB operand	Null operand: operand value supplied via the catalog	BTAM	SAM	ISAM	PAM	Operand in the FILE macro
BLIM		i	x	i	i	x
BLKCTRL	x	x	x	x	x	x
BLKSIZE	x	x	x	x	x	x
BTAMRQS		x	i	i	i	
BUFOFF	x	i	x	i	i	x
CHAINIO		x	i	i	i	x
CHKPT		i	x	i	i	x
CODE	x	x	x	i	i	x
DUPEKY		i	i	x	i	x
EXIT		x	x	x	x	
FCBTYPE	x	x	x	x	x	x
FILE		x	x	x	x	x
FORM		i	x	x	i	
FSEQ	x	x	x	i	x	x
IOAREA1		x	x	x	x	
IOAREA2		x	x	x	x	
IOPERF		i	x	x	x	x
IOREG		x	x	x	i	
IOUSAGE		i	x	x	x	x
KEYARG		i	i	x	i	
KEYLEN	x	i	i	x	i	x
KEYPOS	x	i	i	x	i	x

LABEL		x	x	i	x	x
LBP_ REQUIRED		i	i	i	x	
LINK		x	x	x	x	x
LARGE_FILE		i	x	x	x	x
LOCKENV		i	x	x	x	x
LOGLEN	x	i	i	x	i	x
OPEN		x	x	x	x	x
OPTION		x	x	x	x	x
OVERLAP		i	i	x	i	x
PAD		i	i	x	i	x
PAMREQS		i	i	i	x	
PAMTOUT		i	i	i	x	
PARMOD		x	x	x	x	
PASS		x	x	x	x	
POOLLNK		i	i	x	i	x
RECFORM	x	x	x	x	i	x
RECSIZE	x	x	x	x	i	x
RETPD		x	x	x	x	x
SAM_NODE_ FILE_ENABLE		i	x	i	i	
SECLEV		x	x	i	x	
SHARUPD		i	x	x	x	x
STREAM		x	x	i	i	
TAPEWR		x	x	i	x	x
TPMARK		x	x	i	x	x
TRANS		x	x	i	i	x
TRTADR		x	x	i	i	
TRTADW		x	x	i	i	

UPAM_RAW_ACCESS		i	i	i	x	
VALLEN	x	i	i	x	i	x
VALPROP	x	i	i	x		x
VARBLD	x	i	x	x	i	
WROUT		i	i	x	i	x
WRCHK		i	x	x	x	x

where:

i Operands are ignored..

x Operands may be specified.

Operand BLKSIZE

The user can/must specify “BLKSIZE=(STD,n)” if

- the record is longer than 2048 bytes, or
- the record length is uneconomic in conjunction with a block length of 2048 bytes. For example, if the user has fixed-length records (RECFORM=F), each 1500 bytes long, and defines the block length as BLKSIZE=STD, then 548 bytes would be wasted in each block. If, instead, the user were to specify BLKSIZE=(STD,3), then 6000 out of the 6144 bytes (3 x 2048) would be used, and only 144 bytes would be wasted in every three blocks. Using very large block sizes does, however, lead to an increase in the paging rate.

Operands IOAREA1/2

When a file is opened, the IOAREA addresses are created (if necessary) and validated. They are then moved into system memory. Consequently, any changes relating to these addresses are completely ignored in the FCB. For new addresses to become effective, a CLOSE macro has to be issued, followed by a new OPEN macro.

This method of processing was chosen to minimize internal system processing time (overhead), as the IOAREA addresses do not have to be checked before every action macro. PAM and BTAM allow buffer addresses to be defined in their action macros. These addresses are of course checked whenever action macros are issued.

If a SAM file is opened in UPDATE mode, the IOAREA2 buffer is not used.

If IOAREA1=NO is specified, then the value of IOAREA2 must also be NO. If IOAREA1 is defined with “relexp”, then the value of IOAREA2 must also be “relexp”, or NO. If IOAREA1 is not specified, then IOAREA2 is not allowed either, or must be defined as NO.

Operand LABEL

In the case of a file opened in INPUT, INOUT, EXTEND or REVERSE mode, the system ignores the specification LABEL=(STD,n) and refers to the exchange level (label standard version) specified in the volume label (VOL1).

In the case of an output file, the exchange level specified here applies. If only STD was specified, exchange level 3 applies.

If the allocated tape volume already contains one or more files or file sections, the exchange level must match the standard identifier in the first volume label.

Otherwise, the standard identifier is assigned a value as shown below:

Exchange level	0	1	2	3
Standard identifier	blank	1	2	3

Restrictions

- Exchange level 1: If CODE=ISO7/OWN is specified, STD block specifications are converted to NSTD block specifications and format V records (RECFORM=V) are converted to format D records. If this is not possible, e.g. when RECSIZE/BLKSIZE is greater than 9999, an OPEN error results.
- Exchange level 2: Only SAM files can be processed. STD block specifications are converted as for exchange level 1: in this case, tapes with CODE=EBCDIC are also affected.
- Exchange level 3: RECFORM=U is not allowed for output files.

Rules

- If FSEQ=0/1, then LABEL=(STD,3) is converted to (STD,1).
- If RECFORM=V and CODE=EBCDIC, then LABEL=(STD,1) is implicit.
- If BLKSIZE=STD, then LABEL=(STD,1) is implicit.
- If BLKSIZE=length and RECFORM=U, then LABEL=(STD,2) is implicit.
- If a tape volume already contains one or more files/file sections and the standard identifier in the first volume label is less than the implicitly assumed DIN exchange level, then the version number is taken from the volume label.

Operand WROUT

The WROUT function provides increased security for ISAM file processing, since in the event of a system crash only the records processed by the last macro will be corrupted or lost. (Exception: when the PUT macro is used, only blocks are written.)

The increased number of I/O operations reduces throughput.

The WROUT function takes effect after the ISAM action macros STORE, ELIM, INSRT and PUTX.

ISAM shared-update mode already includes the WROUT function, so in this case the WROUT operand is irrelevant.

A value other than YES or NO will always result in an error message.

The value specified in the FILE macro has priority over the value specified in the FCB macro. The latter value takes effect only if the WROUT operand is omitted from the FILE macro or is entered as a null operand (i.e. WROUT=,...).

FCBAD - Create FCB addresses

Macro type: type O

For all following DMS macros in the program, the code is generated in such a way that the FCBs can be located outside the basic register in symbolic addressing. The addresses of the FCB are then stored in the literal area. The FDBAD macro was created to facilitate the conversion BS2000.

Format

Operation	Operands
FCBAD	

FEOV - Close tape

The FEOV macro initiates a tape swap and file processing is continued on the continuation tape. This macro is ignored for tape files opened with OPEN REVERSE.

If, in the case of input files, the end of the file is on the tape, DMS recognizes “end of file” and activates the EOFADDR routine (see the EXLST macro, "EXLST - Define exit address list"). If the end of the file is not on the tape and no continuation tape is assigned, DMS activates the NODEV routine (see the EXLST macro, "EXLST - Define exit address list").

Format

Operation	Operands
FEOV	**fcbadr / (1)** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB of the file to be processed.

(1)
Register 1 contains the FCB address.

PARMOD

Specifies the generation mode for the macro.

Default value: the value preset for the generation mode by means of the GPARMOD macro or by the assembler.

= 24
The macro is triggered with the expansion for the 24-bit interface. The object can only be executed in 24-bit addressing mode.

= 31
The macro is generated independently of the addressing mode.

Programming note

The FEOV macro destoys registers 0, 1, 14 and 15.

Example of tape swap for a SAM file

The tape swap is triggered with the FEOV macro.

```
FEOVTEST START
         LDBASE 3
         USING  *,3
         .
         .
         FILE   TAPE.TEST,LINK=AUS,DEVICE=T9P,VOLUME=(C1776A,C2921A)
         .
         .
         OPEN  TAPE,OUTPUT          OPEN TAPE FILE 'TAPE'
         .
         .
         PUT   TAPE,RECOUT          WRITE RECORD
         .
         .
         FEOV  TAPE                 INITIATE TAPE SWAP
         .
         .
         CLOSE TAPE                 CLOSE TAPE FILE 'TAPE'
*
END     TERM
*
TAPEND   LR    8,0                  LABEL ROUTINE
         MVC   0(L'BEGIN,8),BEGIN   *
         LBRET TAPE,1               *
*
TAPE     FCB   FCBTYPE=SAM,BLKSIZE=(STD,3),LINK=AUS,RECFORM=F,        -
               RECSIZE=22,EXIT=TAPEXIT
*
TAPEXIT  EXLST COMMON=ENDE,LABEOV=TAPEND
*
RECOUT   DS    CL22
*
BEGIN    DC    CL80'UTL1 USER LABEL'
         .
         .
         END
```

FILE - Define file attributes / control file processing

Macro type: type S (E form/L form/D form/C form); see "Macro types"

The FILE macro processes permanent and temporary files (but not EAM files) and file generations. It can be used to create new files and catalog entries, to change file attributes, and to import files from private volumes.

Except for the retention period (RETPD), the FILE macro cannot be used to define or change file attributes such as the passwords or the access type. If a catalog entry is created by means of FILE, the system default values are used for these attributes. They can, if necessary, then be modified by means of a CATAL macro.

Via the task file table (TFT), the FILE macro establishes a connection between the program and the file and between the file attributes defined in the FILE macro or the catalog entry and the FCB macro.

Main functions of the FILE macro

- creating catalog entries for new files and file generations
- requesting devices and volumes
- allocating and releasing storage space
- creating TFT entries with details of the file processing (data structure, OPEN mode, etc.)
- defining the data organization on tapes.

This introduction is followed by an overview of the functions of the FILE macro at the operand level. The various subjects (such as TFT, TST, etc.) are described in detail in the introduction to this manual, i.e. in the opening chapters.

Catalog entry

If the file or file generation specified in the FILE macro is not yet cataloged, a catalog entry is created. If the file (generation) is already cataloged, DMS accesses the catalog entry when the file is opened and updates it, if necessary, when the file is closed again. The values entered for the operands IOPERF, IOUSAGE, DEVICE, VOLUME, SPACE, DDEVICE, DVOLUME, DSPACE, STATE=FOREIGN (for tape files) and FSEQ (in part) are evaluated and transferred to the catalog entry; otherwise, the corresponding system defaults are set. Entries for the remaining operands in a FILE macro are only evaluated in combination with a file link name and are transferred to the TFT entry.

If the catalog entry contains a basic access control list (BASIC-ACL), DELDATE or GUARDS, it is not possible to create a tape file using FILE (a FILE macro with the corresponding DEVICE operand will be rejected).

If a file (generation) which is to be cataloged is stored on private disk, DMS takes the values for the catalog entry from the F1 label of the first volume containing the file.

The following is specified for cataloging a new file:

for BACKUP:	E for temporary file and working file; otherwise: according to BACKUP system parameter
for MIGRATE:	FORBIDDEN for specification of a disk belonging to an SM pubset in the VOLUME operand; otherwise: INHIBITED for temporary files, ALLOWED for permanent files

for NUM-OF-BACKUP-VERS	0 for temporary file, for file on private disk or on tape and for each file generation; otherwise: according to NUMBACK system parameter
for CCS:	Coded character set from the user catalog of the file owner. If this character set equals `EDF03IRV`, no character set is entered.
for NETCCS:	At node file creation, the Net-Storage coded character set is entered according to its definition in the user entry. The resulting NETCCS of the file is identified based on the following table:

CCS entry [1]	**NETCCS entry [1]**	**Resulting NETCCS in the catalog entry of the node file**
EDF03IRV/*NONE	*ISO	ISO88591; during code conversion, EDF041 is assumed for CCS
EDF03DRV	*ISO	ISO88591; during code conversion, EDF04DRV is assumed for CCS
EDF04DRV	*ISO	ISO88591
EDF04x	*ISO	ISO8859x with x=1,2,..F
ISO8859x	*ISO or *NO-CONV	ISOx
UTFx	*ISO or *NO-CONV	UTFx
<name_a 1..8>	<name_b 1..8>	<name_b 1..8>
<name_a 1..8>	*NO-CONV	<name_a 1..8>

[1] User entry (SYSSRPM) or CATALOG or CREATE-FILE or MODIFY-FILE-ATTRIBUTES entry

If a file has a catalog entry but as yet no disk storage space and is to be assigned space on a private disk through the FILE call, the file may not be encrypted.

If a file has a catalog entry but as yet no tape type and is to be assigned a tape type through

the FILE call, the file may be encrypted only if a file generation is involved. This is then decrypted.

When a new file generation is cataloged, the encryption attributes are transferred from the group entry to the new catalog entry. This does not apply for file generations on tape.

File link name / task file table (TFT)

If a file link name is specified with the LINK operand in the FILE macro, the system creates an entry in the job-specific TFT and transfers to this entry values specified in the current FILE macro, including any NULL operands (see below). When the file is opened, the values from the TFT are placed in the file control block (FCB).

At the OPEN, the specifications for a file are thus contained in:

- the TFT entry
- the file control block (FCB) of the program
- the catalog entry of the file

The catalog entry is subsequently updated with the values contained in the file control block (see OPEN and CLOSE processing, in the “Introductory Guide to DMS” [1]).

Pool link name / ISAM pools

With NK-ISAM, ISAM files are processed in ISAM pools. The connection between the user ISAM pool and the file is established via the pool link name, which is specified by means of the POOLLNK operand. If no pool link name is specified, the file is processed in one of the standard system ISAM pools.

Access methods

Depending on the access method, data structures such as the record length, block size, etc. can be defined by means of the FILE macro.

The operand descriptions point out special features and interactions between the various operands.

NULL operands

Some operands of the FILE macro may be specified as “NULL operands” together with a file link name (null string = an empty character string as the operand value).

When the file is subsequently opened, the appropriate information for these file attributes is obtained from the catalog entry and transferred to the file control block (FCB).

```
FILE ...,LINK=name,FCBTYPE=,RECFORM=,...
```

The following operands may be specified as NULL operands in combination with a file link name:

BLKCTRL, BLKSIZE, BUFOFF, CODE, FCBTYPE, FSEQ, IOPERF, IOUSAGE, KEYLEN, KEYPOS, LOGLEN, RECFORM, RECSIZE, VALLEN and VALPROP

The term “NULL operand” applies only to the above-mentioned operands of the FILE macro. Operand values can also be omitted for other macros; however, but this normally causes the default setting or the default value for the operands to be used.

Version of the FILE macro

The VERSION operand determines which macro format is generated. If VERSION is omitted or if VERSION=0 is specified, the operand list and SVC for the old format (BS2000 V9.0) are generated. The new operands introduced with BS2000 V9.5 and the device types valid as of BS2000 V9.5 are supported as of VERSION=1. New features introduced with BS2000 V10.0 and BS2000/OSD-BC V1.0, especially the new layout of the operand list in which the variable parts are extracted and kept in separate lists (see “Programming notes”, "FILE - Define file attributes / control file processing"), can be used as of VERSION=2.

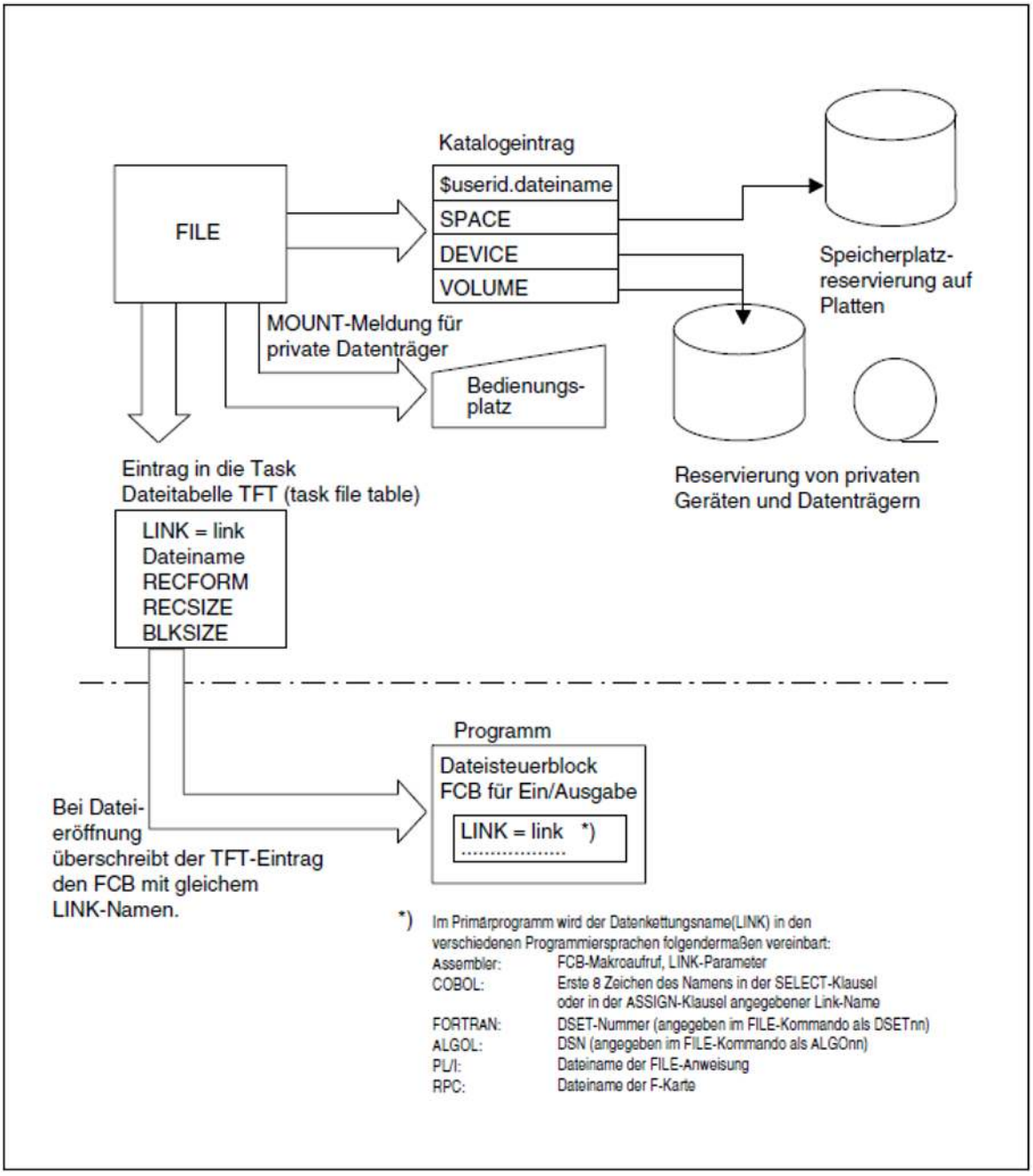


Figure 7: Functions of the FILE macro

Function overview

Operand	Operand value	Function / meaning
Naming and cataloging files, defining link names		

pathname		• Create a catalog entry • Allocate storage space (primary allocation) • Name file/catalog entry to which subsequent operations will refer

DATATTR		Specifies the reference file from which, during creation of a TFT entry, certain values are to be taken which are not explicitly specified with the respective operands
*DUMMY		Define a dummy file
LINK	name	Define a file link name for which a TFT entry is created
POOLLNK	name	For NK-ISAM files: define a pool link name for the user ISAM pool
STATE	FOREIGN	Import a file from private volumes or from a Net-Storage volume
File attributes		
AVAIL	HIGH	Define the requirements regarding availability
BLKCTRL	PAMKEY/ DATA/ DATA2K/ DATA4K/ NO	Define the data format
BLKSIZE	(STD,n) length	Block size as a multiple of a PAM page Tape files: block size for nonstandard blocks
CODE	EBCDIC/ ISO7/ ISO7D/ OWN	Tape files: code
EXC32GB	ALLOWED/ FORBIDDEN	Permit file size > 32 GB for disk files
FCBTYPE	ISAM/ PAM/ SAM/ BTAM	Access method for the file
KEYLEN	number	ISAM files: length of the ISAM key
KEYPOS	number	ISAM files: position of the ISAM key
LOGLEN	number	ISAM files: length of the logical flag
NFTYPE	BS2000/ NODE-FILE	File type for file on Net-Storage: BS2000 file or node file
RECFORM	V/F/U N/M/A	Record format: variable/fixed/undefined Specifies whether printer control characters are to be taken into account
RECSIZE	length r	Record length for RECFORM=F/V Register which contains the length of the current record for RECFORM=U

RETPD	days	Retention period for the file
STOCLAS		Assign a storage class to the file during its creation on an SM pubset
VALLEN	number	ISAM files: length of the value flag
WORKFIL		Specify whether the file is to be created on a volume set for work files or on a volume set with permanent data storage

Requesting devices and volumes		
DDEVICE	device	ISAM files: device type for data section (if separate from index section)
DEVICE	device/ WORK	Define device type/request work tape
DVOLUME	(vsn,...)	ISAM files: private disk for data section (if separate from index section)
FSEQ	UNK/NEW/number	Tape files: position within a file set
MOUNT	(number,...)	Mount request for private volumes
TSET	(name,vsn)	Tape files: define a tape set for extending files or file sets
TVSN	(vsn,...)	Tape files: temporary volume list for current processing
VOLSET		Specify the volume set of an SM pubset on which the file is to be created
VOLUME	(PRIVATE,n) (vsn,...)	Request private volumes Define volume list
VSEQ	(L=(number,...))	Tape files: specify desired file section
Space management for disk		
DSPACE	primary (primary, secondary) (page, number, ABS)	ISAM files: space management for the data section if the index and data sections are separate (see SPACE, "FILE - Define file attributes / control file processing")
SPACE	primary (primary,secondary) (primary,secondary, *KEEP) (page,number,ABS)	Disk files: allocate or release storage space Absolute allocation
OPEN mode and processing attributes		
BLIM	number	Tape files: maximum number of logical blocks per tape
BUFOFF	L/length	Tape files: length of buffer offset
BYPASS	LP/(LP,n)/ (LP,+n)/(LP,-n)	Tape files: bypass label checking

CHAINIO	number	Tape files: chaining factor
CHKPT	NO/ ANY/ BLIM/ FEOV	Tape files: automatic checkpointing
CLOSE	RWD/ INVAL/ REPOS/ DISCON/ LEAVE/ KEEP-DATA-IN-CACHE	Tape/disk files: close mode for the file

CLOSMSG	NO/ YES	SAM files: output of a message after completion of CLOSE processing
DESTOC	NO/ YES	Tape files: overwrite remaining data
DISKWR	BY-CLOSE/ IMMEDIATE	Define the point after a write operation by which the data of the file must be in a consistent state
DUPEKY	YES/ NO	ISAM files: duplicate keys permitted
IOPERF	HIGH/ STD/ USER-MAX/ VERY-HIGH	Define the performance attribute of the file
IOUSAGE	RDWRT/ READ/ WRITE	Specify the I/O operations to which the performance attribute (IOPERF) of the file refers
LABEL	(STD,number) NO NSTD	Tape files: file with standard labels (as per DIN 66029) Tape files without file labels Tape files with nonstandard labels
LOCKENV	HOST/ XCS	Specify whether the file can be opened for writing from different systems simultaneously
OPEN	INPUT/ OUTPUT/ EXTEND/ INOUT/ OUTIN/ UPDATE/ SINOUT/ REVERSE	Specify the OPEN mode for the file
OVERLAP	YES/NO	ISAM files: overlapped processing
PAD	number	ISAM files: padding in data blocks during sequential creation of the file
POOLSIZ	number	ISAM files: size of the file-specific ISAM pool
SECLEV	HIGH/ LOW	Tape files: security level
SHARUPD	YES/ NO/ WEAK	ISAM files: shared-update processing • permitted/not permitted PAM files: shared-update processing • permitted/not permitted/guaranteed protection against write access
STREAM	NO/ YES	BTAM tape files: enable I/Os in streaming mode

TAPEWR	DEVICE-BUFFER/ IMMEDIATE	Files on tape cartridges: buffered or unbuffered output
TPMARK	YES/ NO	Tape files: write tape marks
TRANS	YES/ NO	Convert non-EBCDIC tape files
VALPROP	MIN/ MAX	K-ISAM files: control interpretation of value flags

WRCHK	NO/ YES	Disk files: read-after-write check
WROUT	NO/ YES	ISAM files: write updated blocks immediately
Control of macro generation		
MF	D / C / L / E	Operand lists
PREFIX		Prefix for names in operand lists
VERSION	0 / 1 / 2 / 3	Version setting for the macro

Format

Operation	Operands
FILE	VERSION = 0 / <integer 1..3> ,MF = C / D / L / S / (E,<name>) / (E,(<reg 0..15>)) ,PREFIX = I / * / <name 1..1> ,<pathname 1..54> / *DUMMY ,AVAIL = HIGH ,BLIM = <integer 1..999999> ,BLKCTRL = *BY-PROG / < > / NO / PAMKEY / DATA / DATA4K / DATA2K ,BLKSIZE = *BY-PROG / < > / STD / <integer 1..32767> / (STD,<integer 1..16>) ,BUFOFF = *BY-PROG / < > / L / <integer 0..99> ,BYPASS = LP / (LP,<integer -127..32767>) ,CHAINIO = <integer 1..100> ,CHKPT = (NO / BLIM / FEOV / ANY , ACTIVE / DUMMY) ,CLOSE = RWD / REPOS / DISCON / LEAVE / INVAL / KEEP-DATA-IN-CACHE ,CLOSMSG = NO / YES ,CODE = *BY-PROG / < > / EBCDIC / ISO7 / ISO7D / OWN ,DATATTR = (*FROM-FILE,<c-string: filename 1..54>) ,DDEVICE = <name 1..8> ,DESTOC = NO / YES ,DEVICE = <name 1..8> ,DISKWR = IMMEDIATE / BY-CLOSE ,DSPACE = <integer 0..2147483647> / (<integer 0..2147483647> [,<integer 0..32767>]) / (<integer 0..2147483647>,<integer 0..2147483647>,ABS) ,DUPEKY = NO / YES ,DVOLUME = <@adr> / PRIVATE / (PRIVATE,<integer 1..9>) / list-poss(255): <name 1..6> ,EXC32GB = FORBIDDEN / ALLOWED ,FCBTYPE = *BY-PROG / < > / SAM / ISAM / BTAM / PAM ,FSEQ = < > / UNK / NEW / <integer 0..9999>

```
,IOPERF = < > / STD / HIGH / VERY-HIGH / USER-MAX

,IOUSAGE = < > / READ / WRITE / RDWRT

,KEYLEN = *BY-PROG / < > / <integer 1..255>

,KEYPOS = *BY-PROG / < > / <integer 1..32767>

,LABEL = *BY-PROG / STD / NO / NSTD / ( STD,<integer 0..3> )

,LINK = <name 1..8>

,LOCKENV = HOST / XCS

,LOGLEN = *BY-PROG / < > / <integer 0..255>

,MOUNT = 0 / <@addr> / list-poss(255): <integer 1..255>

,NFTYPE = BS2000 / NODE-FILE

,OPEN = INPUT / OUTPUT / OUTIN / INOUT / SINOUT / EXTEND /
        REVERSE

,OVERLAP = NO / YES

,PAD = <integer 0..99>

,POOLLNK = <name 1..8>

,POOLSIZ = <integer 128..1048576>

,RECFORM = *BY-PROG / < > / F / V / U / ( F / V / U , N / M / A )

,RECSIZE = *BY-PROG / < > / <integer 0..32768> / <reg 2..12>

,RETPD = <integer 0..32767>

,SECLEV = HIGH / LOW / ( HIGH / LOW , OPR )

,SHARUPD = NO / YES / WEAK

,SPACE = <integer -2147483647..2147483647> /
         (<integer -2147483647..2147483647>,<integer 0..32767>) /
         ( <integer -2147483647..2147483647>,*KEEP) /
         ( <integer 02147483647..2147483647>
           ,<integer 0..32767>,*KEEP ) /
         ( <integer 1..2147483647>,<integer 1..2147483647>,ABS )

,STATE = FOREIGN

,STOCLAS = *NONE / <c-string: name 1..8>

,STREAM = NO / YES

,TAPEWR = DEVICE-BUFFER / IMMEDIATE

,TPMARK = NO / YES

,TRANS = YES / NO

,TSET = <name 1..4> / ( <name1..4>,<name 1..6> )
```

```
,TVSN = <@addr> / list-poss(255) <name 1..6>

,VALLEN = *BY-PROG / < > / <integer 0..255>

,VALPROP = *BY-PROG / < > / MIN / MAX

,VOLSET = <c-string: catid 1..4> / *CONTROL

,VOLUME = <@addr> / REMOVE-UNUSED / PRIVATE /
          ( PRIVATE [,<integer 1..9>] ) /
          list-poss(255): <name 1..6>

,VSEQ = <@addr> / <integer 1..255> /
        (L=list-poss(255): <integer 1..255>)

,WORKFIL = NO / YES

,WRCHK = NO / YES

,WROUT = NO / YES
```

Operand default values

The following applies to FILE operands for which no default setting is explicitly described and for which a corresponding FCB operand exists. If the operand is not specified in the FILE macro, this is noted in the TFT entry. The applicable operand value when processing the file is then taken from the specifications in the FCB macro.
If the operand is omitted in the FCB macro as well, the default setting for the FCB macro applies, assuming it is not overwritten by the corresponding value from the catalog entry by OPEN.

Operand descriptions

The forms of the MF operand are described in detail in the appendix, "Macro types". In all macros differentiated by the MF operand (MF=L/E/D/C) the version operand must have the same value.

pathname

Designates the path name of the file(s) or file generation(s)
with: <c-string 1..54: filename 1..54>

“pathname” may not be a file generation group.

If “pathname” is not yet cataloged, a catalog entry is created and space is allocated to the file in accordance with the primary allocation (see the SPACE operand, "FILE - Define file attributes / control file processing").
If a temporary file is specified, it must have been created by the calling task.

The following elements CANNOT be created on a Net-Storage volume:

- Files with a PAM key
- File generations
- Work files
- Temporary files

“pathname” means [:catid:][$userid.]filename

catid
Catalog ID;
Default value: the catalog ID assigned to the user ID.
If the catalog ID belongs to a remote system to which an RFA connection exists, the FILE call is sent to the remote system with the parameter list via RFA.

userid
User ID;
Default value: the user ID specified in the SET-LOGON-PARAMETERS or LOGON command.
If the file has not yet been cataloged as shareable or if storage space has been released, a foreign user ID may only be specified if the calling task possesses the TSOS privilege or is a co-owner of the file.

filename
Fully qualified name of a file or file generation. Tape file names may be suffixed with a version number in parentheses.

***DUMMY**
The FILE command describes a dummy file. If LINK is also specified, a TFT entry with a volume list is created. If TSET is specified with *DUMMY, a TST entry is also created. All other operands are simply checked for syntax errors, but otherwise ignored; neither devices nor storage space are allocated, and no catalog entry is created.

Dummy file as an input file: when the program attempts to read from the file, EOF processing is initiated.
Dummy file as an output file: data are transferred to the I/O areas of program, but output to a physical volume is suppressed.

AVAIL = HIGH

Only as of VERSION=3 and only relevant for files on pubsets and on Net-Storage volumes:
The file is to have an increased availability and is stored on a corresponding volume set (e.g. DRV).

The specification is rejected in the following cases:

- the file already occupies storage space
- SPACE is specified with a non-positive primary allocation
- WORKFIL=YES is specified
- a temporary file is specified
- the file ends up on an SM pubset which does not offer increased availability
- the file ends up on an SM pubset which does not contain a volume set with increased availability
- the file ends up on a private disk
- a tape file is specified

BLIM = number

For the creation of tape files with standard labels which are to be processed with the SAM access method:
“number” specifies how many data blocks may be written on one tape;
1 <= number <= 999999.

When this value is reached, a tape swap is initiated (EOV processing). If requested with the CHKPT operand, a checkpoint is written at the end of the tape before EOV processing is started. If the end of the tape is reached before the specified number of blocks has been written, the user receives an error message in the FCB.

The following entries are rejected if BLIM is specified: FCBTYPE=PAM/BTAM/ISAM, LABEL=NO/STD, FSEQ=n with n>1 and FSEQ=UNK/NEW.

BLKCTRL

Only as of VERSION=1:
Determines whether a file with the conventional K format (with PAM keys) or with the new NK format (without PAM keys) is to be processed. BLKCTRL is relevant for the preliminary file format.

For the processing of NK-SAM and NK-PAM files, the same functions as for the corresponding K files are available, with identical user interfaces. For NK-ISAM files, the access method NK-ISAM provides functions over and above those of K-ISAM, e.g. the processing of ISAM files in ISAM pools or the use of secondary keys (see the "Introductory Guide to DMS" [1]). NK-ISAM and K-ISAM processing differ internally, but there are only minor changes at the user interface with regard to the effects of ISAM-specific operands (see the following table).

Operand	**BLKCTRL = PAMKEY**	**BLKCTRL = DATA/DATA2K/DATA4K**
DDEVICE DVOLUME DSPACE	Separate index and data sections for ISAM files on private disks	Separate index and data sections are not supported, but operands can be specified
DUPEKY		Records with duplicate keys are given an internal time stamp
LOGLEN	Length of the logical flag	This operand is ignored
POOLLNK		Connects the file to a user ISAM pool (otherwise to a standard ISAM pool)
OVERLAP	Read operations are overlapped (with IOAREA2)	Any adjacent blocks are likewise read into the ISAM pool
PAD	Minimum space to be left free in each data block	Maximum space to be left free in each data block
SHARUPD	Block locks	Record or range locks
VALLEN	Length of the value flag	This operand is ignored
VALPROP	The value flag is evaluated	The value flag is ignored

If the BLKCTRL operand is not specified (neither in the TFT nor in the FCB), the following BLKCTRL value is assigned to the file when it is opened (unless the value from the catalog entry is used):

BLKCTRL = PAMKEY	for files on conventional (CKD) disks
BLKCTRL = DATA	for SAM or ISAM files on the new (FBA) disks (without PAM key simulation)
BLKCTRL = NO	for PAM files on the new (FBA) disks (without PAM key simulation)

= *BY-PROG
Only as of Version=3 and only relevant if the DATATTR operand is specified:
The BLKCTRL value from the reference file catalog entry is ignored.

= PAMKEY
The file has the K format: the block control information is kept in a PAM key outside the data block. Such a K file cannot be created on an NK disk (FBA disk without PAM key simulation) or on a Net-Storage volume.

= NO
This value is only meaningful for PAM files and SAM tape files: it is converted into BLKCTRL=DATA for SAM disk files, and into BLKCTRL=DATA2K or BLKCTRL=DATA4K for ISAM files.
If FCBTYPE=PAM is specified, an NK-PAM file containing no block-specific management information is created.
This file can be created on both K disks as well as NK2 disks, without regard to the selected logical block size (BLKSIZE). If the specified logical block size (BLKSIZE) is a multiple of 4K (with the blocking factor “n” even), the file can also be created on an NK4 disk.

= DATA
The file has NK format: the block control information is kept at the beginning of each logical block (for ISAM files: at the beginning of each 2-Kbyte or 4-Kbyte block).
An NK file may be located on K disks, NK2 disks, and – if the appropriate block size is selected – on NK4 disks as well. When a file is created for the first time (OPEN OUTPUT/OUTIN), an NK2 or NK4 file is created:

For files created with an access method other than ISAM, the format of the file depends on the blocking factor “n” in the BLKSIZE specification for the size of a logical block.

- If the blocking factor n is an odd number, an NK2 file is created.
- If the blocking factor n is an even number, an NK4 file is created.

When an NK-ISAM file (OPEN OUTPUT/OUTIN) is created, the format of the file is selected on the basis of the disk format. A file that has already been opened can be opened without regard to the block format.

= DATA2K
Only as of VERSION=2, for ISAM files:
Specialization of “DATA” for NK-ISAM files.

Explicitly creates (OPEN OUTPUT/OUTIN) an NK2-ISAM file. When existing files are opened, every file is checked to determine whether an NK2-ISAM file is involved.
The block-specific management information is stored in the first 16 bytes of each data block. This value cannot be used to create a file on an NK4 disk or to open a file located on one.

= DATA4K
Only as of VERSION=2, for ISAM files:
Specialization of “DATA” for NK-ISAM files.

Explicitly creates (OPEN OUTPUT/OUTIN) an NK4-ISAM file. When existing files are opened, every file is checked to determine whether an NK4-ISAM file is involved.
The block-specific management information is stored in the first 16 bytes of each 4-Kbyte block. If a blocking factor “n” is specified, “n” must be an even number, i.e. the logical block size must be a multiple of 4-Kbytes.
The file can be created and/or opened on K disks, NK2 disks, and NK4 disks.

BLKSIZE

Defines the length of the logical block (data block), i.e. the length of the unit of data transfer from and to I/O devices. BLKSIZE is relevant for the preliminary file format.

If the BLKSIZE is not specified (neither in the TFT nor FCB), the following BLKSIZE value will be assigned to the file when it is opened (unless the value from the catalog entry is taken):

- K/NK2 volumes: BLKSIZE = STD
- NK4 volumes: BLKSIZE = (STD,2)

For disk files, there are interactions between this operand and the SPACE and RECSIZE operands, for tape files between this operand and the LABEL operand (see the two tables under BLKSIZE=length on "FILE - Define file attributes / control file processing").

K disk files/tape files with standard blocks: logical blocks may consist of several PAM pages. The system automatically links the PAM pages belonging to one transfer unit.

Tape files with nonstandard blocks: the block format is not the same as that used by PPAM; a logical block is defined as the number of bytes which are read or written in one read or write operation.

= *BY-PROG
*Only as of Version=3 and only relevant if the DATATTR operand is specified:*The BLKSIZE value from the reference file catalog entry is ignored.

= STD
Equivalent to (STD,1); see below.

The data is transferred in units of 2048 bytes from/to the devices. The usable data length for the user varies depending on what is specified for BLKCTRL (and/or on the disk type).

= (STD,n)
“STD” is a standard block with a block size of 2048 bytes; “n” is the blocking factor (1 <= n <= 16)

Each logical block consists of “n” PAM blocks (1 PAM block or PAM page = 2048 bytes), so the maximum length of a logical block is 16 PAM pages or 32768 bytes.For NK files “n” defines the length of the logical block as a multiple of 2048 bytes: the length of each such block = n * 2048 bytes.

For NK4 files: the value of “n” must be even; the length of the logical block is a multiple of 4K. In the case of NK-ISAM files, the operand value DATA4K must be specified for the BLKCTRL operand.

For SAM files with SETL processing: up to 255 records may be held in each logical block, since the positioning information is held in only one byte. This restriction is not applicable to a 31-bit FCB= length.

= length
Only for tape files:
specifies the maximum block length in bytes and, at the same time, specifies that the file consists of nonstandard blocks; no PAM keys are written.

When specifying “length”, the user must consider, on the one hand, the settings of BUFOFF and RECFORM and, on the other hand, the settings of FCBTYPE and CHAINIO.

Operand RECFORM	Effects
RECFORM=F	“length” specifies the block size including the length of any buffer offset (see the BUFOFF operand). All blocks are the same size.

RECFORM=V/U	“length” specifies the maximum block size including the length of any buffer offset (see the BUFOFF operand). The block size, just like the record length, is variable. If RECFORM=V is used together with CODE=EBCDIC or LABEL=(STD,n) (n > 1), “length” must < 10000

Operand FCBTYPE	Permissible values for “length”
SAM / BTAM	1 <= n <= 32768
PAM	-------

BUFOFF

For tape files with BLKCTRL=DATA or SAM tape files without standard blocking:
defines the buffer offset, i.e. the length of a field which is inserted at the beginning of each data block.

If the BUFOFF is not specified (neither in the TFT nor FCB), the following BUFOFF value will be assigned to the file when it is opened (unless the value from the catalog entry is taken):

- for tape files with BLKCTRL=DATA:
  - for FCBTYPE=SAM: BUFOFF=16
  - for FCBTYPE=PAM: BUFOFF=12
- for SAM tape files without standard blocking
  - for RECFORM=V: BUFOFF=4
  - for RECFORM=F/U: BUFOFF=0

= *BY-PROG
As of Version=3 and only relevant if the DATATTR operand is specified:
The BUFOFF value from the reference file catalog entry is ignored.
= L
The BUFOFF value is taken from the HDR2 label of the file. If there is no HDR2 label, or if the field “buffer offset” in the label contains blanks (X'4040'), the same values apply as when the BUFOFF specification is omitted (neither TFT not FCB).

= length
Specifies the length of the buffer offset.

For SAM files with RECFORM=V: 0 <= length <= 4; if BUFOFF=4 applies, this field contains the length of the current block. For files with BLKCTRL=DATA, this field contains the block control field.

BYPASS

For input files on tape:
Given the appropriate authorization for the user in the user catalog, the user can bypass the label checking routines and specify how the tape is to be positioned. DMS checks that the correct tape is mounted and activates any user routines for label handling in the normal manner. The positioning specification is evaluated only if no OPEN exit is defined.

In addition to label checking, code checking is also bypassed. If the user specifies CODE=OWN, he/she must provide the appropriate code tables.

BYPASS permits processing of tapes created under other operating systems (such as BS1000) or of tapes whose structure and label formats are not known to the system. The BYPASS specification is valid only during file processing; it is not included in the catalog entry for the file.

If specified together with BYPASS, the FSEQ and SECLEV operands are not evaluated.

= LP
No label handling takes place. The header labels are neither checked nor read. The tape position is not changed.

= (LP,n)
No label handling takes place. When the file is opened, the tape is positioned to the n^{th} tape mark, counting from the beginning of the tape. 0 <= n <= 32767.

(LP,0): position to the beginning of the tape.

= (LP,+n)
No label handling takes place. When the file is opened, the tape is positioned forwards by n tape marks from its current position. 0 <= n <= 127.

(LP,+0): the tape is not repositioned.

= (LP,-n)
No label handling takes place. When the file is opened, the tape is positioned backwards by n tape marks from its current position. 0 <= n <= 127.

(LP,-0): the tape is not repositioned.

CHAINIO = number

For BTAM files with chained I/O:
1 <= number <= 16;
“number” is the chaining factor which defines the length of the transport/transfer unit for input and output. “number” is a number of blocks, which means that the length of the transport unit is “number” * BLKSIZE.

Although the value from CHAINIO=number, multiplied by the block size, can be overwritten by specifications in the program (BTAM macro) when processing BTAM files, CHAINIO must still be specified in the FILE macro if chained I /O is to be used.

CHKPT

For tape files only:
controls if and when a checkpoint is to be written automatically to the end of the tape, or how file processing is to continue after a restart (RESTART command).

Default value: CHKPT=(NO,ACTIVE)

= (NO,...)

No automatic checkpoints are written, unless specified otherwise in the FCB of the program.

= (BLIM,...)
When the block limit specified via the BLIM operand is reached, a checkpoint is written automatically; the operand BLIM must be specified.

= (FEOV,...)
A checkpoint is written automatically each time the FEOV macro is called.

= (ANY,...)
A checkpoint is automatically written when the BLIM limit is reached or when the FEOV macro is called. The operand BLIM must be specified.

= (...,DUMMY)
“pathname” is treated like a DUMMY file during a restart by means of the RESTART-PROGRAM command.

= (...,ACTIVE)
The file “pathname” is processed further during a restart by means of the RESTART-PROGRAM command.

CLOSE

Only as of VERSION=2:
Specifies the CLOSE mode for the file. This value may be overwritten by the CLOSE macro when the file is being closed.

Default setting: The CLOSE value is taken from the CLOSE macro.

For more information on CLOSE processing, see also the “Introductory Guide to DMS” [1].

= RWD
For tape processing:
The tape is rewound to the start.

= REPOS
For tape processing:
The tape is positioned to the beginning of the current file section, depending on the LABEL specification.

= DISCON
For tape processing:
The tape is rewound to the start and unloaded/released.

= LEAVE
For tape processing:
Positions the tape to the logical end-of-file, depending on the LABEL specification.

= INVAL
The cached file blocks are invalidated, i.e. declared invalid, and not written back to disk (to be borne in mind for shared-update processing)

= KEEP-DATA-IN-CACHE
Only as of VERSION=3:
Blocks from the file that are in the cache are not written back to the disk, but remain flagged as valid.

CLOSMSG

Only as of VERSION=1:
For files to be processed sequentially (SAM), the user can specify that a message is to be issued (to SYSOUT) after completion of CLOSE processing. If the CLOSMSG operand is not specified (neither the TFT nor the FCB), the following CLOSMSG value is assigned to the file when it is opened:

Disk: CLOSMSG = NO
Tape: CLOSMSG = YES

= NO
The completion message is suppressed.

= YES
The completion message is issued.

CODE

For tape processing:
Specifies, for SAM or BTAM files, whether code translation tables are to be used during input and output and, if so, which tables.

If the CODE operand is not specified (neither the TFT nor the FCB), the following CODE value is assigned to the file when it is opened:

CODE = EBCDIC

For CODE=EBCDIC and CODE=ISO7 the German and international character sets are encoded in the same manner.

For CODE=ISO7/OWN and FCBTYPE=SAM, the following should be noted:

- the block size must be specified by BLKSIZE=length, so that no PAM keys are written;
- for outputs in locate mode with variable-length records (RECFORM=V), the contents of the record length field change.

= *BY-PROG
Only as of VERSION=3 and only relevant if the DATATTR operand is specified:
The CODE value from the reference file catalog entry is ignored.

= EBCDIC
No code conversion during processing is necessary.

= ISO7
The tape file is written in ISO 7-bit code, which means that EBCDIC code is converted to ISO 7-bit code during output and ISO 7-bit code is converted to EBCDIC code during input. The international ISO table is used.

= ISO7D
Only as of VERSION=3:
The tape file is written in ISO 7-bit code, which means that EBCDIC code is converted to ISO 7-bit code during output and ISO 7-bit code is converted to EBCDIC code during input. The German ISO table is used.

= OWN
Conversion is carried out with code tables provided by the user. The addresses of these tables must be specified in an FCB macro (see the TRTADR and TRTADW operands, "FCB - Define file control block"). At the same time, label processing must be switched off via the LABEL operand (LABEL=NO) or carried out in the user program (LABEL=NSTD).

DATATTR = (*FROM-FILE,<c-string: filename 1..54>)

Only as of VERSION=3:
The following values are transferred to the TFT entry when this is created, from the catalog entry of the reference file specified here. Explicitly specified values have precedence.

BLKCTRL, BLKSIZE, BUFOFF, CODE, FCBTYPE, KEYLEN, KEYPOS, LABEL, LOGLEN, RECFORM, RECSIZE, VALLEN, VALPROP

The reference file must be cataloged in the pubset as the file to which the FILE call refers. The caller must have the right to read the reference file catalog entry (using FSTAT or SHOW-FILE-ATTRIBUTES).

The values for BLKCTRL and BLKSIZE contained in the reference file catalog entry are taken into account when forming the preliminary file format.

If the value *BY-PROG is specified for the above operand, transfer of the corresponding value from the reference file catalog entry is suppressed and no value is transferred to the TFT entry.

Example

A reference file is specified whose catalog entry for BLKCTRL contains the valuePAMKEY. The following then applies:

BLKCTRL value specified in FILE call	BLKCTRL value in TFT entry
none	PAMKEY
*BY-PROG	none
DATA	DATA

DDEVICE = <name 1..8>

For ISAM files with index and data sections separated:
DDEVICE designates the disk type for the data section (that for the index section is specified via DEVICE); permissible entries for “device” can be found in the device table in in „System installation“ manual [16]). The new device types introduced with BS2000 V9.5 are only supported as of VERSION=1. DDEVICE must be specified if no storage space has yet been reserved for the file. DVOLUME and DSPACE must also be specified if DDEVICE is specified.

If at least one volume serial number is specified with DVOLUME, every specification of a disk device type which is known to the system is handled like the STDDISK specification.

NK-ISAM does not support index/data separation, but DDEVICE may still be specified (compatibility with K-ISAM).

DESTOC

For tape processing as of VERSION=1:
the user can specify whether any data on the remainder of the tape is to be deleted by overwriting after completion of EOF/EOV processing.

DESTOC is effective only if a TFT entry is created for “pathname” using the LINK operand.

If the DESTOC operand is not specified, the DESTROY specification is taken from the catalog entry when the file is opened.

DESTOC has the same function as the operand DESTROY in the CATAL macro, but the DESTOC specification overrides the DESTROY value in the catalog entry. The value specified for DESTOC is not placed in the catalog entry.

= NO
Any data on the remainder of the tape is not erased.

= YES
After the EOF/EOV labels have been written, any data on the remainder of the tape is erased.

DEVICE

defines the disk device type or tape type.
When VOLUME specifies a private disk which is not contained in the MAREN catalog, the DEVICE operand must be specified.

Defaults

The following applies if the file has no storage space assigned before the FILE call and is allocated storage space by the FILE call:

- If DEVICE=STDDISK is specified and neither VOLUME nor NFTYPE is not, the file will be created on public disks (and also not on a Net-Storage volume in future expansions).
- If DEVICE=NETSTOR (the volume type for Net-Storage volumes) is specified and VOLUME is not, the file will be created on a Net-Storage volume.

If neither DEVICE nor VOLUME is specified, the following applies:

- If NFTYPE is not specified, the file is created on public disk.
- When NFTYPE is specified, the file of the specified file type is created on an arbitrary Net-Storage volume provided one exists.

Future extensions can enable a different behavior here.

= <name 1..8>
Specifies the device type (for disks) or the volume type for Net-Storage volumes and tapes). The possible entries for disk devices are shown in the “System installation” manual [16] (Device type column); permissible values for tape devices are included in the volume type table (see the “Commands” manual [3]).

DEVICE=NETSTOR (the volume type for Net-Storage volumes) specifies a Net-Storage volume.

DEVICE=TAPE cannot be used to request magnetic tape cartridges.

If a tape type is specified for DEVICE when creating a file, but no specification is made for VOLUME, a free tape (SCRATCH-TAPE) with standard labels is requested during OPEN processing and assigned by the operator. A tape is considered free from the viewpoint of DMS if it has not been written as yet or if the retention period of the first file on it has expired, and write access is enabled.

If at least one volume serial number is specified with VOLUME, every specification of a disk device type which is known to the system is handled like the STDDISK specification.

= WORK
Only for tape processing:
causes a work tape with standard labels to be requested during OPEN processing.Work tapes are not assigned to any owner; the corresponding field in the VOL1 label always contains blanks (X'40'). Work tapes should be requested only when they are required during processing and are not to be archived. File protection is not possible on work tapes. Work tapes are allocated by the operator when requested; and values specified for the VOLUME operand are ignored. The operands TSET and STATE=FOREIGN must not be specified together with DEVICE=WORK.

DEVICE=WORK should not be specified for multivolume files, since any available work tape is automatically assigned.

Magnetic tape cartridges cannot be requested as work tapes.

DISKWR

Only as of VERSION=3 and only relevant for files on pubsets or Net-Storage volumes:
Specifies the time after a write operation within which the file data must be in a consistent state. The entry is taken into the catalog entry for files on pubsets or Net-Storage volumes. The entry is taken into account when selecting the volume set for files on SM pubsets.

If DISKWR is not specified and the file has no catalog entry, the value IMMEDIATE is assumed for a permanent file and BY-CLOSE for a temporary file. If DISKWR is not specified and the file has a catalog entry, the value for DISKWR in the catalog entry is used.

The entry is rejected in the following cases:

- the file already occupies storage space
- SPACE is specified with a non-positive primary allocation

= IMMEDIATE
The file data must be in a consistent state immediately after a write operation.

= BY-CLOSE
The file data must be in a consistent state after the file is closed. This allows the file to be processed via a volatile cache.

DSPACE

Used with DDEVICE/DVOLUME for the data section of ISAM files with index and data separation:
DSPACE defines the space allocations for the data section of an ISAM file. The rules for primary and secondary allocation and for absolute allocation are the same as for the SPACE operand, but the entries refer to the volume specified in the DVOLUME operand (see also the DDEVICE and DVOLUME operands and “Index and data separation”, in the “Introductory Guide to DMS” [1]). NK-ISAM does not support separate data and index sections, but DSPACE may still be specified (compatibility with K-ISAM).

= <integer 0..2147483647>
Primary allocation, effective immediately.

= (<integer 0..2147483647>,<integer 0..32767>)
The primary allocation is effective immediately; the secondary allocation value is transferred to the catalog entry; 0 <= secondary <= 32767.

= (<integer 0..2147483647>,<integer 0..2147483647>,ABS)
ABS: absolute allocation;
The number of the PAM page at which the absolute allocation begins is specified followed by the number of PAM pages to be reserved.

DUPEKY

For ISAM files:
Specifies whether or not there may be more than one record with the same primary key value (duplicate keys).

If no value is specified in the FILE macro or FCB, the default value of the FCB macro takes effect on opening the file.

= NO
The file must not contain more than one record with the same primary key value.

= YES
If several records have the same primary key value, they do not overwrite each other, but are written sequentially in the order in which they are created. DUPEKY=YES is of significance only if the ISAM file is created sequentially by means of the PUT macro or extended non-sequentially by means of the STORE macro. The INSRT macro cannot be used to write records with identical primary keys.For DUPEKY=YES, see also "FCB - Define file control block".

DVOLUME

Used together with DDEVICE for K-ISAM files with separate index and data sections on a private volume:
DVOLUME specifies the volume serial number (“vsn” of the volume on which the data section of the ISAM file is to be stored. The VOLUME operand must be specified for the index section. The explanation of the DDEVICE operand applies analogously. NK-ISAM does not support separate data and index sections, but DVOLUME may still be specified (compatibility with K-ISAM). Separate data and index sections are not possible for files on a Net-Storage volume.

= <@addr>
Only as of VERSION=2:
addr is a symbolic address in the program at which a DVOLUME list has been created using the macro FILELST DVOLUME=...
The character “@” is part of the operand value and must be specified.

= PRIVATE
Issues a MOUNT message for a private disk on the console.

= (PRIVATE,<integer 1..9>)
Issues a MOUNT message for the required number of private disks on the console.

= list-poss(255): <name 1..6>
The private disks specified with their VSN are required for the data part of the ISAM file.

EXC32GB

Only for disk files; not for non-PAMKEY files:
The EXC32GB operand determines whether or not the file size may grow beyond 32 GB during data processing (see "Files larger than 32 GB"). The operand is entered in the TFT (Task File Table) and is not evaluated until the file is opened with OPEN.
EXC32GB has no influence on storage space allocations in the event of FILE calls.

= FORBIDDEN
The file size may not exceed 32 GB.

= ALLOWED
The file size may exceed 32 GB.

FCBTYPE

Specifies the access method to be used for file processing.

= *BY-PROG
Only as of VERSION=3 and only relevant if the DATATTR operand is specified:
The FCBTYPE value from the reference file catalog entry is ignored.

= ISAM
“pathname” is an ISAM file. Depending on the BLKCTRL operand, it is processed as an NK-ISAM file (BLKCTRL=DATA) or as a K-ISAM file (BLKCTRL=PAMKEY). The access method ISAM is described in the “Introductory Guide to DMS” [1].

ISAM-specific operands: DUPEKY, KEYLEN, KEYPOS, LOGLEN, POOLLNK, VALLEN, WROUT and DDEVICE, DSPACE, DVOLUME and VALPROP.

= BTAM
“pathname” is a tape file which is to be processed with the access method BTAM. The access method BTAM is described in the “Introductory Guide to DMS” [1].

BTAM-specific operands: CHAINIO, OPEN=SINOUT, STREAM

= PAM
“pathname” is a PAM file and is processed with the access method UPAM The access method UPAM is described in the “Introductory Guide to DMS” [1].

PAM files could be stored on disk or tape.

= SAM
“pathname” is a SAM file on disk or tape. SAM files are generally processed sequentially with the access methods SAM or UPAM. The access method SAM is described in the “Introductory Guide to DMS” [1].

SAM-specific operands: CLOSMSG, OPEN=UPDATE

FSEQ

For tape files which belong to a file set:
specifies the (sequence) number of a file within the file set. If, for example, several files with the same name are stored on one tape, access to a specific file is controlled via FSEQ. This also applies to MF/MV sets.

If no FSEQ value exists in the TFT entry or the FCB at the time the file is opened, FSEQ=1 is entered for files that have not yet been opened. In the case of files that have already been opened, the FSEQ value is taken from the catalog entry.

A catalog entry for the file must exist if FSEQ is specified as a null operand. If a file sequence number is entered there, this is transferred to the TFT entry. Otherwise, no file sequence number is entered in the catalog entry and the null operand is entered in the TFT entry.

= UNK
If the file already has a catalog entry containing a file sequence number, this is transferred to the TFT entry. Otherwise, no file sequence number is entered in the catalog entry and UNK is entered in the TFT entry. This means that when opening a foreign tape file with standard labels, the tape is searched for the file and positioned accordingly.

= NEW
Only permitted if no creation date is entered in the catalog entry:
The value NEW is entered in the TFT entry but not in the catalog entry. If a file sequence number is entered in the catalog entry, it is deleted. This means that a non-existent (NEW) tape file with standard labels is written after the current end of the file set and the file sequence number is incremented by one when the file is opened.

= <integer 0..9999>
If a catalog entry with creation date exists for the file, the FSEQ entry must match the file sequence number in the catalog entry. Otherwise (particularly for foreign files), the number is entered in both the catalog and TFT entries as a file sequence number. The tape is positioned according to the file sequence number when the file is opened.

Both FSEQ=0 and FSEQ=1 designate the first file of the file set.

IOPERF

Only as of VERSION=2 and only relevant for files/file generations on public volumes or Net-Storage volumes:
Specifies the performance attribute of the file. This defines the desired priority level for the I/O operations specified in the IOUSAGE operand. The highest permitted performance attribute is defined in the catalog entry (see the output of the SHOW-USER-ATTRIBUTES command).

When the file is being cataloged, the specification with the highest permissible performance attribute is compared and is transferred to the catalog entry, or STD is entered if IOPERF is not specified.

If the file is cataloged, the corresponding specifications in the catalog entry are not changed. If a file link name is specified in the LINK operand, the value is transferred to the TFT entry if IOPERF was specified.

When a new file is created on an SM pubset, the performance attribute is taken into account when selecting the volume set (e.g. selection of a volume set to which a cache is assigned).

= STD
The file is not processed via a cache.

= VERY-HIGH
If possible, a cache should be used when processing the file, and the whole file should be permanently maintained in the cache (highest performance priority).

= HIGH
The file should be processed via a cache if possible.

= USER-MAX
The file is given the highest I/O attribute that is entered for the user in the user catalog.

IOUSAGE

Only as of VERSION=2 and only relevant for files/file generations on public volumes or Net-Storage volumes:
Specifies the I/O operations to which the performance attributes (IOPERF) of the file apply. When the file is being cataloged, the specification is transferred to the catalog entry, or RDWRT is entered if IOUSAGE is not specified.

If the file is already cataloged, the corresponding specifications in the catalog entry are not changed.

If a file link name is specified in the LINK operand, the value is transferred to the TFT entry if IOUSAGE was specified.

When a new file is created on an SM pubset, the IOUSAGE attribute is taken into account when selecting the volume set (e.g. selection of a volume set to which a read cache is assigned).

= RDWRT
The performance attribute applies to read and write operations.

= WRITE
The performance attribute applies to write operations only.

= READ
The performance attribute applies to read operations only.

KEYLEN = length

For ISAM files:
specifies the length of the ISAM key.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

= *BY-PROG
Only as of Version=3 and only relevant if the DATATTR operand is specified:
The KEYLEN value from the reference file catalog entry is ignored.

= <integer 1..255>
Length of the ISAM key in bytes.

KEYPOS = number

For ISAM files:
specifies the position of the primary key in the record. In variable-length records, 4 bytes for the record length and control fields must be taken into account. The primary key may be anywhere in the record, but must be in the same position in each record of one file.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

= *BY-PROG
*Only as of Version=3 and only relevant if the DATATTR operand is specified:*The KEYPOS value from the reference file catalog entry is ignored.

= <integer 1..255>
Byte position of the primary key.

LABEL

For tape files:
Specifies the label attributes for files on magnetic tape or magnetic tape cartridge; the SECLEV operand determines how the labels are processed.

For existing tape files, the label standard version in the VOL1 label always applies. The LABEL operand is evaluated for output files (OPEN OUTIN/OUTPUT). If the tape already contains files or file sections, the standard identifier in the VOL1 label is set or updated as specified in the LABEL operand.

= *BY-PROG
Only as of Version=3 and only relevant if the DATATTR operand is specified:
The LABEL value from the reference file catalog entry is ignored.

= STD
File and volume already have or are to receive standard labels in accordance with DIN 66029, exchange level 1.

= (STD,<integer 0..3>)
File and volume already have or are to receive standard labels in accordance with the specified DIN 66029 exchange level; exchange level 4 is in preparation.
For specification and effects of the operand, see also the table “Effects of the LABEL operand” in chapter "FCB - Define file control block".

= NO
Labels are neither read nor written (no file label processing). If the tape has standard labels, the system processes the volume labels and checks the access authorization.

= NSTD
The tape file already has or is to receive nonstandard labels and file label processing is performed in the user program. If the volume has standard labels, the system processes them and checks the access authorization.

LINK = <name 1..8>

A TFT entry is to be created for this file link name (“name”). The other operands are then evaluated and their values are placed in this TFT entry (apart from SPACE, DSPACE, AVAIL, WORKFIL, VOLSET, STOCLAS and DISKWR). Volumes are requested from the volume list if necessary.

If the TFT already contains an entry with the same name, it is first implicitly released and then set up again with the current values in the FILE macro. The old TFT entry must not be in the active state. If the old TFT entry had been locked by means of a LOCK-FILE-LINK command, the new entry is also locked. Furthermore, the old volume and device reservations are cleared, but tape devices remain available to the job.
The program and the file are linked together via the file link name and the TFT.

The TFT entry is created in the task of the caller if this is not an RFA case. If the catalog ID in the path name belongs to a remote system to which an RFA connection exists, the TFT entry is created in the remote task and another one is created in the task of the caller. The TFT entry created in the task of the caller must not contain all the information in the remote task TFT entry, e.g.:

- user ID in the file path name
- information that cannot be specified via the FILE macro operands
- information on the TFT entry volume table
- entries for IOPERF, IOUSAGE, DEVICE, DDEVICE, FSEQ, MOUNT

- values transferred with the DATATTR operand from the reference file catalog entry

If the file link name is to be accessed via the command interface, it must correspond to the data type <structured_name 1..8> (see the “Commands” manual [3]).

Note

If the LINK operand is not specified, no TFT entry is created.

Most of the operands of the FILE macro are evaluated only together with a TFT entry. Exceptions to this are the operands whose values are transferred to the catalog entry or operands which control FILE processing such as:
IOPERF, IOUSAGE, DEVICE, VOLUME, SPACE, DDEVICE, DVOLUME, DSPACE, FSEQ (in part) MOUNT, and STATE=FOREIGN.

LOCKENV

Only as of VERSION=3:
Defines whether the file can be opened for writing by multiple systems during processing, dependent on the open mode and SHARUPD value.

= HOST
The file cannot be opened for writing by multiple systems during processing.

= XCS
The file can be opened for writing from different systems during processing by means of SHARUPD=YES if both systems belong to the same XCS network.

LOGLEN = <integer 0..255>

For ISAM files:
specifies the length (in bytes) of the logical flag in the ISAM index; the maximum length is determined by the length of the ISAM key and the length of any existing value flag (see the description of the VALLEN operand, "FILE - Define file attributes / control file processing"), since the entire ISAM index must not be longer than 255 bytes.

The following rule thus applies:

length <= 255 - KEYLEN - VALLEN

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

In the ISAM index, the ISAM key may be followed by a logical flag in which selection criteria are defined bit-by-bit and encoded in binary code. In K-ISAM files, all logical flags of a block are evaluated and the result is placed in the next-higher index entry. NK-ISAM supports logical flags only compatibly, but does not place the flags in the index entry. Any LOGLEN specification is ignored.

= *BY-PROG
Only as of Version=3 and only relevant if the DATATTR operand is specified:
The LOGLEN value from the reference file catalog entry is ignored.

MF

The forms of the MF operand are described in detail in the appendix ("Macro types"). The version operand must have the same value for all macro calls that are distinguished by the MF operand (MF=L/E/D/C/S).
MF=S can only be used for VERSION=0.

Default: MF=S for VERSION=0.

MOUNT

Specifies which volume is requested from the volume list (see VOLUME operand) for FILE processing.

The interactions with the VSEQ operand should be borne in mind:

- VSEQ: MOUNT values must be greater than or equal to the VSEQ value (exception: MOUNT=0). If VSEQ=n, the MOUNT list must begin with “n”
  (MOUNT=(n[,n+1][,n+2][,...); if VSEQ=(L=(n_1, n_2,...)), the MOUNT list must consist of the first k elements of the VSEQ list (MOUNT=(n_1, n_2,...,n_k)).
- If the VSEQ operand is not specified, the MOUNT list must begin with “1” (MOUNT = 1, 2, ..., k)).
  Exception: MOUNT=0.

The following applies when requesting public disks:

- No disks are requested if LINK is specified.
- No disks are requested if the file is migrated.
- No disks are requested if MOUNT=0 is specified.
- All MOUNT specifications apart from MOUNT=0 are rejected if LINK is specified.
- All disks are removed from the volume list if LINK is specified without MOUNT and the file is not migrated.

The following applies when requesting private disks:

- The specification MOUNT=0 is ignored if at least one of the operands SPACE, VOLUME, DSPACE, DVOLUME or REUSE (using the oldest generation volumes when creating a new generation) is specified.
- No private disk is requested if MOUNT=0 is effective.
- If MOUNT=0 is not effective and LINK is not specified, the first private disks are requested with extent and, if necessary, any additional private disks required for the storage allocation.
- All disks from the volume list are requested if LINK is specified and no MOUNT specification is effective.
- The first k disks from the volume list are requested if LINK is specified and k non-zero numbers are specified in MOUNT.

The following applies when requesting Net-Storage volumes:

- When MOUNT=0 but neither SPACE nor VOLUME is specified and the file is already on a Net-Storage volume before the FILE call, no Net-Storage volume is requested.
- In all other cases the Net-Storage volume is requested on which the file resides or is created.

The following applies when requesting tapes:

- No tapes are requested if DEVICE=WORK is specified.
- No tapes are requested if MOUNT=0 or neither LINK nor MOUNT is specified.
- Tapes are requested according to the MOUNT list if MOUNT is specified not equal to zero.
- If LINK is specified without MOUNT, just one tape is requested from the volume list: if

VSEC=n is specified, the nth tape, if VSEQ=(L=(n_1, n_2,...)) the n_1th tape and if VSEQ is not specified, the first tape.

- Every number n>0 in the MOUNT list refers to the nth tape in the volume list.

= 0
For disk files:
The volume is requested only at the time of the OPEN, provided that neither VOLUME/DVOLUME nor SPACE /DSPACE has been specified.

For tape files:
The tape is not requested until the OPEN.

= @addr
Only as of VERSION=2:
addr is a symbolic address in the program at which a MOUNT list was stored using FILELST MOUNT=...
The “@” character is part of the operand value and must be specified.

= list-poss(255): <integer 1..255>
Every number specified here refers to the nth volume in the volume list.

NFTYPE

Only as of VERSION=3 and only relevant for files on Net-Storage volumes:
Specifies the file type for the Net-Storage file to be created.
If this specification contradicts the specifications in the DEVICE and VOLUME operands (e.g. specification of a private disk), the macro is aborted with an error. If the DEVICE and VOLUME operands are not specified, the file is created with the specified file type on an arbitrary Net-Storage volume (if available).

= BS2000
The file is created on Net-Storage as a BS2000 file.

= NODE-FILE
The file is created on Net-Storage as a node file.

OPEN

Specifies the OPEN mode for the file. This setting may be overwritten by the OPEN mode specified in the OPEN macro.
If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

For the possible specifications for the various access methods, see also table “OPEN modes with the FCB macro” in chapter FCB - Define file control block. The various OPEN modes are also described in detail in the descriptions of the access methods.

= INPUT
“pathname” is an input file, i.e. it must exist.

= EXTEND
An existing file is extended, i.e. further data blocks are added to the end of the file or the file is overwritten from a certain position onwards; only sequential write operations are permitted. Labels are created for tape files dependent on the LABEL specification.

= INOUT
An existing file is opened for non-sequential processing; write and read operations are permitted. After OPEN is completed with tape processing, the tape is positioned to the start of tape and no further labels are written.

= OUTIN
A file is created or, if it already exists, overwritten from the beginning. Both read and write operations are permitted (non-sequential). Labels are written for tape files.

= OUTPUT
A file is created or, if it already exists, overwritten from the beginning. Labels are written for tape files.

= REVERSE
The file “pathname” must already exist and is opened as an input file for sequential reading from end-of-file -> beginning-of-file. Disk files file must not extend over several volumes. For tape files, no automatic spool swap is possible. A single section of the file can be processed (the tape concerned is to be selected using VSEQ if necessary). Tape files are positioned to the end of the file section after OPEN processing.

= SINOUT
Only for BTAM tape files:
The file must exist and the tape must not be positioned to the beginning of tape. Data blocks can be read or written. In contrast to INOUT, the tape is not positioned.

= UPDATE
Only for SAM disk files:
The records of the file can be updated by retrieving them with GET and writing them back with PUTX (this is only possible in locate mode).

OVERLAP

For ISAM files:
If this is specified and a second I/O area is defined in the program (IOAREA2 in the FCB), read operations (GET /GETR) can be executed in overlapped mode.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

For NK-ISAM, “overlapped processing” means that neighboring blocks are also read into the ISAM pool. OVERLAP=YES should be used only when reading is predominantly sequential.

= YES
Read operations are executed in overlapped mode.

= NO
Read operations are not executed in overlapped mode.

PAD = <integer 0..99>

For ISAM files created sequentially (using the ISAM macro PUT):
the “padding factor” PAD specifies how much free space is to be left in each data block for subsequent extension of the file (specified as a percentage of the block size defined by means of BLKSIZE). PAD thus has an effect on the block splitting rate when a file is extended non-sequentially.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

The PAD specification has different effects for NK-ISAM and K-ISAM. For NK-ISAM, the block is filled at least up to the PAD limit; for K-ISAM, it is not filled above the PAD limit.

POOLLNK = <name 1..8>

Only as of VERSION=1 and for ISAM files processed in user ISAM pools (NK-ISAM):
“name” is the “pool link name” (up to 8 characters long) which is entered in the TFT. This pool link name must be assigned by means of ADDPLNK to an ISAM pool created using the CREPOOL macro. This name is passed to NK-ISAM at OPEN time; an I/O buffer for the file is created in the appropriate ISAM pool.
Valid character set for “name”: letters, digits, and special characters (in accordance with the rules for file names).
If the pool link name is to be accessed via the command interface, it must correspond to the data type <structured_name 1..8> (see the “Commands” manual [3]).

POOLSIZ = <integer 128..1048576>

Size of the file-specific ISAM pool in units of 2048 bytes.
The specification does not refer to the ISAM pool referenced with POOLLNK.

PREFIX

Only evaluated in conjunction with MF=C or D; this defines the first character of field names and equates that are generated in the data area during macro expansion.

= I
The prefix with which field names and equates generated by the assembler begin.

= *
No prefix is generated.

= <name 1..1>
Prefix with which the generated field names and equates are to begin.

RECFORM

Specifies the record format of the file designated by “pathname” and also specifies which control characters are to be interpreted if the file is sent to a printer.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

The record format is evaluated for the access methods SAM and ISAM. UPAM processes files only on a block basis and any RECFORM specification is ignored. BTAM is also a block-oriented access method, but accepts a RECFORM specification. The record formats are described in detail in the section on access methods in the “Introductory Guide to DMS” [1]. For the relationship between the RECFORM and RECSIZE specifications, see the RECSIZE operand. For information on evaluation of the print control characters, refer to the PRINT-DOCUMENT command (LINE-SPACING operand) in the manuals “Commands” [3] and “SPOOL” [4].

= *BY-PROG
Only as of Version=3 and only relevant if the DATATTR operand is specified:
The RECFORM value from the reference file catalog entry is ignored.

= V
“pathname” consists of variable-length records, which means that the user must remember, when programming, that each record is preceded by a 4-byte field whose first two bytes contain the record length in binary form. Bytes 3 and 4 of this field are used by the system. For input files, the record length field is set by the system; for output files, this must be done by the user. The value specified for RECSIZE is the maximum permissible record length. For BTAM files, the specification RECFORM=V is treated like RECFORM=U.

= F
“pathname” consists of fixed-length records, i.e. the user does not need to worry about any record length and control fields. All records in the file have the same length, which is defined via the RECSIZE operand (see "FILE - Define file attributes / control file processing").

= U
“pathname” consists of records with “undefined” length. Each data block contains only one record, whose length is passed in a register. The system sets this register for input and the user must set it for output (see the BLKSIZE operand, "FILE - Define file attributes / control file processing").
RECFORM=U converts the specification LABEL=(STD,3) into (STD,2).
RECFORM=U is not permitted for ISAM files.

= (...,N)
“pathname” is not a print file and therefore contains no printer control characters. It should not be printed with control character evaluation.

= (...,M)
The first data byte in each record is interpreted as a control character in EBCDIC code. The file can be printed with the command PRINT-DOCUMENT ...,LINE-SPACING=*BY-EBCDIC-CONTROL. For ISAM files, the ISAM index is taken into account.

= (...,A)
The first data byte in each record is interpreted as an ASA control character. The file can be printed with the command PRINT-DOCUMENT ...,LINE-SPACING=*BY-ASA-CONTROL.

RECSIZE

Specifies the record length, depending on the specification in the RECFORM operand.If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

= *BY-PROG
Only as of Version=3 and only relevant if the DATATTR operand is specified:
The RECSIZE value from the reference file catalog entry is ignored.

= <integer 0..32768>
For RECFORM=V: the specification for RECSIZE is ignored, except in the following case: if an ISAM file is read in move mode, and if the value specified for RECSIZE is less than the length of the record which is read, only the length specified for RECSIZE is actually read and error handling (DMS0AAD) is initiated.

For RECFORM=F: the record length in bytes; all records in the file are the same length.

= <reg 2..12>
For RECFORM=U: the RECSIZE operand must specify a general-purpose register (2 <= reg <= 12) which contains the current record length for input and output. The system sets this register for input and the user must set it for output.

RETPD = <integer 0..32767>

By means of “RETPD”, the user can define a retention period during which no write access (update, delete) is possible.

If no value is specified in either the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

Once the retention period has elapsed, the file is not automatically erased; write access is simply permitted again.

The retention period can also be controlled via the CATAL macro:
any RETPD specification in CATAL is immediately placed in the catalog entry. For tape files, the CATAL macro can be used only before the file is opened for the first time.

RETPD is ignored for temporary files.

SECLEV

For tape files:
the operand SECLEV (security level) refers to the TPIGNORE entry in the JOIN file. A SECLEV specification is ignored in interactive mode. In batch mode, users with the appropriate authorization can use the SECLEV operand to specify whether error messages are to be suppressed and/or whether additional label checking is to be executed.

= HIGH
In batch mode, error messages are sent to the console. If the job is running under a user ID with TPIGNORE=YES in its JOIN entry, the operator can ignore the error messages.

= LOW
Permissible only for the tape/file owner if TPIGNORE=YES is defined in the user catalog entry: certain error messages are suppressed in batch mode.

= (...,OPR)
The entry OPR (= overwrite protection) causes the system to execute additional label checking:

- if a file is written on a tape behind an existing file, the labels of the preceding file are checked;
- the expiration date of the new file must not be greater than that of the preceding file.

SHARUPD

For ISAM or UPAM disk files:
specifies whether several jobs may concurrently open the file with an OPEN mode other than OPEN INPUT.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

= NO
As soon as the file is opened by a job with OPENINPUT, it is locked for all other jobs. Concurrent access to the file by several jobs is possible only if the file is used as an input file by all of these jobs, i.e. it is opened with OPEN INPUT. If the file has been opened with OPEN INPUT, any attempt to open it with another OPEN mode is rejected.

= YES
Only for ISAM and PAM files:
the file can be processed concurrently by several jobs, provided SHARUPD=YES is specified in all of these jobs. With UPAM, the user can protect data blocks from access by other jobs as long as he/she is processing them. For ISAM, any locks that are necessary are set automatically by the system: with NK-ISAM, as record key locks; with K-ISAM, as block locks. With NK-ISAM, files which are opened for shared-update processing must be processed in host-specific ISAM pools. SHARUPD=YES for ISAM files simultaneously activates the WROUT function (see operand WROUT, "FILE - Define file attributes / control file processing").

= WEAK
Only as of VERSION=2, *for UPAM processing:*
guarantees write protection but no read protection. Only one job can open the file for an update, but other jobs may use it as an input file at the same time. The user must make allowances in the program for the fact that the contents of the file may change during the period in which it is used as an input file.

SPACE

Only for disk files:
Controls, via the primary, secondary or absolute allocation, the storage space reserved for the file “pathname”. The SPACE operand is always evaluated, even if the LINK operand is not specified in the current FILE macro. Primary, secondary and absolute allocations are described in detail in the section “Requesting storage space” in the “Introductory Guide to DMS” [1].

Default value: The following applies if SPACE is not specified:

- If the file has already been allocated storage space, its storage space and its secondary allocation remain unchanged.
- Otherwise default values apply for primary and secondary allocation. For files on Net-Storage these defaults are permanently defined by the storage space management. For other disk files these defaults are derived from values which can be set by the system administrator.

A FILE macro with a SPACE operand is rejected for files which are open or reserved exclusively for another task. The protection attributes of a file or file generation group are also taken into account.
If the user requests more space on the pubset than is assigned to him in the user catalog entry, the FILE macro is rejected.
If the user is authorized to override the allocated storage space, the system informs the user with a message.

In order to minimize the management overhead for the system and storage space utilization, the following should be noted when making primary and secondary allocations:

- the primary allocation should match the estimated size of the file to be created;
- the secondary allocation should be sufficient to cover the anticipated expansion of the file to be created

When a file with BLKSIZE=(STD,n) is opened, where n >= 2, the following must apply for the number “p” of PAM pages reserved for the file, and for “s”, its secondary allocation:

File type	SPACE operand p	SPACE operand s
SAM	>= 2n	>= n
K-ISAM	> n	
NK-ISAM	> n	
PAM (chained I/O)	> 0	

= <integer -2147483647..2147483647>
The primary allocation, which takes effect immediately.

In the following description, k represents the number of PAM blocks per unit (smallest management unit for the storage space management of disk files; for more information on units see the section “Requesting storage space” in the “Introductory Guide to DMS” [1]).

1...2147483647:
The storage space allocation is rounded up to a multiple of k and the corresponding number of PAM pages is allocated on the pubset or on the private disk. Users should note that each FILE macro with a positive primary allocation reserves space for the file. If the primary allocation is large, this will quickly exhaust the user's storage space contingent.

For files on public disks and private disks the disks for storage allocation are determined as follows:

- Storage space on public volumes is allocated if the VOLUME operand is not specified and the file does not yet occupy storage space.
- The following parameters are considered when selecting the volume set for files on SM pubsets:
  - preliminary file format
  - values of AVAIL, WORKFIL, VOLSET, IOPERF, IOUSAGE, DISKWR
  - permanent/temporary attributes
  - storage class assigned to the file.
- If the VOLUME operand is specified and the file already occupies storage space, the disks already occupied by the file are used for storage space allocation as far as possible.
- If the VOLUME operand is specified, storage allocation starts with the first disk received via the VOLUME operand. If this is not sufficient, allocation is continued with the second disk received via the VOLUME operand, etc.

Storage allocation to a pubset is rejected if the total number of free pages is less than was specified in the primary allocation.

For files on public disks and private disks a partial allocation is made if the VOLUME operand is specified and the disks received via VOLUME contain a total number of free PAM pages which is lower than was specified in the primary allocation (but at least one free unit). However, the FILE call is rejected if public disks are specified and there are fewer PAM pages free on the whole pubset than were specified in the primary allocation (see above).

The user ID entry in the user catalog contains its contingent of public storage space. The following applies if this is exceeded in the request for storage space by the file user ID: if, according to the user catalog entry, the user ID has the right to exceed the contingent, the calling task is informed with a warning that the contingent has been exceeded, otherwise the FILE call is rejected.

If possible, a partial allocation is made with private disks if the request exceeds the free storage space contingent.

Only the maximum possible partial allocation is made if a request would lead to the maximum file size that can be represented in the catalog entry (16777215 PAM pages) being exceeded.

-2147483647..-1:
Amount of storage space released after rounding the primary allocation up to a multiple of k. The space is released from the end of the file, working backwards, as specified in the volume list (any specification in the VOLUME operand is ignored). Only “unused” units are released. For ISAM files, the data and index sections cannot be released separately (see the DSPACE operand, "FILE - Define file attributes / control file processing").

If the file does not occupy storage space after this has been released, the following are deleted from the catalog entry: AVAIL, WORKFIL (not, however, for generations), STOCLAS indicator “File contains defective block”, indicator “S0 migration forbidden” and the interim file format.

If a BS2000 file on a Net-Storage volume has no more space after storage space has been released, it no longer exists on the Net-Storage volume. All references to the Net-Storage volume are removed from its catalog entry.

At least three PAM pages, and in the case of node files at least four PAM pages, remain allocated to the file in the case of files on private disks. For existing files with BLKSIZE=(STD,k), at least as many PAM pages remain allocated as are required for opening the file. The number of remaining PAM pages is defined in this case by storage management.

If DESTROY=YES is defined in the catalog entry, all released PAM pages are overwritten with X'00' (ignoring unit boundaries). Any required private disks are requested in this case. If DESTROY=NO is defined in the catalog entry, the released PAM pages are only overwritten if the destroy level (system parameter DESTLEV) is set high enough.

0:
No change to the storage space reservation; permissible for files on private disks only if the file already occupies storage space. Simultaneous specification of VOLUME is ignored if the file already occupies storage space, otherwise it is rejected.

= (<integer -2147483647..2147483647>,<integer 1..32767>)
Defines the primary and secondary allocations. In contrast to the primary allocation, the secondary allocation does not take effect immediately when the FILE macro is issued, but only if the reserved space runs out during creation or extension of the file. The secondary allocation value is placed in the catalog entry (field S-ALLOC in the output for the SHOW-FILE-ATTRIBUTES command).

<integer -2147483647..2147483647>:
see “primary” above.

<integer 1..32767>:
Secondary allocation, i.e. the number of PAM pages by which the storage is to be extended if required. The secondary allocation is transferred unchanged into the catalog entry. It is only rounded up to a multiple of k when it comes into effect.

SPACE=(0,secondary) defines or changes the secondary allocation and places the (new) value in the catalog entry. This may be specified for a file or file generation on private disk only if space has already been requested for this file or file generation.

(...,0):
Prevents dynamic expansion of the file.

= (<integer -2147483647..2147483647>[,<integer 1..32767>],*KEEP)
Only as of VERSION=2 with release of storage space for a file on public volumes or Net-Storage volumes:
“*KEEP” means that at least one allocation unit remains assigned to the file.

= (<integer -2147483647..2147483647>,<integer -2147483647..2147483647>,ABS)
Absolute allocation (only together with VOLUME). If there is not enough free space on the disk, the FILE macro is rejected; no partial allocation is made. Since the absolute allocation always refers to one volume, a separate FILE macro must be issued for each volume.

If the absolute allocation is the first space request for the file, the secondary allocation is set to 0. The following is specified:

a. Block number of the PAM page at which the space reservation is to start on the private disk. Since space is always allocated in units, “page” must be k*n + 1 (where n >= 0). The first PAM page on a disk at which storage space can be reserved depends on how the disk was formatted.

b. Specifies how many PAM pages are to be reserved on the volume. It must be a multiple of k. As the capacity of a given disk depends on the disk type and how it was formatted, the user should ask the system administrator what the maximum permissible value is. The upper limit for this maximum value is 2147483647 (as for the primary allocation).

c. *ABS:* The keyword “ABS” identifies an absolute allocation

Absolute allocation is not possible for a file on a Net-Storage volume.

STATE = FOREIGN

For files on private volumes or on Net-Storage volumes of the NETSTOR type for which no entry exists in the system catalog, a catalog entry is created (file import). For file generations, the group entry must also be reconstructed (by means of a CATAL macro) before the generations can be imported. Files which are imported with STATE=FOREIGN should be exported from the catalog of their “old” owner (by means of ERASE CATALOG).

The VSNs of the volumes required for file processing must be listed in the VOLUME operand in the correct order. The VOLUME specification can be omitted if MAREN is available and the file volumes are in the MAREN catalog. MAREN then supplies the VSNs.

The following must not be specified together with STATE=FOREIGN:
DEVICE=WORK, TVSN, TSET, VSEQ.

Files on private disk:
The catalog entry is created from the F1 label of the first private disk specified in the VOLUME operand or received via MAREN. The file can only be imported to the user ID contained in the F1 label. The file may also be imported to pubsets other than those on which it was first cataloged. A file cataloged in the F1 label as shareable can be imported to the user ID contained in the F1 label by any task (i.e. regardless of the task user ID).

Files on Net-Storage:
The catalog entry is created from the catalog on the Net-Storage volume of the NETSTOR type which is specified in the VOLUME operand. Specifying the VOLUME operand for a Net-Storage volume of the NETVOL type is not permitted. The file can be imported only to the pubset which is allocated to the Net-Storage volume of the NETSTOR type which is specified in the VOLUME operand.

Tape files:
File attributes of a foreign file cannot be changed by means of a CATAL macro.

If the foreign tape file has standard labels, the file attributes RECFORM, RECSIZE, BLKSIZE and CODE are transferred from the HDR2 label to the catalog entry when the file is opened. The file may be cataloged under more than one user ID; the system then ensures that the catalog entries and the label information are kept consistent.

If the foreign file has nonstandard labels or no labels, the user must specify the operands RECFORM, RECSIZE and BLKSIZE in the FILE macro. If the file is cataloged under more than one user ID, each user is responsible for ensuring that the catalog entry and the label information are kept consistent.

If a foreign tape file with standard labels is to be imported, the following must apply: if the user is not the file owner, the volume and the file must be shareable (indicators in the VOL1 and HDR1 labels).

STOCLAS

Only for VERSION=3:
When a file is created on an SM pubset, it can be assigned a storage class. This then contains an attribute that satisfies the file storage location requirement. If the storage class is assigned to a volume set list, the file is preferably stored on a volume set from this list.

A storage class-relevant entry exists in the following cases:

- If one of the operands AVAIL, DISKWR, VOLUME, VOLSET or WORKFIL is specified.
- If a value other than NETSTOR and other than STDDISK is specified for the DEVICE operand.
- If a value not equal to the null operand is specified for either the IOPERF or IOUSAGE operand.

A default storage class can be stored in the entry for each user ID in an SM pubset's user catalog. This can be displayed using SHOW-USER-ATTRIBUTES INF= PUBSET-ATTR.

When a file or file generation is created on an SM pubset under a user ID which possesses a default storage class on the SM pubset then the following applies: if there is no right to perform physical allocation and the file has not been created on a volume set for work files then specifications relating to storage classes and STOCLAS=*NONE become ineffective, i.e. they are ignored, provided that they are not rejected. (IOPERF and IOUSAGE are nevertheless entered in the TFT entry.)

If a file (not a file generation) is created on an SM pubset under a user ID which possesses a default storage class on the SM pubset then the user-specific default storage class is assigned to the file if no STOCLAS is specified and no storage class-relevant entry exists.

A default storage class can also be stored in an FGG index. This is assigned to an SM pubset when a file generation is created if no STOCLAS is specified and no storage class-relevant entry exists.

If a default storage class is stored in the file user ID entry or the FGG index and this storage class does not exist on the SM pubset concerned or the user is not authorized to access it, the value NONE or another storage class must be specified in the STOCLAS operand in order to create the file or file generation.

If the file is created on an SF pubset or a private volume, it is not assigned a storage class even if a storage class name is specified or a default storage class exists.

A storage class can also be assigned if a file is created on a Net-Storage volume; in this case no work file can be created, nor a file with a PAM key.

= *NONE
The file is not assigned a storage class, an existing default storage class is also not assigned.

= <c-string 1..8>
Name of the storage class assigned to the file. The entry is rejected in the following cases:

- the file already occupies storage space
- SPACE has been specified with a non-positive primary allocation
- an storage class-relevant entry exists
- the storage class does not exist on the SM pubset concerned
- the caller is not authorized to access the storage class.

STREAM

For BTAM tape files:
Enables streaming mode to be used for I/O. This means that the chained I/O jobs (CHAINIO operand) offered in MAV mode (BTAMRQS operand in the FCB call and REQNO in the BTAM call) are themselves chained. It also means that hardware “streaming” mode is to be set if a tape streamer is used.

= NO
Streaming mode is not set unless STREAM=YES is specified in the FCB of the program.

= YES
Streaming mode is set.

TAPEWR

Only as of VERSION=1, for files on tape cartridges:
The user can specify whether or not output is to be buffered.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

= DEVICE-BUFFER
Output is buffered in the tape controller, thus causing a high data transfer rate.

= IMMEDIATE
Output is not buffered.

TPMARK

For tape files without standard labels (LABEL=NO/NSTD):
Specifies whether tape marks are to be written when a tape file is created. Tape files with LABEL=(STD,n) automatically receive tape marks after the labels.

= NO
No tape mark is written for tape files without standard labels, unless TPMARK=YES is defined in the FCB of the program.

= YES
Tape files with nonstandard labels: the tape mark follows the label.

Tape files without labels: the tape mark is written at the beginning of the tape.

TRANS

For tape files used as input files and not created with CODE=EBCDIC:
specifies how the code of the file is to be converted during reading.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

= YES
ISO 7-bit code or OWN code is converted into EBCDIC code.

= NO
ISO 7-bit code is converted into 8-bit format by inserting a leading zero.

TSET

For output tape files with standart labels:

Together with the LINK operand, this creates a tape set for the file via a TST entry (in the tape set table) or sets up a link to an existing TST entry. The corresponding TFT entry then points to the linked TST entry.

A TST entry consists basically of a TSET name and a volume list; the volume list can be defined or extended via the VOLUME operand. Subsequent FILE calls may refer to this volume list by specifying the TSET name, and may extend the list if necessary.

The following applies if TSET is specified together with LINK:

- If the current TSET name does not yet exist in a TST entry, a new entry is created with TSET-SHR=1 (TSET-SHR shows the number of TFT entries linked to this TST entry; see the output of the SHOW-FILE-LINK command).
- If there is already a TST entry with the same name, and if a file link name which does not exist in the TFT is specified, TSET-SHR is incremented by 1.
- When a TFT entry which is linked to a TST entry is released, the TSET-SHR of this entry is decremented by 1. If this results in TSET-SHR=0, the TST entry is also released.

The following conditions must be fulfilled if TSET is specified:

- The volume table in the catalog entry must be empty if a cataloged file is specified.
- The DEVICE operand must be specified if a new file is specified.
- If the DEVICE operand is specified, its value must be a tape type.

The following operands must not be specified together with TSET:
STATE=FOREIGN, DEVICE=WORK, VSEQ, TVSN, FSEQ=UNK, FSEQ=n where n>1, FSEQ=null operand, VOLUME=REMOVE-UNUSED

= <name 1..4>
Tape set name in the TST entry which is used as a reference. If the TST entry does not exist or has no file set identifier and the VOLUME operand is specified, the first VSN received via the VOLUME operand is entered as the file set identifier.

= (<name 1..4>,<name 1..6>)
The four-character name designates a TST entry, the six-character name (VSN) is the file set identifier.

If the TST entry already exists, the file set identifier in the TSET specification must match the file set identifier in the TST entry. When a file is opened, the file set identifiers in the TST entry and in the HDR1 label must be the same.

TVSN

Only for tape files used as input files:
Specifies a temporary list of volume serial numbers for processing, which constitutes the volume list. If the TVSN operand is specified, the volume list in the catalog entry is ignored during file processing; only the volumes specified via TVSN are used. However, the catalog entry is not changed.

The following operands must not be specified together with TVSN:
*DUMMY, STATE=FOREIGN, TSET, VOLUME

= <@addr>
Only as of VERSION=2:
addr is a symbolic address in the program at which a TVSN list was stored with the macro FILELST TVSN=....
The “@” character is part of the operand value and must be specified.

= list-poss (255): <name 1..6>
Defines the volume VSN required for input.

VALLEN = <integer 0..255>

For K-ISAM files:
specifies the length of the value flag in the ISAM index.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

For K-ISAM files, the value flags are evaluated block-by-block and transferred to the next higher index entry as specified in the VALPROP operand. For NK-ISAM files, the VALLEN specification is ignored.

= *BY-PROG
Only as of Version=3 and only relevant if the DATATTR operand is specified:
The VALLEN value from the reference file catalog entry is ignored.

VALPROP

For K-ISAM files:
VALPROP (VALue PROPagation) specifies how the value flag is to be included in the index entries.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

= *BY-PROG
Only as of Version=3 and only relevant if the DATATTR operand is specified:
The VALPROP value from the reference file catalog entry is ignored.

= MIN
The lowest value of the value flag within a data or index block is included in the index entry at the next higher level.

= MAX
The highest value of the value flag in a data or index block is included in the index entry at the next higher level.

VERSION

Specifies which version of the operand list and which SVC are to be generated.

Default value: VERSION=0

Note

The VERSION operand must have the same value in all FILE macros distinguished by the MF operand (MF=L/E/D/C).

= 0
The “old” format (SVC 159) of the operand list (as for BS2000 V9.0) is generated. If support is required for device types and functions which have been introduced since this version, a new format of the operand list (as of VERSION=1) must be used.

= 1
Valid from BS2000 V9.5 onwards:
An operand list with a standard header (SVC 144) is generated. Over and above the operands of BS2000 Version 9.0, this format also supports the operands BLKCTRL, CLOSMSG, DESTOC, POOLLNK and TAPEWR introduced in BS2000 Version 9.5 and the device types valid from V9.5 onwards.

= 2
Valid from BS2000 V10.0 onwards:
An operand list with a standard header (SVC 144) is generated. In contrast to the operand list for VERSION=1, the variable parts (for the operands VOLUME, DVOLUME, TVSN, MOUNT and VSEQ) of this operand list are stored in separate areas. These areas can be created by means of the FILELST macro.

The following operands and operand values are only supported as of VERSION=2:IOPERF, IOUSAGE, CLOSE, BLKCTRL=DATA2K/DATA4K, SHARUPD=WEAK and SPACE=(...,*KEEP)

= 3
Valid as of BS2000/OSD-BC V3.0: an operand list with a standard header (SVC 144) is created.

The following operands and operand values are only supported as of VERSION=3:AVAIL, CLOSE=KEEP-DATA-IN-CACHE, CODE=ISO7D, DATATTR, DISKWR, EXC32GB, LOCKENV, NFTYPE, POOLSIZ, STOCLAS, VOLSET, VOLUME=REMOVE-UNUSED, WORKFIL and the value =*BY-PROG with some operands.

This format of the operand list offers the best programming support; another factor that speaks in its favor is its suitability for use with future developments in the FILE macro.

VOLSET

Only as of VERSION=3 for files on SM pubsets:
Defines the volume set on which the file is to be created. The specification is rejected if the file already occupies storage space, if it is to be created on a Net-Storage volume, or if SPACE is specified with a non-positive primary allocation.

Specifying a volume set with permanent data storage requires permission for physical allocation.

= <c-string: catid 1..4>
Catalog ID of the volume set on which the file is to be created.

= *CONTROL
The file is created on the control volume set of the SM pubset.

VOLUME

Specifies which volumes are required for file processing.
If, when a file is being created, neither DEVICE nor VOLUME is specified, the file is created on public volumes.

Net-Storage volumes are regarded as disks and can be specified without the authorization for physical allocation.

If the first VSN obtained using the VOLUME operand identifies a Net-Storage volume which is assigned to the pubset on which the file resides or is to be created, the VSN remains assigned to the Net-Storage volume even if it also identifies a private disk.

For files on public disks, Net-Storage volumes or private disks:
The volume list contains all disks on which extents of the file are located (possibly after storage space allocation is completed).
DMS attempts to reserve all the space specified via SPACE on the first disk. “Unused” volume serial numbers are moved to the volume list of the catalog entry for subsequent file extensions.
If no storage space allocation is necessary, the specified volume serial numbers will be ignored.

For files on a Net-Storage volume:
The volume list consists of the VSN of the Net-Storage volumes on which the file is located or will be created. This VSN also determines the Net-Storage volume type (NETSTOR or NETVOL) on which the file is created. The volume table in the catalog entry and (if the LINK operand is specified) the volume table in the in the generated TFT entry also consist of this one VSN. The pubset on which the file is or will be cataloged must be assigned to the Net-Storage volume on which the file is located or will be created.

For tape files:
The volume list consists of the volume serial numbers in the catalog entry (if these exist), followed logically by the volume serial numbers from the VOLUME operand. By default, the first volume from the volume list is requested (unless MOUNT=0 is specified). If the uses requests more than one volume, the number of volumes to be mounted concurrently must be specified in the MOUNT operand.

If “pathname” is not yet cataloged, the volume serial numbers from the VOLUME operand are transferred to the catalog entry. Furthermore, the TSET operand can be used to establish a link to a TST entry.

The effects of the VOLUME operand depend on whether the TSET operand is specified. If it is not specified, the volume list is copied unchanged into the catalog entry. If TSET is specified, first the volume list of the TST entry is updated or created and then the volume list of the catalog entry is created according to the TST entry. After a file has been opened, the catalog entry is then updated with the information from the volume list of the TST entry.

If “pathname” is already cataloged, the volume serial numbers from the VOLUME operand form an extension to the volume table of the catalog entry. This means that the VOLUME operand may contain no volume serial numbers which already exist in the catalog entry.

= @addr
Only as of VERSION=2:
addr is a symbolic address in the program at which a VOLUME list has been created by means of the macro FILELST VOLUME=...
The character “@” is part of the operand value and must be specified.

= REMOVE-UNUSED
Only for already cataloged tape files:
All tapes which do not contain data from the file are removed from the catalog entry volume table.
LINK and TSET must not be specified together with REMOVE-UNUSED.

= PRIVATE
A private volume is required for file processing. The operator is requested via a message on the console to enter the volume serial number of the required volume.VOLUME=PRIVATE is ignored in a FILE call for the dummy file *DUMMY.

= (PRIVATE,<integer 1..9>)
A number of private volumes are required for file processing. The operator is requested via a message on the console to enter the volume serial numbers of the required volumes.
VOLUME=(PRIVATE,<integer 1..9>) is ignored in a FILE call for the dummy file *DUMMY.

= list-pos(255): <name 1..6>
Volume serial numbers of the requested volume.

VSEQ

For cataloged tape files with standard labels:
the VSEQ operand permits section-by-section processing of files. A file section is that part of a multivolume file which is stored on one tape (see the programming notes on "FILE - Define file attributes / control file processing" for the effect on the structure of the TFT volume list).

The VSEQ operand refers to the volume list (see the VOLUME operand). The file section numbers correspond to relative volume serial numbers, i.e. they specify the position of the volume serial number in the volume list.

Single value:	If only one file section number is specified in the VSEQ operand, all volumes from the specified entry onwards are transferred to the volume table of the TFT entry.
List:	If a list of file section numbers is specified in the VSEQ operand, the specified entries are transferred to the volume table of the TFT entry.

VSEQ must not be specified together with TSET or STATE=FOREIGN. If the file has not been cataloged or only cataloged with CATAL, all VSEQ specifications apart from VSEQ=1 are rejected.

= @addr
Only as of VERSION=2:
addr is a symbolic address in the program at which a VSEQ list has been created by means of the macro FILELST VSEQ=...
The character “@” is part of the operand value and must be specified.

= <integer 1..255>
Number of the section at which processing is to start.

If “pathname” is an output file (OPEN=OUTPUT/OUTIN), VSEQ=1 must be specified.

If “pathname” is an input file, VSEQ=number designates the file section at which processing is to start.

If “pathname” is opened with OPEN EXTEND, VSEQ specifies the file section at which extension is to begin.

In conjunction with OPEN REVERSE, it is possible to process individual tapes of a file, but tape swapping is inhibited.

= (L=list-poss (255): <integer 1..255>)
Specifies the order in which the file sections are to be processed. this may be used only for input files, not for output files. For files opened with OPEN REVERSE, only one file section number may be specified and automatic tape swapping is not supported.

WORKFIL

Only as of VERSION=3, for files on SM pubsets:
Defines whether the file is created on a work file volume set or a volume set with permanent data retention. Work file volume sets are deleted at a time defined by system administration. Work files cannot be created on a Net-Storage volume. If the file is created on an SM pubset by means of non-physical allocation and WORKFIL is not specified, it is created on a volume set with permanent data retention. The specification of WORKFIL is rejected in the following cases:

- for generations
- if the file already occupies storage space
- if SPACE is specified with a non-positive primary allocation
- if the file ends up on a private disk.

= NO
The file is created on a volume set with permanent data retention.

= YES
The file is created on a work file volume set. This specification is not permitted for temporary files.

WRCHK

For the processing of disk files:
specifies whether a read-after-write check is to be executed. “WRCHK” is not placed in the catalog entry and must therefore be repeated each time before the file is opened or processed.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

The read-after-write check serves to detect recording errors (-> error recovery measures). If the error cannot be recovered, control is passed to the EXLST exit ERRADR. Due to the additional disk revolutions required, the read-after-write function has a decidedly negative effect on system performance.

= NO
No read-after-write check is executed.

= YES
A read-after-write check is executed.

WROUT

For ISAM processing:
WROUT controls how often updated blocks are written back to disk. For shared-update processing or in cross-task ISAM pools and in task-local ISAM pools, WROUT=YES is set implicitly: updated blocks are written back to disk immediately.

If no value is specified in the FILE or FCB macro, the default value of the FCB takes effect on opening the file.

= NO
An updated block is written back to disk only when the contents of the relevant buffer area need to be overwritten or, at the latest, when the file is closed.

= YES
Each updated block is written back to disk immediately, thus always ensuring the consistency of the data on the disk and in virtual memory. However, this also increases the I/O rate.

Programming notes

Structure of the input area if VERSION=0 or VERSION=1 is specified

The operand list of the FILE macro consists of several fixed and variable areas:

1	`Fixed area 1` (a DSECT can be created by means of the IDPFL macro, see below)
2	`Variable area for VOLUME or TVSN specifications` (if the VOLUME or TVSN operand is specified in list form) : :
3	`Variable area for MOUNT specifications` (if the MOUNT operand is specified in list form) : :
4	`Fixed area 2 (FILE extension)` (a DSECT can be created by means of the IDPFX macro, see below)
5	`Variable area for DVOLUME specifications` (if the DVOLUME operand is specified in list form) : :
6	`Variable area for VSEQ specifications` (if the VSEQ operand is specified in list form) : :

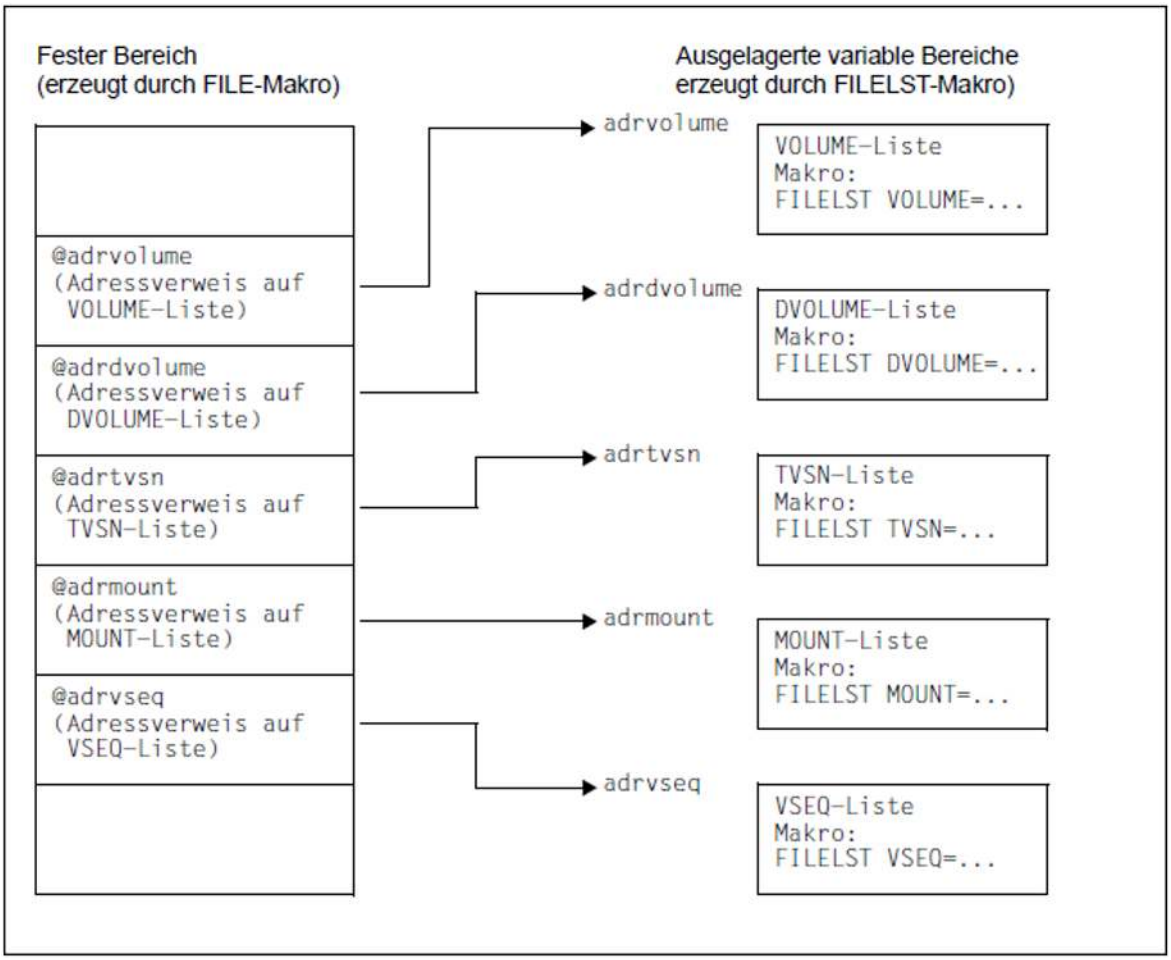
For the two fixed areas of the operand list, two DSECTs are generated by means of MF=D in the FILE macro. The IDPFL macro generates a DSECT for fixed area 1 and IDPFX generates a DSECT for the FILE extension. However, the macros IDPFL and IDPFX support only the old (pre-V9.5) macro format.

Structure of the operand list as of VERSION=2

This format supports all operands and operand values which were introduced after BS2000 V10.0.

If VERSION=2 and higher applies instead of VERSION=0/1, the user can move the variable parts for the VOLUME, TVSN, MOUNT, DVOLUME and VSEQ specifications to separate areas outside the operand list by means of creating a pointer in each operand (specification “@addr”) to a symbolic address “addr” within the program at which he/she has created a list with the corresponding operand values by means of the FILELST macro.

In this case, the operand list created by the FILE macro is an area of fixed length containing solely address pointers to the externally stored variable lists. This list has the following structure:



DSECTs for the fixed area and the externally stored variable areas can be generated using the D form of the FILE and FILEST macros.

Example

The program TAPEFIL reads the tape file TAPE.FILE via the file link name INTAPE.
The list of required volumes is specified by way of the TVSN operand in the FILE macro:

```
TAPEFIL  START
         .
         .
         FILE  TAPE.FILE,LINK=INTAPE,TVSN=@TVSNLIST,VERSION=3,MF=L ———  (1)
         .
         .
TVSNLIST FILELST TVSN=(VOL003,VOL009,VOL017) ————————————————————  (2)
         .
         .
         END
```

(1) The specification @TVSNLIST in the TVSN operand generates in the operand list of the FILE macro an address pointer to the symbolic address TVSNLIST. At this address the FILE macro expects the (variable) area with the list of TVSN values.

(2) The FILELST macro generates at the address TVSNLIST a list containing the values for the TVSN operand in the FILE macro.

Notes on the processing of tape files

Operand STATE=FOREIGN

A FOREIGN indicator is set in the catalog entry, thus making it impossible to change the file attributes by means of a CATAL macro. This FOREIGN indicator is reset when the file is opened.
If sequential file generations of a group belong to the same MF/MV set, DISP=REUSE should never be used in the CATAL macro, since this can lead to the destruction of file generations.
The method for importing foreign files is not the same as that used for private disk files. The reason for this is that the catalog entry for a foreign disk file is unique. For foreign tape files, this uniqueness could be achieved if the user IDs of the file owners already exist in the system into which the file is to be imported. However, if these user IDs do not exist, it is not possible to change the owner identifier on the tape (a hardware restriction would cause the file to be destroyed). Even if the system administrator imports a file for an existing user ID, it cannot be guaranteed that the catalog entry will be unique, since he can also catalog the file under another user ID.
Nevertheless, by virtue of the restrictions in the CATAL macro, tape files with standard labels enjoy the same protection as disk files against conflicts between the file attributes specified in the labels and those in the catalog entry. The only risk factor is that the file owner may change the file attributes by specifying SECLEV=LOW in the FCB. For this reason, there should never be more than one catalog entry in the **same** system for one file, even if the owner of the file is also working in this system.

Return codes

As of version 3, the error code is returned in the parameter list standard header and no longer in general purpose register 15 as in version 2. The standard header must not reside in the read-only area, otherwise the program will be terminated.

Standard header: `ccbbaaaa`

The following code relating to execution of the CATAL macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	No error
	X'40'	X'0501'	Requested catalog not available
	X'82'	X'0502'	Requested catalog in the rest state
	X'40'	X'0503'	Incorrect information in the MRSCAT
	X'82'	X'0504'	Catalog access error
	X'40'	X'0505'	Computer communication error (MRS)
	X'80'	X'0506'	Operation cancelled because of master change

	X'40'	X'0510'	Error while calling an internal function
	X'40'	X'0511'	No allocation because of MVDF inconsistency
	X'40'	X'0512'	Catalog ID not entered in the MRSCAT
	X'40'	X'0515'	Call rejected by the system exit routine
	X'40'	X'051B'	User ID not known in specified pubset
	X'40'	X'051C'	No access right to specified pubset
	X'40'	X'051D'	LOGON password different on specified pubset
X'02'	X'00'	X'051E'	Only partial allocation because of MVDF inconsistency
	X'20'	X'0531'	Unexpected catalog access error
	X'82'	X'0532'	File locked because it is in use
	X'40'	X'0533'	File not found
	X'82'	X'0534'	Private volume cannot be allocated
	X'40'	X'0535'	No access right to the file catalog entry
	X'20'	X'0536'	Error in file management system
	X'40'	X'053A'	Error while updating the F1 label on a private disk
	X'20'	X'053B'	System error during catalog access
	X'82'	X'053C'	Catalog file of the pubset is full
	X'40'	X'053D'	Catalog or F1 label block is destroyed
	X'40'	X'053E'	File on private volume already cataloged
	X'82'	X'053F'	File reserved by another task
	X'40'	X'0540'	Pubset contains no appropriate volume set
	X'82'	X'0541'	No disk storage space assigned
	X'82'	X'0542'	Device not available / disk exclusive
	X'20'	X'0543'	Faulty allocator parameter area
	X'20'	X'0544'	Incorrectly formatted catalog entry
	X'40'	X'0545'	Public volume not connected
X'02'	X'00'	X'0546'	File catalog entry full
	X'40'	X'0547'	Volume cannot be mounted
	X'82'	X'0548'	Not enough storage space

	X'20'	X'0549'	System error with REQM or AQIR call
X'02'	X'00'	X'054A'	Storage space only partially allocated
	X'40'	X'054B'	No volume set available for specified catalog ID
	X'82'	X'054D'	Storage space contingent exhausted
	X'82'	X'0550'	File opened and therefore locked
	X'01'	X'0551'	VSN for tape file specified more than once
	X'01'	X'0553'	Illegal absolute storage space request
	X'01'	X'0554'	Illegal file name format
	X'40'	X'0555'	STATE=FOREIGN: file already cataloged
	X'01'	X'0556'	STATE=FOREIGN: device type invalid or missing
	X'40'	X'0557'	Incorrect VSN specification
	X'01'	X'0558'	Public VSN illegal
	X'01'	X'0559'	Illegal specification with MOUNT
	X'82'	X'055A'	Tape device currently reserved
	X'40'	X'055C'	F1 label missing
	X'40'	X'055D'	User has no physical allocation right
	X'40'	X'055E'	Foreign user ID for non-cataloged file
	X'40'	X'055F'	Volume could not be reserved
	X'01'	X'0576'	Incorrect operand combination or undeleted UNUSED fields
	X'20'	X'0578'	Internal error while checking access rights
	X'01'	X'0579'	Invalid operand for temporary or work file
	X'40'	X'057A'	Storage class incompatible with file attributes
	X'40'	X'057B'	Illegal operand for migrated file
	X'40'	X'057C'	HSMS has rejected recall
	X'40'	X'057E'	HSMS not available
	X'01'	X'0590'	Device type specification missing for private volume
	X'01'	X'0592'	Private disk file has rejected catalog entry without extents or device type definition for public disk rejected
	X'01'	X'0593'	Absolute allocation: illegal number of half-pages

X'82'	X'0594'	Insufficient virtual memory available
X'01'	X'0595'	Illegal mix of public and private VSNs
X'01'	X'0596'	Device type specification not according to catalog entry
X'01'	X'0597'	Absolute allocation: first half-page not on unit border
X'01'	X'0599'	Operand not supported in the remote version
X'01'	X'05A3'	Incorrect SPACE entry
X'01'	X'05A4'	Incorrect use of DSPACE/DVOLUME/DDEVICE
X'01'	X'05A8'	Device type not in system
X'82'	X'05B0'	No suitable tape device available
X'82'	X'05B1'	A file lock is in effect for the file
X'40'	X'05B4'	Volume request was rejected
X'40'	X'05BD'	Illegal combination of file and volume set attributes
X'01'	X'05C2'	Chain name = X'0000000000000000'
X'82'	X'05C3'	File generation to be deleted is locked
X'40'	X'05C4'	An error occurred during operator logon
X'20'	X'05C7'	Internal error in DMS
X'82'	X'05C8'	CE limit for user ID exceeded
X'20'	X'05CA'	Internal error while modifying CE limits
X'40'	X'05D8'	File protected with password
X'40'	X'05DA'	Storage space release on foreign user ID
X'01'	X'05DF'	Illegal specification for BLIM / CHKPT
X'20'	X'05E0'	File locked because of storage management system error
X'01'	X'05E8'	File name invalid for disk file
X'01'	X'05EE'	File name too long after completion
X'01'	X'05EF'	File protection using GUARD only possible for public files
X'01'	X'05FA'	Pubset not locally accessible
X'40'	X'05FC'	User ID not registered
X'40'	X'05FD'	File protected via release date or access type
X'40'	X'0606'	Volume request rejected by MAREN

X'40'	X'0609'	Storage space release not permitted for system file
X'40'	X'060D'	Incorrect name specified for reference file
X'40'	X'060E'	Reference file not found or not accessible
X'40'	X'0613'	Incorrect specification of a storage class
X'40'	X'0640'	Access to Net-Storage is rejected by the ONETSTOR subsystem because of communication problems with the net client
X'40'	X'0641'	File already exists on Net-Storage
X'40'	X'0642'	Large files are not permitted on the specified pubset
X'40'	X'0643'	Net client reports access error
X'40'	X'0644'	Net client reports internal error
X'40'	X'0645'	File does not exist on Net-Storage
X'40'	X'0647'	Specified file does not match the file’s catalog entry
X'40'	X'0648'	Specification of the file type, device and volume are not compatible
X'40'	X'0649'	Net server reports POSIX ACL error
X'40'	X'064A'	Net client reports that access to files on the Net-Storage volume is forbidden
X'40'	X'064B'	Access to node files from the net client not supported
X'40'	X'0652'	Absolute storage space request not permitted on Net-Storage
X'40'	X'0653'	Net-Storage volume of type NETVOL cannot be imported
X'40'	X'0666'	File is protected against requested access
X'40'	X'0683'	File already exists
X'40'	X'0689'	Operand only permitted for file without storage
X'40'	X'06B5'	File is not correctly closed
X'01'	X'06C7'	Invalid generation number
X'01'	X'06C8'	Attribute illegal for file generations
X'40'	X'06CD'	FGG protected against extension
X'01'	X'06CF'	Illegal specification of an FGG
X'40'	X'06D0'	STATE=FOREIGN for non-existent file generation
X'40'	X'06D1'	FGG index locked by another task
X'01'	X'06DA'	Illegal public/private mix for FGG

	X'01'	X'06DF'	Illegal specification for FSEQ/VSEQ/TSET
	X'01'	X'06FD'	Illegal parameter range address
	X'40'	X'06FF'	BCAM connection severed
	X'01'	X'FFFF'	Incorrect function number in standard header
	X'03'	X'FFFF'	Incorrect version number in standard header

Variations in VERSION=0/1/2/3

Operand	Vers=0	Vers=1	Vers=2	Vers=3	Remarks for operand values
MF=E	x	x	x	x	
VERSION	x	x	x	x	
MF=C	-	x	x	x	
PREFIX	-	x	x	x	
VERSION	-	x	x	x	
MF=D	-	x	x	x	
PREFIX	-	x	x	x	
VERSION	-	x	x	x	
MF=L	x	x	x	x	
*DUMMY	x	x	x	x	
pathname	x	x	x	x	
AVAIL	-	-	-	x	
BLIM	x	x	x	x	
BLKCTRL	-	x	x	x	DATA2K and DATA4K not possible for Vers=1 *BY-PROG only as of Vers=3
BLKSIZE	x	x	x	x	*BY-PROG only as of Vers=3
BUFOFF	x	x	x	x	*BY-PROG only as of Vers=3
BYPASS	x	x	x	x	
CHAINIO	x	x	x	x	
CHKPT	x	x	x	x	
CLOSE	-	-	x	x	KEEP-DATA-IN-CACHE only as of Vers=3
CLOSMSG	-	x	x	x	
CODE	x	x	x	x	*BY-PROG only as of Vers=3 ISO7D only as of Vers=3
DATATTR	-	-	-	x	
DDEVICE	x	x	x	x	
DESTOC	-	x	x	x	

DEVICE	x	x	x	x	
DISKWR	-	-	-	x	
DSPACE	x	x	x	x	

DUPEKY	x	x	x	x	
DVOLUME	x	x	x	x	@adr only as of Vers=2
MF=L (cont.)					
EXC32GB	-	-	-	x	
FCBTYPE	x	x	x	x	*BY-PROG only as of Vers=3
FSEQ	x	x	x	x	
IOPERF	-	-	x	x	
IOUSAGE	-	-	x	x	
KEYLEN	x	x	x	x	*BY-PROG only as of Vers=3
KEYPOS	x	x	x	x	*BY-PROG only as of Vers=3
LABEL	x	x	x	x	*BY-PROG only as of Vers=3
LINK	x	x	x	x	
LOCKENV	-	-	-	x	
LOGLEN	x	x	x	x	*BY-PROG only as of Vers=3
MOUNT	x	x	x	x	@adr not possible for Vers=0 and Vers=1
NFTYPE	-	-	-	x	
OPEN	x	x	x	x	
OVERLAP	x	x	x	x	
PAD	x	x	x	x	
POOLLNK	-	x	x	x	
POOLSIZ	-	-	-	x	
RECFORM	x	x	x	x	*BY-PROG only as of Vers=3
RECSIZE	x	x	x	x	*BY-PROG only as of Vers=3
RETPD	x	x	x	x	
SECLEV	x	x	x	x	
SHARUPD	x	x	x	x	WEAK not possible for Vers=0 and Vers=1
SPACE	x	x	x	x	*KEEP not possible for Vers=0 and Vers=1
STATE	x	x	x	x	
STOCLAS	-	-	-	x	

STREAM	x	x	x	x	
TAPEWR	-	x	x	x	
TPMARK	x	x	x	x	

TRANS	x	x	x	x	
TSET	x	x	x	x	
TVSN	x	x	x	x	@adr not possible for Vers=0 and Vers=1
VALLEN	x	x	x	x	*BY-PROG only as of Vers=3
MF=L (cont.)					
VALPROP	x	x	x	x	*BY-PROG only as of Vers=3
VOLSET	-	-	-	x	
VOLUME	x	x	x		@adr only as of Vers=2 REMOVE-UNUSED only as of Vers=3
VSEQ	x	x	x	x	@adr only as of Vers=2
WORKFIL	-	-	-	x	
WRCHK	x	x	x	x	
WROUT	x	x	x	x	

Key

x Operand available in the macro version

\- Operand not available in the macro version

Vers Version

In the above table, the positional operands are arranged before the keyword operands under MF=L.

FILELST - Create variable operand areas for FILE macros

Macro type: type S (L form/D form/C form); see "Macro types"

The FILELST macro creates separate lists for the operands VOLUME, DVOLUME, TVSN, MOUNT and VSEQ of the FILE macro. These lists can then be addressed by specifying the value @addr for these operands in the FILE macro, “addr” being the symbolic address of the FILELST call.

Format

Format 1: L form

Operation	Operands
FILELST	**[MF = L]** **[VOLUME = vsn / (vsn,...)]** **[DVOLUME = vsn / (vsn,...)]** **[TVSN = vsn / (vsn,...)]** **[MOUNT = number / (number,...)]** **[VSEC = number / (number,...)]**

Operand descriptions

VOLUME

A single volume serial number or a list of up to 255 volume serial numbers may be specified.The description of the VOLUME operand in the FILE macro also applies here (see "FILE - Define file attributes / control file processing"), with the restriction that VOLUME=PRIVATE or VOLUME=(PRIVATE,n) may be specified only in the FILE macro.

DVOLUME

A single volume serial number or a list of up to 255 volume serial numbers may be specified.The description of the DVOLUME operand in the FILE macro also applies here (see "FILE - Define file attributes / control file processing"), with the restriction that DVOLUME=PRIVATE or DVOLUME=(PRIVATE,n) may be specified only in the FILE macro.

MOUNT

Up to 255 volumes may be requested.

The description of the MOUNT operand in the FILE macro also applies here (see "FILE - Define file attributes / control file processing"). However, MOUNT=0 may be specified only in the FILE macro.

TVSN

A single volume serial number or a list of up to 255 volume serial numbers may be specified.The description of the TVSN operand in the FILE macro also applies here (see "FILE - Define file attributes / control file processing").

VSEQ

Up to 255 file sections may be requested.

The description of the VSEQ operand in the FILE macro also applies here (see "FILE - Define file attributes / control file processing"). However, only lists may be specified here; single file sections may be specified only in the FILE macro.

Format 2: D form/C form

Operation	Operands
FILELST	**MF = D / C** **[,PREFIX = I / pre]** **[,MACID = DBL / macid]** **[,LIST = VOLUME / (VOLUME,nmbr) /** **DVOLUME / (DVOLUME,nmbr) /** **TVSN / (TVSN,nmbr) /** **MOUNT / (MOUNT,nmbr) /** **VSEQ / (VSEQ,nmbr)]**

Operand descriptions

LIST

This operand defines the lists created by FILELST for which a CSECT or DSECT is to be generated.

= VOLUME
= (VOLUME,nmbr)
A CSECT or DSECT is created for the VOLUME list. If the user specifies the operand value in the form (VOLUME,nmbr) – the parentheses are part of the value and must be specified -, “nmbr” can be used to specify the number of elements in the VOLUME list for which a CSECT or DSECT is to be created.

= DVOLUME
= (DVOLUME,nmbr)
A CSECT or DSECT is created for the DVOLUME list. If the user specifies the operand value in the form (DVOLUME,nmbr) – the parentheses are part of the value and must be specified -, “nmbr” can be used to specify the number of elements in the DVOLUME list for which a CSECT or DSECT is to be created.

= TVSN
= (TVSN,nmbr)
A CSECT or DSECT is created for the TVSN list. If the user specifies the operand value in the form (TVSN, nmbr) – the parentheses are part of the value and must be specified –, “nmbr” can be used to specify the number of elements in the TVSN list for which a CSECT or DSECT is to be created.

= MOUNT
= (MOUNT,nmbr)
A CSECT or DSECT is created for the MOUNT list. If the user specifies the operand value in the form (MOUNT,nmbr) – the parentheses are part of the value and must be specified -, “nmbr” can be used to specify the number of elements in the MOUNT list for which a CSECT or DSECT is to be created.

= VSEQ
= (VSEQ,nmbr)
A CSECT or DSECT is created for the VSEQ list. If the user specifies the operand value in the form (VSEQ, nmbr) – the parentheses are part of the value and must be specified –, “nmbr” can be used to specify the number of elements in the VSEQ list for which a CSECT or DSECT is to be created.

PREFIX

Defines the first character of each field name and equate generated when the macro is expanded.

Default value: PREFIX = I

= pre
Single-character prefix with which the generated field names and equates are to begin.

MACID

Defines the second through fourth characters of the field names and equates generated when the macro is expanded.

Default value: MACID = DBL

= macid
Three-character string defining the second through fourth characters of the generated field names and equates.

FPAMACC - Access FASTPAM files

Macro type: type S (E form/L form/D form/C form/M form); see "Macro types"

The ACCESS FILE function is implemented in the FPAMACC macro. Accesses to the file referenced by OPENID can be formulated in this macro.

Format

Operation	Operands
FPAMACC	**[,OPENID = nmbr / adr(r)]** **[,LEN = length / adr / (r)]** **[,BLOCK = nmbr / adr / (r)]** **[,IOAREA = adr / (r)]** **[,OPCODE = *READ / *WRITE / *READ_WAIT / *WRITE_WAIT /** ***READ_EQUALIZE / *WAIT / adr / (r)]** **[,WAITLST = adr / (r)]** **[,CHAIN = adr / (r)]** **[,POSTCD = nmbr / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = F / pre]
	MF = C / M **[,PREFIX = F / pre]** **[,MACID = ACC / macid]**

Operand descriptions

BLOCK

Specifies the number of the first logical FASTPAM block (within the file) to be transferred.The block size is determined by the BLKSIZE operand in the OPEN function of the FPAMSRV macro. Only integer values are permitted.

Only a direct specification is allowed for the MF=L form.

= nmbr
Direct entry of a decimal numeric value for the number of the first logical block to be transferred. The value is limited to the maximum size of a file in 4-KB pages minus 1:

1 <= nmbr <= 8388606 for LARGE_FILE=*FORBIDDEN (see the FPAMSRV macro)

1 <= nmbr <= 1073741823 for LARGE_FILE=*ALLOWED (see the FPAMSRV macro)

= addr
Symbolic address of a 4-byte field containing the numeric value (binary).

= (r)
Register containing the numeric value.

CHAIN

Specifies the starting address of another FPAMACC parameter list for which the associated job is to be concatenated to that of the current parameter list. Up to 5000 parameter lists can be chained with each other in this way. All jobs chained in this manner are accepted and executed within an SVC.
The jobs are executed in the order of their concatenation. If a parameter error is encountered in one of the jobs, all other jobs will be rejected with a “CHAIN ERROR” return code. For all other errors, no further job will be processed as of the point at which the error is detected (except for return code FACCPNAC), and all jobs which follow will also be rejected. If asynchronous I/O operations with concatenated “wait parameter lists” are involved, wait operations that have been rejected due to an error must therefore be repeated by the user.
If an error is encountered only after all jobs have been successfully initiated (e.g. an I/O error in a chain of asynchronous read/write operations with eventing), all other jobs will be treated as separate from the job containing the error. Each parameter list must therefore be evaluated independently.
See also the examples on "Error handling for chained parameter lists".

Only the symbolic address is allowed for the MF=L form, but no symbolic names may be used within a DSECT, since its address is not known until runtime.

Note

If eventing is enabled, a signal is sent for each individual job.

= addr
Symbolic address (name) of the next parameter list to be processed.

= (r)
Register containing the starting address of the next parameter list to be processed.

IOAREA

Specifies the 4K-aligned starting address of the I/O buffer, which must lie within the I/O area pool that was specified when calling OPEN.
In the case of data spaces, the ALET indicated at the time of calling ENABLE IOAREA POOL is automatically used.

Only the symbolic address is allowed for the MF=L form, but no symbolic names may be used within a DSECT, since its address is not known until runtime.

= addr
Symbolic address (name) of the area.

= (r)
Register containing the starting address of the I/O buffer.

LEN

Determines the length of data to be transferred in logical blocks. The block size is determined by the BLKSIZE operand of the OPEN function in the FPAMSRV macro. Only integer values between 1 and 8 which do not exceed the MAXIOLN value specified in ENABLE ENVIRONMENT are permitted.

Only a direct specification is allowed for the MF=L form.

= length
Direct entry of a decimal numeric value.

= addr
Symbolic address of a 4-byte field containing the numeric value (binary).

= (r)
Register containing the numeric value.

MACID

Defines the second to the fourth characters (inclusive) of the field names and equates that are generated when macros are resolved.

= ACC
Default value: MACID=ACC

= macid
“macid” is three-character string that defines the second to the fourth character (inclusive) of the generated field names and equates.

MF

The forms of the MF operand are described in detail in the appendix on "Macro types".

OPCODE

Identifies the job type.

Only a direct specification is allowed for the MF=L form.

= *READ
Asynchronous reading of logical blocks. A return value is placed in subcode2, indicating whether the job was executed synchronously or asynchronously. If EVENTNG=*NO was specified in the call to OPEN and if the job was not completed synchronously, it must be terminated with OPCODE=*WAIT; otherwise, all subsequent jobs with this FPAMACC parameter list will be rejected.

= *WRITE
Asynchronous writing of logical blocks. A return value is placed in subcode2, indicating whether the job was performed synchronously or asynchronously. If EVENTNG=*NO was specified in the call to OPEN and if the job was not completed synchronously, it must be terminated with OPCODE=*WAIT; otherwise, all subsequent jobs with this FPAMACC parameter list will be rejected.

= *READ_WAIT
Synchronous reading of logical blocks.

= *WRITE_WAIT
Synchronous writing of logical blocks.

= *READ_EQUALIZE
Synchronously reads logical blocks while simultaneously equalizing the DRV disks in the indicated section within the file. In non-DRV mode, *READ_EQUALIZE has the same effect as *READ_WAIT (for more information on DRV, see the “DRV” manual [15]).

= *WAIT
Waits for the end of an asynchronous job (*READ or *WRITE). This operation may be performed only by the task that has initiated the job.
The WAITLIST parameter is used to specify the address of the FPAMACC parameter list, on whose I/O the wait operation is to be performed. This can also be the same parameter list, but it must lie in the same environment and must contain the same OPENID as the WAIT parameter list.

A *WAIT is illegal in the following cases:

- after the synchronous operations *READ_WAIT, *WRITE_WAIT and *READ_EQUALIZE,
- if a *WAIT has already been performed,
- if the asynchronous job was terminated synchronously (subcode2=FACCSYTE after *READ/*WRITE).
- When EVENTNG=*YES was specified in the OPEN

= addr
Symbolic address of a 1-byte field containing the value for OPCODE.

= (r)
Register containing the value for OPCODE.

OPENID

Specifies the short ID of the OPEN for which the FPAMACC operation is to be executed.After successful completion of the OPEN operation, the short ID must be transferred from the FPAMSRV parameter list into the FPAMACC parameter list.

Only a direct specification is allowed for the MF=L form.

= nmbr
Direct entry of a decimal numeric value for the OPEN short ID.

= addr
Address of a 4-byte field containing the short ID.

= (r)
Register containing the short ID.

PARAM

Indicates the address of the operand list. This operand is only evaluated in conjunction with MF=E (see also "Macro types").

POSTCD

Contains data accompanying the bourse signal. This parameter is interpreted only if EVENTNG=*YES is specified.

Only a direct specification is allowed for the MF=L form.

= addr
Address of a 2-byte field containing the POSTCD.

= (r)
Register containing the POSTCD (least-significant 2 bytes).

PREFIX

Defines the first character of field names and equates that are generated when macros are resolved.

= F
Default value: PREFIX=F

= pre
“pre” is a one-character prefix with which the generated field names and equates are to begin.

WAITLST

Specifies the start address of the FPAMACC parameter list, on whose I/O the WAIT operation is to be performed. This may also be the same parameter list, but it must lie in the same environment and must contain the same OPENID.

Any error is reported in the parameter list with which the invalid operation was initiated. In the event of an I/O error, for example, the “wait parameter list” receives the return code “SUCCESSFUL_PROCESSING”; the read/write parameter list, the return code “IO_ERROR”.
WAITLIST is only evaluated in conjunction with OPCODE=*WAIT.

Only the symbolic address is allowed for the MF=L form, but no symbolic names may be used within a DSECT, since its address is not known until runtime.

= addr
Symbolic address (name) of the area.

= (r)
Register containing the starting address of the FPAMACC parameter list.

Programming notes

DRV status

The DRV status is returned when changes are made to the FACCDS field. It is available following the first I/O operation.

End of job message with EVENTING

When the eventing mechanism is used (OPEN with operand EVENTNG=*YES), FASTPAM reports the end of a job in the FACCREQ field of the FPAMACC parameter list. There are two cases:

- FACCREQ = X'00' = FACCTERM means 'job terminated'.
- FACCREQ = X'FF' = FACCACTV means 'job not yet terminated'.

The FACCREQ field is assigned the value FACCACTV (job active) when the job is accepted and the value FACCTERM at the end of the I/O operation; in the latter case, asynchronously by a system task. For this reason, the contents of the FACCREQ field must not be queried by an assembly language instruction that writes to the field contents. The end-of-job message would be lost, for example, if the field contents were queried with the following

instruction: OC FACCREQ,FACCREQ

The instruction: CLI FACCREQ,0 by contrast, would present no problem.

Important note

The FACCREQ field must not be accessed by a machine-language instruction for writing between the time the job is submitted and the end-of-job message. All returned information (return code, DRV status) must be evaluated only after the end-of-job message has been independently queried. It would be incorrect, for example, to copy the contents of the FPAMACC parameter list to a different memory location and to perform the desired actions on this copy.

The job could have been completed synchronously or asynchronously when the application program regains control. The user can obtain information on whether or not the system has sent a signal to the bourse from subcode2; however, this cannot be checked until end-of-job has been reported. In any case, this is irrelevant if the user has enabled EVENT DROPPING.

Synchronous jobs will have always completed when the application program regains control; however, the FACCREQ field is still assigned the appropriate value.

Example: End-of-job handling with eventing

```
*-------------------------------------------------------------------*
* The event ID, FASTPAM environment and the I/O area pool should    *
* have been created by now and the file should have been opened     *
* with EVENTNG=*YES.                                                *
*-------------------------------------------------------------------*
       FPAMACC MF=M,OPCODE=*READ,....
       FPAMACC MF=E,...
       CLI   FACCREQ,FACCTERM        Job already terminated?
       BE    TERM
*-------------------------------------------------------------------*
* Job is not yet terminated;                                        *
* do something else                                                 *
*-------------------------------------------------------------------*
          :
          :
*-------------------------------------------------------------------*
* Wait with SOLSIG (job not synchronously terminated!)              *
*-------------------------------------------------------------------*
         B     SOLS
TERM     CLI   FACCSR2,FACCSYTE
*-------------------------------------------------------------------*
* Job synchronously terminated                                      *
* no SOLSIG!!                                                       *
*-------------------------------------------------------------------*
         BE    PROCEED
SOLS     SOLSIG ...
PROCEED  EQU   *
          :
          :
```

Error handling for chained parameter lists

If FASTPAM detects an error when processing a job chain (possibly when checking parameters or when subsequently processing individual jobs), all following jobs in the chain are rejected by FASTPAM with the return code “CHAIN_ERROR”. The return code “FACCPNAC” (WAIT for an inactive I/O path) is not considered an error in this case, since it occurs normally in a correctly executed program run for synchronously terminated I/Os (with caching). Consequently, “FACCPNAC” does not cause the chain to be aborted, and this in turn simplifies error checking.

- When synchronous jobs are chained, only the return code of the last member in the chain needs to be checked in order to ensure that all jobs have executed successfully.
- For chains of asynchronous I/Os with attached WAIT jobs, only the return code of the last WAIT job and that of the associated I/O needs to be checked. If both are “SUCCESSFUL”, it can be assumed that all previously completed jobs have also executed successfully.
- In the case of chains of asynchronous I/Os with eventing, however, all return codes must be queried separately, since the end-of-job conditions occur independently of one another.

Return codes

Return codes are valid only after the completion of each respective job. They are placed in the (standard) header of the parameter list (see"Layout of the parameter list"):

- The main return code, in a half-word with the name FACCMRET.
- Subcode1, in one byte with the name FACCSR1.
  Subcode1 describes error classes which allow the caller to respond to them. The caller can refer back to the main code as well as to subcode1. (It is better to refer to subcode1, since the information therein does not depend on the software version.)
- Subcode2, in one byte with the name FACCSR2.
  Subcode2 describes the individual main codes more precisely. In the FPAMACC macro, subcode2 is only significant for asynchronous jobs. It indicates whether the job was terminated synchronously for each return code, even in the case of errors.

The field names and the EQU instructions for return codes which are generated by the C or D form of the macro begin with the string FPAM by default. This string can be changed by means of PREFIX and MACID.

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

If the return codes cannot be placed in the header (because it it not accessible, for example), the calling program is terminated with an error message. If the user has defined an STXIT event for an “unrecoverable program error”, this STXIT is activated.

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller
- the list is not aligned on a word boundary
- the list is write-protected.

In the following section, the main return codes are grouped with corresponding subcode1 classes and are described more precisely by subcode2.

Notes

- Error flags are listed in the corresponding system messages with the message code DFPaaaa (where aaaa=main code). Message texts can be output using the command or standard statement HELP-MSG-INFORMATION.
- All addresses passed to FASTPAM must be valid 31-bit addresses. In particular, bit 32 must not be set, otherwise it will be regarded as belonging to the address.

Standard header: `ccbbaaaa`

The following code relating to execution of the FPAMACC macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
X'00'			Synchronous termination. In this case, a following *WAIT for the I/O path is answered by return code FACCPNAC (PATH NOT ACTIVE), and no signal is sent from the system if eventing is enabled.
X'01'			Asynchronous termination. If EVENTNG is set to *NO, the job must be terminated with *WAIT; if EVENTNG=*YES, the system sends a signal to the end of job bourse.
	X'00'	X'0000'	Function executed successfully.
	X'01'		The function could not be executed, since the corresponding operand was not specified correctly.
	X'01'	X'00C8'	Function not executed. Invalid OPEN-ID
	X'01'	X'00C9'	Function not executed. Invalid address for I/O buffer.
	X'01'	X'00CA'	Function not executed. Invalid block specification.
	X'01'	X'00CB'	Function not executed. Invalid WAITLST specification.
	X'01'	X'00CE'	Function not executed. Invalid block number.
	X'01'	X'00CF'	Function not executed. Invalid operation code.
	X'02'		Function not executed. Called function not available.
	X'03'		Function not executed. Interface version not supported.
	X'20'		Internal error.
	X'20'	X'0028'	Function not executed. System error. Run system diagnostics.
	X'40'		CORRECT AND RETRY
	X'40'	X'0037'	System resource bottleneck. Response: inform the system administrator.
	X'40'	X'00C7'	Invalid CFID specified. Response: correct the file name in the program.

X'40'	X'012C'	I/O error. Response: inform the system administrator.
X'40'	X'012D'	Another I/O is active on this path.

X'40'	X'012E'	Only for OPCODE=*WAIT: There is no I/O active on this path. This can also occur with *WAIT operations that are chained with asynchronous jobs in cases when the I/O could be terminated synchronously. The job chain is not aborted for this return code.
X'40'	X'012F'	The job was not executed, since an error occurred in another job chained to it. Response: find the error in the job chain.
X'40'	X'0133'	OPCODE=*WAIT is not allowed with EVENTNG=*YES.
X'40'	X'0134'	Waiting for an I/O of another task is not permitted.
X'40'	X'0140'	I/O after end-of-file. In contrast to UPAM, not even one I/O is executed with this return code.
X'40'	X'0141'	No space could be allocated on disk when making a secondary allocation. Response: inform the system administrator.
X'40'	X'0142'	User ID overloaded. Response: delete files or inform the system administrator.
X'40'	X'0143'	PVS not attached. Response: inform the system administrator.
X'40'	X'0144'	No new files can be added to the catalog. Response: delete files or inform the system administrator.
X'40'	X'0145'	On file access in mode SHARUPD=YES, it was detected that the file size exceeded the value 32 GB even though this value may not be exceeded when OPEN is used with this file.
X'40'	X'014A'	No secondary allocation could be made due to a missing extent list. The missing extent list indicates that the system administration has issued the command REPAIR-DISK-FILES or REMOVE-FILE-ALLOCATION for the file just opened. Response: inform the system administrator.

Examples

Example 1: Detection of errors during the parameter check

In the following example, before the first job in a chain is processed, FASTPAM checks the parameters of all members in the chain and detects the parameter error “INVALID_<parameter>” in the process.

The boxes illustrated in the diagrams below represent FPAMACC parameter lists. The return codes output in each case are shown below the boxes.

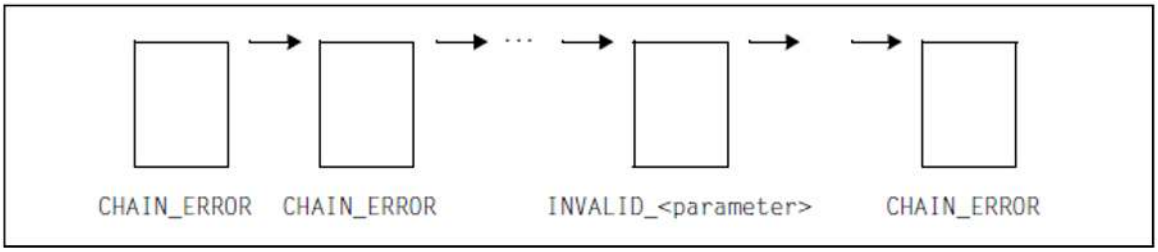


Figure 8: FPAMACC macro: errors during parameter checking

Example 2: Detection of errors after the parameter check

- Asynchronous I/Os with eventing:
  If an error occurs in a chain of asynchronous I/Os with eventing after all I/Os have been initiated (e.g. an “IO_ERROR”), each job is handled separately. A “CHAIN_ERROR” cannot occur.

  Since each I/O is on a file that is opened with “eventing”, reads and writes may be freely exchanged:

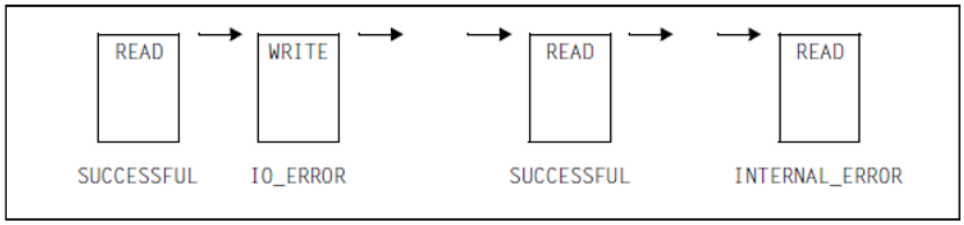


Figure 9: FPAMACC macro: errors after parameter checking (asynchronous I/O with eventing)

- Asynchronous I/Os with “WAIT”:
  If, in a chain of asynchronous I/Os with concatenated WAIT operations, an error occurs at the n-th I/O (“IO_ERROR”), the associated WAIT operation in which the error was detected will still be considered successful; however, since processing is aborted after the error, all following WAIT operations for the outputs 1 to (n-1) will be missing.

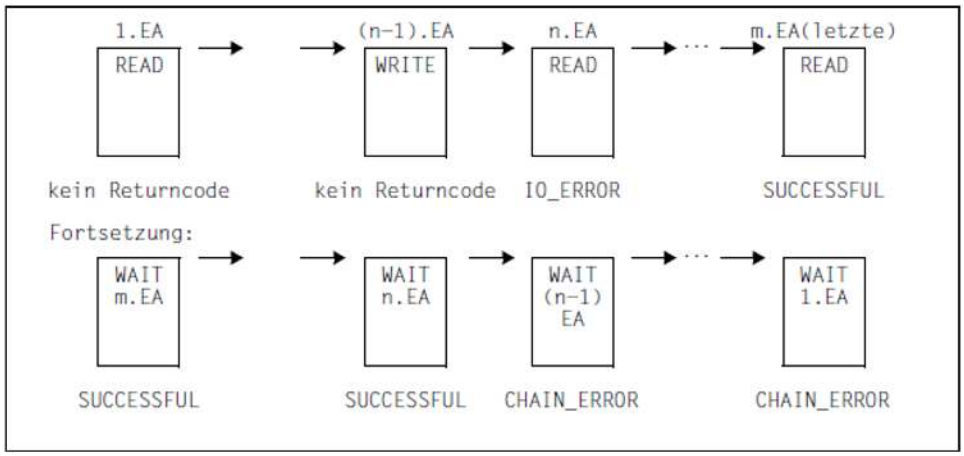


Figure 10: FPAMACC macro: errors after parameter checking (asynchronous I/O with WAIT)

Layout of the parameter list

The following parameter list is issued by an FPAMACC macro:

```
FPAMACC MF=D
1          STACK  PRINT
1          PRINT  NOGEN
2               *,##### PREFIX=F, MACID=ACC #####
```

```
1          #INTF REFTYPE=REQUEST,INTNAME=FPAMACC,INTCOMP=001
1 FACCPA   DS    0F      BEGIN of PARAMETERAREA           _INOUT
1          FHDR  MF=(C,FACC),EQUATES=YES
2          DS    0A
2 FACCFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 FACCIFID DS    0A              0   INTERFACE IDENTIFIER
2 FACCFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 FACCFCT  DS    AL1             2   FUNCTION NUMBER
2 FACCFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 FACCRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 FACCSRET DS    0AL2            4   SUB RETURN CODE
2 FACCSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 FACCR2OK EQU   X'00'               All correct, no additional info
2 FACCR2NA EQU   X'01'               Successful, no action was necessary
2 FACCR2WA EQU   X'02'               Warning, particular situation
2 FACCSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 FACCRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
2 FACCRPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 FACCRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
2 FACCRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 FACCRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 FACCRAER EQU   X'04'               ALIGNMENT ERROR
2 FACCRIER EQU   X'20'               INTERNAL ERROR
2 FACCRCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 FACCRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 FACCRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 FACCRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 FACCRWLR EQU   X'81'                    "     LONG       "
2 FACCRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 FACCRTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
2 FACCRDH  EQU   X'82'               SS IN DELETE / HOLD
```

```
2 *
2 FACCMRET DS    0AL2            6   MAIN RETURN CODE
2 FACCMR2  DS    AL1             6   MAIN RETURN CODE 2
2 FACCMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 FACCRLNK EQU   X'FFFF'             LINKAGE ERROR / REQ. NOT PROCESSED
2 FACCFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *
1 * MAINCODE
1 *
1 FACCMFSP EQU  X'0000'       SUCCESSFUL_PROCESSING             = 0
1 FACCMIER EQU  X'0028'       INTERNAL_ERROR                    = 40
1 FACCSRES EQU  X'0037'       SHORTAGE_OF_RESOURCES             = 55
1 FACCICFI EQU  X'00C7'       INVALID_CFID                      = 199
1 FACCIOPI EQU  X'00C8'       INVALID_OPEN_ID                   = 200
1 FACCIIOA EQU  X'00C9'       INVALID ADDRESS OF IOAREA         = 201
1 FACCIBLK EQU  X'00CA'       INVALID_BLOCK                     = 202
1 FACCILAW EQU  X'00CB'       INVALID LIST ADDRESS FOR WAIT     = 203
1 FACCIBL# EQU  X'00CE'       INVALID_BLOCK_#                   = 206
1 FACCIOP  EQU  X'00CF'       INVALID_OPCODE                    = 207
1 FACCIOER EQU  X'012C'       IO_ERROR = 300
1 FACCPACT EQU  X'012D'       PATH_ACTIVE                       = 301
1 FACCPNAC EQU  X'012E'       PATH_NOT_ACTIVE (WAIT ONLY)       = 302
1 FACCCHE  EQU  X'012F'       CHAIN_ERROR                       = 303
1 FACCWTEV EQU  X'0133'       WAIT_AND_EVENTING                 = 307
1 FACCWTNS EQU  X'0134'       WAIT_NOT_BY_SAME_TASK             = 308
1 FACCEOF  EQU  X'0140'       END_OF_FILE                       = 320
1 FACCNDSA EQU  X'0141'       NO_DISC_SPACE_AVAILABLE           = 321
1 FACCUIDE EQU  X'0142'       USER_ID_EXHAUSTED                 = 322
1 FACCPVNA EQU  X'0143'       PUBLIC_VOLUME_NOT_ATTACHED        = 323
1 FACCCEFL EQU  X'0144'       CATALOG_ENTRY_FULL                = 324
1 FACCLFNS EQU  X'0145'       LARGE_FILE_NOT_SPECIFIED          = 325
1 FACCSAVY EQU  X'014A'       SYSTEM_ADMINISTRATOR_VERIFY       = 330
1 *
1 * SUB RETURN CODE2
1 *
1 FACCSYTE EQU  X'00'         SYNCHRONEOUS TERMINATION
1 FACCASTE EQU  X'01'         ASYNCHRONEOUS TERMINATION
1 *
1 * FPAMACC FUNCTIONS:
1 *
1 FACCACCF EQU  7             ACCESS FILE
1 *
1 * OUTPUT PARAMETER
1 *
1         DS   XL2           RESERVED
1 FACCDS  DS   X             DRV STATUS
1 FACCREQ  DS   X             REQUEST STATUS
1 FACCTERM EQU  X'00'         REQUEST TERMINATED
1 FACCACTV EQU  X'FF'         REQUEST ACTIVE
1 *
1 * INPUT PARAMETER
1 *
1 FACCOPID DS   F             OPEN-ID
1 FACCIOA  DS   A             ADDRESS OF IOAREA
1 FACCBLK  DS   F             BLOCK WITHIN FILE
```

```
1 FACCLAW  DS   A             LIST ADDRESS FOR WAIT OPERATION
1 FACCCHLA DS   A             ADDRESS OF CHAINED LIST
1 FACCPOCO DS   FL2           POSTCODE
1 FACCBLK# DS   FL1           BLOCK NUMBER
1 FACCOP   DS   AL1           OPCODE
1 FACCREAD EQU  1             READ
1 FACCWRIT EQU  2             WRITE
1 FACCRDWT EQU  3             READ AND WAIT
1 FACCWRWT EQU  4             WRITE AND WAIT
1 FACCRDEQ EQU  5             READ AND EQUALIZE
1 FACCWAIT EQU  6             WAIT
1          DS   0F
1 FACC#    EQU   *-FACCPA LENGTH of PARAMETERAREA
```

Example

Creating a file with FASTPAM

```
FPAMTEST START
         BALR  10,0
         USING *,10
         USING FPAMD,9
         LA    9,FPAMPL               R9 -> FPAMSRV-Parameterliste
*-------------------------------------------------------------------*
* Speicher für die ACCESS-Parameterlisten                           *
*-------------------------------------------------------------------*
REQM
         LTR   15,15
         BNZ   ERROR
         LR    8,1                   R8 -> FPAMACC-Parameterliste
*-------------------------------------------------------------------*
* Speicher für den IOAREA-POOL                                      *
*-------------------------------------------------------------------*
REQM  30
         LTR   15,15
         BNZ   ERROR
         LR    7,1                   R7 -> IOAREA-POOL
*-------------------------------------------------------------------*
* ENABLE ENVIRONMENT                                                *
*-------------------------------------------------------------------*
FPAMSRV MF=M,PARAM=FPAMPL,ACCLSTS=(8)
         FPAMSRV MF=E,PARAM=FPAMPL
         CLI   FPAMSR1,FPAMRFSP
         BE    ENAIPO
         CLI   FPAMSR1,FPAMRCAR
         BNE   ERROR
         CLC   FPAMMRET,=Y(FPAMNORE)
         BNE   ERROR
*-------------------------------------------------------------------*
* Behandlung des Fehlers 'RESIDENT SPACE NOT AVAILABLE'             *
* evt. nur Meldung ausgeben und weitermachen                        *
*-------------------------------------------------------------------*
*              .
*              .
*              .
*-------------------------------------------------------------------*
* ENABLE IOAREA-POOL                                                *
```

```
*--------------------------------------------------------------------*
ENAIPO   FPAMSRV MF=M,PARAM=FPAMPL,FCT=*ENAIPO,IPONAME='IOAREA',       -
               IPOADDR=((7),0),IPOSIZE=30
         FPAMSRV MF=E,PARAM=FPAMPL
         CLI   FPAMSR1,FPAMRFSP
         BE    OPEN
         CLI   FPAMSR1,FPAMRCAR
         BNE   ERROR
         CLC   FPAMMRET,=Y(FPAMNORE)
         BNE   ERROR
*--------------------------------------------------------------------*
* Behandlung des Fehlers 'RESIDENT SPACE NOT AVAILABLE'              *
* evt. nur Meldung ausgeben und weitermachen
*--------------------------------------------------------------------*
*              .
*              .
*              .
*--------------------------------------------------------------------*
* Eröffnen der Datei mit OUTIN                                       *
*--------------------------------------------------------------------*
OPEN     FPAMSRV MF=M,PARAM=FPAMPL,FCT=*OPEN,FILE='TESTFILE',         -
               MODE=*OUTIN,SHARUPD=*YES,BLKSIZE=1
         FPAMSRV MF=E,PARAM=FPAMPL
         CLI   FPAMSR1,FPAMRFSP
         BNE   ERROR
*--------------------------------------------------------------------*
* Schreibe nummerierte Blöcke in die Datei                           *
*--------------------------------------------------------------------*
LA    6,1
         LR    4,8                  R4 -> 1. Parameterliste
         USING ACCESSD,4
         LR    3,8                  R3 -> 1. Parameterliste
         LA    2,30                  Schleifenzähler
CYCL1    DS    0F
         ST    6,0(7)
         MVC   0(FACC#,4),FACCPL
         C     2,=A(1)
         BNE   NOTLAST
* IN DER LETZTEN FPAMACC-PARAMETERLISTE KEINE KETTUNG MEHR
         L     3,FFFFFFFF
         B     NEXT
NOTLAST  EQU   *
         A     3,=A(FACC#)            R3 -> nächste Parameterliste
NEXT     EQU   *
         FPAMACC MF=M,PARAM=(4),OPENID=FPAMOPID,BLOCK=(6),            -
               IOAREA=(7),CHAIN=(3)
         A     6,=A(1)
         LR    4,3                  R4 -> nächste Parameterliste
         BCT   2,CYCL1
         FPAMSRV MF=E,PARAM=(8)
* Fehlerauswertung der Parameterlisten                               *
         LR    3,8                  R3 -> 1. Parameterliste
         USING ACCESSD,3
         LA    2,30                  Schleifenzähler
CYCL2    DS    0F
         CLI   FACCSR1,FACCRFSP
         BNE   ERROR
         A     3,=A(FACC#)            R3 -> nächste Parameterliste
         BCT   2,CYCL2
```

```
*
*-------------------------------------------------------------------*
* Schließen der Datei                                               *
*-------------------------------------------------------------------*
FPAMSRV MF=M,PARAM=FPAMPL,FCT=*CLOSE
         FPAMSRV MF=E,PARAM=FPAMPL
         CLI   FPAMSR1,FPAMRFSP
         BNE   ERROR
*-------------------------------------------------------------------*
* DISABLE IOAREA-POOL                                               *
*-------------------------------------------------------------------*
FPAMSRV MF=M,PARAM=FPAMPL,FCT=*DISIPO
         FPAMSRV MF=E,PARAM=FPAMPL
         CLI   FPAMSR1,FPAMRFSP
         BNE   ERROR
*-------------------------------------------------------------------*
* DISABLE ENVIRONMENT                                               *
*-------------------------------------------------------------------*
FPAMSRV MF=M,PARAM=FPAMPL,FCT=*DISENV
         FPAMSRV MF=E,PARAM=FPAMPL
         CLI   FPAMSR1,FPAMRFSP
         BNE   ERROR
*-------------------------------------------------------------------*
* Speicherfreigabe für den IOAREA-POOL                              *
*-------------------------------------------------------------------*
RELM  30,(7)
*-------------------------------------------------------------------*
* Speicherfreigabe für die ACCESS-Parameterlisten                   *
*-------------------------------------------------------------------*
RELM  1,(8)
*
ERROR    DS   0Y
         TERM
*
FPAMPL   FPAMSRV  MF=L,FCT=*ENAENV,ENVNAME='TESTENV',ACCNUMB=30,      -
               MAXIOLN=*MINI,EVENTNG=*NO
FPAMD    FPAMSRV  MF=D
FACCPL   FPAMACC MF=L,LEN=1,                                          -
               OPCODE=*WRITE_WAIT
FFFFFFFF DC    X'FFFFFFFF'
*
         END
```

FPAMSRV - FASTPAM management function

Macro type: type S (E form/L form/D form/C form/M form); see "Macro types"

General

This description begins with an overview of the complete format of the FPAMSRV macro with all possible operands. Regardless of which function is specified (in the FCT operand), **all** operands may be entered in an FPAMSRV macro; the operands to be evaluated are determined by the current FPAMSRV function. The individual functions of the FPAMSRV macro are listed in brief after the format overview.
The format for each function and the operands which are evaluated for it are described separately in each function unit.
Operand values that are not addresses or registers are identified in the operand descriptions as “direct specifications”.
The “direct specifications” are always listed in the operand descriptions if they are theoretically possible in the format, even if the user could not possibly know their value when programming (e.g. the value of an ID assigned by the system).

The various forms of the MF operand are described in detail in the appendix ("Macro types").

Parameter list

The parameter list of the macro contains a header, whose fields are loaded automatically when the list is created with the L form.

If a parameter list is to be created dynamically with the D or C form, it must be initialized beforehand with a parameter list created with the L form. This is the only way of ensuring that the header of a parameter list contains the correct information.

Format

Operation	Operands

FPAMSRV	**[,FCT = *ENAENV / *ENAIPO / *OPEN / *CLOSE / *DISIPO / *DISENV / adr / (r)]** **[,ENVNAME = 'name' / adr / (r)]** **[,IPONAME = 'name' / adr / (r)]** **[,IPOADDR = (elem1,elem2)]** **[,IPOSIZE = size / adr / (r)]** **[,ENVID = nmbr / adr / (r)]** **[,IPOID = nmbr / adr / (r)]** **[,OPENID = nmbr / adr / (r)]** **[,LINK = 'name' / adr / (r)]** **[,FILE = 'pathname' / adr / (r)]** **[,FILE = 'pathname' / adr / (r)]** **[,LASTBLK = nmbr / adr / (r)]** **[,ACCNUMB = number / adr / (r)]** **[,SHARUPD = *NO / *YES / adr / (r)]** **[,MODE = *INPUT / *INOUT / *OUTIN / adr / (r)]** **[,MAXIOLN = *NOT_SPECIFIED / *MINI / *MAXI / adr / (r)]** **[,EVENTNG = *NOT_SPECIFIED / *NO / *YES / adr / (r)]** **[,EIID = nmbr / adr / (r)]** **[,BLKSIZE = size / adr / (r)]** **[,RES = *NOT_SPECIFIED / *NO / *YE / adr / (r)]** **[,ENV = *HOST / *XCS / adr / (r)]** **[,LARGE_FILE = *FORBIDDEN / *ALLOWED / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = F / pre]
	MF = C / M **[,PREFIX = F / pre]** **[,MACID = PAM / macid]**

Functions

Function	Brief description	See

FCT = *ENAENV	Create a FASTPAM environment or connect the user to an existing environment	"FASTPAM function: ENABLE ENVIRONMENT"
FCT = *ENAIPO	Create a FASTPAM I/O area pool or connect the user to an existing pool	"FASTPAM function: ENABLE IOAREA POOL"
FCT = *OPEN	Open a PAM file	"FASTPAM function: OPEN"
FCT = *CLOSE	Close a PAM file	"FASTPAM function: CLOSE"
FCT = *DISIPO	Disable a FASTPAM I/O area pool, i.e. remove a link to the pool and possibly delete it	"FASTPAM function: DISABLE IOAREA POOL"
FCT = *DISENV	Disable a FASTPAM environment, i.e. remove a link to the environment and possibly delete it	"FASTPAM function: DISABLE ENVIRONMENT"

Note

All addresses passed to FASTPAM must be valid 31-bit addresses. In particular, bit 32 must not be set, otherwise it will be regarded as belonging to the address.

FASTPAM function: ENABLE ENVIRONMENT

This function can be used to create a FASTPAM environment or to connect the user with an existing environment. A FASTPAM environment ID (FPAMENID) is entered in the parameter list; this ID must be used in subsequent OPEN calls. If the same parameter list is used for such calls, the ID will have already been entered and need not be filled in by the user.

The ENAENV function only evaluates the operands described below.

Format FCT=*ENAENV

Operation	Operands
FPAMSRV	**[,FCT = *ENAENV / adr / (r)]** **[,ENVNAME = 'name' / adr / (r)]** **[,ACCLSTS = adr / (r)]** **[,ACCNUMB = number / adr / (r)]** **[,MAXIOLN = *NOT_SPECIFIED / *MINI / *MAXI / adr / (r)]** **[,EVENTNG = *NOT_SPECIFIED / *NO / *YESadr(r)]** **[,EIID = nmbr / adr / (r)]** **[,RES = *NOT_SPECIFIED / *NO / *YES / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = F / pre]
	MF = C / M **[,PREFIX = F / pre]** **[,MACID = PAM / macid]**

Operand descriptions

ACCLSTS

ACCLSTS specifies the 4-Kbyte-aligned starting address of the contiguous area containing all the FPAMACC parameter lists. If this area created as a memory-resident area, it must not overlap with the parameter list area of other environments, with I/O area pools, or DIV windows.

If the area is located in a memory pool, the pool must be one that was created with the operand FIXED=YES in the ENAMP macro.

Only the symbolic address is allowed for the MF=L form, but no symbolic names may be used within a DSECT, since its address is not known until runtime.

= addr
Symbolic address (name) of the area.

= (r)
Register containing the starting address of the area.

ACCNUMB

ACCNUMB specifies the number of parameter lists contained in the area defined by ACCLSTS. This area must therefore be requested with a minimum size of ACCNUMB * (length of the FPAMACC parameter list).

If the application is not run with FASTPAM authorization, the number of parameter lists is restricted to 500; otherwise, 5000.

Only a direct specification is allowed for the MF=L form.

= number
Specifies the number of parameter lists to be held in the area defined by ACCLISTS (1 <= number <= 5000).

= addr
Symbolic address of a 4-byte field containing the number of parameter lists as a numeric value (binary).

= (r)
Register containing the number of parameter lists as a numeric value.

EIID

Specifies the short ID for the event item via which the end of a job is indicated during file access. This ID is returned to the user via the EIIDRET operand of the ENAEI macro (see also the “Executive Macros” manual [2]).
If the FPAMACC parameter lists are located in a memory pool, it is important to ensure that the scope of the pool is not larger than that of the event item (return code FPAMEISS).

The EIID operand is not interpreted if EVENTNG=*NO is specified.

Only a direct specification is allowed for the MF=L form.

= nmbr
Decimal numeric value of the short ID for the event item.

= addr
Symbolic address (name) of the 4-byte field containing the short ID for the event item.

= (r)
Register containing the short ID for the event item.

ENVNAME

Designates the name of the environment.

Only a direct specification is allowed for the MF=L form.

= 'name'
Name of the environment.
Name length: 1 <= 'name' <= 54 characters.
Naming conventions:
1st position: a letter or the special character #, @ (or $ for TPR tasks).
2nd - 54th position: any combination from the character set (A,...,Z,0,...,9,$,#,@).

The name is terminated by the first blank (X'40').
The name must be enclosed in single quotes.

= addr
Symbolic address of a 54-byte field containing the name of the environment.

= (r)
Register containing the address of a 54-byte field with the name of the environment.

EVENTNG

Determines whether the end of a job is reported to the user via the eventing mechanism when files are accessed asynchronously (see also the section on “FASTPAM functions, eventing” in the “Introductory Guide to DMS” [1]).

Only a direct specification is allowed for the MF=L form.

= *YES
Indicates that the user wishes to work with eventing, so the short ID of the event item (operand EIID) must be specified. Subsequent OPEN calls can be made with EVENTNG=*YES as well as with EVENTNG=*NO.

= *NO
Means that files can no longer be opened with the parameter EVENTNG=*YES using this environment.

The EIID operand is not interpreted for EVENTNG=*NO.

= *NOT_SPECIFIED
Means that the environment already exists and that the user wishes to join that environment regardless of the setting for the corresponding attribute.

= addr
Symbolic address of a 1-byte field containing the value for EVENTNG.

= (r)
Register containing the value for EVENTNG.

FCT

Defines the FASTPAM function to be executed.

Only a direct specification is allowed for the MF=L form.

= *ENAENV
Direct specification for the ENABLE ENVIRONMENT function.

This function creates a FASTPAM environment or connects the user to an existing environment.
A FASTPAM environment ID (FPAMENID) is entered in the parameter list; this ID must then be used in subsequent OPEN calls. If the same parameter list is used for such calls, the ID will have already been entered and need not be taken into account.

= addr
Symbolic address of a 1-byte field containing the value for the ENABLE ENVIRONMENT function.

= (r)
Register containing the value for the ENABLE ENVIRONMENT function.

MACID

Defines the second to the fourth character (inclusive) of the field names and equates that are generated when macros are resolved.

= PAM
Default value: MACID=PAM

= macid
“macid” is three-character string that defines the second to the fourth characters (inclusive) of the generated field names and equates.

MAXIOLN

Defines the maximum I/O length that is possible with this environment. This length must not be exceeded when accessing files, but smaller I/O lengths may be used.
When working with resident FASTPAM environments, additional system memory (class 3) is reserved for the preformatted I/O paths. The following allocation is made for each I/O path:

- 1 Kbyte with MAXIOLN = *MINI
- 2 Kbytes with MAXIOLN = *MAXI

Only a direct specification is allowed for the MF=L form.

= *MINI
An I/O path for 4-Kbyte transfers is created for each FPAMACC parameter list.

= *MAXI
An I/O path for 32-Kbyte transfers is created for each FPAMACC parameter list.

= *NOT_SPECIFIED
Means that the environment already exists and that the user wishes to join that environment regardless of the setting for the corresponding attribute.

= addr
Symbolic address of a 1-byte field containing the maximum I/O length for this environment.

= (r)
Register containing the value for MAXIOLN.

MF

The forms of the MF operand are described in detail in the appendix, "Macro types".

PARAM

Indicates the address of the operand list. This operand is only evaluated in conjunction with MF=E (see also "Macro types").

PREFIX

Defines the first character of field names and equates that are generated when macros are expanded.

= F
Default value: PREFIX=F

= pre
“pre” is a one-character prefix with which the generated field names and equates are to begin.

RES

Specifies whether the environment is to be made resident.

= *YES
The environment is to be created in resident memory. In this case, a check is performed to determine whether the user ID has the required FASTPAM authorization and whether the number of resident pages requested in the program call (user catalog: “RESIDENT-PAGES” or “CLASSII”) is sufficient for the FPAMACC parameter list area. The size of the FPAMACC parameter list area is determined by the ACCUNUMB operand.
If the result of the check is negative, the user receives the return code “FPAMNORE”, and the FASTPAM environment is made non-resident.

= *NO
A non-resident environment is created.

= *NOT_SPECIFIED
The environment already exists, and the user wishes to join it, regardless of whether or not it was made resident.

= addr
Symbolic address of a 1-byte field containing the value for RES.

= (r)
Register containing the value for RES.

*Possible return codes of the FASTPAM function FCT = * ENAENV*

Standard header: `ccbbaaaa`

The following code relating to execution of the FPAMSRV macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	Function executed successfully.
	X'01'	X'0001'	Function not executed. Invalid environment name.
	X'01'	X'0005'	Function not executed. Invalid address of operand list.
	X'01'	X'0006'	Function not executed. Invalid number of operand lists.
	X'01'	X'000A'	Function not executed. Maximum permissible length of I/O area exceeded (max. 4KB or 32KB).

X'01'	X'000B'	Function not executed. Invalid eventing.
X'01'	X'000D'	Function not executed. Invalid event-item short-ID.
X'01'	X'0012'	Function not executed. • Invalid specification for RES • NOT_SPECIFIED was entered for RES, but the specified environment does not exist.

	X'20'	X'0028'	Function not executed. System error. Run system diagnostics.
	X'40'	X'0032'	The environment or I/O area pool could not be created in resident memory. The application system can still be used for testing, but without the benefits of FASTPAM performance. Subcode2 specifies the cause of the error.
X'01'	X'40'	X'0032'	The user ID of the task that created the environment or I/O area pool does not have the required FASTPAM authorization. Response: inform the system administrator.
X'02'	X'40'	X'0032'	Not enough room in actual memory.
X'03'	X'40'	X'0032'	The amount of resident main memory allocated at the start of the program is not sufficient.
X'04'	X'40'	X'0032'	Connection to a nonresident environment or a nonresident I/O area pool.
X'05'	X'40'	X'0032'	This FASTPAM version only supports non-resident data spaces.
	X'40'	X'0035'	An I/O operation is being executed on the memory pages that are to be fixed during execution of the *ENAENV/*ENAIPO function. This return code only occurs when creating the environment or I/O area pool. There is no restriction on I/Os when joining an existing environment or I/O area pool.
	X'40'	X'0037'	System resource bottleneck. Response: inform the system administrator.
	X'40'	X'0038'	The named FASTPAM resource can be enabled by TPR tasks only.
	X'40'	X'0039'	The named FASTPAM resource can only be enabled by tasks of an ID with FASTPAM privileges.
	X'40'	X'003B'	The user wishes to join an existing environment with some other operand value for ACCLSTS.
	X'40'	X'003C'	The user wishes to join an existing environment with some other operand value for ACCNUMB.
	X'40'	X'003F'	The user wishes to join an existing environment with some other operand value for MAXIOLN.
	X'40'	X'0040'	The user wishes to join an existing environment with some other operand value for EVENTNG.
	X'40'	X'0041'	The user wishes to join an existing environment with some other event item.
	X'40'	X'0046'	The user wishes to join an existing environment, but is not connected to the associated event item short ID.
	X'40'	X'0047'	The scope of the event item is smaller than that of the FPAMACC parameter list memory area.
	X'40'	X'0048'	The task is already connected to the environment.

X'40'	X'004A'	The specified user memory area overlaps a DIV window. This return code only occurs if the environment is made resident.

	X'40'	X'004B'	The specified user memory area overlaps a FASTPAM user memory area that is already in use. This return code is only output if a resident environment or resident I/O area pool is being used.
	X'40'	X'004C'	The request to allocate memory for FPAMACC parameter lists is not complete.
	X'40'	X'005B'	The ' ENAMP' call to create the memory pool for the access lists or I/O area pool was not specified with 'FIXED=YES'. This is required even if 'SCOPE=LOCAL' .

FASTPAM function: ENABLE IOAREA POOL

This function can be used to create an I/O area pool or to connect the caller with an existing pool.
A pool ID (FPAMIPID) is entered in the parameter list; this ID must then be used in subsequent OPEN calls. If the same parameter list is used for such calls, the ID will have already been entered and need not be filled in by the user.

The ENAIPO function only evaluates the operands described below.

Format FCT=*ENAIPO

Operation	Operands
FPAMSRV	**[,FCT = *ENAIPO / adr / (r)]** **[,IPONAME = 'name' / adr / (r)]** **[,IPOADDR = (elem1,elem2)]** **[,IPOSIZE = size / adr / (r)]** **[,RES = *NOT_SPECIFIED / *NO / *YES / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = F / pre]
	MF = C / M **[,PREFIX = F / pre]** **[,MACID = PAM / macid]**

Operand descriptions

FCT

Defines the FASTPAM function to be executed.

Only a direct specification is allowed for the MF=L form.

= *ENAIPO
Direct specification for the ENABLE IOAREA POOL function.

This function creates an I/O area pool or connects the user with an existing pool.A pool ID (FPAMIPID) is entered in the parameter list; this ID must then be used in subsequent OPEN calls. If the same parameter list is used for such calls, the ID will have already been entered and need not be taken into account.

= addr
Symbolic address of a 1-byte field containing the value for the ENABLE IOAREA POOL function to be executed.

= (r)
Register containing the value for the ENABLE IOAREA POOL function.

IPOADDR

Defines the location of the I/O area pool in a data space or a program space by means of a list of 2 elements (see the “Executive Macros” manual [2]).

= elem1
Specifies the 4K-aligned starting address of the memory area in the data space or program space. If this area is created in resident memory, it must not overlap with the memory area of some other I/O area pool, with the parameter list area of an environment, or a DIV window and must be requested beforehand (REQM, REQMP...).

If the area lies in a memory pool, the memory pool must be one that was created by using the ENAMP macro with the operand FIXED=YES.

Only the symbolic address is allowed for the MF=L form, but no symbolic names may be used within a DSECT, since its address is not known until runtime.

elem1 has the following format:

```
elem1 =adr1 / (r1)
```

where:

addr1 is the symbolic starting address of the I/O area pool in the data or program space.

(r1) is a register containing the starting address of the I/O area pool in the data or program space.

= elem2
Identifies the address space.
elem2 = 0: the I/O area pool lies in the program address space.
elem2 0: the I/O area pool lies in a data space with ALET <elem2>
(ALET stands for Access List Entry Token, i.e a pointer to an entry in the access list; see the “Executive Macros” manual [2] for details). Note that only non-resident data spaces are currently supported.

Only a direct specification is allowed for the MF=L form.

elem2 has the following format:

```
elem2 =adr2 / nmbr / (r2)
```

where:

nmbr is the numeric value of the ALET or 0.

addr2 is the symbolic address of a 4-byte field containing the ALET or 0 (binary).

(r2) is a register containing the ALET or 0.

IPONAME

Designates the name of the I/O area pool.

Only a direct specification is allowed for the MF=L form.

= 'name'
Name of the I/O area pool:
1 <= 'name' <= 54 characters.
Naming conventions:
1st position: a letter or the special characters # and @
2nd - 54th position: any combination from the character set (A,...,Z,0,...,9,$,#,@).
The name is terminated by the first blank (X'40')
The name must be enclosed in single quotes.

= addr
Symbolic address of a 54-byte field containing the name of the I/O area pool.

= (r)
Register that holds the address of a 54-byte field containing the name of the I/O area pool.

IPOSIZE

Indicates the size of the memory area for the I/O area pool in units of 4 Kbytes. This area should have already been created with the appropriate size.

Only a direct specification is allowed for the MF=L form.

= size
Size of the memory area for the I/O area pool in 4-Kbyte units: 1 <= size <= 2^{19}.

= addr
Symbolic address of a 4-byte field containing the size of the memory area for the I/O area pool in units of 4 Kbytes.

= (r)
Register containing the size of the memory area in units of 4 Kbytes.

MACID

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

MF

The forms of the MF operand are described in detail in the appendix, "Macro types".

PARAM

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

PREFIX

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

RES

Specifies whether the I/O area pool is to be made resident.

Only a direct specification is allowed for the MF=L form.

= *NOT_SPECIFIED
The I/O area pool already exists, and the user wishes to join it, regardless of whether or not it was made resident.

= *YES
The I/O area pool is to be created in resident memory.
In this case, a check is performed to determine whether the user ID has the required FASTPAM authorization and whether the number of resident pages requested in the program call (user catalog: “RESIDENT-PAGES” or “CLASSII”) is sufficient for the I/O area pool. If the result of the check is negative, the user receives the return code “FPAMNORE”, and the I/O area pool is made non-resident.

Space in data spaces is provided only on a non-resident basis.

= *NO
The I/O area pool is not to be created in resident memory.

= addr
Symbolic address of a 1-byte field containing the value for RES.

= (r)
Register containing the value for RES.

*Possible return codes of the FASTPAM function FCT=*ENAIPO*

Standard header: `ccbbaaaa`

The following code relating to execution of the FPAMSRV macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	Function executed successfully.
	X'01'	X'0002'	Function not executed. Invalid name of I/O area pool.
	X'01'	X'0007'	Function not executed. Invalid memory address of I/O area pool.
	X'01'	X'0008'	Function not executed. Invalid size of I/O area pool.
	X'01'	X'0012'	Function not executed. • Invalid specification for RES • NOT_SPECIFIED was entered for RES, but the specified environment does not exist.
	X'20'	X'0028'	Function not executed. System error. Run system diagnostics.
	X'40'	X'0032'	The environment or the I/O area pool could not be created in resident memory. The application system can still be used for testing, but without the benefits of FASTPAM performance. Subcode2 specifies the cause of the error.

X'01'	X'40'	X'0032'	The user ID of the task that created the environment or I/O area pool does not have the required FASTPAM authorization. Response: inform the system administrator.
X'02'	X'40'	X'0032'	Not enough room in actual memory.
X'03'	X'40'	X'0032'	The amount of resident main memory allocated at the start of the program is not sufficient.
X'04'	X'40'	X'0032'	Connection to a nonresident environment or a nonresident I/O area pool.

X'05'	X'40'	X'0032'	This FASTPAM version only supports non-resident data spaces.
	X'40'	X'0035'	An I/O operation is being executed on the memory pages that are to be fixed during execution of the *ENAENV/*ENAIPO function. This return code only occurs when creating the environment or I/O area pool. There is no restriction on I/Os when joining an existing environment or I/O area pool.
	X'40'	X'0037'	System resource bottleneck. Response: inform the system administrator.
	X'40'	X'0038'	The named FASTPAM resource can be enabled by TPR tasks only.
	X'40'	X'0039'	The named FASTPAM resource can only be enabled by tasks of an ID with FASTPAM privileges.
	X'40'	X'003D'	The user wishes to join an existing I/O area pool with some other operand value for IPOADDR.
	X'40'	X'003E'	The user wishes to join an existing I/O area pool with some other operand value for IPOSIZE.
	X'40'	X'0049'	The task is already connected to the I/O area pool.
	X'40'	X'004A'	The specified user memory area overlaps a DIV window. This return code only occurs if the environment is made resident.
	X'40'	X'004B'	The specified user memory area overlaps a FASTPAM user memory area that is already in use. This return code is only output if a resident environment or resident I/O area pool is being used.
	X'40'	X'0057'	The request to allocate memory for the I/O area pool is not complete.
	X'40'	X'005B'	The ' ENAMP' call to create the memory pool for the access lists or I/O area pool was not specified with 'FIXED=YES'. This is required even if 'SCOPE=LOCAL'.

FASTPAM function: OPEN

This function can be used to open a PAM file with the short IDs returned by ENABLE ENVIRONMENT and ENABLE IOAREA POOL.
A short ID (FPAMOPID) is returned in the parameter list. This ID must be copied to the FPAMACC parameter lists for all following accesses to the file.

The OPEN function only evaluates the operands described below.

Format FCT=*OPEN

Operation	Operands
FPAMSRV	**[,FCT = *OPEN / adr / (r)]** **[,ENVID = nmbr / adr / (r)]** **[,IPOID = nmbr / adr / (r)]** **[,LINK = 'name' / adr / (r)]** **[,FILE = 'pathname' / adr / (r)]** **[,SHARUPD = *NO / *YES / adr / (r)]** **[,MODE = *INPUT / *INOUT / *OUTIN / adr / (r)]** **[,EVENTNG = *NO / *YES / adr(r)]** **[,BLKSIZE = size / adr / (r)]** **[,ENV = *HOST / *XCS / adr / (r)]** **[,LARGE_FILE = *FORBIDDEN / *ALLOWED / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = F / pre]
	MF=C / M **[,PREFIX = F / pre]** **[,MACID = PAM / macid]**

Operand descriptions

BLKSIZE

Defines the block size for following I/O operations in 4K units. The value specified for BLKSIZE must not exceed the maximum value that was defined for the MAXIOLN parameter in the ENABLE ENVIRONMENT function.

Only a direct specification is allowed for the MF=L form.

= size
Specifies the block size in 4K units: 1 <= size <= 8

= addr
Symbolic address of a 1-byte field containing the block size in 4K units (binary).

= (r)
Register containing the block size in 4K units.

ENV

Affects the compatibility of parallel openers dependent on their execution location (cf. “Compatibility matrix: FASTPAM with UPAM/FASTPAM/DIV” in chapter Processing files with FASTPAM).

= *HOST
The maximum permissible parallelism is limited to openers running on the same host.

= *XCS
The openers can run in different hosts in an XCS network without restricting the compatibility (e.g. write operations with SHARUPD=*YES can run in parallel).

= addr
Symbolic address of a 1-byte field containing the value for ENV.

= (r)
Register containing the value for ENV.

ENVID

Designates the short ID of the environment with which the file is to be opened. If the same parameter list is used as for ENABLE ENVIRONMENT, the short ID will have already been entered in the parameter list (FPAMENEV) and need not be taken into account.

The ENVID specification is not allowed for the MF=L form.

= nmbr
Direct entry of the short ID as a decimal numeric value.

=addr
Address of a 4-byte field containing the short ID.

=(r)
Register containing the short ID.

EVENTNG

Determines whether the end of a job is reported to the user via the eventing mechanism when files are accessed asynchronously (see also the section on “FASTPAM functions, eventing” in the “Introductory Guide to DMS” [1]).

Only a direct specification is allowed for the MF=L form.

= *YES
Indicates that the user wishes to work with eventing. This value will only be accepted if EVENTNG=*YES was specified when creating the environment.

= *NO
Indicates that the user does not wish to work with eventing.

= addr
Symbolic address of a 1-byte field containing the value for EVENTNG.

= (r)
Register containing the value for EVENTNG.

FCT

Defines the FASTPAM function to be executed.

Only a direct specification is allowed for the MF=L form.

= *OPEN
Direct specification of the OPEN function.
This function can be used to open a PAM file with the short IDs returned by ENABLE ENVIRONMENT and ENABLE IOAREA POOL.
A short ID (FPAMOPID) is returned in the parameter list. This ID must be entered in the FPAMACC parameter lists for all following accesses to the file.

= addr
Symbolic address of a 1-byte field with the value for the OPEN function.

= (r)
Register containing the value for the OPEN function.

FILE

Specifies the path name of the file. The FILE specification is not evaluated if a value has been specified for the LINK operand.

Only a direct specification is allowed for the MF=L form.

= 'pathname'
<c-string 1..54: filename 1..54>.
The name must be enclosed in single quotes.

= addr
Address of a 54-byte field containing the path name.

= (r)
Register containing the field with the address of the path name.

IPOID

Designates the short ID of the I/O area pool with which the file is to be opened. If the same parameter list is used as for ENABLE IOAREA POOL, the short ID will have already been entered in the parameter list (and is hence not required).

Only a direct specification is allowed for the MF=L form.

= nmbr
Short ID of the I/O area pool as a decimal numeric value.

= addr
Address of a 4-byte field containing the short ID.

= (r)
Register containing the short ID.

LARGE_FILE

Specifies whether the file that is to be opened can grow to become a “large file” with a file size >= 32 GB.

Default value: LARGE_FILE = *FORBIDDEN

In the case of MF=L, only direct specification is permitted.

= *FORBIDDEN
The file may not become a “large file”.

= ALLOWED
The file may become a “large file”.

= addr
The address of an 8-byte field that contains the value for LARGE_FILE.

= (r)
Register containing the address of an 8-byte field with the value for LARGE_FILE.

LINK

Specifies the file link name.

Only a direct specification is allowed for the MF=L form.

= 'name'
File link name with: <c-string 1..8> (enclosed in single quotes)

If the file link name is to be accessed via the command interface it must correspond to the data type <structured_name 1..8> (see the “Commands” manual [3]).

= addr
Address of an 8-byte field containing the file link name.

= (r)
Register containing the address of the field with the file link name.

MACID

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

MF

The forms of the MF operand are described in detail in the appendix on "Macro types".

MODE

Defines the OPEN mode (see the sections “Multiuser mode on one computer” in chapter Processing files with FASTPAM).

Only a direct specification is allowed for the MF=L form.

= *INPUT
The file can only be read.
Parallel INPUT opens are possible, even with the UPAM and DIV access methods, regardless of the SHARUPD mode.
The file must exist, i.e. must have been opened once with OUTIN.

= *INOUT
The file can also be written to.
The SHARUPD mode and ENV operand determine whether parallel opens are possible.
The file must exist, i.e. must have been opened once with OUTIN.

= *OUTIN
The file can also be written to.
It is, however, recreated, i.e. will be empty after the OPEN.
The SHARUPD mode and DIV operand determine whether parallel opens are possible.In multi-user mode (SHARUPD=*YES), any user who wishes to open the file with MODE=*OUTIN must always be the first user to do so; otherwise, access will be denied.

= addr
Symbolic address of a 1-byte field containing the value for the OPEN mode.

= (r)
Register containing the value for the OPEN mode.

PARAM

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

PREFIX

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

SHARUPD

Controls multi-user mode (see also the section on “FASTPAM functions, multi-user mode...” in the “Introductory Guide to DMS” [1]).

Only a direct specification is allowed for the MF=L form.

= *NO
The file can be concurrently read by more than one user (MODE=*INPUT) or can be written to by exactly one user (MODE=*INOUT | *OUTIN).

= *YES
The file can be concurrently read and written by multiple users.

Notes

FASTPAM does not provide any “block-locking” mechanism (LOCK/UNLOCK functions). An appropriate locking mechanism must therefore be supplied by the user. Concurrent access by UPAM applications must be synchronized differently.

The file size is checked whenever the allocator is called.
If this check indicates a file size >= 32 GB and the attribute LARGE_FILE=*FORBIDDEN is set in the associated FCB or the attribute EXCEED-32GB=*FORBIDDEN is set in the TFT then processing is canceled. In this case, FASTPAM returns the code **X'**`00400145`**'** in its local parameter list FPAMACC(I).

= addr
Symbolic address of a 1-byte field containing the value for SHARUPD.

= (r)
Register containing the value for SHARUPD.

*Possible return codes of the FASTPAM function FCT=*OPEN*

Standard header: `ccbbaaaa`

The following code relating to execution of the FPAMSRV macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	Function executed successfully.
	X'01'	X'0009'	Function not executed. Invalid specification for SHARE UPDATE max. 4K or 32K
	X'01'	X'000B'	Function not executed. Invalid eventing.
	X'01'	X'000C'	Function not executed. Invalid specification for MODE
	X'01'	X'000E'	Function not executed. Invalid specification for logical block length (BLKSIZE). *Note* The BLKSIZE is specified in 2-Kbyte units in the ADD-FILE-LINK command; the value specified there is therefore equivalent to half the block size specified in the FASTPAM OPEN.
	X'01'	X'000F'	Function not executed. Invalid short ID for environment.
	X'01'	X'0010'	Function not executed. Invalid short ID for I/O area pool.
	X'01'	X'0014'	Function not executed. Invalid ENV specification.
	X'01'	X'0015'	Function not executed. Invalid LARGE_FILE specification.
	X'20'	X'0028'	Function not executed. System error. Run system diagnostics.

X'40'	X'0033'	General DMS error during OPEN/CLOSE. The DMS return code is passed in the field FPAMDMSC. Response: evaluate DMS return code.
X'40'	X'0037'	System resource bottleneck. Response: inform the system administrator.
X'40'	X'0042'	The value specified for BLKSIZE in the FASTPAM OPEN does not match the value in the catalog entry.
X'40'	X'004D'	The named file is not a PAM file.
X'40'	X'0050'	RFA is not supported by FASTPAM.
X'40'	X'0051'	SPD is not supported by FASTPAM.
X'40'	X'0052'	PPD is not supported by FASTPAM.
X'40'	X'0053'	Tape files are not supported by FASTPAM.

	X'40'	X'0054'	*DUMMY is not supported by FASTPAM.
	X'40'	X'0055'	The specified secondary allocation is too small for a logical block.
	X'40'	X'0056'	An unprivileged user has specified the short ID of an environment or I/O area pool that was created by a privileged user.
	X'40'	X'0058'	FASTPAM only supports files with the attribute 'BLKCTRL = NO'.
	X'40'	X'005A'	'WRCHK=YES' was specified in a call to FILE.

FASTPAM function: CLOSE

This function can be used to close a PAM file. The file to be closed is identified by the short ID (OPENID) that was returned by OPEN.

The CLOSE function evaluates only the function operands described below.

Format FCT=*CLOSE

Operation	Operands
FPAMSRV	**[,FCT = *CLOSE / adr / (r)]** **[,OPENID = nmbr / adr / (r)]** **[,LASTBLK = nmbr / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = F / pre]
	MF = C / M **[,PREFIX = F / pre]** **[,MACID = PAM / macid]**

Operand descriptions

FCT

Defines the FASTPAM function to be executed.

Only a direct specification is allowed for the MF=L form.

= *CLOSE
Direct specification of the “Close file” function.

This function can be used to close a PAM file by calling it with the short ID (OPENID) that was returned by OPEN.

= addr
Symbolic address of a 1-byte field containing the value for the CLOSE function.

= (r)
Register containing the value for the CLOSE function.

LASTBLK

This parameter allows the user to explicitly set the last logical block of the file, provided the user has opened the file with MODE=*INOUT/*OUTIN and SHARUPD=*NO. The specified block must lie within the file.

Only a direct specification is allowed for the MF=L form.

= nmbr
Direct entry of a decimal numeric value for the last 4K block of the file.

= addr
Symbolic address of a 4-byte field containing the numeric value (binary) for the last4-Kbyte-block of the file.

= (r)
Register containing the numeric value for LASTBLK.

MACID

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

MF

The forms of the MF operand are described in detail in the appendix on "Macro types".

PARAM

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

PREFIX

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

OPENID

Refers to the short ID of the OPEN for which this CLOSE function is to be executed.If the same parameter list is used as for the OPEN, the short ID need not be specified, since it will already be contained in the FPAMOPID field of the parameter list.

Only a direct specification is allowed for the MF=L form.

= nmbr
Direct entry of a decimal numeric value for the OPENID.

= addr
Address of a 4-byte field containing the short ID.

= (r)
Register containing the short ID.

*Possible return codes of the FASTPAM function FCT=*CLOSE*

Standard header: `ccbbaaaa`

The following code relating to execution of the FPAMSRV macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	Function executed successfully.
	X'01'	X'0011'	Function not executed. Invalid short ID for the OPEN
	X'01'	X'0013'	Function not executed. Invalid specification for the last block. The CLOSE operation is completed, but for updating the last-page pointer.
	X'20'	X'0028'	Function not executed. System error. Run system diagnostics.
	X'40'	X'0033'	General DMS error during OPEN/CLOSE. The DMS return code is passed in the field FPAMDMSC. Response: evaluate DMS return code.
	X'40'	X'004E'	When calling FPAMSRV with the *CLOSE function, the operand LASTBLK was specified. This operand is ignored, since the file was opened with MODE=*INPUT or SHARUPD=*YES.
	X'40'	X'0059'	A TU task is using a TPR-OPEN-ID.

FASTPAM function: DISABLE IOAREA POOL

This function removes the link between the user and the I/O area pool. If the function is called by the last user, the I/O area pool is disabled. The I/O area pool is addressed by the short ID that is returned by ENABLE IOAREA POOL.

The DISIPO function evaluates only the function operands described below.

Format FCT=*DISIPO

Operation	Operands
FPAMSRV	**[,FCT = *DISIPO / adr / (r)]** **[,IPOID = nmbr / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = F / pre]
	MF = C / M **[,PREFIX = F / pre]** **[,MACID = PAM / macid]**

Operand descriptions

FCT

Defines the FASTPAM function to be executed.

Only a direct specification is allowed for the MF=L form.

= *DISIPO
Direct specification of the DISABLE IOAREA POOL function.

This function removes the link between the user and the I/O area pool. If the function is called by the last user, the I/O area pool is disabled. The I/O area pool is addressed by the short ID that is returned by ENABLE IOAREA POOL.

This function will not be executed if files using the I/O area pool are still open (return code FPAMOFI).

= addr
Symbolic address of a 1-byte field containing the value for the DISABLE IOAREA POOL function.

= (r)
Register containing the value for the DISABLE IOAREA POOL function.

IPOID

Designates the short ID of the I/O area pool to be disconnected or disabled.
If the same parameter list is used as for ENABLE IOAREA POOL, the short ID need not be specified, since it will already be contained in the FPAMIPID field of the parameter list.

Only a direct specification is allowed for the MF=L form.

= nmbr
Direct entry of a decimal numeric value for the short ID of the I/O area pool.

= addr
Address of a 4-byte field containing the short ID of the I/O area pool.

= (r)
Register containing the short ID of the I/O area pool.

MACID

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

MF

The forms of the MF operand are described in detail in the appendix on "Macro types".

PARAM

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

PREFIX

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

*Possible return codes of the FASTPAM function FCT=*DISIPO*

Standard header: ccbbaaaa

The following code relating to execution of the FPAMSRV macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	Function executed successfully.
	X'01'	X'0010'	Function not executed. Invalid short ID for the I/O area pool
	X'20'	X'0028'	Function not executed. System error. Run system diagnostics.
	X'40'	X'0036'	The *DISIPO/*DISENV function cannot be executed, since files opened with the corresponding I/O area pool or environment still exist.
	X'40'	X'0056'	A TU user is attempting to disable a FASTPAM I/O area pool created by a privileged user.

FASTPAM function: DISABLE ENVIRONMENT

This function removes the link between the user and the environment. If the function is called by the last user, the environment is disabled. The environment is addressed by the short ID that is returned by ENABLE ENVIRONMENT.

The DISENV function evaluates only the function operands described below.

Format FCT=*DISENV

Operation	Operands
FPAMSRV	**[,FCT = *DISENV / adr / (r)]** **[,ENVID = nmbr / adr / (r)]** **MF = L**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = F / pre]
	MF = C / M **[,PREFIX = F / pre]** **[,MACID = PAM / macid]**

Operand descriptions

ENVID

Designates the short ID of the environment to be disconnected or disabled. If the same parameter list is used as for ENABLE ENVIRONMENT, the short ID need not be specified, since it will already be contained in the FPAMENEV field of the parameter list.

Only a direct specification is allowed for the MF=L form.

=nmbr
Direct entry of a decimal numeric value for the short ID of the environment.

=addr
Address of a 4-byte field containing the short ID of the environment.

=(r)
Register containing the short ID of the environment.

FCT

Defines the FASTPAM function to be executed.

Only a direct specification is allowed for the MF=L form.

= *DISENV
Direct specification of the DISABLE ENVIRONMENT function.

This function removes the link between the user and the environment. If the function is called by the last user, the environment is disabled. The environment is addressed via the short ID that is returned by ENABLE ENVIRONMENT.

This function will not be executed if there are open files which are still using the environment (return code FPAMOFI).

Only a direct specification is allowed for the MF=L form.

= addr
Symbolic address of a 1-byte field containing the value for the DISABLE ENVIRONMENT function.

= (r)
Register containing the value for the DISABLE ENVIRONMENT function.

MACID

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

MF

The forms of the MF operand are described in detail in the appendix, "Macro types".

PARAM

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

PREFIX

See the description under the format FCT=*ENAENV on "FASTPAM function: ENABLE ENVIRONMENT".

*Possible return codes of the FASTPAM function FCT=*DISENV*

Standard header: `ccbbaaaa`

The following code relating to execution of the FPAMSRV macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	Function executed successfully.
	X'01'	X'000F'	Function not executed. Invalid environment short ID.
	X'20'	X'0028'	Function not executed. System error. Run system diagnostics.
	X'40'	X'0036'	The *DISIPO/*DISENV function cannot be executed, since files opened with the corresponding I/O area pool or environment still exist.
	X'40'	X'0056'	A TU user is attempting to disable a FASTPAM environment created by a privileged user (TPR).

Return codes of the FPAMSRV macro

The field names and the EQU instructions for return codes which are generated by the C or D form of the macro begin with the string FPAM by default. This string can be changed by means of PREFIX and MACID.

The return codes are placed in the header of the parameter list (standard header):

- The main return code, in a half-word with the name FPAMMRET.
- Subcode1, in a byte with the name FASTSR1.
  Subcode1 describes error classes which allow the caller to respond to similar error situations.
  The caller can refer back to the main code as well as to subcode1.
- Subcode2, in a byte with the name FPAMSR2.
  Subcode2 specifies the individual main codes more precisely.

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

If the return codes cannot be placed in the header (because it is not accessible, for example), the calling program is terminated with an error message, and the STXIT event for an unrecoverable program error is generated.

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller
- the list is not aligned on a word boundary
- the list is write-protected.

In the following section, the main return codes are assigned to the appropriate subcode1 classes and are described more precisely via subcode2.

Standard header: `ccbbaaaa`

The following code relating to execution of the CATAL macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	meaning
	X'00'	X'0000'	Function executed successfully.
	X'01'	X'0001'	Function not executed. Invalid environment name.
	X'01'	X'0002'	Function not executed. Invalid name for I/O area pool.
	X'01'	X'0005'	Function not executed. I Invalid address of operand list.
	X'01'	X'0006'	Function not executed. Invalid number of operand lists.
	X'01'	X'0007'	Function not executed. Invalid memory address of I/O area pool.

X'01'	X'0008'	Function not executed. Invalid size of I/O area pool.
X'01'	X'0009'	Function not executed. Invalid specification for SHARE UPDATE.
X'01'	X'000A'	Function not executed. Invalid specification for MAXIOLN.
X'01'	X'000B'	Function not executed. Invalid specification for EVENTNG.
X'01'	X'000C'	Function not executed. Invalid specification for MODE.
X'01'	X'000D'	Function not executed. Invalid event item short ID.
X'01'	X'000E'	Function not executed. Invalid specification for logical block length (BLKSIZE). *Note* The BLKSIZE is specified in 2K units in the ADD-FILE-LINK command; the value specified there is therefore equivalent to half the block size specified in the FASTPAM OPEN.
X'01'	X'000F'	Function not executed. Invalid environment short ID
X'01'	X'0010'	Function not executed. Invalid short ID for I/O area pool
X'01'	X'0011'	Function not executed. Invalid OPEN short ID.
X'01'	X'0012'	Function not executed. • Invalid specification for RES • NOT_SPECIFIED was entered for RES, but the specified environment does not exist.
X'01'	X'0013'	Invalid specification for the last block. The CLOSE operation is completed, but for updating the last-page pointer.
X'01'	X'0014'	Function not executed. Invalid ENV specification
X'01'	X'0015'	Function not executed. Invalid LARGE_FILE specification.
X'02'		Function not executed. The specified function is not available.

X'03'		Function not executed. The specified interface version is not supported.
X'20'	X'0028'	Function not executed. System error. Run system diagnostics.

	X'40'	X'0032'	The environment or I/O area pool could not be created in resident memory. The application system can still be used for testing, but without the benefits of FASTPAM performance. Subcode2 specifies the cause of the error.
X'01'	X'40'	X'0032'	The user ID of the task that created the environment or I/O area pool does not have the required FASTPAM authorization. Response: inform the system administrator.
X'02'	X'40'	X'0032'	Not enough room in actual memory.
X'03'	X'40'	X'0032'	The amount of resident main memory allocated at the start of the program is not sufficient.
X'04'	X'40'	X'0032'	Connection to a non-resident environment or a non-resident I/O area pool.
X'05'	X'40'	X'0032'	This FASTPAM version only supports non-resident data spaces.
	X'40'	X'0033'	General DMS error during OPEN/CLOSE. The DMS return code is passed in the field FPAMDMSC. Response: evaluate DMS return code.
	X'40'	X'0035'	An I/O operation is being executed on the memory pages that are to be fixed during execution of the *ENAENV/*ENAIPO function. This return code only occurs when creating the environment or I/O area pool. There is no restriction on I/Os when joining an existing environment or I/O area pool.
	X'40'	X'0036'	The *DISIPO/*DISENV function cannot be executed, since files opened with the corresponding I/O area pool or environment still exist.
	X'40'	X'0037'	System resource bottleneck. Response: inform the system administrator.
	X'40'	X'0038'	The named FASTPAM resource can be enabled by TPR tasks only.
	X'40'	X'0039'	The named FASTPAM resource can only be enabled by tasks of an ID with FASTPAM privileges.
	X'40'	X'003B'	The user wishes to join an existing environment with some other operand value for ACCLSTS.
	X'40'	X'003C'	The user wishes to join an existing environment with some other operand value for ACCNUMB.
	X'40'	X'003D'	The user wishes to join an existing environment with some other operand value for IPOADDR.
	X'40'	X'003E'	The user wishes to join an existing environment with some other operand value for IPOSIZE.
	X'40'	X'003F'	The user wishes to join an existing environment with some other operand value for MAXIOLN.

X'40'	X'0040'	The user wishes to join an existing environment with some other operand value for EVENTNG.
X'40'	X'0041'	The user wishes to join an existing environment with some other event item.
X'40'	X'0042'	The value specified for BLKSIZE in the FASTPAM OPEN does not match the value in the catalog entry.
X'40'	X'0046'	The user wishes to join an existing environment but is not connected to the associated event item short ID.
X'40'	X'0047'	The scope of the event item is smaller than that of the FPAMACC parameter list memory area.
X'40'	X'0048'	The task is already connected to the environment.
X'40'	X'0049'	The task is already connected to the I/O area pool.
X'40'	X'004A'	The specified user memory area overlaps a DIV window. This return code only occurs if the environment is made resident.
X'40'	X'004B'	The specified user memory area overlaps a FASTPAM user memory area that is already in use. This return code is only output if a resident environment or resident I/O area pool is being used.
X'40'	X'004C'	The request to allocate memory for FPAMACC parameter lists is not complete.
X'40'	X'004D'	The named file is not a PAM file.
X'40'	X'004E'	When calling FPAMSRV with the *CLOSE function, the operand LASTBLK was specified. This operand is ignored, since the file was opened with MODE=*INPUT or SHARUPD=*YES.
X'40'	X'0050'	RFA is not supported by FASTPAM.
X'40'	X'0051'	SPD is not supported by FASTPAM.
X'40'	X'0052'	PPD is not supported by FASTPAM.
X'40'	X'0053'	Tape files are not supported by FASTPAM.
X'40'	X'0054'	*DUMMY is not supported by FASTPAM.
X'40'	X'0055'	The specified secondary allocation is too small for a logical block.
X'40'	X'0056'	A TU task is attempting to release a TPR-created FASTPAM resource or to open a file by with such a resource.
X'40'	X'0057'	The request to allocate memory for the I/O area pool is not complete.
X'40'	X'0058'	FASTPAM only supports files with the attribute 'BLKCTRL = NO'.
X'40'	X'0059'	A TU task is using a TPR_OPEN_ID.

	X'40'	X'005A'	'WRCHK=YES' was specified in a call to FILE.
	X'40'	X'005B'	The 'ENAMP' call to create the memory pool for the access lists or I/O area pool was not specified with 'FIXED=YES'. This is required even if 'SCOPE=LOCAL'.

Notes on return codes

Cause of return codes with subcode1 = X'01' (PARAMETER ERROR):

- The parameter in question was not specified correctly.
- In the case of operand values that can be overwritten by ADD-FILE-LINK, the operand value set by this command is invalid. Only the values that can also be specified in the OPEN are permitted.

If an error that affects the entire FPAMACC parameter list area or the I/O area pool is detected (e.g. if only a part of the I/O area pool is located in a memory pool), return code `X'0005' (INVALID ADDRESS OF ACCESS LISTS)` or `X'0007' (INVALID NUMBER OF IOAREA POOL)` is output instead of `X'0006' (INVALID ADDRESS OF ACCESS LISTS)` or `X'0008' (INVALID SIZE OF IOAREA POOL)`.

Interdependencies of other functions

If the user attempts to free any of the memory areas that were made resident by the system (when executing ENABLE ENVIRONMENT or ENABLE IOAREA POOL) before calling the DISABLE function, the request will be denied by the corresponding RELM, RELMP or DISMP function, and an appropriate return code will be issued. A RELMP will also be denied for common memory pool areas that were defined as resident FASTPAM areas by tasks other than the user's own task.

If the user attempts to free an event item that was specified in ENABLE ENVIRONMENT before the DISABLE ENVIRONMENT function is executed, this request will also be rejected by the corresponding DISABLE EVENT ITEM function (macro DISEI) with a return code.

A call to USER-CLOSE-ALL will not be effective for files opened with FASTPAM.

Layout of the parameter list

The following parameter list is issued by a FPAMSRV macro call:

```
FPAMSRV MF=D
1          STACK  PRINT
1          PRINT  NOGEN
2                *,##### PREFIX=F, MACID=PAM #####
1          #INTF REFTYPE=REQUEST,INTNAME=FPAMSRV,INTCOMP=002
1 FPAMPA   DS    0F    BEGIN of PARAMETERAREA           _INOUT
1          FHDR  MF=(C,FPAM),EQUATES=YES
2          DS    0A
2 FPAMFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 FPAMIFID DS    0A              0   INTERFACE IDENTIFIER
2 FPAMFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 FPAMFCT  DS    AL1             2   FUNCTION NUMBER
2 FPAMFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 FPAMRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
```

```
2 FPAMSRET DS    0AL2            4   SUB RETURN CODE
2 FPAMSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 FPAMR2OK EQU   X'00'               All correct, no additional info
2 FPAMR2NA EQU   X'01'               Successful, no action was necessary
2 FPAMR2WA EQU   X'02'               Warning, particular situation
2 FPAMSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 FPAMRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
2 FPAMRPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 FPAMRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
2 FPAMRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 FPAMRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 FPAMRAER EQU   X'04'               ALIGNMENT ERROR
2 FPAMRIER EQU   X'20'               INTERNAL ERROR
2 FPAMRCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 FPAMRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 FPAMRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 FPAMRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 FPAMRWLR EQU   X'81'                    "     LONG       "
2 FPAMRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 FPAMRTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
2 FPAMRDH  EQU   X'82'               SS IN DELETE / HOLD
2 *
2 FPAMMRET DS    0AL2            6   MAIN RETURN CODE
2 FPAMMR2  DS    AL1             6   MAIN RETURN CODE 2
2 FPAMMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 FPAMRLNK EQU   X'FFFF'             LINKAGE ERROR / REQ. NOT PROCESSED
2 FPAMFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *
1 * SUB RETURN CODE2
1 *
1 FPAMPRIV EQU   X'01'       FASTPAM PRIVILEGE MISSING
1 FPAMRMS  EQU   X'02'       REAL MEMORY SHORTAGE
1 FPAMULE  EQU   X'03'       USER LIMIT EXCEEDED
1 FPAMENR  EQU   X'04'       EXISTING NOT RESIDENT
1 FPAMDSA  EQU   X'05'       DATA SPACE ADDRESS
1 *
1 * MAINCODE
```

```
1 *
1 FPAMMFSP EQU   X'0000'     FUNCTION SUCCESSFULLY PROCESSED    = 0
1 FPAMIENN EQU   X'0001'     INVALID ENVIRONMENT NAME           = 1
1 FPAMIIPN EQU   X'0002'     INVALID IOAREA POOL NAME           = 2
1 FPAMIALA EQU   X'0005'     INVALID ADDRESS OF ACCESS LISTS    = 5
1 FPAMIALN EQU   X'0006'     INVALID NUMBER OF ACCESS LISTS     = 6
1 FPAMIIPA EQU   X'0007'     INVALID ADDRESS OF IOAREA POOL     = 7
1 FPAMIIPS EQU   X'0008'     INVALID SIZE OF IOAREA POOL        = 8
1 FPAMISUP EQU   X'0009'     INVALID SHARE UPDATE               = 9
1 FPAMIMAX EQU   X'000A'     INVALID MAXIMUM IO-LENGTH          = 10
1 FPAMIEVN EQU   X'000B'     INVALID EVENTING                   = 11
1 FPAMIMOD EQU   X'000C'     INVALID MODE                       = 12
1 FPAMIEID EQU   X'000D'     INVALID EVENT-ITEM SHORT-ID        = 13
1 FPAMIBLS EQU   X'000E'     INVALID BLOCKSIZE                  = 14
1 FPAMIENI EQU   X'000F'     INVALID ENVIRONMENT SHORT-ID       = 15
1 FPAMIIPI EQU   X'0010'     INVALID IOAREA POOL SHORT-ID       = 16
1 FPAMIOPI EQU   X'0011'     INVALID OPEN SHORT-ID              = 17
1 FPAMIRES EQU   X'0012'     INVALID RESIDENT                   = 18
1 FPAMILBL EQU   X'0013'     INVALID LAST BLOCK                 = 19
1 FPAMIENV EQU   X'0014'     INVALID ENV-SPECIFICATION          = 20
1 FPAMILRF EQU   X'0015'     INVALID LARGE_FILE-SPECIFICATION   = 21
1 FPAMMIER EQU   X'0028'     INTERNAL ERROR                     = 40
1 FPAMNORE EQU   X'0032'     SPACE NOT RESIDENT                 = 50
1 FPAMDMSE EQU   X'0033'     DMS ERROR DURING OPEN/CLOSE        = 51
1 FPAMRIO  EQU   X'0035'     RUNNING IO                         = 53
1 FPAMOFI  EQU   X'0036'     OPENED FILES                       = 54
1 FPAMSRES EQU   X'0037'     SHORTAGE OF RESOURCES              = 55
1 FPAMTPR  EQU   X'0038'     ENABLE_FROM_TPR_ONLY               = 56
1 FPAMNPRI EQU   X'0039'     NO_PRIVILEGE_FOR_CONNECTION        = 57
1 FPAMDAC@ EQU   X'003B'     DIFFERENT_ACCESS_LISTS_@           = 59
1 FPAMDAC# EQU   X'003C'     DIFFERENT_ACCESS_LISTS_#           = 60
1 FPAMDIP@ EQU   X'003D'     DIFFERENT_IOAREA_POOL_@            = 61
1 FPAMDIPS EQU   X'003E'     DIFFERENT_IOAREA_POOL_SIZE         = 62
1 FPAMDMAX EQU   X'003F'     DIFFERENT_MAXIOLEN                 = 63
1 FPAMDEVE EQU   X'0040'     DIFFERENT_EVENTING                 = 64
1 FPAMDEVI EQU   X'0041'     DIFFERENT_EVENT_ITEM               = 65
1 FPAMDBC  EQU   X'0042'     DIFFERENT_BLKSIZE_IN_CATALOG       = 66
1 FPAMNEIC EQU   X'0046'     NO_EVENT_ITEM_CONNECTION           = 70
1 FPAMEISS EQU   X'0047'     EVENT_ITEM_SCOPE_TO_SMALL          = 71
1 FPAMENEX EQU   X'0048'     ENVIRONMENT_NAME_EXISTING          = 72
1 FPAMINEX EQU   X'0049'     IOAREA_POOL_NAME_EXISTING          = 73
1 FPAMODS  EQU   X'004A'     OVERLAPPING_DIV_SPACE              = 74
1 FPAMOFS  EQU   X'004B'     OVERLAPPING_FASTPAM_SPACE          = 75
1 FPAMALNA EQU   X'004C'     ACCESS_LISTS_NOT_ALLOCATED         = 76
1 FPAMNPAF EQU   X'004D'     NO_PAM_FILE                        = 77
1 FPAMLBIN EQU   X'004E'     LAST BLOCK BUT INPUT               = 78
1 FPAMRFA  EQU   X'0050'     RFA_NOT_SUPPORTED                  = 80
1 FPAMSPD  EQU   X'0051'     SPD_NOT_SUPPORTED                  = 81
1 FPAMPPD  EQU   X'0052'     PPD_NOT_SUPPORTED                  = 82
1 FPAMTAPE EQU   X'0053'     TAPE_NOT_SUPPORTED                 = 83
1 FPAMDUMM EQU   X'0054'     DUMMY_FILE_NOT_SUPPORTED           = 84
1 FPAMSECA EQU   X'0055'     SEC_ALLOCATION_TOO_SMALL           = 85
1 FPAMTPRE EQU   X'0056'     TPR_ENVIRONMENT_OR_IOAREA_POOL     = 86
1 FPAMIPAL EQU   X'0057'     IOAREA_POOL_NOT_ALLOCATED          = 87
1 FPAMBLKN EQU   X'0058'     BLKCTRL_NOT_SUPPORTED              = 88
1 FPAMTPRO EQU   X'0059'     TPR_OPEN_ID                        = 89
1 FPAMWRCN EQU   X'005A'     WRCHK_NOT_SUPPORTED                = 90
1 FPAMMPNF EQU   X'005B'     MEMORY_POOL_NOT_FIXED              = 91
1 *
```

```
1 * &P.PAM FUNCTIONS:
1 *
1 FPAMENEV EQU   1           ENABLE ENVIRONMENT
1 FPAMDIEV EQU   2           DISABLE ENVIRONMENT
1 FPAMENIP EQU   3           ENABLE IOAREA POOL
1 FPAMDIIP EQU   4           DISABLE IOAREA POOL
1 FPAMOPEN EQU   5           OPEN FILE
1 FPAMCLOS EQU   6           CLOSE FILE
1 *
1 * INPUT/OUTPUT PARAMETER
1 *
1 FPAMENID DS    F           ENVIRONMENT SHORT-ID
1 FPAMIPID DS    F           IOAREA POOL SHORT-ID
1 FPAMOPID DS    F           OPEN SHORT-ID
1 *
1 * OUTPUT PARAMETER
1 *
1 FPAMDMSC DS    XL4         DMS-CODE
1 *
1 * INPUT PARAMETER
1 *
1 FPAMENNA DS    CL54        ENVIRONMENT NAME
1 FPAMIPNA DS    CL54        IOAREA POOL NAME
1 FPAMLINK DS    CL8         LINK
1 FPAMFILE DS    CL54        FILENAME
1 FPAMLARF DS    AL1         LARGE_FILE
1 FPAMFRBD EQU   0           LARGE_FILE=FORBIDDEN
1 FPAMALWD EQU   1           LARGE_FILE=ALLOWED
1 FPAMENV  DS    AL1         ENV
1 FPAMHOST EQU   1           ENV=HOST
1 FPAMXCS  EQU   2           ENV=XCS
1 FPAMACLA DS    A           ADDRESS OF ACCESS LISTS
1 FPAMACLN DS    F           NUMBER OF ACCESS LISTS
1 FPAMOFF  DS    A           ADDR. OF IOAREA POOL WITHIN ADDRESS SPACE
1 FPAMALET DS    F           ALET OF DATA SPACE
1 FPAMIPS  DS    F           SIZE OF IOAREA POOL (4K)
1 FPAMEID  DS    F           EVENT-ITEM SHORT-ID
1 FPAMLABL DS    F           LAST BLOCK NUMBER
1 FPAMMODE DS    AL1         OPEN MODE
1 FPAMINPT EQU   1           MODE=INPUT
1 FPAMINOT EQU   2           MODE=INOUT
1 FPAMOUTI EQU   3           MODE=OUTIN
1 FPAMSUPD DS    AL1         SHARE UPDATE
1 FPAMSUNO EQU   1           SHARUPD=NO
1 FPAMSUYE EQU   2           SHARUPD=YES
1 FPAMMAXL DS    AL1         MAXIMUM IO-LENGTH
1 FPAMMINS EQU   0           IO-LENGTH NOT SPECIFIED
1 FPAMMINI EQU   1           IO-LENGTH=4K
1 FPAMMAXI EQU   8           IO-LENGTH=32K
1 FPAMEVEN DS    AL1         EVENTING
1 FPAMEVNS EQU   0           EVENTING=NOT_SPECIFIED
1 FPAMEVNO EQU   1           EVENTING=NO
1 FPAMEVYE EQU   2           EVENTING=YES
1 FPAMRESI DS    AL1         RESIDENT
1 FPAMRENS EQU   0           RESIDENT=NOT_SPECIFIED
1 FPAMRENO EQU   1           RESIDENT=NO
1 FPAMREYE EQU   2           RESIDENT=YES
1 FPAMBLS  DS    FL1         BLOCKSIZE
1 FPAM#    EQU   *-FPAMPA LENGTH of PARAMETERAREA
```

FSTAT - Request catalog information

Macro type: type S (E form, L form/C form/D form); see "Macro types"

The FSTAT macro returns information on catalog entries. The user can request information on one or more files, on file generations, or on file generation groups, as well as on all files under a specified user ID.

Users may retrieve information on all files under their own user IDs and on all other users' files that they are permitted to access (see the selection operands SHARE, BASACL, OWNERAR, GROUPAR, OTHERAR, GUARDS and PROTACT).

The selection of files for which the user requires information can be made as follows:

- By specifying the path name. The catalog ID, user ID, and the fully or partially qualified file name (with or without wildcards) serve as selection criteria.If no path name is specified, all permanent files of the user's own user ID are selected from the standard catalog of the local computer. Temporary files must be addressed with the tempfile prefix (# or @).
- The files selected via the “pathname” can be restricted further by the user by means of selection operands. Only those files which have the file attributes described by the selection operands are selected. If no selection operand is specified or if the value ANY (if allowed) is specified, the corresponding file attribute is not taken into account for the selection.

The scope and structure of the information to be retrieved for the selected files and transferred to the output area can be defined by the user in the OUTPUT operand as follows:

- only the return information concerning macro execution (no information in the output area)
- only the names of the selected files
- statistics (e.g. number of selected files for each type of volume)
- catalog information for each selected file.

The output of catalog information can be restricted by the user to one or more information blocks in which file attributes are grouped into logical units (CEINFO operand):

- history block
- security block
- status block
- backup block
- organization block
- allocation block
- volumes block
- volume extents block
- index-info-block
- FTAM information

Default value: all information blocks (without FTAM information)

The following sources of information can be selected by the user (see the FROM operand):

- the standard catalog of the user ID

- all catalogs of the local pubset
- VTOC of a private disk or of a Net-Storage volume

The FSTAT macro is supported in six versions (see also VERSION operand):

- Version 5 for BS2000 versions >= BS2000 OSD-BC V11.0
- Version 4 for BS2000 versions >= BS2000/OSD-BC V9.0
- Version 3 for BS2000 versions >= BS2000/OSD-BC V3.0
- Version 2 for BS2000 versions >= BS2000/OSD-BC V1.0
- Version 1 (corresponds to version 800) for BS2000 versions >= V8.0
- Version 0 (corresponds to version 710) for BS2000 versions < V8.0

Default setting: Version 0

This description of the FSTAT macro is based on the functionality of Version 5, which must be explicitly specified in the macro as VERSION=5. In the table "Version differences - VERSION=0/1/2/3/4/5", all deviations to the described version are combined for versions 0 to 5.

Format

Operation	Operands
FSTAT	**[pathname]** **[,ACCCNT = ANY / nmbr / (nmbr[,]) / (,nmbr) / (nmbr1,nmbr2)]** **[,ACCESS = ANY / READ / WRITE]** **[,ADMINFO = *ANY / *NONE / <c-string 1..8>]** **[,AVAIL = *ANY / *STD / *HIGH]** **[,BACKUP = ANY / A / B / C / D / E / (list-of-backup)]** **[,BASACL = ANY / NONE / YES]**
	[,BLKCNT = nmbr / (nmbr[,]) / (,nmbr) / (nmbr1,nmbr2)] **[,BLKCTRL = ANY / PAMKEY / DATA4K / DATA2K / DATA / NO / NONE / NK4 / NK2 / (list-of-blkctrl)]** **[,CCS = *ANY / *NONE / ccs-name]** **[,CEINFO = ALL / ALLOCATION / BACKUP / FTAM / HISTORY / INDEX-INFO / ORGANIZATION / SECURITY / STATUS / VOLUMES / VOLUME-EXTENTS / (list-of-ceinfo)]** **[,CRDATE = *ANY / *NONE / date / date(time[,]) / date(time1,time2) / (date[,]) / (date(time)[,]) / (,date) / (,date(time)) / (date1,date2) / (date1(time),date2) / (date1(time),date2(time))]**

```
[,DELDATE = *ANY / *NONE / date / date(time[,]) / date(time1,time2) /
            (date[,]) / (date(time)[,]) / (,date) / (,date(time)) /
            (date1,date2) / (date1(time),date2) /
            (date1(time),date2(time))]

[,DISKWR = ANY / IMMEDIATE / BY-CLOSE]

[,ENCRYPT = *ANY / *NONE / *AES / *DES / (list-of-encrypt)]

[,EXDATE = *ANY / *NONE / date / date(time[,]) / date(time1,time2) /
           (date[,]) / (date(time)[,]) / (,date) / (,date(time)) /
           (date1,date2) / (date1(time),date2) /
           (date1(time),date2(time))]

[,EXTENTS = ANY / nmbr / (nmbr[,]) / (nmbr1,nmbr2)]

[,FCBTYPE = ANY / ISAM / BTAM / SAM / PAM / NONE / (list-of-fcbtype)]

[,FILTYPE = *ANY / *BS2000 / *NODE]

[,FROM = CATALOG / LOCALPVS / (vsn,device)]

[,FSIZE = ANY / SIZE / nmbr / (nmbr[,]) / (nmbr) / (nmbr1,nmbr2)]

[,GEN = NO / YES]

[,GROUPAR = ANY / NO-ACCESS / access-list]

[,GUARDS = *ANY / *NONE / *YES /
           ([READ = *ANY / *NONE / fname]
            [,WRITE = *ANY / *NONE / fname]
            [,EXEC = *ANY / *NONE / fname])]

[,IOPERF = ANY / STD / HIGH / VERY-HIGH / (list-of-ioperf)]

[,IOUSAGE = ANY / RDWRT / WRITE / READ / (list-of-iousage)]

[,LADATE = *ANY / *NONE / date / date(time[,]) / date(time1,time2) /
           (date[,]) / (date(time)[,]) / (,date) / (,date(time)) /
           (date1,date2) / (date1(time),date2) /
           (date1(time),date2(time))]

[,LASTPAG = ANY / nmbr / (nmbr[,]) / (,nmbr) / (nmbr1,nmbr2)]

[,LBPOINT = *ANY / *NO / *YES / *ZERO / *NONZERO]

[,LCDATE = *ANY / *NONE / date / date(time[,]) / date(time1,time2) /
           (date[,]) / (date(time)[,]) / (,date) / (,date(time)) /
          (date1,date2) / (date1(time),date2) /
          (date1(time),date2(time))]

[,MANCLAS = *ANY / *NONE / <c-string 1..8>]

[,MIGRATE = ANY / ALLOWED / INHIBIT / FORBIDDEN / (list-of-migrate)]

[,OTHERAR = ANY / NO-ACCESS / access-list]

[,OUTAREA = (<list-of-elements-002>)]
```

```
[,OUTPUT = RC-ONLY / CEINFO / FNAM-ONLY / STAT-LONG / STAT-SHORT / STAT-
INFO]

[,OWNERAR = ANY / NO-ACCESS / access-list]

[,PASS = ANY / NONE / EXPASS / RDPASS / WRPASS / (list-of-pass)]

[,PASSW = NO / YES]

[,PREFORM = *ANY / *NONE / *K / *NK2 / *NK4 / (list-of-preform)]

[,PROTACT = ANY / LEVEL-0 / LEVEL-1 / LEVEL-2 / (list-of-protact)]

[,RELSPAC = ANY / ALLOWED / IGNORED]

[,SHARE = ANY / YES / NO / SPECIAL / (list-of-share)]

[,SIZE = ANY / FSIZE / nmbr / (nmbr[,]) / (,nmbr) / (nmbr1,nmbr2)]

[,SLEVEL = ANY / S0 / S1 / S2 / (list-of-slevel)]

[,SORT = FILENAM / NO]

[,STATE = ANY / NOCLOS / CLOSED / CACHED / NOT-CACHED / CACHE-NOT-SAVED /
          OPEN-ALLOWED / NO-OPEN-ALLOWED / REPAIR-NEEDED /
          DEFECT-REPORTED / (list-of-state)]

[,STOCLAS = *ANY / *NONE / <c-string 1..8>]

[,STOTYPE = *ANY / *PUBSPACE / *NETSTOR]

[,STOUTAR = (<list-of-elements-002>)]

[,SUPPORT = ANY / PUBLIC / PRDISC / TAPE / (list-of-support)]

[,S0MIGR = *ANY / *ALLOWED / *FORBIDDEN / (list-of-s0migr)]

[,TIMBASE = *UTC / *LTI]

[,TYPE = ANY / FILE / FGG / PLAM / (list-of-type)]

[,USRINFO = *ANY / *NONE / <c-string 1..8>]

[,VERSION-BACKUP = *ANY / *ENABLED / *DISABLED]

[,VOLSET = *ANY / *CONTROL / <c-string 1..4>]

[,VOLUME = *ANY / vsn]

[,VTOC = NO / YES]

[,WORKFIL = *ANY / *NO / *YES]

[,WTQUIET = *YES / *NO]

[,XPAND = PLSHORT / PLLONG / OUTPUT / (PLSHORT,OUTPUT) / (PLLONG,OUTPUT)]

[,MF = L]
,VERSION = 0 / 1 / 2 / 3 / 4 / 5
[,PREFIX = pre]
```

```
MF = (E,adr / E,(r))
,VERSION = 0 / 1 / 2 / 3 / 4 / 5
```

```
[,MF = C / D
[,VERSION = 0 / 1 / 2 / 3 / 4 / 5]
[,PREFIX = pre]
[,XPAND = PLSHORT / PLLONG / OUTPUT / (PLSHORT,OUTPUT) / (PLLONG,OUTPUT)]
```

Operand descriptions

pathname

“pathname” defines the path name of the file(s) for which information is to be returned with: <filename 1..54 with-wild (80) without-gen>

Temporary files are not taken into account.

“pathname” means [:catid:][$userid][filename]

catid
Catalog ID;
Default value: the catalog ID assigned to the user ID

userid

User ID;
“$userid.” designates all files of this user. If the user ID of some other user is specified, information is returned on only those files which the caller of the macro is allowed to access.
Default value: the user's own user ID i.e. the one specified in the SET-LOGON-PARAMETERS or LOGON command.

filename
Fully or partially-qualified file name of permanent or temporary files, of file generations or file generation groups.

Wildcard specification
Nonprivileged users may only specify wildcards in “catid” and “filename” whereas the system administrator may also specify them in “userid” (for information on wildcards see "Wildcards").
Wildcards cannot replace the delimiters in the cat (colons) and user ($ sign and period) name parts.

ACCCNT

Returns information on all files that were accessed as often as specified.
The access counter can be assigned values from 0 to 2147483647.

= ANY
The access counter is not a selection criterion.

= nmbr
Returns information on files for which the access count exactly matches the specified value.

= (nmbr[,])
Returns information on files for which the access count is greater than or equal to the specified value.

= (,nmbr)
Returns information on files for which the access count is less than or equal to the specified value.

= (nmbr1,nmbr2)
Returns information on files whose access counters lie within the specified interval (nmbr1 <= access counter <= nmbr2).

ACCESS

Selects files/file generations on the basis of the access mode.

= ANY
The access mode is not a selection criterion.

= READ
Provides information about files/file generations for which only read access is permitted.

= WRITE
Provides information about files/file generations for which only write access is permitted.

ADMINFO

Returns information on files/file generations dependent on the system administrator metainformation.

= *ANY
The system administrator metainformation is not a selection criterion.

= *NONE
Returns information on files possessing no system administrator metainformation.

= <c-string 1..8>
Returns information on files with the specified system administrator metainformation.

AVAIL

Returns information on files/file generations dependent on their availability.

= *ANY
The availability is not a selection criterion.

= *STD
Returns information on files that are not on a volume set or an SF pubset with high availability.

= *HIGH
Returns information on files that are on disks set with high availability (DRV pubset).

BACKUP

Returns information on files/file generation groups, for which the specified ARCHIVE or HSMS backup level was defined.

= ANY
The backup level is not a selection criterion.

= A
Returns information on files/ FGGs with the attribute BACKUP=A.

= B
Returns information on file/FGGs with the attribute BACKUP=B.

= C
Returns information on files/FGGs with the attribute BACKUP=C.

= D
Returns information on files/FGGs with the attribute BACKUP=D.

= E
Returns information on files/FGGs with the attribute BACKUP=E.

= (list-of-backup)
More than one backup level may be specified in the form of a list. All files/FGGs which satisfy one of the given conditions (ORing) will be selected in such cases.

BASACL

Returns information on files which are selected on the basis of a defined BASIC-ACL.

= ANY
The BASIC-ACL is not a selection criterion.

= NONE
Returns information on all files for which no BASIC-ACL entry is defined.

= YES
Returns information on all files for which a BASIC-ACL entry is defined.

BLKCNT

For tape files only:
Selects files on the basis of the number of blocks on tape.

= ANY
The number of blocks on tape is not a selection criterion.

= nmbr
Returns information on all tape files with exactly the specified number of blocks.

= (nmbr[,])
Returns information on all tape files for which the number of blocks is greater than or equal to the specified value.

= (,nmbr)
Returns information on all tape files for which the number of blocks is less than or equal to the specified value.

= (nmbr1,nmbr2)
Returns information on all tape files for which the number of blocks lies within the specified interval.

Any integers from the range 0 <= value <= 2147483647 may be specified.

BLKCTRL

Selects files on the basis of the block format with which the file was defined (via the BLKCTRL operand in the FILE or FCB macro). The block format is defined when creating the file and is based on the existence and position of the block control field that contains management information for the PAM page.

= ANY
The file format is not a selection criterion.

= DATA
Returns information on all files for which the block control information is located at the start of the data block. Such files are created with BLKCTRL=DATA (see the FILE macro).

= PAMKEY
Returns information on all files which use a separate PAM key for the block control field, i.e. files for which the block control information is stored in a special key field outside the PAM block. Such files are created with BLKCTRL=PAMKEY (see the FILE macro).

= NO
Returns information on all files which contain no block control field. Such files are created with BLKCTRL=NO (see the FILE macro).

= NONE
Returns information on all files for which no BLKCTRL value was defined, i.e. files which have not yet been opened.

= DATA2K
Returns information on all files which were created with BLKCTRL=DATA2K (see the FILE macro).

= DATA4K
Returns information on all files that were created with BLKCTRL=DATA4K (see the FILE macro).

= NK2
Returns information on all NK2 files (files which can be stored on NK2 volumes).

= NK4
Returns information on all NK4 files (files which can be stored on NK4 volumes).

= (list-of-blkctrl)
Returns information on all files that match one of the specified block formats. All values except ANY may be specified in a list.

CCS

Returns information on files selected on the basis of the specified coded character set.The coded character set (CCS) defines how the characters of a national character set are to be stored in binary form. The specified character set has an effect on the representation of characters on the screen, the collating sequence, etc. (see the “XHCS” manual [22]).

= *ANY
The code table is not a selection criterion.

= *NONE
Only files for which no character set is defined are selected.

= ccs-name
Only files with the specified character set are selected.

CEINFO

The information from the catalog is arranged logically into information blocks. Only the information blocks explicitly selected by the user are output.

The CEINFO operand can be used to control the generation of a DSECT or CSECT.
If CEINFO is not specified or if CEINFO=ALL is set, all output blocks (except for the FTAM block) are generated. Otherwise, only those output blocks which are specified in CEINFO are generated.

= ALL
Transfers all the information stored in the catalog to the output area (OUTAREA) for the selected files.
The information shown is divided into the following information blocks: HISTORY / SECURITY / STATUS / BACKUP / ORGANIZATION / ALLOCATION / VOLUMES / VOLUME-EXTENTS (or INDEX-INFO)

File transfer information (FTAM) cannot be output by using CIENFO=ALL. This information, which is only relevant for file transfers, is output with CEINFO=FTAM only.

= HISTORY
Transfers the history block to the output area for the selected files, i.e. all file attributes related to the file history (on UTC basis):

- access counter
- date of last access
- time of last access
- date of last write access
- time of last write access
- creation date
- time of creation
- number of secondary allocations

= SECURITY
Transfers the security block to the output area for all selected files, i.e. all file attributes that affect file security (the bytes intended for file passwords are set to binary 0.
Exception: See the PASSW operand):

- type of access (standard access control)
- file monitoring
- automatic data destruction on deletion
- protection with an execute password
- date on which the file may be updated
- time related to EXPIR-DATE
- protection against read access by a guard
- protection against write access by a guard
- protection against execute access by a guard
- access rights of user class “group” (BASIC-ACL)
- access rights of user class “others” (BASIC-ACL)
- access rights of file owner (BASIC-ACL)
- protection with read password
- protection against release of storage space
- shareability attribute (standard access control)
- protection with write password

= STATUS

Transfers the status block for all selected files to the output area, i.e. all file attributes that affect special characteristics of the file. These include:

- SPECIAL-ACCOUNTING-BIT
- OPEN-CLOSE flag
- REPAIR flag
- PSEUDO-CLOSE flag
- VERIFY-IS-FORBIDDEN flag
- LOCKS (RELEASE-LOCK; ERASE-LOCK; OUTPUT-LOCK; CATALOG-LOCK; SPD-LOCK)

= BACKUP

Transfers the backup block for all selected files to the output area, i.e. all the backup attributes of files. These include:

- the backup level for ARCHIVE or HSMS
- the specification regarding whether the file may be migrated
- the specification regarding whether the file is always to be fully saved
- the version as an internal ARCHIVE attribute
- the number of file versions that the file takes part in the version backup with

= ORGANIZATION

Transfers the organization block for all selected files to the output area, i.e. all file attributes that are related to the file organization:

- block counter (tape files)
- buffer offset (tape files)
- block type (standard or nonstandard block)
- coded character set (CCS) with XHCS support
- coded character set for node files on Net-Storage (NETCCS)
- code specification for tape files
- suitability of file for processing in a cache
- file sequence number (tape file)
- access method when the file was created
- performance requirement for file processing
- type of I/O operations to which the performance requirement applies
- length of ISAM key
- position of ISAM key
- standard version of labels (tape files)
- length of logical ISAM flag
- propagation of ISAM flag
- record format
- record length
- length of the ISAM value flag
- file type on Net-Storage
- file size of node files

= ALLOCATION
Transfers the allocation block for the selected files to the output area, i.e. all file attributes that affect storage space allocation:

- device type for the volume
- total number of extents for the file
- highest used PAM page (last-page pointer)
- secondary allocation for file extension
- storage level for migrated files
- type of volume
- VSN (volume serial number) of the volume used
- last valid byte on the last logical page of the file (last-byte pointer and validity indicator for the last-byte pointer)

= VOLUMES
Transfers the volume tables for the selected files to the output area. These tables contain the following information:

- VOLUME
- DEVICE
- #EXTENTS
- flag indicating whether an ISAM-INDEX or ISAM-DATA volume is involved

= VOLUME-EXTENTS
Transfers the volume tables and the extent lists for the selected files to the output area.

= INDEX-INFO
Transfers the INDEX-INFO block for the selected files to the output area, i.e. all file attributes that affect the file generation group:

- base value for relative generation numbers
- first or last cataloged file generation
- oldest existing file generation
- maximum number of simultaneously cataloged generations
- overflow handling on reaching the maximum number

= FTAM
Transfers the FTAM block for the selected files to the output area, i.e. all file attributes related to file transfer. Since this output block is only relevant for file transfer, the FT block is not output for CEINFO=ALL.

= (list-of-ceinfo)
With the exception of FTAM and ALL, all operand values may be specified in a list. The corresponding information blocks are then transferred to the output area. The order in which the operands are specified is of no consequence.

CRDATE

Allows the user to select files for processing on the basis of their creation dates.

Date values may be supplemented by specifying a time. The rules for date and time specifications are described in section "Format of date specifications". Range specifications are inclusive of both specified limits.

= ANY
The creation date is not a selection criterion.

= NONE
Returns information on all files for which no creation date has been entered in the catalog, i.e. files which have not yet been opened.

= date
Returns information on all files that were created on the specified date.

= (date[,])
Returns information on all files that were created on or after the specified date (creation date >= current date).

= (,date)
Returns information on all files that were created on or before the specified date (creation date <= current date).

= (date1,date2)
Returns information on all files that were created within the specified period (date1 <= creation date <= date2).

= date(time[,])
Returns information on all files that were created on the specified date on or after the specified time.

= date(time1,time2)
Returns information on all files that were created on the specified date within the specified period.

= (date(time)[,])
Returns information on all files that were created on or after the specified date and time.

= (,date(time))
Returns information on all files that were created before the specified date and time.

= (date1(time),date2(time))
Returns information on all files that were created within the specified period. The upper and lower limits of the specified period are defined more precisely by a time specification in both cases.

DELDATE

Allows the user to select files on the basis of the DELETION-DATE (the time as of which the file may be deleted irrespective of its protection attributes).

The user can supplement the date values by specifying a time. It must be noted in this respect that the deletion time of 00:00:00 is currently always entered in the file catalog.The rules for date and time specifications are described in section "Format of date specifications". Range specifications include both specified limits.

= *ANY
The DELETION-DATE is not a selection criterion.

= *NONE
Returns information on all files for which no DELETION-DATE is entered in the catalog.

= date
Returns information on all files for which the specified DELETION-DATE is defined.

= (date[,])
Returns information on all files whose DELETION-DATE is later than or equal to the specified date.

= (,date)
Returns information on all files whose DELETION-DATE is earlier than or equal to the specified date.

= (date1,date2)
Returns information on all files whose DELETION-DATE lies within the specified time period (date1 <= DELETION-DATE <= date2).

= date(time[,])
Returns information on all files for which the specified DELETION-DATE is defined and for which the delete time is later then or equal to the specified time. The delete time (time in relation to the DELETION-DATE) is currently always entered in the catalog as 00:00:00.

= date(time1,time2)
Returns information on all files for which the specified DELETION-DATE is defined and for which the delete time is within the specified time period. The delete time (time in relation to the DELETION-DATE) is currently always entered in the catalog as 00:00:00.

= (date(time)[,])
Returns information on all files whose DELETION-DATE and time is later than or equal to the specified time. The delete time (time in relation to the DELETION-DATE) is currently always entered in the catalog as 00:00:00.

= (,date(time))
Returns information on all files whose DELETION-DATE and time is earlier than or equal to the specified time. The delete time (time in relation to the DELETION-DATE) is currently always entered in the catalog as 00:00:00.

= (date1(time),date2(time))
Returns information on all files whose DELETION-DATE lies within the specified time period (date1 <= DELETION-DATE <= date2). The upper and lower limits of the specified time period are defined more precisely by specifying a time.

DISKWR

Enables the user to select files for processing based on the time at which data consistency is required for them, as defined in the catalog entry.

= ANY
The time at which data consistency is required, as defined in the catalog, is not a selection criterion.

= IMMEDIATE
Returns information on all files for which data consistency is required immediately after a write operation is completed. Such files are not suitable for processing in a write cache.

= BY-CLOSE
Returns information on all files for which data consistency is not required until CLOSE processing. These files are suitable for processing in a write cache.

ENCRYPT

Enables the user to select files based on whether or with which encryption method they are encrypted.

= *ANY
Processes all files regardless of whether or with which encryption method they are encrypted.

= *NONE
Processes only those files which are not encrypted.

= *AES
Processes only those files which are encrypted with the AES encryption method.

= * DES
Processes only those files which are encrypted with the DES encryption method.

= (list_of_encrypt)
Processes only those files which meet one of the specified selection criteria. All values except ANY can be specified in the list.

EXDATE

The user can select files to be processed on the basis of their expiration date. The expiration date of a file is defined in the catalog and specifies when the file may be updated again or deleted. If no expiration date is defined when creating the file, the expiration date is set to the creation date.

The user may supplement date specifications by a time value; however, it should be noted that the time stamp for the expiration date is always set to 00:00:00 in the file catalog at present. The rules for date and time specifications are described in section "Format of date specifications". Ranges defined in intervals include both specified limits.

= ANY
The expiration date is not a selection criterion.

= NONE
Returns information on all files for which no expiration date has been entered in the catalog, i.e. files that have not yet been opened.

= date
Returns information on all files for which the specified expiration date is defined.

= (date[,])
Returns information on all files for which the expiration date is greater than or equal to the specified date.

= (,date)
Returns information on all files for which the expiration date is less than or equal to the specified date.

= (date1,date2)
Returns information on all files for which the expiration date lies within the specified period (date1 <= expiration date <= date2).

= date(time[,])
Returns information on all files with the specified expiration date and with an expiration time that is greater than or equal to the specified time.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= date(time1,time2)
Returns information on all files with the specified expiration date and with a time of expiration that lies within the specified time interval.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= (date(time)[,])
Returns information on all files for which the expiration date and time is greater than or equal to the specified time.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= (,date(time))
Returns information on all files for which the expiration date and time is less than or equal to the specified time.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= (date1(time),date2(time))
Returns information on all files for which the expiration date lies within the specified period (date1 <= expiration date <= date2). The upper and lower limits of the specified period are defined more precisely by time values in both cases.

EXTENTS

Only for files on disk or Net-Storage:
returns information on files selected on the basis of the specified number of extents. An extent is a contiguous area occupied by a file on a disk. The number of extents which make up a file is stored in the catalog.

The possible values for “nmbr” are: 0 <= nmbr <= 65535.

Range specifications include both upper and lower limits.

= ANY
The number of extents is not a selection criterion.

= nmbr
Only the disk files with exactly the specified number of extents are processed.

= (nmbr [,])
Only the disk files with at least the specified number of extents (number of extents >= nmbr) are processed.

= (,nmbr)
Only the disk files that have at most the specified number of extents (number of extents <= nmbr) are processed.

= (nmbr1, nmbr2)
Only the disk files that have at least as many extents as “nmbr1” and at most as many extents as “nmbr2” (i.e. nmbr1 <= number of extents <= nmbr2) are processed.

FCBTYPE

Selects and returns information on files/file generations on the basis of the access method with which they were created. If multiple access methods are listed, all files that satisfy one of the listed conditions (logical OR) are selected by the system.

= ANY
The access method is not a selection criterion.

= NONE
Returns information on files that are cataloged but which contain no data, i.e. files which have not yet been opened or files for which storage space has been released with ERASE..., SPACE.

= ISAM
Returns information on ISAM files.

= BTAM
Returns information on BTAM files.

= SAM
Returns information on SAM files.

= PAM
Returns information on PAM files.

= (list-of-fcbtype)
More than one access method may be specified in a list. Information on all files that were created with one of the specified access methods will be shown.

FILTYPE

Returns information on all files which match the specified file type.

= *ANY
Returns information on both BS2000 files and on node files.

= *BS2000
Returns information only on BS2000 files.

= *NODE
Returns information only on node files (files which can be created and modified by BS2000 and open systems).

FROM

The FROM operand defines the source for the FSTAT information.

= CATALOG
The FSTAT macro retrieves information from the catalog of the default pubset of the user ID, i.e. from the catalog with the default catalog ID.

= LOCALPVS
The FSTAT macro retrieves information from the system catalogs of all selected local pubsets.

= (vsn,device)
The FSTAT macro retrieves information from the directory of the private disk identified by “vsn” or from the catalog of the Net-Storage volume identified by “vsn”.
The device type (“device”) of the private disk must be specified; see the “Device table” in the „System installation“ manual [16] for possible values. The operands VOLUME, SUPPORT and VTOC are not permitted.

FSIZE

For disk files only:
Returns information on files/file generation groups based on the amount of free (i.e. reserved but not used) storage space; 0 <= nmbr <= 16777215

= ANY
The size of the free (i.e. reserved but not used) storage space is not a selection criterion.

= SIZE
Returns information on files for which the number of free PAM pages is equal to the number of reserved pages.

= nmbr
Returns information on files with exactly the specified number of reserved but unused PAM pages.

= (nmbr[,])
Returns information on files with at least the specified number of reserved but unused PAM pages.

= (,nmbr)
Returns information on files with no more than the specified number of reserved but unused PAM pages.

= (nmbr1,nmbr2)
Returns information on files for which the number of free pages lies within the specified range (nmbr1 < nmbr2).

GEN

The “GEN” operand defines whether information on file generations is to be output.The interaction of this operand with the specification TYPE=FGG is illustrated in the following table:

TYPE=FGG	GEN=YES	GEN=NO	Information on FGG	Information on file generations	Information on files
x	x		*	*	-
x		x	*	-	-
	x		*	*	*
		x	*	-	*

x specified in the FSTAT macro

* taken into account for macro processing

\- not taken into account for macro processing

= NO
No information on file generations is output.

= YES
File generation information is output.

The specification GEN=YES is ignored if no “filename” is specified in “pathname”.

GROUPAR

Selects and returns information on files on the basis of the access rights that are defined for members of the file owner's user group in BASIC-ACL entries.

= ANY
The BASIC-ACL entries for members of the file owner's user group are not used as a selection criterion.

= NO-ACCESS
Returns information on all files that cannot be accessed by the user group of the owner.

= access-list
Returns information on all files for which at least one of the access rights specified in the list has been entered for the user group of the file owner in the BASIC-ACL entry.

access-list has the following format:

- Long format:
  ([READ=YES / READ=NO][,WRITE=YES / WRITE=NO][,EXEC=YES / EXEC=NO])

- Short format:
  ([R=Y / R=N][,W=Y / W=N][,X=Y / X=N])

The parentheses constitute part of the operand value and must be specified.

The individual elements of the access list mean the following:

READ=YES or R=Y	Selects all files that may be accessed for reading by the user group of the owner.
READ=NO or R=N	Selects all files that cannot be accessed for reading by the user group of the owner.
WRITE=YES or W=Y	Selects all files that may be accessed for writing by the user group of the owner.
WRITE=NO or W=N	Selects all files that cannot be accessed for writing by the user group of the owner.
EXEC=YES or X=Y	Selects all files that may be executed by the user group of the owner.
EXEC=NO or X=N	Selects all files that cannot be executed by the user group of the owner.

GUARDS

The user can select files to be processed on the basis of the access protection defined by GUARDS (see the “SECOS” manual [8]).

= *ANY
The access protection defined by GUARDS is not a selection criterion.

= *NONE
Returns information on all files which have no access protection defined by GUARDS.

= *YES
Returns information on all files which have access protection defined by GUARDS.

= (READ=...,WRITE=...,EXEC=...)
The type of access protection provided by GUARDS that is to be used as a selection criterion can be defined by the user in a list. For each access mode (read, write, and execute), the defined protection can be specified precisely. If no entry is made for an access mode, the protection defined for that access mode has no effect on the selection.

For each access mode, one of the following values may be specified:

*ANY The defined GUARDS protection is not a selection criterion.

*NONE No guard has been defined for the specified access mode, i.e. the corresponding access is denied.

fname All conditions for granting access in the specified access mode are defined in the guard fname.

IOPERF

Performance attribute of the file; returns information on all files based on the performance attribute that is defined for them in the catalog (see IOPERF operand in the CATAL macro).

= ANY
The performance attribute is not a selection criterion.

= STD
Returns information on all files for which the performance attribute was defined as STD.

= HIGH
Returns information on all files for which the performance attribute was defined as HIGH (high performance priority).

= VERY-HIGH
Returns information on all files for which the performance attribute was defined as VERY-HIGH (highest performance priority).

= (list-of-ioperf)
Up to three performance attributes may be specified in a list. All files that have one of these specified attributes will be shown.

IOUSAGE

Returns information on all files, depending on the type of I/O operations to which the performance attribute applies (see the IOUSAGE operand in the CATAL macro).

= ANY
The performance attribute is not a selection criterion.

= RDWRT
Returns information on all files for which the performance attribute applies to read and write operations.

= WRITE
Returns information on all files for which the performance attribute applies to write operations only.

= READ
Returns information on all files for which the performance attribute applies to read operations only.

= (list-of-iousage)
More than one type of I/O operation may be specified in a list. All files for which the performance attribute applies to at least one of the specified I/O operations will be shown.

LADATE

Returns information on all files with the corresponding date of last access.

The user can supplement date specifications by specifying time values. The rules for date and time specifications are described in section "Format of date specifications".
Ranges specified in intervals include both upper and lower limits.

= ANY
The date of last access is not a selection criterion.

= NONE
Returns information on all files for which no last access date has been entered in the catalog, i.e. files that have not yet been opened.

= date
Selects files which were last accessed on the specified date.

= (date[,])
Returns information on all files that were last accessed on or after the specified date (last access date >= date).

= (,date)
Selects all files that were accessed on or before the specified date (last access date <= date).

= (date1,date2)
Returns information on all files that were last accessed during the specified period (date1 <= last access date <= date2).

= date(time[,])
Returns information on all files that were last accessed on the specified date on or after the specified time.

= date(time1,time2)
Selects all files that were last accessed on the specified date and within the specified period.

= (date(time)[,])
Selects all files that were last accessed on or after the specified date and time.

= (,date(time))
Selects all files that were last accessed before the specified date and time.

= (date1(time),date2(time))
Selects all files that were last accessed within the specified period. The upper and lower limits of the specified period are defined more accurately by means of a time specification.

LASTPAG

Selects and returns information on files based on the amount of storage space used (i.e. the number of PAM pages written).

= ANY
The amount of storage space used is not a selection criterion.

= nmbr
Returns information on all files for which exactly the specified number of PAM pages have been written.

= (nmbr,)
Returns information on all files for which at least the specified number of PAM pages have been written.

= (,nmbr)
Returns information on all files for which no more than the specified number of PAM pages have been written.

= (nmbr1,nmbr2)
Returns information on all files for which the number of written PAM pages lies in the specified area (nmbr1 < nmbr2).

LBPOINT

Informs about files depending on of the attribute of the last-byte pointer (LBP).

= *ANY
The last-byte pointer is not a selection criterium.

= *NO
Informs about files without valid last-byte pointer.

= *YES
Informs about files with valid last-byte pointer.

= *ZERO
Informs about files with valid last-byte pointer with the value zero.

= *NONZERO
Informs about files with valid last-byte pointer with another value than zero.

LCDATE

Returns information on all files based on the date on which they were last accessed for writing.

The user can supplement date specifications by means of time values. The rules for date and time specifications are described in section "Format of date specifications". Ranges defined by intervals include both the specified limits.

= ANY
The date of the last write access is not a selection criterion.

= NONE
Returns information on all files for which no date of last write access is entered in the catalog, i.e. files that have not yet been opened.

= date
Returns information on all files that were last written to (i.e. changed) on specified date.

= (date[,])
Returns information on all files that were last changed on or after the specified date (last change date >= date).

= (,date)
Returns information on all files that were last changed on or before the specified date (last change date <= date).

= (date1,date2)
Returns information on all files that were last changed during the specified period (date1 <= last change date <= date2).

= date(time[,])
Returns information on all files that were last changed on the specified date on or after the specified time.

= date(time1,time2)

Returns information on all files that were last changed on the specified date and within the specified period.

= (date(time)[,])
Returns information on all files that were last changed on or after the specified date and time.

= (,date(time))
Returns information on all files that were last changed before the specified date and time.

= (date1(time),date2(time))
Returns information on all files that were last changed within the specified period. The upper and lower limits of the specified period are defined more precisely by time values.

MANCLAS

Returns in formation on all files according to the HSMS management class for file backup to SM pubsets.

= *ANY
The HSMS management class is not a selection criterion.

= *NONE
All files for which no HSMS management class is defined are selected.

= <c-string 1..8>
All files with the specified HSMS management class are selected.

MF

The forms of the MF operand are described in detail in the appendix ("Macro types"). All macros which differ with respect to the MF operand (MF = L/E/D/..) must have the same value for the VERSION operand.

Special facility with MF=D/C:
The user can control which output blocks are to be generated (XPAND=OUTPUT) by means of the CEINFO operand. With MF=D/C, only those blocks which the user has specified with CEINFO are expanded.

MIGRATE

Returns information on all files that have the specified entry for MIGRATE in the catalog. This entry is evaluated by the Hierarchical Storage Management System (HSMS) for the migration of files (see the MIGRATE operand in the CATAL macro on "CATAL - Process catalog entry"). If multiple values are listed, all files that satisfy one of the conditions are selected.

= ANY
The MIGRATE entry is not a selection criterion.

= ALLOWED
Selects only those files for which MIGRATE=ALLOWED was defined in the catalog entry, i.e. files which may be migrated to storage level S1 or S2.

= INHIBIT
Selects only those files for which MIGRATE=INHIBIT was defined in the catalog entry, i.e. files which may be briefly migrated (e.g. for reorganization purposes).

= FORBIDDEN
Returns information on files for which MIGRATE=FORBIDDEN is entered in the catalog entry, i.e. the files which must not be migrated.

= (list-of-migrate)
The ALLOWED and INHIBIT values may be specified by the user in a list. All files for which one of the specified values was defined in the catalog will be shown.

OTHERAR

Selects and returns information on files based on the access rights that are defined via BASIC-ACL entries for all users other than the file owners's user group.

= ANY
The BASIC-ACL entries for all users other than the file owner's user group do not serve as a selection criterion.

= NO-ACCESS
Returns information on all files that may be accessed by users not belonging to the file owner's user group.

= access-list
Returns information on all files for which at least one of the listed access rights has been defined for users not in the file owner's user group in the BASIC-ACL entries.

access-list has the following format:

- Long format:
  ([READ=YES / READ=NO][,WRITE=YES / WRITE=NO][,EXEC=YES / EXEC=NO])
- Short format:
  ([R=Y / R=N][,W=Y / W=N][,X=Y / X=N])

The parentheses form part of the operand values and are mandatory.

The individual elements of the access list mean the following:

READ=YES or R=Y	Selects all files that can be accessed for reading by users who are not in the owner's user group.
READ=NO or R=N	Selects all files that cannot be accessed for reading by users who are not in the owner's user group.
WRITE=YES or W=Y	Selects all files that can be accessed for writing by users who are not in the owner's user group.
WRITE=NO or W=N	Selects all files that cannot be accessed for writing by users who are not in the owner's user group.
EXEC=YES or X=Y	Selects all files that may be executed by users who are not in the owner's user group.
EXEC=NO or X=N	Selects all files that cannot be executed by users who are not in the owner's user group.

OUTAREA

Output area for information from the catalog entry.

= (<list-of-elements-002>)
The output area is output in list form, consisting of:

- The output area address. It can be output as a constant or equate and must be aligned on a word boundary.
- The length of the output area in bytes. The value to be output depends on the scope of the information requested via the OUTPUT operand.It can be output as a constant or equate.

The format of the output area is described on "Programming notes (VERSION=2, 3 and 4)".

Note

SPECIFIED bits are set internally for the two output areas (OUTAREA and STOUTAR). In order to enable output, the corresponding AREA-SPECIFIED bit must be set. These bits are set when MF=L applies. It is therefore advisable to supply dummy values for the OUTAREA or STOUTAR operands in an MF=L call.

OUTPUT

The output layout in blocks comprises two independent sections containing information from the catalog entry and statistics on all selected catalog entries. For more information on the various information blocks, see "Programming notes (VERSION=2, 3 and 4)".

The information from the catalog entry is transferred to the output area specified by the OUTAREA operand.

Statistics are shown in the output area defined by the STOUTAR operand.

= RC-ONLY
Only the return code is shown, so no output area is required.

= FNAM-ONLY
Only the path names (with PVSID, USERID and FILENAME) are copied to the output area (see the description on "Programming notes (VERSION=2, 3 and 4)").

The path names are copied to the output area specified by the OUTAREA operand.

= CEINFO
Returns information on the catalog entries. The information to be shown can be controlled via the CEINFO operand. No statistics are output (see description on "Programming notes (VERSION=2, 3 and 4)").

The information from the catalog entry is copied to the output area specified by the OUTAREA operand.

= STAT-SHORT
Only the statistics are copied to the corresponding output area (see description on "Programming notes (VERSION=2, 3 and 4)").

The statistics are copied to the output area specified by the STOUTAR operand.

Note

No statistics can be retrieved if the operand FROM=(volume,device) is specified at the same time. Function calls in which this is done are rejected with return code DMS0576.

= STAT-LONG
Only the statistics are transferred to the output area. In addition to the output of STAT-SHORT, individual statistics for the PVSID and USERID are also shown (see the description on "Programming notes (VERSION=2, 3 and 4)").

The statistics are copied to the output area specified by the STOUTAR operand.

Note

No statistics can be retrieved if the operand FROM=(volume,device) is specified at the same time. Function calls in which this is done are rejected with return code DMS0576.

= STAT-INFO
Outputs information on the catalog entries as well as the statistics (corresponds to the output set CEINFO + STAT-LONG).

Both sections of the output are linked by pointers (see the description on "Programming notes (VERSION=2, 3 and 4)").

The information from the catalog entry is transferred to the output area specified by the OUTAREA operand; statistics are copied to the output area specified by the STOUTAR operand.

Note

No statistics can be retrieved if the operand FROM=(volume,device) is specified at the same time. Function calls in which this is done are rejected with return code DMS0576.

OWNERAR

Returns information on files selected on the basis of the access rights that are defined for the file owner in the BASIC-ACL entries.

= ANY
BASIC-ACL entries for the file owner are not used as a selection criterion.

= NO-ACCESS
Returns information on all files that the owner is not allowed to access.

= access-list
Returns information on all files for which at least one of the listed access rights has been defined for the file owner in the BASIC-ACL entry.

access-list has the following format:

- Long format:
  ([READ=YES / READ=NO][,WRITE=YES / WRITE=NO][,EXEC=YES / EXEC=NO])
- Short format:
  ([R=Y / R=N][,W=Y / W=N][,X=Y / X=N])

The parentheses form part of the operand value and must be specified.

The individual elements of the access list mean the following:

READ=YES or R=Y	Selects all files that may be accessed by the owner for reading.
READ=NO or R=N	Selects all files that cannot be accessed by the owner for reading.
WRITE=YES or W=Y	Selects all files that can be accessed by the owner for writing.
WRITE=NO or W=N	Selects all files that cannot be accessed by the owner for writing.
EXEC=YES or X=Y	Selects all files that may be executed by the owner.
EXEC=NO or X=N	Selects all files that the owner is not allowed to execute.

PASS

Selects and returns information on files/file generation groups on the basis of the password protection defined with CATAL. If several password types are specified in the form of a list, all files that satisfy one of the named conditions (logical OR) are selected by the system. Passwords are not output.

= ANY
Password protection is not used as a selection criterion.

= NONE
Selects files for which no password protection exists.

= RDPASS
Selects files which are protected by means of a read password.

= WRPASS
Selects files which are protected by a write password.

= EXPASS
Selects files which are protected by an execute password.

= (list-of-pass)
The user may specify more than one type of password in the form of a list. All files protected by one of the specified password types will be shown.

PASSW

The PASSW operand determines how passwords are copied to the output area.

= NO
Default value: passwords are not explicitly displayed in the output area; the corresponding fields are set to binary 0.

= YES
For privileged users only:
passwords are explicitly represented in the output area.

PREFIX

Only evaluated in conjunction with MF=C or MF=D. Defines which characters the field names and equates generated during macro expansion in the data area are to begin with.

Default value: PREFIX = D

= pre
“pre” is a one-character prefix with which the field names and equates generated by the assembler are to begin.

PREFORM

Returns information on files dependent on their (intended) file format on SM pubsets.

= *ANY
The file format is not a selection criterion.

= *NONE
Returns information on all files for which no PREFORMAT value is defined.

= *K
Returns information on all files with the intended *K file format.

= *NK2
Returns information on all files with the intended *NK2 file format.

= *NK4
Returns information on all files with the intended *NK4 file format.

= (list-of-preform)
Returns information on all files which correspond to one of the specified file formats. Any value apart from ANY can be specified within the list.

PROTACT

Selects and retrieves information on files on the basis of the protection level provided by the highest activated access control.

When the file is accessed, the highest activated protection level applies. The following table shows the method used for access control, the protection attribute to be specified in the CATAL macro and the file protection hierarchy (protection levels):

Access control method	Protection attribute	Protection level
Standard access control	ACCESS and SHARE	0
Basic access control list	BASACL, OWNERAR, GROUPAR, OTHERAR	1
Access control using GUARDS	GUARDS	2

All other protection attributes of the file (e.g. passwords) are evaluated independently, without regard to the implemented protection level.

= ANY
is the default value; returns information on all files without regard to the protection level of the highest activated access control.Time specifications must always be made on **UTC** basis.

= LEVEL-0
Returns information on files for which access is controlled via standard access control.

= LEVEL-1
Returns information on files for which access is controlled via a basic access control list (BASIC-ACL protection).

= LEVEL-2
Returns information on files for which access is controlled by GUARDS.

= (list-of-protact)
The user may specify up to a maximum of three protection levels in a list. All files for which the protection level of the access control method matches one of those specified are selected.

RELSPAC

Returns information on files selected on the basis of the lock to prevent the release of unused memory space (as defined in the FILE macro or the MODIFY-FILE-ATTRIBUTES command). The lock can be defined in the catalog by using the CATAL macro.

= ANY
The lock to prevent the release of unused memory space is not used as a selection criterion.

= ALLOWED
Returns information on files for which unused memory space may be released.

= IGNORED
Returns information on files for which the release of unused memory space is not permitted.

SHARE

Selects and retrieves information on files or file generation groups based on whether or not they are shareable. If a foreign user ID which is not co-owner is specified with “$userid.”, SHARE=YES applies implicitly (see also the SHARE operand in the CATAL macro).

= ANY
Shareability is not used as a selection criterion.

= YES
Returns information on all files that are shareable, i.e. which are also accessible to other user IDs under active standard access control.

= NO
Returns information on all files that are not shareable, i.e. that are only accessible to the owner under active standard access control.

= SPECIAL
Returns information on files which can be accessed by all user IDs, including maintenance IDs (i.e. user IDs with hardware maintenance privileges).

= (list-of-share)
One or more operand values may be specified in a list.

SIZE

Returns information on files/file generations based on the size of the reserved storage space. Whole numbers between 0 and 2147483647 can be used as values.

= ANY
The size of the reserved storage space is not used as a selection criterion.

= FSIZE
Returns information on files for which the number of reserved PAM pages is equal to the number of free pages.

= nmbr
Returns information on files with exactly the specified number of reserved PAM pages.

= (nmbr[,])
Returns information on files with at least the specified number of reserved PAM pages.

= (,nmbr)
Returns information on files no more than the specified number of reserved PAM pages.

= (nmbr1,nmbr2)
Returns information on files for which the number of reserved PAM pages lies in the specified range (nmbr1 < nmbr2).

SLEVEL

Returns information on files selected on the basis of the level in the storage hierarchy at which they are held (see the “HSMS” manual [10]). The following three storage levels are supported by HSMS (Hierarchical Storage Management System):

S0: implemented as fast access disk storage (for online processing)

S1: implemented as high capacity disk storage (background level, available online)

S2: implemented as a magnetic tape and tape cartridge archive (background level, available offline)

= ANY
The storage level is not used as a selection criterion.

= S0
Selects only the files which are being held at level S0.

= S1
Selects only the files which are being held at level S1.

= S2
Selects only the files which are being held at level S2.

= (list-of-slevel)
The user may specify more than one storage level in the form of a list.

SORT

The SORT operand defines how catalog entries/path names are sorted in the output.

= FILENAM
Catalog entries/path names are output in alphabetical order.

= NO
Catalog entries/path names are output in the order in which they are located in the catalog.

STATE

Returns information on files selected on the basis of their current processing state.

= ANY
The current processing status is not a selection criterion.

= NOCLOS
Returns information on all files which have been opened for writing and are not closed. Such files include:

- normally open files (OPEN mode OUTIN, INOUT, OUTPUT)
- files not closed in a previous session
- files not closed in the current session because of job abortion.

Note

STATE=NOCLOS implies GEN=YES, i.e. opened file generations are always output.

= CLOSED
Returns information on all files that have already been closed, i.e. files not selected by NOCLOS.

= CACHED
Returns information on the files which are currently cached.

= NOT-CACHED
Returns information on all files which are not being currently processed in a cache.

= CACHE-NOT-SAVED
Returns information on all files for which it was not possible to write the cache to a disk when they were closed.

= REPAIR-NEEDED
Returns information on all files which were not closed in an earlier session and which have not yet been verified (see the VERIFY macro).

= DEFECT-REPORTED
Returns information on all files which could contain defective disk blocks.

= NO-OPEN-ALLOWED
Returns information on all files which cannot be opened due to data inconsistency.

= OPEN-ALLOWED
Returns information on all files that can be opened.

= (list-of-state)
A list of values may be specified (with a maximum of 7 file states). The output shows all files which are in one of the specified states.

STOCLAS

Returns information on all files according to the file storage class on SM pubsets.

= *ANY
The storage class is not a selection criterion.

= *NONE
All files for which no storage class is defined are selected.

= <c-string 1..8>
All files with the specified storage class are selected.

STOTYPE

Return information on all files according to the storage type.

= *ANY
The storage type is not a selection criterion.

= *PUBSPACE
Only files on public volumes are selected.

= *NETSTOR
Only files on Net-Storage volumes are selected.

STOUTAR

Output area for statistics.

= (<list-of-elements-002>)
The output area is output in list form, consisting of:

- Address of the output area. It may only be specified as a constant or equate and must be aligned on a word boundary.
- Length of the output area (in bytes). The value to be specified here depends on the scope of the information requested with the OUTPUT operand.
  It may only be specified as a constant or equate.

The format of the output area is described on "Programming notes (VERSION=2, 3 and 4)".

Note

SPECIFIED bits are set internally for the output areas OUTAREA and STOUTAR. The appropriate AREA-SPECIFIED bit must be set to enable the output. Since these bits are set with MF=L or MF=M, it is usually meaningful to supply dummy values for these operands in an MF=L call.

SUPPORT

Selects and returns information on files, file generations, or file generation groups (FGGs) on the basis of the type of volume on which they are stored.

= ANY
The volume type is not used as a selection criterion.

= PUBLIC
Returns information on files, file generations or FGGs which are stored on public volumes or Net-Storage volumes.

= PRDISC
Returns information on files, file generations or FGGs which are stored on private disk.

= TAPE
Returns information on files, file generations or FGGs which are stored on tape.

= (list-of-support)
A list of values may be specified (with up to 3 volume types). The output shows all files which are stored on one of the specified volume types.

S0MIGR

Returns information on files dependent on whether reallocation (migration) to S0 level is permitted.

= *ANY
The migration allowance is not a selection criterion.

= *ALLOWED
Returns information on files for which migration within the S0 level is permitted.

= *FORBIDDEN
Returns information on files for which migration within the S0 level is not permitted.

= (list-of-s0migr)
The user can specify the required values in a list. This returns information on all files for which one of the specified values was defined in the catalog.

TIMBASE

Controls whether the absolute date is entered in UTC or local time. The date format of the FSTAT output is also linked to this operand.

= *UTC
Absolute dates are input and output in UTC time.

= *LTI
Absolute dates are input and output in local time.

TYPE

Returns information on files selected on the basis of their type.

= ANY
Returns information on all files irrespective of their type.

= FILE
Returns information on all files with the exceptions of file generation groups.

= FGG
Returns information on file generation groups and, when GEN=YES is specified, also on file generations (see operand "GEN").

= PLAM
Returns information on PLAM libraries. This is a subset of the files which are selected by the specification TYPE=FILE.

= (list-of-type)
Returns information on all files that match one of the specified types. A maximum of three file types can be specified by the user in the form of a list.

USRINFO

Returns information on files/file generations dependent on user metainformation.

= *ANY
The user metainformation is not a selection criterion.

= *NONE
Returns information on files which have no user metainformation.

= <c-string 1..8>
Returns information on files with the specified user metainformation.

VERSION

Specifies which version of the parameter list is to be generated.

Note

If FSTAT is called in VERSION=0/1 in an environment that supports “large files” then a check, and possibly a conversion, will have to be performed. For more detailed information on this subject, refer to "Programming notes for VERSION=0 and VERSION=1".

= 0
Default value: generates the parameter list format that was supported prior to BS2000 V8.0A. This format will, however, only support the parameters which were known at that time. For example, the path name can only be specified without wildcards, and only VOLUME and POS are permitted as selection parameters. The supported operands and operand values can be found in the table "Version differences - VERSION=0/1/2/3/4/5".

= 1
Generates the parameter list format that was supported from BS2000 V8.0 through BS2000 V10.0.
This format will also support only the parameters that were known at the time. The supported operands and operand values can be found in the table "Version differences - VERSION=0/1/2/3/4/5".

= 2
Generates the parameter list format for versions as of BS2000/OSD-BC V1.0.

= 3
Generates the parameter list format for versions as of BS2000/OSD-BC V3.0.

= 4
Generates the parameter list format for versions as of BS2000/OSD-BC V9.0.

= 5
The parameter list format for versions BS2000 OSD/BC V11.0 and higher is generated.

Note

If existing software which manipulates the generated parameter list is to be recompiled or reassembled, the old format must be requested. Otherwise, source code compatibility is ensured.

VERSION-BACKUP

The user can select the files to be processed depending on whether they are part of the version backup.

= *ANY
Files are selected irrespective of whether they take part in the version backup.

= *ENABLED
Informs about all files that are part of the version backup (NUM-OF-BACKUP-VERS > 0).

= *DISABLED
Informs about all files that are not part of the version backup (NUM-OF-BACKUP-VERS = 0).

VOLSET

The user can select the files to be processed via the volume set on which they reside.

= *ANY
The volume set is not a selection criterion.

= *CONTROL
Returns information on all files on the control volume set of the SM pubset.

= <c-string 1..4>
Returns information on all files on the specified volume set.

VOLUME

The user can select files to be processed on the basis of the VSN (volume serial number) of the disk on which they are stored.

= ANY
The VSN of the volume is not a selection criterion.

= vsn
Returns information on all files/file generation groups which contain an entry for the volume with the specified VSN in their volume list. If “vsn” does not designate a private disk, no information is returned on file generation groups.

VTOC

The user can decide whether the requested information is to be obtained from the VTOC (= Volume Table of Contents) of a private disk or of a Net-Storage volume or from the system file catalog TSOSCAT. The VTOC operand cannot be used on partially-qualified file names or in combination with GEN=YES.

= NO
Outputs the current entry in the TSOSCAT.

= YES
Returns the VTOC catalog entries (from the F1 label of a private disk or the catalog of a Net-Storage volume) in accordance with the last current status in the entire computer network. The corresponding volume list must be assigned. The VTOC entry from the volume replaces the corresponding TSOSCAT entry. This makes it possible to restore consistency between the VTOC and TSOSCAT entries. If the specified file no longer exists on the volume shown in the TSOSCAT entry, the TSOSCAT entry is deleted.

WORKFIL

Returns information on files on SM pubsets dependent on whether they can be deleted by the system administrator (work files).

= *ANY
Whether or not the files are work files is not a selection criterion.

= *NO
Returns information on all files which are not work files.

= *YES
Returns information on all files which are work files.

WTQUIET

Controls whether the end of the QUIET state is to be waited for with a shared or remote imported pubset that was in the QUIET state at the time of the FSTAT call. This wait time is defined with DIALOG-WAIT-TIME or BATCH-WAIT-TIME in the corresponding MRSCAT entry.
Otherwise, the job is either aborted immediately with return code DMS0502 or, if the catalog ID contains wildcards, the pubset is skipped.

= *YES
If a pubset is in the QUIET state, this leads to a wait state.

= *NO
If a pubset is in the QUIET state, this either leads to return code DMS0502 or, if the catalog ID contains wildcards, the pubset is skipped.

XPAND

Controls which input parameter area is to be generated (with or without an area for selection parameters). XPAND can also be used to generate data descriptions (DSECTs) for the output area.

= PLSHORT
Generates the input parameter area without an area for selection parameters.

= PLLONG
Generates the input parameter area with an area for selection parameters.

= OUTPUT
Generates all DSECTs to describe the output information blocks.

= (PLSHORT,OUTPUT)
The input parameter area is generated without an area for selection parameters; all DSECTs to describe the output information blocks are generated.

= (PLLONG,OUTPUT)
The input parameter area is generated with an area for selection parameters; all DSECTs to describe the output information blocks are generated.

Return codes

The error code is only returned in the parameter list standard header and no longer in general-purpose register 15 as with version 2.

Note

If the catid entry contains wildcards, the following return codes are suppressed.

X'02000000'
X'00400501'
X'00400503'
X'00400505'
X'00400616'
X'00820502'
X'00820504'
X'00820506'

If no other error occurs and a file was selected, code 0 is returned and, if no file was selected, DMS06CC is returned.

Standard header: `ccbbaaaa`

The following code relating to execution of the FSTAT macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'00'	X'0000'	No error
X'02'	X'00'	X'0000'	Output incomplete because of defective volume set; if IFSVSETI = IFSODVSE (ONE DEFECT VOLUMESET), the volume set ID is in IFSVSET.
	X'40'	X'0501'	Requested catalog not available
	X'82'	X'0502'	Requested catalog in the quiet state
	X'40'	X'0503'	Incorrect information in the MRSCAT
	X'82'	X'0504'	Error in catalog management system
	X'40'	X'0505'	Computer communication error (MRS)
	X'80'	X'0506'	Operation aborted because of master change
	X'40'	X'0510'	Error when calling an internal function
	X'40'	X'0512'	Requested catalog unknown
	X'40'	X'051B'	User ID unknown in specified pubset
	X'40'	X'051C'	No access right to specified pubset
	X'40'	X'051D'	LOGON password for specified pubset is different
	X'40'	X'052E'	Volume no longer available
	X'20'	X'0530'	Storage space request error

	X'20'	X'0531'	Unexpected catalog access error
	X'40'	X'0533'	Specified file not found
	X'82'	X'0534'	Private volume cannot be allocated
	X'20'	X'053B'	System error during file access
	X'40'	X'053D'	Catalog or F1 label block is destroyed
	X'20'	X'054F'	Unexpected error while accessing JOIN file
	X'82'	X'055A'	Device currently reserved
	X'40'	X'055F'	Volume could not be reserved
	X'01'	X'0576'	Conflicting operand combination or the reserved fields of the parameter area were used or selection contains large files
	X'20'	X'0577'	Internal error while accessing the job environment
	X'82'	X'0594'	Not enough virtual memory available (also if wildcards were used and too many files were selected)
	X'01'	X'0599'	Operand is not supported in the RFA-BS version
	X'01'	X'05A8'	Requested device type not found in the system
	X'01'	X'05AB'	Output area address incorrect or not specified
	X'82'	X'05B0'	No suitable device currently available
	X'40'	X'05B4'	Volume cannot be made available
	X'40'	X'05D1'	Device request error
	X'01'	X'05EA'	VTOC=YES is illegal with partially qualified file name or file generation group
	X'01'	X'05EE'	File name too long
	X'40'	X'05FC'	Specified user ID not in home pubset
	X'40'	X'0616'	Output not possible because of defective volume set; if IFSVSETI = IFSODVSE (ONE DEFECT VOLUMESET), the volume set ID is in IFSVSET.
	X'01'	X'06B8'	Invalid operand specified
	X'01'	X'06C7'	Invalid generation number specified
X'xx' X'01' X'02' X'03'	X'00'	X'06CB'	Output area too short Catalog information not completely transferred Statistic information not completely transferred Catalog and statistic information not completely transferred

	X'40'	X'06CC'	Only with selection entry (wildcard): No file matches the selection input
	X'01'	X'06FD'	Parameter area invalid or not accessible
	X'40'	X'06FF'	BCAM connection severed
	X'01'	X'FFFF'	Incorrect function number in parameter area header
	X'03'	X'FFFF'	Incorrect version number in parameter area header

Programming notes for VERSION=4

FSTAT VERSION=4 supports files on Net-Storage volumes.

The STOTYPE operand permits selection according to the storage type: files on public volumes or files on Net-Storage volumes.

The FILTYPE operand enables selection according to the file type: BS2000 files or node files.

In the output area the information on the number of files on Net-Storage volumes is also displayed in the output structures OUTPUT=STAT-LONG/STAT-SHORT/ STAT-INFO in the statistical information of MAIN_HEADER, PVSID_HEADER and USERID_HEADER (see "Programming notes (VERSION=2, 3 and 4)"):

- Number of files on Net-Storage (4 bytes)
- Number of free PAM pages on Net-Storage (4 bytes)

In the output area the following information is also displayed in the output structure OUTPUT=CEINFO:

- in the organization block (see *BLOCK 4 : FILE ORGANIZATION INFORMATION* in Dsect):
  - file type on Net-Storage (`FILE_TYPE`): BS2000 file or node file
  - file size of node files (`NODE_FILE_SIZE`)
  - validity indicator for the node file size (`NODE_FILE_SIZE_VALID`)
- in the allocation block (see *BLOCK 6A: FILE ALLOCATION INFORMATION* in Dsect):
  - last-byte pointer for PAM files and node files (`LAST_BYTE_POINTER`): points to the last valid byte on the last logical page of the file. The value is only valid if the validity indicator is set as well.
  - validity indicator for the last-byte pointer (`LAST_BYTE_POINTER_VALID`)

Further information is provided in the following section "Programming notes (VERSION=2, 3 and 4)".

Programming notes (VERSION=2, 3 and 4)

In FSTAT VERSION=2/3, the corresponding data (extent list and data fields or the file-size and last-page pointers) are returned as 4-byte fields. Consequently, these interfaces do not have to be converted for calls in configurations involving files > 32 GB.
However, it is necessary to take account of the semantic problem affecting the OPEN macro, see "OPEN - Open file". All users of this interface should check whether this problem affects their particular implementation.

The FSTAT macro enables the user to request information from the catalog entry of one or more files.
The OUTAREA operand reserves an output area for the general information retrieved from the catalog entry; the STOUTAR operand assigns an output area for statistics.

The assigned output area should be overwritten with X'00' before data is transferred to it, since the unused parts of the output area are not fully deleted to the end.

The information from the catalog entry is transferred to the output area in blocks, and the requested information blocks are shown directly under one another in the output area. If the number of files is known, the size of the output area can be calculated precisely. Otherwise, if the free space available in the output area is too small to hold an information block, no further data is written to the output area, and the free space is filled with X'00'. The field IROLN (Real Output Length) of the input area returns information on the number of bytes that have been transferred to the output area (if the input parameter area is generated with FSTAT MF=D, VERSION=2, XPAND=PLSHORT).

Depending on the number of information blocks to be transferred and the size that is defined for the output area, the following cases may be differentiated:

The output could be completed fully:

Only the requested information is written to the output area; all unused fields contain X'00'.The caller receives the return code:

`00 00 0000`:
All the requested information was transferred to the output area.

No file matching the selection criteria was found:

No information is written to the output area. The caller receives the following return code:

`00 00 0533`:

The specified file was not found in pubset '(&00)'.

or

`00 00 06CC`:

The specified file was not found in pubset '(&00)'.

No output was possible:

If absolutely no data can be written to the output area, the caller receives the following return code:

`00 01 05AB`:

nvalid address for the area or illegal length specification in the FSTAT macro.

The program should be checked and corrected.

The output could not be completed fully:

If the estimated size of the output area is too small and some of the requested information blocks cannot be transferred as a result, in addition to the main code 06CB (“length specification too small for entry”), an indicator showing which of the two output areas (OUTAREA/STOUTAR) could not be transferred is placed in subcode2.

`01 00 06CB`:

The catalog entry information could not be transferred fully (for OUTPUT=CEINFO/FNAM-ONLY).

`02 00 06CB`:

The requested statistics could not be transferred fully (for OUTPUT=STAT-SHORT/STAT-LONG).

`03 00 06CB`:

Both the catalog entry information and the statistics could not be transferred fully (for OUTPUT=STAT-INFO).

The user can define a larger output area and repeat the FSTAT call in such cases.

The contents and structure of the information to be output depends on which information is selected for retrieval by means of the OUTPUT operand. The following options are available:

OUTPUT = CEINFO / FNAM-ONLY / RC-ONLY / STAT-INFO / STAT-LONG / STAT-SHORT

No output area is required for OUTPUT=RC-ONLY. If OUTPUT=STAT-INFO is specified, the same information as for CEINFO + STAT-LONG is output.

The information blocks can be accessed via an information header. If no blocks are requested, the address pointer for this block is set to null.

The offsets in the respective output areas are calculated relative to the start of the block in which the offset is defined. The offsets in BLOCK_INFORMATION_Header2 are related to BLOCK_INFORMATION_Header1, since both blocks are treated as a single unit.

> All the output structures shown below are typical examples in which the block name (e.g. BLOCK_INFORMATION_HEADER) refers to the block headers of the DSECT that can be generated to evaluate the output area.
> This DSECT can be generated with the macro:
> **FSTAT MF=D, XPAND=OUTPUT, VERSION=3**.

Output structure for OUTPUT=CEINFO

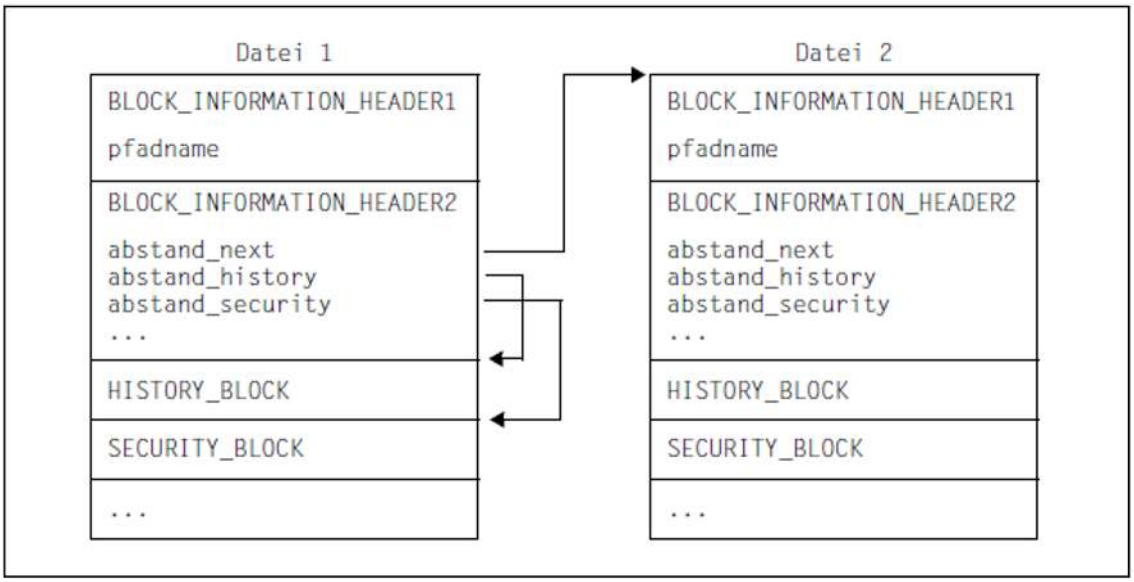
In this output format, all information from the catalog entry excluding the statistics are output in the area defined by OUTAREA. The desired information can be selected via the CEINFO operand.

CEINFO =	HISTORY	Information on accesses to the file
	SECURITY	Access rights and data security attributes
	BACKUP	Backup attributes for the file
	ORGANIZATION	File organization attributes

	STATUS	Special characteristics of files

	ALLOCATION	Information on physical attributes of the file
	VOLUME	Volume list
	VOLUME-EXTENTS	Volume list and extent list
	INDEX-INFO	Information on the file generation group
	FTAM	FTAM (file transfer access method) data

The following diagram shows the output areas for two files:



BLOCK_INFORMATION_HEADER1 contains the following:

pathname Path number of the selected file, subdivided into:

- length of the pvsid (catalog ID) (2 bytes)
- PVSID (catalog ID) (4 bytes)
- length of the user ID (2 bytes)
- user ID (8 bytes)
- length of the file name (2 bytes)
- file name (41 bytes)

BLOCK_INFORMATION_HEADER2 contains the address pointers to the information blocks and to the output area of the next file.

offset_next: Offset to the next BLOCK_INFORMATION_HEADER (2 bytes) (start of the output area of the next file)

offset_history: Offset to the HISTORY_BLOCK (2 bytes)

offset_security: (2 bytes)

offset_backup: (2 bytes)

...

The two headers are followed by the information blocks which contain the actually useful information (HISTORY_BLOCK, SECURITY_BLOCK etc.). The description and length of the individual fields can be obtained from the DSECT.

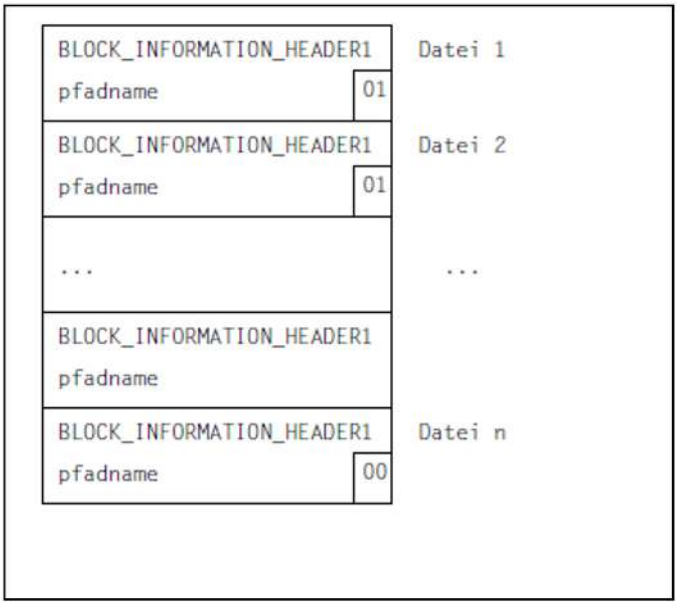
Output structure for OUTPUT=FNAM-ONLY

In this output format, only the path names of the selected files are shown in the output area defined by OUTAREA. Apart from the end criterion in the last byte, the BLOCK_INFORMATION_HEADER1 in the output structure for OUTPUT=FNAM-ONLY is identical to the corresponding header for OUTPUT=CEINFO (see "Programming notes (VERSION=2, 3 and 4)").

End criterion

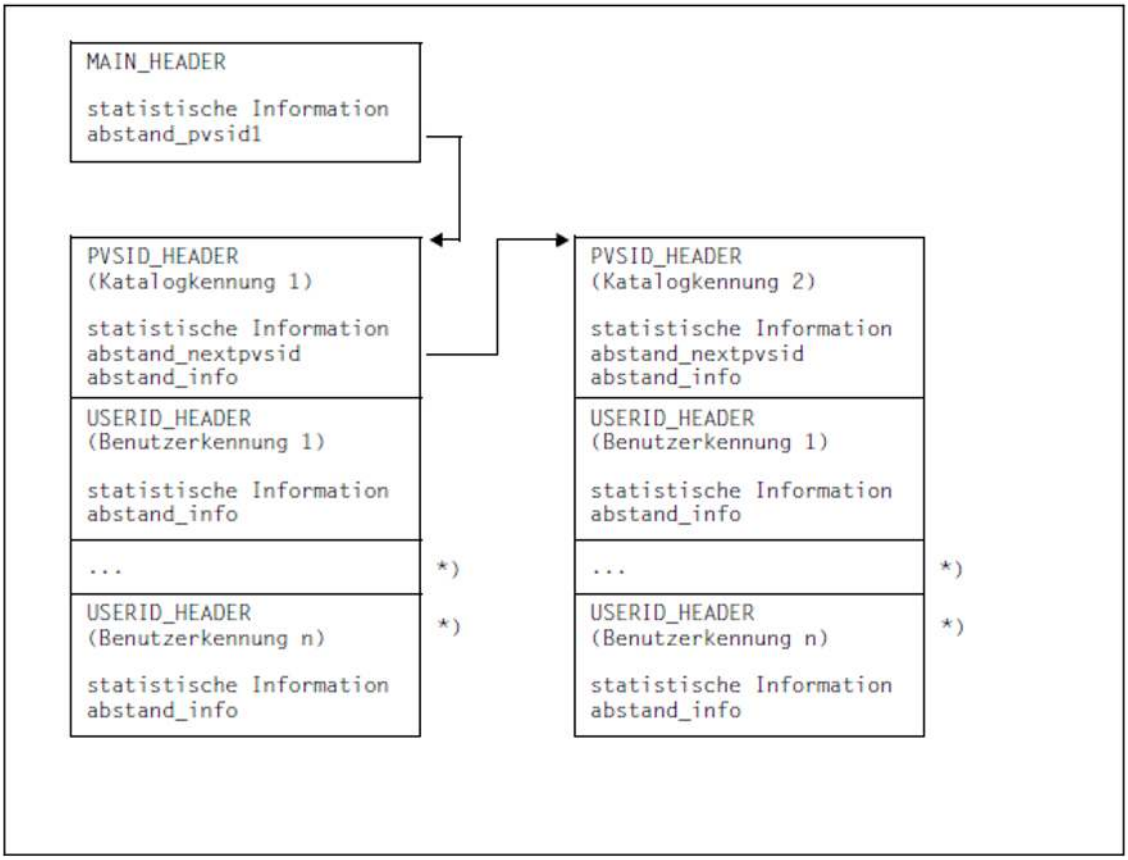
X'00' There are no further path names.

X'01' Additional path names follow.



Output structure for OUTPUT=STAT-LONG

In this output format, only the statistics are shown in the output area defined by the STOUTAR operand.
Nonprivileged users are returned information on files that can be stored on one or more pubsets.
The *system administrator* can also use wildcards to have information on multiple user IDs displayed.



*) Only the *system administrator* can select multiple user IDs.

The **MAIN_HEADER** returns the following information:

Statistics

- Total number of selected files (4 bytes)
- Number of files on public volumes (4 bytes)
- Number of files on private volumes (4 bytes)
- Number of files on Net-Storage (4 bytes)
- Number of files on tape (4 bytes)
- Number of files with migration level 1 (HSMS) (4 bytes)
- Number of files with migration level 2 (HSMS) (4 bytes)
- Number of user IDs (2 bytes)
- Number of free PAM pages on public volumes (4 bytes)
- Number of free PAM pages on private disk (4 bytes)
- Number of free PAM pages on Net-Storage (4 bytes)
- Number of free PAM pages at migration level 1 (4 bytes)
- Number of free PAM pages at migration level 2 (4 bytes)

offset_pvsid1 Offset to the first PVSID_HEADER (2 bytes)

The **PVSID_HEADER** returns the following information:

Statistics

- Catalog ID (4 bytes)
- Total number of selected files (4 bytes)
- Number of files on public volumes (4 bytes)
- Number of files on private volumes (4 bytes)
- Number of files on Net-Storage (4 bytes)
- Number of files on tape (4 bytes)
- Number of files with migration level 1 (HSMS) (4 bytes)
- Number of files with migration level 2 (HSMS) (4 bytes)
- Number of user IDs (2 bytes)
- Number of free PAM pages on public volumes (4 bytes)
- Number of free PAM pages on private disk (4 bytes)
- Number of free PAM pages on Net-Storage (4 bytes)
- Number of free PAM pages at migration level 1 (4 bytes)
- Number of free PAM pages at migration level 2 (4 bytes)

offset_nextpvsid	Offset to the next PVSID_HEADER (2 bytes)
offset_info	Offset to the BLOCK_INFORMATION_HEADER1 of the first selected file for this catalog (4 bytes) (relative to the address of the OUTAREA)

The **USERID_HEADER** returns the following information:

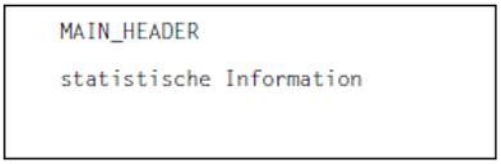
Statistics

- User ID (8 bytes)
- Total number of selected files (4 bytes)
- Number of files on public volumes (4 bytes)
- Number of files on private volumes (4 bytes)
- Number of files on Net-Storage (4 bytes)
- Number of files on tape (4 bytes)
- Number of files with migration level 1 (HSMS) (4 bytes)
- Number of files with migration level 2 (HSMS) (4 bytes)
- Number of free PAM pages on public volumes (4 bytes)
- Number of free PAM pages on private disk (4 bytes)
- Number of free PAM pages on Net-Storage (4 bytes)
- Number of free PAM pages at migration level 1 (4 bytes)
- Number of free PAM pages at migration level 2 (4 bytes)

offset_info	Offset to the BLOCK_INFORMATION_HEADER1 of the first selected file of this user ID that is found under the catalog ID indicated above (4 bytes) (relative to the address of the OUTAREA)

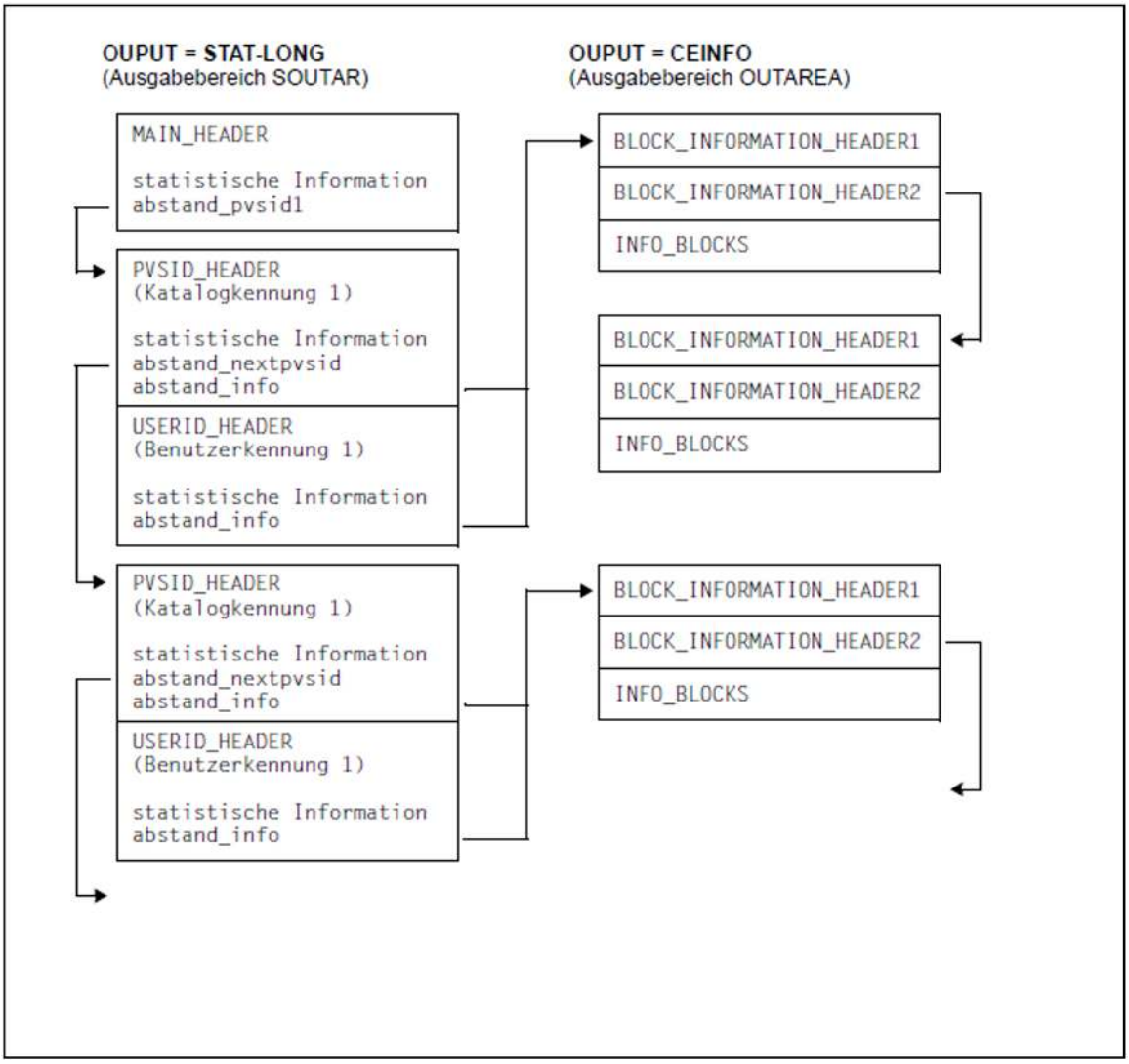
Output structure for OUTPUT=STAT-SHORT

In this output format, only the MAIN_HEADER of the output structure for OUTPUT=STAT-LONG is shown (see "Programming notes (VERSION=2, 3 and 4)").



Output structure for OUTPUT=STAT-INFO

This format combines the output structures for OUTPUT=CEINFO and OUTPUT=STAT-LONG. Two output areas, OUTAREA and STOUTAR, must be defined by the user for this purpose. The output area for statistics, STOUTAR (shown to the left in the diagram below), contains references to the output area OUTAREA (shown on the right). The diagram shows only one user ID in the output area STOUTAR. This corresponds to the selection option for nonprivileged users:


OUPUT = STAT-LONG
(Ausgabebereich SOUTAR)
OUPUT = CEINFO
(Ausgabebereich OUTAREA)
MAIN_HEADER
statistische Information
abstand_pvsid1
PVSID_HEADER
(Katalogkennung 1)
statistische Information
abstand_nextpvsid
abstand_info
USERID_HEADER
(Benutzerkennung 1)
statistische Information
abstand_info
PVSID_HEADER
(Katalogkennung 1)
statistische Information
abstand_nextpvsid
abstand_info
USERID_HEADER
(Benutzerkennung 1)
statistische Information
abstand_info
BLOCK_INFORMATION_HEADER1
BLOCK_INFORMATION_HEADER2
INFO_BLOCKS
BLOCK_INFORMATION_HEADER1
BLOCK_INFORMATION_HEADER2
INFO_BLOCKS
BLOCK_INFORMATION_HEADER1
BLOCK_INFORMATION_HEADER2
INFO_BLOCKS

Programming notes for VERSION=0 and VERSION=1

If the FSTAT macro accesses pubsets with large volumes that do not, however, allow large files, interface behavior is unchanged. Access of this kind is always performed without problems.
Problems can occur if pubsets that also allow large files are accessed.

Interface variants that do not need to be converted

```
FSTAT ...,VERSION=0,<partially-qualified filename>
FSTAT ...,VERSION=0,<file generation group with GEN=YES>
FSTAT ...,VERSION=1,FNAM
```

The semantics problem discussed at OPEN macro must be considered, see "OPEN - Open file".
Each user of this interface must check whether this problem applies to their implementation.

Interface variants that need to be checked/converted

`FSTAT ...,VERSION=0/1,SHORT/LONG`

These variants return the catalog information in BS2000 V10.0 format.
The extent lists and data fields for File-Size and Last-Page-Pointer are output with only 3 bytes. For reasons of compatibility, it is not possible to change the layout of these interfaces.

Calls that refer to large objects cause an overflow in the 3-byte fields.

Two different cases must be distinguished. These depend on the contents of the result list from the FSTAT call:

1. No file >= 32 GB exists in the result list (set of selected files).

   In this instance, FSTAT tolerates the overflow of the 3-byte data field of the PHP in the extent list. The value X'FFFFFF' is assigned to the PHPs that cannot be displayed. It is assumed that the PHPs are never or very rarely evaluated at the interfaces. If this is occasionally not the case, the interface must be converted to version 2 or 3.
   This achieves the following:

   - The introduction of large volumes can be carried out in a compatible manner, user programs do not need to be changed.
   - FSTAT calls with fully-qualified pathnames can be supported compatibly (except for the PHP overflow in the extent list).

   No conversion is necessary!

2. At least one file >= 32 GB exists in the result list (FSTAT is called with a partially-qualified filename or with wildcards).

   Calls of this kind are rejected with the following return code:

X'cc'	X'bb'	X'aaaa'	Explanation
X'00'	X'01'	X'0576'	The selection contains large files.

The following types of FSTAT interfaces are affected:

Type		Comment
I	FSTAT ...	VERSION=0 is set as default
II	FSTAT ...,VERSION=0	
III	FSTAT ...,VERSION=1,SHORT	
IV	FSTAT ...,VERSION=1,LONG	
V	FSTAT ...,VERSION=1	The SHORT operand is set as default

These calls must be converted to VERSION=2/3!

Summary of FSTAT calls

`FSTAT <fully-qualified pathname>,VERSION=0/1`

- Access only to small files (less than 32 GB):
  No action necessary [1)]
- Large files are also accessed (>= 32 GB):
  If the call is made with one of the types `I` to `V`, conversion to VERSION=2/3 is necessary, with the exception of types `I` and `II` where the file generation group and GEN=YES are specified.

`FSTAT <partially-qualified pathname or pathname with wildcards>,VERSION=0/1`

- The result list contains only small files (less than 32 GB):
  No action necessary [1)]

  Warning!
  You must ensure that the requirements are always fulfilled.
  This is probably only possible for pathnames that contain wildcards in the catalog ID and otherwise contain fully-qualified components.

[1] "No action necessary" applies here if the result list contains only small files and the overflow of the 3-byte data field of the PHP in the extent list does not present a problem. If this is not the case, conversion to VERSION=2/3 is necessary.

- The result list can also contain large files (>= 32 GB):
  If the call is made with one of the types `III` to `V`, conversion to VERSION=2/3 is necessary.

Conclusion

When using the FSTAT interface with VERSION < 2 and FORM=LONG or FORM=SHORT, it is necessary to convert to the most recent interface version under the following circumstances:

1. Large files should be accessible with the FSTAT call in the affected program.
2. The pathname in the FSTAT call is partially-qualified or contains wildcards and it cannot be assumed that the result list contains no large files. Calls with wildcards in the catalog ID are particularly critical here.

It may be necessary to convert the data structures in the program as well as the interface.

Control using the system parameter FST32GB

FST32GB only affects the following FSTAT interfaces:

- Version=0 (corresponds to Version=710) when a fully-qualified filename is specified (although not when a file generation group is specified with GEN=YES)
- Version=1 (corresponds to Version=800), where the FNAM operand was not specified

Whether the existence of a file >= 32 GB in the list of selected files leads to the FSTAT call being rejected with the `X'00000576'` return code (FST32GB=0, default setting) or whether an overflow of the 3-byte fields is always tolerated (FST32GB=1) is set globally on the system by systems support staff. In the latter case, the value X'FFFFFF' is assigned to the data fields that cannot be displayed.

Note

The system parameter FST32GB is not evaluated if the FSTAT indicator (see below) is set.

Control using the FSTAT indicator

Behavior equivalent to FST32GB=1, i.e. ignoring the overflow, can be activated for specific interfaces for FSTAT types `I` to `V` using an indicator.

Note

The FSTAT indicator has a higher priority than the system parameter FST32GB. If the FSTAT indicator is set, FST32GB is not evaluated.
The indicator must be set directly in the parameter list; no support is provided in the FSTAT macro.

Description of the bits in the corresponding DSECT:

```
         FSTAT MF=D,PARMOD=31,VERSION=710
IDBFLAG2 DS    X                               FLAGS 2
IDBLOPYE EQU   X'04'                           2-2 S LARGE PUBSET ACCESS=YES
         FSTAT MF=D,PARMOD=31,VERSION=800
IFLAG0   DC    B'10001100'
ILOPY    EQU   X'04'                           2-2 S LARGE PUBSET ACCESS=YES
```

Version differences - VERSION=0/1/2/3/4/5

The following table shows which operands/operand values are supported with VERSION=5/4/3/2/1/0.

- With VERSION=5, all operands/operand values supported in version BS2000 OSD/BC V11.0A and higher can be used.
- In VERSION=4 all operands/operand values can be used which are supported in BS2000/OSD-BC V9.0A and higher.
- All operands and operand values that were supported up to and including BS2000/OSD-BC V8.0A may be used with VERSION=3.
- All operands and operand values that were supported up to and including BS2000/OSD V2.0A may be used in the macro format with VERSION=2.
- All operands and operand values that were supported from BS2000 V8.0A to V10.0A may be used in the macro format with VERSION=1.
- All operands/operand values that were supported in BS2000 Versions < V8.0A can be used in the format with VERSION=0.

Operand	V0	V1	V2	V3	V4	V5	Remarks for operand values
MF=E	x	x	x	x	x	x	
PARMOD	x	x	x	x	x	x	
VERSION	x	x	x	x	x	x	
MF=C	-	-	x	x	x	x	
MF=D	x	x	x	x	x	x	
PARMOD	x	x	x	x	x	x	
PREFIX	-	x	x	x	x	x	
VERSION	x	x	x	x	x	x	
MF=L	x	x	x	x	x	x	
pathname	x	x	x	x	x	x	Vers=0: Temporary files are not taken into account, see also [3)] Vers=1: The length of the path name is specified by means of a positional operand, see [3)]
ACCESS	-	x	x	x	x	x	ANY valid from Vers=2
ACCCNT	-	-	x	x	x	x	
ADMINFO	-	-	-	x	x	x	
AVAIL	-	-	-	x	x	x	
BACKUP	-	x	x	x	x	x	ANY valid from Vers=2
BASACL	-	x	x	x	x	x	ANY valid from Vers=2; Vers=1: value YES with substructure, see[2)]

BLKCNT	-	-	x	x	x	x	
BLKCTRL	-	x	x	x	x	x	Vers=1: Only operand values NONE, DATA, PAMKEY, NO are allowed (also to be specified as a list)

CEINFO	-	-	x	x	x	x	
CCS	-	-	x	x	x	x	
CRDATE	-	x	x	x	x	x	The operand values ANY, NONE and values with (time) specifications are not possible, see also [1)]
DELDATE	-	-	-	x	x	x	
DISKWR	-	-	x	x	x	x	
EXDATE	-	x	x	x	x	x	The operand values ANY, NONE and values with (time) specifications are not possible, see also [1)]
EXTENTS	-	x	x	x	x	x	ANY valid from Vers=2
FCBTYPE	-	x	x	x	x	x	ANY valid from Vers=2
FSIZE	-	x	x	x	x	x	ANY valid from Vers=2
FILTYPE	-	-	-	-	x	x	
FROM	-	x	x	x	x	x	
GEN	x	x	x	x	x	x	
GROUPAR	-	(x)	x	x	x	x	GROUPAR from Vers=2 only, Vers=1: Information can be retrieved with BASACL, see [2)]
GUARDS	-	-	x	x	x	x	
IOPREF	-	-	x	x	x	x	
IOUSAGE	-	-	x	x	x	x	
LADATE	-	x	x	x	x	x	The operand values ANY, NONE and values with (time) specifications are not possible, see also [1)]
LASTPAG	-	x	x	x	x	x	ANY valid from Vers=2
LBPOINT	-	-	-	-	-	x	
LCDATE	-	-	x	x	x	x	
MANCLAS	-	-	-	x	x	x	
MIGRATE	-	x	x	x	x	x	ANY valid from Vers=2; FORBIDDEN valid from Vers=3
VERSION-BACKUP	-	-	-	-	-	x	
OTHERAR	-	(x)	x	x	x	x	Vers=1: The OTHERAR operand is not supported, the information can be retrieved with the BASACL operand, see also [2)]

OUTAREA	(x)	(x)	x	x	x	x	Vers=0/1: The length and address of the output area are represented by positional operands, see [3]); The operand values *REQUEST and *RELEASE are valid from Vers=3.

OUTPUT	(x)	(x)	x	x	x	x	Vers=0: The OUTPUT operand is not supported. The positional operands SHORT and LONG correspond to operand values STAT-SHORT and STAT-LONG of the keyword operand OUTPUT. See also [3)] Vers=1: The OUTPUT operand is not supported. The positional operands SHORT, FNAM and LONG correspond to operand values STAT-SHORT FNAM-ONLY and STAT-LONG of the keyword operand OUTPUT. See also [3)]
OWNERAR	-	(x)	x	x	x	x	Vers=1: The OWNERAR operand is not supported, the information can be retrieved with the BASACL operand, see also [2)]
PASS	-	x	x	x	x	x	ANY valid from Vers=2
PASSW	-	-	x	x	x	x	
PREFIX	-	x	x	x	x	x	
PREFORM	-	-	-	x	x	x	
PROTACT	-	-	x	x	x	x	
RELSPAC	-	-	x	x	x	x	
SHARE	-	x	x	x	x	x	Vers=1: The operand value ANY is illegal, as is specification of a list
SIZE	-	x	x	x	x	x	ANY valid from Vers=2
SLEVEL	-	x	x	x	x	x	ANY valid from Vers=2
SORT	-	x	x	x	x	x	
STATE	x	x	x	x	x	x	Vers=0: Only the operand value NOCLOS is possible Vers=1: only the operand values NOCLOSE and PCLOSE are possible Vers=2: The operand values CACHE-NOT-SAVED and DEFECT-REPORTED are illegal
STOCLAS	-	-	-	x	x	x	
STOTYPE	-	-	-	-	x	x	
STOUTAR	(x)	(x)	x	x	x	x	Vers=0/1: The length and address of the output area are represented by positional operands. In both versions, only one output area is defined. The output information is determined by the positional operands SHORT, LONG and FNAM (Vers=1 only). See [3)]
SUPPORT	x	x	x	x	x	x	specification of a list valid from Vers=1: ANY valid from Vers=2
S0MIGR	-	-	-	x	x	x	
TIMBASE	-	-	-	x	x	x	
TYPE	x	x	x	x	x	x	Vers=0/1: only FGG possible

USRINFO	-	-	-	x	x	x	
VERSION	x	x	x	x	x	x	
VOLSET	-	-	-	x	x	x	
VOLUME	x	x	x	x	x	x	ANY valid from Vers=2
VTOC	x	x	x	x	x	x	
WORKFIL	-	-	-	x	x	x	
WTQUIET	-	-	-	x	x	x	
XPAND	-	-	x	x	x	x	

Table 8: FSTAT - Version differences

Key

x The operand is available in the macro version.

(x) The operand is not available in the macro version under the specified name, but the same function can be executed by an operand of some other name.

- The operand is not available in the macro version.

Vers Version VERSION=710 corresponds to VERSION=0

VERSION=800 corresponds to VERSION=1

Note

The positional operand pathname is indicated before the alphabetically sorted keyword operands.

[1)] The format for the CRDATE, EXDATE and LADATE operands in macro version 1 is as follows:

```
CRDATE / EXDATE / LADATE = NONE / date / (date[,]) /
                                 (,date) / (date1,date2)
```

[2)] The format for the BASACL operand in macro version 1 is as follows.

```
,BASACL=NONE /
          YES([,OWNER=NO-ACCESS / access-list] /
              [,GROUP=NO-ACCESS / access-list] /
              [,OTHERS=NO-ACCESS / access-list])
```

[3)] Representation of positional operands in Vers=0 and Vers=1:

Vers=0:

```
[pathname],outaddr[,length][, SHORT / LONG
```

Vers=1:

```
[pathname] / ([pathname,length1)]
  ,outaddr / (S,outaddr) / (r1)
  ,length2 / (r2)[,SHORT / FNAM / LONG]
```

Version variations in the representation of the output area

Version=0 (corresponds to Version=710)

The scope of information written to the output area depends on whether a partially-qualified file name, a fully-qualified file name of an FGG (file generation group), or a fully-qualified file name is specified.

Partially-qualified file name or fully qualified file name of an FGG:

Only a list of file names is written to the output area.

length	file name		
length	file name		
: :	: :		
length	file name	end	control

length Length of the file name. The value of the length field is equal to the “length of the file name + 1 byte for the size of the length field”. This defines the offset to the next length field.

The name of a group entry is followed by the string '_(FGG)'. The value of the length field is then equal to the “length” of the group name + 1 byte for the size of the length field + 6 bytes for the string '_(FGG)'.

filename File name (maximum 41 bytes).

end Indicates the end of the list (X'00') (1 byte).

control Indicates whether the output area was large enough to accommodate all file names (1 byte).

X'00' Complete output:

all file names were transferred to the output area.

X'01' Incomplete output:

one or more file names could not be transferred to the output area, because it was already full.

Fully-qualified file names

The output format is defined by means of the SHORT and LONG operands:

SHORT The statistics section of the catalog entry is transferred to the output area (at least 60 bytes). The IDCE macro generates a DSECT that produces the layout of the output area.

LONG The complete catalog entry consisting of

- statistics section
- file name
- extension
- volume table
- file table
- FGG suffix

is shown in the output area (at least 2032 bytes).

If the value specified for the length of the output area (in the “length” operand, which corresponds to the length of the operand OUTAREA or STOUTAR in Vers=2) is not sufficient, no information is transferred.
For more information on generating DSECTs for the output area, see also the description of the SHORT operand on "Version variations in the representation of the output area".

Example

```
06 PETER OB ACCOUNTS 00 00
```

Version=1 (correspnds to Version=800)

Returns information on permanent or temporary files, file generations, or file generation groups that are specified by means of fully or partially qualified names.

The size and layout of the output area depends on the scope of the information requested for the files via the operands FNAM, SHORT and LONG.

FNAM A list of file names is transferred to the output area. The address and length of the output area are assigned by means of the operands “outaddr” and “$length_2$”, respectively.

(With Vers=2, these operands correspond to the operand values “addr” and “length” of the operand OUTAREA).

The output area must have a length of at least 73 bytes.

information_header			
data			
: :			
data	end1	end2	control

Total number of selected files (4 bytes)The information_header contains

- Length of the prefix and data that follows (4 bytes)
- Prefix:
  - length of the prefix (2 bytes)
  - length of the catalog ID + 2 (2 bytes)
  - catalog ID for all following entries
  - length of the user ID + 2 (2 bytes)
  - user ID for all following entries

The data area contains:

- length of the usage information that follows + 2 (2 bytes)
- file name (max. 41 bytes)

The output area is terminated by

end1	End-of-list flag for the data area (2 bytes, X'0000')
end2	End-of-list flag for the output area (4 bytes, X'00000000')
control	Return information on the execution of the FSTAT call (1 * byte)
	X'00': All the requested information was transferred to the output area
	X'01': The requested information could not be fully transferred to the output area.

SHORT Returns the statistics stored in the catalog entry for each file name.
The address and length of the output area are assigned by means of the operands “outaddr” and “length$_2$”, respectively.
(With Vers=2, these operands correspond to the operand values “addr” and “length” of the operand STOUTAR.)
The information header and the end1, end2, and control areas correspond to the representation for the FNAM operand.

The data area contains:

- statistics section of the catalog entry; the last byte contain the length of the following file name (60 bytes)
- file name (max. 41 bytes)

In other words, the data area for SHORT has a maximum length of 101 bytes.
The output area for SHORT must be at least 133 bytes long. The IDCE macro generates a DSECT that creates the layout of the output area.

LONG The complete catalog entry is output for each file name. The address and length of the output area are assigned by means of the operands “outaddr” and “length$_2$”, respectively.
(With Vers=2, these operands correspond to the operand values “addr” and “length” of the operand STOUTAR.)
The information header and the end1, end2, and control areas correspond to the representation for the FNAM operand.

The data area contains:

- statistics section
- file name
- extension
- volume table
- file table
- FGG suffix

The output area for LONG must have a length of at least 2064 bytes.

Note

The general-purpose registers 1 and 15 must not be specified for r_1, r_2 and r_3, and the specifications for r_1, r_2, and r_3 must constitute mutually exclusive pairs (logical OR).

GET - Read next record

ISAM:	Macro type:	R for PARMOD=24 0 for PARMOD=31
SAM:	Macro type:	R for PARMOD=24 R for PARMOD=31

Area of application

The GET function can be used when processing files with SAM or ISAM (record-oriented access methods). The file must have been opened with one of the following OPEN modes:

- INPUT (SAM or ISAM)
- INOUT (ISAM)
- OUTIN (ISAM)
- UPDATE (SAM)

Function

The GET function makes a record from a file available to the user. The record returned is always the record referenced by the current record counter. After an OPEN INPUT, this counter is 1. A series of GET macros will thus read the file records sequentially (cf. GETKY, GETFL, GETR for ISAM).

The GET macro does not always issue an SVC. An SVC will only be issued if the next record to be read is no longer in the current buffer area.

If a record located outside the file is requested, DMS detects the “end-of-file” condition. It branches to the EOFADDR address (see the EXLST macro, "EXLST - Define exit address list") and passes control to the user.

Return modes

The user can process a file in two modes which are important for the use of GET:

In **LOCATE mode**, the system transfers the record to a buffer area in the system (IOREG operand). The address of the record is received in a register specified by the user. The record is not transferred to the program area (cf. MOVE mode).

In **MOVE mode**, the user includes in his/her program an area to which the system can copy the record. The address of this record area is passed to the system via register 1 when the macro is issued.

Changing the record pointer

The record pointer can be changed by means of positioning. In this case sequential processing is interrupted and continued elsewhere: cf. SETL, SETLKY.

Special features

*For **SAM** files only*

OPEN UPDATEIn the OPEN mode UPDATE, GET reads records which are to be updated. PUTX changes the record but does not result in a write job; it just places a code in the FCB indicating that this record needs to be written. The actual write job is only executed upon a subsequent GET, RELSE, SETL; PUTX does not issue an SVC.

Tape files and blocking
With tape files in PAMKEY format with high blocking (STD,n), I/O is extended. In this case conversion to NON-KEY format or BLKSIZE=length is advisable.

Access time can be reduced by using large buffer areas. In this case SAM chains together consecutive PAM pages (chained I/O). However, significant time savings can be expected only if the files occupy largely contiguous storage space (see the SPACE operand of the FILE macro, "FILE - Define file attributes / control file processing").

A GET macro results in an SVC only when a buffer transfer is initiated. Consequently, the user cannot expect to receive control in a STXIT process each time a GET macro is issued (when using the STXIT macro with SVC= or SVCLIST=. For more details see the “Executive Macros” manual [2]).

The fact that a record has been updated in the current buffer is “noted” in the TU FCB by PUTX. It is only if this bit is set at the time of the next GET (RELSE, SETL) that the entire logical block is written to disk. PUTX **never** issues an SVC.

*For **ISAM** files only*

Reading via primary keys
With ISAM files, the logical sequence of records is defined via primary keys. With a series of GET macros, the records are read sequentially in ascending order of primary keys. A SETLKEY can be used to position the file to a given primary key. A subsequent GET will then return this record.

Reading via secondary keysWhen processing an NK-ISAM file, it is also possible to read records by specifying a secondary key. If a primary key covers several records with identical secondary keys (DUPEKEY), these records are returned by GET macros in the order in which they were created.

If the file contains records with duplicate primary keys, they will be retrieved on a “first in, first out” (FIFO) basis.

If a file is read sequentially via a secondary key, records with the same secondary key values are returned in the order in which the secondary key values were created.

Where SETL KEY is used to position the file to an existing record, a subsequent GET macro makes this record available for processing.

Format

Operation	Operands
GET	**fcbadr / (1)** **[,area / (0)]** **[,LOCK / NOLOCK]** **[,AIX = NO /** **YES,KEYNAME = name** **/** **YES,KEYNMAD = adr]** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed.

For ISAM files only:
If the file is to be located with the aid of a secondary key, the 31-bit interface of this FCB must be available.

(1)
The FCB address is located in register 1.

area

Address of the field to which the record is to be transferred when the file is processed in MOVE mode. In LOCATE mode, “area” is ignored.

(0)
The address of the field to which the record is to be transferred is stored in register 0.

LOCK

For ISAM files only:
The record or block lock is to remain active after the macro has been executed (explicit lock).

NOLOCK
No explicit lock is set.

AIX

For ISAM files only:
Specifies whether the record is to be located via its primary or secondary key.

= NO
The record is provided via its primary key.

= YES
This may only be specified if

- *the 31-bit interface of the macro is generated (via the operand PARMOD=31 or the macro GPARMOD 31) and*
- *the macro refers to a 31-bit FCB.*

The record is located via the secondary key specified in the operand KEYNAME or KEYNMAD.

KEYNAME = name

For ISAM files only:
Specifies the name of the secondary key via which the record is to be read.
“name” must be the name of a secondary key declared for the current file. The names of all secondary keys defined for a file can be determined with the SHOWAIX macro or the SHOW-INDEX-ATTRIBUTES command.

KEYNMAD = addr

For ISAM files only:
Specifies the symbolic address (the name) of a field in which the user has stored the name of the secondary key via which the record is to be read.
The field containing the symbolic address addr must contain the name of a secondary key declared for the current file at the time the macro is executed.

PARMOD

Specifies the generation mode for the macro.

Default value: the value set in the program by the assembler or by means of the GPARMOD macro.

= 24
The object code generated can run only in the 16-Mb address space (24-bit addressing only).

= 31
The object code generated can run in the 2-Gb address space (24-bit or 31-bit addressing).

Programming note

The GET macro overwrites the contents of registers 0, 1, 14 and 15.

GETFL - Read record by flag

Macro type: R for PARMOD=24
S for PARMOD=24 (with MF specification)
0 for PARMOD=31

The GETFL macro can be used only for files which were created with flags. It evaluates the flags in the ISAM index and returns to the user program the next record in a specified range (see the LIMIT operand, "GETFL - Read record by flag") which fulfills the specified conditions.
GETFL can evaluate both the value flag and the logical flag and can search the file either forwards or backwards.

Note that NK-ISAM does not include flags in the index entries and that a search for records with specific characteristics is thus executed as a sequential read operation.

For K-ISAM files, the flags are evaluated as specified in the VALPROP operand of the FILE/FCB macro and placed in the index entry. A search using GETFL is thus not executed sequentially, but via the index tree. The only difference in flag processing between NK-ISAM files and K-ISAM files which is visible to the user is the drop in performance in the case of NK-ISAM. For this reason, flag processing is not recommended for NK-ISAM files.

If VALTEST and LOGTEST are specified, both conditions must be fulfilled by the record.

If neither VALTEST nor LOGTEST is specified, the GETFL macro acts (within the limits set with LIMIT) like a GET or GETR macro. If the specified limit defined with LIMIT is reached, control passes to the EXLST exit EOFADDR (for LIMIT=END) or the EXLST exit NOFIND (for LIMIT=KEY).

If the GETFL macro is used for a file which was created without flags, control is passed to the EXLST exit USERERR.

The field to which the FCB operand KEYARG points must be large enough to accommodate the complete index (key + value flag + logical flag). The value for the value flag or the mask for the logical flag must comply formally with the corresponding flags in the records (e.g. with respect to their position and length).

Value flags

During a search for a record with a specific value flag (operand VALTEST), the appropriate flag area of each record or in the index entry is compared with the value specified in the GETFL macro. In each case, the first record within the specified limits which complies with the specified conditions is read.

Logical flags

To search for a record on the basis of the logical flag, a bit mask must be defined in the GETFL macro. This mask is then compared bit-by-bit with the logical flag in each record. Depending on the specification for the operand LOGTEST, the first record within the specified limits which complies with either one or all of the specified conditions is read.

Formats

The format of the GETFL macro varies, depending on whether the MF operand is specified and what is specified for this operand:

No MF operand	The macro generates parameters and SVC, PARMOD=24 /31, the FCB address is a symbolic address or is in register 1.
MF=L	List form: the macro generates the operand list, PARMOD=24. The FCB address is not in register 1.
MF=E	Execute form: the macro generates the SVC for MF=L.

As the operands in the format without the MF operand and in the format with MF=L are (apart from the FCB address, PARMOD and MF) identical, these two formats are not shown separately.

Operation	Operands
GETFL	**fcbadr / (1) / (r2)** **[,area / (0)]** **[,LOCK / NOLOCK]** **[,LIMIT = END / KEY]** **[,LOGTEST = ANY / ALL]** **[,REVERSE = YES]** **[,VALTEST = GT / GE / EQ / NE / LE / LT]** **[,PARMOD = 24 / 31]**
	[MF=(E,liste / E,(r1)) / L]

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed.

(1)
The FCB address is stored in register 1.

(r2)
Only in conjunction with PARMOD=24: on execution of GETFL with MF=(E,...), the FCB address is stored in the register designated by “r2” (where r2 1).

area

Address of the area to which the record is to be transferred.

(0)
The address of the area to which the record is to be transferred is stored in register 0.

LOCK

The lock is to remain active once the macro has been executed (explicit lock).

NOLOCK
The lock is not to remain active after execution of the macro.

LIMIT

Defines the end of the area to be searched. The start position is the current position in the file, depending on the preceding macro. The “search direction” depends on whether or not REVERSE=YES is specified. The search begins:

- at the first record of the file after the macro SETL B
- at the last record of the file after the macro SETL E (this is meaningful only together with REVERSE=YES)
- at the current pointer position after the macro SETL KEY
- at the record before or after the current pointer position (dependent on REVERSE=YES) for all other macros

  = END
  The search continues until the end (or beginning) of the file is reached. If the file contains no records whose flags fulfill the specified conditions, the EXLST exit EOFADDR is activated.

  = KEY
  The limit is defined by a key to which the FCB operand KEYARG points. The search is aborted when it encounters a record with the same key as that referenced by KEYARG.

  - without REVERSE=YES: only records with keys less than the key addressed via KEYARG are scanned.
  - with REVERSE=YES: only records with keys greater than the key addressed via KEYARG are scanned.

  If the range defined via a key contains no records whose flags fulfill the conditions in the GETFL macro, control is passed to the EXLST exit NOFIND. This is true even if the file is already positioned to the record which contains the LIMIT key; in this case, the file position remains unchanged.
  If the key addressed via KEYARG is less than the key at the current pointer position (or greater than this if REVERSE=YES is specified), control passes to the EXLST exit USERERR.

LOGTEST

Specifies, for a search with logical flags, whether the records to be retrieved must fulfill all of the specified conditions or whether compliance with only one condition is sufficient. The bit mask must be specified in the field to which the FCB operand KEYARG points. At least one bit in the mask must be set; otherwise control is passed to the EXLST exit USERERR.

= ANY
The search retrieves the next record whose logical flag matches at least one of the bits of the mask.

= ALL
The search retrieves the next record in whose logical flag all of the bits set in the bit mask are also set.

MF = (E,...)

Generates the SVC: the operand list generated with MF=L is used for execution of the macro.

= E,addr
Address of the operand list generated with MF=L. If the addresses of the FCB and “area” are to be passed in registers, these registers must be loaded with valid addresses before the macro is called.

= E,r
The address of the operand list generated with MF=L is stored in register “r”.

MF = L

Only for PARMOD=24:
an 8-byte operand list is generated; the macro is not executed. The operand list is aligned on a word boundary and contains:

- operand byte 1 (see table "Operand byte 1")
- the FCB address or the number of the register which contains the FCB address (=1)
- operand byte 2 (see table "Operand byte 1")
- the address of the area to which the record is to be transferred or the number of the register which will contain this address for a macro call with MF=E.

The operand list must be symbolically addressable (= symbolic address of the macro).

PARMOD

Specifies the generation mode for the macro.

Depending on PARMOD and MF=L, different “operand lists” are generated.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The object code generated can run only in the 16-Mb address space. If MF=L is specified, an 8-byte operand list is generated; if MF=L is not specified, the operands are transferred to registers 0 and 1.

= 31
The object code generated can run in the 2-GByte address space. The information required for macro processing is contained in the FCB.

REVERSE = YES

Processing is performed “in reverse” (towards the beginning of the file).

Default value: processing is performed forwards (towards the end of the file).

VALTEST

Specifies the desired relationship between the value flag of the record to be retrieved and the value to which the FCB operand KEYARG points.

= GT
The value in the record must be greater than the value in the KEYARG field.

= GE
The value in the record must be greater than or equal to the value in the KEYARG field.

= EQ
The value in the record must be equal to the value in the KEYARG field.

= NE
The value in the record must not be equal to the value in the KEYARG field.

= LE
The value in the record must be less than or equal to the value in the KEYARG field.

= LT
The value in the record must be less than the value in the KEYARG field.

Operand list for MF=L

Word 1	...	**Word 2**	...
Operand byte 1	FCB address or register number	Operand byte 2	Area address or register number

Operand byte 1:

Encoded GETFL operand and LOGTEST (see the table "Operand byte 1").

FCB address/register:

Must specify either the address of the FCB or a register (right-justified) containing the FCB address.

Operand byte 2:

Encoded information relating to the GETFL operands LOCK/NOLOCK, fcbaddr, area (see the table "Operand byte 2").

Area address/register:

Must specify either the address of the area or a register (right-justified) containing the address of the area to which the record is to be transferred.

Operand byte 1

Bit position/bit pattern								
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Meaning**
0	0	1	0					VALTEST=GT
0	1	0	0					VALTEST=LT
1	0	0	0					VALTEST=EQ
0	1	1	1					VALTEST=NE
1	0	1	0					VALTEST=GE
1	1	0	0					VALTEST=LE

0	0	0	0					VALTEST=0 or invalid
				1 0				LOGTEST operand specified LOGTEST operand “null” or invalid
					1 0			LOGTEST=ALL LOGTEST=ANY
						1 0		LIMIT=KEY LIMIT=END
							1 0	REVERSE=YES REVERSE=null

Operand byte 2

Bit pattern/bit position (3-0 not used)					
7	**6**	**5**	**4**	**Meaning PARMOD=24**	**Meaning PARMOD=31**
1 0				LOCK specified or default setting NOLOCK specified	LOCK specified or default setting NOLOCK specified
	1 0			FCB address contained in register FCB address specified	- not used -
		1 0		'area' address not specified 'area' address specified	- not used -
			1 0	'area' address contained in register 0 'area' address	- not used -

Programming note

The GETFL macro overwrites the contents of registers 0, 1, 14 and 15.

Overview of the EXLST exits

EXLST exit	**Related GETFL operand**	**Meaning**
EOFADDR	LIMIT = END	No matching record found
NOFIND	LIMIT = KEY	• No matching record in defined range • Limit is identical with current pointer position
USERERR	---	File was created without flags or other user error (such as an invalid OPEN mode)
	LIMIT = KEY	Limit has already been exceeded
	LOGTEST	Bit mask for logical flag contains only zeros (no bits set)

GETKY - Get record with specified key

Macro type: R for PARMOD=24
O for PARMOD=31

The GETKY macro retrieves the record whose primary or secondary key matches the value in the KEYARG field. The KEYARG field is addressed via the KEYARG operand of the FCB macro.

If no record with the specified key value is found, control is returned to the user program via the NOFIND address (see the EXLST macro, NOFIND operand, "EXLST - Define exit address list"). The pointer for the primary or secondary key is set to the value for which the search was made.

If a file contains several records with the same value for the primary or secondary key specified in the GETKY macro, either the first of these records (if the primary key is used) or the record to which the first pointer in the secondary index block refers (if a secondary key is used) is returned.

Format

Operation	Operands
GETKY	**fcbadr / (1)** **[,area / (0)]** **[,LOCK / NOLOCK]** **[,AIX = NO /** **YES,KEYNAME = name** **/** **YES,KEYNMAD= adr]** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed.

If the file is to be read with the aid of a secondary key, the 31-bit interface of this FCB must be available.

(1)
The FCB address is stored in register 1.

area

Address of the area to which the record is to be transferred.; in locate mode, “area” is ignored.

(0)
The address of the area into which the record is to be transferred is stored in register 0.

LOCK

The block or record lock is to be retained after the macro has been executed (explicit lock).

NOLOCK
No explicit lock is set.

AIX

Specifies whether the record is to located via its primary key or via a secondary key.

= NO
The record is located via its primary key (default value).

= YES
This may be specified only if

- *the 31-bit interface of the macro is generated (via the operand PARMOD=31 or the macro GPARMOD 31) and*
- *the macro refers to a 31-bit FCB.*

The record is located via the secondary key specified in the operand KEYNAME or KEYNMAD.

KEYNAME = name

Specifies the name of the secondary key via which the record is to be located.
“name” must be the name of a secondary key defined for the current file. The names of all secondary keys defined for a file can be determined by means of the SHOWAIX macro or the SHOW-INDEX-ATTRIBUTES command.

KEYNMAD = addr

Specifies the symbolic address (the name) of a field in which the user has stored the name of the secondary key via which the record is to be located.
When the macro is executed, the field with the symbolic address “addr” must contain the name of a secondary key defined for the current file.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The object code generated can run only in the 16-Mb address space
(24-bit addressing only).

= 31
The object code generated can run in the 2-Gb address space
(24-bit or 31-bit addressing).

Programming note

The GETKY macro overwrites the contents of registers 0, 1, 14 and 15.

GETR - Get record "reverse"

Macro type: R for PARMOD=24
O for PARMOD=31

The GETR macro reads the next record of the file in the direction of the beginning of the file, i.e. the file is read in reverse.

If a record outside the file is requested, the user is given control via EOFADDR (see the EXLST macro, EOFADDR operand, "EXLST - Define exit address list").

The program can switch from GET to GETR and vice versa at any time, without the file having first to be positioned to beginning-of-file or end-of-file.

If a GET macro which returned a record with the primary or secondary key value K_n is followed by a GETR macro referring likewise to the primary key or the same secondary key, respectively, then this GETR call returns the record with the next lower primary or secondary key value K_{n-1} ($K_{n-1} < K_n$).

If the file contains records with duplicate primary key values, GETR returns the records on the “last in, first out” (LIFO) principle, i.e. the first record to be returned from a group of records which have the same key is the most recent record written to the file.

If a file is read with the aid of a secondary key and if it contains records with identical values for this secondary key, then GETR returns the records in the reverse order to that in which the secondary key values were created.

If the GETR macro is preceded by a SETL KEY macro, the record to which the file was positioned via SETL KEY is returned.

Format

Operation	Operands
GETKY	**fcbadr / (1)** **[,area / (0)]** **[,LOCK / NOLOCK]** **[,AIX = NO /** **YES,KEYNAME = name** **/** **YES,KEYNMAD = adr]** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed. If the file is to be read with the aid of a secondary key, the 31-bit interface of this FCB must be available.

(1)
The FCB address is stored in register 1.

area

Address of the area to which the record is to be transferred. This operand is ignored in locate mode.

(0)
The address of the area to which the record is to be transferred is stored in register 0.

LOCK

The block or record lock is to remain active after the macro has been executed (explicit lock).

NOLOCK
No explicit lock is set.

AIX

Specifies whether the record is to located via its primary key or via a secondary key.

= NO
The record is located via its primary key (default value).

= YES
This may be specified only if

- the 31-bit interface of the macro is generated (via the operand PARMOD=31 or the macro GPARMOD 31) and
- the macro refers to a 31-bit FCB.

The record is located via the secondary key specified in the operand KEYNAME or KEYNMAD.

KEYNAME = name

Specifies the name of the secondary key via which the record is to be located.
“name” must be the name of a secondary key defined for the current file. The names of all secondary keys defined for a file can be determined by means of the SHOWAIX macro or the SHOW-INDEX-ATTRIBUTES command.

KEYNMAD = addr

Specifies the symbolic address (the name) of a field in which the user has stored the name of the secondary key via which the record is to be located.
When the macro is executed, the field with the symbolic address “addr” must contain the name of a secondary key defined for the current file.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The object code generated can run only in the 16-Mb address space
(24-bit addressing only).

= 31
The object code generated can run in the 2-Gb address space (24-bit or 31-bit addressing).

Programming note

The GETR macro overwrites the contents of registers 0, 1, 14 and 15.

IDBPL - Provide BTAM operand list with symbolic names

Macro type: type O

The IDBPL macro is used to generate a dummy section (Dsect) which provides the individual fields of the BTAM macro with symbolic names.

Format

Operation	Operands
IDBPL	**[D]** **[,prefix / ,*]** **[,PARMOD = 24 / 31]**

Operand descriptions

D

Specifies that a Dsect statement is to be generated. If “D” is not specified, no Dsect statement will be generated.

prefix

Specifies the prefix (1 character) with which each symbolic name is to begin.

Default value: each name is prefixed by the letter “I”.

*****
Specifies that no prefix is to be placed in front of the name.

PARMOD

Specifies the generation mode for the macro.

Default value: the value preset for the generation mode by means of the GPARMOD macro or by the assembler.

= 24
The macro is expanded in accordance with the format for the 24-bit interface. The object code is thus executable only in 24-bit addressing mode.

= 31
The macro is generated as addressing mode-independent.

IDFCB - Provide FCB with symbolic names

Macro type: type O

The IDFCB macro serves to generate a (Dsect) for the FCB, so that the user can address all the FCB fields symbolically, provided a base register is initialized appropriately.

Format

Operation	Operands
IDFCB	**[D]** **[,prefix / ,*]** **[,PARMOD = 24 / 31]**

Operand descriptions

D

This operand specifies that a Dsect statement is to be generated. By default, the system does not generate the macro as a Dsect.

prefix
Prefix (1 character) which is to be inserted in front of all Dsect names; by default, names are generated with the prefix “I”.

*****
No prefix will be used.

PARMOD
Specifies the generation mode for the macro.

Default value: the value preset for the generation mode by means of the GPARMOD macro or by the assembler.

= 24
The macro is expanded in accordance with the format for the 24-bit interface. The object code is thus executable only in 24-bit addressing mode. A Dsect is generated for the “old” FCB (BS2000 <= V8.5); i.e. for the FCB extension, a Dsect must be generated as before by means of the IDFCBE macro (see the macro FCB, "FCB - Define file control block", and IDFCBE macro, "IDFCBE - Provide FCBE with symbolic names").

= 31
The macro is generated as addressing-mode-independent. There is no FCB extension for the new FCB (BS2000 >= V9.0), and therefore no IDFCBE macro is required either.

IDFCBE - Provide FCBE with symbolic names

Macro type: type O

The IDFCBE macro generates a Dsect for the FCB extension of the 24-bit TU FCB. If a base register is initialized appropriately, the user can symbolically address the fields in the FCB extension.

This FCB extension is only created if the 24-bit FCB has been generated, and if the BUFOFF, FSEQ and LABEL operands in the FILE or FCB macro were specified for tape processing. If this is not the case, then no IDFCBE macro will be needed.

In order to determine the start address of the FCBE, the IDFCB macro (with PARMOD=24) is required in addition to the IDFCBE macro.

As the 31-bit FCB has no FCB extension (FCBE), the IDFCBE macro is not supported in an XS environment.

Format

Operation	Operands
IDFCBE	**[D]** **[,prefix / ,*]**

Operand descriptions

D

This operand specifies that a Dsect statement is to be generated; by default, no Dsect is generated.

prefix

Prefix (1 character) which is to be inserted in front of all Dsect names; by default, this is the letter “I”.

*****

No prefix will be used.

IDPPL - Provide PAM operand list with symbolic names

Macro type: type O

The IDPPL macro serves to generate a dummy section (Dsect) which supplies the individual PAM macro fields with symbolic names

Format

Operation	Operands
IDPPL	**[D]** **[,prefix / ,*]** **[,PARMOD = 24 / 31]**

Operand descriptions

D

Specifies that a Dsect statement is to be generated. Unless otherwise specified, a CSECT statement is specified rather than a Dsect statement.

prefix
Specifies the prefix, i.e. the character with which each symbolic name is to begin.

*****
No prefix will be used.

PARMOD
Specifies the generation mode for the macro.

Default value: the value preset for the generation mode by means of the GPARMOD macro or by the assembler.

= 24
The macro is expanded in accordance with the format for the 24-bit interface. The object code is thus executable only in 24-bit addressing mode.

= 31
The macro is generated as addressing-mode-independent.

IMPNFIL - Create (import) catalog entries for node files

Macro type: type S (E form / L form / D form / C form / M form) (see "Macro types")

The IMPNFIL macro catalogs node files stored on Net-Storage volumes for which the calling job has the ownership right. DMS creates the catalog entry for a node file in the TSOSCAT and in the file catalog of the Net-Storage volume of the NETSTOR type on the basis of the Inode attributes on the NFS server. For node files on a Net Storage volume of the NETVOL type, the catalog entry is only created in TSOSCAT, also based on the inode attributes on the NFS server. There is no file catalog on the Net-Storage volume for files on a Net-Storage of the NETVOL type.

By specifying a partially qualified file name or wildcard the user can also import more than one file with one call.

Notes

- Co-owners of a user ID may import node files under this ID.
- When entries in the user catalog need to be replaced (REPLACE= *YES/*NFU), these may not be locked and write access must be permissible.

Format

Operation	Operands
IMPNFIL	**,VOLUME=<c-string: 1..6> / <var: char:6>** **,FILENAM=<c-string 1..80>:<filename 1..54** **with-wild-without-cat(80)> / <var: char:80>** **,PUBSET=*STD / <c-string: 1..4> / <var: char:4>** **,FILESTR=*STD / *PAM / *SAM / <var: enum-of_filestr_s: 1>** **,REPLACE=*NO / *YES / *NFU / <var: enum-of_replace_s: 1>** **,IGNPROT=*NO / *YES / <var: enum-of_ignprot_s: 1>** **,LIST=*NO / *SYSOUT / *SYSLST / *BOTH / <var: enum-of_list_s: 1>** **,REPORT=*ERROR / *FULL / <var: enum-of_report_s: 1>** **,EQUATES=*YES / *NO** **MF=L**
	MF=D,PREFIX= D / <pre>
	MF=E,PARAM=<name 1..27>

	MF=C / M
	,PREFIX= D / <pre>
	,MACID= MAN / <macid>

Operand descriptions

VOLUME

Volume serial number (VSN) of the Net-Storage volume on which the node files to be imported are stored.

=<c-string: 1..6>
VSN of the Net-Storage volume.

=<var: char:8>
Only possible with MF=M:
Symbolic address of a memory area of 6 bytes in which the VSN of the Net-Storage volume is stored.

FILENAM
Selection of the node files which are to be imported.

=<c-string 1..80: filename 1..54 with-wild-without-cat(80)>
Path name of the node file on the Net-Storage volume. A catalog ID may not be specified. Specification of a wildcard enables the selection of a file set.

Nonprivileged users can only import files of their user ID. Privileged users (TSOS privilege) can also import files of other users. Wildcards may be specified in the user ID.

=<var: char:80>
Only possible with MF=M:
Symbolic address of a memory area of 80 bytes in which the path name or the wildcard string for the desired file(s) is stored.

PUBSET
Determines the pubset in which the files are to be cataloged. The Net-Storage volume specified in the VOLUME operand must be assigned to the pubset specified here.

=*STD
The catalog entries are configured in the file catalog of the default pubset of the user ID.

=<c-string: 1..4>
Pubset Id of the pubset. The catalog entries are configured in the specified pubset' file catalog.

=<var: char:4>
Only possible with MF=M:
Symbolic address of a memory area of 4 bytes in which the pubset Id is stored.

FILESTR
Determines the FILE-STRUCTURE attribute of the node file which is entered in the file catalog in accordance with the REPLACE operand.

=*STD
The following applies when REPLACE=*NO/*YES: A node file is imported into BS2000 as a PAM file when the file size on the NFS file system is not equal to zero. If the file size on the NFS file system is equal to zero, the imported file is assigned the default attributes of a file generated with CREATE-FILE.
The following applies when REPLACE=*NFU: The catalog entries of the node files are updated in BS2000 irrespective of the FILE-STRUCTURE attribute.

=*PAM
The following applies when REPLACE=*NO/*YES: A node file is imported into BS2000 as a PAM file irrespective of the file size on the NFS file system.
When REPLACE=*NFU, the catalog entries of PAM node files in BS2000 are updated.

=*SAM

For REPLACE=*NO/*YES the following applies: a node file will be imported into BS200 as a SAM file, irrespective of the file size on the NFS file system. At node file creation, the Net-Storage coded character set is entered according to its definition in the user entry. The resulting NETCCS of the file is identified based on the following table:

CCS entry [1]	**NETCCS entry** [1]	**Resulting NETCCS in the catalog entry of the node file**
EDF03IRV/*NONE	*ISO	ISO88591; during code conversion, EDF041 is assumed for CCS

EDF03DRV	*ISO	ISO88591; during code conversion, EDF04DRV is assumed for CCS
EDF04DRV	*ISO	ISO88591
EDF04x	*ISO	ISO8859x with x=1,2,..F
ISO8859x	*ISO or *NO-CONV	ISOx
UTFx	*ISO or *NO-CONV	UTFx
<name_a 1..8>	<name_b 1..8>	<name_b 1..8>
<name_a 1..8>	*NO-CONV	<name_a 1..8>

[1]User entry (SYSSRPM) or CATALOG or CREATE-FILE or MODIFY-FILE-ATTRIBUTES entry

If REPLACE=*NFU, the catalog entries of SAM node files are updated in BS2000.

=<var: enum-of_filestr_s: 1>
Name of the field with the value for FILESTR.

REPLACE
Specifies whether files which already exist in BS2000 are replaced or whether only the catalog entries are updated on the NFS server on the basis of the Inode attributes.

=*NO
Files which already exist are not replaced, nor are their catalog entries updated.

=*YES
Files which already exist on the pubset are replaced by the specified node files. Any files on public space or on Net-Storage are deleted, and files on private disk are exported. When the node files are imported, the entries in the TSOSCAT and in the file catalog of the Net-Storage volume are created anew.

=*NFU
In the case of files which already exist, the entries in the TSOSCAT and in the file catalog of the Net-Storage volume are updated on the basis of the Inode attributes on the NFS server. Here the FILESTR operand determines that the update of the catalog

entries only takes place for files with the specified file structure. When FILESTR=*STD, the files are updated irrespective of the file structure.

=<var: enum-of_replace_s: 1>
Name of the field containing the value for REPLACE.

IGNPROT
This operand is only available to privileged users (TSOS privilege).
Specifies whether files which are already cataloged are to be overwritten regardless of an existing write protection.

=*NO
The write protection is observed.

=*YES
The write protection is ignored.

=<var: enum-of_ignprot_s: 1>
Name of the field containing the value for IGNPROT.

LIST
Specifies whether a processing log is to be output to SYSOUT and/or SYSLST. The default value is *NONE, i.e. no log is created.

=*NO
No output takes plce.

=*SYSOUT
The processing log is output to SYSOUT.

=*SYSLST
The processing log is output to SYSLST.

=*BOTH
The processing log is output to SYSOUT and SYSLST.

=<var: enum-of_list_s: 1>
Name of the field containing the value for LIST.

REPORT
Determines the scope of the log when a processing log was requested in the LIST operand.

=ERROR
Only files which could not be imported are listed. The reason is displayed in each case with a message code.

=*FULL
All files are listed. The reason is specified with a message code for each file which could not be imported.

=<var: enum-of_report_s: 1>
Name of the field containing the value for REPORT.

EQUATES

Control operand only for MF=C and MF=D:
Specifies whether equates should also be generated for the values of the parameter area's fields when the parameter area is expanded.

= *YES
Equates are also generated for the values of the fields of the parameter area when the parameter area is expanded.

= *NO
No equates are generated for the values of the fields of the parameter area when the parameter area is expanded.

Return codes

The return code is placed in the standard header of the parameter area. The parameter area may then not be located in the read-only area, otherwise the program terminates.

Standard header: `ccbbaaaa`

The following code relating to execution of the IMPNFIL macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
X'00'	X'00'	X'0000'	No error
X'00'	X'40'	X'0501'	CMS or FILE: Requested catalog not available
X'00'	X'40'	X'0512'	Pubset Id is not entered in the MRSCAT
X'00'	X'40'	X'051B'	User ID not known in specified pubset
X'00'	X'40'	X'051C'	User has no access right to specified pubset
X'00'	X'40'	X'0535'	There is no access right to the file catalog entry
X'00'	X'20'	X'0578'	Internal error in the file protection check
X'00'	X'82'	X'0594'	Not enough virtual memory available
X'00'	X'20'	X'05C7'	Internal error in DMS
X'00'	X'01'	X'05EE'	Path name too long after completion
X'00'	X'40'	X'05FC'	Specified user ID not on home pubset
X'00'	X'40'	X'0610'	Execution of the function returned a return code for at least one of the selected file names
X'00'	X'01'	X'0624'	File name invalid
X'00'	X'40'	X'0640'	Access to Net-Storage is rejected by the ONETSTOR subsystem because of communication problems with the net client
X'00'	X'04'	X'0642'	Large files not permitted on pubset
X'00'	X'40'	X'0643'	Net client reports access error
X'00'	X'40'	X'0644'	Net client reports internal error
X'00'	X'40'	X'0645'	File does not exist on Net-Storage
X'00'	X'40'	X'0649'	Net server reports ACL error
X'00'	X'40'	X'064A'	Net client reports that access to files on the Net-Storage volume is forbidden
X'00'	X'40'	X'064B'	Access to node files from the net client not supported
X'00'	X'40'	X'064C'	Directory of the specified user ID does not exist on net server
X'00'	X'40'	X'064D'	File is not a node file
X'00'	X'40'	X'064E'	Node file not located on the specified Net-Storage volume
X'00'	X'40'	X'064F'	FCB-TYPE of file and specified file structure do not match
X'00'	X'40'	X'0650'	No node file found which can be imported or updated
X'00'	X'40'	X'0651'	File exists, import not possible

X'00'	X'40'	X'06CC'	Only with wildcard selection: No file matches the specified selection entry
X'00'	X'01'	X'FFFF'	Wrong function number in standard header
X'00'	X'03'	X'FFFF'	Wrong version number in standard header

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller
- the list is not aligned on a word boundary
- the list is write-protected.

Layout of the operand list

Macro expansion with MF=D, and default values for EQUATES, PREFIX and MACID:

```
IMPNFIL MF=D
        MFTST MF=D,PREFIX=D,MACID=MAN,ALIGN=F,
             DMACID=MAN,SUPPORT=(E,D,C,M,L),DNAME=MANGLPL
DMANGLPL DSECT ,
              *,##### PREFIX=D, MACID=MAN #####
*   PARAMETER AREA
DMANHDR  FHDR  MF=(C,DMAN),EQUATES=NO
DMANHDR  DS    0A
DMANFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
*
DMANIFID DS    0A              0   INTERFACE IDENTIFIER
DMANFCTU DS    AL2             0   FUNCTION UNIT NUMBER
*                                  BIT 15    HEADER FLAG BIT,
*                                  MUST BE RESET UNTIL FURTHER NOTICE
*                                  BIT 14-12 UNUSED, MUST BE RESET
*                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
DMANFCT  DS    AL1             2   FUNCTION NUMBER
DMANFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
*
DMANRET  DS    0A              4   GENERAL RETURN CODE
DMANSRET DS    0AL2            4   SUB RETURN CODE
DMANSR2  DS    AL1             4   SUB RETURN CODE 2
DMANSR1  DS    AL1             5   SUB RETURN CODE 1
DMANMRET DS    0AL2            6   MAIN RETURN CODE
DMANMR2  DS    AL1             6   MAIN RETURN CODE 2
DMANMR1  DS    AL1             7   MAIN RETURN CODE 1
DMANFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
*
DMANVOLUM DS    CL6                    VOLUME
DMANFNAME DS    CL80                   FILENAME
DMANPUBID DS    CL4                    PUBSET
DMANFILS DS    FL1                    FILESTRUC
*   FILESTRUC VALUES
DMANSTDF EQU   0                      FILESTRUC = STD
DMANPAMF EQU   1                      FILESTRUC = PAM
DMANSAMF EQU   2                      FILESTRUC = SAM
*
DMANREPL DS    FL1                    REPLACE
```

```
*   REPLACE VALUES
DMANREPN EQU   0                        REPLACE=NO
DMANREPY EQU   1                        REPLACE=YES
DMANREPU EQU   2                        REPLACE=NODE FILE
*
DMANIGNP DS    FL1                      IGNPROT
*   IGNPROT VALUES
DMANIGNO EQU   0                        IGNPROT = NO
DMANIGYE EQU   1                        IGNPROT = YES
*
DMANLIST DS    FL1                      LIST
*   LIST VALUES
DMANLISN EQU   0                        LIST = NO
DMANOUTO EQU   1                        LIST = SYSOUT
DMANOUTL EQU   2                        LIST = SYSLST
DMANOUTB EQU   3                        LIST = BOTH
*
DMANREPO DS    FL1                      REPORT
*   REPORT VALUES
DMANREPE EQU   0                        REPORT = NO
DMANREPF EQU   1                        REPORT = FULL
*
DMANRES1 DS    XL5                      ALIGNMENT
DMAN#    EQU   *-DMANHDR
```

Sample calling sequence

```
MVC   IMPNMFC(DMAN#),IMPNMFL
         IMPNFIL MF=M,                                              -
               PARAM=IMPNMFC,                                       -
               VOLUME='P@BX00',FILENAM='*',                         -
               PUBSET='X'
         IMPNFIL MF=E,PARAM=IMPNMFC
             .
             .
IMPNMFC  IMPNFIL MF=C
IMPNMFL  IMPNFIL MF=L,VOLUME='*DUMMY',FILENAM='AAA'
```

IMPORT - Create catalog entry for files

Macro type: type S (E form/L form/D form/C form); see "Macro types"

The IMPORT macro catalogs files for which the calling job has the ownership rights and which are stored on private disks or Net-Storage volumes of the NETSTOR type. The DMS takes the file attributes from the F1 label of the private disk or from the catalog of the Net-Storage volume of the NETSTOR type and places them in the catalog entry. Files on a Net-Storage volume of the NETVOL type cannot be imported because there is no file catalog on a Net-Storage volume of the NETVOL type. The macro can process partially qualified file names, which means that the user can import several files using a single macro.

When importing file generation groups with generations stored on different disks, it should be noted that generations are cataloged only if the group entry already exists in the system catalog or is kept on the first disk to be imported. Otherwise, the catalog entries for the generations imported before the group entry will be missing. These generations must then be cataloged afterwards by means of an IMPORT or FILE macro (STATE=FOREIGN).

The functions of the macros IMPORT and ERASE (operands CATALOG or DELETE-OR-EXPORT and VOLUME, respectively) are not exact opposites. When a volume is exported, DMS deletes the catalog entries for all files which occupy storage space on this volume. If the same volume is later imported, DMS creates catalog entries only for those files which begin on the volume (i.e. files which received space on the volume during their primary allocation).

Notes

- Co-owners of a user ID can create permanent files under this ID.
- Locked entries can be imported from the F1 label or from the catalog of the Net-Storage volume of the NETSTOR type. However, if entries in the user catalog have to be replaced (REPLACE=YES/ABS), the entries must not be locked and write access must be permitted.

Format

Operation	Operands
IMPORT	**[pathname],VOLUME = vsn,DEVICE = device** **[,AREA = (adr,length)][,REPLACE = NO / YES / ABS]** **[,GEN = YES / NO][,LIST = YES / NO / ONLY]** **[,PVSID = catid]** **[,NUSERID = userid]** **[,MF = L][,VERSION = 1]**
	MF = (E,adr / E,(r))[,VERSION = 1]
	MF = D / C **[,PREFIX = pre / *]** **[,VERSION = 1]**

Operand descriptions

AREA

Specifies the output area for the IMPORT macro. This operand may be omitted if LIST=NO is specified.

= (addr,length)

addr symbolic address of the output area

length length of the output area.

DEVICE = device

Device type on which the volume is to be mounted. See the device table in the „System installation“ manual [16] for possible entries for “vsn”. The new device types introduced after BS2000 Version 9.5 are only supported in conjunction with the VERSION=1 operand. For Net-Storage volumes the volume type NETSTOR must be specified instead of the device type.

Every specification of a disk device type which is known to the system is handled like the STDDISK specification.

GEN

For file generation groups:
specifies whether only the group entry or also the file generations stored on the same private disk is/are to be cataloged.

= YES
If the group entry is on the private disk, DMS catalogs the FGG and all related generations which are on this disk. If there is no group entry on the disk or in the user catalog, no file generations are cataloged.

= NO
DMS transfers only the group entry for the FGG to the catalog.

LIST

Specifies how macro execution is to be logged (see "Programming notes").

= YES
Macro execution is logged.

= NO
No information about macro execution is returned.

= ONLY
Causes execution of the IMPORT macro to be simulated, not actually carried out, i.e. the user receives a SYSLST log which shows how the IMPORT macro would have been executed. The log contains (depending on “pathname”) a list of the files on the volume specified by VOLUME, together with the messages and information returned by the IMPORT macro.

DMS does not check, at this time, whether file locks or protection attributes would prevent files from being imported. For the actual import, the user must ensure that the files are not locked and that write access is permitted.

MF

The forms of the MF operand are described in detail in the appendix ("Macro types"). In all macros differentiated solely by the MF operand (MF=L/E/D/C), the version operand must have the same value.

NUSERID = userid

Only permitted for system administration:
User ID under which the file is to be cataloged. The new user ID is specified without $ and without ".".
A file on private disk is assigned the new user ID both in the file catalog and in the F1 label of the disk.
A BS2000 file on a Net-Storage volume is assigned the new user ID both in file catalog and in the catalog of the Net-Storage volume.
A node file, by contrast, cannot be cataloged under the new user ID as an owner of node files may not be modified. In this case the import is rejected with return code D.

pathname

Designates the files, file generation groups or file generations to be cataloged, with:
<c-string 1..54: filename 1..54> (a partially qualified file name is also permitted).

If "pathname" is not specified, DMS catalogs all files, etc. which are stored under the user ID of the current job on the volume specified in the VOLUME operand.

"pathname" means [$userid.]filename

userid
User ID: the user can only import files for which he/she has owner rights. Default user ID: the user ID of the current job (i.e. of the SET-LOGON-PARAMETERS or LOGON command).

filename
Fully or partially qualified name of a file, file generation group or file generation.

For file generations and file generation groups, the group entry must be created before cataloging the generations!

PREFIX

This operand is only relevant in combination with MF=D/C.

Default value: I

= pre
Specifies a prefix for all names used in a Dsect. Only a single-character is permitted.

= *
Specifies that no prefix is to be used.

PVSID = catid

Specifies the pubset in which the files are to be cataloged. If this is omitted, the catalog entries are created under the default catalog ID of the user ID.

When a Net-Storage volume is specified in the VOLUME operand, this Net-Storage volume must be assigned the pubset to whose catalog the entries are imported. The catalog entry in the pubset is then updated with the data of the Net-Storage volume's catalog entry.

REPLACE

Specifies whether an existing “old” catalog entry is to be overwritten.

= NO
DMS does not overwrite any existing catalog entry.

= YES
The old catalog entry is deleted if it does not agree with the specifications in the IMPORT macro:

- The cataloged file is stored on a public disk: the catalog entry is deleted, which means that the public file is erased (providing the protection attributes permit this and the file is not locked; otherwise, the old catalog entry is left unchanged).
- The cataloged file is on private disk, but begins on a volume other than the one specified in the VOLUME operand: the catalog entry is overwritten (providing the protection attributes permit this and the file is not locked; otherwise, the old entry remains unchanged).
- The cataloged file is stored on a Net-Storage volume. A file on the same Net-Storage volume with the same name is not imported and the catalog entry is not deleted.
- The cataloged file is stored on a Net-Storage volume. A file on private disk, on another Net-Storage volume or on the same Net-Storage volume but which has a different name (not a node file) is imported: the catalog entry is overwritten and the file is thus deleted (if this is not prevented by a file lock or protection attributes, otherwise the old entry is retained). In this case deleting the file means:
  - A BS2000 file on Net-Storage of the type NETSTOR is also deleted on the Net-Storage volume.
  - A node file is retained on the Net-Storage volume.

> ¡ Node files, in contrast to BS2000 files, cannot be imported to a different user ID as an owner of node files may not be modified.

- The cataloged file is contained on private disk and begins on the disk specified in the VOLUME operand: the catalog entry is *not* deleted (exported). A file of the same name is *not* imported.

= ABS
The old catalog entry is overwritten even if the catalog entry and the specifications in the IMPORT command match each other. The return code shows whether the entry was overwritten (return code 8) or whether a file lock prevented overwriting (return code 9).

VERSION = 1

Controls macro generation. The operand list and, if applicable, the SVC valid for BS2000 versions from V9.5 upwards are generated.

Default value: the operand list and the SVC are generated as for BS2000 versions < V9.5.

VOLUME = vsn

The volume serial number (“vsn”) of the volume on which the files to be imported are stored.

If files are to be imported from a Net-Storage (DEVICE = NETSTOR), the archive number ("vsn") is used to decide which type of Net-Storage volume is involved. Files on a Net-Storage volume of the NETSTOR type are imported.

Programming notes

Each element returned is 56 bytes long and has the following structure:

pathname (54 bytes) + Return code (2 bytes)

Return codes

Only the rightmost byte of the return code is of importance to the user. This byte supplies additional information on the processing of the IMPORT macro if the return code in register 15 is X'00'.

Return code	Meaning
C'0'	There was no file with the same name and a new catalog entry has been created.
C'1'	There was already a file with the same name and this was overwritten; together with LIST=ONLY: a file with this name already exists, protection attributes have not been checked.
C'2'	A file with the same name exists but was not overwritten; the REPLACE operand had the value NO
C'3'	A file with the same name exists and could not be erased due to the protection function (ACCESS=READ, WRPASS, etc.) or the file is locked because it is being processed.
C'4'	System error during catalog access.
C'5'	The file is already cataloged and is stored on the volume specified in the VOLUME operand.
C'6'	System error during access to the F1 label of the private disk or to the catalog of the Net-Storage volume.
C'7'	Invalid attempt to import a file generation: the absolute generation number of the generation to be imported conflicts with the limits defined in the group entry.
C'8'	A catalog entry already existed for the specified disk and has been replaced.
C'9'	A catalog entry already exists for the specified disk, but the file is locked.
C'A'	The path name of the file to be imported (together with catalog ID and $userid) is longer than 54 characters.
C'B'	Error while accessing the Net-Storage.
C'C'	The file to be imported is larger than 32 GB, but the pubst specified does not permit files which are larger than 32 GB.
C'D'	Node files may not be imported to a different user ID.

A file on the volume has been processed successfully if the return code is C'0', C'1', C'5' or C'8'.

The following table describes the left byte of the return code. This byte is only significant if the system administrator is using the NUSERID operand. The specifications refer to the entries in the system catalog under Ouserid.

Return code for system administrator	Meaning
C'0'	There is no entry in the system catalog under the userid entered in the F1 label of a private disk or in the catalog of the Net-Storage volume.
C'1'	A file cataloged with the same name was deleted. If the operand LIST=ONLY was specified, this value simply means that a file of the same name already exists. The protection attributes are not checked in this case.
C'2'	A file of the same name already exists; the value of the REPLACE operand was NO.
C'3'	The cataloged file is protected (error on deleting this file).
C'4'	System error on reading the catalog.
C'D'	Node files may not be imported to a different user ID.

System behavior for an overflow of the output area: A user area that is too small is indicated by R15 = 05AB. The import operation is carried out in any case. For a layout of the output area, see above; end criterion X'FF' to offset: last entry +X'36'.

The return code is placed in register 15. If the macro executes normally, the contents of register 15 are set to null. The possible return codes of DMS can be generated by the IDEMS macro.

INSRT - Insert record

Macro type: R for PARMOD=24
0 for PARMOD=31

The INSRT macro transfers a record from the user area to the file and inserts it at the position defined by the value of its record key.

If the file already contains a record with the specified key, the new record is not transferred, even if DUPEKY=YES was defined in the FCB macro. Control passes to EXLST exit DUPEKY.
Insertion of a record with a key that already exists in the file is only possible using STORE; sequential extension of the file is possible by means of PUT.

If RECFORM=V was specified, the user must enter the length of the record to be inserted in the record length field before calling the INSRT macro.

Format

Operation	Operands
INSRT	**fcbadr / (1)** **,area / (0)** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed.

(1)
The FCB address is stored in register 1.

area

The address of the record to be inserted in the file. Even in locate mode, the record must be made available at address “area”.

(0)
The address of the record to be inserted in the file is in register 0.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The object code generated can run only in the 16-Mb address space
(24-bit addressing only).

= 31
The object code generated can run in the 2-Gb address space (24-bit or 31-bit addressing).

Programming note

The INSRT macro destroys the contents of registers 0, 1, 14 and 15.

ISREQ - Unlock data block

Macro type: R for PARMOD=24
0 for PARMOD=31

The ISREQ macro is used in shared-update processing to cancel a lock which was requested explicitly in a read operation and is not cancelled (implicitly) by writing the record back to the file or by any other ISAM action macro.

Locks are cancelled implicitly by the next ISAM action macro unless the read operation which requested the lock is followed by an OSTAT macro or by a read operation with NOLOCK for another file.

The lock may be:

- a record lock (for NK-ISAM, non-sequential processing)
- a range lock ((for NK-ISAM, sequential processing)
- a block lock (for K-ISAM).

Format

Operation	Operands
ISREQ	**fcbadr / (1)** **,ACTION = UNLOCK** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB associated with the file containing the lock.

(1)
The FCB address is stored in register “r”.

ACTION = UNLOCK

Cancels the external lock. After execution of “ISREQ ...,UNLOCK”, ISAM returns to the statement following the ISREQ macro.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The object code generated can run only in the 16-Mb address space
(24-bit addressing only).

= 31
The object code generated can run in the 2-Gb address space
(24-bit or 31-bit addressing).

Programming note

The ISREQ macro overwrites the contents of registers 0, 1, 14 and 15.

Return codes

A return code is placed in field ID1ECB of the FCB. Depending upon the type of error, the contents of register 1 may also be affected:

ID1ECB	R1	Meaning
X'0000'	unchanged	The lock has been cancelled
X'0A01'	changed	For the file whose FCB address is stored in register 1, a lock is in force for the job *)
X'0A02'	unchanged	No lock is in force
X'0AA3'	unchanged	Invalid FCB

*) The user can cancel the lock by simply issuing an ISREQ macro with the contents of register 1 unchanged and with register 1 as the first operand.

LBRET - Return from user label routine

Macro type: type R

The LBRET macro is only required when processing standard user labels. Standard user labels are:

- UVL: user volume header label
- UHL: user file header label
- UTL: user trailer label

The EXRTN macro is used for nonstandard labels.

Format

Operation	Operands
LBRET	**fcbadr / (1)** **,0 / 1 / 2 / (0)** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB, same as the address of the FCB macro.

(1)
The address of the FCB is stored in register 1.

The second operand specifies a function code.

0
Label processing is terminated: for output files, the current label will not be written either.

1
Label processing is terminated: for input files, all subsequent labels in the group will be skipped; for output files, the current label will still be written.

2
Label processing is continued: for input files, the UVL and UTL labels are followed by further user labels which are to be read and processed by the program; for output files, the system returns control to the user program after a user label has been written. The user can write a maximum of 9 UVL and 256 UHL and UTL labels in this way. The system terminates label processing when these limits are reached.

(0)
The rightmost byte of register 0 contains the function code.

PARMOD

Specifies the generation mode for the macro.

Default value: the value preset for the generation mode by means of the GPARMOD macro or by the assembler.

= 24
The macro is expanded in accordance with the format for the 24-bit interface. The object code is thus executable only in 24-bit addressing mode.

= 31
The macro is generated as addressing mode-independent.

Programming note

The LBRET macro destroys the contents of registers 0, 1, 14 and 15.

LFFSNAP - List files from a Snapset

Macro type: type S (E form/L form/D form/C form/M form) (see "Macro format")

The LFFSNAP macro enables the user to obtain information about files which were saved on a Snapset when a pubset was backed up. The information relates to whether files can be restored (using the RFFSNAP macro or the RESTORE-FILE-FROM-SNAPSET command). The associated pubset must be imported.

Nonprivileged users can obtain information about all files which they can access (as with the FSTAT macro or the SHOW-FILE-ATTRIBUTES command, which supplies information from the current file catalog).

Information on all existing Snapsets of a pubset can be obtained using the SHOW-SNAPSET-CONFIGURATION command.

The Snapsets are temporarily not available if the SHC-OSD subsystem was not active when the pubset was imported. In this case the command is aborted with return code 0622. As soon as SHC-OSD is active, the Snapsets are subsequently activated when the SHOW-SNAPSET-CONFIGURATION command is called.

Privileged functions

Systems support (TSOS privilege) can obtain information on the files of all user IDs. Wildcards are not permitted in the user ID here.

Format

Operation	Operands
LFFSNAP	**,PATHNAM=<c-string 1..80: filename 1..54 with-wild(80)> / <var: char:80>** **,SNAPSET=<integer -52..-1> / *LATEST** **,SNAPID=<c-string 1..1: name 1..1 with-low> / <var: char 1..1>** **,OUTAREA=(<var: pointer>,<integer 0..32767>)** **,EQUATES=*YES / *NO** **,EXPAND=PARAM / OUTPUT** **MF=L**
	MF=D,PREFIX= D / <pre>
	MF=E,PARAM=<name 1..27>

	MF=C / M
	,PREFIX= D / <pre>
	,MACID= MAL / <macid>

Operand descriptions

PATHNAM

Selects the files which are to be listed.

=<c-string 1..80: filename 1..54 with-wild(80)>
Path name of the file(s) on the Snapset. Wildcards can be used to specify a set of files.

Only files which satisfy the following requirements are listed:

- They must be cataloged when the Snapset is created.
- The pubset on which they are cataloged must be imported locally.
- They may not reside on private disk.

Aliases may be specified. Individual file generations can be specified. When a file generation group is specified, the file generations are also output.

Privileged users (TSOS privilege) can obtain information on the files of all user IDs. Wildcards are not permitted in the user ID here.

=<var: char:80>
Only possible with MF=M:
Symbolic address of a memory area of 80 bytes in which the path name or wildcard string for the required file (s) is stored.

SNAPSET

This operand may not be specified together with the SNAPID operand.
Specifies the Snapset from which the file information is to be output by means of the relative age.

=<integer -52..-1>
Specifies the Snapset explicitly by means of the relative age. The value -1 specifies the latest Snapset (also corresponds to *LATEST).

=*LATEST
The information from the latest Snapset (i.e. from the most recent pubset backup) is output.

SNAPID *This operand may not be specified together with the SNAPSET operand.* Specifies the Snapset from which the file information is to be output.

=<c-string 1..1: name 1..1 with-low>
Specifies the Snapset explicitly by means of the Snapset ID. The maximum of 52 Snapsets for a pubset are distinguished by means of Snapset IDs specified which comprise letters from the 26 lowercase letters a to z and the 26 uppercase letters A to Z.

=<var: char 1..1>
Only possible with MF=M:
Symbolic address of a memory area of 1 byte in which the Snapset ID is stored.

Note

If neither SNAPSET nor SNAPID is specified, the information from the latest Snapset is output.

OUTAREA

Specifies the output area in which the information is to be stored.

=(<var: pointer>,<integer 0..32767>)
Specifies the address and length of the output area.

EQUATES

Control operand; for MF=C and MF=D only:
Specifies whether equates are also to be generated for the values of the fields of the parameter or output area when the parameter or output area is expanded.

= *YES
When the parameter or output area is expanded, equates are also generated for the values of the fields of the parameter or output area.

= *NO
When the parameter or output area is expanded, no equates are generated for the values of the fields of the parameter or output area.

XPAND

Control operand; for MF=C and MF=D only:
Defines which structure is to be expanded (i.e. generated). This operand is ignored for other MF values.

= PARAM
Expands the layout of the parameter list.

= OUTPUT
Expands the layout of the output area.

Return codes

The return code is placed in the standard header of the parameter area. The parameter area may then not be located in the read-only area, otherwise the program terminates.

Standard header: `ccbbaaaa`

The following code relating to execution of the LFFSNAP macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
X'00'	X'00'	X'0000'	No error
X'00'	X'40'	X'0501'	Requested catalog not available
X'00'	X'40'	X'0505'	Error in host communication
X'00'	X'40'	X'0512'	Requested catalog not found
X'00'	X'40'	X'051B'	Requested user ID not on the pubset
X'00'	X'40'	X'051D'	LOGON password different on specified pubset
X'00'	X'20'	X'0531'	Unexpected error during catalog access
X'00'	X'40'	X'0535'	Specified file not accessible
X'00'	X'82'	X'0594'	Not enough virtual memory

X'00'	X'01'	X'05AB'	Address of output area incorrect/not specified
X'02'	X'00'	X'05B6'	Incorrect time conversion in GTIME macro
X'00'	X'20'	X'05C7'	Internal error in DMS
X'00'	X'40'	X'05FC'	Specified user ID not on home pubset
X'00'	X'40'	X'0615'	File resident on a volume set which is not available
X'00'	X'40'	X'0616'	Volume set cannot be accessed on SM pubset

X'00'	X'40'	X'0622'	Snapset not available
X'00'	X'40'	X'0624'	File name invalid
X'00'	X'40'	X'0684'	File does not exist
X'02'	X'00'	X'06CB'	Output information not transferred in full
X'00'	X'01'	X'06CB'	Output area too small
X'00'	X'40'	X'06CC'	No file name matches the wildcard string specified
X'00'	X'01'	X'06F7'	Invalid operand value
X'00'	X'01'	X'06FD'	Parameter area invalid or not accessible

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller
- the list is not aligned on a word boundary
- the list is write-protected.

Layout of the operand list

Macro expansion with MF=D and EXPAND=PARAM, and default values for EQUATES, PREFIX and MACID:

```
         LFFSNAP MF=D,XPAND=PARAM
DMALLFPL DSECT ,
DMALHDR  DS    0A
DMALFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
DMALIFID DS    0A              0   INTERFACE IDENTIFIER
DMALFCTU DS    AL2             0   FUNCTION UNIT NUMBER
DMALFCT  DS    AL1             2   FUNCTION NUMBER
DMALFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
DMALRET  DS    0A              4   GENERAL RETURN CODE
DMALSRET DS    0AL2            4   SUB RETURN CODE
DMALSR2  DS    AL1             4   SUB RETURN CODE 2
DMALSR1  DS    AL1             5   SUB RETURN CODE 1
DMALMRET DS    0AL2            6   MAIN RETURN CODE
DMALMR2  DS    AL1             6   MAIN RETURN CODE 2
DMALMR1  DS    AL1             7   MAIN RETURN CODE 1
DMALFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
*
DMALPNAM DS    CL80                    PATHNAM
DMALSNAP DS    FL1                     SNAPIND
*   SNAPSET - VALUES
DMALSNIN EQU   0                       SNAPSET=<integer>
DMALSNCH EQU   1                       SNAPSET=<char>
DMALSNLT EQU   2                       SNAPSET=*LATEST
*
DMALSNID DS    CL1                     SNAPID
DMALSNVL DS    H                       SNAPVALUE
DMALARAD DS    A                       OUTAREA=(<addr>,...)
DMALARLN DS    F                       OUTAREA=(...,<length>)
DMAL#    EQU   *-DMALHDR
```

Format of the output area

Macro expansion with MF=D and EXPAND=PARAM, and with default values for EQUATES, PREFIX and MACID:

```
         LFFSNAP MF=D,XPAND=OUTPUT
         MFTST MF=D,PREFIX=D,MACID=MAL,ALIGN=F,
               DMACID=MAL,SUPPORT=(E,D,C,M,L),DNAME=MALOUTL
DMALOUTL DSECT ,
*,##### PREFIX=D, MACID=MAL #####
*   Snapset Output
DMALFSIZ DS    F                        FILESIZE
DMALOPNM DS    CL54                     PATHNAME
DMALSTATE DS    FL1                      STATE
*   STATE = VALUES
DMALSTOP EQU   0                        STATE = OPENED
DMALSTCL EQU   1                        STATE = CLOSED
DMALSTNR EQU   2                        STATE = NOREST
*
DMALFTYPE DS    FL1                      FILETYPE
*   FTYPE = VALUES
DMALFTPB EQU   0                        FTYPE = PUBLIC
DMALFTMG EQU   1                        FTYPE = MIGRATED
DMALFTFG EQU   2                        FTYPE = FGG
DMALFTWR EQU   3                        FTYPE = WORK
DMALFTPD EQU   4                        FTYPE = PRDISK
DMALFTTP EQU   5                        FTYPE = TAPE
DMALFTNT EQU   6                        FTYPE = NET
*
*
DMALCRDT DS    0XL16                    Creation Date
DMALCRYE DS    CL4                      YEAR
DMALCRMO DS    CL2                      MONTH
DMALCRDA DS    CL2                      DAY
DMALCRHO DS    CL2                      HOURS
DMALCRMI DS    CL2                      MINUTES
DMALCRSE DS    CL2                      SECONDS
DMALCRUS DS    CL2                      UNUSED
*
*
DMALLCDT DS    0XL16                    Last Change Date
DMALLCYE DS    CL4                      YEAR
DMALLCMO DS    CL2                      MONTH
DMALLCDA DS    CL2
DAY
DMALLCHO DS    CL2                      HOURS
DMALLCMI DS    CL2                      MINUTES
DMALLCSE DS    CL2                      SECONDS
DMALLCUS DS    CL2                      UNUSED
*
DMALENLT DS    FL1                      END Indicator
*
DMALSNXT EQU   0                        FURTHER ENTRY
DMALSNED EQU   1                        LAST ENTRY
DMALSNNS EQU   2                        NOT ENOUGH SPACE
*
DMALUNUS DS    XL3                      UNUSED
```

DMALOUTPUT# EQU *-DMALFSIZ

The following cases are distinguished when the Snapset information is output to the user’s output area:

- All the information could be output
  The output area is overwritten with the required information, the caller receives return code 0. The output area is not deleted to the end, but only written as far as necessary.
- No files match the selection criteria
  The output area is not written at all. The caller receives return code 0684 or 06CC (in the case of wildcards /partial qualification).
- No output was possible
  The output area could not be written (return code 05AB after validation of the output area or address) or it is too small to transfer output information (return code 06CB).
- Complete output was not possible
  Some file information blocks could not be transferred. In addition to the associated display in the output area (NOT ENOUGH SPACE), return code 06CB with subreturn code2 X'02' is output.

Sample calling sequence

```
         LFFSNAP MF=D,XPAND=OUTPUT
              .
              .
         MVC   LFFSMFC(DMAL#),LFFSMFL
         LFFSNAP MF=M,PATHNAM=':X:T.1',PARAM=LFFSMFC,PREFIX=X,         *
               SNAPSET=-1,OUTAREA=(AREAAD,100)
         LFFSNAP MF=E,PARAM=LFFSMFC
              .
              .
LFFSMFC  LFFSNAP MF=C,PREFIX=X,XPAND=PARAM
LFFSMFL  LFFSNAP MF=L,PATHNAM='X'
AREA     DS    CL100
AREAAD   DC    A(AREA)
             .
             .
```

LJFSNAP - List job variables from a Snapset

Macro type: type S (E form/L form/D form/C form/M form) (see "Macro format")

The LJFSNAP macro enables the user to obtain information about job variables which were saved on a Snapset when a pubset was backed up. The information relates to whether job variables can be restored (using the RJFSNAP macro or the RESTORE-JV-FROM-SNAPSET command). The associated pubset must be imported.

Nonprivileged users can obtain information about all job variables which they can access (as with the FSTAT macro or the SHOW-JV-ATTRIBUTES command, which supplies information from the current file catalog).

Information on all existing Snapsets of a pubset can be obtained using the SHOW-SNAPSET-CONFIGURATION command.

The Snapsets are temporarily not available if the SHC-OSD subsystem was not active when the pubset was imported. In this case the command is aborted with return code 0622. As soon as SHC-OSD is active, the Snapsets are subsequently activated when the SHOW-SNAPSET-CONFIGURATION command is called.

Privileged functions

Systems support (TSOS privilege) can obtain information on the job variables of all user IDs. Wildcards are not permitted in the user ID here.

Format

Operation	Operands
LJFSNAP	**,JVNAME=<c-string 1..80: filename 1..54 with-wild(80)> / <var: char:80>** **,SNAPSET=<integer -52..-1> / *LATEST** **,SNAPID=<c-string 1..1: name 1..1 with-low> / <var: char 1..1>** **,OUTAREA=(<var: pointer>,<integer 0..32767>)** **,EQUATES=*YES / *NO** **,EXPAND=PARAM / OUTPUT** **MF=L**
	MF=D,PREFIX= D / <pre>
	MF=E,PARAM=<name 1..27>

	MF=C / M
	,PREFIX= D / <pre>
	,MACID= MAJ / <macid>

Operand descriptions

JVNAME

Selects the job variables which are to be listed.

=<c-string 1..80: filename 1..54 with-wild(80)>
Path name of the job variables on the Snapset. Wildcards can be used to specify a set of job variables.

The job variables must satisfy the following requirements:

- They must be cataloged when the Snapset is created.
- The pubset on which they are cataloged must be imported locally.

Aliases may be specified.

Privileged users (TSOS privilege) can obtain information on the job variables of all user IDs. Wildcards are not permitted in the user ID here.

=<var: char:80>
Only possible with MF=M:
Symbolic address of a memory area of 80 bytes in which the path name or wildcard string for the required job variable(s) is stored.

SNAPSET

This operand may not be specified together with the SNAPID operand.
Specifies the Snapset from which the job variable information is to be output by means of the relative age.

=<integer -52..-1>
Specifies the Snapset explicitly by means of the relative age. The value -1 specifies the latest Snapset (also corresponds to *LATEST).

=*LATEST
The information from the latest Snapset (i.e. from the most recent pubset backup) is output.

SNAPID
This operand may not be specified together with the SNAPSET operand.
Specifies the Snapset from which the file information is to be output by means of the Snapset ID.

=<c-string 1..1: name 1..1 with-low>
Specifies the Snapset explicitly by means of the Snapset ID. The maximum of 52 Snapsets for a pubsets are distinguished by means of Snapset IDs specified which comprise letters from the 26 lowercase letters a to z and the 26 uppercase letters A to Z.

=<var: char 1..1>
Only possible with MF=M:
Symbolic address of a memory area of 1 byte in which the Snapset ID is stored.

Note

If neither SNAPSET nor SNAPID is specified, the information from the latest Snapset is output.

OUTAREA
Specifies the output area in which the information is to be stored.

=(<var: pointer>,<integer 0..32767>)
Specifies the address and length of the output area.

EQUATES

Control operand; for MF=C and MF=D only:
Specifies whether equates are also to be generated for the values of the fields of the parameter or output area when the parameter or output area is expanded.

= *YES
When the parameter or output area is expanded, equates are also generated for the values of the fields of the parameter or output area.

= *NO
When the parameter or output area is expanded, no equates are generated for the values of the fields of the parameter or output area.

XPAND

Control operand; for MF=C and MF=D only:
Defines which structure is to be expanded (i.e. generated). This operand is ignored for other MF values.

= PARAM
Expands the layout of the parameter list.

= OUTPUT
Expands the layout of the output area.

Return codes

The return code is placed in the standard header of the parameter area. The parameter area may then not be located in the read-only area, otherwise the program terminates.

Standard header: `ccbbaaaa`

The following code relating to execution of the LJFSNAP macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
X'00'	X'00'	X'0000'	No error
X'00'	X'40'	X'0501'	Requested catalog not available
X'00'	X'40'	X'0505'	Error in host communication
X'00'	X'40'	X'0512'	Requested catalog not found
X'00'	X'40'	X'051B'	Requested user ID not on the pubset
X'00'	X'40'	X'051D'	LOGON password different on specified pubset
X'00'	X'20'	X'0531'	Unexpected error during catalog access
X'00'	X'82'	X'0594'	Not enough virtual memory
X'00'	X'01'	X'05AB'	Address of output area incorrect/not specified

X'02'	X'00'	X'05B6'	Incorrect time conversion in GTIME macro
X'00'	X'20'	X'05C7'	Internal error in DMS
X'00'	X'40'	X'05FC'	Specified user ID not on home pubset
X'00'	X'40'	X'0622'	Snapset not available
X'00'	X'40'	X'0624'	JV name invalid
X'00'	X'40'	X'0682'	JV error when accessing JV

X'02'	X'00'	X'06CB'	Output information not transferred in full
X'00'	X'01'	X'06CB'	Output area too small
X'00'	X'01'	X'06F7'	Invalid operand value
X'00'	X'01'	X'06FD'	Parameter area invalid or not accessible

The return codes with the maincode X’04xy’ belong to the component JVS. A list which

includes the meanings can be output using the JVSERROR macro (see also the “Job Variables” manual [21]).

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller
- the list is not aligned on a word boundary
- the list is write-protected.

Layout of the operand list

Macro expansion with MF=D and EXPAND=PARAM, and default values for EQUATES, PREFIX and MACID:

```
         LJFSNAP MF=D,XPAND=PARAM
DMAJLFPL DSECT ,
DMAJHDR  DS    0A
DMAJFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
DMAJIFID DS    0A              0   INTERFACE IDENTIFIER
DMAJFCTU DS    AL2             0   FUNCTION UNIT NUMBER
DMAJFCT  DS    AL1             2   FUNCTION NUMBER
DMAJFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
DMAJRET  DS    0A              4   GENERAL RETURN CODE
DMAJSRET DS    0AL2            4   SUB RETURN CODE
DMAJSR2  DS    AL1             4   SUB RETURN CODE 2
DMAJSR1  DS    AL1             5   SUB RETURN CODE 1
DMAJMRET DS    0AL2            6   MAIN RETURN CODE
DMAJMR2  DS    AL1             6   MAIN RETURN CODE 2
DMAJMR1  DS    AL1             7   MAIN RETURN CODE 1
DMAJFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
*
DMAJJVNM DS    CL80                    JVname
DMAJSNAP DS    FL1                     Snapind
*   SNAPSET - VALUES
DMAJSNIN EQU   0                       SNAPSET=<integer>
DMAJSNCH EQU   1                       SNAPSET=<char>
DMAJSNLT EQU   2                       SNAPSET=*LATEST
*
DMAJSNID DS    CL1                     Snapid
DMAJSNVL DS    H                       SnapValue
DMAJARAD DS    A                       Outarea=(<addr>,...)
DMAJARLN DS    F                       Outarea=(...,<length>)
DMAJ#    EQU   *-DMAJHDR
```

Format of the output area

Macro expansion with MF=D and EXPAND=PARAM, and with default values for EQUATES, PREFIX and MACID:

```
         LJFSNAP MF=D,XPAND=OUTPUT
DMAJOUTL DSECT ,
*   Output List
DMAJJSIZ DS    F                       JVSIZE
DMAJOJVN DS    CL54                    JVNAME
DMAJUNU1 DS    XL2                     UNUSED
*
DMAJCRDT DS    0XL16                   Creation Date
DMAJCRYE DS    CL4                     YEAR
DMAJCRMO DS    CL2                     MONTH
DMAJCRDA DS    CL2                     DAY
DMAJCRHO DS    CL2                     HOURS
DMAJCRMI DS    CL2                     MINUTES
DMAJCRSE DS    CL2                     SECONDS
DMAJCRUS DS    CL2                     UNUSED
*
*
DMAJEXDT DS    0XL16                   Expiration Date
DMAJEXYE DS    CL4                     YEAR
DMAJEXMO DS    CL2                     MONTH
DMAJEXDA DS    CL2                     DAY
DMAJEXHO DS    CL2                     HOURS
DMAJEXMI DS    CL2                     MINUTES
DMAJEXSE DS    CL2                     SECONDS
DMAJEXUS DS    CL2                     UNUSED
*
DMAJENLT DS    FL1                     END Indicator
*
DMAJSNXT EQU   0                       FURTHER ENTRY
DMAJSNED EQU   1                       LAST ENTRY
DMAJSNNS EQU   2                       NOT ENOUGH SPACE
*
DMAJUNU2 DS    XL3                     UNUSED
DMAJOUTPUT# EQU  *-DMAJJSIZ
```

The following cases are distinguished when the Snapset information is output to the user’s output area:

- All the information could be output
  The output area is overwritten with the required information, the caller receives return code 0. The output area is not deleted to the end, but only written as far as necessary.
- No files match the selection criteria
  The output area is not written at all. The caller receives return code 0684 or 06CC (in the case of wildcards /partial qualification).
- No output was possible
  The output area could not be written (return code 05AB after validation of the output area or address) or it is too small to transfer output information (return code 06CB).

- Complete output was not possible
  Some file information blocks could not be transferred. In addition to the associated display in the output area (NOT ENOUGH SPACE), return code 06CB with subreturn code2 X'02' is output.

Sample calling sequence

```
      LJFSNAP MF=D,XPAND=OUTPUT
            .
            .
      MVC   LJFSMFC(DMAL#),LJFSMFL
      LJFSNAP MF=M,PREFIX=X,JVNAME=':X:JV.1',OUTAREA=(AREAAD,100),  *
            PARAM=LJFSMFC
      LJFSNAP MF=E,PARAM=LJFSMFC
            .
            .
LJFSMFC  LJFSNAP MF=C,PREFIX=X,XPAND=PARAM
LJFSMFL  LJFSNAP MF=L,PATHNAM='X'
AREA     DS    CL100
AREAAD  DC    A(AREA)
```

MAILFIL - Send file by email

Macro type: type S (E form/L form/D form/C form/M form) (see "Macro format")

Like the MAIL-FILE command, the MAILFIL macro sends a file as an attachment to an email. A user ID is specified as the receiver of the email. The sender is the user ID of the calling task. MAIL-FILE takes over the email address entered in the EMAIL-ADDRESS field of these user entries as the sender. How you ascertain the receiver and sender addresses, in particular when an address list is used, is described in the section "Selecting email addresses by means of the job name".

A PLAM library member, a SAM or ISAM file and the contents of the system file SYSLST or SYSOUT can be sent. A PAM file can be sent only if the content is available in PDF format. The user task under which MAILFIL is executed must have the required access rights. The file attribute CCS name is evaluated in the case of automatic character set conversion. Optionally the caller can specify that the file is to be deleted automatically after it has been sent.

To execute the macro the “Mail-Sender” function of the software product interNet Services must be available, and at least one email address must be entered in the user entry of the TSOS system ID.

The call is rejected if no email address is entered in the receiver’s user entry. If no email address is entered for the caller, the address of the receiver or TSOS is entered instead as the sender.

If the email cannot be delivered (e.g. because the address is invalid), a bounce mail is sent to the email address of TSOS to request systems support to check the incorrect address. If more than one email address is entered for TSOS, the first address is used for the bounce mail.

The MAIL-FILE functionality is also used by other components of BS2000 to send log files:

- At job termination
  In the EXIT-JOB (or LOGOFF), CANCEL-JOB and ENTER-PROCEDURE commands, transfer to SYSLST or SYSOUT at job termination can be requested instead of a printout. The default value *STDOUT directs output to the output medium defined in the system parameter SSMOUT (printer or email).
- In the case of outputs from utility routines
  Currently HSMS V9.0 and higher and MAREN V12.0 and higher support the transfer of output information and logs.

Format

Operation	Operands

MAILFIL	**,PATHNAM=<c-string 1..54: filename 1..54> / <var: char:54> /** ***SYSOUT / *SYSLST / (*SYSLST,<integer 1..99>)** **,LIBELEM=*NONE /** **(<c-string 1..64: composed-name 1..64> with under /** **<var: char:64>** **,*HIGHEST / *UPPER /** **<c-string 1..24: composed-name 1..24> with under /** **<var: char:24>** **,<c-string 1..8: alphanum-name 1..8> / <var: char:8>)** **,USERID=*OWN / <c-string 1..8: name 1..8> / <var: char:8>** **,SUBJECT=*STD / <c-string 1..256 with low> / <var: char:256>** **,DELETE=*NO / *YES / *DESTROY** **,EQUATES=*YES / *NO** **,VERSION=1 / <integer 1..2>** **MF=L**
	MF=D,PREFIX= D / <pre>
	MF=E,PARAM=<name 1..27>

	MF=C / M
	,PREFIX= D / <pre>
	,MACID= MAM / <macid>

Operand descriptions

PATHNAM

Selects the file which is to be sent or the PLAM library of the library member which is to be sent.

=<c-string 1..54: filename 1..54>
Path name of the file or library name.
The following restrictions apply fo files which are to be sent:

- The file is a SAM or ISAM file. A PAM file is sent only if the content is in PDF format.
- The file may not be empty.
- The name may specify a single file generation but not a file generation group.
- It can also be a temporary file.
  A privileged user (TSOS privilege) can also specify temporary files of another task.
  A privileged user can also specify temporary files which reside on a different pubset from the default pubset of the user ID.
- The file may not be only accessible via an RFA connection.

=<var: char:54>
Only possible with MF=M:
Symbolic address of a memory area of 54 bytes in which the path name of the file to be sent is stored.

=*SYSOUT
Specifies the system file SYSOUT. This specification is possible only if the SYSOUT file is assigned a file or file generation on disk which was created using the access method SAM.
The specification is rejected in the following cases:

- The assigned file is still empty.
- The SYSOUT file has the primary allocation.
- The DUMMY file, a temporary file, a PLAM library member or an S variable is assigned.

=*SYSLST
Specifies the system file SYSLST. This specification is rejected in the following cases:

- SYSLST is empty.
- The DUMMY file, a temporary file, a PLAM library member or an S variable is assigned.
- The assigned file or file generation is not resident on disk or was not created using the access method SAM.

=(*SYSLST,<integer 1..99>)
Specifies a SYSLST file from the set SYSLST01 through SYSLST99. This specification is possible only if the SYSLST file is assigned a file or file generation on disk which was created using the access method SAM.
The specification is rejected in the following cases:

- The assigned file is still empty.
- The SYSLST file has the primary allocation.
- The DUMMY file, a temporary file, a PLAM library member or an S variable is assigned.

LIBELEM

PLAM library member which is to be sent. The name of the PLAM library must also be specified in the PATHNAM operand.
Only text members and PDF files can be sent. Text members are members of the types S, M, J, P, D, X and types derived from these provided they contain no block-oriented records. A member which contains block-oriented records is sent only if its content is in PDF format.

=*NONE
No library member is to be sent. The PATHNAM operand contains the information about the file which is to be sent.

=(<element>,<version>,<typ>)
Specifies the name, version and type of a member which is to be sent which belongs to the PLAM library specified in the PATHNAM operand.

<element>=<c-string 1..64: composed-name 1..64> with under
Name of the library member which is to be sent.

<element>=<var: char:64>
Only possible with MF=M:
Symbolic address of a memory area of 64 bytes in which the name of the library member which is to be sent is stored.

<version>=*HIGHEST
Selects the highest existing version of all members which have the specified name and the specified type.

<version>=*UPPER
Selects the highest possible version (X'FF') of all members which have the specified name and the specified type.

<version>=<c-string 1..24: composed-name 1..24> with under
Version of the library member which is to be sent.

<version>=<var: char:24>
Only possible with MF=M:
Symbolic address of a memory area of 24 bytes in which the version of the library member which is to be sent is stored.

<typ>=<c-string 1..8: alphanum-name 1..8>
Type of the library member which is to be sent.

<version>=<var: char:8>
Only possible with MF=M:
Symbolic address of a memory area of 8 bytes in which the type of the library member which is to be sent is stored.

USERID

User ID whose entry in the user catalog contains the receiver’s email address.

=*OWN
The default value is *OWN, i.e. the logon user ID of the calling task. If the user entry contains a list with more than one email address, a receiver address may be selected in accordance with the job name (see "Selecting email addresses by means of the job name").

=<c-string 1..8: name 1..8>
User ID from whose user entry the receiver’s email address is ascertained.

=<var: char:8>
Only possible with MF=M:
Symbolic address of a memory area of 8 bytes in which the receiver’s user ID is stored.

SUBJECT
Specifies the subject of the email.

Note

As the BCAM name of the sending BS2000 system is already contained in the text of the email which is sent, it need not be entered specially in the subject line.

=*STD
*STD specifies that the email should have a standardized subject text which, in addition to the information “from BS2000”, also contains the sender ID and the file name.

=<c-string 1..256 with-low>
Subject of the email. Case-sensitive. You are recommended to stick to the international character set.

=<var: char:8>
Only possible with MF=M:
Symbolic address of a memory area of 256 bytes in which the subject of the email is stored.

DELETE

Specifies whether the file or the PLAM library member should be automatically deleted after it has been sent successfully:

- If the system file SYSLST is to be sent and SYSLST has the primary allocation, DELETE=*YES applies.
- If the system file SYSLST or SYSOUT is to be sent and the system file is assigned to a file or file generation, it is not deleted automatically.

=*NO
The file or the PLAM library member is not deleted. The file or the PLAM library member is available again immediately after MAIL-FILE is called.

=*YES
The file is automatically deleted after it has been sent successfully. A file is regarded as having been sent successfully even if it cannot be delivered (e.g. because the email address is unknown).

=*DESTROY
This specification has the same effect as DELETE=*YES. In addition, the file or member content is overwritten with binary zeros when it is deleted.

EQUATES

Control operand; for MF=C and MF=D only:
Specifies whether equates are also to be generated for the values of the fields of the parameter area when the parameter area is expanded.

= *YES
When the parameter area is expanded, equates are also generated for the values of the fields of the parameter area.

= *NO
When the parameter area is expanded, no equates are generated for the values of the fields of the parameter area.

VERSION

Controls generation of the parameter area or of the function call. When the function call is generated, the operand must have the same value as when generating the associated parameter area.

= 1
Generates the parameter area or function call applicable for BS2000/OSD V8.0 (it is not possible to specify a library member).

= 2
Generates the parameter area or function call applicable for BS2000/OSD V9.0 and higher.

Layout of the parameter area

The parameter area must be aligned on a word boundary. It begins with a standard header which MAILFIL initializes as follows:

Function Unit Number	22

Function Number	32
Interface Version Number	2 if VERSION=2 is specified, 1 otherwise
Return Code	-1

Macro expansion with MF=D and default values for EQUATES, PREFIX and MACID:

```
         MAILFIL MF=D,VERSION=2
DMAMDEPL DSECT ,
DMAMHDR  DS    0A
DMAMFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
DMAMIFID DS    0A              0   INTERFACE IDENTIFIER
DMAMFCTU DS    AL2             0   FUNCTION UNIT NUMBER
DMAMFCT  DS    AL1             2   FUNCTION NUMBER
DMAMFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
DMAMRET  DS    0A              4   GENERAL RETURN CODE
DMAMSRET DS    0AL2            4   SUB RETURN CODE
DMAMSR2  DS    AL1             4   SUB RETURN CODE 2
DMAMSR1  DS    AL1             5   SUB RETURN CODE 1
DMAMMRET DS    0AL2            6   MAIN RETURN CODE
DMAMMR2  DS    AL1             6   MAIN RETURN CODE 2
DMAMMR1  DS    AL1             7   MAIN RETURN CODE 1
DMAMFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
*
DMAMPNAM DS    CL54                    Dateiname
DMAMUSID DS    CL8                     Benutzerkennung oder Blank
DMAMSUBJ DS    CL256                   Betreff oder Blank
DMAMDELE DS    FL1                     automatisches Loeschen
*   DELETE - values
DMAMDELN EQU   0                       DELETE = NO
DMAMDELY EQU   1                       DELETE = YES
DMAMDELD EQU   2                       DELETE = DESTROY
*
DMAMPNSP DS    FL1                     Typ der PATHNAM-Angabe
*   PATHNAM - values
DMAMBYFN EQU   0                       PATHNAM = fnam
DMAMSLST EQU   1                       PATHNAM = *SYSLST
DMAMBYSN EQU   2                       (*SYSLST,n)
DMAMSOUT EQU   3                       PATHNAM = *SYSOUT
*
DMAMSYSNUM DS    X                     Syslst number
DMAMRES  DS    XL3                     Alignment
DMAMENAM DS    CL64                    Elementname oder Blank
DMAMEVER DS    CL24                    Elementversion oder Blank
DMAMETYP DS    CL8                     Elementtyp oder Blank
DMAMRES2 DS    XL4                     Alignment
DMAM#    EQU   *-DMAMHDR
```

Return code

The return code is placed in the standard header of the parameter area. The parameter area may then not be located in the read-only area, otherwise the program terminates.

The following return codes are generated by MAIL-FILE:

X'cc'	X'bb'	X'aaaa'	Meaning
X'00'	X'00'	X'0000'	No error
X'00'	X'01'	X'0554'	Format of file name invalid
X'00'	X'01'	X'0576'	Incorrect operand combination or UNUSED fields not deleted
X'00'	X'20'	X'05C7'	Internal error in DMS
X'00'	X'40'	X'05FC'	User ID not entered
X'00'	X'40'	X'0694'	Not permissible to send file
X'00'	X'40'	X'0695'	Email address missing
X'00'	X'40'	X'0696'	Email address of TSOS missing

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

Furthermore, return codes of DMS interfaces (see the FSTAT and ERASE macros) and of the Mail-Sender interfaces (YMLSML macro of the software product interNet Services) can be transferred.

When a PLAM library member is specified (LIBELEM operand), return codes from ILAM interfaces (see the PMATCH, PMDTCH, PMOPEN, PMCLOS, PMGETA, PMPOSA, PMDELM macros of the ILAM interface for PLAM) can be transferred. The maincodes of the ILAM interfaces correspond to the decimal message numbers of the PLAM messages (i.e., for example, maincode 00CB corresponds to PLAM message PLA0203).
You can obtain detailed information using `/HELP-MSG <msgid>`.

Sample calling sequence

```
         MVC   MLFLMFC(256),MLFLMFL
         MVC   MLFLMFC+256(XMAM#-256),MLFLMFL+256
         MAILFIL MF=M,PREFIX=X,DELETE=*DESTROY
         MAILFIL MF=E,PARAM=MLFLMFC
             .
             .
MLFLMFC  MAILFIL MF=C,PREFIX=X
MLFLMFL  MAILFIL MF=L,PATHNAM='PN0',USERID='UI0'
```

Selecting email addresses by means of the job name

MAIL-FILE ascertains the email addresses of the receiver and the sender by means of the user entry of each of the user IDs concerned. To execute the command the user IDs of the receiver and of the sender must each contain an email address (see the SHOW-USER-ATTRIBUTES command, EMAIL-ADDRESS output field). The entry can also contain an address list, i.e. multiple email addresses separated by a comma.

When the sender’s user entry contains an address list, the first address is used as the sender address.

When the receiver's user entry contains an address list, MAIL-FILE makes a distinction between whether the caller's user ID (*OWN) or a "foreign" user ID was specified as the receiver. If a foreign user ID is specified, MAIL-FILE sends the email to all addresses. If the home user ID is specified, MAIL-FILE selects the addresses by means of the job name of the calling task:
An address is searched for in which a partial name of the local address part (ahead of the @) begins with the job name (not case-sensitive). Partial names are separated from one another by a period (e.g. first-name.last-name).

For example, the following addresses are selected from the address list `Anna.Huber@xy,Anja.Bauer@xy,Anton.Baumann@xy:`

- `Anna.Huber@xy` with the job names: ANN, HU, HUBER
- `Anja.Bauer@xy` with the job names: ANJ, ANJA, BAUE, BAUER
- `Anton.Baumann@xy` with the job names: ANT, BAUM, BAUMAN

Optionally you can also prefix the addresses in the user entry with "address names" in parentheses.
Example: `(ANH)Anna.Huber@xy,(ANB)Anja.Bauer@xy,(BMN)Anton.Baumann@xy`

The following addresses, for example, are then selected from this address list:

- `Anna.Huber@xy` with the job names: ANH and ANN, HU, HUBER
- `Anja.Bauer@xy` with the job names: ANB and ANJ, ANJA, BAUE, BAUER
- `Anton.Baumann@xy` with the job names: BMN and ANT, BAUM, BAUMAN

If the job name matches more than one address, the address is selected whose partial name which matches the job name is the shortest. From the address list`Beate.Pauli@xy,Pauline.Beck@xy,Paul.Becker@xy`, for example, the following addresses are selected:

- `Beate.Pauli@xy` with the job names: PAULI, BEA
- `Pauline.Beck@xy` with the job names: PAULIN, BE, BECK
- `Paul.Becker@xy` with the job names: P, PAUL, BECKER

If the partial name which matches the job name is equally short in more than one address, the first of these addresses is selected.

If more than one partial name in an address matches the job name, only the first partial name is taken into account.

If the calling task does not have a job name or the job name does not match any address in the address list, the following procedure applies:

- When the receiver address is ascertained, the entire address list is used, i.e. the email is sent to all addresses.
- When the sender address is ascertained, only the first address in the address list is used.

NDWERINF - Evaluate status bytes

The status bytes (sense bytes etc.) in the FCB are set in the case of an error. The NDWERINF macro generates equate statements for the logical error information, thus enabling the user to evaluate these status bytes.

Note

The NDWERINF macro is supported solely for reasons of compatibility and should therefore not be used in new applications. The error information returned by this macro is contained in the logical return code of the FCB and can be evaluated there.

Format

Operation	Operands
NDWERINF	**[prefix / *]** **[,ONLYOSB = YES]**

Operand descriptions

prefix

Prefix (1 character) with which the generated names are to begin.

Default value: I

*****

No prefix is generated.

ONLYOSB=YES

Equates are generated only for the three sense bytes (OSB).

OPEN - Open file

Macro type: type R

Every file has to be opened with the OPEN macro before it can be processed. The default value for the OPEN macro is defined in the OPEN operand of the FCB or FILE macro (via the TFT); if no OPEN mode is specified in these macros, the default value is OPEN=INPUT.

Format

Operation	Operands
OPEN	**fcbadr / (1)** **,mode / (0)** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB for the file to be opened.

(1)
The address of the FCB is stored in register 1.

mode

OPEN processing mode (see first column of table below).

(0)
The OPEN mode is stored in coded form in the least significant byte of register 0. The following table shows the various OPEN modes and the associated codes.

OPEN mode	Code	Meaning
not specified	X'00'	The OPEN mode specified in the TU FCB has priority. If this is also X'00' , an error message is displayed.
INPUT	X'01'	The file is opened as an input file and only read operations are permitted; the file is read “forwards” (from beginning of file -> end of file).
REVERSE	X'02'	The file is opened as an input file and only read operations are permitted; the file is read “backwards” (from end of file -> beginning of file).
OUTPUT	X'04'	The file is opened as an output file and, if it already exists, overwritten from the beginning; only sequential write operations are permitted.

EXTEND	X'08'	The file is opened as an output file for sequential extension; starting at a defined point in the file, only sequential write operations are permitted. In the case of an empty file, OPEN EXTEND is mapped onto OPEN OUTPUT.
UPDATE	X'10'	The file is to be updated; both read and write operations are permitted.

INOUT	X'20'	The file is to be updated; both read and write operations are permitted.
OUTIN	X'40'	The file is first to be created (or overwritten); after this, both read and write operations are permitted.

PARMOD

Specifies the generation mode for the macro.

Default value: the value preset for the generation mode by means of the GPARMOD macro or by the assembler.

= 24
The macro is expanded in accordance with the format for the 24-bit interface. The object code is thus executable only in 24-bit addressing mode.

= 31
The macro is generated as addressing mode-independent.

Programming notes

1. The OPEN macro overwrites the registers 0, 1, 14 and 15.
2. If, during program execution, an FCB address is specified for a file in an OPEN macro, and that address has been used in a previous OPEN, the program is terminated abnormally and error code DMS0D9F returned.

Notes on programming and on large files

The OPEN macro and the access methods are only affected by the introduction of large files, not by the introduction of large volumes.

The OPEN interface checks whether files are permitted to extend beyond 32 GB and whether the creation of files >= 32 GB and access to such files are permitted.

There are two aspects associated with this:

1. Rejection of access to or the creation of large files for access methods that do not allow processing of large files.
2. Indication that a program can create or open files >= 32 GB.

Incompatible interface variants

Interfaces where 3-byte block numbers are used are never able to work with files >= 32 GB. This affects the following cases:

- All files in key format (BLKCTRL=PAMKEY):
  The logical block numbers in the PAMKEY are only 3 bytes wide.
- 24-bit interface of UPAM
  The field for logical block numbers in the UPAM parameter lists and in the TU FCB is only 3 bytes wide.
- 24-bit interface of SAM
  The logical block numbers are affected as part of the retrieval address.

In all the cases described above, the following applies:

- Access to files >= 32 GB is rejected with the return code `X'0000D9D'` or `X'00000D00'`, depending on the size of the storage space allocated to the file (FILE_SIZE).
- Exceeding a file size of 32 GB as a result of secondary allocation is prohibited (LARGE_FILE).

Semantic incompatibilities

It is possible that applications use interfaces that employ 4-byte fields for the data fields described above, but that these 4-byte fields are implicitly or explicitly subject to the former limitation to values less than X'00FFFFFF'. Please notice, that for the modifications in Assembler code for files >= 32 GB in addition to the conversion to 4-byte block numbers and counters, it is necessary to check whether the program logic implicitly assumes that files may not be larger than 32 GB.

The following lists a number of examples of potential problems.

- The highest 3-byte block number X'FFFFFF' has a special meaning.
- “Block numbers” greater than X'00FFFFFF' represent objects other than blocks.
- For calculations with block numbers or counters greater than X'00FFFFFF', overflow can occur.
- The number of digits in input or output fields is not sufficient for displaying such large block numbers or counters.
- When converting hexadecimal numbers to decimal numbers, the field length is too small for the decimal number.
- It is assumed that data structures whose size depends on a file size always find space in virtual memory. This assumption can apply to files less than 32 GB, but not if this size is exceeded.

It is not possible to give a complete list of problem cases. However, one possible source of problems could be coding based on block numbers, file sizes and derived or dependent structures.

The following return codes display information about execution of the macro with regard to large files:

X'cc'	**X'bb'**	**X'aaaa'**	**Explanation**
X'00'	X'00'	X'0D9D'	Error when opening a disk file
X'00'	X'00'	X'0D00'	System error when opening a file

OSTAT - Request information on open files

Macro type: R for PARMOD=24
O for PARMOD=31

The OSTAT macro enables the user to determine:

- how many jobs have opened the file
- how many jobs have opened the file for ISAM shared-update processing (in all possible OPEN/SHARUPD combinations)
- how many jobs have opened the file with an access method other than ISAM

A Dsect can be generated for the output area (operand “area”) by means of the IDOST macro.

The file must be opened before the OSTAT macro is issued. The OPEN/SHARUPD combination of the calling job is then included in the output.

Format

Operation	Operands
OSTAT	**fcbadr / (1)** **,area / (0)** **[,PARMOD = 24** **/ 31]**

Operand descriptions

fcbaddr

Address of the FCB for the file for which the OPEN status is to be returned.

(1)
The FCB address is stored in register 1.

area

Address of the area in which the information is to be placed. This area consists of 23 count fields, each 1 byte long. The maximum value of each counter is therefore 255. If any of the ISAM internal counters reaches a higher value, then the corresponding counter in the information area is set to 255.

A Dsect can be generated for the output area by means of the IDOST macro.

If the “area” operand is not specified, the area for the information is generated in the macro; this means that the resulting code will not be reentrant.

(0)
The address of the area in which the information is to be placed is stored in register 0.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The object code generated can run only in the 16-Mb address space (24-bit addressing only).

= 31
The object code generated can run in the 2-Gb address space (24-bit or 31-bit addressing).

Programming note

The OSTAT macro overwrites the contents of registers 0, 1, 14 and 15.

Return codes

Register 1	Address of the FCB.
Register 0	Address of the output area “area”.
Register 15	X'0000' (execution successful) or an error code in the two low-order bytes.

PAM - Perform UPAM actions

Macro type: type S

All user requests to DMS for UPAM actions are issued using this macro.

Format

Operation	Operands
PAM	**fcbadr** **[,PARMOD = 24 / 31]** **[,RDWT / CHK / LOCK / LRD / LRDWT / RD / RDEQU / SETL /** **SETLPP / SYNC / UNLOCK / WRT / WRTWT / WRTWU / WT]** **[,CHAIN = relexp][,FECB = relexp]** **[,HP = number / +number / -number]** **,KEYFLD = relexp]** **[,LEN = STD / (STD,n) / length]** **[,LOC = 1 / 2 / relexp]** **[,MKEY = NO / YES]** **[,REQNO = number]** **[,POST = postcode]**

Operand descriptions

fcbaddr

Specifies the address of the FCB associated with the file.

CHAIN = relexp

This operand specifies the symbolic address of the next element in a chain of PAM macros in list form. Such a chain must not contain more than 255 elements.

Default value: the list is regarded as the last element in a chain.

CHK

Checks whether the specified I/O request has been completed. If so, the program resumes; if not, control is transferred to the address specified by the LOC operand in the user program.

FECB = relexp

Symbolic address of a file event control block (see "TU eventing: event-driven processing"). This operand may be specified only for the operations RD, LRD and WRT.

HP

Points to a specific PAM page, as follows:

- in non-chained I/O, the PAM page to be transferred (or locked)

- in chained I/O (for disk files or for tape files with nonstandard blocks (NK files)), the first in a series of PAM pages to be transferred (or locked).

Default value: HP = +1 (i.e. sequential processing).

Note

For PAM and SAM files without PAM keys (operand BLKCTRL=DATA or BLKCTRL=NO), the following restrictions result from the fact that only logical blocks (and not the individual 2K blocks) can be accessed in such files:

- For the operations LRD, LRDWT, RD, RDWT, RDEQU, WRT, WRTWT, WRTWU, LOCK and UNLOCK, HP must point to the beginning of a logical block,
  i.e. HP = n * BLKSIZE + 1 (n >= 0).
  If this is not the case, the PAM call will be rejected.
- For the operations SETL and SETLPP, HP must point to the end of a logical block, i.e. HP = n * BLKSIZE + 1 (n >= 0).
  If this is not the case, the PAM call will be rejected. (For SETL operations, the file pointer is thus ready to access the **next** logical block.)

More detailed information is given in the "Programming notes".

= number
The absolute logical block number of the PAM page (LHP) within the file.

= +number
= -number
A relative logical block number (= relative to the file pointer). The corresponding absolute logical block number is calculated by adding the file pointer and the specified relative block number.

KEYFLD = relexp

This is evaluated only for files with PAM keys (operand BLKCTRL=PAMKEY). It is ignored for files with BLKCTRL=DATA or BLKCTRL=NO.
In the case of non-chained I/O, this operand specifies the address of a 16-byte area into which the PAM key is placed during reading or from which it is taken during writing.

In the case of chained I/O with separate processing of all associated PAM keys (operand MKEY=YES), this operand must specify the address of a sufficiently large area (number of PAM pages to be transferred times 16 bytes). If just the PAM key of the first PAM page is to be processed (MKEY=NO), the specified area need only be 16 bytes long. In this case the KEYFLD operand can be omitted, and the default value will come into effect.

The KEYFLD operand is ignored for the operations WT, CHK, SETL and SYNC.

Default value (for files with BLKCTRL=PAMKEY):

- address of the ID1KEY1 field in the FCB (for all read and write operations except RDWT, LRDWT and RDEQU);
- address of the ID1KEY2 field in the FCB (for RDWT,LRDWT and RDEQU).

For further information, see the "Programming notes".

LEN

Specifies the length of the data transferred in a PAM call.

Default value: LEN=STD

The LEN operand is ignored for the operations WT, CHK and SETL.

For its relationship with BLKSIZE, see the description of the UPAM FCB macro ("UPAM - User Primary Access Method").

For further information, see the "Programming notes".

= STD
Data is transferred with the length of a standard block (corresponds to 2048 bytes).

= (STD,n)
Data is transferred with the length of n standard blocks (of 2048 bytes each). n is an integer in the range 1 <= n <= 16.This operand value may be specified only for the 31-bit interface of the macro.

= length
The length in bytes of the data to be transferred (1 <= length <= 32768). A distinction must be made between the following cases:

- *1* <= *length* <= *2048*: no chained processing; each PAM macro transfers a block with a length of 2048 bytes from/to the buffer, i.e. reads or writes one block.
- *2049* <= *length* <= *32768*: chained processing.

For disk and tape files with PAM keys, the number of PAM blocks transferred with one PAM call is determined as follows:

- If *length* is an integral multiple of 2048 (length = n * 2048, n <= 16), the quotient n = length/2048 is the number of PAM blocks to be transferred.
- If *length* is not an integral multiple of 2048, the quotient is rounded up to the next higher integer. With certain hardware configurations, this can result in intermediate buffering and a corresponding decline in performance.

During CLOSE processing, in the case of files with BLKCTRL=PAMKEY the position of the last valid byte is always stored in the last-byte pointer of the last PAM block, and in the case of files with BLKCTRL=NO or DATA in the last-byte pointer of the last logical block of the file.

In the case of node files only data equivalent to the length *length* is transferred at the end of the file. A node file thus ends on a byte boundary in the event of subsequent CLOSE processing.

For a write operation, data is transferred with this length *length*; for a read operation, only the specified number of bytes are valid.

For PAM or SAM disk files **without PAM keys**, the length of a logical block is taken into account when the file pointer is calculated. However, the number of 2048-byte blocks written is never more than necessary to satisfy the LEN specification.

LOC

Points to the I/O buffer in main memory (this does not apply to CHK, see “relexp” below). The buffer must be large enough to hold at least as many PAM pages as specified by the value of LEN. The buffer address can be freely aligned on a byte boundary. If the buffer size is <= 4096 bytes (1 (main memory) page), it should be contained within one page and be aligned on a word boundary. If the buffer size is greater than 4096, it should be aligned on a page boundary. If the buffer alignment is not carried out in this manner, intermediate buffering may be required with certain hardware, and this is likely to have a detrimental effect on performance.

Default value:

- IOAREA2 address in the FCB if a PAM call has already been executed for this FCB and IOAREA1 was specified in the last PAM call;
- IOAREA1 address in the FCB in all other cases.

The LOC operand is ignored for the operations WT, LOCK, UNLOCK, SETL, SETLPP and SYNC.

= 1
This means that the buffer address is located in FCB field IOAREA1.

This specification is not permitted if IOAREA1=NO was specified in the OPEN macro. The (default) facility for switching buffers may be used only if neither IOAREA1=NO nor IOAREA2=NO was specified in OPEN.

= 2
This means that the buffer address is located in FCB field IOAREA2.

This specification is not permitted if IOAREA2=NO was specified in the OPEN macro. The (default) facility for switching buffers may be used only if neither IOAREA1=NO nor IOAREA2=NO was specified in OPEN.

= relexp
This specifies the buffer address or, in the case of a CHK operation, the continuation address if a checked I/O request has not been completed.

LOC=relexp is mandatory for a CHK operation; in other words, LOC must specify an address at which the task is to be continued if the checked I/O operation has not yet terminated.

LOCK

For disk files only:
Requests locks to be set on one or more PAM blocks (see also the operands HP and LEN). Locking a block means that other PAM calls with LOCK or LRD/LRDWT for this block will be rejected.

LRD

For disk files only:
Same as LOCK; if the lock is effected, processing continues as for RD.

LRDWT

For disk files only:
Same as LOCK and LRD; if the lock is effected, processing continues as for RDWT.

MKEY

For disk files only:
This operand is significant only for files with a PAM key (operand BLKCTRL=PAMKEY) and for chained I/O (see also the KEYFLD operand).

Default value: MKEY=NO

= NO
Only the PAM key of the first in a series of PAM pages is expected/provided by the user.

= YES
PAM keys are expected by the user for each PAM page read or provided by the user for each PAM page written.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The macro is expanded in accordance with the 24-bit interface format. The object code can only run in 24-bit addressing mode.

= 31
The macro is generated as addressing mode-independent.

POST = postcode

If UPAM TU eventing is used in conjunction with PARMOD=31, the user may enter a post code here; otherwise any entry in POST is ignored.

When I/O is complete, the post code can be retrieved in accordance with the SOLSIG definition:

- from the two rightmost bytes of the RPOSTAD field
- from the two rightmost bytes of the register specified in RPOSTAD
- from the two rightmost bytes of register 3 in the contingency process initiated by the UPAM event (for more details see the description of the SOLSIG macro in the “Executive Macros” manual [2]).

RD

Initiates the reading of a data block from the file into main memory; the job continues immediately after the read operation has been initiated.

RDEQU

Same as RDWT except that, for disk files with **D**ual **R**ecording by **V**olume (DRV, see the “DRV” [15] manual), the copy is also updated.
This operand value may be specified only for the 31-bit interface of the macro.

RDWT

Same as RD, except that the job does not continue until the read operation has been completed.

REQNO = number

For disk files only:
Specifies the number of the I/O request assigned to this operation. If this operand is specified, the FECB operand must be omitted.

Default value: REQNO=1

SETL

Causes the file pointer to be set to the specified PAM page.

SETLPP

Only for disk files:
Causes the last-page (end-of-file) pointer to be set to the specified PAM page; the PAM page must already belong to the file. This operation is not permitted for input files (OPEN=INPUT) or for files which have been opened with SHARUPD=WEAK/YES. Node files are truncated after SETLPP.

SYNC

Waits for termination of the I/O operation and clears the control buffer for tape cartridge files; for files which are not on a magnetic tape cartridge, SYNC is equivalent to WAIT.This operand value may be specified only for the 31-bit interface of the macro.

UNLOCK

For disk files only:
Unlocks locked PAM blocks (see also the operands HP and LEN).

WRT

Initiates the writing of a data block from main memory into the file; the job is continued immediately after the write operation has been initiated.

WRTWT

Same as WRT, except that the job does not continue until the write operation has been completed.

WRTWU

Same as WRTWT; the PAM page just written is unlocked immediately after completion of the I/O operation.

WT

This operation causes the program to wait for the end of a particular request. The program continues upon completion of this request.

Programming notes

1. The PAM macro overwrites the contents of registers 0, 1, 14 and 15.
2. The first PAM page of a PAM file is designated as number 1. When a PAM file is opened, the file pointer is set to an initial value of 0. The HP operand is ignored in WT, CHK and SYNC operations. WT and CHK refer to I/O requests, not to PAM pages.
3. With the exception of WT, CHK, SYNC and – for tape files- SETLPP operations, any operation not resulting in a branch to an error routine causes the file pointer to be set to the number of the PAM page currently being accessed. This is also true of locking/unlocking even where SHARUPD=NO is specified. When operations lead to an error routine instead of terminating normally, the value of the file pointer does not change.
   For PAM and SAM files without PAM keys (operand BLKCTRL=DATA or BLKCTRL=NO), the file pointer then contains the number of the last 2-Kbyte block which was accessed in the logical block. (During calculation of the file pointer, the length is thus rounded up to the next higher multiple of BLKSIZE.)
4. If, for a disk file, “n” denotes the number of PAM pages already allocated to a file and “k” is the value of the secondary allocation, the following rules must be observed:

PAM operation	**Meaning**
RD RDEQU RDWT LRD LRDWT	HP = number, where 1 <= number <= n
WRT WRTWT WRTWU	HP = number, where 1 <= number <= n + k As soon as a PAM page with LHP > n is written, a secondary allocation is made; the file size thus increases to n + k PAM pages. If several consecutive PAM pages are to be written using one PAM macro (chained I/O), sufficient space for all pages to be written must be available (at the latest) after the first secondary allocation, i.e. LHP <= n+k must be true for the last page.
LOCK UNLOCK	HP = number, where number > 0 PAM pages which are not allocated at the moment can be locked or unlocked later – this locking /unlocking does not result in a secondary allocation.

5. The value specified for REQNO must be less than or equal to the value defined in the PAMREQS operand in the FCB macro.
6. The REQNO operand is ignored for the LOCK, UNLOCK and SETL operations. In WT, CHK and SYNC operations, it serves to specify the I/O request for which the operation is to be carried out.
7. Only one asynchronous I/O operation can be assigned to each request number (REQNO). A waiting period for termination of the asynchronous I/O operation can be defined either explicitly by specifying its request number in a WT operation or implicitly by a second read or write operation for the same request number. If an error is detected during an implicit wait for the end of an I/O request, error code 997 is returned, and the request which caused the implicit WT is not carried out.
8. Request numbers (REQNOs) are system resources; improper use may impair the efficiency of the system.

9. If the user wishes to use the data buffer created by OPEN for I/O operations (operand LOC=1 or LOC=2), the value specified for the LEN operand must not be greater than BLKSIZE, since the buffer will otherwise be too small to hold the data.
10. For tape files, the value specified for LEN must be exactly equal to the length of a tape buffer. For files with BLKCTRL=PAMKEY, this is one PAM block (2048 bytes); this is equivalent to the specification LEN=STD or LEN=(STD,1) or LEN=2048 (16 bytes for the PAM key are added by the system).
11. For tape files without PAM keys (operand BLKCTRL=DATA or BLKCTRL=NO), a nonstandard block with the length specified in LEN is written; when reading, an attempt is made to read a block with the length specified in LEN. For PAM or SAM files without PAM keys, the value specified for LEN must not be greater than BLKSIZE; for ISAM files without PAM keys, it must not be greater than 2048 or (STD,1).
12. If, when reading from a tape file without a PAM key, the value specified for LEN is less than the block length of the file, the read operation is terminated with an error. In order to avoid such errors, one should always read a tape file without a PAM key with LEN=BLKSIZE, i.e. with the maximum permissible value for LEN. For files with BLKCTRL=DATA, the length of the valid data can then be determined from the block control field.
13. In the case of a write operation (WRT, WRTWT, WRTWU) to a file with BLKCTRL=DATA, UPAM treats the first 12 bytes as the block control field and overwrites them with the block control information.
14. For files with BLKCTRL=DATA and operations which require an input/output operation, the value for LEN must be selected such that the block control field is entirely contained in the data buffer. This is the case if the following conditions are fulfilled:
    - For PAM and SAM disk files:
      LEN = n*BLKSIZE or LEN > n*BLKLSIZE + 12 (n = number of data blocks)
    - For ISAM disk files:
      LEN = n*2048 or LEN > n*2048 + 12 (n = number of data blocks)
    - For tape files:
      LEN > 12
15. Files with BLKCTRL=DATA are subject to the following *restriction with regard to parallel processing*:
    when processing a PAM file with BLKCTRL=DATA, the IOAREAs must not be used in parallel for several I/O jobs, as this would mean that the contents of the block control field would be undefined.
    - Illegal approach:

      ```
      PAM   WRT,FCB1,LOC=BUFFER
      PAM   WRT,FCB2,LOC=BUFFER       (Parallel I/Os for BUFFER)
      PAM   WT,FCB1
      PAM   WT,FCB2
      ```

    - Permissible approach:

      ```
      PAM   WRTWT,FCB1,LOC=BUFFER     (Consecutive I/O operations
      PAM   WRTWT,FCB2,LOC=BUFFER     for BUFFER)
      ```

16. For LOCK or UNLOCK operations, LEN=0 is treated like LEN=2048 (or LEN=n, where 1 <= n <= 2048). If any other value is specified for LEN, as many PAM blocks as would be written or read by an I/O operation with this LEN value are locked.
    For PAM and SAM files without PAM keys (operand BLKCTRL=DATA or BLKCTRL=NO), only the first 2-Kbyte section of each affected logical block is actually locked internally, but this causes the entire logical block to be locked.

17. The following points apply to files with a PAM key (BLKCTRL=PAMKEY):
    - If a WT operation is performed (explicitly or implicitly) for a successful read operation, the 16-byte PAM key assigned to the block read is moved to the field designated by the KEYFLD operand. A CHK operation initiated after the completion of an I/O operation has the same effect as a WT operation.
    - If, for a write operation, the user places any information in the last 8 bytes of the 16-byte field defined by the KEYFLD operand, the information is written to the file as part of the PAM key assigned to the PAM page to be written (this is not recommended). UPAM always sets up the first 8 bytes of the KEYFLD area before the write operation begins.
    - The KEYFLD operand is ignored in conjunction with the WT, CHK, UNLOCK, SETL and SETLPP operations.

PUT - Write record

ISAM:	Macro type:	R for PARMOD=24 O for PARMOD=31
SAM:	Macro type:	R for PARMOD=24 R for PARMOD=31

Application area

The PUT function can be used when processing files with SAM or ISAM (record-oriented access methods). The file must have been opened with one of the following OPEN modes:

EXTEND	for SAM and
INOUT/OUTIN	ISAM
OUTPUT	for ISAM
	for SAM

Function

The PUT macro is used for sequential creation or extension of a file. It appends a record to the file.

For K-ISAM only:

The PUT macro can only be used to write records sequentially at the end of a file. If a file was opened in INOUT or OUTIN mode, PUT writes the record at the current end of the file. If the last macro issued for this file was not a PUT, the PUT performs a SETL to the end-of-file position before the record is written into the file.

ISAM:

The PAD factor (block padding) is taken into account during sequential creation or extension of a file. However, the effects for NK-ISAM and K-ISAM are different (For further details see "OPEN modes").

The PUT macro ensures that the key of any record added to the file is equal to or greater than the current highest key value. If the ascending sequence of the keys is broken, control is passed to one of the following two EXLST exits:

- DUPEKY, if the key of the current record is equal to the key of the preceding record and DUPEKY=YES was not specified.
- SEQCHK, if the key of the current record is less than the highest key of the exisiting file.

NK-ISAM:

If IOAREA1=NO is specified in the FCB for an NK-ISAM file, each PUT results in an SVC; the PAD value is ignored.

If IOAREA1 not equal to NO, the I/O area IOAREA1 is filled with records until:

- the buffer is full, or
- the limit specified by PAD in the FILE/FCB macro is exceeded, or
- an ISAM action macro other than PUT is issued.

An SVC is then issued and a new block with the contents of the I/O area is appended to the file.

For shared-update processing, the complete area between the current highest key and the end of the file is locked after a PUT.

Locate mode:

In locate mode, DMS supplies the IOREG register (see FCB macro, IOREG operand, "FCB - Define file control block") with the address of the first free byte within the buffer. The user must then ensure that the record to be included in the output file is made available at this address.

The PUT macro checks the record and updates the address in IOREG for the next record. After the last record has been made available at the address specified in IOREG, it is not necessary to call the PUT macro again (otherwise a CLOSE error will result or a corrupted record will be added to the file).

SAM:

When processing Format V records in locate mode, DMS specifies the buffer capacity remaining after each PUT macro in the VARBLD register (see FCB macro, VARBLD operand, "FCB - Define file control block". The application program must ensure that the remaining buffer area can accommodate a complete record. If there is not enough space left, the RELSE macro must be called to close a block.

Move mode – tape processing:

If, in the case of tape processing, a PUT macro initiates a transfer of the preceding records, and if the end-of-tape mark is detected during this transfer operation, the current record is then written, alone and unblocked, into a block on the tape before the tape swap is initiated. As a result, the file may become one block larger each time the tape is swapped.

Format

Operation	Operands
PUT	**fcbadr / (1)** **,area / (0)** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed.

(1)
The FCB address is stored in register 1.

area
Current address of the record to be transferred to the output buffer; The operand (if the IOREG operand is specified in the FCB) is ignored when processing files in locate mode.

(0)
The address of the record to be transferred to the output buffer is stored in register 0.

PARMOD
Specifies the generation mode for the macro.

Default value: the value preset in the program by means of the GPARMOD macro or by the assembler.

= 24
The macro is generated with the expansion for the 24-bit interface. The object code generated can run only in the 16-Mb address space (24-bit addressing).

= 31
The object code generated can run in the 2-Gb address space (24-bit or 31-bit addressing). The macro is generated such that it is independent of the addressing mode.

Programming notes

1. The PUT macro overwrites the contents of registers 0, 1, 14 and 15.
2. A PUT macro results in an SVC only when a buffer transfer is initiated. Consequently, the user cannot expect to receive control in an STXIT process each time a PUT macro is issued (when using the STXIT macro with SVC= or SVCLIST=; for more details see the “Executive Macros” manual [2]).

PUTX - Replace record

ISAM:	Macro type:	R for PARMOD=24 O for PARMOD=31
SAM:	Macro type:	R for PARMOD=24 R for PARMOD=31

SAM:

The PUTX macro may be used only for disk files processed in locate mode and opened with OPEN UPDATE. It writes an updated record back to the buffer. The record must have been retrieved beforehand by means of the GET macro (for ISAM also GETR, GETKY or GETFL) and its length must not have been changed during updating.

In locate mode, the PUTX macro places the record in the program buffer and sets a flag indicating that the buffer contents have been changed. The buffer contents are written back to the file only when a further macro (such as the next GET) triggers an SVC for this file or FCB.

ISAM:

The PUTX macro overwrites a record which was previously retrieved by means of a GET, GETR, GETKY or GETFL macro. With GETFL in conjunction with SHARUPD=YES, the LOCK operand must be selected.

The record key must not be changed either in move mode or locate mode; in locate mode, the record length must also not be changed.

With the exception of OSTAT, no other ISAM action macro may be issued for this file between the read operation using GET and the write operation using PUTX. For shared-update processing, the lock which was set implicitly or explicitly with the read operation using the LOCK operand must not be canceled. This means that any action macros issued even for other files/FCBs must not set a new lock; only macros such as a read operation with NOLOCK are possible.

K-ISAM:

When processing K-ISAM files in conjunction with SHARUPD=YES: if a record in a group of records with duplicate keys is output in move mode using PUTX, it is not known which of the records with duplicate keys the file will be positioned to following the PUTX.

The record key must not be changed between the read and write operations. If this is attempted, the effects vary depending on whether the file is being processed in move mode or locate mode:

in either case, the EXLST exit USERERR is taken.

If control information in the record is overwritten in locate mode, the results of subsequent processing will be unpredictable. If this makes it impossible for ISAM to process the block, control is passed to EXLST exit USERERR.

Format

Operation	Operands
PUTX	**fcbadr / (1)** **[,PARMOD = 24 / 31]** **[,area / (0)]**

Operand descriptions

area

Address of the record to be written.

(0)
The address of the record is stored in register 0.

fcbaddr

Address of the FCB associated with the file to be processed.

(1)
The FCB address is stored in register 1.

PARMOD

Specifies the generation mode for the macro.

Default value: the value preset in the program by means of the GPARMOD macro or by the assembler.

= 24
The macro is generated with the expansion for the 24-bit interface. The object code generated can run only in the 16-Mb address space (in 24-bit addressing mode only).

= 31
The macro is generated such that it is independent of the addressing mode (24-bit or 31-bit addressing). The object code generated can run in the 2-Gb address space.

Programming notes

1. The PUT macro overwrites the contents of registers 0, 1, 14 and 15.
2. *For SAM only:*
   The fact that a record has been updated in the current buffer is “noted” in the TU FCB by PUTX. It is only if this bit is set at the time of the next GET (RELSE, SETL) that the entire logical block is written to disk. PUTX **never** issues an SVC.

RDTFT - Read TFT and TST information

Macro type: type S (E form/L form/D form/C form); see "Macro types"

By means of the RDTFT macro, the user can fetch information from the task file table (TFT) and output it to a user area. If desired, he can also request information from the TST entry associated with the TFT entry.

Format

Operation	Operands
RDTFT	**outaddr** **[,[length]** **[,SHORT / LONG]** **[,FILE = pathname]** **[,LINK = name]** **[,NUMONLY = NO / YES]** **[,MF = L]** **[,PARMOD = 24 /** **PARMOD = 31 /** **[LINKWC = NO / YES,]VERSION = 2 / 3]**
	MF = (E,adr / (r)) **[,PARMOD = 24 /** **PARMOD = 31 } /** **VERSION = 2 / 3]**
	MF = D / C **[,PREFIX = pre / *]** **,PLIST = INPUT / OUTPUT** **[,PARMOD = 24 /** **PARMOD = 31 /** **VERSION = 2 / 3]**

Operand descriptions

outaddr

Symbolic address of the output area to which the information from the TFT is to be transferred. This address must be specified if MF=L is specified or if no entry is made for the MF operand.

length

Length of the user area
Minimum length: 11 bytes; exception with NUMONLY=YES: 4 bytes

Default value: 140 bytes for SHORT, PARMOD=24/31;
180 bytes for SHORT, VERSION=2/3;
2048 bytes if LONG is specified.

FILE = pathname

Specifies the file(s) on which information is to be supplied
with: <c-string 1..80: filename 1..54 with-wild(80)>

Designates the file or file generation from whose TFT entry information is to be supplied.Wildcards can be used to replace strings in “catid”, “userid” and “filename”; however, the total string length for “pathname” must not exceed 80 characters. Empty file names and the file name “*DUMMY” are not selected when wildcards are used.

If “pathname” is a file generation, the absolute generation number must be specified.

The internal file name is output for temporary files.

“pathname” means [:catid:][$userid.]filename

catid
Catalog ID; default value: the catalog ID assigned to the user ID.

userid
User ID; default value: the user's own ID.

filename
Fully or partially qualified file name.

LINK = name

File link name of the TFT entry from which information is to be passed to the output area (if appropriate, together with information from the associated TST entry).
Wildcards may be used to replace strings in “name”, in which case the following applies:

- The total string length must not exceed 80 characters.
- With the exception of the wildcards, the entire string may only contain characters from the permitted value range of the command interface.
- If wildcards are specified, only those TFT entries that have file link names which were constructed from characters in the permitted value range for the command interface will be selected.

LINKWC

Only as of VERSION=2:
Defines whether placeholders in the link name are to be interpreted as normal characters or as wildcards.

= NO
Placeholders are treated as normal characters.

= YES
Placeholders are interpreted as wildcards.

LONG

The complete TFT entry is placed in the user area, followed by the associated TSTentry including the device list.

MF

The forms of the MF operand are described in detail in the appendix ("Macro types"). In all macros differentiated by MF operands (MF=L/E/D/C), the version operand must contain the same value.
If MF is not specified, you get the S form; however, the explicit specification MF=S is only possible for PARMOD=24.

NUMONLY

Only as of VERSION=3:
Defines whether only the number of selected TFT entries is to be written to the output area.

= NO
The number of selected TFT entries is not transferred to the output area.

= YES
The selected TFT entries are counted and the result written into the first four bytes of the output area. Nothing further is output. The minimum size of the output area is four bytes.

PARMOD

Specifies the generation mode for the macro. If PARMOD and VERSION are specified simultaneously, the PARMOD operand is ignored and an MNOTE message is generated.

Default value: 31, if VERSION is specified,
otherwise the value for the generation mode preset by means of the GPARMOD macro or by the assembler.

= 24
The macro is generated with the expansion for the 24-bit interface. The object code can be executed only in 24-bit addressing mode.

= 31
The macro is generated such that it is independent of the addressing mode.

PLIST

Generates a list with symbolic addresses for the input or output area, depending on the VERSION and PARMOD operands.

= INPUT
Generates a list for the input area.

= OUTPUT
Generates a list for the output area.

PREFIX

Evaluated only in conjunction with MF=C or MF=D; defines the first character of each field name and equate generated in the data area when the macro is expanded.

Default value: PREFIX = I

= pre
The first character of the generated names is replaced by “pre”;
for the input area: 1-2 characters, first character a letter;
for the output area: 1 letter.

= *
No prefix is generated.

SHORT

The static part of the task file table (TFT) entry is transferred to the user area, followed by the static part of the associated TST entry (without the device list); see the LONG operand, below.

VERSION

Controls the generation of the operand list, SVC and output area. If PARMOD and VERSION are specified simultaneously, the PARMOD operand is ignored and an MNOTE message is generated.

The return code is stored exclusively in the parameter list standard header. The RDTFT function also supplies the new output list (see "Programming notes").If a PARMOD value is specified simultaneously, it is ignored.

Default value: generation of the operand list is controlled by the PARMOD operand.

= 2
Creates the operand list valid as of BS2000 V9.5. The return code is stored in the standard header and in register 15.

= 3
Specifies the macro version: the operand list for the macro version valid as of BS2000/OSD-BC V3.0 is generated.

Programming notes

Input area

If the VERSION operand is omitted, the old operand list is generated, depending on the value of the PARMOD operand. A Dsect can be generated for this area by means of the DMARD macro or via the operands MF=D, PLIST=INPUT.

If VERSION=2/3 is specified, the operand list includes the standard header and the field which contains the length of the output area is defined as an address constant (4 bytes). A Dsect for this area is generated using the operands VERSION=2/3, MF=D, PLIST= INPUT.

Output area

The RDTFT macro returns one of two lists, depending on whether or not the operand LINK=name is specified:

- without LINK=name, or if “name” includes wildcards and the operand LINKWC=YES is specified, only a list of the file link names and the file names linked to them is transferred to the output area.

  The list is in chronological order, i.e. in the order in which the TFT entries were created. Each link name/file name pair is preceded by one byte which contains the length of these two fields + 1.
  The list is terminated by one byte containing the value X'00', and the byte after this shows whether or not all requested information has been transferred to the output area.

  X'00' All file link names and the related file names have been transferred to the user area.

  X'01' One or more file link names, with their related file names, could not be placed in the user area, since it was already full.

  If NUMONLY=YES is specified, the number of selected TFT entries is output to the first four bytes of the output area. Nothing further is output.

- with LINK=name, the TFT and TST information for the specified file link name is placed in the output area, provided “name” does not contain wildcards.

  The output area then consists of a fixed part and a variable part. The fixed part contains information from the TFT and TST. The variable part is created only if the LONG operand is specified; it contains further information from the TFT and, possibly, TST volume information.

Field length (bytes) with VERSION	Field length (bytes) without VERSION	Contents
2	2	Length of output area without variable part
8	8	File link name
54	54	Path name
116	76	Static part of TFT entry and associated TST entry
variable	variable	TFT and TST volume information (if LONG is specified)

If the VERSION operand is omitted, the output list has the old format shown in the table above. A Dsect for this output list can be generated by means of the DMADR macro or via the operands MF=D, PLIST=OUTPUT of the RDTFT macro.
If the operand VERSION=2/3 is specified, the new format of the output list is generated. This has a few more fields than the old format. In addition to the information in the previous output list, the new one provides above all information on BLKCTRL, POOLLNK, TAPEWR, CLOSMSG, CLOSE, IOPERF, IOUSAGE, EXC32GBand DESTOC (see the FILE macro, "FILE - Define file attributes / control file processing"). Moreover, the device type is output in printable form (e.g. “D3439-10”). For output with LONG, the TFT volume information was extended.
A Dsect for this output list can be generated via the operands VERSION=2/3, MF=D, PLIST=OUTPUT.

- With PARMOD=31 or VERSION=2/3, the path name is always displayed in full; in all other cases it is output in the form in which it was placed in the TFT entry using FILE or OPEN, i.e. without a catalog ID or user ID if these were omitted. However, it is possible to specify, for the entire system, that the catalog ID and user ID are always to be output.
  If the TFT entry is not linked to a TST entry, the appropriate part of the output area is filled with binary zeros (X'00').

Return codes

The error code is returned in the parameter area standard header. Error code 0 means that no errors occurred. The other error codes are described in the DMAIDEM or DCOIDEM macro.

Program termination with STXIT connection can be initiated in the following cases:

- parameter address incorrect (e.g. shorter than the standard header)
- parameter address not aligned on a word boundary
- UNIT or FUNCTION in header incorrect
- header is write-protected

If the version value is incorrect, the code is returned in the header and in register 15. If the header is not writable, the program is terminated with STXIT connection. The subcodes are only given values as of VERSION=3.

Standard header: `ccbbaaaa`

The following code relating to execution of the RDTFT macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
	X'01'	X'059D'	Invalid link name
	X'01'	X'05AB'	Invalid range or length
	X'40'	X'05E1'	Link name not found
	X'01'	X'06CB'	Output area too small
	X'01'	X'06FD'	Invalid parameter list range
	X'03'	X'FFFF'	Invalid version

Example

```
BEGIN   START
        .
        .
        RDTFT MF=(E,EXTENT),VERSION=3                          RDTFT-SVC
        .
        .
EXTENT  RDTFT OUTPUT,,SHORT,LINK=JOHN,MF=L,VERSION=3       OPERAND LIST
        .
        .
OUTPUT  DS    XL180                                          OUTPUT AREA
        .
INDSK   RDTFT MF=D,PLIST=INPUT,VERSION=3       DSECT INPUT OPERAND LIST
        .
OUTDSK  RDTFT MF=D,PLIST=OUTPUT,VERSION=3     DSECT OUTPUT OPERAND LIST
        .
        .
        END
```

RELSE - Close block

Macro type: type R

The RELSE macro has the following effect, depending on the file type:

- for files opened with OPEN INPUT/REVERSE/UPDATE, the records remaining in the buffer are ignored by the next GET macro, and the first record in the next block is read from the file;
- for files opened with OPEN OUTPUT/EXTEND, the next PUT macro writes the block to the file and the next record becomes the first record of a new block;
  In locate mode, DMS supplies the IOREG register (see FCB macro, IOREG operand, "FCB - Define file control block") with the address of the first free byte within the buffer. The user must then ensure that the record to be included in the output file is made available at this address.
  If format V records are processed in locate mode then the free buffer capacity is displayed in the VARBLD register (see FCB macro, VARBLD operand, "FCB - Define file control block"). After the RELSE call, the free buffer capacity is equal to the block length minus the buffer offset (see BLKSIZE and/or BUFOFF in the FCB macro, "FCB - Define file control block" and "FCB - Define file control block" respectively).
- for files opened with OPEN UPDATE, the current data block is written back to the disk (if a record has been updated with PUTX), and the next GET returns the first record of the next data block.

Format

Operation	Operands
RELSE	**fcbadr / (1)** **[,PARMOD = 24 / 31]** **[,SYNC = NO / YES]**

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed.

(1)
The FCB address is stored in register 1.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The macro is expanded in accordance with the 24-bit interface format. The object code can run only in 24-bit addressing mode.

= 31
The macro is generated as addressing mode-independent.

SYNC

The processing of output files can be synchronized.

Default value: SYNC=NO

= NO
File processing is not synchronized.

= YES
Permitted only in conjunction with PARMOD=31; file processing is synchronized, i.e. after a RELSE..., SYNC=YES all data is on the disk or tape and there is no WAIT outstanding.

For files on tape cartridges, RELSE..., SYNC=YES causes the contents of the device buffer to be written to the tape cartridge.

Programming notes

1. The RELSE macro overwrites the contents of registers 0, 1, 14 and 15.
2. A RELSE macro results in an SVC only when a buffer transfer is initiated. Consequently, the user cannot expect to receive control in an STXIT process each time a RELSE macro is issued (when using the STXIT macro with the SVC= or SVCLIST= operand; for more details, see the “Executive Macros” manual [2]).

RELTFT - Delete TFT entry

Macro type: type S (C form/D form/E form/L form/M form); see "Macro types"

The RELTFT macro deletes from the task file table (TFT) the entry for the specified file link name and releases all private volumes, devices and Net-Storage volumes linked to this entry. When a private volume or a Net-Storage volume with more than one TFT entry is connected, it is released only when the last TFT entry is deleted. It also cancels the reservation of files reserved exclusively by means of the SECURE-RESOURCE-ALLOCATION command. RELTFT is ignored if a LOCK-FILE-LINK is still active for the TFTentry; it is executed only when this lock is released by means of the DROPTFT macro (see "DROPTFT - Release TFT entry") or the UNLOCK-FILE-LINK command (or at LOGOFF time. For further information on the LOCK-FILE-LINK, UNLOCK-FILE-LINK and SECURE-RESOURCE-ALLOCATION commands see the “Commands” [3] manual and the “Introductory Guide to DMS” [1]).

Note

> The RELTFT macro is the earlier REL macro, extended by the use of wildcards in link names. As the previous functionality of the REL macro is still supported, its format is still shown in the appendix (see "Formats of replaced macros"). Its operands, however, are the same as those of the RELTFT macro and are therefore only described here.

Tape files

In the RELTFT macro, the user can select whether the tape devices requested for the tape file (KEEP operand) and /or the requested volumes (UNLOAD operand) are to remain assigned to the job.

The file counter in the TST entry is decremented by 1 if the TFT entry to be released is linked to a TST entry. Once this reaches zero, the TST entry is deleted and DMS releases all devices linked to it. As long as the TST entry is still linked to TFT entries (file counter > zero), DMS releases only those devices that were only requested for the TFT entry named in the RELTFT macro.

If the TFT entry to be released does not reference a TST entry, all devices linked to the TFT entry are released.

Format

Operation	Operands

RELTFT	**KEEP = *NO / *YES**
	,LINK = <c-string 1..80:filename 1..8 with-wild(80) without-gen> /
	<var: char:80: filename 1..8 with-wild(80) without-gen>
	,UNLOAD = *NO / *YES
	,VERSION = <integer 1..1>
	,WILDCRD = *NO / *YES
	,MF = C / D / E / L / M
	,PARAM = <addr> / <(r)>
	,PREFIX = D / <pre>
	,MACID = MAR / <macid>

Operand descriptions

KEEP

Defines whether tape devices linked to this file or TFT entry are not to be returned to the system, but instead are to remain available to the job for reassignment.

= *NO
The tape devices are returned to the system.

= *YES
The tape devices are not returned to the system.

LINK

File link name of the TFT entry to be deleted.

Character strings in the specified name can be represented by wildcards. The following then applies:

1. The total length of the name must not exceed 80 characters.
2. The name may only contain wildcards or characters from the permitted value range of the command interface.
3. If wildcards are used, only those TFT entries are selected whose link names are made up of characters from the permitted value range of the command interface.

Default value: The first TFT entry with the link name *BLANK is deleted.

= <c-string 1..80: filename 1..8 with-wild(80) without-gen>

File link name (enclosed in single quotes).

= <var: char: 80: filename 1..8 with-wild(80) without-gen>
Name of a variable containing the file link name.

MACID

Only evaluated in conjunction with MF=C/D/M and defines the second through fourth characters of field names and equates generated in the data area when the macro is expanded.

Default value: MACID = MAR

= macid
“macid” is a three-character string which defines the second through fourth characters of generated field names and equates.

MF

The forms of the MF operand are described in the appendix ("Macro types").

Default value: Operand list and SVC as previously

PARAM

Designates the address of the operand list and is only evaluated in conjunction with MF=E (see also "Macro types").

= addr
“addr” is the symbolic address (name) of the operand list.

= (r)
“r” is the number of the register containing the address of the operand list. The register must be loaded with this address value before the macro is called.

PREFIX

Only evaluated in conjunction with MF=C/D/M; this defines the first character of field names and equates generated in the data area when the macro is expanded.

Default value: PREFIX = D

= pre
“pre” is a single-character prefix with which the field names and equates generated by the assembler are to begin.

UNLOAD

Only for tape files:
Defines whether the volumes linked to the TFT entry specified under LINK are released and the tape devices concerned are unloaded. They must be requested again if the job is to reaccess these volumes. Automatic allocation by the system is no longer possible.

If a private volume is linked to different TFT entries, it is released only after the last TFT entry is deleted.

= *NO
The tape devices are released.

= *YES
The tape devices are not released.

VERSION = <integer 1..1>

Control operand; controls generation.

WILDCRD

Defines whether wildcard characters in the file link name are interpreted as wildcards or as normal characters.

= *NO
Wildcard characters are interpreted as normal characters.

= *YES
Wildcard characters are interpreted as wildcards.

Programming notes

The error code is only returned in the standard header and no longer in general-purpose register 15, as it was in the REL macro. Program termination with STXIT connection can be initiated in the following cases:

- parameter address faulty (e.g. shorter than the standard header)
- parameter address not aligned on a word boundary

- UNIT or FUNCTION in the header faulty
- header is write-protected

Return codes

Standard header: `ccbbaaaa`

The following code relating to execution of the RELTFT macro is returned in the standard header(cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
X'01'	X'00'	X'059A'	No such link name
	X'01'	X'059D'	Invalid link name
	X'82'	X'059B'	File is currently opened
X'02'	X'00'	X'059C'	Not all selected TFT entries are deleted
	X'01'	X'05C2'	Invalid link name (binary null)
	X'01'	X'06F5'	TPR bit set by TU caller
	X'01'	X'06FD'	Invalid parameter list range
	X'03'	X'FFFF'	Invalid version

REMPLNK - Delete pool link name

Macro type: type S (E form/L form/D form/C form); see "Macro types"

By means of the REMPLNK macro, the user deletes either one specific pool link name or all pool link names from the job's pool table. The file linked to the pool link name must have been closed correctly before the pool link name can be deleted. If REMPLNK is issued for all names in the pool table, any pool link names linked to files which are still open are not deleted, but macro execution is regarded as successfully completed.

Formatt

Operation	Operands
REMPLNK	**MF = L** **,MODE = SINGLE,LINKNAME = name /** **ALL**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = pre]
	MF = C[,PREFIX = pre][,MACID = macid]

Operand descriptions

MACID

Evaluated only in conjunction with MF=C; defines the second through fourth characters of each field name and equate generated in the data area when the macro is expanded.

Default value: MACID = ISR

= macid
Three-character string defining the second through fourth characters of the generated field names and equates.

MF

The forms of the MF operand are described in detail in the appendix ("Macro types").

MODE

Specifies whether one specific pool link name or all pool link names in the job's pool table are to be deleted.

= SINGLE
The user must specify a pool link name.The pool link name specified for LINKNME is to be deleted.

LINKNME = name
Specifies which pool link name is to be deleted. “name” is a pool link name defined by means of the ADDPLNK macro.

= ALL
All pool link names which are not linked to files which are still open are to be deleted.

PARAM

Specifies the address of the operand list; evaluated only in conjunction with MF=E (see "Macro types").

= addr
Symbolic address (name) of the operand list.

= (r)
Number of the register containing the address of the operand list. The register must be loaded with this address value before the macro is called.

PREFIX

Evaluated only in conjunction with MF=C or MF=D; defines the first character of each field name and equate generated in the data area when the macro is expanded.

Default value: PREFIX = D

= pre
One-character prefix with which the field names and equates generated by the assembler are to begin.

Return codes

Unless otherwise specified, the field names and the EQU statements for the return codes generated by the C and D forms of the macro begin with the character string DISR, but this can be modified by PREFIX and MACID.

The return codes are stored in the standard header of the operand list.

Main return code	Meaning
DISROK X'0000'	The macro was executed successfully
DISRNPAR X'0001'	Access to the operand list was not possible
DISRNREM X'0002'	ISAM pools not supported on remote host
DISRINVN X'0003'	Catalog ID unknown
DISRNACC X'0004'	Catalog ID not accessible
DISRINVN X'0005'	The specified pool link name is invalid
DISRNANF X'0006'	The pool link name was not found
DISRPUSE X'0009'	The specified pool link name is linked to a file which is still open
DISDSYSE X'000B'	A system error occurred during macro execution
DISRRLNK X'FFFF'	The macro could not be executed (linkage error): evaluate subsidiary return code 1

RETRY - Repeat macro

Macro type: R for PARMOD=24
O for PARMOD=31

The RETRY macro is needed in routines which are activated when a PGLOCK event occurs (EXLST exit PGLOCK). PGLOCK occurs only when an ISAM action macro is issued for a block or record which is already locked by another job. This “unsuccessful” macro can be repeated using RETRY, and the user can specify in the program whether or not RETRY is to wait for the lock to be cleared.

The RETRY macro may be called only in PGLOCK routines. If it is used elsewhere in a program, control is passed to the EXLST exit USERERR.

When the PGLOCK exit is activated, register 1 contains the FCB address; when the RETRY macro is executed, register 1 must – again – contain this FCB address.

When, after successful execution of the RETRY macro, control is returned to the caller, the contents of the registers are the same as after an action macro which was executed immediately.

If the RETRY macro is unsuccessful and control is passed to the FAIL routine, the contents of the registers are the same as before the RETRY macro was called (except for registers 0, 1, 14 and 15).

For NK-ISAM, the pointers are positioned as they were before the macro was called before the PGLOCK routine is activated (see the table "Rules for ISAM pointers").

For K-ISAM, the following should be noted: when the PGLOCK exit of the EXLST macro is activated, the internal pointers are positioned correctly only if the macro which resulted in the PGLOCK condition was a PUTX or an ELIM (without KEY). The ISAM macros GET, GETR and GETFL have already changed the pointers before they branch to PGLOCK. The RETRY macro can reset the pointers to their previous positions and can restart the macro which led to the PGLOCK condition.

If the last record which was successfully accessed before the PGLOCK exit was activated is one of a group of records with duplicate keys, resetting of the pointers to their previous positions will position the file to the first record in this group.

If ACTION=POS is specified and positioning is executed successfully, control is returned to the calling job at the instruction following the RETRY macro.

Format

Operation	Operands
RETRY	**FAIL = addrexp** **[,ACTION = RETRY[,COUNT = number] / WAIT / POS]** **[,PARMOD = 24 / 31]**

Operand descriptions

FAIL = addrexp

Address to which control is to be transferred if RETRY fails or if the waiting time for access to a block is longer than 30 minutes.

ACTION
Specifies which action is to be taken by RETRY.

= RETRY
The macro is to be retried. The operand COUNT specifies how many times the macro is to be repeated before control is passed to the FAIL address.

= WAIT
The macro is retried. If the block is still not available, the job is placed in a queue. After a wait of 30 minutes (maximum), control is passed to the FAIL routine.

= POS
For K-ISAM: the pointer is to be repositioned in the file.
NK-ISAM supports this value only for compatibility reasons.

COUNT = number
Specifies how many times the macro is to be retried before control is passed to the FAIL routine; 0 <= number <= 255.

Default value: COUNT = 1

PARMOD
Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the
GPARMOD macro or preset by the assembler.

= 24
The object code generated can run only in the 16-Mb address space (24-bit addressing only).

= 31
The object code generated can run in the 2-Gb address space (24-bit or 31-bit addressing).

RFFSNAP- Restore files from Snapset

Macro type: type S (E form/L form/D form/C form/M form) (see "Macro format")

The RFFSNAP macro restores files of a pubset from a pubset copy which was created on an associated Snapset. During the restore operation, single files are copied from the Snapsets onto the active pubset. The process is comparable to an HSMS restore from a backup archive.

The Snapset entry enables a specific backup status (the default is the latest Snapset backup) to be specified, or the user can specify that each file should be restored from the Snapset with the latest file status. Before restoration takes place, the user can use the LFFSNAP macro to obtain information on files which were saved to a Snapset.

All file attributes of a restored file are taken over from the original file unchanged (including the creation date, date of modification and the protection attributes). Only the allocation may differ from the original file, even in the case of files with physical allocation. Files on SM pubsets are restored to the “most suitable” volume set. This need not be the original volume set.

Individual file generations can only be restored with the entire file generation group. Files on private disk are ignored. In the case of migrated files and tape files, only the catalog entries are restored (without checking the availability of the associated tapes). When renaming takes place, these files are also ignored.

Nonprivileged users can only restore a file of a foreign user if they are the co-owner.

Overwriting by the restore operation must be explicitly permitted for existing files (REPLACE operand). For files which are protected against unauthorized overwriting by means of a password, the required password must be entered in the caller’s password table (ADD-PASSWORD command).

Files can also be restored under a new name. They are renamed by specifying another user ID (NUSERID operand) and/or a file name prefix (NPREFIX operand).

Optionally, files which were open in write mode at the time the Snapset was created can be restored (RESTOPN operand). A file restored in this way has the same status as after a system crash. ISAM files may need to be verified. Files with the OPNBACK attribute (see the macro "CATAL - Process catalog entry") which are opened in write mode are restored regardless of this option.

If required, the caller can have a log of restore processing output to SYSOUT (LIST operand). This log can cover either all files or only the files which, for particular reasons, could not be restored.

The Snapsets are temporarily not available if the SHC-OSD subsystem was not active when the pubset was imported. In this case the command is aborted with return code DMS0622. As soon as SHC-OSD is active, the Snapsets are subsequently activated when the SHOW-SNAPSET-CONFIGURATION command is called.

Privileged functions

Systems support (TSOS privilege), as co-owner, can restore all files under their original user IDs.

When a file which still exists is overwritten, systems support can explicitly bypass the file protection by means of the IGNPROT operand.

Systems support can only log the restoration of files via the SECOS component SAT if it has the calls used for deleting files (when overwriting) and for creating an entry in the file catalog logged.

Format

Operation	Operands
RFFSNAP	,PATHNAM=<c-string 1..80: filename 1..54 with-wild(80)> / <var: char:80> ,SNAPSET=<integer -52..-1> / *LATEST / *ALL ,SNAPID=<c-string 1..1: name 1..1 with-low> / <var: char: 1> ,REPLACE=*NO / *YES / <var: enum-of_replace_s: 1> ,IGNPROT=*NO / *YES / <var: enum-of_ignprot_s: 1> ,RESTOPN=*NO / *YES / <var: enum-of_restop_s: 1> ,NUSERID=<c-string 1..8: name 1..8> / <var: char:8> ,NPREFIX=<c-string 1..8: name 1..8> / <var: char:8> ,LIST=*NO / *SYSOUT / *ERRORS-TO-SYSOUT / <var: enum-of_list_s: 1> ,EQUATES=*YES / *NO MF=L
	MF=D,PREFIX= D / <pre>
	MF=E,PARAM=<name 1..27>

	MF=C / M
	,PREFIX= D / <pre>
	,MACID= MAR / <macid>

Operand descriptions

PATHNAM

Selects the files which are to be restored.

=<c-string 1..80: filename 1..54 with-wild(80)>
Path name of the file(s) on the Snapset. Wildcards can be used to specify a set of files.

The files must satisfy the following requirements:

- They must be cataloged when the Snapset is created.
- The pubset on which they are cataloged must be imported locally.
- They may not reside on private disk.

The catalog and user IDs specified must be unique (i.e. contain no wildcards). Aliases (also partially-qualified aliases) may be specified. The name of a file generation group may be specified, but not the name of an individual file generation (individual file generations can only be restored within the group).

Privileged users (TSOS privilege) can restore files of all user IDs.

=<var: char:80>
Only possible with MF=M:
Symbolic address of a memory area of 80 bytes in which the path name or wildcard string for the required file (s) is stored.

SNAPSET
This operand may not be specified together with the SNAPID operand.
Specifies the Snapset from which restoration is to take place by means of the relative age.

=<integer -52..-1>
Specifies the Snapset explicitly by means of the relative age. The value -1 specifies the latest Snapset (also corresponds to *LATEST).

=*LATEST
Specifies the latest Snapset.

=*ALL
All Snapsets of the pubset concerned are used as a basis for restoration. Each file is restored from the Snapset with the latest file status, in other words with the latest backup of the file. A file which cannot be restored with the latest file status is in this case not restorable (i.e. older backup statuses are ignored).

SNAPID

This operand may not be specified together with the SNAPSET operand.
Specifies the Snapset from which restoration is to take place by means of the Snapset ID.

=<c-string 1..1: name 1..1 with-low>
Specifies the Snapset explicitly by means of the Snapset ID. The maximum of 52 Snapsets for a pubsets are distinguished by means of Snapset IDs specified which comprise letters from the 26 lowercase letters a to z and the 26 uppercase letters A to Z.

=<var: char:1>
Only possible with MF=M:
Symbolic address of a memory area of 1 byte in which the Snapset ID is stored.

Note

If neither SNAPSET nor SNAPID is specified, the latest Snapset is used.

REPLACE
Specifies whether the files to be restored may overwrite existing files.

=*NO
Existing files are not overwritten. This means that files with the names of existing files are not restored.

=*YES
Existing files may be overwritten by files which are to be restored provided the protection attributes permit this. For files which are protected against unauthorized overwriting by means of a password, the required password must be entered in the caller’s password table (see the ADD-PASSWORD command).

=<var: enum-of_replace_s: 1>
Name of the field with the value for REPLACE.

IGNPROT
This operand is only available to privileged users (TSOS privilege).
Specifies whether files are to be overwritten without taking into account any write protection which exists.

=*NO
Write protection is taken into account.

=*YES
Write protection is ignored.

=<var: enum-of_ignprot_s: 1>
Name of the field with the value for IGNPROT.

RESTOPN

Specifies whether files which were open in write mode when they were saved to the Snapset and for which the OPNBACK file attribute (see macro "CATAL - Process catalog entry") was not set are also to be restored.

=*NO
These files are not restored. Consequently only files which were not open in write mode when they were saved and files for which the OPNBACK file attribute was set are restored.

=*YES
These files are also restored. The consistency is the same as after a system crash (write accesses in the correct order). ISAM files may need to be verified (REPAIR-DISK-FILE command).

=<var: enum-of_restop_s: 1>
Name of the field with the value for RESTOPN.

NUSERID
Specifies that the files are to be renamed when they are restored and are to be restored under the specified user ID.
This operand may not be specified together with the NPREFIX operand.

=<c-string 1..8: name 1..8>
User ID.

=<var: char:8>
Only possible with MF=M:
Symbolic address of a memory area of 8 bytes in which the user ID is stored.

NPREFIX=
Specifies that the files are to be renamed when they are restored and are to be assigned the specified file name prefix.
This operand may not be specified together with the NUSERID operand.

=<c-string 1..8: name 1..8>
File name prefix.

=<var: char:8>
Only possible with MF=M:
Symbolic address of a memory area of 8 bytes in which the file name prefix is stored.

LIST

Specifies which processing results are to be logged to SYSOUT.

=*NO
No output is directed to SYSOUT.

=*SYSOUT
All files are listed. For files which could not be restored, the reason is displayed by means of a message code.

=*ERRORS-TO-SYSOUT
Only files which cannot be restored are listed. The reason is displayed by means of a message code.

=<var: enum-of_list_s: 1>
Name of the field with the value for LIST.

EQUATES

Control operand; for MF=C and MF=D only:
Specifies whether equates are also to be generated for the values of the fields of the parameter area when the parameter area is expanded.

= *YES
When the parameter area is expanded, equates are also generated for the values of the fields of the parameter area.

= *NO
When the parameter area is expanded, no equates are generated for the values of the fields of the parameter area.

Return codes

The return code is placed in the standard header of the parameter area. The parameter area may then not be located in the read-only area, otherwise the program terminates.

Standard header: `ccbbaaaa`

The following code relating to execution of the RFFSNAP macro is returned in the standard header(cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
X'00'	X'00'	X'0000'	No error
X'00'	X'40'	X'0501'	Requested catalog not available
X'00'	X'40'	X'0505'	Error in host communication
X'00'	X'40'	X'0512'	Requested catalog not found
X'00'	X'40'	X'051B'	Requested user ID not on the pubset
X'00'	X'40'	X'051D'	LOGON password different on specified pubset
X'00'	X'20'	X'0531'	Unexpected error during catalog access

X'00'	X'40'	X'0535'	Specified file not accessible
X'00'	X'82'	X'053C'	No space in the pubset’s catalog
X'00'	X'82'	X'0541'	Not enough disk storage space available
X'00'	X'40'	X'0554'	Format of file name invalid
X'00'	X'40'	X'057F'	Migrated file cannot be renamed
X'00'	X'20'	X'0584'	Internal error
X'00'	X'82'	X'0594'	Not enough virtual memory

X'00'	X'82'	X'05B1'	File lock exists for file
X'02'	X'00'	X'05B6'	Incorrect time conversion in GTIME macro
X'00'	X'40'	X'05BF'	Password not specified
X'00'	X'82'	X'05C3'	File currently locked or in use
X'00'	X'40'	X'05C6	Expiration date not yet reached
X'00'	X'20'	X'05C7'	Internal error in DMS
X'00'	X'01'	X'05CB'	Incorrect first file name or foreign ID specified
X'00'	X'01'	X'05EE'	Path name too long after completion
X'00'	X'40'	X'05FC'	Specified user ID not on home pubset
X'00'	X'40'	X'0610'	Execution of the function returned a return code for at least one of the selected file names
X'00'	X'40'	X'0615'	The file does not reside on the available volume set
X'00'	X'40'	X'0616'	Volume set cannot be accessed on SM pubset
X'00'	X'40'	X'0620'	No restorable file found
X'00'	X'40'	X'0621'	File already cataloged, restoration not performed
X'00'	X'40'	X'0622'	Snapset not available
X'00'	X'01'	X'0623'	Generation cannot be restored
X'00'	X'01'	X'0624'	File name invalid
X'00'	X'40'	X'0681'	DMS error when accessing file
X'00'	X'40'	X'0684'	File does not exist
X'00'	X'01'	X'06C5'	FGG name too long
X'00'	X'01'	X'06C6'	Name of a tape file cannot be changed
X'00'	X'40'	X'06CC'	No file name matches the wildcard string specified
X'00'	X'40'	X'06D5'	File protected and consequently cannot be overwritten
X'00'	X'01'	X'06F7'	Invalid operand value
X'00'	X'01'	X'06FD'	Parameter area invalid or not accessible

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller

- the list is not aligned on a word boundary
- the list is write-protected.

Layout of the operand list

Macro expansion with MF=D, and with default values for EQUATES, PREFIX and MACID:

```
         RFFSNAP MF=D
DMARRFPL DSECT ,
DMARHDR  DS    0A
DMARFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
DMARIFID DS    0A              0   INTERFACE IDENTIFIER
DMARFCTU DS    AL2             0   FUNCTION UNIT NUMBER
DMARFCT  DS    AL1             2   FUNCTION NUMBER
DMARFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
DMARRET  DS    0A              4   GENERAL RETURN CODE
DMARSRET DS    0AL2            4   SUB RETURN CODE
DMARSR2  DS    AL1             4   SUB RETURN CODE 2
DMARSR1  DS    AL1             5   SUB RETURN CODE 1
DMARMRET DS    0AL2            6   MAIN RETURN CODE
DMARMR2  DS    AL1             6   MAIN RETURN CODE 2
DMARMR1  DS    AL1             7   MAIN RETURN CODE 1
DMARFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
*
DMARPNAM DS    CL80                    PATHNAME
DMARSNAP DS    FL1                     SNAPSET
*   SNAPSET - VALUES
DMARSNIN EQU   0                       SNAPSET=<integer>
DMARSNCH EQU   1                       SNAPSET=<char>
DMARSNLT EQU   2                       SNAPSET=*LATEST
DMARSNAL EQU   3                       SNAPSET=*ALL
*
DMARREPL DS    FL1                     REPLACE
 *   REPLACE - VALUES
DMARREPY EQU   0                       REPLACE = YES
DMARREPN EQU   1                       REPLACE = NO
*
DMARIGNP DS    FL1                     IGNPROT
*   IGNPROT VALUES
DMARIGNO EQU   0                       IGNPROT = NO
DMARIGYE EQU   1                       IGNPROT = YES
*
DMARLIST DS    FL1                     LIST
*   LIST - VALUES
DMARLSTN EQU   0                       LIST = NO
DMARLSYO EQU   1                       LIST = SYSOUT
DMARLSYE EQU   2                       LIST = ERRORS
*
DMARNUSR DS    CL8                     NUSERID
DMARNPRE DS    CL8                     NPREFIX
DMARSNVL DS    H                       SnapValue
DMARSNID DS    CL1                     Snapid
DMAROFLG DS    AL1                     FLAG BYTE
DMARNUSP EQU   X'80'                   S: NUSERID SPECIFIED
DMARNPSP EQU   X'40'                   S: NPREFIX SPECIFIED
DMARRESB EQU   X'3F'                   RESERVED
DMARRESO DS    FL1                     RESTOPN
*   RESTOPN VALUES
DMARRONO EQU   0                       RESTOPN = NO
DMARROYE EQU   1                       RESTOPN = YES
*
DMARRES1 DS    XL3                     ALIGNMENT
DMAR#    EQU   *-DMARHDR
```

Sample calling sequence

```
        MVC   RFFSMFC(DMAR#),RFFSMFL
        RFFSNAP MF=M,PATHNAM=':X:TTT',PARAM=RFFSMFC
        RFFSNAP MF=E,PARAM=RFFSMFC
             .
             .
RFFSMFC RFFSNAP MF=C
RFFSMFL RFFSNAP MF=L,...
```

RJFSNAP- Restore job variables from a Snapset

Macro type: type S (E form/L form/D form/C form/M form) (see "Macro format")

The RJFSNAP macro restores job variables of a pubset from a pubset copy which was created on an associated Snapset. During the restore operation, single job variables are copied from the Snapsets onto the active pubset. The process is comparable to an HSMS restore from a backup archive.

The Snapset operand enables a specific backup status (the default is the latest Snapset backup) to be specified, or the user can specify that each job variable should be restored from the Snapset with the latest job variable status. Before restoration takes place, the user can issue the LJFSNAP macro to obtain information on job variables which were saved to a Snapset.

All attributes of a restored job variable are taken over from the original job variable unchanged (including the creation date, date of modification and the protection attributes).

Nonprivileged users can only restore a job variable of a foreign user if they are the co-owner.

Overwriting by the restore operation must be explicitly permitted for existing job variables (REPLACE operand). For job variables which are protected against unauthorized overwriting by means of a password, the required password must be entered in the caller’s password table (see ADD-PASSWORD).

Job variables can also be restored under a new name (NEW-JV-NAME operand). They are renamed by specifying another user ID (NUSERID operand) and/or a name prefix (NPREFIX operand).

If required, the caller can have a log of the restore processing output to SYSOUT (LIST operand). This log can cover either all job variables or only the job variables which, for particular reasons, could not be restored.

The Snapsets are temporarily not available if the SHC-OSD subsystem was not active when the pubset was imported. In this case the macro is aborted with return code 0622. As soon as SHC-OSD is active, the Snapsets are subsequently activated when the SHOW-SNAPSET-CONFIGURATION command is called.

Privileged functions

Systems support (TSOS privilege), as co-owner, can restore all job variables under their original user IDs.

When a job variable which still exists is overwritten, systems support can explicitly bypass the protection by means of the IGNPROT operand.

Systems support can only log the restoration of job variables via the SECOS component SAT if it has the calls used for deleting files (when overwriting) and for creating an entry in the file catalog logged.

Format

Operation	Operands
RJFSNAP	,JVNAME=<c-string 1..80: filename 1..54 with-wild(80)> / <var: char:80> ,SNAPSET=<integer -52..-1> / *LATEST / *ALL ,SNAPID=<c-string 1..1: name 1..1 with-low> / <var: char 1..1> ,REPLACE=*NO / *YES / <var: enum-of_replace_s: 1> ,IGNPROT=*NO / *YES / <var: enum-of_ignprot_s: 1> ,NUSERID=<c-string 1..8: name 1..8> / <var: char:8> ,NPREFIX=<c-string 1..8: name 1..8> / <var: char:8> ,LIST=*NO / *SYSOUT / *ERRORS-TO-SYSOUT / <var: enum-of_list_s: 1> ,EQUATES=*YES / *NO MF=L
	MF=D,PREFIX= D / <pre>
	MF=E,PARAM=<name 1..27>

	MF=C / M
	,PREFIX= D / <pre>
	,MACID= MAR / <macid>

Operand descriptions

JVNAME

Selects the job variables which are to be restored.

=<c-string 1..80: filename 1..54 with-wild(80)>
Path name of the job variable(s) on the Snapset. Wildcards can be used to specify a set of job variables.

The job variables must satisfy the following requirements:

- They must be cataloged when the Snapset is created.
- The pubset on which they are cataloged must be imported locally.

The catalog and user IDs specified must be unique (i.e. contain no wildcards). Aliases (also partially-qualified aliases) may be specified.

Privileged users (TSOS privilege) can restore job variables of all user IDs.

=<var: char:80>
Only possible with MF=M:
Symbolic address of a memory area of 80 bytes in which the path name or wildcard string for the required job variable(s) is stored.

SNAPSET
This operand may not be specified together with the SNAPID operand.
Specifies the Snapset from which restoration is to take place by means of the relative age.

=<integer -52..-1>
Specifies the Snapset explicitly by means of the relative age. The value -1 specifies the latest Snapset (also corresponds to *LATEST).

=*LATEST
Specifies the latest Snapset.

=*ALL
All Snapsets of the pubset concerned are used as a basis for restoration. Each job variable is restored from the Snapset with the latest status of this job variable, in other words with the latest backup. A job variable which cannot be restored with the latest status is in this case not restorable (i.e. older backup statuses are ignored).

SNAPID
This operand may not be specified together with the SNAPSET operand.
Specifies the Snapset from which restoration is to take place by means of the Snapset ID.

=<c-string 1..1: name 1..1 with-low>
Specifies the Snapset explicitly by means of the Snapset ID. The maximum of 52 Snapsets for a pubset are distinguished by means of Snapset IDs specified which comprise letters from the 26 lowercase letters a to z and the 26 uppercase letters A to Z.

=<var: char 1..1>
Only possible with MF=M:
Symbolic address of a memory area of 1 byte in which the Snapset ID is stored.

Note

If neither SNAPSET nor SNAPID is specified, the latest Snapset is used.

REPLACE
Specifies whether the job variables to be restored may overwrite existing job variables.

=*NO
Existing job variables are not overwritten. This means that job variables with the names of existing job variables are not restored.

=*YES
Existing job variables may be overwritten by job variables which are to be restored provided the protection attributes permit this. For job variables which are protected against unauthorized overwriting by means of a password, the required password must be entered in the caller’s password table (see the ADD-PASSWORD command).

=<var: enum-of_replace_s: 1>
Name of the field with the value for REPLACE.

IGNPROT
This operand is only available to privileged users (TSOS privilege).
Specifies whether job variables are to be overwritten without taking into account any write protection which exists.

=*NO
Write protection is taken into account.

=*YES
Write protection is ignored.

=<var: enum-of_ignprot_s: 1>
Name of the field with the value for IGNPROT.

NUSERID
Specifies that the job variables are to be renamed when they are restored and are to be restored under the specified user ID.
This operand may not be specified together with the NPREFIX operand.

=<c-string 1..8: name 1..8>
User ID.

=<var: char:8>
Only possible with MF=M:
Symbolic address of a memory area of 8 bytes in which the user ID is stored.

NPREFIX=

Specifies that the job variables are to be renamed when they are restored and are to be assigned the specified name prefix.
This operand may not be specified together with the NUSERID operand.

=<c-string 1..8: name 1..8>
Name prefix.

=<var: char:8>
Only possible with MF=M:
Symbolic address of a memory area of 8 bytes in which the name prefix is stored.

LIST
Specifies which processing results are to be logged to SYSOUT.

=*NO
No output is directed to SYSOUT.

=*SYSOUT
All job variables are listed. For job variables which could not be restored, the reason is displayed by means of a message code.

=*ERRORS-TO-SYSOUT
Only job variables which could not be restored are listed. The reason is displayed by means of a message code.

=<var: enum-of_list_s: 1>
Name of the field with the value for LIST.

EQUATES

Control operand; for MF=C and MF=D only:
Specifies whether equates are also to be generated for the values of the fields of the parameter area when the parameter area is expanded.

= *YES
When the parameter area is expanded, equates are also generated for the values of the fields of the parameter area.

= *NO
When the parameter area is expanded, no equates are generated for the values of the fields of the parameter area.

Return codes

The return code is placed in the standard header of the parameter area. The parameter area may then not be located in the read-only area, otherwise the program terminates.

Standard header: `ccbbaaaa`

The following code relating to execution of the RJFSNAP macro is returned in the standard header(cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'cc'	X'bb'	X'aaaa'	Meaning
X'00'	X'00'	X'0000'	No error
X'00'	X'40'	X'0433'	JV not cataloged
X'00'	X'40'	X'0435'	JV cannot be accessed
X'00'	X'40'	X'0440'	JV name invalid
X'00'	X'40'	X'04A0'	JV subsystem cannot be accessed
X'00'	X'40'	X'04B1'	Password not specified
X'00'	X'40'	X'04B6'	Expiration date not yet reached
X'00'	X'40'	X'04B8'	Only read access permitted
X'00'	X'40'	X'04BF'	Access not permitted because of JV protection
X'00'	X'40'	X'0501'	Requested catalog not available
X'00'	X'40'	X'0505'	Error in host communication
X'00'	X'40'	X'0512'	Requested catalog not found
X'00'	X'40'	X'051B'	Requested user ID not on the pubset
X'00'	X'40'	X'051D'	LOGON password different on specified pubset
X'00'	X'20'	X'0531'	Unexpected error during catalog access

X'00'	X'82'	X'053C'	No space in the pubset’s catalog
X'00'	X'20'	X'0584'	Internal error
X'00'	X'82'	X'0594'	Not enough virtual memory
X'02'	X'00'	X'05B6'	Incorrect time conversion in GTIME macro
X'00'	X'20'	X'05C7'	Internal error in DMS
X'00'	X'01'	X'05EE'	Path name too long after completion
X'00'	X'40'	X'05FC'	Specified user ID not on home pubset
X'00'	X'40'	X'0610'	Execution of the function returned a return code for at least one of the selected JV names

X'00'	X'40'	X'0620'	No restorable JV found
X'00'	X'40'	X'0621'	JV already cataloged, restoration not performed
X'00'	X'40'	X'0622'	Snapset not available
X'00'	X'01'	X'0624'	JV name invalid
X'00'	X'40'	X'0682'	JV error when accessing JV
X'00'	X'01'	X'06F7'	Invalid operand value
X'00'	X'01'	X'06FD'	Parameter area invalid or not accessible

The return codes with the maincode X’04xy’ belong to the component JVS. A list which includes the meanings can be output using the JVSERROR macro (see also the “Job Variables” manual [21]).

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller
- the list is not aligned on a word boundary
- the list is write-protected.

Layout of the operand list

Macro expansion with MF=D, and with default values for EQUATES, PREFIX and MACID:

```
         RJFSNAP MF=D
*   PARAMETER AREA
DMAKHDR  FHDR  MF=(C,DMAK),EQUATES=NO
DMAKHDR  DS    0A
DMAKFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
*
DMAKIFID DS    0A              0   INTERFACE IDENTIFIER
DMAKFCTU DS    AL2             0   FUNCTION UNIT NUMBER
DMAKFCT  DS    AL1             2   FUNCTION NUMBER
DMAKFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
*
DMAKRET  DS    0A              4   GENERAL RETURN CODE
DMAKSRET DS    0AL2            4   SUB RETURN CODE
DMAKSR2  DS    AL1             4   SUB RETURN CODE 2
DMAKSR1  DS    AL1             5   SUB RETURN CODE 1
DMAKMRET DS    0AL2            6   MAIN RETURN CODE
DMAKMR2  DS    AL1             6   MAIN RETURN CODE 2
DMAKMR1  DS    AL1             7   MAIN RETURN CODE 1
DMAKFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
*
DMAKJNAM DS    CL80                    JVNAME
DMAKSNAP DS    FL1                     SNAPSET
*   SNAPSET - VALUES
DMAKSNIN EQU   0                       SNAPSET=<integer>
DMAKSNCH EQU   1                       SNAPSET=<char>
DMAKSNLT EQU   2                       SNAPSET=*LATEST
DMAKSNAL EQU   3                       SNAPSET=*ALL
*
DMAKREPL DS    FL1                     REPLACE
*   REPLACE - VALUES
DMAKREPY EQU   0                       REPLACE = YES
DMAKREPN EQU   1                       REPLACE = NO
*
DMAKIGNP  DS    FL1                    IGNPROT
*   IGNPROT VALUES
DMAKIGNO  EQU   0                      IGNPROT = NO
DMAKIGYE  EQU   1                      IGNPROT = YES
*
DMAKLIST  DS    FL1                    LIST
*   LIST - VALUES
DMAKLSTN  EQU   0                      LIST = NO
DMAKLSYO  EQU   1                      LIST = SYSOUT
DMAKLSYE  EQU   2                      LIST = ERRORS
*
DMAKNUSR  DS    CL8                    NUSERID
DMAKNPRE  DS    CL8                    NPREFIX
DMAKSNVL  DS    H                      SnapValue
DMAKSNID  DS    CL1                    Snapid
DMAKOFLG  DS    AL1                    FLAG BYTE
DMAKNUSP  EQU   X'80'                  S: NUSERID SPECIFIED
DMAKNPSP  EQU   X'40'                  S: NPREFIX SPECIFIED
DMAKRES1  EQU   X'3F'                  RESERVED
DMAK#     EQU   *-DMAKHDR
```

Sample calling sequence

```
         MVC   RJFSMFC(DMAK#),RJFSMFL
         RJFSNAP MF=M,PATHNAM=':X:JV1',PARAM=RJFSMFC
         RJFSNAP MF=E,PARAM=RJFSMFC
             .
             .
RJFSMFC  RJFSNAP MF=C
RJFSMFL  RJFSNAP MF=L,...
```

SETL - Position file pointer

ISAM:	Macro type:	R for PARMOD=24 0 for PARMOD=31
SAM:	Macro type:	R for PARMOD=24 0 for PARMOD=31

ISAM:

The SETL macro positions the pointer to the beginning or end of a file or, by specifying the record key, to any record within the file.

If SETL KEY is used to position within a file via the primary key and if the file contains records with duplicate primary key values (DUPEKY=YES), the file is positioned to the first of these records.

If SETL KEY is used to position within a file via a secondary key and if the file contains records with duplicate values for this secondary key, the file is positioned to the record whose primary key is indicated by the first pointer in the associated secondary index block.

SAM:

SETL sets the block and record pointer to the position (retrieval address) specified by the user.

The SETL macro sets the position of the internal record pointer, thus enabling the user to define the starting point for subsequent processing of the file.

The retrieval address in the 31-bit TU FCB is updated such that it will be correct after a subsequent GET or PUT macro; in the 24-bit TU FCB, the retrieval address is not changed.

Since the positioning information in the 24-bit TU FCB is 1 byte long, a buffer must not contain more than 255 records.

An invalid SETL operand causes control to be passed to the address USERERR of the EXLST macro.

Format

Operation	Operands
SETL	**fcbadr / (1)** **,B / E / R / KEY / (0)** **[,AIX = NO /** **YES,KEYNAME = name** **/** **YES,KEYNMAD = adr]** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed.
For ISAM only:
If the file is to be positioned with the aid of a secondary key, the 31-bit interface of this FCB must be available.

(1)
The FCB address is stored in register 1.

B

The pointer is to be moved to the beginning of the file.
For ISAM only:
Using SETL B for a null file will cause control to be passed to the EOFADDR error exit of the EXLST macro.
For SAM only:
For multivolume files, the pointer is positioned to the first record on the current tape.

E

The pointer is to be moved to the end of the file.
For SAM only:
For multivolume files, the pointer is positioned to the last record of the current tape, which means that the next GET macro will initiate a tape swap.
With OPEN OUTPUT/EXTEND, specifying “E” will cause control to be passed to the USERERR exit of the EXLST macro.

R

For SAM only:
The positioning information is to be taken from the retrieval address (not permitted for tape files with nonstandard blocks processed with PARMOD=24). For multivolume files, the retrieval address applies to the current volume, not to the file.

KEY

For ISAM only:
The pointer is to be positioned to the primary or secondary key value stored in the field specified in the KEYARG operand of the FCB macro.

If a SETL ...,KEY points to an existing key, a subsequent GET or GETR will read this record.
If SETL ...,KEY points to a non-existent record, a subsequent GET will read the record with the next higher key and a GETR will read the record with the next lower key.

(0)
Register 0 contains a “positioning code”. Before the SETL macro is executed, the positioning code must be loaded into register 0 as follows:

SETL operand	Contents of register 0	Effect
B	0	Position to beginning-of-file

E	1	Position to end-of-file
R	2	*For SAM only:* position according to retrieval address
KEY	Address of KEYARG	Position to specific record

If register 0 contains a value other than 0 or 1, this value is always interpreted as the KEYARG address.

AIX

For ISAM only:
Specifies whether the pointer is to be positioned to a record via its primary key or via a secondary key.

= NO
The pointer is positioned to the record via its primary key (default value).

= YES
This may only be specified if

- the 31-bit interface of the macro is generated (via the operand PARMOD=31 or the macro GPARMOD 31) and
- the macro refers to a 31-bit FCB.

The pointer is positioned to the record via the secondary key specified in the KEYNAME or KEYNMAD operand.

KEYNAME = name

For ISAM only:
Specifies the name of the secondary key via which the pointer is to be positioned to the record.
“name” must be the name of a secondary key defined for the current file. The names of all secondary keys defined for a file can be determined by means of the SHOWAIX macro or the SHOW-INDEX-ATTRIBUTES command.

AIX=YES must be specified.

KEYNMAD = addr

For ISAM only:
Specifies the symbolic address (the name) of a field in which the user has stored the name of the secondary key via which the pointer is to be positioned to the record.
When the macro is executed, the field with the symbolic address “addr” must contain the name of a secondary key defined for the current file.

AIX=YES must be specified.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The macro is generated with the expansion for the 24-bit interface. The object code generated can run only in the 16-Mb address space (24-bit addressing).

= 31
The macro is generated such that it is independent of the addressing mode (24-bit or 31-bit addressing). The object code generated can run in the 2-Gb address space.

Programming notes

1. The SETL macro overwrites the contents of registers 0, 1, 14 and 15.
2. A SETL macro always results in an SVC.

SHOPLNK - Return information on ISAM pool link names

Macro type: type S (E form/L form/D form/C form/M form); see "Macro types"

The SHOPLNK macro provides information on the assignment of ISAM pools to ISAM pool link names. When using the macro, the user can specify either the pool link name or the name of the ISAM pool.
An ISAM pool link name can be assigned to any ISAM pool where there is a link to that ISAM pool, irrespective of the scope of the ISAM pool or of the host computer on which the ISAM pool is located.

Format

Operation	Operands
SHOPLNK	**[,PARAM = adr]** **[,LINK = *ALL / 'name' / adr / (r)]** **[,NAME = *ALL / 'name' / adr / (r)]** **[,CATID = 'name' / adr / (r)]** **[,SCOPE = *TASK / *USERID / *USERGROUP / *HOST / adr / (r)]** **[,AREA = name / (r)]** **[,SIZE = zahl / adr / (r) / *equ]** **[,XPAND = PARAM / DESCHDR / LINKDESC]** **MF = L / M**
	MF = E,PARAM = adr / (r)]
	MF = D[,PREFIX = pre]
	MF = C **[,PREFIX = pre]** **[,MACID = macid]**

Operand descriptions

AREA

Specifies an output area to which the output list is to be transferred. There are no restrictions on the character set and length.Only the operand value addr is allowed for the MF=L form.

Default value: X'FFFFFFFF'

= addr
Symbolic address of a field containing the name of the output area.

= (r)
Register containing the address of a field in which the name of the output area is stored.

CATID

Specifies the catalog ID of the PVS under which the ISAM pool (for which the pool link name is to be shown) was created (of no consequence if the parameter NAME=*ALL is specified).Only the operand value “name” is allowed for the MF=L form.

Default value: The default PVS ID of the job (DEFAULT-PUBSET in the user catalog entry) or the home PVS ID (depending on the setting of the system parameter “ISPLDEFC”).

= 'name'
Name of a field containing the catalog ID of the pubset; four characters in length.

= addr
Symbolic address of a field with the catalog ID of the pubset.

= (r)
Register containing the address of a field with the catalog ID of the pubset.

LINK = *ALL

Returns the assignments of all pool link names defined by the caller to ISAM pools.

= name
Specifies the ISAM pool link name for which the associated ISAM pool is to be shown.

= addr
Symbolic address of a field containing the link name.

= (r)
Register containing the address of the link name.

MACID

Evaluated only in conjunction with MF=C; defines the second through fourth characters of the field names and equates generated in the data area when the macro is expanded.

Default value: MACID = ISL

= macid
Three-character string defining the second through fourth characters of each field name and equate generated.

MF

The forms of the MF operand are described in detail in the appendix (''Macro types'').

NAME

Specifies the name under which the ISAM pool (for which the pool link name is to be shown) was created. The desired ISAM pool is uniquely identified by the specified name, the catalog ID (CATID) and the scope (see the SCOPE operand).
Only the operand values 'name'/*ALL can be specified for the MF=L form.

= *ALL
Requests the pool link names for all ISAM pools created by the caller.

= 'name'
Name of the ISAM pool for which pool link names are to be shown.

= addr
Symbolic address of a field containing the name of the ISAM pool or *ALL.

= (r)
Register containing the address of a field in which the name of the ISAM pool or *ALL is held.

SCOPE

Scope of the specified ISAM pool for which information is to be shown. If NAME=*ALL was specified, the SCOPE operand specification is ignored.
Only the operand values *TASK/*USERID/*HOST are allowed for the MF=L form.

= *TASK
Requests pool link names for the task-local ISAM pool with the specified name.

= *USERID

= *USERGROUP
SCOPE=USERID and SCOPE=USERGROUP which were available up to BS2000/OSD V6.0A are still accepted for reasons of compatibility, but internnaly they are mapped to SCOPE=HOST (cross-task ISAM pool).
However, in each case only the ISAM pools which were created with the specified scope are displayed.

= *HOST
Requests pool link names for the cross-task ISAM pool with the specified name.

= addr
4-byte symbolic address of a field containing the scope of the ISAM pool.

= (r)
Register containing the address at which the scope of the ISAM pool is stored.

SIZE

Specifies the length of the output area.Only the operand values nmbr/*equ are allowed for the MF=L form.

Default value: X'00000000'

= nmbr
Numeric value that specifies the length of the output area.100 <= nmbr <= 10000.

= addr
4-byte symbolic address for the length of the output area.

= (r)
Register containing the length of the output area.

= *equ
Equate with the length of the output area.

XPAND

Control operand; for MF=C and MF=D only:
Defines which structure is to be expanded (i.e. generated). This operand is ignored for other MF values.

= PARAM
Expands the layout of the parameter list.

= DESCHDR
Expands the layout of the header of the output area.

= LINKDESC
Expands the layout of the pool link descriptor.

Return codes

The return codes are placed in the header of the parameter list (standard header):

- The main return code, in a half-word with the name DISLMRET.
- Subcode1, in a byte with the name DISLSR1.
  Subcode1 describes error classes which allow the caller to respond to similar error situations.

  The caller can refer back to the main code as well as to subcode1; however, the evaluation of subcode1 must be given preference, because when main codes are expanded for a macro, evaluations which are exclusively in response to error classes are not taken into account.
- Subcode2 always has the value X'00'.

If return codes cannot be placed in the header of a macro (because it is not accessible, for example), the calling program is terminated with an appropriate error message.

If the return code for an “internal system error on calling a system function” is generated, the field DISL.SYCD in the parameter list of the macro in question will contain a more detailed code to enable diagnostics (see the inserts with the corresponding message for values).

A macro called with MF=D or MF=C generates EQU instructions for the return codes in addition to the field names.

The following overview shows the return codes for the SHOPLNK macro in tabular form. The string DISL must be added to the left of the indicated names for EQU instructions. This string can be modified by using the parameter PREFIX=prefix or MACID=macid (see the operand descriptions).

Standard header: `ccbbaaaa`

The following code relating to execution of the SHOPLNK macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'bb'	X'aaaa'	Meaning
X'00'	X'0000'	The function was executed successfully.
X'01'	X'FFFF'	The header is corrupted, e.g. not correctly initialized. No repetition is possible.
X'02'	X'FFFF'	Linkage error (function not available) The called function is not available (e.g. NK-ISAM is not loaded).

X'03'	X'FFFF'	Linkage error (version not supported) The version specified in the header is not supported (mount error).
X'01'	X'0001'	Parameter list not accessible. No repetition possible.
X'01'	X'0002'	Parameter error. No repetition possible.

X'20'	X'0005'	Internal system error on calling a system function. No repetition possible.
X'40'	X'0008'	The specified ISAM pool link name does not exist. For errors not in error class B, C, E.
X'40'	X'0009'	The caller has not defined an ISAM pool link name. For errors not in error class B, C, E.
X'82'	X'000B'	No virtual memory available. Wait and repeat operation.

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller
- the list is not aligned on a word boundary
- the list is write-protected.

Layout of the operand list

Macro expansion with MF=D and default values for PREFIX and MACID:

```
SHOPLNK MF=D
1          STACK  PRINT
1          PRINT  NOGEN
2                *,##### PREFIX=D, MACID=ISL #####
1          #INTF INTNAME=SHOPLNK,REFTYPE=REQUEST,INTCOMP=002
1 DISLPLA  DS    0F                BEGIN of PARAMETERAREA     _INOUT
1          FHDR  MF=(C,DISL),EQUATES=YES
2          DS    0A
2 DISLFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 DISLIFID DS    0A              0   INTERFACE IDENTIFIER
2 DISLFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 DISLFCT  DS    AL1             2   FUNCTION NUMBER
2 DISLFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 DISLRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 DISLSRET DS    0AL2            4   SUB RETURN CODE
2 DISLSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 DISLR2OK EQU   X'00'               All correct, no additional info
2 DISLR2NA EQU   X'01'               Successful, no action was necessary
2 DISLR2WA EQU   X'02'               Warning, particular situation
```

```
2 DISLSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 DISLRFSP EQU   X'00'              FUNCTION SUCCESSFULLY PROCESSED
2 DISLRPER EQU   X'01'              PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 DISLRFNS EQU   X'01'              CALLED FUNCTION NOT SUPPORTED
2 DISLRFNA EQU   X'02'              CALLED FUNCTION NOT AVAILABLE
2 DISLRVNA EQU   X'03'              INTERFACE VERSION NOT SUPPORTED
2 *
2 DISLRAER EQU   X'04'              ALIGNMENT ERROR
2 DISLRIER EQU   X'20'              INTERNAL ERROR
2 DISLRCAR EQU   X'40'              CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 DISLRECR EQU   X'41'              SUBSYSTEM (SS) MUST BE CREATED
2 *                                 EXPLICITELY BY CREATE-SS
2 DISLRECN EQU   X'42'              SS MUST BE EXPLICITELY CONNECTED
2 *
2 DISLRWAR EQU   X'80'              WAIT FOR A SHORT TIME AND RETRY
2 DISLRWLR EQU   X'81'                   "     LONG       "
2 DISLRWUR EQU   X'82'              WAIT TIME IS UNCALCULABLY LONG
2 *                                 BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 DISLRTNA EQU   X'81'              SS TEMPORARILY NOT AVAILABLE
2 DISLRDH  EQU   X'82'              SS IN DELETE / HOLD
2 *
2 DISLMRET DS    0AL2            6   MAIN RETURN CODE
2 DISLMR2  DS    AL1             6   MAIN RETURN CODE 2
2 DISLMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 DISLRLNK EQU   X'FFFF'             LINKAGE ERROR / REQ. NOT PROCESSED
2 DISLFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 DISLOK   EQU   X'0000'           FUNCTION SUCCESSFUL PROCESSED
1 DISLNPAR EQU   X'0001'           PARAMETERLIST NOT ACCESSIBLE
1 DISLPERR EQU   X'0002'           PARAMETER ERROR
1 DISLSYSE EQU   X'0005'           INTERNAL SYSTEM ERROR
1 DISLLLNE EQU   X'0008'           SPECIFIED POOL LINK NAME NOT FOUND
1 DISLNOLI EQU   X'0009'           NO ISAM POOL LINK EXISTING
1 DISLNOSP EQU   X'000B'           NO MEMORY AVAILABLE
1 DISLNUGR EQU   X'000C'           NO USEGROUP DEFINED
1 DISLINOP EQU   X'001F'           SSTA INOP
1 DISLSSER EQU   X'0020'           SSTA INTERNAL ERROR
1 DISLMEMR EQU   X'0021'           SSTA MEMORY ERROR
1 DISLOPSR EQU   X'0022'           SSTA OPS ERROR
1 DISLOPME EQU   X'0023'           SSTA OPS MEMORY ERROR
1 *
1 *
1 DISLPLNK DS    CL8               LINK-NAME
1 DISLPNAM DS    CL8               POOL-NAME
```

```
1 DISLCID  DS    CL4              CATALOG-IDENTIFIER
1 *
1 DISLSCOP DS    XL1              SCOPE
1 DISLTASK EQU   X'00'                 = TASK
1 DISLUSID EQU   X'01'                 = USERID
1 DISLHOST EQU   X'02'                 = HOST-SYSTEM
1 DISLUSGR EQU   X'03'                 = USERGROUP
1 *
1 DISLSYCD DS    XL1              SYSTEM-ERROR-CODE
1 *
1 DISLADDR DS    A                ADDRESS OF OUTPUT-AREA
1 DISLSIZE DS    F                SIZE OF OUTPUT-AREA
1 *
1 DISL#    EQU   *-DISLPLA        LENGTH of PARAMETERAREA
```

Format of the output area

The SHOPLNK macro returns the assignments of ISAM pools to ISAM pool names in the output area specified by the caller. This area begins with a management header of 16 bytes containing the following information:
The management header of the output area is generated by specifying the control parameter XPAND=DESCHDR when calling the SHOPLNK macro with the control operand MF=D or MF=C:

```
1 DISLADMH  DS    0F               MANAGEMENT HEADER LAYOUT
1 DISLLLG   DS    F                NUMBER OF BYTES TRANSFERRED
1 DISLLCLG  DS    F                LENGTH OF TOTAL INFORMATION
1 DISLL#LN  DS    H                NUMBER OF LINK NAMES INVOLVED
1 DISLLIND  DS    XL1              TRANSFER INDICATOR
1 DISLLCOM  EQU   X'00'            INFORMATION COMPLETE
1 DISLLPAR  EQU   X'01'            INFORMATION INCOMPLETE
1 DISLLRES  DS    CL5              NOT USED
1 DISLLLEN  EQU   *-DISLADMH       LENGTH OF MANAGEMENT HEADER
```

The entry with information can be expanded using the parameter XPAND=LINKDESC when calling the SHOPLNK macro with the control operand MF=C or MF=D:

```
1 DISLLDDS  DS    0F              POOL LINK DESCRIPTOR LAYOUT
1 DISLLNAM  DS    CL8             NAME OF POOL LINK
1 DISLPONA  DS    CL8             NAME OF ASSOCIATED ISAM POOL
1 DISLLCID  DS    CL4             NAME OF ASSOCIATED PVS
1 DISLLSCO  DS    XL1             SCOPE OF ISAM POOL
1 DISLLTSK  EQU   X'00'           SCOPE = TASK
1 DISLLUSR  EQU   X'01'           SCOPE = USERID
1 DISLLHOS  EQU   X'02'           SCOPE = HOST
1 DISLLUGR  EQU   X'03'           SCOPE = USERGROUP
1 DISLLUID  DS    CL8             USERID FOR SCOPE = *USERID
1 *                               GROUP NAME WITH SCOPE = *USERGROUP
1 DISLLRSV  DS    CL3             NOT USED
1 DISLLLNG  EQU   *-DISLLDDS      LENGTH OF ISAM-POOL-DESCR.
```

Explanation of the individual fields in the output area

Management header

The management header (16 bytes in length) contains the following information:

- the number of bytes that were transferred to the output area (length: 4 bytes),
- the total number of bytes contained in the complete information (length: 4 bytes). This value is significant for the caller of the macro if the output area is too small. The caller of the macro must then supply an output area of (at least) this length,
- the number of ISAM pool link names for which the assignments to ISAM pools was shown in the output area (length: 2 bytes),
- an indicator, with a length of 1 byte, which specifies whether the information transferred to the output area is complete or whether only a part of the desired information could be returned because the output area was too small. The following values are assigned to this indicator:

  X'00': The information is complete.

  X'01': The information was truncated and is incomplete.

  In the latter case, the caller must provide a larger output area in order to accommodate the entire information. The length of this larger area can then be obtained from the second word of the management header,
- the remaining 5 bytes are not used.

Descriptor area

In order to identify an ISAM pool uniquely, the name of the ISAM pool, the catalog ID of the host computer on which the pool exists, and the scope of the ISAM pool are required. The output area for the SHOPLNK macro contains 32-byte entries comprising an ISAM pool link name at the beginning, followed by an ISAM pool descriptor with the above information. In other words, the following information is included in such entries:

- the name of an ISAM pool link name with a length of 8 bytes (printable),
- the name of the associated ISAM pool with a length of 8 bytes (printable),
- the catalog ID (CATID) of the host computer on which the ISAM pool in question exists; also printable, and with a length of 4 bytes,
- the scope of the ISAM pool involved, with a length of 1 byte for which values are assigned as follows:

  X'00': SCOPE = *TASK

  X'01': SCOPE = *USERID

  X'02': SCOPE = *HOST

  X'03': SCOPE = *USERGROUP
- the user ID – if the associated ISAM pool has been provided with the attribute SCOPE = USERID,
- the remaining 3 bytes are not used.

The above-named structures are laid out in the output area filled by the SHOPLNK macro as follows:

```
HEADER |
LINKNAME1 | POOLNAME1 | CATID1 | SCOPE1 | userid | UNUSED
LINKNAME2 | POOLNAME2 | CATID2 | SCOPE2 | userid | UNUSED
.......
.......
LINKNAMEn | POOLNAMEn | CATIDn | SCOPEn | userid | UNUSED
```

The “userid” field contains blanks if SCOPE=TASK or SCOPE=HOST applies.

The links between ISAM pools and pool link names are described after the management header. The number of entries created is contained in the management header itself.

SHOPOOL - Return information on ISAM pools

Macro type: type S (E form/L form/D form/C form/M form); see "Macro types"

The SHOPOOL macro returns information on ISAM pools linked to the current job, taking the links to ISAM pools on remote systems into account (except with the SELECT operand). The user may request information on a specific ISAM pool or on all ISAM pools to which a link exists.
The output comprises detailed information on the pool-specific attributes of each ISAM pool (as defined in the CREPOOL macro).
If desired, the user may also have the task sequence numbers (TSNs) of all connected jobs for each ISAM pool displayed.
While the user can set up a connection to an ISAM pool with the CREPOOL macro, a job can also be implicitly connected to standard pools by NK-ISAM.

Note

Cross-task ISAM pools are created automatically in a data space on a file-specific basis when the file is opened. DSCOPE=USERID and SCOPE=USERGROUP which were available up to BS2000/OSD V6.0A are still accepted for reasons of compatibility, but internnaly they are mapped to SCOPE=HOST (cross-task ISAM pool). For further information on NK-ISAM pools in data spaces please refer to the “Introductory Guide to DMS” [1].

Format

Operation	Operands
SHOPOOL	**[,PARAM = adr]** **[,NAME = 'name' / *ALL / adr / (r)]** **[,CATID = 'name' / adr / (r)]** **[,SCOPE = *TASK / *USERID / *USERGROUP / *HOST / adr / (r)]** **[,SELECT = *OWN / *ALL / adr / (r)]** **[,INFO = *ATTR / *ALL / adr / (r)]** **[,AREA = adr / (r)]** **[,SIZE = zahl / adr / (r) / *equ]** **[,XPAND = PARAM / DESCHDR / POOLDESC]** **MF = L / M**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = pre]
	MF = C[,PREFIX = pre][,MACID = macid]

Operand descriptions

AREA

Specifies an output area to which the output list is to be transferred.
Only the operand value addr is allowed for the MF=L form.

Default value: X'FFFFFFFF'

= addr
Symbolic address (name) of the output area. There are no restrictions on the character set and length.

= (r)
Register in which the address of the output area is entered.

CATID

Catalog ID of the pubset to which the ISAM pool (for which information is to be obtained) is assigned (of no consequence if the parameter NAME=*ALL is specified).
Only the operand value 'name' is allowed for the MF=L form.

Default value: The ISAM pool is assigned to the catalog that was set with the system parameter ISPLDEFC (ISAM-POOL-DEFAULT-CATID):

X'00': default catalog ID from the user entry (DEFAULT-PUBSET)

X'01': catalog ID of the home pubset

= 'name'
Catalog ID of the home pubset; 4 characters in length.

= addr
4-byte address of a field containing the catalog ID of the pubset.

= (r)
Register containing the address of a field with the catalog ID of the pubset.

INFO

Defines the scope of the information to be output. The SHOPOOL macro returns the requested information in the output area specified by the caller (see the format of the output area on "SHOPOOL - Return information on ISAM pools").

= *ATTR
Outputs the static attributes for the ISAM pool specified in NAME.

= *ALL
Outputs the job numbers (TSNs) of all jobs connected to the specified ISAM pool/s in addition to the static attributes.

= addr
Address of a field containing the scope of the information to be output.

= (r)
Register containing the address of a field in which the scope of the information to be output is stored.

MACID
Evaluated only in conjunction with MF=C; defines the second through fourth characters of the field names and equates generated in the data area when the macro is expanded.

Default value: MACID = ISP

= macid
Three-character string defining the second through fourth characters of each field name and equate generated.

MF
The forms of the MF operand are described in detail in the appendix on "Macro types".

NAME
Name of the ISAM pool for which information is to be returned. The desired ISAM pool is uniquely identified by the specified pool name, the catalog identifier (“catid”) and the scope.Information is output only if the ISAM pool exists and the current job is connected to it.Only the operand values “name” and “*ALL” can be specified with the MF=L form.

= *ALL
Returns information on all ISAM pools to which the current job is connected.

= 'name'
Name of the ISAM pool for which information is to be output. 'name' may have a length of up to 8 characters.

= addr
Address of a field containing the name of the ISAM pool or *ALL.

= (r)
Register containing an address at which the name of the ISAM pool or *ALL is stored.

PARAM
Indicates the address of the operand list. This operand is only evaluated in conjunction with MF=E (see also "Macro types").

= addr
Symbolic address (name) of the operand list.

= (r)
Number of the register containing the address of the operand list. The register must be loaded with this address value before the macro is called.

PREFIX

Evaluated only in conjunction with MF=C or MF=D; defines the first character of each field name and equate generated in the data area when the macro is expanded.

Default value: PREFIX = D

= pre
“pre” is a one-character prefix with which the field names and equates generated by the assembler are to begin.

SCOPE

Scope of the specified ISAM pool for which information is to be returned. If NAME=*ALL is specified, entries in the SCOPE operand are ignored.
Only the operand values *TASK/*USER-ID/*HOST are allowed for the MF=L form.

= *TASK
Returns information on the cross-task ISAM pool with the specified name.

= *USERID
= *USERGROUP
SCOPE=USERID and SCOPE=USERGROUP which were available up to BS2000/OSD V6.0A are still accepted for reasons of compatibility, but internnaly they are mapped to SCOPE=HOST (cross-task ISAM pool).
However, in each case only the ISAM pools which were created with the specified scope are displayed.

= *HOST
Returns information on the cross-task ISAM pool with the specified name.

= addr
Symbolic name of a field containing the scope of the specified ISAM pool.

= (r)
Register containing the address of a field in which the scope of the ISAM pool is stored.

SELECT

Specifies the criteria by which the ISAM pools specified via the NAME parameter are selected for the output of information.

= *OWN
Returns information on ISAM pools to which the calling task is connected. ISAM pools on remote systems are also shown in the output.

= *ALL
This operand value can only be specified by a task that has TSOS or SW-MONITOR-ADMINISTRATION privileges.
Information is returned on all existing ISAM pools, even if no connection to the ISAM pool exists. Remote ISAM pools are not shown.

= addr
Symbolic name of a field containing the criteria for selecting the ISAM pools for which information is to be output.

= (r)
Register with the address of a field containing the criteria for selecting the ISAM pools for which information is to be output.

SIZE

Specifies the length of the output area.

Only the operand values nmbr/*equ are allowed for the MF=L form.

Default value: X'00000000'

= nmbr
Numeric value that specifies the length of the output area.
100 <= nmbr <= 10000.

= addr
Is the address for the length of the output area.

= *equ
Equate containing the length of the output area.

XPAND

Control operand; for MF=C and MF=D only:
Defines which structure is to be expanded (i.e. generated). This operand is ignored for other MF values.

= PARAM
Expands the layout of the parameter list.

= DESCHDR
Expands the layout of the header of the output area.

= POOLDESC
Expands the layout of a pool descriptor.

Return codes

The return codes are placed in the header of each parameter list:

- The main return code, in a half-word with the name DISPMRET.
- Subcode1, in a byte with the name DISPSR1.
  Subcode1 describes error classes which allow the caller to respond to similar error situations.

  The caller can refer back to the main code as well as to subcode1; however, the evaluation of subcode1 must be given preference, because evaluations which are exclusively in response to error classes are not taken into account when main codes are expanded for a macro.
- Subcode2 always has the value X'00'.

If return codes cannot be placed in the header of a macro (because it is not accessible, for example), the calling program is terminated with an appropriate error message.

If the return code for an "'internal system error on calling a system function" is generated, the field DISPSYCD in the parameter list of the macro in question will contain a more detailed code to enable diagnostics (see the inserts with the corresponding message for values).

A macro called with MF=D or MF=C generates EQU instructions for the return codes in addition to the field names.

The following overview shows the return codes for the SHOPOOL macro in tabular form. The string DISP must be added to the left of the indicated names for EQU instructions. This string can be modified using the parameter PREFIX=prefix or MACID=macid (see the operand descriptions).

Standard header: `ccbbaaaa`

The following code relating to execution of the SHOPOOL macro is returned in the standard header (cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'bb'	**X'aaaa'**	**Meaning**
X'00'	X'0000'	The function was executed successfully
X'01'	X'FFFF'	The header is corrupted, e.g. not correctly initialized. No repetition is possible
X'02'	X'FFFF'	Linkage error (function not available) The called function is not available
X'03'	X'FFFF'	Linkage error (version not supported) The version specified in the header is not supported (mount error)
X'01'	X'0001'	Parameter list not accessible. No repetition possible
X'01'	X'0002'	Parameter error. No repetition possible

X'40'	X'0003'	The specified catalog ID name does not exist Error class not equal to B, C, E
X'40'	X'0004'	The specified ISAM pool link name does not exist Error class not equal to B, C, E
X'20'	X'0005'	Internal system error on calling a system function No repetition possible
X'40'	X'0006'	No ISAM pool exists Error class not equal to B, C, E
X'40'	X'0007'	Privileges required for function not available Error class not equal to B, C, E
X'82'	X'000A'	Specified catalog ID does not exist. Wait and repeat.
X'82'	X'000B'	No virtual memory available. Wait and repeat operation.

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).

The calling program is terminated if one of the following errors occurs with respect to the parameter list:

- the list is not assigned to the caller
- the list is not aligned on a word boundary
- the list is write-protected.

Layout of the parameter list

Macro expansion with MF=D and default values for PREFIX and MACID:

The parameter list of the macro contains a header, whose fields are loaded automatically when the list is created with the L form.

If a parameter list is to be created dynamically with the D or C form, it must be initialized beforehand with a parameter list created with the L form. This is the only way of ensuring that the header of a parameter list contains the correct information.

```
SHOPOOL MF=D
1          STACK  PRINT
1          PRINT  NOGEN
2                 *,##### PREFIX=D, MACID=ISP #####
1          #INTF INTNAME=SHOPOOL,REFTYPE=REQUEST,INTCOMP=002
1 DISPPPA  DS    0F                BEGIN of PARAMETERAREA     _INOUT
1          FHDR  MF=(C,DISP),EQUATES=YES
2          DS    0A
2 DISPFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 DISPIFID DS    0A              0   INTERFACE IDENTIFIER
2 DISPFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
```

```
2 *                                MUST BE RESET UNTIL FURTHER NOTICE
2 *                                BIT 14-12 UNUSED, MUST BE RESET
2 *                                BIT 11-0  REAL FUNCTION UNIT NUMBER
2 DISPFCT  DS    AL1             2   FUNCTION NUMBER
2 DISPFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 DISPRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 DISPSRET DS    0AL2            4   SUB RETURN CODE
2 DISPSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 DISPR2OK EQU   X'00'               All correct, no additional info
2 DISPR2NA EQU   X'01'               Successful, no action was necessary
2 DISPR2WA EQU   X'02'               Warning, particular situation
2 DISPSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 DISPRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
2 DISPRPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 DISPRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
2 DISPRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 DISPRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 DISPRAER EQU   X'04'               ALIGNMENT ERROR
2 DISPRIER EQU   X'20'               INTERNAL ERROR
2 DISPRCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 DISPRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 DISPRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 DISPRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 DISPRWLR EQU   X'81'                    "     LONG       "
2 DISPRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 DISPRTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
2 DISPRDH  EQU   X'82'               SS IN DELETE / HOLD
2 *
2 DISPMRET DS    0AL2            6   MAIN RETURN CODE
2 DISPMR2  DS    AL1             6   MAIN RETURN CODE 2
2 DISPMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 DISPRLNK EQU   X'FFFF'              LINKAGE ERROR / REQ. NOT PROCESSED
2 DISPFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
```

```
2 *
1 DISPOK   EQU   X'0000'           FUNCTION SUCCESSFUL PROCESSED
1 DISPNPAR EQU   X'0001'           PARAMETERLIST NOT ACCESSIBLE
1 DISPPERR EQU   X'0002'           PARAMETER ERROR
1 DISPNCAT EQU   X'0003'           CATID NOT KNOWN
1 DISPPLNE EQU   X'0004'           SPECIFIED ISAM POOL NOT FOUND
1 DISPSYSE EQU   X'0005'           INTERNAL SYSTEM ERROR
1 DISPNOPL EQU   X'0006'           NO ISAM POOL EXISTING
1 DISPNAUT EQU   X'0007'           NO AUTHORIZATION FOR FUNCTION
1 DISPNACC EQU   X'000A'           CATID NOT AVAILABLE
1 DISPNOSP EQU   X'000B'           NO MEMORY AVAILABLE
1 DISPNUGR EQU   X'000C'           NO USERGROUP DEFINED
1 DISPINOP EQU   X'001F'           SSTA INOP
1 DISPSSER EQU   X'0020'           SSTA INTERNAL ERROR
1 DISPMEMR EQU   X'0021'           SSTA MEMORY ERROR
1 DISPOPSR EQU   X'0022'           SSTA OPS ERROR
1 DISPOPME EQU   X'0023'           SSTA OPS MEMORY ERROR
1 *
1 *
1 DISPPNAM DS    CL8               POOL-NAME
1 DISPCID  DS    CL4               CATALOG-IDENTIFIER
1 *
1 DISPSCOP DS    XL1               SCOPE
1 DISPTASK EQU   X'00'                   = TASK
1 DISPUSID EQU   X'01'                   = USERID
1 DISPHOST EQU   X'02'                   = HOST-SYSTEM
1 DISPUSGR EQU   X'03'                   = USERGROUP
1 *
1 DISPSELC DS    XL1               SELECT
1 DISPOWN  EQU   X'00'                   = OWN
1 DISPALLH EQU   X'01'                   = ALL (HOST)
1 *
1 DISPINFO DS    XL1               INFO
1 DISPATTR EQU   X'00'                   = ATTR
1 DISPALLT EQU   X'01'                   = ALL
1 *
1 DISPSYCD DS    XL1               SYS-ERROR-CODE
1 *
1 DISPADDR DS    A                 ADDRESS OF OUTPUT-AREA
1 DISPSIZE DS    F                 SIZE OF OUTPUT-AREA
1 *
1 DISP#    EQU   *-DISPPPA         LENGTH of PARAMETERAREA
```

Format of the output area

The SHOPOOL macro returns the requested information in the output area specified by the caller.

Management header of the output area

This area begins with a management header with a length of 16 bytes. The management header of the output area is generated by specifying the operand XPAND=DESCHDR when calling the SHOPOOL macro with the control operand MF=D or MF=C:

```
1 DISPADMH  DS    0F              MANAGEMENT HEADER LAYOUT
1 DISPPLG   DS    F               NUMBER OF BYTES TRANSFERRED
1 DISPPCLG  DS    F               LENGTH OF COMPLETE INFORMATION
1 DISPP#PO  DS    H               NUMBER OF ISAM POOLS INVOLVED
1 DISPPINF  DS    XL1             INDICATOR FOR SCOPE OF INFO
1 DISPPATT  EQU   X'00'           INFO = *ATTR
1 DISPPALL  EQU   X'01'           INFO = *ALL
1 DISPPIND  DS    XL1             TRANSFER INDICATOR
1 DISPPCOM  EQU   X'00'           INFORMATION COMPLETE
1 DISPPPAR  EQU   X'01'           INFORMATION INCOMPLETE
1 DISPPRES  DS    CL4             NOT USED
1 DISPPLEN  EQU   *-DISPADMH      LENGTH OF HEADER
```

Specified output area

An ISAM pool is described in the output area specified by the caller by means of a so-called ISAM pool descriptor with a length of 32 bytes. ISAM pool descriptors are generated by specifying the control parameter XPAND=DESCHDR when calling the SHOPOOL macro with the control operand MF=C or MF=D:

```
1 DISPPDDS DS    0F              ISAM POOL DESCRIPTOR LAYOUT
1 DISPNAME DS    CL8             NAME OF ISAM POOL
1 DISPPCID DS    CL4             NAME OF ALLOCATED PVS
1 DISPPSIZ DS    F               POOL SIZE (IN 2K UNITS)
1 DISPPSCO DS    XL1             SCOPE OF ISAM POOL
1 DISPPTSK EQU   X'00'           SCOPE = TASK
1 DISPPUSR EQU   X'01'           SCOPE = USERID
1 DISPPHOS EQU   X'02'           SCOPE = HOST
1 DISPPUGR EQU   X'03'           SCOPE = USERGROUP
1 DISPPWRO DS    XL1             WROUT ATTRIBUTE OF ISAM POOL
1 DISPPDEF EQU   X'00'           WROUT = DEFERRED
1 DISPPIMM EQU   X'01'           WROUT = IMMEDIATE
1 DISPPCST DS    XL1             CSTAT RESIDENCE OF ISAM POOL
1 DISPPNRE EQU   X'00'           ISAM POOL NOT CSTAT-RESIDENT
1 DISPPRSD EQU   X'01'           ISAM POOL CSTAT-RESIDENT
1 DISPPEXT DS    XL1             EXISTING ISAM POOL EXTENTS
1 DISPPNEX EQU   X'00'           NO EXTENT EXISTS
1 DISPPEX2 EQU   X'01'           2K EXTENT EXISTS
1 DISPPEX4 EQU   X'02'           4K EXTENT EXISTS
1 DISPPEXA EQU   X'03'           2K AND 4K EXTENT EXISTS
1 DISPPLCI DS    XL1             INDICATOR FOR LOCALITY
1 DISPPLCL EQU   X'00'           POOL ON LOCAL COMPUTER
1 DISPPREM EQU   X'01'           POOL ON REMOTE COMPUTER
1 DISPPUID DS    CL8             USERID FOR SCOPE = *USERID
1 *                              GROUP NAME FOR SCOPE = *USERGROUP
1 DISPPRSV DS    CL3             NOT USED
1 DISPPLNG EQU   *-DISPPDDS      LENGTH OF ISAM POOL DESCRIPTOR
```

Explanation of the individual fields in the output area:

The **management header** contains the following information:

- the number of bytes that were transferred to the output area (length: 4 bytes),
- the total number of bytes contained in the complete information (length: 4 bytes) requested. The caller of the macro must supply an output area of (at least) the same length,
- the number of ISAM pools for which information is to be placed in the output area (length: 2 bytes),
- an indicator, with a length of 1 byte, that represents the scope of information specified in the macro call. The following values apply:

  X'00': INFO = *ATTR

  X'01': INFO = *ALL

- an indicator, with a length of 1 byte, which specifies whether the information transferred to the output area is complete or whether only a part of the desired information could be returned because the output area was too small. The following values are assigned to this indicator:

  X'00': The information is complete.

  X'01': The information was truncated and is incomplete

  (In the latter case, the caller must provide a larger output area in order to receive the entire information. The length of this larger area can then be obtained from the second word of the management header),

- the remaining 4 bytes are not used.

The **specified output area** can contain the following information:

An ISAM pool is described in the output area specified by the caller by means of an ISAM pool descriptor, with a length of 32 bytes, which contains the following information:

- the name of the ISAM pool involved, with a length of 8 bytes (printable),
- the catalog ID (CATID) of the host computer on which the specified ISAM pool exists; also printable, and with a length of 4 bytes,
- the size of the ISAM pool involved (in 2K units), with a length of 4 bytes,
- the scope of the ISAM pool involved, with a length of 1 byte for which values are assigned as follows:

  X'00': SCOPE = *TASK

  X'01': SCOPE = *USERID

  X'02': SCOPE = *HOST

  X'03': SCOPE = *USERGROUP

- the WROUT attribute of the ISAM involved, with the following possible values (1 byte):

  X'00': WROUT = DEFERRED

  X'01': WROUT = IMMEDIATE

- the CSTAT-residence attribute of the ISAM pool involved (1 byte), with the following assignment:

  X'00': the ISAM pool is not CSTAT-resident

  X'01': the ISAM pool is CSTAT-resident

- whether the ISAM pool involved is made up of a 2K extent, a 4K extent, or both extents (for detains concerning 4K-capable NK-ISAM):

  X'00': no extent created as yet

  X'01': existing 2K extent for ISAM pool

  X'02': existing 4K extent for ISAM pool

  X'03': ISAM pool has a 2K extent as well as a 4K extent

- whether the ISAM pool involved belongs to a local or remote host computer, indicated by the following assignment:

  X'00': pool on local host

  X'01': pool on remote host

- the user ID – if the associated ISAM pool has been provided with the attribute SCOPE = USERID,
- the remaining 3 bytes are not used.

The structure of the management header (HDR) and ISAM pool descriptor (DESC) are the main factors that determine the format of the output area. The scope of information that was specified when calling the SHOPOOL macro determines the layout.

INFO = *ATTR

If the caller has specified the parameter INFO = *ATTR in the macro call (or selected this default value by omitting the parameter), the output area of the user will be structured as follows:

If INFO=*ATTR is specified, the indicator provided for it in the management header of the output area is assigned the value X'00'. The management header is followed by the ISAM pool descriptors located below one another in contiguous order (without gaps). The number of descriptors is also indicated in the management header.

```
HDR
DESC1
DESC2
  .......
  .......
DESCn
```

INFO = *ALL

By specifying the parameter INFO = *ALL in the SHOPOOL macro, the caller requests information in the output area on not only the static attributes of the specified ISAM pool, but also the TSNs of tasks connected to that pool. The indicator provided for this in the management header of the output area is assigned the value X'01', and each ISAM pool descriptor is followed by a word containing the number of tasks connected to that pool, followed by a list of the TSNs of the corresponding tasks (in 4 bytes each; printable).

If the caller has specified the parameter INFO = *ALL in the macro call, the output area of the user will be structured as follows:

```
HDR |
 DESC1   |
 TASK#1  | TSN11  | TSN12  |  .....  | TSN1m  |
 DESC2   |
 TASK#2  | TSN21  | TSN22  |  .....  |  ..... | TSN2r  |
  ......| ..........................................
  .....  | ..........................................
 DESCn   |
 TASK#n  | TSNn1  | .....  |  TSNns  |
```

SHOWAIX - Request information on secondary keys

Macro type: type S (E form/L form/D form/C form); see "Macro types"

The SHOWAIX macro returns, in its operand list, information on the secondary keys of an NK-ISAM file. For each secondary key defined in the file, it shows the user

- the name of the key
- the position of the key within the record
- the length of the key
- the DUPEKY setting for the key (whether or not duplicate secondary key values are permitted)
- the completeness of the secondary index blocks belonging to the key.

Format

Operation	Operands
SHOWAIX	**MF = L** **,FILE = pathname / LINK=linkname**
	MF = E,PARAM = adr / (r)
	MF = D[,PREFIX = pre]
	MF = C[,PREFIX = pre][,MACID = macid]

Operand descriptions

FILE = pathname

This specifies the NK-ISAM file for which the secondary key information is to be output with: <c-string 1..54: filename 1..54>..
The value specified for the FILE operand is ignored if the LINK operand is also specified.

“pathname” means [:catid:][$userid.]filename

catid
Catalog ID: if omitted, the default catalog ID for the current user ID is assumed.

userid
User ID: if omitted, the user ID in the SET-LOGON-PARAMETERS or LOGON command is assumed.

filename
Fully qualified file name.

LINK = linkname <1..8>

This specifies the link name of the file for which the secondary key information is to be output.
“linkname” may be up to 8 characters long. If the file link name is to be accessed via the command interface, it must correspond to the data type. <structured_name 1..8> (see the “Commands” manual [3]).

MACID

Specifies the second to fourth characters of each field name and equate generated when the macro is expanded.

Default value: MACID = ISI

= macid
“macid” is a three-character string which specifies the second to fourth characters of each field name and equate generated.

PARAM

Specifies the address of the operand list; it is evaluated only if MF=E applies (see "Macro types").

= addr
Symbolic address (name) of the operand list.

= (r)
Number of the register which contains the address of the operand list. The register must be loaded with this address value before the macro is called.

PREFIX

Specifies the first character of each field name or equate which the assembler generates in the data area when expanding the macro.

Default value: PREFIX=D

= pre
Single-character prefix with which the generated field names and equates are to begin.

Return codes

Standard header: ccbbaaaa

The following code relating to execution of the SHOWAIX macro is returned in the standard header(cc = SUBCODE2, bb = SUBCODE1, aaaa = MAINCODE):

X'bb'	X'aaaa'	Meaning
X'00'	X'0000'	The function was executed successfully.
X'01'	X'0001'	The operand list is not available.
X'40'	X'0002'	Secondary keys are not supported in the remote system (if the macro is called via RFA).
X'40'	X'0003'	The specified catalog ID does not exist.
X'40'	X'0004'	The catalog cannot be accessed.
X'01'	X'0005'	The operand list contains an invalid name.
X'20'	X'000B'	System error.
X'40'	X'000E'	The control block of the file is errored.
X'01'	X'0017'	There was no file specified in the operand list.
X'40'	X'0019'	The file link name is invalid.
X'40'	X'0040'	OPEN error.
X'40'	X'0041'	CLOSE error.
X'40'	X'0044'	The file is not an NK-ISAM file.

Further return codes, whose meanings are defined by conventions valid for all macros, can be found in the table on "Standard header" (standard header).
The calling program is terminated if one of the following errors occurs with regard to the parameter list:

- the list is not assigned to the caller
- the list is not aligned on a word boundary
- the list is write-protected.

Layout of the operand list

Macro expansion with MF=D and default values for PREFIX and MACID:

```
SHOWAIX MF=D
1          MFCHK MF=D,                                                  C
1                PREFIX=D,                                              C
1                MACID=ISI,                                             C
1                DMACID=ISI,                                            C
1                DNAME=ISIAIX,                                          C
1                SUPPORT=(C,D,L,E),                                     C
1                PARAM=,                                                C
1                ALIGN=F,                                               C
1                SVC=32
2 DISIAIX  DSECT ,
2                *,##### PREFIX=D, MACID=ISI #####
1          #INTF INTNAME=SHOWAIX,REFTYPE=REQUEST,INTCOMP=001
1          FHDR  MF=(C,DISI),EQUATES=NO
2          DS    0A
2 DISIFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 DISIIFID DS    0A              0   INTERFACE IDENTIFIER
2 DISIFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 DISIFCT  DS    AL1             2   FUNCTION NUMBER
2 DISIFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 DISIRET  DS    0A              4   GENERAL RETURN CODE
2 DISISRET DS    0AL2            4   SUB RETURN CODE
2 DISISR2  DS    AL1             4   SUB RETURN CODE 2
2 DISISR1  DS    AL1             5   SUB RETURN CODE 1
2 DISIMRET DS    0AL2            6   MAIN RETURN CODE
2 DISIMR2  DS    AL1             6   MAIN RETURN CODE 2
2 DISIMR1  DS    AL1             7   MAIN RETURN CODE 1
2 DISIFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *
1 * SUB RETURN CODE1
1 *
1 DISIRFSP EQU   X'00'              FUNCTION SUCCESSFULLY PROCESSED
1 DISIRPER EQU   X'01'              PARAMETER SYNTAX ERROR
1 *
1 DISIRFNS EQU   X'01'              CALLED FUNCTION NOT SUPPORTED
1 DISIRFNA EQU   X'02'              CALLED FUNCTION NOT AVAILABLE
1 DISIRVNA EQU   X'03'              INTERFACE VERSION NOT SUPPORTED
1 *
1 DISIRIER EQU   X'20'              INTERNAL ERROR
1 DISIRCAR EQU   X'40'              CORRECT AND RETRY
1 *
1 * MAIN RETURN CODE
1 *
1 DISIOK   EQU   0                  AIX DELETED
1 DISINPAR EQU   1                  PARLIST NOT ACCESSIBLE
1 DISINREM EQU   2                  NO SUPPORT ON REMOTE HOST
```

```
1 DISINCAT EQU   3                 CATID NOT KNOWN
1 DISINACC EQU   4                 CATALOG NOT ACCESSIBLE
1 DISIINVN EQU   5                 INVALID NAME
1 DISISYSE EQU   11                SYSTEM ERROR
1 DISISPAC EQU   12                NO ADDRESS SPACE
1 DISIWRCB EQU   14                WRONG CONTROLBLOCK
1 DISIFNSP EQU   23                FILE NOT SPECIFIED
1 DISILNKE EQU   25                LINKNAME ERROR
1 DISIINOP EQU   31                SSTA INOP
1 DISISSER EQU   32                SSTA INTERNAL ERROR
1 DISIMEMR EQU   33                SSTA MEMORY ERROR
1 DISIOPSE EQU   34                SSTA OPS ERROR
1 DISIOPME EQU   35                SSTA OPS MEMORY ERROR
1 DISIOPER EQU   64                FILE OPEN ERROR
1 DISICLER EQU   65                FILE CLOSE ERROR
1 DISINNKF EQU   68                NO NK-ISAM FILE
1 DISIRLNK EQU   X'FFFF'           LINKAGE ERROR
1 *
1 DISIDMSC DS    AL2               DMSCODE
1 DISIFILE DS    CL54              FILE
1 DISILINK DS    CL8               LINK
1 DISIKEY# DS    H                 NUMBER OF KEYS
1 *
1 DISIKNAM DS    CL8               KEYNAME
1 DISIKPOS DS    H                 KEYPOS
1 DISIKLEN DS    AL1               KEYLEN
1 DISIIND  DS    XL1               INDICATOR
1 DISIDUPK EQU   X'80'             SET:   DUPKEY = YES
1 *                                RESET: DUPKEY = NO
1 DISIINCO EQU   X'40'             SET:   SIX IS INCOMPLETE
1 *                                RESET: SIX IS COMPLETE
1          DS    29CL12
1 DISI#    EQU   (*-DISIFHE)        LENGTH OF STRUCTURE
           END
```

STORE - Store record

Macro type: R for PARMOD=24
O for PARMOD=31

The STORE macro transfers a record from the user area to the file, placing it at the position defined by the record key.

In contrast to INSRT, the STORE macro can process records with duplicate keys.
If DUPEKY=YES is specified, the new record is written after the last existing record with the same key. If DUPEKY=YES is not specified, the new record overwrites the existing record with the same key.

The PAD factor is ignored when a file is created or extended sequentially using STORE. In contrast to the PUT macro, sequential use of the STORE macro results in a high block splitting rate and each block will be only about 50% full.

Format

Operation	Operands
STORE	**fcbadr / (1)** **,area / (0)** **[,PARMOD = 24 / 31]**

Operand descriptions

fcbaddr

Address of the FCB associated with the file to be processed.

(1)
The FCB address is stored in register 1.

area

Address of the record to be stored. Even in locate mode, the record to be stored must be provided at address “area”.

(0)
The address of the record to be stored is in register 0.

PARMOD

Specifies the generation mode for the macro.

Default value: the value predefined for the generation mode by means of the GPARMOD macro or preset by the assembler.

= 24
The object code generated can run only in the 16-Mb address space (24-bit addressing only).

= 31
The object code generated can run in the 2-Gb address space (24-bit or 31-bit addressing).

Programming note

The STORE macro overwrites the contents of registers 0, 1, 14 and 15.

VERIF - Recover file

Macro type: type S (E form/L form); see "Macro types"

The VERIF macro is used to recover files, file generations or file generation groups which, due to a system crash or the abortion of a job, have not been closed correctly.

The macro can be used to

- clear a file lock so that the file becomes generally accessible again
- recover a disk file: the catalog entry is updated and, if necessary, the file is closed. ISAM files are recovered on the basis of the existing records.

Note

> If file access was interrupted while there were data buffers in main memory, the last changes made to the file may be missing in the recovered file, since the buffer contents are transferred to external storage only when the buffer becomes full.

The user can unlock disk files and tape files reserved exclusively with the SECURE-RESOURCE-ALLOCATION command if the job causing the lock has been terminated by the system with the console message `TASK PENDED INDEFINITELY`.

Disk files which were not locked via the SECURE-RESOURCE-ALLOCATION command can only be unlocked by the system administrator.

The associated crypto password must be specified to reconstruct an encrypted ISAM file.

Format

Operation	Operands
VERIF	**pathname$_1$[,pathname$_2$][,MF = L]** **[,REPAIR = YES / ABS / NO / CHECK]**
	MF = (E,adr / E,(r))

Operand descriptions

pathname1

Specifies the path name of the permanent or temporary file, file generation group or file generations to be recovered:
<c-string 1..54: filename 1..54>

“pathname1” means [:catid:][$userid.]filename

catid
Catalog ID;
Default value: the catalog ID assigned to the user ID.

userid
User ID;
Default value: the user ID specified in the SET-LOGON-PARAMETERS or LOGON command.

filename
Fully qualified name of a permanent or temporary file, file generation, or file generation group.

pathname2

Designates the file in which the ISAM file “pathname1” is to be reconstructed;
with <c-string 1..54: filename 1..54>

“pathname2” is relevant only for the recovery of ISAM files and must not be the same as “pathname1”. “pathname2” may be the name of a permanent or temporary file or file generation but not the name of a file generation group. If “pathname2” is omitted, the system creates a work file for reconstruction of the ISAM file. However, “pathname2” must be specified if the ISAM file is stored on private disk with its index and data sections on separate disks. If an encrypted file needs to be reconstructed, “pathname2” is assigned the same encryption attributes as “pathname1”.

“pathname2” means [:catid:][$userid.]filename

catid
Catalog ID; default value: the catalog ID belonging to the user ID.

userid
User ID; default value: the user ID specified in the SET-LOGON-PARAMETERS or LOGON command.

filename
Fully-qualified file name of a permanent or temporary file, or a file generation. File generation groups must not be specified.

REPAIR

Specifies how the files defined by “pathname1” are to be recovered. Reconstruction depends on the access method with which the files were created.

“Clearing a file lock” means that the entry for the file in the file lock table is deleted.

Concurrent copy locks remain in place if the concurrent copy session has not yet ended. With REPAIR=YES/ABS /CHECK, a requested reconstruction or consistency check is still performed. For REPAIR=NO, see the operand description under “Notes”.

Only REPAIR=NO is permitted for tape files and file generation groups.

= YES
Only for disk files

PAM:
The pointer to the last PAM page which was written (i.e., the last-page pointer LPP) is set to the highest possible value. This corresponds to the actual size of the file (with BLKCTRL=PAMKEY) or to the file size rounded up to a multiple of the block size (with BLKCTRL=DATA/NO). In this case the last-page pointer is also incremented to a block boundary.

The file is closed.

If a mirrored disk (DRV) is available, the file will be equalized.

SAM:
The highest PAM page written in the file is determined, and the last-page pointer (LPP) is set to this value.

If the file is on a mirrored disk (DRV), the contents of the blocks contained in it are restored by equalization if required.

The file is closed.

ISAM:
The file lock is cleared. If the file is marked as open, it is reconstructed.

= ABS
Only for disk files
The reconstruction is performed regardless of whether or not the file is identified as open.

PAM:
The file lock is cleared and the last-page pointer is updated to point to the last PAM page actually written and the last-byte pointer to a block boundary provided that the file is marked as open; otherwise the last-page pointer and the last-byte pointer remain unchanged.

In the case of files opened with BLKCTRL=PAMKEY/DATA, the highest written PAM page in the file is determined, and the last-page (LPP) pointer is set to that value.

For files opened with BLKCTRL=NO, the LPP is set to the highest possible PAM page (i.e. the file size, rounded up to a multiple of the block size).

If the file is open and on a mirrored disk (DRV), the contents of the blocks contained in it are restored by equalization if required.

The file is then closed (if required).

SAM:
The file lock is cleared. Even if the file is not marked as open, the last-page pointer is updated to point to the last PAM page actually written, and the file is then closed (if required).

ISAM:
The file lock is cleared and the file is reconstructed.

The verification process (with reconstruction) for the SAM and ISAM access methods is analogous to REPAIR =YES.

= CHECK
Only meaningful for NK-ISAM files processed with WROUT=YES:
only files which are marked as open are selected. The file lock is cleared, the last-page pointer is updated to point to the last page which was written, data blocks which consist of more than one PAM page (multiblocks) are checked for consistency of the blocking structure, and the file is closed.

The file lock is cleared. If the file is identified as open, the pointer to the last PAM block is set to the highest written page; multi-blocks are checked for consistency, and the defined secondary keys are verified to ensure that they have been created or deleted completely.

If the file is on a mirrored disk (DRV), the contents of the blocks contained in it are restored by equalization if required.

If no error was detected, the file is closed.

The “consistency of multi-blocks” implies that no abort occurred when writing a multiblock.

= NO
For tape input files:
“filename” must be fully qualified; the file lock is cleared.

For disk files:
PAM – The file lock is cleared. The file is not regarded as closed, i.e. it will still be included in the output from FSTAT ..., STATE= NOCLOS, and will be regarded as a file which is to be repaired for VERIF ..., REPAIR=YES.

SAM:
The file lock is cleared. The file is not regarded as closed, i.e. it will still be included in the output from FSTAT ..., STATE= NOCLOS, and will be regarded as a file which is to be repaired for VERIF ..., REPAIR=YES.

ISAM:
The file lock is cleared. If the file is marked as open, the system executes a privileged close operation and the last-page pointer is updated to point to the last PAM page written. The file is not reconstructed.

If the file is on a mirrored disk (DRV), the contents of the blocks contained in it are restored by equalization if required.

Inconsistencies between the INDEX and data section are neither detected nor removed in the process.

This also applies to inconsistencies with regard to the secondary keys. The file is considered to be closed, i.e. will not be reported by FSTATUS, STATE=NOCLOS and will not be flagged as a file requiring repair by VERIFY ..., REPAIR=YES.

Note on “concurrent copy” save procedure

Concurrent copy locks in a file are set up by the system/HSMS (see the “HSMS” manual [10]). The user can only remove these locks once the save procedure (concurrent copy session) is over. The following should be noted with respect to return codes:

- If a file is locked via both file locks and concurrent copy locks, a positive return code will be issued if the file locks could be removed.
- If, however, a file is locked only via a concurrent copy lock, a positive return code will only be issued once the concurrent copy session is over, otherwise a negative return code will be issued.

MF

For a detailed description of the MF operand, refer to the Appendix ("Macro types").

Programming notes

If the macro is executed successfully, the contents of register 15 are set to zero. The error codes for unsuccessful execution are defined in the IDEMS macro.

Reconstruction of ISAM files

- If “pathname2” is not specified for an ISAM file on public volumes, this file is reconstructed in a work file created by the system. The file “pathname1” is then erased without “DESTROY” (see the CATAL macro, DESTROY operand, "CATAL - Process catalog entry"), and the work file is renamed “pathname1”.

  If “pathname2” is not specified for an ISAM file on private volumes or on a Net-Storage volume, this file is reconstructed in a work file on public volumes. The work file is then copied into the file “pathname1” and then erased by means of “DESTROY” (see the CATAL macro, DESTROY operand, "CATAL - Process catalog entry"). This may take a very long time and it is therefore better to specify “pathname2”.
- If “pathname2” is specified in the VERIF macro, the file “pathname1” is reconstructed in this file and “pathname1” remains unchanged. If “pathname2” is to be on a private disk or on a Net-Storage volume, or if the file “pathname1” is an ISAM file with its index and data sections on separate disks, “pathname2” must be cataloged before the VERIF macro is issued and space must also be reserved for it.

  Records with identical index and data are transferred only once to the reconstructed file.
- No space is reserved in the data blocks of the reconstructed file for subsequent expansion (equivalent to specifying PAD=0 in the FILE macro).

  ISAM files with their data and index blocks on separate volumes can be reconstructed by means of the VERIF macro only if BLKSIZE=STD applies.
- If an ISAM data block contains data which cannot be assigned to a defined record, the entire block is saved in the PAM file “S.filename1.REPAIR”. After VERIF processing has been completed, the user can use this file for his/her own reconstruction efforts. If the new file name would be too long, “filename1” is truncated as required.
- Since a copy of the file is created during reconstruction of an ISAM file, and since this copy occupies part of the public space, the user must ensure that sufficient storage space is available.

Appendix

The Appendix contains:

- Information on the syntax used in macro calls (starting on "Syntax presentation")
- A section on evaluating DMS error messages (starting on "DMS error codes")
- The DIV call interface (starting on "CALL interface for DIV")
- All label formats including information on the processing of the label fields (starting on "Labels")
- Information on Dsect generation (starting on "DMS dummy sections (DSECTs)")

It concludes with the macro formats for the two replaced macros COPY and REL which continue to be supported for reasons of compatibility (starting on "Formats of replaced macros").

Syntax presentation

- Macro format
- Metasyntax used for the macros
- Wildcards
- Format of date specifications
- Macro types
- Standard header

Macro format

A macro format consists of two columns; the first column contains the macro name while the second contains the possible operands.

Macro name	Operands
<macroname>	<operand $_1$> ,<operand $_2$>

When a macro is called, the macro name must be separated from the first operand by at least one blank. Where several operands are specified together, they must be separated by commas.

The macro formats are represented with the aid of certain notational symbols (metacharacters), which are explained in the following table.

Metasyntax used for the macros

Elements of the metasyntax

Representation	Meaning	Examples
UPPERCASE LETTERS	Uppercase letters are used for keywords or constants, which the user must enter exactly as they are shown. Keywords must begin with * in case both keywords and names of constants or variables can be specified.	DIB FORCED=*YES
Lowercase letters	Lowercase letters denote data types of the values or variables which the user can specify.	DIB = <var: pointer>
< >	Angle brackets denote variables whose range of values is described by the data types.	<var: pointer>
Underscoring	Underscoring denotes the default value of an operand. If an operand does not have a default value, another value must be specified for it.	FORCED = *NO / *YES
=	The equals sign connects an operand name with the associated operand values.	DATA = <var: pointer>
/	A slash serves to separate alternative operand values.	FORCED = *NO / *YES
[ ]	Square brackets enclose optional entries, i.e entries which may be omitted. If the comma is inside the square brackets, it is needed only if this optional entry is used and may be omitted for the first operand in a command. If, in contrast, the comma is outside the brackets, it must always be entered, even if no optional entry is specified. Note that normal (round) parentheses must always be entered.	F[REE]SIZE Enter: FREESIZE or the shortened form FSIZE
list-poss(n)	The entry “list-poss” signifies that a list of operand values can be given at this point. If (n) is present, it means that the list must not have more than n elements. A list of more than one element must be enclosed in parentheses.	FLAG=list-poss(3): *SLI / *SKIP / *DC Specification: FLAG=*SKIP FLAG=(*SLI,*DC)

An operand is assigned an operand value from a defined range of values by means of the equals sign.

This value range is determined by a data type. The following table contains the data types of the operand values.

Data types of the operand values

Data type	Character set	Special rules
c-string	EBCDIC character	Must be enclosed within single quotes
integer	[+-] 0..2147483647	Is a decimal number
var:	Introduces a variable specification. The colon is followed by the type of the variable (see table "Data types for variables" below)	<var:var-type>
reg:	Registers 0..15	Specification: (<reg:var-type>)

Suffixes for data types

Suffix	Meaning
n..m	With data type “integer”, n..m means an interval specification; n: minimum value m: maximum value
	With data type “c-string”, n..m means a length specification in bytes; n: minimum length m: maximum length with n < m
n	With data type “c-string”, n means a length specification in bytes; n *must* be adhered to.

The operand values can be entered directly as a character string or integer (see data types “c-string” and “integer”), or indirectly via a variable (see data type “var:”). The following table contains the possible data types for variables.

Data types for variables

Data type	Description	Definition in program
char:n	The variable is a string of n characters. If the length specification is omitted, n=1 is assumed.	CLn
int:n	The variable is an integer occupying n bytes. If the length specification is omitted, n=1 is assumed. Condition: n <= 4	FLn
enum-of E:n	The variable is the list E, which occupies n bytes. If the length specification is omitted, n=1 is assumed. (n <= 4)	XLn

pointer	The variable is an address or an address value.	A

Wildcards

Wildcards may be used in the catalog ID and the file name in the operand “pathname” of the macros ERASE and FSTAT. In the ERASE macro, system files may also be addressed via wildcards.

Wildcard	Meaning
*	Replaces a freely selectable character string, even an empty character string.
/	Replaces precisely one character.
<wildcard1,...>	Replaces all character strings which match at least one of the specified wildcards.
<wildcard1:wildcard2>	Replaces a character string which • is at least as long as the shortest specified wildcard; • is not longer than the longest specified wildcard; • lies in the alphabetical collating sequence between “wildcard1” and “wildcard2” (digits follow letters). “wildcard1” may also be an empty character string; this occupies the first position in the alphabetical collating sequence.
<wildcard1:wildcard2,...>	Wildcards of the type “wildcard1:wildcard2” may also be specified in the form of a list. The rules for each pair of wildcards are the same as described above. The system logically ORs the pairs, i.e. the wildcard list replaces any character string which matches one of the wildcard pairs. The length rule applies separately to each pair, not to all wildcards in the list.
-wildcard	Replaces any character string which does not match the specified wildcard. The minus sign may be specified only at the beginning of the wildcard string.

Format of date specifications

Date specifications are required in the macros ERASE and FSTAT, in the operands CRDATE, DELDATE, EXDATE, LADATE and LCDATE, as well as in the CATAL macro in the DELDATE and EXDATE operands. The user may specify either absolute or relative dates.
A specific time or a time interval may be specified in addition to the date.

The **TIMBASE** operand can be used to control whether absolute dates are specified on the basis of **UTC** (universal time coordinate) or local time **LTI** (relative entries are always in local time).

The date format of the FSTAT output is also linked to this operand.

Absolute date specification

A real date in the form:
YYMMDD or [YY]YY-[M]M-[D]D
(where YY = year, MM = month, DD = day)

Relative date specification

The offset in days from the current date in the form “-n” for the past or “+n” for the future (n=0..99999),
or as Y[ESTERDAY] (correspnds to -1), T[ODAY] (correspnds to ±0)
or TOM[ORROW] (correspnds to +1)

Time specification

A time value (UTC time) related to the date in the format date(hh:mm:ss)(hh = hours, mm = minutes, ss = seconds)

Macro types

Macros are divided into types on the basis of the manner in which their operands are passed. The various types are type O, type R (where the operands are passed in registers) and type S (where the operands are passed in memory; S = storage).

O-type macros

Macros which are neither type R nor type S are known as O-type macros.

Examples of this type are macros in which a register (often only R1) containing the start address of the operand list is specified in the operand field.

The operand list is defined in the data section of the program (using DC statements) and contains the operand values.

R-type macros

Operation	Operands
RTYP	operand1 / (r1),operand2 / (r2)

A macro belongs to type R if all necessary operand values can be loaded into the two registers (0 and 1) used for this purpose. An R-type macro does not generate an operand list.

The operands may be specified directly or placed in registers 0 and 1.

Address operands in R-type macros may be specified as explicit or implicit addresses.

S-type macros

Format 1 (S-, L-, D-, C-, M form)

Name	Operation	Operands
[opadr]	macro	operand1,...operandn [,MF = S / L / C[,PREFIX = p][,MACID = mac] / (C,p) / D[,PREFIX = p] / (D,p) / M[,PREFIX = p][,MACID = mac]]

Format 2 (E form)

Name	Operation	Operands
[opadr]	macro	MF = E / (E,opaddr) / (E,(r))[,PARAM = adr]

For S-type macros, the operand values specified in the macro are passed to the function module in the form of a data area which is part of the macro expansion. This is a suitably structured area which contains the data and memory definitions (DC and DS statements) necessary for passing the operand values.

The following applies to all macros that can be called with a specific macro version (e.g. via the VERSION or PARMOD operand): the version operand must have the same value in all calls with different values for the MF operand (MF=L/E/D/C).

With regard to MF, there are six different macro call forms: S form, E form, L form, D form, C form and M form.

S form = standard form

MF=S is the default value. The instruction section is generated first, followed by the data area with the operand values specified in the macro call. This data area contains no field names and no explanatory equates. The standard header is initialized.

E form = execute form

	Operation	Operands
[label]	**macro**	**MF = E,PARAM = addr / (r) / (1)**

The E form of the MF operand initiates a supervisor call (SVC): the contents of an operand list (see L form) are evaluated and the corresponding operations are executed. For this reason, the “execute” macro call must include the address of the operand list, either as a symbolic address (addr) or in a register (r/1). No other operands are evaluated.

“label” is the symbolic address of the macro in the Assembler program.

L form = LIST form

	Operation	Operands
label	**macro**	**MF=L,operand-list**

The list form uses the other operands specified in the macro to generate an operand list. This list does not contain symbolic addresses for the operands; these are generated by the C or D form of the macro. The address of the operand list must be specified in the macro (E form), which means that the symbolic address “label” must always be specified.

The operand list begins with the standard header (see "Standard header"), whose fields are loaded automatically when the list is created with MF=L. Even if an operand list is to be created dynamically with the D or C form, it must be initialized beforehand with MF=L in order to ensure that the header contains the correct information.

D form = DSECT form

	Operation	Operands
[label]	**macro**	**MF=D[,PREFIX=prefix]**

The D form generates a DSECT for the operand list of the macro. The first character of the generated names can be modified via the PREFIX operand. If a symbolic address is defined for the macro via “label”, the DSECT receives this name. If “label” is not defined, the DSECT receives a macro-specific default name whose first character is likewise modified by PREFIX. The operand list should be initialized with the list form of the macro before the DSECT call in order to ensure that the standard header contains the correct information.

C form

	Operation	Operands
[label]	**macro**	**MF=C[,PREFIX=prefix][,MACID=id**

Like the D form, the C form generates an operand list, but not in the form of a DSECT since no DSECT statement is generated. The operand list remains empty and should be initialized with the L form of the macro call in order to ensure, above all, that the standard header contains the correct information.

Using the PREFIX operand, the first character of the generated names can be changed, while the MACID operand can be used to change the second to fourth characters in these names (a string of up to three characters can be specified for MACID). If the macro is addressed symbolically via “label”, this is also the address of the operand list; if “label” is not specified, the operand list cannot be addressed symbolically.

M form = modification form

	Operation	Operands
[label]	**macro**	**MF=M[,PREFIX=prefix][,MACID=id],operand-list**

Instructions (e.g. MVCs) are generated to overwrite the fields in an initialized data area (operand list) with the operand values specified in the macro call. The M form thus offers an easy way of dynamically matching the operand values with which a macro is called to the requirements of the program.

Since the instructions generated for this purpose use the addresses and equates of the C form or D form, care must be taken, when using the M form, that these names are available for addressing the operand list to be modified. In particular, care must be taken that, if specified, the operands PREFIX and MACID in a macro call with MF=M have the same values as in the associated MF=C or MF=D call.

Standard header

All DMS macros use the standard header for BS2000 macros in their 31-bit interface. This standard header is an 8-byte field at the beginning of the operand list which contains the standardized designation of the interface and provides space for return codes. It is generated by each macro and should – wherever possible – be initialized with the list form of the MF operand.

Structure of the standard header

Field	Byte position	Meaning
UNIT	0-1	Specifies the function unit in which the desired function is implemented
FUNCTION	2	Specifies the function (within the function unit)
VERSION	3	Specifies the version number of the function
SUBCODE2	4	Contains subsidiary return code 2
SUBCODE1	5	Contains subsidiary return code 1
MAINCODE	6-7	Contains the main return code

The fields SUBCODE2, SUBCODE1, MAINCODE contain the return code. The main return code indicates whether or not the operation was executed successfully. In the case of an error, the subsidiary return codes can be used to diagnose the reason for the error.

The following values for the return codes are conventions:

SUB-CODE2	SUB-CODE1	MAIN-CODE	Meaning
X'00'	X'00'	X'0000'	The function was executed successfully; there is no further information for this MAINCODE.
X'01'	X'00'	X'0000'	The function was executed successfully; no further action is necessary.
X'00'	X'01'	X'FFFF'	The requested function is not supported (invalid (entry for UNIT or FUNCTION in the standard header); unrecoverable error.
X'00'	X'02'	X'FFFF'	The specified function is not available; unrecoverable error.
X'00'	X'03'	X'FFFF'	The specified version of the interface is not supported (invalid version entry in the standard header); unrecoverable error.
X'00'	X'04'	X'FFFF'	The parameter list is not aligned on a word boundary.
X'00'	X'41'	X'FFFF'	The subsystem does not exist; it must be generated explicitly.
X'00'	X'42'	X'FFFF'	The calling process is not connected to this interface; it must be connected explicitly.
X'00'	X'81'	X'FFFF'	The subsystem is currently not available.

X'00'	X'82'	X'FFFF'	The subsystem is in the DELETE or HOLD state.

MAINCODE shows the result of execution of the function. SUBCODE1 qualifies the main code. SUBCODE2 subdivides the errors into error classes.

The return code should be passed only in the standard header (except for DMS macros; see "DMS error codes"). However, for a transitional period the return code can also be passed in register R15 or in both the standard header and register R15. In order to check whether a return code has been passed in the standard header, the return code field should be filled with X'FFFFFFFF'. The result of checking the standard header is also always returned in register R15:

X'00000000':	standard header correctly initialized; error-free execution.
X'0001FFFF'	invalid entry for UNIT or FUNCTION.
X'0003FFFF':	invalid entry for VERSION.

DMS error codes

Error codes contain information about errors affecting programs or jobs, enabling the user to determine the type and origin of the error and the appropriate remedial action. The error codes used in DMS offer the following advantages:

- the abbreviated format permits many different error conditions to be encoded
- the type of encoding permits identification of the component which was the source of the error
- in the case of recoverable errors, the program can implement appropriate analysis measures and thus avoid abnormal termination.

DMS errors in program execution

In the following management macro calls, the error code is (with subcodes) placed in the standard header:

- CATAL with VERSION=3
- COMPFIL
- COPFILE
- DECFILE
- DROPTFT
- ENCFILE
- ERASE with VERSION=3
- FILE with VERSION=3
- FSTAT with VERSION=3/4
- LFFSNAP
- LJFSNAP
- MAILFIL
- RDTFT with VERSION=3
- RELTFT
- RFFSNAP
- RJFSNAP

In the following management macro calls, the error code is (without subcodes) placed in the two low-order bytes of register 15:

- CATAL with VERSION<3
- CHNGE
- COPY
- ERASE with VERSION<3
- FILE with VERSION<3
- FSTAT with VERSION<3
- IMPORT
- RDTFT without VERSION and with VERSION=2
- REL

For data management macros (service macros and access methods), error codes are placed in field ID1ECB of the TU FCB (displacement X'98').

If a DMS error occurs during program execution, an error code is placed in the file control block. If the program does not contain a routine for handling such errors, the program is terminated with an error message on SYSOUT in interactive mode or on SYSLST in batch mode.

Coding methods

```
w x y z
```

The error code is a four-digit hexadecimal whose second digit identifies the component that noted the error. The digits w, y and z indicate the error which has occurred.

Error code value table for the components:

x	Component
2	Privileged PAM (PPAM)
3	Catalog administrator (CMS)
4	Data storage administrator (ALLOC)
5 and 6	DMS commands / Assembler macros
7	Privileged tape access method (PTAM)
9	UPAM
A	ISAM
B	SAM
C	BTAM
D	OPEN
E	CLOSE

When components are in the sequence 2_{16}-E_{16} this means that a component with a higher number can call a component with a lower number and not vice versa).

If an error occurs in a called component, the error code is passed to the calling component (and modified in the process).

Example for the generation and modification of error codes

- *single-stage* (error in the calling component): a user calls UPAM and the file is not open: UPAM places error code 0994 in the FCB and branches via $GOTO to USERERR.

- *two-stage* (error in the called component): a user calls UPAM, which in turn calls PPAM. PPAM detects an I/O error: in this case, PPAM passes error code 0227 to UPAM in register 15. UPAM modifies this error code to 0927, places it in the FCB, and branches via $GOTO to ERRADR.

List of DMS error codes

IDEMS macro

The explanations of the error codes are given below in list form. This list can be output, either completely or partially, with the aid of the IDEMS macro.

Operation	Operands	Comments
IDEMS	[ALL=Y]	All error codes are generated
	[,PAM=Y]	Only PPAM codes
	[,CATAL=Y]	Only CMS codes
	[,ALLOC=Y]	Only ALLOC codes
	[,CMDMAC=Y]	Only codes for DMS commands/macros, part 1
	[,CMDNMAC=Y]	Only codes for DMS commands/macros, part 2
	[,NPAM=Y]	Only UPAM codes
	[,ISAM=Y]	Only ISAM codes
	[,SAM=Y]	Only SAM codes
	[,BTAM=Y]	Only BTAM codes
	[,OPEN=Y]	Only codes for OPEN processing
	[,CLOSE=Y]	Only codes for CLOSE processing
	[,P=letter]	Prefix for the symbolic names of the DMS messages; Default value: I “P=” generates no prefix letter.

System message prefix

When system messages are output, the error code is prefixed by the message class “DMS” (0D33 -> DMS0D33). The command or standard statement HELP-MSG-INFORMATION enables a message text and further information on the cause of the error and how it is dealt with to be output in English or German.

Exceptions

- ISAM: the error code for ISAM is always 0AAz (where 0 <= z <= F), even for the system messages.
  Example
  If PPAM detects an error after an ISAM call, error code '0AA9' is always placed in the FCB. The entry in the IDEMS list for this code is “SYSTEM ERROR, HARDWARE”. ISAM issues this message for all errors detected in PPAM.

- OPEN: when ISAM files are opened, errors which cannot be analyzed via the error code may occur.
  Example
  When an ISAM file is opened in INPUT mode, the “first PAM page” is read. this page always contains the control block (NK-ISAM) or the highest-level index block (K-ISAM).

After an I/O error, the following occurs:

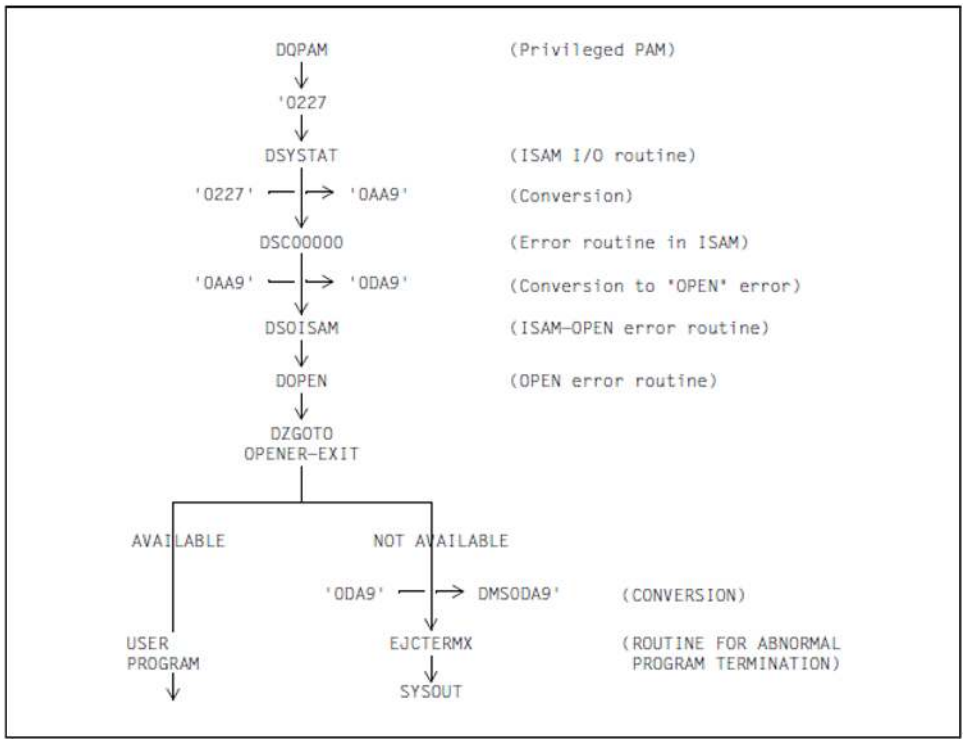


Figure 11: Error sequence after an input/output error

Note
This error has a diffesrent meaning in the IDEMS macro, but stands for an I/O error here.

- CLOSE: in order to complete all outstanding I/O operations, the CLOSE routine calls the access method for the file.

Error analysis aids

The following aids are available for analyzing error codes:

- Look for the meaning in the IDEMS list.
- If file management macros are used in the program, check the error code (in the standard header or in register 15). If the error code0, the program should be terminated or the contents of the error code should be thoroughly analyzed.

CALL interface for DIV

Adapter Window Services can be used in some high-level programming languages to invoke DIV functions from programs via a CALL interface. These programming language are:

- COBOL (Version 2.0 onward)
- FORTRAN (Version 2.2 onward)
- PL/1 (Version 4.1 onward)
- C (Version 2.0 onward)

ILCS linkage is required in order to call DIV functions via a CALL interface.
When programs in COBOL, FORTRAN and C are compiled, the ICLS linkage is provided by default. If PL/1 programs are compiled, an appropriate compiler option must be set.

A special feature must be noted for programs written in C: pointers to data elements must be passed, not the data elements themselves (“call by reference” passing in C).

The semantics of data types in the programming languages indicated above varies widely. The following table lists the data types for which the data representation in the above languages is identical, so that they can be passed as parameters without problems to the extent that is required when calling Window Services.

Data types from programming languages that can be passed as parameters for DIV:

Compiler	Data type binary word	Data type string
PL/1	BIN FIXED (31)	CHAR (i)
COBOL	PIC S9(i) COMP (i = 5,...,9)	USAGE DISPLAY
C	long	char <var>
FORTRAN	INT * 4	(<size>) CHAR * i

The Adapter Window Services are supplied as a runtime library (OML) in a file called SYSLIB.DWS.110 under the system administrator ID TSOS.

When DIV is invoked from the Window Services, not all DIV functions are available fully.

- Windows cannot be located in a data space when using Window Services (SPID is dropped).
- The LOCVIEW=MAP function does not exist for Window Services.
- RELEASE=YES (in the RESET function) is not supported by Window Services.
- In the case of sequential processing, instead of the option of specifying the number of pages to be read (PFCOUNT) in advance, only the sequential access (USAGE) is indicated for Window Services.
  USAGE='RANDOM' has the same effect as PFCOUNT=0,
  USAGE='SEQ' is mapped to PFCOUNT=15.
- Only the main code is available as a return code.

CALLs

A number of different parameters must be passed when the corresponding functions are called (with CALL). The following table shows a summary of all possible values.

Field name	Data type PL/1 environment	Brief explanation
DISPOS	CHAR(6);	/* for function=MAP: 'OBJECT' 'UNCHNG' /* for function=UNMAP: 'UNCHNG' 'FRESH
DMS_CODE	BIN FIXED (31);	
FILENAME	CHAR(54);	
ID	CHAR (8);	* ID of the OPEN, /* (output parameter)
LINKNAME	CHAR(8);	
OFFSET	BIN FIXED (31);	
OPEN-MODE	CHAR(5);	
RETURNCODE	BIN FIXED (31);	/* 'INPUT' INOUT' 'OUTIN'
SHARUPD-MODE	CHAR(4);	/* 'NO' 'WEAK' 'YES'
SIZE	BIN FIXED (31);	
SPAN	BIN FIXED (31);	/* file size, /* (output param)
USAGE	CHAR(6);	/* ' RANDOM' ' SEQ'
WINDOW	AREA(1) BASED ...;	/* aligned on page /* boundary !!

All CALLs for the individual DIV functions (in PL/1) are listed below. The parameters to be specified in each CALL are shown in the individual CALLs.

DIV function OPEN

> CALL DWSOPEN (LINKNAME, FILENAME, OPEN-MODE, SHARUPD-MODE,ID, SIZE, RETURNCODE, DMS-CODE);

DIV function MAP

> CALL DWSMAP (ID, OFFSET, SPAN, WINDOW, USAGE, DISPOS, RETURNCODE);

DIV function SAVE

```
CALL DWSSAVE (ID, OFFSET, SPAN, SIZE, RETURNCODE);
```

REFRESH function (equivalent to the function 'RESET')

```
CALL DWSREFR (ID, OFFSET, SPAN, RETURNCODE);
```

DIV function UNMAP

```
CALL DWSUNMP  (ID, OFFSET, SPAN, WINDOW, DISPOS, RETURNCODE);
```

DIV function CLOSE

```
CALL DWSCLS (ID, RETURNCODE, DMS_CODE);
```

Labels

- Volume header labels
- User volume header labels (UVL1 through UVL9)
- File header labels (HDR1 through HDR9)
- User file header labels (UHL)
- End-of-volume labels (EOV1 through EOV9)
- End-of-file labels (EOF1 through EOF9)
- User file trailer labels (UTL)
- Processing of label fields

Volume header labels

Each volume contains at least one volume header label (VOL1) and at most nine. Volume header labels VOL2 through VOL9 are optional.

First volume header label (VOL1)

The first volume header label identifies the volume, the owner, the access conditions, the implementation, and the edition of the appropriate standard.

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	VOL
4	Label number	1	1
5 – 10	Volume identifier	6	“a” characters; permanently assigned by the owner to identify the volume.
11	Volume access indicator	1	“a” characters; indicates restrictions governing access to the data on this volume. A space indicates that there are no access restrictions for this volume. Any other “a” character means that there are special restrictions governing access to this volume. BS2000: Space or “0”: unrestricted access. “1”: access restricted to owner.
12 – 24	Reserved for future standardization	13	Spaces
25 – 37	System code	13	“a” characters; identifies the implementation that recorded the volume header label.
38 – 51	Owner identifier	14	“a” characters; identifies the owner of the volume.
 38 – 41 42 – 49 50 – 51		 4 8 2	BS2000: Spaces “a” characters: user ID Spaces
52 – 79	Reserved for future standardization	28	Spaces

80	Label standard version	1	Indicates to which version of the standard the labels and data formats on the volume conform. 4 signifies: DIN 66029-4 (September 1987 edition) 3 signifies: DIN 66029-3 (May 1979 edition) 2 signifies: DIN 66029-2 (June 1976 edition) 1 signifies: DIN 66029-1 (August 1972 edition)

Optional volume header labels (VOL2 through VOL9)

The other volume header labels are optional. They are not generated by Fujitsu implementations.

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	VOL
4	Label number	1	Digits 2 through 9.
5 – 80	Reserved for the implementation	76	There are no conventions or restrictions regarding the recording and contents of this field.

User volume header labels (UVL1 through UVL9)

The user volume header labels are optional. If used, they must have the following format:

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	UVL
4	Label number	1	Digits 1 through 9.
5 – 80	Reserved for the installation	76	There are no conventions or restrictions with regard to the recording and contents of this field.

BS2000 supplies the user with the user volume header labels.

File header labels (HDR1 through HDR9)

Each file or file section contains at least two file header labels (HDR1 and HDR2) and at most nine. File header labels HDR3 through HDR9 are optional.

First file header label (HDR1)

The first file header label identifies a file section, describes its location within the file set and defines certain attributes of the file section.

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	HDR
4	Label number	1	1
5 – 21	File name	17	“a” characters; identifies the file.
22 – 27	File set identifier	6	“a” characters; identifies the file set to which this file belongs.
28 – 31	File section number	4	“n” characters; identifies the file section. The number of the first file section in a file is 0001. This number is incremented by one for every subsequent file section in this file.
32 – 35	File sequence number	4	“n” characters; identifies the file in the file set. The file sequence number of the first file in a file set is 0001. This number is incremented by one for each subsequent file in this file set. This field must contain the same number in all the labels of a specific file, regardless of whether the file is contained on one or more volumes.
36 – 39	Generation number	4	“n” characters; distinguishes the successive extensions/updates of the file from 0001 through 9999.
40 – 41	Version number	2	“n” characters; distinguishes the successive repetitions of a generation.
42 – 47	Creation date	6	Spaces or “n” characters; specifies the creation date of the file section. A space indicates the 20th century; the digit 0 indicates the 21st century, followed by two “n” characters (00 – 99) indicating the year within the century, followed by three “n” characters (001 – 366) indicating the day of the year. The value 00000 in the last five places indicates that the creation date is irrelevant.
48 – 53	Expiration date	6	Spaces or “n” characters; specifies the earliest date on which the file section may be deleted. The format is the same as for the “creation date” field (positions 42 – 47). The value 00000 in the last five places indicates that the expiration date is irrelevant and that the file section is obsolete.

54	File access indicator	1	“a” characters; indicates restrictions with regard to access to data in this file. A space indicates that there are no restrictions with regard to accessing the file. Any other “a” character indicates that there are special restrictions governing access to the file. BS2000: “1” or “3”: tape or file owner has access authorization.
55 – 60	Block count	6	000000
61 – 73 61 – 65 66 – 73	System code	13 5 - 8	“a” characters; identifies the implementation responsible for creating the labels. BS2000: BS2000 Spaces
74 – 80	Reserved for later standardization	7	Spaces

Second file header label (HDR2)

The second file header label describes certain attributes of the file and of the implementation.

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	HDR
4	Label number	1	2
5	Record format	1	F, D or S, and V and U (not supported according to DIN); specifies the format of the records in the file. • F: all the records in the file have a fixed record length. • D: all the records in the file have a variable record length and the number of characters (as a decimal) is specified in the record itself. • S: all the records are segmented. • V: all the records have an undefined length (not supported according to DIN). • U: all the records are variable in length and the number of characters (in binary form) is specified in the record itself (not supported according to DIN).
		1	

6 – 10	Block length	5	n” characters; specifies the maximum number of characters per block in the file. BS2000: With standard identifier 1 the following contents are possible:

6 – 7	Standard blocks		'80': standard block identifier.
8 – 10			n” characters: specifies the number of standard blocks.
11 – 15	Record length	5	“n” characters; identifies the record length in conjunction with the record format (position 5). With record format F this field contains the actual record length. With record formats D and V this field contains the maximum length, including the record length word (RLW). With record format S this field contains the maximum record length, excluding the segment control words (SCW). If this field contains 00000 with record format S, it means that the record length may be greater than 99999. With record format U, this field contains the maximum number of characters that may be contained in one record.
16 – 50	Reserved for the implementation	35	There are no conventions or restrictions with regard to the recording and contents of this field. BS2000 assignments up to support of DIN level 4:
16	Recording density	1	0 200 Bpi 1 556 Bpi 2 800 Bpi 3 1600 Bpi 4 6250 Bpi
17	Data position	1	Indicator in the case of track swapping: 0 no 1 yes
18 – 34	Job step identifier	17	ID assigned by task management.
35 – 36	Recording density for tape cartridges	2	' 'BLANK''BLANK'' not compressed ' P'BLANK'' compressed
47 – 50	File name code	4	Used only up to DIN 66029-1 if positions 6 and 7 contain standard blocks.
51 – 52	Buffer displacement	2	“n” characters; specifies the length (in characters) of an additional field that is placed at the beginning of each block.
53 – 80	Reserved for future standardization	28	Spaces

The file header label (HDR3)

The HDR3 label contains the complete file name, the passwords and the access mode relevant for the file owner.

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	HDR
4	Label number	1	3
5 – 12	Owner identifier	8	Identifies the owner of the file (user ID).
13 – 56	File name	44	The first 44 characters of the name of the file or file generation to which the file belongs.
57 – 60	Read password	4	Specifies a password governing read access to the file.
61 – 64	Write password	4	Specifies a password governing read and write access to the file.
65 – 68	Execute password	4	Specifies a password that must be entered in order to execute a load module contained in the file.
69	Access mode	1	Specifies the valid access mode: 0 : read and write access permitted. 1 : only read access permitted.
70 – 80	Reserved	11	Spaces.

Optional file header labels (HDR4 through HDR9)

The other file header labels are optional and contain implementation-specific information.

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	HDR
4	Label number	1	Digits 4 through 9.
5 – 80	Reserved for the implementation	76	There are no conventions or restrictions with regard to the recording or contents of this field.

User file header labels (UHL)

The user file header labels are optional.

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	UHL
4	Label number	1	“a” characters; must be defined by the user. BS2000: “b” characters.
5 – 80	Reserved for the user	76	There are no conventions or restrictions with regard to the recording and contents of this field.

BS2000 supports up to 255 user file header labels. The label number (position 4) is defined by the user.

End-of-volume labels (EOV1 through EOV9)

Every file section extended on a continuation volume contains at least two end-of-volume labels (EOV1 and EOV2) and at most nine. End-of-volume labels EOV3 through EOV9 are optional.

First end-of-volume label (EOV1)

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	EOV
4	Label number	1	1
5 – 54	As for the corresponding fields in HDR1	50	As for the corresponding fields in HDR1.
55 – 60	Block count	6	“n” characters; specifies the number of data blocks that make up the file section.
61 – 80	As for the corresponding fields in HDR1	20	As for the corresponding fields in HDR1.

Second end-of-volume label (EOV2)

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	EOV
4	Label number	1	2
5 – 80	As for the corresponding fields in HDR2	76	As for the corresponding fields in HDR2.

Third end-of-volume label (EOV3)

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	EOV
4	Label number	1	3
5 – 80	As for the corresponding fields in HDR3	76	As for the corresponding fields in HDR3.

Optional end-of-volume labels (EOV4 through EOV9)

The end-of-volume labels contain implementation-specific information.

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	EOV
4	Label number	1	Digits 4 through 9.
5 – 80	Reserved for the implementation	76	There are no conventions or restrictions with regard to the recording and contents of this field.

End-of-file labels (EOF1 through EOF9)

Each file contains at least two end-of-file labels (EOF1 and EOF2) and at most nine. End-of-file labels EOF3 through EOF9 are optional.

First end-of-file label (EOF1)

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	EOF
4	Label number	1	1
5 – 54	As for the corresponding fields in HDR1	50	As for the corresponding fields in HDR1.
55 – 60	Block count	6	“n” characters; specifies the number of data blocks that make up the file section.
61 – 80	As for the corresponding fields in HDR1	20	As for the corresponding fields in HDR1.

Second end-of-file label (EOF2)

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	EOF
4	Label number	1	2
5 – 80	As for the corresponding fields in HDR2	76	As for the corresponding fields in HDR2.

Third end-of-file label (EOF3)

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	EOF
4	Label number	1	3
5 – 80	As for the corresponding fields in HDR3	76	As for the corresponding fields in HDR3.

Optional end-of-file labels (EOF4 through EOF9)

The other end-of-file labels are optional and contain implementation-specific information.

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	EOF
4	Label number	1	Digits 4 through 9.
5 – 80	Reserved for the implementation	76	There are no conventions or restrictions with regard to the recording and contents of this field.

User file trailer labels (UTL)

The user file trailer labels are optional.

Format

Position	Field name	Length	Contents
1 – 3	Label identifier	3	UTL
4	Label number	1	“a” characters; must be defined by the user. BS2000: “b” characters.
5 – 80	Reserved for the user	76	There are no conventions or restrictions with regard to the recording and contents of this field.

BS2000 supports up to 255 user file trailer labels. The label numbers (position 4) are defined by the user.

Processing of label fields

Requirements for a sending system

Files

The records of the files must be passed to the implementation by the application program.

Labels

The installation must transfer the recording information required in each of the label fields listed below to the implementation; otherwise the implementation must supply this information.

For each volume in a volume set:

- volume identifier (VOL1, positions 5 through 10)
- volume access indicator (VOL1, position 11).

For each file in a file set:

- file access indicator (HDR1, position 54)

If the implementation permits the installation to supply the information to be processed in each of the label fields listed below, the implementation must process this information. If the installation does not supply this information, it must be supplied by the implementation.

For each volume in a volume set:

- owner identifier (VOL1, positions 38 through 51)

For each file in a file set:

- file set identifier (HDR1, positions 22 through 27)

The implementation must permit the application program to supply the information to be processed in each of the label fields listed below. If the application program does not supply this information, the implementation must supply the information for the appropriate fields.

For each file in a file set:

- file name (HDR1, positions 5 through 21)
- record format (HDR2, position 5)
- block length (HDR2, positions 6 through 10)
- record length (HDR2, positions 11 through 15)

If the implementation permits the application program to supply the information to be processed in each of the label fields listed below, the implementation must process this information. If the application program does not supply this information, it must be supplied by the implementation.

For each file in a file set:

- generation number (HDR1, positions 36 through 39)
- version number (HDR1, positions 40 and 41)

For each file section of a file set:

- creation date (HDR1, positions 42 through 47)
- expiration date (HDR1, positions 48 through 53)

If the implementation is in a position to process a set of installation volume header labels (UVL), the implementation must permit the installation to supply the information to be processed from the label fields listed below. Processing of the corresponding labels is not requested if the installation does not supply the information.

For each label from a set of installation volume header labels recorded on each volume in a volume set:

- reserved for the installation (UVL, positions 5 through 80)

If the implementation is in a position to process a set of user file header labels (UHL) or of user file trailer labels (UTL), the implementation must permit the application program to supply the information to be entered in the label fields listed below for a label set.
Processing of the corresponding labels is not requested if the application program does not supply the information.

For each label in a set of user file header/trailer labels for a magnetic tape:

- label number (UHL/UTL, position 4)
- reserved for the user (UHL/UTL, positions 5 through 80)

The implementation can impose the following restrictions on the user with regard to the record length (HDR2, positions 11 through 15):

If the implementation is processing segmented records, it can define a maximum record length. This should not be less than the maximum permissible block length minus the length of the buffer displacement field and the length of the segment control word (SCW).

If the implementation is processing variable-length records, it can define a maximum record length corresponding to the maximum block length minus the length of the buffer displacement field and the length of the record length word (RLW).

Requirements for a receiving system

Files

The implementation must supply the application program with the contents of the records and the length of each record. The segment control word (SCW) and the record length word (RLW) are not part of the record.

Labels

The implementation must permit the user to supply enough information to enable him/her to select both the requested files and the volume on which they are recorded.

The implementation must supply the installation with the information contained in the following label fields:

For each volume in a volume set:

- volume identifier (VOL1, positions 5 through 10)
- volume access indicator (VOL1, position 11)

For each file in a file set:

- file access indicator (HDR1, position 54)

The implementation must supply the application program with the information contained in the following label fields:

For each file in a file set:

- file name (HDR1, positions 5 through 21)
- record format (HDR2, position 5)
- block length (HDR2, positions 6 through 10)
- record length (HDR2, positions 11 through 15)

The implementation need not supply the user with the information contained in the following label fields:

For each volume in a volume set:

- owner identifier (VOL1, positions 38 through 51)

For each file in a file set:

- file set identifier (HDR1, positions 22 through 27)
- generation number (HDR1, positions 36 through 39)
- version number (HDR1, positions 40 and 41)

For each file section of a file set:

- creation date (HDR1, positions 42 through 47)
- expiration date (HDR1, positions 48 through 53)

If the implementation is in a position to supply the installation with the information recorded in the set of installation volume header labels (UVL), the information contained in the following label fields must be provided:

For each label of a set of installation volume header labels:

- reserved for the installation (UVL, positions 5 through 80)

If the implementation is in a position to supply the user with the information recorded in the set of user file header labels (UHL) or user file trailer labels (UTL), the information contained in the following label fields must be supplied:

- label number (UHL/UTL, position 4)
- reserved for the user (UHL/UTL, positions 5 through 80)

DMS dummy sections (DSECTs)

In addition to the MF operand, DSECTs are available to the user program for certain operand lists and for dummy sections of DMS tables, FCB entries and catalog entries. The names and sizes of the individual fields are defined, but the order in which the parts appear may be subject to changes.

Users interested in the resolution of a specific dummy section can use a macro with the following format:

Operation	Operands
`makroname`	`[D][,prefix / *][,PARMOD=24 / 31]`

For the meanings of “macroname” see table on the following page.

Default value: no DSECT is generated; the letter I is used as the prefix.

D
Specifies that the macro is being used to generate a DSECT.

prefix

Prefix (1 character) to be placed in front of all names in the DSECT.

*****
Specifies that no prefix is to be used.

PARMOD

The PARMOD operand can be used with the DSECT macros IDFCB, IDMCB, IDFST, IDPPL and DMARD. It specifies which operand list of the DSECT is to be generated.

Default value: the value for the generation mode defined by the GPARMOD macro or preset in the Assembler.

= 24
The (old) 24-bit addressing-mode-dependent operand list is generated.

= 31
The operand list which is independent of the addressing mode is generated. The symbolic names may differ from those in the PARMOD=24 operand list.

Programming notes

- The generation mode can be set globally for all macros in a program with the GPARMOD macro. The PARMOD operand in the DMS macros overrides the default setting made with the GPARMOD macro or (if GPARMOD is not used) preset in the Assembler.
- All PARMOD specifications for an operand list (e.g. MF=E/L/D and the corresponding DSECT macros) and for a file must contain the same value.

DMS dummy program sections (DSECTs) for users:

Macro name	DSECT description
IDBPL	BTAM operand list
IDCAT	CATALOG (CATAL) operand list (old format)
IDCE	Catalog entry
IDCEG	Catalog entry, expansion for file generation groups
IDCEX	Catalog entry expansion
IDCHA	CHANGE (CHNGE) operand list
IDCOP	COPY macro operand list
IDECB	UPAM event control block
IDEE	Catalog entry, extent list
IDEMS	DMS error messages (see section "DMS error codes")
IDERS	ERASE macro operand list (for VERSION=0)
IDFCB	FCB (TU section; valid for all FCB formats)
IDFCBE	FCB extension (for 24-bit TU FCB)
IDFST	FSTAT macro operand list
DMAIMP	IMPORT macro operand list (old format)
IDMCB	EAM control block
IDOST	Information on open files (see the OSTAT macro, "OSTAT - Request information on open files")
IDPFL	FILE macro operand list (old format)
IDPFX	FILE macro operand list extension (old format)
IDPPL	UPAM operand list
IDREL	REL macro (RELEASE) operand list
IDVT	Volume label entry (VT = volume table)
IDVRF	VERIF macro operand list
DMADR	RDTFT output format (if LINK operand specified) (old format)
DMARD	RDTFT macro operand list (old format)

Formats of replaced macros

The following macros have been replaced by new ones. Although the old macros are still supported for reasons of compatibility, their functionality will not be extended. The formats of the old macros are shown below. The operand descriptions are the same as those of the new macros (see earlier).

COPY - Copy file

The COPY macro has been replaced by COPFILE. For a description of the new COPFILE macro and the corresponding COPY operands, see "COPFILE - Copy file".

Operation	Operands
COPY	**$pathname_1$,$pathname_1$** **[,SAME][,WRITE = REPLACE / NEW]** **[,BLKCTRL = IGNORE / CHECK]** **[,IGNORE = SOURCE / TARGET / (SOURCE,TARGET)]**

REL - Delete TFT entry

The REL macro has been replaced by RELTFT. For a description of the new RELTFT macro and the corresponding REL operands, see "RELTFT - Delete TFT entry".

Operation	Operands
REL	**MF = (E,adr / (r))**
	name[,KEEP][,UNLOAD][,MF = L]

Glossary

This glossary provides short definitions of some of the terms used in this manual.

Access Control List (ACL)

File protection with an ACL (Access Control List) has not been supported since SECOS V4.0. The file attribute ACL is still contained in the catalog entry, but normally it contains the value NO (no ACL protection).

If the case should occur that a file has ACL protection (this may be possible following recovery from a long-term archive), the file cannot be accessed. In this case, the owner can protect the file with GUARDS. The protection with GUARDS “overwrites” the ACL protection and thus makes the file accessible again.

access method

A conventional data management technique which defines, for the user, the data organization and the method of data transfer between I/O devices and the main memory of the system. Access methods supported by DMS are:

- EAM (Evanescent Access Method)
- SAM (Sequential Access Method)
- ISAM (Indexed Sequential Access Method)
- UPAM (User Primary Access Method)
- BTAM (Basic Tape Access Method)

Alias Catalog Service (ACS)

Files and job variables can be addressed by means of alias names and stored in special catalogs, the alias catalogs, together with their assignment to the real file/JV. The Alias Catalog Service (ACS) incorporates three basic functions: Alias name definition, catid insertion for temporary spool files and prefix insertion.

alphanumeric

The alphanumeric characters consist of *alpha*betic and *numeric* characters, i.e. the letters A-Z, the special characters $@#, and the digits 0-9.

batch mode

An operating mode in which a job is started with an ENTER-JOB or ENTER-PROCEDURE command; in contrast to interactive mode, the sequence of operations is predefined and stored in an ENTER file (started with ENTER-JOB) or in a procedure file (started with ENTER-PROCEDURE).

Batch processing

see Batch mode

Block

see PAM page; see data block

blocked record

A record in a file in which each data block may contain several records.

BS2000 file

Is a file which is only created and processed by BS2000. BS2000 files on Net-Storage (FILE-TYPE=BS2000) have been supported since BS2000/OSD-BC V9.0. They are located directly on a Net-Storage volume. Open systems may only access them in read mode.

buffer

A contiguous area in main memory into which data is written and from which data is read.

CALL procedure

Sequence of commands/statements executed within a job and called by means of the CALL-PROCEDURE command. For further details see the manual “Commands” [3].

catalog ID

The identifier of a pubset (see pubset ID); it is specified in a full file or path name in the form :catid:.

class 1 memory

That part of virtual user memory which is occupied by the main memory-resident modules of the Executive. All class 1 pages are marked as privileged and non-pageable. The pages are not mapped on paging memory. The pages remain in main memory throughout the session.

class 5 memory

That part of virtual user memory which contains the pageable areas allocated dynamically by the Executive when needed for a user job.

class 6 memory

That part of virtual user memory containing the user programs; space is allocated dynamically by the Executive.

data block

A block which contains one or more records of a file.

double tape mark

Two directly adjacent tape marks which indicate the logical end-of-tape. Two adjacent tape marks may also occur if the tape contains an empty file or an empty file section; in this case, they are not regarded as a double tape mark, but as two single tape marks.
“Empty” in this sense means that there are no data blocks between the tape marks before the file header labels and after the trailer labels.

FCB (file control block)

A file control block which contains all the information abuot a file which is needed for file processing.

file

Related records are grouped together in a named entity called a file. Typical examples of files are: conventional input and output data of programs: load module and object module libraries; text information which is created and processed with a file editor.

file link name

A name with up to 8 characters which establishes the link between the file control block and the file via the task file table.

file section

That part of a file which is stored on a single tape.

file set

A set of files recorded sequentially on one or more tapes. There must be no sections of other files between the sections of a file.

first in – first out (FIFO)

Queue structure according to which information is processed in the order in which it is input (in contrast to last in – first out, LIFO).

fixed-length record

A record in a file in which all records are declared as having the same length; no record length information is needed within the file.

foreign file

A foreign file is a file on a private volume or on a Net-Storage volume which is not cataloged on a pubset.

interactive mode

The operating mode in which a job is initiated at a (remote) terminal and executed; the sequence of processing is not predefined.

job

The totality of all operational sequences between the commands SET-LOGON-PARAMETERS or LOGON and EXIT-JOB or LOGOFF. For this definition, it is immaterial whether the job is already fully defined when it is submitted (batch mode) or whether the individual steps are defined in the course of execution (dialog mode).

label

A record at the beginning or end of a file or tape which is used to identify, describe and/or delimit the tape or file. A label is not regarded as part of the file. Each label is recorded separately in its own block (label block).

label group

An uninterrupted sequence of label sets which delimit a tape, a file section or a file.

label handling routine

A series of statements for the processing of labels.

label identifier

A three-character word which is recorded as part of the label and identifies this label.

label set

An uninterrupted sequence of labels with the same label identifier.

last-byte pointer (LBP)

Pointer to the last valid byte of the last logical block of a PAM file.

last-page pointer (LPP)

Pointer to the last PAM page occupied by a file. In the catalog entry it corresponds to the highest-used page.

locate mode

In locate mode, the user requests the address of the current record, which is stored in a buffer area. The user is responsible for the data transfer from and to the buffer.

logical block number (LBN)

Sequence number of a block in a file.

move mode

In move mode, the user specifies the position of the record in his program. The system is responsible for data transfer from and to the buffer.

net client

Implements access to Net-Storage for the operating system using it.
In BS2000 the net client, together with the BS2000 subsystem ONETSTOR, transforms the BS2000 file accesses to corresponding UNIX file accesses and executes them on the net server using NFS.
The net client for SUs /390 is the HNC, and for SUs x86 and SQ servers the X2000 carrier system.

net server

A file server in the worldwide computer network which provides storage space (Network Attached Storage, NAS) for use by other servers and offers corresponding file server services.

Net-Storage

Storage space provided by a net server in the computer network and storage space released for use by foreign servers. Net-Storage can be a file system or also just a node in the net server’s file system.

Net-Storage file

Is a file which is created on a Net-Storage volume. On Net-Storage a distinction is made between the two file types BS2000 file and node file.

Net-Storage volume

Net-Storage volumes represent Net-Storage in BS2000 which provides systems support as an enhancement of data pubsets.
As of BS2000 V21.0A, there is a second Net-Storage volume type NETVOL in addition to the previous Net-Storage volume type NETSTOR.
Both Net-Storage volumes of type NETVOL and type NETSTOR are addressed for compatibility reasons by means of their Volume Serial Number (VSN) and the volume type NETSTOR. In the released file system of the net server the VSN of the Net-Storage volume corresponds to the directory containing the user files and metadata.

NFS (Network File System)

BS2000 software product that permits distributed data storage in a heterogeneous computer network. It enables users to access remote files as if they were residing on their local computer.
NFS is thus used for connecting systems. Furthermore, the automatic and reliable BS2000 data saving functions can be made available for the files of such systems via NFS.

node file

Is a Net-Storage file (FILE-TYPE=NODE-FILE) which can be created and processed by both BS2000 and open systems. Node files are supported in BS2000 OSD/BC V10.0 and higher. They are located on a Net-Storage volume in a user-specific directory (name of the user ID), and the file names comply with the BS2000 naming conventions.

null file

A file which is logically empty, i.e. a file which is cataloged and to which the system has allocated space, but contains no data.

PAM page

A storage unit of 2048 bytes on disk with or without (depending on the file type) a PAM key; also called a “standard block” for tape files with a PAM key.

privileged mode/program

All parts of the operating system that are not executed in the TU processing state (see function states).

procedure/procedure file

A file which contains a predefined sequence of commands or statements used for program input. Procedures are started with CALL-PROCEDURE or ENTER-PROCEDURE. Only if the file contains an ENTER job will it be started with ENTER-JOB. See the “Commands” manual [3] for details.

public volume set (PVS)

Term previously used for: pubset; seepubset

pubset

A set of disks designated as public. MPVS systems work with several mutually independent pubsets.

pubset Id

The pubset identifier. If the volume serial number of a public volume begins with the three characters “PUB”, the pubset Id is the fourth character; if the volume serial number contains a period, the pubset Id is formed by the characters preceding the period. In the path name the pubset Id is specified in the format :catid: (see ” catid”).

PVS Id

see pubset-Id

record

A group of data items which are treated as a logical unit.

record format

The definition of the length and segmentation of records for a file.
Record format V: if a format V tape is written with non-EBCDIC code, the length is specified in hexadecimal.
Record format D: if a format D tape is written with ISO 7-bit code, the length is specified as a four-digit decimal number.

shareable file

A file which is cataloged with the USER-ACCESS=ALL-USERS. A shareable file can be accessed from all user IDs provided the other protection attributes (e.g. file passwords) of the file permit this.

spoolout

Automatic spoolout: automatic output of the contents of system file SYSLST to a printer or sending it by email at LOGOFF time (EXIT-JOB or LOGOFF)

SYSFILE environment

see system files; the SYSFILE environment consists of all the system files assigned to a job.

system files

System input/output files that are assigned to a job.
The (default) file names SYSDTA, SYSSTMT, SYSCMD, SYSIPT, SYSLST, SYSLST01, SYSLST02, ..., SYSLST99, SYSOPT and SYSOUT refer to the (system) files for data and command input to the operating system or for data output by the operating system. Each of these files is created by the task and specifies predefined I/O areas.
The user can cancel the primary assignment and assign the (default) file names to his/her own cataloged files or compound S variables (when the software product SDFG-P is used).
For detailed information on system files, see the “Commands” manual [3].

tape (volume)

An exchangeable unit of the data storage medium magnetic tape or magnetic tape cartridge. A tape may contain all or part of one file, several files and/or one or more file sections.

tape mark

A tape block which marks the boundary between data blocks and the tape label groups and between certain label groups. The format of the tape mark is defined in the related standards for magnetic tapes.

task sequence number (TSN)

The sequential number assigned by the system to the task (or job) with which the user can identify a task in some commands.

task file table (TFT)

A table created using the LINK-NAME operand of the ADD-FILE-LINK command and from which the file and processing attributes are taken and entered in the file control block when a file is opened.

tape set table (TST)

A table which shows the tapes requested for a job (together with the TFT).

unblocked record

A record in a file in which each data block may contain only one record.

variable-length record

A record in a file in which the records may have different lengths. The record length must be specified in the first word in each record. This record length field is included in the length of the record; it is 4 bytes long and contains the record length left-aligned in decimal or hexadecimal.

volume serial number (VSN)

A six-character string assigned to the volume when it is initialized. It is kept in the standard volume label and is used to identify the volume. The VSNs of public volumes can be recognized by the fact that they either begin with the three characters “PUB” (the fourth character is then the pubset ID or in SM pubsets the volume set ID) or they contain a period (the characters before the period then form the pubset ID or the volume set ID).

volume set

The tape or tapes on which the files of a file set are stored.

volume sequence number (VSEQ)

The number of a file section in a multivolume file.

volume table of contents (VTOC)

The file directory in the F1 label of a private disk or on a Net-Storage volume.

Related publications

You will find the manuals on the internet at *http://bs2manuals.ts.fujitsu.com*.

[1] **BS2000 OS DX**
Introductory Guide to DMS
User Guide

[2] **BS2000 OS DX**
Executive Macros
User Guide

[3] **BS2000 OS DX**
Commands
User Guide

[4] **SPOOL** (BS2000)
User Guide

[5] **DAB** (BS2000)
Disk Access Buffer
User Guide

[6] **RFA** (BS2000)
Remote File Access
User Guide

[7] **BS2000 OS DX**
Introduction to System Administration
User Guide

[8] **SECOS** (BS2000)
Security Control System - Access Control
User Guide

[9] **ARCHIVE** (BS2000)
User Guide

[10] **HSMS** (BS2000)
Hierarchical Storage Management System
User Guide

[11] **HIPLEX MSCF** (BS2000)
BS2000-Processor Networks
User Guide

[12] **Introduction to XS programming**
(for ASSEMBLER programmers) (BS2000)
User Guide

[13] **PERCON** (BS2000)
User Guide

[14] **BS2000 OS DX**
Utility Routines
User Guide

[15] **DRV** (BS2000)
Dual Recording by Volume
User Guide

[16] **BS2000 OS DX**
System Installation
User Guide

[17] **SDF-A** (BS2000)
User Guide

[18] **SDF-P** (BS2000)
Programming in the Command Language
User Guide

[19] **BS2000 OS DX**
Files and Volumes Larger than 32 GB
User Guide

[20] **RSO** (BS2000)
Remote SPOOL Output
User Guide

[21] **JV** (BS2000)
Job Variables
User Guide

[22] **XHCS** (BS2000)
8-Bit Code Processing in BS2000
User Guide

[23] **BS2000 OSD/BC**
System Managed Storage
User Guide

[24] **FUJITSU Server BS2000 SE Series**
Operation and Administration
User Guide

[25] **CRYPT** (BS2000)
Security with Cryptography
User Guide

[26] **VM2000**
Virtual Machine System
User Guide

[27] **MAREN**
Tape management in BS2000
User Guide