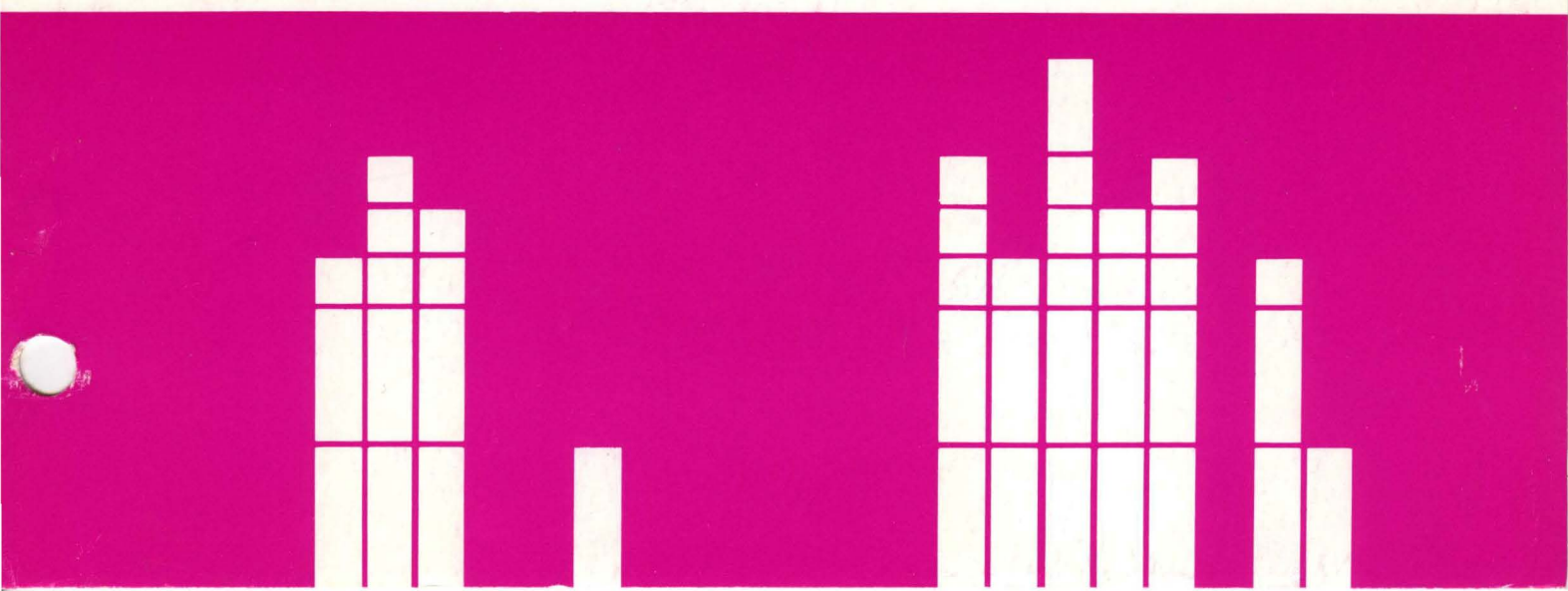
Publication Number
GC31-2067-1

4700 Finance Communication System

Controller Programming Library

Volume 2
Disk and Diskette Programming

IBM



4700 Finance Communication System

Controller Programming Library

Volume 2 Disk and Diskette Programming

Publication Number
GC31-2067-1

File Number
S370/4300/8100/S34-30

Second Edition (January 1984)

This edition applies to Release 3 of the 4700 Finance Communication System and to all subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters (TNLs).

Changes occur often to the information herein; before using this publication to install or operate IBM equipment, consult the latest *IBM System/370 Bibliography of Industry Systems and Application Programs,* GC20–0370, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. Address comments about this manual to IBM Corporation, Information Development, Department 78C, 1001 W.T. Harris Blvd., Charlotte, NC 28257 USA. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This is Volume 2 of the *4700 Controller Programming Library* — one of a set of six volumes for the 4700 programmer. Figure 0-1 on page v summarizes the topics covered in the other volumes. All six volumes are available from your IBM representative or local branch office under a single order number (GBOF–1387).

Volume 2, Disk and Diskette Programming, describes the IBM 4700 Finance Communication System disk and diskette facilities, and tells you how to use them.

Who Should Read This Book

You need this information if your responsibilities include designing, coding, or testing controller application programs that use the 4700 disks or diskettes.

How This Book Is Organized

The first chapter of this manual introduces you to the 4700 disk and diskette. Subsequent sections tell you how to use the facilities to:

- create data sets
- read data sets
- write data sets

This manual can be used both as a guide and as a reference. If you are already familiar with the concepts explained in this manual and simply want to refer quickly to a specific instruction, use the alphabetized descriptions in Chapter 4, "Instruction Descriptions." (There is also an alphabetic index at the back of this book.)

What Else To Read

Before you can use the 4700 disks and diskettes, you must know how to use the other 4700 facilities described in *Volume 1: General Controller Programming,* GC31–2066. Other related publications that you may need to consult are listed in the Bibliography at the end of this manual.

VOLUME 1: GENERAL CONTROLLER PROGRAMMING (GC31-2066)
- Coding Considerations
- Instruction Categories
- Coding Rules
- Instruction Descriptions
- Machine Instruction Formats
- Copy Files
- Error Messages, Program Check Codes, and Status Codes
- Functions Retained for 3600 Compatibility
- Program Communication with the System Monitor

VOLUME 2: DISK AND DISKETTE PROGRAMMING (GC31-2067)
- Basic Disk and Diskette Programming
- Extended Disk and Diskette Access Method
- Instruction Descriptions
- Machine Instruction Formats
- Copy Files
- Status Codes

VOLUME 3: COMMUNICATION PROGRAMMING (GC31-2068)
- SNA/SDLC Communication Programming
- SNA/SDLC Communication Macros
- SNA/SDLC Communication Instructions (Reference)
- BSC3 Host Communication Programming
- BSC3 Communication Instructions (Reference)
- Communication Status Codes
- Communication Parameter List Reference
- Communication Machine Instruction Formats

VOLUME 4: LOOP AND DCA DEVICE PROGRAMMING (GC31-2069)
- General Protocols for Displays
- 4704 and 3604 Displays
- 3270-Compatible Displays and Devices
- 3606 and 3608 Financial Services Terminals
- General Protocols for Printers
- 4710 and 4720 Printers
- 3610, 3611, and 3612 Printers
- 3615 and 3616 Printers
- 3270-Compatible Printers
- 3624 Consumer Transaction Facilities
- Data Stream Mapping (DATSM) Protocols
- Device Status Codes
- Device Parameter List Reference

VOLUME 5: CRYPTOGRAPHIC PROGRAMMING (GC31-2070)
- Cryptographic Concepts and Facilities
- Enciphering and Deciphering Operations
- Generating and Exchanging Cryptographic Keys
- Authenticating Messages
- Validating and Translating PINs
- Using the Encrypting PIN Keypad
- Host Support Encryption Routines (BDKDPRS and BDKDES)
- Cryptographic Programming Instructions (Reference)
- System Cryptography
- Cryptographic Machine Instruction Formats
- Cryptographic Parameter List Reference
- Cryptographic Program Checks and Status Codes

VOLUME 6: CONTROL PROGRAM GENERATION (GC31-2071)
- Overall View of Control Program Generation (CPGEN)
- Sample CPGEN
- CPGEN Macro Statements (Reference)
- Using the Local Configuration Facility (LCF)
- CPGEN Messages
- LCF Messages

Figure 0-1. 4700 Controller Programming Library (GBOF-1387)

Summary of Amendments

Significant changes and additions to this manual are marked with the same change bar that you see at the left of this summary.

This edition reflects the Release 3 change that adds a new LDKT function called Reset Data Set Output Pointer. This document also describes improvements for disk operations when the application programs use LLOAD and APCALL multiple-sector operations.

The coding rules that were Chapter 4 of the previous edition have been deleted. Those coding rules are in *Volume 1: General Controller Programming*.

Contents

Figures

Chapter 1. Introduction to Disk and Diskette Programming

This volume of the Programming Library for the IBM 4700 Finance Communication System describes disk and diskette programming. It contains a brief introduction to 4700 disks and diskettes, and to general concepts of data storage and access.

It also contains the detailed information that you will need to write 4700 Assembler Language application programs using disks and diskettes.

The IBM 4701 Controller Model 1 operates with one-sided diskettes (Diskette 1) or two-sided diskettes (Diskette 2). It can be configured with a diskette expansion unit that contains a second diskette drive.

The IBM 4701 Controller Model 2 operates with two-sided double-density diskette (Diskette 2D) in addition to the one- and two-sided diskettes. The Model 2 can be configured with an expansion unit that provides:

- A second diskette drive
- A second diskette drive and a disk storage drive
- A single disk storage drive
- Two disk storage drives.

If the expansion unit contains one disk storage drive (alone or in combination with a diskette), the disk may have an approximate capacity of either 15 or 30 M bytes. (An M byte equals 1 048 576 bytes.) If the expansion unit contains two disk storage drives, both must be large capacity drives allowing a maximum online disk capacity of 60 M bytes.

The diskette drive that contains the operating diskette during initial load is called the *primary* drive.

Diskette Characteristics

IBM-supplied diskettes are already initialized. Diskettes may also be initialized by a controller using an installation diskette or the System Monitor.

All diskettes must be standard, labeled diskettes as described in the *IBM Diskette General Information Manual*, GA21-9182. More specifically, the 4700 can read and write standard, labeled diskettes on which tracks other than 0, have either 128-byte or 256-byte sectors. A labeled diskette has:

A volume label sector on track 0, side 0, sector 7.

Header label sectors on track 0, side 0, sectors 8 — 26.

Header label sectors, for a two-sided diskette, on track 0, side 1, sectors 1 — 26.

Disk Characteristics

The 4700 disk is a permanently-mounted disk, formatted with 443 cylinders each having either two or four tracks (for the 15- or 30-M byte model, respectively). Each track contains 68 sectors, each sector contains 256 bytes.

The 4700 supports either one or two disk drives in 256-byte sector format only. You can define up to 510 data sets on each disk drive. The header labels for the data sets are found in the 'SYSDSLBL' data set on the disk drive. The individual data sets are located through a hash table data set (named 'SYSDSHSH'). All space allocated on a disk drive is in multiples of sixteen sectors (4096 bytes).

The 4700 provides configuration specifications that can improve disk performance when application programs use LLOAD and APCALL multiple-sector operations. See the FILES and OPTMOD macros in *Volume 6: Control Program Generation* for further information.

Data Set Characteristics

In general, there are 2 types of data set organizations in the 4700. They are referred to as:

- Permanent and Temporary Files.
- EDAM (Extended Disk and Diskette Access Method) data sets.

In this book you will see the terms **permanent** and **temporary**. These terms relate to the retention of data between processing sessions. All 4700 controllers may have a permanent file called SYSPF and always have a temporary file called SYSTF. Permanent and temporary system files are described in Chapter 2.

EDAM data sets offer much flexibility in the ways in which they can be written and read. EDAM also handles exchange data sets that can be processed by systems other than the 4700. EDAM data sets and the ways in which you can use them are described in Chapter 3.

Accessing Data Sets

The 4700 allows you to create and update data sets in 2 ways; sequentially and directly. In general, data sets will be created sequentially and updated in a direct fashion, that is; the third record might be replaced, then the eighth record, and so on.

The word **direct** simply means that the location of the next record to be accessed does *not* depend upon the location of the previously accessed record. Your program can access records in any order.

The 4700 provides *direct* access to data in 2 ways:

1. Relative addressing means that your program can read or replace by record number relative to the beginning of the data set. Without having the location of the beginning of the data set, your program can specify the logical record number to be read or replaced. This means that your program can access the second, fourth, and sixth records, if necessary.

2. Accessing by keys means that you can create data sets; for example, a customer account file; and use the account number or the customer's name as keys to access the appropriate records. The controller will create corresponding data sets containing only pointers (locations of records within your data set) so that your program can access a customer account entry simply by specifying the account number.

Accessing (writing and reading) can also be done sequentially. Sequential access starts with the record located at the beginning of the data set then reads the second, third, fourth, and so on. New records are always written at the end of the data set.

The 4700 also provides absolute diskette addressing which means that your program can access data by diskette side, track, and sector numbers. Another form of absolute addressing allows your program to access any block of data by the physical sector number of any block on a diskette or disk.

The 4700 associates special meanings with the following data set names:

- DUMPAP
- ERRORSET
- SYSAPnnn - where the last three characters are numeric
- SYSBAS
- SYSCPG
- SYSCTL
- SYSDSHSH
- SYSDSLBL
- SYSDSU
- SYSLCF
- SYSOPT
- SYSPF
- SYSSM
- SYSST1
- SYSTF

In this book you will find references to the SMSRPS field that may appear to be similar to the SMSRSN field of earlier books. SMSRSN is a two-byte field that has now been included in a 4-byte field labeled SMSRPS. This change is brought about because of the addressing requirements of the disk drives. Existing application programs that refer to SMSRSN do *not* need to be modified unless you wish to take advantage of the additional capacity of SMSRPS. You are encouraged to use SMSRPS in new application programs so that your programs will not be limited to diskette addressing.

Coding rules and syntax descriptions are found in Volume 1.

Chapter 4, "Instruction Descriptions" on page 4-1 contains the instruction descriptions that you will need to write disk and diskette programs.

Throughout this book the term 'hexadecimal' has been shortened to 'hex' for brevity.

Chapter 2. Basic Disk and Diskette Programming

This chapter discusses basic programming that you can use to access disk and diskette data sets. It is *not* a prerequisite for Chapter 3. You may choose to use only basic instructions to program disk and diskette functions. You will be restricted in the access facilities that you can use.

Basic disk and diskette programming allows access to any data, but only by means of absolute addressing; that is, your program must have track and sector or Physical Block Number (PBN) information. In general, you should not write programs so that they depend upon data at a particular physical location.

Basic programming also provides relative access to the system permanent file (SYSPF) and the system temporary file (SYSTF).

The permanent file on a diskette is retained when you reload the controller's main storage. The temporary file will *not* be retained when you reload unless you specifically request that its contents be saved.

Your system may have a permanent file named SYSPF and always has a temporary file named SYSTF.

- *The permanent file* might contain data that should be retained from day to day, such as a branch bank cash position.

- *The temporary file* might contain data that is not normally retained from day to day, such as a daily audit trail or a teller's cash position.

The permanent file is allocated during the creation of the operating diskette. The permanent file consists of a number of contiguous 256-byte blocks that are read by the LREAD instruction and replaced by the REPLACE instruction.

The temporary file is allocated in groups of 16 blocks called temporary file allocation units (TF Units). Temporary file records are written, read, and replaced in any length up to 252 bytes.

Diskette States

The "state" of operating or operating/test diskettes and data diskettes determines the access functions that can be used, as well as the availability of other diskette functions. The next two sections describe these conditions.

Diskette "Started" State

The only diskette that can be "started"; that is, logically connected to a controller is the operating or diagnostic diskette on the primary drive. It contains the controller load image that is currently in main storage. As long as the diskette is in the logically connected state, the controller application program can read and write temporary and permanent files, load overlay sections, and load transient application programs. System Monitor functions are available and the controller can access diagnostic and restart data on the diskette. Any other diskette that is mounted in place of the current operating or operating/test diskette is logically disconnected from the controller; that is ("stopped"), and the normal diskette functions just noted are not available. If, however, application programs and System Monitor overlays reside on a disk, then transient programs can be called and overlay sections can be loaded.

Diskette "Stopped" State

Except for an absolute read from an operating or operating/test diskette while it is logically connected to the controller, all absolute address operations on diskettes must be done while the diskettes are in the "stopped" state. "Stopped" diskettes are diskettes that are logically disconnected from the controller — overlay program sections *cannot* be loaded, permanent and temporary files *cannot* be accessed, and System Monitor functions are *not* available.

An operating or operating/test diskette, currently mounted on the primary drive, is placed in the "stopped" state by either of two actions by the control operator:

- Issue the stop diskette command (042 1).
- Open and close the diskette-drive door.

The controller application program can prompt the control operator to perform either action.

A data diskette, an operating diskette or an operating/test diskette other than the one used to load the controller, is automatically in the "stopped" state when mounted (as a result of the diskette drive door being opened and closed).

Only the current operating or operating/test diskette can be logically reconnected to the controller. To accomplish this, the control operator must issue the start diskette command (entered as 042 0). An attempt to "start" any other diskette is rejected.

Absolute Addressing

The controller application program can address 128- or 256-byte sectors on diskettes 1, 2, or 2D on either the primary or the secondary drive. You can directly access both operating diskettes (those containing a controller load image) and data diskettes (diskettes to be used for data entry or storage). You can use data diskettes for data transfer between controllers or, in the 128-byte version, between the 4700 and other systems that accept Basic Exchange diskettes.

The controller application program can directly address sectors in either of two ways. Both of these kinds of addressing are called **absolute**.

- For diskette only, you can specify the track number, the sector number, and, for two-sided diskettes, the side. In this case, your program would use the LREAD A, REPLACE A, and REPLACE CR
- REPLACE CR instructions to access the diskette.
- For disk and diskette, you can specify the PBN of the desired block. In this case the application program would use the LREAD PBN and REPLACE PBN instructions to access the data.

For diskette, the PBN is determined by sequentially numbering all blocks starting with the block at track 1, side 0, block 1 as PBN 1, and continuing through track 74. On two-sided diskettes the first block of a track on side 1 is the next PBN after the last block of the same track on side 0. For example, on a diskette 2, PBN 1 is track 1, side 0, sector 1; PBN 15 is track 1, side 0, sector 15; and PBN 16 is track 1, side 1, sector 1. Every block on the diskette except those on track 0 can be addressed by a unique PBN.

Maximum PBNs for disk and diskette are as follows:

```
Diskette 1   (256)      1 110
Diskette 1   (128)      1 924
Diskette 2   (256)      2 220
Diskette 2   (128)      3 848
Diskette 2D  (256)      3 848
Disk (15 Megabyte)     60 247
Disk (30 Megabyte)    120 495
```

Programming Disk Absolute Addressing

You can use absolute addressing for disks only by means of the PBN operand of the LREAD and REPLACE instructions. See Chapter 4, "Instruction Descriptions" on page 4-1 for further information.

Programming Diskette Absolute Addressing

You can use the absolute addressing forms of the LREAD and REPLACE instructions to access diskettes for reading and replacing records. Track and sector information or PBN information can be used to access a diskette. You cannot use the LCHECK and LWRITE instructions in absolute addressing operations.

Several guidelines and rules apply when programming absolute addressing functions:

- An absolute *read* is the only function allowed on an operating or operating/test diskette that is logically connected to the controller. An attempt to replace a sector on a "started" diskette using absolute addressing results in a command reject.

- Diskettes in the "stopped" state may be accessed for both read and replace operations. Data diskettes and operating or operating/test diskettes other than the current diskettes (those used to load the controller for the current session) are always in the "stopped" state when mounted.

- Application programs can use routines contained in overlay sections to process data diskettes or to access operating or operating/test diskettes (other than an absolute read). The overlays must be called into controller storage before the data diskette is mounted or the operating diskette is placed in the "stopped" state. This restriction does not apply when application programs reside on disk.

  Most System Monitor facilities may be resident on the operating or operating/test diskette and not in controller storage. These facilities are not available when a diskette other than the one used to load the controller is mounted, or when the operating or operating/test diskette is in "stopped" state. This restriction does *not* apply when the System Monitor resides on a disk.

Converting Diskette Track and Record Addresses to PBNs

To convert a diskette block address that is given in track, side, sector format, use the following formula:

PBN = (TT-1) x F1 + (H x F2) + RR

where:

TT = track number in decimal

H = side number (0 = side 0, 1 = side 1)

RR = sector number in decimal

F1, F2 = value from following table:

Diskette Type	Sector Size	F1 Value	F2 Value
Diskette 2D	256	52	26
Diskette 2	256	30	15
Diskette 2	128	52	26
Diskette 1	256	15	15
Diskette 1	128	26	26

Examples: Track 24, sector 3 on a Diskette 1, 256-byte sectors.

PBN = (24-1) x 15 + (0 x 15) + 3

PBN = 23 x 15 + 0 + 3

PBN = 348

Track 12, side 1, sector 3 on a Diskette 2, 256-byte sectors.

PBN = (12-1) x 30 + (1 x 15) + 3

PBN = 11 x 30 + 15 + 3

PBN = 348

Converting Diskette PBNs to Track and Record Addresses

To convert a PBN to a track, side, sector address, use the following algorithms:

- For Diskette 1 (1-sided)

```
    (PBN-1)
T = ------- + 1
      MR

Z = (PBN-1) MODULO MR

R = (Z MODULO MR) + 1
```

where:

T = The track number

Z = A temporary work variable used to calculate R

PBN = The physical block number (1 to n)

MODULO = The remainder resulting from the division of the first integer by the second integer

R = The block number

MR = Appropriate maximum of sectors on a track from following table:

Diskette Type	Sector Size	MR Value
Diskette 1	256	15
Diskette 1	128	26

Example:

```
PBN     = 348

Diskette 1, 256-byte sectors

          (348-1)
T       = ------- + 1 = 24 (the track number)
             15

Z       = (348-1) MODULO 15 = 2

R       = (2 MODULO 15) + 1 = 3 (the sector number)

PBN 348 = track 24, sector 3.
```

- For Diskette 2 and 2D (2-sided)

$$T = \frac{(PBN-1)}{MR \times 2} + 1$$

$$Z = (PBN-1) \text{ MODULO } (MR \times 2)$$

$$H = \frac{Z}{MR}$$

$$R = (Z \text{ MODULO } MR) + 1$$

where:

T = The track number

Z = A temporary work variable used in the calculation of H and R

H = The side (0 = side 0, 1 = side 1)

R = The block number

PBN = The physical block number

MODULO = The remainder resulting from the division of the first integer by the second integer

MR = Appropriate maximum number of sectors on a track from the following table:

Diskette Type	Sector Size	MR Value
Diskette 2D	256	26
Diskette 2	256	15
Diskette 2	128	26

Example: PBN = 348

Diskette 2, 256-byte sectors

$$T = \frac{(348-1)}{15 \times 2} + 1 = 12 \text{ (the track number)}$$

$$Z = (348-1) \text{ MODULO } (15 \times 2) = 17$$

$$H = \frac{17}{15} = 1 \text{ (the side)}$$

$$R = (17 \text{ MODULO } 15) + 1 = 3 \text{ (the sector number)}$$

PBN 348 = Track 12, side 1, sector 3

Basic Diskette Programming Considerations

The following two sections describe the general format of diskettes and give information about special diskette records.

Initialized Diskette Formats

The format of track 0 is shown in Figure 2-1 and the formats of initialized one- and two-sided diskettes (128 or 256 byte) are illustrated in Figure 2-2 on page 2-9.

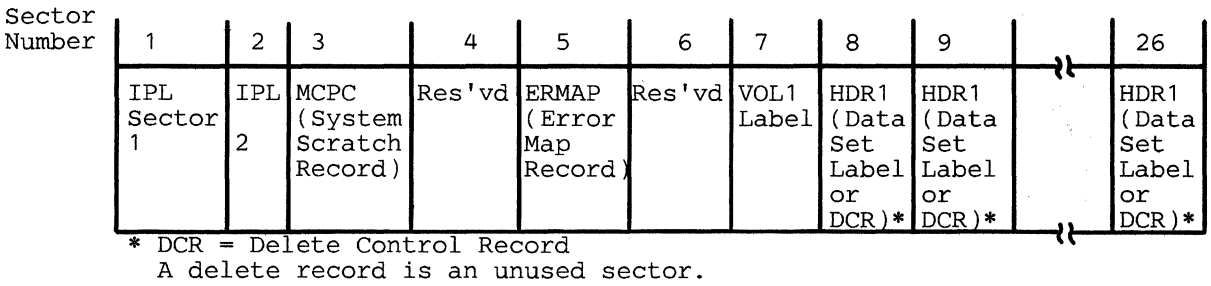


Figure 2-1. Format of Track 0 Sectors

TRACK 0 (LABEL TRACK)

Track 0 contains 26 128-byte records, formatted as follows:

SECTOR 1 (IPL SECTOR 1)

Character Position	Field Name	Length	Contents
0	Label Field	80	X'40's
80	Padding	48	X'00's

SECTOR 2 (IPL RECORD 2)

Same as Record 1.

SECTOR 3 (MCPC RECORD)

Same as Record 1.

SECTOR 4 (RESERVED)

Same as Record 1.

SECTOR 5 (ERROR MAP)

Character Position	Field Name	Length	Contents
0	Label Identifier	5	C'ERMAP'
5	Separator, Defective Track, Reserved, Disk Detect Indicator	18	X'40's
23	Error Directory Indicator	1	C'B'
24	Error Directory	48	X'00's
72	Reserved	8	X'40's
80	Padding	48	X'00's

SECTOR 6 (RESERVED)

Same as Record 1.

SECTOR 7 (VOLUME LABEL)

Character Positon	Field Name	Length	Contents for 256-Byte Diskettes	Contents Basic Exchange Diskettes
0	Label Identifier	3	C'VOL'	C'VOL'
3	Label Number	1	C'1'	C'1'
4	Volume Identifier	6	C'IBMIRD'	C'IBMIRD'
10	Accessibility, Reserved, Owner Identifier, Reserved	65	X'40's	X'40's
75	Physical Record Length	1	C'1'	X'40's
76	Physical Record Sequence Code, Reserved	3	X'40's	X'40's
79	Label Standard Version	1	C'W'	C'W'
80	Padding	48	X'00's	X'00's

SECTORS 8-26 (DATA SET LABELS)

On a newly reformatted diskette, these labels are unused. They are recorded as "delete" control records; i.e., their address marks indicate "control record" rather than "data record". The data portion of each record is as follows:

Character Position	Field Name	Length	Contents
0	Deleted Record Flag	1	C'D'
1	Unused	79	X'40's
80	Padding	48	X'00's

When these data set labels are used, they have the general format shown in *The IBM Diskette General Information Manual.*

TRACKS 1-76

All of these blocks are recorded as delete control records or data records.

Figure 2-2. Format of Initialized Diskettes

Diskette Control Records, Volumes, and Data Sets

If you are using Basic Diskette facilities, then your application program is responsible for all aspects of diskette creation, including:

- Establishing and checking volume labels for multi-volume data sets.
- Establishing and checking data set labels and extents.
- Writing control records.

The three types of control records are:

- Delete Records: These control records indicate an unused block and are indicated by a C'D' (hex C4) as the first character in the data field.
- Sequentially Relocated Records: These control records indicate that the block contains a permanent error and that the data record has been relocated to the next sequential block that does not contain an error. These control records are indicated by a C'F' (hex C6) as the first character of the data field.
- Alternate Relocated Records: These control records indicate that the block contains a permanent error and that the data record has been relocated to an error data set named ERRORSET. These control records are indicated by a C'.' (hex 4B) in the first data byte.

Control records are written by placing the appropriate character in the output segment and issuing the REPLACE instruction with the CR operand. When a control record is read, the type of record is indicated by status in SMSDST and the first character that is placed in the input segment.

Detailed information about the labels on track 0, control records, and relocation of data records is contained in *The IBM Diskette General Information Manual.*

Permanent and Temporary Files

As noted previously, the operating diskette contains space that can be allocated to a permanent and a temporary file:

- *The permanent file.* This file contains data that should be treated as permanent (retained from day to day), such as a 3624 customization image or branch cash position.
- *The temporary file.* This file contains data that is not normally retained from day to day, such as a daily audit trail or teller cash position.

The temporary and permanent files are allocated during the CPGEN process.

- The size of the permanent file is specified in sectors. The data on this file is read and replaced by blocks, at a maximum of 256 bytes per block.

- The space allocated for the temporary file is specified in groups of 16 sectors called temporary file allocation units (TF Units). Don't confuse TF Units with diskette tracks, there is no direct relationship. A TF Unit may begin in the middle of one track and end on the next track. Fifteen sectors of each TF Unit are available to the controller application program. The first sector of each TF Unit is used by the controller for indexes. Temporary file records are written, read, and replaced in any length up to 252 bytes.

The temporary file and permanent file are shared resources. Application programs must therefore be written so that records on these files are read, modified, and replaced by one logical work station at a time.

The Permanent File

The permanent file is created during the creation of the operating diskette. It is never erased or reallocated unless the diskette is used for creating a new operating diskette. The permanent file consists of a number of contiguous 256-byte sectors that are read using the LREAD P instruction and replaced using the REPLACE P instruction. LWRITE and LCHECK are not used for data transmission to the permanent file.

Permanent file sectors are addressed by relative block number; the first sector in the file has the number 1. When the controller addresses a sector, the location of the sector is determined from a known starting location for the file and the number of sectors on each track.

Whenever data transmission occurs between a logical work station and the permanent file, the station issuing the instruction is placed in a wait state. When an LREAD is completed, the wait state ends when the data is available in the station's segment. When a REPLACE is performed, the wait state ends when the block has been written on the diskette. If an error condition occurs that prevents the operation from completing, the wait state ends when status bits are stored in SMSDST.

Allocating the Permanent File

The number of sectors in the permanent file is specified during controller configuration using the PF operand of the FILES macro. For example:

```
FILES   PF=50
```

specifies that a permanent file consisting of fifty 256-byte sectors is required. When the operating diskette is created, the specified number of blocks are allocated and set to binary zeros.

Processing Permanent File Blocks

The information you need to code permanent file read and replace operations is in Chapter 4, "Instruction Descriptions" on page 4-1under the LREAD P and REPLACE P instructions.

Permanent File Errors

If any error, except wrong-length record, is encountered when reading or replacing permanent file records, the requested operation is not performed. If the status bits indicate unit exception, the record number set in SMSRPS by the application program was either less than 1 or greater than the number of records in the permanent file. The controller sets the highest valid permanent file record number in SMSRPS before returning control to the logical work station. Refer to the appropriate appendix for additional information on possible error conditions and associated status bits.

The Temporary File

The temporary file consists of either:

- the space remaining on the operating diskette after all other diskette allocations have been made, or
- the number of 16-sector TF Units specified on the FILES configuration macro, whichever is less.

The temporary file is reused each processing session; records already on the file are retained if the control operator specifies a warm start.

The temporary file consists of TF Units that are sixteen 256-byte sectors. The first sector of each TF Unit is an index and the other 15 sectors are used for data records. Each sector carries a 'session ID' to identify the session during which the records in the sector were created. (The session ID increases by 1 whenever a cold-start controller load is performed.) The temporary file is designed to handle variable-length records of up to 252 bytes. The controller gathers these records into a buffer. When the buffer is full or when a record to be written is larger than the space remaining in the buffer, the entire sector is written to the diskette. As each record is placed in the buffer, a 2-byte prefix is added. When the record is read, the prefix is removed before the record is placed in the segment. The TF Units and data block formats are shown in Figure 2-3 on page 2-13.

Allocating the Temporary File

The number of temporary file subdivisions to be created is specified during controller configuration using the TF operand of the FILES macro. You can specify up to 4 subdivisions. The application program refers to these subdivisions as TF1, TF2, TF3, and TF4. For example:

```
FILES   TF=(2,10)
```

specifies that two temporary file subdivisions are used by the controller application programs; these subdivisions are named TF1 and TF2 and are allocated 10 TF Units. In addition to the major subdivisions, you can specify that the temporary file have as many as 60 subfiles.

If an application program attempts to access a temporary file that has not been created, status (SMSDST) is set to indicate command reject because an invalid request was made.

The controller maintains sets of counters for each temporary file subdivision and for each TF Unit. The controller uses these counters to return a sequence number to the application program when an LWRITE is performed. The sequence number will be one of the following:

- a file sequence number in SMSFSN for TF1 through TF4
- a subfile sequence number in SMSSFW
- a composite sequence number in SMSCSN.

You can retrieve a temporary file data record by specifying the file ID and the file sequence number; subfile sequence number; or composite file sequence number. The controller determines the TF Unit that contains the record and uses the TF Unit counters to determine the exact location of the record.

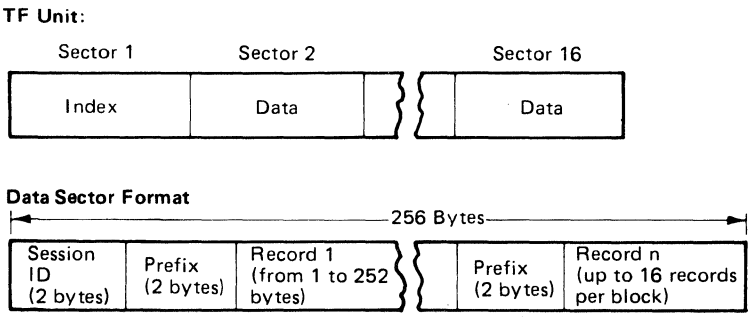


Figure 2-3. Format of the TF Unit

Initializing the Temporary File

The temporary file is initialized in one of three ways:

- During the controller startup procedure performed by the control operator, the operator may respond to the startup message (00001) with a 1 or 8 to indicate that a cold start should be performed.
- The SETDSKT instruction may be issued by the controller application program. One of the functions of SETDSKT is to initialize the temporary file.
- The control operator can use the System Monitor command 063 01 to reset the temporary file.

SETDSKT is normally used after an end-of-file condition has been encountered during a write to a temporary file. At this time, the temporary file write buffer will not contain any data, and any read operation will have been completed so that the temporary file read buffer may be initialized. If SETDSKT is used before an end-of-file is encountered, the temporary file write buffer may contain data that has not been written on the diskette. It is the application program's responsibility to ensure that records in the temporary file write buffer, if those records are required, are saved before SETDSKT is issued. You may use the LCHECK DSK instruction for this purpose. Temporary file read operations will always have been completed before a temporary file is initialized.

Protecting the Primary Diskette

If the door of the diskette drive is inadvertently opened while data is being written to the diskette, a write-error may occur. To prevent this kind of error from occurring during an orderly shutdown of the system, an application program can issue the SETDSKT instruction (with a value of hex '40'), to inhibit access to the diskette on the primary drive. The SETDSKT instruction must be issued before the operator removes the operating diskette. (SETDSKT with a value of hex '20' restores access to the primary drive.) When the drive has been made inaccessible, neither the system nor another application program can write to the diskette and the diskette can be safely removed.

Writing Temporary File Records

The LWRITE instruction is used to write records to a temporary file. The instruction specifies the file number and the location of the record to be written. For example:

```
LWRITE   TF1,3        WRITE THE RECORD FROM SEGMENT 3
BRAN     ST,ERROR     BRANCH IF STATUS RETURNED
```

The record defined by the PFP and SFP of Segment 3 is moved to the temporary file buffer and flagged as a member of temporary file 1. The controller returns the file sequence number in Segment 1 at SMSFSN. The application program could transmit this number to the teller so that he or she could retrieve the record at some later time.

If the field defined by the PFP and SFP of the specified segment is longer than 252 bytes, the first 252 bytes are moved to the temporary file buffer and written on the diskette. A status code is then returned in SMSDST indicating a wrong-length condition because the record is too large.

Reading Temporary File Records

The LREAD instruction is used to read records from a temporary file. Before issuing an LREAD, the controller application program must set the following fields:

- SMSRPS: to the file sequence number (a binary number)
- the PFP of the segment receiving the data: to the beginning of the input area
- the FLI of the segment receiving the data: to the number of bytes of data to be read.

If the FLI is 0, the amount of data read is determined by the space between the PFP and the end of the segment. For example:

```
INPUTSEG    EQUATE    4
RECNO       DEFLD     KBDSEG,5,4          SEQUENCE NUMBER FROM
                                          KEYBOARD
            SETFPL    INPUTSEG,0,0        SETS PFP and FLI of INPUT
                                          AREA
            LDFLDC    WORKREG,RECNO       CONVERTS SEQ NUMBER TO
                                          BINARY
            STFLD     WORKREG,SMSRPS      SET READ SEQ NO FIELD
            LREAD     TF2,INPUTSEG        READS RECORD FROM TEMP
                                          FILE2
            JUMP      ST,ERROR            BRANCHES IF STATUS STORED
```

A record of temporary file 2 is read into Segment 4. The file sequence number of the record was obtained as input from a keyboard. The entire record is read unless Segment 4 is shorter than the record. If Segment 4 is shorter, as much of the record as possible is placed in the segment, and a status code is returned in SMSDST indicating a wrong-length condition because the record is too long. SMSIML is always set to indicate the amount of data actually read into the segment.

The controller tables contain a temporary file read buffer as well as the temporary file write buffer. If the record to be read is located in either of these buffers, the record is moved from the buffer to the logical work station's segment. If the diskette must be read, a temporary file block is brought into the temporary file read buffer, and then the record is moved from the buffer to the logical work station's segment.

The Optional Read Index Buffer

The controller uses the TF Unit index to locate all temporary file records. The TF index for the current TF Unit is in the controller tables. All indexes for filled TF Units are on the diskette as the first sector in each unit. The controller tables normally contain only one temporary file read buffer that is used for TF index and data records. To read a record not in the current TF Unit, the applicable TF Unit index is read into the temporary file read buffer, the record is located, and then the block containing the record is read into the buffer overlaying the index. Because the data block overlays the index, the index must be reread for each data record read. This procedure requires that the diskette be accessed twice for each temporary file read request.

If the temporary file is read sequentially, greater efficiency can be obtained by specifying that the optional 256-byte read index buffer be included in the controller. This buffer allows the controller to read the index into a buffer other than the one used for data. If a subsequent read requires an index already in storage, the controller does not read the index from the diskette. The read index buffer is allocated during controller configuration by specifying the BUF operand of the FILES configuration macro instruction. For example:

```
FILES    TF=(2,10),BUF=Y
```

Each record in the temporary file may be flagged as a member of a subfile. You can do this by placing a 1-byte binary subfile number at SMSSFW before issuing the LWRITE instruction. The subfile number is stored in the record prefix as the record is moved to the temporary file buffer. The subfile sequence number associated with the record is returned in SMSSSN.

For example, temporary file 1 is used to journal transactions when the link to the host processor is inoperative. Each operator has been assigned a subfile, and the subfile number is stored in Segment 0 at OPFILE. Refer to Figure 2-4 on page 2-17.

Each temporary file subdivision can have as many as 60 subfiles. When the LWRITE is issued, the controller examines SMSSFW for a valid subfile number. A 0 indicates no subfile; a number from 1 to 60 causes the record to be flagged as a member of the corresponding subfile. Any other value causes the controller to return a status code indicating that the LWRITE is rejected because an invalid request was made; the record is not stored on the file.

Specifying Subfile Indexes: Subfile indexes may be created during controller configuration so that records can be retrieved by subfile as well as by file. When a record is written to a subfile, the file sequence number is returned in SMSFSN and the subfile sequence number is returned in SMSSSN. An additional set of counters is created for each subfile index specified. For example:

```
FILES   TF=(2,10),INDX1=(1-5)
```

specifies that five sets of subfile counters be created for temporary file 1. This allows subfiles 1 through 5 to be indexed and a subfile sequence number to be returned when a record is written to a subfile. Records can be written to other subfiles, but only file sequence numbers are returned for those records. A record written to temporary file 1, subfile 1, can now be retrieved as either record n of temporary file 1 or as record m of temporary file 1, subfile 1 (where *n* is the file sequence number returned in SMSFSN and *m* is the subfile sequence number returned in SMSSSN when the record is written).

This feature might be used if journaling is done on the diskette rather than on a printer. For example, if all transactions entered are journaled to temporary file 1 and each record is flagged as a member of the subfile that corresponds to the station number, then all records entered at a station can be retrieved by specifying the file and subfile and sequentially reading the records. All records written on temporary file 1 may be retrieved by specifying a subfile number of 0.

```
OPFILE      DEFLD     SEG0,REGEND,1                1

            MVFXD     SMSSFW,OPFILE                2
            LWRITE    TF1,OUTSEG                   3
            BRAN      ST,DSKERR                    4
            LCHECK    DSK                          5
            BRAN      ST,DSKERR                    6
```

1 Defines the subfile number in Segment 0
2 Moves the subfile number into Segment 1
3 Writes the record to a temporary file
4 Branches to an error routine if status was stored
5 Forces a physical write operation to ensure that the record is in the file
6 Branches to an error routine if status was stored.

Figure 2-4. Writing a Temporary File Record Using Subfile Numbers

Reading Records in a Subfile: To retrieve a temporary file record by subfile when the subfile is indexed, the controller application program must first set the following fields:

- SMSSFR: to the subfile number
- SMSRPS: to the subfile sequence number
- the PFP of the segment receiving the data: to the beginning of the input area
- the FLI of the segment receiving the data: to the number of bytes of data to be read.

The LREAD instruction, specifying the temporary file and input segment, is then issued. The controller examines the subfile number and then checks for a corresponding counter; if the counter does not exist (that is, if indexing of that subfile was not specified during controller configuration), a status code is returned in SMSDST indicating that the command is rejected because an invalid request was made.

For example, temporary file 1 is used to journal all credit transactions. Figure 2-5 on page 2-18 shows how TF1 would be read to obtain a listing of these records for a subfile. (The subfile number is obtained from the station number field in segment 1.) The subfile number could be converted to printable form and appended to the record so that the teller who entered the transaction can be identified.

```
BINONE      DEFCON    X'01'                    1
SUBFLNO     DEFCON    X'03'                    2
TRANSFLD    DEFLD     JOURNSEG,,252            3

TFREAD      LDFLD     WORKREG1,BINONE          4
            MVFXD     SMSSFR,SUBFLNO           5
RTRN        STFLD     WORKREG1,SMSRPS          6
            SETFPL    TRANSFLD                 7
            LREAD     TF1,JOURNSEG             8
            BRANL     ST,ERR                   9
            BRANL     PRNTWRT                  10
            ADDFLD    WORKREG1,BINONE          11
            BRAN      RTRN                     12
```

1 Generates a binary 1
2 Defines the subfile number
3 Defines the field to contain the temporary file record
4 Initializes WORKREG1 to a binary 1. WORKREG1 is used as the subfile sequence number, and increases to point to sequential records
5 Sets the read subfile field to 3
6 Stores the subfile sequence number in the read sequence number field
7 Sets the PFP and FLI of the input segment
8 Reads record 1 of temporary file 1, subfile 3
9 Checks for status including no-record-found (The last record was read.)
10 Branches-and-links to a routine to print the record
11 Increases the subfile sequence number by 1
12 Branches to RTRN to read the next record.

Figure 2-5. Reading Temporary File Records Using Subfile Numbers

Composite File Indexing

Composite file indexing is a feature that aids in maintaining a journal if records are written to more than one file. A composite file index is essentially a composite of the subfile indexes for a set of temporary files. A composite index is specified during controller configuration using the COMF and INDXC operands of the FILES configuration macro. For example:

```
FILES   TF=(3,20),COMF=(1,3),INDXC=(1-10)
```

specifies that TF1, TF2, and TF3 be created, that the composite file consist of TF1 and TF3, and that composite file indexes be created for subfiles 1 through 10.

When an LWRITE instruction is issued for a temporary file that is a member of the composite file, the controller examines SMSSFW to determine if the subfile is one of those for which indexing was specified for the composite file. If it is, the controller returns a composite file sequence number in addition to the file sequence number. A subfile sequence number is also returned if a subfile index was specified. The record can later be retrieved as record *n* of a temporary file or record *x* of that subfile of the composite file (where *n* is the file sequence number returned in SMSFSN and *x* is the composite file sequence number returned in SMSCSN when the record is written). The record can also be retrieved as record *m* of a subfile (where *m* is the subfile sequence number returned in SMSSSN) if subfile was specified. If SMSSFW is 0, composite file indexing is not done.

Reading Composite File Records: To read a record written to a file and subfile that are part of the composite file, the controller application program must first set the following fields:

- SMSSFR: to the subfile number
- SMSRPS: to the composite file sequence number
- the PFP of the segment receiving the data: to the beginning of the input area
- the FLI of the segment receiving the data: to the number of bytes to be read.

When the LREAD instruction is issued that refers to the composite file, the controller examines the value in SMSSFR for a valid subfile number; if SMSSFR is 0 or contains a value not specified in the INDXC operand of the FILES macro, a status code is returned in SMSDST indicating that the command is rejected because an invalid request was made.

The following is an example of temporary file indexing. This FILES macro was used:

```
FILES  TF=(3,45),INDX1=(1-5),COMF=(1,3),INDXC=(5-10)
```

Write To:		*Sequence Numbers Returned:*		
File	**Subfile**	**File Sequence Number**	**Subfile Sequence Number**	**Composite File Sequence Number**
1	0	1	0	0
1	2	2	1	0
1	6	3	1	1
1	1	4	0	1
2	5	1	0	0
2		2	0	0
3		1	0	0
3		2	0	2

Figure 2-6. Temporary File Indexing

The first record may be read only as record 1 of TF1. The subfile was specified as 0, so a subfile sequence number does not exist. The third record may be read as record 3 of TF1 or record 1 of TF1, subfile 5, or as record 1 of subfile 5 in the composite file. The last record may be read as record 2 of TF3 or as record 2 of subfile 5 in the composite file. Other records may be read as shown.

Figure 2-7 on page 2-20 is an example of reading a composite file. Temporary file 1 contains records sent to the host processor, temporary file 3 contains records being saved for transmission after normal business hours, and the composite file is made up of both these files. This figure shows how the composite file is read to obtain a journal of all transactions entered by a teller.

```
BINONE      DEFCON    X'01'                    1
TELLID      DEFLD     OPSEG,IDDISP,1           2
TRANSFLD    DEFLD     JOURNSEG,1,252           3

COMPREAD    LDFLD     WORKREG1,BINONE          4
            MVFXD     SMSSFR,TELLID            5
RTRN1       STFLD     WORKREG1,SMSRPS          6
            SETFPL    TRANSFLD                 7
            LREAD     C,JOURNSEG               8
            BRANL     ST,ERR                   9
            BRANL     PRNTWRT                  10
            ADDFLD    WORKREG1,BINONE          11
            BRAN      RTRN1                    12
```

1 Generates a binary 1
2 Defines the field containing the binary teller ID
3 Defines the field to contain the temporary file record
4 Initializes WORKREG1 to a binary 1. WORKREG1 is used as the composite file sequence number, and increases to point to sequential records
5 Sets the read subfile field with the teller ID (subfile sequence number)
6 Stores the composite file sequence number in the read sequence number field
7 Sets the PFP and FLI of the input segment
8 Reads the record
9 Checks for status including no-record-found (The last record was read.)
10 Branches-and-links to a routine to print the record
11 Increases the composite file sequence number by 1
12 Branches to RTRN1 to read the next record.

Figure 2-7. Reading Composite File Records

The System Log

The system log is a special kind of temporary file. It is used by the controller to record messages concerning the status of the system. The controller application program can also write records to the log and may retrieve all records on the log. Log records share space with other records written to the temporary file. The log is always present and is not specified during controller configuration. If a temporary file is not specified, one TF Unit on the operating diskette is automatically allocated for log records. Both the System Monitor and your application program can read the log. The Communications Network Management/Controller Support (CNM/CS) facility in some versions of the System Monitor reads log messages and sends alert messages to the Network Problem Determination Analysis (NPDA) facility at the host system.

Writing Records to the Log: The LWRITE instruction is used to write records to the log. The instruction specifies the log and the location of the record to be written; the value in SMSSFW is ignored. Each record written is assigned a file number that is returned in SMSFSN. Both SMSSSN and SMSCSN are set to 0.

All records written to the log by the controller are time stamped. Records written to the log by the application program (user) will be time stamped only if specified during CPGEN (by specifying the LOGTM parameter).

All log messages have a system byte and an alert byte as the first two bytes. The system byte for controller log messages will always be a hex F1. For the application program that requests time stamps on user log messages, the system and alert bytes will be a duplication of the first two bytes in the user text; however, if a hex F1 is in the first byte, a hex F0 is used as the system byte and only the second byte (alert byte) is duplicated. This elimination of the hex F1 will prevent the application program from writing log messages that resemble controller log messages.

For the application program that does not request time stamps on user log messages, the first two bytes of the record just written also represent the system and alert bytes, but no duplication is required. Again, if a hex F1 is in the first byte, it is replaced by a hex F0 and the alert byte is unaffected.

When any record is written to the log, the controller examines the second byte for a hex F1. If the character is hex F1, the controller sets the flag (bit 7 in the GMSIND field) on and (if specified during the controller configuration process) lights the designated CHECK light on the control operator's console. This light is switched off by any read issued to the log by the System Monitor. The flag must be switched off by the controller application program. A read by the CNM/CS facility switches off both the light and flag.

You can also issue log messages from your program to NPDA by writing messages with number 101 to the log. CNM/CS recognizes these messages as NPDA-directed, and issues alerts to NPDA.

The format of the 101 log message that you must provide to the LWRITE instruction is:

```
01 101 token data
```

where: "token" is a 1 to 8-byte field and "data" is up to 240 bytes in length. (The first 100 bytes of "data" will be sent to NPDA.)

This log message should be time stamped. (Time stamping is an option of the FILES macro in your configuration specifications.) If you choose to time stamp, then this message will appear in the log as follows:

```
01 hhmm 01 101 token data
```

where: "hh" = hours and "mm" = minutes.

If you choose not to time stamp, then you must provide an extra 8 bytes, in place of "01 hhmm ", at the beginning of the message.

All messages written to the log cause the contents of the temporary file buffer to be written to the diskette. If the buffer is only partially filled, the buffer remains available for any additional records written to a temporary file.

A user written log record may contain both EBCDIC and hex data. The hex data must begin with a hex 'FF'. All data after the 'FF' will be considered to be hex.

When the log records are displayed on the terminal using the system monitor commands, only the EBCDIC data will be displayed unless the 'X' option is chosen; then the entire log record will be displayed in hex.

Reading Records from the Log: The LREAD instruction is used to read records from the log. Before issuing the LREAD, the controller application program must place the file sequence number in Segment 1 at SMSRPS. SMSSFR is not used.

The CNM/CS facility reads the log periodically and records any messages that have set the CHECK indicator. The CNM/CS reformats these messages, and passes them to the Network Problem Determination Facility (NPDA) at the host. To prevent duplication of the messages transmitted, CNM/CS maintains an index of messages already read. A cold start or issuing of the SETDSKT instruction resets the index, causing log messages to be lost and may cause them not to be reported to the host as alert messages.

Replacing Temporary File Records

The REPLACE instruction is used to replace records written to a temporary file. Before the REPLACE instruction is issued, the controller application program must set SMSSFR and SMSRPS. The record to be replaced can be identified, as with LREAD, by file and file sequence number; by file, subfile and subfile sequence number; or by composite index, subfile and composite file sequence number. If the record is not exactly the same length as the original record, a status code is returned in SMSDST, and the replacement takes place as follows:

- If the new record is shorter than the old record, the leftmost part of the original record is replaced, a status code is returned to indicate a wrong-length record, and the residual length (how much of the original record was not modified) is returned in SMSIML.

- If the new record is longer than the original record, the leftmost portion of the new record replaces the original record, a status code is returned indicating a wrong-length record and the excess length of the new record is stored in SMSIML. For example, to replace the eighth record in temporary file 1 with the record defined by the secondary and primary field pointers of Segment 3, the following instructions could be used (assuming that register 1 contains a binary value of 8):

```
STFLD       1,SMSRPS     STORE RECORD NUMBER
SUBREG      1,1          CLEAR REGISTER TO ZERO
STFLD       1,SMSSFR     ZERO SUBFILE, USE FILE SEQ NO
REPLACE     TF1,3        REPLACE RECORD FROM SEGMENT 3
JUMP        ST,ERROR     JUMP IF ANY STATUS REPORTED
```

Temporary File Errors

For LWRITE, LREAD, and REPLACE, if status returned is for errors other than incorrect length, the requested operation is not performed.

For LWRITE and LCHECK, if the status is unit exception, the last block written filled the diskette's temporary file area (this status is always presented with prior operation). Records cannot be written to a temporary file of a diskette with a full temporary file area until a cold start is performed with the diskette. Records may still be read and replaced. Before the cold start is performed, any data on a temporary file that is required by the financial institution should be journaled or transmitted to the central processor. If data cannot be immediately journaled or transmitted, another diskette may be loaded and operations restarted. Loading the second diskette requires a controller loading operation which destroys any information in controller storage.

For LREAD and REPLACE, if unit exception is indicated, the sequence number that was present in SMSRPS was either less than 1 or greater than the highest record sequence number assigned from the index just searched. In this case, the controller replaces the original value of SMSRPS with the last sequence number assigned from that index.

The application program can use this information. For example, to read the last record written to the log when its sequence number is unknown, the following instructions could be used:

```
SETFPL    3,0,0        INITIALIZE PFP AND FLI OF INPUT SEG
SUBREG    1,1          GET ZERO VALUE
STFLD     1,SMSRPS     STORE ZERO SEQUENCE NUMBER
LREAD     L,3          REPLACE SMSRPS WITH LAST LOG SEQ NO
LREAD     L,3          READ LAST LOG RECORD INTO SEG 3
JUMP      ST,ERROR     JUMP IF ANY STATUS REPORTED
```

Refer to the Appendix D, "Status Codes for Disk and Diskette" on page D-1 for more detailed status information.

Storage Requirements

The TF Unit counters used in retrieving records of a temporary file occupy storage in the controller table. The amount of space depends on the number of TF Units used for the temporary file and the number of different indexes requested. One set of counters is always created for the log and for each temporary file subdivision specified in the TF operand of the FILES macro. One additional set of counters is created for each subfile specified in any of the INDXn or INDXC operands.

The number of TF Units to be used for the temporary file can be limited by your institution if it does not need the maximum number available and if it wishes to minimize the storage required by the index counters. You can accomplish this by specifying the number of TF Units to be used as the second element of the TF operand. For example:

```
FILES  STF=(2,20),INDX1=(1-5),INDX2=(1-5)
       COMF=(1,2),INDXC(1-5)
```

This FILES macro creates two sets of file counters and 15 sets of subfile counters, each capable of accounting for the records written in a 20-TF Unit area. When the controller is started up using the operating diskette created from the controller configuration procedure containing this specification, only the first 20 TF Units of the available temporary file space are allocated. An attempt by the application program to write more records than can be stored on this number of TF Units results in an error condition, with unit exception returned in SMSDST.

If a greater number of TF Units is specified at controller configuration than is available on the diskette, counters will occupy controller storage, but will be unused. In this instance, unit exception status will result when all of the available TF Units have been used on a diskette.

Data Integrity

Because temporary file records are held in controller storage until a full buffer has been accumulated, some data can be lost when a session is abnormally terminated. When the system is restarted following such an occurrence, a warm start can be requested by the control operator. This will cause the temporary file counters in storage to be reconstructed, but any records in the buffer that had not been written to the diskette are lost.

To minimize the risk of loss, the application program can write a partially filled buffer at periodic intervals by using the LCHECK DSK instruction. To cause a write each time a record is placed in the buffer, however, could seriously degrade performance. This function should be requested only when necessary, and should be designed to minimize, rather than to prevent, loss.

At the end of an operating session, the controller's temporary file buffer should be caused to be written out, so that if a warm start is later required to retrieve records from the current session, all records will be available on the diskette. This can be done by the application program by using either the LCHECK DSK instruction or an LWRITE instruction addressing the log. Also, the control operator can log on and cause a log record to be written.

LCHECK DSK Operation

The controller keeps track of whether its temporary file buffer has been changed since the last time it was written to the diskette. The buffer can be changed either by the addition of a new record using LWRITE or by the replacement of a previously written record using REPLACE.

When the LCHECK DSK instruction is issued, the controller notes whether the buffer has been changed since the last physical write. If so, the current contents of the buffer are written out and the status of the operation is returned to the station issuing the LCHECK. The partially filled buffer remains available for the addition of further records from subsequent LWRITE instructions. If the buffer has not been changed, the write operation does not occur.

The controller maintains a count of blocks written in the temporary file area. This count is available to the application program in Segment 15 at GMSBSN. It is updated each time a full block is written out. The count can be used by the application to determine whether to issue an LCHECK instruction. For example, to ensure that not more than one record per station is lost in the event of an abnormal termination, the application program can save the block number after each LWRITE in an area of station storage. Following the next LWRITE, it can compare the saved block number to the current block number and determine whether it has two data records in the same block. If so, it can issue LCHECK DSK to guarantee that the block is written. The following application instructions could be used for this purpose:

```
SAVE    DEFLD    WORK,,2          DEFINE A SAVE AREA

        LWRITE   TF1,3            WRITE A RECORD FROM SEGMENT 3
        JUMP     ST,ERROR         JUMP IF ANY STATUS REPORTED
        CCFXD    GMSBSN,SAVE      CURRENT BLOCK VERSUS LAST
        JUMP     NE,SKIP          JUMP IF DIFFERENT
        LCHECK   DSK              SAME,WRITE CURRENT BLOCK
        JUMP     ST,ERROR         JUMP IF ANY STATUS REPORTED
        JUMP     NEXT             CONTINUE PROCESSING
SKIP    MVFXD    SAVE,GMSBSN      SAVE LATEST BLOCK NUMBER
NEXT
```

Obtaining Sequence Numbers

You can obtain the highest sequence number for any of the temporary file subdivisions; the composite file; or any of the subfiles by setting SMSRPS to 0 and issuing an LREAD for the file.

The highest sequence number for any of the temporary file subdivisions the composite file, or any of the subfiles can be obtained by setting SMSRPS to 0 and issuing an LREAD for the file in question. This technique causes status to be stored and the highest sequence number to be placed in SMSRPS.

This procedure can be used when you wish to read all records in a file or subfile. For example, the teller may wish to retrieve all records from a subfile to check a previously processed transaction. Figure 2-8 shows the routine used to read a temporary file and write the records to the central processor. REG1 contains a binary 1; the subfile sequence number is set to 0. The input area has already been defined; the input segment is named DISKIN.

```
READDSK   EXOR     SMSRPS,SMSRPS        1
          LREAD    TF1,DISKIN           2
          LDFLD    REG3,SMSRPS          3
RTRN      BRAN     ZO,FINISH            4
          LREAD    TF1,DISKIN           5
          BRAN     ST,DSKERR            6
          BRANL    PRNTWRT              7
          SUBREG   REG3,REG1            8
          STFLD    REG3,SMSRPS          9
          JUMP     RTRN                 10
```

1 Sets the subfile sequence number to 0
2 Obtains the highest sequence number. Status is ignored
3 Loads the subfile sequence number into register 3
4 If 0, indicating that there are no records in the file, branches to a clean-up routine (including transmission of the last record)
5 Reads the record
6 Checks for status information
7 Branches-and-links to a routine that prints or displays the record for the teller
8 Subtracts 1 from the subfile sequence number
9 Stores the result of the subtraction in SMSRPS
10 Branches to step 4 to process the next record.

Figure 2-8. Reading Temporary File Records Starting with the Latest Record Entered

Available Data Set Space

The number of available (unwritten) sectors (SMSADS) in the temporary file is updated after each LWRITE, LREAD, REPLACE, and LCHECK instruction. If the operation is unsuccessful, then the data in the SMSADS field is meaningless.

Diskette Special Multiple-Sector I/O Considerations

When READ and REPLACE instructions are used for multiple-sector operations on diskette, there are no restrictions against crossing track boundaries. However, the controller performs all required operations on one track before initiating a seek to the next track. There are some efficiency implications in this, as discussed below.

When performing a multiple-sector replace operation, the controller writes blocks to the diskette in nonsequential order, using a "skip factor" of 4. If block numbers 1-15 are to be replaced, four diskette revolutions are required, as follows:

Revolution	Blocks Replaced
1	1,5,9,13,...
2	2,6,10,14,...
3	3,7,11,15,...
4	4,8,12,...

When performing a multiple-sector read operation, the controller reads blocks from the diskette in nonsequential order, using a skip factor of 3. If block numbers 1-15 are to be read, three diskette revolutions are required, as follows:

Revolution	Blocks Read
1	1,4,7,10,13,...
2	2,5,8,11,14,...
3	3,6,9,12,15,...

Thus, a full track can be read in three diskette revolutions, or written in seven revolutions (four for writing, three for read-back checking).

A "skip factor" is used in conjunction with the physical record sequence code described under the FORMDKT instruction in Chapter 4, "Instruction Descriptions" on page 4-1. If a 2D diskette is formatted with a hex 0 or 1 value specified for the physical record sequence number, then the multiple-sector operation uses a skip factor as described above. If the 2D diskette is formatted with any physical record sequence number between hex 2 and hex D, the multiple-sector operation uses a skip factor of 1 and the physical sequence of sector assignments on the diskette determines the number of rotations needed to complete the operation.

If more than a full track of data is to be written, an advantage may be gained by synchronizing output with track boundaries. The following example applies to a Diskette 1 or 2, but you can use the same programming technique on a Diskette 2D. If 22 blocks of 256 bytes are to be written starting at block number 9 of one track and continuing through block number 15 of the next track, this can be accomplished in several ways:

- With a 5632-byte buffer, the operation can be performed by the issuing of a single REPLACE instruction. The controller completes the request in 14 diskette revolutions, seven per track.

- With a 3840-byte buffer, the operation can be performed by the issuing of two REPLACE instructions, one for 3840 bytes and one for 1792 bytes. If the I/O requests are made in this sequence, however, the controller requires 14 revolutions to complete the first instruction (seven to write blocks 9 through 15 on the first track, and seven to write blocks 1 through 8 on the second track) plus seven more to complete the second instruction (for blocks 9 through 15 on the second track) — a total of 21 revolutions.

- Reversing the sequence permits the controller to complete the requests in 14 revolutions: seven to write blocks 9 through 15 on the first track, and seven more to write blocks 1 through 15 on the second track.

Note: The preceding discussion refers only to revolutions spent in actual data transfer and to theoretical "best" cases. Additional revolutions may be required for initiating a seek to the next track, for filling dispatch time between requests, or for servicing other I/O requests.

When dealing with the permanent file, your program can refer to blocks by their relative block number in the file, rather than by their absolute block number. By use of a zero-length REPLACE, the program can determine the number of bytes to write on the first request (returned in SMSIML) so that subsequent requests will be track-aligned.

If an I/O error prevents successful completion of the request, the controller attempts to execute the request sequentially before reporting the error. This guarantees that all blocks up to the one for which the error is reported have been processed. It should be noted, however, that blocks on the diskette (REPLACE) or in the input data area (LREAD) may have been modified beyond the point of failure. Make no assumptions about the validity of such modifications, because multiple errors may have occurred. The first error encountered during sequential retry is the one reported.

If your application program does multiple-sector I/O write operations from a shared segment, your program must ensure that the data is not changed by another station while the multiple-sector I/O write is in progress. If some type of interlocking is not implemented by the program, it is possible for incorrect data to be written on the diskette. The CRC error statistical counter might increase when this happens.

Diskette Performance

Factors affecting diskette performance include the amount of arm movement and whether the operation is a read, write, or replace. The time required to read or replace temporary file records is affected by: the existence of a read index buffer; and the presence of the track index or the temporary file block in a buffer. The following are indicative timings for diskette functions:

- The track-to-track access time is approximately 0.006 second for 4701 diskette drives.
- The average physical diskette read time is approximately 0.2 second (assuming an average rotational delay).
- The average physical diskette write time is approximately 0.4 second (assuming an average rotational delay).
- The approximate time to read, write, or replace a diskette record (not including track access time) is:
  - Permanent file read operation: 0.2 second per block.
  - Overlay section read operation: 0.2 second multiplied by the number of blocks read. (To approximate the number of blocks read, divide the size of the overlay section by 256 and round to the next integer.)
  - Temporary file read operation: 0.2 second when the track index, but not the record, is in a controller buffer or 0.4 second when neither the track index nor the record is in a controller buffer (0.2 second to read the track index and 0.2 second to read the record).
  - Permanent file replace operation: 0.4 second.
  - Temporary file write operation: 0.4 second when the write buffer is full, or 0.8 second for the last block on a track (0.4 second to write the track index and 0.4 second to write the record).
  - Temporary file replace operations: 0.4 second when the record and index are in controller buffers, 0.6 second when the index, but not the record, is in a controller buffer (0.2 second to read the record and 0.4 second to write the record), or 0.8 second when neither the record nor the index is in a controller buffer (0.2 second to read the index, 0.2 second to read the record, and 0.4 second to write the record).

The controller diskette drive uses a contact read/write head. The read/write head is physically in contact with the recording surface while data is being read or written. As a result, there is some wear effect on the diskette surface which, over a period of time, will require replacement of the diskette. You can maximize the useful life of a diskette by distributing data so that most tracks are accessed with about the same frequency.

| Disk Performance

All of the factors affecting diskette performance also affect disk performance. The following are indicative timing of disk functions:

- The track-to-track access time is approximately 0.007 second for 4701 disk drives.
- The average seek time is approximately 0.035 second.
- The average rotational delay is approximately 0.0095 second.
- The average physical disk read time is approximately 0.01 second (assuming on average rotational delay).
- The average physical disk write time is approximately 0.03 second (assuming on average rotational delay).

Chapter 3. Extended Disk and Diskette Access Method

The Extended Disk and Diskette Access Method (EDAM) is a set of optional modules that you may include in your CPGEN. It allows application programs to define, create, process, and delete data sets on the primary and secondary diskettes, and on the disks.

Data Set Characteristics

Data sets are named collections of data stored on a disk or diskette.

EDAM allows you to write application programs that can process data sets in 2 major ways:

- process EDAM data sets; that is, keyed and unkeyed data sets, and take advantage of EDAM functions
- process data sets using basic disk and diskette instructions.

This chapter is a description of EDAM and EDAM data sets.

EDAM Data Sets

The following paragraphs briefly describe the characteristics of data sets. We use the four-letter abbreviations to refer to the various types of data sets.

TEMP (Temporary File Data Set) - is a data set that can be created by EDAM and processed using basic temporary file instructions. It should *not* be confused with the system Temporary File (SYSTF).

ESDS (EDAM Sequential Data Set) - is sequential and includes exchange data sets on diskette only; has fixed length records up to sector size.

EDDS (EDAM Direct Data Set) - allows direct access through relative record numbers; has fixed length records up to sector size.

ASDS (Arrival Sequence Data Set) - is a direct data set that provides either fixed or variable length records and allows record lengths up to 1024 bytes. This data set organization also allows records to be deleted.

RKAP (Random Keyed Access Path) - is a data set that contains only pointers to user data and allows random access to records contained in an EDDS or ASDS.

KSAP (Keyed Sequence Access Path) - is also a data set that contains pointers. It allows access to records in either sequential or random order. Like the RKAP it does not contain user data.

Temporary File Data Sets (TEMP)

A TEMP data set is one that has the indexing characteristics of a temporary file; is opened with the temporary file option; and is accessed using temporary file instructions. (There is more information later in this chapter and Chapter 4, "Instruction Descriptions" on page 4-1 has the instruction descriptions.)

Sequential Data Sets (ESDS)

An ESDS is a data set that is always processed sequentially, beginning with the first record, no matter how many stations access the data set. Records are always written at the end; that is after the last record in the data set. Records are always read from a position indicated by a current input pointer maintained by the controller. A single input pointer is maintained for each sequential data set, and initially points to the beginning of the data set. As any station reads a record, this input pointer is moved to the beginning of the next record. The next station to issue a read gets the next record.

The controller also maintains a current output pointer for each sequential data set. The output pointer position is based initially on the values of the End-of-Data address and the Offset-to-Next-Record-Space indicator in the data set label. When any station writes to the data set, the data is stored at the location of this current output pointer, and the pointer increases to the next location. New records can be added until space in the data set is exhausted.

Using this sequential organization, a record once read cannot be reread until either the current input pointer is reset to the beginning of the data set, or until the data set is closed and reopened. A station can request exclusive use of a data set (via LDKT) to read and process an entire sequential data set without interruption.

ESDS characteristics are:

- Sequential
- Blocked or unblocked
- Sector length of 128 or 256 bytes
- Fixed length records up to sector length.

EDAM processes several sequential data set organizations, including Basic Exchange and H Exchange data sets. Exchange data sets can be used for exchanging data with other systems. They are described in the *IBM Diskette General Information Manual*, GA21-9182.

The following sections describe the characteristics of each of the exchange data sets.

Basic Exchange Data Sets: A Basic Exchange data set is identified by a hex (40) in the Exchange Type indicator of the header label. EDAM ignores certain fields in the header label, and uses these default characteristics:

- Sequential organization
- Unblocked logical records
- Sector length: 128 bytes
- Fixed length records, 1 — 128 bytes per record.

Type H Exchange Data Sets: An H Exchange data set has the character *H* in the header label's exchange type indicator, and is valid only on a Type 2D diskette. When EDAM detects an H Exchange data set, it ignores selected header label fields and uses the following default characteristics:

- Sequential organization
- Unblocked logical records
- Sector length: 256 bytes
- Fixed length records ranging from 1 to 256 bytes per record.

Direct Data Sets (EDDS)

Records in a direct data set are assigned relative record numbers. Although records are always written to the end of a direct data set, records can be read and replaced using the relative record numbers. As each record is added to the data set, the controller assigns it a record number relative to the beginning of the data set (1 is the first). Later, records can be read and replaced in any order.

Write requests to a direct data set use an output pointer as used for a sequential data set. Thus, writes by several stations to the same data set can result in interleaved records. Reads and replaces in a direct data set by one station are independent of those by other stations, except that reads and replaces of the same record by 2 or more logical work stations must be synchronized by the application programs to avoid loss of an updated record.

EDDS characteristics are:

- Direct
- Blocked or unblocked
- Sector length of 128 or 256 bytes
- Fixed record length up to sector length.

Arrival Sequence Data Sets (ASDS)

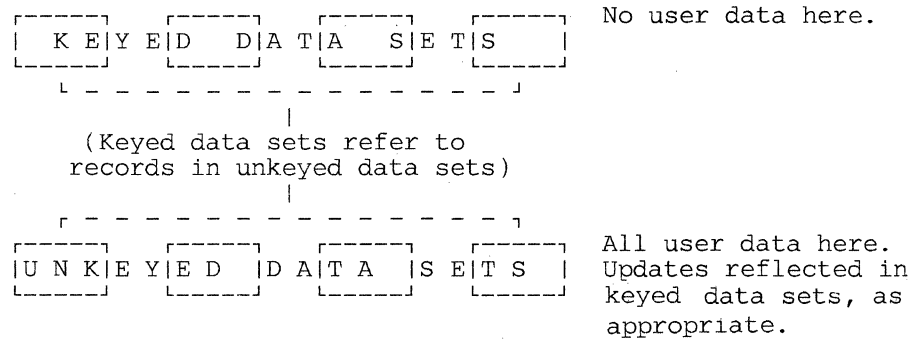
An Arrival Sequence Data Set allows fixed or variable length records that can span physical sectors on the storage medium. You can create an ASDS sequentially and records can be read, updated, or deleted either sequentially or directly by specifying the record sector number and the byte offset of the beginning of the record.

ASDS characteristics are:

- Direct
- Blocked and spanned
- Sector length of 256 bytes
- Fixed and variable record lengths up to 1024 bytes.

Keyed record access provides a way to access records based on specified key fields in those records. Two distinct types of keyed access are provided. One type, RKAP, emphasizes performance and provides only for direct access to records through their keys. The other type of keyed access, KSAP, emphasizes function and, in addition to direct access to keyed records, provides for access to keyed records in logical sequential order based on the values of their keys.

For each type of keyed access, the user processes a keyed data set as though it contains the data records. However, the keyed data set merely points to the data records within the associated unkeyed data set. Updates applied to an unkeyed data set cause it to appear as if those updates were also applied to all keyed data sets that refer to records in the updated unkeyed data set. These data set relationships are illustrated in the following diagram:



When you define a keyed access data set, the location and length of key fields within the unkeyed data set records must be specified. Up to seven keyed data sets, each having its own key field definition, may be defined to access records from one unkeyed data set. Also, one keyed data set can provide for merged, keyed retrieval of records from up to seven unkeyed data sets. The unkeyed data set containing the records referenced through a keyed data set may be either an EDDS or ASDS, but may not be an ESDS or TEMP.

The content of the keyed data sets is defined and maintained entirely by the system; no user data is contained in a keyed data set. From an external view, the keyed data sets appear to be fixed record length data sets with a record length of 256 bytes. Sector length of 256 bytes is required.

For an RKAP, one record in the keyed data set is required for approximately every twenty-five records accessed through it.

For a KSAP, one record in the keyed data set is required for approximately every twenty records accessed through it.

The characteristics of RKAP and KSAP data sets are:

- Direct
- Unblocked
- Sector length of 256 bytes
- Fixed record length of 256 bytes.

Using Basic Disk and Diskette Instructions

EDAM allows use of permanent and temporary file instructions for data sets located on both disk and diskette. Later in this chapter you will find some general discussion of using permanent and temporary file instructions with EDAM data sets.

Data Buffers

All EDAM input and output operations require the use of a buffer to hold the data while it is being read from or written to the device. EDAM uses a single pool of buffers for all drives.

You specify the number of buffers generated for use with EDAM data sets in the EDAM operand of the CPGEN FILES macro. The total number of buffers generated is determined by adding the number of buffers specified in each FILES statement. Each buffer holds data from one physical sector on the storage media and occupies 256 bytes of controller storage.

During system operation, the controller assigns these data buffers to physical sectors as needed, keeping track of the device address associated with the contents of each buffer. It notes each time the contents of a buffer are changed.

When an operation requires access to a particular sector, the controller checks the buffers first to see if the sector is already in storage. If so, a physical read can be avoided; if not, a buffer must be assigned to the sector by the controller.

Buffers are selected for re-assignment according to the following priority:

1. Unreferenced and unchanged
2. Referenced, but unchanged
3. Changed.

If you have changed all buffers, one is chosen and written to the device by the controller before the buffer is reassigned.

EDAM Functions

EDAM provides the ability to define, allocate, and process data sets on disk or diskette. The various service functions for data sets are provided through the LDKT instruction. The primary means for processing data sets is through the LREAD PLR, LWRITE PLR, LCHECK PLR, REPLACE PLR, and DELETE PLR instructions. Alternatively, the application program may process a data set using temporary file instructions (LREAD TFn, LWRITE TFn, REPLACE TFn, LCHECK DSK, LREAD C, LWRITE C, and REPLACE C) or permanent file instructions (LREAD P and REPLACE P). Two additional instructions (LREAD DSID and REPLACE DSID) enable permanent file type of processing for data sets on an unstopped operating diskette.

Data Set Service Functions

The basic service functions relating to data sets concern the existence of the data set. EDAM allows the definition of the data set attributes and the allocation of space for the data set as independent, but combinable functions of the LDKT Allocate instruction. Other LDKT functions include deallocating a data set and its definition, managing data set buffers used in processing data sets, renaming data sets, and querying various data set and device information.

The attributes of a data set include the type of data set, the amount of space the data set may occupy, and the length of records in the data set. For keyed data sets, the attributes include the position and length of the key field in the unkeyed data set records and the names of unkeyed data sets whose records are to be accessed through the keyed data set.

Prior to processing records in a data set, you must open the data set by means of the LDKT Open instruction. A value returned in SMSDID is used to refer to this data set in subsequent instructions. When processing is completed for a data set, you must close the data set by means of the LDKT Close instruction.

Record Processing Using PLR Instructions

The LREAD PLR, LWRITE PLR, LCHECK PLR, REPLACE PLR, and DELETE PLR instructions enable the processing of logical records contained within data sets. Multiple records may be blocked within a physical sector on the disk or diskette or may even span sectors. The PLR instructions provide for accessing the logical records, as appropriate, so that the application program need not be concerned about the physical layout of the records within the device sectors.

The record to be accessed may be identified by its relative location within the data set or, for keyed data sets, by the value of the key within the record. Alternatively, data sets may be accessed sequentially; that is, in the order records were written for unkeyed data sets or in the order defined by the entries in the keyed data sets.

Note: The record processing instructions generally require a buffer supplied by the application program to contain the record. For record processing instructions involving keyed data sets, a second buffer immediately following the first buffer and the same size as the first buffer is used as a work area for keyed data set processing. Therefore segment 14 cannot be issued for keyed record processing.

Diskette Programming

The preceding sections of this chapter have described some of the aspects of disk and diskette programming that are provided by EDAM. The succeeding sections describe the EDAM functions and programming considerations that are unique to diskette usage.

Defining Data Sets

You must allocate a SYSDSLBL direct data set on any diskette on which EDAM data sets will be defined. The 'SYSDSLBL' data set should be created using the 4700 Installation Diskette, or the user data set option of the Host Transmission Facility. This 'SYSDSLBL' data set must contain one 256-byte record for each data set. The number of records must equal the maximum number of data sets that will exist on the diskette. The controller maintains this data set for you. It contains parameters relating to keyed access; parameters used in the allocation of space for the data set; and a copy of the standard diskette header label.

Sector Deletion and Relocation

A diskette qualifies for Alternate Sector Relocation (described later) if it contains an ERMAP record (in discontinuous binary format, that is, the character B in position 23 of the ERMAP record) on track 0, sector 5, and if a data set named ERRORSET is found during initialization.

Defective Physical Sectors

A defective physical sector is one from which data cannot be successfully read because of defects in the diskette recording surface. Such defects may occur because of physical damage to the diskette, or they may be caused by wear that occurs in normal use. They may be detected on either read or write operations, because all writes are followed by a read-back check to confirm the success of the write.

If a defective sector is detected on a read, there is no possible recovery. The application program request that initiated the read is ended with status indicating a data check.

If a defective sector is detected on a write, the recovery attempted, if any, depends on the organization of the data set that contains the defective sector.

Control Records

Each sector on a diskette has an associated field called an address mark. The address mark identifies the sector, and indicates whether the sector contains a data record or a control record.

Three types of control records can be written on a diskette: the Delete control record, the Sequential Relocate control record, and the Alternate Relocate control record.

Delete Control Records

A delete control record is identified by C'D' in its first data byte, and signifies that the sector in which it appears is not currently in use. New and reformatted diskettes may have delete control records in all sectors on tracks 1—74. These records are also used on track 0 to indicate inactive header label sectors.

The controller does not write such records in sectors assigned to a data set, and does not expect to encounter any. However, if a data set is allocated with some blocks preassigned and not initialized, such records may be encountered if those blocks are referred to by the application program. If the application program preassigns blocks without initializing them, it must be prepared to handle the status returned when these control records are encountered.

Sequential Relocate Control Records

A sequential relocate control record is identified by a C'F' in its first data byte, and signifies that the sector in which it appears is defective. The data intended for that sector can be found in the next sequential nondefective sector.

The controller will write such records if it encounters a diskette surface defect while attempting to write to a sequential data set. In reading a sequential data set, a Sequential Relocate control record is skipped and the next sector is read.

When the controller attempts to write a buffer to a sequential data set and cannot successfully read it back, the controller writes a Sequential Relocate control record at the point of failure, unless the point of failure is in the last physical sector of the data set. This write is considered successful if at least the first byte can be read back to permit identification of the control record type.

If the control record is successfully written, the original data block write is retried in the next physical sector.

If the defective sector is the last physical sector in the data set, or if the first byte of the control record cannot be read back successfully, recovery ends and status indicating Data Check is reported to the application program.

Alternate Relocate Control Records

An alternate relocate control record is identified by a period C'.' in its first data byte, and signifies that the sector in which it appears is defective. The data intended for that sector can be found in a data set named ERRORSET.

The controller will write such records if it encounters a diskette surface defect while attempting to write to a direct data set, and if the diskette qualifies for the alternate relocation method of processing defective physical sectors. In reading a direct data set, alternate relocation control records are not normally encountered.

Because the controller keeps the ERMAP directory in storage, and because the alternate relocation method of processing defective sectors requires synchronization between the ERMAP record and the ERRORSET data set, you should not write alternate relocate control records in any sectors on diskettes that meet these criteria. If you do, they will not be processed by the controller.

The application program makes direct data set READ and REPLACE references by giving the number of the record or sector to be accessed relative to the start of the data set. This form of addressing permits accessing a data set in any sequence.

To the controller, each such access is independent of any that preceded it. All requests are handled by converting the given relative number to a specific diskette address by means of a computation based on the Beginning-of-Extent address and, for record level access, the record length and the Block Length obtained from the header label. Obviously, if a data set contained Sequential Relocate control records, this computation would yield incorrect results for any records or sectors beyond the location of the first Sequential Relocate control record. Therefore, the sequential relocation method of handling defective physical sectors is not used in direct data sets.

If you access any sector of a direct data set (including LWRITE, LREAD, REPLACE, and DELETE operations), and the diskette meets the criteria for the alternate relocation method of processing defective physical sectors, the controller will look first for the sector address in the ERMAP directory. If the address is found in the directory, the access is directed to the corresponding sector in the ERRORSET data set rather than to the original address.

If the diskette does not meet the criteria for alternate relocation support, or if the address is not found in the ERMAP directory, the access is directed to the original address.

If a control record is encountered on a read, the request ends with status indicating Control Record Read.

If a write fails because of a diskette surface defect, and if the diskette meets the criteria for alternate relocation support, the controller searches the ERMAP directory for an unused sector in the ERRORSET data set. If one is found, the data intended for the original sector is written to the ERRORSET sector, the ERMAP directory is updated and written to the ERMAP sector on track 0, and an Alternate Relocate control record is written at the location of the defective sector.

A write to the ERRORSET sector; to the updated ERMAP sector; or to the Alternate Relocate control record might fail. This can occur if the diskette does not qualify for alternate relocation support, or if no unused sectors remain in the ERRORSET data set. The operation will be ended with status indicating the cause of the failure.

Unrecoverable Write Errors

If a write error occurs on any data set and either no recovery is possible or any attempted recovery fails, status is returned indicating that the buffer contains an unwritable record.

The unwritable buffer is detected by the application program when a REPLACE operation ends with Data Check status, or when any operation ends with status that either includes the Prior Operation bit or that consists of the Prior Operation bit by itself.

Status consisting of the Prior Operation bit by itself occurs when:

- A data set Close is attempted and one or more blocks of the data set is contained in a buffer flagged unwritable.
- Any EDAM operation requires the assignment of one or more buffers and sufficient buffers are not available because some are flagged as unwritable.

- Volume initialization is attempted and one or more buffers belonging to the drive contains any block that is flagged as changed, even though it is also flagged as unwritable. (This could occur if the drive had previously been made not ready while any of its buffers held blocks that had been changed, but had not been written.)

EDAM provides several facilities by which the application program can attempt its own recovery. These include the Buffer Inquiry and Buffer Release functions and the Ignore Errors option of the Close function. See the LDKT instruction in Chapter 4, "Instruction Descriptions" on page 4-1 for more information.

For a sequential data set, where relative numbers are not used, the method of recovery is as follows. The station can reset the current input pointer, and request exclusive use of the volume (using LDKT). It can then reread the entire data set. The data can be printed, transmitted to the host, or written to another diskette drive. Finally, LDKT can be used to release all buffers. Alternatively, the data set could be closed with the Ignore Errors option, in which case all unwritable buffers are freed but not written to the diskette.

Disk Programming

The following sections of this chapter describe the aspects of EDAM that are unique to disk usage.

Disk Space Allocation

In keeping with support available in other systems, EDAM supports multiple extents for all except TEMP data sets. If you use multiple extents, you must specify, in the data set definition, the number allowed and the size of each extent. By supporting multiple extents, initial allocations need not be extravagantly large and yet the space requirements for a data set can grow with the needs of the data set. You may specify that all sectors are to be assigned when an EDAM data set is allocated (DKT1NA = X'FFFF'). If so, all secondary extents will be allocated initially. All sectors of an RKAP data set are initially assigned, resulting in all secondary extents being allocated initially.

For purposes of space allocation on the disk device, a "page" of space equivalent to sixteen sectors is the unit of space allocation. This is large enough to be a significant amount of space yet small enough that a significant percentage of the total space is not involved. A page thus contains 4096 bytes of usable storage. There are 7531 pages on a large disk drive and 3765 on a small disk drive.

Data Set Labels

The need for compatibility and for logical growth from the diskette to the disk device require that there be a degree of similarity in the way data set header labels are stored by the controller.

For the disk device, the data set header labels are stored in a header label data set (SYSDSLBL) rather than in a reserved area of the drive. Each of sixteen possible extents for the SYSDSLBL occupies two pages of storage and accommodates labels for thirty-two data sets.

The first half of each data set label contains a header label in a format similar to that supported on diskette track 0. The last half of the record contains the attributes for allocating space using a definition; for indexed access; and for multiple extents.

To locate the appropriate header record in SYSDSLBL, an RKAP data set (named SYSDSHSH) is maintained to provide keyed access to the SYSDSLBL records. The SYSDSHSH data set occupies one page of space. Both SYSDSLBL and SYSDSHSH are created when a disk is initialized for EDAM using the 4700 Installation diskette.

Up to 510 user data sets may be defined.

Sector Deletion and Relocation

Defective disk sector relocation is handled automatically. Sequential or alternate relocate control records will not be encountered when processing the disk. Unrecoverable write errors will rarely be encountered when writing to a disk data set because an alternate sector will have been dynamically assigned. When it is not possible to automatically assign an alternate sector, no recovery by the application program is possible.

Accessing Permanent and Temporary Files

When a temporary or permanent file instruction is issued, the controller tests flags to determine if the primary diskette drive is to be accessed (both flags are off). If the primary diskette drive is selected and is not stopped, the temporary or permanent file on the IPL diskette is accessed. If the primary diskette drive is selected and is stopped, no temporary (SYSTF) file processing is allowed.

If the primary diskette drive is not selected, the controller uses the value in SMSDID to identify the data set, device, and drive to be accessed. If the data set has been opened as a temporary file, only the temporary file instructions are allowed. If the data set has been opened as a non-temporary file, permanent file instructions are allowed. If an instruction is issued that is not allowed, it is rejected with appropriate status.

You can create a TEMP data set on a disk or on a diskette in the secondary drive using EDAM. The data set is opened, using LDKT with the temporary file option and can be processed using the basic temporary file instructions.

Any data set opened as a temporary file can be opened with either the warm-start or cold-start option. During a cold start, all previously stored records in the data set are made unavailable; the next LWRITE places data in the first location. During a warm start, all previously stored records are available.

Note: If SYSPF or SYSTF are deallocated and later allocated on the primary drive, problems will occur if data is to be accessed in either of these data sets.

Temporary File Characteristics: A data set accessed using the temporary file instructions, must have these characteristics:

- Sector length 256 bytes
- Logical records defined as unblocked, fixed-length, 256-byte records
- Exchange Type E

- Direct organization
- All sectors assigned
- No secondary extents.

When a station has opened a data set with the temporary file option, all basic temporary file instructions may be used (except those that refer to the system log) to process the data set:

```
LWRITE TFn
LCHECK DSK
LREAD TFn
LREAD C
REPLACE TFn
REPLACE C
```

Using Permanent File Instructions: When a station has opened a *direct* data set without the temporary file option, the station can use either the basic permanent file instructions:

```
LREAD P
REPLACE P
```

or the EDAM instructions:

```
LREAD DSID
REPLACE DSID
```

to access whole sectors of data by relative sector numbers.

Available Data Set Space

The number of available (unwritten and unassigned) sectors (SMSADS) in the current extent of any EDAM data set is updated during LWRITE, LREAD, REPLACE, DELETE, LCHECK, LDKT Open, and LDKT Close operations. If the operation is not successful, the value in SMSADS is meaningless. If the data set is keyed, SMSADS reflects the data set described by SMSUNK. Both SMSADS and SMSUNK are meaningless after an LCHECK for a keyed data set.

Obtaining Disk and Diskette Information

The Global Machine Segment (GMS), described in *Volume 1: General Controller Programming*, Appendix B, contains information that your program might need. Fields within the GMS will contain information such as diskette characteristics (1-sided or 2-sided).

CPGEN Requirements

In order to access the secondary diskette or disk drive, the controller requires an area of storage (for control information) as defined by the FILES macro during configuration. The FILES macro also specifies, for each drive, whether EDAM will be used; the EDAM options that will be included; the number of data sets to be opened concurrently; and the number of buffers required for EDAM processing.

Chapter 4. Instruction Descriptions

COMPDKT--Compress Diskette Data

This instruction makes unused diskette space available. COMPDKT groups data sets on a diskette so that unallocated space between data sets becomes contiguous. COMPDKT can also extend, truncate, or delete as many as three data sets on the compressed diskette according to a parameter list you create and specify by Operand 2.

To prevent data from being lost, the recommended procedure for using COMPDKT is first to copy the diskette to be compressed, and then compress the copy. The original diskette would be useful if COMPDKT ends in an error condition. For example, a power failure occurs; you remove the diskette during compression; or a diskette read or write error occurs.

Before you issue COMPDKT, use SMSDT2M to set the bit that selects the appropriate diskette drive.

To use this instruction, you must code the P41 operand on the OPTMOD configuration macro.

The installation diskette and system monitor also provide diskette compression, as described in the *IBM 4700 Subsystem Operating Procedures*: GC31-2032.

There is no corresponding function for disks.

Do not compress a 3600 operating diskette.

If a 4700 operating diskette is compressed you cannot "start" it without first performing an IPL with the diskette.

The COMPDKT instruction format is:

Name	Operation	Operand
[label]	COMPDKT	{ defld2 | (defrf2) | (reg2) | seg2,disp2 }

Operand 2

This operand specifies the location of the COMPDKT parameter list. To compress without changes, define a parameter list, 1 byte long, containing hex FF. To compress with changes, you must begin the parameter list with a 1-byte count field that has the number of data sets (1 to 3) to be changed. The appropriate parameter list entries must follow this count. You may use DEFCDK (Appendix B, "4700 COPY Files" on page B-1) to describe the parameter list entries.

In the following discussion: EOD = End-of-Data; EOE = End-of-Extent; and BOE = Beginning-of-Extent. Refer to the *IBM Diskette General Information Manual*, GA21-9182, for further information.

Byte	Meaning
0-16	Data set name
17	Flag byte:

Bit	Meaning
0	Truncate data set
1	Set EOD = EOE + 1
2	Track boundary request
3	Delete data set
4-7	Must be zero

Following are the valid combinations of bits 0 and 1 in byte 17:

00 -- Extend the data set extent by n sectors. The sector count, n, may be between 0 and the number of available sectors on the diskette.

01 -- Extend the data to the extent size (that is, extend EOD to EOE+1). The sector count must be zero.

10 -- Truncate the extent by n sectors. The sector count, n, may be between 0 and the current size of the data set. If the sector count equals the extent size, the data set is deleted.

11 -- Truncate the extent size to the data size (truncate EOE to EOD-1). The sector count must be zero. If EOD=BOE, EOE=EOD.

If bit 2 is on, the data set should be aligned on track boundaries before and after being moved (including any size adjustments specified by bits 0 and 1).

If bit 3 of byte 17 is on, the data set is deleted. No other bits can be on with bit 3. The sector count must be zero.

The request is rejected if a conflict results such as bits 0 and 1 or bit 3 on with a nonzero value in the sector count, or any bit on with bit 3.

18-21	Binary count of sectors of change in the existing extent. The high-order 2 bytes of the sector count must be zero, or the parameter list is rejected.

Note: COMPDKT does not guarantee that track alignment can be maintained on sequential data sets. Therefore, track boundary requests on data sets having sequential organization will be ignored.

After you compress data on a diskette, you must open a diskette data set before you access that data set. When you expand or reduce a temporary file, specify a sector count that is a multiple of 16.

Diskette compression can cause extensive movement of the diskette head, which can be heard by anyone standing near the controller. Do not be concerned.

Condition Code: COMPDKT returns condition codes indicating the status of the operation, as follows:

Hex Code	Explanation
01	The instruction executed successfully.
02	Status is returned in SMSDST.

Program Checks (hex): 01, 02, 09, or 27 can be set.

DELETE--Delete Records from Data Sets

The DELETE instruction is provided to delete a record contained in an ASDS and to delete references from keyed access data sets to records contained in unkeyed data sets.

DELETE PLR deletes a record from an ASDS data set and deletes references to an ASDS or EDDS record from RKAP or KSAP data sets. Before issuing DELETE PLR, set:

SMSDID
To contain the data set ID of a data set opened without the temporary file option.

SMSRPS
To contain the relative position, in binary, of the record to be deleted for EDDS and ASDS data sets. For an RKAP or a KSAP data set, this field should contain zero to indicate that the key for the record to be deleted is found within the buffer pointed to by operand 2 of the instruction.

When a record within an ASDS is deleted, the space occupied by that record is not reused until the data set is reorganized (see "Reorganize a Data Set" on page 4-33). The space remains assigned to the deleted record, and the data remains on the record even though the record cannot be retrieved. When an ASDS is read sequentially, deleted records are skipped. Entries in a KSAP or RKAP that would refer to a deleted record are removed from the KSAP or RKAP. Space occupied by those KSAP or RKAP entries will be reused, as necessary, for newly added entries.

A delete request for a record that exists in an EDDS having one or more associated keyed data sets, either directly to the EDDS or through a KSAP or RKAP, causes all references to that record to be deleted from the keyed data set. A portion or all of the EDDS record may be reinitialized by the controller to indicate that the record has been logically deleted. (See the Allocate function of the LDKT instruction.) A delete request directly to a TEMP, an ESDS, or an EDDS having no associated keyed data sets is invalid.

You may use either segment-header, segment-displacement, register, or modified register addressing to specify a buffer for a keyed data set delete. If segment-header addressing is used, the data buffer must be in the segment specified in the instruction. The data must start at the location pointed to by the secondary field pointer (SFP), and continue up to, but not including, the location pointed to by the primary field pointer (PFP). If segment-displacement addressing is used, the displacement to the data and the length of the data are as specified in the instruction.

Name	Operation	Operand
[label]	DELETE	PLR , { defld2 / (defrf2) / (reg2) / seg2,disp2,len2 / seg2 }

PLR
Specifies that a logical record is to be deleted.

operand 2
Specifies the location of the data buffer.

Condition Codes: One of the following is set:

Hex Code	Explanation
01	The operation was successful.
02	Status is stored. The status code is contained in SMSDST.

Program Checks (hex): 01, 02, or 27 can be set.

FORMDKT--Format a Diskette

Use this instruction to format Diskettes 1, Diskettes 2, and Diskettes 2D into 128- or 256-byte sectors according to a 9-byte parameter list you specify with Operand 2. See the "DEFFDK" copy file in Appendix B, "4700 COPY Files" on page B-1.

Before issuing FORMDKT, use SMSDT2M to set the bit to select the appropriate diskette drive. The Installation Diskette and System Monitor also provide diskette formatting.

To use this instruction, you must code the P40 operand on the OPTMOD configuration macro.

There is *no* corresponding instruction for disk, however the function is provided via the 4700 installation facilities. See the *IBM 4700 Subsystem Operating Procedures*: GC31-2032.

The format of the FORMDKT instruction is:

Name	Operation	Operand
[label]	FORMDKT	{ defld2 | (defrf2) | (reg2) | seg2,disp2, }

Operand 2

This operand specifies the location of the parameter list containing the data record length, the diskette type, the data record request, the physical record sequence code, and volume identifier (ID) to be used when formatting.

Byte **Meaning**

0 The data record length to be formatted

hex 00 = 128 bytes per data record
hex 01 = 256 bytes per data record
hex FF = Default to record length of mounted diskette

1 The diskette type

hex 01 = Diskette 1
hex 02 = Diskette 2
hex 03 = Diskette 2D
hex FF = Default to type of mounted diskette

2 Record Request and Sequence Code

Bit 0 - Data record request

0 = format with Delete Control Records
1 = format with data records

Bits 1 - 7 Physical record sequence code

A hexadecimal value from 0 to D.

A value of 0 or 1 indicates
that records are physically sequential.

When formatting type 2D diskettes,
this code can be 0 to D.
A value greater than 1 may improve performance.

For other diskette types this code must be 0.

The physical record sequence code modifies the normal
sequence numbering of the records on the diskette.
For example, assume a physical record sequence code of 5.
The records would be numbered: 1, 6, 11, ... 21, 26,
2, 7, ... 20, 25.

3-8 New volume ID

Blanks = Default to the volume ID of the mounted
diskette.
Other = The specified value is used as the volume ID.
(The ID consists of one to six digits or letters.
The first character must be in position 1 of the
field; any unused positions in the field to the
right of the ID data must be blank. No blanks
are allowed between digits or letters
in this field.)

Notes:

1. Because of the nature of the format utility, host link and loop error counters increase when you format a diskette. The error counters increase because of overruns while the system is disabled.

2. While you are formatting diskettes, the 4700 cannot communicate with the host system.

3. In order to format a blank (unformatted) diskette, do not provide default values in the parameter list.

Condition Code: A condition code is returned indicating the outcome of execution. The condition codes are as follows:

Hex Code	Explanation
01	The instruction executed successfully.
02	Status was returned in SMSDST.

Program Checks (hex): 01, 02, 09, or 27 can be set.

LCHECK--Check the Status of a Data Set

The LCHECK instruction allows you to assure that data for a data set is written from internal buffers to the disk or diskette containing that data set.

LCHECK PLR causes partially-filled buffers associated with the specified data set to be written to the disk or diskette.

Before issuing LCHECK PLR, set:

SMSDID
To contain the data set ID of a data set opened without the temporary file option.

An LCHECK PLR instruction refers only to data set operations initiated by LWRITE PLR instructions or other instructions referring to keyed data sets. The LWRITE PLR instruction causes data to be placed in a buffer that is written to the disk or diskette when the buffer is full. If the buffer is not yet full, LCHECK PLR causes the partially-filled buffer to be written to the data set. When the write operation is complete, the controller stores status in SMSDST, sets the condition code, and proceeds to the next sequential instruction.

Note: A subsequent LWRITE PLR to the data set may fill the buffer, and cause the data in the buffer to be rewritten.

If the SMSDID field refers to an ESDS, EDDS, or ASDS data set, one unwritten buffer (if any) associated with that data set is written to the device and the status of that operation is returned. If the SMSDID field refers to an RKAP or KSAP data set, all unwritten buffers for the device that contains the data set are written. The cumulative status of those operations are returned. If the SMSDID field refers to a TEMP data set, no buffer is written and no status is returned (LCHECK DSK is provided for TEMP data sets).

LCHECK DSK causes a partially-filled buffer for the temporary file on the specified drive to be written to the disk or diskette. Before issuing LCHECK DSK, set:

SMSFG1
To identify the type of device and drive containing the temporary file

Note: A subsequent LWRITE to the temporary file may fill the buffer, and cause the data in the buffer to be rewritten.

Name	Operation	Operand
[label]	LCHECK	{ PLR / DSK }

DSK
Specifies that write operations to a temporary file are to be checked.

PLR
Specifies that write operations to a data set opened without the temporary file option are to be checked.

Condition Codes: One of the following is set:

Hex Code	Explanation
01	No status is returned.
02	Status is returned in SMSDST.

Program Checks: None are set.

LDKT--Disk and Diskette Service Functions

The LDKT instruction provides disk and diskette service functions, including:

- Allocating data sets
- Opening data sets for processing
- Updating data set header labels
- Querying data set information
- Querying volume information
- Utility functions: Renaming and reorganizing data sets; Releasing Buffers; and Resetting input pointers
- Controlling access to a volume
- Closing data sets to further processing
- Deallocating data sets.

The LDKT instruction points to a parameter list (see DEFDKT in Appendix B, "4700 COPY Files" on page B-1) that contains a request code defining the function to be performed and any parameters required for the function. The beginning of the parameter list is defined by the Secondary Field Pointer (SFP) of the segment that contains the list, or by a DEFLD or DEFCON instruction. The length of the parameter list is defined by the request code. If the length from the beginning of the list to the end of the segment is too short to contain all of the parameters required for the particular function requested, a program check occurs when the instruction is executed.

Allocate a Data Set

LDKT defines the attributes of a data set and optionally allocates space for the data set. If space is allocated, it assigns an initial contiguous set of sectors to the data set; fills in the header label record and extension and writes them; and, optionally for an EDDS data set, assigns some or all of the sectors occupied by the data set and initializes them to any specified value. All sectors are assigned for TEMP and RKAP data sets.

The following table summarizes the allowable values for certain fields of the header label. For ASDS, RKAP, and KSAP data sets, these fields are properly filled in by the controller on the basis of the type of data set being created.

	TEMP	ESDS	EDDS	ASDS	RKAP	KSAP
Record Attribute (RCA)	' '	note 1	note 1	'R'	' '	' '
Record/Block Format (RBF)	'F'	'F/b'	'F/b'	'M'	'F'	'F'
Exchange Type Indicator (ETI)	'E'	note 2	'E'	'E'	'E'	'E'
Logical Record Length (LRL)	256	user	user	user	256	256
Data Set Organization (DSO)	'D'	'S/b'	'D'	'D'	'D'	'D'
Block Length (BLL)	256	note 3	note 3	256	256	256
Physical Record Length (PRL)	'1'	user	user	'1'	'1'	'1'

Notes:

1. The record attribute must be set by the user. The valid values are 'B' for blocked records, or blank for unblocked records. Unblocked format for records less than or equal to half the sector size are not provided for an EDDS.

2. The exchange type indicator, for disk only, is set by the controller to 'E'. For diskette, the user must specify a blank, 'E', or 'H'. A blank may be specified for a data set created on a diskette formatted with 128-byte sectors or an 'H' for a data set created on a type 2D diskette formatted with 256-byte sectors. If blank or 'H' is specified, the controller also sets the record attribute, record/block format, and data set organization to blanks. If 'E' is specified (diskette) or controller-set (disk), the controller sets record/block format to 'F' and data set organization to 'S'.

3. Set by the controller.

Figure 4-1. Table of Header Label Values

Before LDKT is issued, you must set SMSFG1 to select the appropriate drive and to designate whether the data set is to be defined or allocated on a diskette or on a disk.

When an RKAP or KSAP data set is allocated that references an allocated unkeyed data set, a work space following the parameter list must exist. This work space is used to read the unkeyed data set records to be accessed through the RKAP or KSAP being allocated. Allocation of a keyed data set will take as much time as is necessary to read all associated unkeyed data sets and to add an entry to the keyed data set for each existing unkeyed data set record. The characteristics of the data set are specified in the LDKT parameter list. The parameter list used when allocating a data set is 32 to 218 bytes long and contains the following fields:

Byte **Value or Meaning**

0 Request Code: hex 01 = Allocate

1 Flag byte

Bit **Meaning**

0 Boundary Requirement
0 = No alignment required.
1 = Allocate on track boundary.

1-2 Allocation Units
00 = Extent size is given in sectors.
01 = Extent size is given in K (1024) bytes.
10 = Extent size is given in tracks.

3-7 Reserved, must be 0.

2-3 Number of tracks, sectors, or K bytes to allocate (in binary).

4-5 Number of sectors to assign (in binary).

6-7 Number of sectors to initialize (in binary).

8 Initialization value (hex).

9 Secondary flag byte.

Bit **Meaning**

0 Reserved, must be zero.

1-3 Type of Data Set
000 = Type undefined
001 = TEMP data set
010 = ESDS data set
100 = EDDS data set
101 = ASDS data set
110 = RKAP data set
111 = KSAP data set

4-5 Allocation Flag
00 = Allocate using parameters provided (no definition).
01 = Allocate using stored definition.
10 = Do not allocate; only store definition provided.
11 = Store definition and allocate from stored definition.

6-7 Reserved, must be zero.

10-89 Header label area

90-267 Extended parameter area

10-137 Header label return area

The header label area is used to pass label parameters to the Allocate function, and contains the following EBCDIC fields:

```
10-12   C'HDR' (set by the controller)
13      C'1' (set by the controller)
14      Blank (set by the controller)
15-31   Data Set Name
32-36   Block Length (set by controller)
37      Record Attribute
        (set by controller for TEMP, ASDS, RKAP, and KSAP)
38-42   Beginning-of-Extent Address (usually set by controller)
43      Physical Sector Length
44-48   End of Extent Address (usually set by controller)
49      Record/Block Format
50      Bypass Indicator
51      Data Set Security
52      Write Protect Indicator
53      Exchange Type Indicator
54      Multi-Volume Data Set Indicator
55-56   Volume Sequence Number
57-62   Creation Date
63-66   Logical Record Length
        (set by the controller for TEMP, RKAP and KSAP)
67-71   Offset to Next Record Space (set by the controller)
72-75   Blanks (set by the controller)
76-81   Expiration Date
82      Verify/Copy Indicator (set by the controller)
83      Data Set Organization
84-88   End of Data Address (set by the controller)
89      Blank (set by the controller)
```

The extended parameter area must be used when a data set definition is stored. The length of the extended parameter area must include through byte number 98. For RKAP or KSAP data sets, the length must include the number of associated data set names described by byte number 98. Only the portion of this area that is actually used is required.

```
90-91   Key Offset from beginning of record
        (binary 0 if not RKAP or KSAP)
92      Key Length (binary 0 if not RKAP or KSAP)
93      Extension Flag Byte
```

Bit	Meaning
0-3	User flags
4-6	Reserved
7	Duplicate key exclusion flag (RKAP and KSAP only) 0 = Duplicate keys not allowed 1 = Duplicate keys allowed

```
94      Number of bytes to reinitialize upon EDDS logical record
        deletion (binary)
95      Number of secondary extents (binary 0 if not disk)
96-97   Size of each secondary extent (K-bytes in binary - disk only)
98      Number of associated data sets (binary 0 if not RKAP or KSAP)
99-217  Names of associated data sets (17 characters each)
```

The parameter list must be at least 32 bytes long when allocating from a stored definition and must be at least 90 bytes long when a stored definition does not exist or is not being created. The parameter list must *not* be in Segment 14, otherwise the LDKT instruction will cause a program check.

When allocating a keyed data set, the segment following the parameter list and the header label return area must be large enough to contain two data set records. This space is used as a work area when inserting entries for existing unkeyed data set records into the keyed data set.

The fields indicated as "set by the controller" need not be initialized by the application program. In addition, if the data set type is specified in byte 9, then the fields described in Figure 4-1 on page 4-14 will also be set by the controller.

The Type of Data Set and Allocation Flag fields in the secondary flag byte are self-explanatory. The settings for the Allocation Flag allow allocation from a saved definition or definition without allocation. The Type of Data Set may not be undefined if the definition is to be saved. If a definition having the same name already exists on the drive, a request to store a new definition or to allocate space using the provided parameters is invalid. You much first purge the old definition (see Deallocate) before such requests are valid.

The number of tracks, sectors, or K-bytes to allocate is given as a 2-byte binary number. The maximum extent size that can be specified for a disk drive is 2048 K bytes (2 097 152 bytes), 512 tracks, or 8192 sectors. If the flag field indicates that this value is given in tracks, the controller multiplies the given value by the number of sectors per track (on the currently mounted diskette) to obtain the number of sectors to allocate. For disk, the number of tracks is always multiplied by 16, which is the number of sectors in a TF Unit and in a page of disk space. All disk allocations are multiples of 16 sectors.

For diskette, this value is used by the controller in conjunction with the Beginning-of-Extent address (whether that address is given in the parameter list or selected by the controller), to determine the End-of-Extent address. If both the Beginning-of-Extent and End-of-Extent addresses are given in the parameter list, the value in this field is ignored. If either the Beginning-of-Extent address or the End-of-Extent address is all zeros or blanks, this field must contain a non-zero value or the request will end with appropriate status.

For purposes of determining the necessary size for data sets, you should consider the following facts:

- Records in ESDS and EDDS data sets do not span sectors. Therefore, an integral number of records will reside in each sector. The amount of unused space will depend on the length of the records in the data set.
- A four-byte descriptor precedes each ASDS record and each portion thereof residing in a sector.
- Each sector of a KSAP provides access to approximately 20 records.
- Each sector of an RKAP provides access to approximately 25 records.

The number of sectors to assign applies only to EDDS, and is given as a 2-byte binary number. All TEMP data set sectors are assigned automatically. If the data set being allocated is sequentially organized, the value in this field is ignored. This value specifies how many sectors, if any, should be considered to be in use (that are, available to LREAD, REPLACE or DELETE operations). If specified as 0, no sectors are assigned. If specified as hex FFFF, all sectors are assigned. Any other value causes that specific number of sectors to be assigned, starting with the first sector. The controller uses this value in conjunction with the Beginning-of-Extent address to determine the initial value of the End-of-Data address. Any sectors not assigned in this manner are available for sequential assignment at a later time. Such sectors are assigned by using the LWRITE instruction. Until a sector has been assigned, any attempt to refer to it is rejected with unit exception status.

All sectors of an RKAP data set are assigned when it is allocated. This includes any secondary extents defined for an RKAP on a disk drive. Similarly, if all sectors are to be assigned for an EDDS data set defined on a disk drive, all secondary extents will be allocated and assigned when the data set is initially allocated.

The number of sectors to initialize applies to EDDS or TEMP data sets only and is given as a 2-byte binary number. If the data set being allocated has the sequential organization, the value in this field is ignored. It specifies how many of the sectors occupied by the data set are to be written from a buffer initialized to a specified value. If the value given in this field is zero, no sectors are initialized. If the value is hex FFFF, all initially allocated sectors are initialized. Sectors in secondary extents that are subsequently allocated are not initialized. Any other value causes that number of sectors to be initialized. All sectors of an RKAP are initialized to binary zeros.

The initialization value specifies the value to be used for initialization of data blocks. It applies only to EDDS and TEMP data sets. If the block length is less than the sector length, the sector is padded with trailing binary zeros. If any sectors are preassigned during allocation, but are not initialized, those sectors may be accessed with LREAD and REPLACE instructions, but with unpredictable results. If the data set being allocated has the sequential organization, the value in this field is ignored. To avoid unexpected results, a TEMP data set or a data set of an undefined type to be used as a temporary file data set, should have all sectors assigned and initialized to hex 00.

The Data Set Name is a 1-to-17 byte alphameric identifier that uniquely identifies each data set on a volume. The name is left-justified in the field and padded with trailing blank characters. The checks performed by the controller on the value given in this field are to confirm:

- That no existing header label on the diskette has the same name
- That the data set name does not begin with 'SYS'
- That the given name does not begin with a blank character.

If a diskette has a data set named ERRORSET, then the diskette is qualified for the Alternate Relocation method of processing defective physical sectors. This is not recognized when the data set is allocated, but will be recognized the next time the diskette is referred to following a not-ready to ready transition.

Block length is determined by the controller, based on the Logical Record Length and Record Attribute fields. If records are to be unblocked, this field is set equal to the Logical Record Length. If records are to be blocked, this field is set to the highest multiple of the Logical Record Length that is not greater than the Physical Sector Length. If sectors are spanned then block length must be 256 bytes.

Block Length is set by the controller for ASDS, RKAP, and KSAP. If the allocation is being made on a diskette, it must be a diskette formatted to a 256-byte block length.

The Record Attribute specifies whether logical records are to be blocked or unblocked. When a Basic Exchange data set is allocated, the controller sets this field to a blank. If not basic exchange, the record attribute must be specified either as a blank (records are unblocked), a C'B' (records are blocked), or a C'R' (records are spanned). If any other value is specified, the request is ended with appropriate status. If a value of C'B' is specified, and the logical record length is such that only one record can be written in each sector, the controller changes this field to a value of blank and no error is indicated. The controller sets this field when ASDS, RKAP, or KSAP data set types are defined.

The Beginning-of-Extent parameter is ignored for disk data sets. For diskette, the Beginning-of-Extent address can either be specified by the user or left to the controller to determine. The value given is the EBCDIC address of the first sector of the extent, in the form C'CCHRR', where:

CC = Track address (01-74)
H = Head Number:
(0 for side 1, 1 for side 2)
RR = Sector address:
(01-26 for 128-byte sectors on Diskettes 1 and 2)
(01-15 for 256-byte sectors on Diskettes 1 and 2)
(01-26 for 256-byte sectors on Diskette 2D)

Note: On an operating diskette, track address can be from 01-73.

If the value given is all zeros (or blanks), the controller will search for an unused extent of the specified number of sectors at the lowest available address, starting with CC = 01, H = 0, and RR = 01. The controller will recognize the track boundary alignment option, if specified. If sufficient space cannot be found, the request is ended with appropriate status. If space is found, it is allocated to this data set and this field is replaced with the address of the first sector. For disk, the value returned is the binary PBN of the first sector of the data set.

If the Beginning-of-Extent field is not all zeros or blanks, the controller determines whether an extent of the specified number of sectors can be allocated starting at the given address without overlapping any existing extents. If so, the space is allocated. If not, the request is ended with appropriate status.

The Physical Sector Length indicates whether this data set is to reside on a 128- or 256-byte formatted diskette. When a Basic Exchange data set is allocated, the controller sets this field to a blank. The physical sector length must be specified as blank if the data set is to reside on a 128-byte diskette or a C'1' if the data set is to reside on a 256-byte diskette. If the currently mounted diskette does not match the requirement specified by this field, the request is ended with appropriate status.

The End-of-Extent parameter is ignored for disk data sets. For diskette, the End-of-Extent can either be specified by the user or left to the controller to determine. It is used only if Beginning-of-Extent is also specified. The value specified is the EBCDIC address of the last sector of the extent in the same format as described for Beginning-of-Extent. If Beginning-of-Extent is given as zeros or blanks, or if this field is given as zeros or blanks, the End-of-Extent is set by the controller. For disk, the value returned is the binary number of sectors in the primary extent.

Record/Block format specifies how records are formatted within the block. Record/Block format may be given as either a blank or C'F', both of which mean fixed-length records, or C'M' for spanned records which may be fixed or variable length. If any other value is specified, the request is ended with appropriate status. When a Basic Exchange data set is allocated, the controller sets this field to a blank. If a definition is being stored, this field is set by the controller for some types of data sets. See Figure 4-1 on page 4-14.

The Bypass indicator has no meaning to the controller and any value will be accepted in this field. It should be a blank if the data set will be allocated on a Basic Exchange diskette. The value of this field may be altered any time LDKT updates the header label instruction.

The Data Set Security indicator has no meaning for the controller and any value will be accepted in this field. It should be a blank if the data set will be read on an exchange diskette. The value of this field may be altered any time LDKT updates the header label instruction.

The Write Protect indicator is not checked at allocation. When a data set is opened for processing and the indicator contains a C'P', the data set is considered write-protected and no data can be written to it. The value of this field may be altered any time LDKT is used to update the header label instruction.

The Exchange Type Indicator is set by the controller when a data set is defined. See Figure 4-1 on page 4-14 for a description of the Exchange Type Indicator.

The Multi-Volume Data Set indicator and Volume Sequence Number have no meaning to the controller. Multi-Volume data sets are not supported.

The Creation Date should contain the date in EBCDIC, in YYMMDD format. If the field is set to blanks, the current date is set by the controller using the system timer value.

The Logical Record Length is a required field given as a 4-byte, EBCDIC number. For data sets having record format C'F' or C' ', the valid range is from 1 to the physical sector length. It is right-justified and padded with leading zeros or blanks. If the value given for an ASDS is all zeros or blanks, variable record length is inferred. If a number is specified for an ASDS data set, the valid range is 1 - 1024 bytes, which results in fixed-length spanned records. If the value given, for other types of data sets, is all zeros or blanks, or is greater than the physical sector length the request is ended with appropriate status. When a data set is defined, the Logical Record Length is set by the controller for TEMP, RKAP, and KSAP types of data sets. See Figure 4-1 on page 4-14.

The Offset-to-Next-Record-Space is provided and maintained by the controller. It indicates the starting output position of the next sequential record relative to the last block preceding the End-of-Data address. It contains a decimal value used as a negative displacement into the block, and is set to an initial value of zero.

The Expiration Date is a 6-byte EBCDIC value in YYMMDD format. This field is used when requesting deallocation. If the date is less than or equal to the current date, it is considered to be expired. Any other value is considered to be the date on which the data set expires.

If the field is blank, the controller sets the current date. If the field is set to '+NNNN+' where NNNN is the number of days after the current date when the data set is to expire, the controller determines the expiration date when the data set is allocated.

The controller sets the Verify/Copy indicator to an initial value of blank. It has no significance to the controller and can be changed at any time by the LDKT Update Header Label instruction. Other valid values are C'V', meaning that the data set has been verified, or C'C', meaning that the data set has been copied to another medium.

The Data Set Organization value indicates whether access to the data set is limited to strictly sequential operations, or whether read and replace operations by location are allowed. It also determines what error recovery may be attempted in the event of write errors because of a defective diskette surface. When a Basic Exchange data set is allocated, the controller sets this field to a blank. Data Set Organization must be given as one of the following:

- hex 40 (blank) or C'S' (sequential organization)
- C'D' (direct organization).

If any other value is given for the Data Set Organization, the request is ended with appropriate status. When a data set is defined, this field is set by the controller. See Figure 4-1 on page 4-14.

The End-of-Data address is provided by the controller. It is determined from the Beginning-of-Extent address and the number of sectors assigned. If no sectors are preassigned, the End-of-Data Address is set equal to the Beginning-of-Extent address. If any sectors are preassigned, this field is set to the address of the first unassigned sector. If all sectors are preassigned, this field is set to the address of the next sector following End-of-Extent.

All remaining fields in the header label area are set by the controller to blanks.

Each of the last 48 bytes of the diskette header label are set to blanks unless the diskette is 1-sided and the data set has an Exchange-Type indicator set to blank (Basic Exchange Standards apply). In this case, the last 48 bytes of the label are set to binary zeros.

For diskette, if the store definition flag bit is set, and no invalid values are detected in the parameter list, the extended header label is written to the SYSDSLBL data set. When the requested space is allocated, the standard HDR1 formatted label is written to track zero. The 'SYSDSLBL' data set must exist on diskette before a store definition allocation request is processed. The 'SYSDSLBL' data set should be created using the 4700 Installation Diskette, or the user data set option of the Host Transmission Facility.

The extended parameter area is used while defining a data set. If the secondary flag byte (byte 9) specifies "store definition," then parameters through byte 98 must be specified, otherwise the extended parameter area is ignored.

The key offset from the beginning of the record is specified as a 2-byte binary value. It must be less than or equal to 1024 and must be specified for a KSAP or an RKAP. It is ignored for other types of data sets.

The key length field is a one-byte binary value that indicates the length of the key for a KSAP or an RKAP. It must be less than or equal to 255. This field must be zero for other types of data sets.

The extension flag byte specifies whether duplicate keys are allowed within a keyed data set. The first 4 bits of this flag byte are available to the user application. Any values are stored, not validated, and can be returned to the application via the LDKT Query Extended Header Label function.

When you logically delete a record from an EDDS data set the record can be reset to the initialization character value. The number of bytes to be reinitialized is a one-byte binary number specified at the time the data set is defined. This function is supported only if the EDDS data set is associated with any keyed data sets. Reinitialization starts from the beginning of the logical record. A value of hex 00 means no bytes will be reinitialized, and a value of hex FF means the entire logical record will be reinitialized.

The number of secondary extents is a one-byte binary number that specifies the maximum number of secondary extents to allocate. This parameter is valid only if the definition is for an ESDS, EDDS, ASDS, RKAP, or KSAP on a disk drive. The maximum value that can be specified is fifteen. Any value that is not in the range hex 00 - 0F is invalid and will cause the allocation request to be ended with an error status.

The size of secondary extents may be used only for an ESDS, EDDS, ASDS, RKAP, or KSAP allocated on the disk drive. It is a 2-byte binary number that specifies the size of each secondary extent that may be allocated, expressed in multiples of 1024 bytes. Because space allocations are in multiples of 4096 bytes, any size specification will be rounded up to the 4K boundary, if necessary. The maximum extent size that can be specified for a disk drive is 2048 K bytes (2 097 152 bytes).

The number of associated data sets is a one-byte binary number. It indicates, for a KSAP or RKAP, the number of ASDS or EDDS data sets that are to be accessed. The names of the data sets follow in the succeeding field. The maximum value that can be specified is seven and any value larger than that is considered an error and will cause the definition request to be ended with an appropriate error indication. This field may not be zero if a KSAP or an RKAP is being allocated, and must be zero for other types of data sets.

The indicated number of data set names must be specified. Each name should be left-justified (with trailing blanks) in a seventeen-character portion of this field. Each named data set must be an existing ASDS or EDDS defined on the same drive as the KSAP or the RKAP being defined. Any associated data set name fields beyond the number specified above will be ignored and do not have to be included in the total parameter length.

If the required space is successfully allocated and no invalid values are detected in the parameter list, the header label is written. A copy of the header label as it is recorded is returned in the parameter list starting at byte 10. As many as 128 bytes are returned, space permitting. The operation completes with zero status and a condition code of hex 01.

Open a Data Set

The LDKT Open function assigns an area of controller storage for the named data set (if an area is not already assigned) and initializes it with data from the header label (and extension). The value assigned to the opened data set is returned to the application, in SMSDID, for future references to the data set.

Before this LDKT is issued, you must allocate the data set. You must set SMSFG1 to indicate which device is to be used and which is the appropriate drive. The full 17-byte name of the data set to be opened is given in the parameter list exactly as it appears in the data set header label.

When a keyed data set is opened, any associated unkeyed data sets are also implicitly opened by the controller. A specific LDKT Open is not required for each associated data set if keyed access is used.

If the write-protect indicator is set in the data set label, the data set is opened with read-only access. The parameter list used when opening a data set is 19 to 138 bytes long and contains the following fields:

Byte	Meaning
0	Request Code: hex 03 = Open
1	Flag byte

Bit	Meaning
0	Temporary File 0 = Do not open as temporary file. 1 = Open as temporary file.
1	Cold/Warm Start 0 = Cold start 1 = Warm start
2	Exclusive Use 0 = Open for general use. 1 = Open for exclusive use.
3-6	Reserved, must be 0.
7	Return header label in input area 0 = Do not return header label. 1 = Return header label.

2-18	Data Set Name
10-137	Header label return area

The parameter list must be at least 19 bytes long whether or not the label is to be returned, and it must not be in Segment 14. Otherwise, a program check will occur when the instruction is executed. Note that the label return area overlaps the data set name parameter. If return of the header label is requested, as many bytes of the label will be stored as there is room for in the list, with a maximum of 128 bytes.

Note: The value returned in SMSDID may vary from one open to the next open of the same data set. You should not code your program assuming the same value is always returned by the open function for any particular data set.

If the open function completes successfully, SMSDID contains a data set ID value. This value uniquely indicates the data set and the disk or diskette drive on which it resides. If a data set is already open by at least one other station, the SMSDID value returned is the same as when the data set was first opened. In addition, the fields SMSADS, SMSCCD, and SMSDST are returned as appropriate. See the DEFSMS copy file in Appendix B, "4700 COPY Files" on page B-1 for definition of these fields.

If the temporary file option is set, and this is the first open of the data set, the characteristics of the data set are checked against those required of a temporary file data set. If they do not match, the request is ended with appropriate status. If the temporary file option is set, and the data set is already open by at least one other station, the previous open must also have specified the temporary file option. If not, the current request is ended with appropriate status.

If the temporary file option is set, and this is the first open of the data set, and the Warm Start flag is not set, the index counters are set to zero. If the temporary file option is set, and this is the first open of the data set, and the Warm Start flag is set, the index counters are initialized to reflect any existing logical records in the data set.

The index counters for a temporary file data set are organized according to the number of TF Units in the data set. The number of counters available depends on the TF operand that was coded on the FILES macro in the configuration specifications. If a data set having more than this number of TF Units in its extent is opened as a temporary file, only the number of TF Units for which there are counters are accessible. The data set is opened, but the request completes with status indicating "Temporary File Data Set Too Large."

If the temporary file option is not set, or if the data set is already open by at least one other station, the Warm Start flag is ignored. Only one temporary file data set can be opened at a given time on each disk or diskette.

If the Exclusive Use flag is set and this is the first open of the data set, a flag is set noting that the data set is open for exclusive use by this station. Subsequent accesses to the data set by this station are permitted, while attempted accesses by other stations are rejected with appropriate status. An open request is ended with appropriate status if the Exclusive Use flag is set and the data set is already open by at least one other station.

If an open of a keyed data set or any of its associated unkeyed data sets fails, then status is returned; none of the data sets are opened; and controller storage for each is reset to zeros. If any of the associated unkeyed data sets were already open from a previous operation, they will not be affected by the request that fails.

If an open request for a keyed data set is processed, the SMSICT field is returned containing a count of the implied-associated data sets that were opened by the keyed data set. If this count is zero, appropriate status is returned. If some of the implied data sets did not open because they were defined but not allocated, zero status is returned and SMSICT is meaningful. The keyed data set is not included in the count. If the keyed data set is already open, then the status and count are both zero.

Update Header Label

There are occasions when you may wish to modify the values of certain fields in a data set header label. The Update Header Label function of the LDKT instruction is provided for this purpose.

Before this LDKT is issued, the data set must be open; the value returned in SMSDID when the data set was opened must be placed in SMSDID. The controller will use the value in SMSDID to determine the data set, device, and drive to be accessed. You should periodically issue the LDKT Update Header Label instruction to allow the system to preserve its End-of-Date pointer.

The corresponding fields of the data set header label are updated with the values supplied in the parameter list along with the current values of the End-of-Data address and Offset-to-Next-Record-Space from the area of user storage associated with the open data set. Optionally, a copy of the updated label may be returned in the parameter list.

The parameter list used when updating a header label is 6 to 138 bytes long and contains the following fields:

Byte	Value or Meaning
0	Request Code: hex 02 = Update Header Label
1	Flag byte

Bit	Meaning
0	Bypass Indicator 0 = Do not update Bypass Indicator. 1 = Update Bypass Indicator.
1	Data Set Security 0 = Do not update Data Set Security. 1 = Update Data Set Security.
2	Write-Protect Indicator 0 = Do not update Write-Protect Indicator. 1 = Update Write-Protect Indicator.
3	Verify/Copy Indicator 0 = Do not update Verify/Copy Indicator. 1 = Update Verify/Copy Indicator.
4-6	Reserved, must be 0
7	Return header label in input area 0 = Do not return header label. 1 = Return header label.

2	New value for Bypass Indicator
3	New value for Data Set Security
4	New value for Write-Protect Indicator
5	New value for Verify/Copy Indicator
6-9	Reserved, must be binary zeros.
10-137	Header label return area

The parameter list must be at least 6 bytes long and must not be in Segment 14. Otherwise, a program check will occur when the instruction is executed. If return of the header label is requested and the list is at least 11 bytes long, as much of the label as will fit in the list will be returned.

Query Data Set Information

Query Extended Header Label

LDKT allows a station to request the extended header label parameters for a data set.

Before this LDKT is issued, SMSFG1 must be set to the drive and device to be accessed.

The parameter list that your application must set may be 4 or 22 bytes long. Bytes 2 and 3, and bytes 5 through 21 may be changed during LDKT operation depending upon the code that your program places in the flag byte (byte 1).

The parameters returned to your application may be from 101 bytes to 220 bytes long (including the first 22 bytes) depending on the number of associated data sets described in byte 100. The format is as described below. If the value in byte 100 is hex 00, then only bytes 0 - 100 will be returned.

Byte **Value or Meaning**

Parameter List

0 Request Code: hex 0F = Query Extended Header Label

1 Flag byte

Bit **Meaning**

0-1 00 = Query the Data Set Name that is in bytes 5 - 21.
10 = Query the Data Set Label that is at the relative position given in bytes 2 - 3.
11 = Query the next valid label that is after the relative position given in bytes 2 - 3.

2-6 Reserved, must be zeros.

7 1 = Return the diskette data set size (displacement 80) in sectors or K-bytes, depending on the setting of flag-byte 82 that coincides with flag-byte 1 of Allocate. If the flag byte indicates tracks, the size is returned in sectors.

2-3 Relative position of the data set label (binary)

4 Blank

5-21 Data Set Name

Returned Parameters

0 Request Code: hex 0F

1 Flag byte

2-3 Relative position of the data set label (binary)

4 Blank

5-21 Data Set Name

22-78 Remaining "HDR1" fields in the same format as the HDR1 label through and including end of data.
Note:
For disk, the BOE, EOE, and EOD fields will be binary values (5 bytes each); for diskette these fields will be character fields (TTHRR).

79 Blank

80-81 Current data set size in K-bytes (in binary) if bit 7 of the flag byte = 0.
For diskette allocations in sectors, the size is rounded down to the next multiple of 1024 bytes.
For diskette only, the size may be set to sectors if bit 7 of the flag byte is set.

82-90 Coincide with bytes 1 - 9 of the LDKT Allocate parameter list

91-98 Coincide with bytes 90-97 of the LDKT Allocate parameter list

99 Number of secondary extents allocated (hex 00 - 0F)

100 Number of associated data sets defined (hex 00 - 07)

101-219 Names of associated data sets (17 characters each padded to the right with blanks).

82-100 If the data set was not previously defined then these bytes will each contain hex 00.

The parameter list must be at least 4 or 22 bytes long depending on the value of the flag byte, otherwise a program check will occur when the instruction is executed. If not enough bytes exist through the end of the data segment to hold all of the query reply, then the reply will be truncated at the end of the data segment and error status hex 0101 is returned.

An attempt to query a specific data set (that is, Byte 1 Bit 0 set off) which does not exist causes the request to be rejected with error status hex 4004 (data set name unknown). If a data set cannot be found using the relative label position or the next relative position, error status is hex 4000 (End of Data).

For an ASDS or EDDS, the associated data sets are those KSAP and RKAP that access the data set being queried. For a KSAP or RKAP, the associated data sets are those ASDS or EDDS that are accessed by the KSAP or RKAP data set.

The current size of a TEMP or RKAP is always the size of its initial allocation. The current size of an ASDS, ESDS, EDDS, or KSAP is the total current size which includes any secondary extents that have been allocated. For diskette, if the user flag-bit 7 is set, the current size returned is in the same unit of measurement as the initial extent. An exception is if the initial extent is tracks; in this case, the current size units returned is sectors.

Buffer Inquiry

This LDKT function allows a station to obtain the identity of unwritable blocks that occupy EDAM buffers. This function, in combination with the LREAD DSID and LDKT Buffer Release functions, can be used in a write-error-recovery procedure.

When the buffer inquiry function is requested, SMSFG1 must be set to identify the drive for which unwritable buffers are to be identified and the SMSDID and SMSRPS fields must be set as the starting point for the unwritable buffer search. The SMSDID value is logically prefixed to the SMSRPS value to produce a combined field, that is the starting point for the unwritable buffer search. All unwritable buffers for the specified drive are searched to find the buffer having the next higher combined SMSDID and SMSRPS value. If any such unwritable buffer is found, the values identifying that buffer are returned in SMSDID and SMSRPS and the request is completed with a zero status. If no such unwritable buffers are found, the request is ended with appropriate status.

If the value returned in SMSDID identifies a keyed data set (RKAP or KSAP), then the relative position of the keyed data set label is returned in the parameter list. If an unwritable buffer for a keyed data set is encountered and the parameter list is too short to contain the relative position, appropriate status is returned. If the parameter list is long enough to contain the relative position field but the returned SMSDID value does not identify a keyed data set, the relative position field will be set to binary zero.

The relative position returned for a keyed data set can be used with the LDKT Query Extended Header Label function to determine the name of the keyed data set having unwritable buffers. After unwritable buffers associated with unkeyed data sets have been reconciled and released, a keyed data set can be recovered by first closing it using the "ignore errors" specification (to get rid of the unwritable buffer). The data set can then be deallocated and reallocated from its stored definition, causing it to be rebuilt to properly reflect existing unkeyed data set records.

The parameter list for buffer inquiry must contain at least one byte, but if keyed data sets are used, should contain three bytes, as follows:

Byte	Value or Meaning
0	Request Code: hex 08 = Buffer Inquiry
1 - 2	Relative position of keyed data set label.

The system handles alternate sector assignment for any write errors on disk; therefore, there should be no unwritable buffers associated with a disk drive. However, the instruction can be issued for a disk drive and will appear to be processed the same as for diskette.

Query Open Status

For any given data set, LDKT allows any station to determine which stations have the data set open by means of a bit map returned in the parameter list.

Before this LDKT is issued, the data set must be open and the value returned in SMSDID when the data set was opened must be placed in SMSDID. The controller will use the value in SMSDID to determine the data set, device, and drive to be accessed.

The bit map is a 64-bit (8-byte) field. Bit 0 is used for data sets opened by the controller; bits 1 - 60 correspond to station IDs 1 - 60; bits 61 - 62 are unused; and bit 63 indicates one or more implied open. For each nonzero bit in the map, the corresponding station has the data set open.

The parameter list used for querying open status is 10 bytes long, and contains the following fields:

Byte	Value or Meaning
0	Request Code: hex 0B = Query Open Status
1	Reserved, must be zero
2-9	Open bit map return area

The parameter list must be at least 10 bytes long and must not be in Segment 14. Otherwise, a program check will occur when the instruction is executed.

Volume ID Inquiry

LDKT allows any station to determine the volume ID of the disk or diskette volume without having to read the volume label.

Before issuing the LDKT, set SMSFG1 to select the appropriate drive.

The parameter list used for the volume ID inquiry is eight bytes in length and contains the following fields:

Byte	Value or Meaning
0	Request Code: hex 0A = Volume ID Inquiry
1	Reserved, must be zero
2-7	Volume ID return area

The parameter list must be at least eight bytes long and must not be in Segment 14, or else a program check will occur when the instruction is performed.

Because the disk media is not removable, it does not have a volume ID. The LDKT Query Volume ID function for the disk device will always return 'SYSDKx' where x is the drive identifier.

Unallocated Space Inquiry

LDKT allows any station to determine the amount of unallocated space on a diskette or a disk. In addition, a summary of the number of contiguous blocks of various sizes is provided.

Before issuing LDKT, you must set SMSFG1 to identify the device involved and to select the appropriate drive.

If a volume is currently mounted on the selected drive when this request is issued, binary numbers representing the space that is not allocated to any data set, are returned in the parameter list. If no volume is mounted on the selected drive or the selected drive is not included in the configuration, the request is ended with appropriate status.

For disk, the flag byte returned and the total unallocated space always indicates K-byte units regardless of the request flag values set by the application. For diskette, when K-bytes are requested, any contiguous blocks that are less than one K-byte in length are not included in the total count.

The parameter list used for unallocated space inquiry is 22 bytes long and contains the following fields (all fields returned by this instruction are in binary):

Byte	Value or Meaning
0	Request Code: hex 0D = Unallocated Space Inquiry
1	Flag byte

Bit	Meaning
0	Units for unallocated space 0 = Return space in sector units 1 = Return space in K-byte units
1	Return largest contiguous diskette block in units indicated by bit 0
2-7	Reserved, must be 0

2-5	Unallocated space return area
6-7	Number of blocks <4 K-bytes
8-9	Number of blocks >=4 K-bytes and <16 K-bytes
10-11	Number of blocks >=16 K-bytes and <64 K-bytes
12-13	Number of blocks >=64 K-bytes and <256 K-bytes
14-15	Number of blocks >=256 K-bytes and <1024 K-bytes
16-17	Number of blocks >=1024 K-bytes
18-21	Largest contiguous diskette block (in sectors or K-bytes)

If the parameter area provided is less than 22 bytes, only as much data as will fit will be returned.

Utility Functions

Buffer Release

LDKT allows any station to release EDAM buffers that contain unwritable blocks belonging to any EDAM data sets, other than temporary file data sets. It can be used to release a specific buffer, all buffers that contain unwritable blocks belonging to a given data set, or all buffers associated with the selected drive.

This function permits EDAM to reuse the buffer, and should be requested only after any pertinent recovery procedures have been completed. Once a buffer has been released, the unwritable block is no longer retrievable.

This release function requires a data set ID in SMSDID, a block number in SMSRPS, and SMSFG1 to select the drive. If the value passed in SMSDID is zero, all unwritable buffers associated with the selected drive (SMSFG1) are released. If the value passed in SMSRPS is zero, all unwritable buffers belonging to the specified data set are released. If neither SMSDID nor SMSRPS contains a zero value, only the buffer containing the specified block for the specified data set is released.

The release request always completes with zero status. Buffer error status is reset for any data set that has an unwritable buffer released.

The parameter list used for the buffer release is one byte long and contains only the request code, as follows:

Byte	Value or Meaning
0	Request Code: hex 09 = Buffer Release

Reset Data Set Input Pointer

LDKT allows any station to reset the current input pointer for a sequential data set to its initial value. This permits rereading of records that have been read previously without the necessity for closing and reopening the data set. In order for a station to request this function, the station must have the data set currently open.

Before this LDKT is issued, the data set must be open and the value returned in SMSDID when the data set was opened must be placed in SMSDID. The controller will use the value in SMSDID to determine the data set, device, and drive to be accessed.

This function can be requested at any time, regardless of the value of the current input pointer. Its effect is to cause the first record of the data set to be returned in response to the next LREAD issued to that data set by any station. If the specified data set does not have sequential organization, no error is reported and the operation has no effect on processing of the data set.

This function is useful in recovering from a permanent write error in the data set. A station receiving status indicating that a permanent write error has occurred can inhibit access to the volume by other stations; reset the input pointer for the data set having the error; and then read the data set in its entirety (either copying the records to a new data set, or handling them in some other appropriate way). After recovery is complete, the station can release the buffer containing the block that was unwritable; close the failed data set for all stations (using the Ignore Errors option); deallocate the data set; and permit other stations to access the volume again.

The parameter list used for resetting a data set input pointer is one byte long and contains only the request code, as follows:

Byte	Value or Meaning
0	Request Code: hex 0C = Reset Data Set Input Pointer

Reset Data Set Output Pointer

LDKT allows any station to reset the current output pointer for the open data set to the start of a data set. This permits re-using the data set as if it had just been allocated. It allows different stations to journal entries to a data set. One station might process the entries periodically and reset the output pointer without forcing journaling stations to reopen the data set. The data set must be an ESDS, EDDS, or an ASDS and cannot have any associated keyed data sets.

When multiple stations are accessing the same data set, the controlling station should issue LDKT Inhibit Access before Reset Output Pointer to temporarily prevent accesses to the common data set. After the data is processed, issue LDKT Permit Access to allow other stations to access the data set again. The MBB option module must be present in order to use the reset data-set output pointer instruction.

Before this LDKT is issued, the data set must be open and the value returned in SMSDID when the data set was opened must be placed in SMSDID.

Byte	Value or Meaning
0	Request Code: hex 11 = Reset Data Set Input Pointer
1	User flag byte: Reserved

Rename a Data Set

This LDKT function allows a data set to be renamed. Before issuing this LDKT, you must set SMSFG1 to identify the device type and to identify the drive on which the data set to be renamed resides. If the data set to be renamed is open to any station, the instruction is rejected.

The parameter list used by this function must be 35 bytes long and contains the following fields:

Byte	Value or Meaning
0	Request Code: hex 10 = Rename Data Set
1	Flag byte

Bit	Meaning
0-7	Reserved, must be 0
2-18	Current Data Set Name
19-35	New Data Set Name

If the parameter list is less than 35 bytes long, a program check will occur when the instruction is executed.

Both the current and the new data set names must be the full 17-byte name for the data set. A data set having the same name as the new name must not already exist on the specified volume. If such a named data set already exists, the instruction will be rejected with appropriate status. Data set names must *not* begin with 'SYS'.

Reorganize a Data Set

LDKT reorganizes an ASDS or a KSAP that has become inefficiently structured through extensive delete/add activity. For an ASDS, the reorganize function compresses the data set to eliminate the space occupied by deleted records. For a KSAP, the reorganize function reorders the entries so that the data set can be searched more efficiently.

Before issuing LDKT, you must set SMSFG1 to identify the device involved and to select the appropriate drive.

The data set must not be open to any station; otherwise, the request is rejected with appropriate status.

The parameter list used when reorganizing a data set is 18 bytes long and contains the following fields:

Byte	Value or Meaning
0	Request Code: hex 0E = Reorganize
1	Flag byte

Bit	Meaning
0-7	Reserved, must be 0
2-18	Data Set Name

The parameter list must be at least 18 bytes long, otherwise, a program check will occur when the instruction is executed.

The data set name is the full 17-byte name of the data set to be reorganized and must be given exactly as it appears in the data set header label.

An attempt to reorganize any other type of data set will cause the request to be rejected with appropriate status.

Reorganization Recovery: When an ASDS is reorganized, any associated KSAP or RKAP data sets are implicitly deallocated (but their definitions not purged) and must again be allocated from the stored definition before they can be used. Should the ASDS reorganize function be interrupted by a power failure or controller malfunction, the ASDS is unusable. For this reason and if sufficient space is available, the ASDS reorganize function may be accomplished with less risk by copying the records from one ASDS data set to another ASDS data set using an LREAD PLR-LWRITE PLR loop. Deleted records will be ignored when the input data set is read. However, any keyed data sets associated with the old ASDS will have to be redefined and associated with the new ASDS data set.

When a KSAP is reorganized, sufficient free space must be available on the device to build a new KSAP from the existing KSAP. The allocation of the space for the new KSAP and the deallocation of the space occupied by the existing KSAP are implicitly handled by the controller. Should the KSAP reorganize function be interrupted, the new KSAP should be deallocated by the user and the reorganize request should be resubmitted. When this interruption occurs, the data set name of the new KSAP data set is the name of the one being reorganized with a $ (dollar sign) substituted for the first character of the name.

Control Access to Volumes

Inhibit Access to Volumes

This LDKT request allows a station to request exclusive use of a volume. When this request is issued, EDAM stores the ID of the requesting station. Subsequently, until a Permit Access request is issued, EDAM accesses to the volume (by the station that requested inhibit) are allowed, but attempted accesses by other stations are rejected with appropriate status.

Before issuing LDKT, set SMSFG1 to select the appropriate drive.

The status of the EDAM buffers and the open status of any data sets on the mounted volume are not affected by the Inhibit function. EDAM monitoring of buffer usage continues normally while access is inhibited. If an operation requested by the inhibiting station requires reassignment of a buffer, and if the buffer assigned had been changed before the inhibit request, the changed sector will be written to the diskette before the buffer is reassigned.

Control operator functions and absolute requests are not affected by the Inhibit Access function.

The parameter list used for inhibiting access is one byte long and contains only the request code, as follows:

Byte	Value or Meaning
0	Request Code: hex 06 = Inhibit Access

Permit Access to Volumes

This LDKT request allows a station that had previously requested an Inhibit Access function to relinquish exclusive access. When this request is issued, EDAM clears the station ID save area, thus enabling all stations to access the volume.

Before issuing LDKT, set SMSFG1 to select the appropriate drive.

If access to the volume is not inhibited when this request is issued, the request has no effect and completes with zero status. If access to the volume is inhibited when this request is issued, this request must be issued by the same station that had previously requested the Inhibit. Otherwise, the request is rejected with appropriate status.

The Permit function does not affect the status of the EDAM buffers and the open status of any data sets on the currently mounted volume.

The parameter list used for permitting access is one byte long and contains only the request code, as follows:

Byte	Value or Meaning
0	Request Code: hex 07 = Permit Access

Close a Data Set

LDKT closes a data set from further processing by the station that requested the close. LDKT close for a keyed data set closes the keyed data set as well as implied opens for the associated unkeyed data sets.

Before this LDKT is issued, the data set must be open and the value returned in SMSDID when the data set was opened must be placed in SMSDID. The controller will use the value in SMSDID to determine the data set, device, and drive to be accessed.

If no other stations have the data set open when the close is requested and if any changed sectors of the data set are in EDAM buffers, they are written to the diskette. If the data set had been extended, the controller reads the label, updates the End-of-Data address and Offset-to-Next-Record-Space fields, and rewrites the label.

When a keyed data set is closed the average number of sector accesses required to find a record using that keyed data set is returned in SMSKEY. This value gives the application an indication of the current performance level. If the number increases significantly it may be advantageous to reorganize the keyed data set if it is a KSAP or reallocate a larger keyed data set if it is an RKAP. This may facilitate fewer accesses. It is returned as a binary integer in the high-order two bytes with a fractional value in the low-order two bytes.

The parameter list used when closing a data set is two bytes long and contains the following fields:

Byte	Value or Meaning
0	Request Code: hex 05 = Close
1	Flag byte

Bit	Meaning
0	Limited/General Close 0 = Close for requesting station only 1 = Close for All Stations
1	Error Option 0 = Report Errors 1 = Ignore Errors
2-7	Reserved, must be 0

The parameter list must be at least two bytes long, otherwise, a program check will occur when the instruction is executed.

If the Close-for-All-Stations flag is set, EDAM zeros the entire bit map for the data set. No other stations that may have had the data set open are then allowed to access the data set unless they reopen it. If a station closes a data set to all stations, the requesting station need not have the data set open. If an explicit close-for-all stations request is issued to an unkeyed data set that also has existing implied opens, then all explicit opens are ended. The implied opens in this case remain in effect and keep the data set open.

If the Close-for-All-Stations flag is set and the close request is for a keyed data set that has associated unkeyed data sets then the keyed data set is closed for all stations. The associated unkeyed data sets are also closed. However, if any of the unkeyed data sets were explicitly opened because of other requests then they will remain open.

The Error Option is ignored if this is not the final close of a data set. If Report Errors is specified and if a data buffer write error occurs during close, or if an EDAM buffer contains an unwritable block, the request is ended with appropriate status and the data set is left open.

If the Ignore Errors flag is set and if a data buffer write error occurs during close, or if any EDAM buffers contain blocks of the data set that had previously been flagged as unwritable, the close completes with zero status. Any buffers that contain unwritable blocks of the data set are freed for other use.

Deallocate a Data Set

This LDKT function deallocates a data set by freeing the space the data set occupied. Optionally, the data set definition may also be purged. For diskette, deallocation involves replacing its diskette header label with a delete control record, and optionally deleting the record in the 'SYSDSLBL' data set that contains the defined parameters for the data set.

Before issuing this LDKT, SMSFG1 must be set to select the appropriate drive.

The data set must not be open to any station when deallocation is requested, otherwise the request is rejected with appropriate status. When an ASDS or an EDDS is deallocated, all associated keyed data sets are implicitly deallocated but their definitions are retained. If your data is sensitive, you should remember that deallocating a data set does not reinitialize any of the sectors containing records of the data set.

The parameter list used when deallocating a data set contains the following fields:

Byte	Value or Meaning
0	Request Code: hex 04 = Deallocate
1	Flag byte

Bit	Meaning
0	Definition Retention Flag 0 = Purge data set definition 1 = Retain data set definition
1-7	Reserved, must be 0

2-18	Data Set Name
19-24	Current date

The parameter list must be at least 25 bytes long, otherwise a program check will occur when the instruction is executed.

The Definition Retention Flag provides for the retention of the data set definition parameters in the SYSDSLBL data set, allowing subsequent allocation from the stored definition. The request to retain a data set definition that had not been previously defined, is not valid. The Deallocate will be done, a definition will not be retained, and attention status (hex 0889) will be returned upon completion.

The Data Set name is the full 17-byte name of the data set to be deallocated, and must be given exactly as it appears in the data set header label.

The Current Date is used to check whether the data set is expired. It is given as a 6-byte EBCDIC value in YYMMDD format and is compared with the Expiration Date field in the header label. If the current date is greater than or equal to the Expiration Date, the deallocate is permitted. If the current date is less than the Expiration Date, the request is ended with appropriate status. If your facility maintains the timer function, you can obtain the current date using the LTIME or LTIMEV instruction. Refer to Volume 1 of the 4700 Controller Programming Library for additional information.

The LDKT instruction and its operands are as follows.

Name	Operation	Operand
[label]	LDKT	{ defcon2 | defld2 | (defrf2) | (reg2) | seg2 | seg2,disp2 }

operand 2
Specifies the DEFDKT parameter list to be used. If seg2 is specified, the controller uses the SFP of the segment to determine the location of the parameter list.

Note: In all cases, the length of the parameter list is determined by the request code set in the first byte of the parameter list (field DKTRCD).

Condition Code: One of the following is set:

Hex Code	Explanation
01	No status is set
02	Status is set in SMSDST

Program Checks (hex): 01, 02, or 27 can be set.

LREAD--Read from Disk or Diskette

LREAD reads either a single logical record or the contents of one or more sectors from a disk or diskette data set. LREAD can retrieve (depending on the first operand of the LREAD instruction):

- The contents of one or more sectors from either diskette drive using absolute (track, head, and sector) addressing (A operand).
- The contents of one or more sectors from disk or diskette using physical block number addressing (PBN operand).
- A logical record from the temporary file or System Log on the operating diskette mounted on the primary drive (TFn, C or L operand).
- A logical record from a data set on the secondary diskette drive or a disk drive opened with the temporary file option (TFn or C operand).
- One or more sectors from the permanent file on the operating diskette (P operand).
- One or more sectors from a data set opened without the temporary file option (P or DSID operand).
- A logical record from a data set on disk or diskette opened without the temporary file option (PLR operand).

A read operation forces the station to wait until the data transmission is completed and status is stored before execution continues with the next sequential instruction. You may use either segment-header or segment-displacement addressing. If you use segment-header addressing the data is read into the specified segment starting at the primary field pointer (PFP). The read operation ends:

- when the end of the message is reached
- when the end of the input field is reached (if the FLI is not zero and less than or equal to the length between the PFP and the end of the segment)
- when the end of the segment is reached (if the FLI is zero or greater than the length between the PFP and the end of the segment).

At the end of the operation, the PFP is unchanged, the length of the data is stored in binary in SMSIML, and the status is stored in SMSDST. If you use segment-displacement addressing, the displacement and length of the data are specified in the instruction. A condition code of hex 01 is set if status is 0. The controller sets a condition code of hex 02 if the status is not 0. If bit 3 of SMSDST is on, the status code pertains to some condition that is not related to the current LREAD. The condition prevents further data transmission, and therefore the current LREAD is not initiated. If bit 3 is not on, status pertains to the current LREAD.

If an invalid value is specified in SMSRPS, the highest valid value (in units consistent with the type of LREAD) is returned in SMSRPS and a unit exception status (hex 4000) is returned in SMSDST. The highest valid value is returned for all types of LREAD except LREAD PLR. For LREAD PLR, the highest valid value is returned for an EDDS only. For LREAD A if the track number is valid but the record number is not, then the highest valid record number for the specified track is returned.

Following an LREAD TFN, LREAD C, or LREAD PLR instruction, SMSUNK contains an identifier of the data set containing the record. SMSUNK together with SMSRPS provide a unique identifier for the record read. If the high-order four bits of SMSUNK are set to zero, the resulting SMSUNK value from LREAD PLR can be used as a relative position of the unkeyed data set label. You can use this relative position value with the LDKT Query Extended Header Label function to determine the identity of the unkeyed data set from which the record was read.

For LREAD instructions, using the A, PBN, P, or DSID operand, you can transfer multiple sectors of data if SMSMBT is set to one. SMSRPS specifies the first sector to be transferred. The specified data area length determines how many data bytes and how many sectors are read. The length also determines if the last data byte ends on a sector boundary, within a sector, beyond the end of the data set, or beyond the last sector on the disk or diskette.

For multiple-sector LREAD operation:

- If the data area ends on a sector boundary, SMSDST is set to hex 0000, and the operation completes as requested.
- If the data area is equal to 0, SMSDST is set to hex 0000, and no data transfers. SMSIML is set to the number of bytes remaining from the beginning sector through the last sector in the data set, on the disk, or on the diskette. The maximum value for SMSIML on disk is equal to FF00.
- If the data area ends within a sector, SMSDST is set to the wrong length record (hex 0101). The number of bytes transferred is equal to the length of the data area, and SMSRPS is set to the last RBN/PBN/A read.
- If the data area ends beyond the last sector of the data set or device, SMSDST is set to unit exception (hex 4000). Data bytes transfer until the last sector of the data set or device is full. SMSIML is set to the number of bytes transferred, and SMSRPS is set to the last RBN/PBN/A read.
- If the request value is 0 or invalid, SMSDST is set to unit exception (hex 4000), and no data transfers. SMSIML is set to 0 and SMSRPS is set to the maximum valid RBN/PBN/A for the data set or the last sector on the disk or diskette.
- If an error occurs during the operation, SMSDST is set to the error status (hex 0200/0204), and data transfers until the system encounters an error. SMSRPS is set to the RBN/PBN/A of the failing sector. SMSIML has no significance.

Figure 4-2 provides a summary of the input conditions and results.

Input	SMSDST Returned	SMSIML Returned	SMSRPS Returned	Data Processed
Length is multiple of sector length	0000	number of bytes requested	RBN/PBN/A last read	number of bytes requested
Length is not multiple of sector length	0101	number of bytes requested	RBN/PBN/A last read	number of bytes requested
Length is 0	0000	number of bytes to end of device or data set	no change	0 bytes
Too long for device or data set	4000	number of bytes processed	RBN/PBN/A last read	number of bytes to end of data set or device
SMSPRS equal 0 or invalid	4000	zero	maximum valid RBN/PBN/A	zero bytes
I/O error while processing	0200 0204	number of bytes to error	failing RBN/PBN/A	number of bytes to error

Figure 4-2. Multiple-Sector LREAD Operation Conditions - Disk and Diskette

LREAD A

LREAD A retrieves the contents of one or more sectors from either the primary or secondary diskette drive. The first (or only) sector is located by its absolute diskette address (track, head, and sector). Before issuing LREAD A, set:

SMSRPS
To contain the absolute address of the first or only sector to be read. The track number, in binary, is placed in the third byte; the sector number, in binary, is placed in the fourth byte. For a 2-sided diskette, set the high-order bit in the sector number to zero for the primary side and to one for the secondary side.

SMSFG1 Bit 6
To indicate which diskette drive to access.

SMSMBT
To indicate whether to read one sector, or enough sectors to fill the data area.

LREAD PBN

LREAD PBN retrieves the contents of one or more sectors from either disk or diskette. The first (or only) sector is located by its physical block number (the relative number of the sector on the device, beginning with 1). Before issuing LREAD PBN, set:

SMSRPS
To contain the physical block number (PBN) of the first or only sector to be read.

SMSFG1
To indicate which device type and drive to access (disk or diskette).

SMSMBT
To indicate whether to read one sector, or enough sectors to fill the data area.

LREAD TFn

The LREAD TFn instruction retrieves a single logical record from the specified (by *n*) file of a temporary file, and places the record in the area indicated by the second operand. LREAD TFn can read records either from the temporary file on the operating diskette, or from a direct data set on another drive opened with the temporary file option. On the operating diskette, the temporary file is implicitly opened at IPL, and continues to be accessed as the temporary file whenever the primary drive is selected and is not stopped. A temporary file data set on any other drive must be opened explicitly by the LDKT instruction, and is then accessed whenever the bits of SMSFG1 are not zero and the data set ID returned by LDKT (Open) is set in SMSDID. Before issuing LREAD TFn, set these fields:

SMSRPS
To contain the file or subfile sequence number of the record to be read, in binary.

SMSSFR
To contain the subfile ID if SMSRPS contains a subfile sequence number, or zero if SMSRPS contains a file sequence number.

SMSFG1
To indicate which temporary file is to be accessed. If you specify the primary diskette, the temporary file on the primary drive is accessed. Otherwise, the temporary file data set that is identified through the value in SMSDID is accessed. If the primary drive is stopped, the request is rejected.

SMSDID
To indicate the data set ID of a temporary file data set on any drive other than the primary diskette drive. If SMSFG1 specifies the primary diskette, this field is ignored and the temporary file on the primary drive is accessed.

LREAD C

LREAD C retrieves a single logical record from a temporary file data set using a composite subfile index, and places the record in the area specified by the second operand. LREAD C can read records from both the temporary file on the operating diskette, or a data set on another drive opened with the temporary file option. On the operating diskette, the temporary file is implicitly opened at IPL, and continues to be accessed as the temporary file whenever the primary drive is selected and is not stopped. A data set on any other drive must be opened (with the temporary file option) explicitly by the LDKT instruction, and is then accessed whenever the bits of SMSFG1 are not zero and the data set ID returned by LDKT (Open) is set in SMSDID. Before issuing LREAD C, set:

SMSRPS
To contain the composite subfile sequence number of the record to be read, in binary.

SMSSFR
To contain the subfile ID.

SMSFG1
To indicate which temporary file is to be accessed. If you specify the primary diskette, the temporary file on the primary drive is accessed. Otherwise, the temporary file data set that is identified through the value in SMSDID is accessed. If the primary drive is stopped, the request is rejected.

SMSDID
To indicate the data set ID of a temporary file data set on any drive other than the primary diskette drive. If SMSFG1 specifies the primary diskette, this field is ignored and the temporary file on the primary drive is accessed.

LREAD L

LREAD L retrieves a single logical record from the system log on the operating diskette. There is only one system log on the operating diskette. Before issuing LREAD L, set:

SMSRPS
To contain the log record number in binary.

LREAD P

LREAD P retrieves the contents of one or more sectors from either the permanent file on the operating diskette, or from a direct data set opened on any other drive without the temporary file option. The first or only sector to be read is located by its number relative to the start (sector 1) of the permanent file or data set. Before issuing LREAD P, set:

SMSRPS
To contain the sector number of the first or only sector to be read, relative to the beginning of the permanent file or data set.

SMSFG1
To indicate whether the permanent file on the primary drive or another data set is to be accessed. If you specify the primary diskette, the permanent file on the primary drive is accessed. Otherwise, the data set that is identified through the value in SMSDID is accessed. If the primary drive is stopped, the request is rejected.

SMSDID
To indicate the data set ID of a data set on any drive. If SMSFG1 specifies the primary diskette, this field is ignored and the permanent file on the primary drive is accessed.

SMSMBT
To indicate whether to read one sector, or enough sectors to fill the data area.

LREAD DSID

LREAD DSID retrieves the contents of one or more sectors from a data set opened on any drive without the temporary file option. The first or only sector to be read is located by its number relative to the start (sector 1) of the data set. Before issuing LREAD DSID, set:

SMSRPS
To contain the sector number of the first or only sector to be read, relative to the beginning of the data set.

SMSDID
To contain the data set ID of a data set opened without the temporary file option.

SMSMBT
To indicate whether to read one sector, or enough sectors to fill the data area.

LREAD PLR

LREAD PLR retrieves a single logical record from data sets on any drive. For a sequential data set, LREAD PLR reads the record pointed to by the current input pointer. For a direct data set, LREAD PLR reads the record identified by SMSRPS. For a keyed data set, LREAD PLR reads the direct data set record found through the keyed data set as specified in SMSRPS, SMSKEY, and the application program data area.

LREAD PLR ESDS: Before issuing LREAD PLR for a sequential (ESDS) data set, set:

SMSDID
To contain the data set ID.

LREAD PLR EDDS or ASDS: Before issuing LREAD PLR for a direct (EDDS or ASDS) data set, set:

SMSRPS
To contain the record sequence number, in binary, for an EDDS data set or the record sector number and offset, in binary, for an ASDS data set.

A value of +1 causes the first available record in the data set to be read (deleted records in an ASDS data set are not available). If the high-order bit (bit 0) of SMSRPS is on, the record following the record identified through the remainder of SMSRPS is read.

For an EDDS data set, if the third bit (bit 2) of SMSRPS is on, the last record in the data set is read. If the second bit (bit 1) of SMSRPS is on, the record preceding the record identified through the remainder of SMSRPS is read.

If the value is zero or is beyond the end of the data set for an EDDS data set, SMSRPS is set to the last record in the data set. For either an EDDS or an ASDS, if the value is not within the data set, an end-of-data-set status is returned in SMSDST.

SMSDID
To contain the data set ID.

LREAD PLR RKAP or KSAP: Before issuing LREAD PLR for a keyed (RKAP or KSAP) data set, set:

SMSRPS
To zero to cause the unkeyed data set record corresponding to the key residing at the proper offset in the application program data area to be read. If no such record is referenced through the keyed data set, a record-not-found status is returned in SMSDST. The position within the keyed data set is returned in SMSKEY such that (primarily for a KSAP data set) the data set may be processed sequentially from that position. For a KSAP data set only, a duplicate-key-follows status is returned in SMSDST if one or more subsequent records having the same key can be accessed through the data set.

A value of +1 causes the first record in the data set to be read. If the high-order bit (bit 0) of SMSRPS is on, the next record following the record identified through SMSKEY is read.

For a KSAP data set, if the third bit (bit 2) of SMSRPS is on, the last record in the data set is read. If the second bit (bit 1) of SMSRPS is on, the record preceding the record identified through SMSKEY is read. When a KSAP is read sequentially (forward or backward), the records are presented in logical sequential order based on the value or their keys. Records containing duplicate key values are presented in first-in-first-out order when the data set is read in the forward direction. They are presented in last-in-first-out order when the data set is read in the backward direction. When the data set is read in the forward direction only, a duplicate key status is returned if one or more subsequent records has the same key as the current record.

When an RKAP data set is read sequentially (forward only), the records are presented in the order they happen to reside in the hash table index (no logical order).

SMSDID
To contain the data set ID.

SMSKEY
To contain the value returned by a prior LREAD PLR operation, for processing sequentially through the RKAP or KSAP data set (ignored if SMSRPS contains zero or +1, for keyed access).

Name	Operation	Operand
[label]	LREAD	{ TFn | C | P | PLR | L | A | DSID | PBN } { defld2 | (defrf2) | (reg2) | seg2 | seg2,disp2,len2 }

A
Indicates absolute (track and sector) addressing (diskette only).

PBN
Indicates physical block number addressing.

TFn
Indicates one of the temporary files, where n is the number of the file (from 1 to 4).

C
Indicates the composite file.

L
Indicates the System Log.

P

Indicates the permanent file or sectors from a data set.

DSID

Permits access to sectors of a data set on any drive.

PLR

Indicates logical records are to be read from a data set.

operand 2

Is the operand to contain the data to be read. Do not specify Segment 14.

Condition Codes: One of the following is set:

Hex Code	Explanation
01	The instruction was executed successfully.
02	Status is returned in SMSDST.

Program Checks (hex): 01, 02, or 27 can be set.

LWRITE--Write to Disk or Diskette

LWRITE writes a single record to a disk or diskette. LWRITE can write (depending on the first operand of the LWRITE instruction):

- A logical record to the temporary file on the operating diskette mounted on the primary drive (TFn operand)
- A logical record to a direct data set on the secondary diskette drive or a disk drive opened with the temporary file option (TFn operand)
- A logical record to the System Log on the operating diskette mounted on the primary drive (L operand)
- A logical record to a data set on disk or diskette opened without the temporary file option (PLR operand).

You may use either segment-header or segment-displacement addressing. If you use segment-header addressing, the data is written from the segment specified in the instruction, starting at the location pointed to by the secondary field pointer (SFP) up to, but not including, the location pointed to by the primary field pointer (PFP). If you use segment-displacement addressing, the displacement to the data and the length of the data are as specified in the instruction.

A write operation does not, in general, cause the data to be written to the device. The data may remain in a buffer in controller storage until the buffer is full; until an LCHECK instruction is issued; or until the data set is closed for all stations having it open. Note than an implied LCHECK is always performed for System Log records, causing an immediate write of the diskette buffer.

At the completion of an LWRITE instruction, SMSDST contains a status code. A condition code of hex 01 is set if status is zero. A condition code of hex 02 is set if the status is nonzero. If bit 3 of SMSDST is on, the status code pertains to some condition that is not related to the current LWRITE. The condition prevents further data transmission, and therefore the current LWRITE is not initiated. If this bit is not on, the status bits pertain to the current LWRITE.

At the completion of an LWRITE instruction, SMSIML contains the absolute value of the difference between the amount of data written and the record length.

LWRITE TFn

LWRITE TFn writes a single logical record either to the temporary file on the operating diskette, or to a data set on another drive opened with the temporary file option. Before issuing LWRITE TFn, set:

SMSSFW
To contain the subfile ID. Set this field to 0 if the record is not to be treated as part of a subfile.

SMSFG1
To indicate the temporary file data set to contain the record. If you specify the primary diskette, the temporary file on the primary drive is selected. If the primary drive is selected, it must *not* be in the stopped state (a write to the primary drive in the stopped state is rejected). If the bits of SMSFG1 are not zero, the temporary file data set that is identified through the value in SMSDID is selected.

SMSDID
To contain the ID of a data set opened with the temporary file option. If SMSFG1 specifies the primary diskette, this field is ignored.

On completion of LWRITE TFn, these fields are set:

SMSFSN
Contains the file sequence number of the stored record if the write is accepted.

SMSSSN
Contains the subfile sequence number of the stored record, if:

- The write was accepted.
- SMSSFW was not 0.
- Subfile indexing was requested during configuration for the subfile ID in SMSSFW.

SMSCSN
Contains the composite subfile sequence number, if:

- The write was accepted.
- SMSSFW was not zero.
- The specified file was defined as part of a composite file and composite subfile indexing for the subfile ID in SMSSFW was requested during configuration.

If the length of the record to be written is zero, a two-byte null record is written with no following text. No status is returned. Sequence numbers are returned, and SMSIML is set to zero.

The maximum record length is 252 bytes. If an attempt is made to write a longer record, only the first 252 bytes are written. A condition code of hex 02 is set and a status code indicating that the record is too long is returned in SMSDST. The difference between the record length and 252 is returned in SMSIML.

LWRITE L

LWRITE L always writes a single logical record to the system log on the operating diskette mounted on the primary drive. SMSSFW, SMSFG1, and SMSDID are ignored; SMSFSN is updated; SMSSSN and SMSCSN are set to zero.

If the length of the record to be written is zero, a two-byte null record is written with no following text. No status is returned. A file sequence number is returned, and SMSIML is set to zero.

The maximum record length is 252 bytes. If an attempt is made to write a longer record, only the first 252 bytes are written; but a condition code of hex 02 is set and a status code indicating that the record is too long is returned in SMSDST. The difference between the record length and 252 is returned in SMSIML.

Note: If the second byte of the data written to the System Log contains a hex F1, the CHECK light on the control operator's 4704 is turned on.

LWRITE PLR

LWRITE PLR writes a single logical record to a data set opened, without the temporary file option, on any drive. If required, LWRITE PLR assigns the next unused sector in the data set. If necessary on disk, and if a secondary extent is allowed and available, it will be allocated to the data set. For a sequential or direct data set, LWRITE PLR writes the record at the end of the data set. For a keyed data set, LWRITE PLR writes the record at the end of the last allocated unkeyed data set associated with the keyed data set.

If a keyed data set excludes duplicate keys, an attempt to add a record through that keyed data set (having the same key as a record already existing in the data set) ends without adding the record, but with a duplicate key status. An attempt to update a keyed data set that excludes duplicate keys by writing a duplicate key record to an unkeyed data set to which it refers, results in the record not being referred to through the keyed data set. A secondary index exception status is returned in this case.

If the record is added to an EDDS data set, the relative number of the new record (relative to the start of the data set) is returned in SMSRPS. Similarly, if the record is added to an ASDS data set, a binary value indicating the relative sector number where the record starts and the byte offset of the record within that sector is returned in SMSRPS. This value can later be used with an LREAD PLR or REPLACE PLR instruction to retrieve or modify the record. Reorganizing an ASDS would invalidate any SMSRPS value saved for records in the ASDS. For a sequential data set, SMSRPS is set to zero.

Before issuing LWRITE PLR, set:

SMSDID
To contain the ID of a data set opened without the temporary file option.

If the data length is less than the defined record length, the data to be written is padded with trailing binary zeros, status is returned indicating wrong-length record, and SMSIML contains the difference between the record length and the data length. If the data length is greater than the record length, the leading portion of the data is written, status is returned indicating wrong-length and over-sized record, and SMSIML contains the difference between the data length and the record length. If the data length equals the record length, no status is returned and SMSIML is set to zero.

After LWRITE PLR or LWRITE TFn is issued, SMSUNK contains a unique identifier for the unkeyed data set containing the record. If the record is accessed through a keyed data set, SMSKEY contains the position in the keyed data set of the entry for the record that was added. These fields are generally not used by an application program after issuing an LWRITE PLR instruction.

Name	Operation	Operand
[label]	LWRITE	{ TFn / L / PLR } , { defcon2 / defld2 / (defrf2) / (reg2) / seg2,disp2,len2 / seg2 }

TFn

Specifies a write to one of the temporary files, where n is the number of the file (from 1 to 4).

L

Specifies a write to the System Log.

PLR

Specifies that a logical record be written to a data set.

operand 2

Refers to the data to be written. When you specify *seg2*, the SFP must point to the start of the field, and the PFP must point 1 byte past the end of the field.

When you code *seg2* as the second operand, a 2-byte machine instruction is generated; but when you code *defld2* or *seg2,disp2,len2*, the machine instruction is 6 bytes long.

Condition Codes: One of the following is set:

Hex Code	Explanation
01	The write operation was successful.
02	Status is returned. Status is stored in SMSDST.

Program Checks (hex): 01, 02, 03 or 27 can be set.

REPLACE--Replace Disk or Diskette Data

REPLACE replaces either a single logical record or the contents of one or more sectors on a disk or diskette data set. Depending on the first operand of the REPLACE instruction, REPLACE can replace:

- The contents of one or more sectors on either diskette drive using absolute addressing (track, head, and sector). Single sectors replaced this way can be written as either data records or control records (A or CR operand).
- The contents of one or more sectors on disk or diskette using physical block number addressing (PBN operand).
- A logical record in the temporary file on the operating diskette mounted on the primary drive (TFn or C operand).
- A logical record in a data set opened with the temporary file option on the secondary diskette drive or disk (TFn or C operand).
- The contents of one or more sectors in the permanent file on the operating diskette mounted on the primary drive (P operand).
- The contents of one or more sectors of a data set opened without the temporary file option (P or DSID operand).
- The contents of a logical record of a data set opened without the temporary file option (PLR operand).

You may use either segment-header or segment-displacement addressing. If you use segment-header addressing, the data is written from the segment specified in the instruction, starting at the location pointed to by the secondary field pointer (SFP) up to, but not including, the location pointed to by the primary field pointer (PFP). If you use segment-displacement addressing the displacement to the data and the length of the data are as specified in the instruction.

A replace operation forces the station to wait until the data transmission is completed and status is stored before execution continues with the next sequential instruction.

On completion of a REPLACE instruction for a single record (TFn, C, or PLR operand), SMSIML contains the absolute difference between the record length and the data length. If the replacement record is shorter than the original record, only the leftmost portion of the original record is replaced, and a status code indicating an incorrect-length record is returned. If the replacement record is longer than the original record, the rightmost portion of the replacement record is truncated and both the incorrect-length and over-sized record indicators are returned. If the replacement length is the same as the length of the original record, SMSIML is set to zero.

If you specify an invalid value in SMSRPS, the highest valid value (in units consistent with the type of REPLACE) is returned in SMSRPS and a unit exception status (hex 4000) is returned in SMSDST. The highest valid value is returned for all types of REPLACE except REPLACE PLR. For REPLACE PLR, the highest valid value is returned for an EDDS only. For REPLACE A if the track number is valid but the record number is not, then the highest valid record number for the specified track is returned.

For REPLACE instructions, using the A, PBN, P, or DSID operand, you can transfer multiple sectors of data if SMSMBT is set to one. SMSRPS specifies the first sector to be transferred. The specified data area length determines how many data bytes and how many sectors are written. The length also determines if the last data byte ends on a sector boundary, within a sector, beyond the end of the data set, or beyond the last sector on the disk or diskette.

For multiple-sector REPLACE operation:

- If the data area ends on a sector boundary, SMSDST is set to hex 0000, and the operation completes as requested.
- If the data area is equal to 0, SMSDST is set to hex 0000 and no data transfers. SMSIML is set to the number of bytes remaining from the beginning sector through the last sector in the data set, on the disk, or on the diskette. The maximum value for SMSIML on disk is equal to FF00.
- If the data area ends within a sector, SMSDST is set to the wrong length record (hex 0100). Binary zeros pad the remaining bytes in the last sector.
- If the data area ends beyond the last sector of the data set or device, SMSDST is set to unit exception (hex 4000). Data bytes transfer until the last sector of the data set or device is full. SMSIML is set to the number of bytes transferred, and SMSRPS is set to the last RBN/PBN/A written.
- If the requested SMSRPS value is 0 or invalid, SMSDST is set to unit exception (hex 4000), and no data transfers. SMSIML is set to 0 and SMSRPS is set to the maximum valid RBN/PBN/A for the data set or the last sector on the disk or diskette.
- If an error occurs during the operation, SMSDST is set to the error status (hex 0200), and data transfers until the system encounters an error. SMSRPS is set to the RBN/PBN/A of the failing sector. SMSIML has no significance.
- If an error occurred previously, bit 3 of the SMSDST is set on (prior op hex 1xxx). The operation does not begin.

Figure 4-3 on page 4-55 provides a summary of the input conditions and results.

Input	SMSDST Returned	SMSIML Returned	SMSRPS Returned	Data Processed
Length is multiple of sector length	0000	zero	RBN/PBN/A last written	number of bytes requested
Length is not multiple of sector length	0100	padding length in last sector	RBN/PBN/A last written	number of bytes requested
Length is 0	0000	number of bytes to end of device or data set	no change	0 bytes
Too long for device or data set	4000	number of bytes to end of device or data set	RBN/PBN/A last written	number of bytes to end of data set or device
SMSPRS equal 0 or invalid	4000	zero	maximum valid RBN/PBN/A	zero bytes
I/O error while processing	0200 0204	number of bytes to error	failing RBN/PBN/A	number of bytes to error
Prior I/O error	1xxx	zero	no change	zero

Figure 4-3. Multiple-Sector REPLACE Operation Conditions - Disk and Diskette

REPLACE A

REPLACE A replaces the contents of one or more sectors on either diskette drive using absolute (track, head, and sector) addressing. Before issuing REPLACE A, set:

SMSRPS
To contain the absolute track, head, and sector numbers. The track number, in binary, is placed in the third byte; the sector number, in binary, is placed in the fourth byte. For a 2-sided diskette, set the high-order bit in the sector number to zero for the primary side and to one for the secondary side.

SMSFG1 Bit 6
To indicate the primary drive (0) or the secondary drive (1). If you select the primary drive, it must be in the stopped state.

SMSMBT
To indicate whether to replace one or more sectors.

REPLACE CR

REPLACE CR replaces the contents of a single diskette sector with a control record, using absolute addressing. Before issuing REPLACE CR, set:

SMSRPS
To contain the absolute track, head, and sector numbers. The track number, in binary, is placed in the third byte; the sector number, in binary, is placed in the fourth byte. For a 2-sided diskette, set the high-order bit in the sector number to zero for the primary side and to one for the secondary side.

SMSFG1 Bit 6
To indicate the primary drive (0) or the secondary drive (1). If you select the primary drive, it must be in the stopped state.

REPLACE PBN

REPLACE PBN replaces the contents of one or more sectors on any disk or diskette. The first (or only) sector is located by its physical block number. Before issuing REPLACE PBN, set:

SMSRPS
To contain the physical block number of the first or only sector to be replaced.

SMSFG1
To indicate the device type and drive (disk or diskette).

SMSMBT
To indicate whether to replace one or more sectors.

REPLACE TFn

REPLACE TFn replaces a single logical record in the specified file of the temporary file on the operating diskette, or in a data set opened on another drive with the temporary file option. On the operating diskette, the temporary file is implicitly opened at IPL and continues to be accessed as the temporary file whenever the nonstopped primary device is selected. A data set on another drive must be opened with the temporary file option and is selected when the data set ID is set in SMSDID. Before issuing REPLACE TFn, set:

SMSRPS
To contain the file or subfile sequence number of the record to be replaced.

SMSSFR
To contain the subfile ID if SMSRPS contains a subfile sequence number, or zero if SMSRPS contains a file sequence number.

SMSFG1
To indicate which temporary file is involved. If you specify the primary diskette, the record is in the temporary file on the primary diskette drive. In this case, the primary diskette drive must be in a stopped state. If the bits of SMSFG1 are not zero, the record is in the data set that is identified through the value contained in SMSDID.

SMSDID
To contain the ID of a data set opened with the temporary file option. If SMSFG1 specifies the primary diskette, this field is ignored.

REPLACE C

REPLACE C replaces the contents of a temporary file record using a composite subfile index. It can access either the temporary file on the operating diskette, or a data set on another drive opened with the temporary file option. Before issuing REPLACE C, set:

SMSRPS
To contain the composite subfile sequence number, in binary, of the record to be replaced.

SMSSFR
To contain the subfile ID.

SMSFG1
To indicate which temporary file is involved. If you specify the primary diskette, the record is in the temporary file on the primary diskette drive. In this case, the primary diskette drive must be in the stopped state. If the bits of SMSFG1 are not zero, the record is in the data set that is identified through the value contained in SMSDID.

SMSDID
To contain the ID of a data set opened with the temporary file option. If SMSFG1 specifies the primary drive, this field is ignored.

REPLACE P

REPLACE P replaces the contents of one or more sectors in either the permanent file on the operating diskette mounted on the primary diskette drive, or in a direct data set opened without the temporary file option. Before issuing REPLACE P, set:

SMSRPS
To contain the sector number of the first or only sector to be replaced, relative to the beginning of the file or data set (relative numbers begin with 1).

SMSFG1
To indicate which data set is involved. If you specify the primary diskette, one or more sectors in the permanent file on the primary diskette drive are replaced. In this case, the primary diskette may not be in the stopped state. If the bits of SMSFG1 are not zero, the data set is identified through the value in SMSDID.

SMSDID
To contain the data set ID of a data set opened without the temporary file option.

SMSMBT
Indicates whether to replace a single sector, or several sectors.

REPLACE DSID

REPLACE DSID replaces the contents of one or more sectors in a direct data set opened without the temporary file option. Before issuing REPLACE DSID, set:

SMSRPS
To contain the sector number of the first or only sector to be replaced, relative to the beginning of the file or data set (relative numbers begin with 1).

SMSDID
To contain the data set ID of a data set opened without the temporary file option.

SMSMBT
Indicates whether to replace a single sector, or several sectors.

REPLACE PLR

REPLACE PLR replaces a logical record in a data set opened without the temporary file option. An unkeyed data set record may be replaced directly (by specifying the data set ID of an unkeyed data set in SMSDID) or indirectly (by specifying the data set ID of a keyed data set in SMSDID). The unkeyed record may reside in either an ASDS or an EDDS data set. The keyed data set may be either an RKAP or a KSAP data set. If the record is replaced indirectly, the keyed data set specified through SMSDID is referred to as the **primary** index. Other keyed data sets that are associated with the data set containing the record being replaced are referred to as **secondary** indexes.

If an unkeyed record is replaced for which one or more secondary indexes exist and if the key fields for those indexes are changed, the operation is equivalent to a delete and a write as far as the secondary indexes are concerned (although the record is replaced within the unkeyed data set). In this way, replacing a record with a changed key field will be properly reflected for future access to the record through a secondary index.

If duplicate keys result from the update and they are allowed within the keyed data set, the updated record will become the last record having that key. If duplicate keys are not allowed, the record would be deleted from the secondary index data set but would not be written to it. The result is that the keyed data set no longer refers to the replaced record. A secondary index exception status is returned in this case. The key field for a primary index data set may not change when replacing a record indirectly through a keyed data set (the key field is the means by which the record to be replaced is located).

Before issuing REPLACE PLR for a direct (unkeyed) data set, set:

SMSRPS
To contain the binary value returned by LWRITE PLR or LREAD PLR for the record to be replaced.

SMSDID
To contain the data set ID of a unkeyed data set opened without the temporary file option.

Before issuing REPLACE PLR for a keyed data set, set:

SMSRPS
To contain zero, to indicate keyed reference.

SMSDID
To contain the data set ID of a keyed data set opened without the temporary file option.

Name	Operation	Operand

```
[label] REPLACE { A    }   , { defcon2          }
                { CR   }     { defld2           }
                { PBN  }     { (defrf2)         }
                { TFn  }     { (reg2)           }
                { C    }     { seg2,disp2,len2  }
                { P    }     { seg2             }
                { DSID }
                { PLR  }
```

A

Specifies that a sector is to be addressed using the absolute track address. This operand can be used only on stopped diskettes.

CR

Specifies that a control sector is to be written using the absolute track address. This operand can be used only on stopped diskettes.

PBN

Replaces contents of one or more disk or diskette sectors using PBN addressing. For diskette processing, the diskette must be stopped.

TFn

Specifies that a record in one of the temporary files is to be replaced, where *n* is the number of the file (from 1 to 4).

C

Specifies that a record in the composite file is to be replaced.

P

Replaces one or more sectors of the permanent file on the operating diskette or of a data set on another drive opened without the temporary file option.

DSID

Replaces one or more sectors of a data set opened without the temporary file option.

PLR

Specifies that a logical record is to be replaced in a direct data set opened without the temporary file option. The data set may be referred to either directly or indirectly through a keyed data set.

operand 2

Refers to the data that is to replace the record. When you specify *seg2*, the SFP must point to the start of the field, and the PFP must point 1 byte past the end of the field.

When you code *seg2* as the second operand, a 2-byte machine instruction is generated; when *defld2* or *seg2,disp2,len2* is coded, the machine instruction is 6 bytes long.

Condition Codes: One of the following is set:

Hex Code	Explanation
01	The replacement operation was successful.
02	Status is stored. Status is stored in SMSDST. (See Appendix D. "Status Codes for Disk and Diskette" on page D-1 for an explanation of status codes.)

Program Checks (hex): 01, 02, or 27 can be set.

SETDSKT--Reset the Temporary File

The SETDSKT instruction allows the controller application program to reset the temporary file on the operating diskette to zero records, at the same time updating the session identifier by one. It also allows the application program to IPL the controller, and to specify the type of start that should take place on the next or on all subsequent automatic starts, or to specify a dump instead of IPL if the system fails. An automatic start occurs when the time-out value has been exceeded (see the TIMEOUT operand of the STARTGEN configuration macro, in Volume 6 of the *4700 Controller Programming Library*).

SETDSKT sets any disk or diskette drive to the not-ready condition (for example, to prevent access to the primary diskette's temporary file during a controlled shutdown). SETDSKT can also reset any drive to the ready condition. When the drive is in the not-ready condition, neither the controller nor any application program can access the drive.

SETDSKT requires optional module P5E to be included in the controller configuration in order to use the instruction. SETDSKT points to a 2-byte parameter list that describes the action the controller program is to take.

Note: Resetting a temporary file containing log messages may cause messages that would be reported by CNM/CS to the host to be lost.

Name	Operation	Operand
[label]	SETDSKT	{ defcon2 | defld2 | (defrf2) | (reg2) | seg2,disp2 }

operand 2

Defines the start of the parameter list. The length associated with this operand is ignored, and the first 2 bytes are assumed to be the parameter list.

Byte	Meaning
0	Is a 1-byte hexadecimal value that indicates the type of request: 01 = Reset the temporary file and update the session identifier. The value byte is ignored. 02 = Use the prompt value specified in the value field on the next auto-IPL only. No time-out occurs and no operator response is permitted. 04 = Use the prompt value specified in the value field on all subsequent automatic starts. Automatic start occurs after the time-out value is exceeded. 08 = Perform a dump if a system failure occurs. The value field is ignored. 10 = Reload the system after a system failure occurs. The value field is ignored. 20 = Set the drive to the ready condition. 40 = Set the drive to the not-ready condition. 80 = IPL the controller immediately. This is the same as pressing the Reset button on the controller.

Notes:

1. Status (hex 0440) will be returned if the diskette is in stopped state when you use the following codes: hex 01, 02, 04, 08 and 10. If the drive select bits are not valid, status (hex 0440) is returned. If you use more than one code, the order of execution is from the lowest to the highest code. For example, if you were to code hex 83, the hex 01 request would be processed first, followed by the hex 02 request, and the hex 80 request last.

2. Drive select bits (SMSFG1) need only be set for parameters 20 (Ready condition) and 40 (Not-ready-condition). All other parameters will go to primary drive. In addition, if parameter 20 or 40 is used in combination with other SETDSKT parameters, the drive select bits will be ignored and the operation will go to the primary drive.

Byte	Meaning
1	Is a 1-byte EBCDIC value the system is to use for the response to the 00001 startup message on an automatic start. See IBM 4700 Subsystem Operating Procedures: GC31-2032 for all valid responses to the 00001 message and their functions.

This field is not validated at the time a SETDSKT instruction is issued; therefore, if an invalid value is specified, a 90050 message is issued to the control operator during the next startup. The prompt value, for either hex 02 or hex 04 code, may be reset by issuing another hex 02 or hex 04 SETDSKT instruction with a prompt value of hex 00.

Condition Codes: One of the following is set:

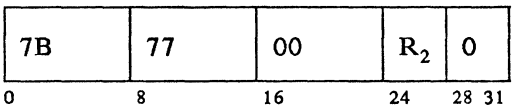
Hex Code	Explanation
01	Request was completed successfully.
02	Request was not completed because of an error on the diskette. Status is stored in SMSDST.

Program Checks (hex): 01, 02, 09, or 27 can be set.

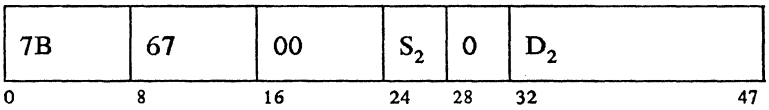
Appendix A. Machine Instruction Formats

This appendix describes the machine formats for the 4700 assembler instructions included in this volume. See *Volume 1: General Controller Programming,* for an explanation of the symbols used in this appendix.

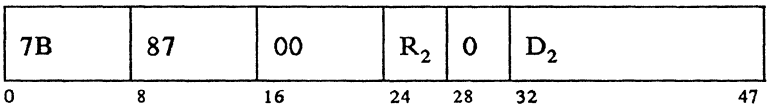
COMPDKT



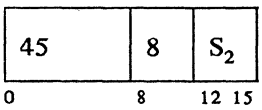
COMPDKT



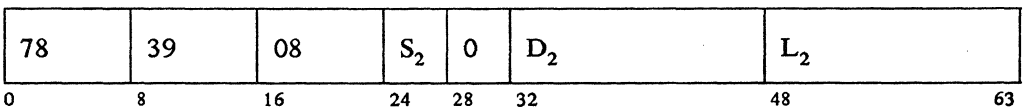
COMPDKT



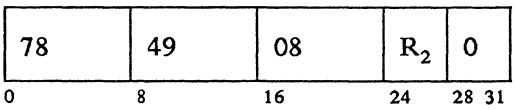
DELETE



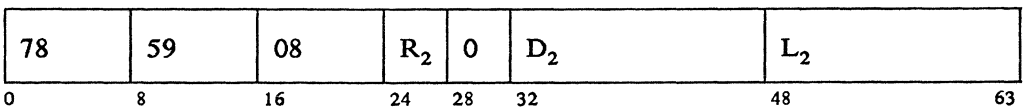
DELETE



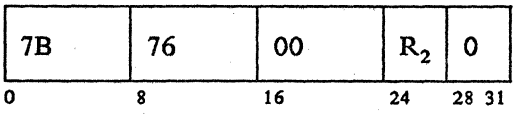
DELETE



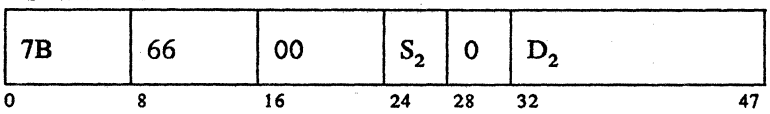
DELETE



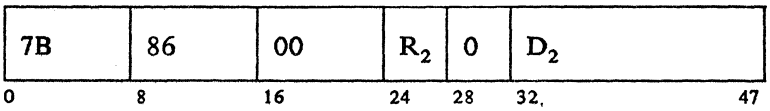
FORMDKT



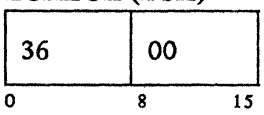
FORMDKT



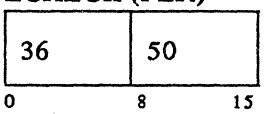
FORMDKT



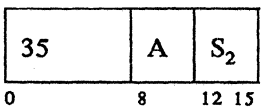
LCHECK (DSK)



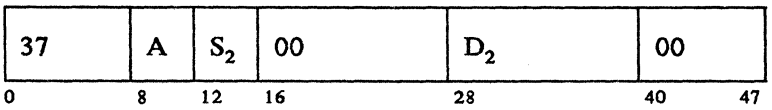
LCHECK (PLR)



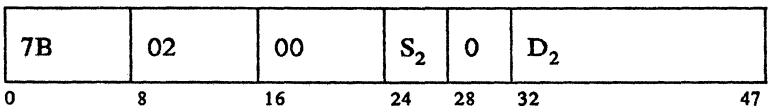
LDKT



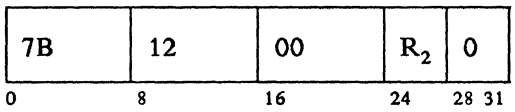
LDKT



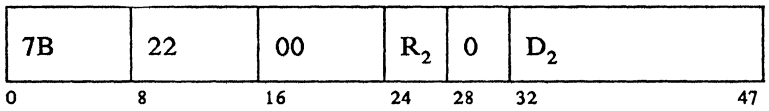
LDKT



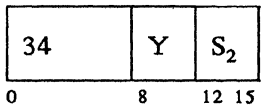
LDKT



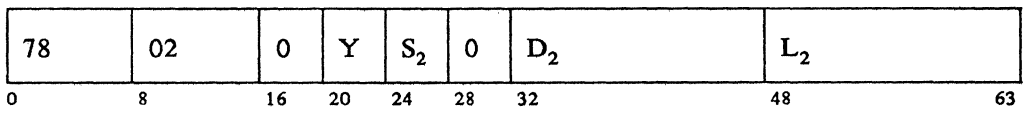
LDKT



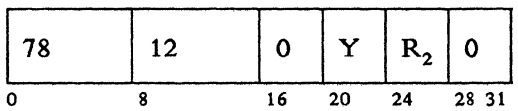
LREAD



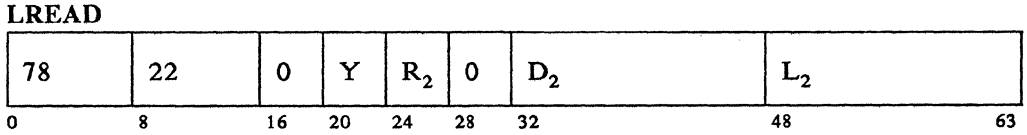
LREAD



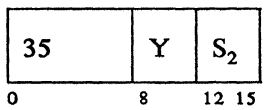
LREAD



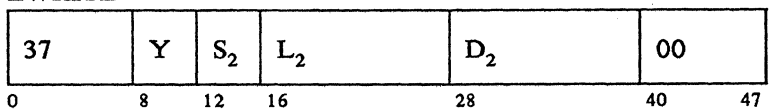
LREAD



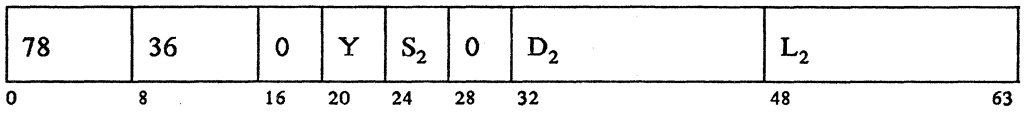
LWRITE



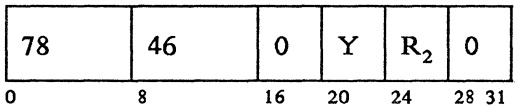
LWRITE



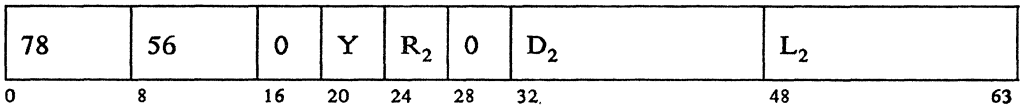
LWRITE



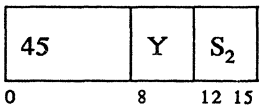
LWRITE



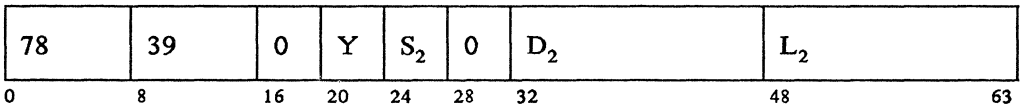
LWRITE



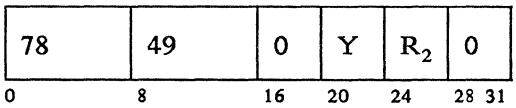
REPLACE



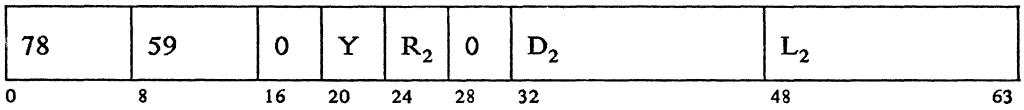
REPLACE



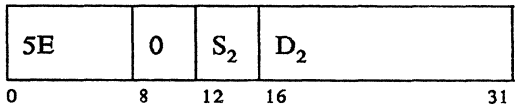
REPLACE



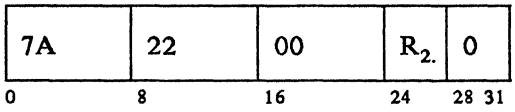
REPLACE



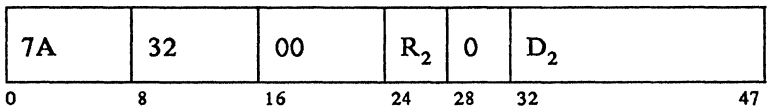
SETDSKT



SETDSKT



SETDSKT



Appendix B. 4700 COPY Files

This appendix gives detailed listings of the system definitions that are appropriate for this book.

Copy files may be included in your program by coding a COPY instruction specifying one of the copy file names in this appendix. For example:

```
COPY   DEFCDK
```

You *must* code either an EQUATE or an LDSECT instruction *before* the COPY instruction to define a segment or register number for the following copy files:

- DEFCDK
- DEFDKT
- DEFESP
- DEFFDK

If you code an EQUATE instruction, for example:

```
DEFCDKS  EQUATE n
```

then the copy file will contain a series of DEFLD instructions. The number that you specify in the EQUATE instruction will become the segment number (the first operand) of each DEFLD.

Note: The segment number must be equated to a specific label; these labels are identified for each copy file.

If you code an LDSECT instruction and specify the BASE= operand, for example:

```
LDSECT BASE=n
COPY   DEFCDK
LEND
```

then the copy file will contain a series of DEFRF instructions. The number that you specify in the BASE= operand of the LDSECT instruction will become the register number (the first operand) of each DEFRF.

If you code an LDSECT and the BASE= operand before the COPY DEFAPB instruction, then the DEFAPB copy file will contain DEFRF instructions and the register number will be as specified by the BASE= operand. Otherwise the DEFAPB copy file will contain DEFLD instructions and the segment number will be 14.

The following copy files always become part of the segment specified:

Copy File	Segment
DEFAPB	14
DEFGMS	15
DEFSMS	1

DEFCDK Compress Diskette Parameter List

Equate DEFCDKS to a segment number.

DEFCDK

```
* * *    'COMPDKT' INSTRUCTION DATA SET MODIFICATION ENTRY

CDKENT   DEFxx   s,22          COMPRESS DISKETTE REQUEST ENTRY
CDKDSN   DEFxx    CDKENT,17    DATA SET NAME
CDKFLG   DEFxx   s,1           FLAG BYTE:
CDKFL0M  EQUATE    X'80'                 TRUNCATE DATA SET
CDKFL1M  EQUATE    X'40'                 SET EOD EQUAL TO EOE
CDKFL2M  EQUATE    X'20'                 MAINTAIN TRACK ALIGNMENT
CDKFL3M  EQUATE    X'10'                 DELETE DATA SET
CDKNUM   DEFxx   s,4           BINARY SECTOR COUNT
```

DEFDKT LDKT Parameter List

Equate DEFDKTS to a segment number.

DEFDKT

```
*************************************************************************
*                                                                       *
*        STANDARD FIELD DEFINITIONS FOR THE 'LDKT' PARAMETER LISTS      *
*                                                                       *
*************************************************************************


         BQKLDS  DEFDKTS,TYPE=START
DKTRCD   DEFXX    ,1           REQUEST CODE
DKTRADM  EQUATE    X'01'                 ALLOCATE DATA SET
DKTRUDM  EQUATE    X'02'                 UPDATE HEADER LABEL
DKTRODM  EQUATE    X'03'                 OPEN DATA SET
DKTRDDM  EQUATE    X'04'                 DEALLOCATE DATA SET
DKTRCDM  EQUATE    X'05'                 CLOSE DATA SET
DKTRIAM  EQUATE    X'06'                 INHIBIT ACCESS TO VOLUME
DKTRPAM  EQUATE    X'07'                 PERMIT ACCESS TO VOLUME
DKTRBQM  EQUATE    X'08'                 BUFFER INQUIRY
DKTRBRM  EQUATE    X'09'                 BUFFER RELEASE
DKTRVQM  EQUATE    X'0A'                 VOLUME ID INQUIRY
DKTROQM  EQUATE    X'0B'                 OPEN STATUS INQUIRY
DKTRRSM  EQUATE    X'0C'                 RESET DATA SET INPUT POINTER
DKTRQSM  EQUATE    X'0D'                 UNALLOCATED SPACE INQUIRY
DKTRRGM  EQUATE    X'0E'                 REORGANIZE ASDS OR KSAP
DKTRQLM  EQUATE    X'0F'                 QUERY EXTENDED LABEL INFO
DKTRRNM  EQUATE    X'10'                 RENAME DATA SET
DKTRROM  EQUATE    X'11'                 RESET OUTPUT POINTER
DKTLST   DEFXX    ,0           BEGINNING OF PARAMETER LIST
         LSPACE
* ALLOCATE DATA SET ( REQUEST CODE: DKTRADM )
         LSPACE
```

```
DKT1F1   DEFXX    DKTLST,1    FLAG BYTE
DKT1FBM  EQUATE    X'80'                ALLOCATE ON TRACK BOUNDARY
DKT1FTM  EQUATE    X'40'                EXTENT SIZE GIVEN IN TRACKS
DKT1FKM  EQUATE    X'20'                EXTENT SIZE GIVEN IN K BYTES
DKT1ES   DEFXX    ,2          EXTENT SIZE (TRKS, SCTS, KBYTS)
DKT1NA   DEFXX    ,2          NUMBER OF SECTORS TO ASSIGN
DKT1NI   DEFXX    ,2          NUMBER OF SECTORS TO INITIALIZE
DKT1IV   DEFXX    ,1          INITIALIZATION VALUE
DKT1F2   DEFXX    ,1          SECONDARY FLAG BYTE
DKT1FTPM EQUATE    X'10'                TEMP TYPE OF DATA SET
DKT1FESM EQUATE    X'20'                ESDS TYPE OF DATA SET
DKT1FEDM EQUATE    X'40'                EDDS TYPE OF DATA SET
DKT1FASM EQUATE    X'50'                ASDS TYPE OF DATA SET
DKT1FRKM EQUATE    X'60'                RKAP TYPE OF DATA SET
DKT1FKSM EQUATE    X'70'                KSAP TYPE OF DATA SET
DKT1FCM  EQUATE    X'04'                ALLOCATE USING STORED DEF'N
DKT1FDM  EQUATE    X'08'                STORE DEF'N; DO NOT ALLOCATE
DKT1FOM  EQUATE    X'0C'                STORE DEF'N; AND ALLOCATE
DKT1HD   DEFXX    ,128        HEADER LABEL (DEFINED BELOW)
DKT1SP   DEFXX    DKT1HD,80   ALLOCATE DATA SET HEADER
         LSPACE
* EXTENDED ALLOCATE DATA SET PARAMETER LIST ( REQUEST CODE: DKTRADM )
         LSPACE
DKT1XP   DEFXX    ,9          EXTENDED ALLOCATE PARAMETER LIST
DKT1KS   DEFXX    DKT1XP,2    KEY STARTING OFFSET
DKT1KL   DEFXX    ,1          KEY LENGTH
DKT1DK   DEFXX    ,1          EXTENSION FLAG BYTE
DKT1DKAM EQUATE    X'01'                DUPLICATE KEYS ALLOWED
DKT1NBI  DEFXX    ,1          NUMBER OF BYTES TO INITIALIZE
DKT1XC   DEFXX    ,1          NUMBER OF SECONDARY EXTENTS
DKT1XS   DEFXX    ,2          SEC EXTENT SIZE IN K BYTES
DKT1AC   DEFXX    ,1          NUMBER OF ASSOCIATED DATA SETS
DKT1AN   DEFXX    ,119        ASSOCIATED DATA SET NAMES
         LSPACE
* UPDATE HEADER LABEL ( REQUEST CODE: DKTRUDM )
         LSPACE
DKT2F1   DEFXX    DKTLST,1    FLAG BYTE
DKT2FBM  EQUATE    X'80'                UPDATE BYPASS INDICATOR
DKT2FSM  EQUATE    X'40'                UPDATE DATA SET SECURITY
DKT2FWM  EQUATE    X'20'                UPDATE WRITE PROTECT INDICATOR
DKT2FVM  EQUATE    X'10'                UPDATE VERIFY/COPY INDICATOR
DKT2FRM  EQUATE    X'01'                RETURN HEADER LABEL
DKT2BI   DEFXX    ,1          NEW BYPASS INDICATOR
DKT2DS   DEFXX    ,1          NEW DATA SET SECURITY VALUE
DKT2WP   DEFXX    ,1          NEW WRITE PROTECT INDICATOR
DKT2VC   DEFXX    ,1          NEW VERIFY/COPY INDICATOR
         LSPACE
* OPEN DATA SET ( REQUEST CODE: DKTRODM )
         LSPACE
DKT3F1   DEFXX    DKTLST,1    FLAG BYTE
DKT3FTM  EQUATE    X'80'                TEMPORARY FILE DATA SET
DKT3FWM  EQUATE    X'40'                WARM START
DKT3FXM  EQUATE    X'20'                EXCLUSIVE USE
DKT3FRM  EQUATE    X'01'                RETURN HEADER LABEL
DKT3DN   DEFXX    ,17         DATA SET NAME
         LSPACE
```

```
* DEALLOCATE DATA SET ( REQUEST CODE: DKTRDDM )
         LSPACE
DKT4F1   DEFXX    DKTLST,1    FLAG BYTE
DKT4FPM  EQUATE    X'80'                RETAIN DATA SET DEFINITION
DKT4DN   DEFXX    ,17         DATA SET NAME
DKT4TD   DEFXX    ,6          TODAY'S DATE
         LSPACE
* CLOSE DATA SET ( REQUEST CODE: DKTRCDM )
         LSPACE
DKT5F1   DEFXX    DKTLST,1    FLAG BYTE
DKT5FAM  EQUATE    X'80'                CLOSE FOR ALL STATIONS
DKT5FIM  EQUATE    X'40'                IGNORE BUFFER ERRORS
         LSPACE
* BUFFER INQUIRY (REQUEST CODE: DKTRBQM)
         LSPACE
DKT8LID  DEFXX    DKTLST,2    RELATIVE POSITION OF KEYED DATA SET
*                              LABEL
         LSPACE
* VOLUME ID INQUIRY ( REQUEST CODE: DKTRVQM )
         LSPACE
DKTAF1   DEFXX    DKTLST,1    FLAG BYTE
DKTAVI   DEFXX ,6              VOLUME ID RETURN AREA
         LSPACE
* OPEN STATUS INQUIRY ( REQUEST CODE: DKTROQM )
         LSPACE
DKTBF1   DEFXX    DKTLST,1    FLAG BYTE
DKTBOM   DEFXX    ,4          OPEN BIT MAP RETURN AREA
DKTBSM   DEFXX    DKTBOM,8    OPEN BIT MAP, 60 STATIONS
         LSPACE
```

```
DKTHDR   DEFXX    DKT1HD,128  HEADER LABEL AREA
DKTHID   DEFXX    DKTHDR,4    HEADER LABEL IDENTIFIER
         DEFXX    ,1          RESERVED
DKTHDN   DEFXX    ,17         DATA SET NAME
DKTHBL   DEFXX    ,5          BLOCK LENGTH
DKTHRA   DEFXX    ,1          RECORD ATTRIBUTE
DKTHBE   DEFXX    ,5          BEGINNING OF EXTENT ADDRESS
DKTHPL   DEFXX    ,1          PHYSICAL SECTOR LENGTH
DKTHEE   DEFXX    ,5          END OF EXTENT ADDRESS
DKTHRB   DEFXX    ,1          RECORD/BLOCK FORMAT
DKTHBI   DEFXX    ,1          BYPASS INDICATOR
DKTHDS   DEFXX    ,1          DATA SET SECURITY
DKTHWP   DEFXX    ,1          WRITE PROTECT INDICATOR
DKTHET   DEFXX    ,1          EXCHANGE TYPE INDICATOR
DKTHBTM  EQUATE    X'40'                BASIC EXCHANGE TYPE
DKTHETM  EQUATE    C'E'                 FULL EXCHANGE TYPE
DKTHMV   DEFXX    ,1          MULTI-VOLUME INDICATOR
DKTHVS   DEFXX    ,2          VOLUME SEQUENCE NUMBER
DKTHCD   DEFXX    ,6          CREATION DATE
DKTHRL   DEFXX    ,4          LOGICAL RECORD LENGTH
DKTHON   DEFXX    ,5          OFFSET TO NEXT RECORD SPACE
         DEFXX    ,4          RESERVED
DKTHXD   DEFXX    ,6          EXPIRATION DATE
DKTHVC   DEFXX    ,1          VERIFY/COPY INDICATOR
DKTHDO   DEFXX    ,1          DATA SET ORGANIZATION
DKTHDDM  EQUATE    C'D'                 DIRECT ORGANIZATION
DKTHDSM  EQUATE    C'S'                 SEQUENTIAL ORGANIZATION
DKTHED   DEFXX    ,5          END OF DATA ADDRESS
         DEFXX    ,1          RESERVED
DKTHPD   DEFXX    ,48         PADDING ZEROS OR SPACES
         LSPACE
* UNALLOCATED SPACE INQUIRY ( REQUEST CODE: DKTRQSM )
         LSPACE
DKTDF1   DEFXX    DKTLST,1    FLAG BYTE
* IF FLAG BIT 0 = 0 RETURN UNALLOCATED SPACE IN SECTORS
* (DISKETTE ONLY)
DKTQUFM  EQUATE   X'80'       RETURN UNALLOCATED SPACE IN K-BYTES
DKTQMAX  EQUATE   X'40'       RETURN LARGEST CONTIGUOUS DISKETTE SLOT
DKTDUDS  DEFXX    ,4          UNALLOCATED SPACE RETURN AREA
DKTDUD0  DEFXX    DKTDUDS,2   FIRST TWO BYTES OF RETURN AREA
DKTDUD2  DEFXX    ,2          SECOND TWO BYTES OF RETURN AREA
DKTDU1   DEFXX    ,2          # BLOCKS <4K
DKTDU2   DEFXX    ,2          # BLOCKS >=4K AND <16K
DKTDU3   DEFXX    ,2          # BLOCKS >=16K AND <64K
DKTDU4   DEFXX    ,2          # BLOCKS >=64K AND <256K
DKTDU5   DEFXX    ,2          # BLOCKS >=256K AND <1024K
DKTDU6   DEFXX    ,2          # BLOCKS >=1024K
DKTDU7   DEFXX    ,2          LARGEST CONTIGUOUS DISKETTE SLOT(HI ORDER)
DKTDU8   DEFXX    ,2          LARGEST CONTIGUOUS DISKETTE SLOT(LO ORDER)
         LSPACE
* REORGANIZE DATA SET REQUEST ( REQUEST CODE: DKTRRGM )
         LSPACE
DKTGF1   DEFXX    DKTLST,1    FLAG BYTE
DKTGDN   DEFXX    ,17         DATA SET NAME
         LSPACE
```

```
* HEADER LABEL EXTENSION INQUIRY ( REQUEST CODE: DKTRQLM )
         LSPACE
DKTQF1   DEFXX    DKTLST,1    FLAG BYTE
DKTQRPM  EQUATE    X'80'              QUERY BY RELATIVE POSITION
DKTQNPM  EQUATE    X'C0'              QUERY TO NEXT RELATIVE POS'N
DKTQSZM  EQUATE    X'01'       RETURN CURRENT DATA SET SIZE IN K-BYTES
*                              OR SECTORS(DISKETTE ONLY)
DKTQPRL  DEFXX    ,2           RELATIVE POS'N OF LABEL
         DEFXX    ,1           UNUSED POSITION (BLANK)
DKTQDN   DEFXX    ,17          DATA SET NAME
DKTQBL   DEFXX    ,5           BLOCK LENGTH
DKTQRA   DEFXX    ,1           RECORD ATTRIBUTE
DKTQBE   DEFXX    ,5           BEGINNING OF EXTENT ADDRESS
DKTQPL   DEFXX    ,1           PHYSICAL SECTOR LENGTH
DKTQEE   DEFXX    ,5           END OF EXTENT ADDRESS
DKTQRB   DEFXX    ,1           RECORD/BLOCK FORMAT
DKTQBI   DEFXX    ,1           BYPASS INDICATOR
DKTQDS   DEFXX    ,1           DATA SET SECURITY
DKTQWP   DEFXX    ,1           WRITE PROTECT INDICATOR
DKTQET   DEFXX    ,1           EXCHANGE TYPE INDICATOR
DKTQBTM  EQUATE    X'40'                BASIC EXCHANGE TYPE
DKTQETM  EQUATE    C'E'                 FULL EXCHANGE TYPE
DKTQMV   DEFXX    ,1           MULTI-VOLUME INDICATOR
DKTQVS   DEFXX    ,2           VOLUME SEQUENCE NUMBER
DKTQCD   DEFXX    ,6           CREATION DATE
DKTQRL   DEFXX    ,4           LOGICAL RECORD LENGTH
DKTQON   DEFXX    ,5           OFFSET TO NEXT RECORD SPACE
         DEFXX    ,4           RESERVED
DKTQXD   DEFXX    ,6           EXPIRATION DATE
DKTQVC   DEFXX    ,1           VERIFY/COPY INDICATOR
DKTQDO   DEFXX    ,1           DATA SET ORGANIZATION
DKTQDDM  EQUATE    C'D'                 DIRECT ORGANIZATION
DKTQDSM  EQUATE    C'S'                 SEQUENTIAL ORGANIZATION
DKTQED   DEFXX    ,5           END OF DATA ADDRESS
         DEFXX    ,1           RESERVED
DKTQCSZ  DEFXX    ,2           CURRENT DATASET SIZE (K-BYTES)
DKTQFG1  DEFXX    ,1           FLAG BYTE
DKTQFBM  EQUATE    X'80'                ALLOCATE ON TRACK BOUNDARY
DKTQFTM  EQUATE    X'40'                EXTENT SIZE GIVEN IN TRACKS
DKTQFKM  EQUATE    X'20'                EXTENT SIZE GIVEN IN K BYTES
DKTQES   DEFXX    ,2           EXTENT SIZE (TRKS/SCTS/KBYTES)
DKTQNA   DEFXX    ,2           NUMBER OF SECTORS TO ASSIGN
DKTQNI   DEFXX    ,2           NUMBER OF SECTORS TO INITIALIZE
DKTQIV   DEFXX    ,1           INITIALIZATION VALUE
DKTQF2   DEFXX    ,1           SECONDARY FLAG BYTE
DKTQFTPM EQUATE    X'10'                TEMP TYPE OF DATA SET
DKTQFESM EQUATE    X'20'                ESDS TYPE OF DATA SET
DKTQFEDM EQUATE    X'40'                EDDS TYPE OF DATA SET
DKTQFASM EQUATE    X'50'                ASDS TYPE OF DATA SET
DKTQFRKM EQUATE    X'60'                RKAP TYPE OF DATA SET
DKTQFKSM EQUATE    X'70'                KSAP TYPE OF DATA SET
DKTQFCM  EQUATE    X'04'                ALLOCATE USING STORED DEF'N
DKTQFDM  EQUATE    X'08'                STORE DEF'N; DO NOT ALLOCATE
DKTQFOM  EQUATE    X'0C'                STORE DEF'N; AND ALLOCATE
```

```
DKTQKS   DEFXX    ,2          KEY STARTING OFFSET
DKTQKL   DEFXX    ,1          KEY LENGTH
DKTQDK   DEFXX    ,1          EXTENSION FLAG BYTE
DKTQDKAM EQUATE    X'01'                DUPLICATE KEYS NOT ALLOWED
DKTQNBI  DEFXX    ,1          NUMBER OF BYTES TO INITIALIZE
DKTQXC   DEFXX    ,1          NUMBER OF SECONDARY EXTENTS
DKTQXS   DEFXX    ,2          SEC EXTENT SIZE IN K BYTES
DKTQXA   DEFXX    ,1          NUMBER OF SEC'Y EXTENTS ALLOC
DKTQAC   DEFXX    ,1          NUMBER OF ASSOCIATED DATA SETS
DKTQAN   DEFXX    ,119        ASSOCIATED DATA SET NAMES
         LSPACE
* RENAME DATA SET REQUEST ( REQUEST CODE: DKTRRNM )
         LSPACE
DKTRF1   DEFXX   DKTLST,1     FLAG BYTE
DKTRDN   DEFXX    ,17         CURRENT DATA SET NAME
DKTRRN   DEFXX    ,17         NEW DATA SET NAME
DKTEND   DEFXX   DKTRCD,(D:DKTQAN+L:DKTQAN-D:DKTRCD) MAX SIZE
         BQKLDS  TYPE=END
```

DEFFDK Format Diskette Parameter List

Equate DEFFDKS to a segment number.

DEFFDK

```
* * *     'FORMDKT' INSTRUCTION PARAMETER LIST DEFINITION

FDKPAR   DEFxx   s,9         'FORMDKT' PARAMETER LIST

FDKLNG   DEFxx    FDKPAR,1   DISKETTE RECORD LENGTH :
FDKLN0   EQUATE    X'00'               128 BYTE RECORD LENGTH VALUE
FDKLN1   EQUATE    X'01'               256 BYTE RECORD LENGTH VALUE
FDKLNF   EQUATE    X'FF'               DEFAULT TO MOUNTED DSK REC LEN

FDKTYP   DEFxx   s,1         DISKETTE TYPE :
FDKTY1   EQUATE    X'01'               DISKETTE 1 VALUE
FDKTY2   EQUATE    X'02'               DISKETTE 2 VALUE
FDKTY2D  EQUATE    X'03'               RESERVED
FDKTYF   EQUATE    X'FF'               DEFAULT TO MOUNTED DSK TYPE

FDKRES   DEFxx   s,1         RESERVED

FDKVID   DEFxx   s,6         DISKETTE VOLUME IDENTIFIER
```

Appendix C. Program Check Codes

If the 4700 controller encounters an execution request that indicates a logic error, a program check results. The following are the hexadecimal codes and the explanations for possible program checks:

Code	Explanation
01	Invalid segment specification: An operand specifies a segment that was not defined during controller configuration procedure, or segment 14 was specified in an instruction that will cause data to be stored or changed in segment 14.
02	Segment overflow: Completion of the instruction requires more storage than the specified segment provides.
03	Field length error: An incorrect field was specified. The length is greater than 2 for an immediate operand; or a SETFPL instruction attempted to adjust the field length indicator to a negative value; or a value is specified which, when added to the PFP, would be greater than the segment length; or The field length was greater than 255 for a PAKSEG instruction.
04	Return-address stack error: An LRETURN instruction was issued, but the return-address stack was empty; or a branch instruction was issued, but the stack was full.
06	Instruction count threshold: The number of instruction executions allowed per transaction has been exceeded.
08	No overlay name: The overlay name is not in the resident overlay directory.
09	Invalid operation or segment code: The instruction operation or segment selection code specified is invalid. Make sure that any required OPTMOD coding for the instruction was entered and that any parameter fields are properly coded.
0A	No entry point: There is no startup entry point specified.
0B	Instruction address error: An addressing error has occurred. In the case of branch instructions, the program check address field of segment 1 will contain the address of the branch instruction.
0C	Instruction count exceeded: 65,535 instructions have been executed without a release of control.
0D	DEFDEL missing or incorrectly used: Either a delimiter request was made but no delimiter table was found or the table is not halfword aligned.
0E	EDIT mask error: The mask used with an EDIT instruction contains an error.

0F	Invalid link write control field: The link write control field or write options are invalid.
10	Communication link write length error: Data length exceeds 4095, data length during an LWRITE in batch mode was too long, command data length is incorrect; negative-response data length is incorrect, or there was a negative response to setting or testing sequence numbers.
11	Invalid parameter list, or parameter space is insufficient.
12	Indexing is not active.
20	Program check in called application program.
21	Called application program not found.
22	APCALL link stack full.
23	Recursive APCALL to an application program defined as USE=STATIC during configuration.
24	APCALL storage pool defined by MAXSTOR=was exceeded.
25	APCALL segment pool defined by MAXSEG=was exceeded.
26	APRETURN issued with no APCALL link stack entry - no calling application program.
27	Register address contains invalid segment space ID.
28	No transient pool: a transient pool was not defined for this station.
29	Transient application size error: the target transient application program will not fit in the largest transient area defined in the pool for this station.
FF	System error.

Appendix D. Status Codes for Disk and Diskette

The list below and tables that follow contain information about the two bytes of status bits that are set in SMSDST when an exceptional condition occurs (condition code= hex 02). The status bits in the first byte (SMSDS1) indicate the general condition:

Bits in SMSDS1		*Condition*
-------1	(hex 01)	Incorrect length
------1-	(hex 02)	Unit check
-----1--	(hex 04)	Command reject
----1---	(hex 08)	Attention
---1----	(hex 10)	Prior operation
--1-----	(hex 20)	Data check
-1------	(hex 40)	Unit exception
1-------	(hex 80)	Intervention required

The status bits in the second byte (together with those in the first) indicate the specific condition, as shown in the tables. The list is common to all tables. To use the tables and list, find the status bits in the leftmost column of the appropriate table, the applicable instruction in the third column; read the explanation of the corresponding condition in the second column of the table.

A status value not in the tables may be a combination of status codes. When such status values occur, review the appropriate table and search for the highest value first, then the next highest value. Remember that a status bit can be shared by more than one status code.

Status Bits	Condition	Instruction
-------- --1----- (hex 0020)	Duplicate keys: This status is generated for keyed or 'NEXT' LREAD through a KSAP if one or more subsequent records having the same key exist. *Action:* Defined by controller application program.	LREAD PLR
-------1 -------- (hex 0100)	Wrong Length Record: LWRITE PLR: The data to be output was shorter than the logical record length specified in the data-set header label. REPLACE TFn, C or PLR: The data to be output was shorter than the original record. REPLACE P, Single Sector: The data to be output was shorter than the sector length on the device. REPLACE P, Multi-Sector: The length of the data to be output was not sufficient to completely fill the final sector. *Action:* Defined by controller application program.	LWRITE PLR REPLACE

Status Bits	Condition	Instruction
-------1 -------1 (hex 0101)	Wrong Length, Oversized Record: LREAD: If this is not a multi-sector read, the input area was shorter than the record or sector to be read. If this is a multi-sector read, the input area was not long enough to fully contain the final sector. LWRITE TFn or L: The data to be output was longer than 252 bytes. LWRITE PLR: The data to be output was longer than the logical record length specified in the header label. REPLACE PLR: The data to be output was longer than the original record length. REPLACE P, Single Sector: The data to be output was longer than a sector on the device. LDKT (Buffer Inquiry, Query Extended Header Label): User data truncated. The user return area was not long enough to return all the requested data. *Action:* Defined by the controller application program.	LREAD LWRITE REPLACE LDKT

Status Bits	Condition	Instruction
-------1 ------1- (hex 0102)	Wrong Length, Record too Long: A 512-byte diskette was loaded, but the controller is not configured for 512-byte buffers.	READ LWRITE
-------1 -1----1- (hex 0142)	Secondary Index Exception: Record too short for secondary index key; duplicate keys not allowed for secondary index update; index EOF for secondary index update; index update exception during KSAP 'Next' or 'Previous' LREAD through a KSAP (index has been updated since last read, sequence may have been impacted).	LREAD LWRITE LDKT REPLACE DELETE
------1- -------- (hex 0200)	Unit Check: Any function: A hardware malfunction was detected. *Action:* Check the statistical error counters. See the *IBM 4700 Finance Communication System, Problem Determination Guide*: GC31-2033.	Any Diskette/Disk Function
------1- -------1 (hex 0201)	Unit Check, Wrong Length Temporary Record: LREAD or REPLACE TFn, C or L: An invalid logical record length was encountered in a temporary file data block while searching for the target record. (For example, the record length was either zero or greater than the number of bytes from the start of the record to the end of the sector.) LDKT (Open): An invalid logical record length was encountered while attempting a warm start open of a temporary file data set. (For example, the record length was either zero or greater than the number of bytes from the start of the record to the end of the sector.) *Action:* (1) If this occurs on the operating diskette or on a data set opened as a temporary file, see the *IBM 4700 Finance Communication System, Problem Determination Guide*: GC31-2033. (2) If the wrong data set was opened with the temporary file option, recovery depends on the controller application program.	LREAD REPLACE LDKT

Status Bits	Condition	Instruction
------1- ------1- (hex 0202)	Unit Check, Session ID Error: LREAD or REPLACE TFn, C or L: A temporary file index block or data block read while searching for the target record did not contain the current session ID. *Action:* If this occurs on the operating diskette, or on a data set opened with the temporary file option, see the *IBM 4700 Finance Communication System, Problem Determination Guide*: GC31-2033. Otherwise, recovery depends on the application program.	LREAD REPLACE
------1- ------11 (hex 0203)	Unit Check, Bad Diskette: Diskette has more than two bad tracks, or track 0 has a bad sector. *Action:* Discard the diskette and use a new one.	FORMDKT
------1- -----1-- (hex 0204)	Unit Check, Control Record Read: Any function: A sector having a control address mark, rather than a data address mark, was read unexpectedly. *Action:* Defined by the controller application program.	Any Diskette Function
------1- ----1--- (hex 0208)	Unit Check, Unreadable Sectors Logged: Diskette compression was completed although one or more data-set sectors were not readable. The Compress Utility Error Log contains information concerning these errors. The Compress Utility Error Log is located in the last sector of track 74, side 0 for Diskette 1 and the last sector on track 74, side 1 for Diskette 2. *Action:* Examine the Compress Utility Error Log that contains the data-set name, the beginning of the extent, and the address of the unreadable sector. The sector addresses listed on sequential data-set read errors correspond to the original data-set locations. The contents of any unreadable sequential sectors are omitted from the compressed diskette.	COMPDKT

Status Bits	Condition	Instruction
------1- ---1---- (hex 0210)	Unit Check, Read Error Count Exceeded: The number of unreadable sectors in sequential data sets exceeded the maximum allowed. *Action:* This diskette cannot be compressed. Retry your copy diskette procedure. The Compress Utility Error Log may be examined. See Status hex 0208 for an explanation of the Compress Error Log.	COMPDKT
------1- -1------ (hex 0240)	Unit Check, Wrong Temporary File Record ID: LREAD or REPLACE TFn, C or L: The temporary file record found by an index search did not meet the search criteria: • If a composite file search was done, the retrieved record subfile ID did not match SMSSFR, or • If a TFn search was done, either the retrieved record did not have the correct file ID, or, if SMSSFR was nonzero the retrieved sub-file ID did not match it. *Action:* If this occurs on the operating diskette, or on a data set opened with the temporary file option, see the *IBM 4700 Finance Communication System, Problem Determination Guide*: GC31-2033. If the wrong data set was opened with the temporary file option, then recovery depends on the controller application program.	LREAD REPLACE

Status Bits	Condition	Instruction
-----1-- -------1 (hex 0401)	Command Reject, Buffer Alignment or Overlay Length Error: LREAD or REPLACE, Multi-Sector: The buffer specified for the option did not begin at a displacement that is an integral multiple of two. LLOAD: The length of the application program overlay section to be loaded is too large. (For example, the length plus the load address exceeds the end of the section of Segment 14 to be overlayed.) *Action:* Depends on the controller application program.	LREAD REPLACE LLOAD
-----1-- ------1- (hex 0402)	Command Reject, Invalid Side Request: LREAD or REPLACE A, Single Sector or Multi-Sector: The absolute diskette address passed in SMSRSN had the side 1 bit set, but the currently inserted diskette is one-sided. *Action:* Depends on the controller application program. If a two-sided diskette is to be accessed, a two-sided diskette must be inserted. Command Reject, Session ID Error: LREAD TFn, C : The current session ID did not match the session ID in the temporary file record when the LREAD was attempted. This may be caused by an action or operation that damages the temporary file on the diskette such as inserting the wrong diskette or through EDAM by opening the wrong data set. *Action:* If the temporary file on the operating diskette has been damaged, see the *IBM 4700 Finance Communication System, Problem Determination Guide*: GC31-2033. If the wrong diskette was inserted or the wrong data set was opened, recovery depends on the controller application program.	LREAD REPLACE

Status Bits	Condition	Instruction
-----1-- ------11 (hex 0403)	Command Reject, Volume Access Inhibited: Access to the volume is inhibited by another station. *Action:* The inhibiting station must give up control.	FORMDKT COMPDKT
	Command Reject, Access Inhibited on Drive: A station issued LDKT (Inhibit Access to Volume). No other operation can be initiated by any other station (except for absolute requests) until the inhibiting station issues an LDKT (Permit Access to Volume).	Any Disk/Diskette Function
-----1-- -----1-- (hex 0404)	Command Reject, Drive not supported: The disk or diskette drive selected is either not attached or is not defined by the CPGEN FILES macro. Any EDAM function: EDAM=Y was specified on a CPGEN FILES macro, but not on the one for the currently selected drive. *Action:* Correct the configuration by supplying the appropriate FILES macro, or use the primary drive.	Any Disk/Diskette Function

Status Bits	Condition	Instruction
-----1-- ----1--- (hex 0408)	Command Reject, Invalid Extent Parameter: LDKT (Allocate): The Beginning-of-Extent address or the End-of-Extent address is not all zeros or spaces, and its: • Track number is either zero or greater than 74, or • Sector number is either zero or greater than the number of sector on a track of the currently mounted diskette, or • Head number is 1 and the currently mounted diskette is 1-sided. The requested primary or secondary extent size for a disk data set is either 0 or greater than the maximum (hex 0800K bytes, hex 2000 sectors, or hex 0200 tracks). Either the Beginning- or End-of-Extent address is all zeros or spaces, **and** the number of sectors or tracks to allocate is zero. The End-of-Extent address is less than the Beginning-of-Extent address, and neither is all zeros or spaces. *Action:* Defined by the controller application program.	LDKT

Status Bits	Condition	Instruction
-----1-- ---1---- (hex 0410)	Command Reject, Data Set Not Open: LWRITE, LCHECK, LREAD or REPLACE, Exchange Data Set, or LDKT (Update Header Label): The data-set ID given in SMSDID is invalid (it is either zero or greater than the number of header labels on the diskette), or the data set has not been opened by the current station. *Action:* If the data-set ID is 0 or invalid, correct the data-set ID. Otherwise, open the data set. LDKT Disk (Allocate, Open, Deallocate, Rename, Reorganize, Query Extended Header Label, Query Unallocated Space, or Query Volume ID) The selected disk drive has not been initialized for EDAM operations. *Action:* Use the EDAM Disk Initialization function of the Installation Diskette to initialize the disk drive.	DELETE LWRITE LREAD LCHECK REPLACE LDKT
-----1-- --1----- (hex 0420)	Command Reject, Data Set Write Protected or Unexpired: The specified data set is write protected. *Action:* Defined by controller application program. Use LDKT to update the header label, changing the Write-Protect indicator to a space. LDKT (Open, Allocate, Deallocate, Rename, Reorganize): The controller tried to open, allocate, deallocate, rename, or reorganize a data set with name beginning SYS. SYS data sets are protected. *Action:* Use another data-set name. LDKT (Deallocate): The current date in the parameter list is less than the expiration date in the header label. *Action:* Defined by controller application program; set the current date in the parameter list to all 9s.	DELETE LWRITE REPLACE LDKT

Status Bits	Condition	Instruction
-----1-- -1------ (hex 0440)	Command Reject, Diskette Stopped: Any temporary file function, or LLOAD: A temporary file operation or an LLOAD instruction was attempted with the primary drive selected, but the drive is in a logically stopped state. **Note:** LLOAD is described in Volume 1 of the *4700 Controller Programming Library.* LREAD or REPLACE P: A permanent file operation was attempted with the primary drive selected and SMSDID set to zero, but the drive is in a logically stopped state. SETDSKT: A SETDSKT code hex 01, 02, 04, 08, or 10 was issued while the diskette was in a stopped state. *Action:* Defined by controller application program. The control operator can be notified to start the diskette drive.	LLOAD LREAD REPLACE SETDSKT
-----1-- -1-----1 (hex 0441)	Command Reject, Keyed Record too Short: The buffer space provided for the unkeyed data set was shorter than the sum of the key offset and the key length.	DELETE LREAD LWRITE REPLACE
-----1-- -1----1- (hex 0442)	Command Reject, Duplicate Keys: Duplicate keys not allowed in primary index.	LWRITE
-----1-- -1---1-- (hex 0444)	Command Reject, Insufficient Buffer Space: When duplicate keys are not allowed, the user data segment must have enough space to hold an additional logical record. The space must be available between the end of the logical record and the end of the data segment. For ASDS variable length records, the maximum size logical record should be provided.	DELETE LREAD LWRITE REPLACE

Status Bits	Condition	Instruction
-----1-- 1------- (hex 0480)	Command Reject, Invalid Request:	COMPDKT FORMDKT DELETE LREAD LWRITE REPLACE

Any function:

The file code in the instruction is incompatible with the operation code (for example, the LOG file code in a REPLACE instruction).

LWRITE TFn:

The file code in the instruction was greater than the number of files specified in the TF operand of the FILES macro for the selected drive, or the subfile given in SMSSFW was greater than 60.

LREAD or REPLACE TFn or C:

No indexing was requested in the CPGEN for either the file specified in the instruction or the subfile specified in SMSSFR.

LREAD or REPLACE C:

SMSSFR contained a value of zero.

Action: Correct either the configuration or the controller application program.

Any EDAM function:

The primary drive is selected and is not logically stopped.

LREAD or REPLACE P, REPLACE PLR, ALLOCATE:

The selected EDAM data set has sequential organization.

Action: Defined by controller application program.

COMPDKT:

Command Reject, Invalid Parameter List:
The parameter list contained an invalid data-set count, an invalid combination in the flag byte, or an invalid flag byte and data-set sector count combination.

Action: Correct the parameter list and retry.

Status Bits	Condition	Instruction
	FORMDKT: The parameter list contained an invalid record length, diskette type, volume ID, physical record sequence code, or specified an invalid record length for the specified diskette type. *Action:* Correct the parameter list and retry.	

Status Bits	Condition	Instruction
-----1-- 1------1 (hex 0481)	Command Reject, Optional Support Module Missing: Any EDAM function:	LDKT LREAD REPLACE DELETE LWRITE

EDAM was not specified on any FILES macro in the CPGEN, or loading of the EDAM support module was suppressed by the control operator.

Extended record format (XRCD) for ASDS was not specified on any FILES macro in the CPGEN.

KEYED was not specified on any FILES macro in the CPGEN.

LDKT (Allocate or Deallocate):

ALLOC was not specified in the EDAM parameter of any FILES macro in the CPGEN, or loading of the EDAM Allocate/ Deallocate support module was suppressed by the control operator.

LDKT (Open):

The temporary file option was specified but the TF operand was omitted from the FILES macro for the secondary drive, or loading of the EDAM temporary file support module was suppressed by the control operator.

LREAD or REPLACE, Multi-Sector:

Optional modules M06 or MB7 for multi-sector support, were not specified during CPGEN, or the loading of the multi-sector support modules was suppressed by the control operator.

Action: Be sure that EDAM is specified, and that the appropriate modules are selected and loaded.

Status Bits	Condition	Instruction
-----1-- 1-----1- (hex 0482)	Command Reject, Data Set Size Conflict: LDKT (Allocate): The number of sectors to assign or initialize is greater than the number of sectors in the primary extent. These parameters must be specified in sectors. Extent size can be specified in K-bytes, sectors, or tracks. *Action:* Defined by the controller application program.	LDKT
-----1-- 1----1-- (hex 0484)	Command Reject, Open Conflict: LDKT (Open): The temporary file option was specified and the data set is presently open by one or more stations, but not as a temporary file. The temporary file option was not specified, but the data set is presently open by one or more stations as a temporary file. LWRITE, LCHECK, LREAD or REPLACE, Temporary File, Exchange Data Set: The specified data set was not opened as a temporary file. LWRITE, LCHECK PLR, DELETE PLR, REPLACE PLR, LREAD P, or REPLACE P: The specified data set was opened as a temporary file. *Action:* Defined by the controller application program. LDKT (Rename): The data set is currently open and must be closed before rename can be performed. *Action:* Close the data set.	DELETE LDKT LWRITE LCHECK LREAD REPLACE

Status Bits	Condition	Instruction
-----1-- 1----1-1 (hex 0485)	Command Reject, Data Set Type Conflict: LDKT (Open): The temporary file option was specified, but the data set does not meet the criteria for a temporary file for one of the following reasons: • Physical sector length is not 256. • Block length is not 256. • Logical record length is not 256. • Not all blocks are assigned. • The data set is organized sequentially. • The primary diskette was specified. *Action:* Defined by the controller application program.	LDKT
----1-1- 1----11- (hex 0486)	Attention, Unit Check, Data Set Defined but Not Allocated. An attempt was made to open a data set that is only defined. You must allocate the data set before opening it.	LDKT

Status Bits	Condition	Instruction
-----1-- 1---1--- (hex 0488)	Command Reject, Invalid Data Set Specification:	LDKT

LDKT (Allocate):

One of the following errors was detected in the parameter list:

- Exchange type indicator is not blank or C'E'.
- Data set name begins with a blank.
- Record attribute is not blank or C'B'.
- Physical record length is not blank or C'1'.
- Record/Block format is not blank or C'F'.
- Data set organization is not blank, C'S' or C'D'.
- Logical Record Length is all zeros or blanks, or is greater than the physical sector length on the currently mounted diskette.
- Secondary extents were requested on a diskette.
- The number of secondary extents requested was greater than 15.
- The number of secondary extents requested is zero and the secondary extent size is not zero.
- Allocation units of both tracks and K-bytes were specified in the same parameter list.

Action: Defined by the application. Correct parameter list. If the logical record length is greater than the sector length, mount a diskette with 256-byte sectors.

Command Reject; Invalid user request:

LDKT (Query Extended Header Label and Query Available Space):

The user request flag is set to an invalid value.

Command Reject; Invalid Data Set Type;

LDKT reorganize:

You specified a data-set type that is not a KSAP or ASDS.

LDKT reset input:

Data type is RKAP, KSAP, or TEMP.

Status Bits	Condition	Instruction
-----1-- 1---1--1 (hex 0489)	Command Reject, Invalid Keyed Access Definition: • No associated data sets were specified for an RKAP or a KSAP. • One or more associated data sets were specified for a data-set type other than RKAP or KSAP. • The number of associated data sets is greater than 7. • An RKAP or a KSAP data set did not specify a valid key length. • A key length is specified for a data-set type other than RKAP or KSAP. An unkeyed data set must specify a key length of hex 00.	LDKT
-----1-- 11----1- (hex 04C2)	Command Reject, Invalid Address: LREAD or REPLACE A; LREAD or REPLACE PBN: The absolute address (TTRR) or Physical Block Number (PBN) was invalid for the disk or diskette. Record not found (KSAP, RKAP, or ASDS). *Action:* Defined by the controller application program.	DELETE REPLACE LDKT
-----1-- 11----11 (hex 04C3)	Command Reject, Data Set is Associated: LDKT Reset Output Pointer was issued for an ASDS or EDDS data set that is associated with a RKAP or KSAP. *Action:* Defined by the controller application program.	LDKT (Reset Output Pointer)

Status Bits	Condition	Instruction
----1--- -------- (hex 0800)	Attention: Presented in conjunction with other status: The coincident status resulted while reading the volume label (LDKT Allocate, Deallocate, or Open), while reading or writing a data header label (LDKT Allocate, Deallocate, Open, or Close), while reading or writing the ERMAP sector (LDKT Allocate), or while reading or writing a temporary file index block or data block (LDKT Open). **Note:** *If this status occurs on a Close function, the data set has been closed but, the label is of dubious integrity. If it occurs on any other function, the function was not completed.* *Action:* Defined by the controller application program and by the other status presented.	LDKT
----1--- 1----111 (hex 0887)	Attention, Data Set Allocation Not Fully Complete: For disk, all secondary extents are requested to be assigned and allocated immediately; but there is not enough contiguous space available for each of the secondary extents. The primary extent and as many secondary extents as possible have been allocated. For diskette, a keyed data set was allocated but the optional module required for keyed data sets is not present in controller storage. The pointers to the associated records could not be generated. *Action:* Verify the attributes of the data set using the Disk/Diskette File Utility program or the LDKT Query Extended Header Label function. If the attributes are not acceptable, then deallocate the data set and reallocate it when the problem has been corrected.	LDKT
----1--- 1---1--- (hex 0888)	Attention, Data Set Definition Nonexistent: Deallocation and definition retention is requested for a definition that does not exist.	LDKT

Status Bits	Condition	Instruction
----1--- 1---1--1 (hex 0889)	Attention, Invalid Associated Data Set Specification: • When defining a keyed data set, 1 or more of its associated data sets had previously been associated with 7 other data sets. • A specified associated data set has not been previously defined as an EDDS or ASDS. This association has *not* been made.	LDKT
----1-1- -------- (hex 0A00)	Attention, Bad Disk or Diskette: A permanent error occurred while reading or writing the volume label (diskette); the BAM sector (disk); or a data-set label (disk or diskette). *Action:* Insert another diskette and/or see the *IBM 4700 Finance Communication System, Problem Determination Guide*: GC31-2033.	COMPDKT LDKT
----1-1- ----1--- (hex 0A08)	Attention, Unit Check, Invalid Extent: Allocate, Deallocate, Open, Query Extended Header Label (Diskettes), Query Volume ID, Rename, Reorganize: COMPDKT: During volume initialization, an existing data-set header label was read in which the Beginning-of-Extent address or End-of-Extent address was invalid because: • It was all zeros. • Track number is either zero or greater than 74. • Sector number is either zero or greater than the number of sectors on a track of the currently mounted diskette. • Head number is 1 and the currently mounted diskette is one-sided. • The End-of-Extent address is less than the Beginning-of-Extent address. *Action:* Release or recreate the disk or diskette data set.	COMPDKT LDKT

Status Bits	Condition	Instruction
	LDKT (Open): The End-of-Data address in the header label for the data set is invalid because: • It is all zeros or spaces. • It is less than the Beginning-of-Extent address. • It is more than 1 sector greater than the End-of-Extent address. • Diskette track number is either zero or greater than 74 (except for the valid case of a data set extending to the end of track 74, in which case End-of-Data address can be 75001). • Sector number is either zero or greater than the number of sectors on a track of the currently mounted diskette. • Head number is 1 and the currently mounted diskette is one-sided. *Action:* Recreate the data set.	
----1-1- --1----- (hex 0A20)	Attention, Unit Check, Invalid Volume Label: LDKT (Allocate, Deallocate, Open, Rename, Query Volume ID, Reorganize, Query Extended Header Label): The volume label sector does not contain C'VOL1' in bytes 0 - 3. *Action:* Format the diskette correctly. FORMDKT: The volume ID in the parameter list or on the mounted diskette is invalid. *Action:* Specify valid volume ID in parameter list and retry. Valid volume IDs contain no leading or imbedded blanks and are made up of EBCDIC alphameric characters. COMPDKT: The volume label sector does not contain C'VOL1' in bytes 0-3. *Action:* Format the diskette correctly.	LDKT FORMDKT COMPDKT COMPDKT

Status Bits	Condition	Instruction
----1-1- -1------ (hex 0A40)	Attention, Unit Check, Unrecoverable BAM Error: A disk is out of synchronization. *Action:* Copy data sets from disk, reinitialize using Installation Diskette, and copy data sets back to disk. LDKT (Allocate, Deallocate or Open): A request was made to allocate an H-Exchange data set on other than a Diskette 2D. *Action:* Mount a Diskette 2D.	LDKT
----1-1- 1------- (hex 0A80)	Attention, Unit Check, Extent Overlap: LDKT (Allocate, Deallocate, Open Rename, Query Volume ID, Reorganize, Query Extended Header Label): During volume initialization, an existing data-set header label was read, in which the defined extent overlapped the extent(s) defined by one or more previously processed header labels. *Action:* Recreate the data set.	COMPDKT LDKT

Status Bits	Condition	Instruction
----1-1- 1---1--- (hex 0A88)	Attention, Unit Check, Invalid Data Set: LDKT (Open): One of the following errors was detected in the header label of the data set being opened: • Block length field is zero or greater than the sector length. • Logical record length field is zero or greater than the block length. • The block length is not an even multiple of the logical record length. • Offset-to-Next-Record is not zeros or spaces, and it is greater than the block length, it is not an even multiple of the logical record length, or the data set does not contain blocked records. The temporary file and warm start options were specified and a logical record was encountered in which the subfile ID was invalid (greater than 60). *Action:* Recreate the data set.	LDKT
----1-1- 1---1--1 (hex 0A89)	Attention, Unit Check, Invalid Associated Data Set Specification: The associated data sets could not be opened because the keyed data-set definition is invalid or the associated data sets are defined but not allocated.	LDKT

Status Bits	Condition	Instruction
---1---- -------- (hex 1000)	Prior Operation:	LDKT LWRITE LREAD REPLACE

If presented by itself:

The requested Data Set operation was not begun because one or more sectors had to be output first and had previously been flagged as unwritable.

LREAD or REPLACE, Data Set, Multi-Sector:

One of the blocks involved in the current request was found to be in one of the controller buffers, and had previously been flagged as unwritable.

REPLACE, Exchange Data Set, Logical Record or Single Sector:

The sector involved in the current request was found to be in one of the controller buffers, and had previously been flagged as unwritable.

LWRITE, LREAD or REPLACE, Exchange Data Set:

A buffer was needed for the current operation, but all buffers presently contain sectors that had previously been flagged as unwritable.

LDKT (Allocate, Deallocate, Open, Rename, Query Extended Header Label, Reorganize, or Update Header Label):

Prior Operation:

A sector was marked unwritable on a previous operation and the number of buffers specified in the CPGEN is not sufficient to complete the current operation.

LDKT (Close):

One or more of the sectors belonging to the data set being closed was found in a controller buffer that had previously been flagged as unwritable.

Action: Recover unwritable data; release unwritable buffers.

Status Bits	Condition	Instruction

If presented with other status:
LWRITE TFn or L, or LCHECK DSK:
The indicated status resulted from a previous attempt to write the current temporary file output buffer.

LWRITE or LCHECK PLR, or LDKT (Close):

The current output buffer for the data set was flagged as unwritable as a result of a previous asynchronous write attempt that failed for the indicated reason.

LREAD or REPLACE, Exchange Data Set, Multi-Sector:

Before beginning the requested operation, the controller attempted to output one of the involved sectors from a buffer that had been flagged as changed and the write failed for the indicated reason.

Action: No recovery possible; reread temporary file from beginning.

Status Bits	Condition	Instruction
---11--- -------- (hex 1800)	Attention, Prior Operation: LDKT (Allocate, Deallocate or Open, Query Volume ID, Rename, Reorganize, Query Extended Header Label): In attempting to initialize the volume for processing, one or more buffers associated with the drive were found to contain a sector previously flagged as unwritable. *Action:* Defined by controller application program. Records can be retrieved from unwritable buffers using LDKT buffer-inquiry and LREAD P.	LDKT
--1----- -------- (hex 2000)	Data Check: Any function: A cyclic redundancy check (CRC) error occurred on a read operation, or a read after write comparison failed on a write operation. *Action:* Defined by controller application program; recover data and reconstruct file.	Any Disk/Diskette Function
--1----- -------1 (hex 2001)	Data Check, Incorrect Record Type: The data set is not an ASDS M-format.	LREAD LWRITE REPLACE DELETE PLR
--1---1- -----1-- (hex 2204)	Data Check, Control Record Read: Any function: A sector having a control address mark, rather than a data address mark, was read unexpectedly and, in addition, its data area CRC did not match the computed CRC. *Action:* Defined by controller application program.	Any Diskette Function

Status Bits	Condition	Instruction
-1------ -------- (hex 4000)	Unit Exception: LWRITE TFn or L: A previous LWRITE filled the temporary file data set. LREAD or REPLACE, Temporary File, Logical Record or Single Sector: The record number given in SMSRSN was zero or greater than the number of records presently contained in the data set, the permanent file, or the temporary file or subfile. LREAD or REPLACE, Multi-Sector: The absolute address passed in SMSRSN specified a track number greater than 74, or a sector number of zero, or a sector number greater than the number of sectors on the specified track. During the multi-sector operation, the end of the diskette, permanent file or exchange data set was reached before the request had been fully satisfied. LWRITE PLR: The data set already contains 65 535 logical records, or there are no unassigned blocks left in the data-set extent. The primary keyed data set is full. Unit Exception: Keyed processing: End-of-File during LREAD or LWRITE. No allocated unkeyed data set to contain record for keyed LWRITE. LDKT (Buffer Inquiry): No buffers were found that matched the search criteria and contained unwritable records. Unit Exception; LDKT (Query Extended Header Label): End-of-File reached during query next valid label by relative position. Invalid SMSRSN specified on query label by relative position. *Action:* Defined by controller application program.	LWRITE LREAD REPLACE LDKT

Status Bits	Condition	Instruction
-1------ -------1 (hex 4001)	Unit Exception, Temporary File Data Set Too Large: LDKT (Open): The data set to be opened as a temporary file has more TF Units in its extent than were specified in the TF operand of the FILES macro for the currently selected drive. The data set was opened, but access is limited to the number of tracks specified. *Action:* If more tracks are to be processed, correct the FILES macro. Otherwise, ignore this status.	LDKT
-1------ ------1- (hex 4002)	Unit Exception, Exclusive Use Conflict: A data set is currently open by one or more stations. LDKT (Open): The Exclusive Use option was specified, but the data set is already open by one or more stations. *Action*: Close the data set before issuing the instruction. The data set is presently open for the exclusive use of another station. LDKT (Deallocate): The data set to be deallocated is presently open by one or more stations. *Action:* Either pause and retry this instruction, or close the data set to all stations and then retry.	COMPDKT FORMDKT LDKT

Status Bits	Condition	Instruction
-1------ -----1-- (hex 4004)	Unit Exception, Data Set Name Unknown:	COMPDKT LDKT

COMPDKT:

A parameter list name has no corresponding data set on the installed diskette, or duplicate data-set names are specified in the parameter list.

LDKT (Open, Deallocate, Rename, Query Extended Header Label, or Reorganize):

There is no data set with the given data-set name on the currently mounted volume.

Unit Exception, Data Set Unknown, LDKT (Allocate):

An Allocate-from-Definition was attempted but an existing data-set definition was not found.

On diskette, a definition was attempted but the SYSDSLBL data set was not previously allocated.

LDKT (Deallocate):

A deallocate function was attempted when an existing data-set label containing either a data-set definition or describing allocated space, was not found.

Action: Either mount the correct diskette, correct the data-set name, or correct the parameter list.

Status Bits	Condition	Instruction
-1------ ----1--- (hex 4008)	Unit Exception, Incompatible Diskette: The volume label indicates the volume has labels in the extended system area, or the volume has a special extent arrangement. *Action:* Mount another diskette. Unit Exception, Incompatible Diskette: LDKT (Allocate or Deallocate): • The volume label of the currently mounted volume indicates that the volume has a special extent arrangement (that is, the byte 72 contains a nonblank). • The volume label of the currently mounted diskette indicates that it has more than cylinder 0 allocated for data-set labels. Allocation and deallocation are not allowed on this diskette. *Action:* Mount another diskette; this diskette cannot be processed by the system. Unit Exception, Incompatible Disk or Diskette: LDKT (Allocate or Deallocate): The disk has not been properly formatted for use by EDAM.	COMPDKT FORMDKT LDKT
-1------ ---1---- (hex 4010)	Unit Exception, Too Many Open Requests: LDKT (Open): There are already as many data sets open as the maximum number allowed to be open at any one time, as specified in the EDAM operand of the FILES macro. *Action:* (1) Pause and retry the instruction, (2) Change the FILES macro, or (3) close one of the data sets and retry.	LDKT

Status Bits	Condition	Instruction
-1------ ---1---1 (hex 4011)	Unit Exception, Multiple Temporary File Opens: LDKT (Open): The temporary file option was specified and another data set is presently open as a temporary file. *Action:* (1) Open the current temporary file data set, (2) wait until this data set is closed, or (3) close the current temporary file data set and open the new data set.	LDKT
-1------ --1----- (hex 4020)	Unit Exception, Data Set Name Not Unique: LDKT (Allocate): A data set with the proposed data-set name is already allocated with or without having a definition. A definition with the proposed data-set name already exists. *Action:* Change the data-set name, mount a new volume, or deallocate the existing data set. LDKT (Reorganize): A data set already exists having the same name as the new KSAP, that is the first character is '$'. *Action:* Deallocate the 'old' data set or rename the KSAP data set. LDKT (Rename): The proposed data-set name is already that of an existing data set or definition. *Action:* Choose a different data-set name.	LDKT

Status Bits	Condition	Instruction
-1------ -1------ (hex 4040)	Unit Exception, No Unused Extents:	LDKT

LDKT (Allocate):

For diskette, no label position is available in SYSDSLBL to hold the data-set definition.

For diskette, there is no position available on Track 0 to hold another header label.

Note: A header label position is used when a data set is defined but not allocated; allocated but not defined; or allocated and defined.

For disk, the maximum of 510 user specified data sets already exists. This limit includes data sets defined, allocated, or defined and allocated.

Action: Either mount another diskette or deallocate any data set on the current volume.

Status Bits	Condition	Instructions
-1------ 1------- (hex 4080)	Unit Exception, Insufficient Diskette Space: The sectors required for the data-set expansion will not be available after compression or sufficient sectors are not available to complete the compress process due to track alignment. Unit Exception, Space Unavailable, LREAD,REPLACE: No space left for implicit secondary extent needed. *Action:* Either mount a new volume, or deallocate the current data set(s) at the desired location containing the needed sectors. Unit Exception, Space Unavailable: LDKT (Allocate): A specific extent location was given and it overlaps one or more existing data-set extents. No specific extent location was given, but the required number of contiguous sectors was not available anywhere on the diskette. *Action:* Either mount a new volume, or deallocate the current data set(s) at the desired location containing the needed sectors. Unit Exception, Space Unavailable: For disk, the amount of contiguous space requested does not exist. For disk, another secondary extent is needed to hold a new data-set label in SYSDSLBL but is not available. *Action:* Truncate or deallocate a data set to provide the needed sectors.	COMPDKT LDKT LWRITE REPLACE
-1------ 1--1---- (hex 4090)	Unit Exception, Too Many Defaults: An attempt was made to format a blank (unformatted) diskette with one or more default values in the parameter list. *Action:* In order to format a blank diskette, no default values can appear in the parameter list.	FORMDKT

Status Bits	Condition	Instruction
-1------ 1--1--1- (hex 4092)	Unit Exception, Invalid Diskette Type: Diskette type in parameter list is invalid for the diskette drive. *Action:* Attempt the operation on a drive that is compatible with the specified diskette type, or correct the parameter list and retry.	FORMDKT
-1------ 1--1--11 (hex 4093)	Unit Exception, Wrong Diskette Type: Diskette type in parameter list is not compatible with the mounted diskette. *Action:* Mount the correct diskette or correct the diskette type in the parameter list and retry.	FORMDKT
-1------ 1--1-1-- (hex 4094)	Unit Exception, Invalid Sector Count: The sectors to truncate from a specified data set (parameter list value) was greater than the current defined extent size. *Action:* Correct the parameter list.	COMPDKT
-1------ 1--1-1-1 (hex 4095)	Unit Exception, Invalid Record Length: The data record length in the parameter list is invalid for the mounted diskette, or the default record length from the diskette is invalid. *Action:* Correct the record length in the parameter list and retry.	FORMDKT
-1-----1 -------- (hex 4100)	Unit Exception, Invalid Record Length: LDKT (Allocate) For RKAP, KSAP, TEMP, or ASDS data-set types, the diskette must be formatted in 256-byte sectors. *Action:* Correct the parameter list and retry. Unit Exception, Incorrect Length: LDKT (Allocate): The physical sector length specified for a data set does not match the sector length on the currently mounted diskette. *Action:* Change either the volume or the parameter list.	LDKT

Status Bits	Condition	Instruction
1------- -------- (hex 8000)	Intervention Required: Any function: The door of the diskette drive is open or the disk/diskette is not ready or not rotating at operational speed. *Action:* Ensure that the diskette is in the correct drive, then close the diskette door and retry the instruction. If the status still occurs, consult your service representative.	Any Disk/Diskette Function
1------- -------1 (hex 8001)	Intervention Required, User-Declared: Any function: A station has executed a SETDSKT instruction, declaring the diskette drive to be not ready. *Action:* Defined by the controller application program. SETDSKT could be issued to declare the drive ready.	Any Diskette Function
1------- ------1- (hex 8002)	Intervention Required, Diskette Not Stopped: FORMDKT or COMPDKT was requested on the primary drive and the primary drive was not logically stopped. *Action:* Remove the diskette currently loaded, then insert the diskette to be formatted or compressed into the primary drive and try again.	FORMDKT COMPDKT
1------1 -------1 (hex 8101)	Intervention Required, Wrong Length Sectors: Any function: The currently mounted diskette is not formatted with either 128- or 256-byte sectors. *Action:* Mount another diskette.	Any Diskette Function
1-----1- -------- (hex 8200)	Intervention Required, Unit Check: The first attempt to access the disk or diskette failed. *Action:* Check that the diskette type is compatible with the diskette drive. Retry the operation; if the problem persists, inform your service representative.	Any Disk/Diskette Function

Appendix E. Statistical Counters

Controller Diskette

Counter	Explanation
1	Intervention required
2	Command reject
3	Record not found
4	Cyclic redundancy check (CRC) errors
5	Disk format error
6	Machine check
7	Seek failure
8	Overrun
9	Permanent write errors in the temporary file

Controller Disk

Counter	Explanation
1	CRC Error
2	Not Ready
3	No Alternate Sectors Available
4	Machine Check
5	Data Unsafe
6	Alternate Assignment Failed
7	Seek failure
8	Equipment Check
9	Record Not Found
10	Successful ECC Correction
11	Alternate Sector Assigned

Appendix F. Diskette Initialization

Although the following description is specifically about diskette initialization and use by a controller, the description of an initialized diskette also applies to new diskettes received from the manufacturer except that a new diskette will not contain defective areas.

One-sided diskettes contain 77 tracks. When these tracks are initialized by the controller, they are numbered sequentially beginning with 0. If a defective track is encountered, that track is not numbered and the numbering continues on the next non-defective track. (Up to two defective tracks can be skipped in this manner.) If more than two defective tracks are found, you cannot use the diskette. An initialized diskette will therefore contain tracks numbered 0—74 (if two defective tracks), 0—75 (if one defective track), or 0—76 (no defective tracks). Track 0 is always the label track.

Two-sided diskettes contain 77 tracks on each recording surface. The initialization process for two-sided diskettes is similar to that for one-sided diskettes. All of the physical track locations are initialized, only nondefective tracks are numbered. If a defective track is encountered, that track *and* the corresponding track on the other side are not numbered. A two-sided diskette will therefore have a minimum of 75 numbered tracks (0 through 74 on each side) and a maximum of 77 numbered tracks (0 through 76 on each side).

Track 0, side 0 is a label track for all diskettes. On a two-sided diskette, track 0, side 1 is also a label track. On data diskettes, tracks 1 through 74 are usable for data on both sides. Tracks 75 and 76 (if present) are reserved on either a one- or two-sided diskette.

Two-sided diskettes are recorded and read by the controller in "cylinder mode". For example, on a Diskette 2 with 256-byte sectors, track 5, sector 15, side 0, is followed by track 5, sector 1, side 1. The label tracks on both Diskettes 1 and 2 always contain twenty six 128-byte sectors.

A Diskette 2D contains twenty-six 128-byte sectors on track 0, side 0, and twenty-six 256-byte sectors on track 0, side 1. These sectors are used for error mapping and for volume and data-set labels.

All tracks other than track 0 on Diskettes 1 and 2 are initialized to contain either fifteen 256-byte sectors per track or twenty-six 128-byte sectors per track (only one sector size is permitted on each diskette); Diskettes 2D contain twenty six 256-byte sectors per track. 4700 installation and operating diskettes are initialized with 256-byte sectors. You can use either 128- or 256-byte sector diskettes for data storage. The actual number of tracks available for files depends on the options specified during the controller configuration procedure and the number and size of the controller application programs. Sector sizes are established when the diskette is initialized and cannot be changed by the controller. the installation diskette supplied by IBM that can be used to initialize diskettes to the one- or two-sided formats.

Data sets are allocated, and controller data and application programs are placed on the operating diskette using system monitor facilities provided with the installation diskette. During operation, the controller maintains the diskette, as necessary, by relocating data records from defective locations to an error data set.

Bibliography

The publications listed below contain information that may be useful to persons programming a 4700 system that includes disks and diskettes.

IBM Vocabulary for Data Processing Telecommunication and Office Systems, GC20-1699

IBM System/370 Bibliography, GC20-0001

IBM System/370 Bibliography of Industry Systems and Application Programs, GC20-0370

IBM 4700 Finance Communication System:

System Summary, GC31-2016

Subsystem Operating Procedures, GC31-2032

Subsystem Problem Determination Guide, GC31-2033

Host Support User's Guide, SC31-0020

4701 Controller Operating Instructions, GC31-2022

Index

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

4700 Finance Communication System
Controller Programming Library
Volume 2
Disk and Diskette Programming

Order No. GC31-2067-1

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*
Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

GC31-2067-1

Reader's Comment Form

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 78C
1001 W.T. Harris Boulevard
Charlotte, NC, USA 28257

Fold and Tape | Please Do Not Staple | Fold and Tape

IBM

Disk and Diskette Programming (S370/4300/8100/S34-30) Printed in U.S.A. GC31-2067-1

Publication Number
GC31-2067-1

File Number
S370/4300/8100/S34-30

Printed in
USA

IBM