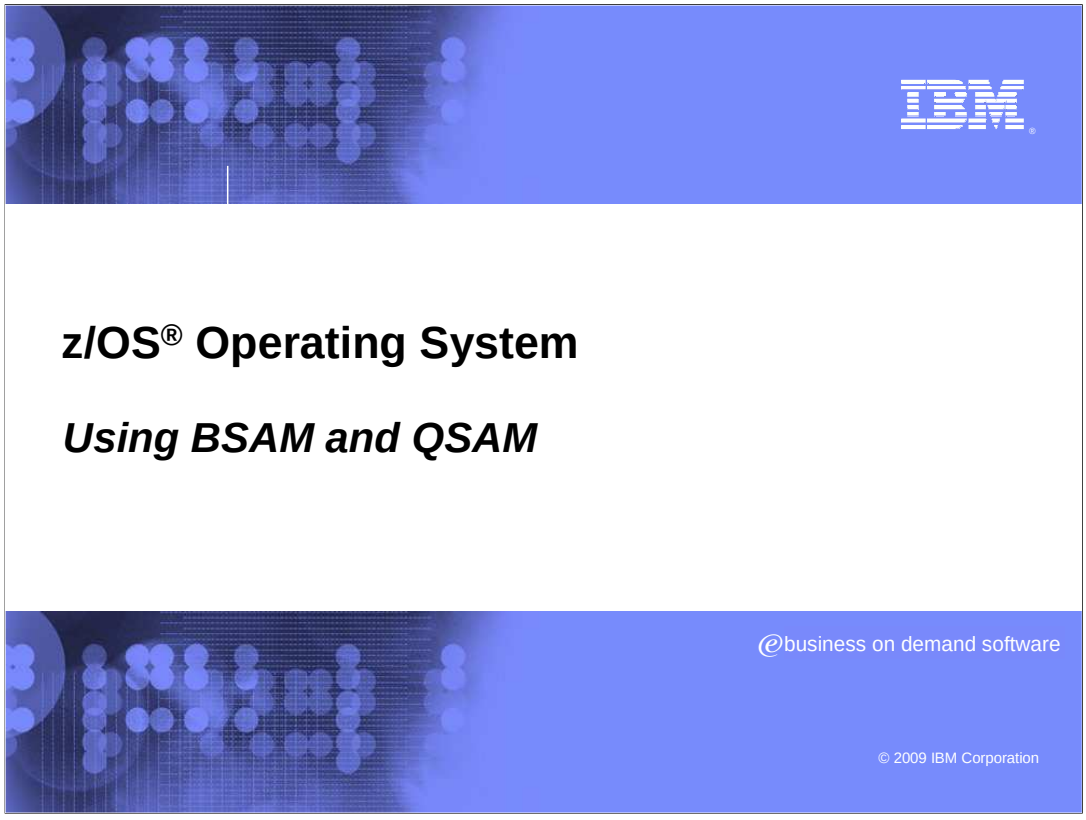

IBM
z/OS® Operating System
Using BSAM and QSAM
ebusiness on demand software
© 2009 IBM Corporation

Basic access method

- What is an access method?

•An access method defines the technique and API by which the data is stored and retrieved.

•Each generally has its own data set structures (data set organization) to organize data.

- An *access method* defines the technique that is used to store and retrieve data.

- Access methods have their own data set structures to organize data, macros to define data sets, and utility programs to process data sets.

Summary of DFSMS™ access methods

Data set organization	Basic	Queued	VSAM
Sequential	BSAM	QSAM	
Partitioned	BPAM, BSAM	QSAM	VSAM
UNIX® file	BSAM	QSAM	VSAM
ESDS			VSAM
RRDS			VSAM
KSDS			VSAM
LDS			VSAM, DIV
Direct	BDAM		

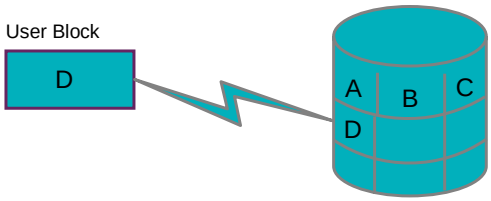
- Access methods are identified primarily by the data set organization.

- For example, use the basic sequential access method (BSAM) or queued sequential access method (QSAM) with sequential data sets.

- However, there are times when an access method identified with one organization can be used to process a data set organized in a different manner. For example, a sequential data set created using BSAM can be processed by the basic direct access method (BDAM) and vice versa.

Basic sequential access method (BSAM)

- Arranges records sequentially in the order in which they are entered to form sequential data sets
- The user organizes records with other records into blocks: *basic access*
- Used to access: sequential data set, partitioned member and UNIX files (such as z/FS)



-BSAM arranges records sequentially in the order in which they are entered.

-A data set that has this organization is a sequential data set.

-The user organizes records with other records into blocks. This is basic access.

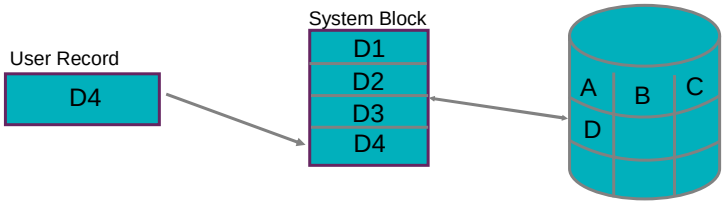
-The application program must block and unblock its own input and output records. BSAM only reads and writes data blocks.

-The application program must manage its own input and output buffers. It must give BSAM a buffer address with the READ macro, and it must fill its own output buffer before issuing the WRITE macro.

-You can use BSAM with the following data types: sequential data set, partitioned member and UNIX files (such as z/FS)

Queued sequential access method (QSAM)

- Arranges records sequentially in the order they are entered to form sequential data sets
- The system organizes records with other records into blocks
- To improve performance, QSAM reads these records into storage before they are requested and writes behind: queued access
- Used to access: sequential data set, partitioned member and UNIX files



•QSAM arranges records sequentially in the order that they are entered to form sequential data sets, which are the same as those data sets that BSAM creates.

•The system organizes records with other records.

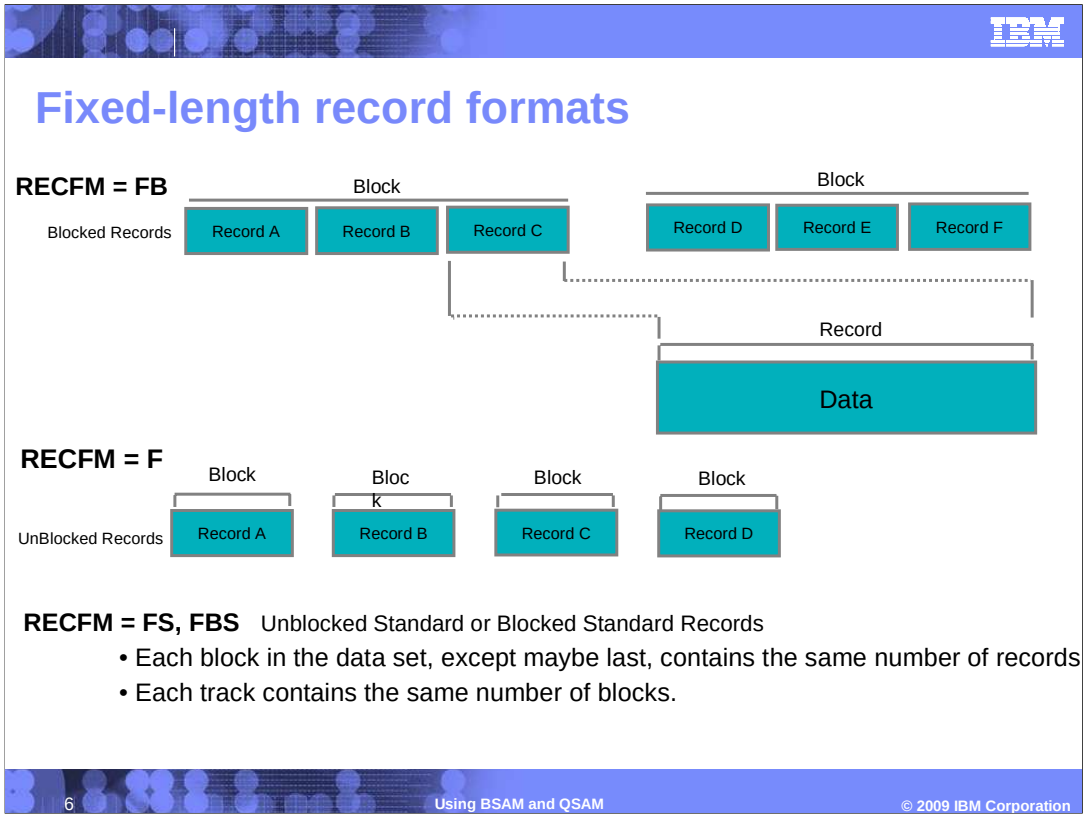
•QSAM blocks and unblocks records for you automatically.

•QSAM manages all aspects of I/O buffering for you automatically. The GET macro retrieves the next sequential logical record from the input buffer. The PUT macro places the next sequential logical record in the output buffer.

•QSAM anticipates the need for records based on their order.

•To improve performance, QSAM reads these records into storage before they are requested. This is called queued access.

•Can use QSAM with the following data types: sequential data set, partitioned member and UNIX files



•The size of fixed-length (format-F or -FB) records is constant for all records in the data set.

•The records can be blocked or unblocked.

•If the data set contains unblocked format-F (format-F) records, one record constitutes one block.

•If the data set contains blocked format-F (format-FB) records, each track contains the same number of blocks, except maybe the last.

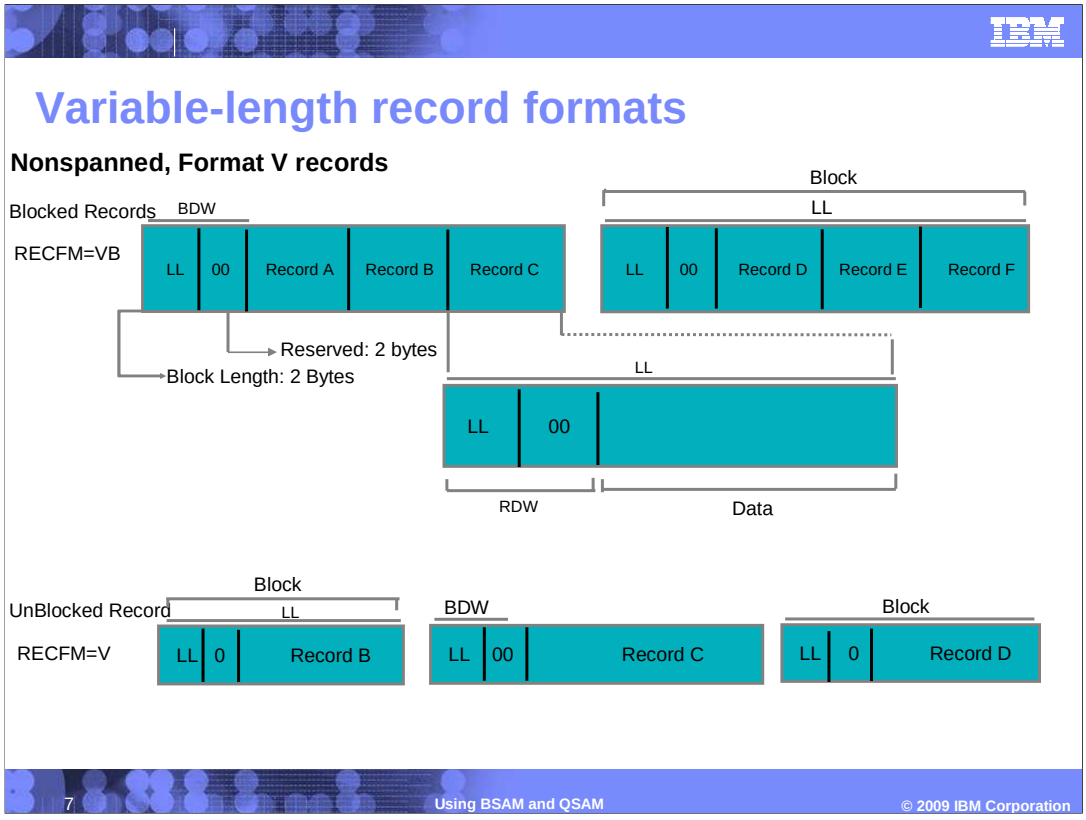
•STANDARD FORMAT (FS, FBS)

•A standard-format data set must conform to these specifications:

- •Only the last block can be truncated.
- •Each block in the data set, except maybe last, contains the same number of records.
- •Each track contains the same number of blocks.

•A sequential data set with standard format records (format-FS or -FBS) sometimes can be read more efficiently than a data set with format-F or -FB records.

•This is because each track contains same number of blocks, and each block except the last contains the same number of records, the system is able to determine the address of each record to be read.



•In a variable-length record data set, each record or record segment can have a different length.

•This figure shows blocked and unblocked variable-length records

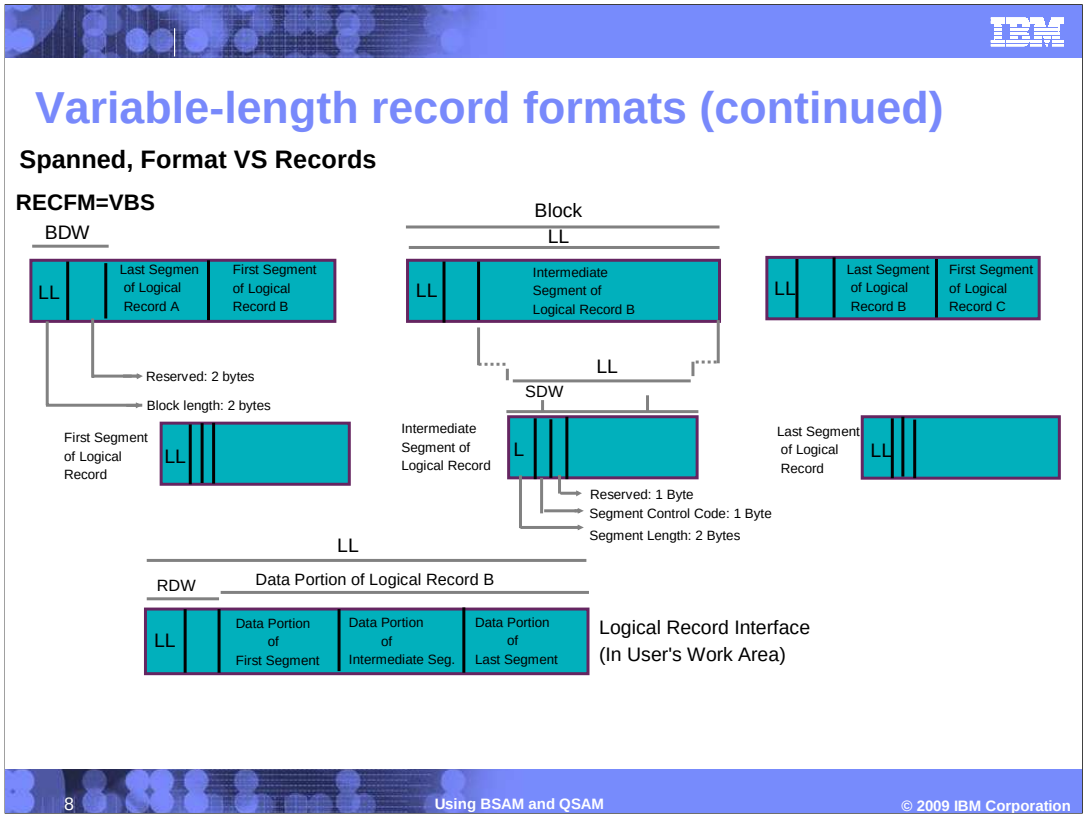
•A block in a data set containing unblocked records is in the same format as a block in a data set containing blocked records. The only difference is that with blocked records each block can contain multiple records.

•Block Descriptor Word (BDW)

- •A variable-length block consists of a block descriptor word (BDW) followed by one or more logical records or record segments.
- •The block descriptor word is a 4-byte field that describes the block.
  - •It specifies the 4 byte block length for the BDW plus the total length of all records or segments within the block.

•Record Descriptor Word (RDW)

- •A variable-length logical record consists of a record descriptor word (RDW) followed by the data.
- •The record descriptor word is a 4 byte field describing the record.
  - •The first 2 bytes contain the length (LL) of the logical record (including the 4 byte RDW).
  - •All bits of the third and fourth bytes must be 0 (because other values are used for spanned records, and this is non-spanned records)



This figure shows how the spanning feature of the BSAM and QSAM lets you create and process variable-length logical records that are larger than one physical block.

•It allows you to split the records into segments so that they can be written into more than one block.

•A block is using the same BDW, but for each record segment, it consists of a segment descriptor word (SDW) followed by the data.

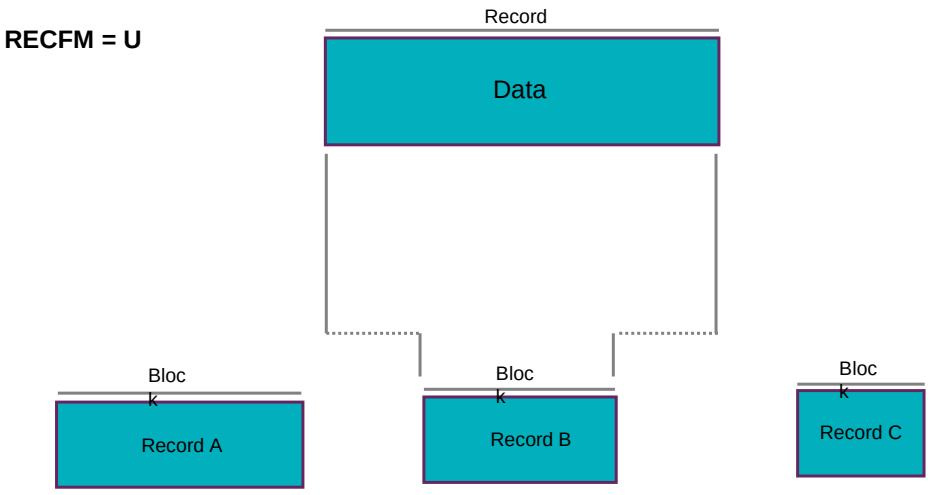
•SDW is similar to RDW

•SDW is a 4 byte field that describes the segment. The first 2 bytes contain the length (LL) of the segment, including the 4 byte SDW.

•The third byte of the SDW contains the segment control code that specifies the relative position of the segment in the logical record.

•The remaining bits of the third byte and all of the fourth byte are reserved for possible future system use and must be 0.

Undefined-length record format



- Format-U permits processing of records that do not conform to the F- or V- format.
- Each block is treated as a record.
- For format-U records, you must specify the record length when issuing the WRITE, PUT, or PUTX macro.
- No error indication will be given if the specified length does not match the buffer size or the physical record size.

Processing a data set with SAM

To access a data set using BSAM or QSAM:

- Connect to the data set by issuing an OPEN macro specifying a DCB which describes the data set.

```
OPEN  mydcb@,[INPUT|OUTPUT]
```

- Add or retrieve data with BSAM/QSAM macros.
- When processing has been completed, issue a CLOSE macro to disconnect the data set from the processing program and free up resources no longer needed.

```
CLOSE mydcb@
```

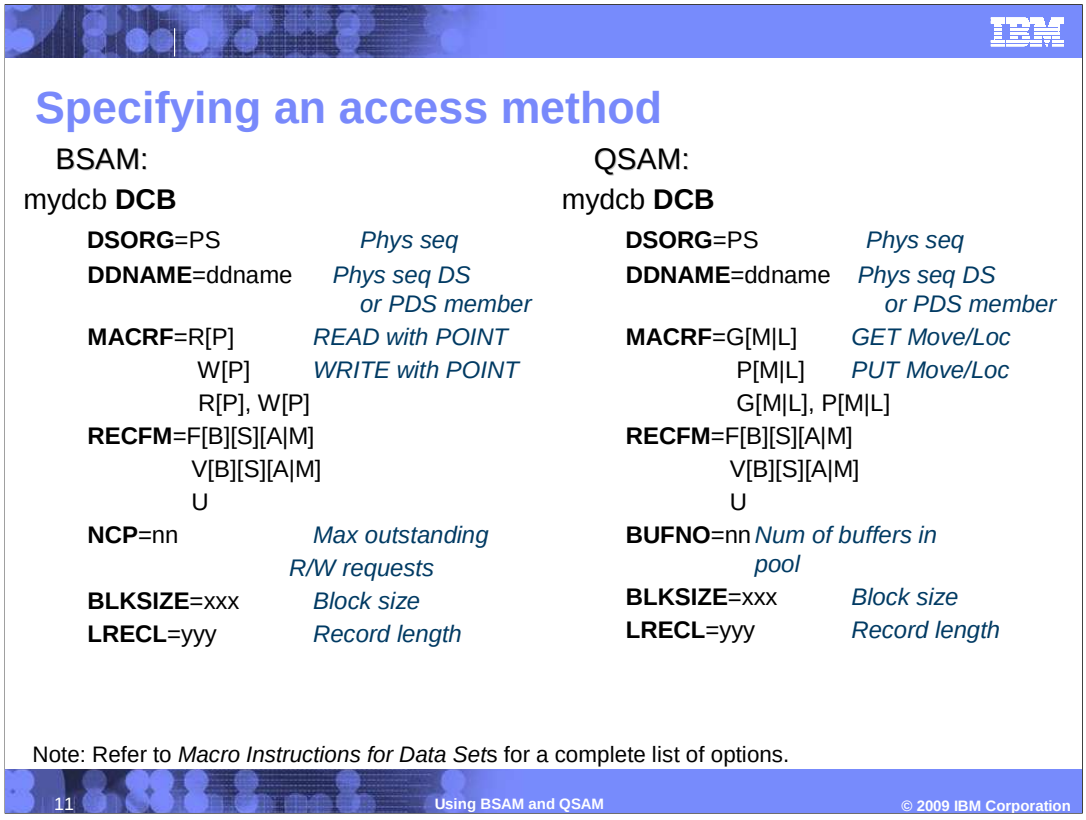

1.A DCB (data control block), which contains the characteristics of the data set, is needed to identify the data set to be opened.

•*dcb address*: Specifies the address of the data control blocks for the data sets to be prepared for processing.

2. Can issue a series of GET or PUT (for QSAM) or READ or WRITE (for BSAM) macros to add or retrieve data.

3. *dcb address*: specifies the address of the data control block for the opened data set to be closed.

•The fields of the data control block (DCB) and DCBE are restored to the condition that existed before the OPEN macro was issued, and the data set is disconnected from the processing program.



Definition of parameters:

DSORG – Data set organization (like physical sequential)

DDNAME – Name that defines the data set being allocated or processed.

MACRF – Specifies type of macros (READ, WRITE, CNTRL, NOTE/POINT for BSAM) or (GET, PUT, PUTX for QSAM) that are used with the data set being created or processed.

RECFM (record format and characteristics of the data set being allocated or processed)

NCP (max number of READ and WRITE macros that can be issued before the first CHECK macro is issued)

BUFNO (buffers to be used for this dataset)

BLKSIZE (maximum block length in bytes)

LRECL (length, in bytes, for fixed-length record. OR max length, in bytes, for variable-length records)

Specifying an access method (continued)

- **The DCB may also reference a DCBE**

```
DCB     DCBE=mydcbe
```

- **The DCBE (DCB extension) expands the functions provided by the DCB.**

```
mydcbe  DCBE
   RMODE31={BUFF|NONE}
  EODAD=myEodad
   SYNAD=mySynad
   GETSIZE={Y|N}
   PASTEOD={Y|N}
   NOVER={Y|N}
   MULTACC=n
   MULTSDN=n
```


Definition of parameters:

•**RMODE31** : specifies whether you request that OPEN get QSAM buffers above the 16 MB line (RMODE31=BUFF) or not (RMODE31=NONE) when acquiring buffers automatically. The default is NONE.

•**EODAD:** specifies the address of an end-of-data routine given control when the end of an input data set is reached.

•**SYNAD:** specifies the address of an error analysis (SYNAD) routine given control when an uncorrectable input/output error occurs.

•**GETSIZE:** specifies that OPEN is to calculate the number of blocks in the data set and store this number in the DCBE (DCBESIZE).

•**PASTEOD:** specifies that the end-of-data marker of the extended format data set, which is saved when the data set is open for INPUT, UPDATE, OUTIN, or INOUT is to be ignored. The default is NO.

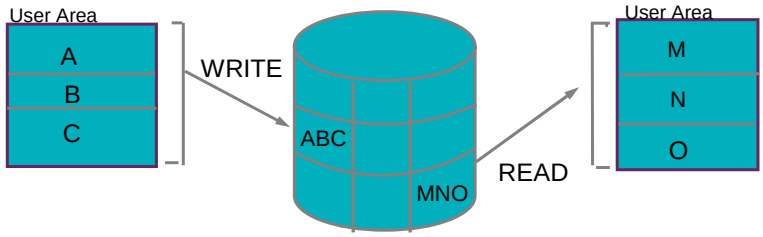
•**NOVER:** specifies that OPEN should bypass any verification to determine whether the size of the stripes of an extended format data set are consistent. The default is NO.

•**MULTACC:** allows the system to process BSAM I/O requests more efficiently by not starting I/O until a number of buffers have been presented to BSAM.

•**MULTSDN:** Requests a system-defaulted NCP. Used to set DCBNCP depending on the data set.

BSAM macros

- READ: Read a Block
  - **The READ macro retrieves a block from a data set and places it into a designated area of storage provided by the user.**



- WRITE: Write a Block
  - **The WRITE macro adds or replaces a block in a data set being created or updated. The data to be written is taken from a designated area of storage provided by the user.**

- The READ macro retrieves a block from a data set and places it into a designated area of storage provided by the user.

- The WRITE macro adds or replaces a block in a data set being created or updated. The data to be written is taken from a designated area of storage provided by the user.

- READ/WRITE: The input/output operation must be tested for completion using a CHECK macro.

BSAM macros (continued)

- CHECK: Wait for Completion of request
- Issued following a READ or WRITE
  - Places the active task in the wait condition
  - Tests input or output operation for errors and exceptional conditions.
  - If the operation completes successfully, control is returned to the instruction following the CHECK macro.
  - If the operation fails, the error analysis routine (SYNAD) or end-of-data routine (EODAD) is given control.

- Issued following a READ or WRITE macro, the CHECK macro places the active task in the wait condition, if necessary, until the associated input or output operation is completed.

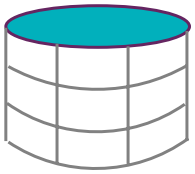
- The input or output operation is then tested for errors and exceptional conditions.

- If the operation completes successfully, control is returned to the instruction following the CHECK macro. Otherwise, the error analysis routine (SYNAD) or end-of-data routine (EODAD) is given control.

- If CHECK fails and if the appropriate routine is not provided, the task is abnormally terminated.

BSAM macros (continued)

Repositioning macros



NOTE: Provide Relative Position

- The NOTE macro returns the position of the last block read from or written into a data set.

- POINT: Position for Access
  - The POINT macro causes repositioning such that the next READ or WRITE operation is for the requested block (denoted by the token passed on input to the request) on the current volume.
- BSP: Backspace
  - The BSP macro repositions to the previous block on the current volume.

Before issuing these three macros, all input and output operations using the same data control block must be tested for completion.

- NOTE: The NOTE macro returns the position of the last block read from or written into a data set. Before issuing a NOTE macro, all input and output operations using the same data control block must be tested for completion.

- POINT: The POINT macro causes the next READ or WRITE operation to be for the specified data set block on the current volume. Before issuing the POINT macro, all input and output operations using the same data control block must be tested for completion.

- BSP: The BSP macro repositions to the previous block on the current volume. All input and output operations must be tested for completion before the BSP macro is issued.

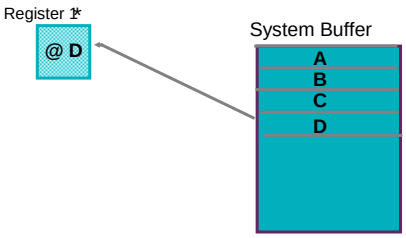
QSAM macros

GET: Obtain next logical record

- The GET macro retrieves (reads) the next record from a data set.

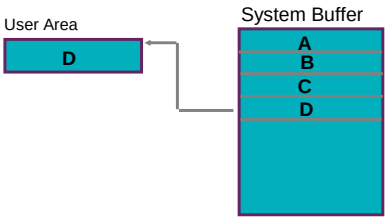
Available modes:

Locate Mode: Omit area address



* Returns address of the location containing the next record

Move Mode: Requires area address



Returns copy of record

-The GET macro retrieves (reads) the next logical record.

-Various modes are available: Locate mode and Move mode.

- Locate mode: In locate mode, the GET macro locates the next sequential record or record segment to be processed. The system returns the address of the record or segment in register 1. You can process the record in the input buffer or move the record to a work area.

- Move mode: In move mode, the GET macro moves the next sequential record to a work area. The system returns the address of the work area in register 1.

QSAM macros (continued)

PUT: Write next record

The PUT macro writes a record in a data set.

Available modes:

Locate Mode: Omit the area address

Move Mode: Requires area address

* Address of the location into which your program later places the next record

-The PUT macro writes a record in a data set.

- Various modes are available: Locate mode and Move mode.

- In the locate mode, the PUT macro returns the address of an area in an output buffer in register 1. This address is where the next sequential record or record segment should be constructed.

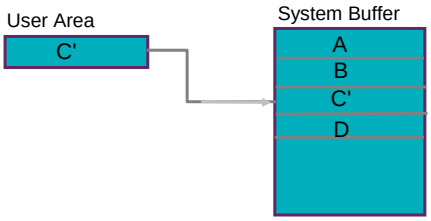
- In the move mode, the PUT macro moves a logical record into an output buffer.

QSAM macros (continued)

PUTX: Update a record in an existing data set

The PUTX macro returns an updated record to a data set.

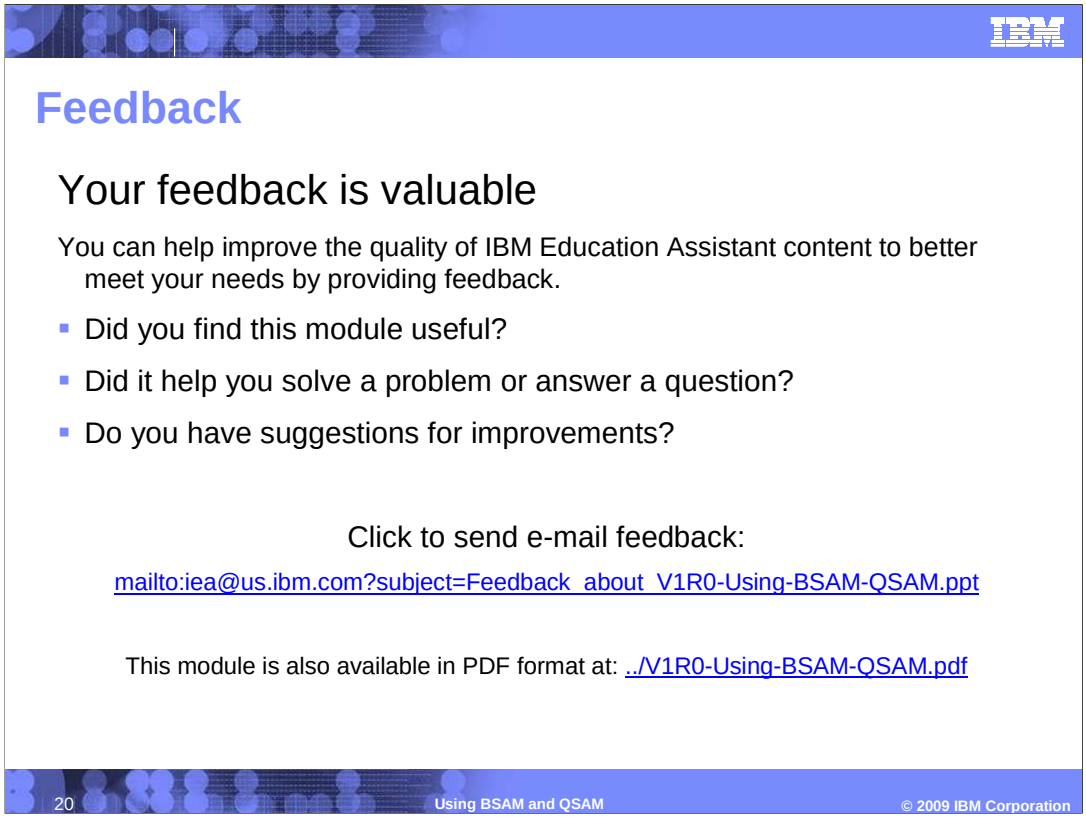
- Record must have been previously obtained by Get with locate or move mode



- PUTX macro returns an updated record to the data set from which it was read. Record must have been previously obtained by GET macro with locate or move mode.

References

- Using Data Sets, SC26-7410
- Macro Instructions for Data Sets, SC26-7408
- SAM Logic, ZW84-4030-03 (IBM confidential)



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers