The data management function is discussed in the specific context of DOS/360 *and* TOS/360, *the disk and tape operating systems for intermediate* SYSTEM/360 *configurations.*

Explained are the processing routines of the data management facilities, collectively referred to as the input/output control system.

Techniques that keep the routines small in size, efficient in operation, and simple to use are emphasized.

Data management concepts for DOS/360 and TOS/360

by A. R. Cenfetelli

The growing diversity and increasing complexity of new applications demand more efficient data-handling devices and methods. The computer designers have responded with a large complement of different input/output (I/O) and auxiliary storage devices. However, the majority of users in the small-to-intermediate computer class desire to spend most, if not all, of their programming effort in solving their own problems. They wish to expend a minimum of effort in coping with the intricacies of the various I/O devices, or in developing new techniques for retrieval of information. The data management facilities for DOS/360 and TOS/360, discussed in this paper,[1] have been designed especially for IBM SYSTEM/360 users in this class.

objectives

The objectives were to provide a collection of data-handling routines with as many big-system capabilities as possible, and yet keep the routines small in size, efficient in operation, and simple to use. A method that well accomplishes these objectives is the tabular-modular technique discussed in this paper. The data management routines for DOS/360 and TOS/360 are collectively referred to as the I/O control system (IOCS). A distinction is made between "logical" and "physical" routines: LIOCS and PIOCS. The logical routines link user programs with the physical routines that are included in the Supervisor of the control program. This paper is mainly concerned with the general LIOCS concepts, but also explains PIOCS in its relation to the logical routines.

In most data processing installations, a typical program processes multiple data files (also called data sets[2]) that are stored

on similar I/O devices. A typical application might be the updating of a data set on tape, with the updating information also coming from tape. In this case, three tapes are involved in the process. Let us call the original tape *oldmaster*, the update tape *update*, and the tape to be created *newmaster*. The characteristics of *oldmaster* and *newmaster* are the same, with the exception that one is an input file and the other an output file. The *update* file is a tape input file, but its data characteristics may be different from those of the *oldmaster* file in length of records, blocking factor,[3] etc. In this case, the normal processing loop would be: retrieve a record from the *update* file, retrieve the corresponding record from the *oldmaster* file, update the *oldmaster* record, and place the updated record in the *newmaster* file. Since there is only one I/O-type device, it follows that only one routine is necessary for record handling. This routine employs the modular technique. Since the routine, or module, can operate on one or more data sets having different characteristics within the same job, it is important that it does not make any permanent self-modification for any of the data sets; this would render the module useless for the simultaneous processing of other data sets. Even temporary self-modification would be quite costly in time and main storage because, each time a date set is serviced, portions of the original module must be saved for later restoration.[4]

It is important that the module recognize the characteristics of each data set it will manipulate as well as the current status of the records being processed (such as their location in the block, etc.). The tabular technique allows the characteristics and status of the data set to be communicated to the module by way of a data set definition table called the DTF (define the file) table. This table consists of a small list of constants or parameters, such as record length, blocking factor, and work areas needed to hold the current status of the data set being processed. For the above example, three DTF tables and one module would be required to perform the necessary data-handling functions. This approach helps to reduce the requirements for main storage and data handling, and also has the simplicity made possible by separating general data-handling characteristics from the detailed specifics required for each data set.

storage labels

Most of today's data processing applications make use of I/O devices that utilize removable storage media, such as tape reels, disk packs, and data cells. In SYSTEM/360, these units of auxiliary storage are referred to as *volumes*. Since these volumes generally contain important data, a means of identification and protection is needed to recognize and prevent inadvertent use of the data sets stored on them. The data management for DOS/360 and TOS/360 accomplishes this by using a set of label records. The following label types can be recorded on each volume:

- A label containing a volume serial number. In the case of direct-access storage devices (DASD's), the volume label also

contains the location of a volume table of contents (VTOC), which is composed of additional labels applicable to the data sets stored on the media.

- A label for each data set stored on the volume, containing such information as the name of the data set, its file identification, the file creation and expiration dates, etc. On a DASD, this label is located in the VTOC; on tape, it follows the volume label, precedes the user's data set, and is called a header label.[5]

A set of routines for standard label-handling procedures, commonly called *open* and *close*, is provided.

processing methods

The LIOCS consists of eight processing routines, also called access methods, for DOS/360 users and five for TOS/360 users.[6] Seven of these methods use the tabular-modular concept, and one requires only the tabular technique. Six of the eight methods accommodate only sequential or consecutive (serial or tapelike) processing. A seventh method is designed primarily for random processing on DASD's, although it can be used for sequential processing to a certain degree. The eighth method can be used for both random and sequential processing on DASD's.

To use the access methods, the appropriate module and associated DTF tables needed must be generated or assembled. These tables and modules can be assembled independently or with the user program. Normally, however, the modules are generated separately because the same module can be used by many programs. Since the DTF tables are data-set dependent, they may or may not be assembled with the user's routine. The ability to assemble components separately and later to combine them into an integrated function offers the advantages of reduced assembly time and minimum housekeeping because a module can be used with a number of different programs, all requiring similar functions for the same (or similar) I/O device types. This is particularly meaningful at debugging time when programs may require many reassemblies. Another advantage of generating the modules is that certain functions not needed by all users can be provided optionally. Thus, users not needing these functions can reduce their overhead and realize savings in main storage.[7]

The user can generate the various modules and DTF tables required by means of a set of parameter cards. These cards contain a description of the function to be performed and the data set to be processed, as well as similar details that cannot be assumed by the data management functions. Generation is accomplished by using the DOS/TOS assembler with its macroinstruction facility and normal assembly techniques.

The access methods are commonly referred to by declarative macroinstructions which are listed in Table 1 by DTF and module type. To execute these macroinstructions, the user must issue an imperative macroinstruction in his program and designate the type of operation to be performed, the DTF table to be used, etc. The imperative macroinstructions and their functions are given

Table 1 Declarative macroinstructions

Table-type instruction	*Defines a data set associated with*
DTFCD	Card device
DTFCN	Printer keyboard (console printer)[8]
DTFDA	DASD (processed randomly; DOS/360 only)
DTFIS	DASD (processed either randomly or sequentially; DOS/360 only)
DTFMT	Magnetic tape
DTFPR	Printer device
DTFPT	Paper tape device
DTFSD	DASD (processed sequentially; DOS/360 only)

Module-type instruction	*Defines the module type to be used with*
CDMOD	DTFCD
DAMOD	DTFDA
ISMOD	DTFIS
MTMOD	DTFMT
PRMOD	DTFPR
PTMOD	DTFPT
SDMODxx	DTFSD[9]

in Table 2. They are explained in detail in the following discussion of the access methods. Key-word designations of macroinstructions are given in parentheses to allow comparison with relevant literature, but they are not needed for an understanding of this paper.

DTFCD
CDMOD

The DTFCD/CDMOD access method provides the user with the ability to read (or punch) a record from a card reader (or on a card punch). Each GET macroinstruction supplies a card record (up to 80-characters) from the reader. Each PUT causes the punching of up to 80 characters. The user has the option of specifying one or two I/O areas (IOAREA1 and IOAREA2), also called buffer areas,[11] to hold the data transferred to main storage from the device, or vice versa. The use of a second buffer area allows the user to do overlap processing while the following record is read or the previous record is punched. The user can process the data read (or data to be punched) in the buffer area if only one area is specified. If two buffer areas are specified, a register (IOREG) can be specified to point to the start of the data field (left-most position) in the current buffer area, or the data can be placed into a work area (WORKA) for processing (or punching). In the latter case, DTFCD/CDMOD transfers the record from the buffer area to the work area and vice versa. The capability for reading a card record and punching information into the same card (CMBND) is also provided if the user has an IBM 1442, an

Table 2 Imperative macroinstructions

Macro-instruction	DTFCD	DTFCN	DTFDA	DTFIS	DTFMT	DTFPR	DTFPT	DTFSD	*Macroinstruction function*
OPEN	*		*	*	*	*	*	*	Does label checking and prepares DTF table for processing
CLOSE	*		*	*	*	*	*	*	Creates trailer labels for tape, updates DASD labels, and terminates processing
GET	*	*		*	*		*	*	Obtains a record in consecutive or sequential order from an input data file
PUT	*	*		*	*	*		*	Places a record in consecutive or sequential order in an output file, or returns a record previously obtained by a GET macroinstruction
READ			*	*	*			*	Reads a record or a block of records[10]
WRITE			*	*	*			*	Writes a record (or a block) or writes back a record (block) previously read[10]
WAITF			*	*					Waits for a completion of a READ or WRITE and verifies proper operation
CHECK					*			*	Waits for a completion of a READ or WRITE and verifies proper operation
RELSE					*			*	Forces the end of an input block
TRUNC					*			*	Forces the end of an output block
CNTRL	*		*		*	*		*	Controls card device, printer, tape, DASD (stacker, select, rewind, seek, etc.)
FEOV					*				Forces an end of volume
PRTOV						*			Tests printer for carriage overflow
SETFL				*					Initializes file creation or load-mode operation
ENDFL				*					Terminates file creation or load-mode operation
SETL				*					Sets lower limit for sequential retrieval
ESETL				*					Terminates sequential retrieval
NOTE					*			*	Notes where a block is read or written[10]
POINT					*			*	Points to a designated block[10]

IBM 2520, or an IBM 2540 with the punch-feed-read feature. Since many data processing installations like to use the stacker selection capabilities, three methods have been provided to achieve this: (1) the user may issue a CNTRL macroinstruction after a GET or before a PUT to select the desired stacker, (2) first character control (CTLCHR) may be used, where the first character of a record may be an ASA (American Standard Association) or SYSTEM/360 control character for stacker selection, and (3) the selection for a given data set may be specified (SSELECT) when the DTF table is generated.

DTFCN

The DTFCN access method allows reading and writing of a record from or to an IBM 1052 printer keyboard by issuing a GET or PUT macroinstruction. The record may be processed in the buffer area or in a work area (WORKA). Only one buffer area can be used with this access method. This file does not require the use of OPEN and CLOSE routines.

DTFDA
DAMOD

Great flexibility in reading or writing a record from or to a direct-access device is achieved by the DTFDA/DAMOD access method. With the WRITE (AFTER) macroinstruction, the user may create a data set in any manner desired. For example, a part number or control field can be converted to a DASD address using a randomizing algorithm, and the record can be written at that address;[12] the user can create a sequential data set with keys (control fields) to be processed later in a skip-sequential fashion.[13] Record retrieval is accomplished by issuing a READ macroinstruction. Two types of READ macroinstructions may be issued for record retrieval. The user may read the record by simply supplying the track and record location (ID), or the track location may be supplied along with the record key (KEY) to be used for record retrieval. Record updating is performed by two corresponding WRITE macroinstructions (ID and KEY). They work in the same fashion as the READ macroinstructions.

In addition to the READ/WRITE macroinstructions, there is a WAITF macroinstruction. This macroinstruction is issued after each READ/WRITE. The WAITF macroinstruction serves a two-fold purpose: (1) it allows overlap processing or issuing of another I/O device macroinstruction (within the limitations of the system configuration) while the present I/O operation is taking place, and (2) it communicates termination of the READ/WRITE operation with which it is associated to the user and returns any status information (ERRBYTE) indicating whether the I/O operation was successful.

Other options, greatly enhancing the capabilities of the access method, are the ability to utilize the multiple track search facility of a DASD (SRCHM) as well as the ability to return the record location (the first five bytes of the record identification, namely CCHHR) when retrieving a record by its key (IDLOC). These features facilitate skip-sequential processing. For example, by issuing a READ (KEY) macroinstruction and starting at the beginning of the data set, the direct-access device searches multiple

tracks until the record is found.[14] The record location is returned and can be used as a reference point in starting the search for the next record to be retrieved. In this manner of processing, only the required records are read, whereas in the normal sequential mode of processing, all records are read even if not all of them are processed. A third feature allows the user to seek ahead by issuing a CNTRL macroinstruction (or restore a strip to an IBM 2321 data cell) and then continue normal processing or issue other I/O device macroinstructions.

When creating a data set using the WRITE (AFTER) macroinstruction, the access method automatically maintains the amount of space available on each track on which the records are stored along with the address of the last record on each track, provided the DASD has been properly initialized (set up) by using either the WRITE (RZERO) macroinstruction or the IBM-supplied utilities for initializing DASD's. This access method does not provide blocking and deblocking facilities. They must be performed by the user.

DTFIS
ISMOD

The DTFIS/ISMOD facility is not only an access method but also a DASD file organization technique for SYSTEM/360 users. Facilities are provided for the user to create a data set, add new records in any order to a previously created data set, and retrieve all records in the data set either randomly or sequentially. The data set is created in sequential order from the input that has been previously sorted on the record keys.

As the data set is created, an index hierarchy is developed. The lowest level is called the *track index* and occupies the first track (in the case of an IBM 2321, one or possibly more tracks) of each cylinder that is contained in the data set area called the prime data area. This index includes a pair of index entries for each track of the cylinder containing the user's data records. The first entry of each pair indexes the highest record key on the appropriate track being referenced. The second entry is used to locate overflow records[15] which can occur from that track when new records (additions) are added to the data set. The second level of index generated is called a *cylinder index*. It is built in a DASD area separate from the prime data area. An entry is made in the cylinder index for the highest record key of each cylinder in the prime data area, and each entry points to the track index on the appropriate cylinder. A third, optional level of index, the *master index*, is generated in the same area as the cylinder index and precedes it. The master index has an entry for the highest key on each track within the cylinder index area. For a small file, this level of index is generally not needed because searching a cylinder index of two or three tracks is as fast as searching the master index and then the cylinder index. Because the data set is sequential with a hierarchy of indices, it is called an indexed sequential data set. The indices provide direct reference to records, allowing their random retrieval with a minimum of search time. The sequential order of the data records, coupled

with the ability to reference overflow records in sequence via the track index, provides sequential retrieval capability. The indexed sequential method consists of four basic functions which provide capabilities for creating a data set, adding new records to it, and retrieving the records. A description of each function follows.

load function

A data set and its associated indices are created by the Load function. Three macroinstructions are used to create the data set. The first one to be issued is a SETFL macroinstruction (set file load mode) which does the initializing needed for data set creation. The next macroinstruction to be issued, a WRITE (NEWKEY), takes the key and data record placed by the user in a work area (WORKL) and creates an output block in the buffer area (IOAREAL) which is written in the prime data area. It also makes the appropriate index entries. After the user has presented all the data records needed to create the data set, the load made must be terminated by issuing an ENDFL macroinstruction (end file load mode). Facility is also provided to extend the data set by adding new records higher in collating sequence than the current last (high) record in the data set. Using the same user program, this can be done by indicating in the delayed Job Control statement (required at execution time of the user's program) that a data-set extension is to take place.

add function

The add function provides the ability to insert new records in the data set. The first of two macroinstructions to be issued for such additions is a WRITE (NEWKEY), which initiates the addition process (searching the indices, etc.) and returns control to the user to allow overlap processing. To complete the addition operation, a WAITF macroinstruction must be issued. This not only ensures that all necessary I/O operations have been completed but also returns status conditions indicating the occurrence of any abnormal operation. When additions are made to the file, the user presents the key and data record to be added in a work area (WORKL). A search is made through the index structure to determine where the record is to be inserted in the data set. The record is either inserted in key sequence in the prime data area, or placed in the overflow area by use of a chaining technique which maintains the proper sequence of the data set.

Two overflow area options, which may be used in any combination, have been provided. One option allows the user to specify that one or more tracks be reserved at the end of each cylinder to store overflow records (CYLOFL). The second option allows an independent overflow area separate from the prime data area to be reserved for storing overflow records. The first option has the advantage of reducing access time for the retrieval of overflow records associated with a given cylinder. The second option has the advantage of utilizing DASD space more efficiently.

random retrieval function

The random retrieval function is used for random retrieval and updating of records. The first of three macroinstructions to be issued for this purpose is a READ (KEY). The READ macro-

instruction performs a search of the indices, using the key of the requested record provided by the user (KEYARG) as the search argument. While the I/O operations that perform this function are taking place, control is returned to the user to allow overlap processing. To complete the READ function and receive the record, the user must issue a WAITF macroinstruction. If the operation is completed successfully, this macroinstruction either places the record into a work area (WORKR) or points to the starting location (leftmost position) of the record within the buffer area by using a general register (IOREG). If the operation was not successful, indications of the resulting abnormal conditions are given. If the user wishes to update and return the record to the data set, he must issue a WRITE (KEY) macroinstruction. The WRITE follows the READ of the record to be updated and precedes the READ for the next record. Again, processing can overlap execution of the WRITE instruction. To complete the operation, the user must issue a WAITF macroinstruction.

sequential retrieval function

The sequential retrieval function makes it possible to sequentially retrieve and update records. The first of four macroinstructions to be issued for this purpose is the SETL (set lower limit), which locates the starting point where retrieval begins. This macroinstruction provides four methods of starting retrieval:

- From the beginning of the data set (BOF)
- At any record location in the prime data area (ID)
- With any record in the data set by supplying the key of the desired starting record (KEY)
- With the first record of a group of records in the same class by supplying the generic key for that class; a class being any group of records that contains identical control information in the first few high-order bytes of the record keys (GKEY, a key equal to or lower than the first record of the desired group)

After the SETL has been successfully executed, the user can issue GET macroinstructions to retrieve each record in the data set in key sequence. The record can be placed in a work area (WORKS), or a general register can be used to point to the starting location of the record in the buffer area. If the user wishes to update the record and return it to the data set, he must issue a PUT macroinstruction. This PUT must follow the GET of the record to be updated and precede the GET of the next record. After all of the desired records have been processed, the sequential retrieval function is terminated by issuing an ESETL macroinstruction. This process of issuing the SETL, GET, PUT, and ESETL macroinstructions can be repeated as many times as desired. By combining the macroinstructions of the sequential retrieval and load functions, the data set can be reorganized. The user can retrieve the current file in its proper sequence (both prime data area records and the associated overflow records) and recreate the file in a new prime data area.

DTFMT
MTMOD

The DTFMT/MTMOD access method provides the ability to create or retrieve magnetic tape records in sequential order. The data set can be created by indicating with the DTFMT parameters that it is an output data set (TYPEFLE) and then issuing PUT macroinstructions. If the records are to be retrieved, an input data set (TYPEFLE) is indicated and a GET macroinstruction is issued. In either case, the records may be processed in a work area (WORKA) or in the buffer area by using a general register (IOREG). When an input data set is to be processed, the access method can also support the read-backward feature for SYSTEM/360 magnetic tape units.

Other optional features which greatly enhance this access method are:

- Specifying two buffer areas for overlap processing capabilities (IOAREA1 and IOAREA2)
- Alternate tape switching between two tape drives
- Bypassing of checkpoint records on input data sets (CKPTREC)

Other macroinstructions allow the user to rewind, rewind and unload, execute various other magnetic tape device functions (CNTRL), or release (skip) the remaining records in a block (RELSE). This can be useful if records are grouped by specific categories. An inverse function of RELSE is the truncating or writing of short blocks of records (TRUNC) for an output data set.

In addition to GET/PUT functions, DTFMT/MTMOD provides the highly useful feature of issuing READ/WRITE macroinstructions to create a data set or retrieve records from a data set generally called a work file (WORK). Overlap processing can take place while the I/O operation is being performed. The user terminates the READ/WRITE by issuing a CHECK macroinstruction that ensures completion of the operation. The particular features which enhance this facility are the NOTE and POINT macroinstructions. By issuing a NOTE macroinstruction, the location of the data block in the data set can then be obtained (three bytes). POINT macroinstructions provide the ability to reposition to a given block in the data set. POINTR can position the tape to the block indicated, POINTW can position the tape after the block indicated, and POINTS can position the tape to the beginning of the data set.

DTFPR
PRMOD

The DTFPR/PRMOD access method is used to print a record on an IBM printer by issuing a PUT macroinstruction. The record to be printed can be presented to the access method via a work area (WORKA) or can be placed into the buffer area. Two buffer areas (IOAREA1 and IOAREA2) can be specified to allow overlap processing. In this case, a work area or a general register must be used to indicate the proper buffer area. Three types of printer-form control are provided by the access method:

- CNTRL macroinstruction for line spacing or page skipping
- PRTOV (printer overflow) macroinstruction for page skipping

or exiting to a user-supplied routine (indicated in the macro-instruction) which can perform certain end-of-page and/or start-of-page functions

- First character control (CTLCHR) which can be used where the first character of a record may be an ASA or SYSTEM/360 control character for line spacing or page skipping

DTFPT
PTMOD

The DTFPT/PTMOD access method provides the ability to retrieve a data record from an IBM 2671 paper tape reader. Two buffer areas (IOAREA1 and IOAREA2) can be used for overlap processing. In this case, a general register must be specified to point to the record in the buffer area currently being used. This access method also provides the ability to handle (1) shifted code for figure shift (FTRANS and SCAN) and/or letter shift (LTRANS and SCAN) or (2) non-shifted code to be translated into SYSTEM/360 code (TRANS). The user must supply the various translation tables needed.

DTFSD
SDMOD

Records can be created or retrieved and updated from a direct-access device by use of the DTFSD/SDMOD access method. The data set can be created by specifying an OUTPUT data set (TYPEFLE) and issuing PUT macroinstructions. If the records are to be retrieved, an INPUT data set is indicated (TYPEFLE) and GET macroinstructions are issued. It is also possible to update those records in the same location on DASD that were retrieved by a GET macroinstruction. In this case, a PUT macroinstruction must be issued for the data set after the GET for the record to be updated and preceding the GET for the next record. This access method also provides double buffering (IOAREA1 and IOAREA2) ability for overlap processing. The user can either process the record in a work area (WORKA) or use a general register to point to the record in the current buffer area. Another macroinstruction allows the user to skip the remaining records in a block (RELSE). This can be useful if records are grouped by specific categories. An inverse function of RELSE, the TRUNC macroinstruction, allows to truncate or write short blocks for an output data set. The CNTRL macroinstruction may also be used to seek the track address of the next record to be processed. In the case of a data cell, CNTRL can restore a strip if the user knows that processing on it has been completed.

DTFSD/SDMOD also provides the READ/WRITE, CHECK, NOTE, and POINT macroinstructions described earlier, although an IBM 2311 disk drive is used as an I/O device in this case. NOTE/POINT uses cylinder, track, and record identification (three bytes) for noting and locating blocks in the data set. Also, if a NOTE follows a WRITE, the unused space on a track can be returned when the data set is being created or when records are to be written in the count, key, and data format of DASD. If a POINTR or POINTW is issued before a READ or WRITE (UPDATE), the READ or WRITE macroinstruction processes the block indicated. If a format write (count, key, and data) is issued, the macro-

instruction WRITE (SQ) writes a new record after the block indicated.

Another feature available with this access method, and which can be used in either of the above two processing modes, is the split cylinder mode. This mode allows two or more data sets to share the same cylinder. Each data set occupies the same track positions through the range of assigned cylinders. This technique has the advantage of minimizing access-arm movement when cross referencing among two or more data sets that perform similar functions. The use of this facility is indicated in one of the job control statements (XTENT) needed at execution time of the user's program.

The main characteristics of the described access methods are summarized in Table 3.

open and close

The purpose of the OPEN and CLOSE routines is to perform the necessary initialization and termination for the access methods. Before issuing any READ/WRITE, GET/PUT, and similar macroinstructions, an OPEN macroinstruction must be issued, indicating the various DTF tables of the data sets to be used. The OPEN routines are called into a special area in main storage (transient area of the Supervisor) where they initialize the various DTF tables, check labels, and verify labeled I/O media. After opening the data sets, normal processing functions may be started (DTFCN does not require an OPEN macroinstruction). After the data set has been processed, the processing function is terminated by a CLOSE macroinstruction, which indicates that the DTF table of the data set is to be closed. The CLOSE routines are called into the transient area where they perform various termination tasks, such as writing trailer labels for magnetic tape output files, and updating DASD labels in certain cases.

physical I/O control system

For the actual execution of the I/O operations necessary for a given function, the logical data management routines (LIOCS) depend on the physical routines (PIOCS) built into the Supervisor of the control system. LIOCS communicates with PIOCS by issuing CCB (Command Control Block), EXCP (Execute Channel Program), and WAIT macroinstructions. The CCB defines a main-storage area used for communicating the appropriate information needed by LIOCS and PIOCS. The logical data management routines (LIOCS) place into the CCB the channel and device address on which the I/O operation is to be performed and the location of a channel program[20] used for the I/O operation. The CCB and the channel program are both located in the DTF table. After the CCB has been properly initialized, the logical data management routines issue an EXCP macroinstruction which causes a transfer of program control to PIOCS. An EXCP parameter points to the CCB so that the PIOCS can obtain the necessary information. PIOCS then either initiates the I/O operation (SIO) or—if the channel and/or device on which the operation is to take place is busy—stores the request in a queue to be executed as soon as channel and/or device are available.

Table 3 Access method characteristics

Characteristic	DTFCD/CDMOD	DTFCN	DTFDA/DAMOD	DTFIS/ISMOD	DTFMT/MTMOD	DTFPR/PRMOD	DTFPT/PTMOD	DTFSD/SDMOD
Data set organization	Sequential	Sequential	Direct	Indexed sequential	Sequential	Sequential	Sequential	Sequential
Basic element of data set	Record (up to 80 characters)	Record	Record	Record	Record	Record	Record	Record
Basic element of access method	Record	Record	Block	GET/PUT record READ/WRITE record	GET/PUT record READ/WRITE block	Record	Record	GET/PUT record READ/WRITE block
Primary input and output macro-instructions	GET/PUT	GET/PUT	READ ID READ KEY WRITE ID WRITE KEY WRITE AFTER WRITE RZERO	SETFL, ENDFL WRITE NEWKEY READ KEY WRITE KEY SETL, GET/PUT ESETL	GET/PUT READ/WRITE NOTE POINTS POINTR POINTW	PUT	GET	GET/PUT READ/WRITE NOTE POINTR POINTW POINTS
Processing overlap ability	Yes (two buffer areas)	NO	Yes (WAITF)	Yes (WAITF)[18]	Yes, GET/PUT (two buffer areas) READ/WRITE CHECK	Yes (two buffer areas)	Yes (two buffer areas)	Yes, GET/PUT (two buffer areas) READ/WRITE CHECK
Record/block format[16]	F, V, U record[17]	F, U record[17]	F, U block	GET/PUT F record READ/WRITE F block[19]	GET/PUT F, V, U record READ/WRITE F, U block	F, V, U record[18]	F, U record[18]	GET/PUT F, V, U record READ/WRITE F, U block
Provisions for data-set search			Track address with record location or record key	Master (optional), cylinder, track indices	READ/WRITE block location in data set with NOTE/POINT			READ/WRITE cylinder track, record (3 bytes) location with NOTE/POINT

* F, V, and U denote fixed, variable, and unspecified lengths.

In either situation, a busy bit (traffic bit) is turned on in the CCB and control is returned to the routine that issued the EXCP. At this point, the logical data management routines providing CHECK or WAITF capabilities usually return to the routine that issued a READ or WRITE macroinstruction. When the data block requested by the I/O operation is desired, a WAIT macroinstruction is either issued automatically by LIOCS at the appropriate time in GET/PUT processing or activated by LIOCS via the WAITF macroinstruction in READ/WRITE processing. The WAIT tests the traffic bit in the CCB to determine whether or not the I/O operation has been completed. The traffic bit is turned off by PIOCS when an interrupt condition is received, indicating that the I/O operation for the channel and device is completed; this is an automatic procedure in SYSTEM/360. If the I/O operation has not been completed, a wait-state condition occurs until completion. When the operation is completed, normal processing by the access method is resumed.

If the user prefers to use the three PIOCS macroinstructions discussed, he must supply his own channel programs, blocking and deblocking routines, and data management techniques. DOS/TOS data management provides a DTFPH (Define the File for Physical IOCS) macroinstruction for the user who only uses PIOCS. This macroinstruction generates a DTF table that can be used by the OPEN/CLOSE routines for creating and checking labels on magnetic tape and direct-access devices. The method affords the same protection and identification capabilities to users of PIOCS as to those using logical DOS/TOS data management routines (LIOCS).

summary comment

With the users of intermediate SYSTEM/360 computers in mind, one of the main considerations in designing and implementing the data management facilities for DOS/360 and TOS/360 was the need for simplicity in usage of the system. The data management provides easy processing of data sets as well as a wide range of capabilities at low main-storage requirements. The tabular-modular design, using read-only coding concepts, provides flexibility not only in operation but also in growth potential.

ACKNOWLEDGMENT

The author wishes to acknowledge the efforts of all those who participated in developing the DOS/TOS Data Management facilities. Their work provided the basis for this paper.

FOOTNOTES

1. The discussion in this paper refers to the first version of DOS/360 and TOS/360, the disk and tape operating systems for intermediate SYSTEM/360 configurations. The second version incorporates multiprogramming and telecommunication capabilities; these additional features do not affect the basic structure and concepts discussed in this paper.

2. A data file, or data set, is a collection or set of records, such as a payroll file, inventory file, etc.
3. Information is usually recorded on such storage devices as tape, direct-access devices, etc., in the form of a block of data. This block may contain one or more records. The number of records contained in a block is considered as the blocking factor. If only one record is contained in a block, this record is considered "unblocked." If a block contains more than one record, we talk about "blocked" records. DOS/TOS permits recording of data blocks in three categories: *fixed-length blocks* if all records are of the same length, *variable-length blocks* if each record in the block can be of any length (in this case, each record must be preceded by a record-length indicator, and the block itself by a block-length indicator), or an *undefined record format* if the records do not fall into the first two categories; in this case, the user must remove and return the record(s) in the block (deblocking and blocking).
4. The method of implementing routines that are not self-modifying (or self-destructive) is referred to as read-only coding.
5. For tape files there is a third label, called a trailer label, which follows the last record block of the data set. Although a standard header (trailer) label procedure is provided, additional user labels can be employed for a given data set. In addition, the user is permitted to employ his own labeling conventions instead of using the standard procedures provided.
6. DASD access methods are not supported in TOS/360.
7. A more detailed description of the module generation and of the linkage between modules and tables is given by D. H. Ricour and V. Mei in "Internal data management techniques for DOS/360," *IBM Systems Journal* **6**, No. 1, 38–48 (1967).
8. The DTFCN table does not require an associated module.
9. The SDMODxx consists of ten module types, xx indicating the module type to be generated: FI, FO, FU, VI, VO, VU, UI, UO, UU, or W.
10. DTFMT and DTFSD provide READ/WRITE, CHECK, NOTE, and POINT macroinstructions which give added flexibility not available from GET/PUT macroinstructions. POINT actually consists of the three imperative macroinstructions POINTR, POINTW, and POINTS.
11. An I/O or buffer area is an area set aside in main storage to receive or transmit a block of information from an I/O device.
12. Historically, random file organization techniques have been utilized where the user took a part number, a control field, etc. and converted this number to a direct-access device address, such as a record location on a track. In SYSTEM/360, DASD's enhance randomizing to a record address because the number of synonyms or overflow records can be reduced. (A synonym occurs when more than one record randomizes to the same record address and cannot be stored in that location.) Records may be randomized to a track address instead of a record location, and written on the track using the WRITE (AFTER) macroinstruction. For further discussion, refer to W. Buchholz, "File organization and addressing," and W. P. Heising, "Note on random addressing techniques," *IBM Systems Journal* **2**, 86–116 (June 1963).
13. In skip-sequential processing, a data set that has been sorted on a key control information field is created in a sequential manner. Records are retrieved by scanning or searching the data set using the key of each record desired. In this way, only these records are retrieved and the job throughput is improved when a large percentage of the data set is to be processed.
14. Due to DASD limitations, the multiple track search can be done only within a cylinder. If a record is not located before the end of a cylinder is reached, an end-of-cylinder condition is returned to the user (ERRBYTE). It is the user's responsibility to reissue the READ macroinstruction. The returned address (IDLOC) points to the first record location on the next cylinder.

15. When new records are added to the data set, they are normally inserted in sequence on the track to which they belong. As a result, the last record on the track becomes displaced and is stored in an area set aside for this purpose. The record itself is called an overflow record, and the area where it is stored is called an overflow area.
16. F, V, and U denote fixed, variable, and unspecified lengths.
17. These access methods permit only the unblocked record format (one record per block).
18. The WAITF can only be used with the WRITE of the ADD function or the READ/WRITE of the random-retrieval function.
19. The access method does record deblocking (READ) and blocking (WRITE).
20. A channel program consists of a set of channel command words (CCW's) that indicate the type of I/O operation to be executed, such as read, write, seek, etc.