AN OS/360 BDAM USERS GUIDE

Wayne E Fisher
IBM Corporation
Education Center
3424 Wilshire Blvd
Los Angeles, Calif 90005

This paper is not intended to replace the IBM publications pertaining to BDAM. Rather it is intended to supplement, clarify, and bring together the BDAM material in the various IBM publications.

It is assumed that the reader is familiar with the Assembly Language and the Supervisor and Data Management macro instructions.

June 30, 1969

TABLE OF CONTENTS

INTRODUCTION

TRACK FORMAT REVIEW

Information is recorded on all direct access volumes in a standard format. In addition to device-dependent data (home address), each track contains a track descriptor record (also called a "capacity record" or R0), and one or more data records. User data is placed in the data records. The system maintains the track descriptor record on each track.

There are only two possible data record formats -- Count-Data and Count-Key-Data -- only one of which can be used for a particular data set. The following illustrates the two possible data record formats.

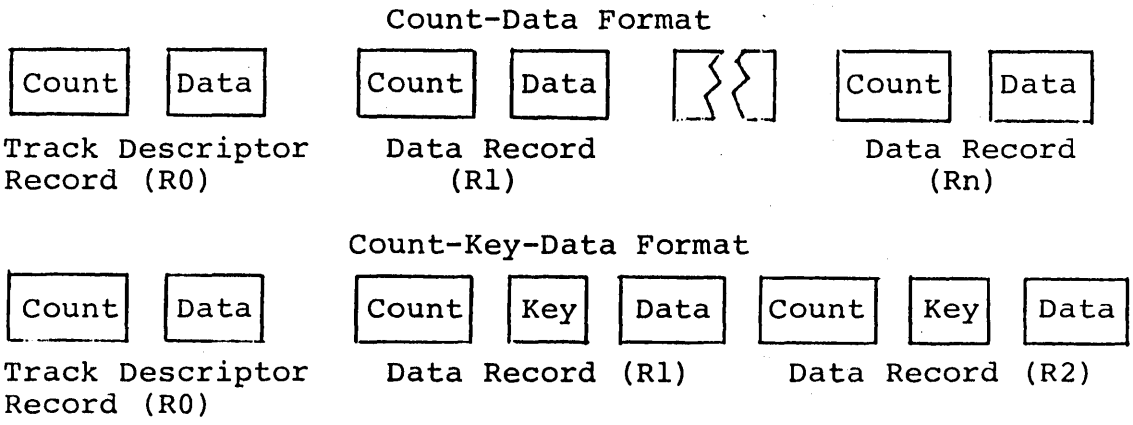

Figure 1. Data Record Formats on DASD Tracks

The Count area of each record contains eight bytes that identify the location of the record in terms of the cylinder, head, and record numbers; its key length (0 if no keys are used); and its data length.

If records are written with keys, the Key area (1-255 bytes) contains a record key that identifies the following Data area. This identifying information might be a part number, account number, sequence number, etc. The hardware is capable of searching the Key area of each record on each track for a particular key. Once more, the keys do not have to be in any particular sequence.

The Data area contains the user's data records. Its length can be up to 32,760 bytes, but realistically is determined by the particular device's track capacity. Each Data area contains a block. Each block can consist of one or more logical records.

BLOCKS VERSUS RECORDS

As previously stated, each Data area contains a block which may consist of one or more logical records. However, since a direct data set can only be processed by the basic access technique, the user must perform any blocking or deblocking if in fact there are more than one logical record per block. The reader should be aware that the term "block" and "record" are used interchangeably in this publication as well as the IBM SRL publications. They both refer to the contents of the Data area on a direct access track.

PROGRAMMING CONSIDERATIONS

FORMATTING THE DATA SET SPACE

Before data records can be placed in a direct data set, the DASD space allocated must be "formatted". Formatting the data set is the process of initializing each track, one after another in a sequential fashion. In fact, a sequential access method (BSAM) is used to perform the formatting.

For fixed length records (RECFM=F), formatting is essentially the process of creating "buckets" which act as place holders for actual records to be added at a later time.

For variable length (RECFM=V) and undefined length (RECFM=V) records, formatting is the process of initializing the Track Descriptor Record (there is one on the front of every track). Each will reflect the fact that there are no records written on the track and that the entire track space is available.

ADDRESSING SCHEMES

Keys

Each record in a data set is comprised of one or more related data fields. One or more of these data fields may serve as an identifier or key field which uniquely distinguishes that record from others in the same data set. Typical keys are: names, part numbers, or chronologically assigned serial numbers such as employee number, invoice number, etc. The key is the means of selecting and retrieving a desired record from the data set.

Every record in a directly organized data set also has a unique address. This address identifies to the access method the location within the data set where the record should be found. The format of the record address will be discussed later.

In a direct data set, there is a definite relationship between the record key and the record address or location. It is this relationship which allows you to directly retrieve any record in the data set without a sequential or index search. This relationship is completely determined by each user -- it might be a direct or an indirect relationship.

Direct Addressing

It is entirely possible to have keys which identify the location of the record in the data set. This is a direct addressing scheme, thus obviating the need for a transformation or mathematical manipulation of the key. One of the characteristics of a direct addressing scheme is that there is a unique DASD address for each record key.

Strict Relationship

The ideal situation would be to use the record's key as its DASD address, that is, there is a strict relationship between the record's key and its address in the data set. An example of this type of direct addressing would be a data set of personnel records where the four-digit employee number is the key and also serves as the location of the record, i.e., the record for employee number 6545 would be the 6,545th record within the data set.

Figure 2 illustrates a card record with a key of 6545 which serves as the address of the corresponding record in the direct data set.

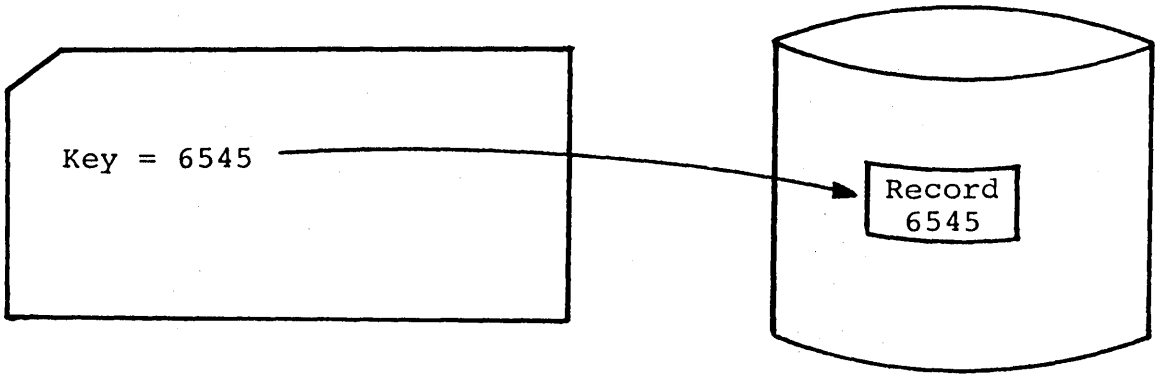

Figure 2. Direct Addressing

This technique assumes that there is an addressable location or "bucket" available for each employee number regardless of whether or not there is an employee with that number. For example, if we have employee numbers which range from 0001 to 9999, then we must have 9,999 buckets even though we may have only a couple hundred employees.

The use of direct address using a strict relationship is usually limited to data sets with small numerical keys. Additionally, in all direct addressing schemes the data records must be fixed length without track overflow.

Cross-Reference Table

There is no unique and simple way of transforming a long key to a shorter unique address. One technique of handling records with a cumbersome key is to build a cross-reference table (index). When a record is written in the data set, you note the physical location and store this, along with its key, in the table. Finding the address of a particular key is achieved by programming a table lookup of the cross-reference table.

Figure 3 illustrates a card record with a key of SMITH which serves as the argument in a programmed table lookup of a cross-reference table. When the argument is found in the table, the corresponding value (623 in this example) will be the address of SMITH's record in the direct data set.

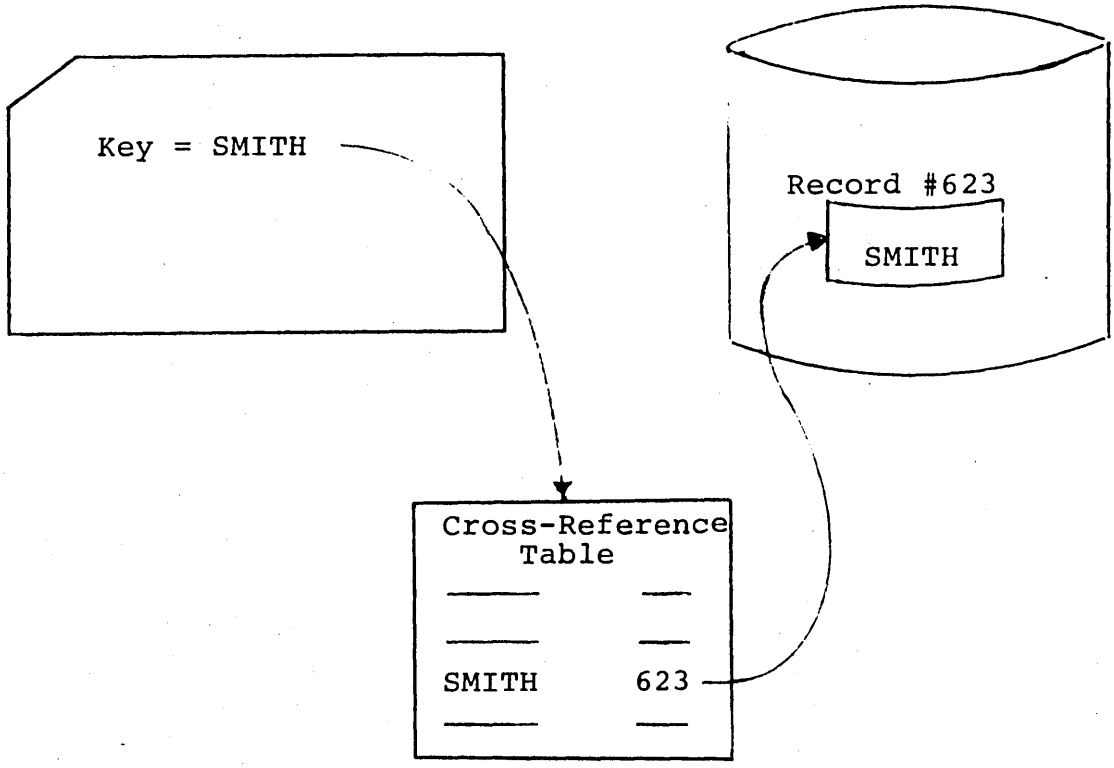

Figure 3. Direct Addressing Using a Cross-Reference Table

This technique of direct addressing allows space to be allocated on the basis of the number of records in the data set rather than on the range of keys. New records can be added sequentially to the end of the data set space and their location noted and placed in the cross-reference table.

The obvious disadvantages are that cross-referencing requires the user to maintain the table, and core storage and processing time is required to search and update the table.

Indirect Addressing

A more common technique for organizing the data set involves the use of indirect addressing. In indirect addressing, the address of each record in the data set is determined by a mathematical manipulation of the key. This manipulation of the key is referred to as randomizing. There are many different techniques of transforming a record key (external identification) into the corresponding record address (internal location). No attempt is made here to describe or explain the various algorithms that might be appropriate for your data set.

Figure 4 illustrates a record with a key of A360 which, when run through a randomizing algorithm, yields the address (127 in this example) of the record in the direct data set.

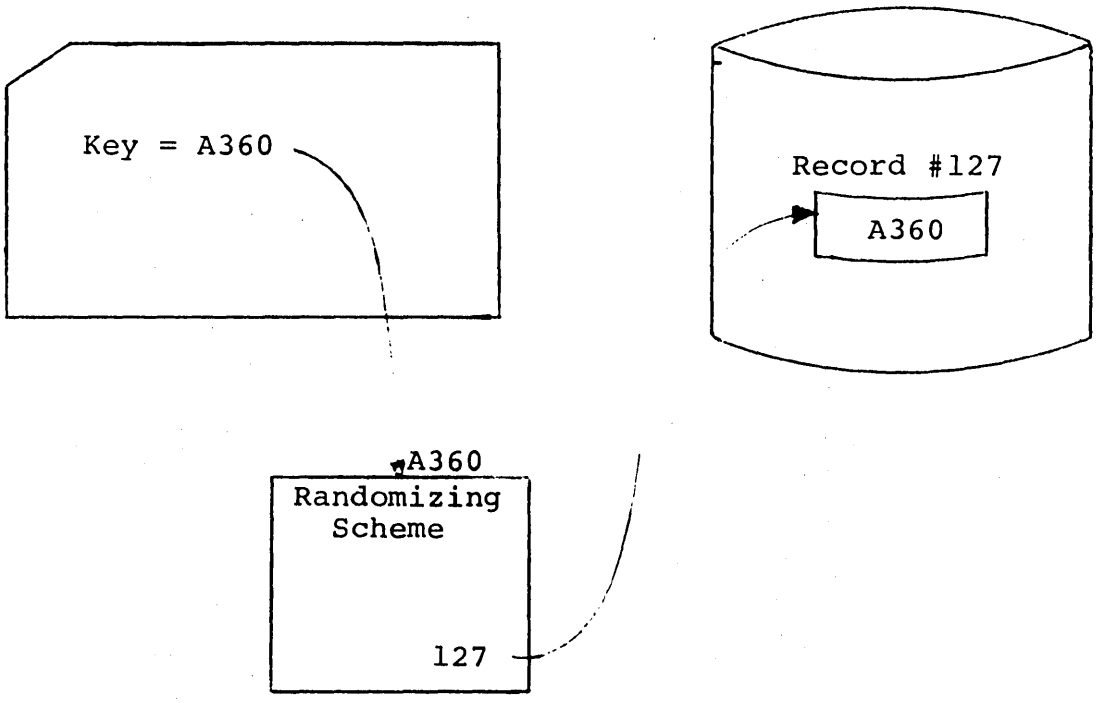

Figure 4. Indirect Addressing

Overflow Record Handling

Characteristic of most randomizing schemes is the synonym problem -- the transformation of two or more unique keys into the same DASD address. The record that is written where it belongs is called the home record. The second and subsequent records with keys which convert to the same address are called overflow records. A procedure must be provided for storing elsewhere those overflow records whose keys convert to an address that is already occupied. There are many different techniques used to handle overflow records. No attempt will be made here to examine them. Rather, we shall discuss the method provided by BDAM called progressive overflow.

Progressive overflow assumes that the entire data set space is not 100% used, that is, there are buckets that are not yet filled and that the overflow record may be stored in one of them. The search for an empty bucket starts at the address produced by the randomizing scheme and continues through consecutive addresses.

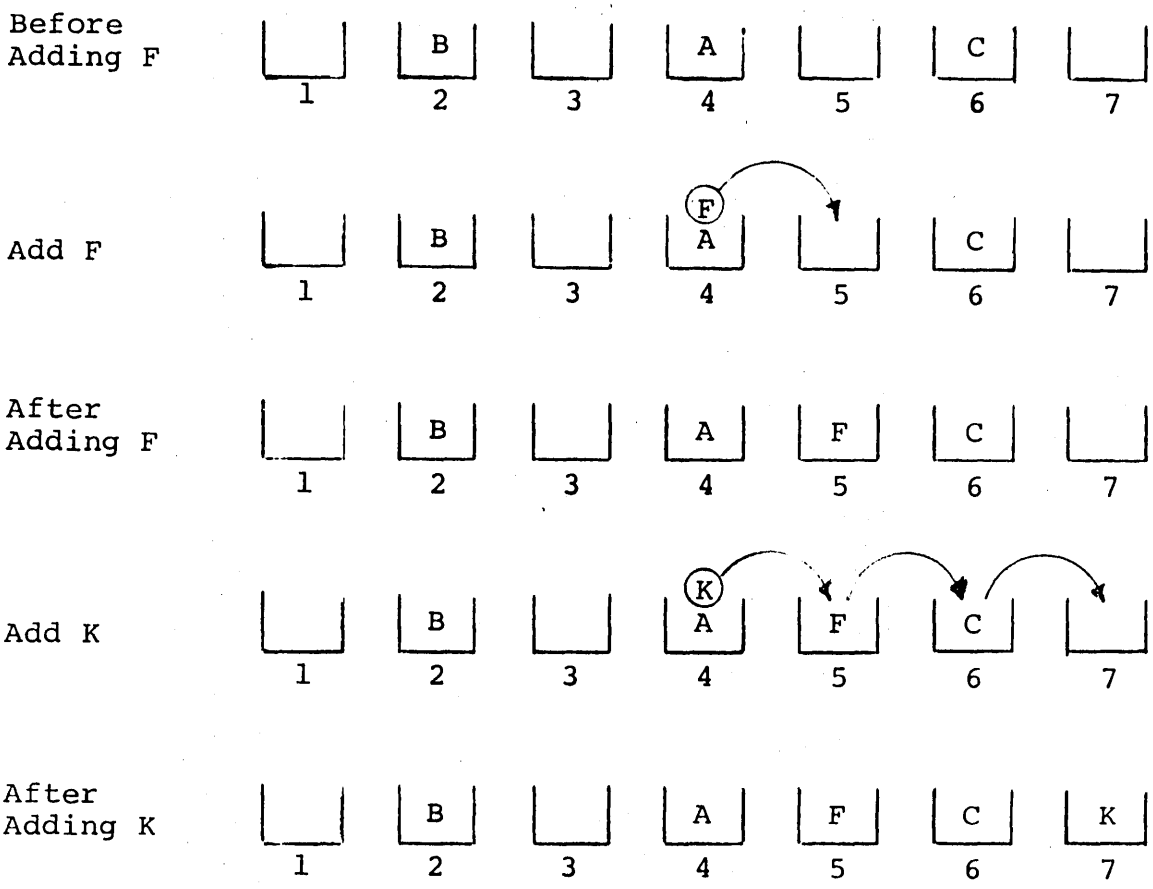

Figure 5. Overflow Record Handling

Figure 5 illustrates a data set containing seven buckets presently having three records. Let's suppose we add a record with a key of F and that our randomizing scheme assigns F to bucket 4. Since bucket 4 is already occupied, a search is made for an empty bucket. Bucket 5 is empty and F is placed in it. Next we add a record with a key of K and once again let's suppose our randomizing scheme assigns K to bucket 4. When we attempt to put K in bucket 4, we find it is occupied so we attempt to put it in bucket 5. But bucket 5 is also occupied so we attempt to put it in bucket 6. This process continues until an empty bucket is found, which in this example is bucket 7.

In searching for an empty bucket in which to store an overflow record, how does the system know when it encounters an empty bucket? How far will the system search for an empty bucket? Once an overflow record is stored in the data set, how does the system ever retrieve it? The answers to these questions will hopefully become answered at a later time. For now, the handling of overflow records by the system requires that the data set be recorded in the Count-Key-Data format and that empty buckets be system dummy records. How far the system will search is a function of the optional "extended search" feature. Both system dummy records and the extended search feature will be explained later.

RECORD REFERENCE METHODS

The READ and WRITE macros are used to retrieve and store individual records (i.e., blocks). Which particular record retrieved or stored is a function of the OPTCD and KEYLEN parameters in the DCB and the "type", "block address", and "key address" parameters in the READ or WRITE macro instructions. Basically, the access method searches either for a record which occupies a specific "bucket" or a record with a specific key.

Referencing a Specific Record

Retrieving or storing a specific record is requested by specifying a DI "type" parameter in the READ/WRITE macro. Which particular record retrieved/stored is determined by the "block address" parameter in the READ/WRITE macro. The "block address" parameter contains the core storage address of the record address (left-most byte) of the particular record. This is illustrated in the following figure.

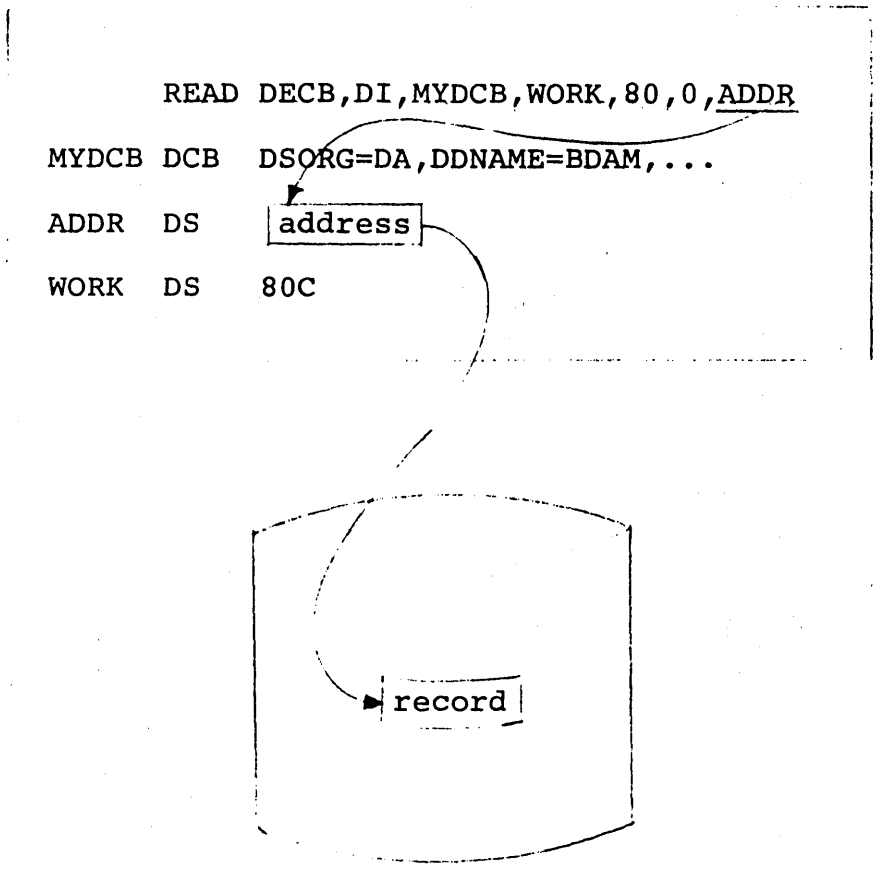

Figure 6. Referencing a Specific Record

It is immaterial to the access method whether the data is recorded in the Count-Key-Data format or the Count-Data format. In fact, the "key address" parameter in the READ/WRITE macro is ignored by the access method.

The contents and length of the record address will vary depending upon the record referencing method you elect to use. There are three methods available: actual addresses, relative block addresses, and relative track addresses. You must notify BDAM of your selection by specifying an A, R, or no R or A in the OPTCD parameter of the DCB. This is how the access method determines how to use the contents of the main storage pointed to by the "block address" parameter. Following is an explanation of the three record referencing methods.

Actual Address

You specify the actual address in the standard eight-byte form -- MBBCCHHR. See the IBM SRL for the specific DASD you are using for the meaning of MBBCCHHR. Remember, the use of an actual address may force you to indicate that the data set is unmovable (DSORG=DAU).

Relative Block Address

You specify the relative location of the record (block) within the data set as a three-byte binary number. This type of reference can be used only with format F records. The access method converts this relative block number into a two-byte binary relative track number and a one-byte binary actual record number. Allocation of noncontiguous tracks does not affect the number.

Relative Track Address

You specify the relative track as a two-byte binary number and the actual record number within that track as a one-byte binary number. The access method computes the actual track location taking into consideration non-contiguous tracks.

Relative Address Limit

When relative addressing is used, the size of the data set is limited to 65,536 tracks. With relative track addressing, you provide a two-byte binary relative track number (0 through 65535). With relative block addressing, you provide a three-byte binary relative block number -- this limits the number of blocks to 16,777,216. However, the access method converts the three-byte relative block address into a two-byte relative track address truncating any high-order bits.

Retrieving a Record with a Specific Key

Retrieving or storing a record with a specific key is requested by specifying a DK "type" parameter in the READ/WRITE macro. The record must be recorded in the Count-Key-Data format as the search is made on the Key area of the DASD track. The user must provide the key of the desired record. This is done by specifying in the "key address" parameter of the READ/WRITE macro the core storage location containing the key. The length of the key is determined by the KEYLEN parameter in the DCB.

Since the records and their associated keys may be stored in a random sequence in the direct data set, the user must specify where the search is to begin. This is done by placing in the "block address" parameter the core storage address (left-most byte) which contains the record address. The contents and length of the record address will vary depending upon the record referencing method you elect to use; i.e., relative block addresses or relative track addresses. Whichever method you use must be indicated in the OPTCD parameter of the DCB. The contents of the relative block and relative track address is explained above.

If relative block addressing is used, the access method converts your relative vlock address to a relative track address. The search always starts at the beginning of the track and continues to the end of the track. If the search is to continue beyond one track, the extended search option nust be requested.

Figure 7 illustrates the retrieval of a record with a key of 'SMITH' using relative block addressing. The record address specifies the 13th block; however, the access method starts the search at the beginning of that track.

```
        READ DECB,DK,DCB,WORK,80,KEY,ADDR
DCB     DCB  DSORG=DA,OPTCD=R,...
KEY     DC   C'SMITH'
ADDR    DC   XL3'00000D'
WORK    DS   80C
```

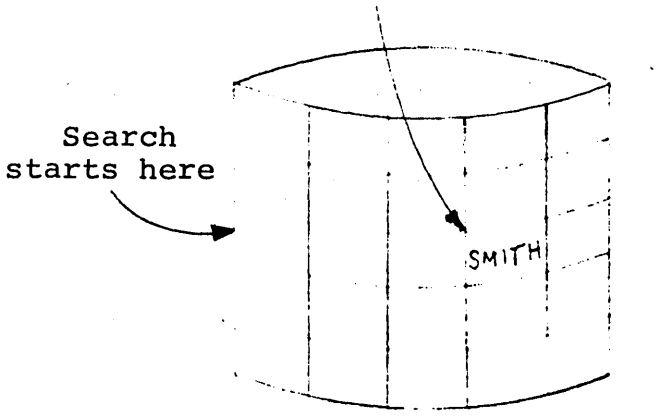

Figure 7. Referencing a Record with a Specific Key

Extended Search Option

The extended search option is available if the data set contains records using the Count-Key-Data format and relative addressing with keys is used. To request extended search, the user must specify OPTCD=E in the DCB. Use of this feature also requires the specification of the LIMCT parameter in the DCB in order to place an upper limit as to how far to search. The same value should be specified for reading and writing as was used when adding records.

When the extended search option is selected and relative track addressing is used, the LIMCT value specified is the number of complete tracks, including the first, to be searched. The search always starts at the beginning of the track (the "one-byte actual record number within the track" portion of the address is ignored).

When the extended search option is selected and relative block addressing is used, the LIMCT value specified

is the number of blocks to be searched. However, the system first converts this value into an integral number of tracks which contain the specified number of blocks, then proceeds in the manner outlined above for relative track addressing. It is possible, therefore, that the number of blocks actually searched may exceed the value specified in the LIMCT parameter.

DYNAMIC BUFFERING

Dynamic buffering is the automatic buffering facilities provided by the control program for the BDAM access method. The control program will construct and manage a buffer pool for each DCB requesting dynamic buffering.

Programming Considerations

MACRF Parameter

The handling of dynamic buffering for BDAM is done by module IGG019LE. This module is brought into core storage at OPEN time only if S has been specified in the MACRF parameter of the DCB. Thus, it is the user's responsibility to request dynamic buffering in the DCB if dynamic buffering is later requested in a READ macro somewhere within the problem program.

BUFL Parameter

The length of a buffer must also be specified by the user if the dynamic buffering feature is to be invoked. This is done by the BUFL parameter in the DCB. The value provided must be maximum BLKSIZE plus KEYLEN if 'S' is specified for the key address parameter of a READ macro.

BUFNO Parameter

The BUFNO parameter in the DCB determines the size of the buffer pool. The number of buffers, by default, is two. The user may override this value by coding the BUFNO parameter. The number of buffers required is a function of the number of READ macros executed without corresponding WRITE or FREEDBUF macros.

Advantages Versus Disadvantages

The advantage of dynamic buffering is that relatively few buffers are needed since the READ requests waiting in the queue do not monopolize buffers. This is because buffers are taken from a buffer pool and assigned just before data transfer begins.

The dynamic buffer module does require core storage (about 460 bytes) as well as CPU time to execute.

Whether or not you use dynamic buffering is a function of the application program processing. In general, if you are doing a serial-type processing (one transaction at a time) you probably will not use the dynamic buffer feature. If your processing is such that you have multiple outstanding I/O requests, you'll probably want to use the dynamic buffer feature to allow the system to manage buffers from a buffer pool.

Obtaining Buffers

As each request to read data is about to be executed, the system checks to see if a request for a dynamic buffer has been specified. If so, the dynamic buffer module ascertains if a buffer is available. If it is, the buffer is assigned and the read is executed. If no buffer is available, the request is queued. When a buffer becomes available it is allocated to the request at the top of the queue.

Releasing Buffers

It is the user's responsibility to release any dynamic buffers obtained upon completion of its use. Buffers may be released in one of two ways: either explicitly by using the FREEDBUF macro, or implicitly by issuing a WRITE macro that specifies dynamic buffering.

To effect the release, you must specify the address of the DECB that was used when the dynamic buffer was obtained, as well as the address of the DCB associated with the direct data set.

FEEDBACK

When relative addressing is used, the BDAM routines must convert this relative address into an actual device address. This conversion does take CPU time, particularly if the data set has multiple extents which may or may not be contiguous. The user can request that the system feed back to him the actual address for subsequent use. For example, if one is to update a record, he could request that the actual address be returned after reading the block so that he could give the system the actual address to perform the write operation. This would save the relative-to-actual conversion process prior to initiating the write operation. If he were only retrieving records, there would be no point in asking for feedback.

Requesting Feedback

Feedback is requested by appending an F to the "type" parameter in the READ/WRITE macro, i.e., READ DECB,DKF,.... It will become apparent to the reader that use of the feedback option is practical only when using relative addressing with keys.

Storing Feedback

Since the system returns back to the user the actual address, it must have someplace in which to place the information. The system uses the area of main storage specified by the "block address" parameter of the READ/WRITE macro instruction for this purpose.

Form of Feedback

The format of the actual address returned to the user by the feedback option is a function of the OPTCD parameter in the associated DCB -- it is in one of two forms. If F is specified in the DCB OPTCD parameter, the device address returned is of the same form presented to the control program. That is, if relative block addressing (OPTCD=RF) is specified, then the true relative block address is returned. In the case of searching for a particular record using a key, the original relative block address would specify where the search started; however, the true relative block address returned as feedback would indicate where the particular record really is. If relative track addressing (OPTCD=F) is specified, the true relative track and actual record number within that track is returned.

If F is not specified in the DCB OPTCD parameter, feedback is in the form of the actual device address of the block. The user should be aware that the actual device address is an eight-byte address (MBBCCHHR). Thus, it is the user's responsibility to allocate an area large enough to hold the feedback. For example:

```
         READ  DECB,DKF,MYDCB,AREA,80,KEY,RELBLK+1
MYDCB    DCB   OPTCD=R,...
RELBLK   DS    1F       Holds Rel Blk & 1st 3 bytes Feedbk
         DS    5C       Holds last 5 bytes of Feedback
AREA     DS    CL80
```

Let's suppose that we are using relative block addressing and that a DKF type READ macro is issued but that the OPTCD parameter does not specify feedback. The relative block address requires only three bytes, but the feedback requires eight bytes. The feedback will overlay the original

three-byte relative block address and the adjacent five bytes. Thus, it is the programmer's responsibility to reserve these five bytes. Figure 8 summarizes the form of feedback based on the form presented and whether or not the OPTCD parameter specifies F for feedback.

If Feedback is Requested in I/O Macro:			Feedback Format	
Type of Reference	Information Supplied	Blk Addr Field Min Lgth	DCB OPTCD	
			≠F	=F
Relative Block Address w/o Key	3-byte Binary Rel Blk No.	3 bytes	8-byte Actual Address	8-byte Actual Address
Relative Block Address with Key	3-byte Binary Rel Blk No. and Key	3 bytes		3-byte True Rel Blk No.
Relative Track Address w/o Key	TTR, where TT=Rel Trk No. R=Actual Rcd Id	3 bytes		8-byte Actual Address
Relative Track Address with Key	TT, where TT=Rel Trk No.	2 bytes		3-byte True TTR
Actual Device Address	MBBCCHHR	8 bytes		8-byte Actual Address

Figure 8. Feedback Format Summary

EXCLUSIVE CONTROL

The exclusive control feature, when requested by the program, prevents the inadvertent processing of the same record by two or more competing tasks within the same job step. Exclusive control is effective only if all tasks referencing the same data set do so through the same DCB, and if all tasks request exclusive control.

The control program implements this feature by maintaining a "read-exclusive list." If the block requested is already in the list, the system places that task in the wait state until the block becomes available.

Requesting Exclusive Control of a Block

If the exclusive control facilities are going to be used, the user must specify an X in the MACRF parameter of the DCB. This will request the loading of the exclusive control module (IGG019LG) at open time.

Exclusive control is applied to blocks that are read (and may or may not be subsequently written) by appending an X to the DI or DK "type" parameter (i.e., DIX or DKX) of the READ macro instruction.

The user should be aware that reading with exclusive control includes feedback. The format of the feedback information is a function of whether or not F has been specified in the OPTCD parameter of the appropriate DCB.

Releasing Blocks Under Exclusive Control

Blocks that have been read under exclusive control must be released from exclusive control. This can be done either by use of a WRITE macro that specifies exclusive control or by use of the RELEX macro instruction.

Release By Writing

To release a block that has been previously read under exclusive control, the user must append an X to the DI or DK "type" parameter of the WRITE macro. If the system finds an outstanding request for the block being released, it is now honored.

Release By RELEX Macro

The RELEX macro may be given to release a block that was read under exclusive control. If the system finds an outstanding request for the block being released, it is now honored.

CHECK MACRO VERSUS WAIT MACRO

When using a basic method to process a data set, it is the user's responsibility to ensure that the I/O operation has completed before processing of the data can commence. This synchronization of I/O with processing can be accomplished using either the CHECK macro or the WAIT macro.

The WAIT macro instruction causes the program to be placed in the wait condition, if necessary, until the associated I/O operation is completed. The user's program must now test the results of the I/O operation for any error conditions.

The CHECK macro performs the same function as the WAIT macro but in addition will test the I/O operation for errors and exceptional conditions. If the operation did not complete successfully, control is given to the user's SYNAD routine specified in the SYNAD parameter of the DCB. See the Supervisor and Data Management Services SRL for an explanation of SYNAD routines. If the SYNAD parameter was not specified, the task is abnormally terminated.

If the CHECK macro is used, the user must have specified C in the MACRF parameter and also specified the address of the user's synchronous error recovery routine in the SYNAD parameter of the DCB. The CHECK module (IGG019LI) will not be loaded into core storage if the C is not present in the MACRF parameter at OPEN time.

If the WAIT macro is used, the MACRF parameter must not contain a C and the SYNAD parameter is ignored. Each I/O request obtains an IOB from the IOB pool. If MACRF contains a C and the CHECK is used, the CHECK Module returns this IOB to the IOB pool; however, if a WAIT macro is used the IOB is not released. Thus, the user should be consistent -- either use the CHECK or the WAIT macro, but not both.

ERRORS

The access method notifies the user of any errors which occur as the result of a READ or WRITE macro by establishing exception codes. These exception codes are returned by the access method after the corresponding WAIT or CHECK macro instruction is issued. The exception codes are placed in the second and third bytes of the DECB used for reading or writing. See Table 11 in the Supervisor and Data Management Macro Instructions SRL for a description of the exception codes.

SPACE ALLOCATION

Space for a direct data set must be allocated using either the SPACE or SUBALLOC parameter in the DD statement. The SPLIT parameter cannot be used. If you request space in units of blocks, and the blocks have keys, you must also give the key length in the KEYLEN subparameter of the DCB parameter in the DD statement.

When using relative addressing, the number of tracks in the data set cannot exceed 65,536.

When creating a multi-volume direct data set, space allocation is accomplished by use of primary and secondary allocation. Neither of these can exceed the capacity of one volume. Primary allocation is allocated only on the first volume; all other space is allocated using the secondary allocation parameter regardless of the number of volumes used.

Once a direct data set has been created, it cannot be expanded without re-creating the data set.

CREATING A DIRECT DATA SET

Formatting the DASD Space

Formatting a fixed length record data set is essentially the process of writing "dummy" and/or actual data records onto each track in order to create gaps on the track. A record, dummy or actual, must be written for each record the user eventually expects to have in the data set. For example, if the user is creating a direct data set for a possible 10,000 records (or addresses), he must issue 10,000 writes.

The tracks can be formatted in either Count-Key-Data or Count-Data format. The format selected is determined by whether or not the user wants keys associated with his data records. The KEYLEN parameter in the DCB identifies to the access method which format is desired.

Formatting a direct data set is almost identical to creating a sequential data set, and in fact, a sequential access method (BSAM) is used to perform this function. The contents of the "dummy" record written is the only thing unique about formatting a direct data set versus creating a sequential data set.

There are two kinds of "dummy" records: system and user. A system dummy record is constructed by the control program. It is defined as a record whose first byte of the key is all one bits and the first byte of the data area has a value indicating the position of the dummy record within the track. A user dummy record is defined and known only by the user. Normally the user places some information in the record which identifies to him the record as a dummy record. It is important that the programmer recognize the distinction between these two types of dummy records when additions to the data set is discussed later on.

The WRITE Macro

Type Parameter

The SD type WRITE macro must be used to request the control program to add a system dummy record to the data set.

The SF type WRITE macro must be used to add a user dummy record or an actual data record to the data set.

Area Address Parameter

This parameter specifies the address of the main

storage area containing the block to be written. If keys are used, the key must immediately precede the data; thus the area address specified would contain the main storage address of the key.

If system dummy records are being written, the area must be large enough to hold the key plus one byte. The control program will construct a system dummy record in this area.

Length Parameter

This parameter may be omitted.

Next Address Parameter

This parameter should be omitted as it is used only with the SFR type WRITE macro.

Return Codes

Execution of the WRITE macro will set a return code in register 15 upon returning control to the user's program. The user should examine it prior to executing the CHECK macro or any other instruction which uses register 15.

A return code of X'00' indicates that the record will be written and that there is more space left on the track.

A return code of X'04' indicates that the record will be written but there is no more space left on the track. The control program will then write a capacity record.

A return code of X'08' indicates that the record will be written; however, there is no more space left on the track and there are no more tracks left in the space allocated. The next record will require secondary space allocation.

Test Completion of WRITE Operation

Each WRITE operation must be tested for completion using a CHECK macro instruction. At the completion of the write operation, control is returned to the user program if no exceptional conditions were encountered. If the execution of the write operation encountered a permanent error condition, control is passed to the user's SYNAD routine. If control is returned from the SYNAD routine, or if there is no SYNAD routine, your task will be abnormally terminated.

DCB Considerations

1) DSORG = PS or PSU.
2) DEVD = DA (default value if omitted).
3) KEYLEN must be specified if keys are used. This will cause the tracks to be formatted in the Count-Key-Data format.
4) MACRF= WL must be specified.
5) RECFM is required.
6) LRECL should be omitted.
7) BLKSIZE is required.

DD Statement Considerations

1) DCB must specify DSORG=DA.
2) SPLIT cannot be used.
3) SPACE: If you request space in units of blocks, and the blocks have keys, you must give the key length in the DCB parameter, i.e., KEYLEN=n. If your program is using relative addressing, the total amount of space allocated should not exceed 65,536 tracks. For best performance you should specify the CONTIG subparameter, especially if the extended search option will be used.
4) DISP=(NEW,...) unless the data set was previously allocated.

PROCESSING FORMAT F WITHOUT KEYS

A "Format F without Keys" data set is a direct data set that is recorded on the DASD in the Count-Data format. When it was created, the following was specified in the DCB: RECFM=F or FT, KEYLEN=0, and BLKSIZE=data field length.

Figure 9 illustrates the structure of a block on a track. CCHHR gives the physical position of the block on the device, key field length is zero, and data field length contains BLKSIZE.

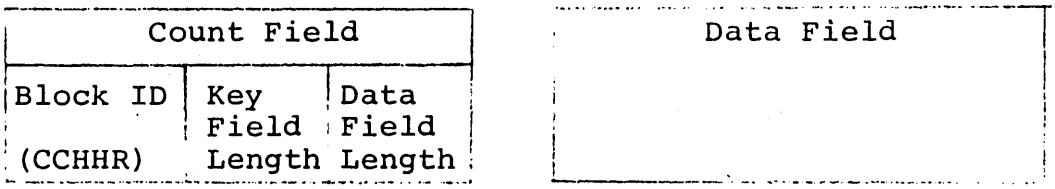

Figure 9. Structure of a Count-Data Block

All I/O operations specify a specific block (i.e., record). The access method searches the Block ID portion of the Count Field using a "search identifier equal" CCW command code.

The DI type READ/WRITE macros must be used to retrieve and store a specific block located in a specified address. The block retrieved or stored is pointed to by the block address parameter. How BDAM interprets the contents of the block address parameter (i.e., is it a three-byte relative block address, or an eight-byte actual address, or a two-byte relative track and one-byte actual record number address) is a function of the OPTCD parameter in the DCB.

To Retrieve

Type Parameter

The DI type READ macro is used to retrieve a specified block. MACRF in the DCB must contain at least RI.

Area Address Parameter

This parameter specifies the main storage address into which the block of data is to be placed. Dynamic buffering is specified by coding 'S' instead of an address; in this case, the address of the main storage area acquired by the system is placed in the DECB specified by the decb name parameter. If 'S' is specified, then S must also be specified in the MACRF parameter of the DCB.

Length Parameter

This parameter specifies the number of data bytes to be read. If 'S' is coded, the number will be taken from the BLKSIZE parameter of the DCB. If a value is specified and it differs from the size of the block read, the "record length check" exception code bit is set on.

Key Address Parameter

This parameter should specify zero. It is completely ignored by the access method.

Block Address Parameter

This parameter specifies the address of the main storage area containing the left-most byte of the relative block, relative track, or actual address of the block to be retrieved. The type of address is determined by the OPTCD parameter in the associated DCB. The device address of the block will be placed in this main storage area by the control program if feedback is requested, i.e., you have specified a DIF type READ macro.

To Update

Type Parameter

The DI type WRITE macro is used to update a specified block. MACRF in the DCB must contain at least WI.

Area Address Parameter

This parameter specifies the main storage area containing

the block. 'S' may be coded if 'S' was coded in the area address of the associated READ macro, in which case the area address in the READ DECB must be moved to the area address of the WRITE DECB. Or, the DECB used in the READ macro can be used as the DECB for the WRITE macro by using the execute form of the READ and/or WRITE macro. 'S' will release the dynamic buffer.

Length Parameter

This parameter specifies the number of data bytes to be written. If 'S' is coded, the number will be taken from the BLKSIZE parameter in the DCB.

Key Address Parameter

This parameter should specify zero. It is completely ignored by the access method.

Block Address Parameter

This parameter specifies the address of the main storage area containing the left-most byte of the relative block, relative track, or actual address of the block to be written. The type of address is determined by the OPTCD parameter in the DCB.

To Make Additions

Additions per se are not possible, rather additions are simply an update of a user-provided dummy record. It is important that the user recognize the the DA type WRITE macro applies only to data sets with keys which contain system dummy records.

DCB Considerations

1) DSORG must specify DA or DAU. If relative addressing is used, DA should be specified. If actual addressing is used, DAU should be specified.
2) MACRF must be specified. All parameter values are valid except a K and A. If S is specified, the BUFL parameter must also be coded.
3) OPTCD should be specified unless the OPTCD field in the DSCB is satisfactory, i.e., beware the the OPTCD field in the data set's DSCB is merged into the user's DCB at OPEN time if the OPTCD parameter is omitted in the DCB macro.
4) SYNAD is required if C is specified in the MACRF parameter.
5) BUFL is required if MACRF specifies S (dynamic buffering). The length specified must be equal to BLKSIZE value.

6) BUFNO should be considered if you have requested dynamic buffering. By default you get two buffers. The number requested should be equal to the number of records you wish to have in core storage at the same time.
7) BLKSIZE and RECFM should be omitted as they are available in the data set's DSCB and will be merged into the DCB at OPEN time.

Examples

Following are a series of examples illustrating the use of different combinations of READ/WRITE macro instruction parameters. Relative track addressing is used. All the examples use a Format F data set without keys (RECFM=F, BLKSIZE=80,KEYLEN=0) and are based on the following definitions of symbols within the same problem program:

```
//BDAM   DD  DSNAME=BDAMFWOK,DISP=OLD

DCB      DCB    DSORG=DA,MACRF=(RIS,WI),DDNAME=BDAM,     C
                BUFL=80,OPTCD=W
AREA     DS     80C
RELTRK   DS     1H      CONTAINS REL TRK NUMBER
         DS     1C      CONTAINS ACTUAL REC NO. WITHIN TRK
```

Note: Relative track addressing is specified by not coding an A or R in OPTCD. No other options were desired; however, the OPTCD parameter could not have been left blank because at OPEN time it would have been filled in from the DSCB and it may not have contained the proper bit setting to give me relative track addressing.

Example 1: READ DECB,DI,DCB,AREA,'S',0,RELTRK

The data portion of the record, pointed to by the RELTRK parameter, is read into core storage starting at location AREA. The length of the block to be read is obtained from DCBBLKSI.

Example 2: READ DECB,DI,DCB,AREA,'S',KEY,RELTRK

The results are the same as example 1 -- the contents of the key address parameter is ignored.

Example 3: READ DECB,DI,DCB,'S','S',0,RELTRK

A dynamic buffer is obtained and the data portion of the record, pointed to by the RELTRK parameter, is placed into it. The core storage address of the block is placed in DECB+12.

Example 4: WRITE DECB,DI,DCB,AREA,'S',0,RELTRK

The contents of AREA is written onto the data portion of the record pointed to by the RELTRK parameter. The length of the block to be written is taken from DCBBLKSI.

Example 5: WRITE DECB,DI,DCB,'S','S',0,RELTRK

The contents of core pointed to by DECB+12 is written onto the DASD track record pointed to by the RELTRK parameter. The dynamic buffer is also freed.

PROCESSING FORMAT F WITH KEYS

A "Format F with Keys" data set is a direct data set that is recorded on the DASD in the Count-Key-Data format. When it was created, the following was specified in the DCB: RECFM=F or FT, KEYLEN=key field length, and BLKSIZE= data field length.

Figure 10 illustrates the structure of a block on a track. CCHHR gives the physical position of the block on the device, key field length contains KEYLEN, and data field length contains BLKSIZE.

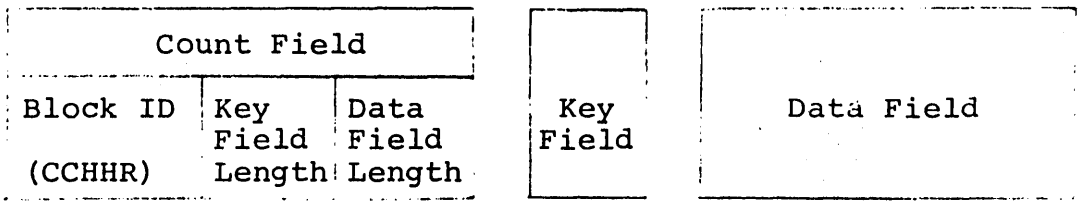

Figure 10. Structure of a Count-Key-Data Block

Two types of I/O operations can be requested: DI or DK. The DI type READ/WRITE macros must be used to retrieve and/or store a specific block located in an address specified by the block address parameter. The access method searches the Block ID portion of the Count Field on the track using a "record identification equal" CCW command code.

The DK type READ/WRITE macros must be used to retrieve and/or store a block with a key as specified by the key address parameter. The block address parameter must still specify the address of where the search is to start. The access method searches the Key Field on the track using a "key equal" CCW command code.

To Retrieve a Specified Block

Type Parameter

The DI type READ macro must be used to retrieve a specific block located in a specified address. The block retrieved is pointed to by the block address parameter. Retrieval of the associated key from the track is optional. MACRF in the DCB must contain at least RI. Extended search, if requested, is ignored.

Area Address Parameter

This parameter specifies the main storage address into which the block of data is to be placed. Dynamic buffering is requested by coding 'S' instead of an address; in this case, the address of the main storage area acquired by the system is placed in the DECB specified by the decb name parameter. If 'S' is specified, S must also be specified in the MACRF parameter of the DCB.

Length Parameter

This parameter specifies the number of data bytes to be read. If 'S' is coded, the number will be taken from the BLKSIZE parameter in the DCB. If a value is specified and it differs from the size of the block read, the "record length check" exception code bit is set on; however, the block is read into core storage. If the value specified is less than block size, only the amount of data specified is read into core storage.

Key Address Parameter

If this parameter specifies an address, the key of the specified block is placed into this main storage address. To suppress the retrieval of the key, code a zero in this parameter. If 'S' was specified in the corresponding area address parameter, then 'S' may be specified in this parameter in which case the key and data are read sequentially into the dynamic buffer; the address of the key will be placed in the DECB at the completion of the read operation.

Block Address Parameter

This parameter specifies the main storage address containing the left-most byte of the relative block, relative track, or actual address of the block to be retrieved. The type of address is determined by the OPTCD parameter in the DCB. The device address of the block will be placed in this area by the control program if feedback is requested, i.e., you have specified a DIF type READ macro.

To Retrieve a Block with a Specified Key

Type Parameter

The DK type READ macro is used to retrieve a block with a specified key. The search for the block starts at the beginning of the track pointed to by the block address parameter and uses the key provided at the key address parameter as a search argument. The number of tracks searched is determined by the LIMCT parameter in the DCB; however, one full track is always searched regardless of whether or not the extended search option is specified. Only the data is retrieved. The MACRF parameter in the DCB must contain at least RK.

Area Address Parameter

This parameter specifies the address of the area into which the block is to be placed. Dynamic buffering is requested by coding 'S' instead of an address; in this case, the address of the main storage area acquired by the system is placed in the DECB specified by the decb name parameter. If 'S' is specified, S must also be specified in the MACRF parameter of the DCB.

Length Parameter

This parameter specifies the number of data bytes to be read. If 'S' is coded, the number will be taken from the BLKSIZE parameter of the DCB. If a value is specified and it differs from the size of the block read, the "record length check" exception code bit is set on.

Key Address Parameter

This parameter must specify the main storage address containing the key of the desired block.

Block Address Parameter

This parameter specifies the address of the main storage area containing the left-most byte of the relative block, relative track, or actual address of where the search for the block to be retrieved starts. The type of address is determined by the OPTCD parameter in the DCB. If it is a relative block address, the system converts it to a relative track address. Thus, when relative addressing is specified, the search always starts at the beginning of the indicated track. The device address of the block will be placed in this area by the control program if feedback is requested, i.e., you have specified a DKF type READ macro.

To Update a Specified Block

Type Parameter

The DI type WRITE macro must be used to update a specified block. Updating of the associated key is optional. MACRF in the DCB must contain at least WI.

Area Address Parameter

This parameter specifies the address of the main storage area containing the block. 'S' may be coded if 'S' was coded in the area address of the associated READ macro, in which case the area address in the READ DECB (DECB+12) must be moved to the area address of the WRITE DECB. Or, the DECB used in the READ macro can be used as the DECB for the WRITE macro by using the execute form of the WRITE macro. 'S' will release the dynamic buffer.

Length Parameter

This parameter specifies the number of data bytes to be written. If 'S' is coded, the number will be taken from the BLKSIZE parameter in the DCB.

Key Address Parameter

This parameter specifies the address of the main storage area containing the key of the block to be written. To suppress the writing of the key, specify zero. 'S' may be coded instead of an address only if the block is contained in an area processed by dynamic buffering, i.e., 'S' was coded in the key address of the associated READ macro.

Block Address Parameter

This parameter specifies the address of the main storage area containing the left-most byte of the relative block, relative track, or actual address of the block to be written. The type of address is determined by the OPTCD parameter in the DCB.

To Update a Block with a Specified Key

Type Parameter

The DK type WRITE macro must be used to write a block using the key provided at the key address parameter as a search argument. The search for the block starts at the address provided at the block address parameter. Only the data is written. The number of tracks searched is determined by the LIMCT parameter in the DCB. MACRF in the DCB must contain at least WK.

Area Address Parameter

This parameter specifies the address of the main storage area containing the block. 'S' may be coded if 'S' was coded in the area address of the associated READ macro, in which case the area address in the READ DECB must be moved to the area address of the WRITE DECB. Or, the DECB used in the READ macro can be used as the DECB for the WRITE macro by using the execute form for the WRITE macro. 'S' will release the dynamic buffer.

Length Parameter

This parameter specifies the number of data bytes to be written. If 'S' is coded, the number of bytes will be obtained from the BLKSIZE parameter in the DCB.

Key Address Parameter

This parameter must specify the main storage address containing the key of the block to be written. 'S' may be coded instead of an address only if the block was read specifying dynamic buffering, i.e., 'S' was coded in the key address parameter of the associated READ macro.

Block Address Parameter

This parameter specifies the address of the main storage area containing the left-most byte of the relative block, relative track, or actual address of where the search for the block to be written starts. The type of address is determined by the OPTCD parameter in the DCB.

To Make Additions

Type Parameter

The DA type WRITE macro is used to "add" a new block wherever there is space. The space the system is looking for is a system dummy record. Thus, it is important to note that additions are really an update to a system dummy record. The search always starts at the beginning of the track pointed to by the block address parameter. The number of tracks searched is determined by the LIMCT parameter in the DCB. One full track is always searched regardless of whether or not the extended search feature (OPTCD=E) is specified. MACRF in the DCB must contain at least WAK.

The user should be aware that the system will add a record which contains the same key as an existing record. To make sure that this doesn't happen, the user should first issue a DK type READ macro and verify that the "record not found" bit is set on in the exception code field of the DECB.

Area Address Parameter

This parameter must contain the address of the main storage area containing the block to be written.

Length Parameter

This parameter specifies the number of data bytes to be written. If 'S' is coded, the number will be obtained from the BLKSIZE parameter in the DCB.

Key Address Parameter

This parameter must contain the address of the main storage area containing the key of the block to be written. The contents of this main storage area will be written in the key portion of the data record. The first byte of the key cannot contain hexadecimal 'FF'.

Block Address Parameter

This parameter must specify the address of the main storage area containing the left-most byte of the relative block, relative track, or actual device address of where the search for a system dummy record starts. The type of address is determined by the OPTCD parameter in the DCB. If feedback is requested, the device address of the block written will be placed in this main storage area by the control program.

DCB Considerations

1) DSORG must specify DA or DAU.

2) MACRF must be specified.

3) OPTCD should be specified unless the OPTCD field in the DSCB is satisfactory, i.e., beware that the OPTCD field in the data set's DSCB is merged into the user's DCB at OPEN time if the OPTCD parameter is omitted in the DCB macro.

4) LIMCT is required if E is specified in the OPTCD parameter.

5) SYNAD is required if C is specified in the MACRF parameter.

6) BUFL is required if MACRF specifies S (dynamic buffering). If 'S' is specified for the key address parameter of a READ or WRITE macro instruction, the buffer length must include the key length.

7) BUFNO should be considered if you have requested dynamic buffering. By default you get two buffers. The number requested should be equal to the number of records you wish to have in core storage at the same time.

8) The following DCB parameters should be omitted as they are available in the data set's DSCB: BLKSIZE, KEYLEN, and RECFM.

Examples

Following are a series of examples illustrating the use of different combinations of READ macro instruction parameters. All the examples use a Format F data set with keys (BLKSIZE=80,KEYLEN=5,RECFM=F) and are based on the following definitions of symbols within the same problem program:

```
//BDAM   DD   DSNAME=TEST.BDAM,DISP=OLD

KEY      DS    CL5
RELBLK   DS    1F
DCB      DCB   DSORG=DA,MACRF=RKISC,DDNAME=BDAM,      C
               SYNAD=ERROR,OPTCD=RE,LIMCT=10,BUFL=85
AREA     DS    80C
```

Example 1: READ DECB,DI,DCB,AREA,'S',0,RELBLK+1

Only the data portion of the data record was read into core storage starting at location AREA. The length of the block is obtained from DCBBLKSI.

Example 2: READ DECB,DI,DCB,AREA,'S',KEY,RELBLK+1

The key portion of the data record is placed into core storage starting at location KEY, and the data portion of the data record is placed in core starting at location AREA. The length of the key and block is obtained from DCBKEYLE and DCBBLKSI respectively.

Example 3: READ DECD,DI,DCB,AREA,'S','S',RELBLK+1

An invalid key address parameter is specified -- the READ will terminate with a "PROT CHECK" error. No data is read into core storage.

Example 4: READ DECB,DI,DCB,'S','S',0,RELBLK+1

A dynamic buffer is obtained and the data portion of the data record is placed into it. The core storage address of the block (data portion) is placed in DECB+12.

Example 5: READ DECB,DI,DCB,'S','S','S',RELBLK+1

A dynamic buffer, large enough for the key and data portion of the data record, is obtained and the key and data from the DASD track is placed into it. The core address of the block is placed in DECB+12 and the core address of the key is placed in DECB+20.

Example 6: READ DECB,DI,DCB,'S','S',KEY,RELBLK+1

A dynamic buffer is obtained and the data portion of the data record is placed into it. The core storage address of the block is placed in DECB+12. The key portion of the data record is placed into core storage at location KEY.

Example 7: READ DECB,DI,DCB,AREA,40,KEY,RELBLK+1

The key and 40 characters of the data portion of the data record was read into core storage starting at the specified locations. However, the "record length check" exception code bit was set on because the length specified (40) was not equal to the data record length.

Example 8: READ DECB,DK,DCB,AREA,'S',KEY,RELBLK+1

The key provided at location KEY is used as a search argument to search the key portion of the DASD tracks. The search starts on the track specified by RELBLK+1. (Note: the access method converts the relative block number into a relative track number). The data portion of the data record was read into core storage starting at location AREA. The contents of KEY is unchanged.

Example 9: READ DECB,DK,DCB,'S','S',KEY,RELBLK+1

Same as Example 8 only a dynamic buffer is obtained and the data is read into it. The core address of the block is placed in DECB+12.