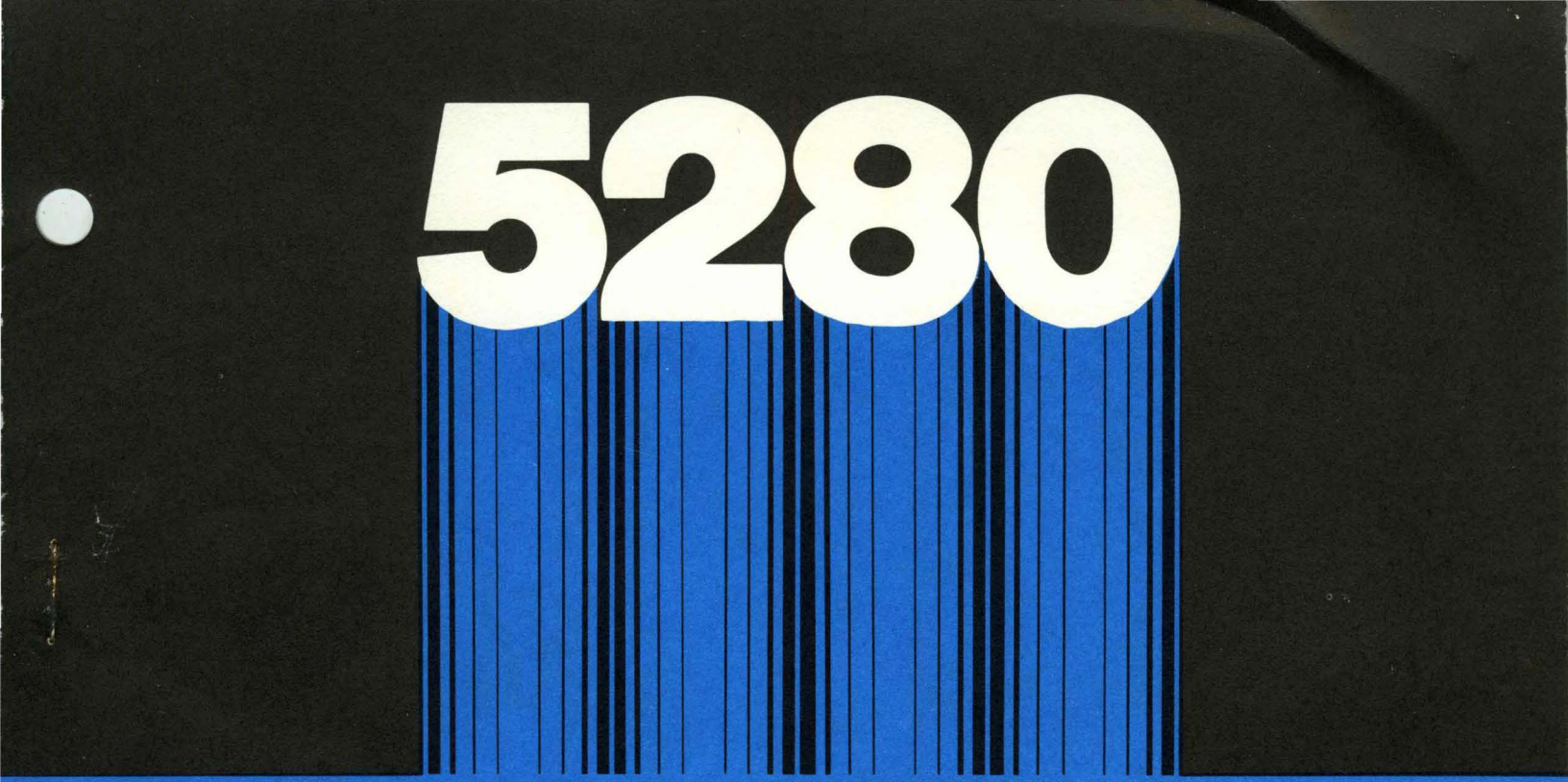


GA21-9352-1
5280-20

IBM 5280 Distributed Data System

System Concepts

GA21-9352-1
5280-20

IBM 5280 Distributed Data System

System Concepts

Second Edition (June 1980)

This is a major revision of, and obsoletes, GA21-9352-0.

Because the changes and additions are extensive, this publication should be reviewed in its entirety.

Changes are periodically made to the information herein; these changes will be reported in technical newsletters or in new editions of this publication.

Use this publication only for the purposes stated in the *Preface*.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address below. Requests for copies of IBM publications and for technical information about the system should be made to your IBM representative or to the branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Product Information Development, Department 997, 11400 Burnet Road, Austin, Texas 78758. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

This manual is intended for operators, supervisors, and programmers using the IBM 5280 Distributed Data System. Persons using this manual should be familiar with the use of DE/RPG programs or system utilities on the 5280.

This manual provides conceptual information about certain parts of the system, such as diskette storage, and how they relate to the DE/RPG language or to the utilities. Although this manual is not intended for the assembler programmer, information such as data exchange between the 5280 and another system might be useful to such a programmer.

Unless otherwise noted, the utilities referred to in this manual are discussed in greater detail in the *IBM 5280 Utilities Reference/Operation Manual*, SC21-7788.

Related Publications

- *IBM 5280 General Information*, GA21-9350
- *IBM 5280 Planning and Site Preparation Guide*, GA21-9351
- *IBM 5280 Functions Reference Manual*, GA21-9353
- *IBM 5280 Message Manual*, GA21-9354
- *IBM 5280 Master Index*, GA21-9356
- *IBM 5280 Operator's Guide*, GA21-9364
- *IBM 5280 User's Setup Procedures*, GA21-9365
- *IBM 5280 Machine Verification Manual*, GA21-9357
- *IBM 5256 Printer Operator's Guide*, GA21-9260-1
- *IBM 5280 System Control Programming Reference/Operation Manual*, GC21-7824
- *IBM 5280 DE/RPG Reference Manual*, SC21-7787
- *IBM 5280 Utilities Reference/Operation Manual*, SC21-7788
- *IBM 5280 Sort/Merge Reference/Operation Manual*, SC21-7789
- *IBM 5280 Assembler Language Reference Manual*, SC21-7790
- *IBM 5280 Introduction to DE/RPG*, SC21-7803
- *IBM 5280 DE/RPG User's Guide*, SC21-7804
- *IBM 5280 DE/RPG Problem Determination Procedures for the Programmer*, SC21-7852
- *IBM 5280 Communications Utilities Reference Manual*, SC34-0247

Contents

Chapter 1. Introduction

The IBM 5280 Distributed Data System is a multipurpose diskette-based system that can be used in a variety of data processing applications, such as data entry, remote job entry, and data processing.

The 5280 can consist of a programmable data station or programmable control unit, auxiliary work stations, diskette drives, and printers, and can possess communications capability.

The 5280 supports two programming languages: DE/RPG and assembler.

The Utilities Program Product available for the 5280 allows you to do things such as allocate or delete data sets from a diskette, enter data, convert existing 3740 programs to 5280 programs, or display system status.

If your 5280 has communications capabilities, the Communications Program Product allows you to transfer data to and receive data from other systems.

The Sort/Merge Program Product allows you to sort the records in a data set or to merge records from two data sets into one data set.

The 5280 user storage is divided into *partitions*. Each individual program is executed in a partition. The 5280 can execute more than one program at the same time.

ABOUT THIS MANUAL

This manual is intended to help you understand the following aspects of the 5280:

- Diskette concepts, such as addressing, initialization, the maximum number of data sets, and available storage.
- Data set concepts, such as access methods, layout, labels, write-protect, security, and multivolume support.
- Data exchange, which is being able to exchange data with other systems.
- Partition concepts, such as the difference between foreground and background partitions, how to initiate a job in a background partition, and how background partitions can access a keyboard.
- Resource allocation table concepts, such as logical and physical device addresses, what the common area is, and a description of the functions in the common area.
- Performance characteristics, such as microprocessor activities and diskette performance, and techniques for improving performance.
- Printer character sets, which include all the character sets available on the 5280.

This manual does not have to be read chapter by chapter. Simply read the chapters as they become appropriate. For example, you might want to read Chapter 3, *Data Set Concepts*, when you need to know the best way to access a particular data set.

USING THE 5280 PUBLICATIONS

The following chart lists topics that are discussed in this and other 5280 publications, and the publications in which the topics can be found. Preferred references are marked with an asterisk.

Topic	Related Publications
Available Features	**General Information* **Planning and Site Preparation Guide* *Operator's Guide* *Functions Reference Manual*
Available programs	**General Information* *Planning and Site Preparation Guide* *Operator's Guide* *Sort/Merge Reference/Operation Manual* *System Control Programming Reference/Operation Manual* *Utilities Reference/Operation Manual*
Common area	*General Information* **System Concepts* **System Control Programming Reference/Operation Manual* *Functions Reference Manual*
Communications	*General Information* **Communications Utilities Reference Manual* *DE/RPG User's Guide* *DE/RPG Reference Manual*
Data entry	*General Information* *Operator's Guide* *Introduction to DE/RPG* **DE/RPG User's Guide* **Utilities Reference/Operation Manual*
Data set organization	*Planning and Site Preparation Guide* **System Concepts* *DE/RPG User's Guide* *DE/RPG Reference Manual*
Data stations	**General Information* *Planning and Site Preparation Guide* *Operator's Guide*
Diskette access methods	**System Concepts* *DE/RPG User's Guide* *DE/RPG Reference Manual*
Diskette care, handling, and wear	*Operator's Guide*

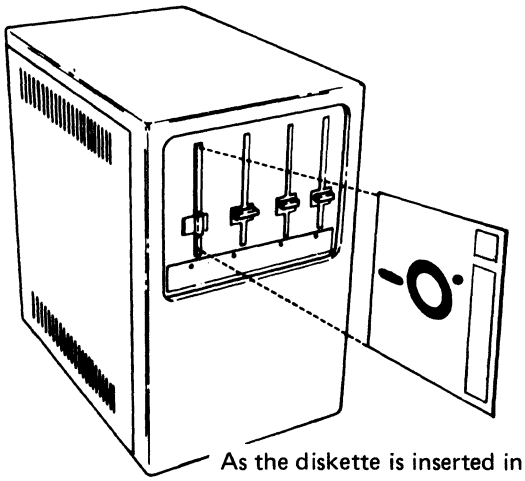
Topic	Related Publications
Diskette formats	*General Information* **System Concepts* *Utilities Reference/Operation Manual*
Error messages	*Message Manual*
Establish job procedures	**Planning and Site Preparation Guide* *Operator's Guide* *Introduction to DE/RPG*
Initial Program Load (IPL)	*Planning and Site Preparation Guide* *Operator's Guide* **System Control Programming Reference/Operation Manual*
Inquiry into a data base	*General Information* *DE/RPG User's Guide* *DE/RPG Reference Manual* **Communications Utilities Reference Manual*
I/O device support	**General Information* *DE/RPG User's Guide* *DE/RPG Reference Manual*
Partitions	*General Information* **Planning and Site Preparation Guide* *System Concepts* *System Control Programming Reference/Operation Manual*
Performance	*System Concepts*
Printer	*General Information* *Planning and Site Preparation Guide* *Functions Reference Manual* **IBM 5256 Printer Operator's Guide*
Problem determination	*Operator's Guide* *Machine Verification Manual* **Problem Determination Procedures for the Programmer* **Message Manual*
Sample configuration	*General Information* **Planning and Site Preparation Guide* *System Control Programming Reference/Operation Manual*
Status line	*Communications Utilities Reference Manual* **Operator's Guide* *Message Manual*
Storage requirements	*General Information* **Planning and Site Preparation Guide* *Functions Reference Manual*

Topic	Related Publications
System architecture	*Functions Reference Manual*
System configuration	*Planning and Site Preparation Guide* **System Control Programming Reference/Operation Manual*
System diskette	**System Concepts* *System Control Programming Reference/Operation Manual*
System facilities	**General Information* *Planning and Site Preparation Guide* **Functions Reference Manual*
System layout	**Planning and Site Preparation Guide* *User's Setup Procedures*
System status	*Operator's Guide* **Utilities Reference/Operation Manual*
System summary	*General Information* **Planning and Site Preparation Guide*
Training data entry operators	*Planning and Site Preparation Guide* **Introduction to DE/RPG* *Utilities Reference/Operation Manual*
Utilities	*General Information* *Planning and Site Preparation Guide* *Operator's Guide* *System Concepts* **Sort/Merge Reference/Operation Manual* **Utilities Reference/Operation Manual*
Writing data entry formats	**Introduction to DE/RPG* *DE/RPG User's Guide* *DE/RPG Reference Manual* **Utilities Reference/Operation Manual*

Note: To find specific references to specific topics, see the *Master Index*. A brief description of each publication is given in the *Planning and Site Preparation Guide*.

Chapter 2. Diskette Concepts

The IBM diskette is a thin, flexible disk permanently enclosed in a semirigid, protective, plastic jacket. When the diskette is properly inserted in the diskette drive, the disk turns freely within the jacket. The diskette is inserted in the diskette drive as follows:



As the diskette is inserted in the diskette drive, the label must be on the left side. The drive latch must be turned down after the diskette is inserted.

The system writes data on the diskette at specific locations (addresses). Data written at an address remains there until it has been replaced by new data. To read data, the desired address is found and the data is read into the 5280. Data and programs are stored in areas on the diskette that are called *data sets*. Chapter 3 is devoted to data set concepts.

DISKETTE TYPES, FORMATS, AND STORAGE CAPACITIES

The 5280 uses three types of diskettes: the one-sided diskette (diskette 1), with data recorded on just one side; the two-sided diskette (diskette 2), with data recorded on both sides; and the two-sided diskette (diskette 2D), with data recorded on both sides at double density. The diskettes come in various formats consisting of the number of sectors per cylinder and the number of characters (bytes) per sector. The diskette formats are:

	Format Number	Sectors per Cylinder	Bytes per Sector	Available Storage in Bytes (cylinders 1-74)
	1	26	128	246 272
Diskette 1	2	15	256	284 160
	3	8	512	303 104
	4	52	128	492 544
Diskette 2	5	30	256	568 320
	6	16	512	606 208
	7	52	256	985 088
Diskette 2D	8	30	512	1 136 640
	9	16	1 024	1 212 416

Note: The diskette type (1, 2, or 2D) is identified on the diskette label, along with the number of bytes per sector. On the label, the number of bytes per sector is called the record length. Diskette formats are described in greater detail in Chapter 5.

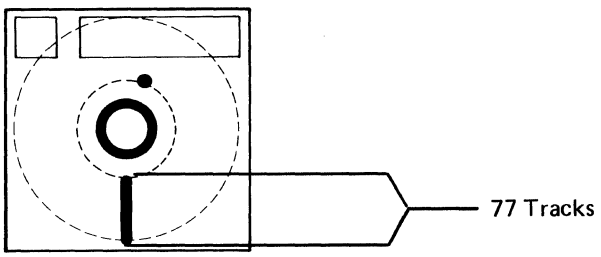
Although the previous chart shows the maximum amount of diskette storage, the amount of diskette storage actually available to you depends on:

- The number of data sets and the size of the data sets allocated on the diskette
- The type of data set structure used (discussed in Chapter 3)
- The allocation of data set space (record placement) as the result of previous delete or allocate options

DISKETTE LAYOUT

Track

A track is the recording area on a single diskette that passes the read/write head while the disk makes a complete revolution. The read/write head is held by a carriage that can be moved to 77 distinct locations along a straight line from the center of the disk. Therefore, each diskette has 77 concentric tracks on which data can be stored. The following illustration shows where the 77 tracks are located on the diskette:



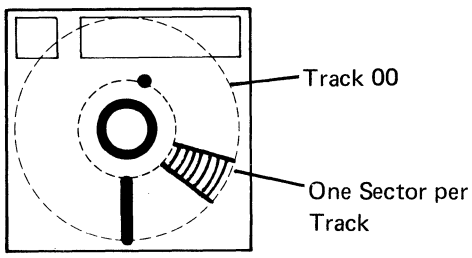
The diskette drive for two-sided diskettes has a read/write head on each side. Each track on side 0 of a two-sided diskette has an associated track on side 1.

Cylinder

A cylinder is one track on a one-sided diskette or a pair of associated tracks (the corresponding tracks on opposite sides of the diskette) on a two-sided diskette. There are 77 cylinders (numbered 0 to 76) on a diskette. Cylinder 0 is called the index cylinder and is reserved for information describing the diskette and its contents. (See *Index Cylinder* later in this chapter.) Cylinders 1 through 74 are used to store data. The last two cylinders are reserved for use as replacements (alternative cylinders) for defective cylinders. (See *Recovering from I/O Errors* in Chapter 3.)

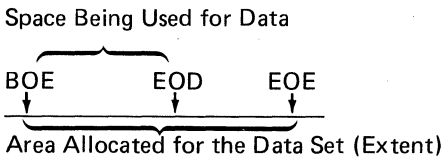
Sector

A sector is a portion of a cylinder, as shown in the following illustration. All sectors on a single track are the same size, and the number of sectors on a cylinder depends on the number of bytes per sector (see *Diskette Types, Formats,* and *Storage Capacities* in this chapter).



Index Cylinder

The index cylinder (cylinder 0) contains information about the diskette, such as volume and owner identification. The index cylinder also contains information associated with each data set on the diskette. This includes the name of each data set and the addresses associated with the data set extents. An extent is the maximum space a data set can occupy. The address at the beginning of this space is called the beginning of extent (BOE). The address at the end of this space is called the end of extent (EOE). A data set might not use all of the space allocated for it by the BOE and EOE addresses; therefore, another address for end of data (EOD) exists.



The EOD address is used to identify the next unused area within the extent or to indicate that data has been written to the EOE address. Although this information is contained in the data set label, the 5280 allows allocation of data sets by specifying the size, format, and number of records required.

Index Cylinder Layout

Sectors 1-7 of the index cylinder (cylinder 0) are reserved for information such as the owner identifier. The rest of the sectors on the cylinder contain pointers to the addresses of the various data sets on the diskette.

Volume ID, Owner ID, and Accessibility

Each initialized diskette has a volume identifier, an owner identifier, and an accessibility field on cylinder 0. The volume ID is a 1 to 6 character name given to the entire diskette (volume). The owner ID is a user-defined 1 to 14 character alphameric name. Both identifiers can be entered with the diskette initialization utility. If no identifiers are specified, the system assigns VOLID and OWNERID.

An accessibility field on the diskette guards against unauthorized access to data on the diskette. If the accessibility field is not left blank, the user must enter the correct owner ID to access the data.

Diskette Addressing

A diskette address consists of a combination of cylinder number, head number, and sector number.

The cylinder number identifies the cylinder onto which a physical record is written or from which it is read.

The head number is the side of the diskette on which the data is to be written or from which it is to be read. This is 0 for all one-sided diskettes and for side 1 of two-sided diskettes. The number is 1 for side 2 of two-sided diskettes.

The sector number is the sector into which the data is to be written or from which the data is to be read.

DISKETTE INITIALIZATION

You can initialize your diskettes by using the diskette initialization utility. Initialization prepares a diskette for use by doing two things. It formats the diskette for use on a system, and it checks for defective sectors. Diskettes must be initialized before they can be used for storing data. All IBM-supplied diskettes are initialized before they are shipped to a customer. Reinitializing is not required unless:

- The diskette was exposed to a strong magnetic field.
- A defect has occurred in one or two cylinders. In this case, initialization can be used to take the bad cylinder(s) out of service and use one or two of the alternative cylinders (cylinders 75 and 76).
- A format other than the existing format is desired.
- The label area of a diskette 2D is to be expanded past 71 labels.

Note: Although IBM-supplied diskettes are initialized, they might contain an empty data set that has already been allocated. Use the diskette label list utility to check for this. If a data set exists, delete it using either the diskette/data set clear or diskette label maintenance utility.

ALLOCATING DATA SET SPACE ON DISKETTE

You can use the diskette label maintenance utility, the key entry utility, or a DE/RPG program to allocate data sets on the diskette. Once you have loaded the program, prompts allow you to enter the data set name and address, then the number of records to be allocated, data exchange type, and record size.

Data sets are not always allocated sequentially on the diskette. For example, data set 2 might be on cylinder 3, data set 3 might be on cylinder 7, and data set 4 on cylinders 8-13. You can control the location of a data set on the diskette only by using a totally unallocated diskette and allocating the data sets in the order you want them.

When the information in a diskette data set is no longer needed, you can use the diskette/data set clear utility or the delete option of the diskette label maintenance utility to delete the data set so that the data set can become available for other uses. However, once a data set is allocated, you cannot increase the size of the data set without copying (reallocating) the data set to a larger data set space.

REALLOCATING DATA SET SPACE

When you delete or reallocate an existing data set, no other data sets are affected. If you reallocate a data set to another data set space on the same diskette, the original space allocated to that data set is unavailable until the diskette is compressed.

For example, assume you want to increase the size of data set 3 from 10 000 to 15 000 records. Using the allocate option of the diskette label maintenance utility, allocate a large enough space for 15 000 records for the data set. The reallocated data set would be located on the diskette following the last allocated data set, provided there is enough continuous storage available. Then use the diskette copy utility to transfer the data set to the larger space. The new data set must have a different name at allocation time in order to copy the data onto the same diskette.

After you have copied the data set to its new space, free up data set 3 using the delete option of the diskette label maintenance utility. The area where the old data set resided is not available for storage. To make the area available, use the diskette compress utility.

Using the diskette compress utility, the entire contents of a diskette can be moved, data set by data set, to the low address end of the diskette. This leaves all unused space at the high address end of the diskette. Allocated data set space is not altered; the sizes of the data sets will not increase or decrease. Only deleted data sets are removed. Compression takes place in both the label area and data area. All of the unallocated space on the diskette is then in one continuous area.

EXTENDING THE AREA FOR DATA SET LABELS

Using the diskette initialization utility, you can extend the area for data set labels on a diskette 2D. In other words, you can increase the number of data sets that can be contained on a diskette. The extension feature should seldom be necessary but is most efficient if you have many small data sets to store on a diskette. The diskette initialization utility displays a prompt allowing you to choose the number of data set labels you want on your diskette 2D, from 71 up to 1007.

A cylinder assigned for additional labels cannot be used for data. Therefore, the amount of data storage is reduced as the storage for data set labels increases. As the storage for labels increases, the time it takes the system to find a specific data set, or determine that it does not exist on the diskette, also increases. Therefore, setting aside additional space for data set labels is not recommended unless absolutely necessary.

NUMBER AND SIZE OF DISKETTE DATA SETS

The diskette volume identification, owner identification, and data set labels are contained on the index cylinder. Each type of diskette can have the following maximum number of data set labels:

	Diskette 1	Diskette 2	Diskette 2D
Maximum Number of Data Set Labels	19	45	71

Note: For a diskette 2D, see the diskette initialization utility for information on how to extend the area for data set labels beyond 71.

If you allocate the maximum number of data set labels without using all the available data set space, the remaining data set space becomes unavailable for storing data. For example, assume you have a diskette 1 with 128 bytes per sector (format 1). This diskette has 246 272 bytes available for storing data; however, you allocate 19 data sets, each containing 50 records. Each record is 128 bytes long.

```
    50 records
x  128 bytes
-----------
 6 400
x   19 data sets
-----------
121 600 bytes of allocated storage
```

Now, by subtracting the allocated storage from the available storage, you can see that 124 672 bytes of storage are not available for you to store data.

```
246 272   bytes available
121 600   bytes used
-----------
124 672   bytes unused
```

You can avoid this by allocating larger data sets, thus using all of the available storage in 19 or fewer data sets.

CREATING A SYSTEM DISKETTE

If you want to copy certain IBM programs and utilities onto another diskette for use as a system diskette, you should be aware that some of these programs and utilities have other data sets that are necessary to run them. Use the diskette copy utility to copy the required programs and their associated data sets. The following table lists those programs and utilities that need additional data sets.

Data Set Name	Descriptive Name	Associated Data Sets
SYSCON	System configuration program	SYSDIPL SYSDPRT1 SYSDPRT2 SYSTBLS SYSSCPO
SYSPTF	PTF/PATCH program	SYSSCPO
SYSDERPG	DE/RPG compiler	SYSCMPO
SYSASM	Assembler	SYSASM1 SYSASM2 SYSASM3 SYSASM4 SYSASM5 SYSASM6 SYSASM6C SYSASM6D SYSASM6I SYSASM6M SYSASM7 SYSASM8 SYSACF
SYSCOPY	Diskette copy utility	SYSUPO
SYSACLC	ACL to assembler language conversion aid	SYSALCO SYSEND

Chapter 3. Data Set Concepts

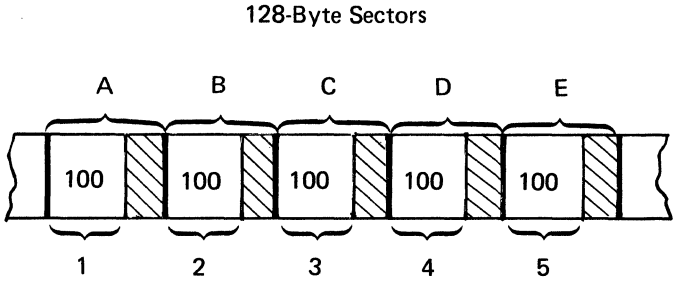
A data set is a group of records that have the same length. This chapter discusses things you need to know about data sets, such as how they are structured, how they are accessed, and what happens when you insert and delete records in a data set.

DATA SET STRUCTURE

A data set can be structured in either of two ways, and the structure you choose determines the type of data exchange you will be using.

Unblocked and Unspanned

Unblocked and unspanned means that records are not blocked together and that the records must each start on sector boundaries. Each record is one block of data. On the 5280, unblocked and unspanned records cannot be longer than sectors.

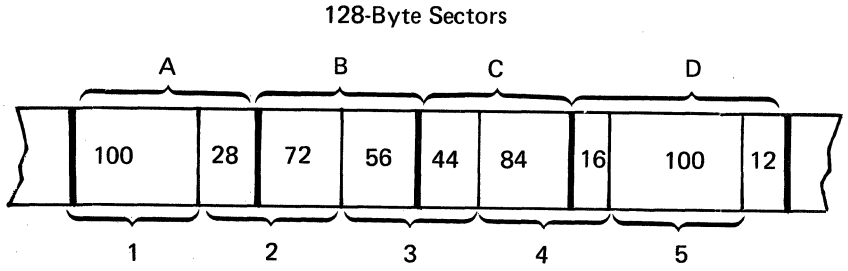


In the preceding example, each 100-byte record starts at the beginning of a 128-byte sector boundary. The remaining 28 bytes in each sector are not available for data storage. Some of the potential storage space is thus wasted.

The basic and H exchange types use the unblocked and unspanned data structure. See Chapter 4 for more information on the basic and H exchange types.

Blocked and Spanned

Blocked and spanned means that records are blocked together, and that sector boundaries are not necessarily related to record positions.



In the preceding example, 100-byte records are placed next to each other with no regard for sector boundaries. No storage space is unused.

The I exchange type uses the blocked and spanned data structure. Unless there is a requirement to exchange data with another system that will accept only unblocked and unspanned data, the blocked and spanned data structure is recommended.

ACCESS METHODS

A data set can be accessed (read and written) in several different ways. The method you choose depends on your particular needs.

A data set consists of records that are grouped together. Although it is not necessarily so, the records are usually similar in content. A data set that stores a mailing list might consist of records that each contain a name field, an address field, and a zip code field.

Sometimes a record can be identified by a field. An employee data set might contain, among other things, an employee number field. The record could be referred to by employee number. The data set might even be in employee number order. In this case, the employee number field would be called the *key field*, and the data set would be in *key sequence*.

Records are stored in the data set in sequential order. Each record has an associated *relative record number*. The first record in the data set is at relative record number 1, the second is at relative record number 2, and so forth.

Sequential Access Method

In the *sequential* access method, records are placed into the data set sequentially. The first record entered occupies the first position in the data set, the second occupies the second position, and so on. Records are retrieved in the same order. The first record is read, then the second, and so on.

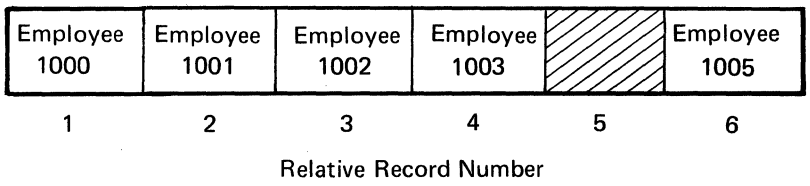
The main advantage of the sequential access method is that records can be processed the fastest when they are processed in sequential order.

Any data set can be accessed sequentially.

Direct Access Method

In the *direct* access method, a relationship exists between records and their positions in the data set. The relative position of a record might be equal to a program counter or a value of a field within a record. The relative position might also be derived by a formula or conversion technique. The relative position is known as the relative record number.

The following is an example of a data set that can be accessed directly:



In this example, a company has a maximum of 1000 employees, and they are assigned employee numbers of 1000 through 1999. Record 1 contains data on employee 1000, record 2 contains data on employee 1001, and so forth. The conversion technique is simply to subtract 999 from the employee number to obtain the relative record number. Notice that record 5 is blank (a non-deleted record). This is because no employee is currently assigned that number.

This access method provides the fastest retrieval of random data. A disadvantage exists when a high percentage of storage spaces are vacant.

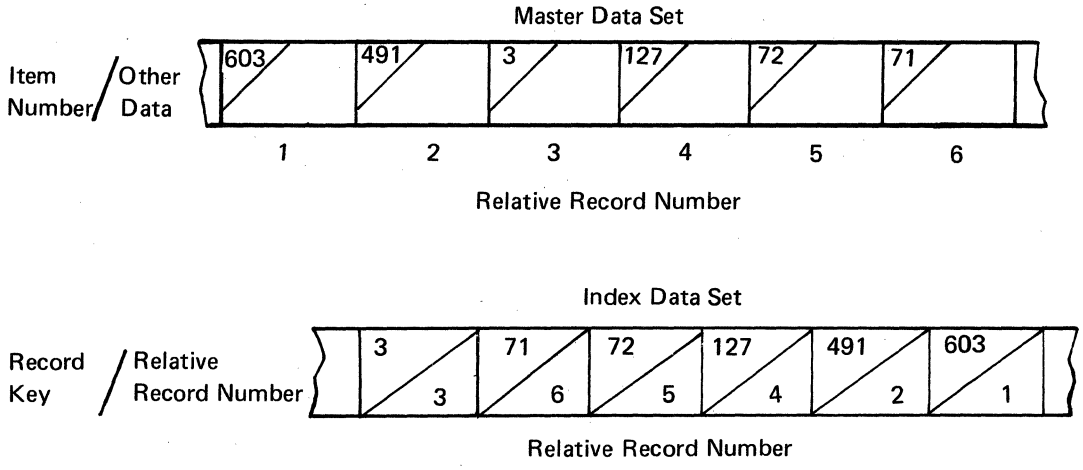
The direct access method can be used to read and write records within any data set. The data set must be created using the sequential access method, however. For example, using the sequential access method you could create a data set with 10 000 blank records. To write records to the data set directly, you would update the existing blank records.

The ADDROUT option of the sort program creates relative record numbers that can be used by DE/RPG for direct access.

Key Indexed Access Method

In the *key indexed* access method, an entry for each record is stored in a separate data set called an index data set. The entry consists of the record's key and the record's location. The index data set allows a program to refer to the records by their record keys. The key for each record must be unique.

The following is an example of a master data set and its index data set:



In this example, a company has an inventory that is identified by item number. The master data set is built sequentially, with item 603 being the first item in the data set, item 491 being the second, and so on. At the same time, an index data set is also being built, which contains the record key (item number) and the relative record position of each record. As the master data set is being built, and as records are being added, the index record is inserted in its proper key sequence in the index data set. Optionally, the index data set can be built using the sort program.

To access a record randomly, the index data set is accessed to find the relative record number of the required record; then the record is directly accessed by relative record number. With DE/RPG, for example, if you want to access item number 127, the system first goes to the index data set and searches it until it comes upon item number 127. The relative record number associated with item number 127 is 4. The system then goes directly to record number 4 of the master data set to access the record.

The main advantage of the key indexed access method is that master data sets can be accessed randomly without knowing the relative record number for any given record, and without maintaining the master data set in any logical order. However, the index data set requires additional storage space.

Direct By Key Access Method

The *direct by key* access method is similar to the key indexed access method except that no second data set exists. The single data set is kept in key sequence by DE/RPG. Each time a record is added, the system inserts the record where it should logically reside in the data set. Records are accessed by key, as in the key indexed access method. Keys must be unique.

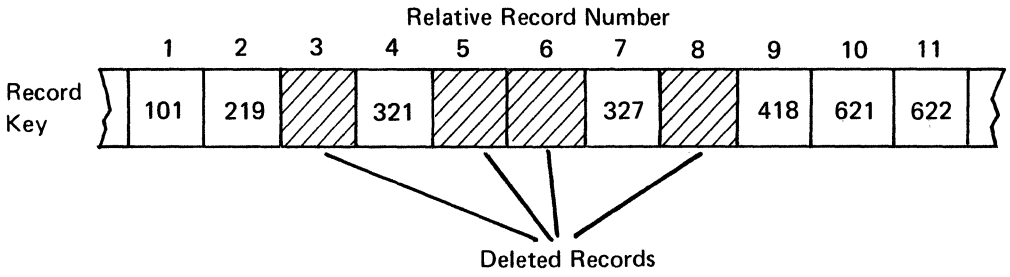
An advantage of the direct by key access method is that the records are kept in logical order without the use of the sort program. A disadvantage is that as the data set grows, the amount of time needed to insert each record might become prohibitive.

Any data set can be accessed by the direct by key access method as long as the key being used is in ascending sequence in the data set.

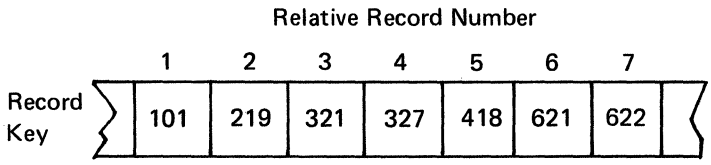
RECORD DELETION

The 5280 can delete records from a data set. When a record is deleted, the system marks the record as deleted. To actually remove the deleted record from the data set, copy the data to a new data set using the diskette copy utility. The default option removes the deleted records.

The following is an example of a data set with several deleted records.



This is the same data set after you run the data set copy utility:



In the direct access method, an attempt to read a deleted record results in an error message. Attempting to read a deleted record in sequential access does not result in an error message; the system skips over deleted records.

Note: It is possible to recover deleted records from I exchange data sets. See *Record Delete Character* under *Data Set Label*.

RECORD INSERTION

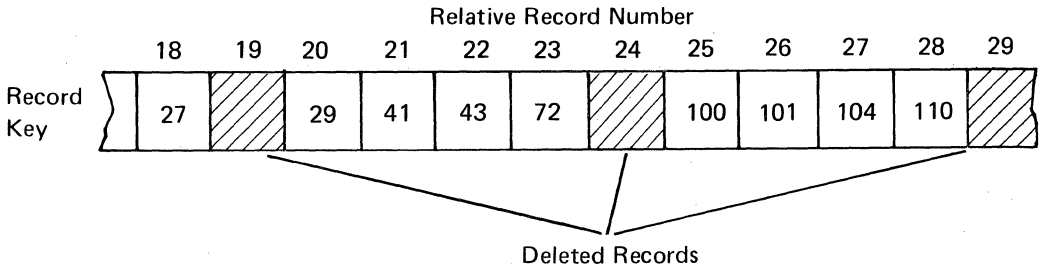
The 5280 can insert one or several records into a data set. When you insert more than one record at the same time, every record from the point of insertion to the end of the data set is moved to make room for the inserted records.

The more records that are beyond the point of insertion, the longer the insertion will take.

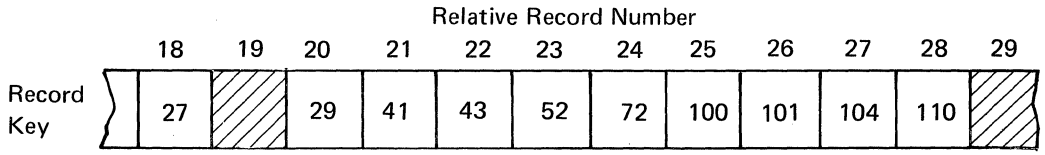
This is a two step process. First the system inserts deleted records. Then you enter the records that are to be inserted, and the system writes those records to the data set, one at a time. For example, you want to insert 10 records at one time. The system inserts 10 deleted records at the point prior to the current record. You then can enter those 10 records. If you enter the first three records and then stop before entering the remaining seven records, you will have seven deleted records.

If a multiple-insertion attempt fails, try to insert one record at a time.

When you need to insert only one record with a DE/RPG program or the key entry utility, there is an option that can speed the process considerably. If the data set contains deleted records following the point of insertion, only those records up to the first deletion need to be moved. The insertion occurs prior to your current position in the data set. The following is an example of what happens during a record insertion in DE/RPG. The data set is currently positioned at record 72 (relative record number 23). Before insertion:



After inserting a record with a key of 52:



Notice that the insertion could have occurred anywhere in the data set. For example, if the current position had been at record 104 (relative record number 27), the insertion would have come between records 101 and 104.

Note: If the end of data (EOD) and end of extent (EOE+1) are equal, no records will be inserted; the data set is considered full. Therefore, data sets that might have records added later should be preallocated with room for growth. In the case of I exchange, EOD and EOE+1 can be equal, but there still might be room to insert records into the data set if the last sector is not full.

RECORD SEARCHING

Several different search functions are available on the 5280. You can find the first, current, and last record in a data set. You can find and display the record in a data set that contains a certain relative record number. Or you can find a record that contains a specific piece of data.

Refer to the *Operator's Guide* for detailed information concerning searches.

OPENING AND CLOSING DATA SETS

To access a data set, a program must first open it. The open procedure checks for such items as the share status and valid exchange type. Depending upon how a data set is opened, the data set might be accessed (shared) by more than one program. In DE/RPG there are three ways that a data set can be opened:

- Unshared
- Shared read
- Shared read/write

When a data set is opened as unshared, only the program that opens the data set has access to it, both for reading and for writing. If another program attempts to access the data set, the requesting program will receive an error code telling it that the data set is already opened.

Shared (shared read and shared read/write) data sets can be accessed by more than one program at a time. Any additions made to a shared data set must be made at the end of the data set. Because no insertions are allowed with a shared data set, transaction data sets should always be opened as unshared data sets.

Whether or not a second program can access a data set that has already been opened depends on:

- How the first program opened the data set
- How the first program is using the data set
- How the second program attempts to open the data set
- How the second program intends to use the data set

The following table shows the combinations that are possible. Allowable combinations are marked with an asterisk.

Program 1 Doing/Allowing	Program 2 Doing/Allowing: R/R	R+W/R	R/R+W	R+W/R+W
R/R	*		*	
R+W/R			*	
R/R+W	*	*	*	*
R+W/R+W			*	*

R = Reading, W = Writing

If program 1 opens a data set allowing shared read, and will be both reading and writing to the data set, program 2 will only be able to open the data set if it opens the data set with shared read/write and only if it is going to read the data set. If program 1 opens the data set with shared read/write and is only going to read the data set, program 2 can open the data set any of the ways that allow the sharing of data sets.

When the program is through with a data set, it issues a close instruction to the data set. Depending on how the data set was opened, this might free the data set to be opened a different way.

RECOVERING FROM I/O ERRORS

The purpose of initializing a diskette is in part to detect defective sectors, that is, sectors where data could be lost. If a defective sector is encountered during initialization, the entire cylinder on which the sector resides is considered defective, and an alternative cylinder is substituted. There can be a maximum of two cylinder substitutions per diskette on the 5280. This is taken care of by the system, and you do not need to be concerned with which cylinders are defective. The one exception is the IPL diskette, where cylinder substitution is not allowed.

A sector can become defective after initialization, even after data resides on it. Any time the system encounters a defective sector, the program running at the time will terminate with an error message, or an error code will be sent to the DE/RPG program.

If a sector does become defective after data is already stored in the data set, the data in the defective sector is lost. Depending on the length of each record, the physical buffer length, and the exchange type, more than one record could be lost. This is a major reason for you to have backup copies of your diskettes.

In the event that you encounter an unrecoverable read error and you do not have a copy of the lost records, proceed as follows:

1. Use the diskette label maintenance utility to create a data set to use for output.

2. Use the close failure recovery program to determine the logical record position of the failure (position n). Use EOJ to terminate the close failure recovery program.

3. Use the diskette copy utility, specify record option, to copy records 1 through n-1.

4. Restart the diskette copy utility, specify record option, to copy a record to position n. The contents of this particular record are unimportant at this time. This will usually be more than one record in the I exchange.

5. Restart the diskette copy utility, specify record option, to copy records n + 1 through the end of the data set.

6. Write a DE/RPG program to replace the lost record or records.

COPYING DATA SETS

There will be many times when you will need to copy data from one diskette to another. The kinds of copy available on the 5280 system are:

- Image copy–The entire contents of one diskette are copied to another diskette of the same type.
- Volume copy–The entire contents of a diskette of one type are copied to a diskette of the same or another type.
- Data set copy–A single data set is copied from one diskette to another diskette. Or up to four data sets can be combined and copied to one data set on the same or another diskette.
- Specify record copy–Records are copied by specifying the relative record number of the first and last record to be copied.
- Specify key copy–Records are copied from one diskette to another diskette if the records contain a specified key or keys. Up to three keys can be specified.

For detailed descriptions of each copy type, refer to the diskette copy utility.

DYNAMIC ALLOCATION AND PREALLOCATION

Sort/Merge Program Product, the DE/RPG compiler, DE/RPG object programs, the source entry program that accompanies the DE/RPG compiler, and the key entry and diskette copy utilities require that an output data set be present for output. Some of these programs give you the option of either preallocating or dynamically allocating the output data set.

Sort/Merge Program Product always allocates the output data set dynamically. The output data set will only be as long as is necessary to complete the operation.

The remaining programs can have their output data sets preallocated with the diskette label maintenance utility. You have the option of making the data set as large as you choose.

If you choose not to preallocate the output data set, the type of dynamic allocation depends on the program. The DE/RPG compiler gives you no choice as to the size of the output data set that will contain the object program. The diskette copy utility will make the output data set only as large as it needs to be.

DE/RPG object programs, the source entry program, and the key entry utility allow you to specify the number of records you want to allocate as a part of dynamic allocation.

It is recommended that you allocate data sets with future growth in mind.

DELETING DATA SETS

To delete a data set, use the diskette/data set clear or the diskette label maintenance utility. With the diskette/data set clear utility, you have the option of dropping the data set but retaining the data set name (and other pertinent information, such as number of records and record length), or of freeing the data set and the data set name from the diskette directory. The diskette label maintenance utility always frees the data set and the data set name.

If you attempt to delete a write-protected data set, you will get a warning message that the data set is write-protected. You then must choose whether or not to delete the data set.

DATA SET LABEL

Each data set has a label that tells the system all it needs to know about the data set. Much of the information in the data set label, such as where the data set actually resides on the diskette, is maintained by the system.

However, much of the information can be changed at your discretion, such as the expiration date and the write-protect indicator. The following table lists each area of the label that you can modify, and its length in bytes.

Name	Length
Data Set Name	8
Bypass Indicator	1
Write-Protect Indicator	1
Exchange Type Indicator	1
Multivolume Indicator	1
Volume Sequence Number	2
Creation Date	6
Record Length	4
Expiration Date	6
Verify/Copy Indicator	1
Record Delete Character	1

The following paragraphs discuss these fields. Refer to the diskette label maintenance utility for information on modifying these fields.

Data Set Name

Each data set is identified by a name. The name must begin with an alphabetic character and can be followed by up to seven alphameric (A through Z and 0 through 9) characters. The following are valid data set names:

SALES200
S
A1B2C3D4

The following are invalid data set names:

200SALES	(begins with a nonalphabetic character)
CHRONICLE	(contains more than 8 characters)
NAME%	(contains a nonalphameric character)

On the 5280, the VOLID specification can be included in the data-set-name field of the load prompt to ensure the use of a particular diskette, if the same data set name is used on more than one diskette. The syntax is *volid.dsname where volid is the 1- to 6-character volume ID, and dsname is the data set name.

Bypass Indicator

The bypass indicator specifies which data sets will be skipped during diskette exchange or copy operations when you are transmitting data sets. The letter B indicates bypass; a blank indicates no bypass. This field is only used by communications.

Write-Protect Indicator

The write-protect feature can be used to protect a data set so that the data can be read but not written to. The letter P indicates write-protect; a blank indicates no write-protect.

Exchange Type Indicator

The exchange type indicator defines the exchange type of the data set. A blank indicates basic exchange, the letter H indicates H exchange, and the letter I indicates I exchange. Refer to Chapter 4 for more information on exchange types.

Multivolume Indicator

The multivolume indicator indicates whether or not a data set is continued on another diskette volume. The letter C indicates that the data set is continued on another diskette. The letter L indicates that this is the last diskette that contains the data set. A blank indicates that the data set is contained on only this diskette.

Volume Sequence Number

Volume sequence numbers indicate the sequence of volumes in a multivolume data set. The sequence must be consecutive, beginning with 01 to a maximum of 99. A blank indicates that the data set is contained on one diskette.

Creation Date

This field can contain the date that the data set was created. The format is YYMMDD. For example, August 25, 1980, would be represented as 800825.

Record Length

This field contains the length of each record in the data set. The maximum record sizes on the 5280 are:

Exchange Type	Record Size
Basic	128 characters
H	256 characters
I	1 024 characters

Expiration Date

This field contains the date (YYMMDD) when the data set and its label can be deleted. There are three possible entries for this field: all blanks, all nines, or a valid date.

At allocation time, the 5280 always assigns all blanks. This indicates that the data set can be deleted at any time on the 5280. The data set will never be deleted unless you request the deletion, however. All blanks also indicate that the data set can be deleted on a system other than the 5280. (On the System/34, for example, a data set with all blanks is considered expired and the system will use this space as if it were unallocated.) You should use the diskette label maintenance utility to modify this field if this diskette might be used on another system and you do not want the data set deleted.

If the field contains 999999, the data set will never expire. To delete the data set, it will first be necessary to change this field.

If the field contains a valid date, the system will check this date against the date that was entered at IPL time. The data set is considered expired on and after the date given. A valid date that has expired is treated the same as all blanks.

If no date was given at IPL time, the data set is considered unexpired, and it will be necessary to change this field to all blanks before you can delete the data set.

Trying to delete an unexpired data set will result in an error message.

Verify/Copy Indicator

This character indicates that data has been copied correctly. A V indicates verification; a blank indicates no verification.

Record Delete Character

For the I exchange, this character is compared to the last character of a record. If they are equal, the system considers this record deleted. By changing this character, it is possible to recover deleted records. If you are using I exchange and are planning to use this function, it is recommended that you reserve the last character of each record as the delete character. Then, should you delete a record and later decide to recover it, no pertinent data will have been lost.

With the basic and H exchanges, deleted records are marked at the beginning of their corresponding sectors. There is no record delete character, and once a record is deleted, it cannot be recovered.

Chapter 4. Data Exchange With Other IBM Systems

Data exchange is the capability to exchange data with other systems. To do this, the data sets that are to be exchanged between two systems must meet certain requirements, depending on the exchange type being used. The 5280 supports these exchange types:

- Basic
- H
- I

The requirements for these exchange types are discussed next.

BASIC EXCHANGE

The basic exchange type has all of the following characteristics:

- The diskette sector must be 128 bytes (format 1 or 4).
- The sectors must be in sequential order.
- All the records in the data set must be the same length.
- The record length and block size must be less than or equal to 128.
- The records in the data set must be unblocked and unspanned. (See Chapter 3 for more information on unblocked and unspanned records.)
- The data set name must be 8 characters or less.
- The records must be written to the data set using the EBCDIC or ASCII character set.
- The data set must be located within cylinders 1 through 73 on a type 1 diskette and 1 through 74 on a type 2 diskette.

H EXCHANGE

The H exchange type is the same as the basic exchange type except that:

- The sector size must be 256 bytes (format 7 only).
- The record length and block size must be less than or equal to 256.
- The data set must be located within cylinders 1 through 74.

I EXCHANGE

The I exchange has all of the following characteristics:

- All records in the data set must be the same length (less than or equal to 1024).
- All records in the data set must be blocked and spanned. (See Chapter 3 for more information on blocked and spanned records.)
- The data set name must be 8 characters or less.
- The records must be written to the data set using the EBCDIC or ASCII character set.
- The data set must be located within cylinders 1 through 74.

DISKETTE FORMATS

The following table shows what diskette formats are needed to exchange data with other systems.

	Diskette Format		
Product	**Basic Exchange**	**H Exchange**	**I Exchange**
3774	1		
S/3	1		
S/32	1		
S/370	1		
3741	1		
3790	1		
S/1	1		
S/34	1	7	1, 3, 7, 9
5260	1	7	
5110	1, 4	7	
8100	1, 4	7	
5280	1, 4	7	1-9

After knowing what diskette format you need, you can then use the following table for additional information, such as diskette type and bytes per sector.

Diskette Format	Diskette Type	Max # of Data Sets	Tracks/ Cylinder	Sectors/ Track	Bytes/ Sector	Storage Capacity
1	1	19	1	26	128	246 272
2	1	19	1	15	256	284 160
3	1	19	1	8	512	303 104
4	2	45	2	26	128	492 544
5	2	45	2	15	256	568 320
6	2	45	2	8	512	606 208
7	2D	71*	2	26	256	985 088
8	2D	71*	2	15	512	1 136 640
9	2D	71*	2	8	1 024	1 212 416

* Formats 7, 8, and 9 permit the extended label area.

Suppose you need to exchange data with a 5260. By using the Diskette Format Table you can see that you can use either the basic or H exchange. If you choose the H exchange, you need to go by diskette format 7. Reading across format 7 in the above table, you can see that it requires a diskette 2D formatted with 256-byte sectors.

Chapter 5. Partition Concepts

To enable the 5280 system to support multiprogramming, the user storage area can be divided into partitions. A partition is a reserved area of user storage that is exclusively available for your programs and utilities. Each partition occupies a specific area of user storage.

The total user storage area available ranges from 32 K to 160 K. Part of this area is occupied by the IBM-supplied program support and tables called the common area, which will occupy approximately 6 K, 15 K, or 16 K bytes. See Chapter 6 of this manual for more about the common area.

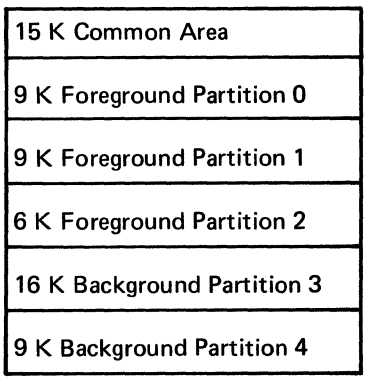
If you have 64 K of storage and 16 K is occupied by the common area, you have 48 K to divide into partitions. The size and type of program to be run in a partition should be taken into consideration when tailoring your system. The smallest partition allowed is 6 K; the largest is 64 K.

There are two kinds of partitions: foreground partitions and background partitions. Foreground partitions are each associated with a particular keyboard. Because there can be a maximum of four keyboards on the 5280, there can be up to four foreground partitions. Foreground partitions are generally used for applications that require regular use of the keyboard, such as data entry applications.

Background partitions are not associated with a particular keyboard until a program is loaded. Programs that operate in the background are normally independent in nature and require minimal operator interaction, for example, compilers. However, a background partition can request a keyboard when necessary. In this case, a solid rectangular block will appear on the side of the display screen that was used to initiate the background program.

There can be a maximum of eight partitions in addition to the common area. Therefore, if your system has three keyboards, you can define five background partitions. However, a typical configuration will have one or two background partitions.

The following figure shows one way that a 64 K system with three keyboards could be partitioned:



The first 15 K is used for the common area. The first three partitions, partition 0, partition 1, and partition 2, are associated with keyboards 0, 1, and 2 respectively. Partitions 3 and 4 are background partitions.

Notice that keyboards 0 and 1 have 9 K, which will allow the running of most utilities in foreground. Keyboard 2 is limited to running programs of 6 K or less in the foreground: these would be programs that you have written and compiled yourself. Any one of the keyboards can have access to either background partition, which means that a program product such as Sort/Merge, which requires 16 K, can be run in partition 3 from any of the keyboards.

PARTITION INTERFACING

Each keyboard can have one foreground and one or more background programs running at the same time. The following paragraphs describe how to load programs into partitions other than the one you are presently in, and how to go from one partition to another when you have more than one program running at the same time.

Program Loading

Loading from Foreground into Background: If you are running a program in the foreground and need to load a background program, press the Sys Req (System Request) key and load the program. The keyboard will then be attached to the background partition. Press the Cncl (Cancel) key to cancel the system request if you decide not to load the background program.

Loading from Background into Foreground: If you are running a program in the background and need to load a foreground program, press the Attn (Attention) key to return to the foreground and load the program. The keyboard will then be attached to the foreground program.

Loading from Background into Background: If you are running a program in the background and need to load a second background program, press the Sys Req key and load the program. The keyboard will remain attached to the first background program.

Any request to load into a partition that already has a program running will result in an error message. A second request will terminate the old program and load the new one if you are loading into the same partition. For example, if you are running a program in foreground in partition 1 and attempt to load a new program in partition 1, you will get an error message. The second time you do this the old program will be terminated and the new program will begin execution.

Switching Partitions

Going from Foreground to Background: If you are running a program in the foreground, you can get to a background program in two different ways. If the solid rectangular block appears on the side of the screen, press the Attn key and you will be in the background program.

If you need to terminate a background program while in the foreground and the solid rectangular block is not displayed, see *Console Mode*, later in this chapter.

Going from Background to Foreground: Normal operation of the 5280 assumes that a program running in the foreground is constantly using the keyboard. Thus, after you are finished servicing a background keyboard request, press the Attn key. If a background program terminates while you are in it, you will automatically be returned to the foreground.

Going from Background to Background: Pressing the Attn key will allow you to go from one background program to another if another background program is requesting the keyboard.

CONSOLE MODE

The console mode should be used to terminate a background program that cannot be accessed with the Attn key. The background program must be associated with the keyboard that invokes this procedure. To terminate a background program:

1. Press the Cmd key, the C key, the B key, and the partition number to connect to the background partition.

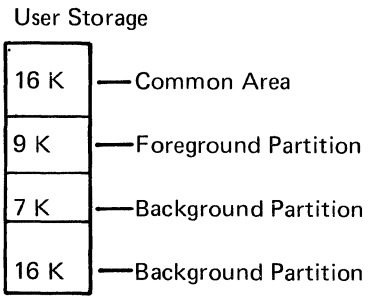
2. Depending on which program is running in the background (for example, DE/RPG or communications), see the appropriate manual for procedures for ending a job.

3. Once the program in the background partition ends, the system automatically returns to the foreground partition.

When the end of job is complete, press the Attn key to restore the prompt that might be associated with the foreground, if such a prompt is not already displayed. If another background job is requesting the keyboard, you can press the Attn key one again to return to the foreground prompt.

Chapter 6. The Common Area

The common area contains information and functions that can be used by any of the devices or programs in the system. The common area is established in the system during the IPL (initial program load) process. Depending upon the common function option selected during system configuration, the common area uses approximately 6 K, 15 K, or 16 K of the user storage. See *Calculating the Size of the Common Area* later in this chapter for more information on the amount of user storage required for the common area. The common area resides in the first part of user storage. The following illustration shows how user storage might be allocated for a 48 K system with three partitions, one keyboard, one diskette drive, and no printers.



CALCULATING THE SIZE OF THE COMMON AREA

The size of the common area is determined by the common function option being used, the ASCII table (if required), the number of keyboards, the number of diskette drives, the number of printers on the system, and the resource allocation table. The following chart shows the approximate size of the common area for a system that has no printers, one diskette drive, and one keyboard:

Common Function Option Name	Common Area Size
SYSDPRT2	6 K
SYSCFA	15 K
SYSHELP	16 K

(See *The Common Functions* later in this chapter for more detailed information about each common function option.)

The tables in the common area that can cause the size of the common area to be greater than 6 K, 15 K, or 16 K are:

- The printer configuration table. This table is present in the common area only if one or more printers are on the system. The size of this table is:

Number of Printers	Additional Storage Required for the Common Area
1	12 bytes
2	20 bytes
3	28 bytes
4	36 bytes
5	44 bytes

- DE/RPG and key entry utility production statistics table. The size of this table depends upon the number of keyboards on the system:

Number of Keyboards	Additional Storage Required for the Common Area
2	64 bytes
3	128 bytes
4	192 bytes

- System error log table. The size of this table depends upon the number of diskettes and printers on the system:

Number of Devices	Additional Storage Required for the Common Area
2-4 diskettes	130 bytes
5-8 diskettes	260 bytes
1-5 printers	130 bytes

- Printer error log table. The size of this table depends upon the number of printers on the system:

Number of Printers	Additional Storage Required for the Common Area
2	20 bytes
3	40 bytes
4	60 bytes
5	80 bytes

- ASCII translation table. If this table is required, its size is 512 bytes.

- Resource allocation table. (The resource allocation table is discussed in detail later in this chapter.) If logical IDs are specified, the size of this table can be determined as follows:
  - 4 bytes for each system shared or partition entry.
  - 4 bytes for the end of the table.

When the system configuration program is run, you can determine the size of the common area before you specify the partition sizes. The system displays the amount of available user storage before the partition sizes are specified. You can subtract the amount of available storage from the total storage (or 64 K if your system is greater than 64 K) to determine the actual common area size.

THE RESOURCE ALLOCATION TABLE

The resource allocation table contains the logical device IDs (identifications) that were specified for the physical devices when the system configuration program was executed. The logical device IDs are two-character identifiers that can be used by your user-written DE/RPG or Sort/Merge Program Product programs in place of the four-character physical device address. This gives programs independence from hardware configurations.

There are two types of entries in the resource allocation table: system shared entries and partition entries. System shared entries are used (shared) by programs that are executing in any partition. For example, if D1 was specified as a system shared entry for diskette drive 4000, programs that are executing in any partition can read or write data to diskette drive 4000 by using the logical ID D1. Partition entries are only used by a program executing in a specific partition. The following illustration shows some entries that might be contained in a resource allocation table.

Physical Device Address	Logical ID	
4000	D1	Partition 0
5000	D2	
4400	D1	Partition 1
4800	D1	Partition 2
4000	D2	System Shared Entries
4400	D3	
4800	D4	
8000	P1	

Following are examples of how the logical ID from this resource allocation table can be used by programs executing in partitions 0, 1, and 2:

- If logical ID D3 is used in a program executing in any partition, the data is read from or written to the diskette at physical device address 4400.
- If logical ID D2 is used in a program executing in partitions 1 or 2, the address 4000 is obtained from the system shared entries. With partition 0, address 5000 is used since the system will always check the partition table first.
- If logical ID D1 is used in a program, the data is read from or written to the diskette at the physical device address specified in the resource allocation table. For example, if the program is executing in partition 2, the data is read from or written to the diskette at physical device address 4800.

When the resource allocation table is created using the system configuration program, you can place blank entries in the table for future use. Then at a later time, you can use the resource allocation table utility to change the blank entries to valid logical IDs for physical device addresses. While any logical IDs could be used, it is recommended that diskettes be D1 through D8 and printers be P1 through P5 to facilitate object program movement between systems without source code changes.

THE COMMON FUNCTIONS

The common functions reside in a shared area that can be used by programs executing in any partition. The program products that use the common functions are the System Control Programming, Utilities, Sort/Merge, Communications, Assembler, and DE/RPG. The source DE/RPG programs do not actually call a common function; however, the DE/RPG compiler supplies the object instructions in the generated object program to call a common function when required. The common functions perform operations such as:

- Displaying machine check and I/O error codes.
- Displaying help messages for some error messages if the Help key is pressed (if the SYSDPRT2 or SYSHELP option is chosen during system configuration).
- Attaching a partition to a keyboard. (See Chapter 5 for more information on attaching partitions.)
- Requesting the owner ID of and allowing access to an access-protected diskette.
- Loading a program into a partition.

Three options are available for selecting the common functions you want in the common area. You must select the appropriate option during execution of the system configuration program. The option you select depends upon what programs are to be run on the system. For example, common functions option SYSCFA or SYSHELP is required for DE/RPG, the source entry program, key entry utility, and 3740 format conversion programs. Following is a description of each common function option.

SYSDPRT2: This option provides the basic IBM-supplied common functions. It also provides help text for certain errors that will be displayed when the Help key is pressed. This option can be used with all utilities (except the key entry and the 3740 format conversion utilities), System Control Programming, Communications, the assembler processor, and the DE/RPG compiler. When SYSDPRT2 is selected, the common area requires approximately 6 K of user storage, which means this option leaves the maximum amount of user storage.

SYSCFA: This option provides the same common functions as SYSDPRT2, plus additional common functions that support DE/RPG programs, the source entry program, the key entry, and 3740 format conversion utilities. However, with SYSCFA, there is no help text provided when the Help key is pressed. In this case you must use the *Message Manual* to determine what the error code means. When this option is selected, the common area requires approximately 15 K of user storage.

SYSHELP: This option provides the same common functions as the SYSCFA option and includes the help text capability. SYSHELP requires approximately 16 K of user storage.

If you try to execute a DE/RPG compiled program, the source entry program, the key entry utility, or the 3740 format conversion utility using the common function SYSDPRT2 option, a program check error occurs. In this case, you must IPL the system using an IPL diskette that contains common functions option SYSCFA or SYSHELP before you can run the program or utility.

Chapter 7. Performance Characteristics

Within the 5280 family there are three programmable controllers: the 5285, the 5286, and the 5288. All three have the same system architecture and differ only in their configuration constraints. All three have various microprocessors to control the operation of the entire system.

All models of the 5280 have the application microprocessor, the keyboard/display microprocessor, and the first diskette microprocessor. The second diskette microprocessor, the printer microprocessor, and the communication microprocessor are optional, depending on the features ordered.

These microprocessors and their major functions are as follows:

- Application Microprocessor
  - Performs applications, arithmetic, and logic
  - Controls the other microprocessors on the system
  - Shares its time among the multiple jobs

- Keyboard/Display Microprocessor
  - Controls all input to and output from the screen
  - Performs character/field edits
  - Supports from one to four keyboards
  - Provides language translation
  - Logs errors

- Diskette Microprocessor
  - Supports from one to four diskettes
  - Supports diskettes 1, 2, and 2D
  - Supports basic, H, and I exchange types
  - Provides diskette data management
  - Supports single and double buffering
  - Blocks and deblocks diskette data
  - Provides device control
  - Initializes diskettes
  - Logs errors

- Printer Microprocessor
  - Supports up to four 5256 printers and one 5225 printer
  - Routes data to the printers
  - Supports single and double buffering
  - Blocks and deblocks diskette data
  - Logs errors

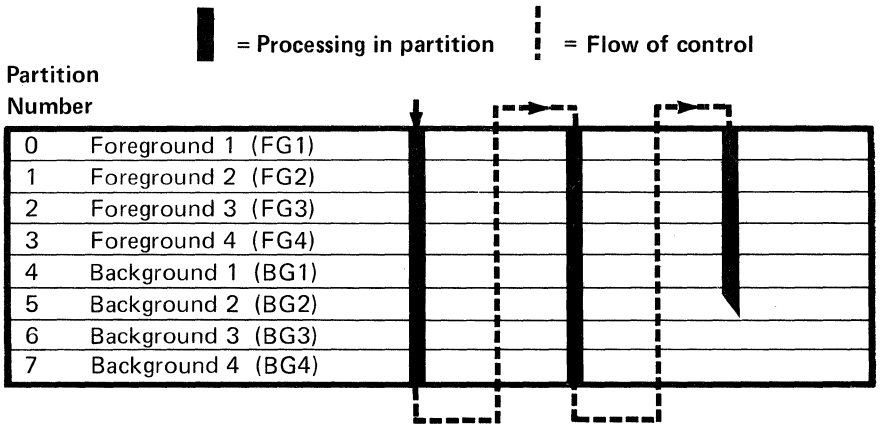
- Communications Microprocessor
  - Provides an interface between the communications line and the communications access method (CAM)
  - Provides link level control
  - Supports binary synchronous communications (BSC) and synchronous data link control (SDLC)
  - Performs error handling
  - Provides diagnostic support for communications

Use of Memory on the 5280

The memory on the 5280 is divided into a common area and user partitions. See Chapter 5, *Partition Concepts*, for more information about partitions and how they work.

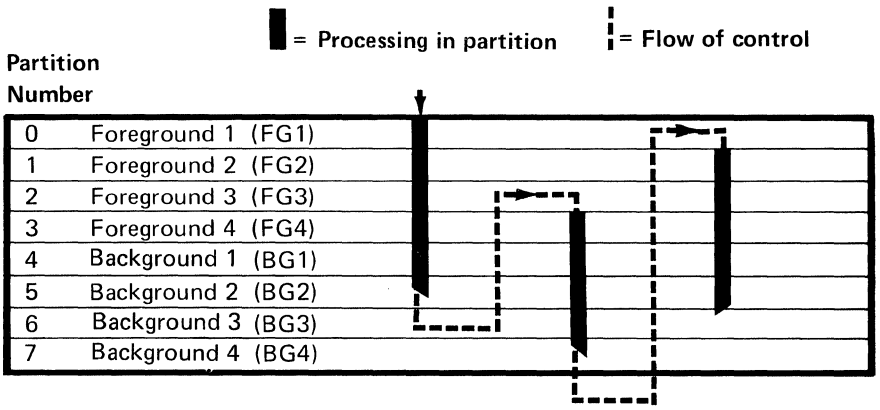
Dispatching of Jobs

The partitions contain the application code and share the available time on the application microprocessor. The base dispatching sequence of jobs begins with the first foreground partition and proceeds to the next partition until all partitions have received their allotted portions of execution time (called time slices). Once the last background partition is serviced, the cycle begins again with the first foreground partition. The following figure shows the normal sequence of partition servicing.



This sequence is interrupted when a record is completed or when a user exit occurs, causing an attention to the application microprocessor. (Examples of functions causing user exits are table look-ups, range checks, and user subroutines). The partition in which the keyboard attention occurs is then serviced. The service is immediate if the processing being interrupted is not responding to a previous attention, or at the end of the time slice if the interrupted partition is responding to an attention request.

Once the partition has received its time slice, processing continues with the next sequential partition if no keyboard attention is pending. As a result of keyboard attention servicing, the cyclical processing of partitions is disrupted, with the result that some partitions receive more frequent time slices while others receive proportionally fewer. The following figure illustrates the dispatching of jobs.



In the preceding figure, processing begins at FG1, proceeds in sequence with FG2, and continues until it begins processing BG2. Before the time slice for BG2 is completed, an attention from the keyboard attached to FG4 causes the application microprocessor to transfer processing to a new time slice for FG4. After FG4 is serviced, processing continues with BG1 and the sequence continues until an attention from FG2 causes the application microprocessor to interrupt BG4 to service FG2. After FG2 is serviced, processing continues with FG3 and so on.

Time Slice Values

The amount of time given to jobs depends on the jobs themselves. Some jobs receive larger time slices than others. The time slice values are as follows:

- The key entry utility, the communications access method (CAM), and DE/RPG using a transaction data set receive 60 milliseconds per time slice.
- DE/RPG in rerun mode receives 4 milliseconds per time slice.
- All other jobs receive 12 milliseconds per time slice.

DISKETTE RECORD I/O PERFORMANCE

This section is intended to act as a guide to diskette performance on the 5280. Individual programmers should determine which of the following factors has the strongest influence in a particular application program and structure the code accordingly to obtain the best performance.

Characteristics of the Media

The 5280 diskette unit drives rotate at a constant speed of 360 revolutions per minute (RPM), or one complete revolution every 1/6 of a second (166 ms). The amount of data passing under the read/write heads and, therefore, the data transfer rates and the read/write rates vary as a function of format and recording techniques. The instantaneous data transfer rate is 1 byte every 32 microseconds for diskettes 1 and 2, and 2 bytes every 32 microseconds for diskette 2D.

The difference between the instantaneous data transfer rate and the read rate is that the read rate adds the overhead for reading sector IDs (address mark, CRC, etc), while the instantaneous data transfer rate does not add these overhead times.

Movement to and within a data set directly affects performance. The factors that must be taken into account in order to access data are:

- Head load time = 40 to 80 ms, depending on how far the head has to travel to reach the intended track
- Track to track access time = 5 ms
- Head settle time = 35 ms
- Rotational delay (average) = 1/2 of full rotation = 1/2 X 166 = 83 ms

For example, you have just closed a data set and the head is at cylinder 0. When you open a new data set, the following occurs:

- Load head = 80 ms
- Rotational delay = 83 ms to find correct header

When the first read or write I/O activity to the data set is initiated, there are two possible results:

- Eight revolutions of the index hole have occurred without another diskette instruction being used, and the heads have been unloaded. The diskette microprocessor must perform all three of the following:
  - Move to record = 5N ms + 35 ms where N is equal to the number of track crossings
  - Head load = 40 to 80 ms
  - Rotational delay to find the correct sector = 83 ms

- The next diskette instruction has been issued prior to eight occurrences of the index hole. The diskette microprocessor must perform both of the following:
  - Move to record = 5N + 35 ms
  - Rotational delay = 83 ms

It would be most efficient to read or write a full cylinder of data (between 3.25 K and 16 K) before the next move or time increment of 5N + 35 ms. Also, the number of head unloads and loads can be minimized when diskette I/O instructions are issued prior to 8 occurrences of the next index hole (on the average 7 1/2 revolutions = 1245 ms).

Average access time can be defined as the average time required to move from one given place on the diskette to another. This requires track crossings (N) equal to 1/3 of the total number of tracks (74), or 25. This definition produces the following equation:

5N + 35 + rotational delay = 5(25) + 35 + 83 = 243 ms

Therefore, the average access time equals 243 ms.

Note: Figures used in *Characteristics of the Media* are approximate.

Characteristics of the Data Set

The relation of multiple data sets (the number of data sets open, the location of the data sets with regard to one another, and the amount and type of activity within a data set) and the size of the data sets are important performance considerations because of the track to track, head load and settle, and rotational delays involved.

Relationship of Multiple Data Sets

Movement between multiple open data sets should be minimized in order to minimize the various delays. If two data sets are to be open in the same program, and you must decide whether the data sets are to reside on the same or separate diskettes that will be in different drives, consider the following:

- If one data set is to be completely processed before the next one is processed, they can both be placed on the same diskette with no significant loss of performance.

- If there is to be constant interaction between the two data sets, they should probably be placed on separate diskettes so that head action and accesses are independent for each data set.

When there is activity between more than two data sets and there are a limited number of drives, in general the data sets should be placed as close to each other as possible. The one to be processed against most frequently should be placed in front of and next to the one to be proccesed with the next most frequency, and so on.

For example, assume you have an index data set and a master data set and you are accessing them using the key indexed access method. The index data set, in general, has multiple accesses for each direct access of the master data set. Assuming either normal distribution of activity or skewed activity towards the first two-thirds of the master data set, head movement and access time are minimized when you place the index data set next to and before the master data set.

Size of the Data Set

Obviously, the smaller the data set, the less time is spent in searching and accessing within the data set, unless the data set is accessed directly.

TECHNIQUES FOR IMPROVING JOB PERFORMANCE

This section is intended for the programmer and regards tips and techniques for improving performance of jobs on the 5280. These guidelines apply in most, but not all, situations.

General Performance Guidelines

As explained earlier in this chapter in *Dispatching of Jobs*, the system gives a time slice to each partition, usually beginning with the low-numbered (foreground) partitions and advancing through the high-numbered (background) partitions. The sequence of operation is interrupted by certain high-priority events, such as keyboard entry of a field that is to be range checked. As a result of interruptions, the performance of a program in the 5280 might be affected by another program that is running concurrently.

To optimize the performance on the 5280, it is recommended that you do the following:

- Configure only those partitions that you expect to use. Unused partitions still require processor time.

- In general, partitions are serviced from lowest to highest partition number. Run your highest-priority programs in the lowest-numbered partitions available. Those programs that require little or no keyboard interaction should generally be run in a background partition.

- Do not put a high-priority nonkeyboard program (for example, CAM) in a partition that having a lower number than a program that causes a high number of attentions to the application microprocessor. The high-priority program could receive significantly fewer time slices than if it were placed in a partition immediately following the highly attention-oriented program. This is particularly true when several partitions are active concurrently.

- Minimize the number of resources, such as diskettes and data sets, that are shared.

- As much as possible, avoid using the application microprocessor resource during data entry. For example, to automatically fill a field with data stored in memory, use the AUXDUP function (handled by the keyboard microprocessor) instead of the INSERT function (handled by the application microprocessor).

- Minimize the use of prompting messages. If prompts are used on a new job or with new operators, prepare additional formats that eliminate all but the important prompts after the training period. This usually improves the response times between records. Various location prompts require more time than fixed location prompts.

Diskette Usage Guidelines

This section discusses how proper placement and condition of diskettes and data sets, memory usage, and in-memory indexes can improve diskette performance.

The 5280 system can have up to eight diskette drives attached. If you have two diskette microprocessors, one of the following must be true of your system:

- There are five or more drives
- There are three or more remote drives
- There are three or more local drives plus one or more remote drives

A local drive is one that is located in your controller — either a 5285, 5286, or 5288. All other drives are remote.

Placement of Diskettes and Data Sets

The following guidelines should assist you in optimizing diskette usage:

- If a job uses multiple data sets, place the data sets on different diskettes to minimize arm movement and contention. The use of multiple drives on the same diskette microprocessor does not always improve performance, however.
- If the system has two diskette microprocessors, assign the diskettes to drives that equalize the workload between microprocessors. The diskettes with device addresses in the range of 4000 through 4C00 are assigned to the first diskette microprocessor; those with 5000 through 5C00 addresses are assigned to the second diskette microprocessor.
- Do not share I/O bound microprocessors. If a microprocessor is already overloaded, any additional burden will adversely affect all the users of the microprocessor.

Condition of Diskettes and Data Sets

In addition to the placement of diskettes and data sets, the condition of the diskettes and data sets can influence the performance of jobs using the diskette. Therefore, consider the following:

- Record insertion (one record only) in a data set provides the best performance when deleted records are placed periodically throughout the data set. In such cases, the insertion of a new record requires that the records already in the data set be moved down only to the next deleted record. The data set copy option of the diskette copy utility can be used to create a data set with deleted records inserted at regular intervals.

- For the basic and H exchanges, deleted records or sectors degrade the performance of the diskette compress and diskette print utilities, as well as the image copy option of the diskette copy utility. For I exchange, deleted records do not degrade performance.

- Unused labels in the label area can have an impact on the time it takes to open a data set. On a diskette 2D, for example, deleted labels in the extended label area can affect the time it takes to open data sets.

Memory Usage

Memory usage within programs can affect the performance of diskette jobs. The amount of buffering and record protection must be considered. These guidelines apply to memory usage:

- For sequential processing of data sets, provide as large a buffer space as possible. The large buffer reduces the number of physical reads.

- For random processing of a data set, use the smallest buffer possible. This reduces the total number of sectors that must be read. Also, the smallest number of records are locked when you use the smallest possible buffer, since all records in a buffer are locked for a shared data set. Therefore, double-buffering in this case is detrimental rather than beneficial.

For a data set processed randomly by key, the performance can be enhanced through the use of an in-memory index. The in-memory index reduces the number of reads that are required to locate the desired record (DE/RPG keyfile).

For cases where an index data set is used (key indexed access method), the in-memory index is most efficient where there is one index per block of indexed data. The number of entries required for the in-memory index can be calculated by the following formula:

$$\frac{\text{number of records in data set X (key length + 4)}}{\text{sector size X number of sectors per buffer}}$$

The following must be considered:

- The formula assumes that the index is in the I exchange.
- The sector size is established by the diskette initialization utility. Depending on the diskette type and format, the value might be 128, 256, 512, or 1024.
- For an index data set, the number of sectors per buffer is one if the expression (key length + 4) is a power of 2. This is true if the key length is 4, 12, or 28; otherwise the number of sectors per buffer is 2.

For cases where the master data set is used without an index (direct by key access method), the number of entries for the in-memory index should be adequate to assure that there are one- or two-index entries per track of data. Allocating an excessive number of entries does not necessarily improve performance. The number of tracks for a data set can be determined by one of the following formulas:

- For the I exchange, the number of tracks equals:

$$\frac{\text{record length X number of records}}{\text{bytes per track}}$$

- For the basic and H exchanges, the number of tracks equals:

$$\frac{\text{number of records}}{26}$$

The number of cylinders can be determined by dividing the result of the above expression by 2 for a diskette 2 or 2D. (For a diskette 1, track and cylinder are equal.)

The values for the above expressions can be found in the final table of Chapter 4.

Memory space is sacrificed for the in-memory index. To determine the amount of memory used, the following formula applies:

bytes of memory = number of entries X (key length + 3)

DE/RPG Guidelines

This section discusses how proper format design, storage management, data set usage, and DE/RPG usage can improve performance.

Format Design

DE/RPG performance can be affected by the manner in which the formats for data entry, verification, and rerun are designed. Consider the following guidelines:

- Using a large number of functions that require either service from the application microprocessor or the heavy use of user-written subroutines might not only affect the performance of the program itself but might also cause degredation in the performance of other partitions. Where applicable, the use of user exits or subroutines should be minimized.

- Verify and rerun mode might not need all the user exit or subroutine functions provided in an entry or update format. If, for example, a format calls for many application microprocessor functions such as self-check, table look-up, range check, and field totals, and the purpose of the rerun is only to recalculate batch totals, then a revised format that only calculates batch totals would provide better performance for the rerun and less potential contention with other partitions. Similarly, using a separate format for verify, rather than specifying verify bypass, aids in reducing contention and in improving performance.

- Use the INSERT function to compute a common expression in a working storage field; then reference the working storage field for subsequent operations such as COMP and other keywords.

Storage Management

Storage management, like proper format design, can improve performance or allow DE/RPG jobs to have a greater flexibility in scheduling. The following provides some guidance for use of storage:

- Some fields on a format could be unreferenced by any other statements within the program. In such cases, a field name might not be needed. You can reduce the program size by eliminating unreferenced named fields in DE/RPG source programs.

- The use of tables in calculation specifications causes an increase in the amount of memory required for a program. Regardless of the number of table references, however, the same amount of supporting code is required for the table. Thus, once a table is required, additional references require only small amounts of memory for each reference.

Data Set Usage

The use of data sets enables the DE/RPG programmer to develop jobs that might have been difficult or impossible on previous systems. In addition to the considerations for diskette usage described in *Diskette Usage Guidelines* earlier in this chapter, the DE/RPG programmer should consider the following:

- Data sets that are to be processed using the key indexed access method can be created most efficiently if they are created sequentially in DE/RPG.

  The sort program can then be used to create the index data set or the properly sequenced master data set.

- For jobs that are not key entry jobs, the use of larger sector sizes in data sets improves throughput. However, the program requires additional memory.

DE/RPG Compiler Usage

You can improve the performance of the DE/RPG compiler by considering the following:

- DE/RPG compiles in a partition with a minimum of 9 K bytes. The performance improves, however, if a larger partition is used. Generally, the larger the partition provided, the better the compile performance becomes until the optimum size of the partition is reached. The optimum size varies for each program; where possible, however, at least a 16 K partition is recommended for compilation.

- The work data sets used by the compiler should be spread across different diskettes. Placing work data set SYSUT001 on a separate diskette might significantly improve performance.

- The use of diskettes with larger sector sizes enhances the performance of the compiler.

- Compute-bound jobs adversely affect the performance of the compile. Thus, when compile times are to be optimized, the jobs should not be run concurrently with heavily compute-bound jobs.

- Diskette I/O bound jobs adversely affect the performance of the compile, particularly if the compiler partition is small (less than 16 K bytes).

Utility Performance Guidelines

This section discusses how proper use of the utilities can improve performance. General guidelines will be discussed first, followed by the diskette copy, diskette print, diskette compress, diskette initialization, key entry, and communications utilities.

General Guidelines

The 5280 utilities provide a wide range of services. The following guidelines apply to the performance of the utilities in general:

- Some utilities perform dynamic buffer allocation, depending on the partition size in which they are executing. The partition size should be maximized for sequential processing.
- In general, a larger partition improves performance.
- The system control programming (SCP) and utilities should be copied onto a single diskette to minimize swapping of diskettes. Where this is done, the initial program load (IPL) data sets must be placed first on the diskette. The volume copy option of the diskette copy utility should be used to copy the IPL data sets.

Diskette Copy Utility

The diskette copy utility has a number of options that can be used for a wide variety of jobs. Depending on the operation desired, the following guidelines can apply:

- For a complete diskette copy, the image copy option is faster than the volume copy option.
- Do not use the single diskette drive copy function unless there is no alternative.
- Except for a single drive copy of the data set copy option, performance of the diskette copy utility is not improved when the partition size is increased beyond 24 K bytes.
- Defective sectors or deleted physical records degrade the performance of the image copy option for basic and H exchange data sets.
- The data set copy option can be used to create data sets with deleted records interspersed among active records. Such a data set improves performance when random addition to a data set is required. The data set must be preallocated to provide space at the end of the data set for additional records.

Diskette Print Utility

The diskette print utility functions can be used to print the contents of data sets. Use of the utility is influenced by the following:

- Deleted physical records degrade the performance of the utility for basic and H exchange data sets.
- The total number of printers in use on the system might affect the throughput of the utility, but only if one of the printers in use at the time is an IBM 5225 Printer.
- Contention for the printers on the system might affect the scheduling or performance of the system, or both.

Diskette Compress Utility

Deleted physical records can degrade the performance of the diskette compress utility for basic and H exchange data sets.

Diskette Initialization Utility

The diskette initialization utility places a heavy load on the diskette microprocessor. Where possible, this utility should be run on a diskette microprocessor being used for no other jobs. If there is only one diskette microprocessor, you should try to run the utility in a dedicated environment.

Key Entry Utility

The key entry utility response times vary when the utility is chaining formats via chaining or manual format selection. The response times vary according to the complexity of the format, since the utility interprets the 3740-like format each time the format is selected. If multiple operators are using the utility and chaining, the response times might be lengthened significantly. When no format change or chaining occurs, the response time is generally better than when formats are being changed or when chaining is used. If a job chains formats frequently, or if the job contains complex formats, it might be desirable to code the job in DE/RPG.

Communications Utility

Communications support on the 5280 permits a wide range of uses. The following should be considered in using communications:

- Communications throughput is not usually affected by concurrent key entry. However, if key entry jobs have a large number of user exits or calculation subroutines, the communications jobs might be adversely affected.

- Compute-bound jobs, such as a DE/RPG compile, can impact the throughput of communications jobs. Wherever possible, such compute-bound jobs should not be scheduled to run concurrently with communications jobs.

- If maximum communications throughput is desired for a job, the job should be scheduled to run in a dedicated environment. A cost advantage might be gained if the job is scheduled at an hour when reduced line rates are available.

- Small record sizes might provide less than optimum line utilization.

The following table lists the character sets available on the 5280 system using the IBM 5256 printer.

Hexadecimal Value	International	United States and Canada	Canada (French)	France	United Kingdom	Spain	Spanish Speaking	Portugal	Brazil	Denmark and Norway	Finland and Sweden	Austria and Germany	Belgium	Italy	Japan (English)	Japan (Katakana)
40																
41																｡
42	¡															｢
43	¢															｣
44	£															､
45	¤															･
46	¥															ｦ
47	¦															ｧ
48	§															ｨ
49	¨															ｩ
4A	[	¢	à	°	$	[	[	[	£	#	§	Ä	[	°	£	£
4B	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^
4C	<	<	<	<	<	<	<	Ç	<	<	<	<	<	<	<	<
4D	(	(	(	(	(	(	(	(	(	(	(	(	(	(	(	(
4E	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
4F	!	|	!	!	|	|	|	!	!	!	!	!	!	!	|	|
50	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&	&
51	é															｡
52	ê															｢
53	ë															｣

Hexadecimal Value	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67
International	è	í	î	ï	ì	ß	]	$	*	)	;	^	-	/	Â	Ä	À	Á	Ã	Å
United States and Canada							!	$	*	)	;	¬	-	/						
Canada (French)							[illegible]	$	*	)	;	^	-	/						
France							§	$	*	)	;	^	-	/						
United Kingdom							!	£	*	)	;	¬	-	/						
Spain							]	Pt	*	)	;	¬	-	/						
Spanish Speaking							]	$	*	)	;	¬	-	/						
Portugal							]	$	*	)	;	^	-	/						
Brazil							$	Ç	*	)	;	^	-	/						
Denmark and Norway							¤	Å	*	)	;	^	-	/						
Finland and Sweden							¤	Å	*	)	;	^	-	/						
Austria and Germany							Ü	$	*	)	;	^	-	/						
Belgium							]	$	*	)	;	^	-	/						
Italy							é	$	*	)	;	^	-	/						
Japan (English)							!	¥	*	)	;	¬	-	/						
Japan (Katakana)	[illegible]	[illegible]	[illegible]		[illegible]		!	¥	*	)	;	¬	-	/						

Hexadecimal Value	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	7A	7B
International	#	$	@	[	\	]	^	`	{	|	}	~	DEL							
United States and Canada	#	$	@	[	\	]	^	`	{	|	}	~	DEL							
Canada (French)	#	$	à	â	ç	ê	î	ô	{	|	}	~	DEL							
France	£	$	à	°	ç	§	^	`	{	|	}	~	DEL							
United Kingdom	#	$	@	[	\	]	^	`	{	|	}	~	DEL							
Spain	₧	$	@	¡	Ñ	¿	^	`	{	ñ	}	~	DEL							
Spanish Speaking	₧	$	á	¡	Ñ	¿	é	í	ó	ñ	ú	~	DEL							
Portugal	₧	$	@	[	Ç	]	^	`	ó	ã	õ	~	DEL							
Brazil	Ó	$	@	[	Ç	]	^	`	ó	ã	õ	~	DEL							
Denmark and Norway	£	$	@	Æ	Ø	Å	^	`	ó	æ	ø	å	DEL							
Finland and Sweden	£	$	É	Ä	Ö	Å	Ü	é	ó	ä	ö	å	DEL							
Austria and Germany	#	$	§	Ä	Ö	Ü	^	`	ó	ä	ö	ü	DEL							
Belgium	£	$	@	°	ç	§	^	`	é	à	è	ù	DEL							
Italy	£	$	@	°	ç	é	^	ù	à	ò	è	ì	DEL							
Japan (English)	#	$	@	[	¥	]	^	`	{	|	}	~	DEL							
Japan (Katakana)	#	$	@	[	¥	]	^	`	{	|	}	~	DEL							

Hexadecimal Value	International	United States and Canada	Canada (French)	France	United Kingdom	Spain	Spanish Speaking	Portugal	Brazil	Denmark and Norway	Finland and Sweden	Austria and Germany	Belgium	Italy	Japan (English)	Japan (Katakana)
7C	@	@	@	à	@	@	@	õ	Ã	Ø	Õ	§	à	§	@	@
7D	'	'	'	'	'	'	'	'	'	'	'	'	'	'	'	'
7E	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
7F	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"
80	Ø															
81	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	ｱ
82	b	b	b	b	b	b	b	b	b	b	b	b	b	b	b	ｲ
83	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	ｳ
84	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	ｴ
85	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	ｵ
86	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	ｶ
87	g	g	g	g	g	g	g	g	g	g	g	g	g	g	g	ｷ
88	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	ｸ
89	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	ｹ
8A	«															ｺ
8B	»															
8C	đ															ｻ
8D	±															ｼ
8E	²															ｽ
8F	±															ｾ

Hexadecimal Value	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3
International	゛	゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
United States and Canada		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Canada (French)		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
France		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
United Kingdom		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Spain		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Spanish Speaking		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Portugal		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Brazil		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Denmark and Norway		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Finland and Sweden		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Austria and Germany		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Belgium		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Italy		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Japan (English)		゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ		｡	｢	｣
Japan (Katakana)	゛	゜	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	イ	ウ	エ	｡	｢	｣

Hexadecimal Value	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7
International	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
United States and Canada	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Canada (French)	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
France	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
United Kingdom	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Spain	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Spanish Speaking	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Portugal	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Brazil	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Denmark and Norway	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Finland and Sweden	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Austria and Germany	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Belgium	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Italy	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Japan (English)	¤	¥	§	¿	Ä	¡	Ñ	ü	§	à	ñ		ä	ö	ñ	ü	à	é	ö	ä
Japan (Katakana)	ｱ	ｲ	ｳ	ｴ	ｵ	ｶ	ｷ	ｸ	ｹ	ｺ	ｻ	ｼ	ｽ	ｾ	ｿ	ﾀ	ﾁ	ﾂ	ﾃ	ﾄ

Hexadecimal Value	International	United States and Canada	Canada (French)	France	United Kingdom	Spain	Spanish Speaking	Portugal	Brazil	Denmark and Norway	Finland and Sweden	Austria and Germany	Belgium	Italy	Japan (English)	Japan (Katakana)
B8																
B9																
BA	¬															ｲ
BB	—															ｳ
BC	¼															ｴ
BD	:															ｵ
BE	¸															ｶ
BF	≡															ｷ
C0	≺	≺	≺	ⓐ	ⓐ	≺	≺	≺	ã:	õ:	Æ	ä:	ä:	ö	å	ｸ
C1	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	ｹ
C2	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	ｺ
C3	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	ｻ
C4	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	ｼ
C5	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	ｽ
C6	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	ｾ
C7	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	ｿ
C8	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	ﾀ
C9	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	ﾁ
CA	-															ﾂ
CB	ö															ﾃ

Hexadecimal Value	International	United States and Canada	Canada (French)	France	United Kingdom	Spain	Spanish Speaking	Portugal	Brazil	Denmark and Norway	Finland and Sweden	Austria and Germany	Belgium	Italy	Japan (English)	Japan (Katakana)
CC	ö															
CD	ò															
CE	ó															
CF	õ															
D0	)	)	è	è	)	)	)	´	é	à	à	ü	è	è	)	
D1	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J	J
D2	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
D3	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
D4	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
D5	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
D6	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
D7	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
D8	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
D9	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
DA	≥															
DB	û															
DC	ü															
DD	ù															
DE	ú															
DF	ÿ															

Hexadecimal Value	International	United States and Canada	Canada (French)	France	United Kingdom	Spain	Spanish Speaking	Portugal	Brazil	Denmark and Norway	Finland and Sweden	Austria and Germany	Belgium	Italy	Japan (English)	Japan (Katakana)
E0	\	\	¸	£	\	\	\	Ç	\	\	£	Ö	£	£	$	$
E1																
E2	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
E3	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
E4	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
E5	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
E6	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
E7	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
E8	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
E9	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
EA	²															
EB	Ö															
EC	Ü															
ED	ò															
EE	ó															
EF	ö															
F0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
F3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Hexadecimal Value	International	United States and Canada	Canada (French)	France	United Kingdom	Spain	Spanish Speaking	Portugal	Brazil	Denmark and Norway	Finland and Sweden	Austria and Germany	Belgium	Italy	Japan (English)	Japan (Katakana)
F4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
F5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
F6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
F7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
F8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
F9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
FA	³															
FB	Ù															
FC	Ü															
FD	Ù															
FE	Ú															
FF																

The following notes apply only to the International character set.

Notes:

1. Hexadecimal values 8D, AD, BD, and DA will appear on the display screen as solid rectangular blocks.
2. Hexadecimal value B7 will appear on the display screen as ¼.
3. Hexadecimal value B8 will appear on the display screen as ½.
4. Hexadecimal value B9 will appear on the display screen as ¾.
5. Hexadecimal value AF will appear on the display screen as ®.

Glossary

access method: A technique for moving data between main storage and input/output devices.

address: A name, label, or number that identifies a register, location in storage, or any other data source.

addrout file: A record address diskette file produced by the sort program. An addrout file contains the binary relative record numbers of records in a diskette file, and can be used to process the file designated as the input file to the sort program.

addrout sort: A type of sort where the output consists of 4-byte, binary, relative record numbers that indicate the relative positions (1st, 20th, 99th) of records in a data set.

alphabetic characters: Letters and other symbols, excluding digits, used in a language.

alphameric characters: Same as alphabetic characters, with the addition of digits 0 through 9.

alternative cylinder: A cylinder on a diskette that is assigned in place of a cylinder that is defective.

assembler language: A source language that includes symbolic machine language statements in which there is a one-to-one correspondence with the instruction formats and data formats of the computer.

attention: An action that causes the applications microprocessor to interrupt the normal processing flow.

background job: A job that is run in a partition which does not have immediate access to a keyboard/display unit.

basic data exchange: A diskette data exchange that uses 128-byte sectors and allows only one record per sector. The logical record length must be $\leq$ 128 bytes and is unblocked and unspanned. The basic data exchange formats allow you to exchange data between the 5280 and other systems that use the basic data exchange format.

blocking: Combining two or more records into one block.

buffer: Storage or programming that compensates for a difference in rate of flow of data, or time of occurrence of events, when transmitting data from one part of a computer system to another.

CAM: See *communications access method*.

collating sequence: The position each character holds in relation to other characters according to the bit structure.

common area: The first part of main storage that contains the system control area, common functions, global tables (such as ASCII and error recording), and so on. Depending upon the common function option selected, this area can be 6 K, 15 K, or 16 K. This area is not available for user programs.

common function area: An area of storage within the common area that contain routines and data that can be accessed by programs in any of the partitions. These routines must be IBM-supplied programs.

common functions: A set of IBM-supplied routines/programs in the common area that is used by programs executing in any partition.

communications access method (CAM): A 5280 program that provides the necessary link between a communications program and the communications line. It performs functions such as data formatting and link protocol.

compile: (ANSI definition) To prepare a machine language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one machine instruction for each symbolic statement, or both, as well as performing the function of an assembler.

copy: To read data from a source, leaving the source data unchanged, and to write the same data elsewhere in a physical form that may differ from that of the source.

cylinder: The tracks that can be accessed without repositioning the diskette drive access mechanism.

data exchange: The ability to exchange diskettes and the data recorded on a diskette data set with a system or device that is different from the one recording the data.

data processing: Performing a series of planned instructions on information to achieve a desired result.

data recovery: Reconstruction of recorded data that cannot be read.

data set: An organized collection of related data records treated as a unit and existing on a diskette. In other systems, this is sometimes referred to as a file.

data set label: A 128-byte area on the diskette index cylinder that describes a data set.

data set name: The name associated with a data set. The first character must be alphabetic, and the remaining characters can be any combination of alphabetic or numeric characters. Blanks cannot appear between characters in a name.

DE/RPG: Data Entry with RPG subroutines. A 5280 program product that provides a means for writing data entry and application programs for the 5280 system.

device address: Two EBCDIC characters (such as D1) or four hex characters (such as 4C00) used to identify a 5280 input/output device such as a diskette drive, printer, or magnetic stripe reader.

direct access: The ability to obtain data from a storage device directly by key or relative record. Contrast with *sequential access method*.

direct access method: An access method for processing files by specifying the address (record number) or key value of each record to be accessed.

direct by key access method: An access method for processing index data files by specifying the key associated with each record to be accessed. The current key specified need not have any relative sequence with the last key or next key to be specified.

diskette: A permanent storage medium used on the 5280, that is, a thin, flexible magnetic disk permanently sealed in a cover that gives protection.

diskette drive: The mechanism used to read and write diskettes.

execute: To cause an instruction, program, utility, or other machine function to be performed. See *execution*.

execution: 1. The process of carrying out the instructions of a computer program by a processor. 2. The machine logic process that causes an instruction to be executed.

extent: A continuous space on a diskette that is occupied by or reserved for a particular data set. The data set header label contains information such as beginning of extent, end of extent, and end of data.

field: One or more bytes of related information in a record.

field length: The number of positions allowed for a given field, determined by the maximum length of information that will be entered in the field.

foreground job: A job that is run in a partition which has immediate access to the keyboard/display unit.

format: A specific arrangement of information in a record or on a display screen.

hex: Hexadecimal. A number system using 16 symbols: 0-9, A-F each representing 4 bits (one-half byte).

H-type data exchange: A diskette data exchange format that uses 256-byte sectors. It allows only one record per sector. The logical record length must be 256 bytes; it is unblocked and unspanned. The H-type exchange allows you to exchange data between the 5280 and other systems that use the H-type data exchange format.

ID: Identification.

index data set: A data set in which the keys from another data set and their record position within that data set are recorded. When index data sets are used, the following access methods can be used: sequential; direct by relative record number; and direct by key value.

initialization: The process of preparing a diskette for data by writing track and sector control information in the volume label.

input data set: A set of records a program uses as source information.

I-type data exchange: A diskette data exchange format that uses 128, 256, 512, or 1024 byte sectors. All records in a data set must be the same length. All records in the data set are blocked and spanned. The I-type exchange allows you to exchange data between the 5280 and other systems that use I-type data exchange.

IPL: Initial program load.

job: For the 5280, a program and associated data that can be executed in a partition.

key: One or more characters included in a data record that are used to identify or control the use of that data.

key field: The field within a record that identifies that record when the direct access method by key value is used. The key and record location for each record in the data set are stored in the index data set when an index is used.

main storage: 1. General purpose storage of a computer. 2. All storage that can be addressed by programs, from which instructions can be executed, and from which data can be loaded directly into registers.

master data set: A collection of permanent information; for example, a customer address data set, which is often processed along with a transaction data set.

multiprogramming: The concurrent execution of 2 or more programs (up to a maximum of 8) in which each program appears to be the only program in the system. Programs can have exclusive use of data sets and/or system I/O resources or can share them, depending upon application requirements.

multivolume data set: A data set that extends beyond a single diskette.

numeric fields: A field that contains one or more numeric characters. Valid numeric characters are the digits 0-9, + (plus sign), - (minus sign), . (decimal point), blank, and , (comma).

object code: For the 5280, the four-byte instructions from the compiler or assembler that are machine executable. The first byte of the object code contains the operation code.

object program: 1. A set of instructions in machine language (object code). The object program is produced by the compiler from the source program. 2. In the 5280, the executable program produced by the DE/RPG compiler from a set of source statements. The object program can be executed to control the operation of the 5280 system to perform user-designed functions.

output: Data delivered or ready to be delivered from a device or program, usually after some processing.

partition: An area of 5280 storage in which only one program can execute at a time.

physical record: A record whose characteristics depend on the manner or form in which it is stored, retrieved, or moved. A physical record may consist of all or part of a logical record.

program: 1) (noun) A set of sequential instructions that tells the controller where to get input, how to process it, and where to put the results. 2) (verb) To design, write, and test computer programs.

program product: An IBM-written, licensed program for which a monthly charge is made. A program product performs functions related to processing user data.

range check: A data check that verifies that the numeric value of a field is within the upper and lower limits.

record: A collection of related data, treated as a unit.

record length: The number of characters (or bytes) forming a record.

relative record number: A number that specifies the location of a record in relation to the beginning of the data set.

resource allocation table: A table in storage that is used to assign a logical device ID (a name) to a physical device address.

search: To find a record in a batch using search arguments provided by the user.

sector: 1. An area on a diskette track reserved to record a unit of data. 2. The smallest amount of data that can be written to or read from a disk or diskette during a single read or write operation.

sequential access method: An access method in which records are accessed in the order in which they occur in the data set. Contrast with *direct access method*.

sequential by key: A method of data set processing that reads records in the order in which a keyed or indexed data set is arranged.

Sort/Merge Program Product: A program product which consists of three programs. The sort program is used to arrange records (or their relative record numbers) into a desired sequence, according to data contained in one or more specific fields within the records. The merge program is used to combine two sequential sets of records into one. The command data set display program is used to display or print the contents of a sort/merge command data set.

source entry program: A part of the DE/RPG Program Product that assists the user in entering DE/RPG source statements onto a diskette.

source program: A set of instructions that represents a particular job as defined by the programmer. These instructions are written in a programming language, such as DE/RPG.

spanned record: A record that crosses a block boundary.

special character: A character other than a digit, a letter, or #, $, and @. For example, *, +, and % are special characters.

status line: For the 5280, the first line on a display screen. This line provides operational information.

system configuration: A process that specifies the various components and devices that form a particular operating system. System configuration combines user-specified options and parameters with IBM programs to produce a system having the desired form and capacity.

system control programming: IBM-supplied programs that are on a diskette. These programs are included with each 5280 system and allow the operator to configure the system, IPL the system, recover from power failures, and patch IBM-supplied programs.

table: A collection of data in which each item is uniquely identified by its position relative to the other items.

track: A circular path or the surface of a diskette upon which information is magnetically recorded and from which recorded information is read.

transaction data set: A data set that contains records associated with a specific transaction. These records are less permanent information, such as customer orders. Contrast with *master data set*.

transfer: (ANSI Definition) To send data from one place and to receive the data at another place.

verify: To determine whether a transcription of data or other operation has been accomplished accurately.

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)

IBM 5280 Distributed Data System System Concepts (File No. 5280-20) Printed in U.S.A. GA21-9352-1