Table of Contents

Notices

IBM, z/OS, and zPDT are registered trademarks of International Business Machines Corporation. MVS is a trademark of International Business Machines Corporation.

Introduction

This document described various structures and conventions related to use of the Fixed Block Architecture (FBA). The Fixed Block Architecture is used by a number of direct access storage devices (DASD) and by the Virtual Storage Access Method (VSAM) with both FBA supporting devices and Count-Key-Data (CKD) supporting devices. The use by VSAM of the FBA is outside of the scope of this document, focusing strictly on use of the FBA with FBA designed DASD.

A FBA device stores data in 512-byte physical sectors. Each sector is identified by an unsigned number. The first sector of the volume is sector 0. The second is sector 1, etc. The last sector of a device supporting n sectors is sector n-1. The CPU can only access in a single operation one or more set of sequentially numbered sectors.

A single sector or a group of sequential sectors are used for storage of application data records. The single sector or group utilize, through software, a logical structure called a control interval. Accessing a control interval depends upon the underlying FBA device physical access to sectors and utilizes its supported channel commands to do so. The control interval is the bridge between the physical aspect of FBA DASD and the logical content placed on the volume.

This document explores

- hardware imposed standards,
- control interval design (the physical-to-logical bridge),
- management of space on FBA devices,
- individual data set considerations, and
- using partitioned organization on an FBA device, and
- channel command usage.

Space management and data sets both depend upon the control interval. For this reason, the control interval is discussed previous to both of these topics. The various channel commands are designed for support of space management and physical access to control intervals and sectors. For this reason, channel commands are discussed last.

Systems Supporting FBA DASD

VSAM from its inception used the control interval as its standard for data storage on disk regardless of the underlying DASD hardware. For Count-Key-Data (CKD) DASD, VSAM would place a control interval within a single CKD physical disk record. If the operating system also supported FBA DASD hardware, the same control interval would be placed within a group of FBA sectors.

Only two IBM® operating systems ever implemented support for FBA DASD devices as described in this document. The support was made available with these two products:

- IBM Virtual Machine/System Product (VM/SP) including CMS, and
- IBM Disk Operating System/Virtual Storage Extended (DOS/VSE).

Recent support by z/OS® for fiber channel connected FBA disks use entirely different mechanisms for access than are described here. The MVS™ family of products never provided support for FBA DASD. Only CKD DASD have been supported by the MVS family.

FBA Hardware Availability

At the time this document was written (2017), FBA DASD hardware described in this document is essentially non-existent. It only survives in emulation form in three environments:

- As a virtual machine exposed device under a version of IBM VM providing FBA minidisks,
- As a hardware device exposed to an operating system or other program under Hercules emulation, or
- As a hardware device exposed to an operating system or other program by IBM zPDT® product.

The latter two cases may be used directly with an operating system that actually provides FBA support, for example, those identified in the preceding section, provided other compatibilities with the emulator exist.

Numerous conditions, particularly error related, that may arise with physical hardware do not arise in emulated environments. As a consequence, many of the error related reporting mechanisms do not apply in these cases. Emulation programs may differ in how closely they conform to real hardware. In particular, if the Hercules emulator is being used, refer to the section “Hercules FBA Device Emulation” for details concerning its FBA emulation.

This document focuses upon the use of FBA devices under emulation. Refer to references, 1, 8 and 9, identified in the following section, for specifics of error situations and responses.

References

The number enclosed in square brackets, [ ], below are used to identify the reference within this document. References listed are in no particular order.

Number	Reference
1	*IBM 3310 Direct Access Storage Reference Manual*, GA26-1660-1, March 1979.
2	*IBM z/OS® DFSMS Using Data Sets Version 2 Release 1*, SC23-6855-00, 2013
3	*IBM z/OS DFSMSdfp Advanced Services Version 2 Release 1*, SC23-6861-01, 2014
4	*IBM VSE/Advanced Functions Handbook Program Number 5746-XE8 Release 2*, LY33-9101-0, June 1980.
5	*IBM Device Support Facilities User's Guide and Reference Release 16 Refresh*, GC35-0033-23, December 1998.
6	*IBM System/360 Operating System: System Control Blocks OS Release 21.7*, GC28-6628-8, April 1973.
7	*DOS/VSE Librarian Logic*, SY33-8557-31, February 1979.
8	*IBM 3370 Direct Access Storage Description*, GA26-1657-1, September 1979.
9	*IBM 3880 Storage Control Models 1, 2, 3, and 4*, GA26-1661-9, September 1987.

FBA Hardware

All FBA DASD devices utilize a fixed sized physical record of 512 bytes on the disk known as a sector. The system accesses the sectors by sector number. To the system, the sectors appear as an uninterrupted linear sequence of sectors. To the physical FBA disk, the sectors are actually written on tracks and accessed by a number of access positions (cylinders) in the same way as CKD records are stored. The number of sectors contained on a single track or accessible to a single access position are attributes of the device that the software can utilize for improved performance if it elects to do so.

Data is transferred between the program and the DASD volume as sector content. The number of sectors participating in the transfer may be one or multiple sequential sectors. Such transferred data is referred to here as a physical data record. Because the content of the sector or sectors containing the physical data record may be transferred by a single FBA DASD command, the physical data record is transferred by a single command.

Records may occur at three levels:

- physical data records,
- logical data records, and
- application data records.

A physical data record may contain one or more logical data records. Correspondingly, a logical data record may contain one or more application data records. Application data records are the unit of data written and read by an application. Application in this context may be a user application or a system application. When more than one application data record is placed within a single logical data record, it can be described as a logical data block and the logical data records are said to be blocked. When a single application data record resides in a single logical data record the application data records are described as unblocked. Additionally, it is possible for a single application data record to reside in multiple logical data records each of which reside in separate physical records, known as a spanned record.

Logical data records of the same size may represent a number of available slots within a physical record. Each slot may contain the content of a logical data record or be available for use, its current content being ignored. Slots allow reuse of a logical data record including the ability to logically “delete” its content.

FBA DASD Model Capacities

FBA DASD storage capacities varied by models:

- 125,664 – 1,672,881 sectors,
- 62,832 – 836,440.5 kilobytes,
- 61.35 – 814.84 megabytes, or
- 6% - 80% or 1 gigabyte.

Capacities for specific models and associated control units are identified in the following table.

Control Unit Type & Model	Device Type and Model	Number of Sectors (decimal)	Number of Sectors (hexadecimal)	Capacity (K)	Capacity (M) (approximate)	Percent of 1 Gigabyte (approximate)
3880-01	3370 3370-1 3370-A1 3370-A2	558,000	X'000883B0'	279,000K	272.46M	27
	3370-2 3370-A2 3370-B3	712,752	X'000AE030'	356,376K	348.02M	34
4331-01	3310 3310-1	125,664	X'0001EAE0'	62,832K	61.35M	6
6310-01	0671	574,560	X'0008C460'	287,280K	280.55M	27
	0671-04	624,456	X'00098748'	312,228K	304.91M	30
	0671-08	513,072	X'0007D430'	256,536K	250.52M	24
	9313	246,240	X'0003C1E0'	123,120K	120.23M	12
	9332 9332-400	360,036	X'00057E64'	180,018K	175.80M	17
	9332-600	554,800	X'00087730'	277,400K	270.90M	26
	9335	804,714	X'000C476A'	402,357K	392.93M	38
	9336 9336-10	920,115	X'000E0A33'	460,057.5K	449.27M	44
	9336-20 9336-25	1,672,881	X'001986B1'	836,440.5K	814.84M	80

Reserved Sectors

Only two sectors are reserved for a specific use. Sector 0 is used for the Initial Program Load function when an FBA DASD device is the target of this operator initiated function. Sector 0 contains a single physical record that contains the set of chained FBA DASD commands and related data that will transfer to the CPU the initial program being loaded.

The program itself may reside in one or more physical data records transferred by their own operations (specified in the content of sector 0) into memory. Frequently the loaded program resides in sectors starting at sector 2 and resides in as many as are needed. There is no requirement for that to be the case. Wherever the program resides, the content of sector 0 must reflect the location.

Sector 1 is reserved for the Volume Label identifying the DASD volume. The volume label is a single physical data record with a standard structure allowing identification of the volume and locating the volume's space management information, know as the Volume Table of Contents (VTOC). The Volume Label is an example of a single application data record residing in a single logical data record in a single physical record, all three records being the contents of sector 1. If the volume does not contain an initial program loaded into memory, the VTOC will typically start at sector 2. When the volume does contain an initial program, the VTOC will reside in the highest numbered sectors of the volume. While these two locations are typical, the VTOC can reside anywhere the user selects other than the first two reserved sectors.

Input/Output Programming for FBA Devices

Programming for FBA devices requires the construction of channel programs that drive a sequence of input/output operations by means of channel command words. This section presumes a familiarity with mainframe channel programming. See references 1 and 8 for details related to each channel command.

Unlike CKD DASD devices, where the I/O programmer has to have intimate knowledge of the way data is recorded on the magnetic media, FBA devices require no such knowledge. All of the “heavy lifting” is provided by the FBA device and its control unit. This greatly simplifies the I/O programmer's task. Additionally the commands are designed for ease of use with standard volume space management conventions.

Five I/O CCW programs are typical of normal operation of a FBA device.

1. Device Discovery channel program
2. Accessing device sectors channel program,

3. Initial Program Load (IPL) program, and
4. Error reporting program.

The third program sequence is really a variation of the second, namely the accessing of device sectors.

Because channel programs consist of a sequence of channel commands issued to the device, the “FBA Channel Commands” section discusses the individual commands, followed by a section that combines the commands into the typical programs, the “FBA Channel Programs” section.

Physical vs. Logical Sector Addressing

As described previously individual sectors are addressed by a number, starting with 0 and consecutively to the number of physical sectors on the disk, less one, or 0 through n-1, where n is the number of sectors on the device. This number is the sector's physical sector number (PSN). Data set placement on a FBA volume will be within one or more groups of sequential physical sectors. Each group is an extent and is identified by the physical sector number of the groups first and last sectors.

The set of extents, taken in order, together create a sequence of sectors that constitute the data set. This sequence of sectors may be addressed with a logical sector number (LSN) relative to the placement of the sector within the data set. Logical sector numbers provide a position independent addressing mechanism for the sectors constituting the data set. Relocation or change in size of the physical extents constituting the data set does not change the logical sector number of the sector within the data set.

An excellent diagram of this process is provided by [1: 3-1, 3-3]. The reference uses the terms “physical block number” and “logical block number” instead of the terms, respectively, “physical sector number” and “logical sector number”. [9:3-5] has an example of logical sector number usage with data set extents on different volumes.

FBA Channel Commands

This table summarizes the possible commands with a brief description. The “Herc” column identifies whether Hercules emulates the command. The “Abrev.” column provides a short form of the command's name. The “Data Length” column indicates the amount of data required in the channel command word. Abbreviated names in **bold** indicate the command has chaining requirements summarized in the next table.

Fixed-Block Architecture Structures

Type	Herc	Command	Abrev.	Name	Data Length	Direction
Control	yes	X'03'	NOP	No-Operation (No-Op)	0	- -
	yes	X'63'	**DXT**	Define Extent	16	CU
	yes	X'43'	**LOC**	Locate	8	CU
Read	yes	X'42'	**READ**	Read	multiples of 512	CPU
	yes	X'02'	**IPL**	Read IPL	512 maximum	CPU
Write	yes	X'41'	**WRITE**	Write	multiples of 512	CU
Sense	yes	X'E4'	SID	Sense Identification (ID)	7	CPU
	yes	X'04'	SNS	Sense	24	CPU
	yes	X'A4'	RRBL	Read and Reset Buffered Log	24	CPU
	yes	X'64'	RDC	Read Device Characteristics	32	CPU
	yes	X'B4'	**RSRV**	Device Reserve	24	CPU
	yes	X]94'	**RLS**	Device Release	24	CPU
	yes	X'13'	**URSRV**	Unconditional Reserve	24	CPU
Diagnostic	no	X'F3'	**DCTL**	Diagnostic Control	4	CU
	no	X'C4'	**DSNS**	Diagnostic Sense	6	CPU
Channel	yes	X'08'	TIC	Transfer-in-channel	0	- -

See [8:6-8] for excellent summary of the function of each command. Use either reference 1 or 7 for command details.

Some channel commands must or must not precede certain other commands. Some must be the first command of a chain. The following table identifies such commands. All others may be first or preceded by another command in the chain.

Command	Be First?	Be Preceded by?
IPL	must	May: IPL
URSRV	must	
RSRV	may	Must not: DXT, IPL
RLS	may	Must not: DXT, IPL
DXT	may	Must not: DXT, IPL
NOP, SID, SNS, RRBL, RDC	may	May: any
TIC	may not	May: any
LOC	may not	Must: DXT, IPL
READ	may not	Must immediately: LOC

Command	Be First?	Be Preceded by?
WRITE	may not	Must immediately: LOC
DCTL	may not	Must: DXT
DSNS	may not	Must: DCTL

The following sections describe the commonly used commands and associated data structures transferred between the CPU and control unit. Refer to references 1, 8, or 9 for details concerning: DCTL, DSNS. Because TIC is not implemented by a control unit, but rather directly by the channel, refer to the relevant Principles of Operation manual for the description of TIC and its constraints.

DEFINE EXTENT – X'63'

The DEFINE EXTENT command transfers 16 bytes of parameters from the channel to the control unit. The parameters define the size and location of a data extent. The data extent area, defined by the parameters transferred to the control unit, establishes limits on the device within which subsequent chained commands are permitted to operate. The parameter list also contains an inhibit mask to determine which types of commands are permitted in the chain.

Chaining Requirements

The DEFINE EXTENT command must not be preceded by another DEFINE EXTENT or READ IPL command in the same chain.

Status

Initial status is normally zero. Channel End and Device End are presented after the parameters have been transferred and checked for validity. Invalid parameters cause the command to be terminated with Channel End, Device End and Unit Check Status. Parameters are invalid if any required values are not present or prohibited values are present.

DEFINE EXTENT Command Parameters

The format of the transferred parameters and related values are provided in the following tables.

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	1	bit string	Command mask. See table below.
+1	+1	1	reserved	Must be X'00'.
+2	+2	2	unsigned	Sector size in bytes. Must be 512.
+4	+4	4	unsigned	Defines the offset, in sectors, from the beginning of the device to the first sector of the extent, the physical sector number of the first sector of the extent being defined.
+8	+8	4	unsigned	Defines the relative displacement, in sectors, from the beginning of the data set to the first sector of the extent, the logical sector number of the first sector of the extent.
+12	+C	4	unsigned	Defines the relative displacement, in sector, from the beginning of the data set to the last sector of the extent,

Offset Dec	Offset Hex	Length	Type	Description
				the logical sector number of the last sector of the extent

Mask Byte Values

Mask / Value	Bits(s)	Description
xx..	0, 1	Inhibited or permitted operations: • 00 – Inhibits format write operations • 01 – Inhibits all write operations • 10 – Must not be used • 11 – Permits all write operations
..00	2, 3	Must be 00.
.... x...	4	Addressed volume area: • 0 – Data area. • 1 – CE area.
.... .x..	5	Diagnostic commands (DIAGNOSTIC CONTROL, DIAGNOSTIC SENSE): • 0 – Inhibited • 1 – Permitted
.... ..00	6,7	Must be 00.

DEVICE RELEASE – X'94'

The DEVICE RELEASE command terminates the reservation of the addressed device from the channel that issued the command if a channel switch feature is installed in the control unit or if a string switch feature is installed in the device. Besides terminating the reservation of the addressed device, the DEVICE RELEASE command transfers up to 24 sense bytes to the channel. A DEVICE RELEASE command will be executed regardless of any abnormal status conditions (such as offline or unsafe).

Chaining Requirements

The DEVICE RELEASE command must not be preceded by a DEFINE EXTENT or READ IPL command in the same chain.

Status

Initial status is normally zero. Channel End and Device End are presented after up to 24 sense bytes have been transferred.

DEVICE RESERVE – X'B4'

The DEVICE RESERVE command reserves the addressed device to the channel that issued the command if a channel switch feature is installed in the control unit or if a string switch feature is installed in the device. Besides reserving the addressed device, the DEVICE RESERVE command transfers up to 24 sense bytes to the channel. A DEVICE RESERVE command will be executed regardless of any abnormal device status conditions (such as offline or unsafe). Device reservation is maintained until the reserving channel successfully completes a DEVICE RELEASE command addressed to the reserved device.

Note: A system reset cancels reservation of a device to the resetting channel only.

Chaining Requirements

The DEVICE RESERVE command must not be preceded by a DEFINE EXTENT or READ IPL command in the same chain.

Status

Initial status is normally zero. Channel End and Device End are presented after up to 24 sense bytes have been transferred.

LOCATE – X'43'

The LOCATE command transfers eight bytes of parameters from the channel to the control unit. The parameters specify the location and amount of the data to be processed. Data transfer between the channel and control unit associated with these operations does not occur during execution of the LOCATE command. Data transfer is initiated by a read or write CCW following the LOCATE command.

Chaining Requirements

The LOCATE command must be preceded by a READ IPL or DEFINE EXTENT command in the same chain or the command is rejected with Channel End, Device End, and Unit Check Status.

Status

Initial status is normally zero. Channel End is presented after the parameters have been transferred and checked for validity. Device End and Unit Check status is presented under the following circumstances:

- prohibited parameter values present or required parameters not present
- sectors identified by the operation fall outside of the extent
- Refer to [9:3-6 to 3-9] for additional conditions associated with the format defective sector and read replicated data operations.

LOCATE Command Parameters

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	1	bit string	Operation Byte specifies the type of record orientation that is required, and the operation to be performed when the desired track position is reached. `000. ....` – modifier bits must be zero `...x ....` – modifier bit ignored `.... xxxx` – operation code: • 0001 – Write data (1) • 0010 – Read replicated data (2) • 0100 – Format defective sector (4) • 0101 – Write and check data (5)

Offset Dec	Offset Hex	Length	Type	Description
				• 0110 – Read data (6) • other values prohibited
+1	+1	1	unsigned	Auxiliary byte, if operation code is • 0001 – must be zero • 0010 – the replication count (See [9:3-9] for details. • 0100 – ignored • 0101 – must be zero • 0110 – must be zero
+2	+2	2	unsigned	Specifies the number of sequential sectors to be processed by the command immediately following the LOCATE command. Must not be zero.
+4	+4	4	unsigned	Defines the relative displacement, in sectors, from the beginning of the data set to the first sector to be processed, the logical sector number of the first sector to be processed.

NO-OPERATION – X'03'

The NO-OPERATION (NO-OP) command is used to maintain channel connection during I/O operations. The NO-OPERATION command is processed as an immediate command. It causes no action at the addressed device. Channel End is signaled immediately upon receipt of the command code.

Chaining Requirements

None.

Status

Channel End and Device End are presented in initial status.

READ – X'42'

The READ command causes data to be transferred from a device to the channel. Upon receipt of the READ command, the control unit verifies correct orientation as specified by the preceding LOCATE command. After verification of orientation the following 512-byte sector is read and transferred to the channel. This process is repeated until the sector count specified by the preceding LOCATE command reaches zero. If the CCW count is greater than the byte count derived from the sector count specified in the LOCATE command, data transfer stops when the sector count reaches zero. If the CCW count is less than the byte count derived from the sector count specified in the LOCATE command, data transfer stops when the CCW count reaches zero.

Chaining Requirements

The READ command must be chained from a LOCATE command or the command is rejected with Channel End, Device End and Unit Check Status.

Status

Initial status is normally zero. Channel End and Device End are presented after the data is transferred to the channel. Unit Check Status is presented when:

- the READ command is not chained from a LOCATE command, or
- the orientation specified by the LOCATE command is not for a read operation.

READ AND RESET BUFFERED LOG – X' A4'

The READ AND RESET BUFFERED LOG command transfers up to 24 bytes of usage and/or error information from the control unit to the channel. The format of the usage/error information is that of format 6 sense information. The usage/error statistics pertain to the logical device addressed. The statistics are reset to zero after data transfer is complete.

Chaining Requirements

None.

Status

Initial status is normally zero. Channel End and Device End are presented after the usage/error information is transferred.

Sense Information Format 6

Columns “3310” and “3370” indicate which fields of the Format 6 sense information is used by the respective devices. Byte 2, bit 3, will be set to 1 indicated environment information is present.

Offset Dec	Offset Hex	Length	Type	3310	3370	Description
+0	+0	7	bit string	X	X	See “SENSE – X'04'” section format 0 sense information format, bytes 0-6 for details.
+7	+7	1	unsigned	X	X	See Format 6 and Message Table Values below.
+8	+8	3	unsigned	X	X	Number of blocks read.
+11	+B	2	unsigned	X		Correctable data checks.
+13	+D	1	unsigned	X		Number of data checks retried.
+14	+E	1	unsigned		X	Number of times access offset invoked.
+15	+F	3	unsigned	X	X	Number of blocks written with verify.
+18	+12	1	unsigned	X		Number of data check errors recovered.
+19	+13	2	unsigned	X	X	Number of access motions.
+21	+15	1	unsigned	X		Number of access errors.
+22	+16	1	unsigned	X	X	Number of service overruns.
+23	+17	1	unsigned		X	Number of command overruns.

Sense Format and Message Table Values (Byte)

Mask / Value	Bits(s)	Description
0110	0-3	Sense format 6. Always X'6'
.... xxxx	4-7	3370 Format 6 Message Table: • 0000-0111 – Not used. • 1000 – Indicates information in bytes 22 and 23 applies to channel A. • 1001 – Indicates information in bytes 22 and 23 applies to channel B. • 1010 - Indicates information in bytes 22 and 23 applies to channel C. • 1011 - Indicates information in bytes 22 and 23 applies to channel D. • 1100-1111 – Not used. 3310 does not use the Format 6 message table. Zeros.

READ DEVICE CHARACTERISTICS – X'64'

The READ DEVICE CHARACTERISTICS command transfers up to 32 bytes of device characteristic information from the control unit to the channel.

READ DEVICE CHARACTERISTICS Information Format

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	1	bit string	Operation modes. See value table below.
+1	+1	1	bit string	Features. See value table below.
+2	+2	1	unsigned	Device Class. See Model Table below.
+3	+3	1	unsigned	Unit Type. See Model Table below.
+4	+4	2	unsigned	Physical record size. Must be 512.
+6	+6	4	unsigned	Number of sectors per cyclical group. See Model Table below.
+10	+A	4	unsigned	Number of sectors per access position. See Model Table below.
+14	+E	4	unsigned	Number of sectors under movable access mechanism. See Model Table below.
+18	+12	6	reserved	Must be zeros.
+24	+18	2	unsigned	Number of sectors in the CE area.
+26	+1A	6	reserved	Must be zeros.

Operation Mode Values (Byte 0)

Mask / Value	Bits(s)	Description
`0... ....`	0	Reserved. Must be 0.
`.x.. ....`	1	Overrunable • 0 – not overrunable • 1 - overrunable
`..x. ....`	2	Channel mode • 0 – byte mode. • 1 – burst mode.

Mask / Value	Bits(s)	Description
...x	3	Data chaining • 0 – not allowed • 1 - allowed
.... 0000	4-7	Reserved. Must be 0000.

Feature Values (Byte 1)

Mask / Value	Bits(s)	Description
0...	0	Reserved. Must be 0.
.x..	1	Removable device • 0 – not removable • 1 - removable
..x.	2	Shared device. • 0 – not shared. • 1 – shared.
...x	3	Reserved. Must be 0.
.... x...	4	Movable access mechanism. • 0 – not movable • 1 - movable
.... .000	5-7	Reserved. Must be 0000.

Model Table Values

Description	Type	Cyclical Group	Blocks / Access Position	Device Sectors
Byte(s)	**3**	**6-9**	**10-13**	**14-17**
0671	X'12'	63	504	574560
0671-04	X'12'	63	504	624,456
0671-08	X'12'	63	504	513,072
3310 3310-1	X'01'	32	352	125,664
3370 3370-1 3370-A1	X'02'	62	744	558,000

Fixed-Block Architecture Structures

Description	Type	Cyclical Group	Blocks / Access Position	Device Sectors
Byte(s)	**3**	**6-9**	**10-13**	**14-17**
3370-A2				
3370-2 3370-A2 3370-B3	X'05'	62	744	712,752
9313	X'08'	96	480	246,240
9332 9332-400	X'07'	73	292	360036
9332-600	X'07'	73	292	554,800
9335	X'06'	71	426	804,714
9336 9336-10	X'11'	63	315	920115
9336-20 9336-25	X'11'	111	777	1672881

READ INITIAL PROGRAM LOAD – X'02'

The READ INITIAL PROGRAM LOAD (READ IPL) command causes the control unit to orient to and then read sector 0. Upon receipt of the READ IPL command, the control unit establishes an extent of maximum allowable size with an offset of zero, a mask byte of zero, and an addressable block size of 512. The control unit then orients to sector zero and reads the entire block.

The READ IPL command will not transfer more than 512 bytes or transfer data from any sector other than 0.

Chaining Requirements

The READ IPL command must be the first command in a chain or must be chained from another READ IPL command or the command is rejected with Channel End, Device End, and Unit Check status.

Status

Initial status is normally zero. Channel End and Device End are presented after the data has been transferred to the channel.

SENSE – X'04'

The SENSE command transfers 24 bytes of sense information from the control unit to the channel. The sense information transferred by this command describes the reasons for Unit Check status, the current status of the device that performed the operation, and system error recovery information.

A Unit Check should always be followed by a SENSE command whether the information is used or not. Otherwise, expected future interrupts may not occur and some I/O paths may not be available. A contingent connection state is established in the control unit after the channel accepts a status byte containing Unit Check. It lasts until a command (other than TESTIO, or NO OPERATION) receives an initial status byte of zero for the control unit and device address which generated the Unit Check. During the contingent connection state, the control unit is busy to all addresses other than the address for which the contingent connection was established.

Sense information is reset to zero after the data transfer is complete, or when an initial status byte of zero is given to any command except Test-I/O or NO OPERATION.

A SENSE command issued during a contingent connection should be a standalone command. This is because much of the device status testing normally performed as part of initial selection is bypassed so that zero initial status can be presented and error information related to the last Unit Check can be reported. If other commands are chained from the SENSE command, the device may not be prepared to execute them, which could result in unpredictable results.

Sense information may occur in seven different formats as specified in byte 7, bits 0-3:

- Format 0 – program or system check
- Format 1 – device equipment check (CE information)
- Format 2 – control unit equipment check (CE information)
- Format 3 – control unit control checks (CE information)
- Format 4 – data check without displacement information (uncorrectable data checks)
- Format 5 – data check with displacement information (correctable data checks)
- Format 6 – usage statistics/overrun errors.

See below for format 0 sense information. Refer to references 1 or 9 for information related

to formats 1-5. See the “READ AND RESET BUFFERED LOG – X'A4'” section for information related to format 6 information.

Chaining Requirements

None.

Status

Initial status is normally zero. Channel End and Device End are presented after the sense bytes are transferred.

Sense Information Format 0

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	3	bit string	Unit Check related sense information. See individual byte formats below. Zeros otherwise.
+3	+3	2	unsigned	Cylinder address of the most recent seek.
+5	+5	1	unsigned	Head address of the most recent seek
+6	+6	1	unsigned	Relative sector address of the most recent sector addressed.
+7	+7	1	unsigned	Sense information format and message code. See table below.
+8	+8	8	Binary	When byte 1, bit 7 (operation incomplete) is set and the error was not detected on a DIAGNOSTIC SENSE command, contains the updated LOCATE parameters. Otherwise, set to zeros.
+16	+10	2	unsigned	When byte 1, bit 7 (operation incomplete) is set and the error was not detected on a DIAGNOSTIC SENSE command, contains the number of blocks transferred to the system (excluding the error block). Otherwise, set to zeros.
+18	+12	6	reserved	Model dependent.

Sense Byte 0 Values

Mask / Value	Bits(s)	Description
x...	0	Command reject.
.x..	1	Intervention required.
..x.	2	Bus Out parity error if set to 1.
...x	3	Equipment check. Bytes 7-23 further define the condition.
.... x...	4	Data check.
.... .x..	5	Overrun.
.... ..00	6,7	Reserved. Set to 0.

Sense Byte 1 Values

Mask / Value	Bits(s)	Description
x...	0	Permanent error.
.x..	1	Sector size exception. Set to 1 when an invalid sector size is specified in bytes 2 and 3 of a DEFINE EXTENT command's parameters. Model dependent.
..0.	2	Not used. Set to 0.
...x	3	Permanent error in a storage director. Model dependent.
.... 0...	4	Not used. Set to 0.
.... .x..	5	File Protected. Set when a DIAGNOSTIC CONTROL or LOCATE command violates the logical extent established by a DEFINE EXTENT command.
.... ..x.	6	Write inhibited. A write operation is attempted on a drive that has its Write inhibit switch in the Read-Only position. Byte 0, bit 0 (command reject) is also set.
....x	7	Operation Incomplete. Refer to references 1 or 9 for details.

Sense Byte 2 Values

Mask / Value	Bits(s)	Description
x...	0	Check data error. Set to 1 when an uncorrectable data check is detected during the read-back verification phase of a WRITE command with write and check-data specified in the preceding

Mask / Value	Bits(s)	Description
		LOCATE command.
.x..	1	Set to 1 when the data check condition indicated by byte 0, bit 4 (data check) is correctable.
..0.	2	Set to 1 to indicate that the error rate for temporary data or seek checks has been exceeded and logging mode has been set for the device.
...x	3	Set to 1 when environment data is present, for example, when a READ AND RESET BUFFERED LOG command is executed.
.... 0000	4-7	Not used. Set to 0000.

Sense Byte 7 Values

Mask / Value	Bits(s)	Description
0000	0-3	Sense information format. Always 0 for format 0.
.... xxxx	4-7	Message code: • 0000 – no messge • 0001 – Storage director received an invalid command. • 0010 – Storage director received an invalid sequence of commands. • 0011 – The count specified in the CCW was less than required. • 0100 – The data argument of the command was invalid • 0101 – A DIAGNOSTIC CONTROL command was issued when prohibited by the DEFINE EXTENT mask. • 0110 – The channel did not indicate chaining when retry status was presented. • 0111 – The command portion of the CCW that was returned did not match the command for which retry was signaled. • 1000-1011 – reserved. • 1100 – A LOCATE command with a format defective block specified in the operation byte was issued when the alternate space was exhausted. • 1101 – A service overrun occurred in the data area. • 1110-1111 – reserved.

SENSE ID – X'E4'

The SENSE ID command transfers seven bytes of sense information from the storage director to the channel. The sense information transferred by this command describes the type and model of the control unit and device being addressed by this command.

Chaining Requirements:

None.

Status

Initial status is normally zero. Channel End and Device End are presented after the sense bytes are transferred.

SENSE ID Command Information Format

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	1	hex	Always X'FF'.
+1	+1	2	hex	Control unit type number.
+3	+3	1	hex	Control unit model number.
+4	+4	2	hex	Device type number.
+6	+6	1	hex	Device model number.

SENSE ID Model Values

Description	constant	CU Type	CU Model	Device Type	Device Model
Byte(s)	**0**	**1, 2**	**3**	**4, 5**	**6**
0671	X'FF'	X'6310'	X'01'	X'0671'	X'00'
0671-04	X'FF'	X'6310'	X'01'	X'0671'	X'04'
0671-08	X'FF'	X'6310'	X'01'	X'0671'	X'08'
3310	X'FF'	X'4331'	X'01'	X'3310'	X'01'
3370 3370-1 3370-A1 3370-A2	X'FF'	X'3880'	X'01'	X'3370'	X'00'

Description	constant	CU Type	CU Model	Device Type	Device Model
Byte(s)	**0**	**1, 2**	**3**	**4, 5**	**6**
3370-2 3370-A2 3370-B3	X'FF'	X'3880'	X'01'	X'3370'	X'04'
9313	X'FF'	X'6310'	X'01'	X'9313'	X'00'
9332 9332-400	X'FF'	X'6310'	X'01'	X'9332'	X'00'
9332-600	X'FF'	X'6310'	X'01'	X'9332'	X'01'
9335	X'FF'	X'6310'	X'01'	X'9335'	X'01'
9336 9336-10	X'FF'	X'6310'	X'01'	X'9336'	X'00'
9336-20 9336-25	X'FF'	X'6310'	X'01'	X'9336'	X'11'

UNCONDITIONAL RESERVE – X'14'

The UNCONDITIONAL RESERVE command breaks device allocation to the primary (failing) path and establishes allocation to the alternate path in the same system if a channel switch feature is installed in the control unit or a string switch feature is installed in the device.

The UNCONDITIONAL RESERVE command is used to recover from hardware malfunctions. It performs all of the functions of the DEVICE RESERVE command, and, in addition, reserves the device to the alternate path even when the device was reserved or in use through the primary path. Reservation or information in use for the primary path is reset in the device and control unit through which the command was issued. The UNCONDITIONAL RESERVE command does not reset the information in the control unit which is not not operational.

Control of the device must be established by the system before the UNCONDITIONAL RESERVE command can be issued. Device control is established if the channel has the device reserved, or the channel has a channel program in progress. If the channel issues an UNCONDITIONAL RESERVE command to a device not assigned to it, one of the following conditions may occur to the other system attached to the string switch.

- If the device is reserved, the reserve is reset and the device becomes reserved to the channel that issued the UNCONDITIONAL RESERVE command.
- An interruption condition will be lost if the device is disconnected between chained commands.
- A recoverable equipment check is presented if the device is active when the command is executed.
- If the device is idle and not reserved, there is no effect.

If the system does not want the device reserved to the alternate path, it must issue a DEVICE RELEASE command. (The DEVICE RELEASE command may be chained to the UNCONDITIONAL RESERVE command.)

The UNCONDITIONAL RESERVE command will be executed regardless of any abnormal device conditions (such as offline or unsafe) unless the device does not respond to the selection tag or there is a preselection check (indicated by Unit Check status and Equipment Check in sense byte 0).

Chaining Requirements

The UNCONDITIONAL RESERVE command must be the first command in a chain or the

command is rejected with Channel End, Device End, and Unit Check status.

Status

Initial status is normally zero. Channel End and Device End are presented after up to 24 sense bytes have been transferred.

WRITE – X'41'

The WRITE command causes data to be transferred from the channel to the control unit. Upon receipt of the WRITE command, the control unit reads the sector ID and verifies correct orientation. After verification of orientation, the following data sector is written with data transferred from the channel. This process is repeated until the sector count specified in the previous LOCATE command reaches zero.

If the CCW count is greater than the byte count derived from the sector count specified in the LOCATE command, data transfer stops when the sector count reaches zero. If the CCW count is less than the byte count derived from the LOCATE command, data transfer stops when the CCW count reaches zero, but the storage director continues to write zeros in the data blocks until the sector count reaches zero.

If access boundaries are encountered during data transfer, the control unit performs the appropriate access movement.

If write and check data is specified in the preceding LOCATE command, the control unit re-initializes the sector count and initiates an access back to the first sector and presents Channel End status. When the access is complete, the control unit reads the sector ID and verifies correct positioning. The following data sectors are then read by the control unit, but the data is not transferred to the channel. After all sectors are read (sector count equals zero), the control unit presents Device End status.

Note: Some software applications post the I/O operation as complete if Unit Status containing Channel End only is stored. As a result, a subsequent Device End and Unit Check may not be made available to the user. This condition can occur if a WRITE CCW with write and check-verify specified in the preceding LOCATE command is the last CCW in the chain. Some software applications provide an option to delay posting until Device End is received. If such an option is not provided, the user must take special action. A write sequence with verify that would otherwise terminate the chain must be followed by a non-write CCW (such as a NO-OP).

Chaining Requirements

The WRITE command must be chained from a LOCATE command or the command is rejected with Channel End, Device End, and Unit Check Status.

Status

Initial status is normally zero. If write data is specified in the operation byte of the preceding

LOCATE command, Channel End and Device End are presented by the control unit after data transfer has been completed. See the preceding description for status presentation when write and check data is specified.

FBA Channel Programs

This section combines the information from the preceding section into a set of typical programs. The assumption, as with the previous section, is made that the reader is familiar with channel programming in general and simply needs to understand how to combine the various FBA channel commands to achieve a specific goal.

FBA DASD Device Discovery

Two commands provide information about the device:

- SENSE ID and
- READ DEVICE CHARACTERISTICS.

Neither command has any chaining requirements, so they can be issued separately or together regardless of the sequence, chaining the second to the first. SENSE ID provides the highest level of information about the device, namely its control unit, type and model, and the device's type and model. Depending upon the discovery process itself, either may make sense as being issued first or together. Review of the supplied information and the discovery process informational needs should dictate how the two are used. If non-standard device sizes are expected as a possibility, READ DEVICE CHARACTERISTICS must be used to determine the actual size of the volume.

Accessing FBA DASD Sector Content

One or more sectors' content is accessed for two purposes:

- reading the current content or
- changing the current content.

The process is nearly identical. Through chained commands, control unit is prepared for the access followed by the actual access. The chaining requirements of the commands dictates that the following sequence is required:

- DEFINE EXTENT,
- followed by LOCATE,

- followed by either READ, accessing existing sector content, or WRITE, changing existing content.

The DEFINE EXTENT command places bounds on the accessed sectors and type of access, and establishes the logical sector numbers used during the access. The extent being defined is usually based upon information within the space management records, converted to logical sector numbers. The LOCATE command dictates the details of the current access and determines the amount of data that will be transferred as well as the direction. The final READ or WRITE command must be consistent with

- the controls imposed by the DEFINE EXTENT mask,
- the LOCATE command operation code (also consistent with the DEFINE EXTENT mask), and
- the quantity of information expected to be exchanged.

The READ and WRITE commands, with appropriate preceding commands, transfer across the channel a single transfer unit, typically being a control interval. See the “Control Interval” section for details on its structure.

Initial Program Load

The design of the initial program load function places some requirements on a device supporting the function. By design 24-bytes of information are transferred from the device and placed at memory addresses 0 through 23, inclusive. This provides only a PSW at address 0 for entering the loaded program and two CCWs for bootstrapping the remainder of the channel program that loads the information. The reading of the remainder of the channel program is usually accomplished by the first CCW at memory address 8. The second CCW at address 16 usually is a TIC command to the remainder of the channel program, placed in memory by the hardware generated command. The remainder of the command chain reads the program into memory. For a FBA DASD device, the remainder of the program is just another access to existing content within specific sectors of the device.

The READ IPL command, which is the same command code for all devices, does double duty. When generated by the CPU hardware during the IPL function, the READ IPL accesses sector zero, transferring just the initial 24 bytes to memory. The first CCW at address 8, is another READ IPL, the second time reading as much as the entire sector zero into memory.

The READ IPL command also implicitly creates an extent consisting of the entire volume. In this extent the logical sector number of the first sector is 0. With the start of the extent being physical sector 0, the remainder of the chained commands operate in an environment where

physical and logical sector numbers are the same. The process of accessing the program stored on the device is simply a read type operation described above. Because the READ IPL provides its own extent definition, the DEFINE EXTENT command is not needed and is effectively replaced by the READ IPL. The program is then read by use of a LOCATE, followed by a READ (as opposed to a READ IPL) command. If the load process needs to load data into different memory locations, multiple LOCATE/READ sequences may be chained together to do so. Each LOCATE command would of course require its own set of parameters.

Putting it together, the sequence of chained commands in memory and its content follows this example. The row in white is a hardware generated command. The rows in red are brought into memory by the hardware generated READ IPL command. The rows in blue are brought into memory by the second READ IPL command. The row in green is brought into memory by the READ command. All CCW's except the last perform command chaining.

Location	Command	Length	Address	Description
(none)	READ IPL	24	X'0000'	This command, generated by the CPU, does not reside in memory but is seen by the FBA DASD control unit as the first command of the chain. It reads bytes 0-23 of sector 0 into memory locations 0 through 23 and automatically continues with the CCW at location X'0008'. The data read by this command is in red.
X'0000'			X'3000'	IPL PSW used to enter the program at completion of the IPL chained commands.
X'0008'	READ IPL	512	X'2000'	Reads sector zero into memory again, containing the rest of the command chain to read the program into memory. For this table we are assuming it is read into memory at X'2000'-X'21FF'. The data read by this command is shown in blue.
X'0010'	TIC	0	X'2018	Transfers control to the rest of the channel program read by the preceding READ IPL command. The transfer can be to anywhere between bytes 24 and 496 of sector 0. This is because the first 24 bytes is the same as that read by the hardware generated READ IPL and at least two CCW's are required to read the program, making byte 496 of the sector the last possible location. For this example, we will assume the remainder of the channel program immediately follows the first 24 bytes, so location X'2018'.
X'2000'			X'3000'	The same as the PSW at X'0000', ignored
X'2008'	READ IPL	512	X'2000'	The same as the CCW at X'0008', ignored
X'2010'	TIC	0	X'2018'	The same as the CCW at X'0010', ignored
X'2018'	LOCATE	8	X'2028'	Identify the sectors using physical sector numbers being read.
X'2020'	READ	1024	X'3000'	Read the sectors where the program is expected to

Location	Command	Length	Address	Description
				reside. The length is the number of sectors multiplied by 512. For this example, it is assumed that the program is 1024 bytes in length and it is read into memory at location X'3000'. The data read by this command is in green.
X'2028'				Parameters for LOCATE command at X'2018'
X'2030-X'21FF'				May contain anything else but is read by the second READ IPL command.
X'3000'-X'3FFF'				Following completion of the READ at X'2020', these bytes contain the program. The IPL PSW at address 0, read by the hardware generated READ IPL command, must contain an address within the range of the program to which control of the CPU is passed. In this example the program is entered at the first byte of the program, X'3000'.

It should be obvious from this example, how much must be planned ahead in terms of where the program resides on the volume and what memory locations will be participating in the program loading. All of the critical information resides in sector 0 for an FBA DASD device that may participate in the IPL function.

Error Reporting

During processing errors can occur. Error information is reported using sense information. On physical hardware this information can exist in seven different formats. Which format is presented is determined by which command is used for its access. Because this document focuses on the information likely to occur with emulated FBA DASD only two of the formats are likely to occur, format 0 and format 6. Only the commands presenting these formats are identified here. The other formats only occur with one of the two diagnostic commands. Refer to references 1 or 9 for details.

The five related commands are:

- DEVICE RELEASE – presenting format 0 sense information
- DEVICE RESERVE – presenting format 0 sense information
- READ AND RESET BUFFERED LOG – presenting format 6 sense information.
- SENSE – presenting format 0 sense information
- UNCONDITIONAL RESERVE – presenting format 0 sense information

The format 0 sense byte 2, bit 2, when set to one, indicates if logging has started for the

device. Further when byte 2, bit 3 is set to one, the buffered log needs to be retrieved for usage statistics. Depending upon the emulation these bits may or may not ever be set. But if bit 3 is set, the program should retrieve the format 6 sense information using the READ AND RESET BUFFERED LOG command.

Hercules FBA Device Emulation

Hercules emulates an FBA volume using a host disk file. Emulation of the device types identified in the “FBA DASD Model Capacities” section are supported by Hercules. Additionally Hercules can emulate non-standard sized FBA DASD volumes in either compressed or uncompressed formats. No CE area is emulated for FBA volumes.

The `dasdinit` utility may be used to create either an uncompressed or a compressed FBA volume. The `devtype` command line argument will dictate the size of the volume unless a specific size is specified in the utility's command line `size` argument. Compressed FBA volumes must be created by the `dasdinit` utility. Uncompressed volumes may be created by any program provided the created emulating file has a length in multiples of whole sectors, that is, multiples of 512 bytes. When the raw flag, the `-r` command line option is not used, a simple Volume Label is written to sector 1. Initialization by an operating system or program is still required for the FBA volume.

The `dasdcopy` utility can create a compressed FBA volume from an uncompressed volume or create an uncompressed FBA volume from a compressed volume. Note the `dasdconv` utility performs specific conversions for CKD volumes only.

No Hercules utility exists for the loading of a FBA volume with data sets. The `dasdload` utility only operates on created CKD volumes. Because the file emulating an uncompressed FBA volume is simply a binary file on the host platform, utilities independent of Hercules can be used to load data onto the volume. Once loaded it can be compressed using the `dasdcopy` utility if desired.

The attributes of the volume as reported by SENSE ID and DEVICE CHARACTERISTICS channel commands are those of the model identified in the Hercules configuration with the exception of the number of supported sectors. Supported sectors always reports the number of sectors being emulated by the emulating file, compressed or uncompressed, and is always complete block groups. If the configuration statement does not identify the device type or it is unrecognized, Hercules assumes the device is a 3370.

Block Groups

The host file emulating an FBA volume is accessed in a single unit, the block group. Regardless of the volume’s sector capacity, the physical host file will contain complete block groups. Each block group contains 120 sectors (each sector being 512 bytes in length) of

61,440 bytes.

Hercules compressed FBA volumes actually uses the Hercules support for compressed CKD volumes. Each block group is handled for compression purposes as a single “track”.

Command Considerations

A major consideration when accessing FBA DASD volumes is the lack of support for data chaining or skipping in the Hercules FBA driver module. Command chaining works fine, but scatter/gather data handling or skipping of data by operations using data chaining within one or more sectors is not supported by Hercules. If data chaining is encountered, Hercules terminates the command with Channel End, Device End, Unit Check status and sense byte 0 indicating an overrun condition occurred.

DEVICE RELEASE

All 24 bytes of format 0 sense information are transferred. However, Hercules never sets any sense information in a byte other than byte 0. All other bytes are always zero.

DEVICE RESERVE

All 24 bytes of format 0 sense information are transferred. However, Hercules never sets any sense information in a byte other than byte 0. All other bytes are always zero.

DIAGNOSTIC CONTROL

Not implemented by Hercules.

DIAGNOSTIC SENSE

Not implemented by Hercules.

DEFINE EXTENT

Record size parameter is ignored and not checked for being in error.

READ AND RESET BUFFERED LOG

Format 0 sense information of all zero bytes is transferred rather than format 6 sense information.

READ DEVICE CHARACTERISTICS

The Hercules configuration file dictates the model of an emulated FBA device. Any emulating

file may be defined as any model. The number of sectors in the CE area is always zero.

Hercules provides the same value for operational mode (Byte 0) for all models: X'30' meaning:

- not overrunable,
- operating in burst mode, and
- data chaining allowed.

Note that the “not overrunnable” flag is inconsistent with Hercules handling of data chaining.

Hercules provides the same value for features (Byte 1) for all models: X'08' meaning”

- device not removable,
- device is not shared and
- device has movable access mechanism.

These reported values for operational mode and features are consistent for a 3310 FBA device. Whether they are for any other model is unknown.

Number of sectors (Bytes 14-17) for non-standard sized device report the actual number of sectors available on the non-standard device. The remaining values for a non-standard device reflect those reported as follows for the standard device found in the Model Table of the command's description in the previous section:

- non-standard 0671 reported as a 0671-08,
- non-standard 3310 reported as a 3310-1,
- non-standard 3370 reported as a 3370-2,
- non-standard 9313 reported as a 9313,
- non-standard 9332 reported as a 9332-600,
- non-standard 9335 reported as a 9335, and
- non-standard 9336 reported as a 9336-25.

SENSE

All 24 bytes of format 0 sense information are transferred. However, Hercules never sets any sense information in a byte other than byte 0. All other bytes are always zero.

The contingent connection state does not exist for Hercules emulated FBA devices. Control

unit busy is never reported.

UNCONDITIONAL RESERVE

All 24 bytes of format 0 sense information are transferred. However, Hercules never sets any sense information in a byte other than byte 0. All other bytes are always zero.

WRITE

Write and check with verification is treated as a normal write command.

Control Interval Format

The lack of hardware end-of-file condition detection and requirement of physical data records being a multiple of 512 bytes, drives the need for a software utilized convention supporting both variable length records and end-of-file detection. The control interval and its standardized fields satisfies these needs. As a software convention, the definitions can be expanded to satisfy the needs of the software.

Once the control interval is created with its application data records and stored, it, in most cases, becomes read-only. Only when the data is rewritten as part of recreating the data set or storing of a new data set in the same location does the control interval's content change.

Applications that provide special storage mechanisms and operate directly on the control interval itself do not need to abide by these rules. A mechanism for random logical record update exists requiring a specific structuring of the control interval and use of its control information. Logical data records organized in this way are referred to as slots. Each slot of the control interval must be of the same length and each must have its own control information (described below in the “Record Definition Field (RDF)” section). An application using slots must have direct access to the control interval in memory for updating of the slot information and control information. Following such updates the entire control interval can be stored back into its original location on the FBA volume.

The control interval is the foundation for managing data on FBA devices. The control interval resides in a single physical data record (in potentially multiple sequential sectors). Management of the volume's content for the placement of application data on the device is managed by a set of application data records within a special data set called the Volume Table of Contents (VTOC). Control intervals are used by the VTOC for its application data records as are most other application containing data sets.

The hardware device manages the physical access to sectors. Software manages the access to application data records by creating and utilizing the information stored within the control interval. By necessity, the size of a control interval must be an integral number of 512-byte sectors. The size of a control interval must therefore be a multiple of 512-bytes. Because the control interval is accessed in a single I/O operation, the maximum number of sectors that can be read by a channel command is 127 sectors (the number of whole sectors not exceeding the maximum bytes transferred by a single I/O command, 65,535 bytes), or 65,024 bytes.

In addition to the following sections, see [2, 185-189] for more details and examples of how the control information within the control interval is used.

Control Interval Definition Field (CIDF)

The control interval tracks the placement of logical data records within the control interval. Starting from the first byte of the control interval, logical data records are added at increasing displacements into the control interval. The **last four bytes** of the control interval contains the control interval definition field (CIDF). The CIDF defines how much free space remains in the control interval and where the free space starts. The CIDF has this format described in this table. “cisize” refers to the length of the control interval (in multiples of 512-bytes).

Position	Length	Type	Description
cisize - 4	2	unsigned binary	Free space offset
cisize - 2	2	unsigned binary	Free space length

Preceding the CIDF are zero or more record definition fields (RDF). Each RDF, described in the next section, is three bytes in length and is placed immediately preceding the CIDF (for the first RDF added) or preceding the most recently added RDF. As each RDF is placed in the control interval the free-space length is reduced at the end by three bytes. Each additional RDF has the exact reverse effect on free space as adding a logical data record. Each logical data record shrinks the free space from the beginning. When an application data record is added the free space offset is increased by the length of the logical data record. The free space resides between the logical data records positioned at the start of the control interval and the RDFs/CIDF placed at the end of the control interval.

For an empty control interval, the CIDF free space offset will be 0 because no logical data records reside in the control interval. The CIDF free space length will be cisize-4 because no RDF's exist in the control interval, only the CIDF. If the control interval is completely filled and has no free space, the CIDF free space offset will be the location within the control interval of the last RDF added to the control interval (which is the byte immediately following the last logical data record in this case). The CIDF free space length will be zero.

The end of a data set or software end-of-file (SEOF) is indicated by a control interval containing a CIDF with both the offset and the length fields set to zero. By setting the free space length to zero, no additional logical data records may be added to the control interval. By setting the free space offset to zero, no logical data records reside in the control interval.

Record Definition Field (RDF)

The record definition field describes various types of logical records within a control interval. In some cases more than one RDF is used. In this case there is a left hand RDF and a right

hand RDF.

It is useful for some circumstances for reuse of a logical data record with new content and a way to indicate when such a logical data record is “available”. In this case all logical data records are required to be of the same length. The position where each logical data record would reside is considered a slot which may be available or in use. In this situation each logical data record has its own RDF indicating the status of its availability.

The three-byte record-definition field contains two fields.

- Byte 0 – a flag field
- Bytes 1,2 – a binary number whose meaning is defined by the flag field

The flag field assigns meaning to the following bits

Mask / Value	Bits(s)	Description
x...	0	Whether logical data record or group of same length logical data records associated with this RDF • 0 – is not the last logical data record or records • 1 – is the last logical data record or group of logical data record contained in this group of one or more control intervals. See Note 1.
.x..	1	Whether there is a paired RDF immediately to the left of this RDF: • 0 – this RDF is not paired with another RDF • 1 – this RDF has a paired RDF to its left.
..xx	2, 3	Indicates whether the logical data record spans control intervals • 00 – no • 01 – yes, this is the first segment • 10 – yes, this is the last segment • 11 – yes, this is an intermediate segment
.... x...	4	Indicates the meaning of the two-byte binary number field: • 0 – the length of the logical data record, segment or slot described by this RDF • 1 – the number of consecutive non-spanned records of the same length or spanned record update count depending upon bits 2 and 3.
.... .x..	5	For a fixed length slot, indicates whether the slot described by this RDF contains a logical data record or is empty or available • 0 – slot contains a logical data record • 1 – slot is available for logical data record content
.... ..xx	6,7	Reserved bits (must be zero)

Note 1: The use of bit 0 to indicate an end-of-file condition is an extension of the standard and eliminates the need for an entire control interval dedicated to software detection of the end-of-file condition. In most cases a data set will have additional unused space within its last or only defined extent and setting aside a single control interval for end-of-file condition has no

additional cost in consumed space. However in a partitioned data set containing multiple end-of-file conditions the use of an entire control interval can become quite expensive in terms of lost space. This extension to the RDF definition, eliminates this cost when used. The description does not restrict the use of bit 0 to partitioned data sets.

The meaning of the two-byte binary field is controlled by the flag field.

Flag Field	Meaning of Two-Byte Binary Number
`.... 0...`	The length of the logical data record, length of a spanned logical data record segment or slot described by this RDF
`..00 1...`	Number of consecutive records of the same length as described by the RDF to the right of this RDF

`..01 1...`
`..10 1...`
`..11 1...`
	The update number of the spanned record described by the RDF to the right of this RDF.

Locating the position of a specific logical data record within the control interval is the result of adding together all of the RDF lengths of each RDF to the right of the specific logical data record's RDF. This gives the offset from the start of the control interval to the start of the specific RDF. In the case where multiple records have the same length, and the RDF is paired with a RDF to its left, the length of the set of same size records is the product of the binary value of the left RDF (the number of same length records) and the binary value of the right RDF (the length of each of the records). The first RDF to the left of the CIDF is always positioned at the start of the control interval, or its offset is always 0.

Sequential Record Definition Fields

Paired and unpaired RDF's may be freely intermixed within the control information of the control interval. How the size of the logical data records vary also influences the use of RDF's. The RDF flag field is identified for each RDF described below.

When the logical data record is of a fixed length, a pair of RDF's may define all of the logical data records in the control interval:

- Left RDF – X'04' – Number of consecutive logical data records of the same length
- Right RDF – X'40' – The size of each consecutive logical data record.

When the logical data records vary in size or there is a single logical data record, a single RDF suffices for each record:

- Logical Data Record RDF – X'00' – The size of this logical data record.

When a logical data record can not be contained within a single control interval it can be spanned between multiple control intervals. Each segment of the spanned logical data record resides in separate control intervals and will be:

- First Segment RDF – X' 10' – Length of the first segment in this control interval
- Middle Segment RDF – X'30' – Length of this middle segment in this control interval
- Last Segment RDF – X'20' – Length of this last segment in this control interval.

The first segment may be preceded in its control interval with other RDF's to its right. Correspondingly, the last segment RDF may have other RDF's to its left. The middle segment will always be the only logical data record in its control interval and is only used if the complete logical data record's length requires it.

Slot Record Definition Fields

When each logical data record position within the control interval is of the same size and is used as a slot, each logical data record position has its own RDF that may be either:

- Used slot RDF – X'00' – The size of this used slot, or
- Available slot RDF – X'02' – The size of this available slot.

All of the RDF's will contain the same size, namely, the size of a slot. By providing an RDF for each slot, its availability for new content can be altered over time by simply changing the slot's RDF flag. Slot RDF usage is mutually exclusive from sequential RDF usage.

Virtual Storage Access Method Spanned Records

Virtual Storage Access Method (VSAM) spanned records, in either entry-sequenced or key-sequenced data sets, use an additional paired RDF for each spanned record's segment, two RDF's per segment.

- Right RDF – X'50', X'70', or X'60' for the length of the first, middle or last spanned segment,
- Left RDF – X'18', X'38', or X'28' containing the update number of the respective first, middle or last spanned segment.

Logical Record Slots

Slots are a specific use of RDF's in a specific configuration. They depend upon one two conditions:

1. each logical record data slot has its own RDF, and
2. each logical record slot is the same length.

The first condition may be true for a control interval containing variable length logical data records, but not the second. Correspondingly, a sequential data set composed of logical data records of the same size will satisfy the second condition, but typically not the first.

These two conditions allow the logical data record slots and RDF fields to be treated as two arrays the offset of either calculated based upon a slot number. It is not necessary to traverse the RDF fields to locate the offset of a specific logical record slot. All that is needed is the length of each slot. Each RDF field is by definition the same length, three bytes.

Control Interval Read Processing

Processing of a control interval is dependent upon the interplay between a number of components. At the foundation of the control interval is its size. The control interval size bounds all other aspects of the control interval related values. All offsets must be less than the control interval size minus four (four for the CIDF).

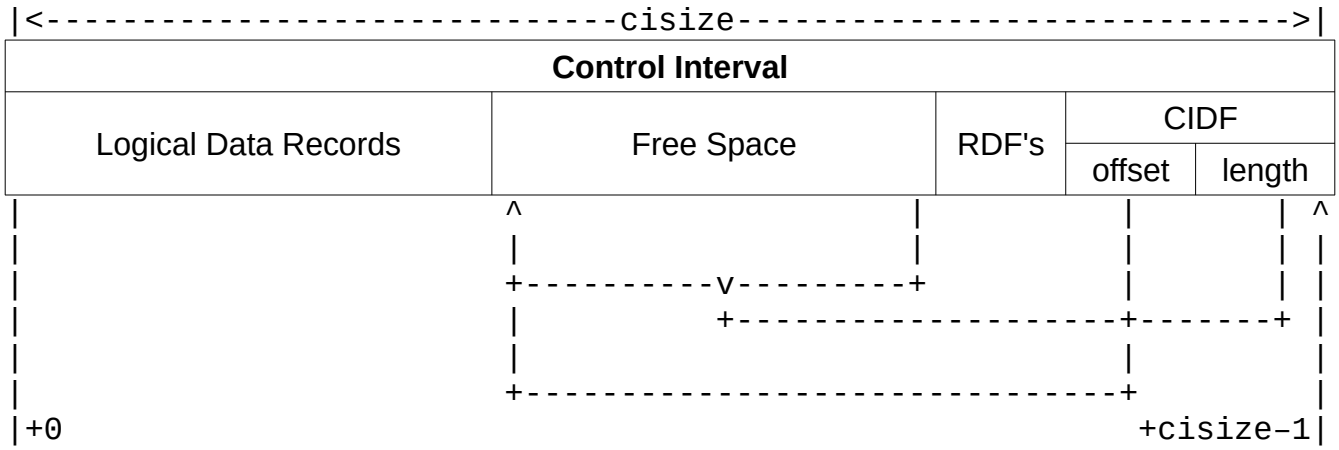
The general format of a control interval is illustrated in the following diagram. Initially the only structural information available comes from the CIDF residing in the last four bytes of the control interval. Before anything is done with the control interval, the CIDF must be inspected for both of the two-byte fields containing zeros. If this is the case, the control interval is a software end-of-file and the end of the data set has been reached. No further processing of this control interval nor any additional control intervals in the data set should occur.

Assuming the control interval is not a software end-of-file, the CIDF defines the starting position and length of any available space, *free space*, within the control interval. The starting position is an offset from the start of the control interval. In this section the word “offset” will always mean “offset from the start of the control interval”. The combination of the free space offset and the length from the CIDF provides the offset of the last RDF (remember RDF's are added from a higher offset to a lower offset), if any are present, or the offset of the CIDF itself, if no RDF's are present.

Sanity Check 1: If the free space offset is 0, then the length **must** be *cisize* minus 4, indicating a control interval containing no logical data records. This is so, because no RDF's will exist if there are no logical data records and the offset of 0 for free space start means exactly that.

Sanity Check 2: If the free space offset is not 0, the free space offset plus the length **must** be less than or equal to *cisize* minus 7 because there must be at least one three byte RDF

present.



With the sanity checks passed, the following control values can be established:

- *rdf* – the offset of the current RDF, initially set to *cisize* – 4.
- *rdf_end* – the offset of the last RDF, initially set to CIDF offset plus CIDF length.
- *rdf_eof* – Whether the current RDF contains an end-of-file flag, when the DSCB-1 DS1OPCD field indicates this convention is being used by the data set. Initialized to False. Initially *rdf* has the offset of the CIDF field. This condition is never True before any RDF's have been processed.
- *ldr* – the offset of the next logical data record, initially set to 0.
- *ldr_len* – the length of the logical data record or records being processed, initialized to 0.
- *ldr_num* – the number of logical data records of this size being processed, initialized to 0.
- *ldr_end* – the offset immediately beyond the last logical data record, initially set to CIDF offset.

Detecting RDF End-of-File Condition

This process is only used if the DSCB-1 DS1OPCD field indicates the RDF EOF convention is being used by the data set. If that is not the case, this process is skipped, defaulting to use of the SEOF control interval EOF convention.

To determine if the RDF end-of-file condition exists, the *rdf_eof* control can be examined for being True. Because this control is always initialized to False at the start of control interval

read processing, the condition is never true before the first RDF is processed and is only set during processing.

Sanity Check 3: When *rdf_eof* is true, no additional RDF's should exist to the left of the current RDF. Hence *rdf* **must** equal *rdf_end*.

Locating the Next Logical Data Record

Locating the next logical data record requires locating the next RDF. Before the next RDF can be processed, it must be determined whether the current data set is at end-of-file because it is using the an RDF flag to signal end-of-file. To determine if the RDF end-of-file condition exists, the *rdf_eof* control can be examined for being True. Because this control is always initialized to False, the condition is never true before the first RDF is processed.

Space Management Records

Managing the allocation of portions of the volume for specific uses is accomplished by a special application, usually provided by the system, in conjunction with a set of logical records residing on the volume. Location of available space may also be maintained by this application's logical records. This section describes the logical records used by the space management application and the content stored on the volume. The content and fields evolved over time and changed with the space management application evolution on different operating systems.

Volume Table of Contents (VTOC)

A collection of related application data records constitutes a data set. Data sets are allocated one or more portions of the volume, each portion being a contiguous region called an extent. For FBA devices, extents are defined by a starting and ending sector number. The data set that maintains the allocation information itself resides in an extent. This data set is called the Volume Table of Contents. The extent of the VTOC itself is recoded within the VTOC in a specific logical record. Within the VTOC extent, control intervals are used to maintain the VTOC's logical data records within slots. Each slot contains a logical data record called a Data Set Control Block (DSCB). Various type of DSCB's provide different types of information. All DSCB records are 140 bytes (X'8C') in length. The following format types are defined. Those in bold are used with FBA devices.

- **Format 0 – a free DSCB containing all X'00' bytes**
- **Format 1 – defines a data set's attributes and up to three allocated extents**
- Format 2 – defines an ISAM data set, used only by CKD devices
- **Format 3 – defines additional allocated extents allocated beyond the first three**
- **Format 4 – defines the VTOC itself and its extent**
- Format 5 – describes unallocated space on the CKD volume and is optional
- Format 6 – shared extent allocations on shared cylinders, used only by CKD devices
- Format 7 – describes free space for certain devices, used only by CKD devices
- Format 8 – describes a data set with EAS, used only by CKD devices
- Format 9 – describes metadata and DSCB pointers, used only by CKD devices
- **Format 10 – describes unallocated space on an FBA volume and is optional.**

This document only discusses those formats and options used with FBA devices. See [3, 2-16] for details not included here for each format. Format 10 is defined only in this document. No implementation of space management on an FBA volume exists that tracks unallocated space.

The first slot of the first control interval of the VTOC extent describes the VTOC itself and provides other information related to the volume. This slot contains a DSCB Type-4 record. The first control interval is found from the logical data record contained in the reserved sector 1, the Volume Label Record. The Volume Label points to the start of the VTOC extent and includes the size of the control interval used by the VTOC allowing the space management application to read the first control interval thereby getting access to the first slot and the DSCB Type 4 record stored in it.

The information contained in this section combines information supplied by references 3, and 4. Formats and content are those defined by [4, III-06 – III-20]. Standardized field names are supplied by [3, 6-14].

VTOC Creation

A utility, ICKDSF, see [5, Appendix D], can create a VTOC on an FBA device volume. This utility can place the VTOC starting at any location other than sectors 0 and 1. The utility determines the VTOC size based upon the number of slots which may be any value between 3 and 999. The actual allocation is determined by the control interval size (a multiple of 512 between 512 and 8,192 bytes) and the number of requested slots. This table provides the number of slots per control interval based upon the sizes supported by ICKDSF.

Size	Slots	Size	Slots	Size	Slots	Size	Slots
512	3	2,560	17	4,608	32	6,656	46
1,024	7	3,072	21	5,120	35	7,168	50
1,536	10	3,584	25	5,632	39	7,680	53
2,048	14	4,096	28	6,144	42	8,192	57

Technically there is no reason for not using larger control interval sizes, but these are the sizes supported by the ICKDSF initializing utility.

The default VTOC is placed starting in sector 2, with a control interval size of 1024 providing 56 slots (8 control intervals in 16 sectors). If the utility places the VTOC at the end of the volume it will contain 99 slots (15 control intervals of 1,024 bytes in 30 sectors) with six unused slots in the last control interval for a total of 105. A system residence volume requires

special placement of the VTOC allowing room for the IPL program and other system specific sectors following sector 1.

Common Structures

Within the DSCB record types, some common structures are used. For conciseness they are described here and referenced by name within the detailed DSCB record type descriptions.

Date

This structure identifies a date in discontinuous binary format.

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	1	unsigned	Year minus 1900. For example 2017 is 117.
+1	+1	2	unsigned	Relative day of year, 1-366. 1 January being 1.

DSCB Slot Address

A DSCB slot address identifies the location within the VTOC of a specific slot. A slot address is 5 bytes in length. A DSCB slot address is described as using the format SSSSR.

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	4	unsigned	Starting VTOC logical sector number of the control interval in which the addressed slot resides
+4	+4	1	unsigned	Slot number starting from 1 of the slot being addressed

Extent Definition

An extent describes an area of the volume assigned for a specific purpose. Extent definition is used in various DSCB record types. Each extent definition is 10 bytes in length.

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	1	unsigned	Extent type • X'00' – Next three fields do not indicate any extent • X'01' – Data area

Offset Dec	Offset Hex	Length	Type	Description
+1	+1	1	unsigned	Extent sequence number. The first extent is 1.
+2	+2	4	unsigned	Lower limit – starting physical sector number of this extent.
+6	+6	4	unsigned	Upper limit – ending physical sector number of this extent

Volume Label Record (Reserved Sector 1)

The Volume Label Record resides in physical data record stored at sector 1. In addition to identifying the volume, the record provides basic information about the Volume Table of Contents.

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	4	EBCDIC	Volume label identifier, always “VOL1'
+4	+4	6	EBCDIC	Volume serial number. See Note 1.
+10	+A	1	hex	Security byte. X'C0' unless user security supported
+11	+B	1	reserved	X’00’
+12	+C	4	unsigned	Starting sector number of the VTOC. See Note 2
+16	+10	5	reserved	EBCDIC spaces X'40'
+21	+15	4	unsigned	VTOC control interval size. See Note 2.
+25	+19	4	unsigned	Number of sectors per control interval. See Note 2.
+29	+1D	4	unsigned	Number of DSCB slots per control interval. See Note 2.
+33	+21	4	reserved	EBCDIC spaces X'40'
+37	+25	14	EBCDIC	Volume owner and address, otherwise EBCDIC X'40'
+51	+33	29	reserved	EBCDIC spaces X'40'
+80	+50	432	reserved	Remainder of reserved sector physical record, X'00'

Note 1: 1 to 6 alphanumeric (A-Z, a-z, 0-9) or national (#, $, @) EBCDIC characters or an EBCDIC hyphen (X'60'), padded on the right with EBCDIC spaces X'40'.

Note 2: These fields allow the first control interval of the VTOC, containing the DSCB Type 4 record, to be accessed.

See [4, III-07] for details.

Data Set Control Block Format 0 – DSCB Available for Use

The Format 0 DSCB identifies a DSCB record not in use. Such records are, therefore, available for other DSCB record content.

The RDF associated with an unused slot should use a flag of X'02' with a length field containing 140. An unused slot should also contain a Format 0 DSCB Record. Software should recognize either condition as indicating slot availability.

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	140	Hex	All 140 bytes set to X'00'

Data Set Control Block Format 1 – Allocated Data Set Description

The DSCB-1 record describes a data set and its allocation by its first three extents. The space management application dictates the number of supported extents.

The RDF associated with a slot containing a Format 4 DSCB must use a flag of X'00' and length of 140.

Offset Dec	Offset Hex	Length	Field Type	Name	Description
+0	+0	44	Hex	DS1DSNAM	Name of the data set. See Note 1.
+44	+2C	1	EBCDIC	DS1IDFMT	Format identifier, EBCDIC 1 (X'F1') or DS1IDC
+45	+2D	6	EBCDIC	DS1DSSN	Volume name of the first or only volume containing the data set.
+51	+33	2	unsigned	DS1VOLSQ	Indicates the order of a volume relative to the first volume on which the data set resides, 1 being first.
+53	+35	3	Date	DS1CRDAT	Creation date.
+56	+38	3	Date	DS1EXPDT	Expiration date.
+58	+3A	1	flags	DS4VTOCI	VTOC indicators. See DS4VTOCI Flags below
+59	+3B	1	unsigned	DS1NOEPV	Number of extents on this volume.
+60	+3C	1	unsigned	DS1NOBDB	Number of bytes used in last directory block.
+61	+3D	1	reserved		X'00'
+62	+3E	13	EBCDIC	DS1SYSCD	System code. Identifies the programming system creating the data set.
+75	+4B	3	Date	DS1REFD	Date last referenced
+78	+4E	2	reserved		X'0000'

Offset Dec	Offset Hex	Length	Field Type	Name	Description
+80	+50	2	unsigned	DS1CISIZ	Control interval size in bytes.
+82	+52	2	bit string	DS1DSORG	Data set organization. See flags below
+84	+54	1	bit string	DS1RECFM	Data set record format. See flags below.
+85	+55	1	bit string	DS1OPTCD	Option code dependent upon DS1DSORG.
+86	+56	2	signed	DS1BLKL	Logical data record length (for unblocked fixed length records) or maximum logical data record length (for fixed block, undefined or variable records). See Note 2.
+88	+58	2	signed	DS1LRECL	Application data record length (for fixed length records), zero (for undefined records), maximum application data record length (for variable unspanned records), maximum application data record length (for variable spanned application records). See Note 3.
+90	+5A	1	unsigned	DS1KEYL	The record's key length. Always zero for FBA devices.
+91	+5B	2	unsigned	DS1RKP	Relative key position. Always zero for FBA devices.
+93	+5D	1	bit string	DS1DSIND	Data set indicators. See indicators below.
+94	+5E	4	binary	DS1SCALO	Secondary allocation request. Not used for FBA devices.
+98	+62	4	unsigned	DS1LSTRS	Last used data set relative sector. Note 4.
+102	+66	1	reserved		Not used for FBA devices. Must be zero.
+103	+67	2	unsigned	DS1PDSDB	Number of sectors allocated to directory blocks if partitioned organization, zeros otherwise. Note 5.
+105	+69	10	extent	DS1EXT1	First extent of this data set.
+115	+73	10	extent	DS1EXT2	Second extent of this data set.
+125	+7D	10	extent	DS1EXT3	Third extent of this data set.
+135	+87	5	SSSSR	DS1PTRDS	DSCB slot address of a Format-3 DSCB or zeros.

Note 1: 1 to 6 alphanumeric (A-Z, a-z, 0-9) or national (#, $, @) EBCDIC characters or an EBCDIC hyphen (X'60'), padded on the right with EBCDIC spaces X'40'.

Note 2: DS1BLKL includes the 4-byte BDW for variable blocked records and the 4-byte RDW for variable unblocked records.

Note 3: DS1LRECL is a signed value. For variable application records the maximum application data record length includes the 4-byte RDW. The value contained in this field for variable spanned records:

- whose maximum record length (including the 4-byte RDW) is less than 32,757, is the length (a positive signed binary number).
- whose maximum record length is greater than 32,756, is X'8000' minus the length (a

negative signed binary number).

Note 4: Bytes 98-102 are redefined for FBA devices, as opposed to the usage for CKD devices. CKD devices define fields DS1LSTAR and DS1TRBAL in these five bytes.

Note 5: Bytes 103 and 104 are redefined for FBA devices, as opposed to the usage for CKD devices. CKD devices define a one byte reserved field and field DS1TTTHI for these two bytes. For partitioned organization, this field indicates the data set relative sector at which member data begins. The value in this field minus 1, is the ending data set relative sector of the directory area.

DS1DSIND Indicators:

Flags	Name	Description
1...	DS1IND80	Last volume on which this file resides.
.xxx xxxx	reserved	Must be zeros

DS1DSORG Byte 1 Flags:

Flags	Name	Description
0100 000x	DS1DSGPS	Physical sequential organization
0010 000x	DS1DSGDA	Direct access organization
0000 001x	DS1DSGPO	Partitioned organization
...x xx..		Reserved must be zeros
0000 000x	DS1DSGUN	Undefined organization
....1	DS1DSGU	Unmovable, contains location sensitive information

DS1DSORG Byte 2 Flags:

Flags	Name	Description
000x 10xx	DS1ORGAM	Virtual Storage Access Method (VSAM)
...x ..xx	reserved	Must be zeros

DS1OPTCD values with DS1DSGDA (direct organization):

Flags	Description
x...	Whether

Flags	Description
	• 0 – SEOF control interval or • 1 – RDF flag bit 0 used for end-of-file detection.
.... 1...	Physical sector addressing. Note 1.
....1	Extent relative sector addressing. Note 1.
.xxx .xx.	Reserved, must be zeros.

Note 1: Bits 4 and 7 being set to 1 are mutually exclusive.

DS1OPTCD values with DS1DSGPO (partitioned organization):

Flags	Description
x...	Whether • 0 – SEOF control interval or • 1 – RDF flag bit 0 used for end-of-file detection.
....1	Extent relative sector addressing, always 1
.xxx xxx.	Reserved, must be zeros.

DS1OPTCD values with DS1DSGPS (sequential organization):

Flags	Description
x...	Whether • 0 – SEOF control interval or • 1 – RDF flag bit 0 used for end-of-file detection.
....1	Extent relative sector addressing, always 1
.xxx xxxx	Reserved, must be zeros.

DS1RECFM Flags:

Flags	Name	Description
10..	DS1RECFF	Fixed length records
01..	DS1RECFV	Variable length records
11..	DS1RECFU	Undefined length

Flags	Name	Description
..x. ...x	reserved	Must be zero
...1	DS1RECFB	Blocked, can not occur with undefined length
.... 1...	DS1RECFS	Fixed length; standard control intervals; no truncated control intervals except possible last control interval. Variable length spanned records
.... .00.		No control character.
.... .10.	DS1RECFA	IS0/ANSI control character.
.... .01.	DS1RECMC	Machine control character.
.... .11.	reserved	

Data Set Control Block Format 3 – Additional Extent Descriptions

The DSCB-3 record identifies additional extents beyond the three provided by DSCB-1 record and allocated on the current volume.

The DSCB-3 record. if it exists for a data set, is located by means of the data set's DSCB-1 DS1PTRDS field.

Offset Dec	Offset Hex	Length	Field Type	Name	Description
+0	+0	4	Hex	DS3KEYID	Always X'03030303'
+4	+4	40	extent	DS3EXTNT	Four extent description, 10 bytes each.
+44	+2C	1	EBCDIC	DS3FMTID	Format identifier, EBCDIC 3 (X'F3') or symbol DS3IDC.
+45	+2D	90	extent	DS3ADEXT	Nine additional extent descriptions, 10 bytes each.
+135	+87	5	SSSSR	DS3PTRDS	Slot address of the next DSCB-3 record

Data Set Control Block Format 4 – VTOC Data Set Description

The DSCB-4 record resides in the first slot of the first control interval of the VTOC. It is located using the information in the Volume Label Record residing in reserved sector 1.

The RDF associated with a slot containing a Format 4 DSCB must use a flag of X'00' and length of 140.

Offset Dec	Offset Hex	Length	Type	Name	Description
+0	+0	44	Hex	DS4KEY	Key, all bytes set to X'04'

Offset Dec	Offset Hex	Length	Type	Name	Description
+44	+2C	1	EBCDIC	DS4IDFMT	Format identifier, EBCDIC 4 (X'F4') or DS4IDC
+45	+2D	5	SSSSR	DS4HPCHR	Slot address of the last DSCB Format 1 record.
+50	+32	2	unsigned	DS4DSREC	Number of label records following initialization
+52	+34	4	unsigned	DS4HCCHH	Next available alternate track. For FBA device, 0
+56	+38	2	unsigned	DS4NOATK	Number of alternate tracks. For FBA device, 0
+58	+3A	1	flags	DS4VTOCI	VTOC indicators. See DS4VTOCI Flags below
+59	+3B	1	unsigned	DS4NOEXT	Number of VTOC extents. Always 1.
+60	+3C	2	reserved		EBCDIC spaces X'40'
+62	+3E	14	reserved		Device Constants, see below.
+76	+4C	19	reserved		VSAM Indicators. Zeros if no VSAM storage.
+95	+5F	10	reserved		Zeros
+105	+69	10	extent	DS4VTOCE	VTOC Data area extent definition.
+115	+73	25	reserved		Zeros.

DS4VTOCI Flags:

Flags	Description
1...	DSCB Type 5 record validity. See Note 1. • 0 – DSCB Type 5 records valid • 1 – DSCB Type 5 records invalid.
...x	Volume use: • 0 – volume managed by system software • 1 – volume reserved for use by emulator programs
.... .x..	DASD Storage Management • 0 – VSAM data sets are not present on the volume • 1 – VSAM data sets are present on the volume
....x	DSCB Type 10 record validity. See Note 2. • 0 – DSCB Type 10 records valid • 1 – DSCB Type 10 records invalid.

Note 1: Format 5 DSCB records are explicitly tied to CKD architecture devices. This bit must always be set to 1.
Note 2: Tracking of free space on an FBA device requires a record unique to FBA devices, the DSCB-10 defined in this document. Such records would be non-standard and would require support by any space management application managing FBA device space. No implementation of space management of free space on a FBA device by means of a VTOC exists. There has never been a need to define free space records for FBA devices. For compatibility the bit selected to indicate DSCB-10 usage is unused by DOS/VSE.

CKD Device Constants

Values used within the device constants area of the DSCB-4 record for a CKD device.

Offset Dec	Offset Hex	Length	Type	Name	Description
+62	+3E	4	Binary	DS4DEVSZ	Size of the device: • Bytes 0,1 – Number of logical cylinders • Bytes 2,3 – Number of tracks per logical cylinder
+66	+42	2	unsigned	DS4DEVTK	Device track length.
+68	+44	2	unsigned	DS4DEVOV	If bit 4 of DS4DEVFG is 1, the binary count of the number of bytes (overhead bytes) occupied by the count field, gaps, and check bytes of a keyed record
+68	+44	1	unsigned	DS4DEVI	If bit 4 of DS4DEVFG is 0, the count of the number of bytes (overhead bytes) occupied by the count field, gaps, and check bits of a keyed record that **is not** the last record on a track.
+69	+45	1	unsigned	DS4DEVL	If bit 4 of DS4DEVFG is 0, the count of the number of bytes (overhead bytes) occupied by the count field, gaps, and check bits of a keyed record that **is** the last record on a track.
+70	+46	1	unsigned	DS4DEVK	The number of overhead bytes to be subtracted from DS4DEVI, DS4DEVL, or DS4DEVOV if the block has no key field.
+71	+47	1	bit string	DS4DEVFG	Device indicator. `xxxx ....` reserved bits, zeros `.... 1...` DS4DEVI is used as a two-byte field (2301) `.... .00.` CCHH is two halfwords (2311) `.... .1..` CCHH is a two-byte continuous value (2321) `.... ..1.` CCHH is four separate values. `.... ...1` Tolerance required on all but last record of the track.
+72	+48	2	unsigned	DS4DEVTL	Value which when divided by 512 is used to determine effective length of a block on a track.
+74	+4A	1	unsigned	DS4DEVDT	Number of full DSCB's that can be contained on one track (44 byte key plus 96 byte data length).
+75	+4B	1	unsigned	DS4DEVDB	Number of full PDS directory blocks that can be contained on one track (8 byte key plus 256 byte data length).

FBA Device Constants

Values used within the device constants area of the DSCB-4 record for a FBA device.

Offset Dec	Offset Hex	Length	Type	Name	Description
+62	+3E	4	Binary	DS4DEVSZ	Number of available sectors.

Offset Dec	Offset Hex	Length	Type	Name	Description
+66	+42	2	unsigned	DS4DEVTK	Not used for FBA device. Must be zero.
+68	+44	2	unsigned	DS4DEVOV	Not used for FBA device. Must be zero.
+68	+44	1	unsigned	DS4DEVI	Not used for FBA device. Must be zero.
+69	+45	1	unsigned	DS4DEVL	Not used for FBA device. Must be zero.
+70	+46	1	unsigned	DS4DEVK	Not used for FBA device. Must be zero.
+71	+47	1	bit string	DS4DEVFG	Device indicator: `0... ....` VTOC uses SEOF for end-of-file indicator `1... ....` VTOC uses RDF flag for end-of-file indicator.
+72	+48	2	unsigned	DS4DEVTL	Not used for FBA device. Must be zero.
+74	+4A	1	unsigned	DS4DEVDT	Number of full DSCB's that can be contained in one VTOC control interval (44 byte key plus 96 byte data length).
+75	+4B	1	unsigned	DS4DEVDB	Not used for FBA device. Must be zero.

Data Set Control Block Format 10 – FBA Free Space Description

Tracking of free space on an FBA device requires a record unique to FBA devices, the DSCB-5 is strictly for use with CKD devices. The format defined here, the DSCB-10, adapts the DSCB-5 for FBA devices, and is defined only in this document.

DSCB-10 describes space on the FBA volume that has not been allocated to a data set (available), and is optional. If an implementation of space management wishes to track unallocated space on the FBA volume, DSCB-10 is suggested.

Use of DSCB-10 records would be non-standard and would require support by any space management application managing FBA device space. No implementation of space management of free space on a FBA device by means of a VTOC exists. VTOC's are only used by the DOS and OS families of operating systems. While the OS family uses DSCB-5 records to manage CKD volume free space, FBA devices are not supported. The DOS family does not manage free space so no need existed for managing free space on FBA. So, there has never been a need to define free space records for FBA devices. Hence the need to create a new record for FBA devices. The following description adapts the DSCB-5 CKD free space extent description for use with FBA devices. The CKD free space extent description manages cylinders and tracks. The FBA free space extent description manages physical sectors.

If used, the first DSCB-10 will be the second logical record in the second slot of the first VTOC control interval, immediately following the DSCB-4 record in the first slot.

Offset Dec	Offset Hex	Length	Field Type	Name	Description
+0	+0	4	Hex	DSAKEYID	Always X'10101010'
+4	+4	8	available	DSAAVEXT	available extent description
+12	+C	32	available	DSAEXTAV	Four available extents, 8 bytes each
+44	+2C	1	EBCDIC	DSAFMTID	Format identifier, EBCDIC A (X'C1') or symbol DSAIDC.
+45	+2D	88	available	DSAADEXT	Eleven additional available extent descriptions, 8 bytes each.
+133	+85	2	reserved		Must be zeros
+135	+87	5	SSSSR	DSAPTRDS	Slot address of the next DSCB-10 record

FBA Available Extent Format

Each available extent description is 8 bytes in length. An extent with a DSANUM field of zero is not in use and its DSASEC value must be ignored.

Offset Dec	Offset Hex	Length	Field Type	Name	Description
+0	+0	4	unsigned	DSASEC	The physical sector number of the first available sector.
+4	+4	4	unsigned	DSANUM	Number of available sectors in the extent including the first.

It is unpredictable how a volume using DSCB-10 would operate with a DOS/VSE or later operating system. Such volumes should only be used with a system prepared to use them.

Adapting Partitioned Data Sets to FBA

Partitioned Data Sets were designed for residency on Count-Key-Data (CKD) architecture DASD. This appendix describes the CKD structures used and how this document adapts the partitioned data set concept to FBA DASD. The first section, “CKD Partitioned Data Sets” describes the structure on CKD DASD. The second section, “FBA Partitioned Data Sets” builds on this information for use of partitioned data sets on FBA DASD.

CKD Partitioned Data Sets

A partitioned data set is a single data set described by a VTOC DSCB Format 1 record, but physically containing multiple data sets of sequentially organized with fixed length records, usually blocked. Each individual sequential data set is called a *member* and is referenced by a *member name*. The sequential records belonging to the member is called *member data*. Member data is sequentially added to the partitioned data set. All of the member data resides in the partitioned data set's *member data area*.

Directory Area

The first sequential data set of the partitioned data set is reserved for keeping track of the name's of the members and where within the partitioned data set each member's data resides in the member data area. This special initial sequential file is the partitioned data set's *directory area*.

Because the directory area and each member's data is physically organized as sequential data sets, the last record in each case is a CKD End-Of-File (EOF) record, a record with key and data area lengths of zero.

The size of the directory area is determined when the partitioned data set is initially created. The number of requested directory blocks are written to the directory area. Following the last physical record of the directory area is the CKD EOF record. Following the initial record creation, the records are updated with new content. Nowhere is the size of the directory area preserved in the VTOC for later reference. So, its size must be fixed from the beginning.

Each physical record, a *directory block*, within the directory area has a count area and key length of 8 bytes plus a data length of 256 bytes, for a total of 272 bytes. While the directory block has a fixed structure, the amount of data within it, the directory entries, vary in number and size. The first two bytes of the CKD record's data area contains an unsigned value of how many bytes of the fixed block are in use. The value includes the first two bytes.

This table illustrates the structure of the CKD directory block as seen by a program:

CKD Directory Block Record (272 Bytes)				
Count Area	**Key Area**	**Data Area**		
8 bytes	8 bytes	256 Bytes		
Record ID and area lengths	Last Member Name or all X'FF'	2 bytes bytes used	used - 2 Directory Entries	256 - used Available

Each directory entry is a minimum of 12 and a maximum of 74 bytes in length. The variation depends on the usage and length of the optional directory entry user data, having a maximum length of 62 bytes. Directory entries for the entire partitioned data set are maintained in collating sequence order based upon the EBCDIC name of the member to which the entry corresponds. The directory block's key area contains the name of the last directory entry in the directory block, the highest in collating sequence. A program can determine if a member's directory entry **may** be in a block by examining the key. If the desired member, in collating sequence, is higher than the key, the block does not contain the member's entry. Only blocks with a key lower or equal need be searched for the member's name. When a directory entry for a member whose name is higher than the desired member is encountered, the search can cease because the member is not present in the partitioned data set.

The last directory entry is an entry for a member with a name field containing eight X'FF' bytes. It is the entry associated with the start of the unused area of the partition data set's member data area. In other words, it points to where the next member is written. The directory block with this key is the last directory block in the directory area in use. Additional unused directory blocks may follow this last directory block. The content of directory blocks following the last used directory block is unpredictable. Searching for a member within the directory must end with the directory block with a key of all X'FF' bytes.

Each directory entry has a common format for the first twelve (and required) bytes.

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	8	EBCDIC	Member name padded on the right with EBCDIC spaces (X'40') or eight X'FF' bytes if the last directory entry of the directory.
+8	+8	3	TTR	Relative track and record number of the start of this member's data. Note 1.
+11	+B	1	bit string	x . . . . . If 1, member name is an alias.

Offset Dec	Offset Hex	Length	Type	Description
				.xx. .... Number of pointers at start of user data ...x xxxx Number of half words of user data.
+12	+C	0-62	optional	User data. Bits 3-7 of byte 11 determines the length, including any optional pointers. A maximum of three pointers may be placed at the start of the user data as specified by bits 1 and 2 of byte 11.

Note 1: Because the start of member data points to a physical CKD record, each new member starts with a new member data block.

A pointer in the user data takes this format.

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	2	binary	Relative track within the data set of the data to which this pointer points
+2	+2	1	binary	Relative record number on the relative track to which this pointer points
+3	+3	1	binary	Number of additional pointers contained in the note list pointed to by this pointer. Zero if the pointer does not point to a note list. See Note 1.

Note 1: A note list consists of additional pointers to member data blocks within the same member of a partition. You can divide a member into subgroups and store a pointer to the beginning of each subgroup in the note list. The member may be a load module containing many control sections, each control section being a subgroup pointed to by the note list. The note list pointer has the same format as the pointer used in the user data. The last byte of the pointer is available for program use.

Member Data Area

Following the directory area is the member data area into which each member's content is written. Member data records are written to the area as sequentially organized records. The record length and block length used are the values identified for the partitioned data set in its DSCB-1 VTOC record. Following a member's data is a CKD EOF record identifying the end of the member's data. The next member's data immediately follows its predecessor's CKD EOF record, or, in the case of the first member, the CDK EOF record of the directory area

provided there is sufficient space.

VTOC Support

DSCB Format 4 provides one field for support of partitioned data sets: DS4DEVDB, the number of directory blocks per track.

DSCB Format 1 uses five fields for a partitioned data set:

- DS1DSORG byte 1 set to DS1DSGPO, X'02, indicating partitioned data set organization and the data set is movable.
- DS1DSORG byte 2 set to zero.
- DS1RECFM set to DS1RECFF, X'80', with optionally DS1RECFB, X'10' indicating fixed block records, or combined, X'90'.
- DS1BLKL set to the length of each member block size, and
- DS1LRECL set to the length of each member record length.

For fixed unblocked records, DS1BLKL and DS1LRECL are the same value. For fixed block records, DS1BLKL is a multiple of DS1LRECL.

FBA Partitioned Data Sets

DOS/VSE provided an implementation of system libraries on FBA DASD. Such libraries resemble partitioned data sets and act very similar to them. The structures used for such libraries are documented in [7, 231-238]. Ref 7 describes how the various programs were structured for library access, but the actual interface used was never intended for end user usage. This adaptation of partitioned data sets to FBA devices has some elements in common with DOS/VSE libraries, but is dissimilar. The primary difference is the use of control intervals which were not used by DOS/VSE.

The design of partitioned data sets specifically leveraged the CKD device architecture. This description of partitioned data sets on FBA DASD, is also informed by and influenced by how FBA DASD devices operate. To the extent that CKD architecture and FBA differ, differences will exist for partitioned data sets. While the structures described here are similar they will not always be the same.

In either architecture, CKD or FBA, a partitioned data set will contain two areas: a directory area and, following it, the member data area. The directory will still allow locating of member data within the partitioned data set. The member data area will continue to contain the content of each of the directory's members.

Differences Between CKD and FBA DASD

The fundamental difference between CKD and FBA devices is their appearance to and operation with a program. These differences are seen:

- when addressing physical records,
- searching for a physical record,
- detecting an end-of-file condition, and
- storing of data.

CKD physical records must be addressed by their global location on the volume as expressed by the physical cylinder and track addresses and record number on the track. This requires the program to convert logical locations, such as a data set's relative track to the corresponding cylinder and track locations. An FBA physical record (sector) can be directly accessed by a program using a sector number relative to a data set's extent without software conversion. In fact, FBA DASD are designed to operate using relative track numbers. Accessing a FBA sector by physical sector takes more program work than it does to use relative sector addressing.

CKD physical records can be located by means of single or multi-track hardware driven search based upon a record's key area content. FBA physical records (sectors), lacking keys, require program inspection of the sector's content for each sector being searched.

CKD devices can alert a program to the end-of-file condition as a hardware event. FBA devices rely upon software to recognize an end-of-file condition.

CKD devices transfer data of varying sizes constrained only by the capacity of a track. FBA devices transfer physical records in fixed sizes constrained only by the amount of data transferable by a single input/output operation. FBA devices must utilize content conventions for storing of variable length data with appropriate software support for them. The control interval convention serves this purpose as well as providing the mechanism for software detection of the end-of-file condition.

Whereas on a CKD DASD device, each directory block and one or more records of a member's content are stored in physical records, on a FBA DASD, they will all be stored in control intervals:

- one control interval per directory block,
- and one or more control intervals for a member's data.

Maximizing the storage capacity of the partitioned data set drives the use of the RDF end-of-

file flag rather than use of a SEOF (control interval with a CIDF of all zeros) control interval to indicate where the data ends. Programs accessing the partitioned data set must understand how to do that. This usage applies to both the directory and member data areas.

Directory Area

The directory area at the beginning of the partitioned data set contains one directory block within each control interval. A control interval is stored for each directory block allocated to the partitioned data set. The following table describes the structure of each directory control interval on a FBA device. The directory entry area of the directory block is extended from 254 bytes (plus the two byte used length) to 503 bytes (plus the two byte used length).

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	8	EBCDIC	Highest member in collating sequence in this directory block. The last directory block sets this field's bits to one, or eight X'FF'.
+8	+8	2	unsigned	Length of variable area of this directory block including this field, between 2 and 497.
+10	+A	495	variable	Variable number of directory entries of varying lengths.
+505	+1F9	3	RDF	Record definition field for the single logical data record containing the used directory block. The length of the RDF is always 505. RDF Flags: • X'00' – for all control intervals except the last • X'80' – for the last directory control interval containing valid directory entries, provided the RDF end-of-file indicator is in use in the DSCB-1 VTOC record. The RDF field is not present in a directory control interval lacking a used directory block.
+508	+1FC	4	CIDF	Control interval definition field. For control intervals containing a used directory block: • Bytes 0,1 – 505, the free space offset • Bytes 2,3 – 0, the length of the free space For unused directory control intervals beyond the last used: • Bytes 0, 1 – 0, the start of free space offset • Bytes 2, 3 – 508, the length of the free space.

This diagram illustrates the structure of a FBA directory area control interval.

FBA Directory Area Control Interval (512 bytes)					
Directory Block (505 bytes)				RDF	CIDF
Highest Member	Bytes Used (2 - 497)	Directory Entries	available		
8 bytes	2 bytes	used - 2 bytes	497 – used bytes	3 bytes	4 bytes

A directory entry has the same structure on an FBA device, but some fields are interpreted differently.

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	8	EBCDIC	Member name padded on the right with EBCDIC spaces (X'40') or eight X'FF' bytes if the last directory entry of the directory.
+8	+8	3	unsigned	Logical sector number of the first sector of this member's first member data control interval. Note 1.
+11	+B	1	bit string	x... .... If 1, member name is an alias. .xx. .... Number of pointers at start of user data ...x xxxx Number of half words of user data.
+12	+C	0-62	optional	User data. Bits 3-7 of byte 11 determines the length, including any optional pointers. A maximum of three pointers may be placed at the start of the user data as specified by bits 1 and 2 of byte 11.

Note 1: This value supports a partitioned data set of 8 gigabytes in length, more than nine times the largest FBA disk capacity.

A pointer in the user data takes this format on FBA devices.

Offset Dec	Offset Hex	Length	Type	Description
+0	+0	3	unsigned	Logical sector number of the first sector of this member data control interval within the data set to which this pointer points.
+3	+3	1	binary	Number of additional pointers contained in the note list pointed to by this pointer. Zero if the pointer does not point to a note list. See Note 1.

Note 1: A note list consists of additional pointers to member data blocks within the same member of a partition. You can divide a member into subgroups and store a pointer to the beginning of each subgroup in the note list. The member may be a load module containing many control sections, each control section being a subgroup pointed to by the note list. The note list pointer has the same format as the pointer used in the user data. The last byte of the pointer is available for program use.

Member Data Area

On a FBA device, the member data reside in one or more control intervals as defined in the DSCB-1 VTOC record for the partitioned data set. The data itself will be either blocked or unblocked, all application data records being of the same length. If the data is blocked, each RDF within the control interval relates to a single block or number of blocks within the control interval when the RDF is paired. If the data is unblocked, each RDF is associated with an individual logical data record or number of logical data records when the RDF is paired.

Blocked vs. Unblocked Data

For CKD devices the disk capacity is better utilized and performance is improved by reducing the number of device accesses when data is blocked. For FBA, everything is already blocked by the use of control intervals. So whether the member data control intervals contain individual member data records, unblocked, or blocked makes little difference.

It really becomes a matter of which portion of the program identifies how to “unblock/block” the logical data records, the portion that reads or constructs the control interval, or the portion that reads or constructs the block. With blocked logical data records, both portions of the program are required. First, each logical data record is added to the block under construction. When completed, the block then has to be added to the control interval under construction. Once the control interval is completed, it is then written to the device.

When reading the blocked data, the reverse occurs. The control interval is read from the device. Each block is sequentially provided from the control interval. Once the program can access the block, it must then unblock each logical data record from the block. One whole step in this process is eliminated with unblocked data.

Member Data Control Interval

Member data records are stored in a member data control interval. The size of the control interval is taken from the DSCB-1 DS1CISIZ field. Member data records are sequentially organized. Each member data record is placed sequentially within the control block. The control interval's RDF fields define the length and position if the logical data records.

For an unblocked partition data set, the RDF defines the length position of the member data record with which it is associated. The length of each member data record is specified by the DSCB-1 DS1LRECL field and the DSCB-1 DS1BLKL field contains the same value. If there are multiple member data records within the control interval (the typical case), the RDF will be paired with one to its left indicating the number of member data records.

For a blocked partition data set, the RDF defines the starting position, relative to the start of the control interval, of the first member data's block within the control interval. The length of the block is determined by the DSCB-1 DS1BLKL field. Each block contains one or more member data records. The member data records length are specified by the DSCB-1 DS1LRECL field. The last member data block of the member may not contain a full block's worth of data. In this case, it will have an RDF reflecting its actual length.

Notes

http://www.ibm.com/support/knowledgecenter/SSLTBW_2.2.0/com.ibm.zos.v2r2.ickug00/fvtoc.htm

This appendix describes the fixed block architecture VTOC (FBAVTOC), shows examples of label record format, and lists the FBAVTOC space requirements.

VTOCs on FBA devices are formatted in a similar manner to VSAM relative record data sets. The 140-byte label records are stored in control intervals. For example, three label records fit into a 512-byte control interval. The control interval size is always an integral multiple of the device's physical block size. The upper limit is 8192 bytes.

The FBAVTOC extent is expressed (to ICKDSF) in number of slots, which is equivalent to the number of file label records. It can range from a minimum of three slots to a maximum of 999 slots, fitting into as many control intervals as required. The program rounds up the extent values to that of the next full control interval.

The FBAVTOC's starting location can be any physical block on the volume, except for blocks 0 and 1, which are reserved for the IPL and volume label blocks (VLB).

A default (non-SYSRES) FBAVTOC has the following default characteristics:

Starting location
: Block 2, immediately after the VLB

Extent
: 56 file label records (8 control intervals)

Control interval size
: 1024 bytes (or FBA block size, if larger)

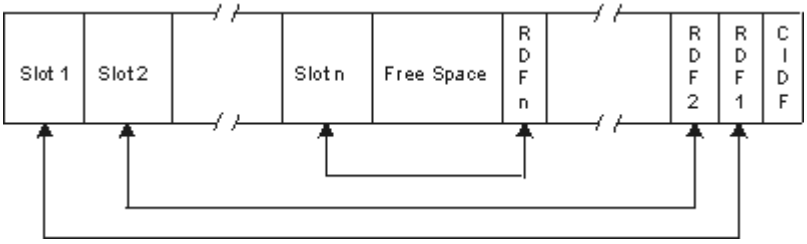
If you specify FBAVTOC(END), ICKDSF creates a VTOC that is 99 entries long with a control interval size of 1024. ICKDSF places the VTOC on the last blocks of the volume.

Each record of the FBAVTOC contains 140 bytes, written as binary zeros. The first two records of the FBAVTOC are reserved for specific records:

- The data set control block of the FBAVTOC (Format-4 DSCB)
- The space management label (Format-5 DSCB)

Each control interval of the FBAVTOC contains a fixed number of slots. Figure 1 shows the FBAVTOC format.

Figure 1. Format of a FBAVTOC control interval



There is one record definition field (RDF) associated with each slot in a control interval. The length field in the RDF (bytes 1 and 2) contains the slot length, which is equal to the label record length. Bit 5 of byte 0 of the RDF indicates whether the associated slot contains a record/label (bit 5=0) or not (bit 5=1). The RDF has one of the following contents:

- X'00008C' if the slot contains a label.
- X'04008C' if the slot does not contain a label, that is, it is empty.

Labels per control interval (LBPCI) is recorded in the standard volume label (VOL1) in bytes 29 through 32.

The control interval definition field (CIDF) has the following format:

```
Bytes 0,1   free-space offset = RECSIZE * LBPCI
Bytes 2,3   free-space length = CISIZE-4-LBPCI * (RECSIZE+3)
```

http://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.idad400/forin.htm

From *z/OS DFSMS Using Data Sets* SC23-6855-00

Index records are stored in control intervals the same as data records, except that only one index record is stored in a control interval, and there is no free space between the record and the control information. So, there is only one RDF that contains the flag X'00' and the length of the record (a number equal to the length of the control interval minus 7). The CIDF also contains the length of the record (the displacement from the beginning of the control interval to the control information); its second number is 0 (no free space). The contents of the RDF and CIDF are the same for every used control interval in an index. The control interval after the last-used control interval has a CIDF filled with 0s, and is used to represent the software end-of-file (SEOF).

From Hercules dasdtab.c dasd_build_fba_devid function:

Fields from dasdtab.h used for devid by FBA devices:

Length is 32 bytes.

Bytes	Source
0	X'30' for operation modes
1	X'08' for features
2	Fba->devclass, always X'21'
3	Fba->type
4,5	Fba->size for block (sector) size, always 512
6-9	Fba->hpg blocks per cyclical group
10-13	Fba->hpp blocks per access position (cylinder)
14-17	Devblk->fbanumblk determined from compressed FBA disk attributes or file size / 512
18-21	0 for sectors under fixed heads
22,23	0 for sectors in alternate areas
24,25	0 for blocks is CE+SA areas
16,27	0 cyclic period in ms.
28,29	0 for minimum period to change access positions in ms
30,31	0 for maximum period to change access positions in ms