Chapter 6

DISK FILE HANDLING INSTRUCTIONS

6.1 INTRODUCTION

The CREDIT instruction set for disk file handling is described in this chapter. The same instructions are used for all packages. The description is based on the instruction set supported by SDM; instructions, options and features not supported by ADM are marked with "not in ADM" and instructions, options and features only supported by EDM are marked with "only for EDM".

Each section starts with a functional description of the instruction, then the parameters are discussed and one or more examples are given at the end.

Statements listed at the foot of the pages should be used for reference to the CREDIT Reference Manual (module M4A), where syntax and further information on the statements can be found. The TOSS release 12.0 version of the CREDIT Reference Manual should be used.

6.1.1 Survey of Disk File Handling Instructions

File handling instructions:

OPEN .DOUT	Create a new file and open for direct output
OPEN .EXT	Open and Extend an existing standard file for sequential output
OPEN .IN	Open an existing file for input only
OPEN .INOUT	Open an existing file for input and output
OPEN .SOUT	Create a new file and open for sequential output
CLOSE	Close file
CLOSE .DROP	Close and delete file
POSIT .DIR	Set Current Record Number on specified record (not ADM)
POSIT .IXDIR	Set Current Record Number on record with specified key (not ADM)
DSC X'19'	Read File Parameters

Record handling instructions

READ .DIR	Read record with specified relative key
READ .SEQ	Read next record using the relative key (not ADM)
READ .IXDIR	Read record with specified symbolic key (not ADM)
READ .IXSEQ	Read record with next symbolic key (not ADM)
WRITE .DIR	Write record with specified relative key
WRITE .SEQ	Write next record using the relative key
WRITE .IXDIR	Write record with specified symbolic key (not ADM)
WRITE .IXSEQ	Write record with next symbolic key (only EDM)
REWRITE .CUR	Rewrite current record (not ADM)
REWRITE .DIR	Rewrite record with specified relative key
REWRITE .IXDIR	Rewrite record with specified symbolic key (not ADM)
DISCARD .CUR	Delete current record (not ADM)
DISCARD .DIR	Delete record with specified relative key
DISCARD .IXDIR	Delete record with specified symbolic key (not ADM)

Transaction Control Instructions (not ADM)

COMMIT	Release the records accessed during the current transaction (dummy instruction for ADM)
COMMIT .PROT	Release the records accessed during the current transaction except those on the specified files (only EDM)
COMMIT .REL	Release the records accessed during the current transaction, on the specified files only
ROLLBCK	Rollback the current transaction (only EDM)

6.2 OPEN FILE

The Open instruction links a data file and any associated indexes to a data set identifier. Open file can be used to open an existing file or to create new files, both indexed (not for ADM) and non-indexed. An existing file or file structure can be opened for input only (read only) or for input and output.

In EDM, an implicit Commit is executed after a successful Open.

Operands to the instruction define:

- the File Parameter block
- the Open mode
- the Sharability

The No Wait option is not allowed for the Open instruction.

File Parameter Block

The application must supply information about the file and any associated index files to be opened, in the File Parameter block.

Chapter 5 contains the layout of a File Parameter block required for opening files. Bytes 1 to 66 are required for all files. Bytes 67 to 90 are required for all indexed files. Bytes 87 to 90 must be repeated for each index item of a concatenated key (only for EDM), and bytes 79 to 90 must be repeated for every index defined for the file.

The parameters required for each Open mode are found in the diagram below. The items that need not be supplied must be filled with binary zeroes, and data management will set them to the proper value if they are relevant for the file, after opening the file.

OPEN .IN
OPEN .INOUT
OPEN .DOUT
OPEN .EXT
OPEN .SOUT

File Parameters required when opening a file.

Parameter	Open Mode				
	.IN	.SOUT	.DOUT	.EXT	.INOUT
Record Length		x	x		
Blocking Factor		x	x		
File Organization	x	x	x	x	x
Device Type		x	x		
I/O Option	x	x	x	x	x
File Name	x	x	x	x	x
Logging Type	x	x	x	x	x
Growth Factor		x	x	x	x
Data Volume Name(s)	x	x	x	x	x
File Section Size(s)		x	x		

```
OPEN .IN
OPEN .INOUT
OPEN .DOUT
OPEN .EXT
OPEN .SOUT
```

Additional parameters for indexed files

Parameter	Open Mode .IN	.SOUT	.DOUT	.INOUT	
Index Volume Name	x	x	x	x	
Index Size		x	x		only EDM
Number of Indexes	x	x	x	x	
Internal Index Id					
Index Type	x	x	x	x	
Conditional Index	x	x	x	x	
Expression Value	x	x	x	x	only EDM
Conditional Item Displacement	x	x	x	x	
Conditional Item Value	x	x	x	x	
Number of Key Items	x	x	x	x	
Key Item Displacement	x	x	x	x	
Key Item Length	x	x	x	x	

The numeric items in the File Parameter block are binary values, the alphanumeric fields such as file name, must contain ISO-7 characters.

Since the File Parameter block must be on a word boundary, it is advisable to place it before any BCD or STRG declarations in the workblock.

Opening an existing indexed file under SDM

When an indexed file structure is opened under SDM, only the indexes specified in the File Parameter block are opened. For each index that is opened, a currency buffer is reserved and the master index is read into memory.

```
OPEN .IN
OPEN .INOUT
OPEN .DOUT
OPEN .EXT
OPEN .SOUT
```

When an indexed file is opened with Open mode Input under SDM, only the indexes that are needed for record access need to be specified. It is also possible to set the Number of Indexes and all other index parameters except Index Volume Name, to zero. In that case all the indexes of the file will be opened.

When an indexed file structure is opened for input/output under SDM, all the indexes of the file should be opened. If not, the indexes that have not been opened will not be updated when the data file is updated and then the files will be inconsistent.

Opening an existing indexed file under EDM

When an indexed file structure is opened under EDM, the I file is opened and all indexes are updated when the data file is updated. Only the indexes needed for record access (either for input only or input/output) need to be specified. The indexes may be specified in any order.

When opening and indexed file under EDM it is also possible to set the number of indexes and all the index parameters except for the Index Volume Name to zero in the File Parameter block. The index descriptor block from the I file is then read into the File Parameter block. The File Parameter block length specified in the instruction determines the number of indexes that are opened. The indexes are opened in the order in which they are defined in the index descriptor block.

Open Mode

The Open mode specifies the type of access for which the file is opened. The instructions Close File and Read File Parameters (DSC) are allowed for every Open mode.

There are five Open modes:

.IN — Input. Records can be read from the file but not written to it. The only record handling instructions allowed are all types of READ and POSIT.

.INOUT — Input/Output. Records can be read from and written to the file. All record handling instructions are allowed.

.SOUT — Sequential Output. A new standard file is created and records can be written to the file sequentially. The only record handling instruction allowed is WRITE .SEQ.

```
OPEN .IN
OPEN .INOUT
OPEN .DOUT
OPEN .EXT
OPEN .SOUT
```

The new file is not formatted at the Open instruction. When the file is closed, the part after the LRN will be formatted automatically.

In EDM only, Open .SOUT can also be used for the creation of a new indexed file. The instruction WRITE .IXSEQ is then allowed. An E file created with Open .SOUT will be formatted by the Open instruction.

.DOUT Direct Output. A new file is created and formatted, and records can be written to the file directly, via the relative key, and for indexed files, via the prime key. The only record handling instructions allowed are WRITE .DIR and WRITE .IXDIR. For Standard files, WRITE .SEQ is also allowed.

.EXT Extend. Records can be added sequentially to an existing standard file. The only record handling instruction allowed is WRITE .SEQ. The added file extent is not formatted. When the file is closed, the part after the LRN will be formatted automatically.

Open modes .IN, .INOUT and .EXT are only allowed for existing files.
Open modes .SOUT and .DOUT can only be used for creation of new files.

In SDM and ADM, a second task opening the same file must specify the same Open mode as the first task that opened the file. In EDM the Open mode need not be the same.

Sharability

The Sharability specifies the protection required for the opened file. There are three types of Sharability:

.NPROT Unprotected. The file can be opened by all tasks, and no records are protected. This is only allowed with Open mode .IN.

.PROT Protected. The file can be opened by all tasks, but any accessed records are held under exclusive access for the requesting task until a Commit, Rollback (EDM only) or Close instruction is executed. This is only allowed with OPEN mode .IN or .INOUT. Under EDM, when a file is opened with Sharability PROT, instructions with the No Wait option (see M21A) are not allowed for the file.

OPEN .IN
OPEN .INOUT
OPEN .DOUT
OPEN .EXT
OPEN .SOUT

.EXCL Exclusive. The complete file is held under exclusive access for the task that issues the Open instruction, and no other task can open it. This Sharability is allowed for all Open modes and it must be used with Open mode .SOUT, .DOUT or .EXT, or if the file is to be deleted with a Close .DROP instruction.

In SDM and ADM, a second task opening the same file must specify the same Sharability as the first task that opened the file. If the Sharability is .EXCL, a second task can not open the same file.

In EDM, when a file is opened protected (.PROT), and transaction logging is not done, file consistency may be lost in the case of an automatic rollback.

In EDM, the Sharabilities need not be the same. If the Sharabilities specified are not in conflict the file will be opened and also be available to the second task. The combinations of Sharabilities allowed in EDM are shown in the diagram below.

task 1 / task 2	.NPROT	.PROT	.EXCL
.NPROT	yes	yes	no
.PROT	yes	yes	no
.EXCL	no	no	no

Return Information

Condition Register

After the execution of the Open instruction, the Condition Register will be set to one of the following values:

CR Value	Meaning
0	File open successful
2	Error

OPEN .IN
OPEN .INOUT
OPEN .DOUT
OPEN .EXT
OPEN .SOUT

More information may be found in the Status Word and the Return Status. The Status Word is obtained with the XSTAT instruction and the Return Status with the RSTAT instruction. All possible values are found in Chapter 10.

Currency

After a successful Open instruction the CRN and the currency for the indexes will be set to zero so that the first data record is accessed by the first Read Sequential instruction and the record associated with the first index entry by a Read Indexed Sequential instruction.

Corrupt EDM File

In EDM the first byte of the I file is a status byte, indicating if the files are in a consistent state (status byte =0) or corrupt (status byte =1). If at an Open it appears that the files are corrupt, the Open is unsuccessful. In that case the value of the Condition Register will be zero, the Status Word has bit 8 set, the value of the Return Status = 2 (I/O error) and the value of the Supplementary Return Status = 254 (File Corrupt). If this occurs it is best to run the recovery and restart the system.

Example of the OPEN Instruction

```
OPEN     DSDK1,.INOUT,.PROT,PBLOK,LEN
```

DSDK1 This is the data set identifier for the disk file.

.INOUT Open mode. An Open mode of .INOUT permits execution of all data management instructions.

.PROT Sharability .PROT causes all records accessed to be held under exclusive access for the task, but allows other tasks to access other records in the file.

PBLOK This is the name of the string data item containing the File Parameter block.

LEN This is a binary data item containing the used length of the File Parameter block. In the above declaration this data item holds the value 66.

```
OPEN .IN
OPEN .INOUT
OPEN .DOUT
OPEN .EXT
OPEN .SOUT
```

Example of a File Parameter Block

The File Parameter block in this example has been defined as a dummy structure in the Data Division.

```
CB1       STRUC
DUMM      STRG      22X'0'
FILEORG   STRG      2X'0'
DUMO      STRG      2X'0'
RWOPTN    STRG      2X'01'
DUM1      STRG      4X'0'
FILENAM   STRG      8C'ACCDEV  '
LOGG      STRG      2X'03'
GRWTH     STRG      2X'10'
VOL1      STRG      6C'DEVMT1'
DUM3      STRG      8X'0'
VOL2      STRG      6C' '
DUM4      STRG      8X'0'
VOL3      STRG      6C' '
DUM5      STRG      8X'0'
VOL4      STRG      6C' '
DUM6      STRG      8X'0'
          STRUCE    CB1
*
DB1       DSTRUC    CB1
PBLOK     STRG      66
          STRUCE    DB1
```

```
OPEN .IN
OPEN .INOUT
OPEN .DOUT
OPEN .EXT
OPEN .SOUT
```

6.3 CLOSE FILE

The Close file instruction is issued by a task to indicate that it no longer requires access to the file. It is also possible to delete a file with the CLOSE instruction.

The Close instruction causes the following events to occur:

- EDM and SDM execute an implicit Commit. Any records held under exclusive access when the instruction is executed are released.
- If the file has been opened exclusive, The exclusive access is now released and the file may be opened by other tasks.
- The block buffer is written to disk unless the same block is currently accessed by another task.
- If no other task currently has this file open, the LRN or, for EDM files, the start of the free record chain, is updated in the VTOC.
- If a standard file has been created or extended the part after the LRN will be formatted.

The No Wait option is not allowed for the Close instruction.

For a file opened with Delay option (see section 8.7.2), only after a successful Close instruction is it certain that all updates have been written to the files on disk.

Close and Delete File

An operand to the instruction may be used to specify that the file must be deleted after execution of the Close.

.DROP Delete file. The file will be deleted after the Close. The VTOC records of the data file and the index file if any, are deleted from the volumes where they reside. The files can no longer be accessed. Deletion is only allowed for files opened with Sharability .EXCL.

Return Information

The Condition Register will be set to one of the following values as a result of the execution of this instruction.

CR Value	Meaning
0	File successfully closed
2	Error

More information may be found in the Status Word and the Return Status. The Status Word is obtained with the XSTAT instruction and the Return Status with the RSTAT instruction. All possible values are found in Chapter 10.

```
CLOSE
CLOSE .DROP
```

If Close .DROP is issued for a file that is not under exclusive access, a normal Close is executed. The Condition Register will be zero and bit 8 is set in the Status Word. The Return Status value will be 6 "Illegal Close option".

When errors occur during execution of a Close, no recovery can be done. EDM will as much as possible leave files and Monitor table in a consistent state. In EDM, the status byte of an I file involved may be set to 1, "file corrupt".

Examples of the Close Instruction

Close File

```
        CLOSE    DSDK1
```

DSDK1 This is the data set identifier for a disk file. The file currently open on this data set identifier will be closed.

Close and Delete File

```
        CLOSE    .DROP,TEMPFL
```

.DROP This is the keyword indicating that the file must be deleted.

TEMPFL This is the data set identifier for a disk file. The file currently open on this data set identifier will be closed and deleted.

```
CLOSE
CLOSE .DROP
```

6.4 SET CURRENT RECORD NUMBER (SET for ADM)

The POSIT instruction is used to set the currency for the data file or index file to such a value that a subsequent Read Sequential or Read Indexed Sequential instruction accesses the record specified.

The value of the CRN after a POSIT instruction will be one less than the relative key of the record specified, subject to the rules mentioned below, because the CRN is incremented prior to a sequential file access.

The POSIT instruction is allowed for files opened successfully for Input or Input-Output.

The record may be specified by the relative key (POSIT Direct) or by (part of) a symbolic key (POSIT Indexed).

In SDM, the CRN set by a POSIT instruction can only be used for Read (indexed) Sequential instructions. In EDM, the currency set by POSIT can also be used for Rewrite and Discard Current.

If the file was opened with Sharability .PROT, the record specified will be held under exclusive access for the task. If the record is already under protected access, the instruction is not successful.

Under EDM, if the file was opened with Sharability .PROT, the No Wait option is not allowed for this instruction.

In SDM and in EDM if no transaction logging is done, it is recommended to use the POSIT instruction to set the currencies for the files at the start of every transaction, after a Commit instruction. After a programmed (in EDM) or automatic Rollback during the transaction, the application may then go back to this point and restore the currencies of the files for reprocessing the transaction.

Operands to the instruction define:

- the access type
- the POSIT type
- the relative record key or the symbolic key

Access Type

There are two possible access types for the POSIT instruction:

.DIR Direct. The record is specified by the relative key. After the POSIT, the record required can be accessed with a READ .SEQ instruction.

.IXDIR Indexed Direct. The record is specified by a symbolic key or the part of a symbolic key. After the POSIT, the record required can be accessed with a READ .IXSEQ instruction.

POSIT .DIR
POSIT .IXDIR

The part of the key to be checked can be indicated by the application by specifying a keylength different from the actual keylength. The remainder of the key will not be checked and Positioning is on the first record with a key starting with the specified characters. This makes it possible to position at the start of a series of keys. In EDM, the keylength is the sum of the lengths of the key items.

POSIT Type

There are three types of positioning, each indicated by a value:

0 - Equal: The CRN is set to the record pointed to by the relative key specified. The record may be "used" or "free".

The index currency is set to the record identified by the specified symbolic key (or part of it) according to the index specified for the instruction. The record is always "used". If the record is not found, the error message "Key not found" is returned.

1 - Greater: The CRN is set to the next "used" record pointed to by the specified relative key.

The index currency is set to the record identified by the symbolic key (or part of it) next to the key specified, according to the index specified for the instruction.

2 - Not-less: The CRN is set to the record pointed to by the specified relative key if this record is "used", or to the next "used" record if this one is "free".

The index currency is set to the record identified by the symbolic key (or part of it) specified if it exists, or to the record identified by the next symbolic key according to the index specified for the instruction.

Examples of POSIT Types

Fig 6 - 1 is a stylised part of a file layout, with examples showing the effect of the POSIT type on the resulting value of the CRN for this file.

```
+-----------------+
|  POSIT .DIR     |
|  POSIT .IXDIR   |
+-----------------+
```

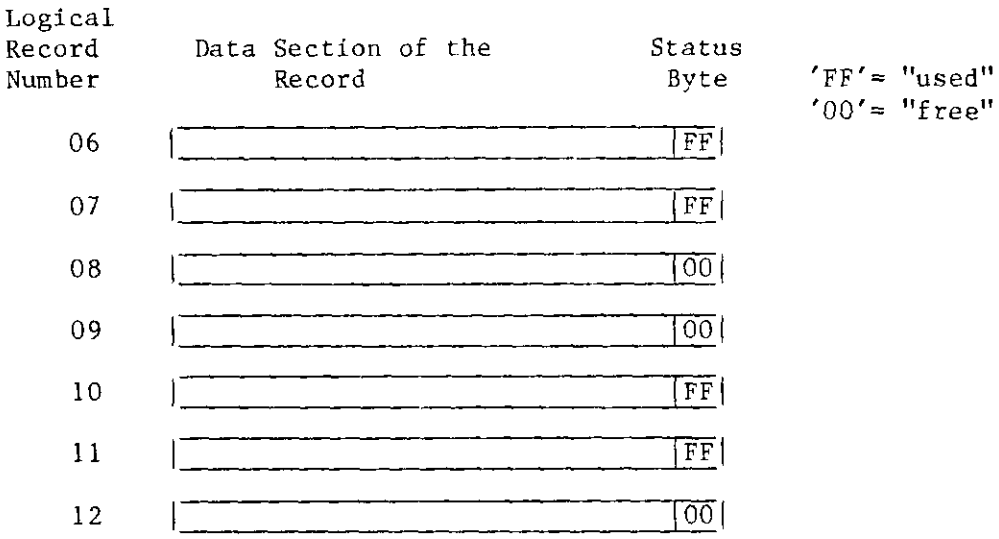


Fig 6-1 Example of POSIT Types

POSIT on Equal (type = 0)

```
TYPE    EQU     X'0'
        MOVE    RECNO,=X'A'
        POSIT   .DIR,DSDK1,RECNO,TYPE
```

The CRN will be set to 9 and the next record to be accessed will be record 10.

POSIT on Greater (type = 1)

```
TYPE    EQU     X'1'
        MOVE    RECNO,=X'7'
        POSIT   .DIR,DSDK1,RECNO,TYPE
```

The CRN will be set to 9 and the next record to be accessed will be record 10. For POSIT on Greater, the search for a used record starts at the record following that specified, so in this case the search for a used record will start at record 8.

POSIT on Not-less (type = 2)

```
TYPE    EQU     X'2'
        MOVE    RECNO,=X'A'
        POSIT   .DIR,DSDK1,RECNO,TYPE
```

The CRN is set to 9 and the next record to be accessed will be record 10. For POSIT on Not-less, the search for a used record starts at record 10.

```
POSIT .DIR
POSIT .IXDIR
```

Return Information

The Condition Register will be set to one of the following values as a result of the execution of this instruction.

CR Value	Meaning
0	Operation successful
1	End of File reached
2	Error
3	Attempt to use record number greater than number of records in file

The value of the currency (CRN) is not changed after an unsuccessful POSIT instruction. After an unsuccessful POSIT .DIR on a standard file, however, the Control Word will contain the relative key specified in the instruction.

After a POSIT .IXDIR with Posti type 0 (equal), the CR is set to 2 and bit 5 and 0 are set in the Status Word if the specified key is not found.

Examples of the POSIT instruction

POSIT .DIR

```
TYPE    EQU     X'1'
        POSIT   .DIR,DSDK1,RECNO,TYPE
```

DSDK1 This is the data set identifier for a disk file.

RECNO This is a decimal data item containing the relative key which will be used to set the CRN pointer.

TYPE This is a value expression indicating the POSIT type required: 0 = equal, 1 = greater, 2 = not-less.

POSIT .IXDIR

```
TYPE    EQU     X'2'
        MOVE    STRG,=C'                BAAAAAAA'
        MOVE    LEN,='8'
        MOVE    IXID,='3'
        POSIT   .IXDIR,DSDK1,STRG,LEN,TYPE,IXID
```

POSIT .DIR
POSIT .IXDIR

DSDK1 This is the data set identifier for a disk file.

STRG This is a string data item containing the key to be used to position the currency of the index file, at the displacement defined for the key in the File Parameter block.

LEN This is a binary data item defining the length of the key which is to be searched.

TYPE This is a value expression indicating the POSIT type required: 2 = not-less.

IXID This is a binary data item or a literal constant specifying the index to be used.

The currency for the index file I3 will be set to such a value that the record with the key "BAAAAAAA" if it exists, else the record with the next higher key value, will be accessed with a READ .IXSEQ instruction.

```
POSIT .DIR
POSIT .IXDIR
```

6.5 READ FILE PARAMETERS

The File Parameters and the Current Record Number (CRN) are obtained by the application with a Data Set Control (DSC) instruction. The File Parameter block is the block specified in the OPEN instruction. Some of the reserved fields will have had values inserted into them by the data management.

The layout of the File Parameter block is found in Chapter 5.

The complete index descriptor is read from disk so that the index descriptors returned are those specified when the file was created, and in that order, independent of the indexes specified when the file was opened.

The instruction is only accepted after a successful Open of the file concerned. The instruction is allowed for each Open mode and Sharability. The No Wait option is not allowed.

Operands to the instruction define:

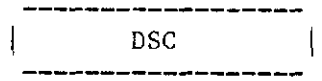
- The DSC control code. For Read File Parameters, this must be X'19'.
- The data item (BCD) where the CRN must be returned.
- The buffer where the File Parameters must be returned.
- The requested length. It is not necessary to read the complete File Parameter block. The file parameters will be copied into the buffer up to the specified length.

Return Information

The Condition Register will be set to one of the following values as a result of the execution of this instruction.

CR Value	Meaning
0	I/O successful
2	Error

The CRN is not affected.



Example of the DSC X'19' Instruction

An example of the DSC instruction used to obtain the File Parameters is shown below:

```
FILEP   EQU     X'19'
        DSC     DSDK1,FILEP,COUNT,STRG,LEN
```

DSDK1 This is the data set identifier for a disk file.

FILEP This is an equate identifier containing the control code. X'19' is the control code for reading the file parameters.

COUNT This is a data item (BCD), which after execution of the instruction, will contain the Current Record Number (CRN).

STRG This is a string data item into which the file parameters will be copied.

LEN This is a binary data item, defining the length of the buffer. After execution of the DSC instruction, LEN will contain the number of bytes actually transferred. The number of bytes transferred never exceeds the specified buffer length.

For example, if the File Parameter of a sequential file without indexes is read and LEN is set to 100, LEN will have the value 66 after the instruction. On the other hand, if LEN is set to 36, only the first 36 bytes of the File Parameter block will be transferred to the buffer.

6.6 READ RECORD

The READ instruction is used to read records into the application buffer. Records can be read from a file opened with Open mode .IN or .INOUT. The record read will become the current record for the requesting task.

If the file was opened with Sharability .PROT, the record read will be held under exclusive access for the requesting task until the task releases it with a COMMIT, ROLLBCK (only in EDM) or CLOSE.

Characters are read from the record into the string buffer specified in the instruction until the number of characters given in the length operand have been transferred or the end of the record is reached. If the number of characters specified in the length operand is less than the record length, the Condition Register will be set to 2 and bit 12 (Illegal length) is set in the Status Word.

Execution of the task issuing the READ will be suspended until the instruction is completed, unless the No Wait option is specified in the instruction. Under EDM, No Wait is not allowed for files opened with Sharability .PROT.

Operands to the instruction define:

- Access type
- Application buffer
- Requested Length

Access Type

The access type may be

.SEQ — Sequential. This access type is allowed for non-indexed and indexed files opened with Open mode .IN or .INOUT. The "used" record following the current record according to the relative key will be read. Records with status "free" are ignored. The CRN is incremented before the record is accessed. When a file is opened, the CRN is preset to zero so the first READ .SEQ instruction will read the first record in the file.

End of File is returned when the last record of the file written by Write Sequential instructions, has been read.

.DIR — Direct. This access type is allowed for standard and indexed files opened with Open mode .IN or .INOUT. The record to be read is identified by the relative key, specified by the application.

```
READ .SEQ
READ .DIR
READ .IXSEQ
READ .IXDIR
```

If the relative key points to a record with status "free", the Condition Register will be 2 and bit 0 and 4, "No Data", is set in the Status word. If the relative key points to a record outside the physical data file, the Condition Register is set to 3 and bit 2, "End of Medium", is set in the Status Word.

.IXSEQ Indexed Sequential. This access type is only allowed for indexed files opened with Open mode .IN or .INOUT. Records are read in the sequence in which the symbolic keys occur in the index specified. If the end of the index has been reached, the condition Register is set to 1 and bit 3, "End of File", is set in the Status Word.

If the current record has the same key as the next record, the Condition Register will be zero and bit 6, "Duplicate Key", is set in the Status Word. This is not an error message but an indication for the application.

When an indexed data file is opened, the currency for all indexes is set to zero so that the first READ .IXSEQ instruction will access the record associated with the first index entry in the index specified.

SDM only maintains one index currency at the time. This is the most recently used index. The index currency of the other indexes is set to zero.

EDM maintains the index currency for all indexes opened for the file.

.IXDIR Indexed Direct. This access type is only allowed for indexed files opened with Open mode .IN or .INOUT. The record to be read is identified by a symbolic key. The key must be supplied by the application, in the application record buffer at the displacement defined in the index descriptor for the index used.

If the key specified is a key for which duplicates exist in the file, the record associated with the first index entry with the same key, is read. The other records with that key can then be accessed by READ .IXSEQ instructions. ("Duplicate Key" is not returned after READ .IXDIR.)

If the requested symbolic key is not present in the specified index, the Condition Register is set to 2 and bit 5 "Key not Found" is set in the Status Word.

READ .SEQ
READ .DIR
READ .IXSEQ
READ .IXDIR

Return Information

After completion of the READ instruction, the following information is returned:

Condition Register

The Condition Register will be set to one of the following values as a result of the execution of a READ instruction:

CR Value	Meaning
0	Read successful
1	End of file reached
2	Error
3	End of device reached

More information can be obtained from the Status Word and the Return Status. All possible values are found in Chapter 10.

Effective Length

The actual number of characters read is returned in the length operand.

The CRN

After successful completion of the READ instruction, the CRN points to the record read, independent of the access type. After Read Indexed Sequential and Read Indexed Direct instructions, the currency of the index file is set to such a value that the record associated with the next index entry is accessed with the next READ .IXSEQ instruction.

If the READ is not successful, the value of the CRN is the same as before execution of the instruction, with the following exception:

After I/O errors during a READ .SEQ on a standard file, the CRN will point to the record that would have been read if the instruction had been successful. The next READ .SEQ will than access the next record. In this way it is possible to pass badspots in a standard file.

The Control Word

After successful completion of the READ instruction, the relative key of the current record is returned in the Control Word. The Control Word can be obtained by the application with a GETCW instruction.

If the READ is not successful, the value of the Control Word is the same as before execution of the instruction, with the following exception:

READ .SEQ
READ .DIR
READ .IXSEQ
READ .IXDIR

After I/O errors during a READ .SEQ or a READ .DIR on a standard file, the Control Word will contain the relative key of the record that would have been read if the instruction had been successful.

Examples of the READ Instruction

Read Sequential (READ .SEQ)

READ .SEQ,DSDK1,STRG,LEN

.SEQ Sequential. The CRN will be incremented and the first "used" record following the current record for data set DSDK1 will be read.

DSDK1 This is the data set identifier of the disk file.

STRG This is a string data item into which characters will be read. It should be large enough to hold one record.

LEN This is a binary data item containing the number of characters to be read (the record length). After execution it will contain the actual number of characters read.

Read Direct (READ .DIR)

READ .DIR,DSDK1,STRG,LEN,RECNO

.DIR Direct. The record with the relative key value specified in the operand RECNO will be read and the CRN pointer will be updated.

DSDK1 This is the data set identifier for a disk file.

STRG This is a string data item into which the characters will be read. It should be large enough to hold one record.

LEN This is a binary data item containing the number of characters to be read (the record length). After execution this will contain the actual number of characters read.

RECNO This is a decimal data item containing the relative key value of the record to be read.

READ .SEQ
READ .DIR
READ .IXSEQ
READ .IXDIR

Read Indexed Direct (READ .IXDIR)

```
READ    .IXDIR,DSDK1,STRG,LEN,IXID
```

.IXDIR Indexed direct access. The record to be read is specified by a symbolic key.

DSDK1 This is the data set identifier for a disk file.

STRG This is a string data item into which the record will be read. Before the instruction is executed, it must contain the symbolic key of the record to be accessed at the displacement defined for the key in this index.

LEN This is a binary data item containing the number of characters to be read (the record length). After execution it will contain the actual number of characters read.

IXID This is a binary data item or literal constant specifying the internal index identifier (index number) of the index to be used.

Read Indexed Sequential (READ .IXSEQ)

```
READ    .IXSEQ,DSDK1,STRG,LEN,IXID
```

.IXSEQ Indexed Sequential. The record to be read is the next in sequence in the specified index.

DSDK1 This is the data set identifier for a disk file.

STRG This is a string data item into which characters will be read. It should be large enough to hold one record.

LEN This is a binary data item containing the number of characters to be read (the record length). After execution this will contain the actual number of characters read.

.IXID This is a binary data item or literal constant specifying the index to be used.

```
READ .SEQ
READ .DIR
READ .IXSEQ
READ .IXDIR
```

6.7 WRITE RECORD

The WRITE instruction is used to write new records from the application buffer to the file. Records can be written to files opened with Open mode .DOUT, .SOUT, .EXT or .INOUT. The record written will become the current record for the requesting task after Write Sequential instructions on files opened with Open mode .SOUT or .EXT.

New records can only be written to records with the status "free". If the WRITE is successful, the record status will be set to "used".

The new record is written from the application record buffer. The buffer should be long enough to hold one record. The characters are transferred sequentially until the end of the buffer is reached. If the buffer is shorter than the number of characters defined in the record length, the Condition Register is set to 2 and bit 12, "Incorrect Length", is set in the Status Word.

If the file is indexed, the new record is identified by the prime key. The new data record is written to the first free position after the LRN (for SDM) or to the first free record in the file (for EDM) and the index entries are inserted in the index files in the correct places. Under SDM, all index files should have been opened. Master indexes are not updated. Under EDM, all indexes are updated.

If the file was opened with Sharability .PROT, the record written will be held under exclusive access for the requesting task until the task releases it with a COMMIT, ROLLBCK (only in EDM) or CLOSE.

Execution of the task issuing the WRITE will be suspended until the instruction is completed, unless the No Wait option is specified in the instruction. Under EDM, No Wait is not allowed for files opened with Sharability .PROT.

Operands to the instruction define:

- Access type
- Application buffer

Access Type

The access type may be

.SEQ Sequential. WRITE SEQ is allowed for standard files opened with Open mode .SOUT, .DOUT, .EXT or .INOUT.

```
WRITE .SEQ
WRITE .DIR
WRITE .IXSEQ
WRITE .IXDIR
```

The record will be written to the first "free" record following the record pointed to by the Last Record Number pointer (LRN). After a successful WRITE .SEQ the LRN is incremented. The CRN is only incremented if the Open mode was .SOUT or .EXT. When a new file is created, the LRN is set to zero so the first WRITE .SEQ instruction will write the first record in the file.

If the end of the physical file space has been reached and the file is not automatically enlarged, the Condition Register will be set to 1 and bit 3, "End of File", is set in the Status Word.

.DIR Direct. WRITE .DIR is allowed for standard files opened with Open mode .DOUT or .INOUT. The record to be written is identified by the relative key, specified by the application. The currency is not affected by Write Direct instructions.

If the relative key points to a record with status "used", the Condition Register is set to 2 and bit 9, "Duplicate Error", is returned in the Status Word.

If the relative key points to a record outside the physical data file, the Condition Register is set to 3 and bit 2, "End of Medium" is set in the Status Word together with bit 0.

.IXDIR Indexed Direct. WRITE .IXDIR is allowed for indexed files opened with Open mode .DOUT or .INOUT. The prime key must be supplied by the application, in the application record buffer at the displacement defined in the index descriptor. The currency is not affected by Write Indexed Direct instructions.

If the prime key exists already in the primary index file, the Write is not executed. The Condition Register is set to 2 and bit 9, "Duplicate Key", is set in the Status Word.

.IXSEQ Indexed Sequential (only EDM). WRITE .IXSEQ is only allowed for indexed files opened with Open mode .SOUT. Records must be supplied by the application in sequence of the prime key. All indexes defined for the file are updated. The currency is not affected by Write Indexed Direct instructions.

If the prime key of the new record is not greater than the prime key of the last record written, the Write is not executed. The Condition Register is set to 2 and bit 5 "Invalid Key", is set in the Status Word.

WRITE .SEQ
WRITE .DIR
WRITE .IXSEQ
WRITE .IXDIR

Return Information

After completion of the WRITE instruction, the following information is returned:

Condition Register

The Condition Register will be set to one of the following values as a result of the execution of a WRITE instruction:

CR Value	Meaning
0	Read successful
1	End of file reached
2	Error
3	End of device reached

More information can be obtained from the Status Word and the Return Status. All possible values are found in Chapter 10.

The CRN

The CRN will be set to the record just written for a standard file opened for Output Sequential or Extend. For other file types and files opened in other Open modes, the CRN and the index currency are not affected.

If the WRITE is not successful, the value of the CRN is the same as before execution of the instruction.

The Control Word

After successful completion of the WRITE instruction, the relative key of the record written is returned in the Control Word. The control Word can be obtained by the application with a GETCW instruction.

If the WRITE is not successful, the value of the Control Word is the same as before execution of the instruction.

```
WRITE .SEQ
WRITE .DIR
WRITE .IXSEQ
WRITE .IXDIR
```

Examples of the WRITE Instruction

Write Sequential (WRITE .SEQ)

```
WRITE    .SEQ,DSDK1,STRG
```

.SEQ — Sequential. The record is written to the position after that indicated by the current contents of the LRN. The LRN pointer held in memory will be incremented.

DSDK1 — This is the data set identifier of the disk file.

STRG — This is a string data item from which characters are written. It should be large enough to hold one record.

Write Direct (WRITE .DIR)

```
WRITE    .DIR,DSDK1,STRG,RECNO
```

.DIR — Direct. The record is written to the position indicated by the relative key supplied by the application.

DSDK1 — This is the data set identifier of the disk file.

STRG — This is a string data item from which characters are written. It should be large enough to hold one record.

RECNO — This is a decimal data item (BCD) containing the relative key of the record to be written.

Write Indexed Direct (WRITE .IXDIR)

```
WRITE    .IXDIR,DSDK1,STRG
```

.IXDIR — Indexed Direct. The contents of STRG is written to the file opened on data set identifier DSDK1, and the corresponding index entries are inserted in the associated index files. In SDM, the record is written to the first free record after the LRN. In EDM, the record is written to the first free record in the free record chain of the data file.

DSDK1 — This is the data set identifier for a disk file.

STRG — This is a string data item from which characters will be written. It should be large enough to hold one record. The string data item must contain the prime record key at the displacement defined for the key of the primary index.

```
WRITE .SEQ
WRITE .DIR
WRITE .IXSEQ
WRITE .IXDIR
```

Write Indexed Sequential (WRITE .IXSEQ)

WRITE .IXSEQ,DSDK1,STRG

.IXSEQ Indexed Sequential. The records must be supplied in sequence of the prime key.

DSDK1 This is the data set identifier for a disk file.

STRG This is a string data item from which characters will be written. It should be large enough to hold one record. The string data item must contain the prime record key at the displacement defined for the key of the primary index.

```
WRITE .SEQ
WRITE .DIR
WRITE .IXSEQ
WRITE .IXDIR
```

6.8 REWRITE RECORD

The REWRITE instruction is used to write an updated record from the application buffer to the file. Records can only be rewritten to files opened with Open mode .INOUT. The Rewrite instruction does not affect the currency.

Records can only be rewritten to records with the status "used". Before a record is rewritten to the file it should have been read, but this is not checked by data management.

The record is written from the application record buffer. The buffer should be long enough to hold one record. The characters are transferred sequentially until the end of the buffer is reached. If the buffer is shorter than the number of characters defined in the record length, the Condition Register is set to 2 and bit 12, "Incorrect Length", is set in the Status Word.

Under SDM, if the file is indexed, all record keys must be unchanged.

Under EDM, if the file is indexed, the value of the prime key must not be changed. If alternate keys and conditional keys have been updated, all indexes are updated. New index entries are generated and the old ones removed from the index file.

If the file was opened with Sharability .PROT, the record rewritten will be held under exclusive access for the requesting task until the task releases it with a COMMIT, ROLLBCK (only in EDM) or CLOSE.

Execution of the task issuing the REWRITE will be suspended until the instruction is completed, unless the No Wait option is specified in the instruction. In EDM, No Wait is not allowed for files opened with Sharability .PROT.

Operands to the instruction define:

- Access type
- Application buffer

Access Type

.CUR Current. Rewrite Current is allowed for standard and indexed files opened with Open mode .INOUT. The current record is rewritten.

SDM: After a POSIT instruction, Rewrite Current will rewrite the record last accessed by a Read instruction.

EDM: After a POSIT instruction, Rewrite Current will rewrite the record selected by the POSIT instruction.

```
+-----------------+
| REWRITE .CUR    |
| REWRITE .DIR    |
| REWRITE .IXDIR  |
+-----------------+
```

.DIR Direct. Rewrite Direct is allowed for standard files opened with Open mode .INOUT. The record to be rewritten is identified by the relative key, specified by the application.

If the relative key points to a record with status "free", the Condition Register is set ot 2 and bit 4, "No Data", is set in the Status Word together with bit 0.

If the relative key points to a record outside the physical data file, the Condition Register is set to 3 and bit 2, "End of Medium" is set in the Status Word together with bit 0.

.IXDIR Indexed Direct. Rewrite Indexed Direct is allowed for indexed files opened with Open mode .INOUT. The prime key of the record must be unchanged.

In SDM, all keys must be unchanged.

In EDM, if alternate keys have been changed, the associated indexes will be updated.

Under EDM, if one of the alternate keys for which duplicates are allowed exist already, the Condition Register will be zero and bit 6 "Duplicate Key" is set in the Status Word.

If one of the alternate keys for which duplicates are not allowed exist already, the condition Register is set to 2 and bit 9 "Duplicate Key not allowed" is set in the Status Word. The Rewrite is not executed.

If the prime key has been changed, the Rewrite is not executed. The Condition Register will be zero and bit 8, "DM rule violated", is set in the Status Word. The Return Status is set to 2 and for EDM the Supplementary Return Status has the value 214.

```
REWRITE .CUR
REWRITE .DIR
REWRITE .IXDIR
```

Return Information

After completion of the REWRITE instruction, the following information is returned:

Condition Register

The Condition Register will be set to one of the following values as a result of the execution of a REWRITE instruction:

CR Value	Meaning
0	Read successful
2	Error
3	End of device reached

More information can be obtained from the Status Word and the Return Status. All possible values are found in Chapter 10.

The CRN

The currency is not affected by Rewrite instructions. In EDM however, if the currency had been set by a POSIT instruction, the currency will be set to the record rewritten with a Rewrite Current so that a subsequent Read Sequential will access the record following the record rewritten.

The Control Word

After successful completion of the REWRITE instruction, the relative key of the record rewritten is returned in the Control Word. The Control Word can be obtained by the application with a GETCW instruction.

If the REWRITE is not successful, the value of the Control Word is the same as before execution of the instruction, with the following exception:

After I/O errors during a REWRITE .CUR or a REWRITE .DIR on a standard file, the Control Word will contain the relative key of the record that would have been rewritten if the I/O had been successful

REWRITE .CUR
REWRITE .DIR
REWRITE .IXDIR

Examples of the REWRITE Instruction

Rewrite Current (REWRITE .CUR)

```
REWRITE    .CUR,DSDK1,STRG
```

.CUR — Current. The contents of the string data item is written back to the record indicated by the current record number. The contents of the CRN are not affected by execution of this instruction.

DSDK1 — This is the data set identifier of the disk file.

STRG — This is a string data item. It contains the characters to be written and must be large enough to hold one record.

Rewrite Direct (REWRITE .DIR)

```
REWRITE  .DIR,DSDK1,STRG,RECNO
```

.DIR — Direct. The contents of the string data item is written back to the record with the specified relative key.

DSDK1 — This is the data set identifier for a disk file.

STRG — This is a string data item from which characters are written. It should be large enough to hold one record.

RECNO — This is a decimal data item containing the relative key of the record to be rewritten.

Rewrite Indexed Direct (REWRITE .IXDIR)

```
REWRITE  .IXDIR,DSDK1,STRG
```

.IXDIR — Indexed Direct. The contents of the string data item is written back to a record in the data file. The record overwritten as a result of this instruction will be the record whose prime key matches the prime key of the record held in the string.

DSDK1 — This is the data set identifier for the disk file.

STRG — This is a string data item from which characters will be written. It should be large enough to hold one record. The string data item must contain the prime record key at the displacement defined for the key of the primary index.

REWRITE .CUR
REWRITE .DIR
REWRITE .IXDIR

6.9 DISCARD RECORD

The Discard instruction is used to delete a record from the file. Records can only be deleted from files opened with Open mode .INOUT. Only records with status "used" can be deleted. The status will be set to "free".

If the file is indexed, the index files opened for the file are updated under SDM. Master indexes are not updated. All indexes should have been opened. Under EDM, all indexes are updated automatically.

If the file was opened with Sharability .PROT, the record deleted will be held under exclusive access for the requesting task until the task releases it with a COMMIT, ROLLBCK (only in EDM) or CLOSE.

Execution of the task issuing the DISCARD will be suspended until the instruction is completed, unless the No Wait option is specified in the instruction. Under EDM, No Wait is not allowed for files opened with Sharability .PROT.

Operands to the instruction define:

- Access type
- Application buffer

Access Type

The access type may be

.CUR Current. Discard Current is allowed for standard and indexed files opened with Open mode .INOUT. The current record is deleted.

SDM: After a POSIT instruction, Discard Current will delete the record last accessed by a Read instruction.

EDM: After a POSIT instruction, Discard Current will delete the record selected by the POSIT instruction.

.DIR Direct. Discard Direct is allowed for standard files opened with Open mode .INOUT. The record to be rewritten is identified by the relative key, specified by the application.

If the relative key points to a record with status "free", the Condition Register is set to 2 and bit 4, "No Data", is set in the Status Word together with bit 0.

```
DISCARD .CUR
DISCARD .DIR
DISCARD .IXDIR
```

If the relative key points to a record outside the physical data file, the Condition Register is set to 3 and bit 2, "End of Medium" is set in the Status Word together with bit 0.

.IXDIR Indexed Direct. Discard Indexed Direct is allowed for indexed files opened with Open mode .INOUT. The record is identified by the prime key.

Return Information

After completion of the DISCARD instruction, the following information is returned:

Condition Register

The Condition Register will be set to one of the following values as a result of the execution of a DISCARD instruction:

CR Value	Meaning
0	Discard successful
2	Error
3	End of Device

More information can be obtained from the Status Word and the Return Status. All possible values are found in Chapter 10.

The CRN

The CRN is not affected by Discard Direct or Indexed Direct instructions. After a Discard Current, the currency will be such that a subsequent Read Sequential or Read Indexed Sequential instruction will access the next record or the record associated with the next index entry according to the last used index.

The Control Word

After successful completion of the DISCARD instruction, the relative key of the record deleted is returned in the Control Word. The Control Word can be obtained by the application with a GETCW instruction.

If the DISCARD is not successful, the value of the Control Word is the same as before execution of the instruction, with the following exception:

```
DISCARD .CUR
DISCARD .DIR
DISCARD .IXDIR
```

After I/O errors during a DISCARD .CUR or a DISCARD .DIR on a standard file, the Control Word will contain the relative key of the record that would have been rewritten if the I/O had been successful.

Examples of the DISCARD Instruction

Discard Current (DISCARD .CUR)

```
DISCARD    .CUR,DSDK1
```

.CUR — Current. The current record is deleted from the file. If the file is indexed, the associated index entries are also deleted from the index files. After this instruction the currency is such that the next "used" record according to the last used access path is read with a Read (Indexed) Sequential instruction.

DSDK1 — This is the data set identifier of the disk file.

Discard Direct (DISCARD .DIR)

```
DISCARD  .DIR,DSDK1,RECNO
```

.DIR — Direct. The record with the specified relative key is deleted.

DSDK1 — This is the data set identifier for a disk file.

RECNO — This is a decimal data item containing the relative key of the record to be deleted.

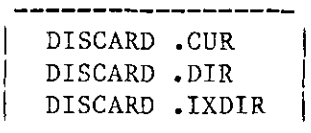
Discard Indexed Direct (DISCARD .IXDIR)

```
DISCARD  .IXDIR,DSDK1,STRG
```

.IXDIR — Indexed Direct. The record with a prime key with the same value as the prime key of the record in application buffer is deleted from the file. If the file is indexed, the associated index entries are also deleted from the index files.

DSDK1 — This is the data set identifier for the disk file.

STRG — This is a string data item from which characters will be written. It should be large enough to hold one record. The string data item must contain the prime record key at the displacement defined for the key of the primary index.



6.10 COMMIT

The COMMIT instruction is used to release records held under exclusive access on a file opened with Open mode .PROT, and to initiate the next transaction if transaction and/or function logging is done (only in EDM). Only a certain number of records may be held under exclusive access at any one time. This number is specified during Monitor generation.

ADM does not support the Commit instruction. Commit instructions may be used for compatibility with SDM or EDM and will be treated as dummy instructions.

In SDM and EDM, a COMMIT is executed automatically when a CLOSE instruction is executed.

In EDM only, a COMMIT is executed automatically after a successful OPEN instruction.

If transaction logging is done for the files, the transaction log information of the current transaction is deleted. If function logging is done, the function log buffers are written to the function log file on disk or tape.

NOTE

Commit does not result in any I/O operations to the files on disk. At what time the contents of the internal block buffer is written to the disk is determined by data management, independent of the transaction control functions.

The No Wait option is not supported for the COMMIT instruction.

Operands to the instruction define:

- Commit type
- The data sets to which the Commit type applies
- The data item where to store the return information

Commit Type

As an option, one of the following types may be specified. If no type is specified, all records under exclusive access for the task are released.

.REL Release. Commit with Release is used to release the records held under exclusive access for a task on the specified files only. Protected records from other files remain protected.

```
COMMIT
COMMIT .PROT
COMMIT .REL
```

Commit with Release defines a "sub-transaction" in the application. The transaction log information is deleted, if transaction logging is being done for the files. This means that the application is responsible for file consistency at the point where a COMMIT .REL is issued.

At a subsequent automatic or programmed (only in EDM) Rollback, the records of the files that were not released with the COMMIT .REL instruction will also be released.

.PROT Protect. (only EDM). Commit with protection is used to release the records held under exclusive access for a task except those on the specified files. This defines a "subtransaction" in the application. The transaction log information is deleted, if transaction logging is being done for the files. This means that the application is responsible for file consistency at the point where a COMMIT .PROT is issued.

At a subsequent automatic or programmed (only in EDM) Rollback, the records of the files specified in the COMMIT .PROT instruction will also be released.

Return Information

The Condition Register and the currency are not affected and the Control Word is not used by the COMMIT instruction.

The return information of the instruction is returned in the binary data item supplied in by the application. This may be set to one of the following values:

Value	Meaning
0	Successful completion.
-1	I/O error. Additional information may be found in the Status word. All possible values are listed in Chapter 10.
-2	Data Management rule violated. Additional information may be found in the Return Status and Supplementary Return Status. All possible values are listed in Chapter 10.

The values -1 and -2 can only be obtained when EDM is used and transaction or function logging is done.

COMMIT
COMMIT .PROT
COMMIT .REL

Examples of the COMMIT Instruction

Commit

```
COMMIT   PARM
```

All records held under exclusive access for the task are released.

PARM — This is a binary data item. After execution of this instruction it contains the return information.

Commit with Release (COMMIT .REL)

```
COMMIT.REL,PARM,DSDK1,DSDK3
```

.REL — Commit with Release. Exclusive access is released only for records from the files opened on data set identifiers DSDK1 and DSDK3.

PARM — This is a binary data item. After execution of this instruction it contains the return information.

Commit with Protection (COMMIT .PROT) (only for EDM)

```
COMMIT.PROT,PARM,DSDK1,DSDK3
```

.PROT — Commit with Protection. Exclusive access is released for all files except from those opened on data set identifiers DSDK1 and DSDK3.

PARM — This is a binary data item. After execution of this instruction it contains the return information.

COMMIT
COMMIT .PROT
COMMIT .REL

6.11 ROLLBACK (only for EDM)

The ROLLBCK instruction is used to abort the current transaction and to release records held under exclusive access on a file opened with Open mode .PROT.

If transaction logging is done, the before images of the records logged during this transaction will be re-applied to the disk files concerned. The files are then in the consistent state they were in at the previous Commit or Open, i.e. at the beginning of the transaction. If transaction logging is done, the currency for data and index files will also be reset to the value at the previous Commit. After that the transaction log information of the current transaction is deleted.

If function logging is done, the function log buffers are written to the function log file on disk or tape.

NOTE

If the previous Commit was a Commit with Protection or Commit with Release, Rollback will also release the records for the files that were not released. Consistency of these files is the responsibility of the program.

Rollback does not necessarily result in the before images of records being written back to the disk. They are restored to the internal block buffers in EDM.

The No Wait option is not supported for the ROLLBCK instruction.

The operand to the instruction defines:

- The data item where to store the return information

Return Information

After completion of the ROLLBCK instruction, the following information is returned:

Condition Register

The Condition Register is not affected and the Control word is not used by the ROLLBCK instruction.

| ROLLBCK |

The return information of the instruction is returned in the binary data item supplied in by the application. This may be set to one of the following values:

Value	Meaning
0	Successful completion.
-1	I/O error. Additional information may be found in the Status word. All possible values are listed in Chapter 10.
-2	Data Management rule violated. Additional information may be found in the Return Status and Supplementary Return Status. All possible values are listed in Chapter 10.

The values -1 and -2 can only be obtained when transaction or function logging is done.

The Currency

Only if transaction logging is done, the CRN and the currencies of the index files are reset to he values they had at the previous COMMIT or OPEN instruction.

If no transaction logging is done, the currencies are not affected.

Examples of the ROLLBCK Instruction

Rollback

```
ROLLBCK   CODE
```

CODE — After execution of the Rollback, the return information will be stored here.

6.11.1 Automatic Rollback

If EDM detects a deadlock situation, Rollback is performed automatically for the task whose request caused the deadlock situation. The records held under protected access by this task are released.

NOTE:

If the previous Commit was a Commit with Protection or Commit with Release, Rollback will also release the records for the files that were not released. Consistency of these files is the responsibility of the program.

If transaction logging is done for the files concerned, the before images of the records are written back to the file and the currency is reset to the value it had at the previous Commit.

When this happens, the Condition Register is set to 2 and bit 11, "Sequence Error / Rollback Performed" is set in the Status Word.

Deadlock

A deadlock situation arises if tasks require protected access to the same records and start waiting for each other to release them.

Before a task goes into a wait state to wait for a record, the protected record administration is checked by EDM. If this task then is already holding a record which the other task is waiting for, this is deadlock and the transaction for this task is rolled back. Deadlock situations with more than two tasks involved are also detected.

The situation of two tasks waiting for each other's records can be partially avoided if different tasks all follow the same sequence of accessing records.

In SDM, an automatic Rollback occurs if access is requested to a record held protected by another task. There is no check if this is a real deadlock situation. A COMMIT, releasing the record protection for all records, is then executed for the second task, with the risk of file inconsistency.

For SDM and for EDM if transaction logging is not used, the following application design is recommended to avoid inconsistent files.

All records needed for a transaction are read by the task, before any record is updated. In this way the task has all records involved under exclusive access for the duration of the transaction. If the transaction is rolled back as a result of a Read for a record which is not available, no information on the files will have been updated yet.

| ROLLBCK |