LEASY V6.2A

Program Interface and Strategies

Edition March 2007

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 manuals@fujitsu-siemens.com.

Certified documentation according to DIN EN ISO 9001:2000

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 9001:2000.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Contents

1 Preface

1.1 Brief product description

LEASY (German acronym for linear input/output system) is a transaction-oriented data management and access system which is run under BS2000.

It provides various security and backup features designed to ensure file consistency.

LEASY supports the following requirements

- simple and uniform access to DMS files
- secondary keys
- transactions
- data security

Files can be accessed from COBOL or Assembler programs. The interface complies with KLDS, the standard for compatible interfaces to linear database systems.

LEASY can be used in timesharing mode (TIAM/Batch) and in transaction mode (openUTM, DCAM).

1.2 Target group

This manual is intended for organizers, developers and administrators of LEASY applications.

Different parts of this manual will be of interest to the members of the various user groups: the differences are explained on the next page. In general, however, users will require the following:

- In order to use LEASY you will need a general knowledge of BS2000, including its data management system (DMS).
- If you are working with LEASY via the COBOL interface you will need to be familiar with the COBOL programming language; if you are using the Assembler interface you will need to be familiar with the macro assemblers and the BS2000 system macros.
- If you are operating LEASY in connection with openUTM, in a DCAM environment or in a multiprocessor system environment, you will need to be familiar with openUTM, DCAM and MRS/RFA respectively.

1.3 Summary of contents

The LEASY software product is described in three manuals:

- LEASY Program Interface and Strategies
- LEASY Utility Routines
- LEASY Ready Reference

The **LEASY Utility Routines** manual describes the LEASY utilities. It is intended primarily for organizers and administrators of LEASY applications.

The **LEASY Ready Reference** manual is aimed at application developers and administrators of LEASY. By bringing together all the LEASY commands and operands and various tables in one handy volume it should make working with LEASY much easier.

This manual, **LEASY Program Interface and Strategies**, provides an overview of the LEASY software product. It contains a detailed description of how to program LEASY applications.

- **Developers** of LEASY applications will find all the information they need in this manual.

  In chapter “Overview of the LEASY program interface” on page 119ff you will find a description of the LEASY program interface independently of the programming language. It is important to read and understand this chapter before proceeding to chapter “COBOL interface” on page 191ff and chapter “Assembler interface” on page 217ff.

- **Administrators** can consult chapter “Basic features of LEASY” on page 13ff for a summary of the most important features and functions of the product. You are also advised to make use of chapter “Technical data” on page 419.

- **Organizers** will find chapter “Basic features of LEASY” on page 13ff and chapter “Save facilities” on page 49ff of particular interest. You will find additional important information in chapter “File management” on page 23ff and chapter “Operating modes” on page 83ff.

First-time users

In chapter “Basic features of LEASY” on page 13ff you will find a brief introduction to LEASY and a description of its most important performance features. If you are new to LEASY, you should read this chapter first.

In chapter “Sample applications” on page 323ff you will find correlated examples of the COBOL and Assembler interface and of how to use the LEASY utilities.

The remaining chapters can be read independently of one another. If other passages have to be read in order to understand a particular section, you are referred to the relevant parts of the manual.

Experienced LEASY users

The section beginning overleaf will introduce you to the major new features of Version 6.2A. You will also find direct references to the sections which provide in-depth information on these new functions.

1.4 Changes since the previous manual

Compared to the manual for LEASY V6.1A, this manual covers the following new features which have been incorporated in LEASY Version 6.2A:

- Controlled release of AIM file generations

  AIM file generations are by default protected against being deleted. Before they can be deleted, they must be released. They are released either automatically by LEASY after the shadow files have been automatically reconstructed, or they must be released by the LEASY administrator using the *AIMA* function of the LEASY-MASTER utility routine when the files are no longer required, see page 58.

- Replacing original files by shadow files during ongoing operation

  The *REPO* function of the LEASY-MASTER utility routine enables files to be replaced by their shadow files without terminating LEASY operation, see page 78.

- Online save

  The *ROMS* function of the LEASY-MASTER utility routine enables a write lock to be assigned to files. These files can then be saved during ongoing operation, see page 80.

- Additional information in the diagnosis file, see page 415.

1.5 Notational conventions

In continuous text the names of commands, operands, files, paths and screen elements are shown in *italics*. Important terms and contrasting pairs appear in **bold**.

Message texts and examples of system outputs are shown in a `fixed-pitch font`.

Texts that you must enter are shown in **`bold fixed-pitch font`**.

⚠ This symbol indicates an important warning which you must heed in the interest of system security or operational reliability, otherwise there may be gaps in the security coverage, loss of data or blockage of computers or lines.

ℹ This symbol precedes an important message which you must heed in the interest of system security or operational reliability.

Where references are made to other publications, the titles are abbreviated. The complete title of each publication referred to is given under “Related publications” section.

The following conventions have been employed in the manual for the formal representation of the statements and their operands:

Formal representation	**Explanation**	**Examples**
UPPERCASE LETTERS and special characters	Uppercase letters and special characters indicate constants which must be entered by the user in exactly this form	*CAT file catalog
lowercase letters	Lowercase letters indicate variables which the user must replace by current values.	The user enters: *CAT TESTCAT
$\left\{ \quad \right\}$	Braces enclose alternatives, i.e. one of the specifications must be selected.	$\left\{\begin{array}{l}\text{file}\\ \text{file.suffix}\\ \text{file.}\end{array}\right\}$ The user enters: FILE1 or FILE1.Z1 or FILE1.
[ ]	Square brackets enclose optional specifications.	keyname[,iub] The user enters: KEY1 oder KEY1,X' 00'
...	Dots indicate a repetition; the preceding syntactical unit can be repeated several times in succession.	(pos,len),... The user enters: (12,4) or (12,4),(14,10),(25,2)
_	Underlining indicates the default value. This is the value set by the utility routine if no specification is made by the user.	INF= $\left\{\begin{array}{c}\text{Y}\\ \underline{\text{N}}\end{array}\right\}$ The user enters: INF=Y or INF=N or nothing (i.e. same as INF=N)

Table 1: Notational conventions

2 Basic features of LEASY

This chapter describes the way in which LEASY works and its most important performance features.

2.1 Preparations for and execution of a LEASY session

To prepare for a LEASY session the first thing you must do is create a LEASY catalog with the **LEASY-CATALOG** utility routine.

With LEASY-CATALOG you can then

- enter new files in the LEASY catalog and create them physically on disk or tape
- add existing DMS files to the LEASY catalog
- delete files from the LEASY catalog
- define the type of security for individual files; a basic distinction is made between BIM (before-image) and AIM (after-image) security.

Once these preparations have been completed for the LEASY session the **LEASY-MAINTASK** utility routine is used to start the main LEASY task. The main task sets up common memory. Common memory (CMMAIN) is the memory area used by all the tasks connected to a particular LEASY catalog. During a LEASY session the main LEASY task processes the requests from the user programs. The term ’LEASY session’ is taken to mean the period between setting up and releasing common memory.

The LEASY-MAINTASK utility routine is also used to make the following settings:

- global validity for file security
- boundary parameters for security (e.g. storing the AIM file on a private disk)
- parameters for common memory (e.g. size).

Once common memory has been set up, the user can start the program. The LEASY runtime system is linked in the **user program**. Users work with LEASY via a predefined interface (COBOL or Assembler interface). Via this interface they can initiate operations for processing files and for controlling **transactions** (see page 15).

Execution of a LEASY session is monitored and controlled with the aid of the **LEASY-MASTER** utility routine.

Monitoring is performed with the aid of the show functions. These provide data on

- the status of the files
- the status of the transactions
- the status of common memory
- the number and type of LEASY operations performed.

The LEASY session and the transactions can be influenced by the following control functions:

- suspending, resuming and terminating a session
- suspending, resuming and resetting a transaction
- changes to the AIM security procedure
- locking and unlocking files.

File processing and the transactions are terminated in the user programs with the appropriate operations.

The LEASY session is terminated by means of the LEASY-MASTER utility routine.

2.2 Performance features

This section summarizes the performance features of LEASY.

Transaction processing

The transaction concept is used to provide consistent units of data. A number of actions form a transaction and should be executed either in their entirety or not at all.

A **LEASY transaction** is a sequence of LEASY operations between the *OPTR* operation (start of transaction) and *CLTR* (end of transaction). These operations are issued by a user program.

A basic requirement for the logical and physical consistency of transactions running in parallel is the ability to roll back an incomplete transaction without affecting the updating of other transactions. A rollback can be effected:

- by making an explicit request in the calling program (see the LEASY operation *CLTR* with the additional function *OPE1=R*)
- by making an explicit request using the LEASY operation *BACK*
- by making an explicit request using the *RLBT* function in the LEASY-MASTER utility routine
- by calling up an STXIT routine, e.g. after an abnormal program termination
- by using the LEASY-MAINTASK utility routine to warm start the system.

Resetting a transaction is always performed with the aid of transaction-related before-image files. Therefore these files must be specified in the LEASY-CATALOG and LEASY-MAINTASK utility routines.

BIM: transaction-oriented data saving

If an error occurs during the execution of a transaction, i.e. the transaction cannot be completed, all updates to the files involved must be rolled back. This is done by providing each transaction with a BIM (**B**efore-**IM**age) file.

The BIM file contains the information on the file access operations performed since the last restart. When a transaction is rolled back the information from the BIM file is used to:

- delete any records or blocks which have been inserted
- insert any records or blocks which have been deleted
- cancel any updates made in records or blocks.

If the transaction was successfully completed or rolled back, the BIM file is defined as "empty".

Automatic restart after a system crash

If a warm start (automatic restart) is performed after a system crash, any open transactions are rolled back. LEASY checks whether there are any BIM files that are not logically empty. Such files are then processed and logically deleted. The files are retained. BIM files are then initialized for the new session.

AIM: file-oriented data saving

In order to be able to reconstruct the current data set of a damaged file all updates of the original file must be logged in a separate file. This is done by keeping **A**fter-**IM**age files. Each LEASY catalog is assigned an AIM file generation group into which the data of updated records is written by all the programs connected to that catalog.

Corrupted files are reconstructed by reading in save files into which the contents of the AIM file are incorporated using the LEASY-RECONST utility routine.

Shadow file support

Shadow files are copies of original files and are updated continuously parallel to processing of the latter. The additional use of shadow files in LEASY enables round-the-clock operation and a quick restart after a physical file breakdown.

To update the shadow files, a switchover occurs during the LEASY session from one AIM file generation to the next, with the shadow file being updated by the closed AIM file with the aid of the LEASY-RECONST utility routine. If the contents of the current AIM file are kept to a minimum, there is only a slight difference between the shadow file and the original. This enables rapid reconstruction of a damaged file.

The AIM file is switched over either by specifying a maximum size in LEASY-MAINTASK (_*AIS_ operand) or under control via the LEASY-MASTER utility routine.

Secondary indexing

In addition to **primary keys**, up to 255 **secondary keys** with partial keys can be defined for accessing each ISAM, DAM or PAM file.

LEASY uses the following procedures: An additional (ISAM) file, the **secondary index file** (SI file), is created for each ISAM, DAM or PAM file for which one or more secondary indices are defined. This file contains pointers from the secondary key value to the primary key value. A secondary index may comprise a number of components (partial keys) and/or can be provided with a repetition factor (multiple secondary index).

The validity of secondary index definitions can be made dependent on the contents of a record type field. The record type field is defined by the user.

LEASY provides two utility routines for secondary index management:

LEASY-CATALOG	For defining the name, position and length of the secondary indices and the record type fields as well as allocating the secondary keys to specific record types. It is also possible to define that changes in the primary file automatically lead to changes in the pointers in the secondary index file.
LEASY-LOADSI	For creating, inserting and deleting secondary index pointers in secondary index files.

The name of a secondary key to be used in direct accessing or positioning can be specified in the LEASY program interface.

Internal linkage with openUTM

When LEASY is linked with openUTM it can also be implemented in inquiry and transaction mode. In a LEASY-openUTM application data security is ensured by the common transaction concept of LEASY and openUTM.

LEASY in a DCAM environment

LEASY provides a DCAM connection for special functions in inquiry and transaction mode which extend beyond the range of functions of openUTM.

DRIVE-LEASY

The LEASY functions can also be accessed via the 4th generation programming language DRIVE. DRIVE-LEASY can also be used under openUTM, but not with DCAM.

3 Locking strategy

LEASY offers a hierarchical, two-level locking strategy for protecting files or records against unauthorized access in parallel tasks or transactions.

3.1 File locking

The **OPEN mode** indicates the type of file opening, as performed by DMS.

The **USAGE mode** indicates the type of file usage in a transaction. The possible USAGE modes depend on the entry for the OPEN mode. Conversely, in the case of implicit file opening (i.e. *OPFL* is not specified before *OPTR*), the OPEN mode is determined from the USAGE mode. Both modes specify

- whether the file is to be created as a new file or whether an existing file is to be processed
- whether the file may be processed with read-only access, write-only access and in one or more tasks.

Both modes should be specified with no more restrictions than necessary since exclusive, protected or shared update access rights may affect and hamper parallel tasks or transactions. Files and transactions should be closed again at the earliest possible opportunity.

3.2 Record locking

LEASY implements record locking for master and model files using the ISAM, DAM and PAM access methods if they have been opened with USAGE modes *UPDT, PRUP, RETR, LDUP* or *PLUP*. Limited coordination via the ISAM and PAM locking mechanisms is possible for LEASY foreign files using the ISAM, DAM and PAM access methods. These mechanisms are not, however, automatically transaction-oriented.

As regards the locking operation, a distinction is made between:

- **implicit locks** which are set automatically for the individual record in the *INSR* and *STOR* operations, and
- **explicit locks** which are set in the *LOCK, RHLD, RNHD* and *RPHD* operations for the specified key or the specified key range.
  "Phantom" locks can also be set explicitly; these are locks on records which do not (yet) exist.

As far as the purpose of the lock is concerned, a distinction is made between READ-LOCK and WRITE-LOCK.

- **READ-LOCK**

  This is assumed for explicit locks in files which have been opened with the USAGE mode *RETR* and for explicit locks with the operation code extension *OPE1=S*.

  The READ-LOCK can be set by several transactions and enables records to be protected against being overwritten in parallel transactions.

  To protect against deadlocks LEASY has a deadlock detection procedure. In the event of a deadlock, transactions are mutually blocked so that they cannot continue.

  Example

Transaction A	locks record 1
Transaction B	locks record 2
Transaction A	wants to lock record 2, but must wait
Transaction B	wants to lock record 1.

  A deadlock occurs.

LEASY detects these deadlock situations and prevents them. The last lock requested (on record 1 by transaction B) is rejected with return code *RC-LC L007*; if necessary, the transaction can be rolled back.

- **WRITE-LOCK**

  This is assumed for implicit and explicit locks without the operation code extension *OPE1*.

  The WRITE-LOCK can only be set in a transaction on an exclusive basis. It must be set if a record is to be updated. The following diagram shows the possible combinations of locks in two transactions (T1 and T2):

T1 T2	READ-LOCK	WRITE-LOCK
READ-LOCK	YES	NO
WRITE-LOCK	NO	NO

Canceling locks

A READ-LOCK or a WRITE-LOCK which refers to a record that has not yet been inserted, updated or deleted in the transaction can be canceled with the LEASY operation *UNLK*.

Record locks are canceled at the end of a transaction (operation *CLTR*).

Access without lock log

The usage modes *ULRT* (Unlocked Retrieval) and *ULUP* (Unlocked Update) permit multiple read accesses and a single write access in parallel without the necessity for a lock log.

These two usage modes cannot be used with SAM files.

Lock log

LEASY implements the record locks by managing a lock log in common memory. Each element of the lock log contains, among other things, the type of lock and the primary key of the locked record.

The elements of the lock log are displayed using the *SHLE* (SHow Lock Elements) function of the LEA.MASTER utility routine.

The following table illustrates how various actions affect the lock log. The notes on the page following the table must also be borne in mind.

Action	**Meaning**	
LOCK, RHLD, RNHD,RPHD	Individual records of an ISAM, DAM or PAM file can be locked (explicit locking).	
LOCK, RHLD	Individual record ranges of an ISAM, DAM or PAM file can be locked.	
LOCK	The *LOCK* operation also serves to lock records/record ranges that do not (yet) exist (so-called phantoms).	
INSR, STOR	Inserted records are automatically locked (implicit locking).	
INSR, STOR, REWR, DLET	Locks on updated, inserted or deleted records are automatically retained until the end of the transaction and cannot be canceled by *UNLK*. If the record is contained in a lock range, LEASY generates an additional lock element for this record. Although the range can then be released by means of *UNLK*, the additional record locks are retained until the end of the transaction.	
DLET, REWR	Records to be deleted or updated must first be locked (implicitly or explicitly).	
UNLK	Locked but not updated records/record ranges are released and, if *OPE1=‘U‘* is specified, updated as well.	
CLTR	All locks are canceled automatically at the end of the transaction (except for foreign files).	
"unprotected read"	Locked and updated records can be read by other transactions (*RDIR, RNXT, RPRI*). This so-called "unprotected read" is authorized to permit a higher degree of parallel transaction processing.	
Initialization of the main LEASY task with the operand **TIME*	The timeout for release of records locked by parallel transactions can be globally specified with this operand.	If timeout occurs without success, a return code informs the user. The locking attempt is repeated at one-second intervals.
OPE-WTIME field in the *RE* area	Timeouts for the release of locked records can be specified for each operation.	

Table 2: Effects of different actions on the lock log

The number of record locks should be kept to a minimum. This is achieved by:

- short transactions
- setting record locks as late as possible within the transaction
- canceling locks as early as possible with *UNLK*
- implementing write operations within the transaction as late as possible.

4 File management

In the LEASY system, processed files can be assigned further file attributes in addition to the DMS attributes. They are entered in a **LEASY catalog** to facilitate management and to enable LEASY to access the files. One or more completely independent LEASY catalogs can be set up under a single user ID.

4.1 Setting up a LEASY catalog

The LEASY catalog is set up by means of the **LEASY-CATALOG** utility routine. It is an ISAM file bearing the name

:catid:$userid.file-catalog.LEASYCAT

This is the DMS name of the LEASY catalog. It is also known as the **physical file name**. Of this name, the user may select only the *file-catalog* part, which is also referred to as the **logical file name** of the LEASY catalog.

The LEASY catalog is cataloged under the user ID used to start the LEASY-CATALOG utility routine. The files managed by the file catalog are created under the same user ID unless another user ID is specified explicitly by entering a DMS name (see the *NAM* operand of the **FIL* statement in the LEASY-CATALOG utility routine).

The LEASY catalog can also be generated via a *CREATE-FILE* command prior to calling the LEASY-CATALOG utility routine. If the catalog has already been opened and closed before being called by LEASY-CATALOG for the first time, the operation

**CAT file-catalog,TYP=N*

is rejected.

The LEASY catalog is protected by an internal DMS write password and can only be modified or deleted via the LEASY-CATALOG utility routine.

4.2 Structure of the LEASY catalog

The LEASY catalog is an ISAM file with the following attributes:

RECORD-FORMAT=V
KEY-POSITION=5
KEY-LENGTH=29
BUFFER-LENGTH=STD(SIZE=8)

During generation, the following entries are made in this file:

- The first record of the file contains management information, such as password, last session number or specifications for forming the names of shadow files.
- There is a management record for each BIM file and for the AIM file.
- A record containing the logical and physical file names, information about the file structure and the LEASY file attributes is created for each file entered in the LEASY catalog.
- If secondary keys are defined for a file, all secondary index definitions are also stored in the record (in the case of model files, only for the model and not for the individual instances).
- File entries for the instances of a model file group are then made for model files. The structure of these records corresponds to the file entry of the model, but here the file name must include the suffix.

4.3 File types supported by LEASY

LEASY supports four different logical file types, each of which can be organized according to the SAM, ISAM, PAM or DAM access method. These are:

- master files
- model files
- temporary files
- foreign files

The main difference between these file types is their lifetime. In addition to DMS attributes they also have special LEASY file attributes.

4.3.1 Master files

Master files are long-term files; in view of their purpose there is only ever one example of a master file. They permit all the functions of LEASY to be utilized, such as secondary indexing, multiple accessing, etc.

The complete set of master files in a file catalog has great similarity with a database, with independent record types stored in different realms.

When master files are entered in the file catalog, they are created or transferred by the LEASY-CATALOG utility routine.

If the LEASY-CATALOG utility routine explicitly assigns a DMS name to the master file, that particular name is used; otherwise the DMS file name (physical file name) is formed as described below.

:catid:$userid.file-catalog.file

where

catid	catalog identifier
userid	user ID under which the LEASY catalog is maintained
file-catalog	name of the LEASY catalog
file	name of the master file (logical file name)

4.3.2 Model files

A LEASY model file group consists of a model and one or more instances. All file attributes that are to apply to the instances are defined for the model. The model itself contains no data.

Instances of a model are those files having the same attributes and name as the model. The instances are given different suffix names.

The LEASY model file system allows a LEASY application to be run optionally on different data sets having a common structure. By entering a suffix name in conjunction with the CATD operation the user can specify which instance in the model file group is to be processed (e.g. in a payroll system the files for particular firms, accounting periods, etc.).

Creating a model file group

A model file called

:catid:$userid.file-catalog.file

is created first of all in the file catalog by the LEASY-CATALOG utility routine and the file attributes defined. Then the individual instances, with the same logical file name (*file*) as the model but with different suffix names (*suffix*), are entered in the catalog:

:catid:$userid.file-catalog.file.suffix

Internally, LEASY copies a duplicate of the model file to the new instances.

4.3.3 Temporary files

Temporary files resemble model files in that a copy with the file attributes is created in the LEASY file catalog. In contrast to master and model files, however, temporary files are generated dynamically when first used in the task; they are always set up on public volumes under the user ID of the user task. This method is particularly useful for transfer files between programs.

Temporary files are set up for individual tasks and are therefore always exclusive, i.e. multi-accessing is not possible. They are recataloged for each new TSN (task sequence number) and should be deleted again by a *DELETE-FILE* command before the task is terminated. This applies only if the task is correctly terminated; if errors occur, the files must be retained for a restart.

Name structure for temporary files

$userid.LEAT.tsn.file

where

userid	task user ID
tsn	task sequence number
file	name of the temporary file

An empty file bearing the following name:

$userid.file-catalog.file

is created in the DMS catalog to store the file attributes.

4.3.4 Foreign files

A file is defined as a LEASY foreign file if it is entered in the LEASY catalog with the **FIL* statement (operand *LEA=F*). If common memory *CMMAIN* is not employed, **all** files are considered as foreign files.

Note that LEASY secondary keys cannot be defined for foreign files.

An example of a foreign file would be the file of a remote user ID.

When a foreign file is defined in the LEASY catalog the DMS attributes specified are *ACCESS-METHOD* and *BLOCK-CONTROL-INFO*. All other DMS operands are rejected. Those DMS attributes absent during execution are taken from the TFT entry of a *ADD-FILE-LINK* command or from the DMS catalog.

Accessing foreign files

Unlike other file types, foreign files are accessed via the file link name:

ADD-FILE-LINK LINK-NAME=file-link-name,...

Consequently, before an existing file is accessed, a *ADD-FILE-LINK* command specifying the file name, file link name and *ACCESS-METHOD* must be issued. *RECORD-FORMAT=FIXED(RECORD-SIZE=...)* must also be specified for DAM files.

When creating a new file, the *ACCESS-METHOD* and the appropriate *OPEN-MODE* operand (*OUTIN* or *OUTPUT*) must be specified in the *ADD-FILE-LINK* command after creating the catalog entry with the *CREATE-FILE* command. All other attributes which do not correspond to the DMS default values must also be specified.

For DAM files,

```
RECORD-FORMAT=FIXED(RECORD-SIZE=...), ACCESS-METHOD=UPAM,BUFFER-LENGTH=...
```

and all other file attributes must be specified in addition. The value specified in the OPEN-MODE operand has priority over the OPEN mode of the *OPFL* statement.

Opening foreign files with SHARED-UPDATE=NO

The order in which LEASY operations may be performed is subject to the same rules as for master files.

Opening foreign files with SHARED-UPDATE=YES

Foreign files are opened with *SHARED-UPDATE=YES* in the *ADD-FILE-LINK* command. If the *SHARED-UPDATE* operand is not specified in the *ADD-FILE-LINK* command, LEASY refers to the OPEN/USAGE mode to determine whether the files are opened with or without *SHARED-UPDATE*. Foreign files for which the LEASY BIM save method is required must be opened with *SHARED-UPDATE=YES* for reading only.
If the files are opened with *SHARED-UPDATE=YES*, the following special rules apply:

- The LOAD USAGE modes (*LOAD, LDUP, EXLD, PLOD, ELOD, PLUP* and *ELUP*) are not supported as no synchronization is available with regard to the highest key number (ISAM), the highest relative record number (DAM) or the highest PAM block number (PAM).
- Updating an ISAM record requires the record to be read with locking directly before the update, since the lock is mapped to the ISAM lock mechanism, i.e. the following sequence of operations is necessary:

  ```
  RHLD/RNHD/RPHD
  REWR
  ```

  The operations *STOR, INSR, REWR, DLET, SETL, RDIR, RHLD, RNXT, RNHD, RPRI, RPHD* and *UNLK* are supported, but not the *LOCK* operation.

  ISAM can retain only one lock; it cancels the ISAM lock at each subsequent macro. LEASY does not automatically cancel the ISAM lock at the end of the transaction.
- For accessing PAM files with *SHARED-UPDATE*, LEASY employs the option for PAM block locking by UPAM instead of the LEASY lock log in common memory CMMAIN.

  In the case of the LEASY operations *RHLD, RNHD* and *RPHD*, the PAM blocks are read using the PAM operation code *LRDWT* (read with locking). For the *REWR, DLET, INSR* and *STOR* operations, the blocks are written using the PAM operation code *WRTWU* (write and cancel lock).

  LEASY checks that the required lock has been set before a block is updated. The user can, therefore, by programming correctly (by means of the LEASY operation sequence *RHLD/RNHD/RPHD* followed by *REWR/DLET*), effect a protected update.

  LEASY supports the operations *LOCK* and *UNLK* on a specific block. These operations are mapped to the PAM operation codes *LOCK* and *UNLOCK*.

DMS restrictions with regard to the locking of PAM blocks

- Up to 255 PAM blocks can be locked at the same time for each task.
- Neither LEASY nor DMS-UPAM is aware of the current locking situation and therefore LEASY is also unable to cancel PAM locks that may remain at the end of a transaction. A physical closure causes the release of all PAM locks of the file to be closed.
- If a PAM block is locked by a task, a further lock request can create a deadlock which can be recognized by neither LEASY nor UPAM. After timeout UPAM sends DMS error code *09B0*, which LEASY maps to error code *99ALL007*. In this case, the user must cancel all PAM locks on this file. After timeout for the first PAM block to be locked, UPAM supplies DMS error code *09B1* which LEASY maps to error code *99ALL006*.
- LEASY passes on the waiting time stored in the *OPE-WTIME* field of the RE area to UPAM.

4.4 File access methods

LEASY supports the following four file access methods:

- sequential access method (SAM)
- indexed sequential access method (ISAM)
- block-oriented primary access method (PAM)
- direct access method (DAM)

Each access method has a number of extensions relating to file access with high-level programming languages and a number of restrictions relating to the BS2000 DMS macro interface.

4.4.1 Sequential access method (SAM)

The access method supported by LEASY for sequentially organized files largely corresponds to the SAM access method of DMS in BS2000 (see the manual “Introductory Guide to DMS”).

Extensions

- The LEASY operation *SETL* can be used to position to a particular record in the file.
- A record can be read directly using the *RDIR* operation. The SAM retrieval address must be specified upon calling.
- Certain USAGE modes (see page 184ff) allow reverse reading of sequential files, i.e. the file is read sequentially in the direction of the start of the file.
- The SAM retrieval address is returned when read and write operations are performed. (see the LEASY operations *RNXT* and *RPRI*).
- 4 or 8 byte SAM retrieval addresses:
  SAM retrieval addresses in AIM and BIM files are stored exclusively in 8 byte format. Transaction rollback and restoring positions within SAM files are fully supported. SAM retrieval addresses can be specified by the user in either 4 or 8 byte format. The 4 byte addresses can be specified in the form *’bbbbbbrr’* where *bbbbbb* = block number and *rr* = record number and 4 bytes each must be used for the block and record numbers with 8 byte addresses.

  If the user specifies 4 byte addresses, a maximum of 255 records are addressable per block. If the number of records in a block exceeds 255 (e.g. via insert operations (*INSR*)), LEASY switches internally to 8 byte addresses. This procedure allows more than 255 records in a block. In this case, the user receives an 8 byte address and the return code 99ALL011.

Restrictions

- Record format *U* (undefined length) is not supported.
- SAM files extending over more than one tape are not supported.
- DMS does not return the SAM retrieval address in the case of files with non-standard blocks. Therefore, LEASY functions which relate to the retrieval address of a record cannot be executed for these files. The operations in question are *OPTR* (*OPE1=W*) and *CLTR* with rollback, and *SETL* and *RDIR*.
- File generations are not supported.

4.4.2 Indexed sequential access method (ISAM)

The access method supported by LEASY for indexed-sequentially organized files largely corresponds to the ISAM access method of DMS in BS2000.

Extensions

- If an ISAM file is processed with LEASY, it is possible to read backwards against the key order (primary and secondary keys).
- LEASY secondary keys can be defined for ISAM files:
  - a maximum of 255 secondary keys per file
  - a maximum of 253 secondary key parts per secondary key
  - a maximum of 512 secondary key parts per file.

  The length of the longest LEASY secondary key + the length of the ISAM primary key of each file must be equal to or less than 254.

Restrictions

- Record format *U* (undefined) is not supported.
- The ISAM key must always be unique; duplicate primary keys are not permitted.
- The BS2000 ISAM lock mechanisms are available to the user for foreign LEASY files only. LEASY provides the transaction-oriented lock log instead.
- LEASY cannot be used to access logically flagged ISAM files, although the flags are set during write operations.
- File generations are not supported.

ISAM pools

For NK-ISAM files the LEASY user has the option of working with ISAM pools. These are virtual memory areas which are used as buffers for ISAM blocks, thereby reducing the I/O rate.

LEASY supports the following ISAM pools

– global standard ISAM pools

– private ISAM pools managed by **LEASY**

– private ISAM pools managed by the **user**.

The following sections summarize, for the above pools, the measures to be taken by the user prior to starting a LEASY session, and by LEASY during a LEASY session.

Global standard ISAM pools

- Before starting a LEASY session

  LEASY-CATALOG utility routine

  – The *PLK/SLK* operands in the **FIL* statement are omitted or specified with the value *PLK=*STD/SLK=*STD*.

  – The **POO* statement is omitted.

- During a LEASY session

  Runtime system

  During the *OPFL* operation LEASY connects to the global standard ISAM pool, which is always available.

Private ISAM pools managed by LEASY

If the user decides to work with private ISAM pools managed by LEASY, the following options exist:

- LEASY creates the ISAM pool on main task startup (operand *PCR=MAINTASK* in the **POO* statement)
- LEASY creates the ISAM pool on the first OPEN for the file in the runtime system (operand *PCR=RUNTIME* in the **POO* statement).

ISAM pools defined with PCR=MAINTASK

The following should be noted for ISAM pools created by LEASY on main task startup:

- Before starting a LEASY session

  LEASY-CATALOG utility routine

  1. The ISAM pool link names must be specified in the *PLK/SLK* operands of the **FIL* statement.
  2. An appropriate **POO* statement with the *PCR=MAINTASK* operand must be issued for each ISAM pool defined in the **FIL* statement.

- During a LEASY session

  Main task

  When a LEASY session is started, the main task creates all ISAM pools defined with *PCR=MAINTASK*. LEASY creates the pools with *CREATION-MODE=NEW*, i.e. the corresponding ISAM pools must not yet exist.

  Runtime system

  During the *OPFL* operation, LEASY connects to the ISAM pool created by the main task.

  - Advantage of PCR=MAINTASK

    The ISAM pools are created at the beginning of a LEASY session and are retained until the LEASY session ends. Following the *CLFL* operation the ISAM pool remains in existence, i.e. it need not be created anew for a new *OPFL* operation.

  - Disadvantage of PCR=MAINTASK

    When a LEASY session is started, LEASY creates *all* ISAM pools defined with *PCR=MAINTASK*, regardless of whether they are actually needed in the LEASY session.

ISAM pools defined with PCR=RUNTIME

The following should be noted for ISAM pools created by LEASY in the runtime system:

- Before starting a LEASY session

  LEASY-CATALOG utility routine

  1. The ISAM pool link names must be specified in the *PLK/SLK* operands of the **FIL* statement.
  2. An appropriate **POO* statement with the *PCR=RUNTIME* operand must be issued for each ISAM pool defined in the **FIL* statement.

- During a LEASY session

  Runtime system

  During the *OPFL* operation, LEASY creates the ISAM pool, or connects to it if it already exists. LEASY creates the pool with *CREATION-MODE=ANY*, i.e. the relevant ISAM pools may, but need not, exist.

  – Advantage of PCR=RUNTIME

    LEASY creates only the ISAM pools that are actually needed.

  – Disadvantage of PCR=RUNTIME

    The ISAM pool is released after every *CLFL* operation. If a large number of *OPFL/CLFL* operations are executed in a LEASY session, ISAM pools will be frequently created and then deleted, with an adverse effect on performance.

i

Users themselves may also create ISAM pools by means of *CREATE-ISAM-POOL* and *ADD-ISAM-POOL-LINK*. With frequent *OPFL/CLFL* operations, this prevents the relevant ISAM pool from being deleted. The pool name (*CREATE-ISAM-POOL*) and the pool link name (*ADD-ISAM-POOL-LINK*) must be identical to the pool link name in the **FIL* statement in LEASY-CATALOG, i.e. the pool name must be identical to the pool link name.
It is the responsibility of users to ensure that any ISAM pools they have created themselves are deleted accordingly by means of *REMOVE-ISAM-POOL-LINK* and *DELETE-ISAM-POOL*.

Private ISAM pools managed by the user

If the user wishes to work with private ISAM pools managed by the user, the following action is required:

- Before starting a LEASY session

  LEASY-CATALOG utility routine

  - The pool link names must be specified in the *PLK/SLK* operands of the **FIL* statement.
  - The **POO* statement is omitted.

  User

  Before the file is opened the user must create the necessary ISAM pool with *CREATE-ISAM-POOL* or *ADD-ISAM-POOL-LINK*.
  The pool name (*CREATE-ISAM-POOL*) and the pool link name (*ADD-ISAM-POOL-LINK*) must be identical to the pool link name in the **FIL* statement in LEASY-CATALOG, i.e. the pool name must be identical to the pool link name.
  It is the responsibility of users to ensure that any ISAM pools they have created themselves are deleted accordingly by means of *REMOVE-ISAM-POOL-LINK* and *DELETE-ISAM-POOL*.

- During a LEASY session

  Runtime system

  The pool link name specified in the **FIL* statement for LEASY-CATALOG is entered by LEASY in the FCB for the OPEN macro call. LEASY assumes that the related ISAM pool has already been created by the user.

4.4.3 Block-oriented access method (PAM)

The access method supported by LEASY for block-oriented files is based on the UPAM access method of DMS in BS2000 (see the manual “Introductory Guide to DMS”).

General characteristics

PAM files with the following properties are possible.

BLOCK-CONTROL-INFO=PAMKEY

BLOCK-CONTROL-INFO=WITHIN-DATA-BLOCK

BLOCK-CONTROL-INFO=NO

The length of the user data block must be defined for each PAM file to be processed by LEASY. If this length exceeds 2048 bytes, the appropriate *BUFFER-LENGTH* operand of the *SET-FILE-LINK* command is required (default blocking: *STD(SIZE=n)*, n ≤ 16, default value: n=1).

Alternatively the length of the user data block can be defined in the *RECORD-SIZE* operand of the *ADD-FILE-LINK* command. The following conditions apply:

- *RECORD-SIZE* ≤ *BUFFER-LENGTH*
- default value: *RECORD-SIZE=BUFFER-LENGTH*

The conditions for NK files are:

- *RECORD-SIZE* ≤ *BUFFER-LENGTH* - *12*
- default value: *RECORD-SIZE=BUFFER-LENGTH* - *12*

Data is always transferred to/from the data volume in the length specified in *RECORD-SIZE* - beyond a certain length, in chained I/O.

For I/O operations the application program must specify the PAM block number (half-page number) of the first PAM block which belongs to the user data block. Thus the PAM block numbers valid for all I/O operations of the LEASY user program are specified implicitly. The following conditions apply:

valid PAM block number = 1 + (n*(i-1))

where:

n = blocking factor in BLKSIZE=STD(SIZE=n)

i = 1, 2, 3, ... (consecutive natural numbers in ascending order).

LEASY does not feature a buffer management system, but instead transfers direct to/from the buffer of the calling program.

Extensions

- In PAM files with *BLOCK-CONTROL-INFO=PAMKEY*, the logical deletion of data blocks is implemented by flags in the PAM key of all PAM blocks belonging to the user data block. The user part of the PAM key is set to *X'FF'*.
- In PAM files with *BLOCK-CONTROL-INFO=WITHIN-DATA-BLOCK*, the logical deletion of data blocks is implemented by putting the delete code *X'FF'* at the start of the block after the control field (*CF*).
- In PAM files with *BLOCK-CONTROL-INFO=NO*, the logical deletion of data blocks is implemented by putting the delete code *X'FF'* in the first data byte of the PAM block.
- Secondary keys can be defined for PAM files:
  - a maximum of 255 secondary keys per file
  - a maximum of 253 secondary key parts per secondary key
  - a maximum of 512 secondary key parts per file.
- With PAM files LEASY enables block numbers to be read backwards, sequentially in descending order.
- As many file blocks are read or written as required for the specified value of *RECSIZE*.

Restrictions

- Only **entire** file blocks can be read or written.
- All I/O operations are performed synchronously (*RDWT*, *WRTWT* macros); asynchronous I/O is not supported (not even by means of eventing mechanisms).
- I/O operations cannot be chained to form a single call.
- The BS2000 UPAM lock mechanisms are available to the user for foreign LEASY files only. LEASY provides the transaction-oriented lock log instead.
- The user cannot process PAM keys.
- Random write access to PAM tape files is not permitted, but a check is not kept on this at present.
- File generations are not supported.

4.4.4 Direct access method (DAM)

The DAM access method supported by LEASY processes blocked records with a fixed length. DAM is derived from the **relative file organization** of COBOL (see the “COBOL85 (BS2000)” manual) and designed in accordance with the KLDS standard. LEASY maps this access method internally onto DMS UPAM files. DAM files with *BLOCK-CONTROL-INFO=PAMKEY*, *BLOCK-CONTROL-INFO=WITHIN-DATA-BLOCK* and *BLOCK-CONTROL-INFO=NO* are possible.

The DAM access method permits:

- direct, sequential forward and reverse reading of data records
- positioning
- insertion, deletion and updating of data records.

General characteristics

- Each record in a DAM file is provided with a unique identification in the form of a relative record number (primary key of the DAM access method). This primary key has a length of 4 bytes, and is not part of the record. It has an integer value (greater than or equal to zero). It must lie within the permissible range.
- The DAM access method must be defined by means of *FCBTYPE=DAM* (LEASY-CATALOG utility routine).
- LEASY implements blocking and unblocking of the records. The block and record lengths must be defined when the file is defined. The following condition applies:
  RECORD-SIZE≤BUFFER-LENGTH

  The condition for NK-DATA files is:
  RECORD-SIZE ≤ BUFFER-LENGTH - 12
- All data blocks started by means of a write access are preformatted (filled with "erased" records).
- A record does not belong to a file if:
  - it is contained in a data block which does not as yet contain a record, or
  - the erase identifier *X’FF’* is contained in the first byte.

Extensions

Secondary keys can be defined for DAM files:

- a maximum of 255 secondary keys per file
- a maximum of 253 secondary key parts per secondary key
- a maximum of 512 secondary key parts per file.

Restrictions

- DAM may only be defined for disk files.
- File generations are not supported.

4.5 Secondary indexing

In addition to primary keys, one or more secondary keys can be defined for accessing records directly. The secondary keys form part of the record.

ISAM (K and NK), DAM and PAM master and model files can be accessed via LEASY secondary keys; NK-ISAM master, model and foreign files can be accessed via ISAM secondary keys.

The *RDIR, RHLD, RNXT, RNHD, RPRI*, *RPHD* and *SETL* operations can be used on ISAM and LEASY secondary keys.

In a LEASY application a file can be accessed via both LEASY and ISAM secondary keys.

Up to 255 secondary keys can be defined for each ISAM, DAM or PAM file in LEASY. The secondary keys can be specified as partial keys (which may overlap).

The **FIL* statement can be used in the LEASY-CATALOG utility routine to create secondary key definitions, add new ones or delete existing ones.

The definition of the secondary keys also specifies whether the same secondary key value can be used more than once, i.e. whether one secondary key value may have several corresponding primary keys.

A secondary index can also be assigned a repetition factor (multiple secondary index). A multiple secondary index is specified as follows:

```
KEY=(rep,(pos1,len1,dist1),(pos2,len2,dist2))
```

The following diagram shows the principle involved; in this example the secondary index consists of two overlapping partial keys.

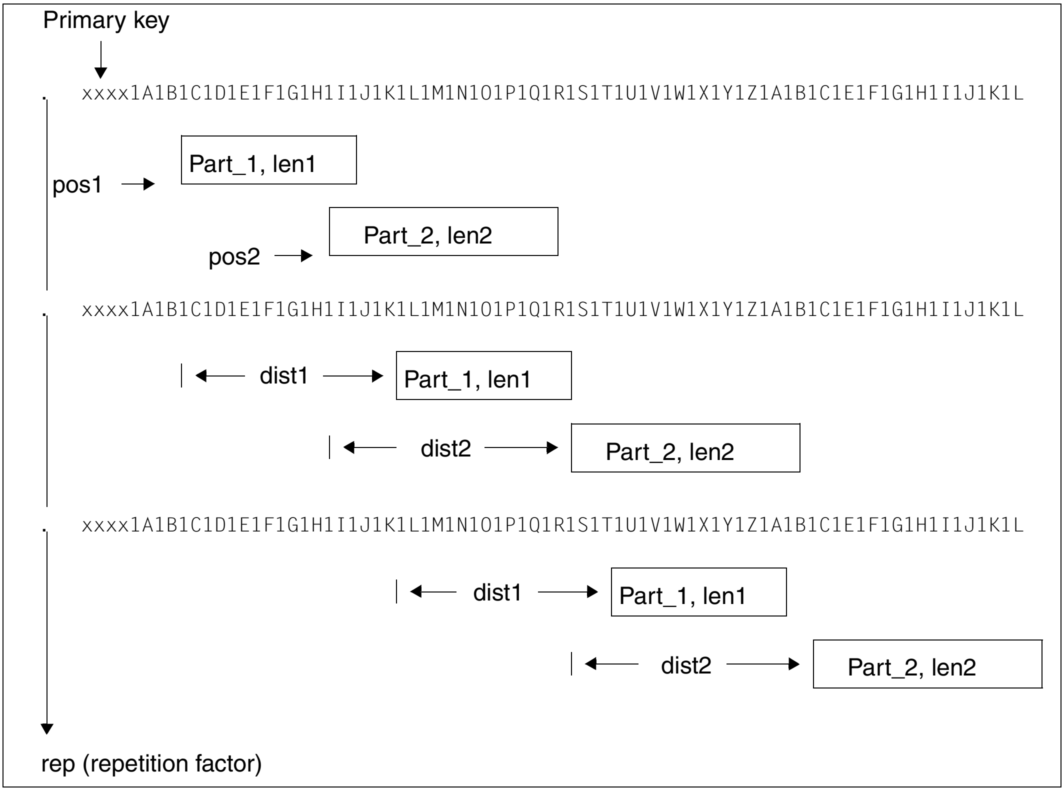


Figure 1: Secondary key with repetition factor

Implementation of secondary index management

A second ISAM file (the secondary index file or SI file) is created for each ISAM, DAM or PAM file for which secondary indices have been defined (primary file). This ISAM file contains only pointers which relate the primary key value to the secondary key value.

The secondary index pointers can be created and inserted in or deleted from the SI file by means of the LEASY-LOADSI utility routine.

If the secondary key values in the primary file are changed, the pointers in the SI file are updated by the LEASY runtime system.

Attributes of an SI file

```
FILENAME=:catid:$userid.file-catalog.file[.suffix]-SI
or
FILENAME=dmsname-SI

ACCESS-METHOD=ISAM
RECORD-FORMAT=V
KEY-POSITION=5
KEY-LENGTH=keylensi (see below)
BUFFER-LENGTH=STD(SIZE=2)
```

Structure of a record in the SI file

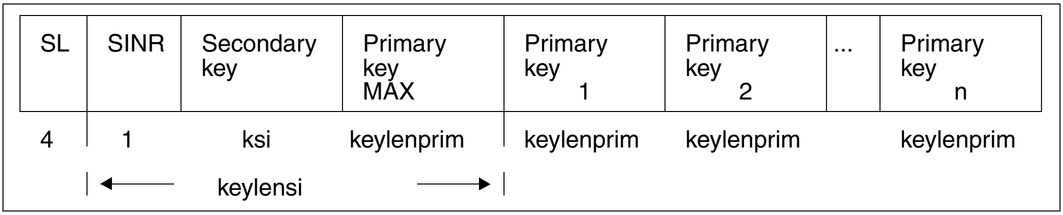


Figure 2: Structure of a record in the SI file

where:

SL	Record length field, length 4 bytes.
keylensi	Length of the ISAM key of the SI file (keylensi=1+ksi+keylenprim).
SINR	internal number of secondary key (1-255), length 1 byte.
Secondary key / ksi	*ksi* is the length of the longest defined secondary key of the corresponding primary file. Since there is only one SI file for all the defined secondary indices of a file, the secondary key field is allocated the length of the longest existing secondary key. Shorter secondary index values are filled with binary zeros.
Primary key MAX	If several primary key values exist for one secondary key value, the primary key values are written consecutively in ascending order to the corresponding record of the SI file. Primary key values can be written to a record until the record length equals the block length (see attributes of an SI file). If the primary key values do not all fit into one record, several records are allocated for one secondary key value, again in ascending order, due to the nature of ISAM keys within the SI file.

	The primary key *MAX* is greater than or equal to the greatest (i.e. the final) primary key value in the SI record. The final or only SI file record corresponding to a secondary key value contains the value *X'FF'* in this field. If an SI key value has no duplicates, *MAX* contains the value *X'00'*.
keylenprim	Primary keys all have the length *keylenprim*; in PAM files *keylenprim=3*; in DAM files *keylenprim=4*.

As can be seen from the structure of the SI file, the primary key chosen should be as short as possible. It should also be considered whether a single, overlong secondary key justifies the resulting extension of the SI file key.

Size of an SI file

An SI file created by the user with the *CREATE-FILE* command is initialized with the following values for the primary and secondary sizes:

```
MAX (user entry, calculated size)
```

If the user does not enter a value or if the value calculated by LEASY is greater than the user entry, the LEASY value is used.

LEASY uses the following formula to calculate the primary and secondary sizes:

```
[(#SPB) * (#BLK) * (#SKY) / (2048 / (2 * LPK + LSK + 5)) + 1]
```

where:

`#SPB`	Number of records in the primary file per block. = block length/record length (for fixed record format) = 5 (for variable ISAM files) = 1 (for PAM files)
`#BLK`	Primary/secondary size of the primary file
`#SKY`	Number of secondary keys
`LPK`	Length of the primary key
`LSK`	Length of the longest secondary key

The calculated value is rounded down to the nearest integer. This result is incremented by the DMS to a multiple of an allocation unit.
For the secondary size the calculated value is limited to *1920*.

ISAM secondary keys for NK-ISAM files

From BS2000 V10 onward, the ISAM DMS access method for NK-ISAM files (non-key ISAM) supports the use of secondary keys.

LEASY can access NK-ISAM records via both LEASY secondary keys and ISAM secondary keys. The criterion for the access mode is the name of the secondary key. If the LEASY and ISAM secondary keys have the same name the LEASY secondary key is used.

The following table shows the main differences between LEASY and ISAM secondary keys:

Criterion	**LEASY SK**	**ISAM SK**
File type	Master files	Master and foreign files
Storage of index	Own SI file	Primary file
Access method	DAM, PAM, ISAM	NK-ISAM
Number of SKs	255	30
Length of SK	PK + SK < 255	< 128
Multiple keys	YES	NO
Suppression	YES	NO
Sequence for identical SKs	Primary key	Time of entry
Definition	Utility routine LEASY-CATALOG	CREATE-ALTERNATE-INDEX command CREAIX macro
Display	Utility routine LEASY-CATALOG	Utility routine LEASY-CATALOG, SHOW-INDEX-ATTRIBUTES command, SHOWAIX macro
Manual setup	Utility routine LEASY-LOADSI	CREATE-ALTERNATE-INDEX command CREAIX macro
Automatic setup via LEASY runtime system	Controlled via LEASY-CATALOG	For each defined SK

Table 3: Differences between LEASY and ISAM secondary keys

A single key value can be specified at the appropriate point in the *AR* or *KB* area. The record read is the one which has the specified key value or - if this record is duplicated - the record that was the first to be written to the file.

The initial value of a key interval is specified in the *KB* range, the end value is the *KE* range.

The first record in the interval is read. If no record is found in this interval, LEASY issues the return code LC=L001.

In the case of duplicates, it is not possible to add a primary key to the secondary key in order to narrow the specification.

ISAM-SK restrictions with respect to LEASY-SK

1. Multiple file positions can be distinguished simultaneously only by different key names and secondary features.
2. Only complete records are transferred to the AR area (the *FA* operand is ignored).
3. If an error occurs the content of the AR area is undefined, i.e. it is not safe to assume that the content of this area remains unchanged.

The LEASY operations *RDIR, RHLD, RNXT, RNHD, RPRA, RPHD* and *SETL* can be used for accessing via the ISAM secondary key.

Note the following when working with ISAM secondary keys in LEASY:

- The name of the ISAM secondary key must be specified in the *SI* operand.
- The value of a single ISAM secondary key must be specified in the *AR* or *KB* range.

  If there is more than one record with the same value the record entered first is returned. If a matching record is not found the return code is *010LL001*.

  A primary key cannot be specified in addition to the secondary key (for the additional specification of duplicates).
- The initial value of the key interval is specified in the *KB* range, the end value is the *KE* range. The following conditions apply:

KB ≤ KE	The record with the smallest secondary key in the specified interval is returned. If the records are identical, the first record entered is returned.
KB > KE	The record with the largest secondary key in the specified interval is returned. If the records are identical, the first record entered is returned.

  If no matching record is found, the return code is *010LL001*. Any subsequent sequential read operations are restricted to the specified interval.
- After positioning, the *RNXT, RNHD, RPRI* and *RPHD* operations can be used for sequential reading. The following conditions apply:

RNXT, RNHD	The record with the next higher SK value is returned. In the case of identical records, the record entered subsequently is returned.
RPRI, RPHD	The record with the next lower SK value is returned. In the case of identical records, the record entered beforehand is returned.

  The return code *010LL003* is returned when the file boundary is reached or if the specified interval is exceeded.
- Information on ISAM secondary keys can be accessed by means of the LEASY-CATALOG utility routine.

4.6 Location of files

Storing files on different volumes is important for two reasons:

- Performance can be enhanced by saving on positioning operations.
- The concept of data security calls for files to be dispersed among different physical locations.

Private volumes can be selected directly; public volumes can be differentiated only by selecting the pubset.

LEASY distinguishes between user files (primary and secondary files) and LEASY system files (LEASY catalog, AIM file, BIM files and LEASY status file).

In dispersing the files, note the following file relationship requirements:

- The LEASY catalog file and the user files should be on the same volume.
- The AIM file, the BIM file and the user files should be on different volumes.
- The BIM files should be distributed over as many volumes as possible (ideally one BIM file per volume).
- In the case of large K-ISAM user files, the option of splitting into index and data sections should be used (*VOL, DEV, DVOL* and *DDEV* operands of the **FIL* command in the LEASY-CATALOG utility routine). The corresponding secondary index file then resides on the same volume as the index section of the primary file.

  This option is not available for NK-ISAM files, but the number of I/O operations can be reduced by using ISAM pools.

4.7 Reserved file names

There are a number of system-internal files in LEASY. The names of these files are generated automatically and are as follows:

`file-catalog.LEASYCAT`

`file-catalog.LEASYAIM`

`file-catalog.BIM#.nnn` (001 ≤ *nnn* ≤ 999)

`file-catalog.LEATSTAT`

`file-catalog.LEASAVE.TAPE`

`file-catalog.LEASAVE.DISK`

`file-catalog.LEADIAG`

These file names are reserved names. With the exception of the file *file-catalog.LEADIAG* the user is not allowed to modify or delete them.

The file names of the secondary index files are generated automatically. They are based on the names of the associated primary files. The character string "-SI" is also used as a suffix to the file names.

In addition, all file names of data files created by the LEASY-CATALOG utility routine and any temporary file names generated during a session are also privileged names. The user is responsible for ensuring that these rules are observed.

5 Save facilities

LEASY offers users two save facilities. These can be selected via the LEASY-MAINTASK and LEASY-CATALOG utility routines on a session-specific and file-specific basis.

- **Transaction saving** enables transactions to be rolled back to their starting point so that consistent files can be maintained.
- **Data saving** enables corrupted files to be restored from backup data. The **shadow file concept** is a special type of data saving which cuts down the time taken for reconstructing files for time-critical applications to a minimum.

These two concepts are described below.

5.1 Transaction saving

Transaction saving enables transactions to be rolled back to their initial status.

5.1.1 Definition of a transaction

Generally speaking, a transaction is a series of actions which is executed either in its entirety or not at all.

A LEASY transaction is a series of LEASY operations between the *OPTR* (open transaction) and *CLTR* (close transaction) LEASY operations. In order to maintain a consistent set of files it is essential to be able to cancel (roll back) incomplete transactions without affecting parallel transactions.

Transactions can be rolled back

- by an explicit request in the program with the *CLTR* (with *OPE1=R*) or *BACK* operation
- by an explicit request via the *RLBT* function in the LEASY-MASTER utility routine
- by activating an STXIT routine (e.g. after abnormal program termination)
- by warm starting the system with LEASY-MAINTASK.

Transaction saving is implemented using the BIM (before-image) save method.

5.1.2 BIM save method

If an error occurs during execution of a transaction, i.e. the transaction cannot be completed, all updates to the relevant files must be canceled. This involves:

- deleting any inserted records or blocks
- reinserting any deleted records or blocks
- canceling any changes in records or blocks.

The BIM save method maintains a BIM file for each of the parallel transactions in which it stores the counter-operation for each operation.

Example

Operation	**Contents of the BIM file**
Delete record 8 in file A	Insert record 8 in file A Contents of record 8
Update record 16 in file B	Update record 16 in file B Old contents of record 16
Insert record 3 in file X	Delete record 3 in file X (record contents not required for deleting)

During rollback the BIM file is processed from the end of the file to the beginning.

When the BIM file has been successfully processed and the transaction has been duly terminated, the BIM file is defined as "logically empty", i.e. the first PAM block is overwritten with binary zeros.

5.1.3 BIM files

BIM files are PAM access files which are created using the LEASY-MAINTASK utility routine.

The following LEASY-MAINTASK statements relate to the BIM file to be created:

*TRA	Number of BIM files
*BCA	Pubset of the BIM files
*BDE, *BVO	Device type and name of the private disks for BIM files
*BIO	Performance attribute for BIM files

Names of the BIM files:

`:catid:$userid.file-catalog.BIM#.nnn`

where

catid	*BCA entry or :catid: of the LEASY catalog
$userid	user ID under which the main LEASY task runs
file-catalog	name of the LEASY catalog allocated to the main LEASY task
BIM#	identifier of the file as a BIM file
nnn	internal transaction number, starting at 001

The BIM files are assigned the following:

BUFFER-LENGTH	=STD(1)	for NK2 disks
	=STD(2)	for NK4 disks
BLOCK-CONTROL-INFO	=PAMKEY	for KEY disks and *CLASS-2-OPTION BLKCTRL=PAMKEY*
	=NO	otherwise

The BIM files are protected by a read password against unauthorized access.

In exceptional cases, users can create BIM files themselves using the *CREATE-FILE* command if they do not want the file attributes allocated by LEASY-MAINTASK.

If LEASY-MAINTASK is started without BIM saving, any existing BIM files are deleted. This can be suppressed via the *KEEP-BIM-FILES* operand of the **LOG* statement.

5.2 Data saving

Data saving enables files damaged or made unreadable by system errors to be restored. Data saving is implemented using the AIM (after-image) save method.

The AIM save method maintains an AIM file for each LEASY catalog. All updates carried out on a file during one or more LEASY sessions are entered in this AIM file. Since the date, time and transaction identifiers are also stored in the AIM file, this file is also transaction-specific.

Once backup copies have been loaded, damaged original files are reconstructed by the LEASY-RECONST utility routine. LEASY-RECONST reads in the AIM file and then updates the LEASY files.

Users can retain generations of copies of original files and AIM files and base the reconstruction of a damaged file on an old copy.

5.2.1 AIM file

The AIM file is a PAM file which must be created as a file generation group. The following options are available to the user for creating AIM file generation groups or new AIM generations:

- Using the LEASY-MAINTASK utility routine

  The **AGE, *ADE, *AVO*, **ASP*, **ACA, *AIO* and **AIS* statements are available for this purpose. In addition, the **DES* statement specifies whether or not the AIM files are to be overwritten with binary zeros when deleted. The AIM files are protected internally by LEASY-MAINTASK with the aid of an encoded read password. The **FAA* and **AGF* statements control the release of AIM file generations.

- Using a *CREATE-FILE-GROUP* command with the following format:

  ```
  /CREATE-FILE-GROUP GROUP-NAME=:catid:$userid.file-catalog.LEASYAIM
  ```

  Users can issue this command to create a new file generation group.

- Using a *CREATE-FILE-GENERATION* command with the following format:

  ```
  /CREATE-FILE-GENERATION
  GENERATION-NAME=file-catalog.LEASYAIM(*n),SPACE=...
  ```

  Users can issue this command to create the **first** generation themselves. In the same way as BIM files, AIM files can only use the formats *BLOCK-CONTROL=PAMKEY* and *BLOCK-CONTROL=NO*.

AIM files can be deleted with the aid of the LEASY-MASTER utility routine.

Example of creating a new AIM file generation group using LEASY-MAINTASK:

```
/START-LEASY-MAINTASK
CAT=LEACAT
LOG=Y
AGE=3
.
.
.
END
```

These statements create the file generation group *LEACAT.LEASYAIM*, of which not more than 3 generations are contained in the catalog at the same time. It can be processed under any user ID.

Whenever a new (empty) AIM file is to be started, the **ASP* statement must be specified in the LEASY-MAINTASK utility routine. The corresponding **ADE* and **AVO* statements must also be specified if the AIM file is to be created on a private volume. A file generation must, however, always be created in its entirety on public or private data volumes. No **ASP* statement should be specified if writing is to be continued in the last (just defined) generation.

Example of creating a new AIM file generation using LEASY-MAINTASK:

```
*ASP=(160,100)
```

This statement is used to create the next-higher AIM file generation. The *ASP* operand defines the primary memory allocation (160 PAM pages) and the secondary memory allocation (100 PAM pages), i.e. 160 PAM pages are reserved when the AIM file generation is generated, and 100 PAM pages are reserved additionally each time more memory is needed.

The secondary allocation must be at least 16 PAM pages, since PAM can be used to write chained I/Os of up to 32 Kb at a time. This value is set automatically by the main LEASY task if a lower value is specified or if the specification is omitted.

Upon reaching a maximum size or by controlling via the LEASY-MASTER utility routine, it is possible to switch from one AIM file generation to the next during a LEASY session. Again the entries for the **ASP, *ADE* and **AVO* statements in LEASY-MAINTASK are used.

AIM files on private disks

The **ADE* and **AVO* statements in the LEASY-MAINTASK utility routine must be used to enter the device type and the VSN, respectively, of the private disk when an AIM file generation group is created on a private disk. All AIM generations must now likewise be on private disks (not necessarily on the same one).

LEASY-MAINTASK creates the AIM file generation group using *OVERFLOW-OPTION=REUSE-VOLUME* in order to permit switching of the AIM file by a LEASY utility routine, since the macros set by the LEASY utility routines do not contain any device specifications. The following applies here:

- Device specifications must be set explicitly only for the first generations up to the maximum number of generations specified by means of the **AGE* statement.
- When creating a generation with a higher generation number, the file is created automatically on that volume having the oldest generation, which is thereby deleted.

AIM files on tapes

The AIM file generation group is created on tape by means of statements to the LEASY-MAINTASK utility routine.

The **ADE* and **AVO* statements must be used to specify the device type and VSN of the tape respectively. The value *Y,M* must be specified in the **LOG* statement so that the main task can write to the AIM file. The **ASP* statement (*TAPE* operand) controls the advance to the next AIM file generation.

Note the following rules for AIM files on tape:

- Only one AIM file generation is permitted on one tape.
- An AIM file generation must not extend over several tapes. Users are responsible for ensuring that a switch is made to the next generation in good time.

  The entries made for **ADE* and **AVO* are only evaluated for the first AIM file generation created in a session. If further AIM file generations are to be created on tape, this can be done by means of

  - renewed starting of LEASY-MAINTASK (**ADE/*AVO* statement)
  - the *CREATE-FILE-GROUP* command
  - the *CREATE-FILE-GENERATION* command.
- If LEASY-MAINTASK is started by means of an ENTER procedure, users should ensure that more CPU time is specified in the *ENTER-JOB* command for AIM on tape than for AIM on disk.

Error when writing to the AIM file

If a DVS error occurs when the AIM file is being written to, AIM logging is no longer possible. The current AIM file generation is then generally unusable.

In such a case LEASY transfers return code `99ALLS75` to the user program and resets the transaction. Any further LEASY request triggers return code `99ALLS81`, which has the following meaning: the AIM file can no longer be written as the result of an error, no further LEASY request is permitted, and the transaction was reset by LEASY.

The following actions are required to enable LEASY to run properly again:

- Terminate LEASY-MAINTASK
- Save the original data set
- Determine the cause of the error and, if required, take precautions to prevent the error from occurring again (e.g. in the event of a shortage of disk storage space, provide sufficient free disk storage space)

The other actions depend on whether or not the LEASY session in which the DVS problem occurred while writing the AIM file generation was executed using automatic reconstruction of shadow files:

- Without automatic reconstruction of shadow files
  - Ensure that at least one AIM file generation is free so that a switchover can be performed.
  - Restart the LEASY-MAINTASK with additional specification of the ASP parameter so that a switchover to a new AIM file generation can be performed.
- With automatic reconstruction of shadow files
  - Generate correct shadow files by copying the backup of the original data set.
  - Restart the LEASY-MAINTASK with AIM logging and automatic RECONST.

5.2.2 Releasing AIM file generations

When creating AIM files, LEASY uses file generations as follows: The LEASY-MAINTASK statement **AGE* is used to specify a maximum number of AIM generations. Provided this threshold value is not reached, a new generation is created when required. However, when the threshold value is reached, the oldest file generation is released and its catalog entry is deleted. To ensure that no data from the oldest AIM generation can be lost during this process, the AIM generations from LEASY V6.2 are by default protected against being released. Before a switchover to the oldest AIM generation can take place, it must be released. The measures required to do this depend mainly on whether automatic reconstruction of shadow files is being used.

If required, the existing method in which the last AIM file generation processed is released immediately after a successful switchover can be set using the LEASY-MAINTASK statement **FAA*.

Working without automatic reconstruction of shadow files

If automatic reconstruction of shadow files is not being used (**AUT=N* parameter or no **AUT* specification in the LEASY-MAINTASK utility routine), the LEASY administrator must release the file generations which are no longer required (*AIMA* function of the LEASY-MASTER utility routine or **AGF* specification in the LEASY-MAINTASK).

CAUTION!

The LEASY administrator is responsible for only releasing AIM file generations which have already been saved or reconstructed.

If the LEASY administrator does not release the AIM file generations in time, it can occur that file generations can no longer be created. In this case no further switchover operations can be performed. If, given this situation, an attempt is nevertheless made to switch over the file generation, as distinction must be made between the following cases:

- Explicit switchover using the LEASY-MASTER utility routine

  Explicit switchover operations using the *AIMC*, *AIMI* or *AIMW* statement of the LEASY-MASTER utility routine are rejected with a corresponding message. The AIM file generation used last is still used.

  ► The LEASY administrator must release AIM file generations which are no longer required using the LEASY-MASTER statement *AIMA*. The LEASY administrator must then repeat the planned switchover operation.

- Implicit switchover as a result of the LEASY-MAINTASK statement **AIS*

  Switchover operations which are required because the file size specified using the *pamblock-number* operand of the LEASY-MAINTASK statement **AIS* are rejected with the corresponding messages. The AIM file generation used last is still used.

The further behavior depends on the *increment* parameter in the **AIS* statement:

- If *increment* is not specified, the current AIM file generation is still used until the maximum file size of 16775000 PAM blocks is reached.
- If *increment* is specified, *pamblock-number* is incremented by the specified value. The current AIM file generation is still used until the new value is reached. The procedure described above is then repeated.

If *increment* has the value 0 or when the maximum file size of 16775000 PAM blocks is reached, the current AIM file generation can no longer be written to. LEASY-MAINTASK issues a corresponding message (*LEA5012*). All other LEASY statements are rejected with the return code 99ALLS75.

- ► The LEASY administrator must release AIM file generations which are no longer required using the LEASY-MASTER statement *AIMA*. The next switchover operation can be executed successfully. The value for *pamblock-number* is reset to the value originally defined following the successful switchover.

- Implicit switchover of the AIM file generation as a result of the LEASY-MAINTASK statement *ASP* or because of a change of version

  If a switchover is required because the LEASY-MAINTASK statement **ASP* is specified or because a change of LEASY version from a version < V6.1 has taken place, the LEASY-MAINTASK utility routine terminates abnormally.

  - ► The LEASY administrator must release at least one AIM file generation. However, the *AIMA* statement of the LEASY-MASTER utility routine which is actually provided for this purpose is only available after the LEASY-MAINTASK utility routine has been started successfully.

    Depending on the current value of the LEASY-MAINTASK statement **AGE*, this problem can be solved as follows:

    1. Value < 255

       The LEASY administrator increments this value by at least 1 and restarts LEASY-MAINTASK.
       The AIM file generation can now be switched over successfully because it is possible to create at least one AIM file generation.

    2. Value = 255

       Because the value cannot be increased any further, the LEASY administrator must use the **AGF* statement in the utility routine to specify a number of file generations which are to be released. This statement is permitted only when **AGE=255*. A maximum of 254 file generations can be released in this way.

       The LEASY administrator must then restart LEASY-MAINTASK.

Working with automatic reconstruction of shadow files

If automatic reconstruction of shadow files is specified in the LEASY-MAINTASK utility routine (_*AUT=Y_), after each update of the shadow files LEASY-MAINTASK automatically releases the AIM generation which is no longer required. In this case generally no intervention by the LEASY administrator is required.

However, this automatic release is possible only if automatic recording of shadow files was defined for **all** files in the LEASY catalog (_AIM=(Y,A)_ or _AIM=(R,A)_ in the _*FIL_ statement of the LEASY-CATALOG utility routine).

If the LEASY catalog contains at least one file for which AIM logging has been defined but automatic recording of shadow files has not been specified (_AIM=Y_ or _AIM=R_), this file's information would be lost after the AIM generation is released. If _AUT=Y_ is specified, LEASY-MAINTASK consequently checks whether files exist which are defined with _AIM=Y_ or _AIM=R_.

The further behavior depends on the specification in the _FAA_ statement:

- _FAA=N_ (default)

  LEASY-MAINTASK is terminated abnormally and issues a message to point out that there is a danger of data being lost. The LEASY administrator must then use the LEASY-CATALOG utility routine to specify all the files concerned using _AIM=N_, _AIM=(Y,A)_ or _AIM=(R,A)_ and, if required, also create shadow files. The LEASY administrator can then restart the LEASY-MAINTASK utility routine.

  > Files for which _AIM=Y_ or _AIM=R_ was specified can also be determined using the _*INF ,M_ statement of the LEASY-CATALOG utility routine.

- _FAA=Y_ (behavior as with LEASY ≤ V6.1)

  LEASY-MAINTASK does issue a message to point out that there is a danger of data being lost, but it continues to operate normally.

5.2.3 AIM elements

The individual entries in the AIM file (referred to as AIM elements) are of variable length and are written consecutively. The element entries are chained both forwards and backwards. Each AIM element consists of two parts:

- a **variable** entry:

  in which the LEASY runtime system writes the data that varies in each LEASY operation, e.g. the file names when physically opening files, or the file name and the new record or block contents after updating.

- a **fixed** entry:

in which the information is written that is required after each action in order to restore the file, i.e. the operation code, TSN, timer, session and transaction numbers, internal LEASY transaction number and two element length fields.

The names of the AIM elements that are listed below are logged in system file *SYSLST* during a restore run using the LEASY-RECONST utility routine.

Name	Action	Meaning
MTSK	Main task entry	LEASY-MAINTASK start
SESS	Session entry	Start of new session (LEASY-MAINTASK)
CATD	CATD entry	Connection to common memory
OPEN	OPEN entry	Physical opening of files
CLOS	CLOSE entry	Physical closing of files
OPTR	OPTR entry	Start of new transaction
CLTR	CLTR entry	End of a transaction
RLBK	Rollback entry	Start of a rollback
STOR[1]	STORE/PUT entry	Addition of a record (ISAM, DAM or SAM) or a block (PAM)
DLET[2]	DLET entry	Deletion of an ISAM or DAM record or a block of a PAM file
PUTX	PUTX entry	Overwriting an ISAM or DAM record or a block of a PAM file
PUTS	PUTXSAM entry	Overwriting of a SAM record
ELIF	ELIMFILE entry	Deletion of a whole ISAM, DAM or PAM file
ELIR	ELIMREC entry	Delete an ISAM, DAM or PAM file beginning at a specified key
SETS	SETLSAM entry	SETL is used to truncate a SAM file
ENDA	End of AIM file entry	Main task has switched AIM file during the session
CSES	Continue session entry	Main task continues session in newly created AIM file
OLDB	Old buffer entry	Main task was unable to write the *ENDA* entry to the old AIM file after switching over to the new AIM file because of an I/O error. For this reason, the main task saves the contents (not yet written) of the AIM buffer in CMMAIN by entering them in the new AIM file and terminating with the *OLDB* element.
CTSK	Continue task entry	A LEASY task has linked itself to the newly created AIM file
FILS	Files list entry	A LEASY task has, at the moment of linkage to the new AIM file, physically opened the files specified in the FILS entry.
PETR	PETR entry	Suspended transaction

Table 4: AIM elements logged in a restore run (part 1 of 2)

Name	**Action**	**Meaning**
STOD	Store DAM buffer entry	Addition or overwriting of a block of a DAM file
CINF	CINF entry	Transfer currency information
LOCK	LOCK entry	Set record lock
RDIR	RDIR entry	Read record directly
RHLD	RHLD entry	Read record directly with record lock
RNXT	RNXT entry	Read next record
RNHD	RNHD entry	Read next record with record lock
RPRI	RPRI entry	Read preceding record
RPHD	RPHD entry	Read preceding record with record lock
UNLK	UNLK entry	Cancel record lock

Table 4: AIM elements logged in a restore run (part 2 of 2)

[1] Line 1 of the AIM file contains the first 24 bytes of the primary key. Line 2 contains the record contents as of byte 1 (including the primary key).

[2] Only the primary key is entered here.

i A record is not entered in the AIM file unless the corresponding LEASY call runs without error.

The only exception to this rule is return code LP11 (specified *CINF* area too small) for the *CINF* operation. An AIM record is written despite this error, because the *CINF* call supplies partial information even though the error resulting in LP11 has occurred.

If the outcome of this call includes an AIM write error as well, the *CINF* return code (LP11) is displayed in the *RC-LC* field (RE areas) and the AIM write return code in the *RC-LCE* field (RE area).

Truncated AIM elements

The user is able to specify that only those sections of a record which have actually been modified are to be written into the AIM file. The LEASY-RECONST utility routine can connect the modified record with the aid of length fields (which provide information on the unmodified record sections) and the associated save file.

The amount of memory space saved is dependent on the ratio of modified record sections to unmodified record sections.

– Performance depends on the specific transaction concerned:
  An *OPTR* ... *RHLD* ... *REWR* ... *CLTR* transaction causes the path length to be increased by the size of the truncated record that is created. This is because an output has to be made to the AIM file for each *CLTR*.
– However, if transactions occur with several *REWR* calls relating to long records, performance can be improved if it is possible to do away with one or maybe more outputs to the AIM file.

Performance is always worsened by a reconstruction run. If LEASY-RECONST detects a truncated record, the non-truncated record must first be read from the save file so that the information from the AIM element can be used to connect the modified record.

The user specifies whether truncated or non-truncated AIM records are to be written by supplying values for the appropriate operands in the LEASY-CATALOG utility routine (*AIM* operand in the **FIL* statement).

A decision to save a file with the aid of truncated AIM records can be reversed via the *AIM* operand in the **FIL* statement of the LEASY-CATALOG utility routine.

Saving with the AIM file

The save concept is based on the following method:

- At certain times the user’s files must be saved to other volumes. This can be achieved, for example, with the *COPY-FILE* command (see the "Commands, Volume 1 - 5" manual), with the ARCHIVE utility routine (see the "ARCHIVE (BS2000/OSD)" manual) or with the LEASY-SAVE utility routine.

  Saving with file converters or other programs which read record-by-record is not permitted with SAM files, since the retrieval addresses of the records may thus be altered.

- Damaged files are overwritten with a save file and then the LEASY-RECONST utility routine processes the AIM files from the moment of saving to the current state.

  By specifying the appropriate operands in the LEASY-RECONST utility routine, the files and/or start of restoration (date or session number) can be selected.

- File saving need not necessarily coincide with the creation of a new generation of the AIM file, since during restoration either a section of an AIM file can be selected or several consecutive AIM files may be incorporated into the user’s file one after another.

  However, files should not be saved during a LEASY session, but only between two sessions, i.e. before the start of LEASY-MAINTASK.

5.3 Shadow files

The shadow file concept is based on keeping a set of copy files parallel to the original files. As part of this method, AIM files are also created.

Copies (shadow files) are made of the original files to be processed. Any subsequent changes made to the original file are then entered in the AIM file. The AIM saving facility is then switched over, i.e. the changes are entered into the new AIM file generation. Finally the shadow file is updated to the latest status with the aid of the closed AIM file generation. The difference between the shadow file and the original is thus at most the contents of one AIM file generation. If the contents of the current AIM file generation are kept to a minimum, the difference between them will also be minimal, and reconstruction will be rapid if a file has been destroyed.

5.3.1 Creating shadow files

The LEASY-CATALOG utility routine is used to define the naming conventions of shadow files for the entire catalog (see the *CPC* and *CPS* operands in the **CAT* statement).

Responsibility for creating shadow files rests with the user.

When this technique is used, one shadow file must be generated for every user file requiring an AIM save. Shadow files must also be created for SI files.

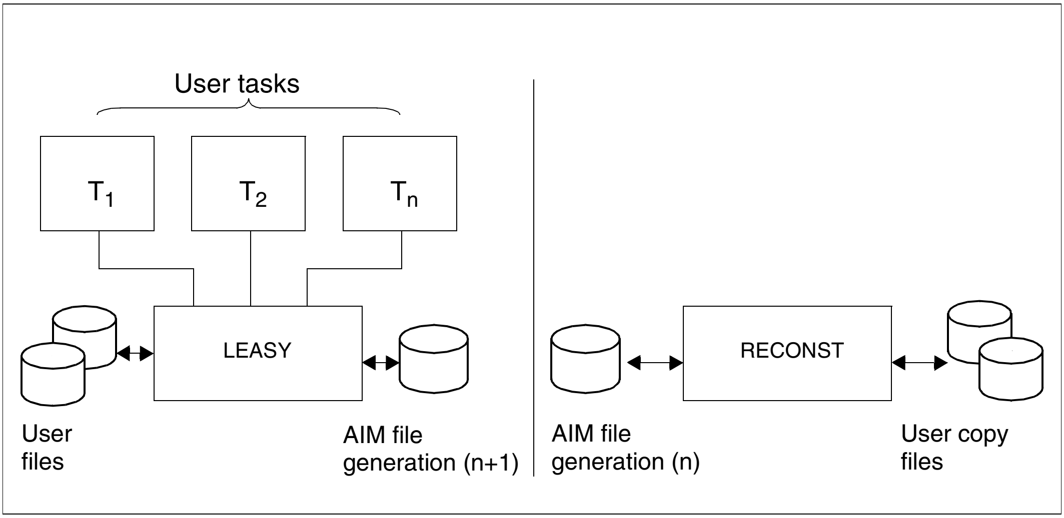


Figure 3: Shadow files

In order to support the concept of shadow files, it must be possible to detach the latest AIM file in the course of the current LEASY session and continue writing in a new (empty) file. To do so, a switchover from the current generation to the next AIM generation is undertaken.

Switching over of the AIM file can be initiated by:

- reaching a predetermined maximum AIM file size. This size is set for each LEASY session via the **AIS* statement in the LEASY-MAINTASK utility routine
- using the *AIMI*, *AIMC* or *AIMW* statement in the LEASY-MASTER utility routine.

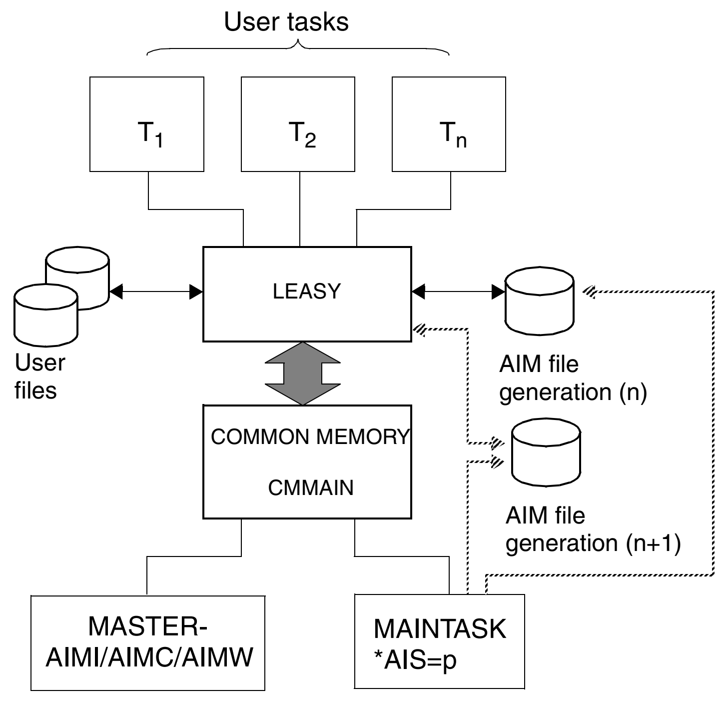


Figure 4: Switching to a new AIM file generation

The LEASY maintask is initiated automatically to switch to the new AIM file generation.

If the AIM buffer is written to the AIM file by the main task, the main task writes to the new generation once switchover takes place. If the AIM buffer is written to the AIM file by application tasks, each application program attached to the old AIM file must be added to the new generation. This is done for every task the next time LEASY is called. Until this next call the old generation of tasks still remains open and therefore cannot yet be used for reconstruction purposes.

During switchover, the elements *ENDA, CSES, OLDB, CTSK* and *FILS* are written to the AIM file. The sequence is as follows:
First the main task writes *CSES* to the new AIM file, then *ENDA* is added to the old AIM file or, if an error occurs (PAM macro), *OLDB* to the new AIM file. Each LEASY user task writes *CTSK* (+*FILS*) to the new AIM file.

Should I/O errors occur during writing to the AIM file it is therefore possible, by means of a statement to the LEASY-MASTER utility, to switch over to a new AIM file during the current session, without losing any elements (from the AIM buffer in CMMAIN) that may not yet have been written to the AIM file.

Switching to a new AIM file can take place even when transactions are open.

5.3.2 Maximum protection against failure

Maximum protection against failure and minimum restart times is a combination best achieved with shadow files distributed across two pubsets as shown below.

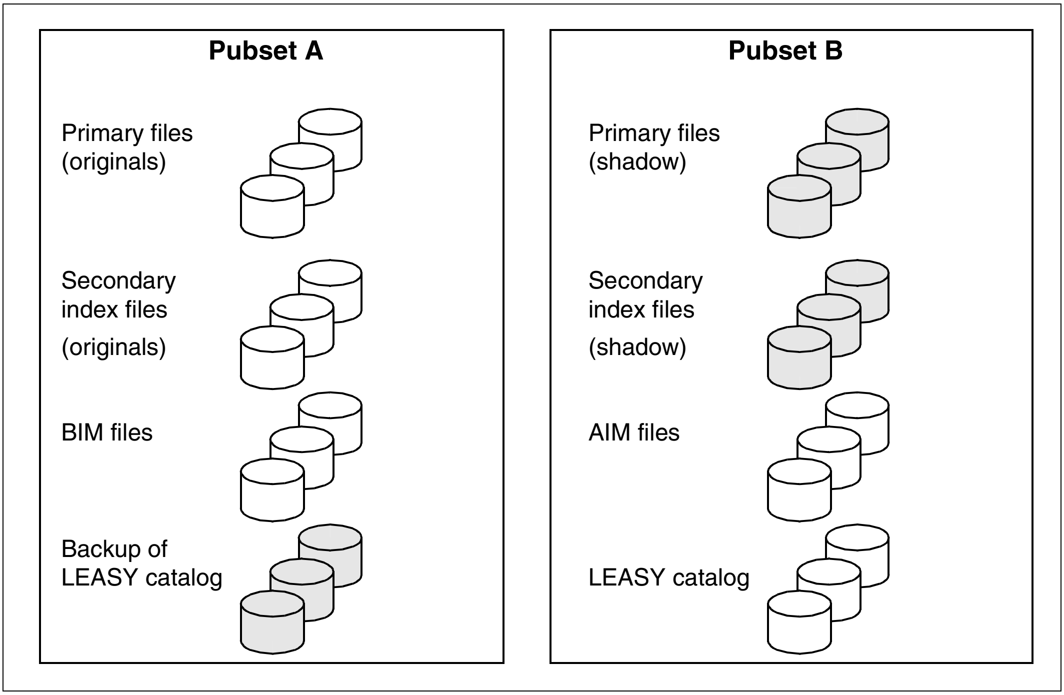


Figure 5: Shadow files distributed to two pubsets

You can set up this configuration as follows:

Pubset A is the default pubset, pubset B is another pubset available for backups.

```
/START-LEASY-CATALOG
```

(Start LEASY-CATALOG).

```
*CAT :B:file-catalog,TYP=N,CPC=:B:copycat[,CID=YES]
```

The CAT statement creates the catalog on pubset B. The shadow files should also be on pubset B. *CPS* would be a possible alternative to *CPC* as a way of defining the names of the shadow files. Make sure that the *CID* parameter is set to *YES,* so that LEASY-MAINTASK will subsequently evaluate the *ACA* or *BCA* parameter, as appropriate. Under normal circumstances it is not necessary to specify the parameter explicitly.

Use *FIL* statements to create the user files (one statement per file):

```
*FIL file,NAM=:A:filename,AIM=(Y,A),...
```

The *NAM* parameter is necessary so that the original files are created on pubset A. The naming convention *file-catalog.file* is recommended for *filename*. AIM saving is set to automatic reconstruction.

```
*END
```

The END statement terminates LEASY-CATALOG.

```
/COPY-FILE :A:original-file,:B:shadow-file
```

```
/COPY-FILE :A:original-si-file,:B:shadow-si-file
```

The shadow files are created by copying the originals to pubset B.

```
/COPY-FILE :B:file-catalog.LEASYCAT,:A:file-catalog.LEASYCAT.SAVE
```

Create the backup copy of the LEASY catalog by copying it to pubset A. This process must be repeated every time the catalog is updated.

```
/START-LEASY-MAINTASK
```

Start LEASY-MAINTASK with the following parameters:

```
*CAT=:B:file-catalog

*ACA=B

*BCA=A

*LOG=Y

*AUT=Y

*REN=enter-command

*AGE=3(or > 3)

.

.

.

*END
```

5.3.3 Repair measures

If a defect occurs on a pubset, it is important to distinguish between the two cases described below with regard to planning repairs:

- the defective pubset cannot be repaired within a reasonable time
- the defective pubset can be repaired within a reasonable time.

Defective pubset cannot be repaired within a reasonable time

The new LEASY session is started on the intact pubset; shadow files are not created. This expedites the resumption of work.

- Pubset A is defective

  Measures:

  1. Declare pubset B as the default pubset.
  2. In the catalog entry for pubset B, set *CID=NO*.
  3. Rename the shadow files as original files.
  4. Perform a reconstruction run of the connected transactions without updated AIM generations (*MOD TRA=V*).
  5. Start a new session with catalog, AIM files and original files on B.

- Pubset B is defective

  Measures:

  1. In the catalog entry for pubset A, set *CID=NO*.
  2. Perform a warm start with catalog A - without AIM write - to bring the original files to a consistent state.
  3. Terminate main task.
  4. Save the original files.
  5. Start a new session with catalog, AIM files and original files on A.

Defective pubset can be repaired within a reasonable time

Bring all repair measures to a conclusion before proceeding

Then restart the session with shadow file saving as follows:

- Pubset A was defective
  1. Delete BIM files.
  2. Warm-start the new session. The shadow files are automatically updated.
  3. Copy the shadow files to the original files.
- Pubset B was defective
  1. Overwrite catalog B with catalog A.
  2. Warm-start the new session without automatic RECONST. This reestablishes the consistency of the original files.
  3. Delete AIM generations.
  4. Terminate LEASY-MAINTASK.
  5. Overwrite the shadow files with the original files.
  6. Save the original files and the shadow files.
  7. Start a new session with automatic RECONST.

5.3.4 Reconstructing an AIM file generation

Reconstruction of the old AIM generation in shadow files, in other words the updating of the shadow files, can be either explicitly performed by the user or automatically initiated by LEASY. Reconstruction of the old AIM file generation is made possible by the LEASY-RECONST utility routine as soon as all tasks have closed the old AIM file. In this case the operand *COPY=YES* must be specified in the *CAT* statement.

For automatic reconstruction of AIM file generations LEASY proceeds as follows:

- it first waits until reconstruction of an AIM file generation is possible
- then it starts the LEASY-RECONST utility routine using appropriate statements
- and triggers a warm start using the AIM file.

AIM file generations are automatically reconstructed only if the user has set up shadow files and the following statements have been specified for the LEASY-CATALOG and LEASY-MAINTASK utility routines:

LEASY-CATALOG utility routine	**Meaning**
CAT file-catalog,...,CPC=...[,CPS=...]	Defines names of shadow files
FIL file,...,AIM=$\left\{\begin{matrix}(Y,A)\\(R,A)\end{matrix}\right\}$	Specifies automatic keeping of shadow files

LEASY-MAINTASK utility routine	**Meaning**
LOG=$\left\{\begin{matrix}A,\underline{M}\\Y,\underline{M}\end{matrix}\right\}$	Session with AIM save and writing of records by the main task
AUT=Y	Automatic reconstruction
REN=enter-command	ENTER command for the RECONST task
[PAS=password]	Password(s) for the RECONST task
AGE ≥ 3	Number of AIM file generations. To guarantee automatic reconstruction with reliable saving the value of this number should be at least 3.

Note the following when working with LEASY shadow files:

- When the mode of operation "automatic keeping of shadow files" is used, a session should be terminated with *CLOS* whenever possible, ensuring that the status of the shadow, original and AIM files is consistent.

- LEASY applications with long-running transactions are not suitable in conjunction with automatic keeping of shadow files. Transactions should not extend beyond more than 2 AIM file generations.
- At least 3 AIM file generations are required for proper and reliable operation.
- AIM file generations should be deleted using the LEASY-MASTER utility routine (*AIME* statement). Only generations which were not opened by LEASY can be deleted. If all generations are to be deleted (“WHOLE” response), LEASY-MAINTASK must be started with *LOG=A,R* or *LOG=Y,R*. No functions for controlling the maintask (TERM/CLOS/SHUT) and no user tasks which use the AIM saving facility may be active.

An example of the automatic reconstruction of AIM file generations can be found in the chapter “Sample applications” starting on page 375.

AIM management record

A separate AIM management record for the AIM file is kept in the LEASY catalog. This record contains entries pertaining to the number and state of the AIM file generations.

The following states are possible (see figure 6 on page 74):

GENFREE	The AIM file has been newly created or reconstruction of the AIM file generations have been successfully completed. If the AIM file is being created for the first time by the main task, all generations have the GENFREE state. If a LEASY application is active, the state of the AIM file must be consistent if a switchover to automatic is taking place. Consistency implies that all shadow files of the AIM file generation have been updated and copied to the original files. The *GENFREE* state is also reached once the contents of an AIM file have been incorporated in the shadow files.
GENINUSE	AIM records are currently being written in this AIM file generation. When a LEASY application is started, the "last" generation is given the *GENINUSE* state. This is the generation to which all connected user tasks write.
GENSWIT	The AIM file generation is being switched over. The user must directly initiate switchover to a new generation by using the LEASY-MASTER utility routine (*AIMI/AIMC/AIMW* functions) or indirectly switch over by setting a maximum file size using the LEASY-MAINTASK utility routine (*AIS* statement).
GENWAIT	Not all transactions written to this AIM file generation have been completed. The old generation assumes the *GENWAIT* state as soon as switchover to the new generation has taken place.

GENREADY	All transactions begun in this AIM file generation have been completed. All transactions written to this generation have been terminated normally. In this case it is of no significance whether the transactions terminated normally with *CLTR* or by means of rollback with *CTLR* and the additional function *OPE1=R*. MAINTASK can initiate reconstruction of the AIM file generations when it has the *GENREADY* state.
GENRECO	The AIM file generation group is being reconstructed. The group has the *GENRECO* state as long as the AIM file generation group is being processed by LEASY-RECONST.

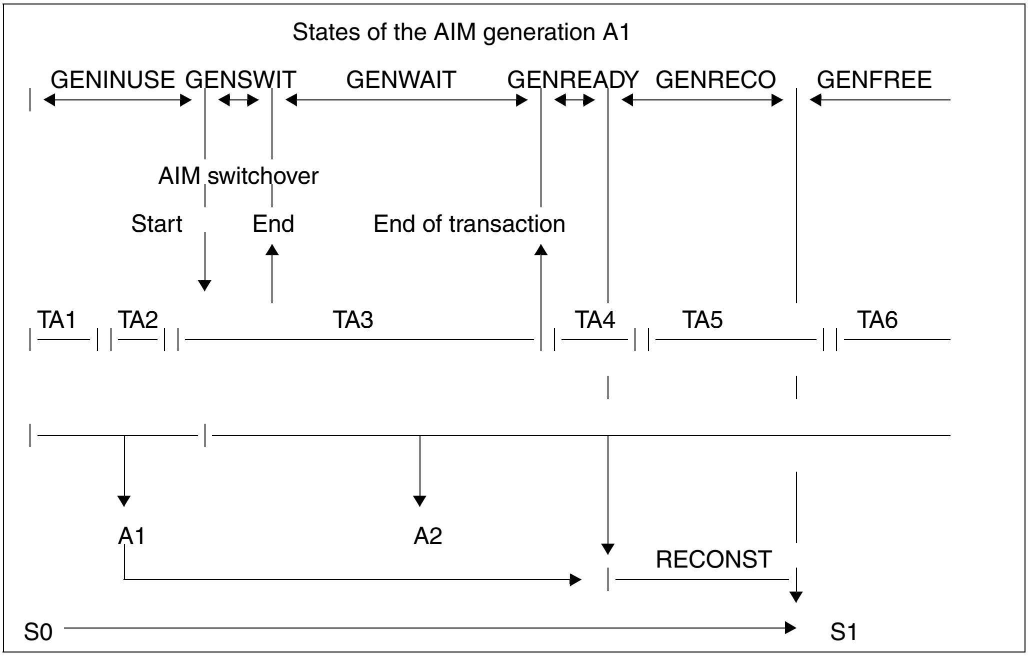


Figure 6: AIM file generation states

Explanation

A1	AIM file generation 1
A2	AIM file generation 2
S0	Shadow file, version 0
S1	Shadow file, version 1
TA1-TA6	Transactions

Starting a LEASY session

To reconstruct the AIM file generations automatically, the statement *AUT=Y* and a *REN* statement must be issued when the main task is started within the LEASY-MAINTASK utility routine. The *REN* statement for LEASY-MAINTASK causes LEASY-RECONST to be started as a **separate task** (RECONST task) by means of an *ENTER-JOB* command. The file specified in the *ENTER-JOB* command is set up by LEASY-MAINTASK only if it does not already exist.

Execution of LEASY-MAINTASK

Following analysis of parameters, LEASY-MAINTASK sets up CMMAIN, also reading in the AIM management record (for the AIM file) from the LEASY catalog. Then a file is created, if it does not already exist, using the name specified by the user in the *REN* statement; the required commands and statements are supplied and an *ENTER-JOB* command is issued using the CMD macro.

Bourse mechanisms are set up to permit communication between the main task and the RECONST task.

The AIM management record is compared with the entry in the DMS catalog and updated as necessary. A new AIM file generation is created. If the AIM file is on disk, the new generation has the size specified by the user (*ASP*=...) or automatically has the same size as the previous generation. If no generation is free, a warm start is already carried out by LEASY-RECONST at this time. Subsequently a cold or warm start is performed, depending on the specification made by the user.

The main task waits in a loop at a bourse for the following events to occur:

- If switchover to a new AIM file generation has been completed (initiated by LEASY), the AIM management record is updated and LEASY-RECONST started.
- If reconstruction has been completed (reported by LEASY-RECONST), the relevant AIM file generation is released.

If LEASY-MAINTASK is terminated using the *TERM* or *SHUT* statement of the LEASY-MASTER utility, LEASY-RECONST also terminates immediately. If the main task has been terminated with the *CLOS* statement of the LEASY-MASTER utility routine, the main task waits for the end of the last transaction before ordering the RECONST task to perform reconstruction of the AIM file generation. If the user requests switchover to the next AIM file generation, the main task checks whether it is free.

Execution of LEASY-RECONST

In a loop, the RECONST task waits for orders from the main task. The following jobs are possible:

LEASY-MASTER jobs

TERM	LEASY-MASTER statement: terminates the RECONST task immediately.
SHUT	LEASY-MASTER statement: terminates the RECONST task immediately.
CLOS	LEASY-MASTER statement: all AIM file generations with the GENREADY state are to be reconstructed.

Internal LEASY jobs

NACH	Reconstruct: all AIM file generations with the GENREADY state are to be reconstructed.
KALT	Cold start: checks whether generations have the GENINUSE, GENWAIT or GENSWIT state. If so, the cold start is rejected.
WARM	Warm start: checks the AIM management record for the state which the generations have and acts accordingly (see also warm start below).

Cold start

A cold start (normal case) is possible only if none of the generations has the GENINUSE, GENWAIT or GENSWIT status and at least one generation is free. AIM generations which have not yet been reconstructed are then reconstructed in parallel to the current operation.

Warm start

The BIM files are no longer the sole criterion for a warm start. The AIM management record includes information as to whether the last LEASY session was properly terminated. This record stores the AIM file generation states. A warm start is mandatory if an AIM file generation does not have the *GENFREE* state. LEASY-RECONST takes the following individual actions in the event of a processor failure (see figure 6 on page 74). The states refer to AIM file generation A1:

– Failure in the *GENFREE* state

  S1 is the final state of the shadow files; AIM file generation 1 has been successfully reconstructed. Transactions from generation 2 which have been completed must be entered in the shadow file, open transactions must be reset to A2; in other words LEASY-RECONST processes A2 using *MOD TRA=V*.

- Failure in the *GENRECO* state

  Reconstruction of generation A1 has not yet been completed. In A1, the only transactions open are those completed in A2. In the case of a warm start, all of A1 must be entered. A2 must be reconstructed as described above; in other words LEASY-RECONST processes A1 using *MOD TRA=A*, A2 using *MOD TRA=V*.

- Failure in the *GENREADY* state

  The equivalent of the actions in the *GENRECO* state.

- Failure in the *GENWAIT* state

  In generations A1 and A2 there are entries from transactions which were not closed at the time the failure occurred. These entries must be reset. Afterwards both generations are entered in S0; in other words LEASY-RECONST processes A1 and A2 using *MOD TRA=V*.

- Failure in the *GENSWIT* state

  The equivalent of the actions for A2 in the *GENFREE* state.

- Failure in the *GENINUSE* state

  The equivalent of the actions for A2 in the *GENFREE* state.

Following successful reconstruction, the user can copy the shadow files to the original files.

Terminating a LEASY session

To terminate the LEASY session, the LEASY-MASTER utility routine provides the statements *TERM, SHUT* and *CLOS*. These statements affect the main task, which passes them on to the RECONST task. In the case of *TERM* and *SHUT*, the LEASY-MAINTASK utility is immediately terminated, likewise LEASY-RECONST is immediately terminated. With *CLOS*,
LEASY-MAINTASK waits until LEASY-RECONST has completely processed all generations in the *GENRECO* state, releases them (*GENFREE* state), and does not finally terminate until then.

Action in the event of errors

This section describes actions to be taken if errors occur while AIM file generations are being reconstructed automatically.

- System crash

  Start the new LEASY session with a warm start.

- Errored termination of an application task

  Start the new LEASY session with a warm start.

- Errored termination of the RECONST task

  The LEASY application can continue to run as long as free generations are available. Start the new LEASY session with a warm start.

- Destruction of a file

  The LEASY session must be terminated using *CLOS*. This results in the shadow files being updated. After the LEASY session has been terminated, the file which has been destroyed can be recovered by copying the shadow file. Start the new LEASY session with a warm start.

  If the start has taken place while using automatic keeping of shadow files and no BIM save has been specified, LEASY will copy the reconstructed shadow files to the original files in the case of a warm start (last session terminated with error and with open transactions). LEASY thus performs the required rollback using the information from the AIM file and shadow files. This copying procedure may take quite some time when large files are involved. All files for which automatic keeping of shadow files was specified and which were involved in open transactions when the session was aborted will be copied.

5.3.5 Replacing original files by shadow files during ongoing operation

The shadow file strategy permits simple recovery of files which have become inconsistent or been destroyed because of an error. These files can be returned to a consistent status by replacing them with the corresponding shadow files.

The *REPO* function of the LEASY-MASTER utility routine enables the LEASY administrator to copy shadow files onto the associated original files without all the LEASY applications and the LEASY maintask having to be terminated.

Requirements

- The function is called in the main MASTER
- Operation takes place with automatic recording of shadow files

- The files concerned are specified in the **FIL* statement of the LEASY-CATALOG utility routine using *AIM= (Y,A)* or *AIM=(R,A)*
- The files are master files with access method ISAM or PAM
- Original and, if required, associated SI files are opened using SHARED-UPDATE=YES
- The *REPO* function necessitates an implicit switchover of the AIM file generation. If this is not possible because no AIM file generation is free, the *REPO* function is aborted.

The LEASY administrator enters the following specifications:

- Selection of the files concerned
- Wait time which applies for completion of all open transactions which affect the files selected
- Reaction if the wait time elapses without it being possible to complete these transactions

LEASY waits for all open transactions which affect the selected files to be completed. The automatic RECONST then brings all the associated shadow files up to date. Finally the original files involved and, if required, the associated SI files are replaced by the shadow files.

While the *REPO* function is being executed, all newly opened transactions are rejected; this can hinder LEASY applications. Implicit switchovers of the AIM file generation resulting from the AIM file size specified in the MAINTASK parameter AIS being reached are not performed either.

Depending on the reaction which has been defined, it can happen that open transactions which are not reset still exist after the wait time has elapsed. Consequently the *REPO* function cannot be executed in full, and not all the original files selected are replaced by their shadow files. This is pointed out by the message *LEA5510*.

i This case can occur in particular in the case of files for which no BIM save was defined. Transactions which process files of this type cannot be reset.

5.3.6 Manual (explicit) online backup

LEASY enables the user to save individual files explicitly using a freely selectable application. To prevent any inconsistencies from occurring, the files to be saved may not be modified during this save operation. In order to ensure this during ongoing LEASY operation, a write lock must be set for the files concerned.

The *ROMS* function in the LEASY-MASTER utility routine is used for this purpose. This allows the LEASY administrator to specify READ-ONLY mode for all or individual master files (with access method ISAM or PAM, including any SI files that exist) of a LEASY catalog during ongoing operation.

Requirements

- The function is called in the main MASTER
- The files concerned are reserved for setting READ-ONLY mode in the **CAT* or **FIL* statement of the LEASY-CATALOG utility routine using *ROM= Y*
- The files are master files with access method ISAM or PAM
- Original and, if required, associated SI files are opened using SHARED-UPDATE=YES

The LEASY administrator enters the following specifications:

- Selection of the files concerned
- Wait time which applies for completion of all open transactions which affect the files selected
- Reaction if the wait time elapses without it being possible to complete these transactions

The LEASY-MASTER utility routine waits for all open transactions which affect the selected files to be completed. It then places the selected files in READ-ONLY mode. When the *ROMS* function has been completed successfullly, a corresponding message (*LEA5512*) is issued. If necessary, other messages are issued beforehand which report that transactions were still open and that these have been reset.

After the *ROMS* function has been completed successfullly, the LEASY administrator can perform the required save operation. The LEASY administrator must then cancel the write lock again using the *ROMR* function so that write operations to the files concerned can take place again. However, the administrator may cancel the write lock only after the save has been completed, otherwise inconsistent save files could be produced. If the save operation is performed using LEASY-SAVE, this is ensured automatically because the *ROMR* function is rejected while this save operation is in progress.

Depending on the reaction which has been defined, it can happen that open transactions which are not reset still exist after the wait time has elapsed. Consequently the *REPO* function cannot be executed in full. It is therefore aborted with the message *LEA5511*. In this case saving using LEASY-SAVE is **not** possible, and a backup is performed using other means only for those files whose transactions were completed or reset.

i This case can occur in particular in the case of files for which no BIM save was defined. Transactions which process files of this type cannot be reset.

While READ-ONLY mode is set, only read operations are permitted for the files concerned. However, transactions which modify the data set and LEASY statements for transactions which were rolled back or reset after the wait time elapsed are rejected.

READ-ONLY mode cannot be set for files which have already been opened by an external user program if the initial access was by a LEASY application.

The current status of the *ROMS* function is stored in the *LEACMST job variable (see page 112).

LEASY-SAVE

The following prerequisites must be satisfied to permit the files to be saved in READ-ONLY mode using LEASY-SAVEt:

- The current LEASY catalog contains only master files with the access method ISAM or PAM.
- All master files (including the SI files) of the current LEASY catalog must be reserved for READ-ONLY mode (*ROM=Y* specification in the LEASY-CATALOG statement **CAT* or **FIL*).
- When the catalog was configured, **no** specifications were made regarding shadow files (*CPC/CPS*).
- When the *ROMS* function is executed in the LEASY-MASTER utility routine, *ALL is specified in screen mask 47 (add file for ROMS).
- The *ROMS* function has been completed successfully; no transactions are open.
- LEASY maintask has been started

6 Operating modes

The LEASY software product can be used in the following operating modes:

- Timesharing

  This mode is described below. figure 7 on page 84 and the associated notes afford a general introduction to this mode.

- Timesharing with openUTM

  This mode is described in section “Inquiry and transaction mode (openUTM)” on page 92ff. The main points are summarized in figure 9 on page 94 and the associated notes.

- Timesharing with DCAM

  This mode is described in section “Inquiry and transaction mode (DCAM)” on page 98ff.

The differences between the modes are discussed in section “Differences between openUTM and DCAM inquiry and transaction processing” on page 103.

6.1 Timesharing mode (TIAM/batch)

There is a strict 1:1 assignment between application tasks and terminals/batch programs. No more than one LEASY transaction may be open in any one user task. The *LEASY* link module must be linked to each user program. This module

- dynamically links the *LEACON* module
- forwards the operations and their operands to *LEACON*
- contains STXIT routines for handling errors.

All LEASY operations are executed in the *LEACONX* module that is loaded dynamically by *LEACON*.

The following diagram provides an overview of the LEASY system in timesharing mode.

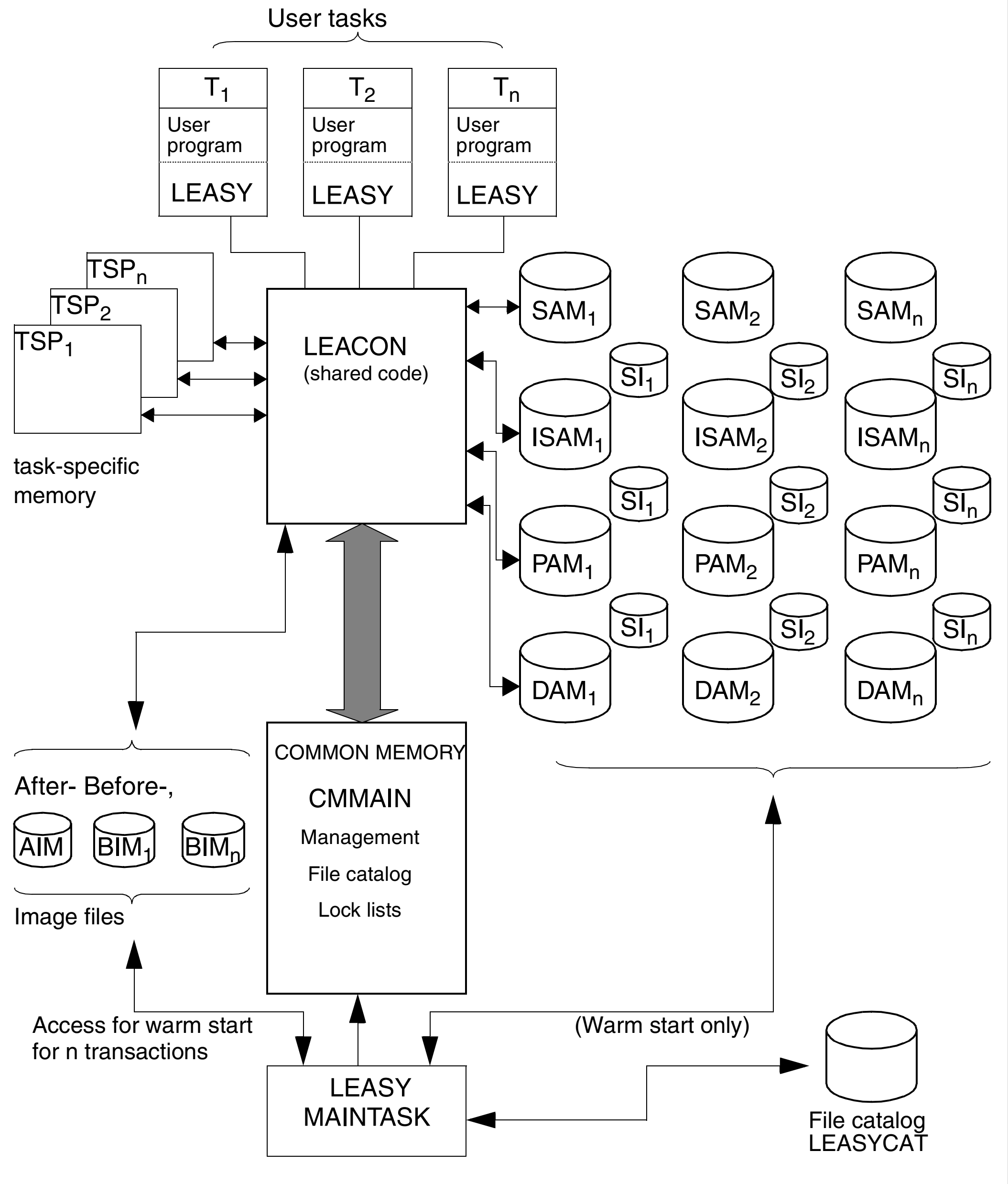


Figure 7: System overview for timesharing mode

6.1.1 Methods of opening and closing files

The *OPFL* operation can be used to open selected files and the *CLFL* operation to close selected files. Note, too, that the *OPTR* operation can also be used to open all files at the start of the program and *CLTR* to close them at the end of the program.

Timesharing without the OPFL and CLFL operations

The following is a typical sequence of operations in this mode:

```
┌─CATD                  Linkage to a LEASY catalog
│
│   ┌─OPTR              1st transaction
│   │   RHLD
│   │   REWR
│   └─CLTR
│
│   ┌─OPTR              2nd transaction
│   │   RNHD
│   │   REWR
│   ├─CLTR(OPE2=T)      Setting a restart point. 3rd transaction
│   │   RNHD
│   │   REWR
│   ├─CLTR(OPE2=T)      Setting a restart point. 4th transaction
│   │
│   └─CLTR
│
├─CATD                  Linkage to another LEASY catalog
│
│   ┌─OPTR              5th transaction
│   │
│   └─CLTR
└─
```

All files involved in a transaction are physically opened by the *OPTR* operation (DMS OPEN macro) and closed by *CLTR*. The *CLTR* operation with the additional function *OPE2=T* is an exception, since it only sets a restart point, i.e. the BIM file is declared "empty", but the file status including file positions is retained.

This operating mode is particularly suitable for batch programs which use *OPTR* at the start of the program to open all files but otherwise only set restart points (*CLTR*,*OPE2=T*) and which close all the files at the end of the program (*CLTR*).

However, it is not advisable to enclose each dialog transaction between *OPTR* and *CLTR* since the continual physical opening and closing of files has adverse effects on the runtime.

Timesharing with the OPFL and CLFL transactions

The following is an example of the structure of a typical operation sequence in timesharing mode with *OPFL/CLFL*:

```
CATD                      Linkage to a LEASY catalog
   OPFL                   Opening files in file list 1

      OPTR                1st transaction
         RHLD
         REWR
      CLTR

      OPTR                2nd transaction
         RHLD
         REWR
      CLTR

   CLFL                   Closing files in file list 1

   OPFL                   Opening files in file list 2

      OPTR                (n)th transaction
         RNXT
         INSR
         RNXT
      CLTR(OPE2=T)        Set a restart point. (n+1)th transaction
         RNXT
      CLTR(OPE2=T)        Set a restart point., (n+2)th transaction

      CLTR

      OPTR                (n+3)th transaction

      CLTR

   CLFL                   Close files in file list 2

CATD                      Link to another LEASY catalog possible

   OPFL

      OPTR

      CLTR
   .
   .
   .
   CLFL
```

The files are physically opened by *OPFL* (DMS OPEN macro).

Each transaction is enclosed by *OPTR* and *CLTR*, which only logically open and close the files. To close the files physically, *CLFL* must be specified.

This operating mode is particularly suitable for complex dialog applications which, in timesharing mode, process a variety of files with different USAGE modes in the individual modules.

i It is possible to use the operating modes alternately, both with and without *OPFL/CLFL* operations.

6.1.2 File access via the I/O task

File access operations are executed in the *LEACONX* module.

In timesharing mode a distinction is made between

- file access in the application task and
- file access in the I/O task.

File access in the application task

The *LEACON* module links the *LEACONX* module to the application program. LEASY can handle a maximum of 255 transactions at the same time. This means that the number of terminals that can be connected and the number of batch programs is restricted to 255.

Reservation of address space for file buffers in each application task can lead to high paging rates.

File access in the I/O task

The *LEACONX* module is no longer in each application program, but (under the name LEAICNX) in one or more separate tasks (I/O tasks). LEASY calls of the application program are no longer passed on by the *LEACON* link module with the aid of subroutine calls but are transferred by means of intertask communication to *LEAICNX* where they are processed and returned to the user.

Up to 1800 application tasks can communicate with LEASY.

Communication between the application program and the I/O task is handled by modules supplied with the *SYSLNK.LEASY.062.IOH* library. For reasons of compatibility these modules also have the names *LEASY* and *LEACON*, as in the version in which *LEACON* is linked dynamically to each application program ("linked-in version").

Users wanting to work with the I/O handler must link in the *LEASY* module from the *SYSLNK.LEASY.062.IOH* library. The *LEASY* module then loads the *LEACON* module from the *SYSLNK.LEASY.062.IOH* library into class 6 memory.

Restrictions at the LEASY interface

The I/O task calls LEASY internally via the DCAM interface. Consequently all restrictions affecting the DCAM interface also apply to the I/O handler. The restrictions are as follows:

- Load mode for DAM:
  each new record is in a new block.
- SAM files can only be read.
- Foreign files can only be read.
- Temporary files are not permitted.

Waiting time for locked records

An I/O task which is waiting for a record lock to be released during the processing of a user job is not available for the processing of other jobs. This is particularly important when there is only one I/O task. This task waits for a lock to be released that it can only release itself. No jobs from other users are processed during the waiting time, i.e. all users have to wait. The user should ensure that the maximum waiting time in the *RE* area is set to 0 when there is only one active I/O task.

Simultaneous operation with and without the I/O handler

Application programs can run simultaneously with and without the I/O handler. This distinction is made in the dynamic loading of *LEACON*.

LEASY operations

If you are working with LEASY for the first time you should familiarize yourself with the LEASY interface before tackling this section. The description of the LEASY interface and all its areas, fields and operations is to be found in chapter “Overview of the LEASY program interface” on page 119ff.

CATD and TERM operations

The *CATD* operation links the application program to the common memory generated by the main task. The *CATD* operation is not passed on to the I/O task. If a blank is transferred as catalog information (*CAT*) for the *CATD* operation, the application program is disconnected from the common memory. The *TERM* operation is converted to the *CLTR* operation with rollback (*OPE1=R*) for an open transaction and then transferred to the I/O task. Otherwise the operation functions like a *CATD* operation with a blank for the catalog information (*CAT*).
If a suffix name is specified for the catalog name, it must be the same as the suffix name in the *CAT* statement of the LEASY-IOTASK utility routine.

OPFL and CLFL operations

The *OPFL* and *CLFL* operations are not executed; nor is a syntax check carried out. The return code issued is always *000LL000*. This enables applications with a central *OPFL/CLFL* to be converted to the I/O handler without changes. This does not apply if alternating OPEN modes are required for processing (particularly in the case of SAM files).

The *OPFL* operation is initiated when the I/O task is started. It therefore applies to all users working with LEASY via the I/O task. The USAGE modes of the transactions must be compatible with the OPEN modes when the I/O task is started.

The files are not closed until the I/O task has been terminated. Thus the OPEN mode cannot be changed during a session with the I/O task.

Transaction operations

All the other operations are mapped to the LEASY-DCAM interface by the *LEACON* module of the *LEASY.SYSLNK.062.IOH* library and sent to the I/O task for processing.
The *OPFL* operation which created the I/O task from the *OPF* statements of the LEASY-IOTASK utility routine and initiated it at the start applies to all operations - and particularly to *OPTR*. SAM files can, for example, only be read in one direction for each session with the I/O task.

Handling of record lengths in the I/O task

How the AR area is sent from the user task to the I/O task and back depends on the operation involved.

Operation	**Record length User task→ I/O task**	**Record length I/O task → user task**
SETL, DLET, LOCK, UNLK	Length specified in I/O task	---
INSR	Current record length	Current record length
RDIR, RHLD	Length specified in I/O task	Current record length
RNXT, RNHD, RPRI, RPHD	---	Current record length
REWR, STOR	Current record length	---

Table 5: Handling of record lengths in the I/O task

If the record format is fixed, the current record length is specified in *RECSIZE*. If it is variable, the record length is read from the first two bytes of the AR area.

6.2 Inquiry and transaction mode (openUTM)

This section describes openUTM-LEASY applications and presupposes familiarity with openUTM.

Generation and structure

If a openUTM application is to be integrated with LEASY, this must be taken into account during generation of the openUTM linkage program (KDCROOT) by means of the following macro:

`KDCDBL.`

The structure of a linked openUTM LEASY application program is as follows:

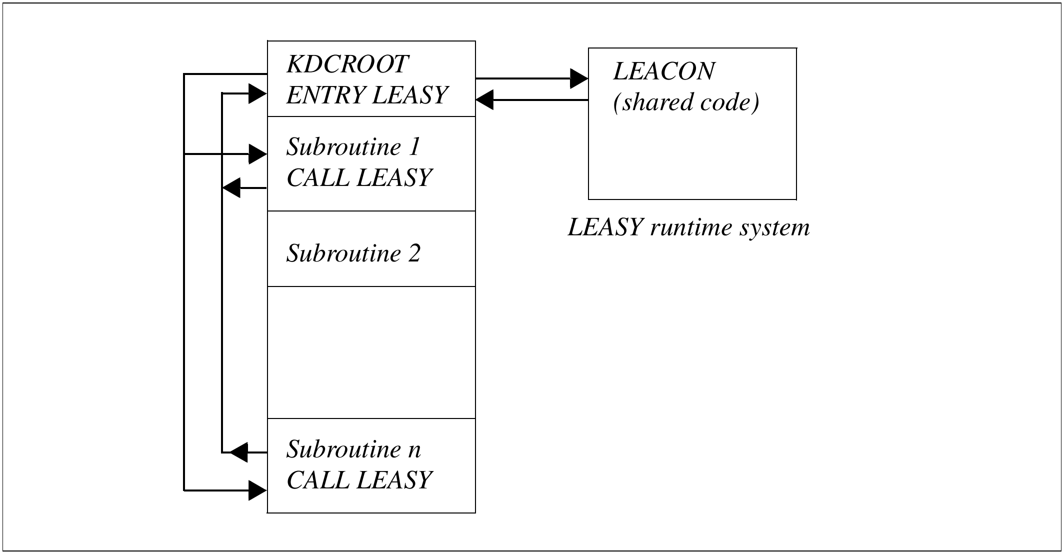


Figure 8: Structure of a openUTM-LEASY application

Each *CALL "LEASY"* in the openUTM subroutines is routed via an ENTRY LEASY in the KDCROOT; the KDCROOT passes the call on to the LEASY runtime system, and also supplies the addresses of a transaction-specific memory and a task-specific memory for each call to LEACON.

The KDCROOT contains all error recovery routines (e.g. STXIT routine) and performs the actions required if errors occur (*CLTR*, *OPE1=R* and *PEND ER*).

KDCROOT also issues calls to LEACON that are not initiated by CALL "LEASY", e.g. in the case of *PEND KP*, in support of multi-step transactions.

openUTM and LEASY share a common transaction concept. The following action is taken by KDCROOT to ensure a common checkpoint at the end of the transaction:

- The LEASY operation *CLTR* is passed on to *LEACON*; *LEACON* accepts the call but does not execute it (*CLTR* operations are optional).
- Not until subsequent *PEND* processing is the LEASY transaction terminated via a special internal call to *LEACON*.

Task and memory structure

The differences for openUTM LEASY applications compared with timesharing mode are as follows:

- The *LEASY* LINK module is omitted; instead KDCROOT provides the link to the runtime system *LEACON*.
- To permit multi-step transactions (task changeover between individual dialog steps), *LEACONX* buffers the current BIM buffer in the common memory CMMAIN when a task is switched (openUTM call *PEND KP*). At the start of the next dialog step this data is transferred from the common memory to the current task area. A task changeover can also be implemented in single-step transactions by means of the *PEND/PA* and *PEND/PR* operations, providing the user uses the TAC class system.

The schematic diagram below shows the user and LEASY modules and their memory assignment.

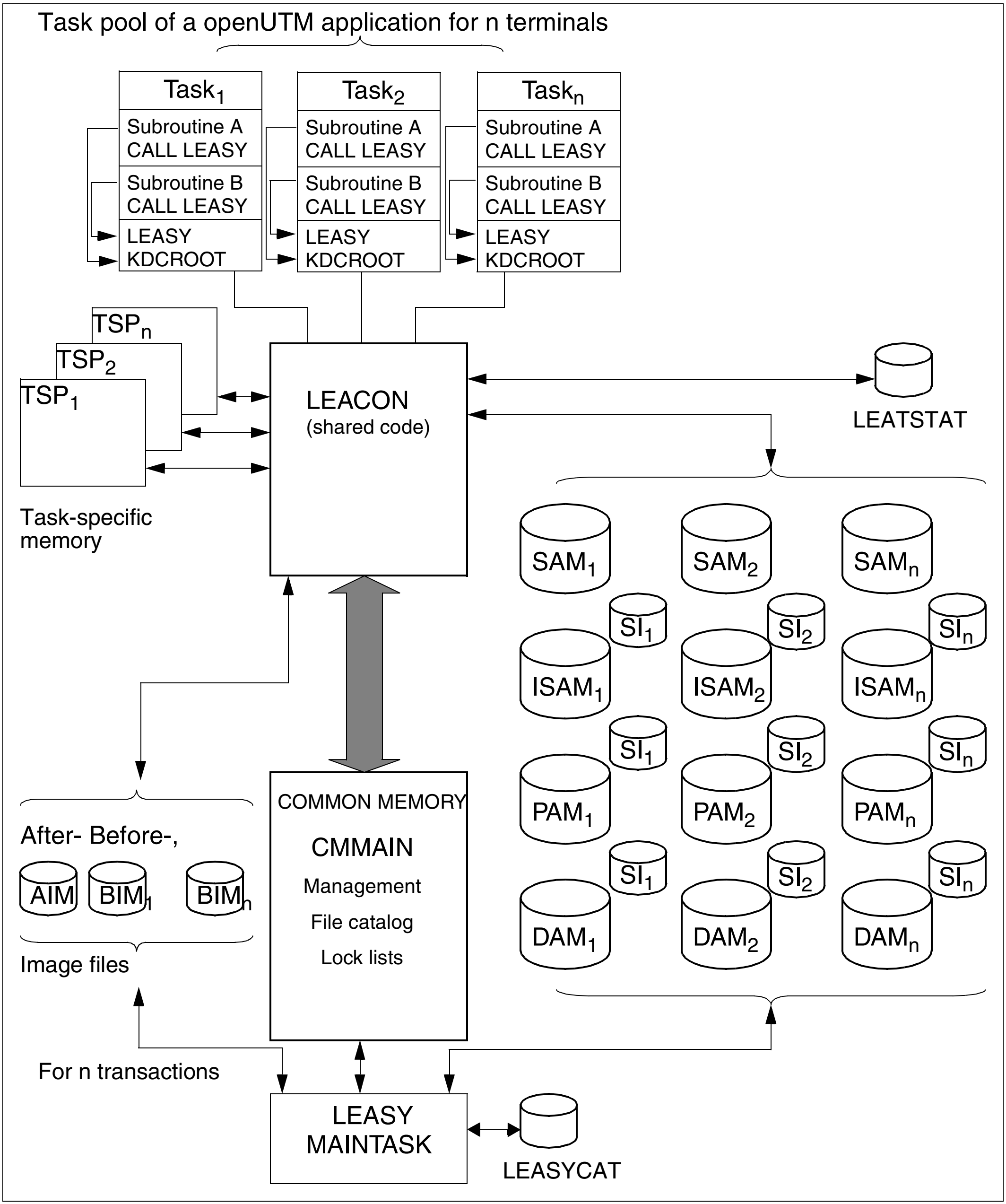


Figure 9: System overview of a LEASY-openUTM application

Starting a openUTM LEASY application

The *START-EXECUTABLE-PROGRAM* command is used to call the openUTM application program.

At present LEASY recognizes 2 start parameters for openUTM:

- Specification of the LEASY catalog

  ```
  .LEASY CATD=[:catid:][$userid.]file-catalog[.suffix]
  ```

  This start parameter establishes a link to the common memory CMMAIN during the startup phase of KDCROOT.

- Specification of the files to be opened

  ```
  .LEASY OPFL=((file1,mod1),...)
  ```

  The file or files are transferred to the openUTM application program in the DB4 format of the file allocation (see page 135ff).

  The start parameter may be repeated, but each file may only appear once in the complete list of the start parameters. All file specifications are entered in a common table, and are subject to an implicit *OPFL* statement during the startup phase, so that all files are physically open in the same manner in all tasks at the time of the first openUTM conversation.

- Specification of whether parameter passing is to take place according to ILCS conventions

  ```
  .LEASY ILCS
  ```

  Further information on parameter passing in accordance with ILCS conventions is provided in the sections “Linking LEASY” on page 119 and “Calling LEASY” on page 121.

LEASY status file in a openUTM environment

In certain cases openUTM requires information from the file access system on the status of individual transactions. LEASY stores this information in a special file - one for each LEASY catalog. This is an ISAM file with the name:

```
file-catalog.LEATSTAT
```

It is written by the LEASY-MAINTASK or LEASY-RECONST utility routine when rolling back transactions from a openUTM environment.

Unless otherwise specified the status file is set up on public volumes. If it is to be written to a private volume, one of the following commands must be issued before the first warm start or reconstruction run in which the file is accessed:

```
/CREATE-FILE file-catalog.LEATSTAT,
SUPPORT=*PRIVATE-DISK(VOL=vsn,DEV-TYPE=device)
```

or

```
/CREATE-FILE file-catalog.LEATSTAT,SUPPORT=*TAPE(VOL=vsn,DEV-TYPE=device)
```

The file only exists if it is created by the user, or if created automatically by a LEASY utility routine with a rollback. An internally allocated password is provided for security reasons.

Restrictions compared with timesharing mode

- It is impossible to change the LEASY catalog because the *CATD* call is executed by openUTM itself during the task initialization phase. *CATD* is not permitted in application programs.
- The *OPFL* operation is only called once per application. When openUTM is used, the appropriate file attributes must be entered for the application in the start parameter *OPFL*. This results in an implicit *OPFL* call during the startup phase. It is forbidden to use *OPFL* and *CLFL* in the user programs in conjunction with openUTM.
- The *CLTR* operation with *OPE2=T* is not permitted since this would be inconsistent with the openUTM transaction concept.
- SAM files are read-only, because DMS does not permit SAM files to be opened in write mode by several tasks simultaneously (openUTM task pool). (Instead an ISAM file with USAGE mode = *LOAD* may be used.)
- Temporary files are assigned by the TSN to the task and not to the transaction or terminal. They are therefore unsuitable for use in a openUTM application due to the application’s task pool, which would assign an undefinable number of different tasks - and thus different physical files - to one logical file. Temporary files are therefore rejected in an *OPFL* operation and an error code (*UTMLLU13*) is issued.
- The LEASY runtime system does not maintain foreign files in common memory CMMAIN or keep lock protocols. Therefore it is not possible for several users to write simultaneously; only the OPEN modes "1" (SAM, ISAM, PAM) and "5" (SAM) are permitted.
- BIM may only be deactivated for read transactions.
- I/O tasks cannot be used in inquiry-and-transaction mode.

Diagnostic information in the openUTM-DB-DIAGAREA

- openUTM documents events which have occurred in task-specific trace areas which are written cyclically. Requests for the LEASY system are documents in the DB-DIAGAREA (see the manual “openUTM Messages, Debugging and Diagnostics”, DB-DIAGAREA).

- LEASY places data concerning the individual request in a 32-byte field (“Secondary DB Trace Information”) in a trace record of the DB-DIAGAREA. This information is used by the Customer Service to facilitate diagnosis when problems occur.

6.3 Inquiry and transaction mode (DCAM)

A DCAM interface is provided by LEASY for special functions in **inquiry and transaction mode** over and above the range of functions of openUTM.

Execution of a DCAM application with LEASY

A DCAM application has two sections:

- a monitor, which handles control and forwarding of messages,
- application program modules, which are used, for example, to access data.

Data access is transaction-oriented at file level with the aid of LEASY.

In order to permit its range of applications to remain as wide as possible, the design of the interface is such that within transactions the control on the DCAM side can be transferred to other transactions at any time.

DCAM can be used in the following ways:

- One DCAM program services several data terminals with interleaved LEASY transactions in several interleaved tasks (cf. openUTM single-task operation with multi-step transactions); after each LEASY operation the transaction which is currently being processed is interrupted, and processing continues with another transaction, which may have been interrupted earlier.

  In such cases files can also be opened without using SHARED UPDATE, providing they are used exclusively in this DCAM application.

- Several DCAM programs service several data terminals with interleaved LEASY transactions in several tasks (cf. openUTM multi-task operation with multi-step transactions); it is possible not only to interrupt and continue processing of transactions within a task, but also to transfer control of processing to any other task.

  It is essential to ensure that each task can execute the LEASY operations *CATD* and *OPFL* without errors before permitting it to process transactions. These operations must be **absolutely identical** for all tasks of a DCAM operation. In the case of *OPFL*, the file sequence must also be identical. This can be ensured, for example, by using a parameter file in all tasks of the application or by means of a shared module. All files to be opened must be specified in an *OPFL* operation.
  Files opened for write processing can only be defined using the LEASY access methods ISAM, PAM or DAM, and must be opened using SHARED UPDATE.

In the case of the *CATD* operation, LEASY must have access to the DCAM application name.

The field in which the transaction identifier will later be entered must be erased prior to the first *OPTR* operation of each transaction. LEASY returns the transaction identifier in this field when the transaction is opened. This identifier must be supplied for each LEASY operation affecting this transaction.

The DCAM monitor section must therefore maintain an internal management table, in which it manages and updates the terminal address and the associated transaction identifier.

This identifier field of the LEASY interface can be erased by means of a *CLTR* operation. The DCAM monitor section must likewise erase the transaction identifier for this screen in its management table.

The following LEASY sequence of operations must be used to log off a DCAM task correctly:

`CLTR [R]`	(in single-task mode for all transactions)
`CLFL`	
`CATD`	with blanks in the field for the LEASY catalog name

When DAM files are processed, **every** LEASY data operation and *CLTR* can lead to a rollback procedure for the LEASY transaction. If the rollback procedure is terminated correctly, the transaction identifier in the *IDE* field of the *RE* area is likewise erased.

Some action functions of the LEASY-MASTER utility routine initiate return codes at the LEASY program interface. This method can also be used to initiate rollback procedures for LEASY transactions in conjunction with any data operation.

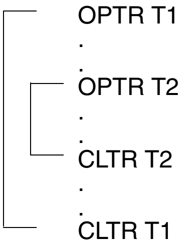
LEADCAM link module

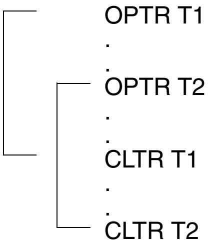
The *LEADCAM* link module is prescribed for DCAM applications in place of the *LEASY* module, in order to permit easier handling. This module is available in the *SYSLNK.LEASY.062.DCAM* library. It contains the LEASY entry address, thereby allowing the user to call LEASY via a subroutine call (*CALL*).

This *LEADCAM* link module contains STXIT routines which are designed for standard applications.

The LEASY-DCAM link permits the user to interleave transactions, even in timesharing mode, by using the *LEADCAM* module instead of the *LEASY* link module and by defining the interface as described above. The transactions may be interleaved either within one another or offset.

Example





The LEASY operations following the second *OPTR* call do **not** belong to both transactions, but merely to the transaction whose identifier they contain. The otherwise independent transactions are thus interleaved with an offset with corresponding results, e.g. with regard to the number of transactions, the lock logic, etc.

This interleaving is not the same as the facility for adding file paths to an existing transaction by means of further *OPTR* operations after the start of the transaction, which has been available since LEASY V3.0.

LEASY operations

If you are working with LEASY for the first time you should familiarize yourself with the LEASY interface before tackling this section. A description of the LEASY interface and all its areas, fields and operations can be found in chapter “Overview of the LEASY program interface” on page 119ff.

CATD operation

The CATD operation is mandatory. In addition to the other parameters, DCAM programs must supply the DCAM application name in the *IDE* field of the *RE* area when logging on to a LEASY application. The application name must not be made up entirely of blanks or binary zeros, though all other bit combinations are permitted. However, only printable characters can be represented in the LEASY utility routines which output the application name to the terminal or to a printer.
The field contents are erased by LEASY following successful execution.

OPFL and CLFL operations

The *OPFL* operation is mandatory when a DCAM program is started following the *CATD* operation; all files to be opened must be specified in a **single** operation.

This ensures that the program reaches the first synchronous point; this point must likewise be reached by all other tasks of the DCAM application. If one or more tasks are already at this first synchronous point (*CATD* and *OPFL*) and are already processing transactions, further tasks can still be started for this application.

In single-task mode this first synchronous point cannot be left by means of any *OPFL/CLFL* operation. If a further task is to be started in addition to such a DCAM task for the same application, the current status of all files of the first task must be reached with an *OPFL* statement.

If in multi-task mode a task destroys this task synchronization by executing **one** *OPFL* or *CLFL* operation, no further tasks can be started for this application from this point onwards. Moreover, this next synchronous point must again be reached by all tasks of the application in the same manner. Only then can synchronism be violated by a task again.

It is not permissible for a task to violate the synchronism by executing several *OPFL* or *CLFL* operations.

In the event of a synchronization error during the *CLFL* operation (return code *DCALLU16*), it will only be possible to execute the *TERM* operation in this task, and only in a user STXIT routine. The task should then be terminated.

It is only permissible to leave the first and all subsequent synchronous points by means of a task if only **one** transaction is open in the **entire** application.

OPTR operation

When the first *OPTR* operation of each LEASY transaction is executed, the *IDE* field in the *RE* area must first be erased (overwritten with binary zeros) and then transferred. If the operation is executed successfully, LEASY returns the transaction identifier in the same field; this identifier must then be supplied for all operations of this transaction (further *OPTR* operations and *CLTR*).

CLTR operation

The transaction identifier must be transferred in the *IDE* field. If the operation is executed successfully, *IDE* will be erased, providing *OPE2=T* has not been set.

CINF operation

The transaction identifier must be transferred in the *IDE* field for operations at file level and for *CINF*.

TERM operation

With DCAM programs the *TERM* operation is only permitted in a user STXIT routine. No further LEASY operations are permitted following a *TERM* operation. The task should be terminated.

If the application program can make a valid transaction identifier available for this operation in the *IDE* field of the *RE* area, this transaction will be rolled back. If the application program cannot specify a valid transaction identifier, the *IDE* field must be erased. In this case no transactions are rolled back.

If the task was the only one for the DCAM application, any open transactions (except the one whose transaction identifier was specified) remain open. These transactions are not rolled back until the LEASY-MAINTASK utility routine is warm-started.

Error recovery

The LEASY operation *TERM* may only be used when logging off from the **last** remaining open transaction, since when used for other transactions, it suppresses further rollback procedures (this operation can also be used to log off a task from common memory CMMAIN).

In case of error, the following applies:

- If several tasks have been activated for the relevant DCAM applications, only a LEASY transaction active in the task (if any) will be rolled back.
- If only the task with the errored program has been activated for the application, all LEASY transactions involved in the DCAM application will be rolled back.
- LEASY transactions are rolled back by the LEASY STXIT routine.

i For further information on STXIT routines see page 104ff.

6.4 Differences between openUTM and DCAM inquiry and transaction processing

There is no common transaction system for DCAM applications and LEASY. The DCAM programs (monitor section) are entirely responsible for supervisory control in the program. The return codes which can be used for control purposes are made available by LEASY in the *RE* area. There is no LEASY status file in a DCAM environment. Write transactions without BIM saving are permitted.

The *CATD* and *OPFL* operations must be used before the task can process LEASY transactions. In multi-task mode it is essential to ensure that all tasks of the application have opened the same files in the same manner and the same order (this requirement that the files for all tasks of an application be opened in the same order also applies to openUTM inquiry and transaction mode). Violations of this requirement are acknowledged by means of error code *DCALLU16*.

Changing the LEASY catalog in single-task mode is permitted without restriction. When the LEASY catalog is changed in multi-task mode within the DCAM application, all tasks must be synchronized, in order to ensure that all LEASY transactions and all files are closed throughout the application.

The *CLFL* and *OPFL* operations are permitted when transactions are closed. Closing and opening of files must be synchronized step by step for each LEASY operation in the same manner as when changing the LEASY catalog.

Violations of synchronization when executing *OPFL* and *CLFL* operations are acknowledged by means of error code *DCALLU16*. Following a synchronization error in conjunction with *CLFL* the task is locked to prevent it from being used again. All operations with the exception of *TERM* are likewise rejected with *DCALLU16*.

The *CLTR* operation is permitted with *OPE2=T*.

DCAM programs must have linked in the *LEADCAM* link module from the *SYSLNK.LEASY.062.DCAM* library.

6.5 Error recovery via the LEASY STXIT routine

The *LEASY* link module contains a STXIT routine for the recovery of errors. This routine is activated by the first LEASY call. The following event classes are handled:

● ABEND	abnormal program termination
● ERROR	unrecoverable program error
● PROCHK	program checking
● RUNOUT	end of program runtime
● TERM	normal program termination (*TERM/TERM DUMP=Y*)

After a dump has been taken and any open LEASY transaction rolled back by *TERM*, the program run is also interrupted (*BKPT* macro) in the STXIT routines in interactive mode, though not in procedure and batch modes. This enables user programs to be tested more efficiently.

The STXIT routine performs the following actions:

- outputs a message to SYSOUT specifying interrupt weight and program count (at time of error)
- effects PDUMP for the whole program (apart from the TERM event class)
- calls *CLTR* with rollback; this prevents lock entries relating to files or records being retained in common memory
- outputs the LEASY return codes of the *CLTR* operation (with suffix *OPE1=R*) to SYSOUT
- effects PDUMP if rollback processing (call *CLTR OPE1=R*) contained errors
- to support error diagnosis the register statuses are set at the time of the program error and the program run is interrupted in interactive mode (*BKPT* macro).
- if an STXIT routine was logged on for the event prior to the first LEASY call, this routine is activated; otherwise the program terminates with

  ```
  TERM MODE=ABNORMAL,UNIT=STEP
  ```

In the case of the event class TERM, the above actions depend on the presence of an open transaction, otherwise the program terminates without any further action being taken.

7 LEASY in a multiprocessor environment

LEASY can also be implemented in an MRS multiprocessor network.

If a LEASY catalog and its LEASY system files are to be accessed on a foreign processor, the catalog identifier (*catid*) of this processor must be specified as part of the DMS file name in the LEASY utility routines (except LEASY-MASTER) and at the user interface. This also applies when specifying explicit DMS file names for application files, or when accessing them.

Format of these file names:

`:catid:$userid.file`

A LEASY catalog is uniquely identified by

`:catid:$userid.file-catalog`

In other words, catalogs which are in different processors, or which are in the same processor but have the same or a different user ID, are nevertheless completely independent of each other.

If a file is given a catalog identifier together with its name, but the MRS is not available, DMS will reject the call.

7.1 LEASY system files in multiprocessor systems

At present the following LEASY system files are available, depending on the LEASY function selected:

- `file-catalog.LEASYCAT`
- `file-catalog.LEASYAIM`
- `file-catalog.BIM#.nnn`
- `file-catalog.LEATSTAT`

Some of these files are set up by the user (e.g. LEASYAIM); others are set up by LEASY utility routines. All LEASY system files belonging to a single file catalog must be on one processor.

7.2 Shareable private disks

User files and LEASY system files can always be stored on shareable private disks as an alternative.

However, a DMS file must not be a member of more than one LEASY catalog.

Problems may also arise if LEASY sessions involving a single catalog file are run alternately on different processors.

7.3 Remote file access and load distribution in an MRS network

Remote file access (RFA) allows distributed user file storage, i.e. in a multiprocessor system user files which logically belong to a single LEASY file catalog can be stored on different processors.

When creating these files a *catid* must be specified in the *NAM* operand of the **FIL* statement in the LEASY-CATALOG utility routine (applies to LEASY master files only), and the *CID* operand of the **CAT* statement must be set to *CID=Y* (default value).

Secondary index files are always on the processor in which the associated primary file is cataloged.

The LEASY-LOADSI, LEASY-MAINTASK and LEASY-RECONST utility routines can also be started from a processor which does not contain the catalog file. In this case a *catid* must be specified in the **CAT* statement (or in the catalog specification of the LEASY-LOADSI utility routine).

Common memory CMMAIN is then also created in the processor on which the LEASY-MAINTASK utility routine is started. Hence locking management is effected via the CMMAIN on the processor on which the main task is started, and not necessarily on the processor on which the LEASY catalog file has been created.

Since the LEASY-MASTER and LEASY-RECONST utility routines access CMMAIN, they must likewise be run on the processor on which LEASY-MAINTASK was started.

To enable application programs to access common memory CMMAIN the following measures must be taken:

- the application programs must be started on the processor on which common memory has been created
- if the processor with common memory CMMAIN is not the one on which the LEASY catalog is stored, a :*catid*: must be specified
  - in the third operand of the LEASY call or
  - in the *CATD* start parameter when in openUTM mode

  when the LEASY catalog is specified.

These features can be used for load distribution in an MRS network.

The LEASY-RECONST utility routine can also be started on a processor other than that on which the catalog file has been created. In this case, a :*catid*: must be specified in the **CAT* statement. The internal storage of the file names ensures that here too the files are correctly reconstructed.

7.4 File consistency in an MRS network

Files cannot be transferred to another processor because the file names in the AIM and BIM files contain the catalog identifier *catid*. Transferring the file would change this catalog identifier and the file would no longer be accessible via the AIM and BIM files.

Users must ensure:

- that all files required for a reconstruction run are on available processors, i.e. check for an MRS environment,
- before performing a warm start of LEASY-MAINTASK with open transactions, that all files are stored on the processor on which they were to be found in the last LEASY session,
- before a reconstruction run with LEASY-RECONST, that all files are on the processor on which they were found by LEASY when updates were made during the life of the AIM file.

Existing LEASY applications which make use of the load distribution facilities in an MRS network can be converted to single-processor operation at any time. This involves the following steps:

- All LEASY system and user files must be entered in the DMS catalog whose catalog identifier is the default value for the DMS of the relevant processor.
- The LEASY-CATALOG utility routine is started, and the *CID* operand of the **CAT* statement set to *CID=N*. Alternatively the catalog identifier can be removed from the path name using the *OLDL* and *NEWL* operands.

8 Special requirements for the use of LEASY

This chapter covers a number of topics:

The first section describes how you should proceed when planning a LEASY operation in order to optimize performance.

In section “Using job variables” on page 111ff you will find a description of how job variables make it easier to use LEASY.

In section “Addressing mode” on page 117 you will find useful information on batch/TIAM and DCAM addressing.

8.1 Planning a LEASY operation

Note the following points in connection with planning a LEASY operation:

Parallel interactive/batch processing

LEASY does not distinguish at the program interface between interactive and batch user programs. Consequently, the programming of the interface is the same for both operating modes. Parallel execution of the two operating modes is possible.

However, when deciding whether batch processing is to be performed in parallel with one or more interactive applications a number of factors must be taken into consideration:

- Since interactive applications are generally time-critical, every unnecessary additional load on the processor should be avoided. This can usually be achieved by means of supervisory measures in the computer center; parallel batch processing should be considered only if this is not possible.
- Batch programs executing in parallel with interactive applications can contain only small transactions, so that the delay in execution of interactive programs caused by locks is kept to a minimum. Consequently, batch programs will likewise contain several transactions. If a batch program is aborted, e.g. because of a program error, LEASY rolls back only the currently open transaction. This means that the data is consistent again, albeit somewhere in the middle of batch processing. Batch programs must therefore be programmed either as restartable or repeatable with regard to the data.

Enhancing performance for OPFL and OPTR

If there are a great many files to be processed, it is best to enter the names of these files in alphabetical order to keep the CPU overhead to a minimum when creating and checking the file elements.

LEASY runtime system

The *OPTR* operation provides one possibility of improving the program runtime by means of the CALL interface. By specifying *N* for the *OPE-LOG* field in the *RE* area, BIM saving is suppressed for the transaction.

Note, however, that this option should be used only for batch programs which are executing exclusively and which are not restartable. Otherwise there is no way to avoid terminating and restarting the LEASY-MAINTASK utility routine with a new set of parameters.

! Data inconsistencies caused by program aborts can then no longer be rectified with a LEASY warm start due to the missing BIM files.

The following situations should be avoided:

- constantly opening and closing user files by omitting the *OPFL* operation
- unnecessarily opening files with *SHARED-UPDATE*
- unnecessarily specifying unused files in the *OPFL* and *OPTR* operations
- selecting the wrong lock level (e.g. a file lock although a lock at record level would suffice).

Another factor of crucial importance for rapid program execution is the sequence of operations within transactions. In this respect, the following combinations of operations should be avoided:

- unnecessary locking of records
- repeated reading of the same record in a transaction
- unnecessary repositioning (explicitly via *SETL* and *RDIR/RHLD* operations, implicitly also via *RNXT/RNHD/RPRI/RPHD* operations by changing the file identifier).

In order to improve performance when writing an ISAM file with the USAGE modes *LOAD* and *LDUP*, users can insert a record with the key *X'FF...FF'*. This record must be written with a USAGE mode that does not entail LEASY assigning the key itself, for example *UPDT*. The record is then ignored by LEASY for key assignment with the USAGE modes *LOAD* and *LDUP*. The result is that when a record is inserted it is not necessary to correct all the index levels of ISAM. See “Explanation of USAGE modes LOAD/PLOD/ELOD and LDUP/PLUP/ELUP” on page 186f.

8.2 Using job variables

LEASY offers the following job variables for monitoring central resources:

*LEACMST	state of common memory
*LEAIOST	number of active I/O tasks

*LEACMST: state of common memory in a job variable

The user can employ a user job variable to start a LEASY application in procedures. Information on the state of common memory is stored in the job variable and can be accessed by the user. Entries during execution of the utility routines LEASY-MAINTASK, LEASY-MASTER and LEASY-RECONST are stored in the job variable.

User action

The user must take the following action:

1. Catalog a job variable by means of a *CREATE-JV* command.

   ```
   /CREATE-JV jvname
   ```

   The job variable name (*jvname*) can be selected by the user.

2. Assign the link name (*LEACMST*) of the job variable by means of the *SET-JV-LINK* command.

   ```
   /SET-JV-LINK LINK=LEACMST,JV-NAME=jvname
   ```

3. Use a *MODIFY-JV* command to preset the job variable with a value which has a length of 50. The freely selectable value with a length of 50 must be entered as of start position 1.

   ```
   /MODIFY-JV (jvname,1,50),SET-VAL=C'...'
   ```

The user can then have the information (which has been transferred by LEASY) output from the job variable by means of the *SHOW-JV* command or the *GETJV* macro.

Action performed by LEASY

When the main LEASY task is started, the job variable is first overwritten with blanks for a length of 50 bytes. Entries which display the current state of the common memory are then made consecutively in the job variable. Information is sorted according to entry and stored by LEASY at five positions within the job variable. Each entry is 10 bytes long. The total length of all 5 entries is 50 bytes. The following table shows the contents of the five entries.

Entry	**1**	**2**	**3**	**4**	**5**
Bytes	1 - 10	11 - 20	21 - 30	31 - 40	41-50
Meaning	State of common memory	Switch AIM file generation	Reconstruction	PETR processing	State of ROMS
Content	INIT NORMAL NOT ACTIVE	IN ACTION IN ERROR END (BLANKS)	END ALL END VALID READY	ACTIVE FINISHED WAITING (BLANKS)	ACTIVE ERROR READY END

Notes on the table

Entry 1	The information on the general state of the common memory (CM) is stored here.
INIT	Initialization of CM is still in progress.
NORMAL	The main task is executing normally.
NOT ACTIVE	The main task is not active. Either it was terminated by LEASY-MASTER or an error has occurred.
Entry 2	The information on AIM file switching is stored here.
IN ACTION	AIM file generation switching is in progress.
IN ERROR	AIM file generation switching contained errors.
END	AIM file generation switching has been terminated properly. At this time the AIM file generation might not yet have been released.
(BLANKS)	The field is deleted after termination.
Entry 3	The information on the reconstruction run is stored here. This entry is updated only if reconstruction takes place (*MOD UPD=YES*) and CMMAIN is not released after reconstruction (*MOD FRE=YES*).
END ALL	End of AIM reconstruction in which all transactions were processed.
END VALID	End of AIM reconstruction in which all valid transactions were processed.

READY	Ready for AIM reconstruction.
Entry 4	Information on transactions which have been terminated provisionally, i.e. transactions of PETR processing (PETR = "preliminary ended transactions"), is stored here.
ACTIVE	PETR processing is active.
FINISHED	All provisionally terminated transactions were processed.
WAITING	Common memory is waiting for PETR processing.
(BLANKS)	When PETR processing is complete, the field is deleted.
Entry 5	The current state of the *ROMS* function is stored here.
ACTIVE	*ROMS* function was started but not yet terminated.
ERROR	*ROMS* function was aborted. A subsequent backup run is not possible any more.
READY	*ROMS* function was terminated normally, i.e. READ-ONLY mode was set. Online backup of the selected files can be started with a backup tool chosen by the user.
END	READ ONLY modus was reset successfully by the *ROMR* function.

If no job variable has been cataloged or assigned, the start of the LEASY application is **not** interrupted.

State and result of a reconstruction run

If a reconstruction run is to be successful, the main task must be started in one of the two following modes:

*USE=RECONST	Applications cannot execute; only reconstruction is possible. This mode is permissible for original and shadow files.
*USE=NORMAL	Applications can execute. Shadow files can be reconstructed manually or automatically parallel to application execution.

Only manual reconstruction is possible for original files; shadow files can be reconstructed either manually or automatically.

If reconstruction is manual, the LEASY-RECONST utility routine - which is called automatically - terminates once the reconstruction is completed.

If reconstruction is automatic, the LEASY-RECONST utility routine is parallel to the applications and is activated whenever a complete AIM generation is due to be updated on the shadow files. The utility routine terminates along with the application.

The RECONST area of the **LEACMST* job variable is updated when original files are reconstructed and when shadow files are reconstructed. In this way, the program can trigger reaction to the end of manual reconstruction while a LEASY application is executing.

Evaluating this information is practical only in the case of manual reconstruction. With automatic reconstruction the information is updated cyclically until the end of the session.

Example of LEACMST

The LEASY-MAINTASK utility routine is to be started as a batch task in a procedure file by means of an *ENTER* command. Processing is delayed until LEASY-MAINTASK is running normally (see the table on page 112; entry 1 has the value "NORMAL"). An application program is then started.

Starting the procedure for LEASY-MAINTASK and the application program:

```
/BEGIN-PROC LOG=*NO,PAR=*YES(PROC-PAR=(&PROG),ESC-CHAR=C'&')
/CREATE-JV JV.PROG  ---------------------------------------------------- (01)
/SET-JV-LINK LINK=LEACMST,JV=JV.PROG ----------------------------------- (02)
/MODIFY-JV (JV.PROG,1,50),SET-VAL=                                        -
/           C'.........0.........0.........0.........0.........0' ----- (03)
/
/ENTER-JOB E.MTSTART --------------------------------------------------- (04)
/WAIT-EVENT UNTIL=JV(COND=((JV.PROG,1,6)=C'NORMAL')),TIME-LIM=600,      -
/                     TIMEOUT=ENDE) ------------------------------------ (05)
/START-EXE &PROG ------------------------------------------------------- (06)
/.ENDE END-PROCEDURE
```

Batch task in the E.MTSTART file:

```
/SET-LOGON-PAR
/SET-JV-LINK LINK=LEACMST,JV=JV.PROG ----------------------------------- (07)
/START-LEASY-MAINTASK -------------------------------------------------- (08)
*CAT=LCAT
*LOG=A
*END
/EXIT-JOB
```

Explanation:

(01) Catalog the user job variable with the name *JV.PROG*

(02) Link the user JV with the link name *LEACMST*

(03) Preset *JV.PROG*

(04) Start the batch task from the *E.MTSTART* file

(05) Wait a maximum of 600 seconds for main task; terminate if unsuccessful

(06) Start the application program

(07) Link the user JV with the link name *LEACMST*

(08) Start LEASY-MAINTASK

*LEAIOST: job variable specifying the number of active I/O tasks

The introduction of a job variable informing the user of the number of active I/O tasks enables programs and procedures to be controlled in accordance with the number of active I/O tasks. This also permits several I/O tasks to be activated under user control.

User action

The user must take the following measures:

1. Catalog a job variable using the *CREATE-JV* command

   ```
   /CREATE-JV jvname
   ```

   The job variable name *jvname* can be selected by the user.

2. Assign the link name *LEAIOST* to the job variable using the *SET-JV-LINK* command

   ```
   /SET-JV-LINK LINK=LEAIOST,JV=jvname
   ```

   Following these measures, the user can retrieve the information passed to the job variable by LEASY via the *SHOW-JV* command or the *GETJV* macro.

   The user must ensure that the same job variable is assigned/queried in all I/O tasks. LEASY does not check whether a job variable has been assigned.

Return information from LEASY

LEASY passes the following 10-byte information in the job variable:

Bytes	1-10
Content	nnn-ACTIVE *where nnn* is the number of active I/O tasks

8.3 Addressing mode

Unless it is running under openUTM, LEASY switches to 31-bit addressing mode each time it is called. Before returning to the application program, it switches back to the original addressing mode.

In openUTM mode LEASY does not switch the addressing mode.

8.4 LEASY as a subsystem

The LEASY runtime system LEACONX can be loaded into class 4 memory.

The DSSM declarations required for this purpose can be found in the supplied *SYSSSC* file *SYSSSC.LEASY.062*.

8.5 Coexistence of different LEASY versions

As of LEASY V6.0 there is in general full coexistence capability for the utility routines and the runtime system, i.e. several different versions of LEASY can be installed simultaneously with IMON and several different versions of LEASY can also be used simultaneously.
For technical reasons, **no** coexistence is possible with LEASY V5.3 or any earlier LEASY versions.

General requirements

Coexistence of different LEASY versions is possible as of BS2000/OSD V3.0. If several different versions of LEASY are to be installed simultaneously as subsystems, this is possible as of DSSM V3.5 and as of SSCM V2.0.

If you wish to use the coexistence of different LEASY versions, you must ensure conformance with the following points:

1. Only the message and SDF file of the latest version installed may be merged in. This condition is satisfied automatically if the last version installed is also the latest version.
2. The different LEASY versions must work with different catalogs, i.e. the catalogs must differ in at least one name component (user ID, catalog ID or catalog name). This also applies for several concurrently running LEASY systems with the same version.
3. No version mix is possible with reference to processing **one** LEASY catalog, i.e. an application program must work with the same LEASY version as the utility routines.

Version selection when starting a LEASY utility routine

It is possible to specify a 4 to 7 character version with both the *SELECT-PRODUCT-VERSION* and the *START-LEASY-utility* commands, i.e. it is also possible to specify correction versions.

A version specified in the *START-LEASY-utility* command always has priority over a version specified in a possibly prior *SELECT-PRODUCT-VERSION* command. If no version is specified in a *START-LEASY-utility* command, the latest installed version is used, if no *SELECT-PRODUCT-VERSION* command was issued before the *START-LEASY-utility* command. Otherwise, the *START-LEASY-utility* command takes over the version specified in the *SELECT-PRODUCT-VERSION* command.

Version selection when starting a LEASY user program

If you installed several LEASY versions with IMON, you can use the *SELECT-PRODUCT-VERSION* command to select the LEASY version with which your user program is to work. The *SELECT-PRODUCT-VERSION* command must be issued before starting the user program, otherwise the latest installed LEASY version is used.

9 Overview of the LEASY program interface

This chapter describes the LEASY program interface independently of the programming language.

The LEASY interface for COBOL is described in chapter "COBOL interface" on page 191ff.

The Assembler macro calls for LEASY are described in chapter "Assembler interface" on page 217ff.

9.1 Linking LEASY

To be able to call LEASY from a user program, you have to link a LEASY link module into the user program which then dynamically loads additional modules from the LEASY runtime system.

The LEASY link module you have to link in depends on the one hand on whether the user program is a batch/TIAM, DCAM or IO task user program and on the other hand whether the parameters are passed to LEASY as with previous LEASY versions (address of parameter list in register 1, last parameter identified by setting the most significant bit in the corresponding address) or according to the ILCS conventions (address of parameter list in register 1, number of parameters in register 0).

The following table shows an overview of all available LEASY link modules and the libraries in which they can be found:

Operating mode	**Type of parameter passing**	**Name of LEASY link module** *leasy_vb_modul*	**Library containing the LEASY link module** *bibliothek*
batch/TIAM	as previously	LEASY	SYSLNK.LEASY.062
batch/TIAM	ILCS convention	LEASYI	SYSLNK.LEASY.062
DCAM	as previously	LEADCAM	SYSLNK.LEASY.062.DCAM
DCAM	ILCS convention	LEADCAMI	SYSLNK.LEASY.062.DCAM
IO task	as previously	LEASY	SYSLNK.LEASY.062.IOH
IO task	ILCS convention	LEASYI	SYSLNK.LEASY.062.IOH
openUTM	If the user wishes to pass parameters according to the ILCS conventions, he must specify the openUTM start parameter *.LEASY ILCS*. If this entry is missing, parameter passing as previously is expected.		

Table 6: Overview of LEASY link modules

You will also find information on parameter passing according to the ILCS convention in section “Calling LEASY” on page 121.

The LEASY link module can be linked in either statically or dynamically. Static linking is made with the BINDER program using the following statement:

```
//INCLUDE-MODULES LIBRARY=bibliothek,ELEM=leasy_vb_modul,TYPE=R
```

The LEASY link module is linked in dynamically with the BIND macro, e.g. as follows:

```
BIND …,SYMBOL=leasy_vb_modul,SYMTYP=MODULE,LIBNAM=bibliothek,…
```

It is thereby particularly important that the *SYMTYP=MODULE* parameter is specified when dynamically loading the LEASY link module *LEASY*. Since there are also other modules in the libraries that have a *LEASY* **Entry**, if the *SYMTYP=MODULE* parameter is not specified (i.e. the default *SYMTYP=ANY* is effective) the search strategy of the BIND macro can lead to another module with a LEASY entry being loaded instead of the LEASY link module and this will generally cause an error.

9.2 Calling LEASY

The LEASY interface, with a few restrictions and extensions, corresponds to KLDS (compatible interface to linear databases).
In KLDS the LEASY interface is accessed via a **subroutine call** (CALL), e.g. in the case of COBOL in the form

```
CALL "LEASY" USING op1, op2, ...
```

The names of the operands *op1*, *op2* ... are freely selectable. However, as is customary with subroutine calls, the sequence is not arbitrary, since the operands are **positional operands**.

Parameter passing according to the ILCS conventions:

Users who wish to pass parameters to LEASY according to the ILCS conventions must note the following:

With a COBOL85 or COBOL2000 compiler COBOL user programs are compiled automatically such that register 0 contains the number of parameters.

With ASSEMBLER user programs, the user himself must ensure that parameters are passed correctly in the program. The LEASY *LEA@*... macros cannot be used.

The program must be relinked after compilation. You must note the information in section "Linking LEASY" on page 119 for this.

In general, LEASY applications that pass the parameters to LEASY as previously and those that pass parameters according to the ILCS conventions can run concurrently.

9.3 Loading the LEASY interface

The operands are identified in the LEASY call by means of their position. Their names are therefore freely selectable, and their contents can be modified at the time of execution. The correct sequence in the LEASY call is mandatory.

The individual LEASY operations require differing numbers of operands (see the table “LEASY OPEN modes” on page 184). Up to 9 operands may be defined. If the operands which are not used are positioned between relevant operands, they must be specified. Moreover their contents may be erased by means of blanks. The only exception is the *US* parameter. This is always the last operand in the operand list regardless of how many operands are specified before it. The field *U-PROT* (=*Y*) in the *RE* area determines that the last operand is to be interpreted as the *US* operand. If the operands not used are positioned at the end of the operand list, they may be omitted.

Table 7 shows all the possible operands and their positions in the LEASY call.
The operands have been provided with standard designations for the sake of clarity. These names, some of which are based on German abbreviations, are used throughout the entire manual.

Position	**Name**	**Meaning**	**Type**
1	OP	Operation code	U
2	RE	Reference area	U/R
	DB	File allocation	U
3	CI	Currency information	U/R
	CAT	Catalog information	U
4	AR	Input/output area	U/R
5	FA	Field selection	U
6	SI	Secondary index	U
7	KB	Key begin	U
8	KE	Key end	U
last	US	User area	U

Table 7: Summary of operands

Key

U	Information passed to LEASY by the user program
R	Information returned from LEASY to the user program

The operands listed under "3" (*DB/CI/CAT*) are alternatives.

Operation code OP

The operation code designates a 4-byte alphanumeric transfer field which determines the operation to be executed by LEASY.

OP Operation

1 4 bytes

table 8 lists the permissible LEASY operations.

LEASY operation	**Meaning**
CATD	Call LEASY catalog
OPFL	Open files
CLFL	Close files
OPTR	Open or extend transaction
CLTR	End transaction
MARK	Create checkpoint
BACK	Execute rollback
RDIR	Directly read record
RNXT	Read next record
RPRI	Read previous record
RHLD	Directly read and lock record
RNHD	Read and lock next record
RPHD	Read and lock previous record
SETL	Position file pointer
INSR	Insert new record
STOR	Insert record
REWR	Rewrite record
DLET	Delete record
LOCK	Lock record
UNLK	Unlock record
CINF	Transfer currency information

Table 8: LEASY operations

The LEASY operations are described in alphabetical order, starting on page 147.

Reference area RE

Via the reference area the user both sends information to and receives information from LEASY.

LEASY returns:

- the return code
- the SAM retrieval address in 24-bit or 31-bit format (see the “Introductory Guide to DMS” manual)
- the PAM block number for sequential reading or access via secondary keys
- the operation code and file name of the last call
- the transaction identifier for DCAM applications with the *OPTR* operation.

The *user* can specify:

- the OPEN mode
- the USAGE mode
- control of the BIM save method in *OPTR* (start of transaction)
- the SAM retrieval address in 24-bit or 31-bit format for positioning with *SETL* and direct reading with *RDIR* / *RHLD*
- the PAM block number
- the version identifier (mandatory entry)
- supplementary information for the *OPTR*, *CLTR*, *RDIR*, *RHLD*, *RNHD*, *RPHD*, *CINF* and *LOCK* operations
- the waiting time for locked records
- the identifiers for DCAM applications
- the transaction identifier for DCAM applications for each operation within a transaction

Unused fields must be filled with blanks (*X’40’*) or binary zeros (*X’00’*).

Structure of the reference area RE

The reference area *RE* is 80 bytes in length; it comprises a compatible part (48 bytes) and the LEASY extension (32 bytes). Table 9 shows the structure of the reference area.

Field names (graduated)	**Position (bytes)**	**Length**	**Type**		**Meaning**	
RC-CC	1-3	3	A	R	Compatible return code	Compatible part of reference area to KLDS
RC-KZ	4	1	A	R	System identifier "L"	
RC-LC	5-8	4	A	R	LEASY return code	
PASS	9-16	8	A	-	Reserved for password	
OPE OPE-STX OPE-OM OPE-LOG -----	17-24 17 18 19 20-24	8 1 1 1 5	A A A A -	U U U U -	Operation extensions STXIT mode OPEN/USAGE mode BIM logging control Reserved	
INT SAMPTR PAMHPNR ----- SAMPTR	25-32 25-28 25-28 29-32 25-32	8 4 4 4 8	A A B B	U/R U/R U/R - U/R	Internal key aspect SAM retrieval address (24-bit) or PAM block number Reserved SAM retrieval address (32-bit)	
NUM	33-40	8	N	R	Number of primary records	
IDE	41-48	8	A	U/R	Identification field for DCAM application	
REOP	49-52	4	A	R	Last operation code	LEASY extension of RE
REDB	53-68	16	A	R	Last file name (+ SI name)	
L-OPT	69	1	A	U	Version identifier "1"	
OPE1 OPE2	70 71	1 1	A A	U U	Operation extensions for OPTR/CLTR/RDIR/RHLD/RNHD /RPHD/LOCK/CINF	
OPE-WTIME	72-74	3	N		Waiting time for locks	
RC-LCE	75-79	5	A	R	LEASY return code extension	
U-PROT	80	1	A	U	User information	

Table 9: Structure of the reference area RE

A	Alphanumeric field
B	Numeric field (binary)
N	Numeric field (printable)
U	Information supplied by the user program to LEASY
R	Information returned by LEASY to the user program

Transfer and return in individual fields

The following table shows the transfer and return information in the individual fields of the reference area *RE*

Field	**Type**	**Contents**
RC-CC	R	Compatible return code from KLDS.
RC-KZ	R	LEASY identifier "L".
RC-LC	R	Error code internally generated by LEASY. This error code is more detailed than the compatible return code. The 4 bytes of RC-LC may be given the following format:
		A ddd B ddd C ddd D ddd J ddd S ddd T ddd DMS-error in processing of AIM file BIM file catalog file primary file job variable (JV) secondary index file status file
		ddd For three-digit DMS message numbers, these are the rightmost 3 bytes of the DMS error code, which has the format 0ddd (see the "System Messages, Volume 1 and Volume 2" manuals) For four-digit DMS message numbers (first digit not 0), these are the string "DMS". The *RC-LCE* field then contains the 4-digit DMS message number. (see the "System Messages, Volume 1 and Volume 2" manuals)
		L eee LEASY-internal error code. An additional error code can be provided in the *RC-LCE* field as supplementary information.
		The compatible return codes together with the return information generated by LEASY are listed in detail with their meanings in the chapter “Return codes” on page 395ff.
PASS		Reserved
OPE-STX	U	Entries in the OPE-STX field are ignored as of LEASY V6.1, the STXIT routine in LEASY remains activated in any case.

Table 10: Transfer and return in the fields of the RE area (part 1 of 8)

<table>
<tr><th>Field</th><th>Type</th><th>Contents</th></tr>
<tr><td>OPE-OM</td><td>U</td><td>
An identifier indicating the method of opening files or file identifiers can be specified in the OPE-OM field for the OPFL and OPTR operations.

OPE-OM= X'FF' means for **both operations** that the DB4 format is selected in the 3rd operand of the LEASY call for the file allocation and that the associated OPEN mode is specified in the DB operand for each file.

n the *OPFL* operation the 1-byte OPEN mode can be specified in the *OPE-OM* field; this mode is then valid in the same way for all those files allocated with *DB1/DB2* format.
In the case of the OPTR operation it is possible to specify not only X' FF' in this field, but also a 1-byte long processing mode, which is then valid in the same way for all file identifiers that are allocated with DB1 or DB2 format. This processing mode is mapped to a particular LEASY USAGE mode according to the table below:
<table>
<tr><th>Processing mode</th><th>USAGE mode</th></tr>
<tr><td>_ (default)</td><td>EXLD (SAM)/
UPDT (ISAM/PAM/DAM)</td></tr>
<tr><td>(ISAM/PAM/DAM)</td><td></td></tr>
<tr><td>A</td><td>RETR</td></tr>
<tr><td>E</td><td>PRUP</td></tr>
<tr><td>G</td><td>EXRT</td></tr>
<tr><td>L</td><td>EXLD</td></tr>
<tr><td>I</td><td>PRRT</td></tr>
<tr><td>O</td><td>EXLD</td></tr>
<tr><td>Q</td><td>EXLD</td></tr>
<tr><td>X</td><td>EXRT</td></tr>
<tr><td>B</td><td>EXUP</td></tr>
<tr><td>R</td><td>ULRT</td></tr>
<tr><td>U</td><td>ULUP</td></tr>
</table>
The specification of a processing mode has the same effect as the specification of the assigned USAGE mode for all declared file identifiers by means of the DB4 format.
</td></tr>
</table>

Table 10: Transfer and return in the fields of the RE area (part 2 of 8)

Field	**Type**	**Contents**
OPE-LOG	U	In the 1st *OPTR* operation of a transaction the BIM save method for this transaction can be canceled by specifying "N". The field is space-filled (X' 40') as standard, i.e. BIM saving is activated for the current transaction if the appropriate operand values are assigned in the LEASY-MAINTASK and LEASY-CATALOG utility routines. If openUTM and LEASY are linked, BIM saving may only be deactivated for read transactions.
SAMPTR	U/R	In the case of SAM files, the current retrieval address is returned in the *SAMPTR* field after each operation. This is specified in the format (24-bit or 31-bit) predefined with the *SETL* or *RDIR* operations (*IDIRPTR*='bbbbbbrr' or *IDIRPTR*='bbbbbbbbrrrrrrrr'). With the *SETL* or *RDIR* operation, the user must store such a retrieval address in the *SAMPTR* field either in 24 bit format ('bbbbbbrr') or in 31 bit format ('bbbbbbbbrrrrrrrr'). If the 24 bit format is used, the second word of the *SAMPTR* field must then be filled with zeros or blanks. This allows correct positioning within the file for a subsequent sequential read operation. With *RNXT/RNHD*, a switchover is made from 24 bit mode to 31 bit mode if the number of the record being read in the block exceeds 255. The 31 bit mode remains activated until it is reset back to 24 bit mode possibly by either *SETL* or *RDIR*.
PAMHPNR	U/R	The PAM block number must be stored in this field in PAM write operations and for direct reading; in sequential read operations and read operations via secondary keys this is done by LEASY.
NUM	R	LEASY supplies the number of primary records belonging to a secondary index value in the *NUM* field for *RDIR/RHLD* operations. This is only possible if the identifier "*N*" is specified in the *OPE2* field, and if no range has been specified for access via a secondary index.
IDE	U/R	No entry is made to this field unless LEASY is called by a DCAM application. The DCAM application name must be supplied in the *IDE* field for the *CATD* operation. *IDE* must be erased prior to the 1st *OPTR* operation of each transaction. LEASY will then return the transaction identifier with the *OPTR* operation. This identifier must be supplied for all LEASY operations affecting this transaction. A *CLTR* operation causes the *IDE* field to be erased.

Table 10: Transfer and return in the fields of the RE area (part 3 of 8)

<table>
<tr><th>Field</th><th>Type</th><th colspan="4">Contents</th></tr>
<tr><td>REOP/
REDB</td><td>R</td><td colspan="4">LEASY always enters the operation code and the file name (+ SI name) of the last call in the <i>REOP</i> and <i>REBD</i> fields. If an error occurs during the <i>OPFL</i> (open files) or the <i>OPTR</i> (open transaction) operations, the file causing the error (together with its OPEN or USAGE mode) is stored in the <i>REDB</i> field. In the <i>CATD</i> operation (call LEASY catalog) the first 16 bytes of the specified catalog name are stored in the <i>REDB</i> field. This allows the user to employ a common error routine when handling errors.</td></tr>
<tr><td>L-OPT</td><td>U</td><td colspan="4">LEASY interface identifier. This field must always be set to "1".</td></tr>
<tr><td rowspan="9">OPE1/OPE2</td><td rowspan="9">U</td><td colspan="4">Additional functions can be specified in the OPE1 and OPE2 fields for the following operations:</td></tr>
<tr><td rowspan="2">OP=OPTR:</td><td>OPE1=</td><td>'_'</td><td>normal transaction start (<i>DB</i> specification)</td></tr>
<tr><td>OPE1=</td><td>'W'</td><td>transaction start and simultaneous file positioning (<i>CI</i> specification in 3rd operand)</td></tr>
<tr><td rowspan="4">OP=CLTR:</td><td>OPE1=</td><td>'_'</td><td>normal end of transaction</td></tr>
<tr><td>OPE1=</td><td>'R'</td><td>resetting of transaction</td></tr>
<tr><td>OPE2=</td><td>'_'</td><td>transaction termination with cancellation of all file access requests</td></tr>
<tr><td>OPE2=</td><td>'T’</td><td>transaction termination and simultaneous transaction start (restart point with release of record locks but retention of resources and file positions)</td></tr>
<tr><td colspan="4">OP=RDIR/RHLD:</td></tr>
<tr><td></td><td>OPE2=</td><td>'N'</td><td>LEASY must transfer the number of primary records to a secondary index value (in the <i>NUM</i> field).</td></tr>
</table>

Table 10: Transfer and return in the fields of the RE area (part 4 of 8)

Field	Type	Contents			
OPE1/OPE2 (continued)	U	OP=RHLD/RNHD/RPHD/LOCK:			
			OPE1=	’S’	READ-LOCK enforced on locking
			OPE1=	’_’	WRITE-LOCK enforced on locking
		OP=RNHD/RPHD:			
			OPE2=	L	If the required record is free, it is transferred to the *AR* area and locked. The pointer is positioned after or before the record that has been read, depending on the direction in which it was read. If the record is locked, LEASY sets the pointer in the same way as if it had been read.
			OPE2=	_	If the required record is free, it is transferred to the *AR* area and locked. The pointer is positioned after or before the record that has been read, depending on the direction in which it was read. If the record is locked, the return code (*99ALL006*) is transferred after the waiting time has elapsed. The record is not read and the pointer is not modified.
		OP=CINF:	OPE1=	'_'	Currency information on all the file identifiers open in the transaction and also their current file pointers. This currency information enables a transaction to be opened (with simultaneous file positioning) by means of *OPTR* and *OPE1=' W'* .
			OPE1=	'F'	Currency information on the files contained in the LEASY catalog and their secondary indices.

Table 10: Transfer and return in the fields of the RE area (part 5 of 8)

Field	Type	Contents			
OPE1/OPE2 (continued)	U		OPE2=	’_’ ’C’	Currency information (type 1) on all the files in the LEASY catalog.
			OPE2=	'O'	Currency information (type 1) on all the files opened by means of *OPFL*.
			OPE2=	'T'	Currency information (type 1) on all the files involved in the transaction.
			OPE2=	'S'	Currency information (type 2) on the file specified in CI.
			OPE2=	'W'	The help function immediately preceding this field is to be continued.
			– The OPE2 entry is only practical if *OPE1='F'* is also specified. – Type 1 currency information only includes general information on the file. Type 2 currency information lists all the tables for the specified file which are for use within LEASY.		
OPE1	U	OP=UNLK	OPE1=	’_’	Normal record release
			OPE1=	’U’	In transactions without BIM saving, modified records are also released
OPE-WTIME	U	A waiting time in seconds for locked records or logical files can be specified individually for each operation in the *OPE-WTIME* field. If the field is not occupied *(X'40'* or *X'00'*), the global waiting time for the session applies (**TIME* operand in the LEASY-MAINTASK utility routine); the default value is 0 if there is no LEASY catalog. Even if an *OPTR* operation encounters a USAGE mode incompatibility with a parallel transaction for a file identifier of the file list specified, the specified or the global waiting time comes into force. If this waiting time expires without the locking transaction having been completed, the user program receives the return code *99ALL110*; otherwise it can continue within its *OPTR* operation.			

Table 10: Transfer and return in the fields of the RE area (part 6 of 8)

Field	**Type**	**Contents**
RC-LCE	R	The 5 bytes of *RC-LCE* can have the following format: 1. 4-character message code for a DMS error in one of the following forms: Axxxx DMS error while processing an AIM file Bxxxx DMS error while processing a BIM file Cxxxx DMS error while processing a catalog file Dxxxx DMS error while processing a primary file Jxxxx DMS error while processing a job variable Sxxxx DMS error while processing a secondary index file Txxxx DMS error while processing a status file xxxx 4-digit DMS message number (see the "System Messages, Volume 1 and Volume 2" manual) 2. Error code extension for the internal LEASY error code stored in the *RC-LC* field in the following form: L_eee eee LEASY-internal error code 3. NKISAM macro error code for the NKISAM macro error stored in the *RC-LC* field, in the form Iiiii iiii Main return code of NKISAM macro (see the "DMS Macros"). 4. Other macro code for the macro error stored in the *RC-LC* field, in the form Mbbaa bbaa corresponds to the return code of the relevant macro in R15 (R15=X' bb0000aa')
U-PROT	U	When user information is specified, the value ' Y' must be set in this field in the case of the operations *BACK, CATD, CLFL, CLTR, DLET, INSR, OPFL, OPTR, REWR* and *STOR*. In the case of the other operations, this field is not evaluated. U-PROT= ' ' No user information specified.

Table 10: Transfer and return in the fields of the RE area (part 7 of 8)

Field	Type	Contents		
		U-PROT=	'Y'	User information specified. The last operand in the operand list is interpreted as user information.

Table 10: Transfer and return in the fields of the RE area (part 8 of 8)

File allocation DB

This operand is used to allocate the files to be processed. The allocation is also known as the "file list".

A distinction is made between the terms:

- **file** (*file*)
- **file identifier** (*file identifier*)

A file identifier consists of the logical name of the file, which can be up to 8 positions long, and an identification code for a sequence identifier (*fm*), which can be up to 3 positions long (optional). The file name and sequence identifier are separated from one another by a slash (/).

```
file-identifier : file[/fm]
```

As many different sequence identifiers as are required may be defined for a logical file. The concept "file identifier" enables several different file positions in a logical file to be current at the same time. It is thus possible, for example, to restart a read command (*RDIR*) at various positions in a file, and to continue independent sequential reading (e.g. *RNXT*) at these positions. The formation of independent read sequences (via different secondary keys) that are identified by the sequence identifier is advantageous for many operations.

LEASY, however, manages lock elements per file and not per file identifier.

The names of logical files (*file*) are allocated using the *DB* operand for the *OPFL* operation only. The names of file identifiers (*file-identifier*) must be specified for all other LEASY operations.

Various formats can be used to specify files/file identifiers depending on their number and use.

Format DB1

This format is used when only **one** file is to be processed. The OPEN or USAGE mode (see page 184f) is taken from the *OPE-OM* field of the reference area (not *X'FF'*).

Format for OPFL

```
file
```

file	logical file name (max. 8 characters)

Format for OPTR and all read and write operations

```
file[/fm]
```

file/fm	file identifier	
	file	logical file name (max. 8 characters)
	fm	sequence identifier (max. 3 characters)

Example of DB1 formats

```
for OPFL          FILE
for OPTR          FILE/ABC
```

Format DB2

This format permits a **variable** number of logical files or file identifiers to be specified. The shared OPEN or USAGE mode (see section “Opening files and transactions” on page 184f) is taken from the *OPE-OM* field of the reference area (not *X’FF’*).

Format for OPFL

```
(file1,file2,...)
```

file	logical file names (max. 8 characters)

Format for OPTR

```
(file1[/fm1],file2[/fm2],...)
```

file/fm	file identifiers	
	file	logical file names (max. 8 characters)
	fm	sequence identifier (max. 3 characters)

i Blanks must not be entered in the parenthesized expression.

Examples of DB2 formats

```
for OPFL         (FILE1,FILE2,FILE3)
for OPTR         (FILE1/ABC,FILE2,FILE3/XYZ)
```

Format DB3

This format may only be used for *CLFL* and *UNLK* operations. *ALL* addresses **all** allocated files.

$$\left\{\begin{array}{l}\text{(ALL)}\\ \text{ALL}\end{array}\right\}$$

If *ALL* is specified without parentheses, the field must be 12 bytes in length.

Format DB4

This format permits a **separate** OPEN or USAGE mode to be defined for each addressed file or file identifier (see page 184f). The *OPE-OM* field of the reference area must be set with *X'FF'*. This is the identifier for specifying the DB4 format.

Formats for OPFL

for *one* file	for *several* files
(file,mode)	((file1,mode1),(file2,mode2)...)

file	logical file name (max. 8 characters)
mode	OPEN mode (1 character)

Formats for OPTR

for *one* file identifier	for *several* file identifiers
(file[/fm],mode)	((file1[/fm1],mode1),(file2[/fm2],mode2)...)

file/fm	file identifier	
	file	logical file name (max. 8 characters)
	fm	sequence identifiers (max. 3 characters)
mode	USAGE mode (4 characters)	

Blanks must not be entered in the parenthesized expression.

Examples of DB4 formats

```
for OPFL        (FILE,4)
                ((FILE1,4),(FILE2,1),(FILE3,2))
for OPTR        (FILE/FM,RETR)
                ((FILE/FM1,RETR),(FILE/FM2,UPDT),(FILE,EXUP))
```

Currency information CI

The currency information contains the following items of information:

- A list of the file identifiers opened in the current transaction
- A list of the current file pointers
  - secondary and primary keys for ISAM, PAM and DAM
  - DMS-internal file position pointers (retrieval address `ID1RPTR`) for SAM
- Range limits

The operation code extensions *OPE1=F* and *OPE2=C,O,T,S* can be used to request currency information on the following:

- all the files contained in the LEASY catalog
- all the files opened with the aid of *OPFL*
- all the files involved in the current transaction
- a particular file which is to be specified

CI is used for the following LEASY operations:

- With *CINF* the currency information is requested.
- With *OPTR* with *OPE1=W* (transaction start and simultaneous file positioning) the currency information is transferred. The CI must be made available in the form in which it was received for the associated *CINF* operation.

Format of the currency information CI

The CI takes the form of a variable-length record with a 4-byte length field at the beginning.

CI	ci-slf	ci-slr	ci-inf	
	1	3	5	n+4 bytes

Field name	Position (bytes)	Length	Type	Meaning
ci-slf	1 - 2	2	U/R	Length field; contains the value n+4
ci-slr	3 - 4	2	R	Length field; contains the necessary minimum length of CI
ci-inf	5 to n+4	n	R	Information field with length n

For the operation *CINF* with *OPE1=_* users must supply the length field *ci-slf* with the estimated length of the information field *ci-sl* prior to the call:
ci-slf=4+m.
In reply they receive the actual length in *ci-slf* and the currency information in *ci-inf*. If no transaction is open, *ci-slf=O* is supplied.

For the operation *CINF* with *OPE1=F* users must supply the length field *ci-slf* with the estimated length of their information field (*ci-slf=4+m*) prior to the call.

For the operation *CINF* with *OPE1=F* and *OPE2=S* users must also store the 8-character logical file name of the desired file in *ci-inf* prior to the call.

In reply, users receive the actual length of the transferred file information in *ci-slf* and the currency information in *ci-inf*. The scope of the list is determined by the operation code extension *OPE2=C*, *O*, *T* or *S* in the *RE* reference area.

If no file fulfills the requirements, *ci-slf=0* is supplied.

If the length specified for *ci-slf* by the user is too short, LEASY issues error code *04XLP11* . If the length of *ci-inf* is sufficient to accept currency information from at least one file, part of the file information is stored in *ci-inf*, and its length is supplied in *ci-slf*. The field *ci-slr* then supplies the necessary minimum length for all the currency information. Thereafter the *CINF* call can be repeated with a larger response area; or the preceding *CINF* call can be continued by means of *OPE2=W*, in which case the next part of the file information is provided in *ci-inf*. Users should note that the values of *ci-slf* and the contents of *ci-inf* in the original length must not be changed if the preceding *CINF* call is continued.

If the area required for transferring the file information is larger than 64K (i.e. more than 819 files), the length cannot be transferred in either *ci-slf* or *ci-slr*. In this case, only one part of the file information is supplied and the *ci-slr* field is filled with *X'FF'*.

The following overview shows the various return values:

	Is ci-inf large enough?					
	yes		no			
	File available?		len > 64K?			
	yes	no	no		yes	
			Space for at least one file?		Space for at least one file?	
			yes	no	yes	no
Error message	-	-	+	+	+	+
ci-slf	len	0	partlen	0	partlen	0
ci-slr	-	-	len	len	X'FFFF'	X'FFFF'
ci-inf	inf	-	partinf	-	partinf	-

where

len	length of all the file information
inf	all the file information
partlen	length of the transferred information section
partinf	part of the file information

Calculating the length of ci-slf

The length of *ci-slf* is calculated as follows:

For OPE1=_

$$ci\text{-}slf = 4 + n*16 + n_1*5 + \sum_{i=1}^{n_2}(KEYLEN_i + 1) + n_3*8 + \sum_{i=1}^{n_4}2*KEYLENINT_i$$

n	number of file identifiers. n = n1 + n2
n_1	number of file identifiers of SAM files
n_2	number of file identifiers of ISAM, PAM and DAM files
$n_3 \leq n_1$	number of file identifiers with current range limits
$n_4 \leq n_2$	(KB, KE)
$KEYLEN_i$	max (*KEYLEN-PRIMFILE,KEYLEN-SIFILE*) of the (i)th file identifier
$KEYLENINT_i$	*KEYLEN-PRIMFILE* or *KEYLEN-SIFILE* of the (i)th file identifier for which the range limits apply.

KEYLEN-PRIMFILE=3 is mandatory for PAM files

KEYLEN-PRIMFILE=4 is mandatory for DAM files

KEYLEN-PRIMFILE=4 or *8* is mandatory for SAM files

For OPE1=F and OPE2=C, O, T or _

$$ci\text{-}slf = 4 + n*88 + v$$

n	number of files
v	16 or 0 space for internal LEASY administrative information if only a section of the file information is to be retrieved and additional sections are to be requested with the aid of *CINF* and *OPE2=W*. The value *v=16* should be used in this case.

For OPE1=F and OPE2=S

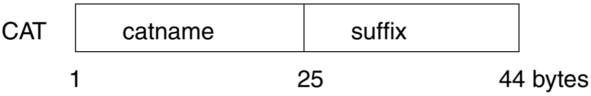
$$\text{ci-slf} = 4 + 111 + s*22 + \sum_{j=1}^{s}(st_j*5 + \sum_{k=1}^{r_j}(rid_{jk}+1))$$

rounded up to a multiple of 4

s	number of secondary index definitions in the file
st_j	number of code sections of secondary index definition j
r_j	number of record type definitions of secondary index definition j
rid_{jk}	length of record type definition k in secondary index definition j

Catalog information CAT

This operand must be specified in the *CATD* operation.



Field name	Position (bytes)	Length	Type	Meaning
catname	1 - 24	24	U	Name of LEASY catalog
suffix	25 - 44	20	U	Suffix for model files

The name of the LEASY catalog *[:catid:][$userid.]file-catalog* must be specified in *catname*. If *LINK=linkname* is specified for *catname*, the

```
/ADD-FILE-LINK LINK=linkname,F-NAME=[:catid:][$userid.]catalogname.LEASYCAT
```

command can be used for allocation purposes so that different file catalogs can be processed without changing the application program. There must not be any blanks in the *LINK=linkname* string.

The user ID may be omitted if the LEASY catalog is cataloged under the same user ID as that under which the calling program is executed.

The suffix is required by LEASY in *OPFL/OPTR* for selecting the correct instances in a model file group (*LEASYTYPE=M*).

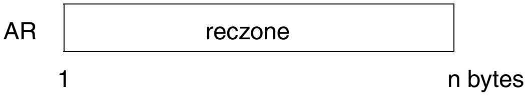
Note the following restrictions:

- *catname* and *suffix* must be padded with blanks to the right.
- Support of MPVS
  If a LEASY application wants to be connected to the CMMAIN of a LEASY catalog which is not generated on the public volume set of the user ID under which the application program is started, a catalog identifier (:*catid*:) must be specified in *catname* for the public volume set containing the LEASY catalog.
- Implementation in multiprocessor systems
  If the LEASY catalog is on a foreign processor, the catalog identifier (:*catid*:) of this processor must be specified as part of the catalog name:

```
[:catid:][$userid.]file-catalog
```

Input/output area AR

The operand *AR* refers to a transfer or return area. This area has a variable length.

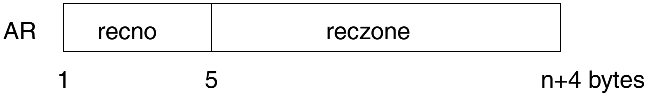


The *AR* operand must be used in read/write operations to make available an I/O area with the length of the record. The record is always transferred in its entire length for write operations and can be restricted to key fields for read operations (see Field selection *FA*); these fields are supplied at their correct positions in the *AR* area.

The I/O area of a file is also known as the **record zone** *AR*.

Where the record format is variable the record length field is sent in the record zone for read operations; it must be supplied by the user for write operations.

In DAM files the *AR* area has the following format:



Field name	Position (bytes)	Length	Type	Meaning
recno	1 - 4	4	U/R	Relative record number (binary)
reczone	5 to n+4	n	U/R	Record zone with length n

The relative record number is not part of the record, though it is supplied in the *AR* area (bytes 1-4) by the user or by LEASY.

Field selection FA

The *FA* operand designates an alphanumeric transfer field. Its contents determine whether the entire record or only key values are to be returned to the record zone *AR*.
This operand must be specified if the operand *SI* (6th operand) follows in the LEASY call, in order to ensure compatibility with KLDS.

FA	string
1	8 bytes

The following character strings can be specified for *string*:

(ALL)	Specifies that the entire record is returned to the record zone *AR* (5-byte character string).
ALL	Specifies that the entire record is returned to the record zone *AR* (8-byte character string, space-filled).
MAINITEM	Specifies that with all read operations for ISAM, PAM and DAM files only the key contents (primary key and the current LEASY secondary key) are returned. There is no reading of the primary record. Thus when reading sequentially via a secondary key, for example, direct accessing of the primary key is no longer required.

Secondary index SI

The operand *SI* designates an 8-byte alphanumeric transfer field.

SI	secondary-index-name	
	1	8 bytes

This field can be used in *RDIR*, *RHLD* and *SETL* operations to specify the name of a secondary index to be used for accessing. This name must previously have been defined for the appropriate file by means of the LEASY-CATALOG utility routine (see the **FIL* statement, *KEY* operand) or as an ISAM secondary index (*CREATE-ALTERNATE-INDEX* command or *CREAIX* macro).

If the name of the secondary index consists of space characters or the character string *MAINITEM*, the **primary key** is used for accessing.

Key begin KB and key end KE

Operands *KB* and *KE* designate transfer fields containing key values. The length of the fields is dependent on the type of key values they are to contain.

KB	key-value		KE	key-value	
	1	m bytes		1	m bytes

m key length

For the *RDIR/RHLD* and *SETL* operations, a key range, within which sequential reading can take place, can be defined for the index specified by the operand *SI* using the *KB* and *KE* operands (primary and/or secondary key).

The format specified for the SAM retrieval address (4 or 8 byte format) in the *SAMPTR* field in the *RE* area determines the entry in *KB* and *KE*. In other words, if an 8 byte address is specified in the *RE* area, 8 byte retrieval addresses are also expected in *KB* and *KE*. The behavior is the same with 4 byte addresses.

For the *LOCK* and *UNLK* operations, a key range, which is to be locked or unlocked, can be defined using the *KB* and *KE* operands.

The contents of *KB* can be greater than, less than or equal to those of *KE*. The differing effects are explained by the individual operations.

The *RNXT/RNHD* or *RPRI/RPHD* operations enable reading in ascending or descending key sequence within the range defined here.

When a range limit is reached, return code `010LL003` is supplied.

When accessing via the primary index (*SI=MAINITEM* or *SI=space*), primary key values with the length of the primary key must be specified in *KB* and *KE*. In the case of PAM files, the PAM block numbers (4 bytes) must be transferred. In the case of SAM files, the retrieval addresses (*ID1RPTR*) must be transferred in 24-bit format (4 bytes) or in 31-bit format (8 bytes). In the case of DAM files, the relative record numbers (4 bytes) must be transferred (similar to the *PAMHPNR/SAMPTR* field in the *RE* area).

When accessing via a secondary index, the secondary key values with the length of the current secondary index must be specified (*SI* operand).

If this logical secondary index is defined as a combination of several secondary key parts (see the **FIL* statement for LEASY-CATALOG), callers must combine the individual parts themselves to form the complete index value.

With the *SETL* operation the secondary index value must be transferred first, followed by the primary index value, when transferring the key values.

Special function when specifying KB without KE

No range is defined in this case (exactly 7 operands specified in the LEASY call) with only a single key value being transferred. The effect obtained is identical to that obtainable if the primary or secondary key were transferred via the record zone *AR* (using 4, 5 or 6 operands).

User area US

This operand defines a USER area for the USER information. This information is transferred to the AIM file and can be logged by the utility routine LEASY-RECONST.

US	us-sl	filler	us-inf	
	1	3	5	n+4

Field name	Position (bytes)	Length	Type	Meaning
us-sl	1 - 2	2	U	Length of USER information (=n+4)
filler	3 - 4	2	U	Filler characters
us-inf	5 to n+4	n	U	USER information of variable length

If a USER area is specified, the reference area *RE* must contain *U-PROT=Y* (see the “U-PROT” line in the table “The following table shows the transfer and return information in the individual fields of the reference area RE” on page 132).

9.4 LEASY operations

LEASY operations can be divided into 4 groups:

Control operations		CATD, OPFL, CLFL
Transaction operations		OPTR, CLTR, MARK, BACK
Operations on file level	read operations	RDIR, RNXT, RPRI
	read operations with lock effect	RDIR, RNXT, RPRI, RHLD, RNHD, RPHD
	positioning operation	SETL
	write operations	INSR, STOR, REWR, DLET
	lock operations	LOCK, UNLK
Operation for currency information retrieval		CINF

Table 11: The LEASY operations

These operations are described below in alphabetical order.

BACK Execute rollback

Operands in the LEASY call:

```
OP,RE[,US]
```

Function

The *BACK* operation rolls back the current transaction; all file updates are canceled using the BIM file.

All record locks managed by LEASY are released and a restart point is set. At the same time a continuation transaction is started in which all those file identifiers that were open at the time of the *BACK* operation are reopened. However, all the file identifier positions point to the start of the file.

This operation is allowed only in timesharing mode (batch and interactive), and not in inquiry and transaction processing with openUTM.

BACK has the same effect as *CLTR* with *OPE1=R* and *OPE2=T*, except that the file positions are not retained.

CATD Call LEASY catalog

Operands in the LEASY call:

```
OP,RE,CAT[,US]
```

Function

The *CATD* operation assigns the LEASY file catalog created with the LEASY-CATALOG utility routine. The files of this LEASY catalog are accessed during the subsequent *OPFL* (open files) and *OPTR* (open transaction) operations.

A suffix may be specified for any model files, i.e. files with *LEASYTYPE=M*. This suffix is then valid for all model files accessed.

The *CATD* operation can only be executed if no files or transactions have yet been opened for the task.
When linked to openUTM this operation is not called by the user, but by openUTM in its start phase.

When accessing a LEASY file catalog which is cataloged under a different user ID than that of the program, the name *$userid.file-catalog* must be specified.

A blank as the first character of the catalog name means that no file catalog is to be used. All files are therefore treated implicitly as foreign files.

If the *CATD* operation is not executed at all, the effect is the same as for a *CATD* operation with *catalog-name=blank*.

If a LEASY catalog has already been allocated with the aid of the *CATD* operand, and an additional *CATD* operation is executed with a blank as the first character of the catalog name, LEASY is detached from the catalog and the original state (processing without LEASY catalog) is reinstated. With this operation, the LEASY-STXIT routine in particular is deactivated again.

i If a user program repeatedly logs on with *CATD catalog* and off again with *CATD*, it must be noted that only a limited number of STXIT administration blocks can be created for a program system (a maximum of 100 with BS2000/OSD-BC V4.0).

The task is linked in the system to the corresponding common memory CMMAIN. The common memory CMMAIN is the common storage for all tasks that are linked to a LEASY file catalog.

If the common memory CMMAIN to be accessed is present in the system but is still being initialized by the LEASY-MAINTASK utility routine, a waiting period of up to 5 minutes comes into force. Each time one second of this period elapses, a check is made to see if initialization has been completed.

If CMMAIN goes into the "ready for application programs" state during this waiting period, the program can be executed. If the entire waiting time elapses without this state being attained, the application program receives one of the return codes *99ALLS01* or *99ALLS04*, depending on the state of CMMAIN.

ℹ If LEASY is called by a DCAM application, the DCAM application name must also be transferred in the *IDE* field of the reference area *RE*.
The application name must not be made up of blanks (*X'40'*) or *X'00'*. All other EBCDIC codes are permissible. The field contents of *IDE* are erased (*X'00'*) if the operation is executed successfully.

CINF Transfer currency information

Operands in the LEASY call:

```
OP,RE,CI
```

Function

The *CINF* operation is used to request currency information, i.e. a list of all file identifiers opened in the transaction and their current file pointers (the keywords of records or blocks accessed or used for positioning in the previous read operation). For ISAM, PAM and DAM files this position is represented by primary and secondary keys; for SAM files the DMS retrieval address *ID1RPTR* is used.

The operation code extensions *OPE1=F* and *OPE2=C,O,T,S* can be used to request currency information on:

- all the files in the LEASY catalog
- all the files opened with the aid of *OPFL*
- all the files involved in the current transaction
- a particular specified file.

The information comprises internal LEASY tables which for the most part contain specifications from the LEASY catalog relating to the files and secondary indices. The currency information is stored as a variable-length record in the *CI* area, which the user must make available in the required length. If there is no transaction open (*OPE1=_*) or if no file fulfills the requirements (*OPE1=F*), *ci-slf=0* is returned.
The currency information (*OPE1=_*) may be used to open a transaction and simultaneously position the file (see the *OPTR* operation, additional function *OPE1=W* with *CI* specified as the 3rd operand).

Additional function (entries in the RE area)

OPE1=_	Currency information on all the file identifiers opened in the transaction and also on their current file pointers. This currency information enables a transaction to be opened (with simultaneous file positioning) by means of *OPTR* and the additional function *OPE1=W*.
OPE1=F	Currency information on the files contained in the LEASY catalog and their secondary indices.
OPE2 = $\begin{Bmatrix} \_ \\ \text{C} \end{Bmatrix}$	Currency information (type 1) on all the files in the LEASY catalog and their secondary indices.
OPE2=O	Currency information (type 1) on all the files opened with the aid of OPFL.
OPE2=T	Currency information (type 1) on all the files involved in the transaction.
OPE2=S	Currency information (type 2) on the file specified in *CI*.
OPE2=W	The help function immediately preceding this is to be continued.

Table 12: Currency Information: specifications in the RE area

Specification of *OPE2* is practical only if *OPE1=F* is also specified.

Type 1 currency information comprises only general information on the file.

Type 2 currency information lists all the internal LEASY tables for the specified file.

CLFL Close files

Operands in the LEASY call:

$$\text{OP,RE[,}\left\{\begin{array}{l}\text{DB1}\\ \text{DB2}\\ \text{DB3}\end{array}\right\}\text{][,US]}$$

Function

The *CLFL* operation closes files specified in the file list. It can close:

- all files (format DB3 or DB operand omitted) or
- selected files (format DB1 or DB2)

opened in previous *OPFL* operations. This operation is not permitted if a **transaction** is open for the task.

The *DB3* operand can be specified to provide compatibility with the KLDS interface. The operation then has the same effect as if only two operands were specified (close all files).

Example

```
OPFL      (D1,D2,D3,D4)
    OPTR  (D1,D2,D3,D4)
    .
    .
    .
    CLTR
CLFL      D2
    OPTR  (D1,D3,D4)
    .
    .
    .
    CLTR
CLFL      (ALL)
```

i This operation is not permissible in openUTM operation.

CLTR Close transaction

Operands in the LEASY call:

```
OP,RE[,US]
```

Function

The *CLTR* operation is used to close the current transaction and to set a restart point, i.e. the corresponding BIM file is defined as being "empty".

Additional functions (entries in the RE area)

OPE1=R	The transaction is reset: all file updates are canceled by means of the BIM file. If this additional function is used when the BIM data saving function is not activated for this transaction (*LOG* parameter of the LEASY-MAINTASK), return code *99ALL014* is output, but the transaction is terminated normally.
OPE2=T	The transaction is closed and a continuation transaction is opened. A restart point is set for the closed transaction and record and block locks are released. However, all file identifiers remain open for the continuation transaction with the same USAGE modes; the file positions are retained. If the BIM data saving function was deactivated for the original transaction (*OPE-LOG=N*), it will also be deactivated for the continuation transaction.

If LEASY is called by a DCAM application, the transaction identifier must be transferred via the *IDE* field. The contents of the *IDE* field are erased (*X'00'*) if the operation is executed successfully, providing *OPE2=T* has not been set.
All record and block locks are canceled.

If an implicit *OPFL* was performed when opening the transaction, i.e. *OPFL* was not used, the assigned files and the BIM file are also physically closed (except for *OPE2=T*). This is known as an implicit *CLFL*.

In timesharing mode all *CLTR* variants are permitted.

When linked to openUTM the suffix *OPE2=T* is not permitted, because it is not compatible with the transaction concept of openUTM.

LEASY cannot cancel DMS locks when using *CLTR*. (This is relevant only for ISAM, DAM and PAM files that are opened as foreign files with SHARED-UPDATE.)

DLET Delete record

Operands in the LEASY call:

```
OP,RE,DB1[,AR[,FA,SI,KB]][,US]
```

Function

The *DLET* operation deletes a record from an ISAM or DAM file, or a block from a PAM file. If updating of the secondary index references is specified, the entries in the secondary index file (SI file) are also deleted. The position pointer for the file identifier kept internally by LEASY remains unchanged.

Transferring the key value

Table 13 below shows the various methods of transferring the key values as a function of the file type and the number of operands in the LEASY call.

File type	No. of operands	Supplied		Deleted
ISAM, DAM	7	DB1 KB	with file name with primary key	Record with the primary key from *KB*
	4	DB1 AR	with file name with the primary key at the defined position for ISAM; in the first 4 bytes for DAM	Record with the primary key from *AR*
	3	DB1 AR	with file name not supplied	Last record read successfully via the same file identifier
PAM	7	DB1 KB	with file name with PAM block numbers	Block with the PAM block number from *KB*
	3	DB1 PAMHPNR (*RE* area)	with file name with PAM block number	Block with the PAM block number from *PAMHPNR*. If *PAMHPNR=0*: last block read successfully via the same file identifier

Table 13: Transfer of key values for the DLET operation

If a file is governed by a lock log, the record or block must have been locked within the transaction (not necessarily via the same file identifier). Deleted records or blocks automatically remain locked until the end of the transaction.

If a **PAM** file is opened as a foreign file with SHARED-UPDATE, the blocks must have been locked beforehand by means of *RHLD/RNHD/RPHD/LOCK*.

INSR Insert new record

Operands in the LEASY call:

```
OP,RE,DB1,AR[,US]
```

Function

A new record or block is entered in the file specified, and all secondary key values required are created in the SI file if this file is to be updated immediately (see the LEASY-CATALOG utility routine, *KEY* operand of the **FIL* statement).
The record or block must be made available in the record zone *AR*.

With a SAM file, the record is appended to the end of the file, and the retrieval address is returned in the *SAMPTR* field of the reference area *RE* in 24-bit format in the case of records with a record number ≤ 255 or in 31-bit format in the case of records with a record number > 255.

With an ISAM, DAM or PAM file the record or block is inserted in accordance with its primary key. No record or block with the specified primary key may already exist. With a PAM file the primary key must be stored in the *PAMHPNR* field of the reference area *RE*.

Records or blocks inserted by *INSR* are automatically locked exclusively until the end of the transaction.

The position pointer for the file identifier kept internally by LEASY is not altered.

LOCK Set record lock

Operands in the LEASY call:

```
OP,RE,DB1[,AR[,FA,SI,KB[,KE]]]
```

Function

The *LOCK* operation enforces lock elements on:

- individual records or blocks identified by means of a primary key
- file sections identified by means of a primary key range

in ISAM, PAM and DAM files.

Since the *LOCK* operation does not access the file, the existence of the record, block or file section is not verified. This means that it is possible to lock non-existent records, blocks or file sections (phantom locks).

Lock elements can only be enforced on files governed by a lock protocol (see also section “File locking” on page 19).

The *LOCK* operation permits transaction-oriented coordination tasks to be implemented using only memory management (i.e. without accessing files) with the aid of the lock elements of a (possibly empty) file.

Transferring the key value

Table 14 below shows the various methods of transferring the primary key values as a function of the file type and the number of operands in the LEASY call.

File type	**No. of operands**	**Supplied:**		**Locked:**
ISAM, DAM	8	DB1	with file name	File section delimited by the primary keys *KB/KE*
		SI	with blanks or "MAINITEM"	
		KB, KE	Primary keys of range limits. The contents of *KB* may be greater than, less than or equal to those of *KE*	
	7	DB1	with file name	Record with the primary key from *KB*
		SI	with blanks or "MAINITEM"	
		KB	with primary key	
	4	DB1	with file name	Record with the primary key from the *AR* area
		AR	with the primary key at the defined position for ISAM; in the first 4 bytes for DAM	
PAM	8	DB1	with file name	File section delimited by the PAM block numbers *KB/KE*
		SI	with blanks or "MAINITEM"	
		KB, KE	Primary keys of range limits. The contents of *KB* may be greater than, less than or equal to those of *KE*	
	7	DB1	with file name	Block with the PAM block number from *KB*
		SI	with blanks or "MAINITEM"	
		KB	PAM block number	
	3	DB1	with file name	Block with the PAM block number from *PAMHPNR*
		PAMHPNR (*RE* area) with PAM block number		

Table 14: Transfer of key values for the LOCK operation

Additional functions (entries in the RE area)

OPE1=S	A READ-LOCK is executed for the file on locking
OPE1=_	A WRITE-LOCK is executed for the file on locking

It is possible to lock a primary key range only if no incompatibility with ranges or key values of other transactions will result for the entire range.

If the file is opened as a foreign file with *SHARED-UPDATE=YES*, the *LOCK* operation is only effective for PAM and DAM files, and not for ISAM files. The lock job is then mapped to the UPAM locking mechanism.

MARK Create checkpoint

Operands in the LEASY call:

```
OP,RE[,US]
```

Function

By means of the *MARK* operation the current transaction is closed and a restart point is set, i.e. the corresponding BIM file is defined as being "empty".

All record locks and those block locks managed by LEASY are canceled.

A continuation transaction is started at the same time, in which all file identifiers open at the time of the *MARK* operation are reopened.

However, the file identifier positions all point to the start of the file.

This operation is permissible only in timesharing mode (batch and interactive) and in DCAM inquiry and transaction processing, and not in inquiry and transaction processing with openUTM.

MARK has the same effect as *CLTR* with *OPE2=T*, except that the file positions are not retained.

OPFL Open files

Operands in the LEASY call:

$$\text{OP,RE,}\left\{\begin{array}{l}\text{DB1}\\ \text{DB2}\\ \text{DB4}\end{array}\right\}\text{[,US]}$$

Function

OPFL physically opens the files specified in the file list according to the relevant OPEN mode (DMS OPEN macro). The relevant OPEN mode is either:

- taken from the *OPE-OM* field in the reference area *RE*;
  it is then identical for all files with the file allocation *DB* (DB1/DB2 formats), or
- specified explicitly for each file (DB4 format).

The associated OPEN modes are described on page 184ff.

The *OPFL* operation is not permissible if **transactions** are open for that task.

The files to be opened can either be specified in a **single** *OPFL* operation or subdivided among **several** consecutive *OPFL* operations. Up to 512 files can be opened (theoretical upper limit).

Example

```
OPFL     (D1,D2) ──────────────────────────── Files D1 and D2 opened
    OPTR (D1,D2)
     .
     .
    CLTR
OPFL     D3 ───────────────────────────────── File D3 opened
    OPTR (D1,D2,D3)
     .
     .
    CLTR
CLFL     (ALL) ────────────────────────────── All files closed
```

OPFL is only mandatory for openUTM and DCAM operation; otherwise it is optional. It is called by openUTM during the start phase (*OPFL* start operand).
In DCAM operation, the files to be opened must be specified in a **single** *OPFL* operation in multitasking operation. With openUTM operation, the files can be spread over several *OPFL* start parameters.

Files larger than 32 GB are not processed by LEASY. An OPFL to such a file is rejected.

OPTR Open or extend transaction

The *OPTR* operation has two different functions with different operands in the LEASY call:

Function 1: Defining the start of a transaction or extending a transaction-oriented file list

Operands in the LEASY call:

```
       ⎧DB1 ⎫
OP,RE,⎨ DB2 ⎬[,US]
       ⎩DB4 ⎭
```

Function

If no user transaction is active at the time of the call, the beginning of a LEASY transaction is defined by *OPTR*. All file identifiers specified in the *DB* operand of the LEASY call are logically opened for the transaction together with their associated USAGE modes.

If the *OPFL* operation has not previously been executed, the files to be processed and the BIM file are also physically opened.
The internally selected DMS OPEN mode can be ascertained from the table “Defined USAGE modes for the OPTR operation” on page 185. This is known as an implicit *OPFL*. If the *OPFL* operation has been executed (i.e. the files are already physically open), a transaction-oriented entry of the file identifiers is only effected in the tables.

All files are positioned to the start or end of the file (SAM files by means of reverse reading) and the primary key.

If before-image saving techniques are used, the files involved in the transaction are simultaneously logged in the first element of the BIM file (see the section “BIM save method” on page 51f).

If at the time of the *OPTR* operation a transaction for this user is already open, this transaction is expanded to include those file identifiers specified in the *DB* operand. The newly opened file identifiers are positioned to the start or end of the file (SAM files by means of reverse reading) and the primary key. This application is only possible in cases where the files have been physically opened by the *OPFL* operation.

Example

```
   OPFL ((D1,4),(D2,4),(D3,3))
⎡--OPTR ((D1,RETR),(D2,UPDT))
|   RNXT D1
|   OPTR (D3,EXUP) ----------- Extension of the transaction-
|                              oriented file list
|   RDIR D3
⎣--CLTR
```

The following applies in both cases:

A file identifier can be opened only once within a transaction.

A file can be logically opened repeatedly within a transaction that has various sequence identifiers, i.e. different file identifiers. In this case, the USAGE mode specified for file identifier 2 is checked first according to the table “Rules for combining the USAGE modes of a logical file” on page 189 for compatibility with those USAGE modes hitherto specified for the same file in the same transaction. The resulting new USAGE mode is used to examine compatibility with parallel foreign transactions (according to the table “Possible combinations of LEASY USAGE modes” on page 188).

Example

```
OPTR   ((D1/FM1,RETR),(D1/FM2,UPDT))
RNXT   D1/FM1
RDIR   D1/FM2
```

If an *OPTR* operation for a file identifier of the specified file list encounters a USAGE mode incompatibility with a parallel transaction, the waiting time set either by the **TIME* operation of the LEASY-MAINTASK utility routine or the *OPE-WTIME* operand in the *RE* area comes into force.

If this waiting time elapses without the locking transaction being ended, the application program receives return code *99ALL110*; otherwise it can continue within its *OPTR* operation.

Function 2: Opening a transaction, and opening and positioning file identifiers in accordance with CI

Operands in the LEASY call:

```
OP,RE(with OPE1=W),CI[,US]
```

Function

A LEASY transaction is opened and the file identifiers stored in the currency information are opened and positioned.

Differences from function 1:

- The names and the USAGE modes of the file identifiers to be opened are stored in the currency information and do not have to be specified in the *DB* operand of the LEASY call.
- After being opened (physically and/or logically), the file identifiers are not positioned to the start of the file but to the positions stored in the currency information.

Defining a restart point using the OPTR operation with the additional function OPE1=W

The statement sequence

`CINF`	save currency information, and
`CLTR (OPE2=T)`	close transaction with simultaneous transaction restart

can be used to define a restart point for the file status while retaining file positions. This restart point can also be used for restarts after a system breakdown.

A restart point is implemented as follows:

1. The file contents are reset during a system warm start by LEASY-MAINTASK.
2. Saved currency information is read in the subsequent program run.
3. Implementation of the *OPTR* (*OPE1=W*) operation with the currency information.

The application program then has the same file status (i.e. open files and file positions) as at the time of *CLTR* (*OPE2=T*). The status of the storage areas in the application program cannot be restored by LEASY.

This approach is *not* required for a normal restart with *OPTR* and the file list (files are positioned at beginning of file).

If an *OPTR* operation for a file identifier of the specified file list encounters a USAGE mode incompatibility with a parallel transaction, the waiting time set either by the **TIME* statement of the LEASY-MAINTASK utility routine or the *OP-WTIME* operand in the *RE* area comes into force.

If this waiting time elapses without the locking transaction being ended, the application program receives return code *99ALL110*; otherwise it can continue within its *OPTR* operation.

i If LEASY is called by a DCAM application, the *IDE* field in the reference area *RE* must be transferred following deletion (*X’00’*) in the first *OPTR* operation of each LEASY transaction. LEASY returns the transaction identifier in the *IDE* field if the operation is executed successfully. The identifier must then be transferred in all operations of this transaction.

RDIR/RHLD

Directly read record / Directly read and lock record

Operands in the LEASY call:

```
OP,RE,DB1,AR[,FA[,SI[,KB[,KE]]]]
```

Functions of the RDIR operation

The *RDIR* function reads a record or block directly into the record zone *AR*:

- via a specified key (records of an ISAM or DAM file)
- via the specified retrieval address in 24-bit or 31-bit format (records of a SAM file)
- via the specified block number (blocks of a PAM file).

If the *SI* operand is missing, is empty (blanks) or contains *MAINITEM*, accessing is performed via the primary index.

Access is performed via the secondary index if it is specified in the *SI* operand (only possible with ISAM, PAM and DAM files). If an ISAM secondary key is used for accessing NK-ISAM files the name of the ISAM secondary key must be specified in the *SI* operand.

If 8 operands are specified - definition of a range (*KB*, *KE*) - the record having the lowest key value of the range is used if $KB < KE$; if $KB > KE$, the record with the highest value is used.

Where a record having a specific key is to be read, this is achieved by:

- $KB = KE$
- specifying 7 operands and supplying *KB*
- specifying 4 to 6 operands and supplying the key in the *AR* area (ISAM and DAM) or in the *RE* area (PAM and SAM).

Additional functions

In addition to reading of the record, the pointer is positioned within the file identifier to the located key and the index used (primary or secondary index).

If 8 operands are specified, a key range is defined within which sequential reading can take place using the *RNXT/RPRI* operations. *RDIR* then supplies the first record available in the range. This need not necessarily be the key value specified for *KB*. If no record is found when a range is specified with the *RDIR* operation, no range is current.

The start/end of the file or the start/end of the secondary index constitute the natural limits of a range where less than 8 operands are specified. If no record is found here using *RDIR*, positioning is effected in the same way as with a corresponding *SETL* call (see the *SETL* operation).

If *MAINITEM* is specified in the *FA* operand, only the key fields are returned when reading. This means that when accessing via the primary index only the existence of the data record is verified and the primary key field is occupied. When accessing via a secondary index only the primary and secondary key values are supplied; there is no direct accessing of the primary data record.

Additional function (entries in the RE area)

OPE2=N:

When accessing via the secondary index, LEASY supplies the number of primary keys for the secondary index value in the *NUM* field of the reference area *RE*, providing no key range has been specified.

Definition of a read range

The ISAM key values, DAM block numbers, PAM block numbers or SAM retrieval addresses, or the secondary key values for ISAM, PAM and DAM must be specified as primary keys in the *KB* and *KE* operands.

The following applies for SAM, PAM, DAM and ISAM files:

The range limits (start/end of file) can be specified by the key values *X'00'* and *X'FF'*.

KB </>/= *KE* can be selected for ISAM, DAM and PAM files. SAM files require that *KB* ≤ *KE* when opening for reading forwards and *KB* ≥ *KE* when opening for reverse reading.

Transfer of key values

Table 15 shows the various methods of transferring the key values as a function of the file type and the number of operands.

File type	**No. of operands**	**Supplied:**	**Returned by LEASY:**
ISAM PAM, DAM, SAM	8	DB1 with file name FA with ALL MAINITEM SI with Secondary index Blanks MAINITEM KB Range limit keys KE (primary or secondary key) SAMPTR (RE area) with SAM: retrieval address 4 byte address: 2nd word binary zero or blanks 8 byte address: 2nd word ≠ binary zero or blanks	FA: ALL If *KB<KE* Record with the lowest key value in the range in the *AR* area If *KB> KE* Record with the highest key value in the range in the *AR* area If *KB=KE* Record with the key value KB in the *AR* area FA: MAINITEM If *KB<KE* Lowest key in the range in the *AR* area or *RE* area (PAM, SAM) If *KB> KE* Highest key in the range in the *AR* area or *RE* area (PAM, SAM) If *KB=KE* Key from *KB* in the *AR* area or *RE* area (PAM, SAM)

Table 15: Transfer of key values for the RDIR/RHLD operation (part 1 of 3)

File type	**No. of operands**	**Supplied:**	**Returned by LEASY:**
ISAM PAM, DAM, SAM	7	DB1 with file name FA with $\left\{\begin{array}{l}\text{ALL}\\\text{MAINITEM}\end{array}\right\}$ SI with $\left\{\begin{array}{l}\text{Secondary index}\\\text{Blanks}\\\text{MAINITEM}\end{array}\right\}$ KB with key (primary or secondary key) SAMPTR (RE area) with SAM: retrieval address 4 byte address: 2nd word binary zero or blanks 8 byte address: 2nd word ≠ binary zero or blanks	FA: ALL Record with the key from *KB* in the *AR* area FA: MAINITEM Key from *KB* in the *AR* area or *RE* area (PAM, SAM)
	6	DB1 with file name AR with primary key: with ISAM at the defined position; with DAM in the first 4 bytes (binary) PAMHPNR (RE area) with PAM: FA with $\left\{\begin{array}{l}\text{ALL}\\\text{MAINITEM}\end{array}\right\}$ SI with $\left\{\begin{array}{l}\text{sec. index}\\\text{MAINITEM}\end{array}\right\}$ with SAM: only entry MAINITEM or blanks allowed SAMPTR (RE area) with SAM: retrieval address	FA: ALL With ISAM and DAM: record with the primary key specified in the *AR* area in the AR area With PAM: block with the primary key specified in *PAMHPNR* in the *AR* area FA: MAINITEM With ISAM and DAM: primary key in the *AR* area With PAM: primary key from *PAMHPNR* in the *RE* area With SAM: record with the retrieval address specified in *SAMPTR*

Table 15: Transfer of key values for the RDIR/RHLD operation (part 2 of 3)

File type	**No. of operands**	**Supplied:**	**Returned by LEASY:**
ISAM PAM, DAM, SAM	5	DB1 with file name AR with primary key: with ISAM at the defined position; with DAM in the first 4 bytes (binary) PAMHPNR (RE area) with PAM: primary key FA with {ALL, MAINITEM} SAMPTR (RE area) with SAM: retrieval address	FA: ALL With ISAM and DAM: record with the primary key specified in the *AR* area in the *AR* area With PAM: block with the primary key specified in *PAMHPNR* in the *AR* area FA: MAINITEM With ISAM and DAM: primary key in the *AR* area With PAM: primary key from *PAMHPNR* in the *RE* area With SAM: record with the retrieval address specified in *SAMPTR*
	4	DB1 with file name AR with primary key: with ISAM at the defined position; with DAM in the first 4 bytes (binary) PAMHPNR (RE area) with PAM: primary key SAMPTR (RE area) with SAM: retrieval address	With ISAM and DAM: record with the primary key specified in the *AR* area in the *AR* area With PAM: block with the primary key specified in *PAMHPNR* in the *AR* area With SAM: record with the recovery address in *SAMPTR* With SAM: record with the retrieval address specified in *SAMPTR*

Table 15: Transfer of key values for the RDIR/RHLD operation (part 3 of 3)

Read access via LEASY secondary index

When accessing via a secondary index, first the appropriate primary key value is ascertained and transferred to the record zone *AR* (for ISAM and DAM files) or to the *PAMHPNR* field of the *RE* area (for PAM files).

If there are several duplicate records for a secondary index value, the lowest primary key value is used where $KB \leq KE$, and the highest one is used where $KB > KE$.

If the primary record cannot be read successfully (e.g. lock is not possible), the caller is still able to interpret the primary key value.

If a multiple secondary index is used, the first occurrence (least distance from the start of the record) is interpreted for positioning in each case.

Read access via ISAM secondary key

Access via the ISAM secondary key is described in section “Secondary indexing” on page 41ff.

Function of the RHLD operation

The *RHLD* operation also locks the record read, in addition to performing the functions of the *RDIR* operation, but only if the read operation was successful, i.e. error code *=000LL000*. If, for example, the record is not found, no lock element is enforced.

Additional functions (entries in the RE area)

The following additional functions, which are requested in the reference area *RE*, are also possible for the *RHLD* operation:

OPE1=S	A READ-LOCK is executed for the file on locking.
OPE1=_	A WRITE-LOCK is executed for the file on locking.

REWR Rewrite record

Operands in the LEASY call:

```
OP,RE,DB1,AR[,US]
```

Function

An existing record or block is updated. The position pointer maintained internally by LEASY for the file identifier is not changed.

If a file is governed by a lock log, the record or block must already have been locked within the transaction.

Updated records or blocks remain locked until the transaction is over.

If updating of the secondary index pointers is specified, the secondary index pointers are automatically maintained as well.

Transferring the key value

SAM file	The record to be updated must have been read beforehand. It can be updated immediately afterwards.
ISAM/DAM file	The record with the primary key stored in the record zone *AR* is updated.
PAM file	The block with the primary key stored in the *PAMHPNR* field of the reference area *RE* is updated.

i The following additional remarks apply if the file was opened as a foreign file with SHARED-UPDATE:

- In the case of an ISAM or DAM file, the record must have been locked and read by means of *RHLD/RNHD/RPHD* immediately preceding the *REWR* operation.
- In the case of a PAM file, the blocks must have been locked beforehand by means of *RHLD/RNHD/RPHD/LOCK*.

RNXT/RNHD

Read next record / Read and lock next record

Operands in the LEASY call:

```
OP,RE,DB1,AR[,FA]
```

Functions of the RNXT operation

The next record or block of the file identifiers specified is read sequentially, beginning at the current position for the file identifier, towards the end of the file (for SAM files) or in ascending order of the primary or secondary key values (for ISAM, PAM or DAM files).

Access is made to the index which was used for positioning during the last *RDIR/RHLD* or *SETL* operation performed for this file identifier (default value = primary key).

The range (*KB,KE*) also applies when specified for this operation (*RDIR/RHLD/SETL*).

If no range was specified in *RDIR/RHLD/SETL*, the return code *010LL003* (EOF) is supplied upon reaching the end of file or the secondary index end of file.

If a read **range** is specified, the return code *LC=L003* is supplied by LEASY when the range limit is exceeded.

An *RNXT* operation following *SETL* causes that record or block to be retrieved which has a key value equal to or greater than the primary or secondary key value specified in *SETL*.

Additional functions

With SAM files the value of the current retrieval address *ID1RPTR* is returned to the *SAMPTR* field in the *RE* area, in which case the first data record on the far left in the data block is given the number 01 within the block number. The retrieval address is supplied in 24-bit or 31-bit format (see the *SAMPTR* field on page 128).

With PAM files the block number of the block read is returned to the *PAMHPNR* field.

In addition to the reading of the record/block, the position within the file identifier is set to the new record if reading was successful (error code=*000LL000*).

Whenever a range limit is exceeded this is recorded in the position management for the file identifier.

By specifying *FA=MAINITEM* in the operand the user can request that only the key fields be supplied.
This means, when sequentially accessing via primary indices, that only the next primary key value is supplied, but otherwise the record zone is unchanged.
If accessing sequentially via a secondary index, the primary key field and the secondary key field (or, in divided secondary keys, all partial fields) in the record zone are occupied.

Reading

When accessing via a secondary index, first the appropriate primary key value is ascertained and stored in the *AR* record zone (ISAM) or in the *PAMHPNR* field of the *RE* area (PAM). If subsequent reading of the primary record is unsuccessful (e.g. it cannot be locked), the user can still analyze the primary key value.

If there are several duplicate records for a single secondary index value, the records or blocks are retrieved in ascending order of their primary key values, beginning with the lowest.

Functions of the RNHD operation

The *RNHD* operation causes the record or block to be locked after being read. It is only locked after a successful operation (error code=*000LL000*).

When accessing via the primary index, the appropriate record is first written directly into the *AR* record zone. Only then is the primary key value known and are locking attempts possible. Should an error occur, the record zone has already been changed.

Additional functions (entries in the RE area)

The following additional functions, which are requested via the reference area *RE*, are possible for the *RHND* operation:

OPE1=S	A READ-LOCK is issued for the file during locking.
OPE1=_	A WRITE-LOCK is issued for the file during locking.
OPE2=L	If the required record is free, it is transferred to the *AR* area and locked. The pointer is positioned after or before the record that has been read, depending on the direction in which it was read. If the record is locked, LEASY sets the pointer in the same way as if the record had been read.
OPE2=_	If the required record is free, it is transferred to the *AR* area and locked. The pointer is positioned after or before the record that has been read, depending on the direction in which it was read. If the record is locked, the return code (*99ALL006*) is transferred after the waiting time has elapsed. The record is not read and the pointer is not modified.

RPRI/RPHD

Read previous record / Read and lock previous record

Operands in the LEASY call:

```
OP,RE,DB1,AR[,FA]
```

Functions of the RPRI operation

The next record in the file identifier specified is read sequentially, beginning at the current position for the file identifier, towards the beginning of the file (SAM files) or in descending order of primary or secondary key values (ISAM, PAM or DAM files).

Accessing is effected via the index at which the pointer was positioned after the last *RDIR*, *RHLD* or *SETL* operation executed for this file identifier.

The range (*KB,KE*) also applies when specified for this operation (*RDIR/RHLD/SETL*).

If no range was specified in *RDIR/RHLD/SETL*, return code *010LL003* (EOF) is supplied upon reaching the start of file or the secondary index start of file.

If a range is current, return code *010LL003* is supplied by LEASY when the range limit is exceeded.

An *RPRI* operation following a *SETL* operation results in the retrieval of the record or block having a key value equal to or less than the secondary or primary key value (*KB*) specified for the *SETL* operation.

Additional function

In addition to reading the record or block, the RPRI operation also effects positioning to the new record or block for the current file identifier. The new positioning depends on a successful read operation, i.e. error code=*000LL000*; otherwise the file position is left unchanged.

The other additional functions and the execution of the *RPRI* operation are as for the *RNXT* operation.

Functions of the RPHD operation

In addition to the functions mentioned above, the *RPHD* operation locks the record or block, but only if the operation was successful (error code=*000LL000*).

When accessing via the primary index, the appropriate record is first written directly into the *AR* record zone. Only then is the primary key record known and are locking attempts possible. Should an error occur, this means that the record zone has already been changed.

Additional functions (entries in the RE area)

The following additional functions, which are requested in the reference area *RE*, are possible for the *RPHD* operation:

OPE1=S	A READ-LOCK is issued for the file during locking
OPE1=_	A WRITE-LOCK is issued for the file during locking
OPE2=L	If the required record is free, it is transferred to the *AR* area and locked. The pointer is positioned after or before the record that has been read, depending on the direction in which it was read. If the record is locked, LEASY sets the pointer in the same way as if the record had been read.
OPE2=_	If the required record is free, it is transferred to the *AR* area and locked. The pointer is positioned after or before the record that has been read, depending on the direction in which it was read. If the record is locked, the return code (*99ALL006*) is transferred after the waiting time has elapsed. The record is not read and the pointer is not modified.

SETL Position file pointer

Operands in the LEASY call:

```
OP,RE,DB1[,AR[,FA,SI[,KB[,KE]]]]
```

Function

SETL positions an internal file pointer to defined keys for the specified file identifier. In addition, the index specified in the *SI* operand (primary or secondary index) and, when 8 operands are specified, a key range for subsequent *RNXT/RNHD/RPRI/RPHD* operations are set.

In the case of a multiple secondary index, the first occurrence (the shortest distance to the start of the record) is always evaluated for positioning.

The following applies with respect to transferring LEASY keys:

- When 8 operands are specified, a range is defined by means of (*KB,KE*); otherwise the file limits constitute the natural range limits.

  The following applies to ISAM, PAM and DAM files:
  The file start and end range limits can be specified with the key values *X'00...'* or *X'FF...'* with the applicable key lengths.

  With SAM files, the file start and end range limits are defined with 4 and 8 byte addresses with the key values *X'00000000'* or *X'FFFFFFFF'*.

  With ISAM, PAM and DAM files, *KB </>/= KE* can be specified. When opening SAM files, you must select *KB ≤ KE* for forward reading and *KB ≥ KE* for reverse reading.

- When accessing via the **primary key**, the operands are supplied in the same way as for the *RDIR/RHLD* operation (see the table “Transfer of key values for the RDIR/RHLD operation” on page 168f).

- When accessing via a **LEASY secondary key**, the secondary key value **and** the primary key value must always be specified in *SETL* (in contrast to *RDIR/RHLD*). This enables positioning to a particular duplicate record. If there are several duplicates of the secondary key value specified in *SETL*, *RNXT* will retrieve the record/block having a primary key value greater than or equal to that specified in *SETL*, while *RPRI* will retrieve that having a primary key value less than or equal to that specified in *SETL*.

- The transfer of a key pair (combination of the LEASY primary and secondary key values) occurs, where 4 to 6 operands are specified, via the record zone *AR* and, in the case of a PAM file, additionally via the *PAMHPNR* field.

  Where 7 to 8 operands are employed, the key pair is transferred via *KB* (and *KE*) only. The key fields addressed via *KB* and *KE* each comprise 2 parts:

The secondary key value, with the length of the current secondary key index, is supplied to the part on the left. The primary key value (ISAM primary key or PAM block number or DAM record number 4 positions long) must then be added.
In this way it is also possible to limit all duplicate records that exist for one secondary key value to a subset when defining a read range (*KB,KE*).

- Access via the **ISAM secondary key** is described in section “Secondary indexing” on page 41ff.

SETL alone does not effect an I/O operation. A subsequent *RNXT/RNHD* or *RPRI/RPHD* operation is needed before a record or block with a key ≥ or ≤ the key specified in *SETL* is actually read.

STOR Insert record

Operands in the LEASY call:

```
OP,RE,DB1,AR[,US]
```

Function

The data record or block is written to the file, regardless of whether or not the record/block already exists (in contrast to the *REWR* and *INSR* operations, which check beforehand whether the record/block exists).

The position pointer maintained internally by LEASY for the file identifier is not altered.

Data records or blocks inserted by *STOR* remain locked until the transaction is closed.

Any secondary index pointers are also automatically maintained if updating is specified for the secondary index pointers.

UNLK Cancel record lock

Operands in the LEASY call:

```
        ⎧{DB1[,AR[,FA,SI,KB[,KE]]] ⎫
OP,RE[, ⎨                          ⎬
        ⎩DB3                       ⎭
```

Function

The *UNLK* operation cancels locking elements in ISAM, PAM and DAM files:

- for individual records or blocks identified by means of a primary key
- for file sections identified by means of a primary key range.

The following applies to locks that are maintained in the common memory CMMAIN by means of the LEASY lock log (the application operates with the LEASY file catalog):

Locks can only be canceled for records or blocks or file sections that were locked within the transaction by a *LOCK* operation or a read operation with locking function (*RHLD*, *RNHD*, *RPHD*), but were not updated.

Records inserted, modified or deleted in the transaction in which they were locked can be unlocked under the following circumstances:

- In the *UNLK* operation, the *OPE1='U'* operand was set in the *RE* area.
- BIM saving was suppressed for the file in question in the current transaction.

  BIM saving is influenced by the following specifications:

  - on a session-specific basis by the *LOG* operand of the main task
  - on a file-specific basis by the *BIM* operand of the catalog
  - on a transaction-specific basis by the *OPE-LOG* field of the *RE* area in the *OPTR* operation.

It is advisable to use *UNLK* explicitly to release locks no longer required. This reduces the waiting time of other transactions attempting to access the locked record/block.

It is, however, not possible to use *UNLK* to release merely a partial range or a single key value from a locking range specified by means of *LOCK*. The range limits specified for *UNLK* and *LOCK* must be identical.

All locks are automatically canceled at the end of the transaction.

If the *UNLK* operation is applied to a modified record which does not comply with the conditions listed above, return code *99AL008* is supplied.

Transfer of the key value

Table16 shows the various methods of transferring key values in accordance with the file type and the number of operands.

File type	**No. of operands**	**Supplied**		**Released**
ISAM, DAM	8	DB1 SI KB KE	with file name with blanks or “MAINITEM” Primary keys of range limits The contents of *KB* may be greater than, less than or equal to the contents of *KE*	File section delimited by the primary keys *KB/KE*
	7	DB1 SI KB	with file name with blanks or “MAINITEM” with primary key	Record with the primary key from *KB*
	4	DB1 AR	with file name with primary key: at the defined position for ISAM; in the first 4 bytes for DAM	Record with the primary key from *AR*
	3	DB1 AR	with file name not supplied	All locked (but unmodified) records in this file

Table 16: Transfer of key values for the UNLK operation (part 1 of 2)

File type	No. of operands	Supplied		Released
PAM	8	DB1 SI KB KE	with file name with blanks or “MAINITEM” primary key of range limits	File section delimited by the PAM block numbers *KB/KE*
	7	DB1 SI KB	with file name with blanks or “MAINITEM” PAM block number	Block with the PAM block number from *KB*
	3	DB1 PAMHPNR	with file name (*RE* area) with PAM block number	Block with the PAM block number from *PAMHPNR*
	3	DB1 PAMHPNR	with file name (*RE* area) with value 0	All locked (but unmodified) blocks in this file
ISAM, PAM, DAM	3	DB3	with “(ALL)”	All locked (but unmodified) records or blocks of all files involved in the transaction
	2	DB1	not supplied	

Table 16: Transfer of key values for the UNLK operation (part 2 of 2)

If a foreign file is opened with SHARED-UPDATE, LEASY maps record/block locks to the ISAM/UPAM lock mechanisms. In this case, only one record/ block at the most can be unlocked by one *UNLK* call.

With ISAM, the locked record of the file specified is unlocked. In the case of PAM files, the block number must be specified explicitly. With DAM files, a record number must be specified. Locks remaining at the end of the transaction cannot be released automatically by LEASY.

If a file is not governed by a lock log or the file is opened without SHARED-UPDATE, the *UNLK* operation is ineffective.

9.5 Table of all LEASY operations and their operands

Opera-tion	RE	OPE1	OPE2	INT	DB1/2/3/4/CI/CAT	AR	FA	SI	KB	KE	US
OPFL	x				DB1/DB2/DB4						[x]
OPTR	x	x			DB1/DB2/DB4/CI						[x]
CLFL	x				[DB1/DB2/DB3]						[x]
CLTR	x	x	x								[x]
MARK	x										[x]
BACK	x										[x]
RDIR	x		x	P+S	DB1	x	[x	[x	[x	[x]]]]	
RHLD	x	x	x	P+S	DB1	x	[x	[x	[x	[x]]]]	
SETL	x			P+S	DB1	[I+D	[x	x	[x	[x]]]]	
LOCK	x	x		P	DB1	[I+D	[I+P+D	I+P+D	I+P+D	[I+P+D]]]	
RNXT	x				DB1	x	[x]				
RNHD	x	x			DB1	x	[x]				
RPRI	x				DB1	x	[x]				
RPHD	x	x			DB1	x	[x]				
INSR	x			P	DB1	x					[x]
STOR	x			P	DB1	x					[x]
REWR	x			P	DB1	x					[x]
DLET	x			P	DB1	[I+D	[I+P+D	I+P+D	I+P+D]]		[x]
UNLK	x			P	[DB3]/[DB1	[I+D	[I+P+D	I+P+D	I+P+D	[I+P+D]]]]	
CINF	x	x	x		CI						
CATD	x				CAT						[x]

Table 17: LEASY operations and their operands

x	Operands must be specified
I	Operands must be specified for ISAM
P	Operands must be specified for PAM
D	Operands must be specified for DAM
S	Operands must be specified for SAM
[ ]	Operands are optional
/	One of the listed operands must be specified

LEASY verifies that the necessary operands are specified. Operands specified, but not required, are ignored.

9.6 Opening files and transactions

When opening files by means of an *OPFL* operation, the OPEN modes defined in Table 18 can be specified.

The OPEN mode selected determines the number of USAGE modes that can be used for that file in a subsequent *OPTR* operation.

LEASY OPEN mode for the OPFL operation

The OPEN mode is used to define the DMS OPEN mode for physical opening of the file specified.

The 1-byte numeric identifier for the LEASY OPEN mode must be stored in the *OPE-OM* field of the *RE* area (formats DB1 and DB2), or in the DB4 format of the file assignment.

LEASY OPEN mode	**S=SAM** **I=ISAM** **P=PAM** **D=DAM**	**DMS OPEN mode**	**USAGE modes permitted for OPTR**
1	I+P+D+S	INPUT	*PRRT, EXRT*
2	I+P+D	INPUT, SHARUPD	*RETR, PRRT, EXRT, ULRT*
3	I+P+D	INOUT	all ISAM, DAM and PAM USAGE modes except *ULRT* and *ULUP*
4	I+P+D	INOUT, SHARUPD	all ISAM, DAM and PAM USAGE modes
5	S	REVERSE	*PRRR, EXRR*
6		(reserved)	---
7	S	UPDATE	*EXUP*
8	S	OUTPUT	*EXLD*
9	S	EXTEND	*EXLD*
A	I+P+D	OUTIN	all ISAM, DAM and PAM USAGE modes except *ULRT* and *ULUP*
B	I+P+D	OUTIN, SHARUPD	all ISAM, DAM and PAM USAGE modes except *ULRT*

Table 18: LEASY OPEN modes

For formats DB1 and DB2, a blank can also be used to specify the OPEN mode. In ISAM, DAM and PAM files this stands for *OPEN mode=4*, and in SAM files for *OPEN mode=9*.

It should be noted that in openUTM LEASY programs *OPEN mode=3* is possible for openUTM applications with one task only.

LEASY USAGE mode for the OPTR operation

The USAGE mode defines the access mode for the user's own transactions and specifies which access modes are allowed in parallel transactions.

The 4-byte alphabetic USAGE mode is specified for the *OPTR* operation and must be stored in format DB4; otherwise the value in the *OPE-OM* field of the *RE* area applies.

Table 19 is valid for master and model files. Foreign and temporary files use the DMS OPEN mode mentioned, but are always opened with *SHARED-UPDATE=NO* if the *OPFL* operation is not specified.

USAGE mode	**SAM**	**ISAM PAM DAM**	**Current transaction**	**Permitted access by parallel transactions**	**DMS OPEN mode**
RETR retrieval	-	+	read	read write	INPUT SHARUPD
PRRT protected retrieval	+	+	read	read	INPUT
PRRR protected retrieval reverse	+	-	read backwards	read	REVERSE
EXRT exclusive retrieval	+	+	read	no access	INPUT
EXRR exclusive retrieval reverse	+	-	read backwards	no access	REVERSE
UPDT update	-	+	read write	read write	INOUT SHARUPD
PRUP protected update	-	+	read write	read	INOUT SHARUPD
EXUP exclusive update	+	+	read write	no access	INOUT or UPDATE
EXLD exclusive	+	+	write in load mode	no access	INOUT or EXTEND

Table 19: Defined USAGE modes for the OPTR operation (part 1 of 2)

USAGE mode	SAM	ISAM PAM DAM	Current transaction	Permitted access by parallel transactions	DMS OPEN mode
LOAD share load	-	+	read write (the ascending record key is allocated by LEASY)	read write	INOUT SHARUPD
PLOD protected load	-	+		read	INOUT SHARUPD
ELOD exclusive load	-	+		no access	INOUT
LDUP load + update	-	+		read write	INOUT SHARUPD
PLUP protected load + update	-	+		read	INOUT SHARUPD
ELUP exclusive load + update	-	+		no access	INOUT
ULRT unlocked retrieval	-	+	read	read write	INPUT SHARUPD
ULUP unlocked update	-	+	read write	read	INOUT SHARUPD

Table 19: Defined USAGE modes for the OPTR operation (part 2 of 2)

A blank can also be used to specify the USAGE mode for formats DB1 and DB2. In ISAM, DAM and PAM files it stands for USAGE mode=*UPDT*, and in SAM files for USAGE mode=*EXLD*.

These values correspond to the values specified for OPEN mode=blank.

The DMS OPEN mode listed in Table 19 only applies if the file was not previously opened by an *OPFL* operation.

Explanation of USAGE modes LOAD/PLOD/ELOD and LDUP/PLUP/ELUP

Multiple write-accessing of SAM files is not possible (DMS restriction).
However, the USAGE modes *LOAD/PLOD/ELOD* and *LDUP/PLUP/ELUP* have been defined to allow several tasks to write sequentially to a shared consecutive data entry file.

In this case LEASY, and not the user, assigns the record key. When records are inserted (*INSR* operation) the key value is increased by 1 (starting at 1) and is returned to the user in binary form as follows:

- in the key field specified in the file definition (ISAM files)
- in the first 4 bytes of the *AR* area (DAM files)
- in the *PAMHPNR* field of the *RE* area (PAM files).

The key length of an ISAM file (*KEYLEN*) must not exceed 4.

The difference between *LOAD/PLOD/ELOD* and *LDUP/PLUP/ELUP* is that only *INSR* is permitted and no lock elements are enforced for *LOAD/PLOD/ELOD* for write operations. With *LDUP/PLUP/ELUP* on the other hand all operations are permitted with the exception of *STOR*. A lock log is therefore kept for *LDUP* and *PLUP* (see the table “The following table shows the operations permitted according to the USAGE mode of the file identifier and the DMS file type (ACCESS-METHOD).” on page 190).

In order that this load procedure with key assignment by LEASY can be performed under protection/exclusively, the USAGE modes *PLOD/PLUP* and *ELOD/ELUP* were defined.

The user can insert a record with the *X’FF...FF’* key in order to improve performance when writing an ISAM file with the USAGE modes *LOAD* and *LDUP*. This record must be written with a USAGE mode in which LEASY itself does not assign the keys (e.g. *UPDT*) and is then ignored by LEASY when keys are assigned with USAGE modes *LOAD* and *LDUP*. This obviates the need to correct all the index levels of ISAM when a record is inserted.

Notes on the USAGE modes ULUP and ULRT

The USAGE modes *ULUP* and *ULRT* permit single write and multiple read access to a file without a lock log. They can be combined with each other, but not with other USAGE modes (see the table “Possible combinations of LEASY USAGE modes” on page 188).

Possible combinations of USAGE modes

The following table indicates the permitted USAGE modes for user *U2*, after user *U1* has opened a file with the USAGE mode specified:

U2 → **U1 ↓**	**RETR**	**UPDT**	**PRRT**	**PRRR**	**PRUP**	**EXRT**	**EXRR**	**EXUP**	**LOAD**	**LDUP**	**EXLD**	**PLOD**	**ELDO**	**PLUP**	**ELUP**	**ULRT**	**ULUP**
RETR	x	x	x	-	x	-	-	-	x	x	-	x	-	x	-	-	-
UPDT	x	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PRRT	x	-	x	x	-	-	-	-	-	-	-	-	-	-	-	-	-
PRRR	-	-	x	x	-	-	-	-	-	-	-	-	-	-	-	-	-
PRUP	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EXRT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EXRR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EXUP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LOAD	x	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-
LDUP	x	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-
EXLD	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PLOD	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ELOD	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PLUP	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ELUP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ULRT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x
ULUP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x	-

Table 20: Possible combinations of LEASY USAGE modes

Table 20 is valid for master and model files. A USAGE mode compatibility check is not performed for foreign and temporary files. In such cases the DMS compatibility rules apply.

If user *U1* opens with *OPFL* (prior to *OPTR*) and user *U2* opens without *OPFL* (prior to *OPTR*), then Table 20 applies only when the DMS OPEN mode selected from table 19 (page 185) for U2 is compatible with that selected from table 18 (page 184) for U1.

The table below shows where compatibility exists between USAGE modes specified by a user within a transaction for the same logical file but for different sequence identifiers

U2 → U1 ↓	RETR	UPDT	PRRT	PRUP	EXRT	EXUP	LOAD	PLOD	ELOD	LDUP	PLUP	ELUP	EXLD	PRRR	EXRR	ULRT	ULUP
RETR	RETR	UPDT	PRRT	PRUP	EXRT	EXUP	LOAD	PLOD	ELOD	LDUP	PLUP	ELUP	-	-	-	-	-
UPDT	UPDT	UPDT	PRUP	PRUP	EXUP	EXUP	-	-	-	-	-	-	-	-	-	-	-
PRRT	PRRT	PRUP	PRRT	PRUP	EXRT	EXUP	PLOD	PLOD	ELOD	PLUP	PLUP	ELUP	-	-	-	-	-
PRUP	PRUP	PRUP	PRUP	PRUP	EXUP	EXUP	-	-	-	-	-	-	-	-	-	-	-
EXRT	EXRT	EXUP	EXRT	EXUP	EXRT	EXUP	ELOD	ELOD	ELOD	ELUP	ELUP	ELUP	-	-	-	-	-
EXUP	EXUP	EXUP	EXUP	EXUP	EXUP	EXUP	-	-	-	-	-	-	-	-	-	-	-
LOAD	LOAD	-	PLOD	-	ELOD	-	LOAD	PLOD	ELOD	PLUP	PLUP	ELUP	-	-	-	-	-
PLOD	PLOD	-	PLOD	-	ELOD	-	PLOD	PLOD	ELOD	PLUP	PLUP	ELUP	-	-	-	-	-
ELOD	ELOD	-	ELOD	-	ELOD	-	ELOD	ELOD	ELOD	ELUP	ELUP	ELUP	-	-	-	-	-
LDUP	LDUP	-	PLUP	-	ELUP	-	PLUP	PLUP	ELUP	LDUP	PLUP	ELUP	-	-	-	-	-
PLUP	PLUP	-	PLUP	-	ELUP	-	PLUP	PLUP	ELUP	PLUP	PLUP	ELUP	-	-	-	-	-
ELUP	ELUP	-	ELUP	-	ELUP	-	ELUP	ELUP	ELUP	ELUP	ELUP	ELUP	-	-	-	-	-
EXLD	-	-	-	-	-	-	-	-	-	-	-	-	EXLD	-	-	-	-
PRRR	-	-	-	-	-	-	-	-	-	-	-	-	-	PRRR	EXRR	-	-
EXRR	-	-	-	-	-	-	-	-	-	-	-	-	-	EXRR	EXRR	-	-
ULRT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	ULRT	-
ULUP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	ULUP

Table 21: Rules for combining the USAGE modes of a logical file

If user *U1* opens with *OPFL* (prior to *OPTR*) and user *U2* opens without *OPFL* (prior to *OPTR*), then Table 20 applies only when the DMS OPEN mode selected from Table 19 for U2 is compatible with that selected from Table 18 for U1.

If a file identifier is opened first of all within a transaction with the USAGE mode *U1* and then subsequently a file identifier of the same logical file with USAGE mode *U2*, this results in the new common USAGE mode *U12* for the logical file according to Table 12.

U12 is compared with the USAGE mode specifications in parallel transactions of other users according to Table 21 and is then considered as the new result USAGE mode *U1* for the file.

It should be noted that Table 21 is diagonally symmetrical; accordingly, the order in which linking occurs is insignificant. When combining via Table 21, there is no difference between specifying the file identifier involved and its USAGE mode within the same *OPTR* operation or in an additional *OPTR* operation within the same transaction.

9.7 LEASY operations and compatible USAGE modes

The following table shows the operations permitted according to the USAGE mode of the file identifier and the DMS file type (*ACCESS-METHOD*).

Operation → **USAGE mode ↓**	RDIR	RHLD	SETL	RNXT	RNHD	RPRI	RPHD	INSR	STOR	REWR	DLET	LOCK	UNLK
RETR + PRRT + EXRT (ISAM + PAM + DAM)	x	x	x	x	x	x	x	-	-	-	-	x	x
UPDT + PRUP + EXUP+ ULUP (ISAM + PAM + DAM)	x	x	x	x	x	x	x	x	x	x	x	x	x
LDUP + PLUP + ELUP (ISAM + PAM + DAM)	x	x	x	x	x	x	x	x	-	x	x	x	x
LOAD + PLOD + ELOD (ISAM + PAM + DAM)	x	x	x	x	x	x	x	x	-	-	-	x	x
ULRT (ISAM + PAM + DAM)	x	-	x	x	-	x	-	-	-	-	-	-	-
EXLD (ISAM + PAM + DAM)	-	-	-	-	-	-	-	x	-	-	-	x	x
PRRT + EXRT (SAM)	x	x	x	x	x	-	-	-	-	-	-	-	x
PRRR + EXRR (SAM)	x	x	x	-	-	x	x	-	-	-	-	-	x
EXUP (SAM)	x	x	x	x	x	-	-	-	-	x	-	-	x
EXLD (SAM)	-	-	-	-	-	-	-	x	-	-	-	-	x

Table 22: LEASY operations as a function of the USAGE mode

Although the release functions are permitted for all modes and the lock functions are permitted for all ISAM, DAM and PAM USAGE modes, a lock log is kept only for the USAGE modes *UPDT*, *PRUP*, *RETR*, *LDUP* and *PLUP*.

10 COBOL interface

This chapter cannot be fully understood without first reading chapter “Overview of the LEASY program interface” on page 119ff. The sections of the two chapters have similar headings, which means they can be consulted without the need for explicit cross-references.

10.1 Calling LEASY

The user program calls LEASY by means of *CALL* statements via subroutine linkage as is common in high-level languages.

The standard registers used for this are:

R1	operand list address
R14	return address
R15	branch address

With a few exceptions, the operand definition is identical to that for KLDS.

Calls from the main program

```
                                 ⎧ CAT ⎫
CALL "LEASY" USING OP,RE,⎨ DB  ⎬,AR,FA,SI,KB,KE,US.
                                 ⎩ CI  ⎭
```

10.2 Defining the COBOL interface

The COBOL statement should include the operands required for the particular LEASY operation.

Operation code OP

OP specifies the operation to be performed by LEASY.

01	OP	PIC X(8).	U	Operation code

The permissible operation codes can be copied with COPY element *LEASYKON* from the OSM library *SYSLIB.LEASY.062* into the user program. They are defined in COPY element *LEASYKON* with the following field names (these field names should be used in the program, i.e. the constants should not be used).

```
01         OP-CODES.

           05 OP-CATD PIC X(4) VALUE     „CATD“.
           05 OP-OPFL PIC X(4) VALUE     „OPFL“.
           05 OP-CLFL PIC X(4) VALUE     „CLFL“.
           05 OP-OPTR PIC X(4) VALUE     „OPTR“
           05 OP-CLTR PIC X(4) VALUE     „CLTR“.
           05 OP-MARK PIC X(4) VALUE     „MARK“.
           05 OP-BACK PIC X(4) VALUE     „BACK“.
           05 OP-RDIR PIC X(4) VALUE     „RDIR“.
           05 OP-RNXT PIC X(4) VALUE     „RNXT“.
           05 OP-RPRI PIC X(4) VALUE     „RPRI“.
           05 OP-RHLD PIC X(4) VALUE     „RHLD“.
           05 OP-RNHD PIC X(4) VALUE     „RNHD“.
           05 OP-RPHD PIC X(4) VALUE     „RPHD“.
           05 OP-INSR PIC X(4) VALUE     „INSR“.
           05 OP-STOR PIC X(4) VALUE     „STOR“.
           05 OP-REWR PIC X(4) VALUE     „REWR“.
           05 OP-DLET PIC X(4) VALUE     „DLET“.
           05 OP-SETL PIC X(4) VALUE     „SETL“.
           05 OP-LOCK PIC X(4) VALUE     „LOCK“.
           05 OP-UNLK PIC X(4) VALUE     „UNLK“.
           05 OP-CINF PIC X(4) VALUE     „CINF“.
```

The LEASY operations are described in alphabetical order in section “LEASY operations” on page 147ff.

Reference area RE

The user both sends information to LEASY (indicated by *U*) and receives information from LEASY (indicated by *R*) via the reference area.

The reference area *RE* can be copied into the user program from OSM library *SYSLIB.LEASY.062* using the COPY element *LEASYPAR*, *LEASYRE* or *LEASYREL*.

Those parts of the COPY elements concerning the reference area have the following format:

```
01   RE.                                       reference area
                                               (80 bytes)
  05 RE-K.                                     compatible part of RE
                                               (48 bytes)
     10 RC-CODE.
        15 RC-CC  PIC X(3).   R                compatible return code
        15 RC-KZ  PIC X.      R                system identifier "L"
        15 RC-LC  PIC X(4).   R                LEASY return code
     10 PASS   PIC X(8).                       reserved for password
     10 OPE.                                   operation extensions
        15 OPE-STX PIC X.     U                STXIT routine control
        15 OPE-OM  PIC X.     U                OPEN mode/USAGE mode
        15 OPE-LOG PIC X.     U                BIM logging control,
                                               only with OPTR
        15 FILLER  PIC X(5).                   not used

     10 INT.                                   internal key aspect
        15 SAMPTR  PIC X(4).  U/R              SAM retrieval address (24-bit)
        15 PAMHPNR REDEFINES SAMPTR PIC 9(8) COMP.
                              U/R              PAM block number
        15 SAMNUM  PIC X(4).  U/R              SAM retrieval address (31-bit)
        15 FILLER  REDEFINES SAMNUM PIC X(4).

     10 NUM        PIC 9(8).  R                No. of primary records
     10 IDE        PIC X(8).  U/R              identification field for
                                               DCAM
  05 RE-LEASY-EXT.                             LEASY extension of
                                               RE (32 bytes)
     10 REOP       PIC X(4).  R                last operation code
     10 REDB       PIC X(16). R                last file name
                                               (+SI name)
     10 L-OPT      PIC X.     U                version identifier,
                                               must contain "1"
     10 OPE1       PIC X.     U                ≥ OPTR/CLTR/LOCK/RDIR/
                                               Ì RHLD/RNHD/RPHD/CINF
     10 OPE2       PIC X.     U                Ó operation extensions
     10 OPE-WTIME  PIC 9(3).  U                timeout for locks
     10 RC-LCE     PIC X(5).  R                LEASY return code extension
     10 U-PROT     PIC X.     U                AIM logging identifier
```

U	transferred by the user
R	returned by the system

File allocation DB

This operation is used to allocate the files or file identifiers to be processed.

Various formats can be used to specify files/file identifiers, depending on their number and use.

Format DB1

This format is used when only **one** file or file identifier is to be processed.

Format for OPFL

01	DB	PIC X(8).	U	Logical file name

Format for OPTR and all read and write operations

01	DB	PIC X(12).	U	Name of file identifier

The OPEN or USAGE mode is indicated in *OPE-OM* (*RE* area) (value not equal to *X'FF'*).

Examples of DB1 formats

For *OPFL*:

```
01       DB1-OPFL.
  05     DB-MITABDAT  PIC X(8) VALUE "MITABDAT".
  05     DB-PROTDAT   PIC X(8) VALUE "PROTDAT".
```

For *OPTR*:

```
01       DB1-OPTR.
  05     DB-MITABDAT  PIC X(12) VALUE "MITABDAT/ABC".
  05     DB-PROTDAT   PIC X(12) VALUE "PROTDAT".
```

Format DB2

This format permits a variable number of logical files or file identifiers to be specified.

Format for OPFL

```
01       DBLISTE    PIC X(m) VALUE "(file,...)".
```

Format for OPTR

```
01       DBLISTE    PIC X(m) VALUE "(file-identifier,...)".
```

i Blanks are not permitted in the parenthesized expression.

The shared OPEN or USAGE mode is indicated in *OPE-OM* (*RE* area) (value not equal to *X'FF'*).

Examples of DB2 formats

For *OPFL*:

```
01       DB2-OPFL.
  05     DBLISTE    PIC X(18) VALUE "(MITABDAT,PROTDAT)".
```

For *OPTR*:

```
01       DB2-OPTR.
  05     DBLISTE    PIC X(22) VALUE "(MITABDAT/ABC,PROTDAT)".
```

Format DB3

This format may only be used for *CLFL* and *UNLK* operations. *ALL* addresses all allocated files.

```
01        DB        PIC X(5)  VALUE "(ALL)".
or
01        DB        PIC X(12) VALUE "ALL".
```

Format DB4

This format permits a *separate* OPEN or USAGE mode to be defined for each addressed file or file identifier.

Format for OPFL

for **one** file *file*

```
01          DB          PIC X(m) VALUE "(file,mod)".
```

for **several** files (*file1* to *filen*)

```
01          DBLISTE     PIC X(m) VALUE "((file1,mod),...)".
```

file	logical file name ≤ 8 characters
mode	OPEN mode (1 byte)
m	length for format DB4

Format for OPTR

for **one** file identifier

```
01          DB          PIC X(m) VALUE "(file-identifier,mode)"
```

for **several** file identifiers

```
01          DB          PIC X(m) VALUE "((file-identifier,mode),...)"
```

file-identifier	file [/fm]	≤ 12 bytes
fm	sequence identifier	≤ 3 bytes
mod	USAGE mode	= 4 bytes
m	length for format DB4	

Blanks are not permitted in the parenthesized expression.

The *OPE-OM* field of the *RE* area must be set with *X’FF’*. This is the identifier for the DB4 specification of the OPEN mode or USAGE mode in the *DB* operand.

The descriptions of the defined OPEN and USAGE modes are in section “Opening files and transactions” on page 184f.

Examples of DB4 formats

For *OPFL*:

```
01       DB4-OPFL.
  05     FILE      PIC X(9)   VALUE "(FILE,4)".
  05     FILE LIST PIC X(34)  VALUE "((FILE 1,4),(FILE 2,1),(FILE 3,2))".
```

For *OPTR*:

```
01       DB4-OPTR.
  05     FILE      PIC X(16)  VALUE "(FILE/FM1,RETR)".
  05     FILE LIST PIC X(48)  VALUE "((FILE/FM1,RETR),(FILE/FM2,UPDT),
                                     (FILE,EXUP))".
```

Currency information CI

This operand defines a variable-length area with currency information. It is prefixed by a 4-byte record length field.

```
01       CI.                                      currency information
  05     CI-SLF PIC S9(4) COMP.  U/R             length field
  05     CI-SLR PIC S9(4) COMP.  R               length field
  05     CI-INF PIC X(m).        R               information field with
                                                  length m
```

- For the operation *CINF* with *OPE1=_* the user must supply the length field *CI-SLF* with the estimated length of the information field prior to the call: *CI-SLF=4+m*.
  In reply he receives the actual length in *CI-SLF* and the currency information in *CI-INF*.
  If no transaction is open, *CI-SLF=0* is returned.
- For the operation *CINF* with *OPE1=F*, the user must supply the length field *CI-SLF* with the estimated length of the information field prior to the call:
  CI-SLF=4+m. For the operation *CINF* with *OPE1=F* and *OPE2=S*, the user must also store the logical file name of the required file (which has a length of 8 characters) in *CI-INF* prior to the call. If no file fulfills these requirements, *CI-SLF=0* is returned.

Catalog information CAT

This operand defines an area for transferring catalog information.

```
01       CAT.                                        catalog specifications
  05     CATNAME PIC X(24).     U                    name of LEASY catalog
  05     SUFFIX  PIC X(20).     U                    suffix for model files
```

- The name of the LEASY catalog *[:catid:][$userid.]file-catalog* must be specified in *CATNAME*.
- *CATNAME* and *SUFFIX* must be padded with blanks to the right.
- Support of MPVS

  If a LEASY application wants to be linked to the CMMAIN of a LEASY catalog which is not generated on the pubset of the ID under which the user program was started, a catalog identifier (*:catid:*) must be specified in *CATNAME* for the public volume set with the LEASY catalog.
- Implementation in multiprocessor systems

  If the LEASY catalog is on a foreign processor, the catalog identifier (*:catid:*) of this processor must be specified as part of the catalog name:

  [:catid:][$userid.]file-catalog

Input/output area AR

This operand defines an input/output area for read/write operations (record zone). The record zone must cover the length of the entire record for write operations. It can be restricted to key fields for read operations.

```
01        AR        PIC X(n).     U/R           record zone with length n
```

The record length field is sent in the record zone for read operations where the record format is variable; it must be loaded by the user for write operations.

Examples

SAM file with fixed record length

```
01        AR        PIC X(20).
```

Output to print file

```
01        AR.
  05      VORSCHUB PIC X.            The user must provide the feed
  05      SATZ     PIC X(20).        control character
```

File with variable record length

```
01        AR.
  05      LAENGE   PIC S9(4) COMP VALUE 104.
  05      FILLER   PIC XX VALUE SPACE.
  05      SATZ     PIC X(100).
```

ISAM file with variable record length and secondary keys

```
01        AR.
  05      SL       PIC S9(4) COMP VALUE 150.
  05      FILLER   PIC XX.
  05      DATEN1   PIC X(12).
  05      PRIMKEY  PIC X(4).
  05      SIKEY1   PIC X(20).
  05      SIKEY2   PIC X(10).
  05      DATEN2   PIC X(100).
```

For ISAM files, the key (or partial keys) must always be given with the same position and length as in the DMS file catalog (or in the LEASY catalog); this cannot be verified.

PAM file with two secondary keys

```
01        AR.
  05      DATEN1   PIC X(10).
  05      SIKEY1   PIC X(10).
  05      DATEN2   PIC X(5).
  05      SIKEY2   PIC X(15).
  05      DATEN3   PIC X(2008).
```

PAM files require the primary key (= PAM block number) to be supplied in the *PAMHPNR* field of the *RE* area; reservation must be made in the record zone for any SI keys.

DAM file with two secondary keys

```
01        AR.
  05      PRIMKEY  PIC S9(8) COMP.
  05      DATEN.
    10    DATEN1   PIC X(10).
    10    SIKEY1   PIC X(5).
    10    DATEN2   PIC X(15).
    10    SIKEY2   PIC X(8).
    10    DATEN3   PIC X(32).
```

DAM files require the primary key (relative record number of the record in the file) to be supplied in the first 4 bytes of the *AR* area **prior** to the data record.

Field selection FA

This operand determines whether the entire record or only the key values are to be returned to the record zone *AR*.

To ensure compatibility with KLDS, the *FA* operand must be specified if the 6th operand of the *CALL* statement *SI* is used.

Format 1

This format determines that the entire record is to be returned to the record zone *AR*.

```
01        FA       PIC  X(5) VALUE "(ALL)".
or
01        FA       PIC  X(12) VALUE "ALL".
```

Format 2

This format determines that only key values are to be returned to the record zone *AR*.

```
01        FA       PIC  X(8) VALUE "MAINITEM".
```

Secondary index SI

This operand defines the name of a secondary index to be used for accessing.

```
01        SI       PIC X(8).      U             name of secondary index
```

The name is 8 characters long. If it comprises the character string *MAINITEM* or blanks, the primary key is used for accessing.

Key begin KB and key end KE

The *KB* and *KE* operands can be used to define a key range for the index specified by the *SI* operand. The contents of *KB* may be greater than, less than or equal to the contents of *KE*.

```
01        KB        PIC X(m).         U          key value with length m
```

```
01        KE        PIC X(m).         U          key value with length m
```

Examples

Access via the ISAM primary key (5 bytes long)

```
01        KB        PIC X(5) VALUE "A".
01        KE        PIC X(5) VALUE "BZZZZ".
```

Smallest key: *Abbbb*, where *b*=blank, on account of the COBOL convention for the PIC clause.

Accessing a PAM file

```
01        KB        PIC 9(8) COMP VALUE 1.
01        KE        PIC 9(8) COMP VALUE 5.
```

The range consists of PAM block numbers 1-5.

Accessing a SAM file

```
01        KB        PIC 9(8) COMP VALUE 257.
01        KE        PIC 9(8) COMP VALUE 511.
```

The range comprises the retrieval addresses *X’00000101*’ through *X’000001FF’* (first to last record of the first block in the file, with 4 byte retrieval addresses)

Accessing a DAM file

```
01        KB        PIC 9(8) COMP VALUE 1.
01        KE        PIC 9(8) COMP VALUE 5.
```

The range comprises the relative DAM record numbers 1-5.

Access via a secondary key (20 bytes long)

```
01        KB        PIC X(20) VALUE "ADAM".
01        KE        PIC X(20) VALUE "BERTA".
```

Access via a secondary key comprising 3 key parts

```
01        KB.
          05 KB-TEIL1 PIC X(2)  VALUE "AA".
          05 KB-TEIL2 PIC X(5)  VALUE LOW-VALUE.
          05 KB-TEIL3 PIC X(10) VALUE LOW-VALUE.

01        KE.
          05 KE-TEIL1 PIC X(2)  VALUE "EZ".
          05 KE-TEIL2 PIC X(5)  VALUE HIGH-VALUE.
          05 KE-TEIL3 PIC X(10) VALUE HIGH-VALUE.
```

User area US

This operand defines a user area for user information.

```
01        US.                                Record zone (n bytes long)
          05 LAENGE   PIC S9(4)     COMP VALUE n.
          05 FILLER   PIC XX        VALUE SPACE.
          05 SATZ     PIC X(n-4).
```

10.3 LEASY operations

The LEASY operations can be divided into 4 groups:

Control operations		CATD, OPFL, CLFL
Transaction operations		OPTR, CLTR, MARK, BACK
Operations at file level	Read operations	RDIR, RNXT, RPRI
	Read operations with lock function	RHLD, RNHD, RPHD
	Positioning operation	SETL
	Write operations	INSR, STOR, REWR, DLET
	Lock operations	LOCK, UNLK
Operation for currency information retrieval		CINF

Table 23: LEASY operations

These operations are described below in alphabetical order.

BACK Execute rollback

The *BACK* operation executes a rollback. The current transaction is canceled using the BIM file. Record locks are released and a restart point is set. See also “BACK Execute rollback” on page 148.

```
CALL "LEASY" USING  OP,RE[,US].
```

CATD Call LEASY catalog

The *CATD* operation assigns the LEASY catalog created with the LEASY-CATALOG utility routine. See also “CATD Call LEASY catalog” on page 149.

```
CALL "LEASY" USING OP,RE,CAT[,US].
```

CINF Transfer currency information

The *CINF* operation supplies all currency information in the *CI* area. It takes the form of a list of all file identifiers opened in the transaction and their current file pointers, or information on the files contained in the LEASY catalog and their secondary indices. See also “CINF Transfer currency information” on page 151.

```
CALL "LEASY" USING OP,RE,CI.
```

The following additional functions can be requested in the *RE* area:

OPE1=_	Currency information on all file identifiers opened in the transaction.
OPE1=F	Currency information on the files contained in the LEASY catalog and their secondary indices.
OPE2=$\left\{\begin{matrix}\_\\ \text{C}\end{matrix}\right\}$	Currency information (type 1) on all files contained in the LEASY catalog and their secondary indices.
OPE2=O	Currency information (type 1) on all files opened with the aid of *OPFL*.
OPE2=T	Currency information (type 1) on all files involved in the transaction.
OPE2=S	Currency information (type 2) on the file specified in *CI*.
OPE2=W	The help function immediately preceding this is to be continued.

CLFL Close files

The *CLFL* operation closes the specified files. These must have been opened by previous *OPFL* operations. See also “CLFL Close files” on page 153.

```
                          ⎧DB1⎫
CALL "LEASY" USING OP,RE[,⎨DB2⎬][,US].
                          ⎩DB3⎭
```

Operand *DB3* can be specified to provide compatibility with the interface KLDS, but it has no other effect.

CLTR Close transaction

The *CLTR* and operation terminates the transaction and sets a restart point. See also “CLTR Close transaction” on page 154.

```
CALL "LEASY" USING OP,RE[,US].
```

The following additional functions can be requested in the *RE* area:

OPE1=R	roll back the transaction.
OPE2=T	terminate the transaction and open a continuation transaction.

DLET Delete record

The *DLET* operation deletes a record from an ISAM or DAM file or a block from a PAM file. See also “DLET Delete record” on page 155.

```
CALL "LEASY" USING OP,RE,DB1[,AR[,FA,SI,KB]][,US].
```

The following options are available depending on the number of operands and the file type:

1. ISAM/DAM/PAM, 7 operands

   The key value of the record to be deleted must be stored in *KB*.

2. ISAM/DAM, 4 operands

   The key value of the record to be deleted is transferred at the defined positions in the *AR* area.

3. ISAM/DAM/PAM, 3 operands

   – In the case of ISAM and PAM files *DLET* deletes the last record to be read successfully using the same file identifier without explicitly transferring the key value.

   – In the case of PAM files an input must be made to the *PAMHPNR* field in the *RE* area. *DLET* deletes the block with the block number specified there. If the *PAMHPNR* field is set to zero, *DLET* deletes the last block to be read successfully using the same file identifier.

INSR Insert new record

The *INSR* operation inserts a new record or block into the specified file. See also "INSR Insert new record" on page 156.

```
CALL "LEASY" USING OP,RE,DB1,AR[,US].
```

In the *AR* area the entire record or block must be transferred.

In the case of PAM files an input must be made to the *PAMHPNR* field in the *RE* area.

LOCK Set record lock

The *LOCK* operation enforces lock elements for individual records or blocks or for file sections from ISAM, DAM or PAM files. See also "LOCK Set record lock" on page 157.

```
CALL "LEASY" USING OP,RE,DB1,[,AR[,FA,SI,KB[,KE]]].
```

The following options are available for transferring keys depending on the number of operands and the file type:

ISAM/DAM/PAM, 8 operands

The key values of the range to be locked must be stored in *KB* and *KE*.

ISAM/DAM/PAM, 7 operands

The key value of the record or block to be locked must be stored in *KB*.

ISAM/DAM, 4 operands

The key value of the record to be locked must be transferred at the defined position in the *AR* area.

PAM, 3 operands

The key value of the block to be locked must be stored in the *PAMHPNR* field.

The following additional functions can be requested in the *RE* area:

OPE1=S	A READ-LOCK is enforced
OPE1=_	A WRITE-LOCK is enforced

MARK Create checkpoint

The *MARK* operation terminates the current transaction and sets a restart point. See also “MARK Create checkpoint” on page 160.

```
CALL "LEASY" USING OP,RE[,US].
```

OPFL Open files

The *OPFL* operation opens the files specified in the file list in accordance with the relevant OPEN mode. See also “OPFL Open files” on page 161.

```
                         ⎧DB1⎫
CALL "LEASY" USING OP,RE,⎨DB2⎬[,US].
                         ⎩DB4⎭
```

OPTR Open or extend transaction

The *OPTR* operation is used to open a transaction (see also “OPTR Open or extend transaction” on page 162). There are two different functions with different operand specifications for this operation.

Function 1: Defining the start of a transaction or extending a transaction-oriented file list (see page 162)

```
                         ⎧DB1⎫
CALL "LEASY" USING OP,RE,⎨DB2⎬[,US].
                         ⎩DB4⎭
```

Function 2: Opening a transaction, and opening and positioning file identifiers in accordance with CI (see page 164).

```
CALL "LEASY" USING OP,RE (with OPE1=W),CI[,US].
```

The *OPE1* field in the *RE* area must be set to *"W"*.

The *CI* area must contain the previously saved currency information.

RDIR/RHLD

Directly read record/Directly read and lock record

The *RDIR* operation reads a record or block. The *RHLD* operation reads a record or block and locks it. See also “RDIR/RHLD Directly read record / Directly read and lock record” on page 166.

```
CALL "LEASY" USING OP,RE,DB1,AR[,FA[,SI[,KB[,KE]]]].
```

If the *SI* operand is missing, access is performed via the **primary index** if this is empty (blanks) or contains *MAINITEM*.

Access is performed via the **secondary index** if it is specified in the *SI* operand (only possible for ISAM, DAM and PAM files).

If 8 operands are specified - definition of a range (*KB,KE*) - and *KB* < *KE*, the record having the lowest key value in the range specified is used; where *KB* > *KE*, the record with the highest key value is used.

If a record with a specific key is to be read, there are three possibilities:

– *KB=KE*
– specifying 7 operands and supplying *KB*
– specifying 4 to 6 operands and supplying the key in the *AR* area (ISAM and DAM) or in the *RE* area (PAM and SAM)

Transfer of a single key value

– When 7 operands are used, the key value (ISAM primary key, PAM block number, SAM retrieval address, DAM record number or secondary key) is transferred via the *KB* operand.

– When 4 to 6 operands are used, a secondary key value or an ISAM primary key value must be stored at the defined positions in the input/output area *AR* before accessing. A DAM record number must be stored in the first 4 bytes. A PAM block number must be specified in the *PAMHPNR* field of the *RE* area, a SAM retrieval address in 24-bit format must be specified in the *SAMPTR* field and a SAM retrieval address in 31-bit format must be specified in the *SAMPTR* and *SAMNUM* field.

When accessing via the secondary index, the following additional information can be requested in the *RE* area if only a single key value is specified:

OPE2=N	The number of primary keys for a secondary key value is transferred in the NUM field of the *RE* area.

The following additional functions can be requested in the *RE* area for the *RHLD* operation:

OPE1=S	A READ-LOCK is enforced
OPE1=_	A WRITE-LOCK is enforced

REWR Rewrite record

The *REWR* operation updates an existing record or block. See also “REWR Rewrite record” on page 172.

```
CALL "LEASY" USING OP,RE,DB1,AR[,US].
```

Transferring the key value

SAM file	The record to be updated must have been read beforehand. It can be updated immediately afterwards.
ISAM/DAM file	The record with the primary key stored in the *AR* record zone is updated.
PAM file	The block with the primary key stored in the *PAMHPNR* field of the *RE* area is updated.

ℹ The following conditions apply if the file is opened as a foreign file with SHARED-UPDATE:

- In the case of ISAM and DAM files, the record must have been read and locked with *RHLD/RNHD/RPHD* immediately prior to the *REWR* operation.
- In the case of PAM files, the blocks must already have been locked with *RHLD/RNHD/RPHD/LOCK*.

RNXT/RNHD

Read next record/Read and lock next record

The *RNXT* operation reads the next record or block of the specified file sequentially. The *RNHD* operation reads the next record or block of the specified file sequentially, and locks it. See also “RNXT/RNHD Read next record / Read and lock next record” on page 173.

The operations are read towards the end of the file for SAM files, or in ascending order of the primary or secondary keys for ISAM, DAM or PAM files.

```
CALL "LEASY" USING OP,RE,DB1,AR[,FA].
```

With SAM files, the value of the current retrieval address is returned in the *SAMPTR* field (24-bit format) and in the *SAMPTR* and *SAMNUM* fields (31-bit format) of the *RE* area. With PAM files, the block number is returned in the *PAMHPNR* field of the *RE* area. With DAM files, the relative record number is returned in the first four bytes of the *AR* area.

If the entire record or block is to be read, the *FA* operand must be set to *ALL* or *(ALL)*, or only 4 operands may be specified.

If *FA* is set to *MAINITEM*, only the key fields will be supplied.

The following additional functions can be requested in the *RE* area for the *RNHD* operation:

OPE1=S	A READ-LOCK is enforced
OPE1=_	A WRITE-LOCK is enforced
OPE2=L	Locked records are skipped
OPE2=_	Locked records are not skipped

RPRI/RPHD

Read previous record/Read and lock previous record

The *RPRI* operation reads the next record or block in the specified file sequentially. The *RPHD* operation reads the next record or block in the specified file sequentially and locks it. The operations are read towards the beginning of the file for SAM files, or in descending order of the primary or secondary keys for ISAM, DAM and PAM files.

See also “RPRI/RPHD Read previous record / Read and lock previous record” on page 175.

```
CALL "LEASY" USING OP,RE,DB1,AR[,FA].
```

With SAM files, the value of the current retrieval address is returned in the *SAMPTR* field (24-bit format) and in the *SAMPTR* and *SAMNUM* fields (31-bit format) of the *RE* area. With PAM files, the block number is returned in the *PAMHPNR* field of the *RE* area. With DAM files, the relative record number is returned in the first four bytes of the *AR* area.

If the entire record or block is to be read, the *FA* operand must be set to *ALL* or *(ALL)*, or only 4 operands may be specified.

If *FA* is set to *MAINITEM*, only the key fields will be supplied.

The following additional functions can be requested in the *RE* area for the *RPHD* operation:

OPE1=S	A READ-LOCK is enforced
OPE1=_	A WRITE-LOCK is enforced
OPE2=L	Locked records are skipped
OPE2=_	Locked records are not skipped

SETL Position file pointer

The *SETL* operation positions an internal file pointer to a specified record or block. See also "SETL Position file pointer" on page 177.

```
CALL "LEASY" USING OP,RE,DB1[,AR[,FA,SI[,KB[,KE]]]].
```

SETL with 7 or 8 operands

- When accessing via the primary key, *KB/KE* must be supplied with the key values in the same way as for the *RDIR/RHLD* operation.
- If it is necessary to transfer a key pair formed by combining the primary and LEASY secondary keys, the special *KB/KE* format must be observed (see page 177f).

SETL with 4 to 6 operands

- The key values must be transferred in the record zone *AR*.
- In the case of PAM files an additional input must be made to the *PAMHPNR* field in the *RE* area; in the case of SAM files an input must be made to the *SAMPTR* field in the *RE* area.

STOR Insert record

The *STOR* operation inserts a record or block in the specified file. See also "STOR Insert record" on page 179.

```
CALL "LEASY" USING OP,RE,DB1,AR[,US].
```

The entire record or block must be transferred in the record zone *AR*.

With PAM files an input must be made to the *PAMHPNR* field.

With DAM files the primary key must be transferred in the first 4 bytes of the *AR* area.

UNLK Cancel record lock

The *UNLK* operation cancels record locks. See also “UNLK Cancel record lock” on page 180.

```
                           ⎧DB1[,AR[,FA,SI,KB[,KE]]]⎫
CALL "LEASY" USING OP,RE,[,⎨                        ⎬].
                           ⎩DB3                     ⎭
```

The key values are transferred in the same way as for the *LOCK* operation.

The third operand (*DB*) must be supplied with the file name or the file identifier in format DB1 or with *ALL* in format DB3.

Unlocking depends on the operation code extension *OPE1*:

OPE1 = ’_’	Only records or blocks locked (but not modified) in a transaction can be unlocked.
OPE1 = ’U’	Modified records and blocks can also be unlocked. See the preconditions on page 180ff.

11 Assembler interface

LEASY provides macros to support programming in Assembler.

They are divided into definition macros and action macros.

- **Definition macros** define data areas. They generate a series of DS and DC instructions and no operation section.
- **Action macros** execute operations (e.g. reading, writing). They generate an operation section and a data section.

The interface generated by the macros is identical to the COBOL interface described in the preceding chapter.

This chapter cannot be fully understood without first reading chapter “Overview of the LEASY program interface” on page 119ff. The sections of the two chapters have similar headings, which means they can be consulted without the need for explicit cross-references.

11.1 Operands used in the LEASY macros

Positional operands and keyword operands are specified for LEASY macros. Symbolic names can be employed for all macros.

Positional operands

Positional operands in action and definition macros are the symbolic addresses of data areas, general-purpose registers and direct operands. The following applies:

symb	Symbolic address of a data area. In action macros, the address is located either in an addressable DSECT or within the current base addressing. In definition macros it must be representable as an A-type constant.
(r)	Number or symbolic name of a general-purpose register (allocated with EQU by the user), in which the address of the data area is stored.
string	Direct operand in the form of a character string. With action macros, *string* must be specified enclosed in apostrophes ('*string*'). With definition macros, *string* is specified without apostrophes. The length restrictions laid down by BS2000 Assembler must be observed. If the definition macro is called within a DSECT or is itself generated as a DSECT, direct operands cannot be entered in the generation. The macro is in this case generated correctly; however, if the operand *MF=D* is specified, reference is made to the above by means of an *MNOTE* with severity *0*.

With the action macros and one definition macro (*LEA@CALL* with the *MF* operand), it is possible, depending on the macro, to specify between 1 and 9 positional operands. They designate the data areas that are defined via the corresponding definition macros and that are used for the transfer of the associated data to/from LEASY. They are equivalent to the LEASY interface operands of the same name with the following designations:

op	operation code
re	reference area
db	file allocation
ar	I/O area
fa	field selection
si	secondary index
kb	key begin
ke	key end
us	user information

The specifications *symb* and (*r*) are valid for all operands. The form '*string*' is permitted additionally for the operands *op*, *db*, *fa* and *si*. If register notation is employed, general-purpose registers 2 to 13 can be specified. The user must ensure that the most significant bit of the register is not set; otherwise the subsequent positional operands are not interpreted.

Keyword operands

Keyword operands comprise a keyword and an operand value. The value is assigned to the keyword using an equals sign. The following may be specified:

PAR=symb	A symbolic address is assigned.
PAR=(r)	A general-purpose register is assigned.
PAR=string	A character string without apostrophes is specified.
PAR=value	A decimal number is specified. The limits are indicated in the individual macro descriptions.
PAR=X'hexvalue'	Hexadecimal numbers are specified.

MF operand

The keyword operands *MF* and *PRE* can be specified for all definition macros (exception: *LEA@OPS*) and all action macros (exceptions: *LEA@PARC*, *LEA@TOLR*).

Permitted format for definition macros:

[,MF=$\left\{\begin{matrix}\underline{L}\\D\end{matrix}\right\}$][,PRE=prefix]

MF=$\underline{L}$	The area is generated as a normal data area (default value). This form enables the definition macro itself to be called within a DSECT or CSECT.
MF=D	The data area to be created is generated as a DSECT. *DS* and *DC* instructions following this macro are assigned to the generated DSECT.
PRE=prefix	A single letter must be specified. All symbolic names of the area generated are given the specified prefix; default value *L*. If the definition macro is preceded by a symbolic name and a DSECT is to be generated (*MF=D*), this name is used only to designate the DSECT. If no symbolic name is specified, the DSECT is given the name `prefix@@areaDS`

Example

Address	Macro	PRE=	DSECT name
-	LEA@OP	B	B@@OPDS
-	LEA@RE	-	L@@REDS
ABCD	LEA@RE	-	ABCD

Permitted format for action macros:

```
       ⎧L               ⎫
[,MF = ⎨D[,PRE=prefix]  ⎬ ]
       ⎩(E,address1)    ⎭
```

MF=L	An operand list of the specified positional operands (e.g. *re, db*) is generated in the macro. It is not possible here to specify the operands in the (*r*) format as a register.
MF=D[,PRE=prefix]	A DSECT is generated for the operand list and can be used for all LEASY calls. *PRE=* determines the prefix of the symbolic names.
MF=(E,address1)	*address1* is the address of an operand list. Within the macro itself only the instruction section is generated. The address is specified as a symbolic address (*symb*) or as a register ((*r*)). The operand list was generated by a macro (e.g. *LEA@CALL*) with the operand *MF=L*. If *MF=(E,address1)* and at least 1 positional operand is specified, the operand list is corrected via the positional operand. This applies also to operands that are irrelevant to the actual macro (e.g. specification of *KB*, *KE* for *LEA@RNXT*). If *MF=(E,address1)* is not specified, the code generated by the action macro is not reusable.

SAVE operand

With action macros, the following operand can also be specified:

`[,SAVE=address2]`

The address can be specified either as a symbolic address (*symb*) or via a register ((*r*)). The area defined is used to save the LEASY reference area (*re*) when corrections have to be made to it. It must therefore be 80 bytes long. This area is referred to as the buffer area.

P and T operands

In the reference area (*re*), individual fields can be modified via key operands as permanent or temporary corrections. The keywords are the names of the areas with the prefix *P* for permanent corrections and *T* for temporary corrections.

In the case of **permanent** corrections the areas addressed are updated. The alterations made here apply also to subsequent action macros as long as the same reference area (*re*) is used.

Temporary corrections are effected only in the buffer area and apply only to the current macro. The reference area is copied into the buffer area for this purpose. If no *SAVE* operand is specified, the buffer area is created within the action macro.

Where a correction is specified both as a permanent correction and as a temporary correction, first the former is effected in the actual operand area. This operand area is then copied into the buffer area and the temporary correction is carried out.

If an operand list is provided (*MF=(E,address1)*), permanent corrections are effective for the reference area specified via *address1*.

If additional temporary corrections are specified, the contents of the reference area (*re*) are moved to a buffer area, the fields of this area are corrected and the address of the buffer area entered permanently in the operand list. This enables the user to access the fields of the *RE* area just used following the LEASY macro.

If temporary corrections are specified and no buffer area is provided (*SAVE* operand), the code generated by the action macro is non-reentrant.

ERRCODE and ERRADDR operands

Where action macros are involved it is possible to distinguish between those error codes that can be tolerated and those that cannot. The following two keyword operands serve this purpose:

$$[\text{,ERRCODE} = \left\{\begin{array}{l}\text{(errorcode,errorcode,...)}\\ \text{address3}\end{array}\right\}\text{,ERRADDR = address4]}$$

[,ERRADDR=address4]

ERRCODE = (error-code,...)	An optional number of LEASY error codes is specified in the form of strings of 1 to 8 characters in length. They represent a list of tolerable error codes that are permitted for the appropriate macro. The code for valid execution need not be specified separately. If a 1, 2 or 4-character error code is specified for *ERRCODE*, a comparison takes place in the *RE* area starting from the address *L@@RCLC*, and in other instances (error code up to 8 characters long) starting from the address *L@@RCLC*. ℹ The length restrictions prescribed by BS2000 Assembler must be observed.
ERRCODE=address3	A symbolic address (*symb*) is specified here. The action macro branches to this address after the LEASY call. An error code distinction can be effected here, as provided, for example, by the macro *LEA@TOLR*. The mechanism is the same as that for error codes specified in the action macro, except that here the same error handling can be effected for several action macros.
ERRADDR=address4	Branching takes place to this address (*symb* or (*r*)) if an error code other than that specified for *ERRCODE* occurs. If *ERRADDR* is specified without *ERRCODE*, branching to *address4* takes place for each error code (except in the case of successful execution). *ERRCODE* must not be specified alone. At the time of entry into the return code analysis routine (*address3*), register 15 contains the address of the error routine (*address4*). Register 14 indicates the address following the action macro causing the error and register 1 is the base address register of the current *RE* area with the return code. In this way, the return code can be analyzed for temporary corrections as well without the *SAVE* operand being specified.

After the error routine has been executed, provision is made via general-purpose register 14 for returning to the address following the action macro that caused the error. This applies even if the error distinction is contained in a separate code section and accessed via *ERRCODE=address3*. The user is responsible for register saving and recovery within the error routine.

Symbolic names

Symbolic names can be specified preceding all definition and action macros. They may be up to 7 characters long.

With definition macros this name is used as a DSECT name if *MF=D* was specified. If there is no *MF=D* operand specification, a location counter is set to the first addressable data element following any necessary alignment.

If no DSECT is generated, the symbolic name specified is used only with the macro *LEA@AR* to form names within the area generated.

With action macros, the name is set before the generated code section with the instruction

```
Name DS OH
```

It can be used as a branch destination.

The names of the data elements of definition macros are allocated by the macros.

Operand errors that prevent the generation of a macro (or a submacro, as used, for example, for error handling) are rejected by an *MNOTE* with severity *1*. If field contents were specified during the formation of a DSECT, this is indicated by an *MNOTE* with severity *0*.

11.2 Register conventions

The action macros use the general-purpose registers 0, 1, 14 and 15. Register saving is not implemented for these registers, which means that the user should not expect the contents of these registers to remain unchanged macro after macro.

General-purpose registers 2 to 13 are safeguarded by the LEASY runtime system and are available again to the user with their original contents after each action macro.

Floating-point registers are not updated.

Branching to an error routine can be programmed for action macros (*ERRCODE=address3*). Returning from the error routine to the address following the action macro that caused the error is possible via general-purpose register 14. The user is responsible for register saving and recovery within the error routine.

11.3 Definition macros

List of definition macros

Macro	**Meaning**
LEA@AR	I/O area
LEA@CALL	Operand list
LEA@CAT	Catalog information
LEA@CI	Currency information
LEA@DB	File allocation (DB2/DB4)
LEA@DB1	File allocation (DB1)
LEA@FA	Field selection
LEA@OP	Operation code
LEA@OPS	Operation code/constant list
LEA@RE	Reference area
LEA@SI	Secondary index
LEA@US	User information

A description of the definition macros in alphabetical order is given below.

The headings for the descriptions of the individual macros include the area generated by the particular macro.

LEA@AR I/O area

This macro generates an I/O area for read and write operations. This area must span the length of a record and is aligned on a half-word boundary. It is also known as the record zone *AR*.

Operands *MF* and *PRE* are permitted (see page 220ff).

Name	Operation	Operands
`name`	`LEA@AR`	`[LEN=length]` `[,FOR=`$\left\{\begin{array}{c}\underline{\mathtt{V}}\\ \mathtt{F}\\ \mathtt{D}\end{array}\right\}$`]`

name	Where *MF=L*, *name* refers to the data area address. Where *MF=D*, *name* is the name of the DSECT. *name* is also used to form the field name (see below).
ℹ	The name which can be specified for the macro is used to form the names of the area fields. If no name is specified for the macro, *prefix@@AR* is used instead of *name* for forming field names, where *PRE=prefix* is evaluated (e.g. *Q@@ARD* for the data area if *PRE=Q* is specified).
LEN=length	The data area is generated with an overall length *length*. If this operand is not specified, only a location counter (with *DS OH*) is generated (specification of *FOR=V* after a standard record length field).
FOR=<u>V</u>	The data area is given a 4-byte standard record length field (default value). The name of the record length field is *nameS*; the name of the data area is *nameD*. *name* refers to the address of the record length field when *MF=L*; when *MF=D*, *name* is the name of the DSECT.
=F	The data area is generated without a record length field. The name of the data area is *nameD*. *name* refers to the address of the data area when *MF=L*; when *MF=D name* is the name of the DSECT.

=D	The data area is preceded by a 4-byte record number field (DAM format). The length of the record number field is not contained in the overall length. The record number field has the name *nameS*; the data area has the name *nameD*. *name* refers to the address of the record number field when *MF=L*; when *MF=D name* is the name of the DSECT.

Example

```
  STAMM    LEA@AR LEN=256
1          LEA@@BP TYP=L,PRE=L,NAM=STAMM,GRU=GEN,ID=AR
1 STAMM    DS    0H
1 STAMMS   DC    Y(256)
1          DC    CL2' '
1 STAMMD   DS    CL(256-4)
  *
           LEA@AR LEN=256,PRE=Q
1          LEA@@BP TYP=L,PRE=Q,NAM=,GRU=GEN,ID=AR
1 Q@@AR    DS    0H
1 Q@@ARS   DC    Y(256)
1          DC    CL2' '
1 Q@@ARD   DS    CL(256-4)
```

LEA@CALL Operand list

This macro generates an operand list for use in action macros. The operand list is generated for all 9 operands. The *MF* operand must be specified for this form of the *LEA@CALL* macro (*MF=L* or *D*); the *PRE* operand is permissible (see page 220ff).

Name	Operation	Operands
`name`	`LEA@CALL`	`[[op],[re],[db],[ar],[fa],[si],[kb],[ke],[us]]`

op	*symb* Address of the operation code area.
	’*string*’ Name of the operation.
re	*symb* Address of the reference area.
db	*symb* Address of the file list or file identifier.
	’*string*’ Logical file name or file identifier in the formats DB1/DB2/DB4.
ar	*symb* Address of the I/O area (record zone).
fa	*symb* Address of the *FA* area (Field Selection).
	’*string*’ Identifier for the field selection.
si	*symb* Address of the *SI* area (Secondary Index).
	’*string*’ Name of a secondary index.
kb	*symb* Address of the *KB* area (Key Begin).
ke	*symb* Address of the *KE* area (Key End).
us	*symb* Address of the *US* area (USer).

If positional operands are specified, their addresses are entered in the appropriate address fields, with the last address used being given the identifier for the last operand (*X’80’* in the highest byte). The operand forms *symb* (for all operands) and *string* (for *op, db, fa, si*) are permitted (the addresses must be displayable as A-type constants or asterisk addresses).

Address constants with address 0 are generated for positional operands that are not specified.

Example

```
 ADDRLIST LEA@CALL L@@OP,                                  -
 L@@RE,                                                    -
 L@@DB1,                                                   -
 L@@AR,                                                    -
 L@@FA,                                                    -
 L@@SI,                                                    -
 LKB01,                                                    -
 LKE01,                                                    -
 MF=L
1        LEA@@BP GRU=GEN,NAM=ADDRLIST,TYP=E,PRE=L
1      LEA@@AL L@@OP,L@@RE,L@@DB1,L@@AR,L@@FA,L@@SI,LKB01,LKE01,PRE=L
2 ADDRLIST DS    0F
2         DC    A(L@@OP+X'00000000')
2         DC    A(L@@RE+X'00000000')
2         DC    A(L@@DB1+X'00000000')
2         DC    A(L@@AR+X'00000000')
2         DC    A(L@@FA+X'00000000')
2         DC    A(L@@SI+X'00000000')
2         DC    A(LKB01+X'00000000')
2         DC    A(LKE01+X'80000000')
```

LEA@CAT Catalog information

This macro generates a data area for transferring catalog information. The area is 44 characters long, of which 24 are used for the catalog name and 20 for a suffix (for instances of model files).

Operands *MF* and *PRE* are permitted.

Name	Operation	Operands
`name`	`LEA@CAT`	`[catalog]` `[,catalog-suffix]`

The area defined by *LEA@CAT* is used only in the LEASY call *CATD* (*LEA@CATD* macro).

catalog	Name of the LEASY catalog in the form `[:catid:][$userid.]file-catalog` 24 characters long. If the LEASY catalog is not generated on the public volume set of the ID under which the user program is started, a catalog identifier :*catid*: must be specified for the public volume set with the LEASY catalog.
catalog-suffix	Suffix for model files (20 characters long).

Example

```
            LEA@CAT $USER.CATALOG,FEBRUARY
 1          LEA@@BP TYP=L,PRE=L,NAM=,GRU=GEN,ID=CAT
 1 L@@CAT   DS    0CL44
 1 L@@CATC  DC    CL24'$USER.CATALOG'
 1 L@@CATZ  DC    CL20'FEBRUARY'
```

LEA@CI Currency information

This macro generates an area of variable length for currency information. It begins with a 4-byte record length field.

The currency information is a list of the file identifiers opened during the current transaction and of current file pointers.

Operands *MF* and *PRE* are permitted.

Name	Operation	Operands
`name`	`LEA@CI`	`[LEN=length]`

The area defined by the macro *LEA@CI* is used only for the LEASY calls *CINF* (macro *LEA@CINF*) and *OPTR* with *OPE1=W* (macro *LEA@OPTR*).

LEN=length	Length of the *CI* area (including record length field) for transferring to LEASY.

Example

```
            LEA@CI LEN=80
  1         LEA@@BP TYP=L,PRE=L,NAM=,GRU=GEN,ID=CI
  1 L@@CI    DS    0H
  1 L@@CIS   DC    Y(80)
  1          DC    CL2' '
  1 L@@CID   DS    CL(80-4)
```

LEA@DB File allocation (DB2/DB4)

This macro generates a data area for allocating files or file identifiers to be processed in the formats DB2 and DB4 (see “File allocation DB” on page 133ff).

Operands *MF* and *PRE* are permitted.

Name	Operation	Operands
`name`	`LEA@DI`	`[string]` `[,LEN=length]`

string Files to be processed.

The following may be specified:

(*file*,...) or (*file-identifier*,...)

or

(*file,mod*) or (*file-identifier,mod*)

or

((*file,mod*),(*file,mod*),...) or ((*file-identifier,mod*), (*file-identifier,mod*),...).

file	file name as a string
file-identifier	file[/fm]
fm	sequence identifier
mod	OPEN or USAGE mode

(see formats DB2 and DB4, page 133ff).

LEN=length Length of the data area to be generated.

- If neither *string* nor *length* is specified, only a location counter (with EQU *) is generated.
- The length restrictions prescribed by BS2000 Assembler must be observed.

Example

```
               LEA@DB ((FILE1,1),(FILE2,2))
1              LEA@@BP TYP=L,PRE=L,NAM=,GRU=GEN,ID=DB
1 L@@DB        DC    '((FILE1,1),(FILE2,2))'
```

LEA@DB1 File allocation (DB1)

This macro generates a data area, 12 bytes long, for allocating the file or file identifier to be processed in the format DB1 (see “File allocation DB” on page 133ff).

Operands *MF* and *PRE* are permitted.

Name	Operation	Operands
`name`	`LEA@DB1`	`[file-identifier]`

file-identifier	Specification of the file or file identifier to be processed as a string (max. 12 characters) in the form:

filename[/fm]

filename	logical file name (max. 8 characters)
fm	sequence identifier of the file (max. 3 characters)

If the operand *file-identifier* is specified, the file name specified is inserted in the area (except when used within a DSECT or when *MF=D* is specified).

Example

```
  STAMDAT   LEA@DB1 STAMDAT
1           LEA@@BP TYP=L,PRE=L,NAM=STAMMDAT,GRU=GEN,ID=DB1
1 STAMDAT   DC    CL12'STAMDAT'
```

LEA@FA Field selection

This macro generates the field selection operand with a length of 8 bytes. It specifies whether the whole record is returned to the record zone (*ar*), or just the key contents (see “Field selection FA” on page 143ff).

Operands *MF* and *PRE* are permitted.

Name	Operation	Operands
`name`	`LEA@FA`	`[string]`

string may be:

{(ALL) ALL}	The file record is returned
MAINITEM	The key contents are returned.

Example

```
            LEA@FA ALL
1           LEA@@BP PRE=L,NAM=,GRU=GEN,ID=FA,TYP=L
1 L@@FA     DC    CL8'ALL'
```

LEA@OP Operation code

This macro generates an operation code area with a length of 4 bytes (see “Operation code OP” on page 123ff).

Operands *MF* and *PRE* are permitted.

Name	Operation	Operands
`name`	`LEA@OP`	`[operation]`

operation	Is specified as a string and must correspond to a valid LEASY operation. If *operation* is specified, the area is preset with the value of this operand, except when used within a DSECT or when *MF=D* is specified.

Example

```
  READNXT  LEA@OP RNXT
1          LEA@@BP TYP=L,PRE=L,NAM=READNXT,GRU=GEN,ID=OP
1 READNXT  DC     'RNXT'
```

LEA@OPS Operation code/Constant list

This macro provides a constant list of all LEASY operation codes.

Operand *PRE* is permitted.

Name	Operation	Operands
`name`	`LEA@OPS`	

The operation codes are defined using:

```
L@@xxxx  DC  C'xxxx'
          .
          .
          .
```

where *xxxx* represents one LEASY operation (e.g. *CATD*, etc.).

The symbolic names in the list are given a prefix via *PRE=*.

Example

```
          LEA@OPS
1 ********      LEASY OPERATION CODES
1 L@@CATD  DC    'CATD'                CONNECT TO / DISCONNECT FROM
1 *                                    COMMON MEMORY
1 L@@OPFL  DC    'OPFL'                OPEN FILE
1 L@@CLFL  DC    'CLFL'                CLOSE FILE
1 L@@OPTR  DC    'OPTR'                OPEN TRANSACTION
1 L@@CLTR  DC    'CLTR'                CLOSE TRANSACTION
1 L@@MARK  DC    'MARK'                MARK A CONSISTENCY POINT
1 L@@BACK  DC    'BACK'                BACK TO LAST CONSISTENCY POINT
1 L@@RDIR  DC    'RDIR'                READ DIRECT
1 L@@RHLD  DC    'RHLD'                READ DIRECT AND LOCK
1 L@@RNXT  DC    'RNXT'                READ NEXT
1 L@@RNHD  DC    'RNHD'                READ NEXT AND LOCK
1 L@@RPRI  DC    'RPRI'                READ PREVIOUSLY
1 L@@RPHD  DC    'RPHD'                READ PREVIOUSLY AND LOCK
1 L@@SETL  DC    'SETL'                SET FILE POINTER
1 L@@INSR  DC    'INSR'                INSERT
1 L@@STOR  DC    'STOR'                STORE
1 L@@REWR  DC    'REWR'                REWRITE
1 L@@DLET  DC    'DLET'                DELETE
1 L@@LOCK  DC    'LOCK'                LOCK
1 L@@UNLK  DC    'UNLK'                UNLOCK
1 L@@CINF  DC    'CINF'               READ CURRENT INFORMATION
```

LEA@RE Reference area

This macro generates a LEASY reference area. In this area the user sends information to LEASY (e.g. OPEN mode) or receives information from LEASY (e.g. return code).

Operands *MF* and *PRE* are permitted.

Name	Operation	Operands
`name`	`LEA@RE`	

The reference area generated is 80 bytes long. It is aligned on the word boundary.

Example

```
LEA@RE PRE=T
1 ********       LEASY COMMUNICATION AREA, VERSION 62A
1          DS    OF
1 T@@RE        DS    0CL80           LEASY COMMUNICATION AREA
1 T@@REK       DS    0CL48           COMPATIBLE PART
1 T@@RCCC      DC    CL3'000'        COMPATIBLE RETURNCODE
1 T@@RCOK      EQU   C'000'          RETURN-CODE: NO ERROR
1 T@@RCKZ      DC    CL1'L'          LEASY SYSTEM CHARACTERISTIC
1 T@@RCLC      DC    CL4'L000'       LEASY RETURN CODE
1 T@@PASS      DC    CL8' '          PASSWORD (FOR FUTURE VERSIONS)
1 T@@OPE       DS    0CL8            SUPPLEMENT FOR LEASY OPERATIONS
1 T@@OPSTX     DC    CL1' '          RESERVED
1 T@@STXA1     EQU   C' '             LEASY-STXIT-ROUTINE
1 T@@STXA2     EQU   X'00'            LEASY-STXIT-ROUTINE
1 T@@STXN      EQU   C' '             * VALUES 'N' AND 'P' NO LONGER
1 T@@STXP      EQU   C' '             * SUPPORTED SINCE LEASY V6.1
1 T@@OPOM      DC    CL1' '          OPEN MODE
1 T@@BLAN      EQU   C' '             STD FOR FORMAT DB1 AND DB2
1 T@@INPUT     EQU   C'1'             DVS OPEN MODE INPUT
1 T@@INPUS     EQU   C'2'             DVS OPEN MODE INPUT, SHARUPD
1 T@@INOUT     EQU   C'3'             DVS OPEN MODE INOUT
1 T@@INOUS     EQU   C'4'             DVS OPEN MODE INOUT, SHARUPD
1 T@@REVER     EQU   C'5'             DVS OPEN MODE REVERSE
1 T@@UPDAT     EQU   C'7'             DVS OPEN MODE UPDATE
1 T@@OUTPU     EQU   C'8'             DVS OPEN MODE OUTPUT
1 T@@EXTEN     EQU   C'9'             DVS OPEN MODE EXTEND
1 T@@OUTIN     EQU   C'A'             DVS OPEN MODE OUTIN
1 T@@OUTIS     EQU   C'B'             DVS OPEN MODE OUTIN, SHARUPD
1 *                              USAGE-MODES
1 T@@EXLD      EQU   C' '             USAGE-MODE EXLD (SAM)
1 T@@UPDT      EQU   C' '             USAGE-MODE UPDT (ISAM/PAM)
1 T@@RETRA     EQU   C'A'             USAGE-MODE RETR
1 T@@PRUPE     EQU   C'E'             USAGE-MODE PRUP
```

```
1 T@@EXRTG     EQU   C'G'             USAGE-MODE EXRT
1 T@@EXLDL     EQU   C'L'             USAGE-MODE EXLD
1 T@@PRRTI     EQU   C'I'             USAGE-MODE PRRT
1 T@@EXLDO     EQU   C'O'             USAGE-MODE EXLD
1 T@@ULRTR     EQU   C'R'             USAGE-MODE ULRT
1 T@@EXLDQ     EQU   C'Q'             USAGE-MODE EXLD
1 T@@ULUPU     EQU   C'U'             USAGE-MODE ULUP
1 T@@EXRTX     EQU   C'X'             USAGE-MODE EXRT
1 T@@EXUPB     EQU   C'B'             USAGE-MODE EXUP
1 *
1 T@@HVAL      EQU   X'FF'            FORMAT DB4
1 T@@OPLG      DC    CL1' '          BIM LOGGING FIELD
1 T@@YBIM      EQU   C' '             WITH BIM LOGGING
1 T@@NBIM      EQU   C'N'             WITHOUT BIM LOGGING
1          DC    CL5' '          UNUSED
1 T@@INT       DS    0XL8            FIELD FOR INTERNAL KEYS
1 T@@SPTR      DC    F'0'            SAM : ID1RPTR(24 BIT) OR
1 *                                         ID1BLK# (31-BIT)
1          ORG   T@@SPTR
1 T@@PAMHP     DC    F'0'            PAM : PAM BLOCK NUMBER
1 T@@SASNR     DC    F'0'            SAM:  ID1REC# (31-BIT)
1          ORG   T@@SASNR
1 T@@UNUSE     DC    F'0'            PAM :UNUSED, SAM UNUSED (24-BIT)
1 T@@NUM       DC    CL8' '          NUMBER OF PRIMARY RECORDS
1 T@@IDE       DC    CL8' '          DCAM IDENTIFIKATION
1 T@@RELE      DS    0CL32           LEASY EXTENSION OF RE
1 T@@REOP      DC    CL4' '          LAST OPERATION CODE
1 T@@REDB      DC    CL16' '         LAST FILE (+SI-NAME)
1 T@@LOPT      DC    CL1'1'          LEASY VERSION BYTE
1 *                               HAS TO CONTAIN '1'
1 T@@OPE1      DC    CL1' '          FOR OPTR,CLTR,RHLD,RNHD,RPHD,
1 *                              LOCK,CINF,UNLK
1 *                              OPTR,CLTR:
1 T@@OCST      EQU   C' '             STANDARD OPEN/CLOSE OF TRANSACTION
1 *                              OPTR:
1 T@@OPW       EQU   C'W'             OPEN TRANSAKTION USING
1 *                               FILE POINTERS IN CI-AREA
1 *                              CLTR:
1 T@@CLR       EQU   C'R'             ROLL BACK TRANSACTION
1 *                              RHLD,RNHD,RPHD,LOCK:
1 T@@SLOC      EQU   C'S'             READ LOCK (SHARE LOCK)
1 T@@WLOC      EQU   C' '             WRITE LOCK
1 *                              CINF:
1 T@@CINFW     EQU   C' '             CINF-AUFRUF FUER OPTR/W
1 T@@CIFI      EQU   C'F'             CINF FUER FILE-INFO
1 *
1 *                              UNLK:
1 T@@ULST      EQU   C' '             STANDARD FOR UNLK
```

```
1 T@@ULUP      EQU   C'U'             UNLK OF UPDATED RECORDS
1 *
1 T@@OPE2      DC    CL1' '      CLOSE TRANSACTION WITH
1 *                                  OR WITHOUT NEW START
1 T@@CLST      EQU   C' '         NO NEW START
1 T@@CLT       EQU   C'T'         NEW START
1 *
1 *
1 T@@NUMY      EQU   C'N'         RDIR/RHLD WITH NUM
1 T@@NUMN      EQU   C' '         RDIR/RHLD WITHOUT NUM
1 *                           NAEHERE ANGABEN FUER CINF
1 T@@CICA      EQU   C' '         INFO UEBER GANZEN KATALOG
1 T@@CICT      EQU   C'C'         INFO UEBER GANZEN KATALOG
1 T@@CIOP      EQU   C'O'         INFO UEBER OFFENE DATEIEN
1 T@@CITA      EQU   C'T'         INFO UEBER DATEIEN DER TA
1 T@@CISP      EQU   C'S'         INFO UEBER SPEZIELLE DATEI
1 T@@CIW       EQU   C'W'         FORTSETZUNG DER AUSKUNFTSFKT.
1 *
1 T@@ROL       EQU   C'L'         READ OVER LOCKED RECORD
1 T@@RNOL      EQU   C' '         DO NOT READ OVER LOCKED REC.
1 *
1 T@@OPWTM     DS   0ZL3          WAIT TIME FOR LOCKING
1          DC    XL3'0'       DEFAULT VALUE
1 *
1 T@@EXRC      DC    CL5' '       ZUSAETZLICHER RETURNCODE
1 T@@UPROT     DC    C' '          AIM PROTOKOLLIERUNGS-KENNZEICHEN
1 T@@NPROT     EQU   C' '          NO USER-INFO AVAILABLE
1 T@@YPROT     EQU   C'Y'          USER-INFORMATION AVAILABLE
```

LEA@SI Secondary index

This macro generates a data area for transferring a secondary index.

Operands *MF* and *PRE* are permitted.

Name	Operation	Operands
`name`	`LEA@SI`	`[index]`

index Is specified as a string and is up to 8 bytes long. The name of a secondary index or *MAINITEM* is given for *index*.
The primary key is accessed if *MAINITEM* is specified.

Example

```
   FAMNAME  LEA@SI FAMNAME
 1          LEA@@BP TYP=L,PRE=L,NAM=FAMNAME,GRU=GEN,ID=SI
 1 FAMNAME  DC    CL8'FAMNAME'
```

LEA@US User area

This macro generates an area of variable length for user information. It begins with a record length field 4 bytes long.

Operands *MF* and *PRE* are permitted.

Name	Operation	Operands
`name`	`LEA@US`	`[LEN=length]`

LEN=length	Length of the *USER* area (including the record length field) for transfer to LEASY; 5<=*length*<=1024

Example

```
             LEA@US LEN=1024
  1          LEA@@BP TYP=L,PRE=L,NAM=,GRU=GEN,ID=US
  1 L@@US    DS    OH
  1 L@@USL   DC    Y(1024)
  1          DC    CL2' '
  1 L@@USD   DS    CL(1024-4)
```

11.4 Action macros

The action macros execute the operations that, together with the positional operands used, are described individually in chapter “Overview of the LEASY program interface” on page 119ff.

List of action macros

Macro	**Meaning**
LEA@BACK	Execute rollback
LEA@CALL	Execute LEASY operation
LEA@CATD	Call LEASY catalog
LEA@CINF	Transfer currency information
LEA@CLFL	Close files
LEA@CLTR	Close transaction
LEA@DLET	Delete record
LEA@INSR	Insert new record
LEA@LOCK	Set record lock
LEA@MARK	Create checkpoint
LEA@OPFL	Open files
LEA@OPTR	Open transaction
LEA@PARC	Correct operand list
LEA@RDIR	Directly read record
LEA@REWR	Rewrite/update existing record
LEA@RHLD	Read and lock record
LEA@RNHD	Read and lock next record
LEA@RNXT	Read next record
LEA@RPHD	Read and lock previous record
LEA@RPRI	Read previous record
LEA@SETL	Position file pointer
LEA@STOR	Insert record
LEA@TOLR	Evaluate error codes
LEA@UNLK	Cancel record lock

The action macros are described below in alphabetical order.

LEA@BACK

Execute rollback

This macro executes a rollback. The current transaction is canceled using the BIM file. Record locks are released and a restart point is set. See “BACK Execute rollback” on page 148.

Operands *MF* and *PRE* are permitted.

Operation	Operands

`LEA@BACK`	`[[re],[us]]`
	`[,SAVE=address2]`
	`[,PIDE=ide][,TIDE=ide]`
	`[,ERRCODE=`$\left\{\begin{array}{l}\texttt{(error code,...)}\\ \texttt{address3}\end{array}\right\}$`,ERRADDR=address4]`
	`[,ERRADDR=address4]`

re	*symb* or (*r*) Address of the reference area.
us	*symb* or (*r*) Address of the *USER* area.
SAVE=adresse2	Address of the buffer area for saving the reference area.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)) Branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
LBACK      LEA@CALL ,L@@RE,MF=L
1          LEA@@BP GRU=GEN,NAM=LBACK,TYP=E,PRE=L
1          LEA@@AL ,L@@RE,,,,,,,PRE=L
2 LBACK    DS    0F
2          DC    A(X'00000000')
2          DC    A(L@@RE+X'80000000')
2          DC    A(X'00000000')
2          DC    A(X'00000000')
2          DC    A(X'00000000')
2          DC    A(X'00000000')
2          DC    A(X'00000000')
2          DC    A(X'00000000')

           LEA@BACK MF=(E,LBACK),                                      –
                ERRADDR=ERROR
1          LEA@CALL 'BACK',,                                           C
1                  MF=(E,LBACK),SAVE=,PRE=L,                           C
1                 PIDE=,TIDE=,                                         C
1                 ERRCODE=,ERRADDR=ERROR
2           LEA@@BP GRU=AKT,NAM=
2           LA    1,LBACK
2           CNOP  2,4
2           LA    15,*+34
2           TM    0(1),X'80'
2           BZ    *+8
2           O     15,=A(X'80000000')
2           ST    15,0(1)
2           LM    14,15,*+6
2           BR    15                     CALL LEASY
2           DC    A(*+12)
2           DC    V(LEASY)
2 *
2           DC    C'BACK'
2           L     1,4(1)                 R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          PRINT OFF
3          XR    15,15
3          BCTR  15,0                    R15=4X‘FF‘
3          LA    14,*+12+4+((L‘$@@RCCC+1)/2)*2     R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L‘$@@RCCC,1),*+8+4        TEST RC
3          BRE   14                      RC OK
3          B     ERROR                   RC: ERROR
3          DC CL(L‘$@@RCCC)‘000‘
```

LEA@CALL

Execute LEASY operation

Every LEASY operation can be executed using this macro. It is used as a submacro by all other action macros.

MF=(E,address1) may be specified.

Operation	Operands
LEA@CALL	

```
[[op],[re],[db],[ar],[fa],[si],[kb],[ke],[us]]

[,SAVE=address2]

[,POPE1=ext1]                  [,TOPE1=ext1]

[,POPE2=ext2]                  [,TOPE2=ext2]

[,POPEOM=openmode]             [,TOPEOM=openmode]

[,POPELOG=log]                 [,TOPELOG=log]

[,POPEWTM=waiting-time]        [,TOPEWTM=waiting-time]

[,PSAMPTR=X'sam-pointer']      [,TSAMPTR=X'sam-pointer']

[,PPAMHP=X'pam-block-number'] [,TPAMHP=X'pam-block-number']

[,PIDE=ide]                    [,TIDE=ide]

           ⎧(error-code,...)⎫
[,ERRCODE=⎨                ⎬ ,ERRADDR=address4]
           ⎩address3        ⎭

[,ERRADDR=address4]
```

op	*symb* or (*r*) Address of the operation code area.
	string Name of the operation.
re	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file list or file identifier.
	string Logical file name or file identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).

fa	*symb* or (*r*) Address of the *FA* area (field selection).
	string Identifier for the field selection.
si	*symb* or (*r*) Address of the *SI* area (Secondary Index).
	string Name of a secondary index.
kb	
	symb or (*r*) Address of the *KB* area (Key Begin).
ke	*symb* or (*r*) Address of the *KE* area (Key End).
us	*symb* or (*r*) Address of the *USER* area

The operation code is specified by means of *op*. The required or legitimate number of positional operands and the validity of keyword operands are dependent on this operation code (see the individual descriptions of the action macros). If *op* is specified in the form *string*, all further operands are subjected to a compatibility check. Otherwise, instead of logical checks, only syntax checks are made.

SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
POPE1=ext1 TOPE1=ext1	Operation code extension *OPE1* (*P*=permanent, *T*=temporary).
=W	Start of transaction with simultaneous file positioning; permitted only with *OPTR*.
=R	Transaction rollback; permitted only with *CLTR*.
=S	Read lock with *LOCK, RHLD, RNHD, RPHD* operations.
=F	Currency information on the files in the LEASY catalog and their secondary indices; permitted only with *CINF*.
=U	Unlock modified records; permitted only with *UNLK*.
=BLANK	Normal end of transaction with *CLTR* Write lock with *LOCK, RHLD, RNHD, RPHD* operations. Currency information on all the file identifiers opened in the transaction (with *CINF*).
POPE2=ext2 TOPE2=ext2	Operation code extension *OPE2* (*P*=permanent, *T*=temporary).

=T	Transaction termination and simultaneous transaction start with *CLTR*. Currency information on all the files involved in the transaction (with *CINF*).
=N	The number of primary keys for a secondary key value is determined; permitted only with *RDIR* and *RHLD*.
=C	Currency information on all the files in the LEASY catalog; permitted only with *CINF*.
=O	Currency information on all the files opened with the aid of *OPFL*; permitted only with *CINF*.
=S	Currency information on the file specified in *CI*; permitted only with *CINF*.
=W	The help function immediately preceding this is to be continued; permitted only with *CINF*.
=L	Locked records are to be skipped; permitted only with *RPHD* and *RNHD*.
=BLANK	Transaction closed and all file access requests canceled with *CLTR*. Currency information on all the files in the LEASY catalog (with *CINF*). The number of primary keys is not determined; permitted only with *RDIR* and *RHLD*.

POPEOM=openmode
TOPEOM=openmode
Open mode of files or transactions (*P*=permanent, *T*=temporary).

The following values are permitted:
1, 2, 3, 4, 5, 7, 8, 9, A, B, E, G, L, I, O, Q, X, BLANK and *X'FF'*; permitted only with *OPFL* and *OPTR*.

POPELOG=log
TOPELOG=log
Activation/deactivation of BIM saving (*P*=permanent,*T*=temporary); permitted only with *OPTR*.

=N	BIM saving deactivated for current transaction.
=BLANK	BIM saving activated.

POPEWTM=waiting-time TOPEWTM=waiting-time	Waiting time for locked records (*P*=permanent, *T*=temporary) permitted with *OPTR, RHLD, RNHD, RPHD, INSR, STOR* and *LOCK*.
=waiting-time	0 ≤ *waiting-time* ≤ 999.
=BLANK	The global waiting time for the session applies (*TIME* statement in LEASY-MAINTASK).

PSAMPTR=sam-pointer TSAMPTR=sam-pointer	Retrieval address for SAM files in 24-bit or 31-bit format (*P*=permanent, *T*=temporary); permitted only with *SETL* and *RDIR*.
=X’bbbbbbrr’	24-bit format.
=X’bbbbbbbbrrrrrrr’	31 bit format. (*b* = block number, *r* = record number within the block)

PPAMHP TPAMHP	PAM block number (*P*=permanent, *T*=temporary).
=X’bbbbbb’	Block number in hexadecimal characters permitted with *SETL, RDIR, RHLD, STOR, INSR, REWR, DLET, LOCK* and *UNLK* if a PAM file is being processed.

PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
ide= C'*string*' X'*string*'	The character string is up to 8 bytes in length, and represents the name of a DCAM application. If *C*’...’ is specified, the name must be filled to the right with blanks; if *X*’...’ is specified, the name must be filled to the right with binary zeros if it is shorter than 8 characters.
=symb	The symbolic address indicates an 8-byte field containing the name of a DCAM application or a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing the name of a DCAM application or a transaction identifier.

i The specifications *C'string'* and *X'string'* should only be used in conjunction with the action macro *LEA@CATD*, since the name of a DCAM application is only supplied with *LEA@CALL*. *ide=symb* or *ide=(r)* must be specified in *LEA@CALL* for all remaining action macros for returning the transaction identifier.

ERRCODE=(error-code,...)	
	List of the tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	
	Address (*symb* or (*r*)) Branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@CALL MF=(E,ADDRLIST)
1          LEA@@BP GRU=AKT,NAM=
1          LA    1,ADDRLIST              R1=A(PARAMS)
1          CNOP  2,4
1          LM    14,15,*+6
1          BR    15                      CALL LEASY
1          DC    A(*+8)
1          DC    V(LEASY)
1          L     1,4(1)                  R1=A(RE)
  *
  ADDRLIST LEA@CALL L@@OP,                                               -
                 L@@RE,                                                  -
                 L@@DB1,                                                 -
                 L@@AR,                                                  -
                 L@@FA,                                                  -
                 L@@SI,                                                  -
                 LKB01,                                                  -
                 LKE01,                                                  -
                 SAVE=T@@RE,                                             -
                 POPEWTM=0,                                              -
                 TOPEWTM=25,                                             -
                 ERRADDR=ERROR,                                          -
                 ERRCODE=(010,L012)
1          LEA@@BP GRU=AKT,NAM=ADDRLIST
1 ADDRLIST LA    14,L@@RE                R14=A(RE)
1          LEA@@KO KORR=PERM,                                            C
1                OPE1=,OPE2=,                                            C
1                OPEOM=,OPELOG=,OPEWTM=0,                                C
1                SAMPTR=,PAMHP=,IDE=
2          MVC   $@@OPWTM-$@@RE(L'$@@OPWTM,14),*+10     MODIFY OPEWTIM
2          B     *+4+((L'$@@OPWTM+1)/2)*2
2          DC    ZL(L'$@@OPWTM)'0'
1          MVC   T@@RE(L'$@@RE),0(14)       SAVE=TEMP. RE
1          LA    14,T@@RE
1          LEA@@KO KORR=TEMP,                                            C
1                OPE1=,OPE2=,                                            C
1                OPEOM=,OPELOG=,OPEWTM=25,                               C
1                SAMPTR=,PAMHP=,IDE=
2          MVC   $@@OPWTM-$@@RE(L'$@@OPWTM,14),*+10     MODIFY OPEWTIM
2          B     *+4+((L'$@@OPWTM+1)/2)*2
2          DC    ZL(L'$@@OPWTM)'25'
1          CNOP  2,4
1          LA    15,L@@OP
1          ST    15,*+78                 A(OP)
1          ST    14,*+78                 A(RE)
1          LA    15,L@@DB1
1          ST    15,*+74                 A(DB)
1          LA    15,L@@AR
1          ST    15,*+70                 A(AR)
1          LA    15,L@@FA
1          ST    15,*+66                 A(FA)
1          LA    15,L@@SI
1          ST    15,*+62                 A(SI)
1          LA    15,LKB01
1          ST    15,*+58                 A(KB)
1          LA    15,LKE01
1          ST    15,*+54                 A(KE)
1          OI    *+50,X'80'              LAST PARAM-ADDRESS
```

```
1          LM    14,15,*+10
1          LA    1,*+14
1          BR    15                      CALL LEASY
1          DC    A(*+40)
1          DC    V(LEASY)
1 *
1          DC    A(0)              PTR(OP)
1          DC    A(0)              PTR(RE)
1          DC    A(0)              PTR(DB)
1          DC    A(0)              PTR(AR)
1          DC    A(0)              PTR(FA)
1          DC    A(0)              PTR(SI)
1          DC    A(0)              PTR(KB)
1          DC    A(0)              PTR(KE)
1          L     1,4(1)                  R1=A(RE)
1          LEA@@FB ERRADDR=ERROR,ERRCODE=(010,L012),R14=ON
2          XR    15,15
2          BCTR  15,0                    R15=4X'FF'
2          LA    14,*+32+4+16+((L'$@@RCCC+1)/2)*2      R14=RETURN ADDRESS
2          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+12      TEST RC
2          BRE   14                      RC OK
2          B     *+4+((L'$@@RCCC+1)/2)*2 RC: ERROR
2          DC    CL(L'$@@RCCC)'000'
2          CLC   $@@RCCC-$@@RE(3,1),*+12      FEASIBLE RETURN CODE?
2          BRE   14                      YES, FEASIBLE
2          B     *+4+4
2          DC    C'010'
2          CLC   $@@RCLC-$@@RE(4,1),*+12      FEASIBLE RETURN CODE?
2          BRE   14                      YES, FEASIBLE
2          B     *+4+4
2          DC    C'L012'
2          B     ERROR                   RC: ERROR
```

LEA@CATD

Call LEASY catalog

This macro assigns the LEASY file catalog created by means of the LEASY-CATALOG utility routine. See “CATD Call LEASY catalog” on page 149.

Operands *MF* and *PRE* are permitted.

Operation	Operands

`LEA@CATD`	`[[re],[cat],[us]]`
	`[,SAVE=address2]`
	`[,PIDE=ide] [,TIDE=ide]`
	`[,ERRCODE={(error-code,... ) / address3} ,ERRADDR=address4]`
	`[,ERRADDR=address4]`

re	*symb* or (*r*) Address of the reference area.
cat	*symb* or (*r*) Address of the *CAT* area for the catalog assignment.
us	*symb* or (*r*) Address of the *USER* area.
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
ide={C’*String*’ X’*string*’}	The character string is up to 8 bytes in length, and represents the name of a DCAM application. If *C*’...’ is specified, the name must be filled to the right with blanks; if *X*’...’ is specified, the name must be filled to the right with binary zeros if it is shorter than 8 characters.
=symb	The symbolic address indicates an 8-byte field containing the name of a DCAM application.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing the name of a DCAM application.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.

ERRADDR=address4	Address (*symb* or (*r*)) Branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
          LA    R5,LCATD
          USING DCATD,R5
  *
  ECATD   LEA@CATD MF=(E,(R5)),
1 ECATD   LEA@CALL 'CATD',,,                                    C
1               MF=(E,(R5)),SAVE=,PRE=L,                        C
1               PIDE=,TIDE=,                                    C
1               ERRCODE=,ERRADDR=
2         LEA@@BP GRU=AKT,NAM=ECATD
2 ECATD   LR    1,R5
2         CNOP  2,4
2         LA    15,*+34
2         TM    0(1),X'80'
2         BZ    *+8
2         O     15,=A(X'80000000')
2         ST    15,0(1)
2         LM    14,15,*+6
2         BR    15                      CALL LEASY
2         DC    A(*+12)
2         DC    V(LEASY)
2 *
2         DC    C'CATD'
2         L     1,4(1)                  R1=A(RE)
```

LEA@CINF

Transfer currency information

This macro stores the entire currency information for the user in the *CI* area. The currency information comprises a list of all file identifiers opened in the transaction and their current file pointers, or information on the files contained in the LEASY catalog and their secondary indices. See “CINF Transfer currency information” on page 151.

Operands *MF* and *PRE* are permitted.

Operation	Operands

`LEA@CINF`	`[[re],[ci]]`
	`[,SAVE=address2]`
	$[,\text{POPE1}=\begin{Bmatrix}\text{F}\\ \\ \text{BLANK}\end{Bmatrix}][,\text{TOPE1}=\begin{Bmatrix}\text{F}\\ \\ \text{BLANK}\end{Bmatrix}]$
	$[,\text{POPE2}=\begin{Bmatrix}\text{C}\\ \text{O}\\ \text{T}\\ \\ \text{S}\\ \text{W}\\ \text{BLANK}\end{Bmatrix}][,\text{TOPE2}=\begin{Bmatrix}\text{C}\\ \text{O}\\ \text{T}\\ \\ \text{S}\\ \text{W}\\ \text{BLANK}\end{Bmatrix}]$
	`[,PIDE=ide]     [,TIDE=ide]`
	$[,\text{ERRCODE}=\begin{Bmatrix}\text{(error-code,...)}\\ \\ \text{address3}\end{Bmatrix},\text{ERRADDR=address4}]$
	`[,ERRADDR=address4]`

re	*symb* or (*r*) Address of the reference area.
ci	*symb* or (*r*) Address of the currency information.
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).

=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
POPE1 TOPE1	Operation code extension *OPE1* (*P*=permanent, *T*=temporary).
=F	Currency information on the files in the LEASY catalog and their secondary indices.
=BLANK	Currency information on all the file identifiers opened in the transaction. This information enables a transaction to be opened (with simultaneous file positioning) with the aid of *OPTR* and *OPE1=W*.
POPE2 TOPE2	Operation code extension *OPE2* (*P*=permanent, *T*=temporary).
$=\left\{\begin{array}{l}\text{C}\\\text{BLANK}\end{array}\right\}$	Currency information on all the files in the LEASY catalog.
=O	Currency information on all files opened with *OPFL*.
=T	Currency information on all files involved in the transaction.
=S	Currency information on the file specified in *CI*.
=W	The help function immediately preceding this is to be continued.
ERRCODE=(error-code,...)	List of the tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@CINF L@@RE,L@@CI,ERRADDR=ERROR
1          LEA@CALL 'CINF',L@@RE,L@@CI,                                  C
1                MF=,SAVE=,PRE=L,                                        C
1                POPE1=,POPE2=,PIDE=,                                    C
1                TOPE1=,TOPE2=,TIDE=,                                    C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          CNOP  2,4
2          LA    15,L@@RE
2          ST    15,*+38                 A(RE)
2          LA    15,L@@CI
2          ST    15,*+34                 A(DB)
2          OI    *+30,X'80'              LAST PARAM-ADDRESS
2          LM    14,15,*+10
2          LA    1,*+14
2          BR    15                      CALL LEASY
2          DC    A(*+24)
2          DC    V(LEASY)
2 *
2          DC    A(*+12)          PTR(OP)
2          DC    A(0)             PTR(RE)
2          DC    A(0)             PTR(DB)
2          DC    C'CINF'
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+12+4+((L'$@@RCCC+1)/2)*2    R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4         TEST RC
3          BRE   14                      RC OK
3          B     ERROR                   RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@CLFL Close files

This macro closes the specified files. These files must have been opened with *OPFL* in previous operations. See “CLFL Close files” on page 153.

Operands *MF* and *PRE* are permitted.

Operation	Operands
LEA@CLFL	see syntax below

```
Operation   Operands
LEA@CLFL    [[re],[db],[us]]

                       ⎧(error-code,...)⎫
            [,ERRCODE= ⎨                ⎬ ,ERRADDR=address4]
                       ⎩address3        ⎭

            [,ERRADDR=address4]
```

re	*symb* or (*r*) Address of the reference area.	
db	*symb* or (*r*) Address of the file list or file identifier.	
	’*string*’	Logical file name or file identifier or file list.
us	*symb* or (*r*) Address of the *USER* area.	
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.	
=address3	Symbolic address for error handling.	
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).	

Example

```
           LEA@CLFL (R4),(R5)
1          LEA@CALL 'CLFL',(R4),(R5),                                   C
1                MF=,PRE=L,                                             C
1                ERRCODE=,ERRADDR=
2          LEA@@BP GRU=AKT,NAM=
2          CNOP  2,4
2          ST    R4,*+34                 A(RE)
2          ST    R5,*+34                 A(DB)
2          OI    *+30,X'80'              LAST PARAM-ADDRESS
2          LM    14,15,*+10
2          LA    1,*+14
2          BR    15                      CALL LEASY
2          DC    A(*+24)
2          DC    V(LEASY)
2 *
2          DC    A(*+12)          PTR(OP)
2          DC    A(0)             PTR(RE)
2          DC    A(0)             PTR(DB)
2          DC    C'CLFL'
2          L     1,4(1)                  R1=A(RE)
```

LEA@CLTR Close transaction

The *LEA@CLTR* macro closes the transaction and sets a restart point. See “CLTR Close transaction” on page 154.

Operands *MF* and *PRE* are permitted.

Operation	Operands
LEA@CLTR	[[re],[us]] [,SAVE=address2] [,POPE1=$\left\{\begin{array}{l}\texttt{R}\\\texttt{BLANK}\end{array}\right\}$][,TOPE1=$\left\{\begin{array}{l}\texttt{R}\\\texttt{BLANK}\end{array}\right\}$] [,POPE2=$\left\{\begin{array}{l}\texttt{T}\\\texttt{BLANK}\end{array}\right\}$][,TOPE2=$\left\{\begin{array}{l}\texttt{T}\\\texttt{BLANK}\end{array}\right\}$] [,PIDE=ide] [,TIDE=ide] [,ERRCODE=$\left\{\begin{array}{l}\texttt{(error-code,...)}\\\texttt{address3}\end{array}\right\}$,ERRADDR=address4] [,ERRADDR=address4]

re	*symb* or (*r*) Address of the reference area.
us	*symb* or (*r*) Address of the *USER* area.
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
POPE1 TOPE1	Operation code extension *OPE1* (*P*=permanent, *T*=temporary).
=R	Transaction rollback.
=BLANK	Normal transaction termination.

POPE2 TOPE2	Operation code extension *OPE2* (*P*=permanent, *T*=temporary).
=T	Transaction termination and simultaneous transaction start.
=BLANK	Transaction termination and cancellation of all file access requests.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable codes (*ERRCODE*).

Example

```
           LEA@CLTR L@@RE,POPE1=BLANK,POPE2=T
1          LEA@CALL 'CLTR',L@@RE,                                           C
1                MF=,SAVE=,PRE=L,                                           C
1                POPE1=BLANK,POPE2=T,PIDE=,                                 C
1                TOPE1=,TOPE2=,TIDE=,                                       C
1                ERRCODE=,ERRADDR=
2          LEA@@BP GRU=AKT,NAM=
2          LA    14,L@@RE                R14=A(RE)
2          LEA@@KO OPCD=CLTR,KORR=PERM,                                     C
2                OPE1=BLANK,OPE2=T,                                         C
2                OPEOM=,OPELOG=,OPEWTM=,                                    C
2                SAMPTR=,PAMHP=,IDE=
3          MVI   $@@OPE1-$@@RE(14),C' '     MODIFY OPE1
3          MVI   $@@OPE2-$@@RE(14),C'T'         MODIFY OPE2
2          CNOP  2,4
2          LA    15,L@@RE
2          ST    15,*+30                 A(RE)
2          OI    *+26,X'80'              LAST PARAM-ADDRESS
2          LM    14,15,*+10
2          LA    1,*+14
2          BR    15                      CALL LEASY
2          DC    A(*+20)
2          DC    V(LEASY)
2 *
2          DC    A(*+8)           PTR(OP)
2          DC    A(0)             PTR(RE)
2          DC    C'CLTR'
2          L     1,4(1)                  R1=A(RE)
```

LEA@DLET Delete record

This macro deletes a record from an ISAM or DAM file or a block from a PAM file. See “DLET Delete record” on page 155.

Operands *MF* and *PRE* are permitted.

Operation	Operands

`LEA@DLET`	`[[re],[db],[ar],[fa],[si],[kb],[us]]`
	`[,SAVE=address2]`
	`[,PPAMHP=X'pam-block-number'][,TPAMHP=X'pam-block-number']`
	`[,PIDE=ide]` `[,TIDE=ide]`
	`[,ERRCODE=` $\left\{\begin{array}{l}\texttt{(error-code,...)}\\ \texttt{address3}\end{array}\right\}$ `,ERRADDR=address4]`
	`[,ERRADDR=address4]`

re	*symb* or (*r*) Address of the reference area.	
db	*symb* or (*r*) Address of the file identifier	
	’string’	File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).	
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).	
	’string’	Identifier for the field selection.
si	*symb* or (*r*) Address of the *SI* area (Secondary Index).	
	’string’	Name of a secondary index.
kb	*symb* or (*r*) Address of the *KB* area (Key Begin).	
us	*symb* or (*r*) Address of the *USER* area.	
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.	
PPAMHP TPAMHP	PAM block number (*P*=permanent, *T*=temporary).	
=X’bbbbbb’	Number of the PAM block to be deleted.	

PIDE=ide
TIDE=ide
Identifier for DCAM (*P*=permanent, *T*=temporary).

=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.

ERRCODE=(error-code,...)
List of tolerable error codes; 1 to 8 characters.

=address3	Symbolic address for error handling.

ERRADDR=address4
Address (*symb* or (*r*));
branch destination, if the particular error code is not in the list of tolerated error codes (*ERRCODE*).

Example

```
           LEA@DLET MF=(E,ADDRLIST),                                     -
                 ERRADDR=ERROR
1          LEA@CALL 'DLET',,,,,,,,                                       C
1                MF=(E,ADDRLIST),SAVE=,PRE=L,                            C
1                PPAMHP=,PIDE=,                                          C
1                TPAMHP=,TIDE=,                                          C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LA    1,ADDRLIST
2          CNOP  2,4
2          LA    15,*+34
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,0(1)
2          LM    14,15,*+6
2          BR    15                    CALL LEASY
2          DC    A(*+12)
2          DC    V(LEASY)
2 *
2          DC    C'DLET'
2          L     1,4(1)                R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                  R15=4X'FF'
3          LA    14,*+12+4+((L'$@@RCCC+1)/2)*2    R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4         TEST RC
3          BRE   14                    RC OK
3          B     ERROR                 RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@INSR Insert new record

This macro inserts a new record or block into the file specified. See “INSR Insert new record” on page 156.

Operands *MF* and *PRE* are permitted.

Operation	Operands
LEA@INSR	see below

```
[[re],[db],[ar],[us]]

[,SAVE=address2]

          ⎧waiting-time⎫                   ⎧waiting-time⎫
[,POPEWTM=⎨            ⎬]        [,TOPEWTM=⎨            ⎬]
          ⎩BLANK       ⎭                   ⎩BLANK       ⎭

[,PPAMHP=X'pam-block-number'][,TPAMHP=X'pam-block-number']

[,PIDE=ide]                     [,TIDE=ide]

          ⎧(error-code,...)⎫
[,ERRCODE=⎨                ⎬ ,ERRADDR=address4]
          ⎩address3        ⎭

[,ERRADDR=address4]
```

re	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file identifier. ’*string*’ File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).
us	*symb* or (*r*) Address of the *USER* area.
SAVE=adresse2	Address (*symb* or (*r*)) of the buffer area for saving the *RE* area.
POPEWTM TOPEWTM	Waiting time for locked records (*P*=permanent, *T*=temporary).
=waiting-time	$0 \leq$ *waiting-time* ≤ 999.

=BLANK	The global waiting time for the session applies; (*TIME* statement in LEASY-MAINTASK).
PPAMHP TPAMHP	PAM block number (*P*=permanent, *T*=temporary).
=X'bbbbbb'	Block number of the block into which information is inserted.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@INSR L@@RE,'PAMFILE     ',L@@AR,                          -
                 TOPEWTM=BLANK,                                          -
                 TPAMHP=X'1',                                            -
                 ERRADDR=ERROR
1          LEA@CALL 'INSR',L@@RE,'PAMFILE     ',L@@AR,                   C
1                MF=,SAVE=,PRE=L,                                        C
1                PPAMHP=,POPEWTM=,PIDE=,                                 C
1                TPAMHP=X'1',TOPEWTM=BLANK,TIDE=,                        C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LA    14,L@@RE                  R14=A(RE)
2          MVC   *+14(L'$@@RE),0(14)        TEMP. RE
2          LA    14,*+8
2          B     *+4+L'$@@RE
2          DS    CL(L'$@@RE)                 TEMP. RE
2          LEA@@KO OPCD=INSR,KORR=TEMP,                                  C
2                OPE1=,OPE2=,                                            C
2                OPEOM=,OPELOG=,OPEWTM=BLANK,                            C
2                SAMPTR=,PAMHP=X'1'
3          MVI   $@@OPWTM-$@@RE(14),C' ' MODIFY OPEWTIM
3          MVC   $@@OPWTM+1-$@@RE(L'$@@OPWTM-1,14),$@@OPWTM-$@@RE(14)
3          MVC   $@@PAMHP-$@@RE(L'$@@PAMHP,14),*+10 MODIFY PAMHP
3          B     *+4+L'$@@PAMHP
3          DC    XL(L'$@@PAMHP)'1'
2          CNOP  2,4
2          ST    14,*+38               A(RE)
2          LA    15,L@@AR
2          ST    15,*+38               A(AR)
2          OI    *+34,X'80'            LAST PARAM-ADDRESS
2          LM    14,15,*+10
2          LA    1,*+14
2          BR    15                    CALL LEASY
2          DC    A(*+40)
2          DC    V(LEASY)
2 *
2          DC    A(*+16)               PTR(OP)
2          DC    A(0)                  PTR(RE)
2          DC    A(*+12)               PTR(DB)
2          DC    A(0)                  PTR(AR)
2          DC    C'INSR'
2          DC    C'PAMFILE     '
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+12+4+((L'$@@RCCC+1)/2)*2 R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4 TEST RC
3          BRE   14                      RC OK
3          B     ERROR                RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@LOCK Set record lock

This macro enforces lock elements on individual records or blocks in ISAM, DAM or PAM files. Since the macro does not access the file, the existence of the record or block is not verified. This means that it is also possible to lock non-existent records or blocks. See “LOCK Set record lock” on page 157.

Operands *MF* and *PRE* are permitted.

```
Operation   Operands
LEA@LOCK    [[re],[db],[ar],[fa],[si],[kb],[ke]]

            [,SAVE=address2]

                    ⎧S    ⎫                      ⎧S    ⎫
            [,POPE1=⎨     ⎬              [,TOPE1=⎨     ⎬
                    ⎩BLANK⎭                      ⎩BLANK⎭

                      ⎧waiting-time⎫                     ⎧waiting-time⎫
            [,POPEWTM=⎨            ⎬]         [,TOPEWTM=⎨            ⎬]
                      ⎩BLANK       ⎭                     ⎩BLANK       ⎭

            [,PPAMHP=X'pam-block-number'][,TPAMHP=X'pam-block-number']

            [,PIDE=ide]                  [,TIDE=ide]

                      ⎧(error-code,...)⎫
            [,ERRCODE=⎨                ⎬ ,ERRADDR=address4]
                      ⎩address3        ⎭

            [,ERRADDR=address4]
```

re	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file identifier.
	’*string*’ File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).
	’*string*’ Identifier for the field selection.
si	*symb* or (*r*) Address of the *SI* area (Secondary Index).
	’*string*’ Name of a secondary index.
kb	*symb* or (*r*) Address of the *KB* area (Key Begin).

ke	*symb* or (*r*) Address of the *KE* area (Key End).
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the *RE* area.
POPE1 TOPE1	Operation code extension *OPE1* (*P*=permanent, *T*=temporary).
=S	A READ-LOCK is enforced.
=BLANK	A WRITE-LOCK is enforced.
POPEWTM TOPEWTM	Waiting time for locked records (*P*=permanent, *T*=temporary).
=waiting-time	0 ≤ *waiting-time* ≤ 999.
=BLANK	The global waiting time for the session applies; (*TIME* statement in LEASY-MAINTASK).
PPAMHP TPAMHP	PAM block number (*P*=permanent, *T*=temporary).
=X’bbbbbb’	Number of the block to be locked.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@LOCK MF=(E,ADDRLIST),                                     -
                 TOPEWTM=30,                                             -
                 ERRADDR=ERROR
1          LEA@CALL 'LOCK',,,,,,,,
1                MF=(E,ADDRLIST),SAVE=,PRE=L,                            C
1                PPAMHP=,POPEWTM=,PIDE=,POPE1=                           C
1                TPAMHP=,TOPEWTM=30,TIDE=,OPE1=                          C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LA    1,ADDRLIST
2          L     14,4(1)                 R14=A(RE)
2          MVC   *+14(L'$@@RE),0(14)        TEMP. RE
2          LA    14,*+8
2          B     *+4+L'$@@RE
2          DS    CL(L'$@@RE)                 TEMP. RE
2          LEA@@KO OPCD=LOCK,KORR=TEMP,                                  C
2                OPE1=,OPE2=,                                            C
2                OPEOM=,OPELOG=,OPEWTM=30,                               C
2                SAMPTR=,PAMHP=,IDE
3          MVC   $@@OPWTM-$@@RE(L'$@@OPWTM,14),*+10 MODIFY OPEWTIM
3          B     *+4+((L'$@@OPWTM+1)/2)*2
3          DC    ZL(L'$@@OPWTM)'30'
2          CNOP  2,4
2          LA    15,*+50
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,0(1)
2          TM    4(1),X'80'
2          BZ    *+8
2          O     14,=A(X'80000000')
2          ST    14,4(1)                 MODIFY RE
2          LM    14,15,*+6
2          BR    15                      CALL LEASY
2          DC    A(*+12)
2          DC    V(LEASY)
2 *
2          DC    C'LOCK'
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+12+4+((L'$@@RCCC+1)/2)*2 R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4 TEST RC
3          BRE   14                      RC OK
3          B     ERROR                RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@MARK

Create checkpoint

This macro closes the current transaction and sets a restart point. See “MARK Create checkpoint” on page 160.

Operands *MF* and *PRE* are permitted.

Operation	Operands

`LEA@MARK`	`[[re],[us]]`
	`[,SAVE=address2]`
	`[,PIDE=ide] [,TIDE=ide]`
	`[,ERRCODE=` $\left\{\begin{array}{l}\texttt{(error-code,...)}\\ \texttt{address3}\end{array}\right\}$ `,ERRADDR=address4]`
	`[,ERRADDR=address4]`

re	*symb* or (*r*) Address of the reference area.
us	*symb* or (*r*) Address of the *USER* area
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@CALL MF=(E,LMARK)
1          LEA@@BP GRU=AKT,NAM=
1          LA    1,LMARK                  R1=A(PARAMS)
1          CNOP  2,4
1          LM    14,15,*+6
1          BR    15                       CALL LEASY
1          DC    A(*+8)
1          DC    V(LEASY)
1          L     1,4(1)                   R1=A(RE)

  LMARK    LEA@MARK L@@RE,MF=L
1 LMARK    LEA@CALL 'MARK',L@@RE,                                    C
1                MF=L,SAVE=,PRE=L,                                   C
                 PIDE=,TIDE=,                                        C
1                ERRCODE=,ERRADDR=
2          LEA@@BP GRU=GEN,NAM=LMARK,TYP=E,PRE=L
2          LEA@@AL 'MARK',L@@RE,,,,,,,PRE=L
3 LMARK    DS    0F
3          DC    A(*+32+X'00000000')
3          DC    A(L@@RE+X'80000000')
3          DC    A(X'00000000')
3          DC    A(X'00000000')
3          DC    A(X'00000000')
3          DC    A(X'00000000')
3          DC    A(X'00000000')
3          DC    A(X'00000000')
3          DC    CL4'MARK'
```

LEA@OPFL

Open files

This macro opens the files specified in the file list according to the relevant OPEN mode. See “OPFL Open files” on page 161.

The *OPFL* operation is not allowed if files or transactions are open for that task.

Operands *MF* and *PRE* are permitted.

Operation	Operands
LEA@OPFL	[[re],[db],[us]] [,SAVE=address2] [,POPEOM=openmode][,TOPEOM=openmode] [,ERRCODE=$\left\{\begin{array}{l}\text{(error-code,...)}\\ \text{address3}\end{array}\right\}$,ERRADDR=address4] [,ERRADDR=address4]

re	*symb* or (*r*) Address of the reference area.	
db	*symb* or (*r*) Address of the file list.	
	’string’	Logical file name.
us	*symb* or (*r*) Address of the *USER* area.	
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the *RE* area.	
POPEOM TOPEOM	Open mode of files/transactions (*P*=permanent, *T*=temporary).	
=openmode	Permitted OPEN mode; (see page 184). The following values are permitted: *1, 2, 3, 4, 5, 7, 8, 9, A, B, X’FF’*.	
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.	
=address3	Symbolic address for error handling.	

ERRADDR=address4

Address (*symb* or (*r*));
branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LA    R5,L@@RE
           LA    R6,ERROR
  *
           LEA@OPFL (R5),'(FILE1,FILE2)',                              -
                 TOPEOM=1,                                             -
                 ERRCODE=ERROUT,                                       -
                 ERRADDR=(R6)
1          LEA@CALL 'OPFL',(R5),'(FILE1,FILE2)',                       C
1                MF=,SAVE=,PRE=L,                                      C
1                POPEOM=,TOPEOM=1,                                     C
1                ERRCODE=ERROUT,ERRADDR=(R6)
2          LEA@@BP GRU=AKT,NAM=
2          LR    14,R5               R14=A(RE)
2          MVC   *+14(L'$@@RE),0(14)         TEMP. RE
2          LA    14,*+8
2          B     *+4+L'$@@RE
2          DS    CL(L'$@@RE)                 TEMP. RE
2          LEA@@KO OPCD=OPFL,KORR=TEMP,                                C
2                OPE1=,OPE2=,                                          C
2                OPEOM=1,OPELOG=,OPEWTM=,                              C
2                SAMPTR=,PAMHP=,IDE=
3          MVI   $@@OPOM-$@@RE(14),C'1'      MODIFY OPEOM
2          CNOP  2,4
2          ST    14,*+30                     A(RE)
2          OI    *+30,X'80'                  LAST PARAM-ADDRESS
2          LM    14,15,*+10
2          LA    1,*+14
2          BR    15                          CALL LEASY
2          DC    A(*+40)
2          DC    V(LEASY)
2 *
2          DC    A(*+12)                     PTR(OP)
2          DC    A(0)                        PTR(RE)
2          DC    A(*+8)                      PTR(DB)
2          DC    C'OPFL'
2          DC    C'(FILE1,FILE2)'
2          L     1,4(1)                      R1=A(RE)
2          LEA@@FB ERRADDR=(R6),ERRCODE=ERROUT,R14=ON
3          LA    14,*+16+2+((L'$@@RCCC+1)/2)*2 R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+12+2 TEST RC
3          BRE   14                      RC OK
3          LR    15,R6          R15=A(ERROR-ROUTINE)
3          B     ERROUT
3          DC    CL(L'$@@RCCC)'000'
```

LEA@OPTR

Open transaction

Macro *LEA@OPTR* opens a transaction. See “OPTR Open or extend transaction” on page 162.

Operands *MF* and *PRE* are permitted.

Operation	Operands
LEA@OPTR	

```
            ⎧db⎫
[[re],[⎨  ⎬],[us]]
            ⎩ci⎭

[,SAVE=address2]

        ⎧W    ⎫             ⎧W    ⎫
[,POPE1=⎨     ⎬]    [,TOPE1=⎨     ⎬]
        ⎩BLANK⎭             ⎩BLANK⎭

[,POPEOM=openmode]  [,TOPEOM=openmode]

          ⎧N    ⎫             ⎧N    ⎫
[,POPELOG=⎨     ⎬]  [,TOPELOG=⎨     ⎬]
          ⎩BLANK⎭             ⎩BLANK⎭

[,PIDE=ide]         [,TIDE=ide]

          ⎧(error-code,...)⎫
[,ERRCODE=⎨                ⎬ ,ERRADDR=address4]
          ⎩address3        ⎭

[,ERRADDR=address4]
```

re	*symb* or (*r*) Address of the reference area.	
db	*symb* or (*r*) Address of the file identifier list.	
	’*string*’	File identifier or file identifier list.
ci	*symb* or (*r*) Address of the currency information.	

Function a (operand db)

A transaction is opened if none is as yet active. All file identifiers specified in the file list (db), together with their USAGE modes, are opened logically for the transaction.

All file pointers are positioned to the start of the file and the primary key.

If the user already has a transaction open at the time of the macro call, this transaction is extended to include the file identifiers that are specified in the file list. Positioning to the start of the file and the primary key is effected.

Function b (operand ci and OPE1=W)

A transaction is opened. The files in the currency information are opened. Positioning is effected to those positions stored in the currency information.

us	*symb* or (*r*) Address of the *USER* area.
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
POPE1=ext1 TOPE1=ext1	Operation code extension *OPE1* (*P*=permanent, *T*=temporary).
=W	Transaction start with simultaneous file positioning.
=BLANK	Normal transaction start.
POPEOM TOPEOM	Open mode of files or transactions (*P*=permanent, *T*=temporary).
	The following values are possible: *A, B, E, G, I, L, O, Q, R, U, X, BLANK* and *X'FF'*.
POPELOG TOPELOG	Activation/deactivation of BIM saving (*P*=permanent, *T*=temporary).
=N	BIM saving for the current transaction is deactivated.
=BLANK	BIM saving is activated.

PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing binary zeros or blanks.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LA    R4,ADDRLIST
           LA    R5,T@@RE
  *
           LEA@OPTR MF=(E,(R4)),                                         -
                 SAVE=(R5),                                              -
                 TOPE1=W,                                                -
                 ERRADDR=ERROR
1          LEA@CALL 'OPTR',,,                                            C
1                MF=(E,(R4)),SAVE=(R5),PRE=L,                            C
1                POPE1=,POPELOG=,POPEOM=,PIDE=,                          C
1                TOPE1=W,TOPELOG=,TOPEOM=,TIDE=                          C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LR    1,R4
2          L     14,4(1)                 R14=A(RE)
2          MVC   0(L'$@@RE,R5),0(14)     SAVE=TEMP. RE
2          LR    14,R5
2          LEA@@KO OPCD=OPTR,KORR=TEMP,                                  C
2                OPE1=W,OPE2=,                                           C
2                OPEOM=,OPELOG=,OPEWTM=,                                 C
2                SAMPTR=,PAMHP=,IDE=
3          MVI   $@@OPE1-$@@RE(14),C'W'  MODIFY OPE1
2          CNOP  2,4
2          LA    15,*+50
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,0(1)
2          TM    4(1),X'80'
2          BZ    *+8
2          O     14,=A(X'80000000')
2          ST    14,4(1)                 MODIFY RE
2          LM    14,15,*+6
2          BR    15                      CALL LEASY
2          DC    A(*+12)
2          DC    V(LEASY)
2 *
2          DC    C'OPTR'
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+12+4+((L'$@@RCCC+1)/2)*2 R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4 TEST RC
3          BRE   14                      RC OK
3          B     ERROR                RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@PARC

Correct operand list

This macro prepares a reference area (*re* or address in *ADDRLST*) or an operand list (*ADDRLST*) for a LEASY call.

Operation	Operands

LEA@PARC	`[[op],[re],[db],[ar],[fa],[si],[kb],[ke]]`
	`[,ADDRLST=address1]`
	`[,LASTPAR=value]`
	`[,POPE1=ext1]`
	`[,POPE2=ext2]`
	`[,POPSTX=stxitmode]`
	`[,POPEOM=openmode]`
	`[,POPELOG=log]`
	[,POPEWTM=$\left\{\begin{array}{l}\text{waiting-time}\\ \text{BLANK}\end{array}\right\}$]
	`[,PSAMPTR=X'sam-pointer']`
	`[,PPAMHP=X'pam-block-number']`
	`[,PIDE=ide]`

The specified positional operands (*op*, *re* etc.) correct the operand list (*ADDRLST*). Operands *POPE1*, *POPE2* etc. correct the *RE* area whose address was specified in the macro (*re*) or is indicated in the operand list (2nd word in *ADDRLST*).

If *ADDRLST* and *LASTPAR* are present, the identifier for the last operand is set in *ADDRLST* at the operand word designated by *value*. If, for example, *LASTPAR=4*, the left-most byte of the address of AR (4th operand word) is set to *X'80'*. A number greater than 8 causes all left-most address bytes to be deleted.

op	*symb* or (*r*) Address of the operation code area.	
	'string'	Name of the operation.
re	*symb* or (*r*) Address of the reference area.	

db	*symb* or (*r*) Address of the file list.
	'*string*' Logical file name or file identifier or file list.
ar	*symb* or (*r*) Address of the I/O area (record zone).
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).
	'*string*' Identifier for the field selection.
si	*symb* or (*r*) Address of the *SI* area (Secondary Index).
	'*string*' Name of a secondary index.
kb	*symb* or (*r*) Address of the *KB* area (Key Begin).
ke	*symb* or (*r*) Address of the *KB* area (Key End).
ADDRLST=address1	Address (*symb* or (*r*)) of the operand list, as expected for the action macro in operand *MF=(E, address1)*.
LASTPAR=value	1 ≤ *value* ≤ 8
	this designates the word at the position *value* in the operand list. The left-most byte of this address is the identifier *X'80'* (last operand).
POPE1=ext1	Operation code extension *OPE1*.
=W	Transaction start with simultaneous file positioning; only permitted with *OPTR*.
=R	Transaction rollback; only permitted with *CLTR*.
=S	Read lock with the *LOCK, RHLD, RNHD, RPHD* operations.
=F	Currency information on the files in the LEASY catalog and their secondary indices; only permitted with *CINF*.
=U	Unlock modified records; permitted only with UNLK.
=BLANK	Normal transaction start or normal end of transaction or write lock.
	Currency information on all the file identifiers opened in the transaction (with *CINF*).
POPE2=ext2	Operation code extension *OPE2*.
=T	Transaction termination and simultaneous transaction start with *CLTR*. Currency information on all the files involved in the transaction (with *CINF*).
=N	The number of primary keys for a secondary key value is determined; only permitted with *RDIR* and *RHLD*.

=C	Currency information on all the files in the LEASY catalog; only permitted with *CINF*.
=O	Currency information on all the files opened with *OPFL*; only permitted with *CINF*.
=S	Currency information on the file specified in *CI*; only permitted with *CINF*.
=W	The help function immediately preceding this is to be continued; only permitted with *CINF*.
=L	Locked records are to be skipped; only permitted with *RPHD* and *RNHD*.
=BLANK	Transaction termination with cancellation of all file access requests (with *CLTR*). Currency information on all the files in the LEASY catalog (with *CINF*). The number of primary keys is not determined (with *RDIR* and *RHLD*).
POPSTX=stxitmode	This operand is still provided for compatibility reasons, but is no longer evaluated.
=P	ignored
=N	ignored
=BLANK	LEASY-STXIT routine.
POPEOM	OPEN mode.
=openmode	Permitted OPEN mode or USAGE mode; (see section “Opening files and transactions” on page 184). The following values are permitted: *1, 2, 3, 4, 5, 7, 8, 9, A, B, E, G, I, L, O, Q, R, U, X, BLANK* and *X’FF’*.
POPELOG=log	Activation/deactivation of BIM saving; only permitted with *OPTR*.
=N	BIM saving deactivated for current transaction.
=BLANK	BIM saving is activated.
POPEWTM	Waiting time for locked records.
=waiting-time	0 ≤ *waiting-time* ≤ 999.
=BLANK	The global waiting time for the session applies (*TIME* statement in LEASY-MAINTASK).

PSAMPTR	SAM retrieval address in 24-bit or 31-bit format.
=X’bbbbbbrr’	24-bit format.
=X’bbbbbbbbrrrrrrrr’	31-bit format.
	(*b* = block number, *r* = record number within the block)
PPAMHP	PAM block number.
=X’bbbbbb’	Number of the PAM block.
PIDE=ide	Identifier for DCAM.
=symb	The symbolic address indicates an 8-byte field containing a DCAM application or a transactions identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a DCAM application or a transaction identifier.

Example

```
           LEA@PARC ,L@@RE,POPEWTM=5,PPAMHP=X'2'
1          LEA@@BP GRU=AKT,NAM=
1          LA    14,L@@RE                  R14=A(RE)
1          LEA@@KO                                                         C
1                OPE1=,OPE2=,                                              C
1                OPEOM=,OPELOG=,OPEWTM=5,                                  C
1                SAMPTR=,PAMHP=X'2',IDE=,                                  C
1                OPSTX=
2          MVC   $@@OPWTM-$@@RE(L'$@@OPWTM,14),*+10 MODIFY OPEWTIM
2          B     *+4+((L'$@@OPWTM+1)/2)*2
2          DC    ZL(L'$@@OPWTM)'5'
2          MVC   $@@PAMHP-$@@RE(L'$@@PAMHP,14),*+10 MODIFY PAMHP
2          B     *+4+L'$@@PAMHP
2          DC    XL(L'$@@PAMHP)'2'
  PARC     LEA@PARC ,L@@RE,POPEWTM=0,ADDRLST=ADDRLIST,LASTPAR=2
1          LEA@@BP GRU=AKT,NAM=PARC
1 PARC     LA    1,ADDRLIST
1          LA    14,L@@RE                  R14=A(RE)
1          LEA@@KO                                                         C
1                OPE1=,OPE2=,                                              C
1                OPEOM=,OPELOG=,OPEWTM=0,                                  C
1                SAMPTR=,PAMHP=,IDE=,                                      C
1                OPSTX=
2          MVC   $@@OPWTM-$@@RE(L'$@@OPWTM,14),*+10 MODIFY OPEWTIM
2          B     *+4+((L'$@@OPWTM+1)/2)*2
2          DC    ZL(L'$@@OPWTM)'0'
1          LA    15,L@@RE
1          TM    4(1),X'80'
1          BZ    *+8
1          O     15,=A(X'80000000')
1          ST    15,4(1)                MODIFY RE
1 *
1          LEA@@LP LASTPAR=2
2          PRINT OFF
2          NI    ($@@POP-$@@POP)(1),X'FF'-$@@LAST
2          OI    ($@@PRE-$@@POP)(1),$@@LAST
```

LEA@RDIR

Directly read record

This macro reads a record or block directly into the input area *AR* as follows:

- A record in an ISAM or DAM file is read via a specified key
- A record in a SAM file is read via the specified retrieval address
- A logical block in a PAM file is read via a specified block number.

In addition to reading, the pointer is positioned to the key found and the index used (primary or secondary index).

Operands *MF* and *PRE* are permitted.

Operation	Operands
LEA@RDIR	[[re],[db],[ar],[fa],[si],[kb],[ke]] [,SAVE=address2] [,POPE2=ext2] [,TOPE2=ext2] [,PPAMHP=X'pam-block-number'] [,TPAMHP=X'pam-block-number'] [,PSAMPTR=X'sam-pointer'] [,TSAMPTR=X'sam-pointer'] [,PIDE=ide] [,TIDE=ide] [,ERRCODE=$\left\{\begin{array}{l}\texttt{(error-code,...)}\\ \texttt{address3}\end{array}\right\}$,ERRADDR=address4] [,ERRADDR=address4]

re	*symb* or (*r*) Address of the reference area.	
db	*symb* or (*r*) Address of the file identifier.	
	’string’	File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).	
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).	
	’string’	Identifier for the field selection.
si	*symb* oder (*r*) Address of the *SI* area (Secondary Index).	
	’string’	Name of a secondary index.

kb	*symb* or (*r*) Address of the *KB* area (Key Begin).
ke	*symb* or (*r*) Address of the *KE* area (Key End).
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
POPE2=ext2 TOPE2=ext2	Operation code extension *OPE2* (*P*=permanent, *T*=temporary).
=N	The number of primary keys for a secondary key value is determined.
=BLANK	The number of primary keys is not determined.
PPAMHP TPAMHP	PAM block number (*P*=permanent, *T*=temporary).
=X'bbbbbb'	Number of the block to be read.
PSAMPTR TSAMPTR	Retrieval address of a record in a SAM file in 24-bit or 31-bit format (*P*=permanent, *T*=temporary).
=X'bbbbbbrr'	24-bit format.
=X'bbbbbbbbrrrrrrr'	31-bit format (*b* = block number, *r* = record number within the block)

ℹ For ISAM and DAM files, the key value must be transferred in the *AR* area. See page 145 and 166ff for use of operands *KB* and *KE*.

PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.

ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LA    R4,ADDRLIST
           LA    R5,ERROR
           LA    R6,PAMDB
  *
           LEA@RDIR ,(R6),MF=(E,(R4)),                                   -
                 SAVE=T@@RE,                                             -
                 TPAMHP=X'ABC',                                          -
                 ERRADDR=ERROR
1          LEA@CALL 'RDIR',,(R6),,,,,,                                   C
1                MF=(E,(R4)),SAVE=T@@RE,PRE=L,                           C
1                PPAMHP=,PSAMPTR=,PIDE=,                                 C
1                POPE2=                                                  C
1                TPAMHP=X'ABC',TSAMPTR=,TIDE=,                           C
1                TOPE2=,                                                 C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LR    1,R4
2          L     14,4(1)                 R14=A(RE)
2          MVC   T@@RE(L'$@@RE),0(14)       SAVE=TEMP. RE
2          LA    14,T@@RE
2          LEA@@KO OPCD=RDIR,KORR=TEMP,                                  C
2                OPE1=,OPE2=,                                            C
2                OPEOM=,OPELOG=,OPEWTM=,                                 C
2                SAMPTR=,PAMHP=X'ABC',IDE=
3          MVC   $@@PAMHP-$@@RE(L'$@@PAMHP,14),*+10 MODIFY PAMHP
3          B     *+4+L'$@@PAMHP
3          DC    XL(L'$@@PAMHP)'ABC'
2          CNOP  2,4
2          LA    15,*+30
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,0(1)
2          TM    4(1),X'80'
2          BZ    *+8
2          O     14,=A(X'80000000')
2          ST    14,4(1)                 MODIFY RE
2          LM    14,15,*+6
2          BR    15                      CALL LEASY
2          DC    A(*+12)
2          DC    V(LEASY)
2 *
2          DC    C'RDIR'
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+12+2+((L'$@@RCCC+1)/2)*2 R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4 TEST RC
3          BRE   14                      RC OK
3          B     ERROR                   RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@REWR

Rewrite record

This macro updates (rewrites) an existing record or block. If a file is governed by a lock log, the record or block must already have been locked. See “REWR Rewrite record” on page 172.

Operands *MF* and *PRE* are permitted.

Operation	Operands

`LEA@REWR`	`[[re],[db],[ar],[us]]`
	`[,SAVE=address2]`
	`[,PPAMHP=X'pam-block-number'][,TPAMHP=X'pam-block-number']`
	`[,PIDE=ide]` `[,TIDE=ide]`
	`[,ERRCODE=` $\left\{\begin{array}{l}\texttt{(error-code,...)}\\ \texttt{address3}\end{array}\right\}$ `,ERRADDR=address4]`
	`[,ERRADDR=address4]`

re	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file identifier.
	’string’ File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).
us	*symb* or (*r*) Address of the *USER* area
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
PPAMHP TPAMHP	PAM block number (*P*=permanent, *T*=temporary).
=X’bbbbbb’	Number of the block to be read.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.

=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LA    R4,ERROR           R4=A(FEHLERROUTINE)
  *
           LEA@REWR MF=(E,ADDRLIST),                                     -
                 PPAMHP=X'2',                                            -
                 ERRADDR=(R4)
1          LEA@CALL 'REWR',,,,                                           C
1                MF=(E,ADDRLIST),SAVE=,PRE=L,                            C
1                PPAMHP=X'2',PIDE=                                       C
1                TRAMHP=,TIDE=,                                          C
1                ERRCODE=,ERRADDR=(R4)
2          LEA@@BP GRU=AKT,NAM=
2          LA    1,ADDRLIST
2          L     14,4(1)                 R14=A(RE)
2          LEA@@KO OPCD=REWR,KORR=PERM,                                  C
2                OPE1=,OPE2=,                                            C
2                OPEOM=,OPELOG=,OPEWTM=,                                 C
2                SAMPTR=,PAMHP=X'2'
3          MVC   $@@PAMHP-$@@RE(L'$@@PAMHP,14),*+10 MODIFY PAMHP
3          B     *+4+L'$@@PAMHP
3          DC    XL(L'$@@PAMHP)'2'
2          CNOP  2,4
2          LA    15,*+22
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000'9
2          ST    15,0(1)
2          LM    14,15,*+6
2          BR    15                      CALL LEASY
2          DC    A(*+12)
2          DC    V(LEASY)
2 *
2          DC    C'REWR'
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=(R4),ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+12+2+((L'$@@RCCC+1)/2)*2 R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+2 TEST RC
3          BRE   14                      RC OK
3          BR    R4                      RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@RHLD

Read and lock record

This macro reads a record or block and locks it.

Accessing with ISAM and DAM files is via the specified key, whilst accessing with PAM files is via the block number.

Operands *MF* and *PRE* are permitted.

Operation	Operands
LEA@RHLD	

```
[[re],[db],[ar],[fa],[si],[kb],[ke]]

[,SAVE=address2]

[,POPE1=ext1]                 [,TOPE1=ext1]

[,POPE2=ext2]                 [,TOPE2=ext2]

          ⎧waiting-time⎫                ⎧waiting-time⎫
[,POPEWTM=⎨            ⎬]       [,TOPEWTM=⎨            ⎬]
          ⎩BLANK       ⎭                ⎩BLANK       ⎭

[,PPAMHP=X'pam-block-number'][,TPAMHP=X'pam-block-number']

[,PIDE=ide]                 [,TIDE=ide]

          ⎧(error-code,...)⎫
[,ERRCODE=⎨                ⎬ ,ERRADDR=address4]
          ⎩address3        ⎭

[,ERRADDR=address4]
```

re	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file identifier.
	'*string*' File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).
	'*string*' File identifier.
si	*symb* or (*r*) Address of the *SI* area (Secondary Index).
	'*string*' Name of a secondary index.
kb	*symb* or (*r*) Address of the *KB* area (Key Begin).

ke	*symb* or (*r*) Address of the *KE* area (Key End).
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
POPE1=ext1 TOPE1=ext1	Operation code extension *OPE1* (*P*=permanent, *T*=temporary).
=S	A READ-LOCK is enforced.
=BLANK	A WRITE-LOCK is enforced.
POPE2=ext2 TOPE2=ext2	Operation code extension *OPE2* (*P*=permanent, *T*=temporary).
=N	The number of primary keys for a secondary key value is determined.
=BLANK	The number of primary keys is not determined..
POPEWTM TOPEWTM	Waiting time for locked records (*P*=permanent, *T*=temporary).
=waiting-time	0 ≤ *waiting-time* ≤ 999.
=BLANK	The global waiting time for the session applies (*TIME* statement in LEASY-MAINTASK).
PPAMHP TPAMHP	PAM block number (*P*=permanent, *T*=temporary).
=X’bbbbbb’	Number of the block to be read.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.

=(r) The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.

ERRCODE=(error-code,...)
List of tolerable error codes; 1 to 8 characters.

=address3 Symbolic address for error handling.

ERRADDR=address4
Address (*symb* or (*r*));
branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@RHLD PAMRE,PAMDB,MF=(E,ADDRLST),                          -
                 POPEWTM=22,                                             -
                 TPAMHP=X'1',                                            -
                 ERRADDR=ERROR
1          LEA@CALL 'RHLD',PAMRE,PAMDB,,,,,,                             C
1                MF=(E,ADDRLIST),SAVE=,PRE=L,                            C
1                PPAMHP=,POPEWTM=22,PIDE=,                               C
1                POPE1=,POPE2=,                                          C
1                TPAMHP=X'1',TOPEWTM=,TIDE=,                             C
1                TOPE1=,TOPE2=,                                          C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LA    1,ADDRLIST
2          LA    14,PAMRE                  R14=A(RE)
2          LEA@@KO OPCD=RHLD,KORR=PERM,                                  C
2                OPE1=,OPE2=,                                            C
2                OPEOM=,OPELOG=,OPEWTM=22,                               C
2                SAMPTR=,PAMHP=,IDE=
3          MVC   $@@OPWTM-$@@RE(L'$@@OPWTM,14),*+10 MODIFY OPEWTIM
3          B     *+4+((L'$@@OPWTM+1)/2)*2
3          DC    ZL(L'$@@OPWTM)'22'
2          MVC   *+14(L'$@@RE),0(14)        TEMP. RE
2          LA    14,*+8
2          B     *+4+L'$@@RE
2          DS    CL(L'$@@RE)                 TEMP. RE
2          LEA@@KO OPCD=RHLD,KORR=TEMP,                                  C
2                OPE1=,OPE2=,                                            C
2                OPEOM=,OPELOG=,OPEWTM=,                                 C
2                SAMPTR=,PAMHP=X'1',IDE=
3          MVC   $@@PAMHP-$@@RE(L'$@@PAMHP,14),*+10 MODIFY PAMHP
3          B     *+4+L'$@@PAMHP
3          DC    XL(L'$@@PAMHP)'1'
2          CNOP  2,4
2          LA    15,*+70
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,0(1)
2          TM    4(1),X'80'
2          BZ    *+8
2          O     14,=A(X'80000000')
2          ST    14,4(1)                            MODIFY RE
```

```
2          LA    15,PAMDB
2          TM    8(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,8(1)                                MODIFY DB
2          LM    14,15,*+6
2          BR    15                    CALL LEASY
2          DC    A(*+12)
2          DC    V(LEASY)
2 *
2          DC    C'RHLD'
2          L     1,4(1)                R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                  R15=4X'FF'
3          LA    14,*+12+4+((L'$@@RCCC+1)/2)*2 R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4 TEST RC
3          BRE   14                    RC OK
3          B     ERROR              RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@RNHD

Read and lock next record

This macro reads sequentially the next record or block in the file specified, either towards the end of the file in the case of SAM files, or in ascending order of the primary or secondary keys for ISAM, DAM or PAM files. If the operation is performed successfully, the record or block is locked. See page 173ff.

Operands *MF* and *PRE* are permitted.

Operation	Operands
LEA@RNHD	

```
[[re],[db],[ar],[fa]]

[,SAVE=address2]

[,POPE1=ext1]          [,TOPE1=ext1]

[,POPE2=ext2]          [,TOPE2=ext2]

          ⎧waiting-time⎫                ⎧waiting-time⎫
[,POPEWTM=⎨            ⎬]     [,TOPEWTM=⎨            ⎬]
          ⎩BLANK       ⎭                ⎩BLANK       ⎭

[,PIDE=ide]            [,TIDE=ide]

          ⎧(error-code,...)⎫
[,ERRCODE=⎨                ⎬ ,ERRADDR=address4]
          ⎩address3        ⎭

[,ERRADDR=address4]
```

e	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file identifier.
	’*string*’ File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).
SAVE=adresse2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
POPE1=ext1 TOPE1=ext1	Operation code extension *OPE1* (*P*=permanent, *T*=temporary).

=S	A READ-LOCK is enforced.
=BLANK	A READ-LOCK is enforced.
POPE2=ext2 TOPE2=ext2	Operation code extension *OPE2* (*P*=permanent, *T*=temporary).
=L	Locked records are skipped.
=BLANK	Locked records are not skipped.
POPEWTM TOPEWTM	Waiting time for locked records (*P*=permanent, *T*=temporary).
=waiting-time	0 ≤ *waiting-time* ≤ 999.
=BLANK	The global waiting time for the session applies (*TIME* statement in LEASY-MAINTASK).
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@RNHD MF=(E,ADDRLIST),                                      -
                 TOPEWTM=12,                                              -
                 ERRADDR=ERROR,                                           -
                 ERRCODE=(L003)
1          LEA@CALL 'RNHD',,,,,,,,                                        C
1                MF=(E,ADDRLIST),SAVE=,PRE=L,                             C
1                POPEWTM=,PIDE=,POPE1=,POPE2=,                            C
1                TOPEWTM=12,TIDE=,TOPE1=,TOPE2=,                          C
1                ERRCODE=(L003),ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LA    1,ADDRLIST
2          L     14,4(1)                 R14=A(RE)
2          MVC   *+14(L'$@@RE),0(14)        TEMP. RE
2          LA    14,*+8
2          B     *+4+L'$@@RE
2          DS    CL(L'$@@RE)                 TEMP. RE
2          LEA@@KO OPCD=RNHD,KORR=TEMP,                                   C
2                OPE1=,OPE2=,                                             C
2                OPEOM=,OPELOG=,OPEWTM=12,                                C
2                SAMPTR=,PAMHP=,IDE=
3          MVC   $@@OPWTM-$@@RE(L'$@@OPWTM,14),*+10     MODIFY OPEWTIM
3          B     *+4+((L'$@@OPWTM+1)/2)*2
3          DC    ZL(L'$@@OPWTM)'12'
2          CNOP  2,4
2          LA    15,*+50
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,0(1)
2          TM    4(1),X'80'
2          BZ    *+8
2          O     14,=A(X'80000000')
2          ST    14,4(1)                 MODIFY RE
2          LM    14,15,*+6
2          BR    15                      CALL LEASY
2          DC    A(*+12)
2          DC    V(LEASY)
2 *
2          DC    C'RNHD'
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=(L003),R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+16+4+16+((L'$@@RCCC+1)/2)*2      R14=RETURN ADDRESS
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+12      TEST RC
3          BRE   14                      RC OK
3          B     *+4+((L'$@@RCCC+1)/2)*2 RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
3          CLC   $@@RCLC-$@@RE(4,1),*+12      FEASIBLE RETURN CODE?
3          BRE   14                      YES, FEASIBLE
3          B     *+4+4
3          DC    C'L003'
3          B     ERROR                   RC: ERROR
```

LEA@RNXT

Read next record

This macro causes the next record or block in the file specified to be read sequentially, either towards the end of the file for SAM files, or in ascending order of the primary or secondary key in the case of ISAM, DAM or PAM files. See page 173ff.

Operands *MF* and *PRE* are permitted.

Operation	Operands

`LEA@RNXT`	`[[re],[db],[ar],[fa]]`
	`[,SAVE=address2]`
	`[,PIDE=ide] [,TIDE=ide]`
	$\texttt{[,ERRCODE=}\left\{\begin{matrix}\texttt{(error-code,...)}\\ \texttt{address3}\end{matrix}\right\}\texttt{,ERRADDR=address4]}$
	`[,ERRADDR=address4]`

re	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file identifier.
	'*string*' File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.

ERRADDR=address4

Address (*symb* or (*r*));
branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@RNXT MF=(E,ADDRLIST),                                     -
                 ERRADDR=ERROR
1          LEA@CALL 'RNXT',,,,,,,,                                       C
1                MF=(E,ADDRLIST),SAVE=,PRE=L,                            C
1                PIDE=,TIDE=,                                            C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LA    1,ADDRLIST
2          CNOP  2,4
2          LA    15,*+34
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,0(1)
2          LM    14,15,*+6
2          BR    15                      CALL LEASY
2          DC    A(*+12)
2          DC    V(LEASY)
2 *
2          DC    C'RNXT'
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+12+4+((L'$@@RCCC+1)/2)*2    R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4         TEST RC
3          BRE   14                      RC OK
3          B     ERROR                   RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@RPHD

Read and lock previous record

This macro causes the next record or block in the file specified to be read, either towards the beginning of the file in the case of SAM files, or in descending order of the primary or secondary key for ISAM, DAM and PAM files. See page 175ff.

If the operation is performed successfully, the record or block is locked.

Operands *MF* and *PRE* are permitted.

Operation	Operands
LEA@RPHD	[[re],[db],[ar],[fa]] [,SAVE=address2] [,POPE1=ext1] [,TOPE1=ext1] [,POPE2=ext2] [,TOPE2=ext2] [,POPEWTM= $\left\{\begin{array}{l}\text{waiting-time}\\ \text{BLANK}\end{array}\right\}$] [,TOPEWTM= $\left\{\begin{array}{l}\text{waiting-time}\\ \text{BLANK}\end{array}\right\}$] [,PIDE=ide] [,TIDE=ide] [,ERRCODE= $\left\{\begin{array}{l}\text{(error-code,...)}\\ \text{address3}\end{array}\right\}$,ERRADDR=address4] [,ERRADDR=address4]

re	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file identifier.
	’*string*’ File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the *RE* area.
POPE1=ext1	Operation code extension *OPE1* (*P*=permanent, *T*=temporary).
TOPE1=ext1	

=S	A READ-LOCK is enforced.
=BLANK	A WRITE-LOCK is enforced.
POPE2=ext2 TOPE2=ext2	Operation code extension *OPE2* (*P*=permanent, *T*=temporary).
=L	Locked records are skipped.
=BLANK	Locked records are not skipped.
POPEWTM TOPEWTM	Waiting time for locked records (*P*=permanent, *T*=temporary).
=waiting-time	0 ≤ *waiting-time* ≤ 999.
=BLANK	The global waiting time for the session applies (*TIME* statement in LEASY-MAINTASK).
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@RPHD MF=(E,ADDRLIST),                                     -
                 POPEWTM=3,                                              -
                 ERRADDR=ERROR
1          LEA@CALL 'RPHD',,,,,,,,                                       C
1                MF=(E,ADDRLIST),SAVE=,PRE=L,                            C
1                POPEWTM=3,PIDE=,POPE1=,POPE2=,                          C
1                TOPEWTM=,TIDE=,TOPE1=,TOPE2=,                           C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LA    1,ADDRLIST
2          L     14,4(1)                 R14=A(RE)
2          LEA@@KO OPCD=RPHD,KORR=PERM,                                  C
2                OPE1=,OPE2=,                                            C
2                OPEOM=,OPELOG=,OPEWTM=3,                                C
2                SAMPTR=,PAMHP=,IDE=
3          MVC   $@@OPWTM-$@@RE(L'$@@OPWTM,14),*+10     MODIFY OPEWTIM
3          B     *+4+((L'$@@OPWTM+1)/2)*2
3          DC    ZL(L'$@@OPWTM)'3'
2          CNOP  2,4
2          LA    15,*+34
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,0(1)
2          LM    14,15,*+6
2          BR    15                      CALL LEASY
2          DC    A(*+12)
2          DC    V(LEASY)
2 *
2          DC    C'RPHD'
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+12+4+((L'$@@RCCC+1)/2)*2    R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4         TEST RC
3          BRE   14                      RC OK
3          B     ERROR                   RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@RPRI

Read previous record

This macro causes the next record in the file specified to be read sequentially, either towards the start of the file for SAM files, or in descending order of the primary or secondary key for ISAM, DAM or PAM files. See page 175.

Operands *MF* and *PRE* are permitted.

```
Operation    Operands

LEA@RPRI     [[re],[db],[ar],[fa]]

             [,SAVE=address2]

             [,PIDE=ide][,TIDE=ide]

                       ⎧(error-code,...)⎫
             [,ERRCODE=⎨                ⎬ ,ERRADDR=address4]
                       ⎩address3        ⎭

             [,ERRADDR=address4]
```

re	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file identifier.
	’*string*’ File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the *RE* area.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.

ERRADDR=address4

Address (*symb* or (*r*));
branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
          LEA@RPRI MF=(E,ADDRLIST)
1         LEA@CALL 'RPRI',,,,,,,,,                                     C
1               MF=(E,ADDRLIST),SAVE=,PRE=L,                           C
1               PIDE=,TIDE=,                                           C
1               ERRCODE=,ERRADDR=
2         LEA@@BP GRU=AKT,NAM=
2         LA    1,ADDRLIST
2         CNOP  2,4
2         LA    15,*+34
2         TM    0(1),X'80'
2         BZ    *+8
2         O     15,=A(X'80000000')
2         ST    15,0(1)
2         LM    14,15,*+6
2         BR    15                      CALL LEASY
2         DC    A(*+12)
2         DC    V(LEASY)
2 *
2         DC    C'RPRI'
2         L     1,4(1)                  R1=A(RE)
```

LEA@SETL

Position file pointer

This macro positions an internal file pointer to a specified record or block. In addition, the index specified for the operand *si* and, if 8 operands are specified, a key range are set for subsequent *LEA@RNXT*/*LEA@RPRI* operations. See “SETL Position file pointer” on page 177.

Operands *MF* and *PRE* are permitted.

```
Operation   Operands
LEA@SETL    [[re],[db],[ar],[fa],[si],[kb],[ke]]

            [,SAVE=address2]

            [,PPAMHP=X'pam-block-number'] [,TPAMHP=X'pam-block-number']

            [,PSAMPTR=X'sam-pointer']     [,TSAMPTR=X'sam-pointer']

            [,PIDE=ide]                   [,TIDE=ide]

                       ⎧(error-code,...)⎫
            [,ERRCODE= ⎨                ⎬ ,ERRADDR=address4]
                       ⎩address3        ⎭

            [,ERRADDR=address4]
```

re	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file identifier.
	’string’ File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).
	’string’ Identifier for the field selection.
si	*symb* or (*r*) Address of the *SI* area (Secondary Index).
	’string’ Name of a secondary index.
kb	*symb* or (*r*) Address of the *KB* area (Key Begin).
ke	*symb* or (*r*) Address of the *KE* area (Key Begin).

SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.
PPAMHP TPAMHP	PAM block number (*P*=permanent, *T*=temporary).
=X'bbbbbb'	Number of the block to be read.
PSAMPTR TSAMPTR	Retrieval address of a record in a SAM file in 24-bit or 31-bit format (*P*=permanent, *T*=temporary).
=X'bbbbbbrr'	24-bit format.
=X'bbbbbbbbrrrrrrrr'	31-bit format. (*b* = block number, *r* = record number within the block)
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LA    R4,ADDRLIST
  *
           LEA@SETL ,,,,'MAINITEM',MF=(E,(R4)),                          -
                 TSAMPTR=X'1001',                                        -
                 ERRADDR=ERROR
1          LEA@CALL 'SETL',,,,,'MAINITEM',,,                             C
1                MF=(E,(R4)),SAVE=,PRE=L,                                C
1                PPAMHP=,PSAMPTR=,PIDE=,                                 C
1                TPAMHP=,TSAMPTR=X'1001',TIDE=,                          C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LR    1,R4
2          L     14,4(1)                 R14=A(RE)
2          MVC   *+14(L'$@@RE),0(14)        TEMP. RE
2          LA    14,*+8
2          B     *+4+L'$@@RE
2          DS    CL(L'$@@RE)                 TEMP. RE
2          LEA@@KO OPCD=SETL,KORR=TEMP,                                  C
2                OPE1=,OPE2=,                                            C
2                OPEOM=,OPELOG=,OPEWTM=,                                 C
2                SAMPTR=X'1001',PAMHP=,IDE=
3          MVC   $@@SPTR-$@@RE(L'$@@SPTR,14),*+10     MODIFY SAMPTR
3          B     *+4+L'$@@SPTR
3          DC    XL(L'$@@SPTR)'1001'
2          CNOP  2,4
2          LA    15,*+70
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,0(1)
2          TM    4(1),X'80'
2          BZ    *+8
2          O     14,=A(X'80000000')
2          ST    14,4(1)                 MODIFY RE
2          LA    15,*+38
2          TM    20(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,20(1)                MODIFY SI
2          LM    14,15,*+6
2          BR    15                      CALL LEASY
2          DC    A(*+20)
2          DC    V(LEASY)
2 *
2          DC    C'SETL'
2          DC    C'MAINITEM'
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+12+4+((L'$@@RCCC+1)/2)*2    R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4         TEST RC
3          BRE   14                      RC OK
3          B     ERROR                   RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

LEA@STOR

Insert record

This macro inserts a record or block in the file specified, regardless of whether or not the record/block already exists. The inserted records or blocks remain locked until the transaction is closed. See “STOR Insert record” on page 179.

Operands *MF* and *PRE* are permitted.

Operation	Operands
LEA@STOR	

```
[[re],[db],[ar],[us]]

[,SAVE=address2]

           ⎧waiting-time⎫             ⎧waiting-time⎫
[,POPEWTM=⎨            ⎬]  [,TOPEWTM=⎨            ⎬]
           ⎩BLANK       ⎭             ⎩BLANK       ⎭

[,PPAMHP=X'pam-block-number'] [,TPAMHP=X'pam-block-number']

[,PIDE=ide]                   [,TIDE=ide]

           ⎧(error-code,...)⎫
[,ERRCODE=⎨                ⎬ ,ERRADDR=address4]
           ⎩address3        ⎭

[,ERRADDR=address4]
```

re	*symb* or (*r*) Address of the reference area.	
db	*symb* or (*r*) Address of the file identifier.	
	’*string*’	File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).	
us	*symb* or (*r*) Address of the *USER* area.	
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area.	
POPEWTM TOPEWTM	Waiting time for locked records (*P*=permanent, *T*=temporary).	
=waiting-time	0 ≤ *waiting-time* ≤ 999.	

=BLANK	The global waiting time for the session applies (*TIME* statement in LEASY-MAINTASK).
PPAMHP TPAMHP	PAM block number (*P*=permanent, *T*=temporary).
=X'bbbbbb'	Number of the block to be read.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@STOR L@@RE,L@@DB1,L@@AR,                                 -
                 POPEWTM=3
1          LEA@CALL 'STOR',L@@RE,L@@DB1,L@@AR,                          C
1                MF=,SAVE=,PRE=L,                                       C
1                PPAMHP=,POPEWTM=3,PIDE=,                               C
1                TPAMHP=,TOPEWTM=,TIDE=,                                C
1                ERRCODE=,ERRADDR=
2          LEA@@BP GRU=AKT,NAM=
2          LA    14,L@@RE               R14=A(RE)
2          LEA@@KO OPCD=STOR,KORR=PERM,                                 C
2                OPE1=,OPE2=,                                           C
2                OPEOM=,OPELOG=,OPEWTM=3,                               C
2                SAMPTR=,PAMHP=,IDE=
3          MVC   $@@OPWTM-$@@RE(L'$@@OPWTM,14),*+10     MODIFY OPEWTIM
3          B     *+4+((L'$@@OPWTM+1)/2)*2
3          DC    ZL(L'$@@OPWTM)'3'
2          CNOP  2,4
2          LA    15,L@@RE
2          ST    15,*+46                A(RE)
2          LA    15,L@@DB1
2          ST    15,*+42                A(DB)
2          LA    15,L@@AR
2          ST    15,*+38                A(AR)
2          OI    *+34,X'80'             LAST PARAM-ADDRESS
2          LM    14,15,*+10
2          LA    1,*+14
2          BR    15                     CALL LEASY
2          DC    A(*+28)
2          DC    V(LEASY)
2 *
2          DC    A(*+16)         PTR(OP)
2          DC    A(0)            PTR(RE)
2          DC    A(0)            PTR(DB)
2          DC    A(0)            PTR(AR)
2          DC    C'STOR'
2          L     1,4(1)                 R1=A(RE)
```

LEA@TOLR

Evaluate error codes

This macro generates a code section which identifies the list of tolerable error codes specified for *ERRCODE*. If such an error code is encountered, control branches back to the address following the action macro. Any other error code causes a branch to the address *address4*, which is specified under *ERRADDR* for the action macro.

Any number of error codes may be specified in the error code list. In the error routine a return can be enabled via general-purpose register 14 to the address following the macro that caused the error.

Operation	Operands
`LEA@TOLR`	`ERRCODE=(error-code,...)`

ERRCODE=(error-code,...)

List of tolerable error codes;

1 to 8 characters.

Example

```
            LEA@CLFL L@@RE,L@@DB1,                                            -
                  ERRCODE=TOLR,                                               -
                  ERRADDR=ERROR
1           LEA@CALL 'CLFL',L@@RE,L@@DB1,                                     C
1                 MF=,PRE=L,                                                  C
1                 ERRCODE=TOLR,ERRADDR=ERROR
2           LEA@@BP GRU=AKT,NAM=
2           CNOP  2,4
2           LA    15,L@@RE
2           ST    15,*+38                  A(RE)
2           LA    15,L@@DB1
2           ST    15,*+34                  A(DB)
2           OI    *+30,X'80'               LAST PARAM-ADDRESS
2           LM    14,15,*+10
2           LA    1,*+14
2           BR    15                       CALL LEASY
2           DC    A(*+24)
2           DC    V(LEASY)
2 *
2           DC    A(*+12)           PTR(OP)
2           DC    A(0)              PTR(RE)
2           DC    A(0)              PTR(DB)
2           DC    C'CLFL'
2           L     1,4(1)                   R1=A(RE)
2           LEA@@FB ERRADDR=ERROR,ERRCODE=TOLR,R14=ON
3           LA    14,*+16+4+((L'$@@RCCC+1)/2)*2    R14=RETURN-ADDR.
3           CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+12+4         TEST RC
3           BRE   14                       RC OK
3           LA    15,ERROR                 R15=A(ERROR-ROUTINE)
3           B     TOLR
3           DC    CL(L'$@@RCCC)'000'
  *

  TOLR      LEA@TOLR ERRCODE=(091LL118,043)
1           LEA@@BP GRU=AKT,NAM=TOLR
1           LEA@@FB ERRCODE=(091LL118,043),ERRADDR=(15),R14=OFF
2 TOLR      CLC   $@@RCCC-$@@RE(8,1),*+12       FEASIBLE RETURN CODE?
2           BRE   14                       YES, FEASIBLE
2           B     *+4+8
2           DC    C'091LL118'
2           CLC   $@@RCCC-$@@RE(3,1),*+12       FEASIBLE RETURN CODE?
2           BRE   14                       YES, FEASIBLE
2           B     *+4+4
2           DC    C'043'
2           BR    15                       RC: ERROR
```

LEA@UNLK

Cancel record lock

This macro cancels record locks. Locks can only be canceled for records or blocks that were locked within the transaction but were not updated.

Locks that are no longer required should be released in order to reduce the waiting time for locked records or blocks. All locks are canceled automatically when the transaction is terminated. See “UNLK Cancel record lock” on page 180.

Operands *MF* and *PRE* are permitted.

Operation	Operands
`LEA@UNLK`	`[[re],[db],[ar],[fa],[si],[kb],[ke]]`
	`[,SAVE=address2]`
	`[,POPE1=ext1]` `[,TOPE2=ext2]`
	`[,PPAMHP=X'pam-block-number'][,TPAMHP=X'pam-block-number']`
	`[,PIDE=ide]` `[,TIDE=ide]`
	$\texttt{[,ERRCODE=}\left\{\begin{array}{l}\texttt{(error-code,...)}\\ \texttt{address3}\end{array}\right\}\texttt{,ERRADDR=address4]}$
	`[,ERRADDR=address4]`

re	*symb* or (*r*) Address of the reference area.
db	*symb* or (*r*) Address of the file identifier.
	’*string*’ File identifier.
ar	*symb* or (*r*) Address of the I/O area (record zone).
fa	*symb* or (*r*) Address of the *FA* area (Field Selection).
	’*string*’ Identifier for the field selection.
si	*symb* or (*r*) Address of the *SI* area (Key Begin).
	’*string*’ Name of a secondary index.
kb	*symb* or (*r*) Address of the *KB* area (Key Begin).
ke	*symb* or (*r*) Address of the *KE* area (Key End).
SAVE=address2	Address (*symb* or (*r*)) of the buffer area for saving the reference area

POPE1=ext1 TOPE1=ext1	Operation code extension *OPE1* (*P*=permanent, *T*=temporary).
=U	Modified records are also released.
=BLANK	Only records that have not been modified can be released..
PPAMHP TPAMHP	PAM block number (*P*=permanent, *T*=temporary).
=X’bbbbbb’	Number of the block to be released.
PIDE=ide TIDE=ide	Identifier for DCAM (*P*=permanent, *T*=temporary).
=symb	The symbolic address indicates an 8-byte field containing a transaction identifier.
=(r)	The general-purpose register *r* contains the address of an 8-byte field containing a transaction identifier.
ERRCODE=(error-code,...)	List of tolerable error codes; 1 to 8 characters.
=address3	Symbolic address for error handling.
ERRADDR=address4	Address (*symb* or (*r*)); branch destination, if the particular error code is not in the list of tolerable error codes (*ERRCODE*).

Example

```
           LEA@UNLK MF=(E,ADDRLIST),                                      -
                 SAVE=T@@RE,                                              -
                 TPAMHP=X'3',                                             -
                 ERRADDR=ERROR
1          LEA@CALL 'UNLK',,,,,,,,                                        C
1                MF=(E,ADDRLIST),SAVE=T@@RE,PRE=L,                        C
1                PPAMHP=,PIDE=,                                           C
1                TPAMHP=X'3',TIDE=,                                       C
1                ERRCODE=,ERRADDR=ERROR
2          LEA@@BP GRU=AKT,NAM=
2          LA    1,ADDRLIST
2          L     14,4(1)                 R14=A(RE)
2          MVC   T@@RE(L'$@@RE),0(14)        SAVE=TEMP. RE
2          LA    14,T@@RE
2          LEA@@KO OPCD=UNLK,KORR=TEMP,                                   C
2                OPE1=,OPE2=,                                             C
2                OPEOM=,OPELOG=,OPEWTM=,                                  C
2                SAMPTR=,PAMHP=X'3',IDE=
3          MVC   $@@PAMHP-$@@RE(L'$@@PAMHP,14),*+10     MODIFY PAMHP
3          B     *+4+L'$@@PAMHP
3          DC    XL(L'$@@PAMHP)'3'
2          CNOP  2,4
2          LA    15,*+50
2          TM    0(1),X'80'
2          BZ    *+8
2          O     15,=A(X'80000000')
2          ST    15,0(1)
2          TM    4(1),X'80'
2          BZ    *+8
2          O     14,=A(X'80000000')
2          ST    14,4(1)                 MODIFY RE
2          LM    14,15,*+6
2          BR    15                      CALL LEASY
2          DC    A(*+12)
2          DC    V(LEASY)
2 *
2          DC    C'UNLK'
2          L     1,4(1)                  R1=A(RE)
2          LEA@@FB ERRADDR=ERROR,ERRCODE=,R14=ON
3          XR    15,15
3          BCTR  15,0                    R15=4X'FF'
3          LA    14,*+12+4+((L'$@@RCCC+1)/2)*2    R14=RETURN-ADDR.
3          CLC   $@@RCCC-$@@RE(L'$@@RCCC,1),*+8+4         TEST RC
3          BRE   14                      RC OK
3          B     ERROR                   RC: ERROR
3          DC    CL(L'$@@RCCC)'000'
```

11.5 Macros for the interpretation of currency information (CI)

In order to facilitate interpretation of the CI currency information which is returned when *OPE1=F* is specified, LEASY provides macros which generate structures in the form of DSECTs. The following macros are permitted:

- LEA@@DDL
- LEA@@DSI
- LEA@@DPL
- LEA@@DRI

Structure of the ci-inf area

The CI currency information, which is returned when *OPE1=F* is specified, is structured as follows in the *ci-inf* area:

ci-inf area for OPE2=C, BLANK, O, T, W

A subarea which contains the structure *LEA@@DDL* in the length *L@@DL1* is provided for each of the selected files. *LEA@@DDL* contains general information on the file, e.g. file name, file type.

LEA@@DDL	Subarea file 1
LEA@@DDL	Subarea file 2
. . .	. . .
LEA@@DDL	Subarea file n

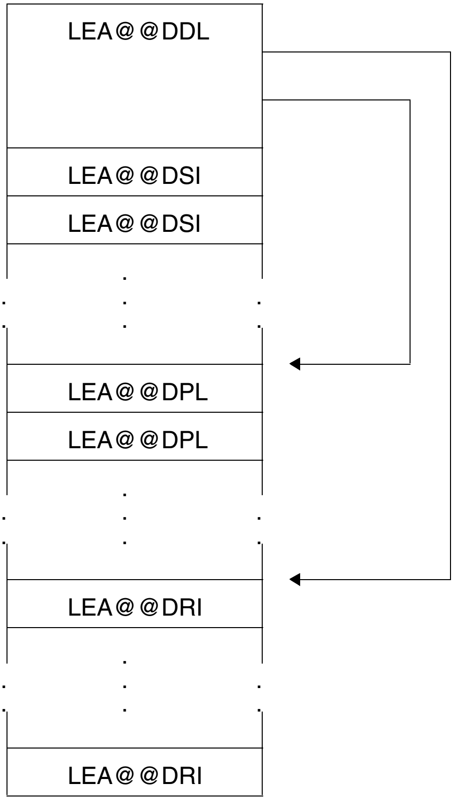
ci-inf area for OPE2=S

The *LEA@@DDL* structure is at the beginning of the *ci-inf* area. This is the only structure for files without an *SI* definition and contains general information on the file, e.g. file name, file type.

In the case of files with *SI* definitions, this is followed by an *LEA@@DSI* structure for each *SI* definition of the file. This structure contains general information on the secondary index, e.g. name, number of partial keys.

This is followed by *LEA@@DPL* structures, one such structure being provided for each partial key which is defined for this file. The structure contains entries which specify the length and position of the partial keys.

Finally, there is an *LEA@@DRI* structure for each record type defined for this file. The structure contains the record type designation and its length.



Macros

LEA@@DDL

This macro generates an *LEA@@DDL* dummy section. All the names in this DSECT are prefixed by *L@@D*. The prefix can be modified with the aid of the *PRE* operand.

Operation	Operands
LEA@@DDL	[PRE=prefix]

Example

```
           LEA@@DDL
1 *
1 *        LAYOUT OF AN ELEMENT FOR FILE DESCRIPTION
1 *
1 L@@DDL   DSECT
1 *
1           DS     A
1 *
1 L@@DLOGN DS     CL8                  LOGICAL FILE NAME
1 *
1          DS     A
1 *
1 L@@DPAD  DS     X                    PAD FACTOR
1 *
1 L@@DIND  DS     X                    INDICATOR
1 L@@DINO  EQU    X'00'                  NEITHER AIM NOR BIM SAVING
1 L@@DIA   EQU    X'80'                  AIM SAVING
1 L@@DIB   EQU    X'08'                  BIM SAVING
1 L@@DISA  EQU    X'02'                  TRUNCATED AIM RECORDS
1 L@@DRDP  EQU    X'20'                  READ PASSWORD SPECIFIED
1 L@@DWRP  EQU    X'10'                  WRITE PASSWORD SPECIFIED
1 *
1          DS     2X
1 *
1 L@@DFN   DS     CL54                 FILE NAME
1 *
1 L@@DFT   DS     C                    LEASY FILE TYPE
1 L@@DFTS  EQU    'S'                    MASTER FILE
1 L@@DFTM  EQU    'M'                    MODEL FILE
1 L@@DFTF  EQU    'F'                    FOREIGN FILE
1 L@@DFTT  EQU    'T'                    TEMPORARY FILE
1 *
1 L@@DFCB  DS     X                    FCBTYPE
1 L@@DFCBI EQU    X'40'                  ISAM
1 L@@DFCBS EQU    X'00'                  SAM
1 L@@DFCBP EQU    X'C0'                  PAM
1 L@@DFCBD EQU    X'C1'                  DAM
1 *
1 L@@DRSM  DS     H                    MAXIMUM RECORD LENGTH
1 *
1 L@@DKEYL DS     X                    KEY LENGTH
1 *
1 L@@D#SI  DS     X                    NUMBER OF SI DEFINITIONS
```

```
1 *                                      = NUMBER OF LEA@@DSI ELEMENTS
1 L@@DNOSI EQU   X'00'                    NO SI DEFINITIONS
1 *
1 L@@DL1   EQU   *-L@@DDL               LENGTH OF LEA@@DDL WITHOUT SI DEFINITIONS
1 *
1          DS    X
1 *
1 L@@DSILM DS    X                      MAXIMUM SI KEY LENGTH
1 *
1 L@@DPPLD DS    Y                      DISTANCE OF FIRST LEA@@DPL ELEMENT FROM
1 *                                      BEGINNING OF LEA@@DDL
1 *
1 L@@DRPOS DS    H                      RECORD TYPE FIELD POSITION - 1
1 *
1 L@@DRLEN DS    H                      RECORD TYPE FIELD LENGTH
1 *
1 L@@D#RID DS    H                      NUMBER OF DEFINED RECORD TYPES
1 *
1 L@@DPRID DS    Y                      DISTANCE OF FIRST LEA@@DRI ELEMENT FROM
1 *                                      BEGINNING OF LEA@@DDL
1 *
1 L@@DL2   EQU   *-L@@DDL               LENGTH OF LEA@@DDL WITH SI DEFINITIONS
1 *
1 L@@DSI   EQU   *                      BEGINNING OF FIRST LEA@@DSI ELEMENT
1 *
1          CSECT
1 *
```

LEA@@DSI

This macro generates an *LEA@@DSI* dummy section. All the names in the DSECT are prefixed by *L@@S*. The prefix can be modified with the aid of the *PRE* operand.

Operation	Operands
`LEA@@DSI`	`[PRE=prefix]`

Example

```
         LEA@@DSI
1 *
1 *      LAYOUT OF A SECONDARY INDEX DEFINITION
1 *
1 L@@SSI   DSECT
1 *
1 L@@SKEY  DS    CL8                SI NAME
1 *
1          DS    X
1 *
1 L@@SIUB  DS    X                  SI SUPPRESSION BYTE
1 *
1 L@@SIND  DS    X                  INDICATOR BYTE
1 L@@SDUP  EQU   X'01'                DUPEKY = YES
1 L@@SUPD  EQU   X'02'                KEYUPD = YES
1 L@@SIIUB EQU   X'04'                SI SUPPRESSION BYTE VALID
1 *
1 L@@S#SIP DS    HL1                NUMBER OF KEY SECTIONS
1 *                                  = NUMBER OF LEA@@DPL ELEMENTS FOR SI
1 *
1 L@@SKLSI DS    X                  LENGTH OF SECONDARY KEYS
1 *
1          DS    X
1 *
1 L@@SPPLD DS    Y                  ADDRESS POINTER TO FIRST LEA@@DPL ENTRY
1 *                                  RELATIVE TO START ADDRESS OF L@@DPPLD
1 *                                  IN LEA@@DDL
1 *
1 L@@SREP  DS    H                  REPETITION FACTOR
1 *
1 L@@S#RID DS    H                  NUMBER OF RECORD TYPE DEFINITIONS
1 *
1 L@@SPRID DS    Y                  ADDRESS POINTER TO FIRST LEA@@DRI ENTRY
1 *                                  RELATIVE TO START ADDRESS OF L@@DPRID
1 *                                  IN LEA@@DDL
1 *
1 L@@SL    EQU   *-L@@SSI           LENGTH OF A LEA@@DSI ENTRY
1 *
1          CSECT
1 *
```

LEA@@DPL

This macro generates an *LEA@@DPL* dummy section. All the names in the DSECT are prefixed by *L@@P*. The prefix can be modified with the aid of the *PRE* operand.

Operation	Operands
LEA@@DPL	[PRE=prefix]

Example

```
          LEA@@DPL
1 *
1 *       LAYOUT OF AN ELEMENT FOR KEY SECTION DEFINITION
1 *
1 L@@PPL   DSECT
1 *
1 L@@PPOS  DS    AL2                KEY SECTION START POSITION - 1
1 *
1 L@@PLEN  DS    HL1                KEY POSITION LENGTH - 1
1 *
1 L@@PDIST DS    HL2                KEY SECTION DISPLACEMENT
1 *
1 L@@PL    EQU   *-L@@PPL
1 *
1          CSECT
1 *
```

LEA@@DRI

This macro generates an *LEA@@DRI* dummy section. All the names in the DSECT are prefixed by *L@@R*. The prefix can be modified with the aid of the *PRE* operand.

Operation	Operands
`LEA@@DRI`	`[PRE=prefix]`

Example

```
          LEA@@DRI
1 *
1 *       LAYOUT OF AN ELEMENT FOR RECORD TYPE DEFINITION
1 *
1 L@@RRI   DSECT
1 *
1 L@@RLRID DS    X                  LENGTH OF RECORD TYPE DEFINITIONS
1 *
1 L@@RRID  EQU   *                  RECORD TYPE DEFINITION
1 *
1          CSECT
1 *
```

12 Sample applications

12.1 COBOL program

This example demonstrates how LEASY is called in a COBOL program via the CALL interface.
Trace listings illustrating the use of LEASY utilities at runtime can be found in the sections "Trace listing 1" (page 357) and "Trace listing 2" (page 375).

Source program listing

```
 ID DIVISION.
 PROGRAM-ID. PERSDAT.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     DATE-ISO4 IS DATE4
     TERMINAL IS DSS.
 DATA DIVISION.
* IN DIESEM PROGRAMM WERDEN ZWEI DATEIEN UEBER LEASY BEARBEITET:
* LEASY-DATEIKATALOG: LCAT
* DATEIEN:            MITABDAT:  ISAM
*                                KEYPOS=1
*                                KEYLEN=4
*                                RECFORM=F
*                                RECSIZE=74
*                                SI: ABT,POS=45,LEN=5
*                                SI: NAME,POS=5,LEN=20
*                     PROTDAT:   SAM
*                                RECSIZE=45
 WORKING-STORAGE SECTION.
 01  LEASY-PARAMS.
*
```

```
           COPY LEASYKON.                                                 (1)
*>
       *>PM L61004
       *>PM L61006
       *>PM L61029
       *>
       *>COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006
       *>                              ALL RIGHTS RESERVED
       *>          LEASY-OPERATIONSCODES.
        02        OP-CODES.
          05      OP-CATD          PIC  X(4)  VALUE  "CATD".
          05      OP-OPFL          PIC  X(4)  VALUE  "OPFL".
          05      OP-CLFL          PIC  X(4)  VALUE  "CLFL".
          05      OP-OPTR          PIC  X(4)  VALUE  "OPTR".
          05      OP-OPDB          PIC  X(4)  VALUE  "OPDB".
          05      OP-CLTR          PIC  X(4)  VALUE  "CLTR".
          05      OP-CLSE          PIC  X(4)  VALUE  "CLSE".
          05      OP-MARK          PIC  X(4)  VALUE  "MARK".
          05      OP-BACK          PIC  X(4)  VALUE  "BACK".
          05      OP-RDIR          PIC  X(4)  VALUE  "RDIR".
          05      OP-RNXT          PIC  X(4)  VALUE  "RNXT".
          05      OP-RPRI          PIC  X(4)  VALUE  "RPRI".
          05      OP-RHLD          PIC  X(4)  VALUE  "RHLD".
          05      OP-RNHD          PIC  X(4)  VALUE  "RNHD".
          05      OP-RPHD          PIC  X(4)  VALUE  "RPHD".
          05      OP-INSR          PIC  X(4)  VALUE  "INSR".
          05      OP-STOR          PIC  X(4)  VALUE  "STOR".
          05      OP-REWR          PIC  X(4)  VALUE  "REWR".
          05      OP-DLET          PIC  X(4)  VALUE  "DLET".
          05      OP-SETL          PIC  X(4)  VALUE  "SETL".
          05      OP-LOCK          PIC  X(4)  VALUE  "LOCK".
          05      OP-UNLK          PIC  X(4)  VALUE  "UNLK".
          05      OP-CINF          PIC  X(4)  VALUE  "CINF".
          05      OP-TERM          PIC  X(4)  VALUE  "TERM".
       *>
       *>          LEASY-OPENMODES
        02        OP-MODES.
          05      OP-INPUT         PIC  X     VALUE "1".
          05      OP-INPUT-SHARUPD PIC  X     VALUE "2".
          05      OP-INOUT         PIC  X     VALUE "3".
          05      OP-INOUT-SHARUPD PIC  X     VALUE "4".
          05      OP-REVERSE       PIC  X     VALUE "5".
          05      OP-UPDATE        PIC  X     VALUE "7".
          05      OP-OUTPUT        PIC  X     VALUE "8".
          05      OP-EXTEND        PIC  X     VALUE "9".
          05      OP-OUTIN         PIC  X     VALUE "A".
          05      OP-OUTIN-SHARUPD PIC  X     VALUE "B".
       *>
```

```
*>          LEASY-USAGEMODES
*>          LEASY-USAGE ABKUERZUNGEN FUER OPE-OM
 02        USAGE-MODES-OPE-OM.
*>                                                 RETR
   05      OP-A             PIC  X     VALUE "A".
*>                                                 EXUP
   05      OP-B             PIC  X     VALUE "B".
*>                                                 PRUP
   05      OP-E             PIC  X     VALUE "E".
*>                                                 EXRT
   05      OP-G             PIC  X     VALUE "G".
*>                                                 PRRT
   05      OP-I             PIC  X     VALUE "I".
*>                                                 EXLD
   05      OP-L             PIC  X     VALUE "L".
*>                                                 EXLD
   05      OP-O             PIC  X     VALUE "O".
*>                                                 EXLD
   05      OP-Q             PIC  X     VALUE "Q".
*>                                                 ULRT
   05      OP-R             PIC  X     VALUE "R".
*>                                                 ULUP
   05      OP-U             PIC  X     VALUE "U".
*>                                                 EXRT
   05      OP-X             PIC  X     VALUE "X".
*>
*>          LEASY-USAGES FUER DB1, DB2, DB4
 02        USAGE-MODES.
*>          RETRIEVAL
   05      RETR             PIC  X(4)  VALUE  "RETR".
*>          UNLOCKED  RETRIEVAL
   05      ULRT             PIC  X(4)  VALUE  "ULRT".
*>          PROTECTED RETRIEVAL
   05      PRRT             PIC  X(4)  VALUE  "PRRT".
*>          PROTECTED RETRIEVAL REVERSE
   05      PRRR             PIC  X(4)  VALUE  "PRRR".
*>          EXCLUSIVE RETRIEVAL
   05      EXRT             PIC  X(4)  VALUE  "EXRT".
*>          EXCLUSIVE RETRIEVAL REVERSE
   05      EXRR             PIC  X(4)  VALUE  "EXRR".
*>          UPDATE
   05      UPDT             PIC  X(4)  VALUE  "UPDT".
*>          UNLOCKED UPDATE
   05      ULUP             PIC  X(4)  VALUE  "ULUP".
*>          PROTECTED UPDATE
   05      PRUP             PIC  X(4)  VALUE  "PRUP".
*>          EXCLUSIVE UPDATE
   05      EXUP             PIC  X(4)  VALUE  "EXUP".
```

```
      *>          SHARE LOAD
         05      LOAD             PIC  X(4)  VALUE  "LOAD".
      *>          PROTECTED LOAD
         05      PLOD             PIC  X(4)  VALUE  "PLOD".
      *>          EXCLUSIVE LOAD
         05      ELOD             PIC  X(4)  VALUE  "ELOD".
      *>          SHARE LOAD + UPDATE
         05      LDUP             PIC  X(4)  VALUE  "LDUP".
      *>          PROTECTED LOAD + UPDATE
         05      PLUP             PIC  X(4)  VALUE  "PLUP".
      *>          EXCLUSIVE LOAD + UPDATE
         05      ELUP             PIC  X(4)  VALUE  "ELUP".
      *>          EXCLUSIVE LOAD
         05      EXLD             PIC  X(4)  VALUE  "EXLD".
      *>          UNLOCKED RETRIEVAL
         05      ULRT             PIC  X(4)  VALUE  "ULRT".
      *>
      *>          LEASY-KONSTANTE FUER RE-BEREICH
      02        OPE-CONST.
      *>          WERTE FUER FELD OPE1
         05      OPE1-F           PIC  X     VALUE  "F".
         05      OPE1-R           PIC  X     VALUE  "R".
         05      OPE1-S           PIC  X     VALUE  "S".
         05      OPE1-U           PIC  X     VALUE  "U".
         05      OPE1-W           PIC  X     VALUE  "W".
      *>          WERTE FUER FELD OPE2
         05      OPE2-C           PIC  X     VALUE  "C".
         05      OPE2-L           PIC  X     VALUE  "L".
         05      OPE2-N           PIC  X     VALUE  "N".
         05      OPE2-O           PIC  X     VALUE  "O".
         05      OPE2-S           PIC  X     VALUE  "S".
         05      OPE2-T           PIC  X     VALUE  "T".
         05      OPE2-W           PIC  X     VALUE  "W".
      *>
          COPY LEASYPAR.                                              (2)
000010*
      *>
      *>COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006
      *>                              ALL RIGHTS RESERVED
      *>          LEASY-OPERATIONSCODE
       02        OP              PIC  X(4)  VALUE SPACE.
      *>
      *>          LEASY-VERSTAENDIGUNGSBEREICH.
       02        RE.
         05      RE-K.
           10    RC-CODE.
             15  RC-CC           PIC  X(3)  VALUE SPACE.
             15  RC-KZ           PIC  X     VALUE SPACE.
```

```
       15  RC-LC           PIC  X(4)  VALUE SPACE.
        88 RC-OK                      VALUE "L000".
        88 RC-NOFIND                  VALUE "L001".
        88 RC-DUPEKY                  VALUE "L002".
        88 RC-EOF                     VALUE "L003".
        88 RC-SEQCHK                  VALUE "L004".
        88 RC-NOLOCK                  VALUE "L005".
        88 RC-LOCKED                  VALUE "L006".
        88 RC-DEADLOCK                VALUE "L007".
        88 RC-CHANGED                 VALUE "L008".
        88 RC-RECNOLOCK               VALUE "L009".
        88 RC-RECLERROR               VALUE "L010".
        88 RC-MORE255                 VALUE "L011".
        88 RC-NOACTREC                VALUE "L012".
        88 RC-KEYOUTRNG               VALUE "L013".
        88 RC-NOBIM                   VALUE "L014".
        88 RC-TSKDEADLOCK             VALUE "L015".
        88 RC-L016                    VALUE "L016".
     10    PASS            PIC  X(8)  VALUE SPACE.
     10    OPE.
       15  OPE-STX         PIC  X     VALUE SPACE.
       15  OPE-OM          PIC  X     VALUE SPACE.
       15  OPE-LOG         PIC  X     VALUE SPACE.
       15  FILLER          PIC  X(5)  VALUE SPACE.
     10    INT.
       15  SAMPTR          PIC  X(4)  VALUE LOW-VALUE.
       15  PAMHPNR         REDEFINES SAMPTR
                           PIC  9(8) COMP.
       15  SAMNUM          PIC  X(4)  VALUE LOW-VALUE.
       15  FILLER          REDEFINES SAMNUM
                           PIC  X(4)            .
     10    NUM             PIC  9(8)  VALUE ZERO.
     10    IDE             PIC  X(8)  VALUE SPACE.
  05      RE-LEASY-EXT.
     10    REOP            PIC  X(4)  VALUE SPACE.
     10    REDB            PIC  X(16) VALUE SPACE.
     10    L-OPT           PIC  X     VALUE "1".
     10    OPE1            PIC  X     VALUE SPACE.
     10    OPE2            PIC  X     VALUE SPACE.
     10    OPE-WTIME-GLOBAL           VALUE LOW-VALUE.
       15  OPE-WTIME       PIC  9(3).
     10    RC-LCE          PIC  X(5)  VALUE SPACE.
     10    U-PROT          PIC  X     VALUE SPACE.
*>
*>          LEASY-DATEILISTEN
 02        DB1.
   05      DB1-NAME        PIC  X(12) VALUE  SPACE.
*>
```

```
  02         DB3             PIC  X(12) VALUE  "ALL".
 *>
 *>           LEASY-KATALOGINFORMATIONEN
  02         CAT.
    05       CATNAME         PIC  X(24) VALUE SPACE.
    05       ZUSATZ          PIC  X(20) VALUE SPACE.
 *>
 *>           KLDS-FELDAUSWAHL
  02         FA              PIC  X(8)  VALUE  "ALL".
 *>           LEASY-SEKUNDAERINDEX
  02         SI              PIC  X(8)  VALUE SPACE.
 *>
*
 01   DB1-USED.                                  }
      05  DB-MITABDAT PIC X(12) VALUE "MITABDAT". } ──────────────  (3)
      05  DB-PROTDAT  PIC X(12) VALUE "PROTDAT".  }
*
 01   DB4-OPFL.
      05  FILLER      PIC X(27) VALUE "((MITABDAT,4),(PROTDAT,9))".  (4)
*
 01   DB4-OPTR-UPDT.
      05  FILLER      PIC X(33) VALUE "((MITABDAT,UPDT),(PROTDAT,EX  (5)
-                                     "LD))".
*
 01   DB4-OPTR-LIST.
      05  FILLER      PIC X(17) VALUE "(MITABDAT,RETR)". ────────  (6)
 01   SI-ABT          PIC X(8)  VALUE "ABT". ─────────────────────  (7)
 01   SI-NAME         PIC X(8)  VALUE "NAME". ────────────────────  (8)
*
* EIN-AUSGABEBEREICH AR
*
 01   MITABSATZ.                               }
      05  PNUM        PIC 9(6) BINARY.         }
      05  NAME        PIC X(20).               }
      05  VORNAME     PIC X(10).               }
      05  WOHNORT     PIC X(10).               }
      05  ABTEILUNG   PIC X(5).                }
      05  STRASSE     PIC X(22).               }
*                                              } ──────────────────────────  (9)
 01   PROTSATZ.                                }
      05  PAKTION     PIC X.                   }
      05  PNUM        PIC 9(6).                }
      05  NAME        PIC X(20).               }
      05  DATUM       PIC X(10).               }
      05  ZEIT        PIC 9(8).                }
*                                              }
```

```
* TERMINALEIN-AUSGABE
*
 01  TERMINALSATZ.
     05  PNUM        PIC 9(6).
     05  PNUMX       REDEFINES PNUM PIC X(6).
         88  END-KZ            VALUE "*END".
     05  FILLER      PIC X     VALUE SPACE.
     05  ABTEILUNG   PIC X(5).
     05  FILLER      PIC X     VALUE SPACE.   } ————————————————— (10)
     05  NAME        PIC X(20).
     05  FILLER      PIC X     VALUE SPACE.
     05  VORNAME     PIC X(10).
     05  FILLER      PIC X     VALUE SPACE.
     05  WOHNORT     PIC X(10).
     05  FILLER      PIC X     VALUE SPACE.
     05  STRASSE     PIC X(22).
*
 01  ERRORLINE.
     05  FILLER      PIC X(13) VALUE "LEASY-FEHLER ".
     05  ERR-KODE    PIC X(8).
*
 01  OP-LINE.                                                  } — (11)
     05  FILLER      PIC X(23) VALUE "GEWAEHLTE OPERATION: ".
     05  LAST-OP     PIC X(4).
     05  FILLER      PIC X(20) VALUE ", GEWAEHLTE DATEI: ".
     05  LAST-DB     PIC X(16).
*
 01  AUSGABE.
     05  AUSGABE1   PIC X(44) VALUE "BITTE GEBEN SIE DIE GEWUENSCH
-                                   "TE AKTION EIN:".
     05  AUSGABE2   PIC X(30) VALUE "   I..MITARBEITER EINFUEGEN".
     05  AUSGABE3   PIC X(29) VALUE "   D..MITARBEITER LOESCHEN".
     05  AUSGABE4   PIC X(30) VALUE "   L..MITARBEITER AUFLISTEN".
     05  AUSGABE5   PIC X(24) VALUE "   E..PROGRAMM BEENDEN".
*
 01  EINGABE.
     05  MITARB-EING PIC X(51) VALUE "BITTE GEBEN SIE DIE DATEN IM
-                                    " ANGEZEIGTEN FORMAT EIN".
     05  END-EING    PIC X(28) VALUE "(*END: ENDE DER EINGABE)".
     05  T-AUS.
         10  FILLER  PIC X     VALUE SPACE.
         10  FILLER  PIC X(34) VALUE "PERSNR ABTLG NAME".
         10  FILLER  PIC X(47) VALUE "VORNAME    WOHNORT    STRASS
-                              "E".
     05  ALLGEMEIN   PIC X(27) VALUE "BITTE ERROR-CODE BEACHTEN".
     05  AKTION      PIC X     VALUE SPACE.
         88  EINFUEGEN         VALUE "I".
         88  LOESCHEN          VALUE "D".
```

```
         88  AUSGEBEN          VALUE "L".
         88  ENDE              VALUE "E".
     05  TEXT-NUMERIC PIC X(30) VALUE
         "PERSONALNUMMER NICHT NUMERISCH".
     05  KODE         PIC X.
*
*
*
 01  TABELLE.
     05  FILLER      PIC X     VALUE SPACE.
     05  FILLER      PIC X(6)  VALUE ALL "*".
     05  FILLER      PIC X     VALUE SPACE.
     05  FILLER      PIC X(5)  VALUE ALL "*".
     05  FILLER      PIC X     VALUE SPACE.
     05  FILLER      PIC X(20) VALUE ALL "*".
     05  FILLER      PIC X     VALUE SPACE.
     05  FILLER      PIC X(10) VALUE ALL "*".
     05  FILLER      PIC X     VALUE SPACE.
     05  FILLER      PIC X(10) VALUE ALL "*".
     05  FILLER      PIC X     VALUE SPACE.
     05  FILLER      PIC X(22) VALUE ALL "*".
*
 PROCEDURE DIVISION.
 MENUE SECTION.
 PROG-ANF.
*        VERBINDUNG ZU LEASY KATALOG
     DISPLAY "NAME LEASY-DATEIKATALOG ?" UPON DSS. ---------------- (12)
     ACCEPT CATNAME FROM DSS. ------------------------------------ (13)
     CALL "LEASY" USING OP-CATD RE CAT. -------------------------- (14)
     IF NOT RC-OK PERFORM LEASY-ERROR
             GO TO PROG-END.          } -------------------------- (15)
*
     MOVE HIGH-VALUE TO OPE-OM. ---------------------------------- (16)
     CALL "LEASY" USING OP-OPFL RE DB4-OPFL. --------------------- (17)
     IF NOT RC-OK PERFORM LEASY-ERROR
             GO TO PROG-END.
*
     PERFORM AKTION ---------------------------------------------- (18)
     UNTIL ENDE. ------------------------------------------------- (19)
*
     CALL "LEASY" USING OP-CLFL RE. ------------------------------ (20)
     IF NOT RC-OK PERFORM LEASY-ERROR.
 PROG-END.
     STOP RUN.
```

```
/
 AKTION.
     DISPLAY AUSGABE1 UPON DSS.
     DISPLAY AUSGABE2 UPON DSS.
     DISPLAY AUSGABE3 UPON DSS.
     DISPLAY AUSGABE4 UPON DSS.
     DISPLAY AUSGABE5 UPON DSS.
     ACCEPT AKTION FROM DSS.
     MOVE AKTION TO PAKTION.           } ------------------------- (21)
     IF EINFUEGEN
     THEN PERFORM EINFUEGEN
     ELSE IF LOESCHEN
          THEN PERFORM LOESCHEN
          ELSE IF AUSGEBEN
               THEN PERFORM AUSGEBEN.
*
 EINFUEGEN SECTION.
 OPTR-INSERT.
*
*         TRANSAKTIONSEROEFFNUNG FUER INSERT
*
     CALL "LEASY" USING OP-OPTR RE DB4-OPTR-UPDT. --------------- (22)
     IF NOT RC-OK PERFORM LEASY-ERROR
         GO TO INSERT-END.
*
 TERMINAL-INPUT.
     DISPLAY MITARB-EING UPON DSS.
     DISPLAY END-EING UPON DSS.
     DISPLAY T-AUS UPON DSS.        } ---------------------------- (23)
     DISPLAY TABELLE UPON DSS.
 *
     ACCEPT TERMINALSATZ FROM DSS.
     IF END-KZ GO TO CLTR-INSRT.
     IF PNUM OF TERMINALSATZ NOT NUMERIC
         DISPLAY TEXT-NUMERIC UPON DSS    } --------------------- (24)
         GO TO TERMINAL-INPUT.
     MOVE CORR TERMINALSATZ TO MITABSATZ.
*
*         EINFUEGEN DES MITARBEITERS
 INSRT-MITABSATZ.
*
```

```
     CALL "LEASY" USING OP-INSR RE DB-MITABDAT MITABSATZ.  ------- (25)
 *
     IF RC-DUPEKY
         DISPLAY "SATZ BEREITS VORHANDEN"
         UPON DSS
         GO TO TERMINAL-INPUT             }  --------------------- (26)
     ELSE IF RC-LOCKED
             DISPLAY "SATZ GESPERRT"
             UPON DSS
             GO TO TERMINAL-INPUT.
     IF RC-OK PERFORM PROTOKOLL-WRITE.  -------------------------- (27)
     IF RC-OK GO TO TERMINAL-INPUT.
     PERFORM LEASY-ERROR.
*
*       BEENDEN DER TRANSAKTION FUER INSRT
*
 CLTR-INSRT.
     CALL "LEASY" USING OP-CLTR RE.  ----------------------------- (28)
     IF NOT RC-OK PERFORM LEASY-ERROR.
*
 INSERT-END.
         EXIT.
*
 LOESCHEN SECTION.
 OPTR-DELETE.
*
*       TRANSAKTIONSEROEFFNUNG FUER DELETE
*
     CALL "LEASY" USING OP-OPTR RE DB4-OPTR-UPDT.  --------------- (29)
     IF NOT RC-OK PERFORM LEASY-ERROR
         GO TO DELETE-END.
 DLET-EING.
     DISPLAY "BITTE GEBEN SIE DIE PERSONALNR. (6-STELLIG) EIN"
         UPON DSS.                                             }   (30)
     DISPLAY END-EING UPON DSS.
     ACCEPT PNUMX OF TERMINALSATZ FROM DSS.
     IF END-KZ GO TO CLTR-DELETE.
     IF PNUM OF TERMINALSATZ NOT NUMERIC
         DISPLAY TEXT-NUMERIC UPON DSS                 }  -------- (31)
         GO TO DLET-EING.
     MOVE PNUM OF TERMINALSATZ TO PNUM OF MITABSATZ.
*
     CALL "LEASY" USING OP-RHLD RE DB-MITABDAT MITABSATZ.  ------- (32)
     IF RC-NOFIND
         DISPLAY "SATZ NICHT VORHANDEN" UPON DSS
         GO TO DLET-EING
     ELSE IF RC-LOCKED
         DISPLAY "SATZ GESPERRT" UPON DSS
```

```
         GO TO DLET-EING.
     MOVE CORR MITABSATZ TO TERMINALSATZ.
     DISPLAY TERMINALSATZ UPON DSS.       } ——————————————————— (33)
*
 DISP.
     DISPLAY "LOESCHEN ? (J/N)" UPON DSS.
     ACCEPT KODE FROM DSS
     IF KODE = "Y" OR "J" OR "N" NEXT SENTENCE  } ————————— (34)
     ELSE GO TO DISP.
     IF KODE = "N" GO TO DLET-EING.
     CALL "LEASY" USING OP-DLET RE DB-MITABDAT MITABSATZ. ———— (35)
     IF RC-OK PERFORM PROTOKOLL-WRITE
         GO TO DLET-EING.
     PERFORM LEASY-ERROR.
*
 CLTR-DELETE.
     CALL "LEASY" USING OP-CLTR RE. —————————————————————————— (36)
     IF NOT RC-OK PERFORM LEASY-ERROR.
*
 DELETE-END.
     EXIT.
*
 AUSGEBEN SECTION.
*
*         TRANSAKTIONSEROEFFNUNG FUER LIST
*
 OPTR-LIST.
     MOVE SPACES TO TERMINALSATZ. ———————————————————————————— (37)
     CALL "LEASY" USING OP-OPTR RE DB4-OPTR-LIST. ———————————— (38)
     IF NOT RC-OK PERFORM LEASY-ERROR
         GO TO LISTOUT-END.
*
 LDISP.
     DISPLAY "SORTIERT NACH PERSNR (P), NAMEN (N) ODER ABTEILUNG (
-    "A) ?"
     UPON DSS. —————————————————————————————————————————————— (39)
     ACCEPT KODE FROM DSS.
     IF KODE = "A" MOVE SI-ABT TO SI
     ELSE IF KODE = "N" MOVE SI-NAME TO SI  } ———————————————— (40)
     ELSE MOVE SPACE TO SI.
*
*        POSITIONIEREN AUF DATEIANFANG
     MOVE LOW-VALUE TO MITABSATZ. ———————————————————————————— (41)
     CALL "LEASY" USING OP-SETL RE DB-MITABDAT
         MITABSATZ FA SI. ——————————————————————————————————— (42)
         IF NOT RC-OK PERFORM LEASY-ERROR
         GO TO CLTR-LIST.
```

```
 RNXT.
     CALL "LEASY" USING OP-RNXT RE DB-MITABDAT MITABSATZ  ---------  (43)
     IF RC-OK
        MOVE CORR MITABSATZ TO TERMINALSATZ  }
        DISPLAY TERMINALSATZ UPON DSS        }  ----------------------  (44)
        GO TO RNXT.                          }
        IF NOT RC-EOF PERFORM LEASY-ERROR.
*
 CLTR-LIST.
     CALL "LEASY" USING OP-CLTR RE.  ------------------------------------  (45)
     IF NOT RC-OK PERFORM LEASY-ERROR.
*
 LISTOUT-END.
*
 PROTOKOLL-WRITE SECTION.                                           }
 PROT.                                                              }
     MOVE CORR MITABSATZ TO PROTSATZ.                               }
     ACCEPT DATUM FROM DATE4.                                       }
     ACCEPT ZEIT FROM TIME.                                         }  ---------  (46)
     CALL "LEASY" USING OP-INSR RE DB-PROTDAT PROTSATZ.             }
 PROT-END.                                                          }
     EXIT.                                                          }
*                                                                   }
 LEASY-ERROR SECTION.
 ERROR-DISPLAY.
*                                   }
     MOVE RC-CODE TO ERR-KODE.      }
     MOVE REOP TO LAST-OP.          }
     MOVE REDB TO LAST-DB.          }  --------------------------------  (47)
     DISPLAY ERRORLINE UPON DSS.    }
     DISPLAY OP-LINE UPON DSS.      }
     DISPLAY ALLGEMEIN UPON DSS.    }
*
 ERROR-END.
     EXIT.
```

Explanation

(1) The COPY element LEASYKON, which contains the definitions of LEASY operations, OPEN modes and USAGE modes, is copied.

(2) The COPY element LEASYPAR, which contains the RE communication area and the remaining definitions of the operands of the CALL-LEASY statement, is copied.

(3) The files MITABDAT and PROTDAT are defined in the file format DB1 so that they can be accessed individually.

(4) All files to be opened in the file format DB4 are assigned together with their OPEN mode:

MITABDAT in OPEN mode 4: INOUT,SHARUPD
PROTDAT in OPEN mode 9: EXTEND

(5) The files accessed in the UPDT transaction are assigned:

MITABDAT in USAGE mode UPDT
PROTDAT in USAGE mode EXLD

(6) The file accessed in the LIST transaction is assigned.

(7) The secondary key ABT is defined.

(8) The secondary key NAME is defined.

(9) The AR input/output area is defined with the name MITABSATZ for the MITABDAT file and with the name PROTSATZ for the PROTDAT file.

(10) The area for input/output to/from the data display terminal is defined.

(11) The output in the event of an error is defined (see LEASY-ERROR SECTION).

(12) The LEASY catalog to be edited is specified.

(13) The input is supplied in the CATNAME field, which is defined in the CAT catalog information.

(14) CALL-LEASY call for the CATD operation. The RE communication area and the CAT catalog information are also transferred.

(15) The error behavior is defined.

(16) Since files are transferred in DB4 format in the next CALL-LEASY statement, the OPE-OM field must be set to HIGH-VALUE.

(17) CALL-LEASY call for the OPFL operation. The files defined with DB4-OPFL are physically opened.

(18) Messages are output on the data display terminal requesting the user to select the action to be performed. Depending on the user’s choice, the program branches to the appropriate section.

(19) This continues until the user enters the end criterion.

(20) The CALL-LEASY call with the CLFL operation closes the files after the end criterion is specified.

(21) See step (18).

(22) This CALL-LEASY statement opens the INSERT transaction. The files defined with DB4-OPTR-UPDT are logically opened.

(23) Messages on the data display terminal informing the user of the input format.

(24) The input is supplied to the TERMINALSATZ area. Provided an end criterion is not entered and the input is numeric, the contents of TERMINALSATZ are transferred to the input/output area MITABSATZ of the file MITABDAT.

(25) This CALL-LEASY call writes the data in the AR area MITABSATZ to the file MITABDAT assigned using DB-MITABDAT.

(26) If the specified record already exists or is locked, appropriate messages are output.

(27) If the INSERT transaction is executed without errors, the program branches to the PROTOKOLL-WRITE SECTION.

(28) If the end criterion is entered on the data display terminal or if an error occurs while inserting a record, the INSERT transaction is terminated by issuing a CALL-LEASY call with the CLTR operation.

(29) The DELETE transaction is opened. The files defined with DB4-OPTR-UPDT are logically opened.

(30) The personnel number of the employee to be deleted must be entered. When converted to binary format (4 bytes), this forms the primary key of the record to be deleted.

(31) The input is supplied to the PNUMX field of the TERMINALSATZ area. Provided an end criterion is not entered and the input is numeric, the contents of the PNUM field of the TERMINALSATZ area are transferred to the PNUM field of the MITABSATZ area.

(32) This CALL-LEASY call is used to read the record whose primary key is stored in the MITABSATZ record zone from the MITABDAT file and lock it.

(33) The record contents just read into the record zone are transferred to the TERMINALSATZ area and output on the data display terminal.

(34) A message is output on the data display terminal requesting the user to confirm whether or not the record is to be deleted. The next program step depends on the user’s response.

(35) The record just read and locked is deleted from the MITABDAT file by issuing a CALL-LEASY statement with the DLET operation.

(36) If the end criterion is entered on the data display terminal or if an error occurs while deleting a record, the DELETE transaction is terminated by issuing a CALL-LEASY call with the CLTR operation.

(37) The TERMINALSATZ area is cleared.

(38) The LIST transaction is opened. The file defined with DB4-OPTR-LIST is logically opened.

(39) This message requests the user to select the sort criterion for the records to be output.

(40) Depending on the user's specification, either the name of the corresponding secondary key to which the program navigates in the next SETL operation is written in the field of the secondary index SI, or the SI field is overwritten with blanks in which case the program navigates to the primary key.

(41) If the MITABSATZ input/output area is cleared, the program navigates to the start of the file in the next SETL operation.

(42) This CALL-LEASY statement enables the program to navigate to the start of the file.

(43) This CALL-LEASY statement reads the next record in the ascending primary or secondary key sequence (depending on the contents of the SI field) and transfers this to the MITABSATZ area. The program then navigates to the new record.

(44) The contents of the MITABSATZ area are written to the TERMINALSATZ area and output on the data display terminal. The new record is then read.

(45) If an error occurs when navigating or reading or if the end of the file is reached, the LIST transaction is terminated by issuing a CALL-LEASY call with the CLTR operation.

(46) If a record was inserted or deleted, the program branches to the PROTOKOLL-WRITE SECTION. Here, the contents of the PNUM and NAME fields of MITABSATZ are transferred to the corresponding fields of the PROTSATZ input/output area of the PROTDAT file. The date and time are also entered in this area. The contents of the PROTSATZ area are inserted in the PROTDAT file by issuing a CALL-LEASY statement with the INSR operation.

(47) If an error occurs in a CALL-LEASY call, the program branches to this point. The error codes stored by LEASY in the RE communication area or the last operation code and file name are output.

12.2 Assembler program

This example demonstrates how LEASY is called in an Assembler program via the macro interface.
Trace listings illustrating the use of LEASY utilities at runtime can be found in the sections “Trace listing 1” (page 357) and “Trace listing 2” (page 375).

Source program listing

```
  PERSDAT  START    0
           TITLE    'P E R S D A T'
           PRINT    NOGEN
  PERSDAT  AMODE    ANY
  PERSDAT  RMODE    ANY
           GPARMOD  31
2                *,VERSION 010
  *
  *
  ******************************************************************
  * IN DIESEM PROGRAMM WERDEN ZWEI DATEIEN UEBER LEASY BEARBEITET: *
  * LEASY-DATEIKATALOG: LCAT                                       *
  * DATEIEN:            MITABDAT:  ISAM                            *
  *                                KEYPOS=1                        *
  *                                KEYLEN=4                        *
  *                                RECFORM=F                       *
  *                                RECSIZE=74                      *
  *                                SI: ABT,POS=45,LEN=5            *
  *                                SI: NAME,POS=5,LEN=20           *
  *                     PROTDAT:   SAM                             *
  *                                RECSIZE=45                      *
  ******************************************************************
  *
  *
  ANF      BALR     R3,0
           USING    *,R3,R4
           USING    GTIMED,R5
           LA       R4,2048(0,R3)
           LA       R4,2048(0,R4)
           BIND     SYMBOL=I@GTIME,SYMBLAD=AENTRY
  *
  *
  MENUE    DS       0H
  *
  *                 VERBINDUNG ZU LEASY-KATALOG
  *
           MVC      AUSLEN,=Y(30)
```

```
           MVC      AUSTEXT(25),='NAME LEASY-DATEIKATALOG ?' -----------  (1)
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      EINTEXT,EINTEXT-1
           RDATA    EINB,TIAMERR
2                *,@DCEI      999    921011   53531002
1                *,RDATA      009    920320   53121057
           MVC      L@@CAT,EINTEXT -------------------------------------  (2)
           LA       R10,PROGEND
           LEA@CATD L@@RE,L@@CAT,ERRADDR=LEASYERR ----------------------  (3)
  *
           LA       R10,PROGEND
           LEA@OPFL L@@RE,DB4OPFL,POPEOM=X'FF',ERRADDR=LEASYERR --------  (4)
  *
  *                 AUSWAHL DER FUNKTIONEN
  AKTION   DS       0H -------------------------------------------------  (5)
           MVC      AUSLEN,=Y(49)
           MVC      AUSTEXT(44),AUSGABE1
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      AUSLEN,=Y(35)
           MVC      AUSTEXT(30),AUSGABE2
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      AUSLEN,=Y(34)
           MVC      AUSTEXT(29),AUSGABE3
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      AUSLEN,=Y(35)
           MVC      AUSTEXT(30),AUSGABE4
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      AUSLEN,=Y(29)
           MVC      AUSTEXT(24),AUSGABE5
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      EINTEXT,EINTEXT-1
           RDATA    EINB,TIAMERR
2                *,@DCEI      999    921011   53531002
1                *,RDATA      009    920320   53121057
           MVC      PAKTION,EINTEXT
           CLI      PAKTION,'I'
```

```
           BE       EINFUEGN
           CLI      PAKTION,'D'
           BE       LOESCHEN
           CLI      PAKTION,'L'
           BE       AUSGEBEN
           CLI      PAKTION,'E' ---------------------------------------  (6)
           BNE      AKTION
  *
  *                 ABSCHLUSS-ROUTINEN
  *
  ENDE     LA       R10,PROGEND
           LEA@CLFL L@@RE,ERRADDR=LEASYERR ----------------------------  (7)
  *
  PROGEND  TERM
2                *,VERSION 100
  *
  *
  EINFUEGN DS       0H
  *
  *                 TRANSAKTIONSEROEFFNUNG FUER INSERT
  *
  OPTRINSR LA       R10,INSREND
           LEA@OPTR L@@RE,DB4OPTRU,ERRADDR=LEASYERR -------------------  (8)
  *
  *                 TERMINAL-EINGABE
  TERMINP  MVC      AUSLEN,=Y(56)
           MVC      AUSTEXT(51),MITARBEG
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      AUSLEN,=Y(33)
           MVC      AUSTEXT(28),ENDEING
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058  ----------------  (9)
           MVC      AUSLEN,=Y(84)
           MVC      AUSTEXT(79),TAUS
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      AUSLEN,=Y(84)
           MVC      AUSTEXT(79),TSTARS
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
  *
```

```
           MVC      EINTEXT,EINTEXT-1
           RDATA    EINB,TIAMERR
2                *,@DCEI      999    921011   53531002
1                *,RDATA      009    920320   53121057
           MVC      TSATZ,EINTEXT
           CLC      TSATZ(4),ENDKZ
           BE       CLTRINSR
           LA       R10,TERMINP
           BAL      R11,TESTNUM                                    ———— (10)
           PACK     DOWO,TPNUM
           CVB      R12,DOWO
           ST       R12,MPNUM
           MVC      MNAME,TNAME
           MVC      MVORNAME,TVORNAME
           MVC      MWOHNORT,TWOHNORT
           MVC      MABTLNG,TABTLNG
           MVC      MSTRASSE,TSTRASSE
  *
  *                 EINFUEGEN DES MITARBEITERS
  INSMSATZ LA       R10,CLTRINSR
           LEA@INSR L@@RE,DBMDAT,MSATZ,                                   *
                 ERRCODE=(L002,L006),ERRADDR=LEASYERR ——————————————— (11)
  TST02A   CLC      L@@RCLC,='L002'
           BNE      TST06A
           MVC      AUSLEN,=Y(27)
           MVC      AUSTEXT(22),='SATZ BEREITS VORHANDEN'
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           B        TERMINP                                        ———— (12)
  TST06A   CLC      L@@RCLC,='L006'
           BNE      RCD00A
           MVC      AUSLEN,=Y(18)
           MVC      AUSTEXT(13),='SATZ GESPERRT'
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           B        TERMINP
  RCD00A   LA       R10,TERMINP
           B        PROTWRT ——————————————————————————————————————————— (13)
  *
  *                 BEENDEN DER TRANSAKTION FUER INSERT
  *
  CLTRINSR LA       R10,INSREND
           LEA@CLTR L@@RE,ERRADDR=LEASYERR —————————————————————————— (14)
  *
  INSREND  B        AKTION
  *
```

```
    *
    LOESCHEN DS       0H
    *
    *                 TRANSAKTIONSEROEFFNUNG FUER DELETE
    *
    OPTRDLET LA       R10,DLETEND
             LEA@OPTR L@@RE,DB4OPTRU,ERRADDR=LEASYERR  ——————————————————— (15)
    *
    *                 TERMINAL-EINGABE
    DLETEING MVC      AUSLEN,=Y(55)
             MVC      AUSTEXT(50),='BITTE GEBEN SIE DIE PERSONALNUMMER (6-S*
                   TELLIG) EIN'                                       ⎫
             WROUT    AUSB,TIAMERR                                    ⎪
  2                *,@DCEO      999    921011   53531004              ⎪
  1                *,WROUT      006    920316   53121058              ⎪
             MVC      AUSLEN,=Y(33)                                   ⎬  (16)
             MVC      AUSTEXT(28),ENDEING                             ⎪
             WROUT    AUSB,TIAMERR                                    ⎪
  2                *,@DCEO      999    921011   53531004              ⎪
  1                *,WROUT      006    920316   53121058              ⎭
             MVC      EINTEXT,EINTEXT-1
             RDATA    EINB,TIAMERR                                    ⎫
  2                *,@DCEI      999    921011   53531002              ⎪
  1                *,RDATA      009    920320   53121057              ⎪
             MVC      TPNUM,EINTEXT                                   ⎪
             CLC      TPNUM(4),ENDKZ                                  ⎪
             BE       CLTRDLET                                        ⎬  (17)
             LA       R10,DLETEING                                    ⎪
             BAL      R11,TESTNUM                                     ⎪
             PACK     DOWO,TPNUM                                      ⎪
             CVB      R12,DOWO                                        ⎪
             ST       R12,MPNUM                                       ⎪
             LA       R10,CLTRDLET                                    ⎭
             LEA@RHLD L@@RE,DBMDAT,MSATZ,                                *
                   ERRCODE=(L001,L006),ERRADDR=LEASYERR  ————————————— (18)
    TST01B   CLC      L@@RCLC,='L001'
             BNE      TST06B
             MVC      AUSLEN,=Y(25)
             MVC      AUSTEXT(20),='SATZ NICHT VORHANDEN'
             WROUT    AUSB,TIAMERR
  2                *,@DCEO      999    921011   53531004
  1                *,WROUT      006    920316   53121058
             B        DLETEING
    TST06B   CLC      L@@RCLC,='L006'
             BNE      RCD00B
             MVC      AUSLEN,=Y(18)
             MVC      AUSTEXT,='SATZ GESPERRT'
             WROUT    AUSB,TIAMERR
```

```
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           B        DLETEING
  RCDOOB   L        R12,MPNUM
           CVD      R12,DOWO
           UNPK     TPNUM,DOWO
           OI       TPNUM+5,X'FO'
           MVC      TNAME,MNAME
           MVC      TVORNAME,MVORNAME
           MVC      TWOHNORT,MWOHNORT                                      (19)
           MVC      TABTLNG,MABTLNG
           MVC      TSTRASSE,MSTRASSE
           MVC      AUSLEN,=Y(83)
           MVC      AUSTEXT(78),TSATZ
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
  DISP     MVC      AUSLEN,=Y(21)
           MVC      AUSTEXT(16),='LOESCHEN ? (J/N)'
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      EINTEXT,EINTEXT-1
           RDATA    EINB,TIAMERR
2                *,@DCEI      999    921011   53531002                  (20)
1                *,RDATA      009    920320   53121057
           CLI      EINTEXT,'N'
           BE       DLETEING
           CLI      EINTEXT,'J'
           BE       DELMSATZ
           CLI      EINTEXT,'Y'
           BNE      DISP
  *
  *                 LOESCHEN DES MITARBEITERS
  *
  DELMSATZ LA       R10,CLTRDLET
           LEA@DLET L@@RE,DBMDAT,MSATZ,ERRADDR=LEASYERR  ------------   (21)
           LA       R10,DLETEING
           B        PROTWRT
  *
  *                 BEENDEN DER TRANSAKTION FUER DELETE
  *
  CLTRDLET LA       R10,DLETEND
           LEA@CLTR L@@RE,ERRADDR=LEASYERR  -------------------------   (22)
  *
  DLETEND  B        AKTION
  *
  *
```

```
  AUSGEBEN DS       OH
  *
  *                 TRANSAKTIONSEROEFFNUNG FUER LIST
  *                                                                   }
  OPTRLIST MVI      TSATZ,' '                                         } ---- (23)
           MVC      TSATZ+1(77),TSATZ                                 }
           LA       R10,LISTEND
           LEA@OPTR L@@RE,DB4OPTRL,ERRADDR=LEASYERR ------------------------ (24)
  *
  *                 TERMINAL-EINGABE
  LDISP    MVC      AUSLEN,=Y(61)
           MVC      AUSTEXT,='SORTIERT NACH PERSNR (P), NAMEN (N) ODER AB*
                 TEILUNG (A) ?'                                       }
           WROUT    AUSB,TIAMERR                                      } ---- (25)
2                *,@DCEO      999    921011   53531004                }
1                *,WROUT      006    920316   53121058
           MVC      EINTEXT,EINTEXT-1                                 }
           RDATA    EINB,TIAMERR                                      }
2                *,@DCEI      999    921011   53531002                }
1                *,RDATA      009    920320   53121057                }
  LDA      CLI      EINTEXT,'A'                                       }
           BNE      LDN                                               }
           MVC      L@@SI,SIABT                                       }
           B        POSANF                                            }
  LDN      CLI      EINTEXT,'N'                                       } ---- (26)
           BNE      LDP                                               }
           MVC      L@@SI,SINAME                                      }
           B        POSANF                                            }
  LDP      CLI      EINTEXT,'P'                                       }
           BNE      LDISP                                             }
           MVC      L@@SI,=CL8' '                                     }
           B        POSANF                                            }
  *                                                                   }
  *                 POSITIONIEREN AUF DATEIANFANG
  POSANF   XC       MSATZ,MSATZ ------------------------------------------- (27)
           LA       R10,CLTRLIST
           LEA@SETL L@@RE,DBMDAT,MSATZ,L@@FA,L@@SI,ERRADDR=LEASYERR ---- (28)
  *
  *                 NAECHSTEN SATZ LESEN UND AUSGEBEN
  *
  RNXT     LA       R10,CLTRLIST
           LEA@RNXT L@@RE,DBMDAT,MSATZ,                                     *
                 ERRCODE=(L003),ERRADDR=LEASYERR -------------------------- (29)
           CLC      L@@RCLC,='L003'
           BE       CLTRLIST
```

```
                L        R12,MPNUM
                CVD      R12,DOWO
                UNPK     TPNUM,DOWO
                OI       TPNUM+5,X'F0'
                MVC      TNAME,MNAME
                MVC      TVORNAME,MVORNAME
                MVC      TWOHNORT,MWOHNORT                        ——— (30)
                MVC      TABTLNG,MABTLNG
                MVC      TSTRASSE,MSTRASSE
                MVC      AUSLEN,=Y(83)
                MVC      AUSTEXT(78),TSATZ
                WROUT    AUSB,TIAMERR
2                    *,@DCEO      999    921011   53531004
1                    *,WROUT      006    920316   53121058
                B        RNXT
  *
  *                      BEENDEN DER TRANSAKTION FUER LIST
  *
  CLTRLIST LA       R10,LISTEND
           LEA@CLTR L@@RE,ERRADDR=LEASYERR ———————————————————— (31)
  *
  LISTEND  B        AKTION
  *
  *
  PROTWRT  DS       0H
  *
  *                 PROTOKOLL-SATZ AUSGABE
  *
  PROT     L        R12,MPNUM
           CVD      R12,DOWO
           UNPK     PPNUM,DOWO
           OI       PPNUM+5,X'F0'
           MVC      PNAME,MNAME
           LA       R13,SAVE                                      ——— (32)
           LA       R5,GTIMEL
           GTIME    MF=E,PARAM=GTIMEL,LINKADR=AENTRY
           MVC      PDATUM,NTIGDTIC
           MVC      PZEIT,NTIGTDI
           LEA@INSR L@@RE,DBPDAT,PSATZ,ERRADDR=LEASYERR
  *
  PROTEND  BR       R10
  *
  *
  TESTNUM  DS       0H
  *
  *                 PRUEFUNG AUF NUMERISCHEN WERT
  *
           LA       R12,6
```

```
           LA       R13,TPNUM
  TSTNUM1  CLI      0(R13),'9'
           BH       NOTNUM
           CLI      0(R13),'0'
           BL       NOTNUM
           LA       R13,1(0,R13)
           BCT      R12,TSTNUM1
           BR       R11
  NOTNUM   MVC      AUSLEN,=Y(35)
           MVC      AUSTEXT,TEXTNUME
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           BR       R10
  *
  *
  LEASYERR DS       0H
  *
  *                 ERROR-DISPLAY
  *
           MVC      ERRKODE,L@@RCCC
           MVC      LASTOP,L@@REOP
           MVC      LASTDB,L@@REDB
           MVC      AUSLEN,=Y(26)
           MVC      AUSTEXT(21),ERRLINE
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      AUSLEN,=Y(68)                                         (33)
           MVC      AUSTEXT(63),OPLINE
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
           MVC      AUSLEN,=Y(32)
           MVC      AUSTEXT(27),ALGEMEIN
           WROUT    AUSB,TIAMERR
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
  *
  ERROREND BR       R10
  *
  *
  TIAMERR  WROUT    TIAMERRL,TERMD
2                *,@DCEO      999    921011   53531004
1                *,WROUT      006    920316   53121058
  *
  *
  TERMD    TERM     UNIT=STEP,MODE=ABNORMAL,DUMP=Y
```

```
2                *,VERSION 100
           PRINT    GEN
  *
  *
  *                 LEASY-PARAMS
  *
           LEA@RE -----------------------------------------------------  (34)
1 ********       LEASY COMMUNICATION AREA, VERSION 62A
1          DS    OF
1 L@@RE        DS    0CL80           LEASY COMMUNICATION AREA
1 L@@REK       DS    0CL48           COMPATIBLE PART
1 L@@RCCC      DC    CL3'000'        COMPATIBLE RETURNCODE
1 L@@RCOK      EQU   C'000'          RETURN-CODE: NO ERROR
1 L@@RCKZ      DC    CL1'L'          LEASY SYSTEM CHARACTERISTIC
1 L@@RCLC      DC    CL4'L000'       LEASY RETURN CODE
1 L@@PASS      DC    CL8' '          PASSWORD (FOR FUTURE VERSIONS)
1 L@@OPE       DS    0CL8            SUPPLEMENT FOR LEASY OPERATIONS
1 L@@OPSTX     DC    CL1' '         RESERVED
1 L@@STXA1     EQU   C' '            LEASY-STXIT-ROUTINE
1 L@@STXA2     EQU   X'00'           LEASY-STXIT-ROUTINE
1 L@@STXN      EQU   C' '            *VALUES ’N’ AND ’P’ NO LONGER
1 L@@STXP      EQU   C' '            *SUPPORTED SINCE LEASY V6.1
1 L@@OPOM      DC    CL1' '          OPEN MODE
1 L@@BLAN      EQU   C' '             STD FOR FORMAT DB1 AND DB2
1 L@@INPUT     EQU   C'1'             DVS OPEN MODE INPUT
1 L@@INPUS     EQU   C'2'             DVS OPEN MODE INPUT, SHARUPD
1 L@@INOUT     EQU   C'3'             DVS OPEN MODE INOUT
1 L@@INOUS     EQU   C'4'             DVS OPEN MODE INOUT, SHARUPD
1 L@@REVER     EQU   C'5'             DVS OPEN MODE REVERSE
1 L@@UPDAT     EQU   C'7'             DVS OPEN MODE UPDATE
1 L@@OUTPU     EQU   C'8'             DVS OPEN MODE OUTPUT
1 L@@EXTEN     EQU   C'9'             DVS OPEN MODE EXTEND
1 L@@OUTIN     EQU   C'A'             DVS OPEN MODE OUTIN
1 L@@OUTIS     EQU   C'B'             DVS OPEN MODE OUTIN, SHARUPD
1 *                              USAGE-MODES
1 L@@EXLD      EQU   C' '             USAGE-MODE EXLD (SAM)
1 L@@UPDT      EQU   C' '             USAGE-MODE UPDT (ISAM/PAM)
1 L@@RETRA     EQU   C'A'             USAGE-MODE RETR
1 L@@PRUPE     EQU   C'E'             USAGE-MODE PRUP
1 L@@EXRTG     EQU   C'G'             USAGE-MODE EXRT
1 L@@EXLDL     EQU   C'L'             USAGE-MODE EXLD
1 L@@PRRTI     EQU   C'I'             USAGE-MODE PRRT
1 L@@EXLDO     EQU   C'O'             USAGE-MODE EXLD
1 L@@ULRTR     EQU   C'R'             USAGE-MODE ULRT
1 L@@EXLDQ     EQU   C'Q'             USAGE-MODE EXLD
1 L@@ULUPU     EQU   C'U'             USAGE-MODE ULUP
1 L@@EXRTX     EQU   C'X'             USAGE-MODE EXRT
1 L@@EXUPB     EQU   C'B'             USAGE-MODE EXUP
```

```
1 *
1 L@@HVAL      EQU   X'FF'            FORMAT DB4
1 L@@OPLG      DC    CL1' '          BIM LOGGING FIELD
1 L@@YBIM      EQU   C' '             WITH BIM LOGGING
1 L@@NBIM      EQU   C'N'             WITHOUT BIM LOGGING
1         DC    CL5' '          UNUSED
1 L@@INT       DS    0XL8            FIELD FOR INTERNAL KEYS
1 L@@SPTR      DC    F'0'            SAM : ID1RPTR (24-BIT) OR
1 *                                        ID1BLK# (31-BIT)
1         ORG   L@@SPTR
1 L@@PAMHP     DC    F'0'            PAM : PAM BLOCK NUMBER
1 L@@SASNR     DC    F'0'            SAM : ID1REC# (31-BIT)
1         ORG   L@@SASNR            PAM : UNUSED, SAM UNUSED (24-BIT)
1 L@@UNUSE     DC    F’0’
1 L@@NUM       DC    CL8' '          NUMBER OF PRIMARY RECORDS
1 L@@IDE       DC    CL8' '          DCAM IDENTIFIKATION
1 L@@RELE      DS    0CL32           LEASY EXTENSION OF RE
1 L@@REOP      DC    CL4' '          LAST OPERATION CODE
1 L@@REDB      DC    CL16' '         LAST FILE (+SI-NAME)
1 L@@LOPT      DC    CL1'1'          LEASY VERSION BYTE
1 *                               HAS TO CONTAIN '1'
1 L@@OPE1      DC    CL1' '          FOR OPTR,CLTR,RHLD,RNHD,RPHD,
1 *                              LOCK,CINF,UNLK
1 *                              OPTR,CLTR:
1 L@@OCST      EQU   C' '             STANDARD OPEN/CLOSE OF TRANSACTION
1 *                              OPTR:
1 L@@OPW       EQU   C'W'             OPEN TRANSAKTION USING
1 *                               FILE POINTERS IN CI-AREA
1 *                              CLTR:
1 L@@CLR       EQU   C'R'             ROLL BACK TRANSACTION
1 *                              RHLD,RNHD,RPHD,LOCK:
1 L@@SLOC      EQU   C'S'             READ LOCK (SHARE LOCK)
1 L@@WLOC      EQU   C' '             WRITE LOCK
1 *                              CINF:
1 L@@CINFW     EQU   C' '             CINF-AUFRUF FUER OPTR/W
1 L@@CIFI      EQU   C'F'             CINF FUER FILE-INFO
1 *
1 *                              UNLK:
1 L@@ULST      EQU   C' '             STANDARD FOR UNLK
1 L@@ULUP      EQU   C'U'             UNLK OF UPDATED RECORDS
1 *
1 L@@OPE2      DC    CL1' '      CLOSE TRANSACTION WITH
1 *                                  OR WITHOUT NEW START
1 L@@CLST      EQU   C' '         NO NEW START
1 L@@CLT       EQU   C'T'         NEW START
1 *
1 *
1 L@@NUMY      EQU   C'N'         RDIR/RHLD WITH NUM
```

```
1 L@@NUMN      EQU   C' '         RDIR/RHLD WITHOUT NUM
1 *                           NAEHERE ANGABEN FUER CINF
1 L@@CICA      EQU   C' '         INFO UEBER GANZEN KATALOG
1 L@@CICT      EQU   C'C'         INFO UEBER GANZEN KATALOG
1 L@@CIOP      EQU   C'O'         INFO UEBER OFFENE DATEIEN
1 L@@CITA      EQU   C'T'         INFO UEBER DATEIEN DER TA
1 L@@CISP      EQU   C'S'         INFO UEBER SPEZIELLE DATEI
1 L@@CIW       EQU   C'W'         FORTSETZUNG DER AUSKUNFTSFKT.
1 *
1 L@@ROL       EQU   C'L'         READ OVER LOCKED RECORD
1 L@@RNOL      EQU   C' '         DO NOT READ OVER LOCKED REC.
1 *
1 L@@OPWTM     DS    0ZL3         WAIT TIME FOR LOCKING
1          DC    XL3'0'       DEFAULT VALUE
1 *
1 L@@EXRC      DC    CL5' '       ZUSAETZLICHER RETURNCODE
1 L@@UPROT     DC    C' '          AIM PROTOKOLLIERUNGS-KENNZEICHEN
1 L@@NPROT     EQU   C' '          NO USER-INFO AVAILABLE
1 L@@YPROT     EQU   C'Y'          USER-INFORMATION AVAILABLE
  *
           LEA@CAT ----------------------------------------------------- (35)
1          LEA@@BP TYP=L,PRE=L,NAM=,GRU=GEN,ID=CAT
1 L@@CAT   DS    0CL44
1 L@@CATC   DC    CL24' '
1 L@@CATZ   DC    CL20' '
  *
           LEA@FA   ALL ------------------------------------------------ (36)
1          LEA@@BP PRE=L,NAM=,GRU=GEN,ID=FA,TYP=L
1 L@@FA    DC    CL8'ALL'
  *
           LEA@SI ------------------------------------------------------ (37)
1          LEA@@BP TYP=L,PRE=L,NAM=,GRU=GEN,ID=SI
1 L@@SI    DC    CL8' '
  *
  DBMDAT   LEA@DB1  MITABDAT ------------------------------------------- (38)
1          LEA@@BP TYP=L,PRE=L,NAM=DBMDAT,GRU=GEN,ID=DB1
1 DBMDAT   DC    CL12'MITABDAT'
  *
  DBPDAT   LEA@DB1  PROTDAT
1          LEA@@BP TYP=L,PRE=L,NAM=DBPDAT,GRU=GEN,ID=DB1
1 DBPDAT   DC    CL12'PROTDAT'
  *
  DB4OPFL  LEA@DB   ((MITABDAT,4),(PROTDAT,9)) ------------------------- (39)
1          LEA@@BP TYP=L,PRE=L,NAM=DB4OPFL,GRU=GEN,ID=DB
1 DB4OPFL  DC    '((MITABDAT,4),(PROTDAT,9))'
  *
  DB4OPTRU LEA@DB   ((MITABDAT,UPDT),(PROTDAT,EXLD)) ------------------- (40)
1          LEA@@BP TYP=L,PRE=L,NAM=DB4OPTRU,GRU=GEN,ID=DB
```

```
1 DB4OPTRU DC    '((MITABDAT,UPDT),(PROTDAT,EXLD))'
  *
  DB4OPTRL LEA@DB   (MITABDAT,RETR) ------------------------------------ (41)
1          LEA@@BP TYP=L,PRE=L,NAM=DB4OPTRL,GRU=GEN,ID=DB
1 DB4OPTRL DC    '(MITABDAT,RETR)'
  *
  *
  SIABT    DC       CL8'ABT' ------------------------------------------- (42)
  *
  SINAME   DC       CL8'NAME' ------------------------------------------ (43)
           PRINT    NOGEN
  *
  *
  *                 EIN-AUSGABEBEREICHE AR
  *
           DS       0F
  MSATZ    DS       0CL74                                           }
  MPNUM    DS       F                                               }
  MNAME    DS       CL20                                            }
  MVORNAME DS       CL10                                            }
  MWOHNORT DS       CL10                                            }
  MABTLNG  DS       CL5                                             }
  MSTRASSE DS       CL22                                            }
           DS       CL3                                             } ---- (44)
  *                                                                 }
  PSATZ    DS       0CL45                                           }
  PAKTION  DS       CL1                                             }
  PPNUM    DS       CL6                                             }
  PNAME    DS       CL20                                            }
  PDATUM   DS       CL10                                            }
  PZEIT    DS       CL8                                             }
  *
  *                 TERMINAL-EIN-AUSGABEBEREICHE
  *                                                                 }
  TSATZ    DS       0CL78                                           }
  TPNUM    DS       CL6                                             }
           DC       CL1' '                                          }
  TABTLNG  DS       CL5                                             }
           DC       CL1' '                                          }
  TNAME    DS       CL20                                            } ---- (45)
           DC       CL1' '                                          }
  TVORNAME DS       CL10                                            }
           DC       CL1' '                                          }
  TWOHNORT DS       CL10                                            }
           DC       CL1' '                                          }
  TSTRASSE DS       CL22                                            }
  *
```

```
ERRLINE  DS       0CL21
         DC       CL13'LEASY-FEHLER '
ERRKODE  DS       CL8
*
OPLINE   DS       0CL63                                               (46)
         DC       CL23'GEWAEHLTE OPERATION: '
LASTOP   DS       CL4
         DC       CL20', GEWAEHLTE DATEI: '
LASTDB   DS       CL16
*
AUSGABE1 DC       CL44'BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:'
AUSGABE2 DC       CL30'   I..MITARBEITER EINFUEGEN'
AUSGABE3 DC       CL29'   D..MITARBEITER LOESCHEN'
AUSGABE4 DC       CL30'   L..MITARBEITER AUFLISTEN'
AUSGABE5 DC       CL24'   E..PROGRAMM BEENDEN'
*
MITARBEG DC       CL51'BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT *
               EIN'
ENDEING  DC       CL28'(*END: ENDE DER EINGABE)'
TAUS     DC       CL79' PERSNR ABTLG NAME                 VORNAME    WO*
               HNORT    STRASSE'
TSTARS   DC       CL79' ****** ***** ******************** ********** ***
               ******** **********************'
*
ALGEMEIN DC       CL27'BITTE ERROR-CODE BEACHTEN'
*
TEXTNUME DC       CL30'PERSONALNUMMER NICHT NUMERISCH'
*
         DS       0H
EINB     DS       0CL260
EINLEN   DS       CL2
         DC       CL2' '
EINTEXT  DS       CL256
*
AUSB     DS       0CL260
AUSLEN   DS       CL2
         DC       CL3' '
AUSTEXT  DS       CL255
*
TIAMERRL DS       0CL22
         DC       AL2(22)
         DC       CL3' '
         DC       CL17'TIAM-MACRO-ERROR'
*
*
*                 DATENFELD FUER GTIME
*
GTIMEL   GTIME    MF=L,DATE=YES,TOD=YES
```

```
GTIMED   GTIME    MF=D
PERSDAT  CSECT
*
*
*                 SONSTIGE FELDER
*
DOWO     DS       D
*
AENTRY   DS       F
*
SAVE     DS       18F
*
ENDKZ    DC       CL4'*END'
*
*
R0       EQU      0
R1       EQU      1
R2       EQU      2
R3       EQU      3
R4       EQU      4
R5       EQU      5
R6       EQU      6
R7       EQU      7
R8       EQU      8
R9       EQU      9
R10      EQU      10
R11      EQU      11
R12      EQU      12
R13      EQU      13
R14      EQU      14
R15      EQU      15
*
*
         END      ANF
               ='LOESCHEN ? (J/N)'
               ='SORTIERT NACH PERSNR (P), NAMEN (N) ODER ABTEILUNG (A)
               =CL8' '
               ='L002'
               ='L006'
               ='L001'
               ='SATZ NICHT VORHANDEN'
               ='L003'
               =Y(30)
               =Y(49)
               =Y(35)
               =Y(34)
               =Y(29)
               =Y(56)
```

```
=Y(33)
=Y(84)
=Y(27)
='SATZ BEREITS VORHANDEN'
=Y(18)
=Y(55)
='BITTE GEBEN SIE DIE PERSONALNUMMER (6-STELLIG) EIN'
=Y(25)
=Y(83)
=Y(21)
=Y(61)
=Y(26)
=Y(68)
=Y(32)
='NAME LEASY-DATEIKATALOG ?'
='SATZ GESPERRT'
=X'9906021124049467' CONSISTENCY CONSTANT FOR AID
```

Explanation

(1) The name of the LEASY catalog to be edited must be entered.

(2) The input is supplied to the LEA@CAT field, which is defined in the catalog information CAT (see step (35)).

(3) Macro call for the CATD operation. The RE communication area (see step (34)) and the catalog information CAT (see step (35)) are transferred.

(4) Macro call with the OPFL operation. The files defined with DB4OPFL (see step (39)) are physically opened. For DB4 format, the OPE-OM field in the RE area must be set to X'FF'.

(5) Messages are output on the data display terminal requesting the user to select the action to be performed. Depending on the user’s choice, the program branches to the appropriate section.

(6) This continues until the user enters the end criterion.

(7) The macro call with the CLFL operation closes the files after the end criterion is entered.

(8) This macro call opens the INSERT transaction. The files defined with DB4OPTRU (see step (40)) are logically opened.

(9) Messages on the data display terminal informing the user of the input format.

(10) The input is supplied to the TSATZ area. Provided an end criterion is not entered and the input is numeric, the contents of TSATZ are transferred to the AR area MSATZ of the file MITABDAT.

(11) This macro call writes the data in the AR area MSATZ to the file MITABDAT assigned using DBMDAT (see step (38)).

(12) If the specified record already exists or is locked, appropriate messages are output.

(13) If the INSERT transaction is executed without errors, the program branches to protocol record output PROTWRT.

(14) If the end criterion is entered on the data display terminal or if an error occurs while inserting a record, the INSERT transaction is terminated by issuing a macro call with the CLTR operation.

(15) The DELETE transaction is opened. The files defined with DB4OPTRU (see step (40)) are logically opened.

(16) The personnel number of the employee to be deleted must be entered. When converted to binary format (4 bytes), this forms the primary key of the record to be deleted.

(17) The input is supplied to the TPNUM field of the TSATZ area. Provided an end criterion is not entered and the input is numeric, the contents of the TPNUM field of the TSATZ area are transferred to the MPNUM field of the AR area MSATZ.

(18) This macro call is used to read the record whose primary key is stored in the MSATZ record zone from the MITABDAT file and lock it.

(19) The record contents just read into the record zone are transferred to the TSATZ area and output on the data display terminal.

(20) A message is output on the data display terminal requesting the user to confirm whether or not the record is to be deleted. The next program step depends on the user’s response.

(21) The record just read and locked is deleted from the MITABDAT file by issuing a macro call with the DLET operation.

(22) If the end criterion is entered on the data display terminal or if an error occurs while deleting a record, the DELETE transaction is terminated by issuing a macro call with the CLTR operation.

(23) The TSATZ area is cleared.

(24) The LIST transaction is opened. The file defined with DB4OPTRL (see step (41)) is logically opened.

(25) This message requests the user to select the sort criterion for the records to be output.

(26) Depending on the user’s specification, either the name of the corresponding secondary key to which the program navigates in the next SETL operation is written in the L@@SI field of the SI area, or the SI field is overwritten with blanks in which case the program navigates to the primary key.

(27) If the MITABSATZ input/output area is cleared, the program navigates to the start of the file in the next SETL operation.

(28) The macro call with the SETL operation enables the program to navigate to the start of the file.

(29) This macro call reads the next record in the ascending primary or secondary key sequence (depending on the contents of the SI field) and transfers this to the AR area MSATZ. The program then navigates to the new record.

(30) The contents of the AR area MSATZ are written to the TSATZ area and output on the data display terminal. The new record is then read.

(31) If an error occurs when navigating or reading or if the end of the file is reached, the LIST transaction is terminated by issuing a macro call with the CLTR operation.

(32) If a record was inserted or deleted, the program branches to protocol record output PROTWRT. Here, the contents of the MPNUM and MNAME fields of MITABSATZ are transferred to the corresponding fields of the AR area PSATZ of the PROTDAT file. The date and time are also entered in this area. The contents of the AR area PSATZ area are inserted in the PROTDAT file by issuing a macro call with the INSR operation.

(33) If an error occurs in a macro call, the program branches to this point. The error codes stored by LEASY in the RE communication area or the last operation code and file name are output.

(34) This macro call generates the RE communication area.

(35) This macro call generates the area for the catalog information CAT.

(36) This macro call generates the field selection operand FA.

(37) This macro call generates the area for the secondary index SI.

(38) The MITABDAT and PROTDAT files are defined in the file format DB1 so that they can be accessed individually.

(39) All files to be opened in the file format DB4 are assigned together with their OPEN mode:

MITABDAT in OPEN mode 4: INOUT,SHARUPD
PROTDAT in OPEN mode 9: EXTEND

(40) The files accessed in the UPDT transaction are assigned:

MITABDAT in USAGE mode UPDT
PROTDAT in USAGE mode EXLD

(41) The MITABDAT file accessed in the LIST transaction in USAGE mode RETR is assigned.

(42) The secondary key ABT is defined.

(43) The secondary key NAME is defined.

(44) The AR input/output area is defined with the name MSATZ for the MITABDAT file and with the name PSATZ for the PROTDAT file.

(45) The TSATZ area for input/output to/from the data display terminal is defined.

(46) Definitions for output in the event of an error.

12.3 Trace listings

12.3.1 Trace listing 1

This trace listing shows the sequence of the PERSDAT user program (see the COBOL program starting on page 323 and the Assembler program starting on page 338) in conjunction with the LEASY-CATALOG, LEASY-MAINTASK, LEASY-MASTER and LEASY-RECONST utilities. The reconstruction of AIM file generations is carried out by the user.

```
/start-leasy-catalog ------------------------------------------------------------  (1)
%  BLS0523 ELEMENT 'CATALOG', VERSION '06.2A00', TYPE 'L' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-CATALOG', VERSION '06.2A00' OF '2006-03-08 01:27:31' LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS RESERVED
%  LEA0101 LEASY CATALOG PROGRAM VERSION V6.2A STARTED
*CAT LCAT,TYP=N,PAS=C'LCAT'  ----------------------------------------------------  (2)
*FIL MITABDAT,AIM=Y,KEY=(ABT,45,5),KEY=(NAME,5,20),SHA=Y,RECFORM=F,  -
RECSIZE=74,KEYPOS=1,KEYLEN=4,SPACE=(12,3)  --------------------------------------  (3)
*FIL PROTDAT,LEA=T,FCBTYPE=SAM,RECFORM=F,RECSIZE=45,SPACE=(12,12)  --------------  (4)
*INF ,A  ------------------------------------------------------------------------  (5)
%   DNAME=MITABDAT
%   FNAME=LCAT.MITABDAT
0000000012 :SPVS:$USER.LCAT.MITABDAT
  ------------------------------- HISTORY      -------------------------------
  CRE-DATE   = 2006-04-21  ACC-DATE   = 2006-04-21  CHANG-DATE = 2006-04-21
  CRE-TIME   =   07:35:01  ACC-TIME   =   07:35:01  CHANG-TIME =   07:35:01
  ACC-COUNT  = 1           S-ALLO-NUM = 0
  ------------------------------- SECURITY     -------------------------------
  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
  USER-ACC   = ALL-USERS   ACCESS     = WRITE       ACL        = NO
  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2006-04-21
  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
  SP-REL-LOCK= NO
  ------------------------------- BACKUP       -------------------------------
  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
  MIGRATE    = ALLOWED
  ------------------------------- ORGANIZATION -------------------------------
  FILE-STRUC = ISAM        BUF-LEN    = STD(1)      BLK-CONTR  = DATA (2K)
  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
  REC-FORM   = (F,N)       REC-SIZE   = 74
  KEY-LEN    = 4           KEY-POS    = 1
  AVAIL      = *STD
  ------------------------------- ALLOCATION   -------------------------------
  SUPPORT    = PUB         S-ALLOC    = 3           HIGH-US-PA = 2
  EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
      1       SPVS.0       D3435
  NUM-OF-EXT = 1
:SPVS: PUBLIC:      1 FILE  RES=        12 FRE=        10 REL=          9 PAGES
```

```
0000000012 :SPVS:$USER.LCAT.MITABDAT-SI
  ------------------------------- HISTORY       -------------------------------
  CRE-DATE   = 2006-04-21  ACC-DATE   = 2006-04-21  CHANG-DATE = 2006-04-21
  CRE-TIME   =   07:35:01  ACC-TIME   =   07:35:01  CHANG-TIME =   07:35:01
  ACC-COUNT  = 1           S-ALLO-NUM = 0
  ------------------------------- SECURITY      -------------------------------
  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
  USER-ACC   = ALL-USERS   ACCESS     = WRITE       ACL        = NO
  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2006-04-21
  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
  SP-REL-LOCK= NO
  ------------------------------- BACKUP        -------------------------------
  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
  MIGRATE    = ALLOWED
  ------------------------------- ORGANIZATION -------------------------------
  FILE-STRUC = ISAM        BUF-LEN    = STD(2)      BLK-CONTR  = DATA (2K)
  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
  REC-FORM   = (V,N)       REC-SIZE   = 0
  KEY-LEN    = 25          KEY-POS    = 5
  AVAIL      = *STD
  ------------------------------- ALLOCATION    -------------------------------
  SUPPORT    = PUB         S-ALLOC    = 4           HIGH-US-PA = 2
  EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
      1       SPVS.0       D3435
  NUM-OF-EXT = 1
:SPVS: PUBLIC:      1 FILE  RES=        12 FRE=        10 REL=         6 PAGES
%   LEASYTYPE=...S.......LOCK=.......NO
%   FCBTYPE=..ISAM.......PAD=........15
%   RECSIMAX=00074.......KEYLEN=....004
%   AIM=.......YES.......BIM=.......YES
%   WRPASS=.....NO.......RDPASS=.....NO
%   ROM=........NO
%   KEY=.(ABT     ,      (00045,005),YES,YES)        (001)
%   .....................HAS NO POINTERS IN SI FILE
%   KEY=.(NAME    ,      (00005,020),YES,YES)        (002)
%   .....................HAS NO POINTERS IN SI FILE
%   DNAME=PROTDAT
%   FNAME=LCAT.PROTDAT
0000000012 :SPVS:$USER.LCAT.PROTDAT
  ------------------------------- HISTORY       -------------------------------
  CRE-DATE   = 2006-04-21  ACC-DATE   = 2006-04-21  CHANG-DATE = 2006-04-21
  CRE-TIME   =   07:35:01  ACC-TIME   =   07:35:01  CHANG-TIME =   07:35:01
  ACC-COUNT  = 1           S-ALLO-NUM = 0
  ------------------------------- SECURITY      -------------------------------
  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
  USER-ACC   = OWNER-ONLY  ACCESS     = WRITE       ACL        = NO
  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2006-04-21
  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
  SP-REL-LOCK= NO
  ------------------------------- BACKUP        -------------------------------
  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
  MIGRATE    = ALLOWED
```

```
  ------------------------------- ORGANIZATION --------------------------------
  FILE-STRUC = SAM         BUF-LEN    = STD(1)      BLK-CONTR  = DATA
  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
  REC-FORM   = (F,N)       REC-SIZE   = 45
  AVAIL      = *STD
  ------------------------------- ALLOCATION   -------------------------------
  SUPPORT    = PUB         S-ALLOC    = 12          HIGH-US-PA = 0
  EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
      1       SPVS.0       D3435
  NUM-OF-EXT = 1
:SPVS: PUBLIC:      1 FILE  RES=        12 FRE=        12 REL=         9 PAGES
%   LEASYTYPE=...T.......LOCK=.......NO
%   FCBTYPE=...SAM.......RECSIMAX=00045
%   AIM=........NO.......BIM=.......YES
%   WRPASS=.....NO.......RDPASS=.....NO
%   ROM=........NO
*END  --------------------------------------------------------------------------   (6)
%  LEA0110 NORMAL TERMINATION OF LEASY CATALOG PROGRAM
/show-file e.mtsk.1  -------------------------------------------------------------   (7)

/SET-LOGON-PAR JOB-NAME=MAINTASK
/ASS-SYSLST  O.MTSK.1
/START-LEASY-MAINTASK ------------------------------------------------------------   (8)
CAT=LCAT  ------------------------------------------------------------------------   (9)
FILES=2  ------------------------------------------------------------------------- (10)
LOG=Y  --------------------------------------------------------------------------- (11)
TRANS=3  ------------------------------------------------------------------------- (12)
TIM=40  -------------------------------------------------------------------------- (13)
END
/EXIT-JOB SYSTEM-OUTPUT=*NONE

end                                                         S*SOF+     1(      1)

%  SHO0500 ':SPVS:$USER.E.MTSK.1' CLOSED
/enter-job e.mtsk.1,resources=*par(cpu-lim=10)  --------------------------------- (14)
%  JMS0066 JOB 'MAINTASK' ACCEPTED ON 06-04-21 AT 07:42, TSN = 91DS
/show-user-stat
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
 MAINTASK 91DS 2 BATCH    9 210      0.4962      10 FSC  ------------------------- (15)
 DIATASK  91C7 3 DIALOG   0 210      0.4857    9000 FSC
%  SRO0376 NO RSO JOB OF TYPE 'T7' PRESENT
/show-file-attr lcat.  ----------------------------------------------------------- (16)
      18 :SPVS:$USER.LCAT.BIM#.001
      18 :SPVS:$USER.LCAT.BIM#.002
      18 :SPVS:$USER.LCAT.BIM#.003
       3 :SPVS:$USER.LCAT.LEADIAG
       0 :SPVS:$USER.LCAT.LEASYAIM (FGG)
      18 :SPVS:$USER.LCAT.LEASYCAT
      12 :SPVS:$USER.LCAT.MITABDAT
      12 :SPVS:$USER.LCAT.MITABDAT-SI
      12 :SPVS:$USER.LCAT.PROTDAT
:SPVS: PUBLIC:      9 FILES RES=       111 FRE=        92 REL=        69 PAGES
```

```
/start-exe persdat ------------------------------------------------------------  (17)
%  BLS0500 PROGRAM 'C.PERSDAT', VERSION ' ' OF '2006-03-27' LOADED
NAME LEASY-DATEIKATALOG ?
*LCAT
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*I
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*101721 AB212 SCHMITTINGER         FRANZ      MOOSBURG   ANDERLSTRASSE 3
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*062224 AB212 SCHNIEIDINGER        GERHARD    BRUECKENAU SORTSTRASSE 7
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*122510 AB21  EISBOSS              ROLF       MUENCHEN   WALTERWEG 25
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*071531 AP360 EXBOSS               MONIKA     MUTTERHAUS BACHUFERSTR. 17
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
**END
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*L
SORTIERT N.PSNR(P),NAME(N) ODER ABT(A)?
*N
122510 AB21  EISBOSS              ROLF       MUENCHEN   WALTERWEG 25
071531 AP360 EXBOSS               MONIKA     MUTTERHAUS BACHUFERSTR. 17
101721 AB212 SCHMITTINGER         FRANZ      MOOSBURG   ANDERLSTRASSE 3
062224 AB212 SCHNIEIDINGER        GERHARD    BRUECKENAU SORTSTRASSE 7
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
```

```
*E  ---------------------------------------------------------------------------- (18)
/show-user-stat  --------------------------------------------------------------- (19)
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
DIATASK  91C7 3 DIALOG   0 210      0.4857    9000 FSC
DIATASK  91C8 3 DIALOG   0 210      1.0013    9000 FSC
%  SPS0171 NO LOCAL SPOOLOUT JOB PRESENT
/copy-file lcat.mitabdat,save.mitabdat  ---------------------------------------- (20)
/start-leasy-catalog  ---------------------------------------------------------- (21)
%  BLS0523 ELEMENT 'CATALOG', VERSION '06.2A00', TYPE 'L' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-CATALOG', VERSION '06.2A00' OF '2006-03-08 01:27:31' LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS RESERVED
%  LEA0101 LEASY CATALOG PROGRAM VERSION V6.2A STARTED
*CAT LCAT,PAS=C'LCAT',INF=Y,CPC=SAVE  ------------------------------------------ (22)
%  LEA5101 LAST SESSION: NUMBER = 1, DATE = 2006-04-21, TIME = 07:42:32-S.  ----- (23)
%  LEA5002 SHADOW DIRECTORY NAME: $USER.SAVE SHADOW SUFFIX NAME:
%  LEA5105 DMS FILENAMES WITH CATID
%  LEA5108 ROM = NO IS SPECIFIED
*END
%  LEA0110 NORMAL TERMINATION OF LEASY CATALOG PROGRAM
/show-file e.mtsk.2  ----------------------------------------------------------- (24)

/SET-LOGON-PAR JOB-NAME=MAINTASK
/ASS-SYSLST  O.MTSK.2
/START-LEASY-MAINTASK
CAT=LCAT
FILES=2
LOG=Y
TRANS=3
TIM=40
ASP=(36,36)
END
/EXIT-JOB SYSTEM-OUTPUT=*NONE

END
/enter-job e.mtsk.2,resources=*par(cpu-lim=10)  -------------------------------- (25)
%  JMS0066 JOB 'MAINTASK' ACCEPTED ON 06-04-21 AT 07:58, TSN = 91EL
/show-user-stat
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
MAINTASK 91EL 2 BATCH    9 210      0.4792      10 FSC      ---------------------- (26)
DIATASK  91C7 3 DIALOG   0 210      0.4857    9000 FSC
DIATASK  91C8 3 DIALOG   0 210      1.1819    9000 FSC
%  SRO0376 NO RSO JOB OF TYPE 'T7' PRESENT
/show-file-attr lcat.leasyaim,inf=*all-attr
0000000000 :SPVS:$USER.LCAT.LEASYAIM (FGG)
  ------------------------------- HISTORY     -------------------------------
  CRE-DATE   = 2006-04-21  ACC-DATE   = NONE        CHANG-DATE = NONE
  CRE-TIME   =   07:42:32  ACC-TIME   = NONE        CHANG-TIME = NONE
  ACC-COUNT  = 0           S-ALLO-NUM = 0
  ------------------------------- SECURITY    -------------------------------
  READ-PASS  = YES         WRITE-PASS = NONE        EXEC-PASS  = NONE
  USER-ACC   = ALL-USERS   ACCESS     = WRITE       ACL        = NO
```

```
  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2006-04-21
  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
  SP-REL-LOCK= NO
  ------------------------------- BACKUP       -------------------------------
  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 0
  MIGRATE    = ALLOWED
  ------------------------------- GENERATION-INFO ----------------------------
  MAXIMUM    = 3           BASE-NUM   = 2           OVERFL-OPT = REUSE-VOL
  FIRST-GEN  = 1           LAST-GEN   = 2                                        (27)
:SPVS: PUBLIC:      1 FILE  RES=         0 FRE=         0 REL=         0 PAGES
/start-exe persdat                                                               (28)
%  BLS0500 PROGRAM 'C.PERSDAT', VERSION ' ' OF '2006-03-27' LOADED
NAME LEASY-DATEIKATALOG ?
*LCAT
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*L
SORTIERT N.PSNR(P),NAME(N) ODER ABT(A)?
*P
062224 AB212 SCHNIEIDINGER        GERHARD    BRUECKENAU SORTSTRASSE 7
071531 AP360 EXBOSS               MONIKA     MUTTERHAUS BACHUFERSTR. 17
101721 AB212 SCHMITTINGER         FRANZ      MOOSBURG   ANDERLSTRASSE 3
122510 AB21  EISBOSS              ROLF       MUENCHEN   WALTERWEG 25
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*D
BITTE GEBEN SIE DIE PERSONALNR. (6-STELLIG) EIN
(*END: ENDE DER EINGABE)
*071531
071531 AP360 EXBOSS               MONIKA     MUTTERHAUS BACHUFERSTR. 17
LOESCHEN ? (J/N)
*J
BITTE GEBEN SIE DIE PERSONALNR. (6-STELLIG) EIN
(*END: ENDE DER EINGABE)
**END
 BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*I
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*094711 AB212 NEUBOSS              HARDY      MUENCHEN   AM KNACKEPUNKT 1
```

```
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*141719 AB212 HINHOLD              ANTJE      MUENCHEN   RAUCHERSTEG 3
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*291018 DP212 WALDI                BERND      MUENCHEN   SCHWABENSTR.30
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
**END
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*L
SORTIERT N.PSNR(P),NAME(N) ODER ABT(A)?
*A
122510 AB21  EISBOSS              ROLF       MUENCHEN   WALTERWEG 25
062224 AB212 SCHNIEIDINGER        GERHARD    BRUECKENAU SORTSTRASSE 7
094711 AB212 NEUBOSS              HARDY      MUENCHEN   AM KNACKEPUNKT 1
101721 AB212 SCHMITTINGER         FRANZ      MOOSBURG   ANDERLSTRASSE 3
141719 AB212 HINHOLD              ANTJE      MUENCHEN   RAUCHERSTEG 3
291018 DP212 WALDI                BERND      MUENCHEN   SCHWABENSTR.30
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*E ────────────────────────────────────────────────────────────────────────────── (29)
*delete-file lcat.mitabdat ────────────────────────────────────────────────────── (30)
/start-exe c.persdat ──────────────────────────────────────────────────────────── (31)
%  BLS0500 PROGRAM 'C.PERSDAT', VERSION ' ' OF '2006-03-27' LOADED
NAME LEASY-DATEIKATALOG ?
*LCAT ─────────────────────────────────────────────────────────────────────────── (32)
LEASY-FEHLER 99ALDD33
GEWAEHLTE OPERATION:   OPFL, GEWAEHLTE DATEI:  MITABDAT    4
BITTE ERROR-CODE BEACHTEN
/help-msg dms0d33 ─────────────────────────────────────────────────────────────── (33)
%  DMS0D33 SPECIFIED FILE NOT CATALOGED.
%  ? The requested file has not been cataloged in the system.
%    For the file no catalog entry could be found.
%  ! Correct the error and try again.
/start-leasy-reconst ──────────────────────────────────────────────────────────── (34)
%  BLS0523 ELEMENT 'RECONST', VERSION '06.2A00', TYPE 'L' FROM LIBRARY
':50SH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-RECONST', VERSION '06.2A00' OF '2006-03-08 01:28:19' LOADED
```

```
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS RESERVED
%  LEA0401 LEASY AFTER IMAGE PROGRAM (RECONST) VERSION V6.2A STARTED
*CAT LCAT,FRO=1,COP=Y ——————————————————————————————————————————————————————————  (35)
*SES FRO=2 —————————————————————————————————————————————————————————————————————  (36)
*MOD SIU=N —————————————————————————————————————————————————————————————————————  (37)
*END ———————————————————————————————————————————————————————————————————————————  (38)
SESS#=00002 TSN#=91EL D=2006-04-21 T=07:58:43-S ————————————————————————————————  (39)
%  LEA0410 NORMAL TERMINATION OF LEASY AFTER IMAGE PROGRAM  (RECONST)
/copy-file save.mitabdat,lcat.mitabdat —————————————————————————————————————————  (40)
/start-exe persdat —————————————————————————————————————————————————————————————  (41)
%  BLS0500 PROGRAM 'C.PERSDAT', VERSION ' ' OF '2006-03-27' LOADED
NAME LEASY-DATEIKATALOG ?
*LCAT
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
  I..MITARBEITER EINFUEGEN
  D..MITARBEITER LOESCHEN
  L..MITARBEITER AUFLISTEN
  E..PROGRAMM BEENDEN
*L
SORTIERT N.PSNR(P),NAME(N) ODER ABT(A)?
*N —————————————————————————————————————————————————————————————————————————————  (42)
122510 AB21  EISBOSS             ROLF       MUENCHEN   WALTERWEG 25
141719 AB212 HINHOLD             ANTJE      MUENCHEN   RAUCHERSTEG 3
094711 AB212 NEUBOSS             HARDY      MUENCHEN   AM KNACKEPUNKT 1
101721 AB212 SCHMITTINGER        FRANZ      MOOSBURG   ANDERLSTRASSE 3
062224 AB212 SCHNIEIDINGER       GERHARD    BRUECKENAU SORTSTRASSE 7
291018 DP212 WALDI               BERND      MUENCHEN   SCHWABENSTR.30
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
```

Printout of the selected function of the LEASY-RECONST utility (43)

This printout is generated automatically for each reconstruction run.

```
LEASY AFTER-IMAGE-RECONSTRUCTION       DATE OF RECONSTRUCTION: 2006-04-21  TIME:
08:08:40-S
SELECTED PARAMETERS:
--------------------

LEASY CATALOG  (*CAT):     SELECTED CATALOG NAME:............. :SPVS:$USER.LCAT
                           AFTER IMAGE FILE GENERATION NUMBER: 0001 UNTIL 0002
                           UPDATING SHADOW FILES:............. YES

OPERATION MODE (*MOD):     PRINT:............................. NORMAL
                           UPDATING SHADOW FILES:............. YES
                           UPDATING SHADOW SI-FILES:.......... NO
                           TRANSACTIONS SELECTED:............. ALL
                           UNLOAD CLASS-5-MEMORY:............. YES
                           SET FREE CMMAIN FOR RUNTIME-SYSTEM: USE OF CMMAIN NOT
CHANGED

LIST REPORT    (*REP):     REQUESTED LENGTH OF RECORD OUTPUT:. SHORT
                           RECORD EXTRACTION:................. NOT SELECTED
                           LIST INVALID RECORDS:.............. YES
                           RECORD SELECTION:.................. ALL RECORDS
                           LIST USER-INFORMATION:............. NO
                           LIST PROTOCOL RECORDS:............. NO

DATE FILTER    (*DAT):     FROM DATE (YYYY-MM-DD):............ START OF FILE
                           TO   DATE (YYYY-MM-DD):............ END   OF FILE

SESSION FILTER (*SES):     FROM SESSION-NUMBER:...............     2
                           TO   SESSION-NUMBER:............... END   OF FILE
                           LAST TRANSACTION:.................. END   OF TO-SESSION

FILE FILTER    (*FIL):     NOT SPECIFIED

RANGE FILTER   (*RAN):     NOT SPECIFIED
```

Printout of the reconstruction log ——— (44)

```
LEASY AFTER IMAGE RECONSTRUCTION, AIMFILE=:SPVS:$USER.LCAT.LEASYAIM(*0002) NEW PAMBLOCK-LINK: BLOCK# = 0000001

OP  XYS POS SESSION TRANS ITR TSN  FILE                                   TIME        RECORD
------------------------------------------------------------------------------------------------------------------
SESS     4210 00002     0   0 91EL                                        07:58:43-S  D=2006-04-21
CATD     4270 00002     0   0 91C8                                        08:01:56-S  T=91C8 U=USER P=C.PERSDA I=TSN-91C8
OPEN     4372 00002     0   0 91C8                                        08:01:56-S  MITABDAT:SPVS:$USER.LCAT.MITABDAT
OPTR     4429 00002     2   1 91C8                                        08:01:58-S  B=         H=         A=$DIALOG  #=
DLET     4468 00002     2   1 91C8 :SPVS:$USER.SAVE.MITABDAT              08:01:58-S  KEY=X'0001176B'
CLTR     4500 00002     2   1 91C8                                        08:01:58-S
OPTR     4557 00002     3   1 91C8                                        08:01:58-S  B=         H=         A=$DIALOG  #=
STOR     4666 00002     3   1 91C8 :SPVS:$USER.SAVE.MITABDAT              08:02:01-S  KEY=X'000171F7'
STOR     4775 00002     3   1 91C8 :SPVS:$USER.SAVE.MITABDAT              08:02:01-S  KEY=X'00022997'
STOR     4884 00002     3   1 91C8 :SPVS:$USER.SAVE.MITABDAT              08:02:01-S  KEY=X'000470CA'
CLTR     4916 00002     3   1 91C8                                        08:02:01-S
CLOS     4951 00002     0   0 91C8                                        08:02:02-S
CATD     5011 00002     0   0 91C8                                        08:04:20-S  T=91C8 U=USER P=C.PERSDA I=TSN-91C8
ENDA     5083 00002     0   0 91EL                                        08:07:08-S  D=2006-04-21
```

Log of the ENTER procedure E.MTSK.1 ——— (45)

```
/START-LEASY-MAINTASK
%  BLS0523 ELEMENT 'MAINTASK', VERSION '06.2A00', TYPE 'L' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-MAINTASK', VERSION '06.2A00' OF '2006-03-08 01:28:12'
LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS
RESERVED
%  LEA0301 LEASY MAINTASK VERSION V6.2A STARTED
CAT=LCAT
FILES=2
LOG=Y
TRANS=3
TIM=40
END
%  LEA5303 WARM/COLD START SUCCESSFUL
%  LEA5307 NEW LEASY SESSION CREATED: SESSION NUMBER = 00003, DATE = 2006-04-
21, TIME = 08:21:04-S
%  LEA5304 *LEASY MAINTASK :SPVS:$USER.LCAT INITIALIZATION COMPLETED
%  LEA0310 NORMAL TERMINATION OF LEASY MAINTASK BECAUSE OF CLOSE DOWN
FUNCTION
```

Log of the ENTER procedure E.MTSK.2 ——— (46)

```
/START-LEASY-MAINTASK
%  BLS0523 ELEMENT 'MAINTASK', VERSION '06.2A00', TYPE 'L' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-MAINTASK', VERSION '06.2A00' OF '2006-03-08 01:28:12'
LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS
RESERVED
%  LEA0301 LEASY MAINTASK VERSION V6.2A STARTED
CAT=LCAT
FILES=2
LOG=Y
TRANS=3
TIM=40
ASP=(36,36)
END
%  LEA5303 WARM/COLD START SUCCESSFUL
%  LEA5307 NEW LEASY SESSION CREATED: SESSION NUMBER = 00002, DATE = 2006-04-
21, TIME = 08:36:20-S
%  LEA5304 *LEASY MAINTASK :SPVS:$USER.LCAT INITIALIZATION COMPLETED
%  LEA5003 START OF AIM FILE GENERATION SWITCHING ON 2006-04-21 AT 08:38:06-S
%  LEA5004 AIM FILE GENERATION SWITCHING SUCCESSFUL
%  LEA0310 NORMAL TERMINATION OF LEASY MAINTASK BECAUSE OF CLOSE DOWN
FUNCTION
```

Log of the LEASY-MASTER utility routine

```
/start-leasy-master
%  BLS0523 ELEMENT 'MASTER', VERSION '06.2A' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-MASTER', VERSION '06.2A00' OF '2006-03-08 01:28:19' LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS RESERVED
%  LEA0501 LEASY MASTER PROGRAM VERSION V6.2A STARTED
```

```
LEASY MASTER PROGRAM VERSION V6.2A................SCREEN 001: MAINTASK SELECTION
................................................................................
PLEASE TYPE IN NAME OF LEASY DIRECTORY..........................................
(*END=END OF PROGRAM)...........................................................
*LCAT                                                                             -(47)
................................................................................
PLEASE ENTER PASSWORD
*C'LCAT'                                                                          -(48)
```

```
LEASY MASTER PROGRAM VERSION V6.2A.............SCREEN 003: GENERAL INFORMATION
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
CURRENT SESSION NUMBER:..........00001
CMMAIN STATUS:...................NORMAL WORKING
CMMAIN CONTROL:..................NO CONTROL FUNCTION IS ACTIV
USE BEFORE IMAGE LOGGING:........YES
USE AFTER IMAGE LOGGING:.........YES, AIM GEN#=0001                               -(49)
NUMBER OF ACTIVE TASKS:..........000 OF MAX. 003
NUMBER OF ACTIVE TRANSACTIONS:...000 OF MAX. 003
NUMBER OF OPEN FILES:............000 OF MAX. 002
NUMBER OF ACT. TA APPLICATIONS:..000 OF MAX. 003
BUCKET POOL MEMORY SIZE:.........00029696 BYTES
SIZE OF ONE BUCKET IN POOL:......00001024 BYTES
NUMBER OF BUCKETS IN BUCKET POOL:00000029
USED BUCKETS FOR LOCK ELEMENTS:..00000000
USED BUCKETS FOR TRANSACTIONS:...00000000 UNUSED
BUCKETS:..................00000029
NUMBER OF LOCKED DATA RECORDS:...00000000
NUMBER OF FREE LOCK ELEMENTS:....00000004
SYSLST PRINTOUT SWITCH IS SET:...ON
UPD. COMMANDS ON CMMAIN ALLOWED:.YES
FUNCTION SELECTION (OR BLANK=MAINTASK SELECTION; OR *END=END OF PROGRAM)
*CLOS                                                                             -(50)
```

```
LEASY MASTER PROGRAM VERSION V6.2A.............SCREEN 003: GENERAL INFORMATION
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
CURRENT SESSION NUMBER:..........00001
CMMAIN STATUS:...................NORMAL WORKING
CMMAIN CONTROL:..................TERMINATE MAINTASK
CMMAIN CONTROL:..................ACCEPT NO FURTHER TRANSACTIONS
USE BEFORE IMAGE LOGGING:........YES
USE AFTER IMAGE LOGGING:.........YES, AIM GEN#=0001
NUMBER OF ACTIVE TASKS:..........000 OF MAX. 003
NUMBER OF ACTIVE TRANSACTIONS:...000 OF MAX. 003
NUMBER OF OPEN FILES:............000 OF MAX. 002
NUMBER OF ACT. TA APPLICATIONS:..000 OF MAX. 003
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
FUNCTION SELECTION (OR +=CONTINUE; OR BLANK; OR *END)
**END                                                                              -(51)
```

```
%  LEA0510 NORMAL TERMINATION OF LEASY MASTER PROGRAM
/start-leasy-master
%  BLS0523 ELEMENT 'MASTER', VERSION '06.2A' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-MASTER', VERSION '06.2A00' OF '2006-03-08 01:28:19' LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS RESERVED
%  LEA0501 LEASY MASTER PROGRAM VERSION V6.2A STARTED
```

```
LEASY MASTER PROGRAM VERSION V6.2A................SCREEN 001: MAINTASK SELECTION
................................................................................
PLEASE TYPE IN NAME OF LEASY DIRECTORY..........................................
(*END=END OF PROGRAM)...........................................................
*LCAT                                                                              -(52)
................................................................................
PLEASE ENTER PASSWORD
*C'LCAT'                                                                           -(53)
```

```
LEASY MASTER PROGRAM VERSION V6.2A.............SCREEN 003: GENERAL INFORMATION
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
CURRENT SESSION NUMBER:..........00002
CMMAIN STATUS:...................NORMAL WORKING
CMMAIN CONTROL:..................NO CONTROL FUNCTION IS ACTIV
USE BEFORE IMAGE LOGGING:........YES
USE AFTER IMAGE LOGGING:.........YES, AIM GEN#=0002                                 -(54)
NUMBER OF ACTIVE TASKS:..........000 OF MAX. 003
NUMBER OF ACTIVE TRANSACTIONS:...000 OF MAX. 003
NUMBER OF OPEN FILES:............000 OF MAX. 002
NUMBER OF ACT. TA APPLICATIONS:..000 OF MAX. 003
BUCKET POOL MEMORY SIZE:.........00028672 BYTES
SIZE OF ONE BUCKET IN POOL:......00001024 BYTES
NUMBER OF BUCKETS IN BUCKET POOL:00000028
USED BUCKETS FOR LOCK ELEMENTS:..00000000
USED BUCKETS FOR TRANSACTIONS:...00000000
UNUSED BUCKETS:..................00000028
NUMBER OF LOCKED DATA RECORDS:...00000000
NUMBER OF FREE LOCK ELEMENTS:....00000003
SYSLST PRINTOUT SWITCH IS SET:...ON
UPD. COMMANDS ON CMMAIN ALLOWED:.YES
FUNCTION SELECTION (OR BLANK=MAINTASK SELECTION; OR *END=END OF PROGRAM)
*AIMI                                                                               -(55)
```

```
................................................................................
................................................................................
................................................................................
%  LEA5003 START OF AIM FILE GENERATION SWITCHING ON 2006-04-21 AT 08:38:06-S
%  LEA5004 AIM FILE GENERATION SWITCHING SUCCESSFUL
................................................................................
................................................................................
................................................................................
FUNCTION SELECTION (OR BLANK=MAINTASK SELECTION; OR *END=END OF PROGRAM)
*CLOS                                                                               -(56)
```

```
LEASY MASTER PROGRAM VERSION V6.2A.............SCREEN 003: GENERAL INFORMATION
...............................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
CURRENT SESSION NUMBER:..........00002
CMMAIN STATUS:...................NORMAL WORKING
CMMAIN CONTROL:..................TERMINATE MAINTASK
CMMAIN CONTROL:..................ACCEPT NO FURTHER TRANSACTIONS
USE BEFORE IMAGE LOGGING:........YES
USE AFTER IMAGE LOGGING:.........YES, AIM GEN#=0003                            -(57)
NUMBER OF ACTIVE TASKS:..........000 OF MAX. 003
NUMBER OF ACTIVE TRANSACTIONS:...000 OF MAX. 003
NUMBER OF OPEN FILES:............000 OF MAX. 002
NUMBER OF ACT. TA APPLICATIONS:..000 OF MAX. 003
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
FUNCTION SELECTION (OR +=CONTINUE; OR BLANK; OR *END)                          -(58)
**END
```

```
%  LEA0510 NORMAL TERMINATION OF LEASY MASTER PROGRAM
```

Explanation

(1) The LEASY-CATALOG utility routine is called.

(2) The LEASY catalog LCAT is created with the password C'LCAT'.

(3) The master file MITABDAT is created with the specified properties.

(4) The temporary file PROTDAT is created with the specified properties.

(5) The properties of all files contained in the catalog are to be output in full on the data display terminal.

(6) The LEASY-CATALOG utility routine is terminated.

(7) The ENTER file E.MTSK.1, which starts the main task, is output.

(8) LEASY-MAINTASK is started in the ENTER file.

(9) The LEASY catalog LCAT is assigned.

(10) It is possible to have two files open simultaneously.

(11) The session is conducted with AIM and BIM security.

(12) It is possible to process three transactions simultaneously.

(13) The program should wait up to 40 seconds for a session to become available.

(14) The E.MTSK.1 batch jobs are started. The main task that initializes the first LEASY session is started.

(15) The main task is started.

(16) The files contained in the LEASY catalog LCAT are listed.

(17) The user program PERSDAT is called.

(18) The user program is terminated.

(19) The LEASY session was terminated by the LEASY-MASTER utility using the CLOS function (see step (50)). For the main task log, see step (45).

(20) A shadow file with the name SAVE.MITABDAT is created for the MITABDAT personnel file.

(21) LEASY-CATALOG is called.
To allow for reconstruction of the shadow files at a later point in time, the CPC operand must be set before starting the main task.

(22) The naming system used when reconstructing shadow files is defined. SAVE is the catalog name in the shadow file name.

(23) The specification INF=YES in the *CAT statement causes the LEASY-CATALOG utility to output the session number, the date, the time of the last LEASY session, and the value of CPC.

(24) The ENTER file E.MTSK.2, which starts the main task, is output. In addition to the statements of the ENTER file E.MTSK.1 (see step (7)), the *ASP statement switches to the next AIM file generation.

(25) The E.MTSK.2 batch job is started. The main task that initializes the second LEASY session is started.

(26) The main task is started.

(27) The program switches from the first to the second AIM file generation.

(28) The user program PERSDAT is restarted.

(29) The user program is terminated.

(30) The primary file LCAT.MITABDAT is inadvertently deleted.

(31) The user program PERSDAT is called.

(32) An attempt to access the files with the user program causes an abort.

(33) The HELP-MSG command displays the cause of the error.

(34) After switching to the next AIM file generation (see steps (54) and (55)), the LEASY-RECONST utility is called.
When reconstructing the shadow files, the main task may be in the state *USE=N or *USE=R. A new main task need not be started.

(35) The LEASY catalog LCAT is assigned. All AIM file generations, from the first (FRO=1) to the most recent, must be used for reconstruction.
The shadow files are to be reconstructed (COP=Y).

(36) Reconstruction is to begin with session number 2.

(37) The SI files are not to be reconstructed.

(38) Statement input is concluded.

(39) Information on the reconstructed sessions is output. The reconstruction logs are created automatically (see steps (43) and (44)).

(40) The original file LCAT.MITABDAT is copied from the reconstructed shadow file SAVE.MITABDAT.

(41) The user program PERSDAT is called.

(42) The files are processed further.

(43) Printout of the selected functions of the LEASY-RECONST program.

(44) Printout of the reconstruction log.

(45) Main task log from the first session (see step (14)).

(46) Main task log from the second session (see step (25)).

(47) After the LEASY-MASTER utility is called, the LEASY catalog for which the MASTER functions are to be executed is assigned.

(48) The password C'LCAT' is requested.

(49) General information is output, indicating among other things that this description refers to the first AIM file generation.

(50) The main task is terminated using the CLOS function.

(51) The LEASY-MASTER utility is terminated with *END.

(52) After the LEASY-MASTER utility is called again, the LEASY catalog for which the MASTER functions are to be executed is assigned.

(53) The password C'LCAT' is requested.

(54) General information is output, indicating among other things that this description refers to the second AIM file generation.

(55) The program switches immediately to the next AIM file generation.

(56) The main task is terminated using the CLOS function.

(57) General information is output, indicating among other things that this description refers to the third AIM file generation.

(58) The LEASY-MASTER utility routine is terminated with *END.

12.3.2 Trace listing 2

This trace listing shows the sequence of the PERSDAT user program (see the COBOL program starting on page 323 and the Assembler program starting on page 338) in conjunction with the LEASY-CATALOG, LEASY-MAINTASK, LEASY-MASTER and LEASY-RECONST utilities. The reconstruction of AIM file generations is carried out automatically.

```
/start-leasy-catalog  ------------------------------------------------------------   (1)
%  BLS0523 ELEMENT 'CATALOG', VERSION '06.2A00', TYPE 'L' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-CATALOG', VERSION '06.2A00' OF '2006-03-08 01:27:31' LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS RESERVED
%  LEA0101 LEASY CATALOG PROGRAM VERSION V6.2A STARTED
*CAT LCAT,TYP=N,PAS=C'LCAT',CPC=SAVE  --------------------------------------------   (2)
*FIL MITABDAT,AIM=(Y,A),KEY=(ABT,45,5),KEY=(NAME,5,20),SHA=Y,RECFORM=F,-
RECSIZE=74,KEYPOS=1,KEYLEN=4,SPACE=(12,3)  ---------------------------------------   (3)
*FIL PROTDAT,LEA=T,FCBTYPE=SAM,RECFORM=F,RECSIZE=45,SPACE=(12,12)  ---------------   (4)
*INF ,A  -------------------------------------------------------------------------   (5)
%   DNAME=MITABDAT
%   FNAME=LCAT.MITABDAT
0000000012 :SPVS:$USER.LCAT.MITABDAT
  ------------------------------- HISTORY      -------------------------------
  CRE-DATE   = 2006-04-21  ACC-DATE   = 2006-04-21  CHANG-DATE = 2006-04-21
  CRE-TIME   =   08:45:13  ACC-TIME   =   08:45:13  CHANG-TIME =   08:45:13
  ACC-COUNT  = 1           S-ALLO-NUM = 0
  ------------------------------- SECURITY     -------------------------------
  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
  USER-ACC   = ALL-USERS   ACCESS     = WRITE       ACL        = NO
  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2006-04-21
  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
  SP-REL-LOCK= NO
  ------------------------------- BACKUP       -------------------------------
  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
  MIGRATE    = ALLOWED
  ------------------------------- ORGANIZATION -------------------------------
  FILE-STRUC = ISAM        BUF-LEN    = STD(1)      BLK-CONTR  = DATA (2K)
  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
  REC-FORM   = (F,N)       REC-SIZE   = 74
  KEY-LEN    = 4           KEY-POS    = 1
  AVAIL      = *STD
  ------------------------------- ALLOCATION   -------------------------------
  SUPPORT    = PUB         S-ALLOC    = 3           HIGH-US-PA = 2
  EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
      1       SPVS.0       D3435
  NUM-OF-EXT = 1
:SPVS: PUBLIC:      1 FILE  RES=        12 FRE=        10 REL=         9 PAGES
0000000012 :SPVS:$USER.LCAT.MITABDAT-SI
  ------------------------------- HISTORY      -------------------------------
  CRE-DATE   = 2006-04-21  ACC-DATE   = 2006-04-21  CHANG-DATE = 2006-04-21
  CRE-TIME   =   08:45:13  ACC-TIME   =   08:45:13  CHANG-TIME =   08:45:13
```

```
  ACC-COUNT  = 1           S-ALLO-NUM = 0
  ------------------------------- SECURITY     -------------------------------
  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
  USER-ACC   = ALL-USERS   ACCESS     = WRITE       ACL        = NO
  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2006-04-21
  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
  SP-REL-LOCK= NO
  ------------------------------- BACKUP       -------------------------------
  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
  MIGRATE    = ALLOWED
  ------------------------------- ORGANIZATION -------------------------------
  FILE-STRUC = ISAM        BUF-LEN    = STD(2)      BLK-CONTR  = DATA (2K)
  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
  REC-FORM   = (V,N)       REC-SIZE   = 0
  KEY-LEN    = 25          KEY-POS    = 5
  AVAIL      = *STD
  ------------------------------- ALLOCATION   -------------------------------
  SUPPORT    = PUB         S-ALLOC    = 4           HIGH-US-PA = 2
  EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
      1       SPVS.0       D3435
  NUM-OF-EXT = 1
:SPVS: PUBLIC:      1 FILE  RES=        12 FRE=        10 REL=         6 PAGES
%   LEASYTYPE=...S.......LOCK=.......NO
%   FCBTYPE=..ISAM.......PAD=........15
%   RECSIMAX=00074.......KEYLEN=....004
%   AIM=YES, AUTOM.......BIM=.......YES
%   WRPASS=.....NO.......RDPASS=.....NO
%   ROM=........NO
%   KEY=.(ABT     ,      (00045,005),YES,YES)        (001)
%   .....................HAS NO POINTERS IN SI FILE
%   KEY=.(NAME    ,      (00005,020),YES,YES)        (002)
%   .....................HAS NO POINTERS IN SI FILE
%   DNAME=PROTDAT
%   FNAME=LCAT.PROTDAT
0000000012 :SPVS:$USER.LCAT.PROTDAT
  ------------------------------- HISTORY      -------------------------------
  CRE-DATE   = 2006-04-21  ACC-DATE   = 2006-04-21  CHANG-DATE = 2006-04-21
  CRE-TIME   =   08:45:13  ACC-TIME   =   08:45:13  CHANG-TIME =   08:45:13
  ACC-COUNT  = 1           S-ALLO-NUM = 0
  ------------------------------- SECURITY     -------------------------------
  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
  USER-ACC   = OWNER-ONLY  ACCESS     = WRITE       ACL        = NO
  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2006-04-21
  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
  SP-REL-LOCK= NO
  ------------------------------- BACKUP       -------------------------------
  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
  MIGRATE    = ALLOWED
  ------------------------------- ORGANIZATION -------------------------------
  FILE-STRUC = SAM         BUF-LEN    = STD(1)      BLK-CONTR  = DATA
  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
  REC-FORM   = (F,N)       REC-SIZE   = 45
```

```
  AVAIL      = *STD
  -------------------------------- ALLOCATION   --------------------------------
  SUPPORT    = PUB         S-ALLOC    = 12          HIGH-US-PA = 0
  EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
      1       SPVS.1       D3435
  NUM-OF-EXT = 1
:SPVS: PUBLIC:      1 FILE  RES=        12 FRE=        12 REL=         9 PAGES
%   LEASYTYPE=...T.......LOCK=.......NO
%   FCBTYPE=...SAM.......RECSIMAX=00045
%   AIM=........NO.......BIM=.......YES
%   WRPASS=.....NO.......RDPASS=.....NO
%   ROM=........NO
*END ---------------------------------------------------------------------------   (6)
%  LEA0110 NORMAL TERMINATION OF LEASY CATALOG PROGRAM
/copy-file lcat.mitabdat,save.mitabdat -----------------------------------------   (7)
/copy-file lcat.mitabdat-si,save.mitabdat-si -----------------------------------   (8)
/show-file e.mtsk.3 ------------------------------------------------------------   (9)

/SET-LOGON-PAR JOB-NAME=MAINTASK
/ASS-SYSLST O.MSTK.3
/START-LEASY-MAINTASK ----------------------------------------------------------  (10)
CAT=LCAT -----------------------------------------------------------------------  (11)
FILES=2 ------------------------------------------------------------------------  (12)
LOG=Y,M ------------------------------------------------------------------------  (13)
AUT=Y --------------------------------------------------------------------------  (14)
REN=ENTER-JOB E.RECONST.AUT,JOB-NAME=RECOAUT -----------------------------------  (15)
TRANS=3 ------------------------------------------------------------------------  (16)
TIM=40 -------------------------------------------------------------------------  (17)
END
/EXIT-JOB SYSTEM-OUTPUT=*NONE
  end                                                      S*SOF+     1(      1)

/enter-job e.mtsk.3,resources=*par(cpu-lim=10) ---------------------------------  (18)
%  JMS0066 JOB 'MAINTASK' ACCEPTED ON 06-04-21 AT 08:50, TSN = 91KN
/show-user-stat
 NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
  DIATASK  91C7 3 DIALOG   0 210      3.6877    9000 FSC
  DIATASK  91C8 3 DIALOG   0 210      2.9012    9000 FSC
  DIATASK  91C9 3 DIALOG   0 210      3.3986    9000 FSC
  MAINTASK 91KN 2 BATCH    9 210      0.5239      10 FSC -----------------------  (19)
  RECOAUT  91KP 2 BATCH    9 210      0.4036     200 FSC  ----------------------  (20)
  %  SPS0171 NO LOCAL SPOOLOUT JOB PRESENT
  %  SRO0376 NO RSO JOB OF TYPE 'T7' PRESENT
  %  SCP1095 DPRINTSV WARNING : SOME DPRINT PRINT-JOBS CANNOT BE DISPLAYED
 /show-file-attr lcat. ---------------------------------------------------------  (21)
     18 :SPVS:$USER.LCAT.BIM#.001
     18 :SPVS:$USER.LCAT.BIM#.002
     18 :SPVS:$USER.LCAT.BIM#.003
      3 :SPVS:$USER.LCAT.LEADIAG
      0 :SPVS:$USER.LCAT.LEASYAIM (FGG)
     18 :SPVS:$USER.LCAT.LEASYCAT
     12 :SPVS:$USER.LCAT.MITABDAT
```

```
     12 :SPVS:$USER.LCAT.MITABDAT-SI
     12 :SPVS:$USER.LCAT.PROTDAT
:SPVS: PUBLIC:      9 FILES RES=       111 FRE=        92 REL=        69 PAGES
/start-exe persdat ------------------------------------------------------------- (22)
%  BLS0500 PROGRAM 'C.PERSDAT', VERSION ' ' OF '2006-03-27' LOADED
NAME LEASY-DATEIKATALOG ?
*LCAT
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*I
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*094711 AB212 NEUBOSS              HARDY      MUENCHEN   AM KNACKEPUNKT 1
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*151921 DP212 BUSCHBADER           REINHARD   MUENCHEN   AM WEICH 7
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*291018 AB212 WALDI                BERND      MUENCHEN   SCHWABENSTR. 30
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                 VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
**END
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*L
SORTIERT N.PSNR(P),NAME(N) ODER ABT(A)?
*N
151921 DP212 BUSCHBADER           REINHARD   MUENCHEN   AM WEICH 7
094711 AB212 NEUBOSS              HARDY      MUENCHEN   AM KNACKEPUNKT 1
291018 AB212 WALDI                BERND      MUENCHEN   SCHWABENSTR. 30
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*E ----------------------------------------------------------------------------- (23)
/show-user-stat
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
```

```
DIATASK  91C7 3 DIALOG   0 210      4.2662    9000 FSC --------------------------- (24)
DIATASK  91C8 3 DIALOG   0 210      2.9012    9000 FSC
DIATASK  91C9 3 DIALOG   0 210      3.5208    9000 FSC
NAME      TSN TYPE  PRI   SIZE  COPIES PRSIZE   RTSN OPT
RECOAUT  91K2 4 FT  240      2       0      0   91KP
%  SRO0376 NO RSO JOB OF TYPE 'T7' PRESENT
%  SCP1095 DPRINTSV WARNING : SOME DPRINT PRINT-JOBS CANNOT BE DISPLAYED
/enter-job e.mtsk.3,resources=*par(cpu-lim=10) ---------------------------------- (25)
%  JMS0066 JOB 'MAINTASK' ACCEPTED ON 06-04-21 AT 08:59, TSN = 91K5
/show-user-stat
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
MAINTASK 91K5 2 BATCH    9 210      0.4825      10 FSC --------------------------- (26)
DIATASK  91C7 3 DIALOG   0 210      4.3008    9000 FSC
DIATASK  91C8 3 DIALOG   0 210      2.9012    9000 FSC
RECOAUT  91K6 2 BATCH    9 210      0.4015     200 FSC --------------------------- (27)
DIATASK  91C9 3 DIALOG   0 210      3.5208    9000 FSC
%  SPS0171 NO LOCAL SPOOLOUT JOB PRESENT
%  SRO0376 NO RSO JOB OF TYPE 'T7' PRESENT
%  SCP1095 DPRINTSV WARNING : SOME DPRINT PRINT-JOBS CANNOT BE DISPLAYED
/start-exe persdat ---------------------------------------------------------------- (28)
%  BLS0500 PROGRAM 'C.PERSDAT', VERSION ' ' OF '2006-03-27' LOADED
NAME LEASY-DATEIKATALOG ?
*LCAT
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*I
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*281731 AB212 BLONDIE             OTTILIE    MUENSTER   SUDSCHWEDENW. 22
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
*302015 DP212 WUPPI               HARTMUT    MUENCHEN   WALDTALSTR. 19
BITTE GEBEN SIE DIE DATEN IM ANGEZEIGTEN FORMAT EIN
(*END: ENDE DER EINGABE)
 PERSNR ABTLG NAME                VORNAME    WOHNORT    STRASSE
 ****** ***** ******************** ********** ********** **********************
**END
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*L
SORTIERT N.PSNR(P),NAME(N) ODER ABT(A)?
*A
```

```
094711 AB212 NEUBOSS              HARDY      MUENCHEN   AM KNACKEPUNKT 1
281731 AB212 BLONDIE              OTTILIE    MUENSTER   SUDSCHWEDENW. 22
291018 AB212 WALDI                BERND      MUENCHEN   SCHWABENSTR. 30
151921 DP212 BUSCHBADER           REINHARD   MUENCHEN   AM WEICH 7
302015 DP212 WUPPI                HARTMUT    MUENCHEN   WALDTALSTR. 19
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*E ------------------------------------------------------------------------------  (29)
/delete-file lcat.mitabdat ------------------------------------------------------  (30)
/start-exe c.persdat ------------------------------------------------------------  (31)
%  BLS0500 PROGRAM 'C.PERSDAT', VERSION ' ' OF '2006-03-27' LOADED
NAME LEASY-DATEIKATALOG ?
*LCAT --------------------------------------------------------------------------  (32)
LEASY-FEHLER 99ALDD33
GEWAEHLTE OPERATION:   OPFL, GEWAEHLTE DATEI:  MITABDAT    4
BITTE ERROR-CODE BEACHTEN
/help-msg dms0d33 ---------------------------------------------------------------  (33)
%  DMS0D33 SPECIFIED FILE NOT CATALOGED.
%  ? The requested file has not been cataloged in the system.
%    For the file no catalog entry could be found.
%  ! Correct the error and try again.
/show-user-stat
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
MAINTASK 91K5 2 BATCH    9 210      0.5174      10 FSC
DIATASK  91C7 3 DIALOG   0 210      5.1639    9000 FSC
DIATASK  91C8 3 DIALOG   0 210      2.9012    9000 FSC
RECOAUT  91K6 2 BATCH    9 210      0.4565     200 FSC
DIATASK  91C9 3 DIALOG   0 210      3.6355    9000 FSC
%  SPS0171 NO LOCAL SPOOLOUT JOB PRESENT
%  SRO0376 NO RSO JOB OF TYPE 'T7' PRESENT
%  SCP1095 DPRINTSV WARNING : SOME DPRINT PRINT-JOBS CANNOT BE DISPLAYED
/show-fil-att lcat.mitabdat
          12 :SPVS:$USER.LCAT.MITABDAT ------------------------------------------  (34)
  :SPVS: PUBLIC:      1 FILE  RES=        12 FRE=         9 REL=         9 PAGES
/start-exe c.persdat ------------------------------------------------------------  (35)
%  BLS0500 PROGRAM 'C.PERSDAT', VERSION ' ' OF '2006-03-27' LOADED
NAME LEASY-DATEIKATALOG ?
*LCAT
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*L
SORTIERT N.PSNR(P),NAME(N) ODER ABT(A)?
*P
094711 AB212 NEUBOSS              HARDY      MUENCHEN   AM KNACKEPUNKT 1
151921 DP212 BUSCHBADER           REINHARD   MUENCHEN   AM WEICH 7
281731 AB212 BLONDIE              OTTILIE    MUENSTER   SUDSCHWEDENW. 22       --  (36)
```

```
291018 AB212 WALDI                BERND      MUENCHEN   SCHWABENSTR. 30
302015 DP212 WUPPI                HARTMUT    MUENCHEN   WALDTALSTR. 19
BITTE GEBEN SIE DIE GEWUENSCHTE AKTION EIN:
   I..MITARBEITER EINFUEGEN
   D..MITARBEITER LOESCHEN
   L..MITARBEITER AUFLISTEN
   E..PROGRAMM BEENDEN
*E
```

SYSLST logs of the LEASY-RECONST utility (37)

```
LEASY AFTER-IMAGE-RECONSTRUCTION                         DATE OF RECONSTRUCTION: 2006-04-21   TIME: 09:14:18-S

SELECTED PARAMETERS:
--------------------
LEASY CATALOG  (*CAT):     SELECTED CATALOG NAME:............. :SPVS:$USER.LCAT
                           AFTER IMAGE FILE GENERATION NUMBER: 0002 UNTIL 0002
                           UPDATING SHADOW FILES:............. YES, AUTOMATIC

OPERATION MODE (*MOD):     PRINT:............................. NORMAL
                           UPDATING SHADOW FILES:............. YES
                           UPDATING SHADOW SI-FILES:.......... YES
                           TRANSACTIONS SELECTED:............. ALL
                           UNLOAD CLASS-5-MEMORY:............. YES
                           SET FREE CMMAIN FOR RUNTIME-SYSTEM: USE OF CMMAIN NOT CHANGED

LIST REPORT    (*REP):     REQUESTED LENGTH OF RECORD OUTPUT:. SHORT
                           RECORD EXTRACTION:................. NOT SELECTED
                           LIST INVALID RECORDS:.............. YES
                           RECORD SELECTION:.................. ALL RECORDS
                           LIST USER-INFORMATION:............. NO
                           LIST PROTOCOL RECORDS:............. NO

DATE FILTER    (*DAT):     FROM DATE (YYYY-MM-DD):............ START OF FILE
                           TO   DATE (YYYY-MM-DD):............ END   OF FILE
SESSION FILTER (*SES):     FROM SESSION-NUMBER:............... START OF FILE
                           TO   SESSION-NUMBER:............... END   OF FILE
                           LAST TRANSACTION:.................. END   OF TO-SESSION
FILE FILTER    (*FIL):     SELECTED FILES:.................... :SPVS:$USER.LCAT.MITABDAT

RANGE FILTER   (*RAN):     NOT SPECIFIED
```

```
LEASY AFTER IMAGE RECONSTRUCTION, AIMFILE=:SPVS:$USER.LCAT.LEASYAIM(*0002) NEW PAMBLOCK-LINK: BLOCK# = 0000001

OP  XYS POS SESSION TRANS ITR TSN  FILE                                      TIME        RECORD
------------------------------------------------------------------------------------------------------------------
MTSK     4138 00001     0   0 91K5                                           08:59:13-S  D=2006-04-21
SESS     4210 00002     0   0 91K5                                           08:59:14-S  D=2006-04-21
CATD     4270 00002     0   0 91C7                                           09:02:22-S  T=91C7 U=USER P=C.PERSDA I=TSN-91C7
OPEN     4372 00002     0   0 91C7                                           09:02:22-S  MITABDAT:SPVS:$USER.LCAT.MITABDAT
OPTR     4429 00002     1   1 91C7                                           09:02:23-S  B=         H=         A=$DIALOG  #=
STOR     4538 00002     1   1 91C7 :SPVS:$USER.SAVE.MITABDAT                 09:02:23-S  KEY=X'00044C83'
STOR     4647 00002     1   1 91C7 :SPVS:$USER.SAVE.MITABDAT                 09:02:24-S  KEY=X'00049BBF'
CLTR     4679 00002     1   1 91C7                                           09:02:24-S
CLOS     4714 00002     0   0 91C7                                           09:02:25-S
CATD     4774 00002     0   0 91C7                                           09:11:03-S  T=91C7 U=USER P=C.PERSDA I=TSN-91C7
ENDA     4846 00002     0   0 91K5                                           09:14:18-S  D=2006-04-21
```

```
LEASY AFTER-IMAGE-RECONSTRUCTION                              DATE OF RECONSTRUCTION: 2006-04-21   TIME: 09:20:54-S

SELECTED PARAMETERS:
--------------------
LEASY CATALOG  (*CAT):     SELECTED CATALOG NAME:............ :SPVS:$USER.LCAT
                           AFTER IMAGE FILE GENERATION NUMBER: 0002 UNTIL 0003
                           UPDATING SHADOW FILES:............ YES, AUTOMATIC

OPERATION MODE (*MOD):     PRINT:............................ NORMAL
                           UPDATING SHADOW FILES:............ YES
                           UPDATING SHADOW SI-FILES:......... YES
                           TRANSACTIONS SELECTED:............ ALL
                           UNLOAD CLASS-5-MEMORY:............ YES
                           SET FREE CMMAIN FOR RUNTIME-SYSTEM: USE OF CMMAIN NOT CHANGED

LIST REPORT    (*REP):     REQUESTED LENGTH OF RECORD OUTPUT:. SHORT
                           RECORD EXTRACTION:................ NOT SELECTED
                           LIST INVALID RECORDS:............. YES
                           RECORD SELECTION:................. ALL RECORDS
                           LIST USER-INFORMATION:............ NO
                           LIST PROTOCOL RECORDS:............ NO

DATE FILTER    (*DAT):     FROM DATE (YYYY-MM-DD):........... START OF FILE
                      TO   DATE (YYYY-MM-DD):........... END  OF FILE

SESSION FILTER (*SES):    FROM SESSION-NUMBER:.............. START OF FILE
                      TO   SESSION-NUMBER:.............. END  OF FILE
                           LAST TRANSACTION:................. END  OF TO-SESSION

FILE FILTER    (*FIL):     SELECTED FILES:................... :SPVS:$USER.LCAT.MITABDAT

RANGE FILTER   (*RAN):     NOT SPECIFIED


LEASY AFTER IMAGE RECONSTRUCTION, AIMFILE=:SPVS:$USER.LCAT.LEASYAIM(*0002) NEW PAMBLOCK-LINK: BLOCK# = 0000001

OP  XYS POS SESSION TRANS ITR TSN  FILE                                      TIME       RECORD
---------------------------------------------------------------------------------------------------------------------------------
MTSK     4138 00001     0   0 91K5                                           08:59:13-S  D=2006-04-21
SESS     4210 00002     0   0 91K5                                           08:59:14-S  D=2006-04-21
CATD     4270 00002     0   0 91C7                                           09:02:22-S  T=91C7 U=USER P=C.PERSDA I=TSN-91C7
OPEN     4372 00002     0   0 91C7                                           09:02:22-S  MITABDAT:SPVS:$USER.LCAT.MITABDAT
OPTR     4429 00002     1   1 91C7                                           09:02:23-S  B=         H=         A=$DIALOG  #=
STOR     4538 00002     1   1 91C7 :SPVS:$USER.SAVE.MITABDAT                 09:02:23-S  KEY=X'00044C83'
STOR     4647 00002     1   1 91C7 :SPVS:$USER.SAVE.MITABDAT                 09:02:24-S  KEY=X'00049BBF'
CLTR     4679 00002     1   1 91C7                                           09:02:24-S
CLOS     4714 00002     0   0 91C7                                           09:02:25-S
CATD     4774 00002     0   0 91C7                                           09:11:03-S  T=91C7 U=USER P=C.PERSDA I=TSN-91C7
ENDA     4846 00002     0   0 91K5                                           09:14:18-S  D=2006-04-21


LEASY AFTER IMAGE RECONSTRUCTION, AIMFILE=:SPVS:$USER.LCAT.LEASYAIM(*0003) NEW PAMBLOCK-LINK: BLOCK# = 0000001
OP  XYS POS SESSION TRANS ITR TSN  FILE                                  TIME       RECORD
--------------------------------------------------------------------------------------------------------------------------
CSES     4138 00002     0   0 91K5                                       09:14:18-S  D=2006-04-21
CATD     4198 00002     0   0 91C7                                       09:19:15-S  T=91C7 U=USER P=C.PERSDA I=TSN-91C7
OPEN     4300 00002     0   0 91C7                                       09:19:15-S  MITABDAT:SPVS:$USER.LCAT.MITABDAT
```

Log of the ENTER procedure E.MTSK.3 (38)

```
/START-LEASY-MAINTASK
%  BLS0523 ELEMENT 'MAINTASK', VERSION '06.2A00', TYPE 'L' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-MAINTASK', VERSION '06.2A00' OF '2006-03-08 01:28:12' LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS RESERVED
%  LEA0301 LEASY MAINTASK VERSION V6.2A STARTED
CAT=LCAT
FILES=2
LOG=Y,M
AUT=Y
REN=ENTER-JOB E.RECONST.AUT,JOB-NAME=RECOAUT
TRANS=3
TIM=40
END
%  JMS0066 JOB 'RECOAUT' ACCEPTED ON 06-04-21 AT 08:59, TSN = 91K6
%  LEA5303 WARM/COLD START SUCCESSFUL
%  LEA5307 NEW LEASY SESSION CREATED: SESSION NUMBER = 00002, DATE = 2006-04-21, TIME =
08:59:14-S
%  LEA5304 *LEASY MAINTASK :SPVS:$USER.LCAT INITIALIZATION COMPLETED
%  LEA5003 START OF AIM FILE GENERATION SWITCHING ON 2006-04-21 AT 09:14:18-S
%  LEA5004 AIM FILE GENERATION SWITCHING SUCCESSFUL
%  LEA0310 NORMAL TERMINATION OF LEASY MAINTASK BECAUSE OF CLOSE DOWN FUNCTION
END
```

Log of the ENTER procedure E.RECONST.AUT (39)

```
/START-LEASY-RECONST
%  BLS0523 ELEMENT 'RECONST', VERSION '06.2A00', TYPE 'L' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-RECONST', VERSION '06.2A00' OF '2006-03-08 01:28:19' LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS RESERVED
%  LEA0401 LEASY AFTER IMAGE PROGRAM (RECONST) VERSION V6.2A STARTED
CAT :SPVS:$USER.LCAT    ,COP=(Y,A)
END
SESS#=00002 TSN#=91K5 D=2006-04-21 T=08:59:14-S
%  LEA0410 NORMAL TERMINATION OF LEASY AFTER IMAGE PROGRAM  (RECONST)
```

Log of the LEASY-MASTER utility routine

```
/start-leasy-master
%  BLS0523 ELEMENT 'MASTER', VERSION '06.2A' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-MASTER', VERSION '06.2A00' OF '2006-03-08 01:28:19' LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS RESERVED
%  LEA0501 LEASY MASTER PROGRAM VERSION V6.2A STARTED
```

```
................................................................................
LEASY MASTER PROGRAM VERSION V6.2A................SCREEN 001: MAINTASK SELECTION
................................................................................
PLEASE TYPE IN NAME OF LEASY DIRECTORY..........................................
(*END=END OF PROGRAM)...........................................................
*LCAT                                                                              -(40)
................................................................................
PLEASE ENTER PASSWORD
*C'LCAT'                                                                           -(41)
```

```
LEASY MASTER PROGRAM VERSION V6.2A.............SCREEN 003: GENERAL INFORMATION
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
CURRENT SESSION NUMBER:..........00001
CMMAIN STATUS:...................NORMAL WORKING
CMMAIN CONTROL:..................NO CONTROL FUNCTION IS ACTIV
USE BEFORE IMAGE LOGGING:........YES
USE AFTER IMAGE LOGGING:.........YES, AIM GEN#=0001                                -(42)
NUMBER OF ACTIVE TASKS:..........000 OF MAX. 003
NUMBER OF ACTIVE TRANSACTIONS:...000 OF MAX. 003
NUMBER OF OPEN FILES:............000 OF MAX. 002
NUMBER OF ACT. TA APPLICATIONS:..000 OF MAX. 003
BUCKET POOL MEMORY SIZE:.........00029696 BYTES
SIZE OF ONE BUCKET IN POOL:......00001024 BYTES
NUMBER OF BUCKETS IN BUCKET POOL:00000029
USED BUCKETS FOR LOCK ELEMENTS:..00000000
USED BUCKETS FOR TRANSACTIONS:...00000000
UNUSED BUCKETS:..................00000029
NUMBER OF LOCKED DATA RECORDS:...00000000
NUMBER OF FREE LOCK ELEMENTS:....00000003
SYSLST PRINTOUT SWITCH IS SET:...ON
UPD. COMMANDS ON CMMAIN ALLOWED:.YES
FUNCTION SELECTION (OR BLANK=MAINTASK SELECTION; OR *END=END OF PROGRAM)
*CLOS                                                                              -(43)
```

```
LEASY MASTER PROGRAM VERSION V6.2A.............SCREEN 003: GENERAL INFORMATION
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
CURRENT SESSION NUMBER:..........00001
CMMAIN STATUS:...................NORMAL WORKING
CMMAIN CONTROL:..................TERMINATE MAINTASK
CMMAIN CONTROL:..................ACCEPT NO FURTHER TRANSACTIONS
USE BEFORE IMAGE LOGGING:........YES
USE AFTER IMAGE LOGGING:.........YES, AIM GEN#=0001
NUMBER OF ACTIVE TASKS:..........000 OF MAX. 003
NUMBER OF ACTIVE TRANSACTIONS:...000 OF MAX. 003
NUMBER OF OPEN FILES:............000 OF MAX. 002
NUMBER OF ACT. TA APPLICATIONS:..000 OF MAX. 003
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
FUNCTION SELECTION (OR +=CONTINUE; OR BLANK; OR *END)
**END                                                                             –(44)
```

```
%  LEA0510 NORMAL TERMINATION OF LEASY MASTER PROGRAM
/start-leasy-master
%  BLS0523 ELEMENT 'MASTER', VERSION '06.2A' FROM LIBRARY
':5OSH:$TSOS.SYSPRG.LEASY.062' IN PROCESS
%  BLS0524 LLM 'LEASY-MASTER', VERSION '06.2A00' OF '2006-03-08 01:28:19' LOADED
%  BLS0551 COPYRIGHT (C) FUJITSU SIEMENS COMPUTERS GMBH 2006. ALL RIGHTS RESERVED
%  LEA0501 LEASY MASTER PROGRAM VERSION V6.2A STARTED
```

```
................................................................................
LEASY MASTER PROGRAM VERSION V6.2A................SCREEN 001: MAINTASK SELECTION
................................................................................
PLEASE TYPE IN NAME OF LEASY DIRECTORY..........................................
(*END=END OF PROGRAM)...........................................................
*LCAT                                                                             –(45)
................................................................................
PLEASE ENTER PASSWORD
*C'LCAT'                                                                          –(46)
```

```
LEASY MASTER PROGRAM VERSION V6.2A............SCREEN 003: GENERAL INFORMATION
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
CURRENT SESSION NUMBER:..........00002
CMMAIN STATUS:...................NORMAL WORKING
CMMAIN CONTROL:..................NO CONTROL FUNCTION IS ACTIV
USE BEFORE IMAGE LOGGING:........YES
USE AFTER IMAGE LOGGING:.........YES, AIM GEN#=0002                          -(47)
NUMBER OF ACTIVE TASKS:..........000 OF MAX. 003
NUMBER OF ACTIVE TRANSACTIONS:...000 OF MAX. 003
NUMBER OF OPEN FILES:...........000 OF MAX. 002
NUMBER OF ACT. TA APPLICATIONS:..000 OF MAX. 003
BUCKET POOL MEMORY SIZE:.........00028672 BYTES
SIZE OF ONE BUCKET IN POOL:......00001024 BYTES
NUMBER OF BUCKETS IN BUCKET POOL:00000028
USED BUCKETS FOR LOCK ELEMENTS:..00000000
USED BUCKETS FOR TRANSACTIONS:...00000000
UNUSED BUCKETS:..................00000028
NUMBER OF LOCKED DATA RECORDS:...00000000
NUMBER OF FREE LOCK ELEMENTS:....00000002
SYSLST PRINTOUT SWITCH IS SET:...ON
UPD. COMMANDS ON CMMAIN ALLOWED:.YES
FUNCTION SELECTION (OR BLANK=MAINTASK SELECTION; OR *END=END OF PROGRAM)
*REPO                                                                         -(48)
```

```
LEASY MASTER PROGRAM VERSION V6.2A..................SCREEN 039: COPY SHADOWFILE
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
................................................................................
TIME TO WAIT FOR END OF TRANSACTIONS:...005
REACTION FOR UNFINISHED TRANSACTIONS:.....R
................................................................................
................................................................................
FUNCTION SELECTION (OR R=REACTION, IN CASE OF OPEN TRANSACTIONS AFTER
WAITING TIME; OR W=ENTER WAITING TIME; OR F=FILE SELECTION;
OR S=START FUNCTION PROCESSING; OR BLANK=MAINTASK SELECTION;
OR *END=END OF PROGRAM)
*W                                                                            -(49)
```

```
LEASY MASTER PROGRAM VERSION V6.2A................SCREEN 040: ENTER WAITING TIME
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
................................................................................
PLEASE ENTER THE TIME TO WAIT FOR THE COMPLETION OF NOT YET CLOSED TRANSACTIONS
(0<=WAITING TIME<=120; BLANK=5 MINUTES (STANDARD VALUE))
*1                                                                                 -(50)
```

```
LEASY MASTER PROGRAM VERSION V6.2A...................SCREEN 039: COPY SHADOWFILE
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
................................................................................
TIME TO WAIT FOR END OF TRANSACTIONS:...001
REACTION FOR UNFINISHED TRANSACTIONS:.....R
................................................................................
................................................................................
FUNCTION SELECTION (OR R=REACTION, IN CASE OF OPEN TRANSACTIONS AFTER
WAITING TIME; OR W=ENTER WAITING TIME; OR F=FILE SELECTION;
OR S=START FUNCTION PROCESSING; OR BLANK=MAINTASK SELECTION;
OR *END=END OF PROGRAM)
*F                                                                                 -(51)
```

```
LEASY MASTER PROGRAM VERSION V6.2A....................SCREEN 042: FILE SELECTION
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
................................................................................
NO FILES SELECTED FOR FUNCTION REPO
................................................................................
FILE SELECTION (A=ADD FILENAME; OR R=REMOVE FILENAME;
OR E=END OF FILE SELECTION)
*A                                                                                 -(52)
```

```
LEASY MASTER PROGRAM VERSION V6.2A......................SCREEN 043: ADD FILENAME
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
................................................................................
PLEASE ENTER LOGICAL FILENAME TO BE ADDED
(BLANK=STOP ADDING LOGICAL FILENAMES):
*MITABDAT                                                                          -(53)
```

```
LEASY MASTER PROGRAM VERSION V6.2A....................SCREEN 042: FILE SELECTION
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
................................................................................
SELECTED FILES:
................................................................................
MITABDAT
................................................................................
FILE SELECTION (A=ADD FILENAME; OR R=REMOVE FILENAME;
OR E=END OF FILE SELECTION)
*E                                                                                –(54)
```

```
LEASY MASTER PROGRAM VERSION V6.2A...................SCREEN 039: COPY SHADOWFILE
................................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
................................................................................
TIME TO WAIT FOR END OF TRANSACTIONS:...001
REACTION FOR UNFINISHED TRANSACTIONS:.....R
................................................................................
................................................................................
FUNCTION SELECTION (OR R=REACTION, IN CASE OF OPEN TRANSACTIONS AFTER
WAITING TIME; OR W=ENTER WAITING TIME; OR F=FILE SELECTION;
OR S=START FUNCTION PROCESSING; OR BLANK=MAINTASK SELECTION;
OR *END=END OF PROGRAM)
*S                                                                                –(55)
```

```
%  LEA5003 START OF AIM FILE GENERATION SWITCHING ON 2006-04-21 AT 09:14:18-S
%  LEA5004 AIM FILE GENERATION SWITCHING SUCCESSFUL
%  LEA5536 FILE MITABDAT COPIED
%  LEA5536 FILE MITABDAT-SI COPIED
%  LEA5509 FUNCTION REPO NORMALLY TERMINATED
................................................................................
FUNCTION SELECTION (OR BLANK=MAINTASK SELECTION; OR *END=END OF PROGRAM)
*CLOS                                                                             –(56)
```

```
LEASY MASTER PROGRAM VERSION V6.2A............SCREEN 003: GENERAL INFORMATION
...............................................................................
CURRENT LEASY DIRECTORY:.........:SPVS:$USER.LCAT
CURRENT SESSION NUMBER:..........00002
CMMAIN STATUS:...................NORMAL WORKING
CMMAIN CONTROL:..................TERMINATE MAINTASK
CMMAIN CONTROL:..................ACCEPT NO FURTHER TRANSACTIONS
USE BEFORE IMAGE LOGGING:........YES
USE AFTER IMAGE LOGGING:.........YES, AIM GEN#=0003                                -(57)
NUMBER OF ACTIVE TASKS:..........000 OF MAX. 003
NUMBER OF ACTIVE TRANSACTIONS:...000 OF MAX. 003
NUMBER OF OPEN FILES:............000 OF MAX. 002
NUMBER OF ACT. TA APPLICATIONS:..000 OF MAX. 003
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
...............................................................................
FUNCTION SELECTION (OR +=CONTINUE; OR BLANK; OR *END)
**END                                                                              -(58)
```

```
%  LEA0510 NORMAL TERMINATION OF LEASY MASTER PROGRAM
```

Explanation

(1) The LEASY-CATALOG utility routine is called.

(2) The LEASY catalog LCAT is created with the password C'LCAT'.

(3) The master file MITABDAT is created with the specified properties. The associated shadow file is to be kept up-to-date automatically (AIM=(Y,A)). The naming system used when reconstructing shadow files is defined (CPC=SAVE).

(4) The temporary file PROTDAT is created with the specified properties. It is not recorded in the AIM file (AIM=N by default).

(5) The properties of all files contained in the catalog are to be output in full on the data display terminal.

(6) The LEASY-CATALOG utility routine is terminated.

(7) A shadow file with the name SAVE.MITABDAT is created for the personnel data file MITABDAT.

(8) A shadow file with the name SAVE.MITABDAT-SI is created for the associated SI file.

(9) The ENTER file E.MTSK.3, which starts the main task, is output.

(10) LEASY-MAINTASK is started in the ENTER file.

(11) The LEASY catalog LCAT is assigned.

(12) It is possible to have two files open simultaneously.

(13) The session is conducted with AIM and BIM security. The AIM buffer is written by the main task.

(14) The AIM file generations are automatically reconstructed in the shadow files.

(15) An ENTER-JOB command for starting the RECONST task with the ENTER procedure E.RECONST.AUT is defined.

(16) It is possible to process three transactions simultaneously.

(17) The program should wait up to 40 seconds for a session to become available.

(18) The E.MTSK.3 batch job is started. The main task and the RECONST task are started. The first LEASY session is initialized.

(19) The main task is started.

(20) The RECONST task was started by the main task.

(21) The files contained in the LEASY catalog LCAT are listed.

(22) The user program PERSDAT is called.

(23) The user program is terminated.

(24) The LEASY session was terminated by the LEASY-MASTER utility routine using the CLOS function (see step (43)). This means that the first AIM file generation is automatically reconstructed in the shadow files. The main task then terminates the RECONST task.

(25) The E.MTSK.3 batch job is started again (for information on outputting the contents of the ENTER file, see step (9)). The main task and the RECONST task are started. The second LEASY session is initialized.

(26) The main task is started.

(27) The RECONST task is started.

(28) The user program PERSDAT is called.

(29) The user program PERSDAT is terminated.

(30) The primary file LCAT.MITABDAT is inadvertently deleted.

(31) The user program PERSDAT is called.

(32) An attempt to access the files with the user program causes an abort.

(33) The HELP-MSG command displays the cause of the error.

(34) The REPO function is used to replace the MITABDAT file by its shadow file during ongoing operation (see steps (48) through (55)).

(35) The user program PERSDAT is called.

(36) The files are processed further.

(37) Printout of the LEASY-RECONST logs (selected functions and reconstruction log).

(38) Main task log from the second session.

(39) RECONST task log from the second session.

(40) After the LEASY-MASTER utility routine is called, the LEASY catalog for which the MASTER functions are to be executed is assigned.

(41) The password C'LCAT' is requested.

(42) General information is output, indicating among other things that this description refers to the first AIM file generation.

(43) The LEASY session is terminated using the CLOS function.

(44) The LEASY-MASTER utility routine is terminated with *END.

(45) After the LEASY-MASTER utility routine is called again, the LEASY catalog for which the MASTER functions are to be executed is assigned.

(46) The password C'LCAT' is requested.

(47) General information is output, indicating among other things that this description refers to the second AIM file generation.

(48) The REPO function is designed to replace the MITABDAT file by its shadow file during ongoing operation.

(49) The mask for entering a wait time is called.

(50) 1 minute is defined as the wait time.

(51) The mask for file selection is called.

(52) A file is to be added to the selection list.

(53) The MITABDAT file is selected.

(54) The file selection is completed.

(55) The REPO function is started with the parameters displayed. The file is copied and REPO terminates normally.

(56) The LEASY session is terminated using the CLOS function.

(57) General information is output, indicating among other things that this description refers to the third AIM file generation.

(58) The LEASY-MASTER utility routine is terminated with *END.

13 Return codes

The messages of the LEASY interface are represented in this chapter in different tables. The first table (table 24) is arranged in ascending order of RC-LC error codes, and the other tables (table 25 on page 403 and table 26 on page 404) in ascending order of compatible return codes of KLDS **and** RC-LC error codes.

Runtime system messages are also output in some cases, these messages are described in the manual “LEASY (BS2000/OSD) Utility Routines“.

LEASY-internal error code RC-LC arranged in ascending order:

RC-LC	Meaning
L000	Function correctly executed (all operations)
L001	Record with key not located (RDIR, RHLD, REWR, DLET)
L002	Duplicate (RNXT, INSR for primary or secondary key, REWR, STOR for secondary key where DUPEKY = NO)
L003	EOF for sequential reading (at file end for RNXT and RNHD, at file beginning for RPRI and RPHD) or positioning error: sequential read instruction without current range (RNXT, RNHD, RPRI, RPHD) or EOF for INSR in the case of ISAM (USAGE modes LOAD/PLOD/ELOD and LDUP/PLUP/ELUP)
L004	Sequence error in load mode (INSR)
L005	Record not locked (DLET, REWR)
L006	Timeout for locking attempt expired (LOCK, RHLD, RNHD, RPHD, INSR, STOR)
L007	Deadlock during locking attempt (LOCK, RHLD, RNHD, RPHD, INSR, STOR)
L008	Record cannot be unlocked because it was updated in the transaction (UNLK)
L009	Warning: record to be unlocked has not been locked (UNLK)
L010	Length error in variable-length record (INSR, REWR, STOR)
L011	Warning: more than 255 records per block (RNXT, RPRI; SAM) when using a SAM retrieval address in 24-bit format
L012	No current record exists (REWR; SAM) or no valid read instruction for the file identifier (before DLET without key specification)
L013	Key outside permitted range; highest PAM block number of block to be written must be ≤ (FILESIZE + SECONDARY ALLOCATION) (INSR, STOR; PAM and DAM)

Table 24: LEASY-internal error code RC-LC in ascending order (part 1 of 8)

RC-LC	Meaning
L014	Rollback not possible as transaction without BIM saving
L015	openUTM: task deadlock
L016	Writing of a DAM file record or PAM file record with BLOCK-CONTROL-INFO= WITHIN-DATA-BLOCK or BLOCK-CONTROL-INFO=NO is not possible since X'FF' is set in the first byte of the record (erase identifier for DAM) (INSR, STOR, REWR)
L017	No /ADD-FILE-LINK command issued for the specified link name.
L018	In terms of syntax, the name of the file assigned via the /ADD-FILE-LINK command is not a LEASY catalog
L019	During a sequential read operation via an ISAM secondary key the record read immediately beforehand cannot be found.
L101	File not specified in OPTR of this transaction (all operations whose 3rd operand specifies a file identifier)
L102	Operation not permitted - contrary to FCBTYPE and/or USAGE mode (all operations whose 3rd operand specifies a file identifier)
L103	No transaction open (CLTR for all operations whose 3rd operand specifies a file identifier)
L104	Transaction opened with CATD or DISCONNECT/openUTM
L105	File name or suffix not defined in LEASY catalog (OPFL, OPTR)
L106	USAGE mode incompatible with OPEN mode (OPTR after OPFL)
L107	Additional specification for model file missing (OPFL, OPTR)
L108	FILE table overflow (OPTR) – increase *FILE statement in LEASY-MAINTASK
L109	Secondary index name not defined in LEASY catalog (RDIR, RHLD, SETL) or ISAM secondary index specified for SETL.
L110	File/file identifier cannot be opened with USAGE mode or result USAGE mode requested, as it has already been opened by another transaction with a higher USAGE mode (OPTR)
L111	USAGE mode incompatible with already opened file/file identifier in the same transaction
L112	KEYLEN (ISAM file) > *KEY statement for LEASY-MAINTASK (OPTR)
L113	KEYLEN > 4 for USAGE modes LOAD, ELOD, PLOD, LDUP, PLUP, ELUP (OPTR; ISAM)
L114	Record length incompatible with block length or invalid BLKSIZE (OPFL, OPTR)
L115	The required sequence identifier was not specified for this file in earlier OPTR operations of this transaction (all operations specifying a file identifier in the 3rd operand)
L116	No CLFL executed (CATD after OPFL) or the file has already been opened (OPFL)
L117	No CLTR executed (OPFL after OPTR)
L118	CLFL: at least one of the specified files has not been opened by OPFL
L119	No CLTR executed (CLFL after OPTR)

Table 24: LEASY-internal error code RC-LC in ascending order (part 2 of 8)

RC-LC	Meaning
L120	File (OPTR) not specified in previous file list (OPFL) (OPTR after OPFL)
L122	File identifier already open
L123	AIM buffer too small (*AIB in LEASY-MAINTASK) in relation to maximum RECSIZE (OPFL, OPTR) or warm start with LEASY-MAINTASK without AIM saving, although this was activated for the transaction to be rolled back
L124	2nd OPTR call without using OPFL
L125	Entries in the LEASY catalog and those in the DMS catalog are inconsistent
L126	Incorrect file format (BLKCTRL=NO)
L130	File size exceeds 32 GB
LI01	CATD call is missing (foreign files are not permitted)
LI02	No transaction is active (DCAM LU80)
LI03	Overflow in transfer area; maximum number of application programs has been exceeded
LI04	Internal IOH error: waiting time for the I/O task has expired (*WAI statement)
LI05	Internal IOH error: I/O task has been terminated with errors when processing a LEASY call; the transaction is reset
LI06	Internal IOH error: I/O task has been terminated with errors when processing a LEASY call; the transaction is not reset
LI07	Internal IOH error: initialization error; common memory is not released
LI08	Version error; the internal version is incompatible with I/O task
LI09	Internal IOH error: semaphore (protected variable) cannot be accessed; error in internal synchronization
LI10	Internal IOH error: the record length in the CINF area is greater than the length specified in the DBL statement
LI11	File not specified in the OPF statement
LI12	Record length greater than 0 or greater than the value in the ARL statement
LI20	Versions of runtime system and I/O task do not match
LI26	Version of link module < V5.1
LP01	Operation code is incorrect (all operations)
LP02	Too few operands (all operations)
LP04	OPE1/OPE2 incorrect (CLTR)
LP06	USAGE mode incorrect or invalid (OPTR)
LP07	OPEN mode incorrect or invalid OPFL: foreign file, SHAREUP=YES, BIM=YES, OPEN mode for write. OPTR: USAGE mode not compatible with OPEN mode.

Table 24: LEASY-internal error code RC-LC in ascending order (part 3 of 8)

RC-LC	Meaning
LP08	Field selection incorrect, "(ALL)" (SETL, RDIR, RHLD)
LP09	Syntax error in file list (OPFL, OPTR, CLFL)
LP10	Syntax error in catalog name (CATD)
LP11	CI area too small for currency information (CINF) or no information in the CI area (ci-slf=0
LP12	L-OPT incorrect, ≠'1' (all operations)
LP14	PAMHPNR/SAMPTR invalid (in all operations in which these fields are evaluated)
LP15	OPE-WTIME non-numeric (all operations)
LP16	OPE-OM in RE area is set incorrectly
LP17	Invalid combination of (KB, KE) (SETL, RDIR for SAM file)
LP18	Syntax error in file identifier (for all operations with specification of DB1)
LP19	OPE-STX incorrect (CATD)
LP20	The length of the USER area is not in the range 5 < len < 1024
LS01	Common memory CMMAIN of main task not created for specified LEASY catalog (CATD, OPTR)
LS02	Operation is rejected because of CLOS or SHUT function (CATD, OPFL, OPTR)
LS03	Too many transactions - transaction table overflow (OPTR); increase *TRANS statement in LEASY-MAINTASK
LS04	Common memory CMMAIN is locked for the runtime system (*USE=R in LEASY-MAINTASK)
LS05	No operation at all possible at the moment because of HOLD function
LS06	No new transaction possible at the moment because of QUIE function
LS07	No operation for this transaction possible at the moment because of LOCT or QUIE function
LS08	Rollback due to second LS12
LS09	OPE2=T is ignored in CLTR operations because of SHUT, CLOS, RLBT or REPO function
LS10	Operation is converted to CLTR with OPE1=R because of RLBT or SHUT function
LS11	Virtual memory exhausted (REQM, ENAMP macros)
LS12	Overflow of the transaction element area (in the case of OPTR) or the lock protocol element area while attempting to enforce a new lock element; increase *MEM statement in LEASY-MAINTASK
LS13	The file is locked by the LEASY-MASTER utility routine (OPFL and OPTR)
LS14	The file is locked against opening in write mode by the LEASY-MASTER utility routine (OPFL and OPTR)
LS15	Task table overflow, increase *TSK operand in LEASY-MAINTASK utility routine

Table 24: LEASY-internal error code RC-LC in ascending order (part 4 of 8)

RC-LC	Meaning
LS17	Error in job variable function
LS18	DVS error with CATALOG file
LS19	DVS error with SI file
LS20	General DVS error
LS21	DVS error with BIM file
LS22	DVS error with AIM file
LS23	Error during rollback (CLTR,OPE1=R)
LS26	Version of link module < V5.1
LS30	STXIT macro error in LEASY module
LS31	Error in dynamic loading of a module
LS32	ENASI macro error
LS33	RELM macro error
LS34	DISSI macro error
LS35	ENAMP macro error
LS36	Version of LEACON module is incompatible with version of LEASY module
LS37	ENQAR macro error
LS38	DEQAR macro error
LS40	LEASY system error: enforced lock element not located
LS41	LEASY system error: internal lock for record splitting frozen in secondary file
LS42	LEASY system error: duplicate in secondary file when splitting record
LS43	Inconsistency between primary and secondary index files: no primary record exists for SI entry, or it contains an incorrect secondary key value. Record with primary key not found Record found, but record-type field is invalid Record found, but does not contain an SI key.
LS44	Format error in BIM file (during rollback)
LS45	LEASY system error: inconsistency in common memory (internal secondary index number not located)
LS47	LEASY system error: logic error in LEAWRAIM
LS48	LEASY system error: MVC lock frozen in LEAWRAIM
LS49	LEASY system error: WRT lock frozen in LEAWRAIM
LS51	LEASY system error: AIM buffer is full and cannot be cleared because of an error in PAM-WRITE
LS52	Format error in PAM file

Table 24: LEASY-internal error code RC-LC in ascending order (part 5 of 8)

RC-LC	Meaning
LS53	LEASY system error: AIMSWITCH lock frozen in LEALAISW
LS54	LEASY system error: open file table frozen in LEASPERR
LS55	LEASY system error: transaction table lock frozen in LEASPERR
LS56	LEASY system error: free chain lock frozen in LEASPERR
LS57	LEASY system error: release lock frozen in LEASPERR
LS58	LEASY system error: file table lock frozen in LEAFTIN
LS59	Error when writing a DAM data block: error in S1 or AIM processing has forced an automatic rollback of the transaction (CLTR, all operations whose 3rd operand specifies a file identifier)
LS60	LEASY system error: lock of deadlock bit matrix is frozen
LS61	Error in ENAEI macro
LS62	Error in ENACO macro
LS63	Error in SOLSIG macro
LS64	Error in POSSIG macro
LS65	Main task has been terminated with errors (e.g. when writing the AIM buffer to tape)
LS66	LEASY system error: error in the truncation of AIM records
LS67	LEASY system error: incorrect call for LEAKMP module
LS68	Version of the runtime system is not identical with the version of CMMAIN common memory
LS69	Error in the DISMP macro
LS70	Error in the DISEI macro
LS71	Error in CREPOOL macro (for NK-ISAM)
LS72	Error in DELPOOL macro (for NK-ISAM)
LS73	Error in ADDPLNK macro (for NK-ISAM)
LS74	Error in REMPLNK macro (for NK-ISAM)
LS75	The LEASY statement cannot be processed. The AIM file generation has reached the maximum size or it cannot be switched over (for system reasons, e.g. pubspace limit reached or because no AIM file generation is free and the value 0 was specified as an increment in the AIS statement of LEASY-MAINTASK).
LS76	Transaction semaphore could not be obtained.
LS77	Because of ROMS function, currently no LEASY statements which modify the data set (DLET, INSR, REWR, STOR) are possible.
LS78	No new transactions permitted because of REPO.

Table 24: LEASY-internal error code RC-LC in ascending order (part 6 of 8)

RC-LC	Meaning
LS79	Transaction already reset because of READ-ONLY mode (LEASY-MASTER, ROMS) or copying of shadow files (LEASY-MASTER, REPO).
LS80	No statements expect CLTR permitted because of REPO.
LS81	AIM file can no longer be written because of an error, no further LEASY request permitted, transaction was reset by LEASY.
LU01	openUTM: invalid start operand
LU02	openUTM: syntax error in start operand
LU10	openUTM: missing or insufficient start operands DCAM: error in start operation sequence (CATD and/or OPFL omitted)
LU11	openUTM/DCAM: less than 2 LEASY operands
LU12	openUTM/DCAM: OPEN mode not permitted for foreign or SAM files (file is read-only) (OPFL)
LU13	openUTM/DCAM: LEASY temporary file not permitted (OPFL)
LU14	openUTM: after a delayed CLTR a CALL-LEASY is not permitted in the same dialog step (all operations)
LU15	openUTM: file must not be opened for writing in transactions without BIM saving (OPTR)
LU16	openUTM/DCAM: error in intertask synchronization for OPFL or CLFL, or different sequence for OPFL
LU17	DCAM: error; open transaction within DCAM application for OPFL or CLFL
LU18	DCAM: error; transaction cannot be active in more than one task at the same time
LU50	openUTM/DCAM: application table overflow
LU51	openUTM/DCAM: inconsistency in the application table
LU52	openUTM/DCAM: internal lock of the application table is frozen
LU53	DVS error with STATUS file
LU54	openUTM: status inquiry for the current LEASY session with openUTM application transactions still open
LU80	openUTM: error in openUTM call sequence at openUTM database interface DCAM: DCAM application name missing (CATD); transaction identifier missing or errored (all operations within a LEASY transaction)
LU81	openUTM: OPFL call missing (OPTR)
LU82	openUTM: start operand does not start with ".LEASY_"
LU83	openUTM: incorrect operation code
LU84	openUTM: status call: operation code neither "inquiry" nor "delete"
LU85	openUTM: error in processing of suspended transactions
Addd	DMS error during processing of an AIM file

Table 24: LEASY-internal error code RC-LC in ascending order (part 7 of 8)

RC-LC	Meaning
Bddd	DMS error during processing of a BIM file
Cddd	DMS error during processing of a catalog file
Dddd	DMS error during processing of a primary file
Jddd	Error during processing of a job variable
Sddd	DMS error during processing of a secondary file
Tddd	DMS error during processing of a LEASY status file

Table 24: LEASY-internal error code RC-LC in ascending order (part 8 of 8)

Messages of the LEASY interface

The following LEASY interface messages are sorted by compatible return code.

Return codes for program control

RC-CC	RC-LC
000	L000 Function correctly executed (all operations)
010	No record satisfies the search condition L001 Record with key not located (RDIR, RHLD, REWR, DLET) L003 EOF for sequential reading (at file end for RNXT and RNHD, at file beginning for RPRI and RPHD) or positioning error: sequential read instruction without current range (RNXT, RNHD, RPRI, RPHD) or EOF for INSR in the case of ISAM (USAGE modes LOAD/PLOD/ELOD and LDUP/PLUP/ELUP)
051	Contents of sort field already contained in file or outside permitted range L002 Duplicate (RNXT, INSR for primary or secondary key, REWR, STOR for secondary key where DUPEKY=NO) L013 Key outside permitted range; the highest permitted PAM block number of the block to be written must be ≤ (FILESIZE + SECONDARY ALLOCATION) (INSR, STOR; PAM and DAM)

Table 25: Return codes for program control

Return codes indicating errors

RC-CC	RC-LC
01A	Record not retained prior to update L005 Record not locked (DLET, REWR) L012 No current record available (REWR; SAM) or no valid read command for file identifier (before DLET without key specification)
01B	Modification of contents of sort field L012 No current record exists (REWR; SAM)
02A	System I/O error Dddd I/O error during primary file processing LS20 I/O error
031	L003 EOF for sequential reading (at file end for RNXT and RNHD; at file beginning for RPRI and RPHD) or positioning error: sequential read instruction without current range (RNXT, RNHD, RPRI, RPHD) or EOF for INSR in the case of ISAM (USAGE modes LOAD/PLOD/ELOPD and LDUP/PLUP/ELUP)
043	Invalid file name L105 File name or suffix not defined in LEASY catalog (OPFL, OPTR) L107 Suffix for model file missing (OPFL, OPTR) LP18 Syntax error in file identifier (for all operations with specification of DB1) LU13 openUTM/DCAM: LEASY temporary file not permitted (OPFL)
04A	Field selection entry invalid LP08 Field selection incorrect, ’(ALL)’ (SETL, RDIR, RHLD)
04B	Invalid operation key LP01 Operation code incorrect (all operations)

Table 26: Return codes indicating errors (part 1 of 10)

RC-CC	RC-LC	
04D	Invalid operation extension	
	LP04	OPE1/OPE2 incorrect (CLTR)
	LP06	USAGE mode incorrect or invalid (OPTR)
	LP07	OPEN mode incorrect or invalid OPFL: foreign file, SHAREUP=YES, BIM=YES, OPEN mode for write. OPTR: USAGE mode not compatible with OPEN mode.
	LP12	L-OPT incorrect, ≠'1' (all operations))
	LP14	PAMHPNR/SAMPTR invalid (all operations in which these fields are evaluated)
	LP15	OPE-WTIME non-numeric (all operations)
	LP16	OPE-OM in RE area set incorrectly
	LP19	OPE-STX incorrect (CATD)
04X	Format error in LEASY call	
	LP02	Too few operands (in all operations)
	LP10	Syntax error in catalog name (CATD)
	LP11	CI area too small for currency information (CINF) or no information in the CI area (ci-slf=0)
	LP20	The length of the USER area is not within the range 5 < len < 1024
04Y	Syntax errors in multiple operands	
	LP09	Syntax error in file list (OPFL, OPTR, CLFL)
05A	Invalid search condition	
	L109	Secondary index name not defined in LEASY catalog (RDIR, RHLD, SETL) or ISAM secondary index specified for SETL
	LP17	Invalid combination of (KB, KE) (SETL, RDIR for SAM file)
07A	DLET system error	
	Addd	DMS error in processing an AIM file
	Bddd	DMS error in processing a BIM file
	Dddd	DMS error in processing a primary file
	Sddd	DMS error in processing a secondary index file
	LSxx	System errors, as described for RC-CC='99A' with RC-LC='LSxx' , can also occur here
07B	System error with INSR or STOR RC-LC as with RC-CC = ' 07A'	
07C	System error with REWR RC-LC as with RC-CC = ' 07A'	

Table 26: Return codes indicating errors (part 2 of 10)

RC-CC	RC-LC	
091	File not available	
	L101	File not specified in OPTR of this transaction (all operations whose 3rd operand specifies a file identifier)
	L103	No transaction open (CLTR, all operations whose 3rd operand specifies a file identifier)
	L115	The required sequence identifier was not specified for this file in previous OPTR operations of this transaction (all operations whose 3rd operand specifies a file identifier)
	L118	CLFL: At least one of the specified files was not opened by OPFL.
	L120	File (OPTR) not specified in previous file list (OPFL) (OPTR after OPFL)
	L126	Incorrect file format (BLKCTRL=NO)
	L130	File size exceeds 32 GB
092	Invalid mode in file processing (e.g. incorrect sequence of operations)	
	L102	Operation not permitted - contradicts FCBTYPE and/or USAGE mode (all operations whose 3rd operation specifies a file identifier)
	L106	USAGE mode incompatible with OPEN mode (OPTR after OPFL)
	L119	No CLTR executed (CLFL after OPTR)
	L124	2nd OPTR call without using OPFL
	LU12	openUTM/DCAM: OPEN mode not permitted for foreign or SAM files (file is read-only) (OPFL)
	LU14	openUTM: after a delayed CLTR a CALL LEASY is not allowed in the same dialog step (all operations)
093	File already open	
	L104	Transaction opened with CATD or DISCONNECT/openUTM
	L116	No CLFL executed (CATD after OPFL) or the file has already been opened (OPFL)
	L117	No CLTR executed (OPFL after OPTR)
	L122	File identifier already open
094	MARK error	
	Lxxx	Errors as described for RC-CC='99A' with RC-LC='xxx' can also occur here
095	BACK error	
	L014	Rollback not possible as transaction without BIM saving
	Lxxx	Errors as described for RC-CC='99A' with RC-LC='xxx' can also occur here

Table 26: Return codes indicating errors (part 3 of 10)

RC-CC	RC-LC	
802	Memory space in file full	
	Dddd	Memory space for primary file exhausted
	LS20	No memory with file
99A	Other errors	
	Addd	DMS error while processing an AIM file
	Bddd	DMS error while processing a BIM file
	Cddd	DMS error while processing a catalog file
	Dddd	DMS error while processing a primary file
	Jddd	Error while processing a job variable
	Sddd	DMS error while processing a secondary index file
	Tddd	DMS error while processing a LEASY status file
	L004	Sequence error in load mode (INSR)
	L006	Timeout exceeded during locking attempt (LOCK, RHLD, RNHD, RPHD, INSR, STOR)
	L007	Deadlock during locking attempt (LOCK, RHLD, RNHD, RPHD, INSR, STOR)
	L008	Record to be released cannot be unlocked because it was updated in the transaction (UNLK)
	L009	Warning: record to be released was not locked (UNLK)
	L010	Length error in variable-length record (INSR, REWR, STOR)
	L011	Warning: more than 255 records per block (RNXT, RPRI; SAM) when using a SAM retrieval address in 24-bit format
	L014	Rollback not possible as transaction without BIM saving
	L015	openUTM: task deadlock
	L016	Writing of a DAM file record or PAM file record with BLOCK-CONTROL-INFO= WITHIN-DATA-BLOCK or BLOCK-CONTROL-INFO=NO is not possible since X'FF' is set in the first byte of the record (erase identifier for DAM) (INSR, STOR, REWR)
	L017	No /ADD-FILE-LINK command issued for specified link name
	L018	The name of the file assigned via the /ADD-FILE-LINK command is syntactically not a LEASY catalog
	L019	During a sequential read operation via an ISAM secondary key the record read immediately beforehand cannot be found.
	L108	FILE table overflow (OPTR) – increase *FILE statement in LEASY-MAINTASK

Table 26: Return codes indicating errors (part 4 of 10)

RC-CC	RC-LC	
99A	L110	File/file identifier cannot be opened with USAGE mode or result USAGE mode requested as it has already been opened by another transaction with a higher USAGE mode (OPTR)
	L111	USAGE mode incompatible with already opened file/file identifier in the same transaction
	L112	KEYLEN (ISAM file) > *KEY statement in LEASY-MAINTASK (OPTR)
	L113	KEYLEN > 4 for USAGE modes LOAD, ELOD, PLOD, LDUP, PLUP, ELUP (OPTR; ISAM)
	L114	Record length incompatible with block length or invalid BLKSIZE (OPFL, OPTR)
	L116	No CLFL executed (CATD after OPFL) or the file has already been opened (OPFL)
	L120	The user has updated to LEASY V6.1 but has started an IO task of a version earlier than LEASY V6.1. It is only allowed to use IO tasks from LEASY V6.1. It is no longer possible to use older version IO tasks.
	L123	AIM buffer too small (*AIB in LEASY-MAINTASK) in relation to maximum RECSIZE (OPFL, OPTR) or warm start performed with LEASY-MAINTASK without AIM saving, although this was activated for the transaction to be rolled back
	L125	Entries in the LEASY catalog and those in the DMS catalog are inconsistent
	LS01	Common memory CMMAIN of main task not created for specified LEASY catalog (CATD, OPTR)
	LS02	Operation rejected because of CLOS or SHUT function (CATD, OPFL, OPTR)
	LS03	Too many transactions - transaction table overflow (OPTR); increase *TRANS statement in LEASY-MAINTASK
	LS04	Common memory CMMAIN is locked for the runtime system (*USE=R in LEASY-MAINTASK)
	LS05	No operation at all possible at the moment because of HOLD function
	LS06	No new transactions possible at the moment because of QUIE function
	LS07	No operation possible at the moment for this transaction because of LOCT or QUIE function
	LS08	Rollback due to second LS12
	LS09	OPE2=T is ignored in CLTR operations because of SHUT, CLOS, RLBT or REPO function

Table 26: Return codes indicating errors (part 5 of 10)

RC-CC	RC-LC	
99A	LS10	Operation is converted to CLTR with OPE1=R because of RLBT or SHUT function
	LS11	Virtual memory exhausted (REQM, ENAMP macros)
	LS12	Overflow of transaction element area (for OPTR) or lock log element area when trying to enforce a new lock element; increase *MEM in LEASY-MAINTASK
	LS13	The file is locked by the LEASY-MASTER utility routine (OPFL and OPTR)
	LS14	The file is locked against opening in write mode by the LEASY-MASTER utility routine (OPFL and OPTR)
	LS15	Task table overflow, increase *TSK operand in LEASY-MAINTASK utility routine
	LS17	Error in job variable function
	LS18	DVS error with CATALOG file
	LS19	DVS error with SI file
	LS20	General DVS error
	LS21	DVS error with BIM file
	LS22	DVS error with AIM file
	LS23	Error during rollback (CLTR,OPE1=R)
	LS26	Version of link module < V 5.1
	LS30	STXIT macro error in LEASY module
	LS31	Error in dynamic loading of a module
	LS32	ENASI macro error
	LS33	RELM macro error
	LS34	DISSI macro error
	LS35	ENAMP macro error
	LS36	Version of LEACON module is incompatible with version of LEASY module
	LS37	ENQAR macro error
	LS38	DEQAR macro error
	LS40	LEASY system error: enforced lock element not located
	LS41	LEASY system error: internal lock for record splitting frozen in secondary file
	LS42	LEASY system error: duplicate in the secondary index file when splitting record

Table 26: Return codes indicating errors (part 6 of 10)

RC-CC	RC-LC	
99A	LS43	Inconsistency between primary and secondary index files: no primary record exists for SI entry, or it contains an incorrect secondary key value. Record with primary key not found Record found, but record-type field is invalid Record found, but does not contain an SI key.
	LS44	Format error in BIM file (during rollback)
	LS45	LEASY system error: inconsistency in common memory (internal secondary index number not located)
	LS47	LEASY system error : logic error in LEAWRAIM
	LS48	LEASY system error : MVC lock frozen in LEAWRAIM
	LS49	LEASY system error : WRT lock frozen in LEAWRAIM
	LS51	LEASY system error : AIM buffer is full and cannot be cleared because of an error in PAM-WRITE
	LS52	Format error in PAM file
	LS53	LEASY system error: AIMSWITCH lock frozen in LEALAISW
	LS54	LEASY system error: open file table lock frozen in LEASPERR
	LS55	LEASY system error: transaction table lock frozen in LEASPERR
	LS56	LEASY system error: free chain lock frozen in LEASPERR
	LS57	LEASY system error: release lock frozen in LEASPERR
	LS58	LEASY system error: file table lock frozen in LEAFTIN
	LS59	Error when writing a DAM data block: Error in S1 or AIM processing has forced an automatic rollback of the transaction (CLTR, all operations whose 3rd operand specifies a file identifier)
	LS60	LEASY system error: deadlock bit matrix lock is frozen
	LS61	Error in the ENAEI macro
	LS62	Error in the ENACO macro
	LS63	Error in the SOLSIG macro
	LS64	Error in the POSSIG macro
	LS65	Main task has been terminated with errors (e.g. when writing the AIM buffer to tape)
	LS66	LEASY system error: error in the truncation of AIM records
	LS67	LEASY system error: incorrect call for the LEAKMP module
	LS68	Version of the runtime system is different to the version of common memory CMMAIN

Table 26: Return codes indicating errors (part 7 of 10)

RC-CC	**RC-LC**	
99A	LS71	Error in CREPOOL macro (for NK-ISAM) The RC-LCE field contains the main return code of the macro (see the ”DMS Macros” manual)
	LS72	Error in DELPOOL macro (for NK-ISAM) The RC-LCE field contains the main return code of the macro (see the ”DMS Macros” manual)
	LS73	Error in ADDPLNK macro (for NK-ISAM) The RC-LCE field contains the main return code of the macro (see the ”DMS Macros” manual)
	LS74	Error in REMPLNK macro (for NK-ISAM) The RC-LCE field contains the main return code of the macro (see the ”DMS Macros” manual)
	LS75	The LEASY statement cannot be processed. The AIM file generation has reached the maximum size or it cannot be switched over (for system reasons, e.g. pubspace limit reached or because no AIM file generation is free and the value 0 was specified as an increment in the AIS statement of LEASY-MAINTASK).
	LS76	Transaction semaphore could not be acquired.
	LS77	Because of ROMS function, currently no LEASY statements which modify the data set (DLET, INSR, REWR, STOR) are possible.
	LS78	No new transactions permitted because of REPO.
	LS79	Transaction already reset because of READONLY mode (LEASY-MASTER, ROMS) or copying of shadow files (LEASY-MASTER,REPO).
	LS80	No statements expect CLTR permitted because of REPO.
	LS81	AIM file can no longer be written because of a n error, no further LEASY request permitted, transaction was reset by LEASY.
UTM	Errors in openUTM mode	
	LU01	Invalid start parameter
	LU02	Syntax error in start parameter
	LU10	Missing or insufficient start parameters
	LU11	Less than 2 LEASY operands
	LU12	openUTM/DCAM: OPEN mode impermissible for foreign or SAM file (file is read-only) (OPFL)
	LU13	openUTM/DCAM: LEASY temporary file not allowed (OPFL)
	LU14	openUTM: After delayed CLTR, CALL-LEASY in the same dialog step is not allowed (all operations)

Table 26: Return codes indicating errors (part 8 of 10)

RC-CC	RC-LC	
UTM	LU15	openUTM: In transactions without BIM saving file cannot be opened for writing (OPTR)
	LU16	Error in intertask synchronization for OPFL or CLFL, or different sequence of file names for OPFL
	LU50	Application table overflow
	LU51	Inconsistency in the application table
	LU52	Internal lock of the application table is frozen
	LU53	DVS error with STATUS file
	LU54	Status inquiry for the current LEASY session with openUTM application transactions still open
	LU80	Error in openUTM call sequence at openUTM database interface
	LU81	OPFL call missing (OPTR)
	LU82	Start parameter does not begin with ".LEASY_"
	LU83	Incorrect operation code
	LU84	Status call: operation code neither "inquiry" nor "delete"
	LU85	Error in processing of suspended transactions
DCA	Errors in DCAM mode	
	LU10	Error in start operation sequence (CATD and/or OPFL omitted)
	LU11	Less than 2 LEASY operands
	LU12	openUTM/DCAM: OPEN mode impermissible for foreign or SAM file (file is read-only) (OPFL)
	LU13	openUTM/DCAM: LEASY temporary file not allowed (OPFL)
	LU16	Error in intertask synchronization for OPFL or CLFL, or different order of file names for OPFL
	LU17	Error: open transaction within DCAM application for OPFL or CLFL
	LU18	Error: transaction cannot be active simultaneously in more than one task
	LU50	Application table overflow
	LU51	Inconsistency in the application table
	LU80	DCAM application name missing (CATD); transaction identifier missing or errored (all operations within a LEASY transaction)
DRV	Error in DRIVE call	
	LD01	No free entry in DRIVE user table
	LD02	Entry already in DRIVE user table
	LD03	Entry not in DRIVE user table

Table 26: Return codes indicating errors (part 9 of 10)

RC-CC	**RC-LC**	
IOH	Error in the I/O handler	
	L104	CATD error: CATD for an open transaction
	LI01	CATD call is missing (foreign files are not permitted)
	LI02	No transaction is active (DCAM LU80)
	LI03	Overflow in transfer area; maximum number of application programs has been exceeded
	LI04	Internal IOH error: waiting time for I/O task has expired (*WAI statement)
	LI05	Internal IOH error: I/O task terminated with errors when processing a LEASY call; transaction is reset
	LI06	Internal IOH error: I/O task terminated with errors when processing a LEASY call; transaction is not reset
	LI07	Internal IOH error: initialization error; common memory is not released
	LI08	Version error. Internal version is not compatible with I/O task
	LI09	Internal IOH error: the semaphore (protected variable) cannot be accessed; error in internal synchronization
	LI10	Internal IOH error: the record length in the CINF area is larger than the length specified in the *DBL statement
	LI11	File not specified in OPF statement
	LI12	Record length greater than 0 or greater than the value in the ARL statement
	LI20	Versions of runtime system and I/O task do not match
	LI26	Version of link module < V5.1
	LP02	CATD error: CATD has too few operands
	LP10	CATD error: syntax error in catalog name
	LS01	Common memory CMMAIN of main task not created for specified LEASY catalog (CATD, OPTR
	LS35	Macro error: error in the ENAMP macro
	LS61	Macro error: error in the ENAEI macro
	LS63	Macro error: error in the SOLSIG macro
	LS64	Macro error: error in the POSSIG macro
	LS69	Macro error: error in the DISMP macro
	LS70	Macro error: error in the DISEI macro

Table 26: Return codes indicating errors (part 10 of 10)

14 Diagnosis file

A central diagnosis file exists (called "*leadiag*“ in the following text), that is valid for one LEASY catalog. All applications that work with this catalog write certain messages into this file. The diagnosis file is used by customer support to examine the causes of errors.

Properties of the file:

- To fulfill the parallel read/write access requirements, it is an ISAM shared update file with the following properties:
  `FCBDAT  FCB  FCBTYPE=ISAM, EXIT=EXLST, KEYARG=SATZKEY, KEYPOS=5, KEYLEN=8, SHARUPD=YES, OPEN-MODUS=INOUT`
- Name of the diagnosis file: *catalog-name.LEADIAG*.
  The file is created by the LEASY main task if it does not already exist. This is possible because the LEASY main task always runs in the ID in which the catalog is located. Any *CATID* specified with the catalog is also taken into account, i.e. the diagnosis file *leadiag* is on the same pubset as the LEASY catalog.

Writing to the diagnosis file

The procedure is as follows when a request is issued to write into the diagnosis file *leadiag* (see page 416):

- A general check is made when the catalog (*CATD*) is opened:
  FSTAT is used to determine whether the catalog with the fixed name *catalog-name.LEADIAG* already or still exists in the ID containing the catalog. If not, the file is created.
- The message is written into the diagnosis file in a standardized format with the next higher key. Among other things, this format contains the following information: date, time, user name, DB name, module name/ID, where the error occurred. For operating system and DMS call errors there are special formats for outputting the relevant error codes. Further special formats exist for utility routines and for starting and terminating user programs.

The central diagnosis file always remains existent. The administrator can delete it exclusively (i.e. as long as the LEASY runtime system or LEASY main task has not opened it) or reduce it in size by deleting obsolete records.

When is a record written into the diagnosis file?

- Generally important information, e.g. creation of the common memory pool by the LEASY main task and termination of the LEASY main task are output to the diagnosis file.
- When system errors occur, generally with a return code *LSxxx*, an error record is written into the diagnosis file and a dump is created.
- If a dump declaration exists for error codes other than *LSxxx*, a record is written to the diagnosis file. In this case, "*DPRC 00*“ is output as the module ID (*DPRC* is the statement for enabling a dump) because the causing module is not known.
- When an event which is relevant for the logbook occurs, the associated information is written to the diagnosis file. These events are:
  - The following inputs for administration using the LEASY-MASTER utility routine: QUIE, HOLD, CONT, RLBT, LOCT, UNLT, LOCF, UNLF, AIMI, AIMC, AIME, AIMW, IOTE, AIMA, REPO, ROMS, ROMR
  - Starting the LEASY-MAINTASK utility routine
    All start parameters with the exception of **COM*, **PAS* and **REN* are logged together with their operand values.
  - Starting the LEASY-RECONST utility routine
    The start parameters **CAT*, **DAT*, **FIL*, **MOD* and **SES* are logged together with their operand values.
  - Starting or terminating user programs.

Errors with leadiag

If DVS errors occur when creating, opening or writing to the diagnosis file (e.g. if insufficient space is available), LEASY reacts as follows:

- **LEASY main task:** it must be possible to open or create the diagnosis file and it must be possible to output creation of the common memory pool, otherwise a hard abort occurs because the system is generally not functional. A message with DVS code is thereby output to SYSOUT to allow the administrator to take the necessary steps.
- **LEASY runtime system:** the message actually provided for *leadiag* is written to SYSOUT and the DVS code of the failed call (*OPEN* or *PUT*) is additionally output to allow the administrator to take the necessary steps. To highlight this message in the SYSOUT log, the messages `LEA5014` and `LEA5015` are output before and after it.

Limiting the volume of data in the diagnosis file

The diagnosis file *leadiag* is automatically stored after 100,000 records have been logged (this corresponds to a volume of data of approx. 8 MB when the record length is 80 bytes).

For this purpose it is cataloged under a new name which is assigned a suffix containing the current date and current time in the format *yyyy-mm-dd.hhmmss*, e.g. `LEATEST.LEADIAG.2006-07-24.110523`.

The diagnosis file which is currently to be written is created with the default name (*catalog-name.LEADIAG*). This procedure ensures a continuous file sequence. The LEASY administrator must limit the number of files himself/herself, i.e. delete old files (if required, after saving them to tape).

15 Technical data

Scope of supply

- *SYSLNK.LEASY.062* library:
    - LEACON: connection module which loads the LEASY runtime module *LEACONX* dynamically.
    - LEASY: (non-reentrant, size approx. 3 KB) connection module containing the entry *LEASY*. At the first branch, the connection module *LEACON* is loaded dynamically.
    - LEASYI: (non- reentrant, size approx. 3 KB) connection module (parameter passing according to ILCS conventions) containing the entry *LEASY*. When called the first time, the connection module *LEACON* is loaded dynamically.
    - Dynamically loadable modules:

LEACNV	LEASY-CONVERT
LEACONX	LEASY runtime system (size approx.114 KB)
LEACTX	LEASY-CATALOG
LEAICNX	I/O-TASK module
LEAILCS	ILCS connection module
LEAITX	LEASY-IOTASK
LEALDX	LEASY-LOADSI
LEAMSX	LEASY-MASTER
LEAMTX	LEASY-MAINTASK
LEARCX	LEASY-RECONST
LEASVX	LEASY-SAVE

- *SYSLNK.LEASY.062.DCAM* library:
  - LEADCAM: (non-reentrant, size approx. 5 KB) with the entry address LEASY as a substitute for the *LEASY* module for DCAM applications.
  - LEADCAMI: (non-reentrant, size approx. 5 KB; parameter passing according to ILCS conventions) with entry address LEASY as a replacement for the *LEASY* module with DCAM applications.
- *SYSLNK.LEASY.062.IOH* library:
  - LEASY: (non-reentrant, size approx. 3 KB) mandatory if the I/O-TASK is always to be used irrespective of the job variables.
  - LEASYI: (non-reentrant, size approx. 3 KB) the *LEASYI* module (parameter passing according to ILCS conventions) mandatory if the I/O TASK is always to be used irrespective of the job variables.
  - LEACON: connection module which loads the I/O-TASK module *LEAICNX* dynamically.
- *SYSLIB.LEASY.062* library:
  - COPY elements for the COBOL interface

LEASYKON	Constants for the interface
LEASYPAR	Parameters for the interface
LEASYRE	LEASY RE area for the DATA DIVISION (WORKING-STORAGE-SECTION)
LEASYREL	LEASY RE area for the DATA DIVISION (LINKAGE-SECTION)

  - Macros for UTM applications

KDCDB	Macro for generating KDCROOT
KDCDBL	Macro for generating KDCROOT with multi-db operation

  - Macros for the Assembler interface
- *SYSPRG.LEASY.062* library:
  - *LEA.xxx* program files for the old call interface *START-PROGRAM phase* (see the Release Notice for further information)
  - Start modules (LLMs) of the LEASY utilities for the SDF call interface *START-LEASY-xxx*
- Message file *SYSMES.LEASY.062*
- Subsystem declarations *SYSSSC.LEASY.062*
- System syntax file *SYSSDF.LEASY.062*

- Information on IMON *SYSSII.LEASY.062*
- Extraction procedure *SINPRC.LEASY.062* (see the Release Notice for further information)
- Release Notice *SYSFGM.LEASY.062.D* (German) and *SYSFGM.LEASY.062.E* (English)

Size of dynamic memory

During the run *LEACON* requests additional task-specific memory using REQM (class 6 memory); the size of this memory depends primarily on the number of files used. Its size can be calculated by adding the various memory requirements (see the Release Notice for the applicable figures):

1 Stack area for internal procedure data

1 BIM buffer

1 FCB area for each BIM file opened plus internal management

1 FCB area for the AIM file plus internal management

1 FCB area for the catalog file

1 area for each LEASY file (SAM, ISAM, DAM or PAM) for FCB plus internal management

Area for trace information

Area for global management data

1 work area for DAM file processing

In addition, DMS (in class 5 memory) requests two IOAREAs with the size *BLKSIZE* for each SAM, DAM or ISAM file opened. IOAREAs are not created for PAM files.

If secondary keys are defined for an ISAM, DAM or PAM file, an additional ISAM secondary key file is created for each primary file; memory space for this additional file is generated as follows:

1 Area for FCB and internal LEASY SI buffer (class 6 memory

2 Areas for the IOAREAs (class 5 memory)

Size and format of common memory

The required size of the common memory CMMAIN can be calculated using the following formula:

```
SIZE [bytes]=  292 + 64 + 16 + 88*d + 8*tk*dam + (t+7)/8 + 33*m + 22*si +

                    k
          5*pl + Σ (1 + l(SADEFk)) +  80*tk + 22*app + 144*t + 32*f +
                  i=1

         (5 + 2* maxkey)*10*t + buc*t [+ 4096*ai] [+ 4096*t]
d          number of files in the LEASY catalog (except model file instances)
m          number of model file instances in LEASY catalog
dam        number of DAM files in the LEASY catalog
si         number of secondary indices of all files
pl         number of (pos, len, dist) definitions of all secondary keys
           of all files
k          number of record type interdependences for secondary key
           definitions
l(SADEF)   length of a record type definition for secondary keys
           contingent on record type
           (l(SADEF) = 0 where RTP=NONE in the FIL statement,
                       LEASY-CATALOG)
tk         number of tasks permitted in parallel         *TSK  ⎫
           (timesharing, batch and inquiry and                 ⎪
           transaction tasks).                                 ⎪ MAINTASK
app        number of inquiry and transaction mode              ⎪ state-
           applications permitted in parallel            *APP  ⎪ ments for
t          number of transactions permitted in parallel  *TRA  ⎬ defining
f          maximum number of files that may be open            ⎪ the
           simultaneously                                *FIL  ⎪ appro-
maxkey     highest "keylen" from ISAM or PAM             *KEY  ⎪ priate
buc        size of a bucket in a bucket pool             *MUS  ⎪ values
ai         number of pages (4K) of the AIM buffer        *AIB  ⎭
```

Generally the default size of the common memory (the LEASY-MAINTASK statement **MEM=1* corresponds to one segment, i.e. 64 KB) is sufficient.
If necessary a higher value should be specified in the **MEM* statement.

It should be noted that the AIM buffer is contained in common memory CMMAIN and, depending on its size, can have a strong effect on the value of the **MEM* statement.

Common memory CMMAIN comprises permanent sections at the beginning and end of the memory area with a section between that is managed dynamically.

The permanent sections comprise:

- management block
- coordination block
- image of the LEASY catalog
- task table
- inquiry-and-transaction mode application table
- transaction table
- after-image buffer, if *LOG=A/Y
- before-image buffer, if *LOG=B/Y.

The dynamically managed memory section contains:

- a contiguous area for the lock table, which has space for 100 lock elements per transaction
- a memory area (bucket pool) subdivided into memory units (buckets) of equal size.

The user can set the size of the memory units (buckets) using the **MUS* (Memory Unit Size) statement. At least one bucket for storing the transaction-oriented file identifier management must be available for each transaction running in parallel.
The transaction is assigned further buckets from the bucket pool should one bucket be insufficient. At the end of the transaction all these buckets are released again and free space management for buckets is provided.

If the contiguous memory for lock elements overflows, buckets from the bucket pool are also assigned dynamically to the lock table. Those buckets which are called upon for the lock elements remain allocated to the lock table for as long as the CMMAIN exists. When the lock elements are released, they are incorporated in the free space management for lock elements.

Glossary

AIM

After-Image Saving:
- storage of the logical data contents after updating
- protection in the case of hardware faults (destruction of data fields)
- reconstruction after system crash
- implementation using AIM file and save copies of the original files.

BIM

Before-Image Saving:
- storage of data contents before updating
- protection in case of software errors (program abortion)
- no interruption of or interference with other transactions
- warm start capability (restart after software errors)
- implementation using BIM files.

common memory

Common memory for all tasks connected to a catalog.

DAM

Direct Access Method:
Derived from the relative file organization of COBOL, and designed in accordance with the KLDS standard, additionally supported by secondary indices (see page 41ff).

DCAM

Data Communication Access Method:
Inquiry and transaction mode in BS2000.

deadlock

General state of waiting for system resources which, due to the particular configuration involved, can never terminate without outside intervention.

ISAM

Indexed Sequential Access Method as defined by the Data Management System (see the "Introductory Guide to DMS" manual), additionally supported by secondary indices (see page 41ff).

lock

Feature of the primary sort key of a data record for the logical separation of write accesses to one record by different transactions.

openUTM

Universal Transaction Monitor. UTM is used for implementing inquiry and transaction mode in BS2000.

PAM

BS2000 primary, block-oriented access method (see the "Introductory Guide to DMS" manual), additionally supported by secondary indices (see page 41ff).

primary index

The primary index is a unique sort key for files which permit direct access.

- The ISAM key of the file is the primary sort key of ISAM files.
- The block number (half-page number) of the PAM block is the primary sort key of PAM files, and is at the beginning of the user's logical data block (the data block can extend over several PAM blocks of up to 2048 bytes each).
- A 4-byte value greater than or equal to zero, which is not part of the file record, is the sort key of DAM files.

SAM

Sequential files as defined by the Data Management System (see the "Introductory Guide to DMS" manual).

secondary index

Sort term for records of an ISAM, DAM or PAM file, which, like the primary sort key, enables direct accessing of the records in a file.

SI

Secondary index.

SI file

Secondary index file.
This file contains the secondary key pointers to primary keys of the user file.

transaction

A series of file access operations which are processed as a single unit.

UPAM

See PAM.

UTM

See openUTM.

Related publications

The manuals are available as online manuals, see *http://manuals.fujitsu-siemens.com*, or in printed form which must be paid and ordered separately at *http://FSC-manualshop.com*.

LEASY (BS2000/OSD)
Utility Routines
User Guide

LEASY (BS2000)
Ready Reference

COBOL85 (BS2000)
COBOL Compiler
User's Guide

COBOL2000 (BS2000/OSD))
COBOL Compiler
User's Guide

Assembler (BS2000)
Reference Manual

SDF (BS2000/OSD)
Introductory Guide to the SDF Dialog Interface
User Guide

BS2000/OSD-BC
Commands, Volume 1 - 5
User Guide

BS2000/OSD-BC
Executive Macros
User Guide

BS2000/OSD-BC
Utility Routines
User Guide

SORT (BS2000/OSD)
User Guide

ARCHIVE (BS2000/OSD)
User Guide

BS2000/OSD-BC
System Messages, Volume 1 and Volume 2
User Guide

***open*UTM** (BS2000/OSD)
Generating and Handling Applications
User Guide

***open*UTM** (BS2000/OSD, UNIX, Windows NT)
Administering Applications
User Guide

***open*UTM** (BS2000/OSD, UNIX, Windows NT)
Programming Applications with KDCS for COBOL, C and C++
Core Manual

DCAM (TRANSDATA)
Program Interfaces
Reference Manual

DCAM (TRANSDATA)
COBOL Calls
User Guide

JV (BS2000/OSD)
Job Variables
User Guide

BS2000/OSD-BC
Security Handbook for Systems Support

DRIVE/WINDOWS (BS2000)
Programming System
User Guide

BS2000/OSD-BC
Introductory Guide to DMS
User Guide

BS2000/OSD-BC
DMS Macros
User Guide

Index

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *…@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Copyright Fujitsu Technology Solutions, 2009

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *…@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.

Copyright Fujitsu Technology Solutions, 2009