MVS/Extended Architecture System – Data Administration

Licensed Program

Order Number
GC26-4149-2

Data Facility Product
5665-XA2

Version 2
Release 3.0

| **Third Edition (June 1987)**

| This is a major revision of, and makes obsolete, GC26-4149-1.

| This edition applies to Version 2 Release 3.0 of MVS/Extended Architecture Data Facility Product, Licensed Product 5665-XA2, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Changes" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Preface

This publication provides information for system programmers about MVS/Extended Architecture Data Facility Product, and how to modify and extend the data management capabilities of the operating system.

Organization

This publication contains the following chapters and appendixes:

- Chapter 1, "Managing the Volume Table of Contents (VTOC)" on page 1, defines and discusses the structure of the VTOC and VTOC index, and the use of system macros to read a data set control block (DSCB), rename a data set, delete a data set from the VTOC, or obtain DASD volume free space information.

- Chapter 2, "Executing Your Own Channel Programs (EXCP)" on page 63, defines and discusses the use of the EXCP macro to control the organization of data based on device characteristics with your own channel programs.

- Chapter 3, "Reading from and Writing to Direct Access Devices (XDAP)" on page 99, defines and discusses the use of the XDAP macro to read, verify, and update blocks without using an access method.

- Chapter 4, "Password Protecting Data Sets" on page 111, defines and discusses system password protection and how to create and maintain the PASSWORD data set.

- The information formerly in Chapter 5, "Exit Routines" on page 125 has been moved to *Data Facility Product: Customization.*

- Chapter 6, "System Macro Instructions" on page 127, defines and discusses the system macros used to refer to, validate, and modify system data areas.

- Chapter 7, "Maintaining SYS1.IMAGELIB" on page 203, defines and discusses adding a UCS or FCB image to the system image library, and maintaining the UCS image tables.

- Chapter 8, "JES2 Support for the IBM 1403, 3203 Model 5, and 3211 Printers" on page 225, defines and discusses JES2 support for UCS alias names and the 3211 indexing feature.

- The information formerly in Chapter 9, "CATALOG, SCRATCH, and RENAME Dummy Modules" on page 227 has been moved to *Data Facility Product: Customization.*

- Chapter 10, "Specifying Buffer Numbers for DASD Data Sets" on page 229, defines and discusses the performance considerations when using the BUFNO keyword and subparameter.

- Appendix A, "CVAF VTOC Access Macros" on page 231, defines and discusses the format of the VTOC access macros: CVAFDIR, CVAFDSM, CVAFFILT, CVAFSEQ, and CVAFTST, and their return codes.

- Appendix B, "Examples of VTOC Access Macros" on page 259, defines and discusses examples of using the VTOC access macros in your programs.

- Appendix C, "VTOC Index Error Message and Associated Codes" on page 297, defines and discusses the error message and field codes issued by the Common VTOC Access Facility (CVAF).

- The information formerly in Appendix D, "Example of an OPEN Installation Exit Module" on page 305 has been moved to *Data Facility Product: Customization.*

- Appendix E, "DFP ISMF Services" on page 307, defines and discusses the DFP user services available with ISMF.

Prerequisite Knowledge

In order to use this book efficiently, you should be familiar with the following topics:

- Assembler language

- Standard program linkage conventions

- The utility programs IEHLIST and IEHPROGM

- Data management access methods and macro instructions

Required Publications

You should be familiar with the information presented in the following publications:

- *Assembler H Version 2 Application Programming: Language Reference*, GC26-4037, and *Assembler H Version 2 Application Programming: Guide*, GC26-4036, contain more information on coding in assembler language.

- *MVS/Extended Architecture System Programming Library: Supervisor Services and Macro Instructions*, GC28-1154, contains a description of standard linkage conventions.

- *MVS/Extended Architecture Data Administration: Utilities*, GC26-4150, describes how to use IEHLIST to maintain the VTOC, and IEHPROGM to protect data sets.
- *MVS/Extended Architecture Data Administration Guide*, GC26-4140, and *MVS/Extended Architecture Data Administration: Macro Instruction Reference*, GC26-4141, contain information on using access methods and macro instructions to do input and output.

Specific prerequisite reading is listed at the beginning of some chapters, as it relates to the particular topic.

Related Publications

Within the text, references are made to the publications listed in the table below.

Short Title (as it appears in the text)	**Publication Title**	**Order Number**
Access Method Services Reference	*MVS/Extended Architecture Integrated Catalog Administration: Access Method Services Reference* *MVS/Extended Architecture VSAM Catalog Administration: Access Method Services Reference*	GC26-4135 GC26-4136
Assembler H V2 Application Programming: Guide	*Assembler H Version 2 Application Programming: Guide*	SC26-4036
Assembler H V2 Application Programming: Language Reference	*Assembler H Version 2 Application Programming: Language Reference*	GC26-4037
Checkpoint/Restart User's Guide	*MVS/Extended Architecture Checkpoint/Restart User's Guide*	GC26-4139
Conversion Notebook	*MVS/Extended Architecture Conversion Notebook*	GC28-1143
Data Administration Guide	*MVS/Extended Architecture Data Administration Guide*	GC26-4140

Short Title (as it appears in the text)	Publication Title	Order Number
Data Administration: Macro Instruction Reference	*MVS/Extended Architecture Data Administration: Macro Instruction Reference*	GC26-4141
Data Facility Product: Customization	*MVS/Extended Architecture Data Facility Product: Version 2 Customization*	GC26-4267
Debugging Handbook	*MVS/Extended Architecture Debugging Handbook*, Volumes 1 through 5	LC28-1164[1] LC28-1165 LC28-1166 LC28-1167 LC28-1168
Device Support Facilities User's Guide and Reference	*Device Support Facilities User's Guide and Reference*	GC35-0033
IBM System/370 Principles of Operation	*IBM System/370 Principles of Operation*	GA22-7000
IBM 2821 Control Unit Component Description	*IBM 2821 Control Unit Component Description*	GA24-3312
IBM 3203 Printer Component Description and Operator's Guide	*IBM 3203 Printer Component Description and Operator's Guide*	GA33-1515
IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide	*IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide*	GA24-3543
IBM 3262 Printer Model 5 Product Description	*IBM 3262 Printer Model 5 Product Description*	GA24-3936

Note:

[1] All five volumes may be ordered under one order number, LBOF-1015.

Short Title (as it appears in the text)	Publication Title	Order Number
IBM 3800 Printing Subsystem Programmer's Guide	*IBM 3800 Printing Subsystem Programmer's Guide*	GC26-3846
IBM 4245 Printer Model 1 Component Description and Operator's Guide	*IBM 4245 Printer Model 1 Component Description and Operator's Guide*	GA33-1541
IBM 4248 Printer Description	*IBM 4248 Printer Description*	GA24-3927
Initialization and Tuning	*MVS/Extended Architecture System Programming Library: Initialization and Tuning*	GC28-1149
JCL User's Guide	*MVS/Extended Architecture JCL User's Guide*	GC28-1351
JCL Reference	*MVS/Extended Architecture JCL Reference*	GC28-1352
JES2 Initialization and Tuning	*MVS/Extended Architecture System Programming Library: JES2 Initialization and Tuning*	SC23-0065
JES3 Data Areas	*MVS/Extended Architecture Data Areas (MVS/JES3)*	LYB8-1195
JES3 Initialization and Tuning	*MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning*	SC23-0059
Linkage Editor and Loader User's Guide	*MVS/Extended Architecture Linkage Editor and Loader User's Guide*	GC26-4143
Magnetic Tape Labels and File Structure Administration	*MVS/Extended Architecture Magnetic Tape Labels and File Structure Administration*	GC26-4145
Open/Close/EOV Logic	*MVS/Extended Architecture Open/Close/EOV Logic*	LY26-3966
RACF General Information	*Resource Access Control Facility (RACF): General Information Manual*	GC28-0722

Short Title (as it appears in the text)	**Publication Title**	**Order Number**
Service Aids	*MVS/Extended Architecture System Programming Library: Service Aids*	GC28-1159
Supervisor Services and Macro Instructions	*MVS/Extended Architecture System Programming Library: Supervisor Services and Macro Instructions*	GC28-1154
System Generation	*MVS/Extended Architecture Installation: System Generation*	GC26-4148
System Logic Library	*MVS/Extended Architecture System Logic Library: Volume 8 of 17*, Parts 1 and 2 (IOS)	LY28-1234 (Part 1) LY28-1235 (Part 2)
System Macros and Facilities	*MVS/Extended Architecture System Programming Library: System Macros and Facilities*, Volumes 1 and 2	GC28-1150 GC28-1151
System Messages	*MVS/Extended Architecture Message Library: System Messages*, Volumes 1 and 2	GC28-1376 GC28-1377
TSO Command Language Reference	*MVS Extended Architecture TSO Command Language Reference* (OS/VS2 TSO Command Language Reference, as updated by Supplement SD23-0259 for MVS/XA)	GC28-0646
TSO/E Data Areas	*MVS/Extended Architecture TSO/E Data Areas* (plus Supplement LDB3-0276)	LYB8-1119
Utilities	*MVS/Extended Architecture Data Administration: Utilities*	GC26-4150
VSAM Administration: Macro Instruction Reference	*MVS/Extended Architecture VSAM Administration: Macro Instruction Reference*	GC26-4152

Notational Conventions

A uniform system of notation describes the format of data management macro instructions. This notation is not part of the language; it simply provides a basis for describing the structure of the commands.

The command format illustrations in this book use the following conventions:

- Brackets [ ] indicate an optional parameter.
- Braces { } indicate a choice of entry; unless a default is indicated, you must choose one of the entries.
- Items separated by a vertical bar (|) represent alternative items. No more than one of these items may be selected.
- An ellipsis (...) indicates that multiple entries of the type immediately preceding the ellipsis are allowed.
- Other punctuation (such as parentheses, commas, and spaces) must be entered as shown. A space is indicated by a blank.
- **BOLDFACE** type indicates the exact characters to be entered, except as described in the bullets above. Such items must be entered exactly as illustrated.
- *Lowercase italic* type specifies fields to be supplied by the user.
- **<u>BOLDFACE UNDERSCORED</u>** type indicates a default option. If the parameter is omitted, the underscored value is assumed.
- Parentheses ( ) must enclose subfields if more than one is specified. If only one subfield is specified, you may omit the parentheses.

Address and Register Conventions

The following describes the meaning of each notation used to show how an operand can be coded:

symbol
: The operand can be any valid assembler-language symbol.

(*0*)
: General register 0 can be used as an operand. When used as an operand in a macro instruction, the register must be specified as the decimal digit 0 enclosed in parentheses as shown above.

(*1*)
: General register 1 can be used as an operand. When used as an operand in a macro instruction, the register must be specified as the decimal digit 1 enclosed in parentheses as shown above. When you use register 1, the instruction that loads it is not included in the macro expansion.

(2-12)

The operand specified can be any of the general registers 2 through 12. All registers as operands must be coded in parentheses; for example, if register 3 is coded, it is coded as (3). When one of the registers 2 through 12 is used, it can be coded as a decimal digit, symbol (equated to a decimal digit), or an expression that results in a value of 2 through 12.

RX-Type Address

The operand can be specified as any valid assembler-language RX-type address. The following shows examples of each valid RX-type address:

Name	Operation	Operand
ALPHA1	L	1,39(4,10)
ALPHA2	L	REG1,39(4,TEN)
BETA1	L	2,ZETA(4)
BETA2	L	REG2,ZETA(REG4)
GAMMA1	L	2,ZETA
GAMMA2	L	REG2,ZETA
GAMMA3	L	2,=F'1000'
LAMBDA1	L	3,20(,5)

Both ALPHA instructions specify explicit addresses; REG1 and TEN have been defined as absolute symbols. Both BETA instructions specify implied addresses, and both use index registers. Indexing is omitted from the GAMMA instructions. GAMMA1 and GAMMA2 specify implied addresses. The second operand of GAMMA3 is a literal. LAMBDA1 specifies an explicit address with no indexing.

A-Type Address

The operand can be specified as any address that can be written as a valid assembler-language A-type address constant. An A-type address constant can be written as an absolute value, a relocatable symbol, or relocatable expression. Operands that require an A-type address are inserted into an A-type address constant during the macro expansion process. For more details about A-type address constants, see *Assembler H Version 2 Application Programming: Language Reference.*

absexp

The operand can be an absolute value or expression. An absolute expression can be an absolute term or an arithmetic combination of absolute terms. An absolute term can be a nonrelocatable symbol, a self-defining term, or the length attribute reference. For more details about absolute expressions, see *Assembler H Version 2 Application Programming: Language Reference. OS/VS–DOS/VSE–VM/370 Assembler Language.*

relexp

The operand can be a relocatable symbol or expression. A relocatable symbol or expression is one whose value changes by n if the program where it appears is relocated n bytes away from its originally assigned area of storage. For more details about relocatable symbols and expressions, see *Assembler H Version 2 Application Programming: Language Reference.*

Summary of Changes

Release 3.0, June 1987

Enhancements and New Support

Data sets may now be retained beyond the year 1999 or be retained indefinitely (never-scratch). "Deleting a Data Set from the VTOC (SCRATCH and CAMLST SCRATCH)" on page 33 describes *never-scratch* designations.

Information about the new LSPACE macro instruction has been added. Chapter 1, "Managing the Volume Table of Contents (VTOC)" on page 1, explains how to use the LSPACE macro to determine the amount of free space and the degree of space fragmentation on a direct access volume. The LSPACE macro also allows you to obtain VTOC status information.

Open, Close, and End-of-Volume parameter lists may now reside at an address above 16 megabytes. Chapter 2, "Executing Your Own Channel Programs (EXCP)" on page 63 explains the changes to the EOV macro format and description in support of parameter lists above 16 megabytes.

The retrieval area for information extracted from a JFCB can now be allocated at an address above 16 megabytes. Chapter 6, "System Macro Instructions" on page 127 now includes the following information in support of the retrieval area above 16 megabytes.

- The differences in the RDJFCB and OPEN macro operands and parameter lists.
- The new X'13' exit list entry code.
- The use of the IHAARL macro.

Information related to customizing the Data Facility Product for individual users has been moved to *Data Facility Product: Customization*, a new book. This includes the information in Chapter 2 under "Appendages" on page 71, all of Chapter 5 (including the new information about the conventions that installation-written DADSM preprocessing and postprocessing modules must follow for 31-bit versus 24-bit addressing), all of Chapter 9, and the example formerly in Appendix D.

Service changes have been made throughout the manual, and are indicated in the text by revision bars.

Release 2.0, June 1986

Enhancements and New Support

Information has been added for support of CVAF Filter Services.

- The format of the CVPL in Chapter 1, "Managing the Volume Table of Contents," has been updated.
- Appendix A now contains the CVAFFILT macro syntax and explains the parameters, register contents, and return codes.
- Appendix B now contains an example of a CVAFFILT invocation.

Information has been added for support of DASD Calculation Services.

- Chapter 5, "Exit Routines," describes the use of the DASD Calculation Services precalculation and postcalculation installation exits. It also explains the parameters, register contents, and return codes.

Appendix E has been added to provide interface information for DFP/ISMF services.

Chapter 1, "Managing the Volume Table of Contents," has been reorganized and rewritten.

Chapter 7, "Maintaining SYS1.IMAGELIB," has been reorganized and rewritten.

Service changes have been made throughout the manual, and are indicated in the text by revision bars.

Release 1.0, April 1985

Enhancements and New Support

The SCRATCH and CAMLST SCRATCH macro descriptions have been updated to include support for the erasure of residual DASD data.

Information has been added to the description of the REALLOC macro for DADSM support of DFDSS that:

- Describes the new support and explains what it does.
- Adds new keywords.

- Adds new return codes.

The PARTREL macro has been added to the partial release section that:

- Describes the new support and explains what it does.
- Adds all keywords and descriptions.
- Adds new return codes.

Information supporting the ERASE-ON-SCRATCH option for RACF-defined data sets has been added to "Deleting a Data Set from the VTOC (SCRATCH and CAMLST SCRATCH)" on page 33.

Information has been added to Figure 29 on page 134 to support the:

- 3262 Model 5 Printer.
- 4245 Printer.
- 4248 Printer.

Information has been added to support the IBM 3380 (all models).

The following IBM 3480 Magnetic Tape Subsystem support information has been added:

- The high-speed positioning feature.
- Chapter 6, "System Macro Instructions" on page 127, has been updated.
- The MSGDISP macro has been added to permit loading a message display on the 3480.
- The 3480 has been added to Figure 29 on page 134.

"High-Speed IBM 3480 Positioning" on page 149 in Chapter 6, "System Macro Instructions" on page 127 has been added and describes how to set the tape block ID for the IBM 3480 Magnetic Tape Subsystem in full function mode.

Chapter 1, formerly titled "Controlling Space on DASD Volumes," has been renamed, "Managing the Volume Table of Contents (VTOC)."

Version 2 Publications

The Preface includes new order numbers for Version 2.

Contents

Figures

Chapter 1. Managing the Volume Table of Contents (VTOC)

The direct access device storage management (DADSM) routines control allocation of space on direct access volumes through the volume table of contents (VTOC) of that volume, and through the VTOC index if one exists. This chapter gives an overview of the VTOC and the VTOC index and discusses how to use system macros to access the VTOC and VTOC index.

The VTOC

The VTOC is a data set on a direct access volume that describes the contents of that volume. It resides in a single extent (that is, it is a continuous data set) anywhere on the volume after cylinder 0, track 0. Its address is located in the VOLVTOC field of the standard volume label (see Figure 1 on page 2).

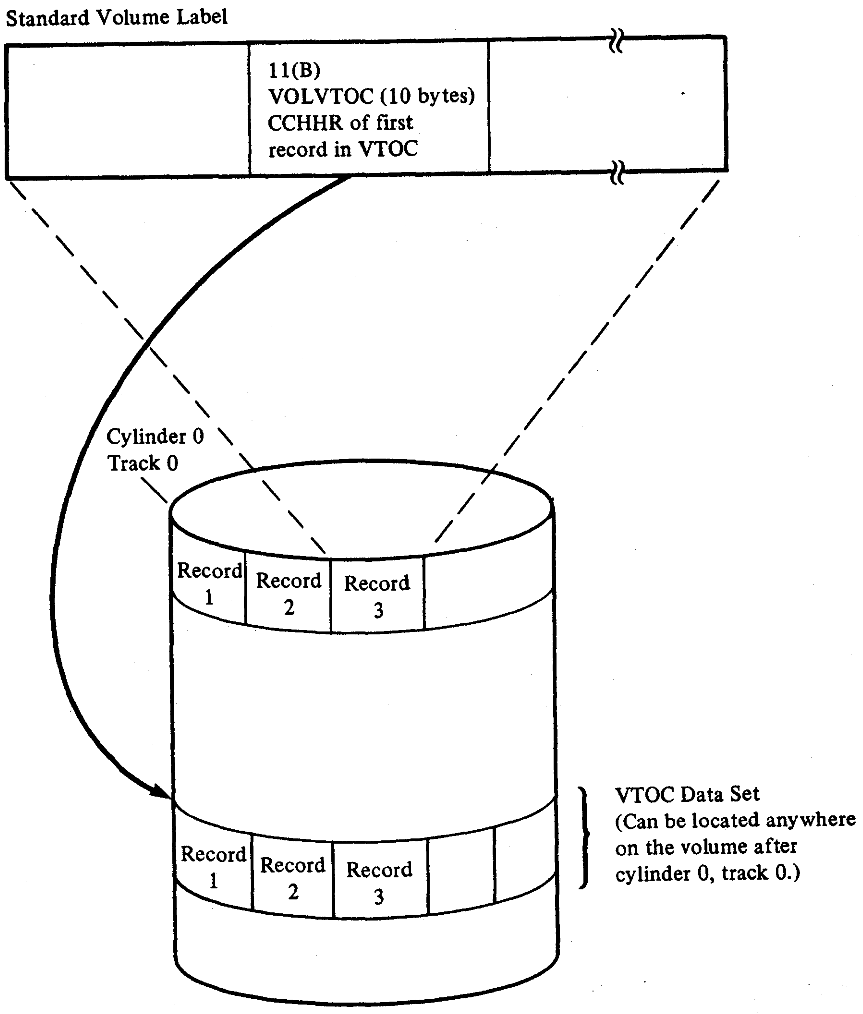


Figure 1. Locating the Volume Table of Contents (VTOC)

The VTOC is composed of 140-byte[1] data set control blocks (DSCBs) that correspond either to a data set or VSAM data space currently residing on the volume, or to contiguous, unassigned tracks on the volume. DSCBs for data sets or data spaces describe their characteristics. DSCBs for contiguous, unassigned tracks indicate their location.

1 The 140 bytes are defined as a 44-byte key portion followed by a 96-byte data portion. You may make references to the logical 140-byte DSCB or to either of its parts.

Data Set Control Block (DSCB) Format Types

The VTOC has seven different kinds of DSCBs. This section lists the different kinds of DSCBs, what they are used for, how many exist on a volume, and how they are found.

The first record in every VTOC is the VTOC (format-4) DSCB that describes (1) the device that the volume resides on, (2) the attributes of the volume itself, and (3) the size and contents of the VTOC data set itself.

The format-4 DSCB is followed by a free-space (format-5) DSCB that, for a nonindexed VTOC, lists the extents on the volume that have not been allocated to a data set or VSAM data space. Each format-5 DSCB contains 26 extents. If there are more than 26 available extents on the volume, another format-5 DSCB will be built for every 26 extents. The format-5 DSCBs are chained, using the last field of each format-5 DSCB. An indexed VTOC does not use format-5 DSCBs for describing free space; however, one empty format-5 DSCB is provided to allow a basis for converting back to a nonindexed VTOC.

The third and subsequent DSCBs in the VTOC have no prescribed sequence.

A data set or VSAM data space is defined by one or more DSCBs in the VTOC of each volume on which it resides. The number of DSCBs needed to define a data set or VSAM data space is determined by (1) the organization of the data set (ISAM data sets need a format-2 DSCB to describe the index) and (2) the number of extents the data set or VSAM data space occupies (a format-3 DSCB is needed to describe the 4th through the 16th extents; additional format-3 DSCBs may be required to describe the extents for a VSAM data set cataloged in an Integrated Catalog Facility catalog). Figure 2 on page 7 shows the general makeup of a VTOC and the DSCBs needed to define two types of data sets (ISAM and non-ISAM).

Data set A (in Figure 2 on page 7) is an ISAM data set; three DSCBs, a format-1, format-2, and format-3, are identified. Data sets B, C, and D could be sequential, partitioned, or direct data sets or they could be VSAM data spaces. Data set B has more than three extents and therefore requires both a format-1 and a format-3 DSCB.

Data sets C and D have three or fewer extents and need only a format-1 DSCB. The format-6 DSCB, pointed to by the format-4 DSCB, is used to keep track of the extents allocated in order to be shared by two or more data sets (split-cylinder data sets). For example, if data sets C and D share an extent made up of one or more cylinders, this extent would be described in the format-6 DSCB. Note that split-cylinder data sets cannot be allocated, but existing split-cylinder data sets can still be processed.

Format-0 DSCB

Name: Free VTOC Record

Function: Describes an unused record in the VTOC (contains 140 bytes of binary zeros). To delete a DSCB from the VTOC, a format-0 DSCB is written over it.

How Many: One for every unused 140-byte record on the VTOC. The DS4DSREC field of the format-4 DSCB is a count of the number of format-0 DSCBs on the VTOC. This field is not maintained for an indexed VTOC.

How Found: Search on key equal to X'00' (sometimes X'00000000') for a nonindexed VTOC; for an indexed VTOC, the VTOC map of DSCBs is used to find a format-0 DSCB.

Format-1 DSCB

Name: Identifier

Function: Describes the first three extents of a data set or VSAM data space.

How Many: One for every data set or data space on the volume, except the VTOC.

How Found: Search on key equal to the data set name. For an indexed VTOC, a CCHHR pointer for each data set name is in the VTOC index.

Format-2 DSCB

Name: Index

Function: Describes the indexes of an ISAM data set.

How Many: One for every ISAM data set (for a multivolume ISAM data set, a format-2 DSCB exists only on the first volume).

How Found: Chained from a format-1 DSCB that represents the data set.

Format-3 DSCB

Name: Extension

Function: Describes the 4th through 16th extents of a data set or VSAM data space. Data sets and VSAM data spaces are restricted to 16 extents per volume. VSAM data sets cataloged in an Integrated Catalog Facility catalog may be extended to a maximum of 123 extents, in which case there may be as many as ten format-3 DSCBs.

How Many: One for each data set or VSAM data space on the volume that has more than three extents. There may be as many as ten for a VSAM data set cataloged in an Integrated Catalog Facility catalog.

How Found: Chained from a format-2 or a format-1 DSCB that represents the data set or VSAM data space. In the case of a VSAM data set cataloged in an Integrated Catalog Facility catalog, the chain may be from a preceding format-3 DSCB.

Format-4 DSCB

Name: VTOC

Function: Describes the extent and contents of the VTOC and provides volume and device characteristics. If the VTOC is indexed, certain fields of this DSCB are not maintained by DADSM. See "Structure of an Indexed VTOC."

How Many: One on each volume.

How Found: VOLVTOC field of the standard volume label contains its address. It is always the first record in the VTOC.

Format-5 DSCB

Name: Free Space

Function: On a nonindexed VTOC, describes the space on a volume that has not been allocated to a data set or to a VSAM data space (available space). For an indexed VTOC, format-5 is zero, and the volume pack space map describes the available space.

How Many: One for every 26 noncontiguous extents of available space on the volume for a nonindexed VTOC; for an indexed VTOC, there is only one.

How Found: The first format-5 DSCB on the volume is always the second DSCB of the VTOC. If there is more than one format-5 DSCB, it will be chained from the previous format-5 DSCB via the DS5PTRDS field of each format-5 DSCB.

Format-6 DSCB

Name: Shared Extent

Function: Describes the extents shared by two or more data sets (split-cylinder extents).

How Many: One for every 26 split-cylinder extents on the VTOC.

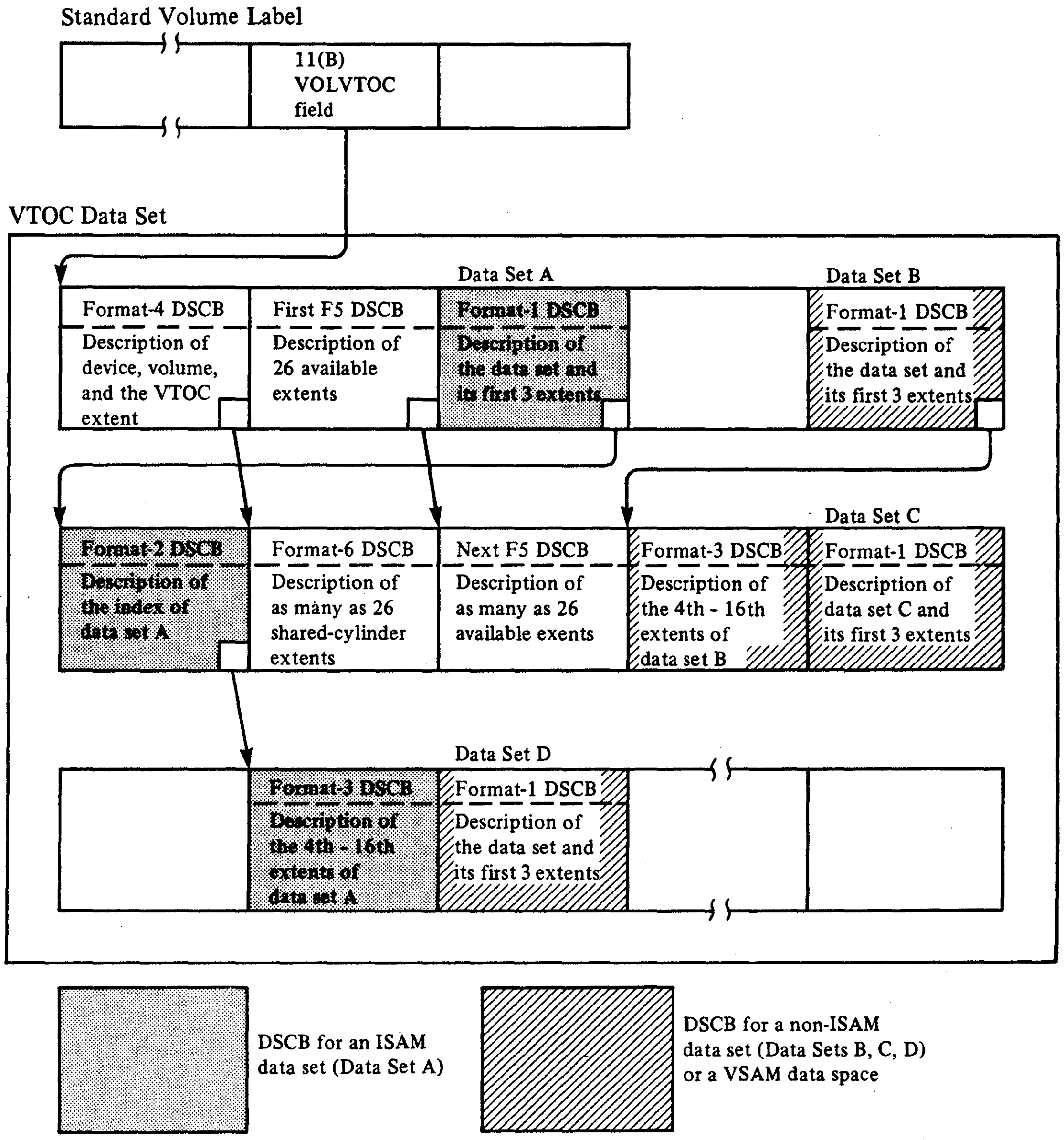
How Found: The address of the first format-6 DSCB is contained in the DS4F6PTR field of the format-4 DSCB. If there is more than one format-6 DSCB on the volume, it will be chained from the previous format-6 DSCB via the DS6PTRDS field of the format-6 DSCB.

Allocating and Releasing DASD Space

The DADSM allocate and extend routines assign tracks and cylinders on direct access volumes for new data sets and VSAM data spaces. The DADSM extend routine obtains additional space for a data set or VSAM data space that has already exceeded its original, primary allocation. The DADSM scratch and partial release routines are used to release space that is no longer needed on a direct access volume.

The DADSM routines allocate and release space by adding, deleting, and modifying the DSCBs. When space is needed on a volume, the allocate routines search the appropriate DSCBs for enough contiguous, available tracks to satisfy the request. If there are not enough contiguous tracks, the request is filled, using as many as five noncontiguous groups of free tracks. The appropriate DSCBs are modified to reflect the assignment of the tracks.

When space is released, the scratch routines free the DSCBs of the deleted data set or data space. For a nonindexed VTOC, to indicate that the tracks containing the affected data set or data space can be reallocated, a free space (format-5) DSCB is built (or modified if existent). For an indexed VTOC, the index is updated.


Standard Volume Label
11(B) VOLVTOC field
VTOC Data Set
Data Set A
Data Set B
Format-4 DSCB
Description of device, volume, and the VTOC extent
First F5 DSCB
Description of 26 available extents
Format-1 DSCB
Description of the data set and its first 3 extents
Format-1 DSCB
Description of the data set and its first 3 extents
Data Set C
Format-2 DSCB
Description of the index of data set A
Format-6 DSCB
Description of as many as 26 shared-cylinder extents
Next F5 DSCB
Description of as many as 26 available exents
Format-3 DSCB
Description of the 4th - 16th extents of data set B
Format-1 DSCB
Description of data set C and its first 3 extents
Data Set D
Format-3 DSCB
Description of the 4th - 16th extents of data set A
Format-1 DSCB
Description of the data set and its first 3 extents
DSCB for an ISAM data set (Data Set A)
DSCB for a non-ISAM data set (Data Sets B, C, D) or a VSAM data space

Note: Empty boxes in the VTOC data set represent free VTOC Records (Format-0 DSCBs)

Figure 2. Contents of VTOC—DSCBs Describing Data Sets

The VTOC Index

The VTOC index is a physical-sequential data set, residing on the same volume as the VTOC. It contains an index of data set names of format-1 DSCBs in the VTOC and free space information. The index is searched instead of the hardware keys.

The VTOC index is optional. You may build it when you initialize the volume, or for an existing VTOC (with the volume online or offline). You may subsequently inactivate it (online or offline) so that the VTOC is processed without using the index.

Each VTOC index is formatted by Device Support Facilities with physical blocks 2048 bytes in length. These physical blocks are the VTOC index records (VIRs), the basic structural units of the index. The kind of information they contain depends on the part of the index they belong to.

Several different kinds of records, each built from one or more VIRs, are in a VTOC index:

- The VTOC index entry record (VIER) that is used to access format-1 DSCBs and the format-4 DSCB
- The VTOC pack space map (VPSM) that shows what space has been allocated on a disk pack
- The VTOC index map (VIXM) that shows which VIRs have been allocated in the VTOC index
- The VTOC map of DSCBs (VMDS) that shows which DSCBs have been allocated in the VTOC

An Example of a VTOC and Its Index

A format-1 DSCB in the VTOC contains the name and extent information of the VTOC index. The name of the index must be 'SYS1.VTOCIX.xxxxxxxx', where 'xxxxxxxx' can be anything valid in a data set name and is generally the serial number of the volume containing the VTOC and its index. The name must be unique within the system to avoid ENQ contention. The relationship of a VTOC to its index is shown in Figure 3 on page 9. Each of the components of the index is discussed separately in the following sections.

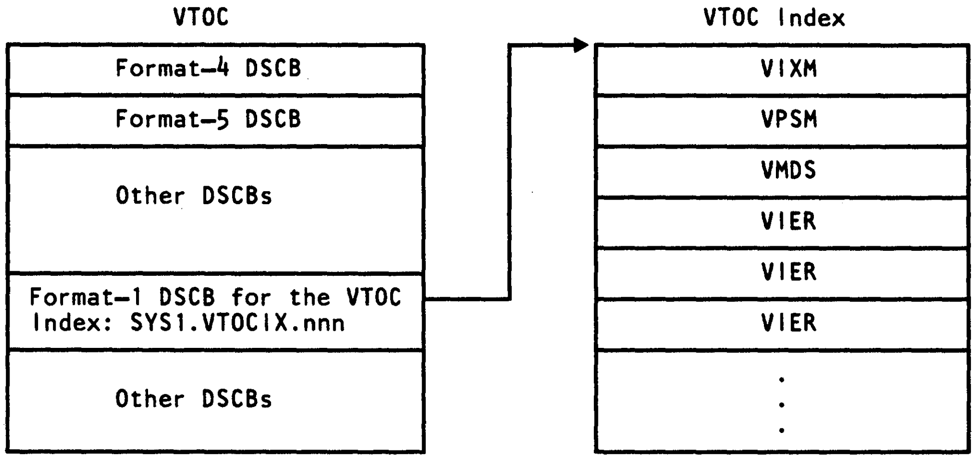


Figure 3. Relationship of a VTOC to Its Index

The VTOC Index Entry Record (VIER)

VIERs have these characteristics:

- A VIER uses one VIR and contains variable-length index entries. The number of VIERs in an index varies depending upon the number of data sets on the volume.
- VIERs in a VTOC index may be on one or many levels. All index entries in a VIER are at the same index level. VIERs have a hierarchic relationship. Index entries in higher-level VIERs point to lower-level VIERs. Index entries in level-one VIERs (those at the lowest level) point to format-1 DSCBs for data sets on the volume.
- A higher-level VIER is created when the fourth lower-level VIER is created. When that new higher-level VIER is filled with pointers to lower-level VIERs, a new VIER at the same level is created. Again, when the fourth VIER at the same level is created, a VIER at a still higher level is created, adding another level to the index.

Contents of VIER Fields

Each VIER contains a header and sections (see Figure 4 on page 10). The VIER header contains:

- A field identifying the VTOC index record as a VIER.
- The relative byte address (RBA) of the VIER.
- A pointer to a VIER at the same level (hence, a "horizontal" pointer). The VIER pointed to contains index entries whose keys are greater than any key in the pointing VIER.

- The level number (LVL) of this VIER.
- The number (SECNO) of sections (a VIER contains eight sections).
- The length (SECL) of the sections (each section is 246 bytes in length).
- The offsets to the first-used and the last-used sections.
- The 44-byte high key of the VIER.

Each section contains:

- An offset to the last entry in the section (or zero if the section is empty)
- Index entries

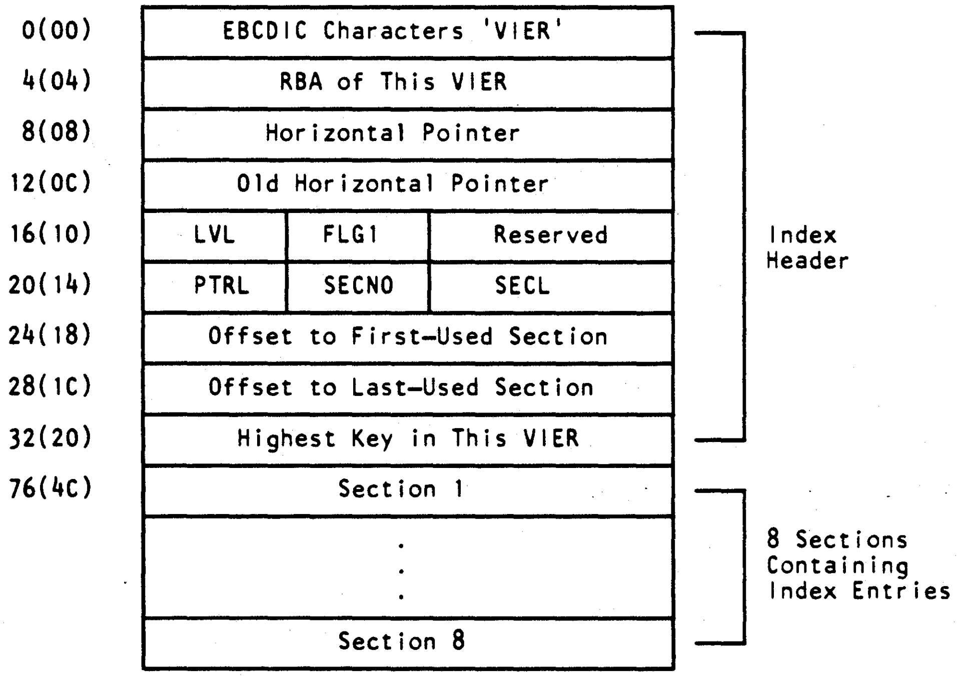


Figure 4. Format of the VTOC Index Entry Record (VIER)

Format of a VTOC Index Entry

The format of an index entry is:

FLG	KEYL	Unused	Record Pointer	Key

NAME	OFFSET	BYTES	DESCRIPTION
VXEFLG	00(00)	1	Flag byte
VXEKEYL	01(01)	1	Length of the VXEKEY field
VXEFC	02(02)	1	Unused
VXERPTR	03(03)	4 or 5	Record pointer
VXEKEY	07(07) or 08(08)	1 to 44	Name of a data set, if a level-one VIER; if not, the high key in the header of a lower-level VIER

Each index entry contains:

- A flag byte.
- A keylength field (containing a value of 1 to 44, depending on the length of the data set name).
- A record pointer (VXERPTR) that is one of the following:
  - In level-one VIERs, the 5-byte CCHHR of the format-1 or format-4 DSCB that represents the data set whose name is the key in the entry
  - In other VIERs, the 4-byte RBA of the lower-level VIER whose high key is the key in the entry
- A key that, for level 1 VIERS, is the data set name, and for level 2 or higher VIERs is the high key of a lower-level VIER. Trailing blanks are suppressed in the VTOC index entry.

When a VIER Is Created

The first level-one VIER is created when the VTOC index is created. Subsequent VIERs are created when a data set name is to be added to the VTOC index but the VIER where it should be added is full. A new VIER is created in the following manner:

- A new VIER is allocated.
- Half of the sections from a full VIER (those containing the highest keys) are moved into the new VIER, leaving each VIER half empty.
- The new index entry is added to one of the two VIERs, depending on its key.

A Tree of Linked VIERs

Figure 5 on page 13 shows how VIERS are related to each other. Note that the VIERs (which are simplified here—only the high key is shown in the header) form a type of "tree structure."

How to Find a Format-1 DSCB

In the search for the format-1 DSCB for a particular data set, one path along the tree structure is followed.

As seen in Figure 4 on page 10, a field in the header of a VIER contains the highest key of any index entry in that VIER. Beginning with this field in the first high-level VIER, the following search logic is used: Is the key of the data set (the data set name) lower than or equal to the VIER's high key? If neither, the test is again applied with the VIER having a greater high key pointed to by the horizontal pointer. This procedure continues until a VIER is found having a high key that is greater than or equal to the key of the data set. Comparisons are then made with the entries in the VIER's sections. Eventually, an entry is found with a key greater than or equal to the data set key. This entry points to a VIER at the next-lower level.

The search proceeds to successively lower levels until an entry in a level-two VIER is found whose key is greater than or equal to the key of the data set. This entry points to a level-one VIER that, in turn, contains an entry with a key that is equal to the data set key and that points to the format-1 DSCB for the desired data set.

Special Cases in a DSCB Search

If there is only one level in the VTOC index, the entries in the VIERs all point to format-1 DSCBs, so that only one level need be searched.

If an update to the VTOC index requires a new VIER and the update is interrupted (for example, because of an I/O error or a system failure), the entry in the level-n VIER may contain a key that is greater than the high key in the lower-level VIER pointed to by that entry. In this case, two VIERs at level n-1 may have to be searched. This situation is corrected when DADSM next processes the volume.

The VTOC Pack Space Map (VPSM)

The VPSM accounts for space on a disk pack. It shows what space on the volume has been allocated and what space remains free.

The map contains bit maps of the cylinders and tracks on the volume. A value of one indicates that the cylinder or track has been allocated; a value of zero, that it has not been allocated. The bit representing a cylinder is set to zero if no tracks on the cylinder have been allocated; it is set to one if any track has been allocated. Tracks assigned as alternate tracks are marked as allocated.

The VPSM replaces the chain of format-5 DSCBs, but one empty format-5 DSCB is left in the VTOC to allow for conversion back to a nonindexed VTOC, a process that requires reconstruction of a format-5 DSCB chain.

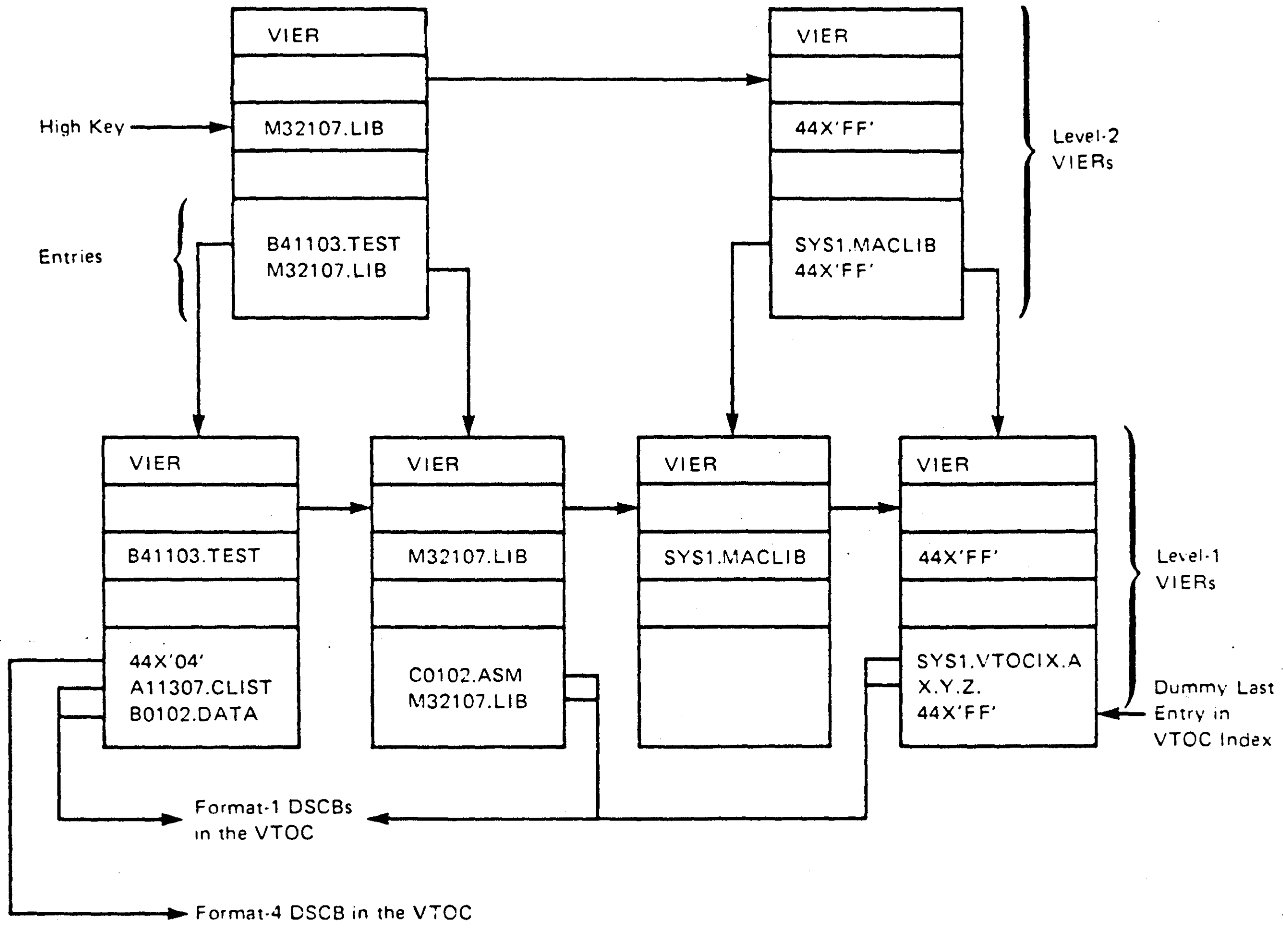


Figure 5. Structure of Linked VIERs

The format of an index map (including the VPSM) is shown in Figure 6 on page 14.

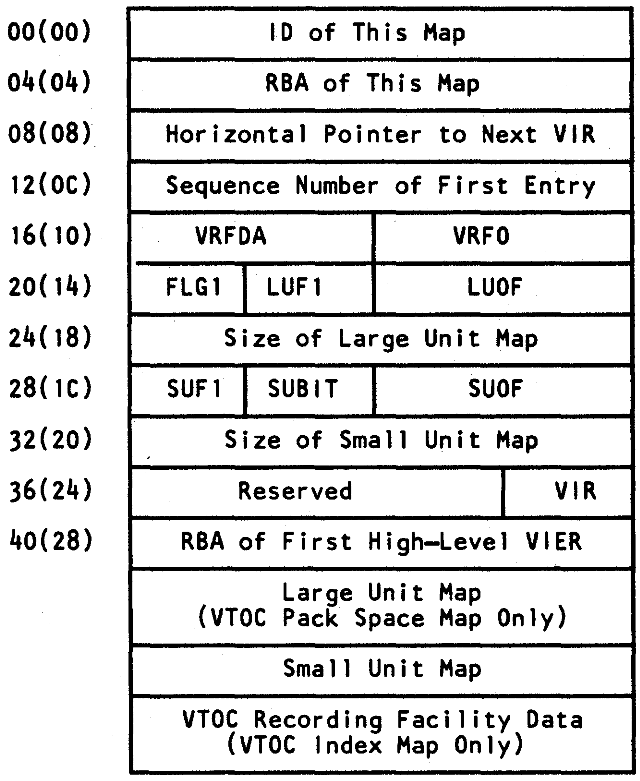


Figure 6. An Index Map

The VTOC Index Map (VIXM)

The VIXM contains a bit map in which each bit represents one VTOC index record (VIR). The status of the bit indicates whether the VIR is allocated (1) or unallocated (0).

An area of the VIXM is reserved for VTOC recording facility (VRF) data. (This is the facility that allows detection of and recovery from certain errors in an indexed VTOC.)

A field in the first VIXM record points to the first high-level VIER. Another field in the first VIXM record (VIR in Figure 7 on page 15) contains the number of VTOC index records that contain all the space maps.

The VTOC Map of DSCBs (VMDS)

The VMDS contains a bit map where each bit represents one DSCB in the VTOC. The status of the bit indicates whether the DSCB is allocated (1) or unallocated (0).

Name	Offset	Bytes	Description
VIMAP	00(X'00')	2048	VTOC map
VIMH	00(X'00')	44	VTOC map header
VIMID	00(X'00')	4	Map ID in EBCDIC ('VPSM', 'VIXM', or 'VMDS')
VIMRBA	04(X'04')	4	RBA of this map
VIMHZPTR	08(X'08')	4	Horizontal RBA pointer to next VIR of this map
VIMORG	12(X'0C')	4	Sequence number of the first entry in the map
VIMVRFDA	16(X'10')	2	Offset to current VRF data (if VIMVRFSW=1) or offset where VRF data may be written (if VIMVRFSW=0), (first VIXM only)
VIMVRFO	18(X'12')	2	Offset to VRF area (first VIXM VIR only)
VIMFLG1	20(X'14')	1	Flag byte
VIMVRFSW		X'80'	VRF data exists if 1
		.xxx xxxx	Reserved
VIMLUF1	21(X'15')	1	Large unit flag byte
VIMLUOF	22(X'16')	2	Offset into VIR of large unit map (zero if none)
VIMLUSZ	24(X'18')	4	Size in bits of large unit map

Figure 7 (Part 1 of 2). Format of a VTOC Map

Name	Offset	Bytes	Description
VIMSUF1	28(X'1C')	1	Small unit flag byte
VIMSUBIT	29(X'1D')	1	Number of small unit bits per large unit (zero if none)
VIMSUOF	30(X'1E')	2	Offset into VIR of small unit map
VIMSUSZ	32(X'20')	4	Size in bits of small unit map
	36(X'24')	3	Reserved
VIMVIR	39(X'27')	1	Number of map records (VIXM only)
VIMFHLV	40(X'28')	4	RBA of first high-level VIER (VIXM only)
VIMLUMAP	44(X'2C')	kk	Large unit map (kk is VIMLUSZ/8, rounded up)
VIMSUMAP	mm	nn	Small unit map (mm is VIMSUOF, nn is VIMSUSZ/8, rounded up)
VIMVRF	pp	qq	VRF area (pp is VIMVRFO, qq is remainder of first VIXM)

Figure 7 (Part 2 of 2). Format of a VTOC Map

Structure of an Indexed VTOC

An indexed VTOC is identical to a nonindexed VTOC, except that, for an indexed VTOC, only a single format-5 DSCB exists and is empty, and certain format-4 DSCB data (the number of format-0 DSCBs and the CCHHR of the highest format-1 DSCB) is not maintained by DADSM. The DOS bit (bit 0 in field DS4VTOCI), set to one in the format-4 DSCB, indicates that these fields (and the format-5 DSCB) cannot be relied on. The index bit (bit 7 in field DS4VTOCI) is set in the format-4 DSCB; it indicates that a VTOC index exists.

Scratch/Rename/Allocate Restrictions

A VTOC index data set may not be scratched if the VTOC index is active. Neither may a VTOC index data set be renamed if the VTOC index is active, unless it is being renamed to another name beginning with 'SYS1.VTOCIX.'. A data set may not be renamed to a name beginning with 'SYS1.VTOCIX.' if there is already such a data set on the volume. Only one data set whose name begins with 'SYS1.VTOCIX.' may be allocated on a volume.

Initializing and Maintaining the VTOC

Creating the VTOC and VTOC Index

To prepare a volume for use (to initialize it), the Device Support Facilities utility is used. One of the things this utility does is to build the VTOC. After initialization, this VTOC will contain a format-4 DSCB and a format-5 DSCB. For a nonindexed VTOC, the format-5 DSCB contains an extent entry for all the free space on the volume; the initial number of extents in the format-5 DSCB is one or two, depending on where the VTOC is located on the volume. If the VTOC is located somewhere other than at the beginning or end of the volume, two extent entries are needed to describe the free space that precedes and follows it. For an indexed VTOC, the format-5 DSCB contains a zero.

A VTOC index can be created when a volume is initialized by using the Device Support Facilities command INIT and specifying the INDEX key word.

A nonindexed VTOC can be converted to an indexed VTOC by using the command BUILDIX and specifying the IXVTOC keyword. The reverse is also possible by using the BUILDIX command and specifying the OSVTOC keyword.

For more detailed information, see *Device Support Facilities User's Guide and Reference.*

Protecting the VTOC and VTOC Index

Resource Access Control Facility (RACF)

You can protect the VTOC and VTOC index by using the Resource Access Control Facility (RACF). This is done by defining the volume serial entity under the RACF class DASDVOL. A user must be authorized to the DASDVOL/volume serial entity at the following levels:

- At the UPDATE level, to open the VTOC for output processing.
- At the UPDATE level, to open for output processing any data set whose name begins with 'SYS1.VTOCIX.'.
- At the ALTER level, to allocate, rename, or scratch any data set whose name begins with 'SYS1.VTOCIX.'.
- At the ALTER level, to rename a data set to any name that begins with 'SYS1.VTOCIX.'.

Neither the VTOC nor the VTOC index is protected from being opened for input processing by the DASDVOL/volume serial entity.

Note that neither the VTOC nor the VTOC index can be protected through the RACF class DATASET.

Authorized Program Facility (APF) Requirements

A program must be authorized by the authorized program facility (APF) to perform any of the following functions:

- Opening a VTOC for output processing
- Opening for output processing a data set whose name begins with 'SYS1.VTOCIX.'
- Allocating, renaming, or scratching any data set whose name begins with 'SYS1.VTOCIX.'
- Renaming a data set to any name that begins with 'SYS1.VTOCIX.'

Password Protection

The VTOC index data set may be password protected. The protection is the same as for any password-protected data set. Password checking is bypassed if the volume in which the VTOC index resides is protected by RACF through the DASDVOL class.

Copying/Restoring/Initializing the VTOC

Operations on Volumes Containing a Nonindexed VTOC

- *Restoring a Volume from a Dump Tape.* There are no operational requirements if you change the volume serial number or do a partial restore that does not modify the VTOC. If you do a restore and change the VTOC size without changing the volume serial number, the volume must be varied offline after it is restored. You should not do a restore on a volume with an indexed VTOC.
- *Copying a Volume.* There are no operational requirements if you change the volume serial number or do not modify the VTOC of the receiving volume. If you do a copy and change the VTOC size without changing the volume serial number, the volume must be varied offline after it is copied. You should not do a copy from a volume with an indexed VTOC.

Operations on Volumes Containing an Indexed VTOC

You should use Device Support Facilities to convert a VTOC to a nonindexed format to update the volume. If you do not, take note of the following information:

- *Initializing a Volume.* If you do not change the volume serial number, the volume should be varied offline before starting the job.
- *Restoring a Volume from a Dump Tape.* There are no operational requirements if you change the volume serial number or do a partial restore that does not modify the VTOC or VTOC index. If you do a restore and modify the VTOC or VTOC index without changing the volume serial number, the volume should be varied offline after it is restored.

- *Copying a Volume.* There are no operational requirements if you change the volume serial number of the receiving volume or do a partial dump without modifying the VTOC or VTOC index. If you modify the VTOC or VTOC index without changing the volume serial number, the receiving volume should be varied offline after it is copied.

- *Shared DASD Considerations.* In shared DASD environments, whenever the VTOC index is modified or relocated or whenever the volume is changed from indexed VTOC to OS VTOC or from OS VTOC to indexed VTOC, the device should be varied offline to the sharing system or systems.

Accessing the VTOC with DADSM Macros

You may use DADSM or CVAF to access the VTOC and its index. (CVAF access is described in "Accessing the VTOC and its Index with CVAF Macros" on page 42.) DADSM macros and associated tasks include:

```
LSPACE   - Obtain free space, volume fragmentation, and VTOC
           status information for a DASD volume.
OBTAIN   - Read a DSCB from a VTOC.
PARTREL  - Release unused space from a SAM or PAM data set.
REALLOC  - DASD space allocation.
RENAME   - Rename a non-VSAM data set.
SCRATCH  - Release all space and DSCBs for a non-VSAM data set.
```

The PARTREL macro is described in "Releasing Unused Space from a DASD Data Set" on page 169. The REALLOC macro is described in "Allocating a DASD Data Set" on page 174.

This section tells how to use the LSPACE, OBTAIN, SCRATCH, and RENAME macro instructions. These macros are most commonly used by the operating system and the data set utility programs (IEHMOVE, IEBCOPY, and IEHPROGM), but you may use them in your own routines. The functions you can perform with these macros are:

LSPACE	Obtaining free space, volume fragmentation, and VTOC status information for a DASD volume
OBTAIN	Reading a data set control block from the VTOC
RENAME	Changing the name of a data set
SCRATCH	Deleting a data set

You can obtain free space, volume fragmentation, and VTOC status information for a DASD volume by using the LSPACE macro instruction. LSPACE returns information to any of three user-specified areas.

You can read a data set control block (DSCB) into virtual storage by using the OBTAIN and CAMLST macro instructions. There are two ways to specify the DSCB that you want to read: by using the name of the data set associated with the DSCB, or by using the absolute track address of the DSCB. You must provide a 140-byte data area in virtual storage, into which the DSCB is to be read. When you specify the name of the data set, an identifier (format-1 or format-4) DSCB is read into virtual storage. To read a DSCB other than a format-1 or a format-4 DSCB, you must specify an absolute track address (see "Example" on page 32).

You can change a data set name by using the RENAME and CAMLST macro instructions. This causes replacement of the data set name in the data set's format-1 DSCB with the new name.

You can delete a non-VSAM data set by using the SCRATCH and CAMLST macro instructions. This causes deletion of the data set's DSCBs.

Coding examples, programming notes, and exception return code descriptions accompany the following macro instruction descriptions.

Note: You cannot use LSPACE, OBTAIN, SCRATCH, or RENAME macro instructions with either a SYSIN or SYSOUT data set.

Obtaining DASD Volume Information (LSPACE)

You can use the LSPACE macro to obtain free space, volume fragmentation, and VTOC status information for a DASD volume. LSPACE normally returns status information (such as LSPACE subfunction, return code, and reason code) to the parameter list. The format of the LSPACE parameter list is shown in Figure 8 on page 24. You may request that LSPACE return additional information such as the total number of free extents on the volume, or the fragmentation index. This additional information can be returned in either:

- *A message return area:* "Message Return Area" on page 27 describes the format and content of the message return area.

- *A data return area:* "Data Return Area" on page 27 describes the format and content of the data return area.

- *A Format-4 DSCB return area:* "Format 4 DSCB Return Area" on page 28 describes the format and content of the Format-4 DSCB return area.

The format of the LSPACE macro is:

[*symbol*]	**LSPACE**	**[UCB={*addr* | *(reg)*}]** **[,MSG={*addr* | *(reg)* | 0}** **| DATA={*addr* | *(reg)* | 0}]** **[,SMF={TEST | YES | NONE}]** **[,F4DSCB={*addr* | *(reg)* | <u>0</u>}]** **[,MF={<u>I</u> | D | (D,MSG) | (D,DATA)** **| L | (L,MSG) | (L,DATA)** **| (E,*addr*) | (E,*(reg)*)}]**

UCB={*addr* | *(reg)*}
specifies the address of the UCB for the volume whose free space information you are requesting.

addr—RX-type address
specifies the address of a fullword that contains the address of the UCB.

(reg)—(2-12)
specifies a register containing the UCB address for the device.

When using the standard (MF=I) form of the macro, you must provide a UCB address.

MSG={*addr* | *(reg)* | 0} | DATA={*addr* | *(reg)* | 0}
specifies the way LSPACE is to return free space information.

Note: The MSG and DATA parameters are mutually exclusive.

MSG={*addr* | *(reg)* | 0}
specifies the address of a caller-provided 30-byte message return area into which LSPACE returns either a free space message or, for unsuccessful requests, status information. For a description of this area, see "Message Return Area" on page 27.

addr—RX-type address
specifies the address of the message return area.

(reg)—(2-12)
specifies a register containing the address of the message return area.

0
specifies that you do not want the free space message. This is the default for all forms of the macro except execute.

DATA={*addr* | *(reg)* | 0}
specifies the address of a caller-provided 36-byte data return area into which LSPACE returns free space and volume information. For a description of this area, see "Data Return Area" on page 27.

addr—RX-type address
specifies the address of the data return area.

(reg)—(2-12)
specifies a register containing the address of the data return area.

0
specifies that you do not want the free space and volume information.

SMF={TEST | YES | NONE}
specifies the type of SMF processing desired.

TEST
specifies that LSPACE is to test for an SMF system and whether SMF volume information is desired. *Only programs executing in supervisor state, protect key 0-7, or APF-authorized may specify this operand.*

YES

specifies that you want LSPACE to provide SMF volume information. *Only programs executing in supervisor state, protect key 0-7, or APF-authorized may specify this operand.*

NONE

specifies that you do not want LSPACE to provide SMF volume information. This is the default for all forms of the macro except execute.

F4DSCB={*addr* | *(reg)* | <u>**0**</u>}

specifies the address of a 96-byte DSCB return area provided by the calling program, into which LSPACE returns the volume's format-4 DSCB. For a description of the format-4 DSCB fields, see the DSCB4 data area section in *Debugging Handbook.*

addr—RX-type address

specifies the address of the format-4 DSCB return area.

(*reg*)—**(2-12)**

specifies a register containing the address of the format-4 DSCB return area.

0

specifies that you do not want the data portion of the format-4 DSCB for the volume. This is the default for all forms of the macro except execute.

MF={<u>**I**</u> | **D** | **(D,MSG)** | **(D,DATA)** | **L** | **(L,MSG)** | **(L,DATA)** | **(E,***addr***)** | **(E,***(reg)***)**}

specifies the form of the LSPACE macro.

I

specifies the inline (standard) form of the macro. This generates an inline parameter list containing the required variables, loads the address of the parameter list in register 1, and issues an SVC 78. This form is the default.

D

generates a DSECT that maps the LSPACE parameter list. See Figure 8 on page 24 for the format of the LSPACE parameter list.

(D,MSG)

generates a DSECT that maps the message return area. For the format of the area, see "Message Return Area" on page 27.

(D,DATA)

generates a DSECT that maps the data return area. For the format of the area, see "Data Return Area" on page 27.

L

generates the required constants in the calling program. You may then issue the execute form of the macro, which uses these constants.

(L,MSG)
: generates the required message return area constants in the calling program.

(L,DATA)
: generates the required data return area constants in the calling program.

(E,*addr*)
: loads the address of the parameter list specified by *addr* into register 1, puts the specified variables into the parameter list, and issues an SVC 78.

(E,*(reg)*)
: loads the address of the parameter list specified by *(reg)* into register 1, puts the specified variables into the parameter list, and issues an SVC 78.

Name	Offset	Bytes	Description
LSPAPL			
LSPAPLID	00(X'00')	4	EBCDIC 'LSPA'
LSPANGTH	04(X'04')	2	Length of parameter list
LSPAFLAG	06(X'06')	1	Parameter flag byte
LSPASMFY		X'80'	SMF=YES
LSPASMFT		X'40'	SMF=TEST
LSPADATA		X'20'	Free space data request
LSPARSVB		...x xxxx	Reserved
LSPARSVD	07(X'07')	1	Reserved
LSPAERCD	08(X'08')	1	LSPACE return code
LSPASFID	09(X'09')	1	LSPACE subfunction
LSPASFPC		X'00'	Processing complete
LSPASFVP		X'01'	Validate parameters
LSPASFUS		X'02'	Check UCB status
LSPASFNQ		X'03'	Enq on SYSZDMNT
LSPASF45		X'04'	Read F4 and first F5 (EXCP)
LSPASFN5		X'05'	Read next F5 (EXCP)
LSPASFRV		X'06'	Read volume label (EXCP)
LSPASF4X		X'80'	Read F4 and maps (CVAFDIR)
LSPASFEX		X'81'	Get free extents (CVAFDSM)
LSPASFF0		X'82'	Get F0 count (CVAFDSM)
LSPASFVR		X'83'	Get VIR count (CVAFDSM)
LSPASFVD		X'84'	Check for VRF (CVAFVRF)
LSPASFRT	10(X'0A')	1	Subfunction return code
LSPASFRS	11(X'0B')	1	Subfunction reason code
LSPARS01		X'01'	Check parameter list storage key
LSPARS02		X'02'	Check parameter list ID
LSPARS03		X'03'	Check LSPACE flag
LSPARS04		X'04'	Check authorization for SMF flag
LSPARS05		X'05'	Check message or data return area storage key
LSPARS06		X'06'	Check format-4 DSCB return area storage key
LSPARS07		X'07'	Check UCB address
LSPARS08		X'08'	Check for virtual UCB address
LSPARS09		X'09'	Check for zero VTOC pointer
LSPAUCB	12(X'0C')	4	UCB address
LSPAFRSP	16(X'10')	4	Address of message or data return area
LSPAFMT4	20(X'14')	4	Address of format-4 DSCB

Figure 8. Format of the LSPACE Parameter List (MF=D)

Note: For more information about the LSPAERCD, LSPASFID, LSPASFRT, and LSPASFRS fields, see "LSPACE Status Information" on page 26.

Return Codes from LSPACE

Return codes from LSPACE are as follows:

Code	Meaning
0(X'00')	Successful processing
4(X'04')	Permanent I/O Error
8(X'08')	Non-Standard OS Volume
12(X'0C')	Invalid Parameter or UCB Not Ready
16(X'10')	Invalid Parameter List

LSPACE Status Information

Register 0 and the LSPACE macro's parameter list[2] contain additional diagnostic information. Figure 9 shows the relationship between the following LSPACE parameter list fields:

- LSPAERCD (return code)
- LSPASFID (subfunction identifier)
- LSPASFRT (subfunction return code)
- LSPASFRS (subfunction reason code)

LSPAERCD	LSPASFID	LSPASFRT	LSPASFRS	Description
16 (X'10')	01 (X'01')	N/A	01 (X'01')	Bad parm list storage key
16 (X'10')	01 (X'01')	N/A	02 (X'02')	Bad parm list ID
12 (X'0C')	01 (X'01')	N/A	03 (X'03')	Invalid LSPACE flag
12 (X'0C')	01 (X'01')	N/A	04 (X'04')	Not authorized for SMF
12 (X'0C')	01 (X'01')	N/A	05 (X'05')	Bad MSG/DATA area storage key
12 (X'0C')	01 (X'01')	N/A	06 (X'06')	Bad FMT4 area storage key
12 (X'0C')	01 (X'01')	N/A	07 (X'07')	UCB not found
12 (X'0C')	01 (X'01')	N/A	08 (X'08')	UCB not direct access device
12 (X'0C')	01 (X'01')	N/A	09 (X'09')	UCB VTOC pointer is zero
12 (X'0C')	02 (X'02')	N/A	N/A	Invalid UCB status
12 (X'0C')	03 (X'03')	ENQ RETC	N/A	Failed ENQ on SYSZDMNT
08 (X'08')	04 (X'04')	N/A	N/A	F5s are invalid
04 (X'04')	04 (X'04')	ECB STAT	N/A	Error reading F4 and first F5
04 (X'04')	05 (X'05')	ECB STAT	N/A	Error reading next F5
04 (X'04')	06 (X'06')	ECB STAT	N/A	Error reading volume label
04 (X'04')	80 (X'80')	DIR RETC	CVSTAT	Error getting F4/space maps
04 (X'04')	81 (X'81')	DSM RETC	CVSTAT	Error getting free extents
04 (X'04')	82 (X'82')	DSM RETC	CVSTAT	Error getting F0 count
04 (X'04')	83 (X'83')	DSM RETC	CVSTAT	Error getting VIR count
04 (X'04')	84 (X'84')	VRF RETC	CVSTAT	Error checking for VRF
00 (X'00')	00 (X'00')	N/A	N/A	No problems

Figure 9. LSPACE Status Information Relationships

LSPACE Subfunction Return Code and Reason Code

The following table identifies the information returned in the LSPASFRT and LSPASFRS fields of the LSPACE macro's parameter list.

```
N/A - Not Applicable
CVSTAT - CVSTAT field of CVAF parameter list
ENQ RETC - Return code from ENQ
DIR RETC - Return code from CVAFDIR
DSM RETC - Return code from CVAFDSM
VRF RETC - Return code from CVAFVRF
ECB STAT - ECB completion code
```

[2] Status information does not appear in the parameter list for return code 16.

LSPACE Information Return Areas

The LSPACE macro returns status information to the parameter list and, optionally, the return of volume information to any of the three caller requested return areas described below.[3]

Message Return Area: LSPACE returns information to a 30-byte message return area (Figure 10). If you provide a message return area with the MSG option, LSPACE returns EBCDIC text, qualified by return codes as shown in Figure 11.

```
LSPMSG   DSECT             Message Area
LSPMTEXT DS    CL30        Message Text
```

Figure 10. DADSM LSPACE Free Space Information Format, MF=(D,MSG)

Return Code	Text or Explanation
16(X'10')	No text returned (invalid parameter list or SMF indicator)
12(X'0C')	Text: LSPACE—UCB NOT READY
	Text: LSPACE—UCBVTOC IS ZERO
	Text: LSPACE—INVALID PARAMETER
	Text: LSPACE—NOT A DIRECT ACCESS VOL
08(X'08')	Text: LSPACE—NON-STANDARD OS VOLUME
04(X'04')	Text: LSPACE—PERMANENT I/O ERROR
00(X'00')	Text: SPACE=aaaa,bbbb,cccc/dddd,eeee

where:

aaaa	= Total number of free cylinders
bbbb	= Total number of additional free tracks
cccc	= Total number of free extents
dddd	= Number of cylinders in largest free extent
eeee	= Number of additional tracks in largest free extent

Figure 11. DADSM LSPACE Message Area Contents

Data Return Area: If you provide a data return area with the DATA option, LSPACE returns information as described in Figure 12.

[3] Requests for the MSG and DATA areas are mutually exclusive. LSPACE checks to ensure that the storage key of each information return area is equal to the caller's key or that the caller is authorized prior to its use.

Name	Offset	Bytes	Description
LSPDRETN	00(X'00')	1	Return area status byte
LSPDSPAC		X'80'	Returned space information
LSPDF0CN		X'40'	Returned format-0 DSCB count
LSPDVRCN		X'20'	Returned free VIR count
LSPDRRES		...x xxxx	Reserved
LSPDSTAT	01(X'01')	1	Status byte
LSPDIXDS		X'80'	Index exists for VTOC
LSPDIXAC		X'40'	Index VTOC active
LSPDIRES		..xx xxxx	Reserved
LSPDRSV1	02(X'02')	2	Reserved
LSPDNEXT	04(X'04')	4	Number of free extents
LSPDTCYL	08(X'08')	4	Total free cylinders
LSPDTTRK	12(X'0C')	4	Total additional free tracks
LSPDLCYL	16(X'10')	4	Number of cylinders in largest free extent
LSPDLTRK	20(X'14')	4	Number of additional tracks in largest free extent
LSPDF0S	24(X'18')	4	Format-0 DSCB count
LSPDVIRS	28(X'1C')	4	Free VIR count
LSPDFRAG	32(X'20')	4	Fragmentation index[1]

Figure 12. Format of the LSPACE Data Return Area

Note to Figure 12:

[1] The fragmentation index is a numeric representation of the relative size and distribution of free space on the volume. A large index value indicates a high degree of fragmentation.

Format 4 DSCB Return Area: If you provide a format-4 DSCB return area with the F4DSCB option, LSPACE returns information as described by the DSCB4 data area in *Debugging Handbook*.

Example of LSPACE Using Message Return Area

The following example requests that LSPACE return free space information in the message return area.

```
LSPAMFIM LSPACE MSG=MYMSG,UCB=(R10),MF=I
```

Example of LSPACE Using Data Return Area

The following example requests that LSPACE return free space information in the data return area.

```
LSPAMFID LSPACE DATA=MYDATA,UCB=(R10),MF=I
```

Example of LSPACE Specifying List and Execute Forms

The following example uses the list form of the macro to define the parameter list, and the execute form to refer to the same parameter list

```
LSPALIST LSPACE MSG=MYDATA,MF=L
         .
         .
         .
LSPAEX   LSPACE MF=(E,LSPALIST),UCB=(R10)
```

Reading a Control Block from the VTOC

Reading a DSCB by Name (OBTAIN and CAMLST SEARCH)

If you specify a data set name using OBTAIN and the CAMLST SEARCH option, the OBTAIN routine reads the 96-byte data portion of the identifier (format-1) DSCB and the absolute track address of the DSCB into virtual storage. The absolute track address is a 5-byte field in the form CCHHR. The absolute track address field contains zeros for VSAM and VIO data sets.

Because the VTOC does not contain a format-1 DSCB for a suballocated VSAM data space, an OBTAIN request, which searches the VTOC for such a data space's DSCB, fails. If the volume contains VSAM data sets, the OBTAIN routine uses information from the VSAM catalog to build a pseudo format-1 DSCB, setting its CCHHR to zeros.

The format is:

[symbol] *listname*	**OBTAIN** **CAMLST**	*listname-addrx* **SEARCH** *,dsname-relexp* *,vol-relexp* *,wkarea-relexp*

listname-addrx
: points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

SEARCH
: this operand must be coded as shown.

dsname-relexp
: specifies the virtual storage location of a fully qualified data set name. The area that contains the name must be 44 bytes long.

Note: A DSNAME of 44 bytes of X'04' (X'040404...04') can be used to read a format-4 DSCB.

vol-relexp
: specifies the virtual storage location of the 6-byte volume serial number on which the DSCB is located.

wkarea-relexp
: specifies the virtual storage location of a 140-byte work area that you must define.

Example: In the following example, the identifier (format-1) DSCB for data set A.B.C is read into virtual storage using the SEARCH option. The serial number of the volume containing the DSCB is 770655.

```
         OBTAIN   DSCBABC             READ DSCB FOR DATA
*                                     SET A.B.C INTO DATA
*                                     AREA NAMED WORKAREA

DSCBABC  CAMLST   SEARCH,DSABC,VOLNUM,WORKAREA
DSABC    DC       CL44'A.B.C'         DATA SET NAME
VOLNUM   DC       CL6'770655'         VOLUME SERIAL NUMBER
WORKAREA DS       140C                140-BYTE WORK AREA
```

Note: Check the return codes.

The OBTAIN macro instruction points to the CAMLST macro instruction. SEARCH, the first operand of CAMLST, specifies that a DSCB be read into virtual storage, using the data set name you have supplied at the address indicated in the second operand. DSABC, the second operand, specifies the virtual storage location of a 44-byte area into which you have placed the fully qualified name of the data set whose format-1 DSCB is to be read. VOLNUM, the third operand, specifies the virtual storage location of a 6-byte area into which you have placed the serial number of the volume containing the required DSCB. WORKAREA, the fourth operand, specifies the virtual storage location of a 140-byte work area into which the DSCB is to be returned.

Control is returned to your program at the next executable instruction following the OBTAIN macro instruction. If the DSCB has been successfully read into your work area, register 15 contains zeros. Otherwise, register 15 contains one of the following return codes. The return codes are shown in decimal, with hexadecimal values in parentheses.

Return Codes from OBTAIN (Reading by data set name)

Code	Meaning
4(X'04')	The required volume was not mounted.
8(X'08')	The format-1 DSCB was not found in the VTOC of the specified volume.
12(X'0C')	A permanent I/O error was encountered, or an invalid format-1 DSCB was found when processing the specified volume, or an unexpected error return code was received from CVAF (Common VTOC Access Facility).
16(X'10')	Invalid work area pointer.

After execution of these macro instructions, the first 96 bytes of the work area contain the data portion of the identifier (format-1 or format-4) DSCB; the next 5 bytes contain the absolute track address (CCHHR) of the DSCB. These 5 bytes contain zeros for VSAM or VIO data sets.

Reading a DSCB by Absolute Device Address (OBTAIN and CAMLST SEEK)

You can read any DSCB from a VTOC using OBTAIN and the CAMLST SEEK option. You specify the SEEK option by coding SEEK as the first operand of the CAMLST macro and by providing the absolute device address of the DSCB you want to read, unless the DSCB is for a VIO data set. Only the SEARCH option can be used to read the DSCB of a VIO data set.

The format is:

[*symbol*]	**OBTAIN**	*listname-addrx*
listname	**CAMLST**	**SEEK** *,cchhr-relexp* *,vol-relexp* *,wkarea-relexp*

listname-addrx
: points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

SEEK
: this operand must be coded as shown.

cchhr-relexp
: specifies the virtual storage location of the 5-byte absolute device address (CCHHR) of a DSCB.

vol-relexp
: specifies the virtual storage location of the 6-byte volume serial number on which the DSCB is located.

wkarea-relexp
: specifies the virtual storage location of a 140-byte work area that you must define.

Example: In the following example, the DSCB at actual-device address X'00 00 00 01 07' is returned in the virtual storage location READAREA, using the SEEK option. The DSCB resides on the volume with the volume serial number 108745.

```
         OBTAIN   ACTADDR          READ DSCB FROM
*                                  LOCATION SHOWN IN CCHHR
*                                  INTO STORAGE AT LOCATION
*                                  NAMED READAREA

ACTADDR  CAMLST   SEEK,CCHHR,VOLSER,READAREA
CCHHR    DC       XL5'0000000107' ABSOLUTE TRACK ADDRESS
VOLSER   DC       CL6'108745'     VOLUME SERIAL NUMBER
READAREA DS       140C            140-BYTE WORK AREA
```

Note: Check the return codes.

The OBTAIN macro points to the CAMLST macro. SEEK, the first operand of CAMLST, specifies that a DSCB be read into virtual storage. CCHHR, the second operand, specifies the storage location that contains the 5-byte actual-device address of the DSCB. VOLSER, the third operand, specifies the storage location that contains the serial number of the volume where the DSCB resides. The fourth operand, READAREA, specifies the storage location to which the 140-byte DSCB is to be returned.

Control is returned to your program at the next executable instruction following the OBTAIN macro instruction. If the DSCB has been successfully read into your work area, register 15 contains zeros. Otherwise, register 15 contains one of the following return codes. The return codes are shown in decimal, with hexadecimal values in parentheses.

Return Codes from OBTAIN (Reading by Absolute Device Address)

Code	Meaning
4(X'04')	The required volume was not mounted.
8(X'08')	The format-1 DSCB was not found in the VTOC of the specified volume.
12(X'0C')	A permanent I/O error was encountered or an unexpected error return code was received from CVAF.
16(X'10')	Invalid work area pointer.
20(X'14')	The SEEK option was specified and the absolute track address (CCHHR) is not within the boundaries of the VTOC.

Deleting a Data Set from the VTOC (SCRATCH and CAMLST SCRATCH)

You can use the SCRATCH and CAMLST macro instructions to delete a non-VSAM data set. SCRATCH processing deletes the associated data set control blocks (DSCBs) and makes the space occupied by the data set available for reallocation. Be aware that this process may not erase the data from the disk. Data sets that contain sensitive data should be erased (overwritten with zeros) before their space is made available. This erase can either be done before issuing the SCRATCH macro, or be requested in scratch processing by

- Providing an associated RACF ERASE attribute, or
- Activating bit 21 (X'00 00 04 00') of the SCRATCH parameter list.

Authorized callers of SCRATCH may set bit 22 to '1' to override the RACF profile ERASE attribute.

If you want to scratch a data set being processed using virtual input/output (VIO), the data set must have been allocated for use by your job. Scratching VIO data sets not allocated to your job is not allowed.

If the data set to be deleted is sharing one or more cylinders with one or more data sets (a split-cylinder data set), the space will not be made available for reallocation until all data sets on the shared cylinders are deleted.

A data set cannot be deleted if the expiration date in the identifier (format-1) DSCB has not passed, unless you choose to ignore the expiration date. You may specify that DADSM is to ignore the expiration date by specifying the OVRD option in the CAMLST macro instruction.

DADSM SCRATCH processing supports three never-scratch dates. To ensure that a data set will never be scratched, specify the expiration date as either of the following:

- 1999.365
- 1999.366
- 1999.999

For information on RACF-defined data sets, see *RACF General Information Manual*. You may scratch a RACF-defined data set (that is, the DSCB indicates RACF-defined) only if you have alter access authority to either the data set/volume serial in the DATASET class, or to the volume serial in the DASDVOL class (if the volume is RACF-defined).

If a data set to be deleted is stored on more than one volume, either a device must be available for mounting the volumes or at least one volume must be mounted. In addition, all other required volumes must be serially mountable.

When deleting a data set, you must build a volume list in virtual storage. This volume list consists of an entry for each volume on which the data set resides. The first two bytes of the list indicate the number of entries in the list. Each 12-byte entry consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte scratch status code that should be initialized to zero.

If the space to be deleted is a VSAM data space, you must use the DELETE command provided by access method services. For complete information about the DELETE command, see *Access Method Services Reference*.

Volumes are processed in the order that they appear in the volume list. The volume at the beginning of the list is processed first. If a volume is not mounted, a message is issued to the operator requesting that a volume be mounted. (A volume mount message will not be issued for a mass storage system (MSS) virtual volume; however, a status code will be returned to your program.) This is only done if register 0 has been loaded with the address of the UCB associated with the device where unmounted volumes are to be mounted. (The device must be allocated to your job.) If you do not load register 0 with a UCB address, its contents must be zero, and at least one of the volumes in the volume list must be mounted before the SCRATCH macro instruction is issued.

If the requested volume cannot be mounted, the operator issues a reply indicating that the request cannot be fulfilled. A status code is then set in the last byte of the volume pointer (the second byte of the scratch status code) for the unavailable volume, and the next volume indicated in the volume list is processed.

The format is:

[*symbol*] *listname*	**SCRATCH** **CAMLST**	*listname-addrx* **SCRATCH** *,dsname-relexp* *,,vol list-relexp* **[,,OVRD]**

listname-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

SCRATCH

this operand must be coded as shown.

dsname-relexp

specifies the virtual storage location of a fully qualified data set name. The area that contains the name must be 44 bytes long. The name must be defined by a C-type Define Constant (DC) instruction.

vol list-relexp

specifies the virtual storage location of an area that contains a volume list. The area must begin on a halfword boundary.

OVRD

when coded as shown, specifies that the expiration date in the DSCB should be ignored.

Example: In the following example, data set A.B.C is deleted from two volumes. The expiration date in the identifier (format-1) DSCB is ignored.

```
         SR        0,0                SET REG 0 TO ZERO
         SCRATCH   DELABC             DELETE DATA SET A.B.C
*                                     FROM TWO VOLUMES,
*                                     IGNORING EXPIRATION
*                                     DATE IN THE DSCB

DELABC   CAMLST    SCRATCH,DSABC,,VOLIST,,OVRD
DSABC    DC        CL44'A.B.C'        DATA SET NAME
VOLIST   DC        H'2'               NUMBER OF VOLUMES
         DC        X'3030200E'        3380 DISK DEVICE CODE
         DC        CL6'000017'        VOLUME SERIAL NO.
         DC        H'0'               SCRATCH STATUS CODE
         DC        X'3030200E'        3380 DISK DEVICE CODE
         DC        CL6'000018'        VOLUME SERIAL NO.
         DC        H'0'               SCRATCH STATUS CODE
```

Note: Check the return codes and SCRATCH status codes.

The SCRATCH macro instruction points to the CAMLST macro instruction. SCRATCH, the first operand of CAMLST, specifies that a data set be deleted. DSABC, the second operand, specifies the virtual storage location of a 44-byte area where you have placed the fully qualified name of the data set to be deleted. VOLIST, the fourth operand, specifies the virtual storage location of the volume list you have built. OVRD, the sixth operand, specifies that the expiration date in the DSCB of the data set to be deleted be ignored.

When you attempt to delete a password-protected data set that is not also RACF-protected, the operating system issues a message (IEC301A) to ask the operator at the console or the terminal operator of a remote console to enter the password. The data set will be scratched only if the password supplied is associated with a WRITE protection mode indicator. The protection mode indicator is described in Chapter 5, "Password Protecting Data Sets."

Control is returned to your program at the next executable instruction following the SCRATCH macro instruction. If the data set has been successfully deleted, register 15 will contain zeros, and the scratch status code in the volume list entry for each volume will be set to zero. Otherwise, register 15 will contain one of the return codes that follow. To determine whether the data set has been successfully deleted from each volume on which it resides, you must examine the scratch status code, that is, the last byte of each entry in the volume list.

Return Codes from SCRATCH

Code	Meaning
4(X'04')	No volumes containing any part of the data set were mounted, nor did register 0 contain the address of a unit that was available for mounting a volume of the data set. The data set may be a VIO data set that was not allocated during your job. (This return code is accompanied by a scratch status code of 5 in each entry of the volume list.)
8(X'08')	An unusual condition was encountered on one or more volumes.
12(X'0C')	The volume list passed was invalid. The scratch status code (the last byte of each volume list entry) will not have been modified during scratch processing.

Status Codes from SCRATCH

After the SCRATCH macro instruction is executed, the last byte of each 12-byte entry in the volume list indicates one of the following conditions in binary codes:

Scratch Status Code	Meaning
0	All DSCBs for the data set have been deleted from the VTOC on the volume pointed to.
1	The VTOC of this volume does not contain the format-1 DSCB for the data set to be deleted.
2	The macro instruction failed when the correct password was not supplied in the two attempts allowed, or an attempt was made to scratch a VSAM data space or data set cataloged in an Integrated Catalog Facility catalog.
3	The data set was not deleted from this volume because either the OVRD option was not specified or the retention cycle has not expired.
4	A permanent I/O error was encountered, or an invalid format-1 DSCB was found when processing this volume, or an unexpected error return code was received from CVAF.
5	It could not be verified that this volume was mounted, and no device was available for mounting this volume.
6	The operator was unable to mount this volume. For IBM Mass Storage Systems (MSS), a volume mount failure occurred. For a JES3-managed virtual volume, JES3 would not allow the volume to be mounted.
7	The specified data set could not be scratched because it was being used.
8	The DSCB indicates the data set is defined to RACF, but either the user is not authorized to access the data set or the volume, or the data set is a VSAM data space, or the data set is cataloged in an Integrated Catalog Facility catalog, or the data set is not defined to RACF.

Renaming a Data Set in the VTOC (RENAME and CAMLST RENAME)

You rename a data set that is not cataloged in an Integrated Catalog Facility catalog or VSAM catalog by using the RENAME and CAMLST macro instructions. These cause the data set name in all format-1 DSCBs for the data set to be replaced by the new name you supply. (VIO data sets cannot be renamed.)

If a data set to be renamed is stored on more than one volume, either a device must be available for mounting the volumes, or at least one volume must be mounted. In addition, all other volumes of the data set must be serially mountable.

For information on RACF-defined data sets, see *RACF General Information Manual*. Only a user with alter access authority may rename a RACF-defined data set.

When renaming a data set, you must build a volume list in virtual storage. This volume list consists of an entry for each volume on which the data set resides. The first two bytes of the list indicate the number of entries in the list. Each 12-byte volume list entry consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte rename status code that should be initialized to zero. Volumes are processed in the order in which they appear in the volume list. The first volume on the list is processed first. If a volume is not mounted, a message is issued to the operator requesting that the volume be mounted. (A volume mount message will not be issued for an MSS volume; however, a status code is returned to your program.) This is only done if you indicate the direct access device on which unmounted volumes are to be mounted by loading register 0 with the address of the UCB associated with the device to be used. (The device must be allocated to your job.) If you do not load register 0 with a UCB address, its contents must be zero, and at least one of the volumes in the volume list must be mounted before the RENAME macro instruction is executed.

If the operator cannot mount a volume in the volume list, a reply is issued that the request cannot be fulfilled. A status code is then set in the last byte of the volume list entry (the second byte of the rename status code) for the unavailable volume, and the next volume indicated in the volume list is processed or requested.

The format is:

[*symbol*] *listname*	**RENAME** **CAMLST**	*listname-addrx* **RENAME** *,dsname-relexp* *,new name-relexp* *,vol list-relexp*

listname-addrx
: points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

RENAME
: this operand must be coded as shown.

dsname-relexp
: specifies the virtual storage location of a fully qualified data set name to be replaced. The area that contains the name must be 44 bytes long. The name must be defined by a C-type Define Constant (DC) instruction.

new name-relexp
: specifies the virtual storage location of a fully qualified data set name that is to be used as the new name. The area that contains the name must be 44 bytes long. The name must be defined by a C-type Define Constant (DC) instruction.

vol list-relexp
: specifies the virtual storage location of an area that contains a volume list. The area must begin on a halfword boundary.

Example: In the following example, data set A.B.C is renamed D.E.F. The data set resides on two volumes.

```
         SR      0,0              SET REG 0 TO ZERO
         RENAME  DSABC            CHANGE DATA SET
                                  NAME A.B.C TO D.E.F

DSABC    CAMLST  RENAME,OLDNAME,NEWNAME,VOLIST
OLDNAME  DC      CL44'A.B.C'      OLD DATA SET NAME
NEWNAME  DC      CL44'D.E.F'      NEW DATA SET NAME
VOLIST   DC      H'2'             TWO VOLUMES
         DC      X'3030200E'      3380 DISK DEVICE CODE
         DC      CL6'000017'      VOLUME SERIAL NO.
         DC      H'0'             RENAME STATUS CODE
         DC      X'3030200E'      3380 DISK DEVICE CODE
         DC      CL6'000018'      VOLUME SERIAL NO.
         DC      H'0'             RENAME STATUS CODE
```

Note: Check the return codes and RENAME status codes.

The RENAME macro instruction points to the CAMLST macro instruction. RENAME, the first operand of CAMLST, specifies that a data set be renamed. OLDNAME, the second operand, specifies the virtual storage location of a 44-byte area where you have placed the fully qualified name of the data set to be renamed. NEWNAME, the third operand, specifies the virtual storage location of a 44-byte area where you have placed the new name of the data set. VOLIST, the fourth operand, specifies the virtual storage location of the volume list you have built.

Control is returned to your program at the next executable instruction following the RENAME macro instruction. If the data set has been successfully renamed, register 15 will contain zeros, and the rename status code in the volume list entry for each volume will be set to zero. Otherwise, register 15 will contain one of the return codes below. To determine whether the data set has been successfully renamed on each volume where it resides, you must examine the rename status code, the last byte of each entry in the volume list.

Code	Meaning
4(X'04')	No volumes containing any part of the data set were mounted, nor did register 0 contain the address of a unit that was available for mounting a volume of the data set to be renamed. The data set may be a VIO data set and cannot be renamed. (This return code is accompanied by a rename status code of 5 in each entry of the volume list.)
8(X'08')	An unusual condition was encountered on one or more volumes.
12(X'0C')	The volume list passed was invalid. The rename status code, the last byte of each volume list entry, will not have been modified during rename processing.

After the RENAME macro instruction is executed, the last byte of each 12-byte entry in the volume list indicates one of the following conditions in binary code:

Rename Status Code	Meaning
0	The format-1 DSCB for the data set has been renamed in the VTOC on the volume pointed to.
1	The VTOC of this volume does not contain the format-1 DSCB for the data set to be renamed.
2	The macro instruction failed when the correct password was not supplied in the two attempts allowed, or the user tried to rename a VSAM data space or VSAM data set cataloged in an Integrated Catalog Facility catalog.
3	A data set with the new name already exists on this volume.
4	A permanent I/O error was encountered, or an invalid format-1 DSCB was found when trying to rename the data set on this volume, or an unexpected error return code was received from CVAF.
5	It could not be verified that the volume was mounted, and no device was available for mounting the volume.
6	The operator was unable to mount this volume. For Mass Storage Systems (MSS), a volume mount failure occurred. For a JES3-managed virtual volume, JES3 would not allow the volume to be mounted.
7	The specified data set could not be renamed on this volume because it was being used.
8	The data set is defined to RACF, but either the user is not authorized to alter the data set or the data set is defined to RACF on multiple volumes.

When you attempt to rename a password-protected data set, the operating system issues a message (IEC301A) to ask the operator or remote console operator to verify the password. The data set will be renamed only if the password supplied is associated with a WRITE protection mode indicator. The protection mode indicator is described in Chapter 4, "Password Protecting Data Sets" on page 111.

Accessing the VTOC and its Index with CVAF Macros

You may use CVAF or DADSM to access the VTOC or its index. DADSM access is described in "Accessing the VTOC with DADSM Macros" on page 19.

CVAF macros and associated tasks include:

```
CVAFDIR   - Directly access DSCBs or VTOC index records.
CVAFDSM   - Obtain volume free space information.
CVAFFILT  - Read sets of DSCBs for one or more DASD data sets.
CVAFSEQ   - Retrieval of the following:
              - Data set names from an active VTOC index.
              - DSCBs in physical-sequential order.
              - DSCBs in data set name order (index required).
CVAFTST   - Determine if a DASD volume has an active VTOC index.
```

Appendix A, "CVAF VTOC Access Macros" on page 231, contains detailed descriptions of these macros. Appendix B, "Examples of VTOC Access Macros" on page 259, contains examples of their use.

Serialization and Updating

CVAF requires that you provide all necessary system resource serialization for your request. You can only ensure the integrity of multiple data elements (sets of DSCBs and/or VIRs) returned by CVAF if you serialize system resources adequately. You compound this exposure if you must make multiple CVAFFILT requests for a desired set of DSCBs and/or VIRs.

You must weigh possible system performance loss because of serialization against the potential loss of data integrity. If you make updates without adequate serialization, you may compromise the integrity of the volume's VTOC, the VTOC index, and/or any associated data set.

CVAF only honors requests to modify the volume's VTOC and/or index from authorized programs.

CVAF assumes that an authorized program holds an exclusive RESERVE (or ENQ) on the qname (major name) of SYSVTOC, rname (minor name) of the volume's serial number, with the scope of SYSTEMS. The SYSVTOC qname does not serialize access to the format-1 DSCB for a data set. You may provide this serialization by allocating the data set with disposition OLD, MOD, or NEW (not SHR). This causes the proper ENQ, ensuring that no other job can update that data set's format-1 DSCB.

Identifying the Volume

If you are authorized, you may identify the volume to the CVAFDIR, CVAFDSM, CVAFFILT, and CVAFSEQ macros by specifying the address of its UCB. If you are not authorized, you must identify the volume by specifying the address of a SAM or EXCP DEB opened to the volume's VTOC.

The DEB can be obtained by opening a DCB using the RDJFCB and OPEN TYPE=J macros. The DCBs DDNAME is that of a DD statement allocated to the unit whose VTOC is to be accessed. After issuing the RDJFCB macro, the JFCBDSNM field is overlaid with the data set name of the format-4 DSCB: 44X'04'. You open the DCB for INPUT by using OPEN TYPE=J. The DEB address is in DCB field DCBDEBA. The OPEN macro is described under "OPEN—Initialize Data Control Block for Processing the JFCB" on page 148; the RDJFCB macro is described under "RDJFCB—Read a Job File Control Block" on page 137.

If a CVAF macro call specifies IOAREA=KEEP, a subsequent CVAF call using a different CVPL may omit the UCB and DEB keywords and supply the IOAREA address from the other CVPL. You can use the IOAREA keyword to do this.

The requirements cited above do not apply to the CVAFTST macro. The CVAFTST macro only allows you to identify the VTOC by specifying a UCB, and does not require that you be authorized.

Using Registers

Register 1 contains the address of the CVAF parameter list (CVPL). Register 15 contains the return code when processing for a function is complete.

Generating a CVPL (CVAF Parameter List)

All the CVAF macros except CVAFTST use the CVPL to pass parameters to CVAF. The CVAFTST macro expands to provide its only parameter (UCB address) in register 1, and calls the associated CVAF module. CVAF uses the CVPL to return information related to the CVAF request.

CVAF generates a CVPL when you specify the CVAFDIR, CVAFDSM, CVAFFILT, or CVAFSEQ macro with MF=L or MF=I as a subparameter. If you do not specify the MF subparameter, MF=I is the default. Upon return, the CV1IVT bit indicates whether an indexed or nonindexed VTOC was accessed. The CVSTAT field contains feedback when an error occurs. The address of the map records buffer list is returned in the CVMRCDS field. The address of the VIER buffer list is returned in the CVIRCDS field. CVAF returns the CVAF I/O area address in the CVIOAR field. CVAF returns the CVAF Filter Save Area address in the CVFSA field.

You may use the CVPL generated by the MF=L or MF=I form of the CVAFDIR, CVAFDSM, CVAFFILT, or CVAFSEQ macros (by using the MF=E keyword) to execute a different function than that specified by the macro that originally generated the CVPL. If you specify a CVAF filter request, you must use a CVPL generated by the CVAFFILT macro. (To support the CVFSA field, the CVAFFILT macro generates a CVPL four bytes larger than that generated by the other CVAF macros.)

The ICVAFPL macro maps the CVPL. The format of the CVPL is shown in Figure 13 on page 44.

Name	Offset	Bytes	Description
CVPL			
CVLBL	00(X'00')	4	EBCDIC 'CVPL'
CVLTH	04(X'04')	2	Length of CVPL 64(X'40') for macros other than CVAFFILT 68(X'44') for CVAFFILT macro
CVFCTN	06(X'06')	1	Function byte (See Figure 14 on page 45)
CVSTAT	07(X'07')	1	Status information
CVFL1	08(X'08')	1	First flag byte
CV1IVT		X'80'	Indexed VTOC accessed
CV1IOAR		X'40'	IOAREA=KEEP
CV1PGM		X'20'	BRANCH=(YES,PGM)
CV1MRCDS		X'10'	MAPRCDS=YES
CV1IRCDS		X'08'	IXRCDS=KEEP
CV1MAPIX		X'04'	MAP=INDEX
CV1MAPVT		X'02'	MAP=VTOC
CV1MAPVL		X'01'	MAP=VOLUME
CVFL2	09(X'09')	1	Second flag byte
CV2HIVIE		X'80'	HIVIER=YES
CV2VRF		X'40'	VRF data exists
CV2CNT		X'20'	COUNT=YES
CV2RCVR		X'10'	RECOVER=YES
CV2SRCH		X'08'	SEARCH=YES
CV2DSNLY		X'04'	DSNONLY=YES
CV2VER		X'02'	VERIFY=YES
CV2NLEVL		X'01'	New highest level VIER (output)
CVFL3	10(X'0A')	1	Third flag byte
CV3FILT		X'80'	FLTAREA=KEEP
CV3IXERR		X'40'	Index error found
		..xx xxxx	Reserved
	11(X'0B')	1	Reserved
CVUCB	12(X'0C')	4	UCB address
CVDSN	16(X'10')	4	Data set name address
CVFCL	16(X'10')	4	Filter criteria list address
CVBUFL	20(X'14')	4	Buffer list address
CVIRCDS	24(X'18')	4	Index VIRs buffer list address
CVMRCDS	28(X'1C')	4	Map VIRs buffer list address
CVIOAR	32(X'20')	4	I/O area address
CVDEB	36(X'24')	4	DEB address
CVARG	40(X'28')	4	Argument address
CVSPACE	44(X'2C')	4	SPACE parameter list address
CVEXTS	48(X'30')	4	Extent table address
CVBUFL2	52(X'34')	4	New VRF VIXM buffer list address
CVVRFDA	56(X'38')	4	VRF data address
CVCTAR	60(X'3C')	4	Count area address
CVFSA	64(X'40')	4	Filter save area

Figure 13. Format of the CVAF Parameter List (CVPL)

Note: The CVAFFILT macro generates a CVPL four bytes longer (total length = X'44') than that generated by the other CVAF macros (total length = X'40').

The possible contents of the CVFCTN field in the CVPL and their meanings are as follows:

Name	Description		
CVDIRD	X'01'	-CVAFDIR	ACCESS=READ
CVDIWR	X'02'	-CVAFDIR	ACCESS=WRITE
CVDIRLS	X'03'	-CVAFDIR	ACCESS=RLSE
CVSEQGT	X'04'	-CVAFSEQ	ACCESS=GT
CVSEQGTE	X'05'	-CVAFSEQ	ACCESS=GTEQ
CVDMIXA	X'06'	-CVAFDSM	ACCESS=IXADD
CVDMIXD	X'07'	-CVAFDSM	ACCESS=IXDLT
CVDMALC	X'08'	-CVAFDSM	ACCESS=ALLOC
CVDMRLS	X'09'	-CVAFDSM	ACCESS=RLSE
CVDMMAP	X'0A'	-CVAFDSM	ACCESS=MAPDATA
CVVOL	X'0B'	-CVAFVOL	ACCESS=VIBBLD
CVVRFRD	X'0C'	-CVAFVRF	ACCESS=READ
CVVRFWR	X'0D'	-CVAFVRF	ACCESS=WRITE
CVFIRD	X'0E'	-CVAFFILT	ACCESS=READ
CVFIRES	X'0F'	-CVAFFILT	ACCESS=RESUME
CVFIRLS	X'10'	-CVAFFILT	ACCESS=RLSE

Figure 14. CVFCTN Field of CVPL—Contents and Definitions

Buffer Lists

A buffer list consists of one or more chained control blocks, each with a header and buffer list entries, obtained and initialized by your program before calling CVAF. The header indicates whether the buffer list is for DSCBs or VTOC index records. The entries point to and describe the buffers.

You can create buffer lists in two ways:

- Directly, when you fill in the arguments and buffer addresses of DSCBs or VIRs to be read or written
- Indirectly (by CVAF), when you code the IXRCDS=KEEP and/or MAPRCDS=YES keywords

The ICVAFBFL macro maps CVAF buffer lists. Figure 15 on page 46 shows the format of a buffer list header. Figure 16 on page 47 shows the format of a buffer list entry.

Buffer List Header: The buffer list header indicates whether the buffer list describes buffers for DSCBs or for VTOC index records. The DSCB bit must be set to one and the VIR bit to zero for CVAF to process a request to read or write a DSCB. CVAF requires that you provide buffer lists and buffers in your program's protect key. CVAF uses the protect key and subpool fields in the buffer list header only if you code ACCESS=RLSE.

Each buffer list header contains a count of the number of entries in the buffer list that directly follows the header.

The forward chain address chains buffer lists together. You must not chain DSCB buffer lists to VIR buffer lists, or VIR buffer lists to DSCB buffer lists.

The format of the buffer list header is shown in Figure 15.

Name	Offset	Bytes	Description
BFLHDR	0(X'00')	8	Buffer list header
BFLHNOE	0(X'00')	1	Number of entries
BFLHFL	1(X'01')	1	Flag byte and key
BFLHKEY		xxxx	Protect key of buffer list and buffers
BFLHVIR		X'08'	Buffer list entries describe VIRs
BFLHDSCB		X'04'	Buffer list entries describe DSCBs
		xx	Reserved
	2(X'02')	1	Reserved
BFLHSP	3(X'03')	1	Identifies the subpool of buffer list and buffers
BFLHFCHN	4(X'04')	4	Forward chain address of next buffer list

Figure 15. Format of a Buffer List Header

Buffer List Entry: A buffer list contains one or more entries. Each entry provides the buffer address, the length of the DSCB or VIR buffer, the argument, and an indication whether the argument is an RBA, a TTR, or a CCHHR.

The fields and bit uses are listed below.

- For a VIR buffer, the TTR and CCHHR bits must be 0, and the RBA bit must be 1.
- For a DSCB buffer, the RBA bit must be 0, and one of either the TTR or CCHHR bits must be set to 1 (they must not both be 1).
- The BFLEAUPD bit is an output indicator from CVAF that the BFLEARG field of a VIR buffer list was updated.
- The BFLEMOD bit indicates that a VIR buffer was modified and must be written; if no BFLEMOD bits are on in any of the entries for a CVAFDIR ACCESS=WRITE, all buffers are written.
- The BFLESKIP bit is used to cause an entry to be ignored.
- The BFLEIOER bit is an output indicator from CVAF to indicate an I/O error occurred during reading or writing of the DSCB or VIR.
- The BFLELTH field is the length of the buffer; for a DSCB buffer, the length must be 96 or 140; for a VIR buffer, the length must be the length of the buffer divided by 256.
- The BFLEARG field is the argument of the DSCB or VIR. Specify the desired format of the 5-byte field by setting either the BFLECHR, BFLETTR, or

BFLERBA bit to 1. The respective BFLEARG values and formats are as follows:

- CCHHR=5 byte CCHHR
- TTR=0TTR0
- RBA=One byte of 0 followed by a 4-byte RBA

The optional and required values for BFLEARG are dependent upon the variables associated with a given request. These are described in the following request-oriented topics.

The format of the buffer list entry is shown in Figure 16.

Name	Offset	Bytes	Description
BFLE	0(X'00')	12	Buffer list entry
BFLEFL	0(X'00')	1	Flag byte
BFLERBA		X'80'	Argument is RBA
BFLECHR		X'40'	Argument is CCHHR
BFLETTR		X'20'	Argument is TTR
BFLEAUPD		X'10'	CVAF updated argument field
BFLEMOD		X'08'	Data in buffer has been modified
BFLESKIP		X'04'	Skip this entry
BFLEIOER		X'02'	I/O error
		x	Reserved
	1(X'01')	1	Reserved
BFLELTH	2(X'02')	1	Length of VIR buffer divided by 256, or length of DSCB buffer
BFLEARG	3(X'03')	5	Argument of VIR or DSCB
BFLEATTR	4(X'04')	3	TTR of DSCB
BFLEARBA	4(X'04')	4	RBA of VIR
BFLEBUF	8(X'08')	4	Buffer address

Figure 16. Format of a Buffer List Entry

Accessing the DSCB Directly

You may use the CVAFDIR macro to read or write a DSCB. You may also use it to read or write VTOC index records for indexed VTOCs. "CVAFDIR Macro" on page 231 discusses detailed information about the CVAFDIR VTOC access macro.

After a CVAFDIR call, you may test the CVAF parameter list bit, CV1IVT, to determine whether the VTOC is indexed or nonindexed.

Specifying a Data Set Name to Read or Write a DSCB: If you want to read or write a single DSCB by specifying only the data set name (that is, BFLEARG is zero) you must specify either ACCESS=READ or ACCESS=WRITE.

Specify the address of the data set name in the DSN keyword. Specify the address of the buffer list in the BUFLIST keyword. Each of these areas and the associated buffers must be in your program's protect key.

The buffer list must contain at least one buffer list entry with the skip bit off and a pointer to a 96-byte buffer. You must not provide 140-byte buffers. You may chain buffer lists together, but CVAF only uses the first eligible entry.

For an indexed VTOC, CVAF searches the index for the data set name and, if it is found, puts the DSCB argument into the buffer list entry and uses it to read or write the DSCB. If CVAF cannot find the data set name in the index, CVAF does a key search of the VTOC.

For a nonindexed VTOC, CVAF uses a channel program to do a key search of the VTOC to locate the data set name and read or write the DSCBs. If CVAF finds the data set name, CVAF puts the DSCB argument into the buffer list entry.

The DSCB argument returned in the buffer list entry is in the format determined by the buffer list entry bits BFLECHR or BFLETTR.

If CVAF does not find the data set name in the VTOC, it provides a return code of '4' in register 15, and an error code of '1' in the CVSTAT field.

Specifying the DSCB Location: If you want to read or write a DSCB by specifying the DSCB's location (that is, BFLEARG), you must specify either ACCESS=READ or ACCESS=WRITE.

Specify the address of the data set name in the DSN keyword. Specify the address of the buffer list in the BUFLIST keyword. Each of these areas and the associated buffer(s) must be in your program's protect key.

The buffer list must have at least one buffer list entry with the skip bit off and a pointer to a 96-byte or 140-byte buffer. You may chain buffer lists together, but CVAF only uses the first eligible entry.

If the buffer is for a 96-byte read or write, CVAF issues a channel program to verify that the key in the DSCB is the same as the 44-byte data set name you provide. CVAF does not execute the read or write unless the keys match. If they do not match, CVAF ignores the specified BFLEARG and reads or writes the DSCB according to the rules given in "Specifying a Data Set Name to Read or Write a DSCB" on page 48.

If the buffer is for a 140-byte read or write, CVAF issues a channel program to read or write the DSCB at the location specified in the buffer list entry. CVAF does not use the data set name you specified. If you specify VERIFY=YES, CVAF verifies that the designated DSCB is a format-0 DSCB before issuing the write channel program.

Reading or Writing VTOC Index Records: You may read or write VIRs explicitly by using the BUFLIST keyword. You may read them implicitly by using the IXRCDS and MAPRCDS keywords. You may supply a buffer list address in the BUFLIST keyword to read or write one or more VIRS. The buffer list header must have the VIR bit set to one and the DSCB bit set to zero. CVAF inspects each entry in the buffer list (and any chained extensions). If the skip bit is set to zero, the RBA bit is set to one (and the CCHHR and TTR bits are set to zero), and the buffer address is nonzero, CVAF processes the entry. CVAF uses the RBA in the argument field of the buffer list entry to read or write a VIR using the buffer address. CVAF processes read and write requests in the order of their appearance in the buffer list.

Each of the storage areas you provide must be in your program's protect key.

For a write request, CVAF inspects the modification bit in the buffer list entries. If the bit is not set to '1' in any entry, CVAF writes all the entries. CVAF sets the modification bit to zero for entries whose VIR is written.

If you specify the keywords MAPRCDS=YES and/or IXRCDS=KEEP and, at the same time, you do not provide an address in the CVMRCDS/CVIRCDS fields of the CVPL, CVAF reads the map records and the first high-level VTOC index entry record.

Reading Map Records and VIERS: If you want to read the VTOC index map records and first high-level VIER, and retain them in virtual storage, you must code either ACCESS=READ or ACCESS=WRITE. CVAF does not require either the DSN or BUFLIST fields.

If you want to read and retain map records, you must code MAPRCDS=YES. The CVAF parameter list field CVMRCDS must be zero. CVAF obtains a buffer list with the number of entries and buffers required to read all the map VIRs. CVAF puts the buffer list address into the CVMRCDS field.

If you want to read and retain the first high-level VIER and (if this requires an index search) all VIERs read, you must code IXRCDS=KEEP. If the CVAF parameter list field CVIRCDS is zero, CVAF obtains a buffer list with entries and buffers, and reads the first high-level VIER. CVAF determines the number of entries and buffers. If CVIRCDS is not zero, CVAF reads only the VIERs required for an index search.

You can only ensure the integrity of the maps and VIER that CVAF reads if you enqueue the VTOC and (for shared DASD) issue a reserve to the unit.

You must release the map and VIER buffers acquired and retained by CVAF by issuing a subsequent CVAF call.

Releasing Buffers and Buffer Lists Obtained by CVAF: You may release buffers and buffer lists acquired by CVAF in the three following ways:

- To free the MAP records buffer list, code MAPRCDS=NO or MAPRCDS=(NO,addr) specifying any ACCESS.

- To free the index records buffer list, code IXRCDS=NOKEEP or IXRCDS=(NOKEEP,addr) specifying any ACCESS.

- Issue a CVAF call with ACCESS=RLSE, and specify a buffer list address with the BUFLIST keyword.

CVAF frees all eligible buffers and any buffer lists if they become empty. Eligible buffers are those pointed to by buffer list entries with the skip bit off. CVAF frees a buffer list if none of its buffer list entries have the skip bit on. If buffer lists are chained together, CVAF checks and frees all buffer lists if appropriate.

Ensure that you do not request CVAF to release the same buffer list twice by specifying its address in more than one place.

Accessing DSNs or DSCBs in Sequential Order

Each CVAFSEQ call may request the return of one of the following:

- One format-1 or format-4 DSCB in indexed (data-set-name) order.

- One or more DSCBs in physical-sequential order (if you are unauthorized, you can only request one DSCB).

- The next data set name in the index.

CVAF reads the DSCBs into buffers supplied with the BUFLIST keyword. "CVAFSEQ Macro" on page 251 discusses detailed information about the CVAFSEQ VTOC access macro.

Use the buffer list to specify the argument of each DSCB to be read. For indexed access, you must request 96-byte DSCBs in the buffer list. For physical-sequential access, you must request 140-byte DSCBs.

If you select indexed order, CVAF returns each format-1 or format-4 DSCB whose name is in the index. If you want CVAF to return only the data set names in the index (not the DSCBs), specify DSNONLY=YES. In this case, CVAF returns the CCHHR of the DSCB in the argument area supplied through the ARG keyword. CVAF updates the DSN area you specify, with the data set name of each DSCB read, each time you issue CVAFSEQ.

Initiating Indexed Access (DSN Order): To initiate indexed access (DSN order), either supply in the area coded through the DSN keyword 44 bytes of binary zeros (to indicate the first data set name in the index) or supply the data set name you want to serve as the starting place for the index search.

The name that CVAF returns in the DSN area is the one equal to or greater than the DSN supplied, depending on the specification of the ACCESS keyword. CVAF updates the DSN field.

The ACCESS keyword determines whether the search is for a DSN greater than or equal to that which you specify.

If you specify DSNONLY=NO, CVAF returns the DSCB and argument to you, using the buffer list you provide with the BUFLIST keyword. CVAF uses the first entry in the buffer list with the skip bit set to '0' and a nonzero buffer address. You must specify the argument value if you set either the TTR or CCHHR bit in the buffer list entry to '1'. The default is CCHHR. For indexed access, the DSCB size in the buffer list entry must be 96 bytes.

If you specify DSNONLY=YES, you must specify the CCHHR argument in the ARG area.

Note that the data set name of the format-4 DSCB is in the index and that CVAF may return its name (44 bytes of X'04'). The format-4 DSCB's name is likely to be the first data set name in the VTOC index.

Initiating Physical-Sequential Access: To initiate physical-sequential access, you must either specify DSN=0, or not specify the DSN parameter at all. To begin the read, you must initialize the argument field in the first buffer list entry to zero or to the argument of the DSCB. If the argument is zero, CVAF uses the argument of the start of the VTOC.

You must set the DSCB size to 140 in buffer list entries.

The ACCESS= specification determines whether CVAF reads the DSCB whose argument is supplied or the DSCB following it.

For example, to read the first DSCB (the format-4 DSCB) in the VTOC, you may set the BFLEARG in the first buffer list entry to zero and specify ACCESS=GTEQ in the CVAFSEQ macro. If you subsequently specify ACCESS=GT, CVAF reads the second DSCB (the first format-5 DSCB).

If you are authorized, CVAF reads as many DSCBs as there are entries in the buffer list, with a single CVAF call. If you are not authorized, CVAF only reads one DSCB.

CVAF only uses one buffer list. CVAF does not inspect a second buffer list chained from the first. If you are authorized, CVAF uses all entries in the buffer list. CVAF does not inspect the skip bit. Each entry must have a buffer address, the length field set to 140, and the TTR or CCHHR bit set to 1 (if neither bit is set, CVAF sets the CCHHR bit on). If you are unauthorized, CVAF only uses the first entry. CVAF updates the argument field of each buffer list entry with the argument of the DSCB. The argument value is returned in either TTR or CCHHR format, depending on whether you set the TTR or CCHHR bit to 1 in the buffer list entry. The default is CCHHR.

CVAF uses only the argument in the first entry to begin the search. CVAF does not inspect arguments in subsequent entries. If you specify a nonzero argument value in the first entry, a DSCB with that argument must exist.

CVAF indicates an end-of-data condition by providing return code 4 in register 15, and a value of X'20' in the CVSTAT field. CVAF sets the argument fields of all buffer list entries following the last DSCB read, to zero (the first entry is zero if CVAF does not read any DSCBs).

Note that CVAF reads all DSCBs, including format-0 DSCBs. You cannot be certain that you have read all format-1 through format-6 DSCBs until CVAF had read the entire VTOC. For a nonindexed VTOC, the format-4 DSCB field DS4HPCHR contains the CCHHR of the last format-1 DSCB. Format-2 through format-6 DSCBs may reside beyond that location. For an indexed VTOC, the VMDS contains information about which DSCBs are format-0 DSCBs.

Reading Sets of DSCBs with CVAF Filter

The CVAF filter service retrieves sets of DSCBs into buffers provided by the calling program. The following text summarizes this service and its requirements.

- You may invoke the CVAF filter service by issuing the CVAFFILT macro. "CVAFFILT Macro" on page 245 describes the macro's syntax and parameters.

- You request DSCBs by specifying either one or more fully qualified data set names, or one partially qualified name. See "Filter Criteria List" on page 53 and "Examples of Partially Qualified Names for CVAFFILT" on page 250 for further information.

- For each of the qualifying data sets, CVAF Filter returns DSCBs in the order that they are chained in the VTOC: format 1, format 2, then format 3. CVAF does not return DSCBs of other formats.

- CVAF filter service returns complete DSCB chains for one or more qualifying data sets into caller-provided buffers. See "Example of CVAFFILT Macro Sequences" on page 57 and "Example 3: Using the CVAFFILT Macro" on page 271 for further information. CVAF filter service does not return a partial DSCB chain:

  - If you do not provide enough buffers to hold all of the requested DSCBs, CVAF filter service returns one or more complete DSCB chains and/or a status code (CVSTAT in the CVPL). The status code indicates whether or not you may use a "RESUME" CVAF call to retrieve the rest (or more) of the requested DSCBs. See "RESUME Capability" for specific information.

  - If the total number of buffers provided is not sufficient to contain a data set's complete DSCB chain, CVAF filter service sets a status byte (FCLDSNST in the FCL), ignores the data set, and processes the next qualifying data set. You can avoid this situation by providing a minimum of eleven DSCB buffers (enough for a data set at the 123 extent limit).

- You must identify a single DASD volume in the CVAF parameter list (CVPL). CVAF filter service supports both indexed and nonindexed VTOCs.

- When calling CVAF, your program can be in either 24-bit or 31-bit addressing mode. If it is in 31-bit mode, the control blocks shown in Figure 17 on page 53 may reside above the 16Mb line. All these areas must be accessible in your program's storage key.

RESUME Capability: If CVAF filter service terminates before returning a data set's DSCBs because you did not provide enough buffers, CVAF filter service saves the information necessary for a RESUME function in the filter save area (You must specify FLTAREA=KEEP on the initial CVAFFILT call to cause CVAF filter service to obtain and keep the filter save area).

To allow RESUME processing to execute correctly, you *must* maintain the relationship between the requested volume (identified by CVDEB, CVUCB, or a kept IOAREA), your FCL, *and* CVAF's FSA. If you observe this requirement, you can initiate and resume multiple CVAF filter service operations asynchronously on one or more DASD volumes. You can ensure this relationship by providing a unique CVPL and FCL for the duration of the READ/RESUME/RELEASE sequence associated with each logical request.

If you issue an ACCESS=RESUME without having previously specified FLTAREA=KEEP, CVAF filter service provides return code '4' in register 15 and '66' in the CVSTAT field.

If you specify FLTAREA=KEEP, you must issue a subsequent CVAFFILT call with the ACCESS=RLSE keyword to release the filter save area storage.

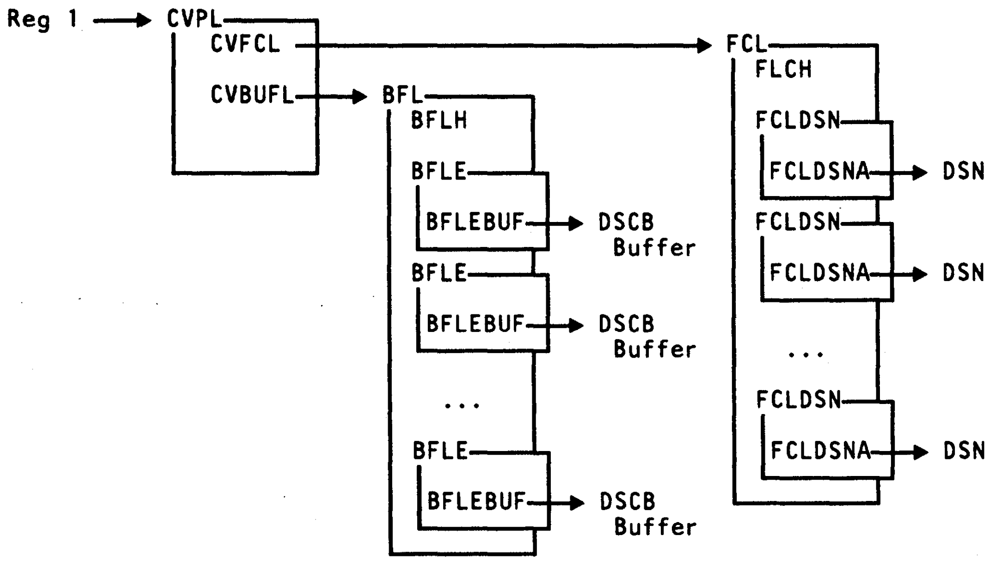


Figure 17. Control Blocks Required for CVAF Filter Services

Filter Criteria List

The filter criteria list consists of two kinds of elements; a list header, and a variable number of list entries. The list entries immediately follow the header, and each entry represents a different data set name to be processed by CVAF filter. The header and entries, shown in Figure 18 and Figure 19 are mapped by the ICVFCL macro. The format of the FCL header is shown in Figure 18.

Name	Offset	Bytes	Description
FCLID	00(X'00')	4	EBCDIC 'FCL '
FCLCOUNT	04(X'04')	2	Number of data set name entries provided in the list.
FCLDSCBR	06(X'06')	2	Number of DSCBs returned
FCL1FLAG	08(X'08')	1	Request flag byte
FCL1LIST		X'80'	List contains fully qualified data set names
FCL1ORDR		X'40'	FCL data set name order requested
		..xx xxxx	Reserved
FCL2FLAG	09(X'09')	1	Status flag byte
FCL2SEQ		X'80'	CVAFFILT executed sequential VTOC access
FCL2SDIR		X'40'	CVAFFILT executed sequential VTOC access, but did at least one direct DSCB read
		..xx xxxx	Reserved
FCLDRSV	10(X'0A')	6	Reserved

Figure 18. Format of a Filter Criteria List (FCL) Header

FCLID

Must be a 4-character EBCDIC constant of 'FCL '.

FCLCOUNT

Specifies the number of data set name entries (FCLDSN) supplied in the list. You must not change this parameter between the initial CVAFFILT call and any subsequent RESUME operations.

- When you specify a partially-qualified data set name, you must specify FCLCOUNT = 1. See "Examples of Partially Qualified Names for CVAFFILT" on page 250 for the format of partially qualified data set names.

- When you specify a list of fully qualified names, CVAFFILT processes only the number of names specified in FCLCOUNT.

FCLDSCBR

Indicates the total number of DSCB entries (including format-1, format-2, and format-3) returned to the caller's buffers by a single CVAFFILT call.

Because CVAF may encounter an error after successfully processing a data set, you may:

1. Initialize FCLDSCBR to 0 before each READ and RESUME call.

2. Upon return from CVAF filter service, process the number of DSCBs indicated by FCLDSCBR,.

3. Then, interpret the CVAF return code and CVSTAT.

FCL1FLAG

Define your request for ACCESS=READ with this flag byte. Any subsequent RESUME requests refer to a copy of these bits in the filter save area (FSA).

FCL1LIST
Set this bit to 1 if you specify a list of fully qualified data set names. Set it to 0 if you specify a single partially qualified data set name.

FCL1ORDR
If you specify that CVAF return DSCB chains in the data set name sequence implied by the placement of the FCLDSN elements, set this bit to 1. Note that:

- If you allow CVAF to determine the sequence of return for format-1 DSCBs, you may realize a performance gain.
- CVAF always returns DSCBs for a given data set in format-1, format-2, format-3 order.
- If you specify a single partially-qualified data set name, CVAF filter does not use this field.

FCL2FLAG
CVAF filter indicates the following status conditions in this byte.

FCL2SEQ
CVAF filter sets this bit to 1 if it determines that its sequential VTOC access path is most efficient. If CVAF filter selects the direct VTOC access path, it sets this field to 0.

FCL2SDIR
CVAF filter sets this bit to 1 if storage limitations within its sequential VTOC access path require direct DSCB reads. CVAF initializes this bit to 0 on each ACCESS=READ and ACCESS=RESUME request. You may test this bit when CVAF filter returns control to you, to determine if you must take some action to relieve the storage limitation.

The format of the FCL entry is shown in Figure 19.

Name	Offset	Bytes	Description
FCLDSN	16(X'10')	8	Data set name information entry
FCLDSNST	00(X'00')	1	Data set name status
		X'00'	Data set name not yet processed
		X'01'	DSCBs returned successfully
		X'02'	Data set name not found
		X'03'	Error in DSCB chain. RESUME function recommended.
		X'04'	Error in CVAFFILT processing. RESUME not recommended.
		X'05'	Insufficient user buffer list elements. RESUME function recommended.
FCLDSNLG	01(X'01')	1	Data set name length
FCL3FLAG	02(X'02')	1	Flag byte
FCL3UPDT		X'80'	This data set name processed during this invocation
		.xxx xxxx	Reserved
FCLDSNRV	03(X'03')	1	Reserved
FCLDSNA	04(X'04')	4	Data set name address

Figure 19. Format of a Filter Criteria List (FCL) Entry

FCLDSN
Contains data set name information. This, and the following fields are repeated in the FCL as a "set," as many times as indicated by the value in FCLCOUNT.

FCLDSNST
Indicates DSCB retrieval status.

- CVAF filter initializes this byte to 0 for ACCESS=READ requests.
- CVAF filter updates this byte after processing this data set name for either ACCESS=READ or ACCESS=RESUME.
- ACCESS=RESUME requests do not process data set names whose FCLDSNST field is non-zero, thus results may be unpredictable if you alter this field.
- For partially-qualified data set name requests, CVAF filter does not post the FCLDSNST field until it has returned all DSCB chains for all qualifying data sets. CVAF filter posts the highest numeric value which applied during the its processing.
- For fully-qualified data set name requests, CVAF filter returns a FCLDSNST byte for each data set name. If the value is greater than 1, CVAF filter has not returned any DSCBs for the associated data set name.

See Figure 19 for the meanings of the possible values in this field.

FCLDSNLG
Indicates length of data set name. You must provide this value.

FCL3FLAG
This is the status flag byte associated with the data set name pointed to by FCLDSNA.

FCL3UPDT
This bit indicates that CVAF filter processed the associated data set name during the current invocation of CVAFFILT.

- When initializing for either a READ or RESUME request, CVAF filter sets this bit to 0.

- When CVAF filter has completed processing for the associated data set name, it sets this bit to 1.

FCL3DSNRV
Reserved, unused.

FCLDSNA
Specifies the address of a fully-qualified data set name, or, if this is the only data set name and FCL1LIST is 0, a partially-qualified data set name. You must provide both this address and the storage area to which it points.

Example of CVAFFILT Macro Sequences: The following example demonstrates the order in which you might issue CVAFFILT macro calls to:

1. Request the DSCBs for a list of data sets.

2. Resume CVAFFILT processing interrupted because of insufficient user buffers.

3. Release the kept filter save area.

The example assumes the following conditions:

- You are an authorized caller (that is, you are specifying a UCB address and IOAREA=KEEP).

- You have initialized a CVAF buffer list as follows:

  - You have specified four buffers.

  - You have defined the buffer list address in your program with the label 'BUFADDR'.

  - You will use the same buffer list for ACCESS=READ and ACCESS=RESUME processing.

- You have initialized a filter criteria list (FCL) as follows:

  - FCLCOUNT = 6 (You are requesting DSCB chains for six data set names.)

  - FCL1LIST = '1'B (The data set names are fully qualified.)

  - FCL1ORDR = '1'B (You want the DSCB chains returned in the order implied by data set name elements in the FCL.)

  - You have initialized each of the six data set name elements such that they form a list requesting SYS1.A, SYS2.B, SYS3.C, SYS4.D, SYS5.E, and SYS6.F respectively.

- The first five data sets have DSCB chain lengths or 1, 5, 2, 3, and 1 respectively on the volume.

- The sixth data set (SYS6.F) is not defined on the volume.

To obtain an initialized CVPL, you could issue the following CVAFFILT macro (list form—does not call CVAF). This example requests the branch entry to CVAF and specifies that the caller is in supervisor state.

```
CVPLIST  CVAFFILT BRANCH=(YES,SUP),MF=L
```

To obtain the first set of DSCB chains, you could issue the following CVAFFILT macro (execute form—calls CVAF). This example specifies that the filter save area is to be kept to allow for ACCESS=RESUME calls. The IOAREA is to be kept for improved efficiency.

```
         CVAFFILT ACCESS=READ,BUFLIST=bufaddr,FCL=fcladdr,
                  UCB=ucbaddr,FLTAREA=KEEP,IOAREA=KEEP,
                  MF=(E,CVPLIST)
```

This CVAFFILT call would return DSCBs as follows:

```
  Buffer     Contents of Buffer

     1       Format-1 DSCB, SYS1.A
     2       Format-1 DSCB, SYS3.C
     3       Format-3 DSCB, SYS3.C
     4       Undefined (unused)
```

CVAF filter would provide return code = 4, CVSTAT = X'40' (RESUME recommended), and FCLDSCBR = 3 (CVAF would return a total of three DSCBs for the two data sets). CVAF would not return DSCBs for data set SYS2.B because its chain contains more DSCBs than the total number of buffers provided. To retrieve SYS2.B's DSCBs, you would have to specify at least five buffers AND execute another ACCESS=READ. (Even though CVAF allows you to specify a different buffer list for each READ OR RESUME, or modify the existing list between READ and RESUME calls, modifying the FCL would result in unpredictable results.) Buffer entry 4 would not have any DSCBs returned, because SYS4.D's DSCB chain size is larger than the number of remaining buffers. The FCL status information would be as follows:

DSN	FCLDSNST	FCL3UPDT	Comments
SYS1.A	1	1	DSCBs returned from this call
SYS2.B	5	1	DSCB chain exceeds total buffers
SYS3.C	1	1	DSCBs returned from this call
SYS4.D	0	0	DSCBs may be returned by RESUME
SYS5.E	0	0	DSCBs may be returned by RESUME
SYS6.F	0	0	DSCBs may be returned by RESUME

Because this CVAFFILT invocation recommends RESUME, and you specified FLTAREA=KEEP, you could use the following execute form of CVAFFILT to obtain more DSCB chains:

```
         CVAFFILT ACCESS=RESUME,MF=(E,CVPLIST)
```

This CVAFFILT call would return DSCBs as follows:

```
  Buffer   Contents of Buffer
     1     Format-1 DSCB, SYS4.D
     2     Format-2 DSCB, SYS4.D
     3     Format-3 DSCB, SYS4.D
     4     Format-1 DSCB, SYS5.E
```

CVAF filter would provide return code = 0, CVSTAT = 0 (request completed), and would have updated the FCL status as follows:

```
  DSN        FCLDSNST    FCL3UPDT    Comments
  SYS1.A        1           0        DSCBs returned from prior call
  SYS2.B        5           0        DSCB chain exceeds total buffers
  SYS3.C        1           0        DSCBs returned from prior call
  SYS4.D        1           1        DSCBs returned from this call
  SYS5.E        1           1        DSCBs returned from this call
  SYS6.F        2           1        Data set name not found
```

FCLDSCBR would contain 4. (This CVAFFILT call returned a total of four DSCBs.) CVAF Filter would not return any DSCBs for SYS6.F, because its format-1 DSCB cannot be found on the volume (FCLDSNST = '2').

Because this status indicates that CVAF Filter has returned all requested DSCBs, and you requested FLTAREA=KEEP and IOAREA=KEEP on the previous call, you should request the RLSE function as follows:

```
         CVAFFILT ACCESS=RLSE,FLTAREA=NOKEEP,IOAREA=NOKEEP,
                  MF=(E,CVPLIST)
```

Obtaining Information from the VTOC Index

You may use ACCESS=MAPDATA to obtain information contained in the space maps. "CVAFDSM Macro" on page 239 discusses detailed information about the CVAFDSM VTOC access macro.

To count the number of unallocated VIRs in the VTOC index space map (VIXM), you must code COUNT=YES and MAP=INDEX. CVAF returns the number of unallocated VIRs in the 4-byte area specified by the CTAREA keyword.

To count the number of format-0 DSCBs, you must code COUNT=YES and MAP=VTOC. CVAF returns the number of format-0 DSCBs in the VTOC map of DSCBs VMDS in the 4-byte area specified by the CTAREA keyword.

To obtain one or more free space extents from the VTOC pack space map (VPSM), you must code COUNT=NO and MAP=VOLUME. CVAF returns the extents in the area specified by the EXTENTS keyword. Each extent is returned in a 5-byte XXYYZ format, the same as for a format-5 DSCB extent, where XX is the relative track address (RTA) of the first track of the extent, YY is the number of whole cylinders in the extent, and Z is the number of additional tracks in the extent. The RTA specified by your program to CVAF in the first (or only) extent serves as a starting point for the VPSM search; the extent returned is the next free extent with a higher starting RTA than the one your program specified.

If all the unallocated extents in the VPSM are supplied before filling in all the extents supplied, the remaining extents are set to zero. CVAF provides return code 4 in register 15 and indicates end-of-data condition by putting a value of X'20' in the CVSTAT field.

Diagnosing VTOC Errors

Actions Taken When an Error Occurs

These actions are taken if an error occurs:

- If an index structure error is detected, DADSM or CVAF causes the VTOC index to be disabled. The indexed VTOC bit is zeroed in the format-4 DSCB. A software error record is written to SYS1.LOGREC. A system dump is taken. The VTOC is converted to a nonindexed format at the next DADSM allocate or extend call.

- If a program check, machine check, or other error occurs while using a VTOC access macro, a SYS1.LOGREC message is written, and a system dump is taken.

- An error code is put in the CVSTAT field of the CVPL. The values and explanations of these error codes are listed in Appendix C, "VTOC Index Error Message and Associated Codes" on page 297.

Recovering from System or User Errors

Because an unauthorized user cannot modify a VTOC, neither the VTOC nor the VTOC index need be recovered from a user error caused by an unauthorized user.

A system error affects a VTOC and VTOC index, probably by interrupting DADSM while it is updating, thus leaving the VTOC and/or the VTOC index in a partially updated state. Both the VTOC and the VTOC index are designed to allow DADSM to recover from such an interruption.

For a nonindexed VTOC (or a VTOC with an index that has been disabled), a subsequent call to DADSM ALLOCATE or EXTEND causes VTOC convert routines to reestablish the free space (format-5 DSCBs).

For an indexed VTOC, a subsequent call to any DADSM function causes the recovery of the previous interrupt (either by backing out or completing the interrupted function).

GTF Trace

A trace function exists to trace all CVAF calls for VTOC index output I/O, all VTOC output I/O, and all VTOC index and space map modifications. For information on this function, see *DFP Diagnosis*.

Listing a VTOC and VTOC Index

You may obtain dump, formatted, or abridged listings of the VTOC and the VTOC index by using the LISTVTOC command of the IEHLIST utility program.

Chapter 2. Executing Your Own Channel Programs (EXCP)

The execute-channel-program (EXCP) macro instruction provides you with complete control of the data organization based on device characteristics. This chapter contains a general description of the function and application of the EXCP macro instruction, accompanied by descriptions of specific control blocks and macro instructions used with EXCP. Factors that affect the operation of EXCP, such as device variations and program modification, are also discussed.

Before reading this chapter, you should be familiar with system functions and with the structure of control blocks, as well as with the operational characteristics of the I/O devices required by your channel programs. Operational characteristics of specific I/O devices are described in IBM publications for each device.

You also need to understand the information in these publications:

- *Data Administration Guide* contains the standard procedures for I/O processing under the operating system.

- *Assembler H Version 2 Application Programming: Guide* contains the information necessary to code programs in the assembler language.

- *Data Administration: Macro Instruction Reference* describes the system macro instructions that can be used in programs coded in the assembler language.

- *Conversion Notebook* describes the factors to consider when converting from MVS/370 at the MVS/SP Version 1 level to MVS/XA.

The execute-channel-program (EXCP) macro instruction causes a supervisor-call interruption to pass control to the EXCP processor. (I/O process is the name we will use for the EXCP processor and the I/O supervisor. For our purposes, it's unnecessary to understand how input/output processing is divided between the two.) EXCP also provides the I/O supervisor with control information regarding a channel program to be executed. When an IBM access method is being used, an access method routine is responsible for issuing EXCP. If you are not using an IBM access method, you must issue EXCP in your program. (The EXCP macro instruction cannot be used to process SYSIN or SYSOUT data sets.)

You issue EXCP primarily for I/O programming situations to which the standard access methods do not apply. If you are writing your own access method, you must include EXCP for I/O operations. EXCP must be used for processing nonstandard labels, including reading and writing labels and positioning magnetic tape volumes.

To issue EXCP, you must provide a channel program (a list of channel command words) and several control blocks in your program area. The I/O process then schedules I/O requests for the device you have specified, executes the specified I/O commands, handles I/O interruptions, directs error recovery procedures, and posts the results of the I/O requests.

Using EXCP in System and Problem Programs

This section explains the procedures performed by the system and the programmer when EXCP is issued by the routines of IBM access methods. The additional procedures you must perform when issuing EXCP yourself are then described by direct comparison.

How the System Uses EXCP

When using an IBM access method to perform I/O operations, the programmer is relieved of coding channel programs and constructing the control blocks necessary for the execution of channel programs. To permit I/O operations to be handled by an access method, the programmer need only issue the following macro instructions:

- A DCB macro instruction that produces a data control block (DCB) for the data set to be retrieved or stored
- An OPEN macro instruction that initializes the data control block and produces a data extent block (DEB) for the data set
- A macro instruction (for example, GET or WRITE) that requests I/O operations

Access method routines will then:

1. Create a channel program that contains channel commands for the I/O operations on the appropriate device
2. Construct an input/output block (IOB) that contains information about the channel program
3. Construct an event control block (ECB) that is later posted with a completion code each time the channel program terminates
4. Issue an EXCP macro instruction to pass the address of the IOB to the routines that initiate and supervise the I/O operations

The I/O process consists of:

5. Constructing a request queue element (RQE) for scheduling the request
6. If the requestor is in a V=V address space, fixing the buffers so that they cannot be paged out and translating the requestor's virtual channel program into a real channel program
7. Issuing a start subchannel (SSCH) instruction to cause the channel to execute the real channel program

8. Processing I/O interruptions and scheduling error recovery procedures when necessary

9. Posting a completion code in the event control block after the channel program has been executed

Note: If the requestor is an authorized program in a V=R address space, a real channel program is provided; thus, item 6 is not performed.

The programmer is not concerned with these procedures and does not know the status of I/O operations until they are completed. Device-dependent operations are limited to those provided by the macro instructions of the particular access method selected.

How To Use EXCP in Problem Programs

To issue the EXCP macro instruction directly, you must follow the procedures that the access methods would perform, as summarized in items 1 through 4 of the preceding discussion. In addition to constructing and opening the data control block with the DCB and OPEN macro instructions, you must construct a channel program, an input/output block, and an event control block before you can issue EXCP. The I/O process generally handles items 5 through 9.

After issuing EXCP, you should issue a WAIT macro instruction, specifying the address of the event control block, to determine whether the channel program has terminated. If volume switching is necessary, you must issue an EOV macro instruction. When all processing of the data set has been completed, you must issue a CLOSE macro instruction to restore the data control block.

All external interfaces for EXCP are compatible between MVS/370 and MVS/XA, except for the restrictions noted below. These restrictions relate only to the support of virtual and real addresses above 16 megabytes.

EXCP will be available to programs executing in either 24-bit or 31-bit addressing mode. However, in order to maintain the required compatibility, the following restrictions apply:

- EXCP will only support a 24-bit virtual storage interface. In addition, all areas related to I/O operations (for example, I/O buffers, channel command words, IOBs, DEBs, appendages, and so forth), must remain 24-bit virtual addressable. EXCP (channel command word translator) will allow 24-bit virtual I/O buffers to be fixed above 16-megabyte real. When a channel command word (CCW) references a real address above 16-megabyte, the CCW translator will build an indirect addressing word (IDAW) for that CCW. Note that this is not supported for format-1 CCWs. All virtual addresses must be below 16-megabyte. For V=R users, CCWs and IDAWs must be below 16-megabyte real.

- Only format-0 CCWs are accepted as input.

- All user-specified appendage routines are given control in 24-bit addressing mode and must return in the same mode.

Note: Access methods run in 24-bit addressing mode. Users running in 31-bit mode must interface to the access methods by using a user-written routine that is resident below 16-megabyte virtual (because the access methods will be able to return control only to a 24-bit addressable location). All addresses, buffers, parameters, control blocks, save areas and exit addresses must be below 16-megabyte virtual. All access methods (except VSAM), for example, GET or PUT, must be called in 24-bit addressing mode.

31-Bit IDAW Requirements

A virtual channel program provided by the EXCP caller may have one or more CCWs with the IDA flag set and the address portion of these CCWs pointing to a single 4-byte IDAW. This EXCP function is referred to as virtual IDAWs.

The 4-byte IDAW can contain a virtual address that ranges from 0 to the maximum 31-bit address. Virtual IDAWs are supported on all virtual CCWs except:

- Transfer in channel (TIC) commands.
- All non data-transfer type commands: for example, recalibrate, rewind, set space, fold, block data check, no operation, control commands.
- Read, read backward, and sense commands, with the skip flag set.

The same addressing restrictions apply to EXCPVR inputs with the exception that 31-bit real data areas may be specified by the user-created CCWs through the use of IDAWs. All CCWs and IDAWs must be below 16-megabyte real.

Only format-0 CCWs are accepted as input.

All other areas related to the EXCP/EXCPVR I/O operation (for example, CCWs, IDAWs, IOBs, DEBs, DCBs, appendages, and so forth) must remain 24-bit addressable.

Note, however, that the EXCP processor will allow both 24-bit and 31-bit virtual I/O buffers to be fixed above 16-megabyte real.

How EXCP Operates in a V=R Address Space

User-constructed channel programs for I/O operations of an authorized program in a V=R address space are not translated. Because the address space is V=R, any CCWs created by the user have correct real data addresses. (Translation would only re-create the user's channel program, so the CCWs are used directly.)

Modification of an active channel program by data read in or by processor instructions is legitimate in a V=R address space, but not in a V=V address space.

EXCP Requirements

This section describes the channel program that you must provide in order to issue EXCP. This section also describes the control blocks that you must either directly construct or cause to be constructed by using macro instructions.

All areas related to the EXCP/EXCPVR I/O operation (for example, CCWs, IDAWs, IOBs, DEBs, DCBs, appendages, and so forth) must remain 24-bit addressable.

Note, however, that the EXCP processor will allow both 24-bit and 31-bit virtual I/O buffers to be fixed above 16-megabyte real.

Channel Program

The channel program supplied by you and executed through EXCP is composed of CCWs on doubleword boundaries. Each channel command word specifies a command to be executed and, for commands initiating data transfer, the area to or from which the data is to be transferred.

Channel command word operation codes used with specific I/O devices can be found in IBM publications for those devices. All channel command word operation codes described in these publications can be used. In addition, both data chaining and command chaining may be used.

To specify either data chaining or command chaining, you must set appropriate bits in the channel command word and indicate the type of chaining in the input/output block. Both data and command chaining should not be specified in the same channel command word; if they are, data chaining takes precedence.

EXCP does not support channel programs that modify themselves, regardless of the method of modification: data chaining, command chaining, or a program to do the modification. The intended modification in virtual storage has no effect on the running real-channel program (see "Modification of a Channel Program during Execution" on page 70).

Control Blocks

When using EXCP, you must be familiar with the function and structure of the IOB, the ECB, the DCB, the DEB, and the IDAW. IOB and ECB fields are illustrated under "Control Block Fields" on page 88. DCB fields are illustrated under "Macro Specifications for Use with EXCP" on page 73. The handling of IDAWs is described under "SIO Appendage" on page 95. Descriptions of these control blocks follow.

Input/Output Block (IOB)

The input/output block is used for communication between the problem program and the system. It provides the addresses of other control blocks, and maintains information about the channel program, such as the type of chaining and the progress of I/O operations. You must define the input/output block and specify its address as the only parameter of the EXCP macro instruction.

Event Control Block (ECB)

The event control block provides you with a completion code that describes whether the channel program was completed with or without error. A WAIT macro instruction, which can be used to synchronize I/O operations with the problem program, must identify the event control block. You must define the event control block and specify its address in the input/output block.

Data Control Block (DCB)

The data control block provides the system with information about the characteristics and processing requirements of a data set to be read or written by the channel program. A data control block must be produced by a DCB macro instruction that includes parameters for EXCP. If you are not using appendages, a short DCB is constructed. Such a DCB does not support reduced error recovery. You specify the address of the data control block in the input/output block.

All DCBs must be located in storage that is not fetch-protected, or, if the task is authorized, in storage that is in the key of the task (TCB KEY).

Data Extent Block (DEB)

The data extent block contains one or more extent entries for the associated data set and other control information. An extent defines all or part of the physical boundaries on an I/O device occupied by, or reserved for, a particular data set. Each extent entry contains the address of a unit control block (UCB) that provides information about the type and location of an I/O device. More than one extent entry can contain the same UCB address. For all I/O devices supported by the operating system, the data extent block is produced during execution of the OPEN macro instruction for the data control block. The system places the address of the data extent block into the data control block. All DEBs must be located in storage that is not fetch-protected, or, if the task is authorized, in storage that is in the key of the task (TCB key). Only authorized tasks (APF-authorized or TCB PKF=0-7) may build DEBs to be used for I/O operations.

How the Channel Program Executes

This section explains how the system uses your channel program and control blocks after you issue EXCP.

Initiation of the Channel Program

By issuing EXCP, you request the execution of the channel program specified in the input/output block. The I/O process validates the request by checking certain fields of the control blocks associated with this request. If the I/O process detects invalid information in a control block, it initiates abnormal termination procedures.

The EXCP processor gets:

- The address of the data control block from the input/output block
- The address of the data extent block from the data control block

- The address of the unit control block from the data extent block

It places the IOB, TCB, DEB, and UCB addresses and other information about the channel program into an area called a request queue element (RQE). (Unless you are providing appendage routines (described under "Appendages" on page 71) you should not be concerned with the contents of RQEs.)

If you have provided an SIO (start I/O) appendage, the EXCP processor now passes control to it. The return address from the SIO appendage determines whether the EXCP processor must:

- Execute the I/O operation normally, or
- Skip the I/O operation.

For a description of the SIO appendage and its linkage to the EXCP processor, see "Appendages" on page 71.

If you are issuing EXCP from a V=V address space, the channel program you construct contains virtual addresses. Because channel subsystems cannot use virtual addresses, the EXCP processor must:

- Translate your virtual channel program into one that uses only real addresses.
- Fix in real storage the pages used as I/O areas for the data transfer operations specified in your channel program.

The EXCP processor builds the translated (real) channel program in a portion of real storage.

For direct access devices, specify the seek address in the input/output block. The I/O supervisor constructs a CCW chain to issue the seek and the set file mask specified in the data extent block, and to pass control to your real channel program.

If your channel program begins with a locate-record CCW, the I/O process builds a define-extent CCW and passes control to your real channel program. (You cannot issue the initial seek, set file mask, or define extent CCWs. The file mask is set to prohibit seek-cylinder CCWs, or, if space is allocated by tracks, seek-head commands. If the data set is open for INPUT, write CCWs are also prohibited.)

For a magnetic tape device, the I/O supervisor constructs a CCW chain to set the mode specified in the data extent block and passes control to your real channel program. (You cannot set the mode yourself.)

If the I/O device is other than a direct access device or a magnetic tape device, the I/O supervisor then places the starting CCW of the channel program into the operation request block (ORB) and issues a start subchannel (SSCH) instruction.

Modification of a Channel Program during Execution

Any problem program that modifies an active channel program with processor instructions or with data read in by an I/O operation must be run in a V=R address space. It cannot run in a V=V address space because of the channel program translation performed by the I/O supervisor. (In a V=V address space, an attempt to modify an active channel program affects only the virtual image of the channel program, not the real channel program being executed by the channel subsystem.)

A program of this type can be changed to run in a V=V address space by issuing another EXCP macro for the modified portion of the channel program.

Completion of Execution

The system considers the channel program completed when it receives an indication of a channel-end condition in the subchannel status word (SCSW). Unless a CHE (channel-end) or ABE (abnormal-end) appendage directs otherwise, the request queue element for the channel program is made available, and a completion code is placed into the event control block. The completion code indicates whether errors are associated with channel end. If device end occurs simultaneously with channel end, errors associated with device end (that is, unit exception or unit check) are also accounted for.

If device end occurs after channel end and if an error is associated with device end, the completion code in the event control block does not indicate the error. However, the status of the unit and channel is saved by the I/O supervisor for the device, and the UCB is marked as intercepted. The input/output block for the next request directed to the I/O device is also marked as intercepted. The error is assumed to be permanent, and the completion code in the event control block for the intercepted request indicates interception. The DCBIFLGS field of the data control block is also flagged to indicate a permanent error. Note that, if a write-tape-mark or erase-long-gap CCW is the last or only CCW in your channel program, the I/O process will not attempt recovery procedures for device end errors. In these circumstances, command chaining a NOP CCW to your write-tape-mark or erase-long-gap CCW ensures initiation of device-end error recovery procedures.

To be prepared for device-end errors, you should be familiar with device characteristics that can cause such errors. After one of your channel programs has terminated, you should not release buffer space until you have determined that your next request for the device has not been intercepted. You may reissue an intercepted request.

Interruption Handling and Error Recovery Procedures

An I/O interruption allows the processor to respond to signals from an I/O device that indicate either termination of a phase of I/O operations or external action on the device. A complete explanation of I/O interruptions is contained in *IBM System/370 Principles of Operation*. For descriptions of interruption by specific devices, see the IBM publications for each device.

If error conditions are associated with an interruption, the I/O supervisor schedules the appropriate device-dependent error routine. The channel subsystem is then restarted with another request that is not related to the channel program in error. (The following paragraphs discuss "related" channel programs.) If the error recovery procedures fail to correct the error, the system places ones in the first two bit positions of the DCBIFLGS field of the data control block. You are informed of the error by an error code in the event control block.

If a channel program depends on the successful completion of a previous channel program (as when one channel program retrieves data to be used in building another), the previous channel program is called a "related" request. Such a request must be identified to the EXCP processor. To find out how to do this, see "Input/Output Block (IOB) Fields" on page 89 and "Purging and Restoring I/O Requests" on page 156.

If a permanent error occurs in the channel program of a related request, the EXCP processor removes the request queue elements for all dependent channel programs from their queue and makes them available.

The related request queue (RRQ) reflects the order in which request queue elements are removed from their queue.

For all requests dependent on the channel program in error, the system places completion codes into the event control blocks. The DCBIFLGS field of the data control block is also flagged. Any requests for a data control block with error flags are posted complete without execution. To reissue requests dependent on the channel program in error, you must reset the first two bits of the DCBIFLGS field of the data control block to zeros. You then reissue EXCP for each channel program desired.

With the IBM 3800 Printing Subsystem, a cancel key or a system-restart-required paper jam causes both a lost data indicator to be set in DCBIFLGS and a lost page count and channel page identifier to be stored in the UCB extension. (See *JES3 Data Areas*, *TSO/E Data Areas*, and *IBM 3800 Printing Subsystem Programmer's Guide*.)

Appendages

The detailed information about appendages that appeared in this section has been moved to *Data Facility Product: Customization.*

An appendage is a user-written routine that provides additional control over I/O operations. By using appendages, you can examine the status of I/O operations and determine the actions to be taken for various conditions.

Channel Programming Considerations

Command retry is a function of the channel supporting the IBM 2305-2, 3330/3333, 3340/3344, 3350, 3375, and 3380 direct access devices. When the channel subsystem receives a retry request, it repeats the execution of the CCW, requiring no additional input/output interrupts. For example, a control unit may initiate a retry procedure to recover from a transient error.

A command retry during the execution of a channel program may cause any of the following conditions to be detected by the initiating program:

- Modifying CCWs: A CCW used in a channel program must not be modified before the CCW operation has been successfully completed. Without the command retry function, a command was fetched only once from storage by a channel. Therefore, a program could determine through condition codes or program controlled interruptions (PCI) that a CCW had been fetched and accepted by the channel. This permitted the CCW to be modified before reexecution. With the command retry function, this cannot be done, because the channel will fetch the CCW from storage again on a command retry sequence. In the case of data chaining, the channel will retry commands starting with the first CCW in the data chain.

- Program Controlled Interrupts (PCI): A CCW containing a PCI flag may cause multiple program-controlled interrupts to occur. This happens if the PCI-flagged CCW was retried during a command retry procedure and if a PCI could be generated each time the CCW is reexecuted.

- Residual Count: If a channel program is prematurely terminated during the retry of a command, the residual count in the channel status word (CSW) will not necessarily indicate how much storage was used. For example, if the control unit detects a "wrong-length record" error condition, an erroneous residual count is stored in the CSW until the command retry is successful. When the retry is successful, the residual in the CSW reflects the correct length of the data transfer.

- Command Address: When data chaining with command retry, the CSW may not indicate how many CCWs have been executed at the time of a PCI. For example:

CCW#	**Channel Program**
1	Read, data chain
2	Read, data chain
3	Read, data chain, PCI
4	Read, command chain

  In this example, assume that the control unit signals command retry on Read #3 and the processor accepts the PCI after the channel resets the command address to Read #1 because of command retry. The CSW stored for the PCI will contain the command address of Read #1 when the channel has actually progressed to Read #3.

- Testing Buffer Contents on Data Read: Any program that tests a buffer to determine when a CCW has been executed and continues to execute based on this data may get incorrect results if an error is detected and the CCW is retried.

Macro Specifications for Use with EXCP

If you are using the EXCP macro instruction, you must also use DCB, OPEN, CLOSE, and, in some cases, the EOV macro instruction. The parameters of the DCB, EOV, and EXCP macro instructions are described here. The parameters and different forms of the OPEN and CLOSE macro instructions are described in *Data Administration: Macro Instruction Reference.* A diagram of the data control block (DCB) is included in this section with the description of the DCB macro instruction.

Defining Data Control Blocks for EXCP (DCB)

The EXCP form of the DCB macro instruction produces a data control block that can be used with the EXCP macro instruction. You must issue a DCB macro instruction for each data set to be processed by your channel programs. (Notation conventions and format illustrations of the DCB macro instruction are given in *Data Administration: Macro Instruction Reference.*) DCB parameters that apply to EXCP may be divided into four categories, depending on the following portions of the data control block that are generated when they are specified:

- Foundation block. This portion is required and is always 12 bytes in length. You must specify two of the parameters in this category.

- EXCP interface. This portion is optional. If you specify any parameter in this category, 20 bytes are generated.

- Foundation block extension and common interface. This portion is optional and is always 20 bytes in length. If this portion is generated, the device-dependent portion is also generated.

- Device dependent. This portion is optional and is generated only if the foundation block extension and common interface portion is generated. Its size ranges from 4 to 20 bytes, depending on specifications in the DEVD parameter. However, if you do not specify the DEVD parameter (and the foundation extension and common interface portion is generated), the maximum 20 bytes for this portion are generated.

Some of the procedures performed by the system when the data control block is opened and closed (such as writing file marks for output data sets on direct access volumes) require information from optional data control block fields. You should make sure that the data control block is large enough to provide all information necessary for the procedures you want the system to handle.

Figure 20 on page 75 shows the relative position of each portion of an opened data control block. The fields corresponding to each parameter of the DCB macro instruction are also designated, with the exception of DDNAME, which is not included in a data control block that has been opened. The fields identified in parentheses represent system information that is not associated with parameters of the DCB macro instruction.

Sources of information for data control block fields other than the DCB macro instruction are data definition (DD) statements, data set labels, and data control block modification routines. You may use any of these sources to specify DCB parameters. However, if a particular portion of the data control block is not generated by the DCB macro instruction, the system does not accept information intended for that portion from any alternative source.

You may provide symbolic names for the fields in one or more EXCP DCBs by coding a DCBD macro to generate a dummy control section (DSECT). To map the common interface, foundation block extension, and foundation block, you code DSORG=XE. To map the foundation block and EXCP interface, you code DSORG=XA. You may code DSORG=(XA,XE) to map both. For further information, see *Data Administration: Macro Instruction Reference.*

Foundation Block Parameters

DDNAME=*symbol*
: The name of the data definition (DD) statement that describes the data set to be processed. This parameter must be given.

MACRF=(E)
: The EXCP macro instruction is to be used in processing the data set. This operand must be coded.

REPOS={Y | N}
: Magnetic tape volumes: This parameter indicates to the dynamic device reconfiguration (DDR) routine whether the user is keeping an accurate block count. If the user is keeping an accurate block count, the DDR routine can attempt to swap the volume. (You must maintain the block count in the DCBBLKCT field.)

  Y—The user is keeping an accurate block count, and the DDR routine can attempt to swap the volume.

  N—The block count is unreliable, and the DDR routine cannot and will not attempt to swap the volume.

  If the operand is omitted, N is assumed.

EXCP Interface Parameters

EOEA=*symbol*
: 2-byte identification of an EOE appendage that you have entered into SYS1.LPALIB or SYS1.SVCLIB.

PCIA=*symbol*
: 2-byte identification of a PCI appendage that you have entered into SYS1.LPALIB or SYS1.SVCLIB.

SIOA=*symbol*
: 2-byte identification of a SIO appendage that you have entered into SYS1.LPALIB or SYS1.SVCLIB.

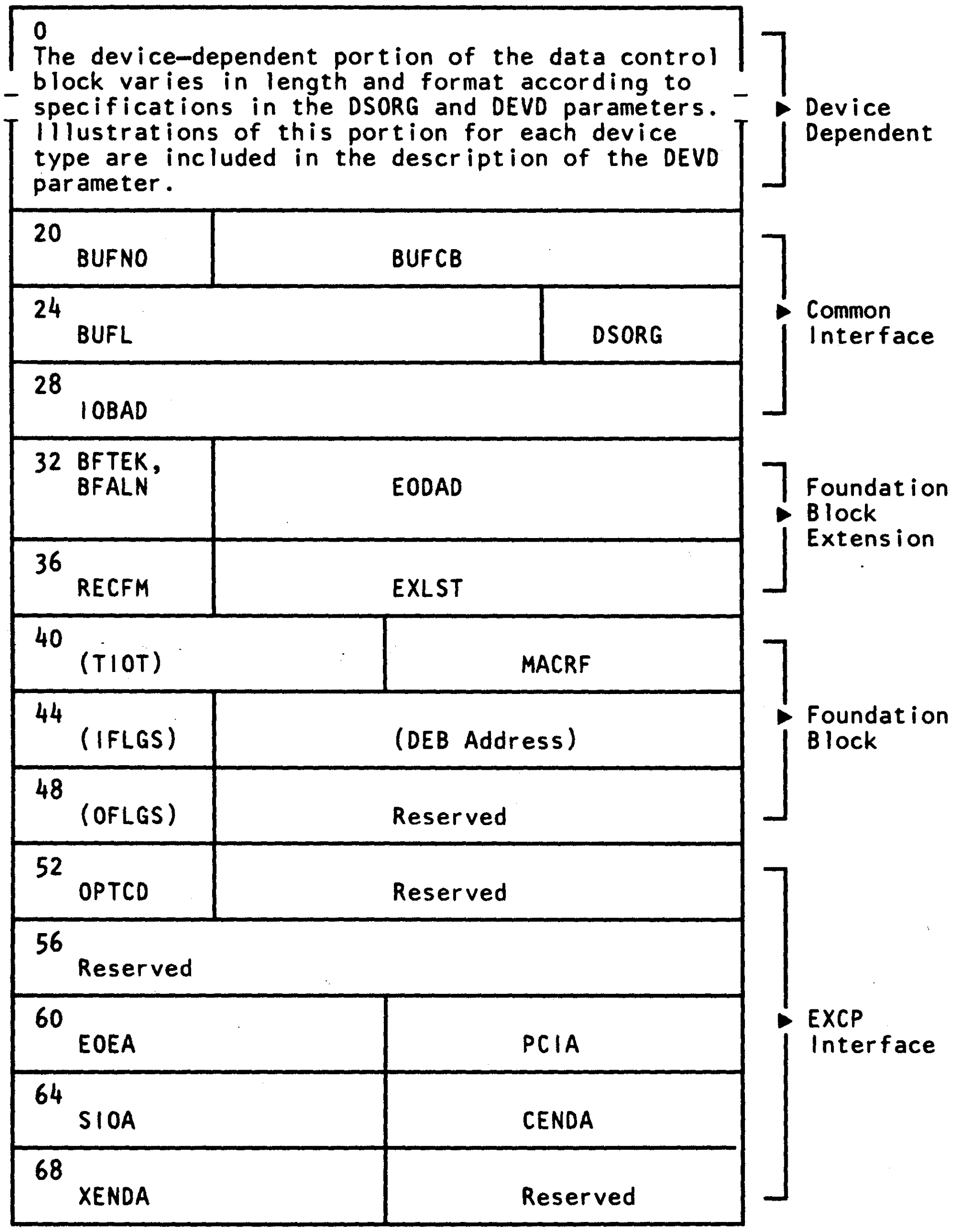

0
The device–dependent portion of the data control block varies in length and format according to specifications in the DSORG and DEVD parameters. Illustrations of this portion for each device type are included in the description of the DEVD parameter.
Device Dependent
20 BUFNO
BUFCB
24 BUFL
DSORG
28 IOBAD
Common Interface
32 BFTEK, BFALN
EODAD
36 RECFM
EXLST
Foundation Block Extension
40 (TIOT)
MACRF
44 (IFLGS)
(DEB Address)
48 (OFLGS)
Reserved
Foundation Block
52 OPTCD
Reserved
56 Reserved
60 EOEA
PCIA
64 SIOA
CENDA
68 XENDA
Reserved
EXCP Interface

Figure 20. Data Control Block (DCB) Format for EXCP (After OPEN)

CENDA=*symbol*
2-byte identification of a CHE appendage that you have entered into SYS1.LPALIB or SYS1.SVCLIB.

XENDA=*symbol*
2-byte identification of an ABE appendage that you have entered into SYS1.LPALIB or SYS1.SVCLIB.

OPTCD=Z

indicates that, for magnetic tape (input only), a reduced error recovery procedure (5 reads only) will occur when a data check is encountered. It should be specified only when the tape is known to contain errors and the application does not require that all records be processed. Its proper use would include error frequency analysis in the SYNAD routine. Specification of this parameter will also cause generation of a foundation block extension. This parameter is ignored unless it was selected at system generation.

IMSK=*value*

Any specification indicates that the system will not use IBM-supplied error routines.

Foundation Block Extension and Common Interface Parameters

EXLST=*address*

the address of an exit list that you have written for exception conditions. The format of the exit list is provided in *Data Facility Product: Customization.*

EODAD=*address*

the address of your end-of-data-set routine for input data sets. If this routine is not available when it is required, the task is abnormally terminated.

DSORG={PS | PO | DA | IS}

the data set organization (one of the following codes). Each code indicates that the format of the device-dependent portion of the data control block is to be similar to that generated for a particular access method:

Code	**DCB Format for**
PS	QSAM or BSAM
PO	BPAM
DA	BDAM
IS	QISAM or BISAM

For direct access devices, if you specify PS or PO, you must maintain the following fields of the device-dependent portion of the data control block so that the system can write a file mark for output data sets:

- The track balance (DCBTRBAL) field that contains a 2-byte binary number that indicates the remaining number of bytes on the current track. This number can be obtained from the system track algorithm routine.

- The full disk address (DCBFDAD) field that indicates the location of the current record. The address is in the form MBBCCHHR.

These fields are written into the format-1 DSCB and are used by Open routines for staging MSS data sets. Staging is done only up through the last cylinder specified by these fields if the data set is reopened for OUTPUT, INOUT, OUTIN, OUTINX, or EXTEND.

If you specify PO for a direct access device, the DCBDIRCT field will not be updated. Therefore, you should be careful when using EXCP with the STOW macro.

IOBAD=*address*

the address of an input/output block (IOB). If a pointer to the current IOB is not required, you may use this field for any purpose.

The following parameters are not used by the EXCP routines. They provide additional information that the system will store for later use by access methods that read or update the data set.

RECFM=*code*

the record format of the data set. (Record format codes are given in *Data Administration: Macro Instruction Reference.*) When writing a data set to be read later, RECFM, LRECL, and BLKSIZE should be specified to identify the data set attributes. LRECL and BLKSIZE can only be specified in a DD statement, because these fields do not exist in a DCB used by EXCP.

BFTEK={S | E}

the buffer technique, either simple or exchange.

BFALN={F | D}

the word boundary alignment of each buffer, either fullword or doubleword.

BUFL=*length*

the length in bytes of each buffer; the maximum length is 32767.

BUFNO=*number*

the number of buffers assigned to the associated data set; the maximum number is 255. See Chapter 10, "Specifying Buffer Numbers for DASD Data Sets" on page 229, for further details and performance considerations.

BUFCB=*address*

the address of a buffer pool control block, that is, the 8-byte field preceding the buffers in a buffer pool.

Device-Dependent Parameters

DEVD=*code*

the device in which the data set may reside. The codes are listed in order of descending space requirements for the data control block:

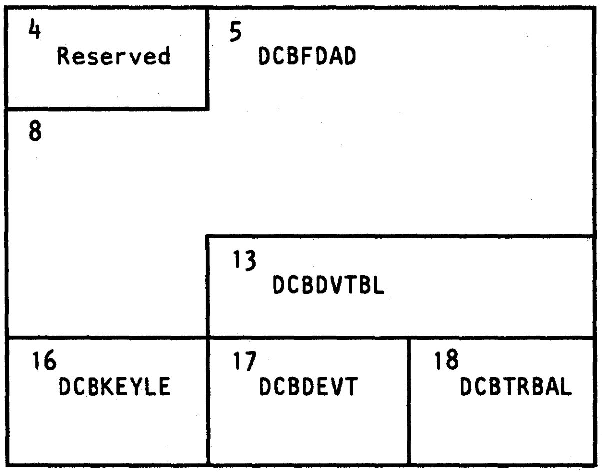
Code	**Device**
DA	Direct access
TA	Magnetic tape
PR	Printer
PC	Card punch
RD	Card reader

Note: For MSS virtual volumes, DA should be used.

If you do not want to select a specific device until job setup time, you should specify the device type requiring the largest area; that is, DEVD=DA.

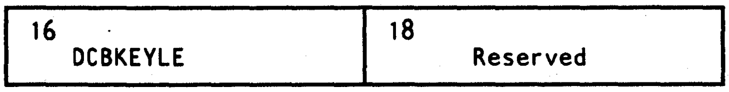
The following diagrams illustrate the device-dependent portion of the data control block for each combination of device type specified in the DEVD parameter and data set organization specified in the DSORG parameter. Fields that correspond to device-dependent parameters in addition to DEVD are indicated by the parameter name. For special services, you may have to maintain the fields shown in parentheses. The special services are explained in the note that follows the diagram.

Device-dependent portion of data control block when DEVD=DA and DSORG=PS:

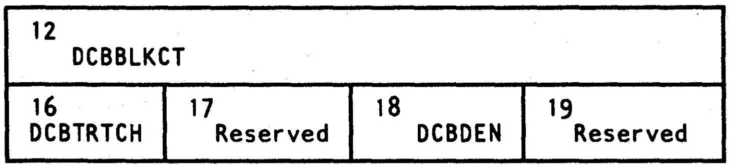


For output data sets, the system uses the contents of the full disk address (DCBFDAD) field, plus one, to write a file mark when the data control block is closed, provided the track balance (DCBTRBAL) field indicates that space is available. If DCBTRBAL is less than 8, the file mark is written on the next sequential track. You must maintain the contents of these two fields yourself if the system is to write a file mark. OPEN will initialize DCBDVTBL and DCBDEVT.

Device-dependent portion of data control block when DEVD=DA and DSORG=DA:



Device-dependent portion of data control block when DEVD=TA and DSORG=PS:



The system uses the contents of the block count (DCBBLKCT) field to write the block count in trailer labels when the data control block is closed or when the EOV macro instruction is issued. You must maintain the contents of this field yourself if the system is to have the correct block count. (**Note**: The I/O supervisor increments this field by the contents of the IOBINCAM field of the IOB at the completion of each I/O request.)

When using EXCP to process a tape data set open at a checkpoint, you must be careful to maintain the correct count; otherwise, the system may position the data set incorrectly when restart occurs. If REPOS=Y, the count must be maintained by you for repositioning during dynamic device reconfiguration.

Device-dependent portion of data control block when DEVD=PR and DSORG=PS:

16 DCBPRTSP	18 Reserved

Device-dependent portion of data control block when DEVD=PC or RD and DSORG=PS:

16 DCBMODE,DCBSTACK	18 Reserved

The following DCB operands pertain to specific devices and may be specified only when the DEVD parameter is specified.

KEYLEN=*length*
for direct access devices, the length in bytes of the key of a physical record, with a maximum value of 255. When a block is read or written, the number of bytes transmitted is the key length plus the record length.

DEN=*value*
for magnetic tape, the tape recording density in bits per inch:

Density

Value	**7-track tape device**	**9-track tape device**
1	556	—
2	800	800 (NZRI)
3	—	1600 (PE)
4	—	6250 (GCR)

NRZI—Non-return-to-zero change to ones recording
PE—phase encoded recording
GCR—group coded recording

If this parameter is omitted, the highest density available on the device is assumed.

TRTCH=*value*
for 7-track magnetic tape, the tape recording technique:

Value	**Tape Recording Technique**
C	Data conversion feature is available.
E	Even parity is used. (If omitted, odd parity is assumed.)
T	BCDIC to EBCDIC translation is required.

MODE=*value*
for a card reader or punch, the mode of operation. Either C (column binary mode) or E (EBCDIC code) may be specified.

STACK=*value*
for a card punch or card reader, the stacker bin to receive cards, either 1 or 2.

PRTSP=*value*
for a printer, the line spacing, either 0, 1, 2, or 3.

DSORG Parameter of the DCBD Macro

In addition to the operands described in *Data Administration: Macro Instruction Reference* for the DSORG parameter of the DCBD macro, you may specify the following operands.

DSORG=

XA	specifies a DCB with the EXCP interface section (including appendage names)
XE	specifies a DCB with the foundation block extension

Initializing Data Control Blocks (OPEN)

The OPEN macro instruction initializes one or more data control blocks so that their associated data sets can be processed. You must issue OPEN for all data control blocks that are to be used by your channel programs. (A dummy data set may not be opened for EXCP.) Some of the procedures performed when OPEN is executed are:

- Reading in the JFCB (job file control block), unless the TYPE=J option of the macro instruction was coded
- Construction of the data extent block (DEB)
- Transfer of information from the JFCB and data set labels to the DCB
- Verification or creation of standard labels
- Tape positioning
- Loading of your appendage routines

The parameters and different forms of the OPEN macro instruction are described in *Data Administration: Macro Instruction Reference*.

If you intend to process a multivolume direct data set, you must cause the open routines to build a data extent block for each volume and issue mount messages for them. This can be done by reading in the JFCB with a RDJFCB macro instruction and opening each volume of the data set. See " Using RDJFCB to Process a Multivolume Direct Data Set" on page 141 for an example of how to code a routine to do this, and "Reading and Modifying a Job File Control Block" on page 136 for further uses of the RDJFCB macro.

Executing a Channel Program (EXCP)

The EXCP macro instruction requests the initiation of the I/O operations of a channel program. You must issue EXCP whenever you want to execute one of your channel programs. The format of the EXCP macro instruction is:

[*symbol*]	**EXCP**	*iob-addr*

iob-addr—A-type address, (2-12), or (1)
: the address of the input/output block of the channel program to be executed.

End of Volume (EOV)

The EOV macro instruction identifies end-of-volume and end-of-data-set conditions. For an end-of-volume condition, EOV causes switching of volumes and verification or creation of standard labels. For an end-of-data-set condition, EOV causes your end-of-data set routine to be entered. Before processing trailer labels on a tape input data set, you must decrement the DCBBLKCT field. Your program issues EOV if switching of magnetic tape or direct access volumes is necessary, or if secondary allocation is to be performed for a direct access data set opened for output.

For magnetic tape, you must issue EOV when either a tapemark is read or a reflective spot is written over. In these cases, bit settings in the 1-byte DCBOFLGS field of the data control block determine the action to be taken when EOV is executed. Before issuing EOV for magnetic tape, you must make sure that appropriate bits are set in DCBOFLGS. Bit positions 2, 3, 6, and 7 of DCBOFLGS are used only by the system; you are concerned with bit positions 0, 1, 4, and 5. The use of these DCBOFLGS bit positions is as follows:

Bit 0
: set to 1 indicates that a write command was executed and that a tapemark is to be written.

Bit 1
: indicates that a backward read was the last I/O operation.

Bit 4
: indicates that data sets of unlike attributes are to be concatenated.

Bit 5
: indicates that a tapemark has been read.

If bits 0 and 5 of DCBOFLGS are both off when EOV is executed, the tape is spaced past a tapemark, and standard labels, if present, are verified on both the old and new volumes. The direction of spacing depends on bit 1. If bit 1 is off, the tape is spaced forward; if bit 1 is on, the tape is backspaced.

If bit 0 is on, but bit 5 is off, when EOV is executed, a tapemark is written immediately following the last data record of the data set. Standard labels, if specified, are created on the old and the new volume.

After issuing EOV for sequentially organized output data sets on direct access volumes, you can determine whether additional space was obtained on the same or a different volume. You do this by examining the data extent block (DEB) and the unit control block (UCB). If neither the address of the UCB, as shown in the DEB, nor the volume serial number, as shown in the UCB, has changed, additional space was obtained on the same volume. Otherwise, space was obtained on a different volume.

The parameters of the EOV macro instruction are:

[*symbol*]	**EOV**	(*dcb addr* | *acb addr*) **[,MODE={<u>24</u> | 31}]**

dcb addr | acb addr—A-type address, (2-12), or (1)
the address of the data control block or access method control block that is opened for the data set. If this parameter is specified as (1), register 1 must contain this address.

MODE={<u>24</u> | 31}
indicates whether the EOV parameter list may reside above or below the 16 megabyte line in virtual storage. The modes are:

24
If you do not specify the **MODE** operand, this mode is assumed. The expansion of the EOV macro generates a parameter list of the standard form (4 bytes per entry). The parameter list *must be below* the 16 megabyte line, but the calling program may be above the line. If your program is in 24-bit mode and you do not use a register to provide the address of the DCB or ACB, the DCB or ACB *must be below* the 16 megabyte line.

31
The expansion of the EOV macro generates a parameter list in the 31-bit addressing mode format (8 bytes per entry). The parameter list may reside above or below the 16 megabyte line. The first byte (byte 0) in each entry contains option information and the last four bytes (bytes 4-7) contain the 4-byte DCB or ACB address. The DCB (and all ACBs except VSAM/VTAM ACBs) *must be below* the 16 megabyte line; therefore, byte 4 *must be zeros.* Bytes 1 through 3 *must also be zeros.*

Note: Failure to provide a DCB below the 16 megabyte line causes an ABEND 50D.

Note: To determine how the system disposes of a tape volume when an EOV macro is issued, see the description of the DISP parameter of the OPEN macro in *Data Administration: Macro Instruction Reference.*

Restoring Data Control Blocks (CLOSE)

The CLOSE macro instruction restores one or more data control blocks so that processing of their associated data sets can be terminated. You must issue CLOSE for all data control blocks that were used by your channel programs. Some of the procedures performed when CLOSE is executed are:

- Release of data extent block (DEB)
- Removal of information transferred to data control block fields when OPEN was executed
- Verification or creation of standard labels
- Volume disposition
- Release of programmer-written appendage routines

When CLOSE is issued for data sets on magnetic tape volumes, labels are processed according to bit settings in the DCBOFLGS field of the data control block. Before issuing CLOSE for magnetic tape, you must set the appropriate bits in DCBOFLGS. The significant DCBOFLGS bit positions are listed in the EOV macro instruction description.

The parameters and different forms of the CLOSE macro instruction are described in *Data Administration: Macro Instruction Reference.*

Assigning an Alternate Track and Copying Data from the Defective Track (ATLAS)

A program that uses the EXCP macro instruction for input and output and that is APF authorized may, during the execution of the program, use the ATLAS macro instruction to obtain an alternate track and to copy a defective track onto the alternate track. With the use of ATLAS, the program can recover from permanent (hard) errors encountered in the execution of the following types of I/O commands:

- Search ID.
- Write. (The error condition must be confirmed during the execution of the channel program by a CCW that checks the data written.)
- Read count. Errors in the CCHHR part of the count area can be recovered from, unless the record is the home address or record zero. Errors in the KDD part of the count area cannot be recovered from, unless the user has identified the defective record.

Note: ATLAS may be used for all direct access devices with the exception of MSS volumes (3330V).

Your DCB must include the DCBRECFM field, and the field must show whether the data set is in the track overflow format. If it is, recovery from errors in last records on tracks depends on your identifying the track overflow record segments.

Recovery takes the form of obtaining a good alternate track and copying the defective track onto the good alternate one. Unless a reexecution of the channel program by ATLAS can correct the defect, the user should examine, and if necessary replace, defective records in a subsequent job if the data set is to be processed again.

The format is:

[*symbol*]	**ATLAS**	**PARMADR={*addr*}** **[,CHANPRG={R̲ | NR}]** **[,CNTPTR={P̲ | F}]** **[,WRITS={YES | NO}]**

PARMADR

Address of a parameter address list of the following format:

0	Address of IOB for the channel program that encountered the error
4	Address of count area field

The count area field contains the CCHHRKDD of a defective record or the CCHH of a track that is to be copied.

addr—A-type address, (2-12), or (1)

CHANPRG={R̲ | NR}

specifies whether the channel program that encountered the error can be executed again.

R	Channel program may be executed again by ATLAS. Before permitting reexecution of the channel program by ATLAS, you must reset the error indications of the previous execution fields in the DCBIFLGS. (See the example of the use of ATLAS below.)
NR	Channel program may not be executed again.

If this parameter is omitted, R is assumed.

CNTPTR

specifies whether the count area field contains a full count area (CCHHRKDD) or a partial count area (CCHH).

P	Part of the count area (the CCHH address of the track to be copied).
F	Full count area (CCHHRKDD count of the record that was found defective).

If this parameter is omitted, P is assumed.

WRITS
track overflow segment identification.

If your data set is in the track overflow format, this identification determines recovery from errors in last records on tracks.

YES	If this is the last record on the track, it is a segment other than the last of a track overflow record.
NO	If this is the last record on the track, it is the last or only segment of a track overflow record.

If this parameter is omitted, it is assumed that it cannot be established whether a last record is a segment of an overflow record.

Using ATLAS

If a channel program encounters a unit check condition (shown in the CSW) in its execution, the EXCP processor program will place the sense bytes in the IOB. ATLAS can be used to recover from sense conditions shown by the following bit settings:

```
IOBSENS0   X'08'    Data check

IOBSENS1   X'80'    Permanent
```

Also, before using ATLAS, you must reset error indications as follows:

```
NI  DCBIFLGS,X'3F'    Reset the DCBIFLGS error indications.
```

The ATLAS program will attempt to find a good alternate track and will attempt to copy the defective track onto the good track, including all error conditions in either key or data areas. The error conditions may be rectified by reexecuting the channel program or through the use of the IEHATLAS utility program in a subsequent step.

Example: The following illustrates the use of the ATLAS macro instruction.

```
         EXCP     MYIOB
         WAIT     ECB=MYECB
         TM       MYECB,X'7F'        TEST FOR I/O ERROR
         BO       NEXT               NO, SUCCESSFUL, GO TO
*                                    ANOTHER ROUTINE
         TM       IOBCSW+3,X'02'     UNIT CHECK
         BZ       OTHER              NO, DO OTHER ERROR
*                                    PROCESSING
         TM       IOBSENS0,X'08'     DATA CHECK
         BNO      OTHER              NO, CAN'T HANDLE
         TM       IOBSENS1,X'80'     PERMANENT
         BNO      OTHER              NO, CAN'T HANDLE
         NI       DCBIFLGS,X'3F'     RESET ERROR
*                                    INDICATORS
         ATLAS    PARMADR=THERE,CHANPRG=R
```

Operation of the ATLAS Program

The ATLAS program (SVC 86):

- Establishes the availability and address of the next alternate track from the format-4 DSCB of the VTOC.
- Brings all count fields from the defective track into storage to establish the description of the track.
- Initializes the alternate track. (Writes the home address and record zero.)
- Brings the key and data areas of each record into storage, one at a time, and combines them with their new count area to write the complete record onto the alternate track.
- When the copying is finished, chains the alternate to the defective track and updates the VTOC.

Control is returned to your program at the next executable instruction following the ATLAS macro instruction.

Return Codes from the ATLAS Program

The success of the ATLAS macro instruction can be determined by examining the contents of register 15, which will contain one of the return codes described below. If register 15 contains decimal 0, 36, 40, or 44, the contents of register 0 may be significant.

Code	Meaning
0(X'00')	Successful completion. Key and data areas have been copied from the defective track onto a good alternate one. The only error encountered was in the record identified by the user's CCHHRKDD value. If the channel program is reexecutable, it has been successfully reexecuted.
4(X'04')	This device type does not have alternate tracks that can be assigned by programming.
8(X'08')	All alternate tracks for the device have been assigned.
12(X'0C')	A request for storage (GETMAIN macro instruction) could not be satisfied.
16(X'10')	All attempts to initialize and transfer data to an alternate track failed. The number of attempts made is equal to 10% of the assigned alternates for the device.
20(X'14')	The type of error shown by the sense byte cannot be handled through the use of the ATLAS macro instruction. The condition is other than a data check (in the count or data areas) or a missing address marker.
24(X'18')	The format-4 DSCB of the VTOC cannot be read; therefore alternate track information is not available to ATLAS.
28(X'1C')	The record specified by the user was the format-4 DSCB, and it could not be read.
32(X'20')	An error found in count area of last record on the track cannot be handled because last-record-on-track identification is not supplied.
36(X'24')	An error was encountered when reading or writing the home address record or record zero. No error recovery has taken place. If register 0 contains X'01 00 00 00', the defect is in record zero.
40(X'28')	Successful completion. Key and data areas have been copied from the defective track onto a good alternate one. However, the alternate track may have records with defective key or data areas. Register 0 identifies the first three found defective as follows: `n R R R` n—The number of record numbers that follow (0, 1, 2, or 3).

Code	Meaning
	R—The hexadecimal number of the record found defective but copied anyway. If the channel program is reexecutable, it has been successfully reexecuted.
44(X'2C')	Errors encountered and no alternate track has been assigned. The return parameter register (register 0) will contain the R of a maximum of three error records. Error conditions that return this code are: • ATLAS received an error indication for a record with a data length in the count field of zero. Recovery was not possible because a distinction cannot be made between an EOF record and an invalid data length. • An error occurred while reading the count field of a record, and the KDD (key length-data length) was found to be defective. • More than three records on the specified track contained errors in their count fields.
48(X'30')	No errors found on the track, no alternate assigned. ATLAS will not assign an alternate unless a track has at least one defective record.
52(X'34')	I/O error in reexecuting user's channel program. A good alternate is chained to the defective track, and data has been transferred. The user's control blocks will give indication of the error condition causing failure in reexecution of the channel program.
56(X'38')	The DCB reflects a track overflow data set, but the UCB device type shows that the device does not support track overflow.
60(X'3C')	The CCHH of the user-specified count area is not within the extents of the data set.
64(X'40')	The device is an MSS virtual device, which is not supported.

Control Block Fields

The fields of the input/output block, event control block, and data extent block are illustrated and explained here; the data control block fields are described with the parameters of the DCB macro instruction under "EXCP Requirements" on page 67.

Input/Output Block (IOB) Fields

The input/output block (IOB) is not automatically constructed by a macro instruction; it must be defined as a series of constants and must be on a fullword boundary. For unit-record and tape devices, the IOB is 32 bytes in length. For direct access, teleprocessing, and graphic devices, 8 additional bytes must be provided. You may want to use the system mapping macro IEZIOB, which expands into a DSECT, to help in constructing an IOB.

In Figure 21 on page 90 the diagonally ruled areas indicate fields in which you must specify information. The other fields are used by the system and must be defined as all zeros. You may not place information into these fields, but you may examine them.

IOBFLAG1 (1 byte)
: You must set bit positions 0, 1, and 6. One-bits in positions 0 and 1 indicate data chaining and command chaining, respectively. (If both data chaining and command chaining are specified, the system does not use error recovery routines except for the direct access devices.) A one-bit in position 6 indicates that the channel program is not a "related" request; that is, the channel program is not related to any other channel program. If you intend to issue an EXCP macro with a BSAM, QSAM, or BPAM data control block, you may want to turn on bit 7 to prevent access-method appendages from processing the I/O request.

IOBFLAG2 (1 byte)
: If you set bit 6 in the IOBFLAG1 field to zero, bits 2 and 3 in this field must then be set to:

- 00, if any channel program or appendage associated with a related request might modify this IOB or channel program.
- 01, if the conditions requiring a 00 setting don't apply, but the CHE or ABE appendage might retry this channel program if it completes normally or with the unit-exception or wrong-length-record bits on in the CSW.
- 10 in all other cases.

The three combinations of bits 2 and 3 represent the three kinds of related requests, known as type 1 (00), type 2 (01), and type 3 (10). The type you use determines how much the EXCP processor can overlap the processing of related requests. Type 3 allows the greatest overlap, normally making it possible to quickly reuse a device after a channel-end interruption. (Related requests that were executed on a pre-MVS system are executed as type-1 requests if not modified.)

IOBSENS0 and IOBSENS1 (2 bytes)
: are placed into the input/output block by the EXCP processor when a unit check occurs. On occasion, the system is unable to obtain any sense bytes because of unit checks when sense commands are issued. In this case, the system simulates sense bytes by moving X'10FE' to IOBSENS0 and IOBSENS1.

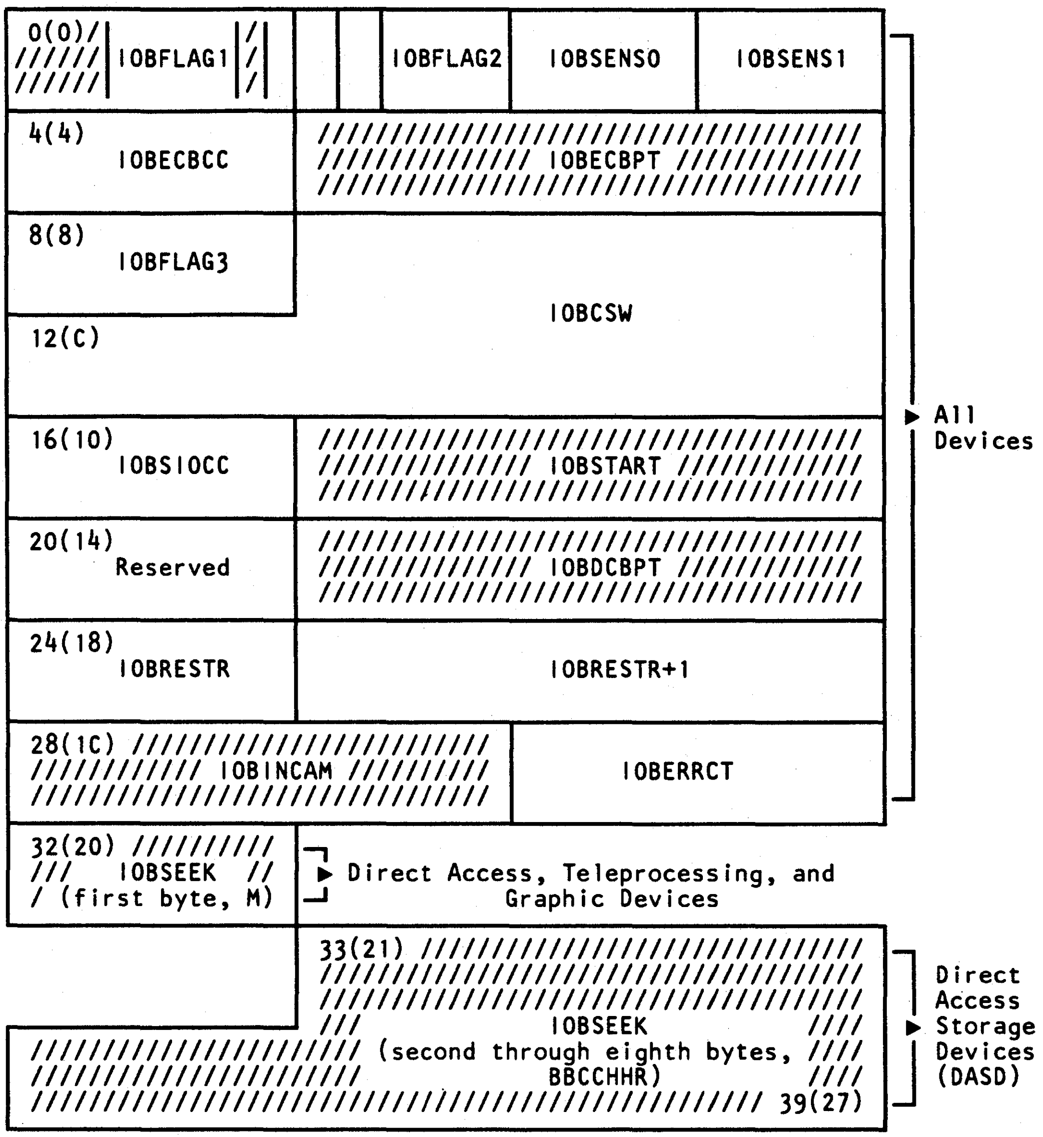


Figure 21. Input/Output Block (IOB) Format

IOBECBCC (1 byte)
: the first byte of the completion code for the channel program. The system places this code in the high-order byte of the event control block when the channel program is posted complete. The completion codes and their meanings are listed under "Event Control Block (ECB) Fields" on page 92.

IOBECBPT (3 bytes)
: the address of the 4-byte event control block you have provided.

IOBFLAG3 (1 byte)
: is used only by the system.

IOBCSW (7 bytes)
: the low-order seven bytes of the channel status word that are placed into this field each time a channel-end or PCI interruption occurs.

IOBSIOCC (1 byte)
: in bits 0 and 1, the instruction-length code; in bits 2 and 3, the start subchannel (SSCH) condition code for the instruction the system issues to start the channel program; and, in bits 4 through 7, the program mask.

IOBSTART (3 bytes)
: the starting address of the channel program to be executed.

Reserved (1 byte)
: used only by the system.

IOBDCBPT (3 bytes)
: the address of the data control block of the data set to be read or written by the channel program.

IOBRESTR (1 byte)
: used by the system for volume repositioning in error recovery procedures.

IOBRESTR+1 (3 bytes)
: if a related channel program is permanently in error, used by the system to chain together IOBs that represent dependent channel programs. To learn more about the conditions under which the chain is built, see "Interruption Handling and Error Recovery Procedures" on page 70.

IOBINCAM (2 bytes)
: for magnetic tape, the amount by which the block count (DCBBLKCT) field in the device-dependent portion of the data control block is to be incremented. You may alter these bytes at any time. For forward operations, these bytes should contain a binary positive integer (usually +1); for backward operations, they should contain a binary negative integer. When these bytes are not used, all zeros must be specified.

Reserved (2 bytes)
: used only by the system.

IOBSEEK (first byte, M)
: for direct access devices, the extent entry in the data extent block that is associated with the channel program (0 indicates the first entry; 1 indicates the second, and so forth). For teleprocessing and graphic devices, it contains the UCB index.

IOBSEEK (last 7 bytes, BBCCHHR)
: for direct access devices, the seek address for your channel program.

Event Control Block (ECB) Fields

You must define an event control block (ECB) as a 4-byte area on a fullword boundary. When the channel program has been completed, the input/output supervisor places a completion code containing status information into the ECB (Figure 22 on page 93). Before examining this information, you must test for the setting of the "complete bit." If the complete bit is not on, and your problem program cannot perform other useful operations, you should issue a WAIT macro instruction that specifies the event control block. Under no circumstances should you construct a program loop that tests for the complete bit.

Data Extent Block (DEB) Fields

The data extent block (DEB) is constructed by the system when an OPEN macro instruction is issued for the data control block. You may not modify the fields of the DEB, but you may examine them.

Executing Fixed Channel Programs in Real Storage (EXCPVR)

The EXCPVR macro instruction provides you with the same functions as the EXCP macro instruction (that is, a device-dependent means of performing input/output operations). In addition, it allows your program to improve the efficiency of the I/O operations in a paging environment by translating its own virtual channel programs to real channel programs. Authorized programs are allowed to execute in a V=V area and provide the EXCP processor with real channel programs. This eliminates the translation of channel programs by the EXCP processor. The program issuing the EXCPVR must remain in authorized state until the completion of the channel programs.

Problem programs are authorized to use the EXCPVR macro instruction under the authorized program facility (APF). A description of how to authorize a program can be found in *Supervisor Services and Macro Instructions.*

[symbol]	**EXCPVR**	*iob-addr*

iob-addr—A-type address, (2-12), or (1)
the address of the input/output block of the channel program to be executed.

To use EXCPVR, you must do all the things you would do to execute an EXCP request; in addition you must:

1. Code PGFX=YES in the DCB associated with the EXCPVR requests and provide a page-fix (PGFX) appendage by specifying SIOA=symbol in the DCB.

2. Fix the data area that contains your channel program, the data areas that are referred to by your channel program, your PCI appendage (if your program can generate program-controlled interrupts), and any area referred to by your PCI appendage. To cause EXCP to fix these data areas, you build a list that

WAIT bit=0	COMPLETE bit=1	Remainder of completion code
bit 0	1	2 … 31

Wait bit
A one bit in this position indicates that the WAIT macro instruction has been issued, but the channel program has not been completed.

Complete bit
A one bit in this position indicates that the channel program has been completed; if it has not been completed, a zero bit is in this position.

Completion code
This code, which includes the wait and complete bits, may be one of the following 4-byte hexadecimal expressions:

CODE	MEANING
7F000000	The channel program has terminated without error.
41000000	The channel program has terminated with a permanent error.
42000000	The channel program has terminated because a direct access extent address has been violated.
44000000	The channel program has been intercepted because of a permanent error associated with a device end for the previous request. You may reissue the EXCP macro instruction to restart the channel program.
48000000	The request queue element for a channel program has been made available after it has been purged.
4B000000	One of the following errors occurred during error recovery processing for a tape device.
	• The CSW command address in the IOB is zeros.
	• An unexpected load point was encountered.
4F000000	Error recovery routines have been entered because of direct access error but are unable to read the home address or record 0.

Figure 22. Event Control Block (ECB) after Posting of Completion Code (EXCP)

contains the addresses of these virtual areas. You should build the list in your PGFX appendage.

3. Determine whether the data areas in virtual storage specified in the address fields of your CCWs cross page boundaries. If they do, you must build an indirect data address list (IDAL) and put the address of the IDAL in the affected CCW.

4. Translate the addresses in your CCWs from virtual to real addresses.

All other areas related to the EXCP/EXCPVR I/O operation (that is, CCWs, IDAWs, IOBs, DEBs, DCBs, appendages, and so forth) must remain 24-bit addressable. Note, however, that the EXCP processor will allow both 24-bit and 31-bit virtual I/O buffers to be fixed above 16 megabytes real.

Items 3 and 4 must be done in your start-I/O (SIO) appendage. A description of the SIO appendage is presented under "Appendages" on page 71.

Building the List of Data Areas to Be Fixed

The EXCP processor expects programs using the EXCPVR macro instruction to pass a list of data areas to be fixed. This list is to be built in the PGFX appendage, as described below.

The data areas you must fix in real storage (if not already fixed in real storage) are:

1. The channel program. If the channel program is already in a fixed subpool, it does not have to be fixed.
2. The data areas to which your channel program will write and from which your channel program will read. If the data areas are already in a fixed subpool, they do not have to be fixed.
3. The PCI appendage, if used, and any areas accessed by the PCI appendage (DEB, IOB, and so forth).

EXCPVR users can specify 31-bit real data areas by creating CCWs through the use of IDAWs.

Page Fix (PGFX) and Start-I/O (SIO) Appendage

This appendage consists essentially of two independent appendages. The complete appendage can be viewed as a re-enterable subroutine having two entry points, one for the SIO appendage and one for the PGFX appendage.

The SIO entry point is located at offset 0 in the subroutine; any other location in the appendage may be branched to from this entry point. The entry point of the PGFX appendage is at offset +4 in the SIO subroutine, which is set in register 15 as the entry point of the PGFX appendage.

Page Fix (PGFX) Appendage: The purpose of this appendage is to list all the areas that must be fixed to prevent paging exceptions during the execution of the current I/O request. This appendage may be entered more than once. However, each time it is entered, it must create the same list of areas to be fixed. The appendage may use the 16-word save area pointed to by register 13. Registers 10, 11, and 13 may be used as work registers.

Page-Fix List Processing

Each page-fix entry placed in the list by the appendage must have the following doubleword format:

0	1 31	32	33 63
0	Starting virtual address of area to be fixed	0	Ending virtual address of area to be fixed + 1

On return from your PGFX appendage to the EXCP processor (via the return address provided in register 14), register 10 must point to the first page-fix entry and register 11 must contain the number of page-fix entries in the work area. The EXCP processor then fixes the pages corresponding to the areas listed by the PGFX appendage. The pages remain fixed until the associated I/O request terminates.

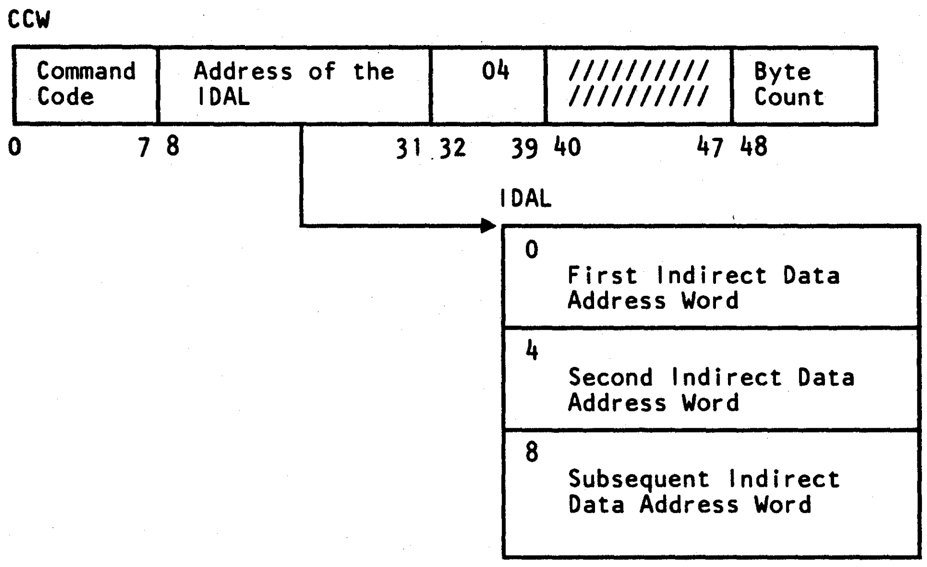
If either the channel end appendage or the abnormal end appendage returns via the return address in register 14 plus 8, the PGFX appendage is not normally reentered. Instead, the SIO appendage is entered, and the page-fix list built by the PGFX appendage is still active. However, the PGFX appendage is entered after either the channel end appendage or the abnormal end appendage returns via the return address in register 14 plus 8 when a PURGE macro has been issued (for instance, when a storage swap has occurred). In this case, when I/O is restored, the PGFX appendage is entered.

Note: The page-fix list must be in page-fixed storage.

SIO Appendage: If you are using EXCPVR to execute your channel program, you must translate the virtual addresses in the operands of your channel program to real addresses. This should be done in your SIO appendage. If indirect data addressing is required, the SIO appendage should also build the indirect data address lists (IDALs) and turn on the IDA indicators in the associated CCWs.

Translating Virtual Addresses and Building the IDAL: You must convert the virtual addresses in the channel program to real addresses. You must also check the areas whose addresses appear in bits 8 through 31 of your CCWs to determine whether the data areas cross 2K-byte boundaries. If they do, you must provide an entry in the IDAL for each 2K-byte boundary crossed. The channel subsystem uses the IDAL to identify the address where it will continue reading or writing when a 2K-byte boundary is crossed during a read or write operation. The IDAL must contain real addresses when it is processed by the channel.

In MVS/XA, the LRA instruction returns a 31-bit real address regardless of the addressing mode. You must be careful when you construct the IDAW to ensure that the real storage obtained by GETMAIN (or branch entry) is below 16 megabytes. Do your page fixing before you issue the LRA. (See *Supervisor Services and Macro Instructions* or *System Macros and Facilities* for information on how to obtain real storage below 16 megabytes real.)


CCW
Command Code
Address of the IDAL
04
//////////
//////////
Byte Count
0
7
8
31
32
39
40
47
48
IDAL
0
First Indirect Data Address Word
4
Second Indirect Data Address Word
8
Subsequent Indirect Data Address Word

Notes:

1. *You must put one entry in the IDAL for each 2K-byte page boundary your data area crosses.*

2. *If the CCW has an IDAL address rather than a data address, bit 37 must be set to signal this to the channel.*

3. *The maximum number of entries needed in the IDAL is determined from the count in the CCW as follows:*

   Number of IDAL entries=((CCW byte-count – 1)/2048) + 1.
   (Round up division to next highest integer if remainder is not zero.)

The number of IDAL entries required ultimately depends on the number of 2K-byte boundaries crossed by the data. For example, if your data is 800 bytes long and does not cross a 2K-byte page boundary, no IDAL entries are required. If your data crosses a 4K-byte page boundary, then two IDAL entries are required. If your data is 5000 bytes long, at least two IDAL entries are required. If your data crosses two 4K-byte page boundaries, four IDAL entries are required.

The first indirect address is the real address of the first byte of the data area. The second and subsequent indirect addresses are the real addresses of the second and subsequent 2K-byte boundaries of the data area.

For example, if the data area real address is X'707FF' and the byte count is X'1802', the IDAL would contain the following real addresses (assuming the real addresses are contiguous, which may not always be the case):

```
707FF
70800
71000
```

If the data area real address is X'707FF' and the byte count is X'800', the IDAL would contain the following addresses:

```
707FF
70800
```

Chapter 3. Reading from and Writing to Direct Access Devices (XDAP)

Execute direct access program (XDAP) is a macro instruction that you may use to read, verify, or update a block on a direct access volume. This chapter explains what the XDAP macro instruction does and how you can use it. The control block generated when XDAP is issued and the macro instructions used with XDAP are also discussed.

Because most of the specifications for XDAP are similar to those for the execute channel program (EXCP) macro instruction, you should be familiar with the "Executing Your Own Channel Programs (EXCP)" chapter of this publication, and with the information contained in *Data Administration Guide* that provides how-to information for using the access method routines of the system control program.

If you are not using the standard IBM data access methods, you can, by issuing XDAP, generate the control information and channel program necessary for reading or updating the records of a data set. (However, XDAP cannot be used to read, verify, or update a SYSIN or SYSOUT data set.)

You cannot use XDAP to add blocks to a data set, but you can use it to change the keys of existing blocks. Any block configuration and any data set organization can be read or updated.

Although the use of XDAP requires less storage than do the standard access methods, it does not provide many of the control program services that are included in the access methods. For example, when XDAP is issued, the system does not block or unblock records and does not verify block length.

To issue XDAP, you must provide the actual track address of the track containing the block to be processed. You must also provide either the block identification or the key of the block, and specify which of these is to be used to locate the block. If a block is located by identification, both the key and data portions of the block may be read or updated. If a block is located by key, only the data portion can be processed.

For additional control over I/O operations, you may write appendages that must be entered into the LPA library. Descriptions of these routines and their coding specifications are included under Chapter 2, "Executing Your Own Channel Programs (EXCP)" on page 63.

XDAP Requirements

When using the XDAP macro instruction, you must, somewhere in your program, code a DCB macro instruction that produces a data control block (DCB) for the data set to be read or updated. You must also code an OPEN macro instruction that initializes the data control block and produces a data extent block (DEB). The OPEN macro instruction must be executed before any XDAP macro instructions are executed.

When the XDAP macro instruction is assembled, a control block and executable code are generated. This control block may be logically divided into three sections:

- An event control block (ECB) that is supplied with a completion code each time the direct access channel program is terminated.
- An input/output block (IOB) that contains information about the direct access channel program.
- A direct access channel program that consists of three or four channel command words (CCWs). The type of channel program generated depends on specifications in the parameters of the XDAP macro instruction. When executed, it locates a block by either its actual address or its key and reads, updates, or verifies the block.

When the channel program has terminated, a completion code is placed into the event control block. After issuing XDAP, you should therefore issue a WAIT macro instruction, specifying the address of the event control block, to regain control when the direct access program has terminated. If volume switching is necessary, you must issue an EOV macro instruction. When processing of the data set has been completed, you must issue a CLOSE macro instruction to restore the data control block.

Macro Specifications for Use with XDAP

When you are using the XDAP macro instruction, you must also code DCB, OPEN, CLOSE, WAIT, and, in some cases, the EOV macro instructions. The parameters of the XDAP macro instruction are listed and described here. For the other required macro instructions, special requirements or options are explained, but you should see "Macro Specifications for Use with EXCP" on page 73 for listings of their parameters.

Defining a Data Control Block (DCB)

You must issue a DCB macro instruction for each data set to be read, updated, or verified by the direct access channel program. To learn which macro instruction parameters to code, see "Defining Data Control Blocks for EXCP (DCB)" on page 73.

Initializing a Data Control Block (OPEN)

The OPEN macro instruction initializes one or more data control blocks so that their associated data sets can be processed. You must issue OPEN for all data control blocks that are to be used by the direct access program. Some of the procedures performed when OPEN is executed are:

- Construction of data extent block (DEB)
- Transfer of information from DD statements and data set labels to the data control block
- Verification or creation of standard labels
- Loading of programmer-written appendage routines

The two parameters of the OPEN macro instruction are the address(es) of the data control block(s) to be initialized and the intended method of I/O processing of the data set. The method of processing may be specified as INPUT, OUTPUT, or EXTEND; however, if nothing is specified, INPUT is assumed. The parameters and different forms of the OPEN macro instruction are described in *Data Administration: Macro Instruction Reference.*

Executing Direct Access Programs (XDAP)

The XDAP macro instruction produces the XDAP control block (that is, the ECB, IOB, and channel program) and executes the direct access channel program. The format of the XDAP macro instruction is:

[*symbol*]	**XDAP**	*ecb-symbol* *,type* *,dcb-addr* *,area-addr* *,length-value* **,[(***key-addr,keylength-value***)]** *,blkref-addr* **,[***sector-addr***]** **[,MF={E | L}]**

ecb-symbol—symbol or (2-12)
the symbolic name to be assigned to the XDAP event control block. Registers can be used only with MF=E.

type—**{RI | RK | WI | WK | VI | VK}**
the type of I/O operation intended for the data set and the method by which blocks of the data set are to be located. One of the combinations shown must be coded in this field.

The codes and their meanings are:

R	Read a block.

W	Update a block.
V	Verify that the device is able to read the contents of a block, but do not transfer data.
I	Locate a block by identification. (The key portion, if present, and the data portion of the block are read, updated, or verified.)
K	Locate a block by key. (Only the data portion of the block is read, updated, or verified.) If you code this value, you must code the 'key-addr,keylength-value' operands.

dcb-addr—A-type address or (2-12)
the address of the data control block for the data set. If this data control block is also being used by a sequential access method (BSAM, BPAM, QSAM), you must reassemble the XDAP macro instruction. Otherwise, sequential access method appendages will be called at the conclusion of the XDAP channel program.

area-addr—A-type address or (2-12)
the address of an input or output area for a block of the data set.

length-value—absexp or (2-12)
the number of bytes to be transferred to or from the input or output area. If blocks are to be located by identification and the data set contains keys, the value must include the length of the key. The maximum number of bytes transferred is 32767.

key-addr—RX-type address or (2-12)
when blocks are to be located by key, the address of a virtual storage field that contains the key of the block to be read, updated, or verified.

keylength-value—absexp or (2-12)
when blocks are to be located by key, the length of the key. The maximum length is 255 bytes.

blkref-addr—RX-type address or (2-12)
the address of a field in virtual storage containing the actual track address of the track containing the block to be located. The actual address of a block is in the form **MBBCCHHR**, where **M** indicates which extent entry in the data extent block is associated with the direct access program; **BB** is not used, but must be zero; **CC** indicates the cylinder address; **HH** indicates the actual track address; and **R** indicates the block identification. **R** is not used when blocks are to be located by key. (For more detailed information, see "Converting a Relative Track Address to an Actual Track Address" on page 106.)

sector-addr—RX-type address or (2-12)
the address of a 1-byte field containing a sector value. The sector-address parameter is used for rotational position sensing (RPS) devices only. The parameter is optional, but its use will improve channel performance. When the parameter is coded, a set-sector CCW (using the sector value indicated by the data address field) precedes the search-ID-equal command in the channel program. The sector-address parameter is ignored if the type parameter is coded as RK, WK, or VK. If a sector address is specified in the

execute form of the macro, then a sector address, not necessarily the same, must be specified in the list form. The sector address in the executable form will be used.

Note: No validity check is made on either the address or the sector value when the XDAP macro is issued. However, a unit check/command reject interruption will occur during channel-program execution if the sector value is invalid for the device or if the sector-addr operand is used when accessing a device without RPS. (For more detailed information, see "Obtaining Sector Number of a Block on a Device with the RPS Feature" on page 108.)

MF=
you may use the L-form of the XDAP macro instruction for a macro expansion consisting of only a parameter list, or the E-form for a macro expansion consisting of only executable instructions.

MF=E
The first operand (ecb-symbol) is required and may be coded as a symbol or supplied in registers 2 through 12. The type, dcb-addr, area-addr, and length-value operands may be supplied in either the L- or E-form. The blkref-addr operand may be supplied in the E-form or moved into the IOBSEEK field of the IOB by you. The sector-addr is optional; it may be coded either in both the L- and E-form or in neither.

MF=L
The first two operands (ecb-symbol and type) are required and must be coded as symbols. If you choose to code length-value or keylength-value, they must be absolute expressions. Other operands, if coded, must be A-type addresses. (blkref-addr is ignored if coded.)

Note: The XDAP macro builds a channel program containing A-type addresses. These addresses refer to storage within the L-form of the macro. If you copy the L-form of the macro to a workarea so that the program may be reentrant, the E-form of the XDAP macro does not update the A-type addresses. This results in an invalid channel program.

The dcb-addr, area-addr, blkref-addr, and sector-value operands may be coded as RX-type addresses or supplied in registers 2 through 12. The length-value and keylength-value operands can be specified as absolute expressions or decimal integers or supplied in registers 2 through 12.

End of Volume (EOV)

The EOV macro instruction identifies end-of-volume and end-of-data-set conditions. For an end-of-volume condition, EOV causes switching of volumes and verification or creation of standard labels. For an end-of-data-set condition, EOV causes your end-of-data-set routine to be entered. When using XDAP, you issue EOV if switching of direct access volumes is necessary or if secondary allocation is to be performed for a direct access data set opened for output.

For details about the parameters of the EOV macro instruction, see "End of Volume (EOV)" on page 81.

Restoring a Data Control Block (CLOSE)

The CLOSE macro instruction restores one or more data control blocks so that processing of their associated data sets can be terminated. You must issue CLOSE for all data sets that were used by the direct access channel program. Some of the procedures performed when CLOSE is executed are:

- Release of data extent block (DEB)
- Removal of information transferred to data control block fields when OPEN was executed
- Verification or creation of standard labels
- Release of programmer-written appendage routines

The CLOSE macro instruction must identify the address of at least one data control block to be restored, and may specify other parameters. The parameters and different forms of the CLOSE macro instruction are described in *Data Administration: Macro Instruction Reference.*

Control Blocks Used with XDAP

The three control blocks generated during execution of the XDAP macro instruction are described here.

Event Control Block (ECB)

The event control block (ECB) begins on a fullword boundary and occupies the first 4 bytes of the XDAP control block. Each time the direct access channel program terminates, the I/O supervisor places a completion code containing status information into the event control block (Figure 23 on page 105). Before examining this information, you must wait for the completion of the channel program by issuing a WAIT macro instruction that specifies the address of the event control block.

Input/Output Block (IOB)

The input/output block (IOB) is 40 bytes in length and immediately follows the event control block. "Control Block Fields" on page 88 contains a diagram of the input/output block (Figure 23 on page 105). You may want to examine the IOBSENS0, IOBSENS1, and IOBCSW fields if the ECB is posted with X'41'.

WAIT bit	COMPLETE bit	Completion code	
bit 0	1	2	31

Wait bit
A one bit in this position indicates that the direct access channel program has not been completed.

Complete bit
A one bit in this position indicates that the channel program has been completed; if it has not been completed, a zero bit is in this position.

Completion code
This code, including the wait and complete bits, may be one of the following 4-byte hexadecimal expressions:

CODE	MEANING
7F000000	Direct access program has terminated without error.
41000000	Direct access program has terminated with permanent error.
42000000	Direct access program has terminated because a direct access extent address has been violated.
4F000000	Error recovery routines have been entered because of direct access error but are unable to read home address or record 0.

Figure 23. Event Control Block (ECB) after Posting of Completion Code (XDAP)

Direct Access Channel Program

The direct access channel program is 24 bytes in length (except when set sector is used for RPS devices) and immediately follows the input/output block. Depending on the type of I/O operation specified in the XDAP macro instruction, one of four channel programs may be generated. The three channel command words for each of the four possible channel programs are shown in Figure 24 on page 106.

When a sector address is specified with an RI, VI, or WI operation, the channel program is 32 bytes in length. Each of these channel programs in Figure 24 on page 106 would be, in this case, preceded by a set sector command.

Converting a Relative Track Address to an Actual Track Address

To issue XDAP, you must provide the actual track address of the track containing the block to be processed. If you know only the relative track address, you can convert it to the actual address by using a resident system routine. The entry point to this conversion routine is labeled IECPCNVT. The address of the entry point (CVTPCNVT) is in the communication vector table (CVT). The address of the CVT is in location 16.

Type of I/O Operation	CCW	Command Code
Read by identification	1	Search ID Equal
	2	Transfer in Channel
Verify by identification[1]	3	Read Key and Data
Read by key	1	Search Key Equal
	2	Transfer in Channel
Verify by key[1]	3	Read Data
Write by identification	1	Search ID Equal
	2	Transfer in Channel
	3	Write Key and Data
Write by key	1	Search Key Equal
	2	Transfer in Channel
	3	Write Data

[1] For verifying operations, the third CCW is flagged to suppress the transfer of information to virtual storage.

Figure 24. The XDAP Channel Programs

The conversion routine does all its work in general registers. You must load registers 0, 1, 2, 14, and 15 with input to the routine. Register usage is as follows:

Register	Use
0	Must be loaded with a 4-byte value of the form TTRN, where TT is the track number relative to the beginning of the data set, R is the the block identification on that track, and N is the concatenation number of a BPAM data set. (0 indicates the first data set in the concatenation, an nonconcatenated BPAM data set, or a non-BPAM data set.)
1	Must be loaded with the address of the data extent block (DEB) of the data set.
2	Must be loaded with the address of an 8-byte area that is to receive the actual address of the block to be processed. The converted address is of the form MBBCCHHR, where M indicates which extent entry in the data extent block is associated with the direct access program (0 indicates the first extent, 1 indicates the second, and so forth); BB is two bytes of zeros; CC is the cylinder address; HH is the actual track address; and R is the block number.

3-8	Are not used by the conversion routine.
9-13	Are used by the conversion routine and are not restored.
14	Must be loaded with the address to which control is to be returned after execution of the conversion routine.
15	Is used by the conversion routine as a base register and must be loaded with the address where the conversion routine is to receive control.

Return Codes from the Conversion Routine

When control is returned to your program, register 15 will contain one of the following return codes:

Code	Meaning
0(X'00')	Successful conversion.
4(X'04')	The relative block address converts to an actual track address outside the extents defined in the DEB.

Converting an Actual Track Address to a Relative Track Address

To get the relative track address when you know the actual track address, you can use the conversion routine labeled IECPRLTV. The address of the entry point (CVTPRLTV) is in the communication vector table (CVT). The address of the CVT is in location 16.

The conversion routine does all its work in general registers. You must load registers 1, 2, 14, and 15 with input to the routine. Register usage is as follows:

Register	Use
0	Will be loaded with the resulting TTR0 to be passed back to the caller.
1	Must be loaded with the address of the data extent block (DEB) of the data set.
2	Must be loaded with the address of an 8-byte area containing the actual address to be converted to a TTR. The actual address is of the form MBBCCHHR.
3-8	Are not used by the conversion routine.
9-13	Are used by the conversion routine and are not restored.

14	Must be loaded with the address to which control is to be returned after execution of the conversion routine.
15	Is used by the conversion routine as a base register and must be loaded with the address where the conversion routine is to receive control.

Obtaining Sector Number of a Block on a Device with the RPS Feature

To obtain the performance improvement given by rotational position sensing, you should specify the sector-addr parameter in the XDAP macro. For programs that can be used with both RPS and non-RPS devices, the UCBRPS bit (bit 3 at an offset of 17 bytes into the UCB) should be tested to determine whether the device has rotational position sensing. If the UCBRPS bit is off, a channel program with a "set sector" command must not be issued to the device.

The sector-addr parameter on the XDAP macro specifies the address of a 1-byte field in your region. You must store the sector number of the block to be located in this field. You can obtain the sector number of the block by using a resident conversion routine, IECOSCR1. The address of this routine is in field CVTOSCR1 of the CVT, and the address of the CVT is in location 16. The routine should be invoked via a BALR 14,15 instruction. If you are passing the track balance to the routine, invoke the routine using a BAL 14,8(15). If you are computing the sector value on modulo devices (3375 and 3380) with variable length records, you must pass the track balance to the sector convert routine.

For RPS devices, the conversion routine does all its work in general registers. You must load registers 0, 2, 14, and 15 with input to the routine. Register usage is as follows:

Register	Use
0	For fixed, standard blocks or fixed, unblocked records not in a partitioned data set: Register 0 must be loaded with a 4-byte value in the form XXKR, where XX is a 2-byte field containing the physical block size, K is a 1-byte field containing the key length, and R is a 1-byte field containing the number of the record for which a sector value is desired. The high-order bit of register 0 must be turned off (set to 0) to indicate fixed-length records. Passing the track balance: Register 0 must be loaded with the 4-byte value of the track balance of the record preceding the required record. For all other cases: Register 0 must be loaded with a 4-byte value in the form BBIR, where BB is the total number of key and data bytes on the track up to, but not including, the target record; I is a 1-byte key indicator (1 for keyed records, 0 for records without keys); and R is a 1-byte field containing the number of the record for which a sector value is desired. The high-order bit of register 0 must be turned on (set to 1) to indicate variable-length records.
1	Not used by the sector-convert routine.
2	Must be loaded with a 4-byte field where the first byte is the UCB device type code for the device (obtainable from UCB+19), and the remaining three bytes are the address of a 1-byte area that is to receive the sector value.
3-8	Not used.
9-11	Used by the convert routine and are not saved or restored.
12,13	Not used.
14	Must be loaded with the address in which control is to be returned after execution of the sector conversion routine.
15	Used by the conversion routine as a base register and must be loaded with the address of the entry point to the conversion routine.

Chapter 4. Password Protecting Data Sets

The password protection described in this chapter does not apply to VSAM data sets. Information about VSAM data set protection is in *VSAM Administration: Macro Instruction Reference* and *Access Method Services Reference*. (For information on RACF and its relationship to password protection, see *RACF General Information*.) To use the data set protection feature of the operating system, you must create and maintain a PASSWORD data set consisting of records that associate the names of the protected data sets with the password assigned to each data set. There are four ways to maintain the PASSWORD data set:

- You can write your own routines.
- You can use the PROTECT macro instruction.
- You can use the utility control statements of the IEHPROGM utility program.
- If you have TSO, you can use the TSO PROTECT command.

This chapter discusses only the first two of the four ways: It provides technical detail about the PASSWORD data set that is necessary for writing your own routines, and it describes how to use the PROTECT macro instruction. (The last two of the four ways are discussed in other publications, as indicated in the list of publications below.)

Before using the information in this chapter, you should be familiar with information in several related publications. The following publications are recommended:

- *Data Administration Guide* contains a general description of the data set protection feature.
- *System Messages* contains a description of the operator messages and replies associated with the data set protection feature.
- *JCL Reference* contains a description of the data definition (DD) statement parameter used to indicate that a data set is to be password protected.
- *Utilities* contains a description of how to maintain the PASSWORD data set using the utility control statements of the IEHPROGM utility program.
- *TSO Command Language Reference* describes the use of the TSO PROTECT command.

Providing Data Set Security

In addition to the usual label protection that prevents the opening of a data set without the correct data set name, the operating system provides data set security options that prevent unauthorized access to confidential data. Password protection prevents access to data sets until a correct password is entered by the system operator, or, for TSO, by a remote terminal operator.

The following are the types of access allowed to password-protected data sets:

- PWREAD/PWWRITE—A password is required for read or write.
- PWREAD/NOWRITE—A password is required for read. Writing is not allowed.
- NOPWREAD/PWWRITE—Reading is allowed without a password. A password is required to write.

To prepare for use of the data set protection feature of the operating system, you place a sequential data set, named PASSWORD, on the system residence volume. This data set must contain at least one record for each data set placed under protection. In turn, each record contains a data set name, a password for that data set, a counter field, a protection mode indicator, and a field for recording any information you desire to log. On the system residence volume, these records are formatted as a "key area" (data set name and password) and a "data area" (counter field, protection mode indicator, and logging field). The data set is searched on the "key area."

Note: The area allocated to the data set should not have been previously used for a PASSWORD data set, as this may cause unpredictable results when adding records to the data set.

You can write routines to create and maintain the PASSWORD data set. If you use the PROTECT macro instruction to maintain the PASSWORD data set, see "Maintaining the PASSWORD Data Set (PROTECT Macro)" on page 116. If you use the IEHPROGM utility program to maintain the PASSWORD data set, see *Utilities*. These routines may be placed in your own library or in the system's library (SYS1.LINKLIB). You may use a data management access method or EXCP programming to read from and write to the PASSWORD data set.

If a data set is to be placed under protection, it must have a protection indicator set in its label (format-1 DSCB or header 1 tape label). This is done by the operating system when the data set is created, by the IEHPROGM utility program, or by the PROTECT macro when creating or adding the control password. The protection indicator is set in response to a value in the LABEL= operand of the DD statement associated with the data set being placed under protection. *JCL Reference* describes the LABEL operand.

Note: Data sets on magnetic tape are protected only when standard labels are used.

Password-protected data sets can only be accessed by programs that can supply the correct password. When the operating system receives a request to open a protected data set, it first checks to see whether the data set has already been opened for this job step. If so, only the access mode will be checked to determine whether it is compatible with the protection mode under which it was previously opened. If the data set has not been previously opened by this job step or if the access mode is not compatible with the protection mode under which it was previously opened, a message is issued that asks for the password; the message goes to the operator console. If the program requesting that the data set be opened is running under TSO in the foreground, the message goes to the TSO terminal operator. If you want the password supplied by another method in your installation, you can modify the READPSWD source module or code a new routine to replace READPSWD in SYS1.LPALIB.

PASSWORD Data Set Characteristics

The PASSWORD data set must reside on the same volume as your operating system. The space you allocate to the PASSWORD data set must be contiguous, that is, its DSCB must indicate only one extent. The amount of space you allocate depends on the number of data sets your installation wants to protect. Each entry in the PASSWORD data set requires 132 bytes of space. The organization of the PASSWORD data set is physical-sequential; the record format is unblocked, fixed-length records (RECFM=F). Each record that forms the data area is 80 bytes long (LRECL=80,BLKSIZE=80) and is preceded by a 52-byte key (KEYLEN=52). The key area contains the fully qualified data set name of as many as 44 bytes and a password of one to eight bytes, left justified with blanks added to fill the areas. The password assigned may be from one to eight alphameric characters in length.

Note: For data sets on magnetic tape designed according to the specifications of the International Organization for Standardization (ISO) 1001-1979, the equivalent American National Standards Institute (ANSI) X3.27-1978, or the Federal Information Processing Standards (FIPS) 79, do not include generation and version numbers as part of generation data set names. The generation and version numbers are not included as part of the names in the tape labels and are ignored if included in the PASSWORD data set.

You can protect the PASSWORD data set itself by creating a password record for it when your program initially builds the data set. Thereafter, the PASSWORD data set cannot be opened (except by the operating system routines that scan the data set) unless the operator enters the password.

Note: If a problem occurs on a password-protected system data set, maintenance personnel must be provided with the password in order to access the data set and resolve the problem.

Creating Protected Data Sets

A data definition (DD) statement parameter (LABEL=) may be used to indicate that a data set is to be password protected. For data sets on DASD, an alternative method is to use the PROTECT macro instruction for a previously allocated data set. A data set may be created and the protection indicator set in its label without entering a password record for it in the PASSWORD data set.

Operating procedures at your installation must ensure that password records for all data sets currently password-protected are entered in the PASSWORD data set. Installations where independent computing systems share common DASD resources must ensure that PASSWORD data sets on all systems contain the appropriate password records for any protected data set on shared DASD.

Under certain circumstances, the order in which data sets are allocated and unallocated from multiple systems on shared DASD may result in loss of password protection. For example, if an unprotected data set is allocated and opened by a user on System A and then scratched by a different user on System B, the first user is given a "window" to the unallocated (free) area. If any data set, protected or unprotected, is allocated in that space by a user on either system during the time the "window" is open, the new data set has no protection from the user with the "window."

While the allocation disposition is still NEW, a password-protected data set can be used without supplying a password. However, once the data set is unallocated, any subsequent attempt to open will result in termination of the program unless the password record is available and the correct password is supplied. Note that, if the protection mode is NOPWREAD and the request is to open the data set for input or read backward, no password will be required.

Tape Volumes Containing More Than One Password-Protected Data Set

To password protect a data set on a tape volume containing other data sets, you must password protect all the data sets on the volume. (Standard labels—SL, SUL, AL, or AUL—are required. For definitions of these label types and the protection-mode indicators that can be used, see *Magnetic Tape Labels and File Structure Administration.*)

If you issue an OPEN macro instruction to create a data set following an existing, password-protected data set, the password of the existing data set will be verified during open processing for the new data set. The password supplied must be associated with a PWWRITE protection-mode indicator.

Protection Feature Operating Characteristics

The topics that follow provide information concerning actions of the protection feature in relation to termination of processing, volume switching, data set concatenation, SCRATCH and RENAME functions, and counter maintenance.

Termination of Processing

Processing is terminated when:

1. The operator cannot supply the correct password for the protected data set being opened after two tries.

2. A password record does not exist in the PASSWORD data set for the protected data set being opened.

3. The protection-mode indicator in the password record and the method of I/O processing specified in the Open routine do not agree, for example, OUTPUT specified against a read-only protection-mode indicator.

4. There is a mismatch in data set names for a data set involved in a volume switching operation. This is discussed in the next paragraph.

Volume Switching

The system ensures a continuation of password protection when volumes of a multivolume data set are switched. It accepts a newly-mounted tape volume to be used for input or a newly-mounted direct access volume, regardless of its use, if these conditions are met:

- The data set name in the password record for the data set is the same as the data set name in the JFCB. (This ensures that the problem program has not changed the data set name in the JFCB since the data set was opened.)
- The protection-mode indicator in the password record is compatible with the processing mode, and a valid password has been supplied.

The system accepts a newly-mounted tape volume to be used for output under any of these conditions:

- The security indicator in the HDR1 label indicates password protection; the data set name in the password record is the same as the data set name in the JFCB; and the protection-mode indicator is compatible with the processing mode. (If the data set name in the JFCB has been changed, a new password is requested from the operator.)
- The security indicator in the HDR1 label does not indicate password protection. (A new label will be written with the security indicator indicating password protection.)
- Only a volume label exists. (A HDR1 label will be written with the security indicator indicating password protection.)

Data Set Concatenation

A password is requested for every protected data set that is involved in a concatenation of data sets, regardless of whether the other data sets involved are protected or not.

SCRATCH and RENAME Functions

To delete or rename a protected data set, it is necessary that the job step making the request be able to supply the password. The system first checks to see if the job step is currently authorized to write to the data set. If not, message IEC301A is issued to request the password. The password provided must be associated with a "WRITE" protection-mode indicator.

Counter Maintenance

The operating system increments the counter in the password record on each usage, but no overflow indication will be given (overflow after 65535 openings). You must provide a counter maintenance routine to check and, if necessary, reset this counter.

Maintaining the PASSWORD Data Set (PROTECT Macro)

To use the PROTECT macro instruction, your PASSWORD data set must be on the system residence volume. The PROTECT macro can be used to:

- Add an entry to the PASSWORD data set.
- Replace an entry in the PASSWORD data set.
- Delete an entry from the PASSWORD data set.
- Provide a list of information about an entry in the PASSWORD data set; this list will contain the security counter, access type, and the 77 bytes of security information in the "data area" of the entry.

In addition, the PROTECT macro updates the DSCB of a protected direct access data set to reflect its protection status; this feature eliminates the need for you to use job control language when you protect a data set.

PASSWORD Data Set Characteristics and Record Format (With PROTECT macro)

When you use the PROTECT macro, the record format and characteristics of the PASSWORD data set are no different from the record format and characteristics that apply when you use your own routines to maintain it.

Number of Records for Each Protected Data Set

When you use the PROTECT macro, the PASSWORD data set must contain at least one record for each protected data set. The password (the last 8 bytes of the "key area") that you assign when you protect the data set for the first time is called the control password. In addition, you may create as many secondary records for the same protected data set as you need. The passwords assigned to these additional records are called secondary passwords. This feature is helpful if you want several users to have access to the same protected data set, but you also want to control the way they can use it. For example: One user could be assigned a password that allowed the data set to be read and written, and another user could be assigned a password that allowed the data set to be read only.

Note: The PROTECT macro will update the protection-mode indicator in the format-1 DSCB in the protected data set only when you issue it for adding, replacing, or deleting a control password.

Protection-Mode Indicator

You can set the protection-mode indicator (third data byte) in the password record to one of four different values:

- X'00' to indicate that the password is a secondary password and the protected data set is to be read only (PWREAD).
- X'80' to indicate that the password is the control password and the protected data set is to be read only (PWREAD).

- X'01' to indicate that the password is a secondary password and the protected data set is to be read and written (PWREAD/PWWRITE).

- X'81' to indicate that the password is the control password and the protected data set is to be read and written (PWREAD/PWWRITE).

Because of the sequence in which the protection status of a data set is checked, the following defaults will occur:

If control password is:	Secondary password must be:
1. PWREAD/PWWRITE or PWREAD/NOWRITE	PWREAD/PWWRITE or PWREAD/NOWRITE
2. NOPWREAD/PWWRITE	NOPWREAD/PWWRITE

If the control password is set to either of the settings in item 1 above, the secondary password will be set to PWREAD/PWWRITE if you try to set it to NOPWREAD/PWWRITE.

If the control password is changed from either of the settings in item 1 to the setting in item 2 above, the secondary password will be automatically reset to NOPWREAD/PWWRITE.

If the control password is changed from the setting in item 2 to either of the settings in item 1 above, the secondary password is set by the system to PWREAD/PWWRITE.

Because the DSCB of the protected data set is updated only when the control password is changed, you may request protection attributes for secondary passwords that conflict with the protection attributes of the control password.

PROTECT Macro Syntax

The format is:

[*symbol*]	**PROTECT**	*parameter list address*

parameter list address—A-type address, (2-12), or (1)
indicates the location of the parameter list. The parameter list must be set up before the PROTECT macro is issued. The address of the parameter list may be passed in register 1, in any of registers 2 through 12, or as an A-type address. The first byte of the parameter list must be used to identify the function (add, replace, delete, or list) you want to perform. See Figure 25 on page 118 through Figure 28 on page 123 for the parameter lists and codes used to identify the functions.

Note: The parameter lists and the areas addressed by the list must reside below 16 megabytes virtual.

PROTECT Macro Parameter Lists

0 X'01'	13 Control password pointer
1 00 00 00	16 Number of volumes
4 Data set name length	17 Volume list pointer
5 Data set name pointer	20 Protection code
8 00	21 New password pointer
9 00 00 00	24 String length
12 00	25 String pointer

Notes:

0 X'01'
Entry code indicating ADD function.

4 Data set name length.

5 Data set name pointer.

13 Control password pointer.
The control password is the password assigned when the data set was placed under protection for the first time. The pointer can be 3 bytes of binary zeros if the new password is the control password.

16 Number of volumes.
If the data set is not cataloged and you want to have it flagged as protected, you must specify the number of volumes in this field. A zero indicates that the catalog information should be used.

17 Volume list pointer.
If the data set is not cataloged and you want to have it flagged as protected, you provide the address of a list of volume serial numbers in this field. Zeros indicate that the catalog information should be used.

20 Protection code.
A one-byte number indicating the type of protection: X'00' indicates default protection (for the ADD function; the default protection is the type of protection specified in the control password record of the data set); X'01' indicates that the data set is to be read and written; X'02' indicates that the data set is to be read only; and X'03' indicates that the data set can be read without a password, but a password is needed to write into it. The PROTECT macro will use the protection code value, specified in the parameter list, to set the protection-mode indicator in the password record.

Figure 25 (Part 1 of 2). Parameter List for ADD Function

21 New password pointer.
If the data set is being placed under protection for the first time, the new password becomes the control password. If you are adding a secondary entry, the new password is different from the control password.

24 String length.
The length of the character string (maximum 77 bytes) that you want to place in the optional information field of the password record. If you don't want to add information, set this field to zero.

25 String pointer.
The address of the character string that is going to be put in the optional information field. If you don't want to add additional information, set this field to zero.

Figure 25 (Part 2 of 2). Parameter List for ADD Function

0 X'02'	13 Control password pointer
1 00 00 00	16 Number of volumes
4 Data set name length	17 Volume list pointer
5 Data set name pointer	20 Protection code
8 00	21 New password pointer
9 Current password pointer	24 String length
12 00	25 String pointer

Notes:

0 X'02'.
Entry code indicating REPLACE function.

4 Data set name length.

5 Data set name pointer.

9 Pointer to current password.
The address of the password that is going to be replaced.

13 Control password pointer.
The address of the password assigned to the data set when it was first placed under protection. The pointer can be set to 3 bytes of binary zeros if the current password is the control password.

16 Number of volumes.
If the data set is not cataloged and you want to have it flagged as protected, you have to specify the number of volumes in this field. A zero indicates that the catalog information should be used.

17 Volume list pointer.
If the data set is not cataloged and you want to have it flagged as protected, you have to provide the address of a list of volume serial numbers in this field. If this field is zero, the catalog information will be used.

20 Protection code.
A one-byte number indicating the type of protection: X'00' indicates that the protection is default protection (for the REPLACE function the default protection is the protection specified in the current password record of the data set); X'01' indicates that the data set is to be read and written; X'02' indicates that the data set is to be read only; and X'03' indicates that the data set can be read without a password, but a password is needed to write into the data set.

Figure 26 (Part 1 of 2). Parameter List for REPLACE Function

21 New password pointer.
The address of the password that you want to replace the current password.

24 String length.
The length of the character string (maximum 77 bytes) that you want to place in the optional information field of the password record. Set this field to zero if you don't want to add additional information.

25 String pointer.
The address of the character string that is going to be put in the optional information field of the password record. Set the address to zero if you don't want to add additional information.

Figure 26 (Part 2 of 2). Parameter List for REPLACE Function

0 X'03'	9 Current password pointer
1 00 00 00	12 00
4 Data set name length	13 Control password pointer
5 Data set name pointer	16 Number of volumes
8 00	17 Volume list pointer

Notes:

0 X03'.
Entry code indicating DELETE function.

4 Data set name length.

5 Data set name pointer.

9 Current password pointer.
The address of the password that you want to delete. You can delete either a control entry or a secondary entry.

13 Control password pointer.
The address of the password assigned to the data set when it was placed under protection for the first time. The pointer can be 2 bytes of binary zeros if the current password is also the control password.

16 Number of volumes.
If the data set is not cataloged and you want to have it flagged as protected, you must specify the number of volumes in this field. A zero indicates that the catalog information should be used.

17 Volume list pointer.
If the data set is not cataloged and you want to have it flagged as protected, you must provide the address of a list of volume serial numbers in this field. If this field is zero, the catalog information will be used.

Figure 27. Parameter List for DELETE Function

0 X'04'	5 Data set name pointer
1 80-byte buffer pointer	8 00
4 Data set name length	9 Current password pointer

Notes:

0 X'04'.
Entry code indicating LIST function.

1 80-byte buffer pointer.
The address of a buffer where the list of information can be returned to your program by the macro instruction.

4 Data set name length.

5 Data set name pointer.

9 Current password pointer.
The address of the password of the record that you want listed.

Figure 28. Parameter List for LIST Function

Return Codes from the PROTECT Macro

When the PROTECT macro finishes processing, register 15 contains one of the following return codes:

Code	Meaning
0(X'00')	The updating of the PASSWORD data set was successfully completed.
4(X'04')	The PASSWORD of the data set name was already in the password data set.
8(X'08')	The password of the data set name was not in the PASSWORD data set.
12(X'0C')	A control password is required or the one supplied is incorrect.
16(X'10')	The supplied parameter list was incomplete or incorrect.
20(X'14')	There was an I/O error in the PASSWORD data set.
24(X'18')[1]	The PASSWORD data set was full.
28(X'1C')	The validity check of the buffer address failed.
32(X'20')[2]	The LOCATE macro failed. LOCATE's return code is in register 1, and the number of indexes searched is in register 0.
36(X'24')[2]	The OBTAIN macro failed. OBTAIN's return code is in register 1.
40(X'28')[2]	The DSCB could not be updated.
44(X'2C')	The PASSWORD data set does not exist.
48(X'30')[2]	Tape data set cannot be protected.
52(X'32')[2]	Data set in use.
56(X'38')[2]	The data set uses the virtual storage access method (VSAM).

Notes:

1 For this return code, a message is written to the console indicating that the PASSWORD data set is full.

2 For this return code, the PASSWORD data set has been updated, but the DSCB has not been flagged to indicate the protected status of the data set.

Chapter 5. Exit Routines

The detailed information about installation-written exit modules has been moved to *Data Facility Product: Customization.*

This chapter discussed how exit modules can:

- Take control before and after direct access device storage management (DADSM) processing
- Take control during Open for a DCB
- Determine whether a missing data set control block (such as for a data set that has been moved to another volume) can be restored to a volume
- Recover from errors that may occur during the opening, closing, or handling of an end-of-volume condition for a data set associated with the user's task
- Bypass, limit, or override system-calculated values that assist you in selecting optimum DASD data set block size/CI size.

Chapter 6. System Macro Instructions

This chapter describes miscellaneous macro instructions that allow you to:

- Modify control blocks (RDJFCB macro)
- Obtain information from control blocks and system tables (DEVTYPE macro)
- Perform track capacity calculations (TRKCALC macro)
- Allocate a data set based on a partial DSCB (REALLOC macro)
- Load a message display on an IBM 3480 Magnetic Tape Subsystem (MSGDISP macro)

Before reading this chapter, you should be familiar with the following publication:

- *Assembler H Version 2 Application Programming: Guide* contains the information necessary to code programs in the assembler language.

Introduction

The system macro instructions are described in these functional groupings:

- Mapping (IEFUCBOB, IEFJFCBN, and CVT)
- Obtaining device characteristics (DEVTYPE)
- Manipulating the JFCB (RDJFCB)
- Data security (DEBCHK)
- Manipulating queues (PURGE and RESTORE)
- Performing track capacity calculations (TRKCALC)
- Allocating a DASD data set (REALLOC)
- Releasing unused space from a DASD data set (PARTREL)
- Loading a message display on an IBM 3480 Magnetic Tape Subsystem (MSGDISP)

Mapping System Data Areas

The IEFUCBOB, IEFJFCBN, and CVT macro instructions are used as DSECT expansions that define the symbolic names of fields within the unit control block (UCB), job file control block (JFCB), and communication vector table (CVT), respectively.

The CVT, IEFUCBOB, and IEFJFCBN macro definitions are in a distribution library named SYS1.AMODGEN. Before you can issue the macros, you must copy them from SYS1.AMODGEN into SYS1.MACLIB (the IEBCOPY utility can be used to copy the macros), or SYS1.AMODGEN may be concatenated to the macro library before reference is made to SYS1.AMODGEN.

IEFUCBOB—Mapping the UCB

This macro instruction defines the symbolic names of the fields in the unit control block (UCB). The macro does not include a DSECT statement. However, if you specify PREFIX=YES, the DSECT statement is provided.

The format is:

[*symbol*]	**IEFUCBOB**	**[LIST={<u>NO</u> | YES}]** **[,PREFIX={<u>NO</u> | YES}]**

LIST={<u>NO</u> | YES}

NO

~~specifies that only the UCB prolog is to be printed.~~

YES

specifies that the UCB prolog and the rest of the UCB are to be printed.

PREFIX={<u>NO</u> | YES}

NO

specifies that no prefix is to be printed.

YES

specifies that the prefix and main body of the UCB are to be printed. A DSECT statement is included if you specify PREFIX=YES.

IEFJFCBN—Mapping the JFCB

This macro instruction defines the symbolic names of the fields in the job file control block (JFCB). The macro does not include a DSECT statement. If you require one, code a DSECT statement before the macro statement.

The format is:

[*symbol*]	IEFJFCBN	[LIST={<u>NO</u> | YES}]

LIST={<u>NO</u> | YES}

NO
: specifies that only the JFCB prolog is to be printed.

YES
: specifies that the JFCB prolog and the rest of the JFCB are to be printed.

CVT—Mapping the CVT

This macro instruction defines the symbolic names of all fields in the communication vector table (CVT).

The format is:

[*symbol*]	CVT	[DSECT={<u>NO</u> | YES}] [,LIST={<u>NO</u> | YES}]

DSECT={<u>NO</u> | YES}

NO
: specifies that you do not want a DSECT.

YES
: specifies that you want a DSECT.

LIST={<u>NO</u> | YES}

NO
: specifies that only the CVT prolog is to be printed.

YES
: specifies that the CVT prolog and the rest of the CVT are to be printed.

Obtaining I/O Device Characteristics

Use the DEVTYPE macro instruction to request information relating to the characteristics of an I/O device, and to cause this information to be placed into a specified area. (The results of a DEVTYPE macro instruction executed before a checkpoint is taken should not be considered valid after a checkpoint/restart occurs.) The IHADVA macro maps the data returned by the DEVTYPE macro.

The topics that follow discuss the DEVTYPE macro, device characteristics, and particular output for particular devices.

DEVTYPE Macro Specification

The format is:

[*symbol*]	**DEVTYPE**	*ddloc-addrx* *,area-addrx* **[,DEVTAB]** **[,RPS]**

For the UCBLIST function, the format is:

[*symbol*]	**DEVTYPE**	*(area-addrx,area-size)* **,UCBLIST=***(ucbl-addr,ucbl-num)*

ddloc-addrx
: the name of an 8-byte field that contains the symbolic name of the DD statement to which the device is assigned. The name must be left justified in the 8-byte field, and must be followed by blanks if the name is fewer than eight characters. The doubleword need not be on a doubleword boundary.

area-addrx
: the name of an area into which the device information is to be placed. If your program does not specify the UCBLIST function, the area can be two, five, or six fullwords long, depending on whether or not you specify the DEVTAB and RPS operands. If your program specifies the UCBLIST function, the area must be 6 fullwords long. The area must be on a fullword boundary.

area-size
: the size of the area into which the device information is to be placed.

DEVTAB
: This operand is only required for direct access devices. If DEVTAB is specified, the following number of words of information is placed in your area:

- For direct access devices: 5 words
- For non-direct access devices: 2 words

If you do not code DEVTAB, one word of information is placed in your area if the reference is to a graphics or teleprocessing device; for any other type of device, two words of information are placed in your area.

RPS
: If RPS is specified, DEVTAB must also be specified. The RPS parameter causes one additional full word of RPS information to be included with the DEVTAB information.

UCBLIST
: UCBLIST provides a list service in which the caller passes a list and count of the addresses of UCBs. The information returned is always given in 6-word entries (one entry per UCB address) regardless of the device type. The

words that would contain information not applicable to the device for that entry are not altered.

Note: Any reference for a DUMMY data set in the DEVTYPE macro instruction will cause eight bytes of zeros to be placed in the output area. Any reference to a SYSIN or SYSOUT data set causes X'00000102' to be placed in word 0 and 32760 (X'00007FF8') to be placed in word 1 in the output area. Any reference to a file allocated to a TSO terminal causes X'00000101' to be placed in word 0 and 32760 (X'00007FF8') to be placed in word 1 in the output area.

Device Characteristics Information

The following information is placed into your area as a result of issuing a DEVTYPE macro:

Word 0
Describes the device as defined in the UCBTYP field of the UCB.

Word 1
Maximum block size. For direct access devices, this value is the smaller of either the maximum size of a nonkeyed block or the maximum block size allowed by the operating system; for magnetic tape devices, this value is the maximum block size allowed by the operating system. For all other devices, this value is the maximum block size accepted by the device.

If your program specifies either DEVTAB or UCBLIST, the next three fullwords contain the following information about direct access devices:

Word 2

Bytes 0-1	The number of physical cylinders on the device, including alternates.
Bytes 2-3	The number of tracks per cylinder. *Note:* Before you use bytes 2 and 3, read the description of word 4, byte 1, bit 0.

Word 3

Bytes 0-1	Maximum track length. Note that for the IBM 3375 and 3380 direct access devices, this value is not equal to the value in word 1 (maximum block size) as it is for other IBM direct access devices.

Note: Before using bytes 2 and 3, read the description of word 4.

Byte 2	Block overhead, keyed block—the number of bytes required for gaps and check bits for each keyed block other than the last block on a track.
Byte 3	Block overhead—the number of bytes required for gaps and check bits for a keyed block that is the last block on a track.

Bytes 2-3	Block overhead—the number of bytes required for gaps and check bits for any keyed block on a track including the last block. Use of this form is indicated by a 1 in bit 4, byte 1 of word 4.	
	Basic overhead—the number of bytes required for the count field. Use of this form is indicated by a 1 in bit 3, byte 1 of word 4.	

Word 4

Byte 0	Block overhead, block without key—the number of bytes to be subtracted from word 3, bytes 2 or 3 or bytes 2 and 3, if a block is not keyed.	
	If bit 3, byte 1 of word 4 is 1, this byte contains the modulo factor for a modulo device.	
Byte 1		
	Bit 0	If on, the number of cylinders, as indicated in word 2, bytes 0 and 1 are invalid.
	Bit 1	Reserved.
	Bits 2-3	If on, indicates a 3380 is attached to a 3880 Model 13 or 23.
	Bit 3	If on, indicates a modulo device (3375, 3380). For information on how to calculate the number of data bytes required for a data block for a modulo device, see the device data in *Data Administration Guide*.
	Bit 4	If on, bytes 2 and 3 of word 3 contain a halfword giving the block overhead for any block on a track, including the last block.
	Bits 5-6	Reserved.
	Bit 7	If on, a tolerance factor must be applied to all blocks except the last block on the track.
Bytes 2-3	Tolerance factor—this factor is used to calculate the effective length of a block. The calculation should be performed as follows:	
	Step 1	add the block's key length to the block's data length.
	Step 2	test bit 7 of byte 1 of word 4. If bit 7 is 0, perform step 3. If bit 7 is 1, multiply the sum computed in step 1 by the tolerance factor. Shift the result of the multiplication 9 bits to the right.

Step 3	add the appropriate block overhead to the value obtained above.

If bit 3, byte 1 of word 4 is 1, bytes 2 and 3 contain the overhead for the data or key field.

If your program specifies DEVTAB and RPS, or specifies UCBLIST, the next fullword contains the following information:

Word 5

Bytes 0-1	R0 overhead for sector calculations
Byte 2	Number of sectors for the device
Byte 3	Number of data sectors for the device

Figure 29 on page 134 shows the output for each device type that results from issuing the DEVTYPE macro.

Note: If your program specifies UCBLIST, the output consists of one 6-word entry for every UCB address contained in the UCB list.

Return Codes from the DEVTYPE Macro

Control is returned to your program at the next executable instruction following the DEVTYPE macro instruction. Register 15 contains a return code from the DEVTYPE macro. The return codes and their meanings are as follows:

Code	Meaning
00(X'00')	Indicates that the information concerning the ddname you specified has been successfully moved to your work area.
04(X'04')	Indicates that the ddname you specified was not found.

IBM Device[1] [2]	Maximum Record Size (Word 1, in Decimal)	DEVTAB (Words 2, 3, and 4, in Hexadecimal)	RPS (Word 5, in Hexadecimal)
2540 Reader	80	Not Applicable	Not Applicable
2540 Reader w/CI	80	Not Applicable	Not Applicable
2540 Punch	80	Not Applicable	Not Applicable
2540 Punch w/CI	80	Not Applicable	Not Applicable
2501 Reader	80	Not Applicable	Not Applicable
2501 Reader w/CI	80	Not Applicable	Not Applicable
3890 Document Processor	80	Not Applicable	Not Applicable
3505 Reader	80	Not Applicable	Not Applicable
3505 Reader w/CI	80	Not Applicable	Not Applicable
3525 Punch	80	Not Applicable	Not Applicable
3525 Punch w/CI	80	Not Applicable	Not Applicable
1403 Printer	120[2]	Not Applicable	Not Applicable
1403 w/UCS	120[2]	Not Applicable	Not Applicable
3203 Model 5 Printer	132	Not Applicable	Not Applicable
3211 Printer	132[2]	Not Applicable	Not Applicable
3262 Model 5 Printer	132	Not Applicable	Not Applicable
4245 Printer	132	Not Applicable	Not Applicable
4248 Printer	132[4]	Not Applicable	Not Applicable
3800 Printing Subsystem	136[3]	Not Applicable	Not Applicable
3400 (9-track, p.e.)	32760	Not Applicable	Not Applicable
3400 (9-track, d.d.)	32760	Not Applicable	Not Applicable
3400 (7-track)	32760	Not Applicable	Not Applicable
3480 (18-track)	32760	Not Applicable	Not Applicable
2305 Model 2 Fixed-Head Storage	14660	006000083A0A01215B080200	0140B4B1
3330/3333 Disk Storage	13030	019B0013336DBFBF38000200	00ED807C
3330V MSS Virtual Volume	13030	019B0013336DBFBF38000200	00ED807C

Figure 29 (Part 1 of 2). Output from DEVTYPE Macro

IBM Device[1] [2]	Maximum Record Size (Word 1, in Decimal)	DEVTAB (Words 2, 3, and 4, in Hexadecimal)	RPS (Word 5, in Hexadecimal)
3330 Model 11 (or 3333 Model 11) Disk Storage	13030	032F0013336DBFBF38000200	00ED807C
3340 Disk Storage (35 megabytes)	8368	015D000C2157F2F24B000200	0125403D
3340/3344 Disk Storage (70 megabytes)	8368	0230001E4B36010B52080200	0125403D
3350 Disk Storage	19069	0230001E4B36010B52080200	0185807B
3375 Disk Storage	32760	03BF000C8CA000E0201000BF	0340C4BB
3380 Models A04, AA4, and B04 Disk Storage	32760	0376000FBB6001002010010B	04E0DED6
3380 Models A04, AA4, and B04 Disk Storage (attached to a 3880 Model 13 or 23)	32760	0376000FBB6001002030010B	04E0DED6
3380 Models AD4 and BD4 Disk Storage	32760	0376000FBB6001002010010B	04E0DED6
3380 Models AE4 and BE4 Disk Storage	32760	06EB000FBB6001002010010B	04E0DED6
2250 Model 3 Display Unit		Not Applicable	Not Applicable
1030,1050,83B3, TWX,2250,S360	Not Applicable	Not Applicable	Not Applicable
115A,1130	Not Applicable	Not Applicable	Not Applicable
2780	Not Applicable	Not Applicable	Not Applicable
2740	Not Applicable	Not Applicable	Not Applicable

Figure 29 (Part 2 of 2). Output from DEVTYPE Macro

Notes to Figure 29:

1 CI—card image feature; d.c.—data conversion; d.d.—dual density; p.e.—phase encoding; UCS—universal character set; w/—with.

2 Although certain models can have a larger line size, the minimum line size is assumed.

3 The IBM 3800 Printing Subsystem can print 136 characters per line at 10-pitch, 163 characters per line at 12-pitch, and 204 characters per line at 15-pitch. The machine default is 136 characters per line at 10-pitch.

4 The IBM 4248 Printer returns 132 characters even if the 168 Print Position Feature is installed on the device.

DEVTYPE—List Form

The list form of the DEVTYPE macro is only valid with the UCBLIST function. It is used to construct an empty parameter list. By specifying MF=L you construct a parameter list, and you can subsequently supply the values by specifying the execute form of the macro.

name1	**DEVTYPE**	*,(area-addrx,area-size)* **,UCBLIST=YES** **,MF=L**

name1
: label of the parameter list to be constructed. It must also be specified in the corresponding MF=E form.

DEVTYPE—Execute Form

The execute form of the DEVTYPE macro is only valid with the UCBLIST function. It can be used to modify a parameter list and call the DEVTYPE function.

[*name1*]	**DEVTYPE**	*,(area-addrx,area-size)* **,UCBLIST=***(ucbl-addrx,ucbl-num)* **,MF=(E,***name1***)**

name1
: label of the parameter list constructed by the corresponding MF=L form.

Reading and Modifying a Job File Control Block

To accomplish the functions that are performed as a result of an OPEN macro instruction, the open routine requires access to information that you have supplied in a data definition (DD) statement. This information is stored by the system in a job file control block (JFCB).

In certain applications, you may find it necessary to modify the contents of a JFCB before issuing an OPEN macro instruction. For example, suppose you are adding records to the end of a sequential data set. You might want to add a secondary allocation quantity to allow the existing data set to be extended when the space currently allocated is exhausted. To assist you, the system provides the RDJFCB macro instruction. This macro instruction causes a specified JFCB to be moved from the scheduler work area (SWA), where it is stored, to an area specified in an exit list. The use of the RDJFCB macro instruction with an exit list is shown under "Example" on page 138. When you subsequently issue the OPEN macro instruction, you must indicate, by specifying the TYPE=J operand, that you want to open the data set using the JFCB in the area you specified.

The RDJFCB macro also allows you to retrieve allocation information (all JFCBs and all volume serial numbers) for the data sets in a concatenation. You may either select data sets or, by default, retrieve the information for all data sets in the concatenation. Figure 33 on page 146 illustrates how you can use RDJFCB to retrieve this information.

Caution: If you set the bit JFCNWRIT in the field JFCBTSDM to 1 before you issue the OPEN macro instruction, the JFCB is not written back to the SWA at the conclusion of open processing. OPEN TYPE=J normally moves your program's modified copy of the JFCB to the scheduler work area (SWA), replacing the system copy. To ensure that this move is done, your program must set bit zero (0) of the JFCBMASK+4 field to 1. The JFCBMASK format is shown in the Internal Data Areas section of *Open/Close/EOV Logic.* If the user JFCB, which the system used to open the data set, is not written back to SWA (JFCNWRIT set on), then errors may occur during EOV or close processing.

Some of the modifications that are commonly made to the JFCB include:

- Moving the creation and expiration date fields of the DSCB into the JFCB (see "Using RDJFCB for MSS Virtual Volumes" on page 140).
- Moving the secondary allocation quantity from the DSCB into the JFCB (see "Using RDJFCB for MSS Virtual Volumes" on page 140).
- Moving the DCB fields from the DSCB into the JFCB.
- Adding volume serial numbers to the JFCB (see "Using RDJFCB for MSS Virtual Volumes" on page 140 and "RDJFCB Security" on page 140).

  Volume serial numbers in excess of five are written to the JFCBX (extension) located in the SWA. The JFCBX cannot be modified by user programs.
- Modifying the data set sequence number field in the JFCB.
- Modifying the number-of-volumes field in the JFCB (see "Using RDJFCB for MSS Virtual Volumes" on page 140).
- Setting bit JFCDQDSP in field JFCBFLG3 to invoke the tape volume DEQ at demount facility (see "DEQ at Demount Facility for Tape Volumes" on page 147).
- Modifying the JFCRBIDO field in the JFCB to cause high-speed positioning to a specific data block on a 3480 tape volume.

RDJFCB—Read a Job File Control Block

The RDJFCB macro instruction causes a job file control block (JFCB) to be moved from the SWA (scheduler work area) into an area of your choice as identified via the EXLST parameter of the DCB macro for each data control block specified.

[*symbol*]	**RDJFCB**	(*dcb-address* ,[(*options*)],...)

dcb-address,[(*options*)]

(same as the dcb-address, option1, and option2 operands of the OPEN macro instruction, as shown in *Data Administration: Macro Instruction Reference*), except for the MODE operand, which is not valid with the RDJFCB macro.

The option operands do not affect RDJFCB processing. You can, however, specify them in the list form of the RDJFCB macro instruction and refer to the generated parameter list with the execute form of the macro.

Example: In Figure 30, the macro instruction at EX1 creates a parameter list for two data control blocks: INVEN and MASTER. In creating the list, both data control blocks are assumed to be opened for input; option2 for both blocks is assumed to be DISP. The macro instruction at EX2 moves the system-created JFCBs for INVEN and MASTER from the SWA into the area you specified, thus making the JFCBs available to your problem program for modification. The macro instruction at EX3 modifies the parameter list entry for the data control block named INVEN and indicates, through the TYPE=J operand, that the problem program is supplying the JFCBs for system use.

```
EX1       RDJFCB (INVEN,,MASTER),MF=L
          .
          .
          .
EX2       RDJFCB MF=(E,EX1)
          .
          .
          .
EX3       OPEN (,(RDBACK,LEAVE)),TYPE=J,MF=(E,EX1)
          .
          .
          .
INVEN     DCB       EXLST=LSTA,...
MASTER    DCB       EXLST=LSTB,...
LSTA      DS        0F
          DC        X'07'
          DC        AL3(JFCBAREA)
          .
          .
          .
JFCBAREA  DS        0F,176C
          .
          .
          .
LSTB      DS        0F
          .
          .
          .
```

Figure 30. Sample Code Using RDJFCB Macro

Multiple data control block addresses and associated options may be specified in the RDJFCB macro instruction. This facility makes it possible to read several job file control blocks in parallel.

An exit list address must be provided in each DCB specified by an RDJFCB macro instruction. Each exit list must contain an active entry that specifies the virtual storage address of the area into which a JFCB is to be placed.

Two kinds of JFCB entries may appear in the exit list. Each is briefly explained in the following text. A full discussion of the exit list and its use is contained in *Data Facility Product: Customization.*

Type 07 JFCB Exit List Entry

The type 07 JFCB exit list entry allows a variety of functions to the user, as described in the following text.

The format of the Type-07 JFCB exit list entry is as follows:

Types of Exit List Entry	Hexadecimal Code (High-Order Byte)	Contents of Exit List Entry (Low-Order Bytes)
Job file control block	07	Address of a 176-byte area to be provided if the RDJFCB or OPEN (TYPE=J) macro instruction is used. This area must begin on a fullword boundary and must be located within the user's region. Also, users running in 31-bit addressing mode must ensure that this area is located below 16 megabytes virtual.

The virtual storage area into which the JFCB is read must be at least 176 bytes long. Each exit list entry must be 4 bytes long. The system recognizes only the first occurrence of an exit list entry code. Indicate the end of the exit list by setting the high order bit in the entry code to 1.

The DCB may be either open or closed when this macro instruction is executed.

If the JFCB is read successfully for all DCBs in the parameter list, return code 0 is placed in register 15. If the JFCB is not read for any of the DCBs because the DDNAME is blank, or a DD statement is not provided, return code 4 is placed in register 15.

Warning: The following errors cause the results indicated:

Error	Result
A DD statement has not been provided.	Return code 4 is placed in register 15.
DDNAME field in DCB is blank.	A write-to-programmer is issued, the request for this DCB is ignored, and return code 4 is placed in register 15.

Error	Result
A virtual storage address has not been provided.	Abnormal termination of task.

If you want to open a VTOC data set to change its contents (that is, open it for OUTPUT, OUTIN, INOUT, UPDAT, OUTINX, or EXTEND), your program must be authorized under the Authorized Program Facility (APF). APF provides security and integrity for your data sets and programs. Details on how you authorize your program are provided in *System Programming Library: Supervisor Services and Macro Instructions.*

If the RDJFCB routine fails while processing a DCB associated with your RDJFCB request, your task is abnormally terminated. None of the options available through the DCB ABEND exit, as described in *Data Facility Product: Customization*, are available when a RDJFCB macro instruction is issued.

When using concatenated data sets, the RDJFCB routine modifies only the first JFCB.

Using RDJFCB for MSS Virtual Volumes: Care must be taken in using RDJFCB if the data set resides on MSS virtual volumes such that:

- The expiration date added does not conflict with other volumes within the specified MSVGP.
- The secondary allocation quantity should be in cylinder increments and be a multiple or submultiple of the primary allocation quantity to avoid fragmentation.
- The number of volumes must not exceed the number available in the specified MSVGP.
- Any volume serial numbers added to the JFCB should exist in the MSVGP.

RDJFCB Security: The volume serial numbers specified in the user-supplied JFCB will be compared with the volume serial numbers in the system JFCB located in the SWA. Each different volume serial number will be enqueued exclusively. The volumes will stay enqueued until the job step terminates, because the close routines will not dequeue the volumes. If the job step already has the volume open, OPEN TYPE=J will continue. If the volume is enqueued by another job step, a 413 abend will occur with a return code of 04.

Some JFCB modifications can compromise the security of existing password-protected data sets. The following modifications are specifically not allowed, unless the program making the modifications is authorized or can supply the password:

- Changing the disposition of a password-protected data set from OLD or MOD to NEW.
- Changing the data set name of one or more of the volume serial numbers when the disposition is NEW.

- Changing the label processing specifications to bypass label processing.

Note: An authorized program is one that is either in supervisor state, executing in one of the system protection keys (keys 0 through 7), or authorized under the Authorized Program Facility.

RDJFCB Use by Authorized Programs: If you change the data set name in the JFCB, you should do a system enqueue on the major name of "SYSDSN" for the substituted data set name. To use the correct interface with other system functions (for example, partial release), the ENQUEUE macro should include the TCB of the initiator and the length of the data set name (with no trailing blanks). When you complete processing of the data set, you should use the DEQ macro to release the resources. If the substituted data set name is enqueued by another job step, a 913 abend occurs with a return code of X'1C'.

Using RDJFCB to Process a Multivolume Direct Data Set: This use of RDJFCB and OPEN TYPE=J permits your program to process a multivolume data set. To do so, your program must cause the open routines to build a data extent block for each volume and issue mount messages for them. Your program must use the RDJFCB macro to read in the JFCB, and open each volume of the data set. The following code illustrates the procedure:

```
         RDJFCB     DCB1                 READS IN THE JFCB
         SR         R3,R3                CLEARS REG 3; IT WILL
*                                        HOLD COUNT OF VOLS TO
*                                        BE OPENED
         IC         R3,JFCBNVOL          PUTS # OF VOLS
*                                        IN REG 3
         LA         R4,DCB1              R4 POINTS TO DCB FOR
*                                        VOL TO BE OPENED
         LA         R5,1                 PUTS SEQUENCE # OF
*                                        FIRST VOL TO BE
*                                        OPENED IN REG 5
LOOP     EQU        *
         STH        R5,JFCBVLSQ          PUTS SEQ # OF VOL
*                                        TO BE OPENED WHERE
*                                        OPEN RTNS LOOK
         OPEN ((R4),OUTPUT),TYPE=J       OPENS ONE VOL
*    NOTE THAT THE TYPE=J OPTION OF THE MACRO MUST BE USED
         LA         R4,DCB2-DCB1(R4)     INCREMENT REG 4 TO
*                                        POINT TO THE DCB FOR
*                                        THE NEXT VOL TO BE
*                                        OPENED
         LA         R5,1(R5)             INCREMENT TO SEQ # OF
*                                        NEXT VOL TO BE OPENED
         BCT        R3,LOOP              LOOP UNTIL ALL VOLS
*                                        OPEN
         .
         .
         .
JFCB     DS         CL176               JFCB READ IN HERE
         ORG        JFCB+70
JFCBVLSQ DS         H                   SEQ # OF VOL TO BE
*                                       OPENED
         ORG        JFCB+117
JFCBNVOL DS         FL1                 # OF VOLS IN DATA SET
         ORG
* MAPPING MACRO IEFJFCBN MAY ALSO BE USED
DCB1 DCB DDNAME=SYSUT1,MACRF=(E),EXLST=EXITS,DSORG=PS
DCB2 DCB DDNAME=SYSUT1,MACRF=(E),EXLST=EXITS,DSORG=PS
DCB3 DCB DDNAME=SYSUT1,MACRF=(E),EXLST=EXITS,DSORG=PS
DCB4 DCB DDNAME=SYSUT1,MACRF=(E),EXLST=EXITS,DSORG=PS
DCB5 DCB DDNAME=SYSUT1,MACRF=(E),EXLST=EXITS,DSORG=PS
* THIS PROCEDURE WORKS FOR 5 VOLS OR LESS; THE JFCB
* EXTENSION, WHICH IDENTIFIES ADDITIONAL VOLS, CAN'T
* BE READ IN
EXITS    DS         0F
         DC         X'87',AL3(JFCB)      87 IDENTIFIES THIS AS
*                                        THE EXIT LIST ENTRY
*                                        THAT SHOWS WHERE JFCB
*                                        WILL BE READ IN
```

Type 13 JFCB Exit List Entry

The type 13 JFCB exit list entry allows you to retrieve selected allocation information, as described in the following text.

The format of the type 13 JFCB exit list entry is as follows:

Types of Exit List Entry	Hexadecimal Code (High-Order Byte)	Contents of Exit List Entry (Low-Order Bytes)
Job file control block	13	Address of an allocation retrieval list. OPEN TYPE=J does not recognize this exit list entry. This entry allows you to use RDJFCB to obtain copies of the JFCBs for data sets in a concatenation, and lists of all the volume serial numbers for those data sets. You may select JFCBs in the concatenation or by default, retrieve all of them. See Figure 31 on page 144 for the allocation retrieval list format, and Figure 33 on page 146 for an example of usage.

Using RDJFCB to Retrieve Allocation Information: RDJFCB uses DCB exit list entry type 13 to retrieve allocation information (JFCBs and volume serial numbers) for concatenated data sets. The exit list entry code is X'13', and is defined as "retrieve allocation information." The second through fourth bytes of this entry must point to an "Allocation Retrieval List," as described in Figure 31 on page 144. When you issue RDJFCB, this DCB exit list entry causes retrieval of JFCBs for the specified concatenated data sets, and lists of all volume serial numbers for these data sets. You may either select JFCBs in the concatenation or by default, retrieve all of them. OPEN TYPE=J does not recognize this exit list entry. RDJFCB uses the parameter list to receive and return information about the request. You may use the IHAARL macro (shown below) to generate and map the allocation retrieval list.

[*symbol*]	**IHAARL**	**[DSECT={<u>YES</u> | NO}]** **[,PREFIX=prefix]** **[,DESCR={YES | <u>NO</u>}]**

DSECT={<u>YES</u> | NO}
specifies whether the symbol at the beginning of the generated area appears on a DSECT instruction or a DC instruction. For DSECT=NO, the symbol appears on a DC instruction. The default is DSECT=YES.

PREFIX=prefix
allows you to invoke the macro more than once per assembly. Specifies a character string with which all generated symbols are to be prefixed. Do not specify delimiters, such as quotation marks. If you omit this operand or specify a null value, the prefix defaults to the characters ARL.

DESCR={YES | <u>NO</u>}
specifies whether the macro expansion includes the macro description (prolog). The default is DESCR=NO.

Figure 31 and Figure 32 describe the formats of the allocation retrieval list and allocation retrieval area, respectively.

Name	Offset	Bytes	Description
The Following Fields Are Set by the Caller of RDJFCB:			
ARLLEN	00(X'00')	2	Length of this area. Value must be 36 or more.
ARLIDENT	02(X'02')	2	EBCDIC 'AR'
ARLOPT1	04(X'04')	1	Option byte.
ARLLANY		0...	Allocation retrieval area must be below 16Mb line.
		1...	Allocation retrieval area may be above 16Mb line.
		.xxx xxxx	Reserved. Must be zero.
ARLRSVD1	05(X'05')	7	Reserved. Must be zero.
ARLRETRV	12(X'0C')	2	Number of data sets for which to retrieve information. If 0, retrieve all in the concatenation.
ARLFIRST	14(X'0E')	2	Number of first data set in concatenation for which to retrieve information. 0 or 1 specifies retrieval of information beginning with first data set in the concatenation.
The Following Fields Are Set by RDJFCB:			
ARLAREA	16(X'10')	4	Address of allocation retrieval area. See Figure 32.
ARLPOOL	20(X'14')	1	Storage subpool containing allocation retrieval area.
ARLRLEN	21(X'15')	3	Length of allocation retrieval area.
ARLRTRVD	24(X'18')	2	Number of concatenated data sets for which JFCBs were retrieved.
ARLCONC	26(X'1A')	2	Number of concatenated data sets. If no concatenation, this value is 1.
ARLRCODE	28(X'1C')	1	Reason Code: 0 = Requested information was read. The following reason codes are related to return code 8: 4 = ARLFIRST is greater than ARLCONC. 8 = Insufficient storage to read information. ARLPOOL and ARLRLEN describe what could not be obtained.
ARLRSVD2	29(X'1D')	7	Reserved. Used by RDJFCB.

Figure 31. Format of the Allocation Retrieval List

)ffset	Bytes	Description
)0(X'00')	2	Length of the information for this data set (including this field). The length is rounded up to a multiple of four so the starting address of the allocation retrieval area plus the value in the length field designates the address of the retrieval area for the next data set in the concatenation, if requested.
02(X'02')	2	Reserved. Set to zero.
04(X'04')	176(dec)	JFCB
180(X'B4')	variable	Sixth and subsequent volume serial numbers. Determined by the value in JFCBNVOL. If the number of volume serial numbers is fewer than the specified volume count, entries at the end of the list may contain all blanks. If the first byte of an entry is X'FF', the JCL specified VOL=REF and the volume could not be determined.

Figure 32. Format of the Allocation Retrieval Area

Return code 0 indicates that RDJFCB has filled in the allocation retrieval list fields. When you have finished using information from the retrieval areas, you should issue FREEMAIN to free the areas. To determine whether the release of the Data Facility Product on your system is capable of using exit list entry type X'13' to retrieve allocation information, set the ARLAREA field in the allocation retrieval list to zeros before issuing RDJFCB. If the ARLAREA is still zeros and the ARLRCODE field contains reason code 00 when RDJFCB returns control to your program, your release of DFP does not support this function.

Example: In Figure 33 on page 146, the macro instruction at ALLOCINF creates a parameter list for one DCB (INDCB), assumed to be open for input. The JFCBs and volume serial numbers are retrieved for all data sets allocated to DD name SYSLIB.

```
  ***JCL FOR FOLLOWING INVOCATION OF RDJFCB:

//SYSLIB DD     DISP=SHR,DSN=DEPT61.ROUTINES
//        DD     DISP=SHR,DSN=CORPORAT.ROUTINES
//        DD     DISP=SHR,DSN=SYS1.FORTLIB


  ***EXAMPLE CODE TO INVOKE RDJFCB ALLOCATION INFORMATION RETRIEVAL:

*   GET A COPY OF THE JFCB FOR THE FIRST OR ONLY DATA SET ALLOCATED
*   TO SYSLIB AND TRY TO READ THE JFCBS AND VOLUME SERIAL NUMBERS
*   FOR ALL DATA SETS ALLOCATED TO SYSLIB.
*
ALLOCINF RDJFCB (INDCB)
         LTR   R15,R15          TEST RDJFCB RETURN CODE
         BNZ   NOJFCB           BRANCH IF INFORMATION NOT AVAILABLE
         ICM   R1,X'F',SLBAREA  GET AND TEST ADDRESS OF ARL
         BZ    OLDSYSTM         GO IF SYSTEM DOES NOT SUPPORT ARL
         USING SLBSTRT,R1       ESTABLISH ADDRESSABILITY TO AREA
         CLI   SLBRCODE,0       TEST RDJFCB REASON CODE
         BNE   NOJFCB           BRANCH IF INFORMATION NOT AVAILABLE
*
*  LOOP THROUGH THE JFCBS IN THE AREA TO WHICH SLBAREA POINTS.
*  CODE CAN BE INSERTED HERE TO PRINT THE DATA SET NAMES AND
*  VOLUME SERIAL NUMBERS.
         .
         .
OLDSYSTM DS    0H               ROUTINE TO HANDLE JUST LIBJFCB
         .
         .
*
NOJFCB   DS    0H               ROUTINE TO HANDLE INABILITY TO GET THE
*                               JFCB. THE DATA SET MAY NOT BE ALLOCATED.
         .
         .
*
SLBOPNX  DS    0H               DCB OPEN EXIT ROUTINE FOR SYSLIB.
*                               HANDLES RECFM, LRECL, AND BLKSIZE.
         .
         .
INDCB    DS    DSORG=PO,DDNAME=SYSLIB,MACRF=R,SYNAD=INERROR,           X
               EXLST=INEXLST
INEXLST  DC    0F'0',X'05'      ENTRY CODE FOR OPEN EXIT ROUTINE
         DC    AL3(SLBOPNX)     ADDR OF DCB OPEN EXIT ROUTINE
         DC    X'13'            ENTRY CODE TO RETRIEVE ALLOCATION
*                                INFORMATION
         DC    AL3(SLB)         ADDR OF ALLOCATION RETRIEVAL LIST
         DC    X'87'            ENTRY CODE TO RETRIEVE FIRST JFCB AND
*                                INDICATE LAST ENTRY IN LIST
         DC    AL3(LIBJFCB)     ADDR OF JFCB FOR FIRST DATA SET
*
*  AN ALLOCATION RETRIEVAL LIST FOLLOWS, POINTED TO BY DCB EXIT LIST.
*
SLBSTRT  IHAARL DSECT=NO,PREFIX=SLB
         DC    0F'0'
LIBJFCB  DC    CL176' '         FIRST JFCB
         .
         .
```

Figure 33. Sample Code Retrieving Allocation Information. IHAARL PREFIX=SLB requires Assembler H.

DEQ at Demount Facility for Tape Volumes

This facility is intended to be used by long-running programs that create an indefinitely long tape data set (such as a log tape). Use of this facility by such a program permits the processed volumes to be allocated to another job for processing (such as data reduction). This processing is otherwise prohibited unless the indefinitely long data set is closed and dynamically unallocated.

You may invoke this facility only through the RDJFCB/OPEN TYPE=J interface by setting bit JFCDQDSP (bit 0) in field JFCBFLG3 (offset 163 or X'A3') to 1. The volume serial of the tape is dequeued when the volume is demounted by OPEN or EOV with message IEC502E when all the following conditions are present:

- The tape volume is verified for use by OPEN or EOV.
- JFCDQDSP is set to 1.
- The program is APF authorized (protect key and supervisor/problem state are not relevant).
- The tape volume is to be immediately processed for output. That is, either OPEN verifies the volume and the OPEN option is OUTPUT, OUTIN, or OUTINX; or EOV verifies the volume and the DCB is opened for OUTPUT, OUTIN, INOUT, or EXTEND, and the last operation against the data set was an output operation (DCBOFLWR is set to 1).

Note that, in order for EOV to find JFCDQDSP set to 1, the program must not inhibit the rewrite of the JFCB by setting bit 4 of JFCBTSDM to 1.

The tape volume is considered verified after file protect, label type, and density conflicts have been resolved. The volume is dequeued when demounted after this verification, even if further into OPEN or EOV processing the volume is rejected because of expiration date, security protection, checkpoint data set protection, or an I/O error.

When the volume serial is dequeued, the volume becomes available for allocation to another job. However, because the volume DEQ is performed without unallocating the volume, care must be exercised both by the authorized program and the installation to prevent misuse of the DEQ at demount facility. A discussion of such misuse follows.

1. The authorized program must not close and reopen the data set using the tape volume DEQ at demount facility. If it does, one of the following can occur:

   a. The dequeued volume may be mounted and in use by another job. When the volume is requested for mounting, for the authorized program, the operator is unable to satisfy the mount. Therefore, the operator must either cancel the requesting job, cancel the job using the volume, wait for the requesting job to time out, or wait for the job using the volume to terminate.

   b. The dequeued volume may be allocated to another job but not yet in use. The operator mounts the volume to satisfy the mount request of the authorized job. When the volume is requested for mounting by the other job, the operator is unable to satisfy the mount request, and is faced with the same choices as in a, above.

c. The dequeued volume may not yet be allocated to another job and the volume is mounted to satisfy the mount request of the authorized job. Another job may allocate the volume and, when the volume is requested for mounting, the situation is the same as in b, above.

It is the responsibility of the installation that permits a program to run with APF authorization to ensure that it does not close and reopen a data set using the DEQ at demount facility.

2. Care should be exercised when an authorized program uses the DEQ at demount facility (data set 1) but processes another tape data set (data set 2). Assume the same volume serial numbers have been coded in the DD statements for data set 1 and data set 2. As the volumes of data set 1 are demounted, they are dequeued even though those volumes may yet be requested for data set 2. All the problems explained in a, b, and c in 1, above, may occur as data set 2 and another job contend for a dequeued volume.

   This problem should not occur, given the intended use of the DEQ at demount facility; that is, a long-running application creating an indefinitely long tape data set. This type of application is not normally invoked through batch execution with user-written DD statements.

3. After a volume has been demounted and dequeued because of the DEQ at demount facility, the volume is not automatically rejected by the control program when mounted in response to a specific or nonspecific mount request. Without the use of the facility, the control program can recognize (by the ENQ) that the volume is in use, and reject the volume. Therefore, operations procedures, in effect to prevent incorrect volumes from being mounted, should be reviewed in the light of reduced control program protection from such errors when the DEQ at demount facility is used. Specifically, if a volume is remounted for an authorized program and the volume had been used previously by that authorized program, duplicate volume serial numbers will exist in the JFCB, and the control program will be unable to release the volume during EOV processing.

4. Checkpoint/restart considerations are discussed in *Checkpoint/Restart User's Guide*.

OPEN—Initialize Data Control Block for Processing the JFCB

The OPEN macro instruction initializes one or more data control blocks so that their associated data sets can be processed.

A full explanation of the operands of the OPEN macro instruction is contained in *Data Administration: Macro Instruction Reference*. The TYPE=J option, because it is used in conjunction with modifying a JFCB, should be used only by the system programmer or under the system programmer's supervision.

The parameters of the OPEN TYPE=J macro instruction are:

[*symbol*]	**OPEN**	(*dcb-addr* ,[(*options*)],...) [,**TYPE=J**]

TYPE=J

specifies that, for each data control block referred to, you have supplied a job file control block (JFCB) to be used during initialization. A JFCB is an internal representation of information in a DD statement.

During initialization of a data control block, its associated JFCB may be modified with information from the data control block or an existing data set label or with system control information.

The system always creates a job file control block for each DD control statement. The job file control block is placed in the SWA (scheduler work area). Its position, in relation to other JFCBs created for the same job step, is noted in a table in virtual storage.

When the TYPE=J operand is specified, you must also supply a DD statement. However, the amount of information given in the DD statement is at your discretion, because you can modify many fields of the system-created job file control block. If you specify DUMMY on your DD statement, the open routine will ignore the JFCB DSNAME and open the data set as dummy. (See Figure 30 on page 138 for an example of coding that modifies a system-created JFCB.)

Notes:

1. *The DD statement must specify at least:*
   - *Device allocation (refer to JCL User's Guide for methods of preventing share status)*
   - *A ddname corresponding to the associated data control block DCBDDNAM field*
2. *The* **MODE** *operand is not shown here because it is not allowed with the TYPE=J operand of the OPEN macro instruction.*

High-Speed IBM 3480 Positioning

High-speed positioning for 3480 tape drives is available when opening a tape data set on an IBM standard-labeled tape for either EXTEND (OUTINX, EXTEND, or DISP=MOD), or when opening to the **beginning** of such a data set. To invoke high-speed positioning, your program must modify certain fields in the JFCB and use OPEN TYPE=J to open the data set.

You should use the following procedure to modify the JFCB:

1. Issue the RDJFCB macro to have the system move the JFCB into your work area.

2. Set the JFCPOSID flag in the JFCBFLG3 flag byte to indicate that you are providing a block ID for a high speed search.

3. Move the block ID into the JFCRBIDO field of the JFCB. If you are opening to the beginning of a data set, use the block ID of the first header label record of that data set. If you are opening to the end of a data set (for example, to extend it), use the block ID of the tape mark immediately following the last block of user data in that data set.

4. Issue the OPEN TYPE=J macro to have the system use your modified JFCB.

After the tape is positioned, OPEN processes the trailer labels for the data set being extended.

If you set the JFCPOSID flag off, OPEN positions the volume normally, as though the high-speed positioning feature were not active.

If you set the JFCPOSID flag on, but do not provide a block ID in the JFCRBIDO field, OPEN positions the volume normally and does one of the following:

- If you are opening to the beginning of a data set, OPEN inserts the block ID of the first header label record of that data set into the JFCRBIDO field.

- If you are opening to the end of the data set, OPEN inserts the the block ID of the tape mark immediately following the last block of user data for that data set into the JFCRBIDO field.

If the JFCPOSID flag is on during CLOSE processing, CLOSE inserts the block ID for the first header label record of the **next** data set (which may not exist) into the JFCRBIDC field. Therefore, if you deallocate the 3480 device and want to use the current block ID for subsequent processing, you must save the block ID before you CLOSE the data set.

OPEN resets the JFCPOSID flag if either of the following conditions exists:

- Your program issues an OPEN which is not TYPE=J.

- The requested tape volume is not an IBM standard-labeled volume.

- The requested unit is not a buffered tape device

Notes:

1. *If you specify dynamic unallocation (with SVC 99, FREE=CLOSE on the DD statement, or the FREE option on the CLOSE macro), then the block ID for the next data set will not be available to your program. This is because dynamic unallocation frees the JFCB.*

2. *When using high-speed positioning, specify the data set sequence number normally, either explicitly by LABEL=(seqno,SL) on the DD statement, or by default.*

After the system routines have used the JFCRBIDO field for high-speed positioning, they clear JFCRBIDO in the system's copy of the JFCB to prevent misinterpretation during a subsequent OPEN.

Ensuring Data Security by Validating the Data Extent Block

Protecting one user's data from inadvertent or malicious access by an unauthorized user depends on protection of the data extent block (DEB). The DEB is a critical control block because it contains information about the device a data set is mounted on, and describes the location of data sets on direct access device storage volumes. The DEB also contains the address of the appendage vector table (AVT). Using the AVT, an unauthorized user can modify the AVT to give control to a routine in supervisor state to read from and write to data sets to which access would otherwise be denied.

To guarantee protection of the DEB, the DEBCHK macro instruction is provided. The DEBCHK macro instruction can be found in SYS1.MACLIB. The DEBCHK macro is issued by several components of the system control program. For example:

- The open access method executors issue the macro to add the address of a DEB they have built to a list of valid addresses called the DEB table. The DEB validity-checking routine builds and maintains a DEB table for each job step.
- The EXCP processor uses the macro to verify that the DEB passed with each EXCP request is in the DEB table.
- The close component issues the macro to remove a DEB from the DEB table.

If you code a routine that builds a DEB, you must add the address of the DEB you built to the DEB table. If you code a routine that depends on the validity of a DEB that is passed to your routine, you should verify that the DEB passed to your routine has a valid entry in the DEB table and points to your DCB or access method control block (ACB). Use the **TYPE=ADD** and the **TYPE=VERIFY** operands of the macro, respectively.

To prevent an asynchronous routine from changing or deleting, or assigning a new DEB to a DCB, you must hold the local lock. In this case, you must use the branch entry to the DEBCHK verify routine.

Additional details about the functions provided by the DEB validity-checking routine and about the contents of the DEB table are available in *Open/Close/EOV Logic*.

The DEBCHK macro instruction provides four functions:

- Adds the address of a DEB to the DEB table, which is located in protected storage. The DEB table contains the address of every user DEB associated with a given job step. Every system control program component that builds a user DEB must add the address of that DEB to a DEB table.
- Verifies that the DEB table associated with a given job step contains the address of a valid DEB and that the DEB points to the DCB (or ACB). Any system control program component or problem program can use this function to verify that a DEB is valid.

- Deletes the address of a DEB from the DEB table. Any program that deletes a user DEB must, before it deletes the DEB, issue a DEBCHK macro with a **TYPE=DELETE** operand to delete the address of the DEB from the DEB table. If the DEB validity-checking routine encounters an error while deleting the address from the DEB table, the job step is abnormally terminated.

- Deletes the address of a DEB from the DEB table in the same way as the preceding function, except that, instead of terminating the job step, this function merely returns an error code in register 15. This function is provided to prevent recurring abnormal termination. The format of the DEBCHK and a description of the operands follow:

DEBCHK—Macro Specification

[*symbol*]	**DEBCHK**	*cbaddr* **[,TYPE={<u>VERIFY</u> | ADD | DELETE | PURGE}]** **[,AM={***amtype* **|** *(amaddr)* **|** *((amreg))***}]** **[,BRANCH={<u>NO</u> | YES}]** **[,TCBADDR=***address***]** **[,KEYADDR=***address***]** **[,SAVREG=***req***]** **[,MF=L]**

cbaddr

for **BRANCH=NO**
RX-type address, (2-12), or (1)

A control block address passed to the DEBCHK routine. This operand is ignored if MF=L is coded. For verify, add, and delete requests, cbaddr is the address of a DCB or ACB that points to the DEB whose address is either verified to be in the DEB table, added to the DEB table, or deleted from the DEB table. For the purge function, cbaddr is the address of the DEB whose pointer is to be purged from the table: No reference is made to the DCB or ACB.

Note: A spooled DCB's DEB does not point back to the DCB, but to the spooled ACB; in this case, the DEBCHK should be issued against the ACB.

for **BRANCH=YES**
The A-type address of a 4-byte field, or a register (1) or (3-12), that points to the DCB or ACB containing the DEB to be verified.

TYPE={<u>VERIFY</u> | ADD | DELETE | PURGE}
indicates the function to be performed. If MF=L is coded, TYPE is ignored. The functions are:

VERIFY
This function is assumed if the **TYPE** operand is not coded. The control program checks the DEB table to determine whether the DEB pointer is in the table at the location indicated by the DEBTBLOF field of the DEB. The DEB is also checked to verify that DEBDCBAD points to the DCB (or ACB) passed to DEBCHK. The

DEBAMTYP field in the DEB is compared to the AM operand value, if given. The two must be equal. **TYPE=VERIFY** may be issued in either supervisor or problem state.

ADD

The DEB and the DCB (or ACB) must point to each other before the DEB address can be added to the DEB table. Before the DEB pointer can be added to the table, the DEB itself must be queued on the current TCB DEB chain (the TCBDEB field contains the address of the first DEB in the chain). The DEB address is added to the DEB table at some offset into the table. That offset value is placed in the DEBTBLOF field of the DEB, and the access method type is inserted into the DEBAMTYP field of the DEB. A zero is placed in the DEBAMTYP field if the AM operand is not coded. **TYPE=ADD** can be issued only in supervisor state.

DELETE

The DEB and the DCB (or ACB) must point to each other before the DEB address can be deleted from the DEB table. **TYPE=DELETE** can be issued only in supervisor state.

PURGE

The DEB pointer is removed from the DEB table without checking the DCB (or ACB). **TYPE=PURGE** can be issued only in supervisor state.

AM

specifies an access method value. Each value corresponds to a particular access method type (note that BPAM and SAM have the same values):

Type	**Value**
TCAMAP	(X'84')
SUBSYS	(X'81')
ISAM	(X'80')
BDAM	(X'40')
SAM	(X'20')
BPAM	(X'20')
TAM	(X'10')
GAM	(X'08')
TCAM	(X'04')
EXCP	(X'02')
VSAM	(X'01')
NONE	(X'00')

The operand can be coded in one of the following three ways, only the first of which is valid for the list form (MF=L) of the instruction.

amtype

refers to the access method: **ISAM, BDAM, SAM, BPAM, TAM** (which refers to BTAM only), **GAM, TCAM, EXCP**, or **VSAM**. **TCAMAP** identifies a TCAM application-program DEB. **SUBSYS** identifies a subsystem of the operating system, such as a job entry subsystem. **NONE** indicates that no access method or subsystem is specified.

(amaddr)

is the RS-type address of the access method value. This format may not be coded when **MF=L** is used.

((amreg))

is one of the general registers 1 through 14 that contains the access method value in its low-order byte (bit positions 24 through 31). The high-order bytes are not inspected. This form may not be used when **MF=L** is coded.

The use of amaddr and amreg should be restricted to those cases where the access method value has been generated previously by the **MF=L** form of DEBCHK. If **MF=L** is not coded, the significance of the AM operand depends upon the **TYPE.**

If **TYPE** is **ADD** and **AM** is specified, the access method value is inserted in the DEBAMTYP field of the DEB, and all subsequent DEBCHK macros referring to this DEB must either specify the same **AM** or omit the operand. When the **AM** operand is omitted for **TYPE=ADD**, a null value (0) is placed in the DEB and all subsequent DEBCHK macros must omit the **AM** operand.

If **AM** is specified when the **TYPE** is **PURGE, DELETE,** or **VERIFY**, the access method value is compared to the value in the DEBAMTYP field of the DEB. If **AM** is omitted, no comparison is made.

BRANCH={NO | YES}

specifies whether you want to use the branch entry to the DEBCHK verify routines.

NO

specifies branch entry is not to be used. The operands SAVREG, TCBADDR, and KEYADDR are ignored.

YES

specifies the branch entry is to be used. TYPE=VERIFY must be implicitly or explicitly specified. The operands TCBADDR and KEYADDR are required. AM and MF are ignored. Notes for BRANCH=YES:

- Registers 1, 2, 10, 11, 14, and 15 must not be used for SAVREG=.
- Registers 1, 2, 10, 11, 14, 15, and the register specified for SAVREG= must not be used for cbaddr, TCBADDR=, or KEYADDR=.
- The contents of registers 10, 11, and 14 are unpredictable on completion. Also, if you do not specify SAVREG=, the contents of register 2 are unpredictable.
- At completion time, register 1 contains the address of the DEB, and register 15 contains either 0, 4, or 16 (see "Return Codes from the DEBCHK Macro" on page 155 for codes and their meanings).

TCBADDR=*address*—A-type address or (3-12)
specifies the location or register containing the address of the TCB to be used by the DEBCHK verify routine. Use this operand only when BRANCH=YES.

KEYADDR=*address*—A-type address or (3-12)
specifies the location, or a register pointing to the location of a field containing the key to be used when accessing the DCB (or ACB). Use this operand only when BRANCH=YES.

SAVREG=*reg*
specifies the register in which register 2 is to be saved. Use this operand only when BRANCH=YES.

MF=L
indicates the list form of the DEBCHK macro instruction. When MF=L is coded, a parameter list is built consisting of the access method value that corresponds to the **AM** keyword. This value may be referenced by name in another DEBCHK macro by coding **AM=(amaddr)**, or it may be inserted into the low-order byte of a register before issuing another DEBCHK macro by coding **AM=((amreg))**.

Return Codes from the DEBCHK Macro

If the DEBCHK routine completes successfully, register 15 will be set to 0 and register 1 will contain the address of the DEB when control is returned to your program. Otherwise, register 15 will contain one of the following decimal codes:

Code	Meaning
04(X'04')	Either (a) the DEB table associated with the job step does not exist; or (b) the DEBTBLOF field of the DEB was set to zero or a negative number, or was larger than the DEB table; or (c) register 1 did not contain the same address as the DEB table entry.
08(X'08')	An invalid TYPE was specified. (The DEBCHK routine was entered by a branch, not by the macro.)
12(X'0C')	Your program was not authorized and TYPE was not VERIFY.
16(X'10')	DEBDCBAD did not contain the address of the DCB (or ACB) that was passed to the DEBCHK routine.
20(X'14')	The AM value does not equal the value in the DEBAMTYP field.
24(X'18')	The DEB is not on the DEB chain and TYPE=ADD was specified.
28(X'1C')	TYPE=ADD was specified for a DEB that was already entered in the DEB table.
32(X'20')	The DEB table exceeded the maximum size (32760 bytes) and TYPE=ADD.

Purging and Restoring I/O Requests

The system's purge routines, guided by a parameter list you pass them, perform either a halt or a quiesce operation. In a halt operation, the purge routines stop the processing of specified I/O requests that were initiated with an EXCP macro instruction. In a quiesce operation, the purge routines:

- Allow the completion of I/O requests that were initiated with an EXCP macro instruction and have been passed to the I/O supervisor for execution
- Stop the processing of those requests that have not as yet been passed to the I/O supervisor, but save the IOBs of the requests so that they can be reprocessed (restored) later.

The system's restore routines make it possible to reprocess I/O requests that are quiesced. (*Note*: Not covered here is the purge and restore processing that takes in I/O requests not initiated by an EXCP macro instruction. If you want to learn the full scope of purge and restore processing, see the I/O supervisor logic section of *System Logic Library,* Volume 8.)

You can give control to the purge and restore routines in one of two ways: (1) by loading register 1 with the address of the parameter list and issuing specific SVC instructions or (2) by issuing the PURGE and RESTORE macro instructions. If your installation requires the use of macro instructions, you must add the macro definitions to the macro library (SYS1.MACLIB) or place them in a partitioned data set and concatenate this data set to the macro library. The macro definitions, JCL, and utility statements needed to add the macros to your macro library are presented in Figure 34 on page 157 and Figure 35 on page 158. Whether you issue the macro instructions or the SVC instructions, you must first build a parameter list. The SVC instructions are SVC 16 for PURGE and SVC 17 for RESTORE.

```
RESTORE Macro Definition

        MACRO
&NAME   RESTORE       &LIST
        AIF           ('&LIST' EQ '').E1
&NAME   IHBINNRA      &LIST              LOAD REG 1
        SVC           17                 ISSUE SVC FOR RESTORE
        MEXIT
.E1     IHBERMAC      01,150             LIST ADDR MISSING
        MEND

Control Statements Required

//jobname     JOB         {parameters}
//stepname    EXEC        PGM=IEBUPDTE,PARM=NEW
//SYSPRINT    DD          SYSOUT=A
//SYSUT2      DD          DSNAME=SYS1.MACLIB,DISP=OLD
//SYSIN       DD          DATA
./  ADD       NAME=RESTORE,LIST=ALL
              .
              .
              .
              RESTORE macro definition
              .
              .
              .
./  ENDUP
/*
```

Figure 35. Macro Definition, JCL, and Utility Statements for Adding RESTORE Macro to the System Macro Library

PURGE—Halt or Finish I/O-Request Processing

The macro instruction used to call the purge routines is coded as follows:

[*symbol*] *address*	**PURGE**	*parameter-list*

parameter list address—RX-type address, (2-12) or (1)
address of a parameter list, 12 or 16 bytes long, that you have built on a fullword boundary in your storage. The parameter list address can be specified as an RX-type constant or in registers 2 through 12, or 1.

The format and contents of the parameter list are as follows:

Byte	**Contents**
0	A byte in which you specify what the purge routines will do. These are the bit settings and their meanings:

1...	Purge I/O requests to a single data set.

0...	Either purge I/O requests associated with a TCB or address space, or purge I/O requests to more than one data set.
.1..	Post ECBs associated with purged I/O requests.
..1.	Halt I/O-request processing. (Quiesce I/O-request processing, if 0.)
...1	Purge related requests only. (Valid only if a data-set purge is requested.)
.... 0...	Reserved—must be zero.
.... .1..	Do not purge the TCB request-block chain of asynchronously scheduled processing.
.... ..1.	Purge I/O requests associated with a TCB.
....1	This is a 16-byte parameter list. Additional purge options are specified in bytes 12 to 15. (If this bit is off, the list is 12 bytes long, and the purge routines do not put a return code in byte 4 of this list or in register 15.)

1,2,3	The address of a DEB if you are purging I/O requests to a single data set. The address of the first DEB in a chain of DEBs if you are purging I/O requests to more than one data set. (The next-to-the-last word of each DEB must point to the next DEB in the chain; the second word of the last DEB must contain zeros.)
4	A byte of zeros. (If bit 7 of byte 0 is on, the purge routines will put a code in this byte: X'7F' if the purge operation is successful; X'40' if it is not successful. If bit 7 of byte 0 is off, then X'7F' appears in this byte.)
5,6,7	The address of the TCB associated with the I/O requests you want purged (but only if you turned on bit 6 of byte 0). May be zeros if the TCB is the one you are running under.
8	Driver 10. (Default value of X'00' implies that EXCP is the owner.
9,10,11	The address of a word in your storage or the address of the DEBUSPRG field (which is X'11' bytes more than the DEB address in this parameter list). At whichever address you specify, the purge routines store a pointer to the purged I/O restore list, PIRL. In the PIRL is a pointer to the first IOB in the chain of IOBs. The location of the pointer and format of the chain are shown in Figure 36 on page 161. *Note:* This field is relevant for quiesce options only.
12	A byte in which you can specify additional purge options. These are the bit settings and their meanings:

Note: The following applies only if bit 7 of byte 0 is set to one.

..1.	Purge I/O requests associated with an address space. (You must be in supervisor state.)
...1	Check the validity of all the DEBs associated with the purge operation if this is a data-set purge. Validate this parameter list, whatever the type of purge operation, by ensuring that there are no inconsistencies in the selection of purge options. (If you are in problem state, these actions are taken regardless of the bit setting.)
.... 1...	Ensure that I/O requests will be reprocessed (restored) under their original TCB. (If zero, and this byte is meaningful (bit 7 of byte 0 is on), the I/O requests will be reprocessed under the TCB of the program making the restore request.)
.... .0..	Must be zero.

13	A byte of zeros.
14,15	The 2-byte ID of the address space associated with the I/O requests you want purged. (Only meaningful if bit 2 of byte 12 is on.)

Control is returned to your program at the instruction following the PURGE macro instruction.

Return Codes from PURGE Macro

If the purge operation was successful, register 15 will contain zeros. Otherwise, register 15 will contain one of the following return codes:

Code	Meaning
04(X'04')	Your request to purge I/O requests associated with a given TCB was not honored because that TCB did not point to the job step TCB, as it must when the requestor is in problem state.
08(X'08')	Either you requested an address-space purge operation, but were not in supervisor state, or you requested a data-set purge operation, but supplied no data-area address in bytes 1, 2, and 3 of the purge parameter list.
20(X'14')	Another purge request has preempted your request. You may want to reissue your purge request in a time-controlled loop.

Note: Register 15 will contain zeros, regardless of the outcome of the purge operation, if you set bit 7 in byte 0 of the parameter list to zero.

Modifying the IOB Chain

Note that, although this procedure is not recommended, if you want to change the order in which purged I/O requests are restored or prevent a purged request from being restored, you may change the sequence of IOBs in the IOB chain or remove an IOB from the chain. The address of the IOB chain can be obtained from the PIRL (see Figure 36). (The address of the PIRL is at the location pointed to by bytes 9 through 11 of the purge parameter list.)

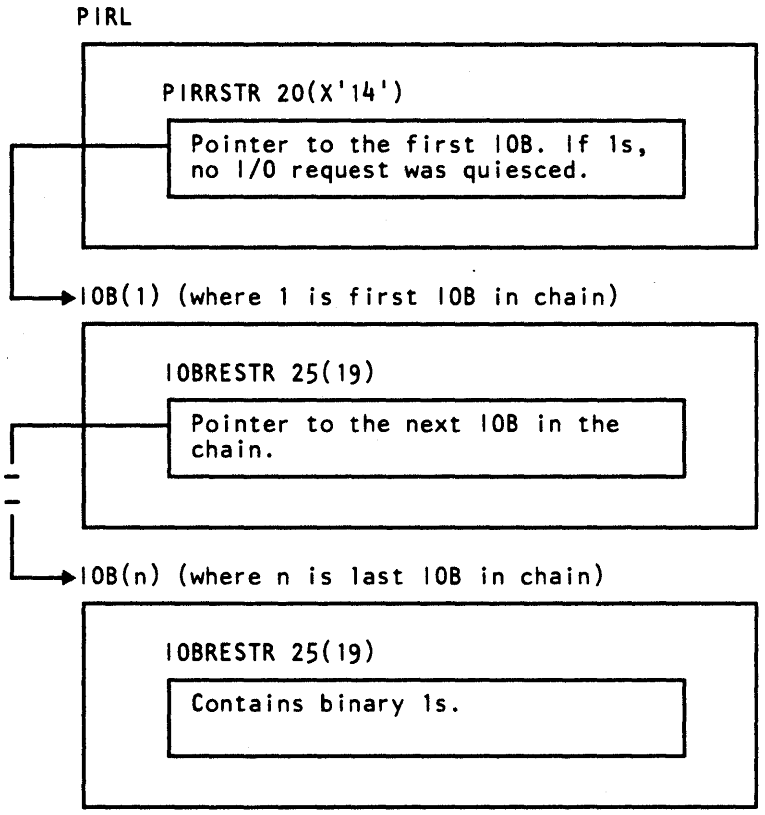

PIRL
PIRRSTR 20(X'14')
Pointer to the first IOB. If 1s, no I/O request was quiesced.
IOB(1) (where 1 is first IOB in chain)
IOBRESTR 25(19)
Pointer to the next IOB in the chain.
IOB(n) (where n is last IOB in chain)
IOBRESTR 25(19)
Contains binary 1s.

Figure 36. The PIRL and IOB Chain

RESTORE—Reprocess I/O Requests

The RESTORE macro is coded as follows:

[*symbol*]	**RESTORE**	*restore address*

restore address—RX-type address, (2-12) or (1)
address you specified at byte 9 of the purge parameter list.

Performing Track Calculations

The TRKCALC macro performs track capacity calculations. The standard, list, execute, and DSECT forms of the macro are described. Examples of the TRKCALC macro follow the macro descriptions. Using TRKCALC, you may do the following:

- Perform track capacity calculations
- Determine the number of records of a given size that can be written on a full track or on the remainder of a track
- Perform track balance calculations as follows:
  - Determine whether a given record size can be written in the space remaining on the track and return the new track balance.
  - Determine the maximum size record that can be written on the track if the given record does not fit.
  - Determine the track balance if the last physical record is removed from the track.

TRKCALC—Standard Form

The format of the TRKCALC macro is:

[*symbol*]	**TRKCALC**	**FUNCTN={TRKBAL | TRKCAP}** **{,DEVTAB=*addr* | ,UCB=*addr* | ,TYPE=*addr*}** **[,BALANCE=*addr*]** **[,REMOVE={YES | NO}]** **[,MAXSIZE={YES | NO}]** **{,RKDD=*addr* | ,R=*addr*,K=*addr*,DD=*addr*}** **[,REGSAVE={YES | NO}]** **[,MF=I]**

FUNCTN={TRKBAL | TRKCAP}
specifies the function to be performed.

Note: You must specify one of the three keywords, DEVTAB, UCB, or TYPE, to provide the macro a source for information.

TRKBAL
if REMOVE=NO is specified, TRKBAL calculates whether an additional record fits on the track and what the new track balance would be if the record were added. If REMOVE=YES is specified, TRKBAL calculates what the track balance would be if a record were removed from the track. The record to be added or removed from the track is defined by the RKDD parameter, or by the R, K, and DD parameters.

If R=1 (or the R value in the RKDD parameter is 1) and REMOVE=NO is specified, record 1 is added to an empty track; if R=1 and REMOVE=YES is specified, record 1 is deleted from the track, leaving an empty track.

If R≠1, the specified record is added to or removed from the track. The input track balance may be supplied through the BALANCE parameter; if it is not supplied, it is assumed that the track contains equal-sized records as specified in the RKDD parameter (or in the R, K, and DD parameters).

When REMOVE=NO is specified, one of the following occurs:

- If the record fits on the track, register 0 contains the new track balance.
- If the record does not fit on the track and MAXSIZE=NO is specified, a "record does not fit" return code is given in register 15.
- If the record does not fit and MAXSIZE=YES is specified, one of the following happens:
  - The data length of the largest record that fits in the remaining space is returned in register 0.
  - A code is returned that indicates no record fits in the remaining space.

When REMOVE=YES is specified, one of the following occurs:

- If R=1, register 0 contains the track capacity.
- If R≠1, registers 0 contains the input track balance (supplied through the BALANCE parameter) incremented by the track balance used by the input record. If the input balance is not supplied, register 0 contains the track capacity left after R−1 records are written on the track.

TRKCAP
calculates, and returns in register 0, the number of fixed-length records that may be written on a whole track (R=1) or on a partially filled track (R≠1). The records are defined by the K and DD values of the RKDD parameter, or by the K and DD parameters.

One of the following occurs:

- If R=1, the BALANCE parameter is ignored and the calculation is made on an empty track.
- If R≠1 and the BALANCE parameter is omitted, the calculation is made for a track that already contains R−1 records of the length defined by the K and DD values.

- If R≠1 and the BALANCE parameter is supplied, the calculation is made for a track whose remaining track balance is the value of the BALANCE parameter.

DEVTAB=*addr*—RX-type address, (2-12), (0), (14)
addr specifies a word that contains the address of the device characteristics table entry (DCTE). If you specify a register, it contains the address of the DCTE, not the address of a word containing the address of the DCTE. The address of the DCTE can be found in the DCBDVTBA field of an opened DCB.

UCB=*addr*—RX-type address, (2-12), (0), (14)
addr specifies the address of a word that contains the address of the UCB. If you specify a register, it contains the address of the UCB, not the address of a word containing the address of the UCB. You must ensure that the UCB address is valid by verifying that byte 3 (UCB+2) in the UCB contains X'FF'.

TYPE=*addr*—RX-type address, (2-12), (0), (14)
you may specify the address of the UCB device type (UCBTBYT4), or you may specify the 1-byte UCB device type in the low-order byte of a register.

BALANCE=*addr*—RX-type address, (2-12), (0), (14)
you may specify either the address of a halfword containing the current track balance, or you may specify the balance in the low-order two bytes of a register. The value supplied may be the value returned when you last issued TRKCALC. If R=1, the balance is reset to track capacity by TRKCALC, and your supplied value is ignored. This is an input value and is not modified by the TRKCALC macro. The resulting track balance is returned in register 0 and in the TRKCALC parameter list field STARBAL.

REMOVE={YES | <u>NO</u>}
indicates if a record is to be deleted from the track.

YES
specifies that the record identified by the record number (specified in the R keyword) is being deleted from the track. The track balance is incremented instead of decremented.

Note: YES is valid only on a FUNCTN=TRKBAL call.

NO
specifies that a record is not to be deleted from the track. NO is the default.

MAXSIZE={YES | <u>NO</u>}

YES
If the specified record does not fit, the largest length of a record with the specified key length that fits is returned (register 0).

Note: YES is valid only on a FUNCTN=TRKBAL call.

NO

Maximum size is not returned. NO is the default.

RKDD=*addr*—RX-type address, (2-12), (0), (14)

addr specifies a word containing a record number (1 byte), keylength (1 byte), and data length (2 bytes) (bytes 0, 1, and 2 and 3, respectively) or a register containing the record number, key length, and data length. R, K, and DD may be specified by this keyword, or you may use the following three keywords instead.

R=*addr*—RX-type address, (2-12), (0), (14), or n

you may specify either the address of the record number, or you may specify the record number using the low-order byte of a register or immediate data (n). Specify a decimal digit for n (immediate data).

K=*addr*—RX-type address, (2-12), (0), (14), or n

you may specify either the address of a field containing the hexadecimal value of the record's key length, or you may specify the record's key length using the low-order byte of a register or immediate data (n). Specify a decimal digit for n (immediate data).

DD=*addr*—RX-type address, (2-12), (0), (14), or n

you may specify either the address of a field containing the hexadecimal value of the record's data length, or you may specify the record's data length using the low-order two bytes of a register or immediate data (n). Specify a decimal digit for n (immediate data).

REGSAVE={YES | NO}

YES

specifies registers 1 through 14 are saved and restored in the caller-provided save area (pointed to by register 13) across the TRKCALC call. Otherwise, registers 1, 9, 10, 11, and 14 are modified. Registers 0 and 15 are always modified by a TRKCALC call.

NO

specifies registers are not saved across a TRKCALC call. NO is the default.

MF=I

specifies storage definition for the TRKCALC parameter list and parameter list initialization, using the given keywords, then calling the TRKCALC function. MF=I is the default.

TRKCALC—Execute Form

A remote parameter list is referred to and can be modified by the execute form of the TRKCALC macro. The TRKCALC routine is called. The description of the standard form of the macro provides the explanation of the function of each operand.

[*symbol*]	**TRKCALC**	**[FUNCTN={TRKBAL | TRKCAP}]** **[{,DEVTAB={*addr* | *} |** **,UCB={*addr* | *} | ,TYPE={*addr* | *}}]** **[,BALANCE={*addr* | *}]** **[,REMOVE={YES | <u>NO</u>}]** **[,MAXSIZE={YES | <u>NO</u>}]** **[{,RKDD=*addr* | ,R=*addr*,K=*addr*,DD=*addr*}]** **[,REGSAVE={YES | <u>NO</u>}]** **,MF=(E,*addr*)**

FUNCTN={TRKBAL | TRKCAP}
is coded as shown in the standard form. If this keyword is omitted, any specification of REMOVE, MAXSIZE, LAST, and the RX form of BALANCE is ignored. In addition, DEVTAB is assumed if UCB is coded and a failure occurs, or if TYPE is specified. When you use FUNCTN, one of the keywords (DEVTAB, UCB, or TYPE) must be specified to provide an information source.

DEVTAB=*addr* **| ***—RX-type address, (2-12), (0), (14)
is coded as shown in the standard form except for the * subparameter. Specify an * when you have inserted the address of the device characteristics table entry (DCTE) in the parameter list.

UCB=*addr* **| ***—RX-type address, (2-12), (0), (14)
is coded as shown in the standard form except for the * subparameter. Specify an * when you have inserted the address of the UCB in the parameter list.

TYPE=*addr* **| ***—RX-type address, (2-12), (0), (14)
is coded as shown in the standard form except for the * subparameter. Specify an * when you have inserted the address of the UCB type (UCBTYP) in the parameter list.

BALANCE=*addr* **| ***—RX-type address, (2-12), (0), (14)
is coded as shown in the standard form except for the * subparameter. Specify an * when you have inserted the balance in the parameter list.

REMOVE={YES | <u>NO</u>}
is coded as shown in the standard form.

MAXSIZE={YES | <u>NO</u>}
is coded as shown in the standard form.

RKDD=*addr*—RX-type address, (2-12), (0), (14)
is coded as shown in the standard form.

R=*addr*—RX-type address, (2-12), (0), (14) or n
is coded as shown in the standard form.

K=*addr*—RX-type address, (2-12), (0), (14), or n
is coded as shown in the standard form.

DD=*addr*—RX-type address, (2-12), (0), (14), or n
is coded as shown in the standard form.

REGSAVE={YES | NO}
is coded as shown in the standard form.

MF=(E,*addr*)
This operand specifies that the execute form of the TRKCALC macro instruction and an existing data management parameter list are used.

E
Coded as shown.

***addr*—RX-type address, (0), (1), (2-12), or (14)**
specifies an in-storage address of the parameter list.

TRKCALC—List Form

The list form of the TRKCALC macro constructs an empty, in-line parameter list. By coding only MF=L, you construct a parameter list, and the actual values can be supplied by the execute form of the TRKCALC macro. Any parameters other than MF=L are ignored.

[*symbol*]	**TRKCALC**	**MF=L**

TRKCALC—DSECT Only

This call gives a symbolic expansion of the parameter list for the TRKCALC macro. No DSECT statement is generated. If a name is specified on the macro call, it applies, after any necessary boundary alignment, to the beginning of the list. The macro-generated symbols all begin with "STAR".

[*symbol*]	**TRKCALC**	**MF=D**

Input Register Usage for All Forms of MF

Registers 0, 2 through 12, and **14** are available to provide input for keywords.

Register 1 is used only to provide the address of the parameter list for an MF=E call.

Register 13 may be used as input for keywords, if REGSAVE=YES is not specified.

Register 15 is used as a work register to build the TRKCALC parameter list for the MF=E call; it is not available as an input register.

Output from TRKCALC

FUNCTN=TRKBAL

Register 15=0
The record fits on the track. Register 0 and STARBAL contain the new track balance.

Register 15=4
Record does not fit on the track. If MAXSIZE=YES is specified, a partial record does not fit either. Register 0 and STARBAL are set to zero.

Register 15=8
Record does not fit on the track. MAXSIZE=YES is specified, and a partial record does fit. Register 0 and STARBAL are set to the maximum number of data bytes that fit on the remainder of the track with the specified keylength.

Note: The keylength is excluded from the count of maximum data bytes.

STARBAL
This is the track balance field of the TRKCALC parameter list. This field is first set to the track capacity if R=1, or to the supplied BALANCE value if R≠1, or to the calculated balance if R≠1 and BALANCE are omitted. STARBAL is updated to the new track balance if the record fits; otherwise, STARBAL is left with the input track balance value.

FUNCTN=TRKCAP

Register 15=0
Register 0 contains the number of records that fit on the track if R = 1, or the number of records that fit on the remainder of the track if R ≠ 1.

Register 15=4
No records of the length specified fit on a full track (R = 1) or a partial track (R ≠ 1). Register 0 is set to zero.

STARBAL
This is the track balance field of the TRKCALC parameter list. This field is first set to the track capacity if R=1, or to the supplied BALANCE value if R≠1, or to the calculated balance if R≠1 and BALANCE are omitted.

Return Codes from TRKCALC

The TRKCALC macro passes a return code in register 15. The return codes and their meanings are as follows:

Contents	Meaning
00(X'00')	Indicates that register 0 contains the new track balance
04(X'04')	Indicates that the record did not fit (register 0 = 0)
08(X'08')	Indicates that the record did not fit (Register 0 contains the maximum data length that does fit)

TRKCALC Macro Examples

In this example, TRKCALC is coded to determine how many records of a given size with 10-byte keys fit on an IBM 3380 track. After issuing the macro, the number of records is saved in NUMREC:

```
TRKCALC  FUNCTN=TRKCAP,TYPE=UTYPE,R=1,K=10,DD=DL,MF=(E,(1))
         .
         .
         ST   0,NUMREC    SAVE NUMBER OF RECORDS
         .
         .
DL       DC   H'xxxx'     DATA LENGTH
UTYPE    DC   X'0E'
NUMREC   DS   F           MAX # OF RECORDS
```

In this example, TRKCALC is coded to determine whether another record can fit on a track of a 3380, given a track balance.

```
TRKCALC  FUNCTN=TRKBAL,TYPE=UTYPE,R=REC,K=KL,DD=DD,BALANCE=BAL,
         MAXSIZE=YES,MF=(E,(1))
         .
         .
UTYPE    DC   X'0E'
REC      DC   X'xx'
KL       DC   X'xx'
DD       DC   H'xxxx'
BAL      DC   H'xxxx'
```

Releasing Unused Space from a DASD Data Set

Direct Access Device Storage Management (DADSM) supports the release of unused space that is allocated to sequential or partitioned data sets.[4] This partial release function is called when:

- The data set is closed (if the RLSE subparameter of SPACE was specified on its DD card).
- A restart is processing from a checkpoint in which the data set was extended after a checkpoint.
- A PARTREL macro is issued.

The PARTREL Macro

The PARTREL macro builds a parameter list and issues a LOAD, BASSM, DELETE sequence. This sequence partially releases the space allocated to a data set without an associated OPEN/CLOSE.

The PARTREL macro supports sequential and partitioned data sets on volumes with or without an indexed VTOC. The macro may be coded in the execute, DSECT, and list forms, but not the standard form. The calling program:

- Must be APF authorized.
- Must have allocated the volume to this task and must ensure it stays mounted during the PARTREL function.
- Must ensure that the data set is not open.
- Must not hold any locks.
- Must provide the address of an available standard register save area in general register 13.
- Must provide the associated parameter list and parameters in storage below 16 megabytes virtual.
- May be in any storage key.
- May run in either supervisor or problem program state.
- May include the CVAFTBL mapping macro ICVAFPRM, and test the CVFDFPFT field. If the CVFPREL bit is on, PARTREL is supported as described.

[4] The format-1 DSCB for the data set contains an identifier (DS1LSTAR) for the last data record written.

PARTREL—Execute Form

The execute form of the PARTREL macro is as follows:

[*symbol*]	**PARTREL**	**MF={(E,*addr*) | (E,(*reg*))}** **[,DSN={*addr* | (*reg*))}]** **[,ERASE={YES | NO | <u>TEST</u>}]** **[,MODE={<u>PGM</u> | SUP}]** **[,TIOT={<u>ENQ</u> | NOENQ}]** **[,UCB=(*reg*)]**

Except for MODE, all parameters default to the current contents of the parameter list. The MODE parameter defaults to PGM.

To provide a better understanding of these parameters, their descriptions include information about DADSM execution-time processing. These descriptions use the term "effective value" to designate the value used by DADSM for this request. The effective value may be:

- Specified as a parameter on the PARTREL macro.
- Provided as the parameter's associated value in the parameter list.
- Defined by DADSM from the information provided in the request.

MF={(E,*addr*) | (E,(*reg*))}
specifies the execute form of the macro and the address of an existing PARTREL parameter list.

addr—RX-type address, (*reg*)—(0-12)
specifies the PARTREL parameter list address.

DSN={*addr* | (*reg*))}
specifies the address of a 44-byte area that contains the data set name. The data set name must be left-justified, with any unused bytes defined as blanks.

addr—RX-type address, (*reg*)—(0), (2-12)
You must provide an effective value for DSN.

| **ERASE={YES | NO | TEST}]**
| specifies a residual data erase attribute (see "Deleting a Data Set from the
| VTOC (SCRATCH and CAMLST SCRATCH)" on page 33 for a
| description of erase attributes). ERASE=YES and ERASE=NO are
| mutually exclusive. The default is ERASE=TEST.

| **ERASE=YES**
| specifies that the area being released should be erased (overwritten
| with zeros) before it is made available for new allocations.

| **ERASE=NO**
| specifies that the area should not be erased. This specification
| overrides and RACF erase attribute.

ERASE=TEST
specifies that the associated RACF erase attribute is to be used.

MODE={PGM | SUP}
specifies that PARTREL is requested by a caller in problem program state (MODE=PGM) or in supervisor state (MODE=SUP). MODE=PGM is the default.

If the calling program is in supervisor state (and wants to be returned in supervisor state), the effective value of MODE must be SUP. If the calling program is in problem program state, the effective value of MODE must be PGM.

TIOT={ENQ | NOENQ}]
specifies the desired SYSZTIOT and SYSDSN ENQ processing within partial release. The default is ENQ.

TIOT=ENQ
specifies that partial release is to do its normal, exclusive ENQ on SYSZTIOT and SYSDSN. If either of these ENQ requests fails, PARTREL will terminate the request with a return code of X'08'.

TIOT=NOENQ
specifies that the caller has provided the necessary serialization. If partial release finds that the caller does not have exclusive use of SYSDSN, PARTREL will terminate the request with a return code of X'08'.

UCB=(*reg*)
specifies the address of the UCB for the volume on which the subject data set resides. The volume must be mounted, and you must ensure that it remains mounted.

(*reg*)—(0), (2-12)
You must provide an effective value for the UCB parameter.

PARTREL—List Form

The list form of PARTREL is specified as follows:

[*symbol*]	**PARTREL**	**MF=L** **[,DSN=*addr*]** **[,ERASE={YES | NO | TEST}]** **[,MODE={PGM | SUP}]** **[,TIOT={ENQ | NOENQ}]**

Notes:

1. *The execute form of the UCB parameter can not be specified on the list form.*
2. *The list form MODE parameter is for documentation only. The effective value of MODE is as specified or defaulted on the execute form.*

For an explanation of the parameters, see the execute form.

An example of the list form's expansion is:

```
 PRELPL    PARTREL MF=L
+PRELPL    DS     0F
+          DC     CL4'PREL'              EBCDIC 'PREL' FOR PARTREL
+          DC     AL2(PRL1E-PRELPL)      LENGTH OF PARAMETER LIST
+          DC     H'0'                   ERROR CODE RETURNED FROM
+*                                         PARTIAL RELEASE
+          DC     XL1'00'                PARAMETER FLAG BYTE
+          DC     XL3'00'                RESERVED
+          DC     A(0)                   ADDRESS OF DATA SET NAME
+          DC     A(0)                   UCB POINTER
+PRL1E     EQU    *                      END OF PARAMETER LIST
```

PARTREL—DSECT Form

The DSECT form of PARTREL is specified as follows:

[*symbol*]	**PARTREL**	**MF=D**

An example of the DSECT form's expansion is:

```
 PRELPL    PARTREL MF=D
+PRELPL    DSECT                         DSECT FOR PARAMETER LIST
+PRLPLID   DS     CL4                    EBCDIC 'PREL' FOR PARTREL
+PRLNGTH   DS     AL2                    LENGTH OF PARAMETER LIST
+PRERRCDE  DS     H                      ERROR CODE RETURNED FROM
+*                                         PARTIAL RELEASE
+PRLFLAG   DS     XL1                    PARAMETER FLAG BYTE
+PRLPGM    EQU    X'00'                  MODE=PGM (PROBLEM PROGRAM)
+PRLSUP    EQU    X'80'                  MODE=SUP (SUPERVISOR STATE)
+PRLTIOT   EQU    X'40'                  TIOT=NOENQ
+PRLNERAS  EQU    X'20'                  ERASE=NO
+PRLERASE  EQU    X'10'                  ERASE=YES
+PRLFRES   EQU    X'0F'                  RESERVED
+PRLRSVD   DS     XL3                    RESERVED
+PRLDSN    DS     A                      DATA SET NAME POINTER
+PRLUCB    DS     A                      UCB POINTER
+PRLEND    EQU    *                      END OF PARAMETER LIST
+PRLENGTH  EQU    PRLEND-PRELPL          LENGTH OF PARAMETER LIST
```

Return Codes From PARTREL

Control returns to the instruction following the last instruction generated by the PARTREL macro. If the data set was successfully processed, register 15 contains zeros. Otherwise, register 15 contains one of the following return codes. This is a cumulative list of DADSM partial release return codes. Some of these codes may not apply to the PARTREL macro.

Code	Meaning
02(X'02')	Unable to find extent in format-1 DSCB.
04(X'04')	Unable to find extent in format-3 DSCB.
08(X'08')	Either the required SYSZTIOT or SYSDSN ENQ failed, or an unrelated DEB indicates that another DCB is open to the data set.
12(X'0C')	Invalid parameter list.
16(X'10')	Permanent I/O error or unexpected CVAF error return code or installation exit rejected the request.
20(X'14')	DSN, or DSN pointer is invalid.
24(X'18')	Invalid UCB pointer.
28(X'1C')	Given DSORG is not supported.
32(X'20')	No room in the VTOC.

Allocating a DASD Data Set

The REALLOC macro builds a parameter list and issues an SVC 32 to allocate a new data set. You can code the macro in the execute, DSECT, and list forms, but not in the standard form. The calling program:

- Must be APF authorized.
- Must have allocated the volume to this task and must ensure it will stay mounted during the REALLOC function.
- Must not hold any locks.
- Must provide the associated parameter list and parameters in storage below 16-megabyte virtual.
- May use any storage key.
- May run in either supervisor or problem program state.

- Must note that REALLOC does not call RACF or catalog management.
- May include the CVAFTBL mapping macro ICVAFPRM, and test the CVFDFPFT field:
  - If the CVFPREL bit is on, REALLOC is supported as it is documented in this release.
  - If the CVFPREL bit is off and the CVFREALL bit is on, REALLOC is supported as documented in a prior release. (Absolute allocation is not supported.)
  - If neither bit is on, REALLOC is not supported.

The calling program must provide the REALLOC macro with one or more model DSCBs. You can use the OBTAIN macro to get the DSCBs from other data sets and modify them for the request. DADSM uses these model DSCBs to validate the allocation request, and to construct those DSCBs that are written to the VTOC for the requested allocation.

The ALLOC parameter for the REALLOC macro defines the allocation request as either absolute (ABS) or movable[5] (MOV).

An absolute request provides a set of allocation parameters, a full format-1 DSCB, an optional format-2 DSCB, and an optional format-3 DSCB that describe the attributes of the desired data set:

- Support is provided for data sets with a user label extent and for ISAM data set allocations,[6] but is not limited to these two types.
- The number of extents to be allocated, and their absolute placement on the volume, are defined by the format-1 DSCB and one (optional) format-3 DSCB.

An absolute request is limited to a single volume with indexed VTOC support.

A movable request provides a set of allocation parameters and a partial DSCB[7] that describe the attributes of the desired data set:

- Data sets with a user label extent, ISAM data sets, and absolute track allocated data sets are not supported.

---

5 The requested data set's allocation is not sensitive to its placement on the volume. This is specifically NOT a reference to the format-1 DSCB bit DS1DSGU (unmovable bit), which may be either on or off in an ALLOC=MOV request's partial DSCB. That is, the data set may subsequently contain location-dependent information.

6 See the description of the F2DSCB parameter in the REALLOC macro execute form for more information.

7 The partial DSCB (mapped by the IECPDSCB macro) consists of the first 98 bytes of a format-1 DSCB followed by two full words: PDPRIQTY (primary space request in tracks), and PDDIRQTY (number of directory blocks).

- The maximum number of extents that may be allocated is determined by the data set organization (PD1DSORG) and the data set indicator (PD1DSIND) bytes in the partial DSCB. If PD1DSORG indicates a VSAM data set organization and PD1DSIND indicates that the data set is cataloged in an Integrated Catalog Facility catalog, the maximum number of extents is 123. Otherwise, the maximum number of extents is 16.

A movable request is limited to a single volume with or without indexed VTOC support.

REALLOC—Execute Form

The format of the REALLOC macro in execute form is:

[*symbol*]	**REALLOC**	**MF={(E,***addr***) | (E,(***reg***))}** **{,ALLOC={ABS | MOV}}** **[,DSSIZE={***addr* **| (***reg***)}]** **[,F2DSCB={***addr* **| (***reg***)}]** **[,F3DSCB={***addr* **| (***reg***)}]** **[,MINAU={***addr* **| (***reg***)}]** **[,PDSCB={***addr* **| (***reg***)}]** **[,PDSDIR={***addr* **| (***reg***)}]** **[,UCB=***(reg)***]**

All parameters except ALLOC default to the current contents of the referenced parameter list. The ALLOC parameter defaults to MOV.

To provide a better understanding of this macro's parameters, their descriptions include information about DADSM execution-time processing. These descriptions use the term "effective value" to designate the value used by DADSM for this request. The effective value may be:

- Specified as a parameter for the REALLOC macro.
- Provided as the parameter's associated value in the referenced parameter list.
- Defined by DADSM from information provided in the request.

MF={(E,*addr* **| (E,(***reg***))}**
specifies the execute form of the macro and the address of a REALLOC parameter list.

addr—RX-type address, (*reg*)—(0-12)
specifies the address of the REALLOC parameter list.

ALLOC={ABS | ***MOV*****}**
specifies that the REALLOC request is for absolute extents (ALLOC=ABS) or for a movable allocation (ALLOC=MOV). ALLOC=MOV is the default.

If you want absolute allocation, the effective value of ALLOC must be ABS.

DSSIZE={*addr* **| (***reg***)}**
specifies the size of the data set to be allocated in tracks. The DSSIZE parameter is invalid for an ALLOC=ABS request.

addr—RX-type address
specifies the address of a full word that contains the data set size.

(*reg*)—(0), (2-12)
specifies a register that contains the size of the data set.

You must provide an effective value for DSSIZE for an ALLOC=MOV request. The PDPRIQTY field of the partial DSCB is ignored.

DADSM assumes that you have provided the effective value of DSSIZE in tracks even if the PD1SCALO flag byte of the partial DSCB indicates a cylinder request, X'C0', or an average block request, X'40'.

If the PD1SCALO flag byte of the partial DSCB indicates a cylinder request, X'C0', or an average block with round request, X'41', the effective value of DSSIZE is rounded up to the next full cylinder.

F2DSCB={*addr* | (*reg*)}
specifies the in-storage address of a format-2 DSCB. This DSCB is used as a model to construct the allocated data set's format-2 DSCB.

The F2DSCB parameter is invalid for an ALLOC=MOV request.

addr—**RX-type address, (*reg*)—(0), (2-12)**

You may provide an effective value for F2DSCB in an ALLOC=ABS request when the DS1DSORG flag byte of the given format-1 DSCB is X'80' (indexed sequential organization).

Because REALLOC allocates on the basis of a single volume for each request and because a multivolume ISAM data set is defined with one format-2 DSCB (on the first volume only), the associated DADSM allocation processing routines do not require a format-2 DSCB (that is, multiple REALLOC requests may be used to to allocate a multivolume ISAM data set).

F3DSCB={*addr* | (*reg*)}
specifies the in-storage address of a format-3 DSCB. This DSCB is used as a model to construct the allocated data set's format-3 DSCB.

The F3DSCB parameter is invalid for an ALLOC=MOV request.

addr—**RX-type address, (*reg*)—(0), (2-12)**

You must provide an effective value for F3DSCB in an ALLOC=ABS request when the DS1NOEPV byte of the format-1 DSCB indicates more than three extents (or when the DS1NOEPV byte indicates more than two extents and the DS1EXT1 extent type indicator is X'40'; a user label extent).

The REALLOC request is limited to a maximum of 16 extents. No more than one format-3 DSCB can be specified.

You must provide an effective value of zero for the F3DSCB in an ALLOC=ABS request when the DS1NOEPV byte of the format-1 DSCB indicates that there are less than four extents (or when the DS1NOEPV byte indicates that there are less than three extents and the DS1EXT1 extent type indicator is X'40'; a user label extent).

MINAU={*addr* | (*reg*)}
specifies the size of the minimum allocation unit in tracks. All primary extents for this data set are in multiples of this minimum allocation unit. This minimum does not apply to subsequent extensions of the data set.

The MINAU parameter is invalid on an ALLOC=ABS request.

addr—RX-type address
specifies the address of a full word containing the minimum allocation unit.

(*reg*)—**(0), (2-12)**
specifies a register containing the minimum allocation unit.

The MINAU parameter has no effect on the requested allocation if:

- You provide an effective value of zero.
- The PD1SCALO flag byte of the partial DSCB indicates either a cylinder request, X'C0', or an average block with round request, X'41'.

Otherwise, the effective value of DSSIZE must be a multiple of the effective value of MINAU.

PDSCB={*addr* | (*reg*)}
specifies the address of a partial DSCB (for ALLOC=MOV) or the in-storage address of a full format-1 DSCB (for ALLOC=ABS). This DSCB is used as a model to construct the allocated data set's format-1 DSCB.

addr—**RX-type address,** (*reg*)—**(0), (2-12)**

You must provide an effective value for the PDSCB parameter.

PDSDIR={*addr* | (*reg*)}
specifies the number of 256-byte directory blocks for a partitioned data set (PDS).

addr—RX-type address
specifies an in-storage address of a full word containing the number of 256-byte PDS directory blocks.

(*reg*)—**(0), (2-12)**
specifies a register containing the number of 256-byte PDS directory blocks.

You must provide an effective value for PDSDIR when partitioned organization is indicated:

- The DS1DSORG flag byte of the format-1 DSCB is X'02' (ALLOC=ABS).
- The PD1DSORG flag byte of the partial DSCB is X'02' (ALLOC=MOV).

For an ALLOC=MOV request, you can specify the effective value of PDSDIR in the PDDIRQTY field of the partial DSCB. The PDDIRQTY field is used if and only if, the effective REALLOC parameter list value of PDSDIR is zero.

Do not specify an effective value for PDSDIR when a PDS is not indicated.

UCB=(*reg*)
specifies the address of the UCB for the volume in which the data set is to be allocated. The volume must be mounted, and you must ensure that it remains mounted.

(*reg*)—(0), (2-12)

You must provide an effective value for the UCB parameter.

REALLOC—List Form

The list form of the REALLOC macro is specified as follows:

[*symbol*]	**REALLOC**	**MF=L** **[,ALLOC={ABS | <u>MOV</u>}]** **[,F2DSCB=*addr*]** **[,F3DSCB=*addr*]** **[,PDSCB=*addr*]**

Notes:

1. *The execute form parameters DSSIZE, MINAU, PDSDIR, and UCB can not be specified on the list form.*

2. *The list form's ALLOC parameter affects the tests made by the REALLOC macro at assembly time and the contents of the parameter list.*

3. *The effective value of ALLOC is as specified or defaulted on the execute form.*

See the execute form for an explanation of the parameters.

An example of the list form expansion is:

```
 REALPL    REALLOC MF=L
+REALPL    DS     0F
+          DC     CL4'REAL'                  EBCDIC 'REAL' FOR REALLOC
+          DC     AL2(RAL1E-REALPL)          LENGTH OF PARAMETER LIST
+          DC     H'0'                       ERROR CODE RETURNED FROM
+*                                              ALLOCATE (SVC 32)
+          DC     XL1'00'                     PARAMETER FLAG BYTE
+          DC     XL3'00'                     RESERVED
+          DC     F'0'                        DATA SET SIZE
+          DC     F'0'                        MINIMUM ALLOCATION UNIT
+          DC     A(0)                        PARTIAL DSCB POINTER
+          DC     A(0)                        UCB POINTER
+          DC     F'0'                        PDS DIRECTORY QUANTITY
+          DC     A(0)                        FORMAT 2 DSCB POINTER
+          DC     A(0)                        FORMAT 3 DSCB POINTER
+RAL1E     EQU    *                           END OF PARAMETER LIST
```

REALLOC—DSECT Only

The DSECT form of REALLOC is specified as follows:

[*symbol*]	**REALLOC**	**MF=D**

An example of the DSECT form expansion is:

```
 REALPL    REALLOC MF=D
+REALPL    DSECT                            DSECT FOR PARAMETER LIST
+RALPLID   DS     CL4                       EBCDIC 'REAL' FOR REALLOC
+RALNGTH   DS     AL2                       LENGTH OF PARAMETER LIST
+RAERRCDE  DS     H                         ERROR CODE RETURNED FROM
*                                             ALLOCATE (SVC 32)
+RALFLAG   DS     XL1                       PARAMETER FLAG BYTE
+RALMOV    EQU    X'00'                     ALLOC=MOV
+RALABS    EQU    X'80'                     ALLOC=ABS
+RALFRES   EQU    X'7F'                     RESERVED
+RALRSVD   DS     XL3                       RESERVED
+RALDSSZ   DS     F                         DATA SET SIZE
+RALMAU    DS     F                         MINIMUM ALLOCATION UNIT
+RALPDSCB  DS     A                         PARTIAL DSCB POINTER
+RALUCB    DS     A                         UCB POINTER
+RALDQTY   DS     F                         PDS DIRECTORY QUANTITY
+RAL2DSCB  DS     A                         FORMAT 2 DSCB POINTER
+RAL3DSCB  DS     A                         FORMAT 3 DSCB POINTER
+RALEND    EQU    *                         END OF PARAMETER LIST
+RALENGTH  EQU    RALEND-REALPL             LENGTH OF PARAMETER LIST
```

Return Codes from REALLOC

Control returns to the instruction following the SVC 32 generated by the REALLOC macro. If the data set was successfully allocated, register 15 contains zeros. Otherwise, register 15 contains one of the following return codes:

Note: This is a cumulative list of DADSM allocation return codes. Some of these codes may not apply to the REALLOC macro.

Return Code (R15)	**Reason Code (R0)**	**Meaning**
004(X'04')		Data set name of request already exists on this volume. Initial allocation not possible under the name given.
008(X'08')		No room available in the VTOC or VTOC index.
012(X'0C')		One of the following errors was encountered: • Permanent I/O error • Error returned by CVAF
016(X'10')		Requested absolute track not available.
020(X'14')		Requested quantity not available.
024(X'18')		Average record length greater than 65535 bytes.
028(X'1C')		Incorrect DSORG or DISP in an ISAM index request.
032(X'20')		No prime area requested for ISAM data set.
036(X'24')		ISAM prime area must be requested before overflow.
040(X'28')		Space requested must begin on a cylinder boundary.
044(X'2C')		Duplicate ISAM DSNAME element.
048(X'30')	01(X'01')	Invalid REALLOC parmlist ID.
	02(X'02')	Invalid REALLOC parmlist length.
	03(X'03')	REALLOC request, but neither ALLOC=ABS nor ALLOC=MOV is specified.

Return Code (R15)	Reason Code (R0)	Meaning
	04(X'04')	Invalid data set size specified for ALLOC=MOV.
	05(X'05')	The data set is not a PDS for an ALLOC=MOV.
	06(X'06')	The data set is not a PDS for an ALLOC=ABS.
	07(X'07')	No directory quantity specified for a PDS for an ALLOC=ABS.
052(X'34')		Invalid JFCB or partial DSCB pointer.
056(X'38')		Requested directory space is larger than the space available on this volume.
060(X'3C')		Nonindexed VTOC not supported for REALLOC ALLOC=ABS request.
064(X'40')		Invalid user label request.
068(X'44')		Invalid UCB pointer.
072(X'48')		DOS VTOC cannot be converted to an OS VTOC.
076(X'4C')		No space parameter given for a new data set or zero space requested at absolute track zero.
080(X'50')		Invalid request for ISAM index.
084(X'54')		ISAM multivolume index not allowed.
088(X'58')		Invalid ISAM DSNAME element.
092(X'5C')		ISAM multivolume overflow request not allowed.
096(X'60')		ABSTR and CYL requests conflict.
100(X'64')		CYL and CONTIG requests conflict.
104(X'68')		Invalid space subparameter.
108(X'6C')		Primary space request for an ISAM data set is zero, or primary space for an ABSTR request is zero.
112(X'70')		Duplicate ISAM index request.
116(X'74')		User labels not supported.
120(X'78')		Invalid combination of values for DSSIZE and MINAU.

Return Code (R15)	Reason Code (R0)	Meaning
124(X'7C')		DSSIZE is not a multiple of MINAU.
128(X'80')		Directory space requested is larger than primary space.
132(X'84')		Space request must be ABSTR for DOS volume.
136(X'88')		Invalid F3DSCB pointer.
140(X'8C')		ISAM index must be requested before prime area.
144(X'90')		Last concatenated DD card unnecessary or invalid for this ISAM data set.
148(X'94')		Overlapping extents in the VTOC.
152(X'98')		Overlapping DOS split cylinder extents in the VTOC.
156(X'9C')		DADSM allocation terminated because of possible VTOC errors.
160(X'A0')		ISAM allocation terminated because of possible VTOC errors.
164(X'A4')		Allocation terminated because of DOS stacked pack format.
168(X'A8')		RACF define failed, data set profile already defined.
172(X'AC')		User not authorized to define data set.
176(X'B0')		Installation exit rejected this request with a return code of 8. No further volumes should be attempted.
180(X'B4')		Installation exit rejected this request with a return code of 4. For a nonspecific volume request, another volume may be attempted.
184(X'B8')		RACF define with modeling specified and model not found.
188(X'BC')		Invalid F2DSCB pointer.

Message Displays on the IBM 3480 Magnetic Tape Subsystem

The MSGDISP macro displays a message on the IBM 3480. With MSGDISP, you can specify the message to be displayed and how to display it (for example, steady or flashing). The six main parameters of the macro and their functions are:

MOUNT	Displays an "M" in position 1 of the display area during a mount request until a volume is loaded and made ready. The "M" is followed by the volume serial number and label type.
VERIFY	Shows that a volume has been accepted by displaying its serial number and label type in positions 2 through 8.
RDY	Displays text in positions 2 through 7 while a data set is open.
DEMOUNT	Displays a volume disposition indicator in position 1 until a volume is demounted.
RESET	Clears the display area.
GEN	Provides the full range of display options, including the option to alternate two messages.

All except the RDY parameter require that you be in supervisor state, have a storage protect key of 0 through 7, or be authorized by the authorized program facility.

For MVS/XA, you may specify the IOSLEVEL (priority) of the request with the FORCE parameter. IOSLEVEL support replaces single-level I/O quiescing with multilevel quiescing; the higher the IOSLEVEL value, the greater your priority to control the device.

The MSGDISP macro generates a parameter list as input to an SVC routine.

MSGDISP may be coded in the standard, execute, and list forms.

The formats for specifying MSGDISP with the six main parameters, and the return codes generated by MSGDISP, are given in the sections that follow.

MSGDISP—Displaying a Mount Message

The format for specifying MSGDISP with the MOUNT parameter is:

[*symbol*]	**MSGDISP**	**MOUNT** **,UCB=***(reg)* **[,FORCE={NO | YES |** *n* **|** *keyword* **| (***reg***)}]** **[,LABEL={'A' | 'N' | 'S' | 'X' |** *addr***}]** **[,MF={L | (E,***addr***)}]** **[,SER={'***volser***' |** *addr***}]** **[,TEST={NO | YES}]** **[,WAIT={NO | YES}]**

MOUNT

displays an "M" in position 1 of the display area during a mount request. The "M" is followed by a volume serial number and label type. The display flashes on and off until a volume is loaded and ready. If the device is ready at the time a mount request is issued, the "M" is not displayed.

UCB=*(reg)*—(2-12)

specifies a register containing the UCB address for the device.

FORCE={NO | YES | *n* | *keyword* | *(reg)*}

specifies the priority (IOSLEVEL) for the display request's I/O. The higher the IOSLEVEL value, the greater the priority.

If you do not specify the FORCE parameter, the default is FORCE=NO.

NO

prevents execution of a display request for a device whose I/O is being quiesced. The IOSLEVEL is set to the installation default, as indicated in the CVTIONLV field of the CVT.

YES

forces execution of a display request for a device even if its I/O is being quiesced. The IOSLEVEL is set to 9, the highest priority.

n

specifies a decimal number from 1 to 9, to be used as the IOSLEVEL value. A high number indicates a higher priority request for the device.

keyword

specifies a label equated to an IOSLEVEL value:

NORMAL	1
QUIESCE	2
DAVV	3
DDR	4
DYNPATH	5
UNCRSV	6
CHPRCVY	7
SCHRCVY	8
FDEV	9

(reg)

specifies that the low-order byte of the indicated register (2 through 12) contains a value between 1 and 9, indicating the IOSLEVEL.

LABEL={'A' | 'N' | 'S | 'X' | *addr*}

displays the label type of the mounted volume in position 8. If you specify an unknown label type other than a blank, a "?" is displayed.

'A'

specifies ISO/ANSI/FIPS (AL) or ISO/ANSI/FIPS with user labels (AUL). Specify in apostrophes.

'N'

specifies no labels (NL), LTM (DOS), or bypass label processing (BLP). Specify in apostrophes.

'S'

specifies IBM Standard (SL) or IBM Standard with user labels (SUL). Specify in apostrophes.

'X'

specifies nonstandard labels (NSL). Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)

specifies an in-storage address of an area containing an "A", "N", "S", or "X" (see the following explanations of these characters). For MF=L, you may only specify an A-type address.

MF={L | (E,*addr*)}

specifies either the execute or the list form of MSGDISP. If you do not specify this parameter, the standard form of the macro is used.

L

specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)

specifies that the execute form of the macro and an existing parameter list are used.

addr—RX-type address, (1), or (2-12)

specifies an in-storage address of the parameter list.

SER={'*volser*' | *addr*}

specifies the serial number of the volume to be mounted. The serial number is displayed in positions 2 through 7. If you do not specify SER, the system supplies the volume serial number. If the serial number is not available, a scratch volume is used, unless the volume use attribute indicates a default of "PRIVAT".

'*volser*'

specifies the volume serial number as a literal. Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)

specifies an in-storage address of the volume serial number. For MF=L, you may only specify an A-type address.

TEST={NO | YES}

specifies whether to test the UCB to determine if the device is capable of displaying messages.

NO

specifies that the SVC routine will test the UCB.

YES

specifies testing the UCB before the SVC call.

Note: TEST=YES requires you to include the UCB mapping macro (IEFUCBOB) in the source code.

WAIT={NO | <u>YES</u>}

specifies when control is returned to you.

NO

specifies that control is to be returned before I/O is complete. I/O return codes are not returned, and I/O errors are recorded in the same manner as any permanent error by the error recovery procedure.

<u>YES</u>

specifies that control is to be returned after I/O is complete.

MSGDISP—Displaying a Verify Message

The format for specifying MSGDISP with the VERIFY parameter is:

[*symbol*]	**MSGDISP**	**VERIFY** **,UCB=***(reg)* **[,FORCE={<u>NO</u> | YES |** *n* **|** *keyword* **|** *(reg)***}]** **[,LABEL={'A' | 'N' | '<u>S</u>' | 'X' |** *addr***}]** **[,MF={L | (E,***addr***)}]** **[,SER={'***volser***' |** *addr***}]** **[,TEST={<u>NO</u> | YES}]** **[,WAIT={NO | <u>YES</u>}]**

VERIFY

displays the serial number and label type of a volume that has been accepted in positions 2 through 8. Position 1 remains blank. The display lasts until the next display request is executed.

UCB=*(reg)*—(2-12)

specifies a register containing the UCB address for the device.

FORCE={<u>NO</u> | YES | *n* **|** *keyword* **|** *(reg)***}**

specifies the priority (IOSLEVEL) for the display request's I/O. The higher the IOSLEVEL value, the greater the priority.

If you do not specify the FORCE parameter, the default is FORCE=NO.

<u>NO</u>

prevents execution of a display request for a device whose I/O is being quiesced. The IOSLEVEL is set to the installation default, as indicated in the CVTIONLV field of the CVT.

YES

forces execution of a display request for a device even if its I/O is being quiesced. The IOSLEVEL is set to 9, the highest priority.

n

specifies a decimal number from 1 to 9, to be used as the IOSLEVEL value. A high number indicates a higher priority request for the device.

keyword

specifies a label equated to an IOSLEVEL value:

NORMAL	1
QUIESCE	2
DAVV	3
DDR	4
DYNPATH	5
UNCRSV	6
CHPRCVY	7
SCHRCVY	8
FDEV	9

(*reg*)

specifies that the low-order byte of the indicated register (2 through 12) contains a value between 1 and 9, indicating the IOSLEVEL.

LABEL={'A' | 'N' | 'S' | ' ' | *addr*}

specifies label type of the mounted volume in position 8 of the display. If an unknown label type other than a blank is specified, a "?" is displayed.

'A'

specifies ISO/ANSI/FIPS (AL) or ISO/ANSI/FIPS with user (AUL) labels. Specify in apostrophes.

'N'

specifies no labels (NL), LTM (DOS), or bypass label processing (BLP). Specify in apostrophes.

'S'

specifies IBM Standard (SL) or IBM Standard with user (SUL) labels. Specify in apostrophes.

'X'

specifies nonstandard (NSL) labels. Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)

specifies an in-storage address of an area containing an "A", "N", "S", or "X" (see explanations below for these characters). For MF=L, you may only specify an A-type address.

MF={L | (E,*addr*)}

specifies either the execute or list form of MSGDISP. If you do not specify this parameter, the standard form of the macro is used.

L

specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)

specifies that the execute form of the macro and an existing parameter list is to be used.

addr—RX-type address, (1), or (2-12)

specifies an in-storage address of the parameter list.

SER={'*volser*' | *addr*}

specifies the serial number of the volume that has been verified. The serial number displays in positions 2 through 7. If you do not specify SER, the system supplies the volume serial number. If the serial number is not available, a scratch volume is used, unless the volume use attribute indicates a default of "PRIVAT".

'*volser*'

specifies the volume serial number as a literal. Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)

specifies an in-storage address of the volume serial number. For MF=L, you may only specify an A-type address.

TEST={NO | <u>YES</u>}

specifies whether to test the UCB to determine if the device is capable of displaying messages.

<u>NO</u>

specifies that the SVC routine will test the UCB.

YES

specifies testing the UCB before the SVC call.

Note: TEST=YES requires you to include the UCB mapping macro (IEFUCBOB) in the source code.

WAIT={NO | <u>YES</u>}

specifies when control is to be returned to you.

NO

specifies that control is to be returned before I/O is complete. I/O return codes are not returned, and I/O errors are recorded in the same manner as any permanent error by the error recovery procedure.

<u>YES</u>

specifies that control is to be returned after I/O is complete.

MSGDISP—Displaying a Ready Message

The format for specifying MSGDISP with the RDY parameter is:

[*symbol*]	**MSGDISP**	**RDY** **,DCB=*addr*** **[,MF={L | (E,*addr*)}]** **[,TXT={'*msgtxt*' | *addr*]**

RDY

displays the text supplied in the TXT parameter in positions 2 through 7 while the data set is open. The display is steady (not flashing) and is enclosed in parentheses. The display is also written to the tape pool console (routing code 3, descriptor code 7).

DCB=*addr*

specifies the address of a DCB opened to a data set on the mounted volume. If multiple devices are allocated, the message display is directed to the one containing the volume currently in use.

Note: If multiple devices or multiple volumes are allocated, you may update a message display after an end-of-volume condition by using the EOV exit specified in a DCB exit list. In the case of a concatenated data set with unlike characteristics, the DCB OPEN exit may be used to update the display.

addr—RX-type address, A-type address, or (2-12)

specifies an in-storage address of the opened DCB. For MF=L, you may only specify an A-type address.

MF={L | (E,*addr*)}

specifies either the execute or list form of MSGDISP. If this parameter is not specified, the standard form of the macro is used.

L

specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)

specifies that the execute form of the macro and an existing parameter list is to be used.

addr—RX-type address, (1), or (2-12)

specifies an in-storage address of the parameter list.

TXT={'*msgtxt*' | *addr*}

specifies up to six characters to display in positions 2 through 7 of the display. If you do not specify TXT, blanks are displayed.

'*msgtxt*'

specifies the text as a literal. Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)

specifies an in-storage address of an area containing the text to be displayed. For MF=L, you may only specify an A-type address.

MSGDISP—Displaying a Demount Message

The format for specifying MSGDISP with the DEMOUNT parameter is:

[*symbol*]	**MSGDISP**	**DEMOUNT** **,UCB=***(reg)* **[,DISP={'D' | 'K' | 'R' |** *addr***}]** **[,FORCE={NO | YES |** *n* **|** *keyword* **| (***reg***)}]** **[,MF={L | (E,***addr***)}]** **[,MLABEL={'A' | 'N' | 'S' | 'X' |** *addr***}]** [,MSER={'*volser-to-mount*' | *addr*}] [,SER={'*volser*' | *addr*}] [,TEST={NO | YES}] [,WAIT={NO | YES}]

DEMOUNT
Displays a volume disposition indicator in position 1 until the volume is demounted. Optionally, you may display the serial number of the volume to be demounted at the same time. The display flashes on and off. If a volume is not mounted on the device when the display request is executed, blanks are displayed.

The demount message may be displayed alternately (flashing) with a mount message for the next volume by specifying the MSER parameter.

UCB=*(reg)*—(2-12)
specifies a register containing the UCB address for the device.

DISP={'D' | 'K' | 'R' | *addr***}**
specifies the character to display in position 1, representing the volume disposition.

'D'
Demount a public volume. Specify in apostrophes.

Note: "D" also displays when you specify an invalid character or when the volume use attribute is unknown (as in an automatic volume recognition (AVR) error when reading a label).

'K'
Keep a private volume and return it to the library. Specify in apostrophes.

'R'
Retain a private volume near the device for further use. Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)
specifies an in-storage address of an area containing a "D", "K", or "R". For MF=L, you may only specify an A-type address.

FORCE={NO | YES | *n* **|** *keyword* **| (***reg***)}**
specifies the priority (IOSLEVEL) for the display request's I/O. The higher the IOSLEVEL value, the greater the priority.

If you do not specify the FORCE parameter, the default is FORCE=NO.

NO
: prevents execution of a display request for a device whose I/O is being quiesced. The IOSLEVEL is set to the installation default, as indicated in the CVTIONLV field of the CVT.

YES
: forces execution of a display request for a device even if its I/O is being quiesced. The IOSLEVEL is set to 9, the highest priority.

n
: specifies a decimal number from 1 to 9, to be used as the IOSLEVEL value. A high number indicates a higher priority request for the device.

keyword
: specifies a label equated to an IOSLEVEL value:

NORMAL	1
QUIESCE	2
DAVV	3
DDR	4
DYNPATH	5
UNCRSV	6
CHPRCVY	7
SCHRCVY	8
FDEV	9

(*reg*)
: specifies that the low-order byte of the indicated register (2 through 12) contains a value between 1 and 9, indicating the IOSLEVEL.

MF={L | (E,*addr*)}
specifies either the execute or list form of MSGDISP. If you do not specify this parameter, the standard form of the macro is used.

L
: specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)
: specifies that the execute form of the macro and an existing parameter list is to be used.

addr—RX-type address, (1), or (2-12)
: specifies an in-storage address of the parameter list.

MLABEL={'A' | 'N' | 'S' | 'X' | *addr*}
displays the label type of the volume to be loaded and made ready following a demount, in position 8. If you specify an unknown label type other than a blank, a "?" is displayed. You may only specify this parameter if you also specify the MSER parameter.

'A'

specifies ISO/ANSI/FIPS (AL) or ISO/ANSI/FIPS with user (AUL) labels. Specify in apostrophes.

'N'

specifies no labels (NL), LTM (DOS), or bypass label processing (BLP). Specify in apostrophes.

'S'

specifies IBM Standard (SL) or IBM Standard with user (SUL) labels. Specify in apostrophes.

'X'

specifies nonstandard (NSL) labels. Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)

specifies an in-storage address of an area containing an "A", "N", "S", or "X" (see the following explanations of these characters). For MF=L, you may only specify an A-type address.

MSER={'*volser-to-mount*' | *addr*}

displays the mount message for the next volume alternately (flashing) with the demount message. The display continues untilyou demount the current volume. At that time, the mount message will display (flashing) until you load the volume and make the device ready. If no volume is mounted at the time the demount and mount messages are executed, only the mount message will display (flashing) until the volume is loaded and ready.

'*volser-to-mount*'

specifies the volume serial number of the volume to be mounted, as a literal. Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)

specifies an in-storage address of the volume serial number of the volume to be mounted. For MF=L, you may only specify an A-type address.

SER={'*volser*' | *addr*}

specifies the serial number of the volume to be demounted. The serial number is displayed in positions 2 through 7. If you do not specify SER, the system supplies the volume serial number. If the serial number is not available, a scratch volume is used, unless the volume use attribute indicates a default of "PRIVAT".

'*volser*'

specifies the volume serial number as a literal. Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)

specifies an in-storage address of the volume serial number. This parameter is not valid for the MF=L form. For MF=L, you may only specify an A-type address.

TEST={NO | YES}

specifies whether to test the UCB to determine if the device is capable of displaying messages. message display SVC routine.

NO

specifies that the SVC routine will test the UCB.

YES

specifies testing the UCB before the SVC call.

Note: TEST=YES requires you to include the UCB mapping macro (IEFUCBOB) in the source code.

WAIT={NO | YES}

specifies when control is to be returned to you.

NO

specifies that control is to be returned before I/O is complete. I/O return codes are not returned, and I/O errors are recorded in the same manner as any permanent error by the error recovery procedure.

YES

specifies that control is to be returned after I/O is complete.

MSGDISP—Resetting the Message Display

The format for specifying MSGDISP with the RESET parameter is:

[*symbol*]	**MSGDISP**	**RESET** **,{UCB=*(reg)* | ,UCBL=*addr*}** **[,FORCE={NO | YES | *n* | *keyword* | *(reg)*}]** **[,MF={L | (E,*addr*)}]** **[,TEST={NO | YES}]** **[,WAIT={NO | YES}]**

RESET

clears all existing data on the display. If you specify WAIT=NO and the last service requested was a demount, the display is not cleared.

After being cleared, the display will show the device's *internal* status message (for example, a message indicating that the device is ready).

UCB=*(reg)*—(2-12)

specifies a register containing the UCB address for the device.

UCBL=*addr*—RX-type address, A-type address, (0), or (2-12)

specifies the address of a list containing a maximum of 64 words. Each word in the list contains the address of a UCB representing a device whose display is to be reset. The end of the list is indicated by a '1' in the high order bit of the last address in the list. If an error is encountered while processing the list, register 1 points to the associated UCB when you regain control.

You cannot specify UCBL with TEST=YES and WAIT=NO.

FORCE={NO | YES | *n* | *keyword* | *(reg)*}

specifies the priority (IOSLEVEL) for the display request's I/O. The higher the IOSLEVEL value, the greater the priority.

If you do not specify the FORCE parameter, the default is FORCE=NO.

NO

prevents execution of a display request for a device whose I/O is being quiesced. The IOSLEVEL is set to the installation default, as indicated in the CVTIONLV field of the CVT.

YES

forces execution of a display request for a device even if its I/O is being quiesced. The IOSLEVEL is set to 9, the highest priority.

n

specifies a decimal number from 1 to 9, to be used as the IOSLEVEL value. A high number indicates a higher priority request for the device.

keyword

specifies a label equated to an IOSLEVEL value:

NORMAL	1
QUIESCE	2
DAVV	3
DDR	4
DYNPATH	5
UNCRSV	6
CHPRCVY	7
SCHRCVY	8
FDEV	9

(*reg*)

specifies that the low-order byte of the indicated register (2 through 12) contains a value between 1 and 9, indicating the IOSLEVEL.

MF={L | (E,*addr*)}

specifies either the execute or the list form of MSGDISP. If you do not specify this parameter, the standard form of the macro is used.

L

specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)

specifies that the execute form of the macro and an existing parameter list is to be used.

addr—RX-type address, (1), or (2-12)

specifies the address of the parameter list.

TEST={NO | YES}

specifies whether to test the UCB to determine if the device is capable of displaying messages.

<u>**NO**</u>

specifies that the SVC routine will test the UCB.

YES

specifies testing the UCB before the SVC call. You cannot specify TEST=YES if you also specify the UCBL parameter.

Note: TEST=YES requires you to include the UCB mapping macro (IEFUCBOB) in the source code.

WAIT={NO | <u>YES</u>}

specifies when control is to be returned to you.

NO

specifies that control is to be returned before I/O is complete. I/O return codes are not returned, and I/O errors are recorded in the same manner as any permanent error by the error recovery procedure.

You cannot specify WAIT=NO if you also specify the UCBL parameter.

<u>**YES**</u>

specifies that control is to be returned after I/O is complete.

Note: Demount messages can be reset only if WAIT=YES is specified.

MSGDISP—Providing the Full Range of Display Options

The format for specifying MSGDISP with the GEN parameter is:

[*symbol*]	MSGDISP	**GEN** **,UCB=***(reg)* **[,FLASH={STEADY | STEADY2** **| <u>BLINK</u> | BLINK2 | ALT}]** **[,FORCE={<u>NO</u> | YES |** *n* **|** *keyword* **| (***reg***)}]** **[,MF={L | (E,***addr***)}]** **[,TEST={<u>NO</u> | YES}]** **[,TXT={'***msgtxt***' |** *addr***}]** **[,TXT2={'***altmsgtxt***' |** *addr***}]** **[,VOL={<u>STATIC</u> | REMOVE | INSERT | SWAP}]** **[,WAIT={NO | <u>YES</u>}]**

GEN
specifies the full range of display options.

UCB=*(reg)*—(2-12)
specifies a register containing the UCB address for the device.

FLASH={STEADY | STEADY2 | <u>BLINK</u> | BLINK2 | ALT}
specifies message display mode.

Note: If you specify VOL=SWAP, messages will always be displayed as if you had specified FLASH=ALT

STEADY
specifies that the primary message (TXT) is to be displayed without flashing.

STEADY2
specifies that the alternate message (TXT2) is to be displayed without flashing.

<u>BLINK</u>
specifies that the primary message (TXT) flash on and off at a rate of approximately two seconds on and one-half second off.

BLINK2
specifies that the alternate message (TXT2) flash on and off at a rate of approximately two seconds on and one-half second off.

ALT
specifies that the primary and alternate messages (TXT and TXT2) flash on and off alternately, at a rate of approximately two seconds on and one-half second off.

FORCE={<u>NO</u> | YES | *n* **|** *keyword* **| (***reg***)}**
specifies the priority (IOSLEVEL) for the display request's I/O. The higher the IOSLEVEL value, the greater the priority.

If you do not specify the FORCE parameter, the default is FORCE=NO.

<u>NO</u>

prevents execution of a display request for a device whose I/O is being quiesced. The IOSLEVEL is set to the installation default, as indicated in the CVTIONLV field of the CVT.

YES

forces execution of a display request for a device even if its I/O is being quiesced. The IOSLEVEL is set to 9, the highest priority.

n

specifies a decimal number from 1 to 9, to be used as the IOSLEVEL value. A high number indicates a higher priority request for the device.

keyword

specifies a label equated to an IOSLEVEL value:

NORMAL	1
QUIESCE	2
DAVV	3
DDR	4
DYNPATH	5
UNCRSV	6
CHPRCVY	7
SCHRCVY	8
FDEV	9

(*reg*)

specifies that the low-order byte of the indicated register (2 through 12) contains a value between 1 and 9, indicating the IOSLEVEL.

MF={L | (E,*addr*)}

specifies either the execute or the list form of MSGDISP. If you do not specify this parameter, the standard form of the macro is used.

L

specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)

specifies that the execute form of the macro and an existing parameter list is to be used.

addr

specifies an in-storage address of the parameter list. Specify either an RX-type address or a register in the range of 2 through 12.

TEST={<u>NO</u> | YES}

specifies whether to test the UCB to determine if the device is capable of displaying messages.

<u>NO</u>
specifies that the SVC routine will test the UCB.

YES
specifies testing the UCB before the SVC call.

Note: TEST=YES requires you to include the UCB mapping macro (IEFUCBOB) in the source code.

TXT={'*msgtxt*' | *addr*}
specifies 8 characters to be shown in positions 1 through 8 of the display. If you do not specify TXT, blanks are displayed.

'*msgtxt*'
specifies the 8 characters as literals. Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)
specifies an in-storage address of an area containing the 8 characters. For MF=L, you may only specify an A-type address.

TXT2={'*altmsgtxt*' | *addr*}
specifies 8 alternate characters to display in positions 1 through 8 of the display. If you do not specify TXT2, blanks are displayed.

'*altmsgtxt*'
specifies the 8 characters as literals. Specify in apostrophes.

addr—RX-type address, A-type address, or (2-12)
specifies an in-storage address of an area containing the 8 characters. For MF=L, you may only specify an A-type address.

VOL={<u>STATIC</u> | REMOVE | INSERT | SWAP}
specifies message display mode, based on volume status.

<u>STATIC</u>
specifies that messages will display without regard to volume status until the next message request is executed, or until the next command initiates volume movement.

REMOVE
specifies that messages will display until the current volume is demounted. This parameter is ignored if a volume is not mounted when the request is executed.

INSERT
specifies that messages will display until a volume is present, the tape threaded, and the active/inactive switch is in the active position. This parameter is ignored if a volume is loaded and ready when the request is executed.

SWAP
specifies that messages will always display as if FLASH=ALT were specified. The data from TXT and TXT2 displays alternately (flashing) until the current volume has been demounted. Then only TXT2 will display (flashing) until a new volume is loaded and ready.

If no volume is mounted when this parameter is specified, only TXT2 data will display (flashing) until a new volume is loaded and ready.

WAIT={NO | YES}
specifies when control is to be returned to you.

NO
specifies that control is to be returned before I/O is complete. I/O return codes are not returned, and I/O errors are recorded in the same manner as any permanent error by the error recovery procedure.

YES
specifies that control is to be returned after I/O is complete.

Return Codes from MSGDISP

When the system returns control to the problem program, the low-order byte of register 15 contains a return code. The low-order byte of register 0 may contain a reason code as follows:

Return Code (R15)	Reason Code (R0)	Meaning
00(X'00')		Successful completion.
04(X'04')		Device does not support MSGDISP.
08(X'08')		Unauthorized request (failed TESTAUTH for proper authority level) or invalid input parameters (including DCB or UCB).
08(X'08')	01(X'01')	Invalid parameter.
	02(X'02')	Invalid DCB or DEBCHK error.
	03(X'03')	Environmental error.
	04(X'04')	Authorization violation.
	05(X'05')	Invalid UCB.
	06(X'06')	Invalid request.
	11(X'0B')	Unsuccessful ESTAE macro call.
	12(X'0C')	Unsuccessful GETMAIN request.

Return Code (R15)	Reason Code (R0)	Meaning
12(X'0C')		I/O error (I/O Supervisor posted the request for an error). *Note:* An I/O error occurs for load display if the drive display has a hardware failure.

If you get return code X'04' or X'0C' on a RESET UCBL operation, register 1 points to the UCB associated with the error when you regain control.

Chapter 7. Maintaining SYS1.IMAGELIB

This chapter describes how to maintain the system image library (SYS1.IMAGELIB) and UCS images for the IBM 1403, 3203, and 3211 Printers, and FCB images for the IBM 3203, 3211, and 4245 Printers. Sample JCL jobstreams for adding a UCS image to SYS1.IMAGELIB for the IBM 1403, 3203, and 3211 Printers are shown in Figure 37 on page 206, Figure 38 on page 207, and Figure 39 on page 208, respectively.

SYS1.IMAGELIB does not contain UCS images for the IBM 3262 Model 5, 4245, or 4248 Printers, but instead contains image tables. By means of these image tables, the system relates the user-requested UCS image to the corresponding print band. Figure 40 on page 210 defines and describes the structure of an image table entry. The contents of IBM-supplied image tables for the IBM 4245 and 4248 Printers are shown in Figure 41 on page 211 and Figure 42 on page 212, respectively. The IBM 3262 Model 5 Printer uses the same image table as the 4248.

This chapter also describes How to maintain the UCS image table in SYS1.IMAGELIB for the IBM 3262 Model 5, 4245, and 4248 Printers. To determine which print bands are available, see:

- *IBM 3262 Printer Model 5 Product Description*, containing information on band IDs for the 3262 Model 5 Printer
- *IBM 4245 Printer Model 1 Component Description and Operator's Guide*, containing information on band IDs for the 4245 Printer
- *IBM 4248 Printer Description*, containing information on band IDs for the 4248 Printer

SYS1.IMAGELIB also contains control modules for the IBM 3800 Printing Subsystem. You can use the IEBIMAGE utility program to create and maintain these control modules (character arrangement table modules, graphic character modification modules, copy modification modules, library character set modules, and FCB modules).

You can also use IEBIMAGE to create and maintain FCB modules in SYS1.IMAGELIB for the 4248 Printer. You can use FCB modules created for the 4248 with the 3262 Model 5 Printer. However, the 3262 Model 5 does not support variable printer speeds or the horizontal copy feature of the 4248. For more information about IEBIMAGE, see *Utilities*.

This chapter also describes how to retrieve an FCB image from SYS1.IMAGELIB for modification.

To use the information presented in this chapter, you should be familiar with the subjects of the following publications:

- *Data Administration: Macro Instruction Reference* describes the SETPRT macro that you can use to specify the images or modules that you want.
- *JCL Reference* describes the CHARS, MODIFY, UCB, and FCB parameters of the DD statement that are processed at OPEN time.
- *IBM 2821 Control Unit Component Description* contains information on creating a user-designed chain/train for the 1403 Printer.
- *IBM 3203 Printer Component Description and Operator's Guide* contains information on creating a user-designed train for the 3203 Printer.
- *IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide* contains information on creating a user-designed train for the 3211 Printer.
- *JES2 Initialization and Tuning* and *Network Job Entry Facility for JES2* contain reference information for JES2.
- *JES3 Initialization and Tuning*

You can use the SPZAP service aid to display and modify an existing member of SYS1.IMAGELIB. *Service Aids* describes the use of SPZAP.

UCS Images in SYS1.IMAGELIB

Most IBM standard character set images are included in SYS1.IMAGELIB at system generation time, through the DATAMGT macro and an IODEVICE macro for the specified printer. (For details on the DATAMGT and IODEVICE macros, see *System Generation.*) The standard character set images for the 1403, 3203, and 3211 Printers are shown below.

Printer	**Images**
1403 or 3203	AN, HN, PCAN, PCHN, PN, QNC, QN, RN, SN, TN, XN, YN
3211	A11, G11, H11, P11, T11

Adding a UCS Image to the Image Library

Using the assembler and linkage editor, you may add a UCS image to those that reside in SYS1.IMAGELIB. No executable code is generated; the assembler prepares DCs, and the linkage editor puts them into SYS1.IMAGELIB. The new UCS image must be structured according to the following rules:

1. The member name must be 5 to 8 characters long; the first 4 characters must be the appropriate UCS prefix, as shown below.

UCS1 - 1403 Printer

UCS2 - 3211 Printer (or 3211-compatible printer)

UCS3 - 3203 Printer

These first four characters must be followed by a character set code, one to four characters long. Any valid combination of letters and numbers under assembler language rules is acceptable. However, the single letters U or C must not be used, because they are symbols for special conditions recognized by the system. The assigned character set code must be specified on the DD statement or SETPRT macro to load the image into the UCS buffer.

You can supply an alias name for a new image with the ALIAS statement of the linkage editor. (For more information on the ALIAS statement, see *Linkage Editor and Loader User's Guide.*)

2. The first byte of the character set image load module specifies whether the image is a default. (Default images may be used by the system for jobs that do not request a specific image.) Specify the following in the first byte:

   For JES2:

   X'80' indicates a default image.

   X'40' indicates that the output is to be folded.

   X'C0' indicates default image and folding.

   X'00' indicates that the image is not to be used as a default.

   For non-JES2:

   X'80' indicates a default image.

   X'00' indicates that the image is not to be used as a default.

3. The second byte of the load module indicates the number of lines (n) to be printed for image verification. See "Verifying the UCS Image" on page 215 for more information on image verification.

4. Each byte of the next n bytes indicates the number of characters to be printed on each verification line. For the 3211 Printer, the maximum number of characters printed per line is 48; the bytes of associative bits (see note 5) are not printed during verification.

5. The UCS image itself must follow the previously described fields. The image must fill the number of bytes required by the printer; see the table below for image lengths. Note that, because of Assembler language syntax, two apostrophes or two ampersands must be coded to represent a single apostrophe or a single ampersand, respectively, within a character set image.

Printer	Image Length
1403	240 bytes
3203	304 bytes (240 characters followed by 64 bytes of associative bits)
3211	512 bytes (432 characters followed by 15 bytes of X'00' 64 bytes of associative bits, and one reserved byte of X'00')

Associative bits must be coded to prevent data checks when adding a UCS image to SYS1.IMAGELIB. See the appropriate printer manual for more information on coding associative bits.

UCS Coding Examples

- Figure 37 contains an example of adding a 1403 UCS image, YN, to SYS1.IMAGELIB or the image library. Notes follow Figure 39 on page 208.
- Figure 38 on page 207 shows the code used to add a 3203 UCS image, YN, to SYS1.IMAGELIB or the image library.
- Figure 39 on page 208 shows the code used to add a 3211 UCS image, A11, to SYS1.IMAGELIB or the image library.

```
//ADDYN      JOB  MSGLEVEL=1
//STEP       EXEC PROC=ASMFCL,PARM.ASM='NODECK,LOAD',
//                PARM.LKED='LIST,OL,REFR,RENT,XREF'
//ASM.SYSIN DD *
UCS1YN   CSECT
         DC   X'80'         (THIS IS A DEFAULT IMAGE)
         DC   AL1(6)        (NUMBER OF LINES TO BE PRINTED)
         DC   AL1(39)       (39 CHARACTERS TO BE PRINTED ON LINE 1)
         DC   AL1(42)       (42 CHARACTERS TO BE PRINTED ON LINE 2)
         DC   AL1(39)       (39 CHARACTERS TO BE PRINTED ON LINE 3)
         DC   AL1(39)       (39 CHARACTERS TO BE PRINTED ON LINE 4)
         DC   AL1(42)       (42 CHARACTERS TO BE PRINTED ON LINE 5)
         DC   AL1(39)       (39 CHARACTERS TO BE PRINTED ON LINE 6)
*    THE FOLLOWING SIX LINES REPRESENT THE TRAIN IMAGE
         DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.#-$'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.#-$'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRSTUVWXYZ*,.'
         END
/*
//LKED.SYSLMOD DD DSNAME=SYS1.IMAGELIB(UCS1YN),DISP=OLD,
//              SPACE=        (OVERRIDE SECONDARY ALLOCATION)
```

Figure 37. Sample Code to Add a 1403 UCS Image to SYS1.IMAGELIB

Notes to Figure 37 on page 206, Figure 38 on page 207, and Figure 39:

1. The RENT and REFR linkage editor attributes are *required.*

2. For the 3203 and 3211 Printers, the 64 bytes of associative bits must be coded to avoid data checks. To determine how to code these bits for a particular image, see *IBM 3203 Printer Component Description and Operator's Guide* or *IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide.*

3. Executing the ASMFCL procedure does not actually generate executable code. The assembler/linkage editor merely places the UCS image into SYS1.IMAGELIB.

4. The SPACE parameter is overridden here because the ASMFCL cataloged procedure has secondary allocation specified. By eliminating the override you can use the original secondary allocation amount.

UCS Image Tables in SYS1.IMAGELIB

SYS1.IMAGELIB does not contain UCS images for the IBM 3262 Model 5, 4245, or 4248 Printers, but contains image tables. The UCS image for each band is stored, instead, in the printer, and is automatically loaded into the UCS buffer when the machine is powered on or a new band is installed. See Figure 40 on page 210 for the format of image table entries, and "Adding or Modifying a UCS Image Table Entry" on page 213 for information on how to add or modify an image table entry.

SYS1.IMAGELIB contains one UCS image table for each type of printer that supports image tables. An image table contains an entry for most installation-standard IBM-supplied bands. The 4245 image table is named UCS5. The shared 4248 and 3262 Model 5 image table is named UCS6.

Alias Names in Image Tables

The image tables also define alias names for most installation-standard print bands used on the IBM 4245 and 4248 Printers. The IBM-supplied image tables do not provide alias names for the IBM 3262 Model 5 Printer.

Some print chains/trains/bands, such as SN and KA22, do not have alias names because there is no equivalent chain/train/band on other printers. You can assign an alias for these chains/trains with the linkage editor ALIAS statement. (For more information on the ALIAS statement, see *Linkage Editor and Loader User's Guide*.) For the 3262 Model 5, 4245, or 4248 Printer, you can add an alias name by adding or modifying an entry in the UCS image table. See "Adding or Modifying a UCS Image Table Entry" on page 213. A typical UCS image table entry is shown in Figure 40 on page 210.

```
//ADDA11     JOB  MSGLEVEL=1
//STEP       EXEC PROC=ASMFCL,PARM.ASM='NODECK,LOAD',
//              PARM.LKED='LIST,OL,REFR,RENT,XREF'
//ASM.SYSIN DD *
UCS2A11  CSECT
         DC   X'80'          (THIS IS A DEFAULT IMAGE)
         DC   AL1(9)         (NUMBER OF LINES TO BE PRINTED)
         DC   AL1(48)        (48 CHARACTERS TO BE PRINTED ON LINE 1)
         DC   AL1(48)        (48 CHARACTERS TO BE PRINTED ON LINE 2)
         DC   AL1(48)        (48 CHARACTERS TO BE PRINTED ON LINE 3)
         DC   AL1(48)        (48 CHARACTERS TO BE PRINTED ON LINE 4)
         DC   AL1(48)        (48 CHARACTERS TO BE PRINTED ON LINE 5)
         DC   AL1(48)        (48 CHARACTERS TO BE PRINTED ON LINE 6)
         DC   AL1(48)        (48 CHARACTERS TO BE PRINTED ON LINE 7)
         DC   AL1(48)        (48 CHARACTERS TO BE PRINTED ON LINE 8)
         DC   AL1(48)        (48 CHARACTERS TO BE PRINTED ON LINE 9)
*    THE FOLLOWING NINE LINES REPRESENT THE TRAIN IMAGE
*    NOTE 2 AMPERSANDS MUST BE CODED TO GET 1 IN ASSEMBLER SYNTAX
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   15X'00'        (RESERVED FIELD, BYTES 433-447)
*    THE FOLLOWING FOUR DC INSTRUCTIONS DEFINE THE ASSOCIATIVE BITS,
*    UCSB BYTE POSITIONS 448-511
         DC   X'C0101010101010101010000404042400004010'
         DC   X'10101010101010100004040410000404010 10'
         DC   X'101010101010000404000000010101010101 0'
         DC   X'101010100004040444800'
         DC   X'00'          (RESERVED FIELD, BYTE 512)
         END
/*
//LKED.SYSLMOD DD DSNAME=SYS1.IMAGELIB(UCS2A11),DISP=OLD,
//              SPACE=         (OVERRIDE SECONDARY ALLOCATION)
```

Figure 39. Sample Code to Add a 3211 UCS Image to SYS1.IMAGELIB

yourself. "Adding or Modifying a UCS Image Table Entry" on page 213 describes how to add entries to the UCS image table. For a list of the bands available for the 3262 Model 5, see *IBM 3262 Printer Model 5 Product Description.*

Adding or Modifying a UCS Image Table Entry

If you plan to use a new UCS image name/alias with the 3262 Model 5, 4245, or 4248 Printer, you must add an entry for that image name/alias to the appropriate UCS image table. As shown below, use the assembler to compile the image table module, then link-edit the object file into SYS1.IMAGELIB. Similarly, if you want to select a new default image or change the description on an old image, you must change the image table.

To build new UCS table entries, or to change the format of old entries, use the following procedure. Also, see "Example 1: Adding a New Band ID to the 4245 UCS Image Table (UCS5)" on page 216 and "Example 2: Adding a New Default Entry to the 4248 UCS Image Table (UCS6)." on page 216.

1. Issue the IGGUCSIT macro, as described below, to build a new UCS image table entry. If you are updating the image table as shown in the following two examples, the linkage editor builds a new entry at the start of the table, even if you intended to replace an existing entry. When the system subsequently uses the table, it encounters the new entry first, thus the old one is effectively replaced.

2. Include the UCS image table source, using the IGGUCS5 or IGGUCS6 macro, both of which are found in SYS1.MACLIB.

3. Assemble the image table module (UCS5 or UCS6).

4. Link-edit the assembled module into SYS1.IMAGELIB.

Note: RENT and REFR are required attributes.

The IGGUCSIT macro instruction has the following format:

IGGUCSIT	**MF={<u>LIST</u> | DSECT}**
	,NAME=*image name*
	[,ALIAS=*image alias***]**
	[,DEFAULT={YES | <u>NO</u>}]
	[,DESCR=*description***]**
	[,DEVICE={<u>4245</u> | 4248}]
	[,VLENGTH=(*n1,n2,...n***)]**
	[,FOLD={YES | <u>NO</u>}]

MF={<u>LIST</u> | DSECT}
specifies the form of the macro instruction.

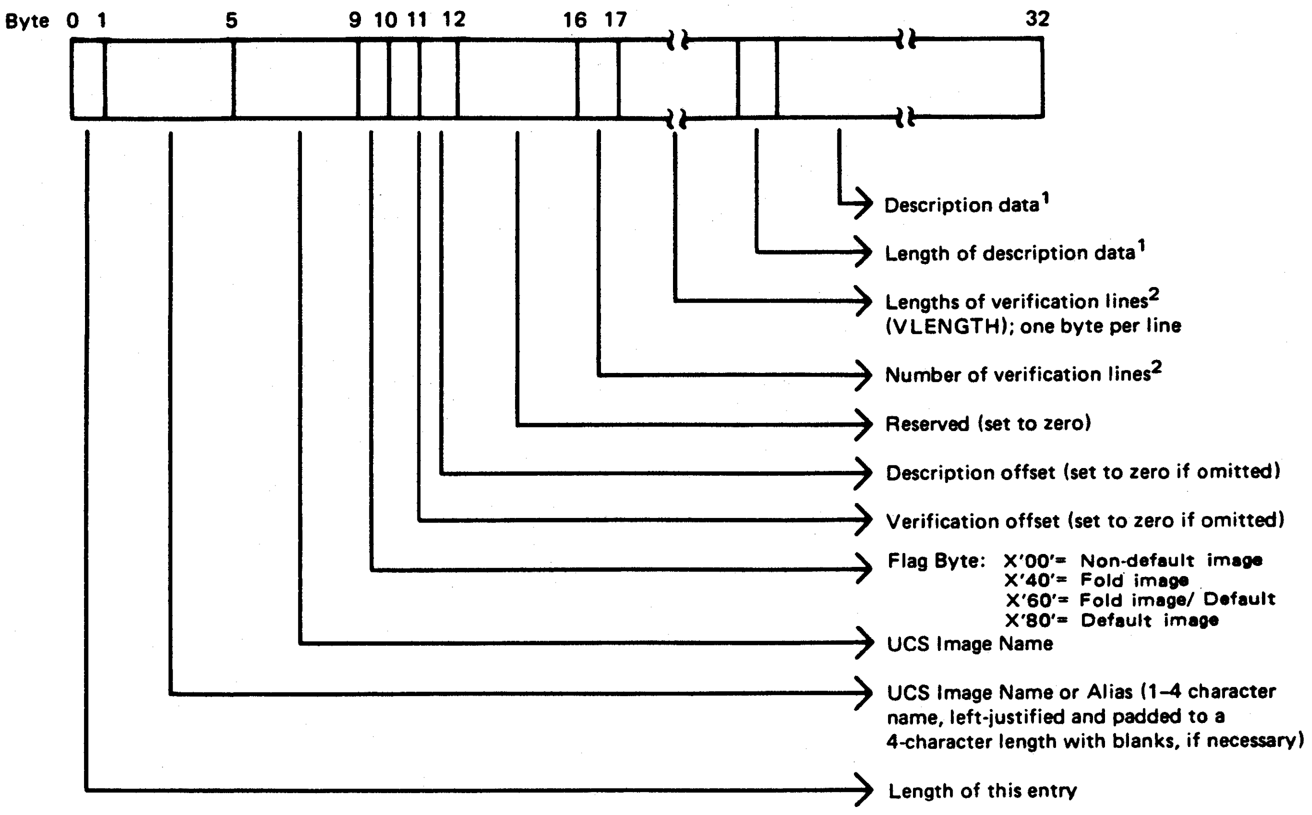

Byte 0 1 5 9 10 11 12 16 17 32
Description data[1]
Length of description data[1]
Lengths of verification lines[2] (VLENGTH); one byte per line
Number of verification lines[2]
Reserved (set to zero)
Description offset (set to zero if omitted)
Verification offset (set to zero if omitted)
Flag Byte: X'00'= Non-default image
X'40'= Fold image
X'60'= Fold image/ Default
X'80'= Default image
UCS Image Name
UCS Image Name or Alias (1–4 character name, left-justified and padded to a 4-character length with blanks, if necessary)
Length of this entry

Figure 40. UCS Image Table Entry Format

Notes to Figure 40:

1. This field is optional.

2. This field is optional for the 4245 Printer. For the 3262 Model 5 and the 4248, this field does not apply and is set to X'00'.

For the 3262 Model 5 Printer, DEVICE=4248 should be specified in order to create the appropriate form of the image table entry.

VLENGTH=(*n1*,*n2*,... *n*)
specifies the length(s) of each line in the UCS verification display. The length of each line must be specified separately, even if all lines are of the same length.

n1 is the length of print line 1; *n2* is the length of print line 2; *n* is the length of the last print line. To display the complete image, the sum of the verification line lengths should equal 350.

For details on the verification report, see "Verifying the UCS Image."

The VLENGTH parameter is not valid for the 3262 Model 5 or 4248 Printer.

FOLD={YES | <u>NO</u>}
indicates whether the UCS image is to be folded.

YES
indicates that the UCS image is to be folded. Allows printing only uppercase characters from either upper- or lowercase data codes. Folding continues until an UNFOLD command is received.

<u>NO</u>
indicates that the UCS image is not to be folded. This is the default.

Verifying the UCS Image

For the 1403 (with the UCS feature), 3203, 3211, 3262 Model 5, 4245, and 4248 Printers, you can print the UCS image for visual verification using either of the following parameters:

- In JCL: **UCS=(*character set code*,,VERIFY)**
- In the SETPRT macro: **UCS=(*character set code*,,V)**

You can also use these parameters for the 3262 Model 5 and 4248 Printers. However, because the UCS image cannot be read directly from the 3262 Model 5, or 4248, only the header information is printed. The verification display header appears in the format shown below.

```
UCS IMAGE VERIFICATION image id [,FOLD] [description]
```

image id
The 1- to 4-character name of the UCS image.

description
The descriptive information supplied for this UCS image in the UCS image table.

The contents of the UCS image table UCS6 (IGGUCS6 macro), for the 4248 Printer, are shown in Figure 42.

Name	**Alias**	**Default**	**Description**
40E1	40E1	YES	Default UCS image
40E1	AN21	NO	4245 AN21 image
40E1	AN	NO	1403/3203 AN image
40E1	A11	NO	3211 A11 image
4101	4101	NO	Nondefault UCS image
4101	HN21	NO	4245 HN21 image
4101	HN	NO	1403/3203 HN image
4101	H11	NO	3211 H11 image
41C1	41C1	NO	Nondefault UCS image
41C1	GN21	NO	4245 GN21 image
41C1	G11	NO	3211 G11 image
4121	4121	NO	Nondefault UCS image
4121	PL21	NO	4245 PL21 image
4121	PN	NO	1403/3203 PN image
4121	P11	NO	3211 P11 image
4181	4181	NO	Nondefault UCS image
4181	TN21	NO	4245 TN21 image
4181	TN	NO	1403/3203 TN image
4181	T11	NO	3211 T11 image
4061	4061	NO	Nondefault UCS image
40C1	40C1	NO	Nondefault UCS image
4161	4161	NO	Nondefault UCS image
4161	FC21	NO	4245 FC21 image
4201	4201	NO	Nondefault UCS image
4201	SN21	NO	4245 SN21 image
4041	4041	NO	Nondefault UCS image
4041	KA21	NO	4245 KA21 image

Figure 42. UCS6 Image Table Contents

Note: The image tables for the 4245 and 4248 Printers include USA and Canada band IDs only. To support other national band IDs, you must modify the UCS image table. See "Adding or Modifying a UCS Image Table Entry" on page 213.

The 3262 Model 5 Printer uses the 4248 UCS image table, UCS6. However, no 3262 Model 5 band names or aliases are provided by IBM in UCS6. In order to use 3262 Model 5 UCS images, you must add the names and aliases to UCS6

```
                                                                    72
//UCS6     JOB  . . .
//         EXEC ASMFCL,
//               PARM.ASM='NODECK,LOAD',
//               PARM.LKED='OL,RENT,REUS'
//SYSPRINT  DD SYSOUT=A
//ASM.SYSIN DD *
         TITLE 'UPDATED UCS6 IMAGE TABLE'
UCS6     CSECT
         IGGUCSIT NAME=40E1,                                        X
               DEVICE=4248,                                         X
               ALIAS=HN21,                                          X
               DEFAULT=YES,                                         X
               DESCR='40E1 DEFAULT BAND'
         IGGUCS6
         END
/*
//LKED.SYSLMOD DD DSN=SYS1.IMAGELIB(UCS6),DISP=OLD,
//               SPACE=  (OVERRIDE SECONDARY ALLOCATION)
```

Notes to Example 2:

1. This method creates a duplicate entry for 40E1 that becomes the first entry in the table. Because the table is searched sequentially, the new entry is always found before the old entry, thus effectively replacing the old entry.

2. The RENT and REUS linkage editor attributes are *required.*

3. Executing the ASMFCL procedure does not generate executable code. The assembler/linkage editor places the updated UCS image table into SYS1.IMAGELIB.

4. The SPACE parameter is overridden because the ASMFCL cataloged procedure has secondary allocation specified. Elimination of the override causes the original secondary allocation amount to be used.

CB Images in SYS1.IMAGELIB

Two standard FCB images, STD1 and STD2, are included in SYS1.IMAGELIB during system generation for the following printers:

3203

3211

3262 Model 5

4245

4248

The 4248 and 3262 Model 5 Printers also accept FCBs that can be used with the 3203, 3211, and 4245 Printers. (These are referred to as 3211 format FCBs.)

LIST

produces a UCS image table entry based on the information supplied in other IGGUCSIT parameters. If LIST is selected or allowed to default, the NAME parameter must also be coded.

DSECT

produces a DSECT for a single UCS image table entry, similar to the sample entry shown in Figure 40 on page 210. If you code DSECT, all other parameters of IGGUCSIT are ignored.

LIST is the default.

NAME=*image name*

specifies the 1 to 4 character UCS image name.

ALIAS=*image alias*

specifies a 1 to 4 character alias name for the UCS image. If ALIAS is not specified, the image name coded in the NAME parameter will be entered in the UCS image table.

DEFAULT={YES | NO}

indicates whether the new UCS image is to be used as a default value.

YES

indicates that this UCS image is a default. Default images are used by the system for jobs that do not request a specific image.

NO

indicates that this UCS image should not be used as a default.

If the DEFAULT parameter is not specified, the new UCS image is not used as a default.

DESCR=*description*

specifies descriptive information about the new UCS image. *description* can be up to 32 EBCDIC or hexadecimal characters long. You cannot use EBCDIC and hexadecimal characters in combination.

Descriptive information is placed in the header line of the verification display, following the real UCS image name. If you omit the DESCR parameter, no description appears in the display. For more information on the verification display, see "Verifying the UCS Image" on page 215.

If VLENGTH is not specified for the 4245 Printer, the DESCR parameter is ignored.

DEVICE={4245 | 4248}

specifies the type of device for which an image table entry is to be created.

If you specify MF=LIST on the first invocation of the IGGUCSIT macro, DEVICE defaults to 4245. The default for subsequent invocations is the printer type that you specified (or the default) on the first invocation. Table entries with different DEVICE specifications are not allowed.

```
FCB2STD2  CSECT
          DC     X'80'                     DEFAULT
          DC     AL1(66)              FCB IMAGE LENGTH = 66
          DC     X'000000'            LINE 1, 2, 3
          DC     X'01'                LINE 4, CHANNEL 1
          DC     X'0000000000'        LINE 5, 6, 7, 8, 9
          DC     X'02'                LINE 10, CHANNEL 2
          DC     X'0000000000'        LINE 11, 12, 13, 14, 15
          DC     X'03'                LINE 16, CHANNEL 3
          DC     X'0000000000'        LINE 17, 18, 19, 20, 21
          DC     X'04'                LINE 22, CHANNEL 4
          DC     X'0000000000'        LINE 23, 24, 25, 26, 27
          DC     X'05'                LINE 28, CHANNEL 5
          DC     X'0000000000'        LINE 29, 30, 31, 32, 33
          DC     X'06'                LINE 34, CHANNEL 6
          DC     X'0000000000'        LINE 35, 36, 37, 38, 39
          DC     X'07'                LINE 40, CHANNEL 7
          DC     X'0000000000'        LINE 41, 42, 43, 44, 45
          DC     X'08'                LINE 46, CHANNEL 8
          DC     X'0000000000'        LINE 47, 48, 49, 50, 51
          DC     X'0A'                LINE 52, CHANNEL 10
          DC     X'0000000000'        LINE 53, 54, 55, 56, 57
          DC     X'0B'                LINE 58, CHANNEL 11
          DC     X'0000000000'        LINE 59, 60, 61, 62, 63
          DC     X'0C'                LINE 64, CHANNEL 12
          DC     X'00'                LINE 65
          DC     X'19'                LINE 66, CHANNEL 9-END OF FCB IMAGE
          END
```

Figure 44. Sample of the Standard FCB Image STD2

Adding an FCB Image to the Image Library

You may add a 3211-format FCB image to those that reside in SYS1.IMAGELIB, using the assembler and linkage editor. No executable code is generated; the assembler prepares DCs, and the linkage editor links them into SYS1.IMAGELIB. The new FCB image must be structured according to the following rules:

1. The member name may not exceed eight bytes. The first four characters of the name must be FCB2. The characters that follow identify the FCB image and are referred to as the "image identifier" (ID). Any combination of valid assembler language characters can be used, with the exception of a single "C" or "U," because these are used by the system to recognize special conditions. The image identifier must be specified in the FCB keyword of a DD statement or in the SETPRT macro to load the image into the FCB buffer.

2. The first byte of the FCB load module specifies whether the image is a default. (Default images may be used by the system for jobs that do not request a specific image.) Specify the following in the first byte:

   X'80' indicates a default image
   X'00' indicates a nondefault image

For more information about the UCS VERIFY parameters, see *JCL Reference* and *Data Administration: Macro Instruction Reference*.

Examples of Adding to the UCS Image Table

Example 1: Adding a New Band ID to the 4245 UCS Image Table (UCS5)

In this example, the band name RPQ1 with description "RPQ BAND" is added to UCS5. In the UCS verification display, 7 lines of 50 characters each are printed. Macro IGGUCS5 causes the UCS image table source (as distributed by IBM) to be included in the table entry.

```
                                                                      72
//UCS5    JOB  . . .
//        EXEC ASMFCL,
//              PARM.ASM='NODECK,LOAD',
//              PARM.LKED='OL,RENT,REUS'
//SYSPRINT  DD SYSOUT=A
//ASM.SYSIN DD *
          TITLE 'UPDATED UCS5 IMAGE TABLE'
UCS5      CSECT
          IGGUCSIT NAME=RPQ1,                                          X
                VLENGTH=(50,50,50,50,50,50,50),                        X
                DESCR='RPQ BAND'
          IGGUCS5
          END
/*
//LKED.SYSLMOD DD DSN=SYS1.IMAGELIB(UCS5),DISP=OLD,
//              SPACE=  (OVERRIDE SECONDARY ALLOCATION)
```

Notes to Example 1:

1. The RENT and REUS linkage editor attributes are *required.*
2. Executing the ASMFCL procedure does not actually generate executable code. The assembler/linkage editor places the updated UCS image table into SYS1.IMAGELIB.
3. The SPACE parameter is overridden here because the ASMFCL cataloged procedure has secondary allocation specified. Elimination of the override causes the original secondary allocation amount to be used.

Example 2: Adding a New Default Entry to the 4248 UCS Image Table (UCS6).

In the following example, the band name 40E1 with description "40E1 DEFAULT BAND" is added to UCS6 and defined as a default band. An alias name, HN21, is also defined for band 40E1. Macro IGGUCS6 causes the UCS image table source (as distributed by IBM) to be included in the table entry.

```
//ADDFCB          JOB  MSGLEVEL=1
//STEP            EXEC PROC=ASMFCL,PARM.ASM='NODECK,LOAD',
//                     PARM.LKED='LIST,OL,REFR,RENT,XREF'
//ASM.SYSIN       DD   *
FCB2ID1           CSECT
*THIS EXAMPLE IS FOR A FORM LENGTH OF 11 INCHES WITH 8 LPI (88 LINES)
                  DC   X'80'     THIS IS A DEFAULT IMAGE
                  DC   AL1(89)   LENGTH OF FCB IMAGE AND INDEXING BYTE
                  DC   X'8F'     OFFSET 15 CHARACTERS TO THE RIGHT
                  DC   X'10'     8 LINES PER INCH-NO CHANNEL FOR LINE 1
                  DC   XL4'0'    4 LINES NO CHANNEL
                  DC   X'01'     CHANNEL 1 IN LINE 6
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'02'     CHANNEL 2 IN LINE 13
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'03'     CHANNEL 3 IN LINE 20
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'04'     CHANNEL 4 IN LINE 27
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'05'     CHANNEL 5 IN LINE 34
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'06'     CHANNEL 6 IN LINE 41
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'07'     CHANNEL 7 IN LINE 48
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'08'     CHANNEL 8 IN LINE 55
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'09'     CHANNEL 9 IN LINE 62
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'0A'     CHANNEL 10 IN LINE 69
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'0B'     CHANNEL 11 IN LINE 76
                  DC   XL6'0'    6 LINES NO CHANNEL
                  DC   X'0C'     CHANNEL 12 IN LINE 83
                  DC   XL4'0'    4 LINES NO CHANNEL
                  DC   X'10'     POSITION 88 LAST LINE IN IMAGE
                  END
/*
//LKED.SYSLMOD    DD   DSNAME=SYS1.IMAGELIB(FCB2ID1),DISP=OLD,
//                     SPACE=    (OVERRIDE SECONDARY ALLOCATION)
```

Figure 45. Sample Code to Assemble and Add an FCB Load Module to SYS1.IMAGELIB

Notes to Figure 45:

1. The RENT and REFR linkage editor attributes are *required.*
2. Executing the ASMFCL procedure does not actually generate executable code. The assembler/linkage editor is used to place the FCB image into SYS1.IMAGELIB.
3. The SPACE parameter is overridden here because the ASMFCL cataloged procedure has secondary allocation specified. Elimination of the override causes the original secondary allocation amount to be used.

STD1 sets line spacing at 6 lines per inch for an 8-1/2 inch form; STD2 is a default FCB image that sets line spacing at 6 lines per inch for an 11-inch form. Channels for both images are evenly spaced, with Channel 1 on the fourth line and Channel 9 on the last line. See Figure 43 on page 218 and Figure 44 on page 219 for sample STD1 and STD2 images.

The 3262 Model 5, the 4245, and the 4248 Printer each load a default FCB image into the buffer when they are powered on. The 3262 Model 5 default FCB image is an 11-inch form with 6 lines per inch, a Channel 1 on the third print line, and a Channel 12 on line 64. The 4245 default FCB image is an 11-inch form with 6 lines per inch and a Channel 1 on the first print line. The 4248 default FCB image is the last FCB image loaded.

The standard FCB images STD3 is included in SYS1.IMAGELIB during system generation for the following printer:

3800

You should use the IEBIMAGE utility to create and modify FCB modules for the 3800 Printing Subsystem. You should also use it to create and modify FCB images for the 3262 Model 5 or 4248 Printer (4248 format FCBs).

Note: FCB module CSECT names for the 3262 Model 5 and 4248 Printers must begin with the letters "FCB4" For information on IEBIMAGE and the format of the 4248 FCB image, see *Utilities.*

```
FCB2STD1  CSECT
          DC    X'80'                 DEFAULT
          DC    AL1(48)               FCB IMAGE LENGTH = 48
          DC    X'000000'             LINE 1, 2, 3
          DC    X'01'                 LINE 4, CHANNEL 1
          DC    X'000000'             LINE 5, 6, 7
          DC    X'02'                 LINE 8, CHANNEL 2
          DC    X'000000'             LINE 9, 10, 11
          DC    X'03'                 LINE 12, CHANNEL 3
          DC    X'000000'             LINE 13, 14, 15
          DC    X'04'                 LINE 16, CHANNEL 4
          DC    X'000000'             LINE 17, 18, 19
          DC    X'05'                 LINE 20, CHANNEL 5
          DC    X'000000'             LINE 21, 22, 23
          DC    X'06'                 LINE 24, CHANNEL 6
          DC    X'000000'             LINE 25, 26, 27
          DC    X'07'                 LINE 28, CHANNEL 7
          DC    X'000000'             LINE 29, 30, 31
          DC    X'08'                 LINE 32, CHANNEL 8
          DC    X'000000'             LINE 33, 34, 35
          DC    X'0A'                 LINE 36, CHANNEL 10
          DC    X'000000'             LINE 37, 38, 39
          DC    X'0B'                 LINE 40, CHANNEL 11
          DC    X'000000'             LINE 41, 42, 43
          DC    X'0C'                 LINE 44, CHANNEL 12
          DC    X'000000'             LINE 45, 46, 47
          DC    X'19'                 LINE 48, CHANNEL 9-END OF FCB IMAGE
          END
```

Figure 43. Sample of the Standard FCB Image STD1

Return Code	Meaning
8(X'08')	Either SYS1.IMAGELIB does not exist on the volume to which the catalog points, or SYS1.IMAGELIB is not cataloged.
12(X'0C')	An error occurred in reading the catalog or VTOC.

BLDL and LOAD are the only macros that may refer to the DCB built by the IMGLIB macro.

3. The second byte of the load module indicates the number of bytes to be transferred to the control unit to load the FCB image. This count includes the byte, if used, for the print position indexing feature.

4. The third byte of the load module (the first byte of the FCB image) is either the print position indexing byte, or the lines-per-inch byte. The print position indexing byte is optional and, when used, precedes the lines-per-inch byte. The 3262 Model 5, 4245, and 4248 Printers accept and discard the index byte if it is present, because neither printer supports the indexing feature. A description of the print position indexing feature and its use will be found in *IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide.*

   The special index flag in the third byte contains X'80' plus a binary index value, from 1 to 32 (the default is 1). This index value sets the left margin: 1 indicates flush-left; any other value indicates a line indented that many spaces.

   The form image begins with the lines-per-inch (LPI) byte. The LPI byte defines the number of lines per inch (6 or 8) and also represents the first line of the page.

   Note: Printers controlled by JES2 require a channel 1 identifier here.

   Typically, the length of an FCB image is consistent with the length of the form it represents. For example, an 8-1/2 inch form to be printed at 6 LPI has an FCB image that is 51 bytes long (8-1/2 inches times 6 LPI).

   The LPI byte appears as follows:

   X'1n' sets 8 LPI

   X'0n' sets 6 LPI

5. All remaining bytes (lines) must contain X'0n', except the last byte, which must be X'1n'. The letter n can be a hexadecimal value from 1 to C, representing a channel (one to 12), or it can be 0, which means no channel is indicated.

In Figure 45 on page 221, an FCB load module is assembled and added to SYS1.IMAGELIB. The image defines a print density of 8 lines per inch on an 11-inch form, with a right shift of 15 line character positions (1-1/2 inches).

Chapter 8. JES2 Support for the IBM 1403, 3203 Model 5, and 3211 Printers

UCS Alias Names

The system assigns an alias for each installation-standard print chain not actually defined on a given printer. This provides JES2 with flexibility in scheduling printers for SYSOUT data sets. For example, a request for the 1403 TN train would be assigned the T11 train if the data set were printed on a 3211. The assigned alias names that follow the naming conventions currently used in SYS1.IMAGELIB are:

Image	**Alias**
UCS1AN	UCS1A11
UCS1HN	UCS1H11
UCS1PN	UCS1P11
UCS1TN	UCS1T11
UCS2A11	UCS2AN
UCS2H11	UCS2HN
UCS2P11	UCS2PN,UCS2RN,UCS2QN
UCS2T11	UCS2TN

The image and alias names are included in SYS1.IMAGELIB at system generation.

Some trains, such as SN and G11, do not have aliases because neither has an equivalent train on the other printer. An installation can assign an alias, if it so chooses. (For details about the ALIAS statement, see *Linkage Editor and Loader User's Guide.*) If an alias is supplied, JES2 will use it. If an alias is not supplied, an installation-defined SYSOUT class or a printer routing code (specified via the DEST parameter) should be used to assign the data set to the correct printer. If a SYSOUT class or a printer routing code is not used and if JES2 is directed to print a data set on a printer for which the proper image is not supplied, JES2 notifies the operator. The operator can then print the data set with a valid train or redirect the data set to the proper printer via the '$E' command.

If an installation defines a new train, it can supply an alias name for that train, via the linkage editor ALIAS statement, when including the image in SYS1.IMAGELIB.

Retrieving an FCB Image from SYS1.IMAGELIB

If you want to modify an FCB image in virtual storage before loading it into a forms control buffer, you can use this sequence of macro instructions to read the FCB image into virtual storage.

1. An IMGLIB macro instruction, along with the OPEN parameter

2. A BLDL macro instruction to determine whether the FCB image you want is in the image library

3. A LOAD macro instruction to load the image into virtual storage

After the image has been read in, you should issue the IMGLIB macro instruction with the CLOSE parameter and the address of the DCB that was built by the first IMGLIB macro. A SETPRT macro instruction can be used to load the forms control buffer with the modified image. Printers other than the 3800 require the use of an FCB entry in an exit list, as described in *Data Administration Guide.*

The format of the BLDL and SETPRT macros is given in *Data Administration: Macro Instruction Reference*; the format of the LOAD macro is given in *Supervisor Services and Macro Instructions.*

The format of the IMGLIB macro is shown below:

[*symbol*]	**IMGLIB**	**{OPEN | CLOSE,*addr*}**

OPEN
specifies that a DCB is to be built for SYS1.IMAGELIB and that SYS1.IMAGELIB is to be opened. The address of the DCB is returned in register 1.

CLOSE
specifies that SYS1.IMAGELIB is to be closed.

addr
specifies the RX-type address of the word that points to the DCB. If coded in the form (*reg*), the register in parentheses then contains the address of the DCB, not the address of the fullword.

Return codes from the IMGLIB OPEN macro are shown below:

Return Code	Meaning
0(X'00')	Operation successful.
4(X'04')	Either the volume containing SYS1.IMAGELIB is not mounted or a required catalog volume is not mounted.

Chapter 9. CATALOG, SCRATCH, and RENAME Dummy Modules

The detailed information about installation-replaceable Catalog, Scratch, and Rename dummy modules that appeared in this chapter has been moved to *Data Facility Product: Customization.*

You can replace these modules to invoke special processing either before or after CATALOG (SVC26), SCRATCH (SVC 29), or RENAME (SVC 30) processing.

Chapter 8. JES2 Support for the IBM 1403, 3203 Model 5, and 3211 Printers

UCS Alias Names

The system assigns an alias for each installation-standard print chain not actually defined on a given printer. This provides JES2 with flexibility in scheduling printers for SYSOUT data sets. For example, a request for the 1403 TN train would be assigned the T11 train if the data set were printed on a 3211. The assigned alias names that follow the naming conventions currently used in SYS1.IMAGELIB are:

Image	Alias
UCS1AN	UCS1A11
UCS1HN	UCS1H11
UCS1PN	UCS1P11
UCS1TN	UCS1T11
UCS2A11	UCS2AN
UCS2H11	UCS2HN
UCS2P11	UCS2PN,UCS2RN,UCS2QN
UCS2T11	UCS2TN

The image and alias names are included in SYS1.IMAGELIB at system generation.

Some trains, such as SN and G11, do not have aliases because neither has an equivalent train on the other printer. An installation can assign an alias, if it so chooses. (For details about the ALIAS statement, see *Linkage Editor and Loader User's Guide*.) If an alias is supplied, JES2 will use it. If an alias is not supplied, an installation-defined SYSOUT class or a printer routing code (specified via the DEST parameter) should be used to assign the data set to the correct printer. If a SYSOUT class or a printer routing code is not used and if JES2 is directed to print a data set on a printer for which the proper image is not supplied, JES2 notifies the operator. The operator can then print the data set with a valid train or redirect the data set to the proper printer via the '$E' command.

If an installation defines a new train, it can supply an alias name for that train, via the linkage editor ALIAS statement, when including the image in SYS1.IMAGELIB.

The 3211 Indexing Feature

JES2 supports the 3211 Indexing Feature in two ways:

1. Specification of the INDEX parameter on the /*OUTPUT card.

2. The extended FCB image:

   JES2 supplies two special FCBs: FCB26 for 6 lines per inch and FCB28 for 8 lines per inch (specified as FCB=6 and FCB=8, respectively). These FCBs contain a channel 1 indication in position 1, a special index flag in the third byte, and the number of lines per inch in the fourth byte of the image.

   The special index flag in the third byte of FCB26 and FCB28 contains X'80' plus a binary index value, in the range 1 to 32 (default=1). The index value sets the left margin (1 indicates flush-left position; other values cause indentation of the print line by N-1 positions).

   If any other FCB images are to be used by JES2, they must specify channel 1 in position 1; otherwise, JES2 incorrectly positions the forms in the printer. (STD1 and STD2 do not specify channel 1 in position 1 and therefore must not be specified, unless altered, for JES2.)

   If the third byte of any other FCB image contains a data character (specifying the number of lines per inch) other than X'80', JES2 uses that specification and supplies an index value of 1.

IBM 3203 Model 5 Printer

The IBM 3203 Model 5 Printer is treated as a 3211 Printer by JES2, except that the 3203 Model 5 does not support the 3211 indexing feature, and any indexing commands from JES2 are ignored by the 3203 Model 5. The 3203 Model 5 uses 3211 FCB images and its own unique UCS images. UCS images are listed in *System Generation.*

Chapter 10. Specifying Buffer Numbers for DASD Data Sets

The BUFNO keyword in the DCB macro and the BUFNO subparameter of the DCB keyword in the DD statement determine how many buffers are allocated when accessing a partitioned or sequential data set using QSAM. The NCP keyword in the DCB macro determines how many un-CHECKed READ or WRITE macro instructions are allowed when accessing a sequential or partitioned data set using BSAM; one buffer is used for each READ or WRITE macro instruction.

The sequential access method can construct a channel program to transfer as many as 30 buffers or 240000 bytes of data, whichever is less. If BUFNO or NCP is less than 30, no more than that number of buffers can be transferred with a single channel program.

BUFNO is defaulted in OPEN to five if it is not specified for a QSAM DCB; NCP is defaulted to one in OPEN if it is not specified. The QSAM access method manages buffers. The user program must manage buffers when it uses BSAM.

Performance Considerations

Buffer number and block size influence the rate at which data can be transferred and the operating system overhead per block. The use of more buffers reduces (per block transferred) the EXCP and IOS overhead and the time waiting for the DASD device to seek to the requested cylinder and rotate to the requested record (device latency time). However, if more buffers are allocated than a program can effectively process, the virtual pages containing those buffers will be paged out, effectively adding to the system overhead for the job. A large number of buffers also cause a large amount of real storage to be allocated to the job while the data is being transferred.

A job in a low-performance group may get swapped out more frequently than a higher priority job. The number of buffers allocated for the job contributes to the number of pages that have to be swapped out.

Programs that access data sets with small block size (for example, 80) can easily make effective use of 30 buffers, which fit in, at most, two 4096-byte pages. The advantage of 30 buffers over the default of five buffers is great: one channel program versus six channel programs to transfer 30 blocks.

At the other end of the spectrum, usage of data sets with large blocking factors such as full-track blocking on 3350 or half-track blocking on 3380 can still be effective when only three or four buffers, rather than five or more, are specified.

DSN: Specify the Name of the DSCB

DSN=*addr*
DSN specifies the address of a 44-byte data set name of the DSCB to be accessed.

DSN is required if ACCESS=READ or WRITE and the request is to read or write a DSCB. If a 140-byte DSCB is specified:

- CVAF validity checks the storage location, but ignores the contents of the location.
- You must specify an argument that points to an extent within the VTOC.

BUFLIST: Specify One or More Buffer Lists

BUFLIST=*addr*
The BUFLIST keyword contains the address of a buffer list used to read or write a DSCB or VIRs.

VERIFY: Verify that a DSCB is a Format-0 DSCB

<u>VERIFY=YES</u>
CVAF will verify that the DSCB is a format-0 DSCB before writing the DSCB. The first four bytes of the key will be compared with binary zeros. If the key does not start with four bytes of zeros, the DSCB will not be written and an error code will be returned.

VERIFY=NO
CVAF will not test the key of the DSCB.

Note: VERIFY applies only when writing a 140-byte DSCB. VERIFY is ignored when a VIR is written.

UCB | DEB: Specify the VTOC to Be Accessed

UCB=*(reg)*
Supplies the address of the UCB for the unit whose VTOC is to be accessed. An unauthorized caller must not use this parameter.

Note: Code the address of the UCB parameter as register (2-12). Coding an RX-Type address here gives you unpredictable results.

If the address of a previously obtained I/O area is supplied through the IOAREA keyword, neither UCB nor DEB need be supplied. Otherwise, either a UCB or DEB must be supplied. If a UCB address is supplied, it will be overlaid in the CVPL by the UCB address present in the I/O area.

DEB=*addr*
Specifies the address of a DEB opened to the VTOC you want to access. CVAF does not allow output requests to the VTOC or VTOC index if you specify the DEB subparameter. If you are not authorized, you cannot

The slightly lower DASD performance and small increase in EXCP and IOS instruction costs should be more than offset by a reduction in paging or swapping in a constrained environment.

It can be seen that proper selection of buffer number can have a positive effect on the elapsed time of a job and the system overhead associated with the job. The DCB OPEN installation exit can use installation criteria for a default buffer number for QSAM DCBs (for a description of the OPEN installation exit, see *Data Facility Product: Customization.*) The NCP field of the DCB must be set by the program for BSAM DCBs.

ACCESS: Read or Write a DSCB or VIR(S), or Release Buffer Lists

When ACCESS is READ or WRITE, a single DSCB is accessed for an indexed or nonindexed VTOC, or one or more VIRs are accessed for an indexed VTOC.

ACCESS=READ

Specifies that a single DSCB or one or more VIR(s) are to be read into a buffer whose address is in a buffer list.

If the buffer list if for a DSCB, only one entry is used in the buffer list. The first entry with the skip bit set to zero and a nonzero buffer address is used.

All VIR(s) whose buffer list entry has the skip bit off will be read into a buffer.

DSN and BUFLIST are required if ACCESS=READ for a DSCB buffer list.

ACCESS=WRITE

Specifies that a single DSCB or one or more VIRs are to be written from buffer(s) whose address is in a buffer list.

WRITE is permitted with BRANCH=NO only if the caller is authorized by APF.

DSN and BUFLIST are required if ACCESS=WRITE for a DSCB buffer list.

If any buffer list entry has its modified bit set, only those entries with the modified bit set will be written. If no modify bits are on, all VIRs will be written.

ACCESS=RLSE

Applies only to VIR buffer lists. It requests the release of one or more buffers in the VIR buffer list chain identified in the BUFLIST keyword, and the release of each buffer list for which all buffers are released.

DSN and BUFLIST are not required if ACCESS=RLSE.

Only buffers in the buffer list with the skip bit set to zero and with a nonzero buffer address are released. The buffer list is not released if any entry has the skip bit set to one.

For an indexed VTOC, if ACCESS=RLSE is coded, buffer lists and buffers pointed to by the BUFLIST keyword will be released, along with buffer lists supplied in the CVAF parameter list CVMRCDS and CVIRCDS fields. If the CVMRCDS or the CVIRCDS buffers are supplied in the BUFLIST field, either directly or indirectly through chaining, the keyword MAPRCDS=YES, IXRCDS=KEEP, or MAPRCDS=(NO,0), IXRCDS=(NOKEEP,0) must be coded to prevent CVAF from freeing the buffers more than once. If buffers are released, the CVAF parameter list field pointing to the buffer list will be updated.

perform any asynchronous activity (such as EXCP, CLOSE, EOV) against the data set represented by the DEB because CVAF removes the DEB from the DEB table for the duration of the CVAF call. If you are not authorized (neither APF authorized nor in a system key) you must specify a DEB address, not a UCB, to CVAFDIR. See "Identifying the Volume" on page 42 for further details.

If you supply both the DEB and the UCB in the CVPL, the DEB address will be used and the UCB address will be overlaid in the CVPL by the UCB address in the DEB.

IOAREA: Keep or Free the I/O Work Area

IOAREA=KEEP
Specifies the CVAF I/O area associated with the CVAF parameter list is to be kept upon completion of the CVAF request. IOAREA=KEEP may be coded with BRANCH=NO only if the caller is authorized (APF or system key).

If IOAREA=KEEP is coded, the caller must issue CVAF with IOAREA=NOKEEP specified at some future time, whether or not any further VTOC access is required: for example, the recovery routine of the caller of CVAF.

Coding IOAREA=KEEP allows subsequent CVAF requests to be more efficient, as certain initialization functions can be bypassed. Neither DEB nor UCB need be specified when a previously obtained CVAF I/O area is supplied; neither can they be changed.

When IOAREA=KEEP is first issued, CVAF returns the CVAF I/O area in the CVAF parameter list (CVIOAR). Subsequent calls of CVAF may use that same parameter list, and CVAF will obtain its I/O area from the CVIOAR.

When processing on the current volume is finished, release all areas that were kept.

IOAREA=(KEEP,*addr*)
Provides the address of a previously obtained I/O area. If a different CVAF parameter list is used, the previously obtained I/O area may be passed to CVAF by coding its address as the second parameter of the IOAREA keyword.

<u>IOAREA=NOKEEP</u>
Causes the work area to be freed upon completion of the CVAF request.

IOAREA=(NOKEEP,*addr*)
Causes a previously obtained work area to be freed upon completion of the CVAF request.

MAPRCDS: Keep or Free MAPRCDS Buffer List and Buffers

This keyword applies to an indexed VTOC only and specifies the disposition of the MAPRCDS buffer list and buffers.

MAPRCDS=YES
Specifies that the buffer list and buffers are to be retained at the end of processing.

If no buffer list address is in the CVAF parameter list, CVAF will read the MAP VIRs into buffers it obtains. The buffer list that contains the address and RBAs of the VIRs can be accessed after processing from the CVAF parameter list field, CVMRCDS. The buffer list and VIR buffers are in your protect key: subpool 0 if you are not authorized; 229 if you are.

When processing on the current volume is finished, release all areas that were kept.

MAPRCDS=(YES,*addr*)
If YES is coded and the buffer list address (CVMRCDS in CVAF parameter list) is supplied, VIRs are not read.

The CVMRCDS buffer list used in CVAFDIR macro can be passed to another CVAF macro call through the MAPRCDS keyword.

If MAPRCDS=YES is coded for a nonindexed VTOC, the function is performed, but an error code will be returned.

MAPRCDS=NO
If MAPRCDS=NO is coded, all the buffers without the skip bit on in the buffer list whose address is in the CVMRCDS field of the CVPL will be freed. If all the buffers are freed, the buffer list will also be freed.

MAPRCDS=(NO,*addr*)
Causes buffer lists and buffers previously obtained by CVAF to be freed.

You must free buffer lists and buffers obtained by CVAF. This can be done in one of three ways:

- By coding MAPRCDS=NO on the CVAFDIR macro that obtained the buffers
- By coding MAPRCDS=NO on a subsequent CVAF macro
- By coding CVAFDIR ACCESS=RLSE and providing the address of the buffer list in the BUFLIST keyword

Note: You must enqueue the VTOC and reserve the unit to maintain the integrity of MAP records read.

IXRCDS: Retain VIERS in Virtual Storage

This keyword applies to indexed VTOCs only.

IXRCDS=KEEP
Specifies that VIERs read into storage are to be kept in virtual storage. The VIERs are retained even if processing cannot complete successfully. The CVAF parameter list in field CVIRCDS contains the address of a buffer list with the VIR buffer addresses and RBAs of the VIERs read.

The index search function will dynamically update the buffer list and, when necessary, obtain additional buffer lists and chain them together.

If KEEP is specified and no buffer list is supplied to CVAF in the CVPL, CVAF will obtain a buffer list and buffers and read the first high-level VIER. The address of the buffer list is placed in the CVMICDS field of the CVPL. The first high-level VIER will be checked for the VXFHLV bit and to see whether the VXVISE bit is off.

The buffer list and VIR buffers are in your protect key. The subpool is 0 if you are not authorized; it is subpool 229 if you are.

If IXRCDS=KEEP is coded for a nonindexed VTOC, a request to read or write a DSCB will be performed, but an error code will be returned.

When processing on the current volume is finished, release all areas that were kept.

IXRCDS=(KEEP,*addr*)
The index records buffer list address from one CVAF request is being passed to this CVAF parameter list by specifying its address as the second parameter in the IXRCDS keyword.

IXRCDS=NOKEEP
If NOKEEP is coded, the VIERs that are accessed (if any) are not retained. Furthermore, the buffer list supplied in the CVIRCDS field in the CVAF parameter list is released, as are all buffers found in the buffer list. If the skip bit is set in any entry in the buffer list, the buffer and buffer list will not be freed.

IXRCDS=(NOKEEP,*addr*)
Specifies that previously accessed VIERs are not to be retained.

You must free buffer lists and buffers obtained by CVAF. This can be done in one of three ways:

- By coding IXRCDS=NOKEEP on the CVAFDIR macro that obtained the buffers
- By coding IXRCDS=NOKEEP on a subsequent CVAF macro
- By coding CVAFDIR ACCESS=RLSE and providing the address of the buffer list in the BUFLIST keyword

Note: You must enqueue the VTOC and reserve the unit to maintain the integrity of the VIERs read.

BRANCH: Specify the Entry to the Macro

BRANCH=(YES,SUP)
Requests that the branch entry to CVAFDIR be used. You must be in supervisor state. Protect key checking is bypassed.

An 18-word save area must be supplied if BRANCH=YES is coded. No lock may be held on entry to CVAF. SRB mode is not allowed.

BRANCH=YES
Equivalent to BRANCH=(YES,SUP), because SUP is the default when YES is coded. Protect key checking is bypassed.

BRANCH=(YES,PGM)
Requests the branch entry. You must be authorized by APF and be in problem state. Protect key checking is bypassed.

BRANCH=NO
Requests the SVC entry. You must be authorized by APF if any output operations are requested. Protect key checking is performed.

MF: Specify the Form of the Macro

This keyword specifies whether the list, execute, or normal form of the macro is requested.

MF=I
If I is coded or if neither L nor E is coded, the CVAF parameter list is generated and CVAF is called. This is the normal form of the macro.

MF=L
L indicates the list form of the macro. A parameter list is generated, but CVAF is not called.

MF=(E,*addr*)
E indicates the execute form of the macro. The CVAF parameter list whose address is in X'addr' can be modified by this form of the macro.

Return Codes from the CVAFDIR Macro

On return from CVAF, register 1 contains the address of the CVPL (CVAF parameter list), and register 15 contains one of the following return codes:

Code	Meaning
00(X'00')	The request was successful. However, if the CVAFDIR request is to read or write a DSCB and a VTOC index structure error is encountered, the CVSTAT field indicates the structure error encountered. (CVSTAT code descriptions are in Appendix C, "VTOC Index Error Message and Associated Codes" on page 297.)
04(X'04')	An error occurred. The CVSTAT field in the CVPL contains an indication of the cause of the error. (CVSTAT code descriptions are in Appendix C, "VTOC Index Error Message and Associated Codes" on page 297.)
08(X'08')	Invalid VTOC index structure while processing a request to read or write a VTOC index record. The CVSTAT field in the CVPL contains an indication of the cause of the error. (CVSTAT code descriptions are in Appendix C, "VTOC Index Error Message and Associated Codes" on page 297.)
12(X'0C')	The CVAF parameter list is not in your protect key or is invalid (the ID is invalid, or the length field is incorrect, or the CVFCTN field is invalid). The CVPL has not been modified.
16(X'10')	An I/O error was encountered.

CVAFDSM Macro

Overview of the CVAFDSM Macro

The CVAFDSM macro may be used for an indexed VTOC to:

- Obtain one or more extents that describe unallocated space on the volume
- Obtain a count of free DSCBs on the VTOC
- Obtain a count of free VTOC index records in the VTOC index.

Syntax

[*label*]	**CVAFDSM**	**ACCESS=MAPDATA**
		,MAP=INDEX | VOLUME | VTOC
		[,EXTENTS=*addr*]
		[,MAPRCDS=YES[1] | (YES,*addr*) | NO[2] |
		(NO,*addr*)]
		[,UCB=*(reg)* | DEB=*addr*]
		[,COUNT=YES | NO]
		[,CTAREA=*addr*]
		[,IOAREA=KEEP | (KEEP,*addr*) | NOKEEP |
		(NOKEEP,*addr*)]
		[,BRANCH=NO | YES[3] | (YES,SUP) | (YES,PGM)]
		[,MF=I | L | (E,*addr*)]

[1] Default if MF=I.

[2] Default if MF=L or MF=(E,addr).

[3] Default is SUP if YES is coded.

ACCESS=MAPDATA: Request Information from the Index Space Maps

ACCESS=MAPDATA
Obtains data from the index space maps. Three kinds of data are available:

- The number of format-0 DSCBs (the data is obtained from the VTOC map of DSCBs)
- The number of unallocated VIRs in the index (the data is obtained from the VTOC index map)
- The number (and location) of extents of unallocated pack space (the data is obtained from the VTOC pack space map)

MAP: Identify the Map to Be Accessed

MAP=INDEX
Specifies that the VTOC index map (VIXM) is to be accessed and a count of unallocated VIRs returned. COUNT=YES must also be coded.

MAP=VOLUME
Specifies that the VTOC pack space map (VPSM) is to be accessed and information on unallocated extents of pack space returned. EXTENTS=addr and COUNT=NO must also be coded.

MAP=VTOC
Specifies that the VTOC map of DSCBs (VMDS) is to be accessed and a count of format-0 DSCBs returned. COUNT=YES must also be coded.

EXTENTS: Identify Where Extents from the VPSM Are Returned

EXTENTS=*addr*
If one or more extents from the VPSM are requested, EXTENTS is the address of a 1-byte count field containing the number of 5-byte extents that follow. In the first two bytes of the first 5-byte extent, you must supply the relative track address (RTA) at which CVAF should start the VPSM search. The first extent area is updated with information on the next free extent found that has a higher starting RTA than that supplied. Each subsequent extent area is filled in with information on free space extents (in ascending track address order).

Information on free extents has the format, XXYYZ, where:

- XX is the relative track address of the first track of the extent.
- YY is the number of whole cylinders in the extent.
- Z is the number of additional tracks in the extent.

Only XX is supplied by the caller in the first extent area. CVAF will start searching the VPSM at relative track address XX.

If all the unallocated extents in the VPSM are provided before filling in all the supplied extent areas, the remaining extent areas are set to zero. Register 15 is set to 4 on return, with the CVSTAT field in the CVPL set to X'20' to indicate end of data.

MAPRCDS: Keep or Free MAPRCDS Buffer List and Buffers

MAPRCDS=YES
Specifies that the buffer list and buffers are to be retained at the end of the function.

If YES is specified and no buffer list is supplied through the CVAF parameter list, CVAF will read the MAP VIRs into buffers obtained by CVAF. The buffer list that contains the address and RBAs of the VIRs can be accessed after the CVAF call from the CVAF parameter list field, CVMRCDS. The buffer list and VIR buffers are in the caller's protect key: subpool 0 if the caller is not authorized; subpool 229 if the caller is authorized.

YES is the default if MF=I is specified or defaulted.

When processing on the current volume is finished, release all areas that were kept.

MAPRCDS=(YES,*addr*)
If YES is coded, but the buffer list address (CVMRCDS in CVAF parameter list) is supplied, the VIRs are not read.

The CVMRCDS buffer list from one CVAF call can be passed to another CVAF macro call through the MAPRCDS keyword.

MAPRCDS=NO
If MAPRCDS=NO is coded, the MAP records buffers and buffer list will be freed upon completion of the CVAFDSM function.

NO is the default if MF=L is specified.

MAPRCDS=(NO,*addr*)
Causes buffer lists and buffers previously obtained by CVAF to be freed.

Buffer lists and buffers obtained by CVAF must be freed by the caller. This can be done in one of three ways:

- By coding MAPRCDS=NO on the call that obtained the buffers.
- By coding MAPRCDS=NO on a subsequent CVAF call.
- By calling CVAFDIR ACCESS=RLSE and providing the buffer list in the BUFLIST keyword.

  If MF=(E,addr) is coded and MAPRCDS is not coded, the parameter list value of MAPRCDS is not changed.

Note: You must enqueue the VTOC and reserve the unit to maintain the integrity of the MAP records read.

UCB | DEB: Specify the VTOC to Be Accessed

UCB=*(reg)*
Supplies the address of the UCB for the unit whose VTOC is to be accessed. An unauthorized caller may not supply a UCB to CVAF.

Note: Code the address of the UCB parameter only as register (2-12). Coding an RX-Type address here gives you unpredictable results.

DEB=*addr*
Specifies the address of a DEB opened to the VTOC you want to access. CVAF does not allow output requests to the VTOC or VTOC index if you specify the DEB subparameter. If you are not authorized, you cannot

perform any asynchronous activity (such as EXCP, CLOSE, EOV) against the data set represented by the DEB because CVAF removes the DEB from the DEB table for the duration of the CVAF call. If you are not authorized (neither APF authorized nor in a system key) you must specify a DEB address, not a UCB, to CVAFDSM. See "Identifying the Volume" on page 42 for further details.

If a previously obtained CVAF I/O area is supplied through the IOAREA keyword, neither UCB nor DEB need be supplied. Otherwise, either a UCB or DEB must be supplied. If a UCB address is supplied, it will be overlaid in the CVPL with the UCB address in the I/O area.

If DEB and UCB are supplied in the CVPL, the DEB will be used, and the UCB address supplied will be overlaid in the CVPL with the UCB address obtained from the DEB.

COUNT: Obtain a Count of Unallocated DSCBs or VIRs

COUNT=YES
Indicates that a count of unallocated DSCBs or VIRs in the designated space map is requested. MAP=VTOC or MAP=INDEX must be specified if COUNT=YES is coded.

COUNT=NO
Indicates that a count of unallocated DSCBs or VIRs is not desired but, rather, information on free space on the pack is desired. MAP=VOLUME must be coded if COUNT=NO is coded or defaulted.

CTAREA: Supply a Field to Contain the Number of Format-0 DSCBs

CTAREA=*addr*
Gives the address of a 4-byte field to contain the number of format-0 DSCBs when COUNT=YES, MAP=VTOC is specified; or the number of unallocated VIRs in the VTOC index when COUNT=YES, MAP=INDEX is specified.

IOAREA: Keep or Free the I/O Work Area

IOAREA=KEEP
Specifies that the CVAF I/O area associated with the CVAF parameter list is to be kept upon completion of the CVAF request. IOAREA=KEEP may be coded with BRANCH=NO only if the caller is authorized (APF or system key).

If IOAREA=KEEP is coded, the caller must issue CVAF with IOAREA=NOKEEP specified at some future time, whether or not any further VTOC access is required: for example, the recovery routine of the caller of CVAF.

Coding IOAREA=KEEP allows subsequent CVAF requests to be more efficient, as certain initialization functions can be bypassed. Neither DEB nor UCB need be specified when a previously obtained CVAF I/O area is supplied; neither can they be changed.

When IOAREA=KEEP is first issued, CVAF returns the CVAF I/O area in the CVAF parameter list (CVIOAR). Subsequent calls of CVAF may use that same parameter list, and CVAF will obtain its I/O area from the CVIOAR.

When processing on the current volume is finished, release all areas that were kept.

IOAREA=(KEEP,*addr*)
Provides the address of a previously obtained I/O area. If a different CVAF parameter list is used, the previously obtained CVAF I/O area may be passed to CVAF by coding its address as the second parameter of the IOAREA keyword.

<u>IOAREA=NOKEEP</u>
Causes the work area to be freed upon completion of the CVAF request.

IOAREA=(NOKEEP,*addr*)
Causes a previously obtained work area to be freed upon completion of the CVAF request.

BRANCH: Specify the Entry to the Macro

BRANCH=(YES,SUP)
Requests that the branch entry to CVAFDIR be used. The caller must be in supervisor state. Protect key checking is bypassed.

An 18-word save area must be supplied if BRANCH=YES is coded. No lock may be held on entry to CVAF. SRB mode is not allowed.

BRANCH=YES
Is equivalent to BRANCH=(YES,SUP), because SUP is the default when YES is coded. Protect key checking is bypassed.

BRANCH=(YES,PGM)
Requests the branch entry. The caller must be APF authorized and in problem state. Protect key checking is bypassed.

<u>BRANCH=NO</u>
Requests the SVC entry. The caller must be APF authorized if any output operations are requested. Protect key checking is performed.

MF: Specify the Form of the Macro

This keyword specifies whether the list, execute, or normal form of the macro is requested.

<u>MF=I</u>
If I is coded or if neither L nor E is coded, the CVAF parameter list is generated, as is code, to call CVAF. This is the normal form of the macro.

MF=L

L indicates the list form of the macro. A parameter list is generated, but code to call CVAF is not generated.

MF=(E,*addr*)

E indicates the execute form of the macro. The remote CVAF parameter list supplied as X'addr' is used in, and can be modified by, the execute form of the macro.

Return Codes from the CVAFDSM Macro

On return from CVAF, register 1 contains the address of the CVPL (CVAF parameter list), and register 15 contains one of the following return codes:

Code	Meaning
0(X'00')	The request was successful.
4(X'04')	End of data (CVSTAT is set to decimal 32), or an error was encountered. The CVSTAT field in the CVPL contains an indication of the cause of the error. (CVSTAT code descriptions are in Appendix C, "VTOC Index Error Message and Associated Codes" on page 297)
8(X'08')	Invalid VTOC index structure. CVSTAT contains an indication of the cause of the error. (CVSTAT code descriptions are in Appendix C, "VTOC Index Error Message and Associated Codes" on page 297)
12(X'0C')	The CVAF parameter list is not in your protect key or is invalid (the ID is invalid, or the length field is incorrect, or the CVFCTN field is invalid). The CVPL has not been modified.
16(X'10')	An I/O error was encountered.

CVAFFILT Macro

Overview of the CVAFFILT Macro

You may use the CVAFFILT macro to invoke the CVAF filter service. You may also use it to map or initialize the CVAF parameter list (CVPL). To accommodate the FSA address, the CVPL generated by CVAFFILT is 4 bytes longer than the CVPL generated by the other CVAF macros. CVAF filter retrieves data set DSCB chains from an indexed or nonindexed VTOC and places them in buffers you provide. You may request the DSCBs for a single partially qualified data set name or for a list of fully qualified data set names. You must identify a specific DASD device and provide both a filter criteria list (FCL) defining the request, and a CVAF buffer list (with buffers) for DSCB return. The format of the two elements of the FCL is shown in Figure 18 on page 54 and Figure 19 on page 56. The format of the buffer list is shown in "Buffer Lists" on page 45. CVAFFILT returns a complete set of DSCBs in the order that they are chained in the VTOC (format-1, format-2, then format-3).

Note: Keywords coded on the list form of the macro need not be coded on the execute form. Keywords coded on one CVAFFILT call remain in effect for subsequent calls unless overridden, if you use the same CVAFFILT parameter list.

Syntax

[*label*]	**CVAFFILT**	**[,ACCESS={READ | RESUME | RLSE}]**
		[,BUFLIST={*addr* | (*reg*)}]
		[,{UCB={*addr* | (*reg*)} | DEB={*addr* | (*reg*)}]
		[,FCL={*addr* | (*reg*)}]
		[,FLTAREA={KEEP | KEEP,{*addr* | (*reg*)} |
		NOKEEP | NOKEEP,{*addr* | (*reg*)}}]
		[,IOAREA={KEEP | KEEP,{*addr* | (*reg*)} |
		NOKEEP | NOKEEP,{*addr* | (*reg*)}}]
		[,BRANCH={NO | YES | (YES,{SUP | PGM})}]
		[,{MF=I | MF=D | MF=L | MF=(E,{*addr* | (*reg*))}]

Note: For the first operand following CVAFFILT, you must not code the leading comma.

Control Block Address Resolution:

Keyword=*addr* | *(reg)*

You, as the caller, either define or reference the control blocks needed by CVAF filter (caller-defined control blocks are: BUFLIST, CVPL, and FCL. Caller-referenced control blocks are: DEB, FLTAREA, IOAREA, and UCB). The CVAFFILT macro generates different instructions for keyword=addr and keyword=(reg) depending upon whether you are specifying a "defined" or "referenced" control block.

1. When you specify any control block's address as '(reg)', the CVAFFILT macro assumes that the register specified contains that address.

2. When you specify a "defined" control block's address as 'addr', the CVAFFILT macro assumes that the specified location is that of the control block itself. The macro generates a load address instruction (LA) to obtain the control block's address.

3. When you specify a "referenced" control block's address as 'addr', the CVAFFILT macro assumes that the specified location is that of a fullword containing the address of the control block. The macro generates a load instruction (L) to obtain the control block's address.

ACCESS: Retrieve or Resume Retrieving a DSCB, or Release FLTAREA and/or IOAREA

ACCESS=READ

Retrieves all DSCBs associated with the data set name(s) specified in the filter criteria list (FCL), placing them in your buffers. You can select (filter) the retrieved DSCBs by providing either a list of one or more fully qualified names, or a single partially qualified name, using single or double asterisk notation. (See the example of partially qualified names in "Examples of Partially Qualified Names for CVAFFILT" on page 250.)

If the number of buffers is not large enough to hold all the requested DSCBs, CVAFFILT indicates this in the CVSTAT status byte of the CVAF parameter list (CVPL). You can resume the READ function by issuing a call with ACCESS=RESUME. See "Codes Put in the CVSTAT Field" on page 298.

When selecting DSCBs by partially qualified name, CVAFFILT uses only the first data set name in the FCL list. You must set the FCLCOUNT count field in the FCL to '1' or CVAFFILT returns error code 63 in the CVSTAT status byte of the CVPL. The DSCBs returned by CVAFFILT may not be in sequence by data set name; however, the DSCBs for each data set are always in order (format-1, format-2, format-3).

When selecting DSCBs by fully qualified names, you may request that CVAF filter return the DSCBs for the selected data set names in the data set name order implied by the FCL. See the FCL1ORDR flag in Figure 18 on page 54.

You should always test the status byte of each data set name in the FCL list to ensure successful completion (Some error conditions result in failure to return a data set's DSCBs). See the FCLDSNST byte in Figure 19 on page 56.

ACCESS=RESUME

Resumes a previously initiated READ or RESUME function that was terminated because you did not provide enough buffers to contain all the requested DSCBs. For the RESUME function to execute correctly, you must have coded the keyword FLTAREA=KEEP in each of the previous READ and RESUME function calls.

ACCESS=RLSE
Releases the previously kept filter save area (FLTAREA) and/or CVAF I/O work area (IOAREA).

UCB | DEB: Specify the VTOC to Be Accessed

UCB=*addr* | *(reg)*
Supplies the address of the UCB for the unit whose VTOC is to be accessed. If you are not authorized, you cannot supply a UCB to CVAF. CVAF returns CVSTAT '8' and return code '4' if you specify a UCB and you are not authorized.

DEB=*addr* | *(reg)*
Specifies the address of a DEB opened to the VTOC you want to access. If you are not authorized, you cannot perform any asynchronous activity against the data set represented by the DEB (such as EXCP, CLOSE, EOV), because CVAF removes the DEB from the DEB table for the duration of the CVAF call. If you are not authorized (neither APF authorized nor in a system key) you must specify a DEB address, not a UCB, to CVAFFILT. See "Identifying the Volume" on page 42 for further details.

BUFLIST: Specify a Buffer List

BUFLIST=*addr* | *(reg)*
The BUFLIST keyword specifies the address of a buffer list used to read DSCBs. When you specify ACCESS=RLSE, the BUFLIST keyword is required for the standard form of the macro. See the format of the buffer list header and buffer list entry in Figure 15 on page 46 and Figure 16 on page 47, respectively.

FCL: Specify a Filter Criteria List

FCL=*addr* | *(reg)*
The FCL keyword specifies the address of a filter criteria list. It is required when ACCESS=READ is specified on the standard form of the macro. The format of the two elements of the filter criteria list is shown in Figure 18 on page 54 and Figure 19 on page 56.

FLTAREA: Keep or Free the Filter Save Area

FLTAREA=KEEP
Specifies keeping the filter save area. You must code this operand if the RESUME function might be called later (to resume processing prematurely terminated because the number of caller-supplied buffers is not enough to contain all the returned DSCBs).

CVAFFILT returns the address of the kept filter save area in the CVAFFILT parameter list (CVFSA field). If you specify the same parameter list in subsequent RESUME calls, CVAFFILT reuses the same filter save area.

Note: If you code this operand, you must subsequently issue CVAFFILT with ACCESS=RLSE to release the filter save area.

FLTAREA=(KEEP,{*addr* | (*reg*)})
Specifies the address of a previously obtained filter save area. See the description of FLTAREA=KEEP operand for additional concerns.

<u>FLTAREA=NOKEEP</u>
Frees the filter save area upon completion of the CVAF request.

FLTAREA=(NOKEEP,{*addr* | (*reg*)})
Frees a previously obtained filter save area upon completion of the CVAF request.

IOAREA: Keep or Free the I/O Work Area

IOAREA=KEEP
Specifies keeping the CVAF I/O work area. For authorized callers, CVAFFILT returns the address of the kept I/O work area in the CVAFFILT parameter list (CVIOAR). If you specify the same parameter list in subsequent calls, CVAFFILT reuses the same I/O work area.

Note: If you code this operand, you must subsequently issue CVAFFILT with ACCESS=RLSE to release the I/O work area.

IOAREA=(KEEP,{*addr* | (*reg*)})
Provides the address of a previously obtained filter save area. See the description of IOAREA=KEEP operand for additional concerns.

<u>IOAREA=NOKEEP</u>
Frees the filter save area upon completion of the CVAF request.

IOAREA=(NOKEEP,{*addr* | (*reg*)})
Frees a previously obtained CVAF I/O work area upon completion of the CVAF request.

BRANCH: Specify the Entry to the Macro

<u>BRANCH=NO</u>
Requests the SVC (default) entry. Protect key checking is performed.

BRANCH=YES
Equivalent to BRANCH=(YES,SUP), because SUP is the default when you code YES. You must be in supervisor state. Protect key checking is bypassed.

BRANCH=(YES,SUP)
Requests the branch entry. You must be in supervisor state. Protect key checking is bypassed. You must supply an 18-word save area if you specify BRANCH=YES. You cannot hold a lock at entry to CVAF. You cannot be in SRB mode.

BRANCH=(YES,PGM)
Requests the branch entry. You must be APF authorized and be in problem state. Protect key checking is bypassed.

MF: Specify the Form of the Macro

Specifies whether the DSECT, list, execute, or normal form of the macro is requested. You can be in either 24-bit or 31-bit addressing mode. If you are not authorized, you must pass the address of a DEB built by OPEN. If you are authorized, you may pass either the DEB address or the UCB address. You must ensure that the volume is allocated and will remain mounted (for example, by dynamic allocation).

MF=I
Specifies the standard form of the macro. The CVAF parameter list is generated and CVAF is called. 'MF=I' is the default value.

MF=D
Specifies the DSECT form of the macro. The macro generates a request for the ICVAFPL macro to map the unique CVAF filter CVPL (4-bytes longer than standard CVPL).

MF=L
Specifies the list form of the macro. The CVAF parameter list is generated, but CVAF is not called.

MF=(E,{*addr* | (reg)})
Specifies the execute form of the macro. The CVAF parameter list whose address is in 'addr' or 'reg' is used. You can modify the parameter list with this form of the macro.

Return Codes from the CVAFFILT macro

CVAF filter service does not issue any messages. Upon completion of a filter request, register 15 contains one of the following return codes:

Code	Meaning
00(X'00')	The request was successful.
04(X'04')	Logical error; status information in CVSTAT.
08(X'08')	Invalid VTOC structure.
12(X'0C')	CVAFFILT parameter list in wrong key, or invalid.
16(X'10')	I/O error.

Note: CVSTAT in the CVAF parameter list explains the status codes. See "Codes Put in the CVSTAT Field" on page 298 for a list of the status codes.

Examples of Partially Qualified Names for CVAFFILT

CVAFFILT supports partially qualified data set names using single or double asterisk notation as shown below:

- You may use a single asterisk to represent a single qualifier. For example, SYS1.*.LOAD designates any data set with three qualifiers, the first being SYS1, the second being any qualifier, and the third being LOAD.

- You may also use a single asterisk to represent zero or more unspecified characters. For example, LOAD.*LIB designates any data set having only two qualifiers, with LOAD being the first, and the second qualifier ending with the character string LIB (for example, LINKLIB). The asterisk may appear anywhere within the qualifier. You may use two single asterisks in the following way: LOAD.A*B*.LIB. CVAFFILT does not support the use of two or more single asterisks with any other character within a single qualifier (for example, LOAD.B**.LIB is invalid).

- A double asterisk represents a place holder for zero or more qualifiers. For example, SYS1.** designates any data set having SYS1 as its first or only qualifier.

CVAFSEQ Macro

Overview of the CVAFSEQ Macro

The CVAFSEQ macro may be used to:

- Read an indexed VTOC sequentially in data-set-name (DSN) order
- Read an indexed VTOC or a nonindexed VTOC in physical-sequential order

Syntax

[*label*]	**CVAFSEQ**	**ACCESS=GT | GTEQ**
		[,BUFLIST=*addr*]
		[,DSN=*addr*]
		[,UCB=*(reg)* | DEB=*addr*]
		[,DSNONLY=NO | YES]
		[,ARG=*addr*]
		[,IOAREA=KEEP | (KEEP,*addr*) | NOKEEP | (NOKEEP,*addr*)]
		[,IXRCDS=KEEP | (KEEP*addr*) | NOKEEP | (NOKEEP,*addr*)]
		[,BRANCH=NO | YES[1] | (YES,SUP) | (YES,PGM)]
		[,MF=I | L | (E,*addr*)]

1 If YES, default is SUP.

ACCESS: Specify Relationship between Supplied and Returned DSN

ACCESS=GT
: Specifies that the DSN or argument value is to be used to return a DSCB whose DSN or argument is greater than that supplied.

ACCESS=GTEQ
: Specifies that the DSN or argument value is to be used to return a DSCB whose DSN or argument is greater than or equal to that supplied.

Note: A CVAF call specifying ACCESS=GTEQ should be followed by an ACCESS=GT request, or the same DSCB or name will be returned.

BUFLIST: Specify One or More Buffer Lists

BUFLIST=*addr*
: The BUFLIST keyword supplies the address of a buffer list used to read or write DSCBs and VIRs.

DSN: Specify Access by DSN Order or by Physical-Sequential Order

DSN=*addr*
Specifies that access of an indexed VTOC is by DSN order. BUFLIST is required if DSNONLY=NO is coded or defaulted.

DSN omitted
If you omit the DSN keyword, access of an indexed or nonindexed VTOC is by physical-sequential order. BUFLIST is required.

Note: If the order is physical-sequential, you must initialize the argument field in the first buffer list entry to zero or to the argument of the DSCB. If the argument is zero (BFLEARG=00), the read begins at the start of the VTOC. You must be authorized (APF or system key) to read multiple DSCBs with a single invocation of the CVAFSEQ macro. See "Initiating Physical-Sequential Access" on page 51 for more information.

UCB | DEB: Specify the VTOC to Be Accessed

UCB=*(reg)*
Supplies the address of the UCB for the unit whose VTOC is to be accessed. An unauthorized caller may not supply a UCB to CVAF.

Note: Code the address of the UCB parameter only as register (2-12). Coding an RX-type address here gives you unpredictable results.

DEB=*addr*
Specifies the address of a DEB opened to the VTOC you want to access. CVAF does not allow output requests to the VTOC or VTOC index if you specify the DEB subparameter. If you are not authorized, you cannot perform any asynchronous activity (such as EXCP, CLOSE, EOV), against the data set represented by the DEB because CVAF removes the DEB from the DEB table for the duration of the CVAF call. If you are not authorized (neither APF authorized nor in a system key), you must specify a DEB address, not a UCB, to CVAFSEQ. See "Identifying the Volume" on page 42 for further details.

If a previously obtained CVAF I/O area is supplied through the IOAREA keyword, neither UCB nor DEB need be supplied.

Otherwise, either a UCB or DEB must be supplied. If a UCB address is supplied, it will be overlaid in the CVPL with the UCB address in the I/O area.

If you specify both DEB and UCB in the CVPL, the DEB will be used, and the UCB address supplied will be overlaid in the CVPL with the UCB address obtained from the DEB.

DSNONLY: Specify That Only the Data Set Name Be Read

This keyword is applicable only to accessing an indexed VTOC in DSN order.

DSNONLY=NO
Requests that the data set name be obtained from the VTOC index and the DSCB be read into a buffer supplied through the BUFLIST keyword. BUFLIST is required.

DSNONLY=YES
Requests that only the data set name be obtained from the VTOC index. If the ARG keyword is coded, the argument of the DSCB is returned.

ARG: Specify Where the Argument of the DSCB Is to Be Returned

This keyword is applicable only to accessing an indexed VTOC in DSN order with DSNONLY=YES coded.

ARG=*addr*
Provides the address of the 5-byte area where the CCHHR of each data set name in the VTOC index is returned when DSNONLY=YES is coded.

IOAREA: Keep or Free the I/O Work Area

IOAREA=KEEP
Specifies that the CVAF I/O area associated with the CVAF parameter list is to be kept upon completion of the CVAF request. IOAREA=KEEP may be coded with BRANCH=NO only if the caller is authorized (APF, or system key).

If IOAREA=KEEP is coded, the caller must issue CVAF with IOAREA=NOKEEP specified at some future time, whether or not any further VTOC access is required: for example, the recovery routine of the caller of CVAF.

Coding IOAREA=KEEP allows subsequent CVAF requests to be more efficient, because certain initialization functions can be bypassed. Neither DEB nor UCB need be specified when a previously obtained CVAF I/O area is supplied; neither can they be changed.

When IOAREA=KEEP is first issued, CVAF returns the CVAF I/O area in the CVAF parameter list (CVIOAR). Subsequent calls of CVAF may use that same parameter list, and CVAF will obtain its I/O area from the CVIOAR.

When processing on the current volume is finished, release all areas that were kept.

IOAREA=(KEEP,*addr*)
Provides the address of a previously obtained I/O area. If a different CVAF parameter list is used, the previously obtained CVAF I/O area may be passed to CVAF by coding its address as the second parameter of the IOAREA keyword.

IOAREA=NOKEEP
Causes the work area to be freed upon completion of the CVAF request.

IOAREA=(NOKEEP,*addr*)
Causes a previously obtained work area to be freed upon completion of the CVAF request.

IXRCDS: Retain VIERs in Virtual Storage

This keyword applies to an indexed VTOC only.

IXRCDS=KEEP
Specifies that the VIERs read into storage during the CVAF function are to be kept in virtual storage. The VIERs are retained even if the index function is unsuccessful. The VIERs are accessed from the CVAF parameter list (CVIRCDS). CVIRCDS is the address of a buffer list containing the VIR buffer addresses and RBAs of the VIERs read.

Index search function will dynamically update the buffer list and, when necessary, obtain additional buffer lists and chain them together.

If KEEP is specified and no buffer list is supplied to CVAF in the CVPL, CVAF will obtain a buffer list and buffers and read the first high-level VIER. The address of the buffer list is placed in the CVIRCDS field of the CVPL. The first high-level VIER will be checked for the VXFHLV bit and to see if the VXVISE bit is off.

The buffer list and VIR buffers are in the caller's protect key. The subpool is 0 if the caller is not authorized; subpool 229 if the caller is authorized.

If IXRCDS=KEEP for an nonindexed VTOC, a request to read a DSCB may be performed, but an error code will be returned.

When processing on the current volume is finished, release all areas that were kept.

IXRCDS=(KEEP,*addr*)
The CVIRCDS from one CVAF call can be passed to another CVAF parameter list by specifying the address as the second parameter in the IXRCDS keyword.

IXRCDS=NOKEEP
If NOKEEP is coded, the VIERs that are accessed (if any) are not retained. Furthermore, the buffer list supplied in the CVIRCDS field in the CVAF parameter list is released, as are all buffers found in the buffer list. If the skip bit is set in any entry in the buffer list, the buffer and buffer list will not be freed.

IXRCDS=(NOKEEP,*addr*)
Specifies that previously accessed VIERs are not to be retained.

You must free buffer lists and buffers obtained by CVAF. This can be done in one of three ways:

- By coding IXRCDS=NOKEEP on the CVAFSEQ macro that obtained the buffers
- By coding IXRCDS=NOKEEP on a subsequent CVAF macro
- By coding CVAFDIR ACCESS=RLSE and providing the address of the buffer list in the BUFLIST keyword

Note: You must enqueue the VTOC and reserve the unit to maintain the integrity of the VIERs read.

BRANCH: Specify the Entry to the Macro

BRANCH=(YES,SUP)
Requests that the branch entry to CVAFDIR be used. The caller must be in supervisor state. Protect key checking is bypassed.

An 18-word save area must be supplied if BRANCH=YES is coded. No lock may be held on entry to CVAF. SRB mode is not allowed.

BRANCH=YES
Is equivalent to BRANCH=(YES,SUP), because SUP is the default when YES is coded. Protect key checking is bypassed.

BRANCH=(YES,PGM)
Requests the branch entry. The caller must be APF authorized and in problem state. Protect key checking is bypassed.

BRANCH=NO
Requests the SVC entry. The caller must be APF authorized if any output operations are requested. Protect key checking is performed.

MF: Specify the Form of the Macro

This keyword specifies whether the list, execute, or normal form of the macro is requested.

MF=I
If I is coded, or neither L nor E is coded, the CVAF parameter list is generated, as is code, to call CVAF. This is the normal form of the macro.

MF=L
L indicates the list form of the macro. A parameter list is generated, but code to call CVAF is not generated.

MF=(E,*addr*)
E indicates the execute form of the macro. The remote CVAF parameter list supplied as 'addr' is used in and can be modified by the execute form of the macro.

Return Codes from the CVAFSEQ Macro

On return from CVAF, register 1 contains the address of the CVPL (CVAF parameter list), and register 15 contains one of the following return codes:

Code	Meaning
00(X'00')	The request was successful.
04(X'04')	End of data (CVSTAT is set to decimal 32), or an error was encountered. The CVSTAT field in the CVPL contains an indication of the cause of the error. Error descriptions are in Appendix C, "VTOC Index Error Message and Associated Codes" on page 297.
08(X'08')	Invalid VTOC index structure. CVSTAT contains an indication of the cause of the error. Error descriptions are in Appendix C, "VTOC Index Error Message and Associated Codes" on page 297.
12(X'0C')	The CVPL (CVAF parameter list) is not in your protect key, or is invalid (the ID is invalid, or the length field is incorrect, or the CVFCTN field is invalid). The CVPL has not been modified.
16(X'10')	An I/O error was encountered.

CVAFTST Macro

Overview of the CVAFTST Macro

The CVAFTST macro determines whether the system supports an indexed VTOC, and, if it does, whether the VTOC on the unit whose UCB is supplied is indexed or nonindexed.

You will get a return code of 12 if CVAFTST cannot determine whether an indexed or nonindexed VTOC is on the unit's volume. You should not receive a return code of 12 from CVAFTST if you have opened a data set (including the VTOC) on the volume.

You need no authorization to issue the CVAFTST macro.

Syntax

[*label*]	CVAFTST	UCB=*(reg)*

UCB: Specify the VTOC to Be Tested

UCB=*(reg)*
: Supplies the address of the UCB for the volume whose VTOC is to be tested.

  Note: Code the address of the UCB parameter only as register (2-12). Coding an RX-type address here gives you unpredictable results.

Return Codes from the CVAFTST Macro

On return from CVAF, register 15 contains one of the following return codes:

Code	Meaning
0(X'00')	The system does not support an indexed VTOC. The volume should be considered to have a nonindexed VTOC. The UCB was not inspected to determine its validity or status.
4(X'04')	The system supports an indexed VTOC, but the volume has a nonindexed VTOC.
8(X'08')	The system supports an indexed VTOC and the volume has an indexed VTOC.
12(X'0C')	The system supports an indexed VTOC, but the volume is not mounted or the VIB is not initialized for it; thus, the status (indexed or nonindexed) of the VTOC cannot be determined.
16(X'10')	The system supports an indexed VTOC, but the unit is not a DASD or has a VIO UCB, or the UCB address is invalid.

Appendix B. Examples of VTOC Access Macros

The examples that follow are partial assembler listings that include expansions of each VTOC access macro. The expansions are provided to show how the VTOC macros can be substituted for existing procedures.

Example 1: Using the CVAFDIR Macro with an Indexed or Nonindexed VTOC

This example uses the CVAFDIR macro to read a DSCB of a given data set name and determines whether the DSCB is for a partitioned data set. The address of the 44-byte data set name is supplied to the program in register 5 (labeled RDSN in the example). The address of a DEB open to the VTOC is supplied to the program in register 4 (labeled RDEB in the example).

The buffer list is in the program and is generated by the ICVAFBFL macro. The DSCB buffer is in the program and is generated by the IECSDSL1 macro.

```
EXAMPLE1 CSECT
         STM   14,12,12(RSAVE)
         BALR  12,0
         USING *,12
         ST    RSAVE,SAVEAREA+4
         LA    RWORK,SAVEAREA
         ST    RWORK,8(,RSAVE)
         LR    RSAVE,RWORK
***************************************************************
*
*        REGISTERS
*
***************************************************************
REG1     EQU   1                        REGISTER 1
RWORK    EQU   3                        WORK REGISTER
RDEB     EQU   4                        DEB ADDRESS
RDSN     EQU   5                        ADDRESS OF DATA SET NAME
RSAVE    EQU   13                       SAVE AREA ADDRESS
REG15    EQU   15                       RETURN CODE REGISTER 15
```

```
************************************************************
*
*        RETURN CODES
*
************************************************************
PDSRTN   EQU   0                  DATA SET A PDS RETURN CODE
NOTFND   EQU   4                  DATA SET NOT FOUND RETURN CODE
NOTPDS   EQU   8                  DATA SET NOT A PDS RETURN CODE
UNEXPECD EQU   12                 UNEXPECTED ERROR RETURN CODE
************************************************************
*
*        READ DSCB INTO DS1FMTID.
*        DATA SET NAME ADDRESS SUPPLIED IN RDSN.
*        ADDRESS OF DEB OPEN TO VTOC SUPPLIED IN RDEB.
*        DETERMINE IF DATA SET IS A PARTITIONED DATA SET.
*        THIS PROGRAM IS NEITHER REENTRANT NOR REUSABLE.
*
************************************************************
         XC    BUFLIST(BFLHLN+BFLELN),BUFLIST ZERO BUFFER LIST
         OI    BFLHFL,BFLHDSCB      DSCBS TO BE READ WITH BUFFER LIST
         MVI   BFLHNOE,1            ONE BUFFER LIST ENTRY
         LA    RWORK,DS1FMTID       ADDRESS OF DSCB BUFFER
         ST    RWORK,BFLEBUF        PLACE IN BUFFER LIST
         OI    BFLEFL,BFLECHR       CCHHR OF DSCB RETURNED BY CVAF
         MVI   BFLELTH,DSCBLTH      DATA PORTION OF DSCB READ - DSN     *
                                    SUPPLIED IN CVPL
         MVC   DS1DSNAM,0(RDSN)     MOVE IN DATA SET NAME TO WORKAREA
         CVAFDIR ACCESS=READ,DSN=DS1DSNAM,BUFLIST=BUFLIST,DEB=(RDEB)
+        CNOP  0,4
+        BAL   1,ICV1E                   LOAD CVPL LIST ADDRESS
+ICV1S   EQU   *                         START OF CVPL
+        DC    CL4'CVPL'                 EBCDIC 'CVPL'
+        DC    AL2(ICV1E-ICV1S)          LENGTH OF CVPL
+        DC    XL1'01'                   FUNCTION CODE
+        DC    XL1'00'                   STATUS INFORMATION
+        DC    B'00000000'               FIRST FLAG BYTE
+        DC    B'00000000'               SECOND FLAG BYTE
+        DC    H'0'                      RESERVED
+        DC    A(0)                      UCB ADDRESS
+        DC    A(DS1DSNAM)               DATA SET NAME ADDRESS
+        DC    A(BUFLIST)                BUFFER LIST ADDRESS
+        DC    A(0)                      INDEX VIR'S BUFFER LIST ADDRESS
+        DC    A(0)                      MAP VIR'S BUFFER LIST ADDRESS
+        DC    A(0)                      I/O AREA ADDRESS
+        DC    A(0)                      DEB ADDRESS
+        DC    A(0)                      ARGUMENT ADDRESS
+        DC    A(0)                      SPACE PARAMETER LIST ADDRESS
+        DC    A(0)                      EXTENT TABLE ADDRESS
+        DC    A(0)                      NEW VRF VIXM BUFFER LIST ADDR
+        DC    A(0)                      VRF DATA ADDRESS
+        DC    A(0)                      COUNT AREA ADDRESS
+ICV1E   EQU   *                         END OF CVPL
+        ST    RDEB,36(,1)               STORE DEB PTR IN PARM LIST
+        SVC   139
         USING CVPL,REG1            ADDRESSABILITY TO CVPL
         LTR   REG15,REG15          ANY ERROR
         BZ    NOERROR              BRANCH IF NOT
```

```
***********************************************************
*
*        DETERMINE WHAT ERROR IS
*
***********************************************************
         C    REG15,ERROR4         IS RETURN CODE 4
         BNE  OTHERERR             BRANCH IF NOT 4
         CLI  CVSTAT,STAT001       IS IT DATA SET NAME NOT FOUND?
         BNE  OTHERERR             BRANCH IF NOT
         DROP REG1                 ADDRESSABILITY TO CVPL NOT NEEDED
***********************************************************
*
*        DATA SET NAME NOT FOUND
*
***********************************************************
         L    RSAVE,4(,RSAVE)
         RETURN (14,12),RC=NOTFND  SET UP DATA SET NOT FOUND ERROR
+        LM    14,12,12(13)                   RESTORE THE REGISTERS
+        LA    15,NOTFND(0,0)                 LOAD RETURN CODE
+        BR    14                             RETURN
 NOERROR EQU  *                    DSCB READ
         MVC  F1CCHHR,BFLEARG      MOVE CCHHR OF FORMAT 1/4 DSCB TO    *
                                   WORKAREA
         CLI  DS1FMTID,C'4'        IS DSCB A FORMAT 4 DSCB
         BE   NOTF1                BRANCH IF YES. NOT A FORMAT 1
         TM   DS1DSORG,DS1DSGPO    IS FORMAT 1 DSCB FOR PARTITIONED    *
                                   DATA SET
         BO   PDS                  BRANCH IF PDS
 NOTF1   EQU  *                    DSCB IS NOT A PDS
         L    RSAVE,4(,RSAVE)
         RETURN (14,12),RC=NOTPDS  SET UP NOT PDS RETURN CODE
+        LM    14,12,12(13)                   RESTORE THE REGISTERS
+        LA    15,NOTPDS(0,0)                 LOAD RETURN CODE
+        BR    14                             RETURN
 PDS     EQU  *                    DATA SET IS PARTITIONED
         L    RSAVE,4(,RSAVE)
         RETURN (14,12),RC=PDSRTN  SET UP PDS RETURN CODE
+        LM    14,12,12(13)                   RESTORE THE REGISTERS
+        LA    15,PDSRTN(0,0)                 LOAD RETURN CODE
+        BR    14                             RETURN
 OTHERERR EQU *                    UNEXPECTED ERROR
         L    RSAVE,4(,RSAVE)
         RETURN (14,12),RC=UNEXPECD
+        LM    14,12,12(13)                   RESTORE THE REGISTERS
+        LA    15,UNEXPECD(0,0)               LOAD RETURN CODE
+        BR    14                             RETURN

 ERROR4  DC   F'4'                 ERROR RETURN CODE 4
 BUFLIST ICVAFBFL DSECT=NO         BUFFER LIST
+**********************************************************************
+*       BUFFER LIST HEADER
+**********************************************************************
+BUFLIST DS    0F                          BUFFER LIST HEADER
+BFLHNOE DS    XL1                         NUMBER OF ENTRIES
```

```
+BFLHFL   DS    XL1                       KEY AND FLAG BYTE
+         ORG   BFLHFL
+BFLHKEY  DS    XL1                       PROTECT KEY (FIRST 4 BITS)
+BFLHVIR  EQU   X'08'                     BUF. LIST ENTRIES DESCRIBE VIRS
+BFLHDSCB EQU   X'04'                     BUF. LIST ENTRIES DESCRIBE DSCBS
+         DS    XL1                       RESERVED
+BFLHSP   DS    XL1                       SUBPOOL OF BUF. LIST/BUFFERS
+BFLHFCHN DS    A                         FORWARD CHAIN PTR TO NEXT BUF.
+*                                         LIST
+BFLHLN   EQU   *-BUFLIST                 LENGTH OF BUFFER LIST HEADER
+***********************************************************************
+*        BUFFER LIST ENTRY
+***********************************************************************

+BFLE     DS    0F                        BUFFER LIST ENTRY
+BFLEFL   DS    XL1                       BUFFER LIST ENTRY FLAG
+BFLERBA  EQU   X'80'                     ARGUMENT IS RBA
+BFLECHR  EQU   X'40'                     ARGUMENT IS CCHHR
+BFLETTR  EQU   X'20'                     ARGUMENT IS TTR
+BFLEAUPD EQU   X'10'                     CVAF UPDATED ARGUMENT FIELD
+BFLEMOD  EQU   X'08'                     DATA IN BUF. HAS BEEN MODIFIED
+BFLESKIP EQU   X'04'                     SKIP THIS ENTRY
+BFLEIOER EQU   X'02'                     I/O ERROR
+         DS    XL1                       RESERVED
+BFLELTH  DS    XL1                       LENGTH OF DSCB BUFFER OR
+*                                         LENGTH OF VIR DIVIDED BY 256
+BFLEARG  DS    XL5                       ARGUMENT OF VIR OR DSCB (CCHHR)
+         ORG   BFLEARG+1
+BFLEATTR DS    XL3                       'TTR' OF ARGUMENT
+         ORG   BFLEARG+1
+BFLEARBA DS    XL4                       'RBA'  OF ARGUMENT
+BFLEBUF  DS    A                         BUFFER ADDRESS
+BFLELN   EQU   *-BFLE                    LENGTH OF A BUFFER LIST ENTRY
          IECSDSL1 (1)                FORMAT 1 DSCB DATA SET NAME AND      *
                                      BUFFER
+IECSDSL1 EQU   *                     FORMAT 1 DSCB
+IECSDSF1 EQU   IECSDSL1
+DS1DSNAM DS    CL44                  DATA SET NAME
+DS1FMTID DS    CL1                   FORMAT IDENTIFIER
+DS1DSSN  DS    CL6                   DATA SET SERIAL NUMBER
+DS1VOLSQ DS    XL2                   VOLUME SEQUENCE NUMBER
+DS1CREDT DS    XL3                   CREATION DATE
+DS1EXPDT DS    XL3                   EXPIRATION DATE
+DS1NOEPV DS    XL1                   NUMBER OF EXTENTS ON VOLUME
+DS1NOBDB DS    XL1                   NUMBER OF BYTES USED IN LAST
+*                                        DIRECTORY BLOCK
+         DS    XL1                   RESERVED
+DS1SYSCD DS    CL13                  SYSTEM CODE
+         DS    XL7                   RESERVED
```

```
+DS1DSORG DS     XL2                      DATA SET ORGANIZATION
+*                           FIRST BYTE OF DS1DSORG
+DS1DSGIS EQU    X'80'                    IS - INDEXED SEQUENTIAL          @01A
+*                                        ORGANIZATION
+DS1DSGPS EQU    X'40'                    PS - PHYSICAL SEQUENTIAL         @01A
+*                                        ORGANIZATION
+DS1DSGDA EQU    X'20'                    DA - DIRECT ORGANIZATION         @01A
+DS1DSGCX EQU    X'10'                    CX - BTAM OR QTAM LINE GROUP     @01A
+*        EQU    X'08'                    RESERVED                         @01A
+*        EQU    X'04'                    RESERVED                         @01A
+DS1DSGPO EQU    X'02'                    PO - PARTITIONED ORGANIZATION    @01A
+DS1DSGU  EQU    X'01'                    U - UNMOVABLE, THE DATA          @01A
+*                                        CONTAINS LOCATION DEPENDENT
+*                                        INFORMATION
+*
+*                            SECOND BYTE OF DS1DSORG
+DS1DSGGS EQU    X'80'                    GS - GRAPHICS ORGANIZATION       @01A
+DS1DSGTX EQU    X'40'                    TX - TCAM LINE GROUP             @01A
+DS1DSGTQ EQU    X'20'                    TQ - TCAM MESSAGE QUEUE          @01A
+*        EQU    X'10'                    RESERVED                         @01A
+DS1ACBM  EQU    X'08'                    ACCESS METHOD CONTROL BLOCK      @01A
+DS1DSGTR EQU    X'04'                    TR - TCAM 3705                   @01A
+*        EQU    X'02'                    RESERVED                         @01A
+*        EQU    X'01'                    RESERVED                         @01A
+DS1RECFM DS     XL1                      RECORD FORMAT
+DS1OPTCD DS     XL1                      OPTION CODE
+DS1BLKL  DS     XL2                      BLOCK LENGTH
+DS1LRECL DS     XL2                      RECORD LENGTH
+DS1KEYL  DS     XL1                      KEY LENGTH
+DS1RKP   DS     XL2                      RELATIVE KEY POSITION
+DS1DSIND DS     XL1                      DATA SET INDICATORS
+DS1SCALO DS     XL4                      SECONDARY ALLOCATION
+DS1LSTAR DS     XL3                      LAST USED TRACK AND BLOCK ON TRACK
+DS1TRBAL DS     XL2                      BYTES REMAINING ON LAST TRACK USED
+         DS     XL2                      RESERVED
+DS1EXT1  DS     XL10                     FIRST EXTENT DESCRIPTION
+*        FIRST BYTE                      EXTENT TYPE INDICATOR
+*        SECOND BYTE                     EXTENT SEQUENCE NUMBER
+*        THIRD - SIXTH BYTES             LOWER LIMIT
+*        SEVENTH - TENTH BYTES           UPPER LIMIT
+DS1EXT2  DS     XL10                     SECOND EXTENT DESCRIPTION
+DS1EXT3  DS     XL10                     THIRD EXTENT DESCRIPTION
+DS1PTRDS DS     XL5                      POSSIBLE PTR TO A FORMAT 2 OR 3 DSCB
+DS1END   EQU    *
 DSCBLTH  EQU    *-IECSDSL1-L'DS1DSNAM LENGTH OF DATA PORTION OF DSCB
 F1CCHHR  DS     XL5                      CCHHR OF DSCB
 SAVEAREA DS     18F                      SAVE AREA
 CVPL     ICVAFPL ,                       CVPL MAPPING MACRO

+***********************************************************************
+*        CVAF PARAMETER LIST
+***********************************************************************

+CVPL     DSECT                              CVAF PARAMETER LIST
+         DS     0F
+CVLBL    DS     CL4                         EBCDIC 'CVPL'
+CVLTH    DS     H                           LENGTH OF CVPL
```

```
+CVFCTN    DS    XL1                    FUNCTION BYTE
+CVDIRD    EQU   X'01'                  CVAFDIR ACCESS=READ
+CVDIWR    EQU   X'02'                  CVAFDIR ACCESS=WRITE
+CVDIRLS   EQU   X'03'                  CVAFDIR ACCESS=RLSE
+CVSEQGT   EQU   X'04'                  CVAFSEQ ACCESS=GT
+CVSEQGTE  EQU   X'05'                  CVAFSEQ ACCESS=GTEQ
+CVDMIXA   EQU   X'06'                  CVAFDSM ACCESS=IXADD
+CVDMIXD   EQU   X'07'                  CVAFDSM ACCESS=IXDLT
+CVDMALC   EQU   X'08'                  CVAFDSM ACCESS=ALLOC
+CVDMRLS   EQU   X'09'                  CVAFDSM ACCESS=RLSE
+CVDMMAP   EQU   X'0A'                  CVAFDSM ACCESS=MAPDATA
+CVVOL     EQU   X'0B'                  CVAFVOL ACCESS=VIBBLD
+CVVRFRD   EQU   X'0C'                  CVAFVRF ACCESS=READ
+CVVRFWR   EQU   X'0D'                  CVAFVRF ACCESS=WRITE
+CVSTAT    DS    XL1                    STATUS INFORMATION (SEE LIST    *
+                                       BELOW)
+CVFL1     DS    XL1                    FIRST FLAG BYTE
+CV1IVT    EQU   X'80'                  INDEXED VTOC ACCESSED
+CV1IOAR   EQU   X'40'                  IOAREA=KEEP
+CV1PGM    EQU   X'20'                  BRANCH=(YES,PGM)
+CV1MRCDS  EQU   X'10'                  MAPRCDS=YES
+CV1IRCDS  EQU   X'08'                  IXRCDS=KEEP
+CV1MAPIX  EQU   X'04'                  MAP=INDEX
+CV1MAPVT  EQU   X'02'                  MAP=VTOC
+CV1MAPVL  EQU   X'01'                  MAP=VOLUME
+CVFL2     DS    XL1                    SECOND FLAG BYTE
+CV2HIVIE  EQU   X'80'                  HIVIER=YES
+CV2VRF    EQU   X'40'                  VRF DATA EXISTS
+CV2CNT    EQU   X'20'                  COUNT=YES
+CV2RCVR   EQU   X'10'                  RECOVER=YES
+CV2SRCH   EQU   X'08'                  SEARCH=YES
+CV2DSNLY  EQU   X'04'                  DSNONLY=YES
+CV2VER    EQU   X'02'                  VERIFY=YES
+CV2NLEVL  EQU   X'01'                  OUTPUT-NEW HIGHEST LEVEL VIER
+*                                      CREATED
+          DS    H                      RESERVED
+CVUCB     DS    A                      UCB ADDRESS
+CVDSN     DS    A                      DATA SET NAME ADDRESS
+CVBUFL    DS    A                      BUFFER LIST ADDRESS
+CVIRCDS   DS    A                      INDEX VIR'S BUFFER LIST ADDRESS
+CVMRCDS   DS    A                      MAP VIR'S BUFFER LIST ADDRESS
+CVIOAR    DS    A                      I/O AREA ADDRESS
+CVDEB     DS    A                      DEB ADDRESS
+CVARG     DS    A                      ARGUMENT ADDRESS
+CVSPACE   DS    A                      SPACE PARAMETER LIST ADDRESS
+CVEXTS    DS    A                      EXTENT TABLE ADDRESS
+CVBUFL2   DS    A                      NEW VRF VIXM BUFFER LIST ADDR
+CVVRFDA   DS    A                      VRF DATA ADDRESS
+CVCTAR    DS    A                      COUNT AREA ADDRESS
+CVPLNGTH  EQU   *-CVPL

+*               VALUES OF CVSTAT
+*(THIS PART OF THE ICVAFPL MACRO EXPANSION IS NOT SHOWN)
          END
```

Example 2: Using the CVAFDIR Macro with an Indexed VTOC

This example uses the CVAFDIR macro to read one or more DSCBs on a VTOC. The UCB is supplied to the program in register 4 (labeled RUCB). The TTR of each DSCB read is to be returned to the caller. This program must be APF authorized.

The address of a parameter list is supplied to the program in register 5 (labeled RLIST). The parameter list contains one or more 3-word entries. The format of each 3-word entry is mapped by the LISTMAP DSECT. The first word contains the address of the data set name of the DSCB to be read. The second word contains the address of the 96-byte buffer into which the DSCB is to be read. The third word contains the address of the 3-byte TTR of the DSCB read.

The CVPL is generated by a list form of the CVAFDIR macro at label CVPL. The BUFLIST, IXRCDS, IOAREA, and BRANCH keywords are coded on the list form of the macro. IXRCDS=KEEP and IOAREA=KEEP are coded to avoid overhead if two or more DSCBs are to be read. BRANCH=(YES,PGM) is coded in the list form of the CVAFDIR macro to cause the CVPL to have the CV1PGM bit set to one; this will indicate to CVAF that the caller is authorized by APF and not in supervisor state. The execute forms of the CVAFDIR macro then specify BRANCH=YES, and not BRANCH=(YES,PGM), because the CV1PGM bit is set in the list form of the macro.

The CVAFDIR macro with ACCESS=RLSE is coded before the program exits in order to release the CVAF I/O area and the index records buffer list. BUFLIST=0 is coded because no user-supplied buffer list is to be released; BUFLIST was coded on the list form of the CVAFDIR macro and, therefore, is in the CVBUFL field of the CVPL. This field must be set to zero for the release.

```
EXAMPLE2 CSECT
         STM   14,12,12(13)
         BALR  12,0
         USING *,12
         ST    13,SAVEAREA+4
         LA    RWORK,SAVEAREA
         ST    RWORK,8(,13)
         LR    13,RWORK
***************************************************************
*
*        REGISTERS
*
***************************************************************
RWORK    EQU   3                     WORK REGISTER
RUCB     EQU   4                     UCB ADDRESS SUPPLIED BY CALLER
RLIST    EQU   5                     ADDRESS OF PARAMETER LIST
RDSN     EQU   6                     ADDRESS OF DATA SET NAME
RTTR     EQU   7                     ADDRESS OF TTR
REG15    EQU   15                    RETURN CODE REGISTER 15
```

```
*************************************************************
*
*        READ DSCB OF DATA SET NAME SUPPLIED. RETURN TTR OF DSCB.
*        UCB ADDRESS SUPPLIED IN RUCB.
*        ADDRESS OF PARAMETER LIST IN RLIST.
*          WORD 1 OF PARAMETER LIST = ADDRESS OF DATA SET NAME
*          WORD 2 OF PARAMETER LIST = ADDRESS OF DSCB TO BE RETURNED
*          WORD 3 OF PARAMETER LIST = ADDRESS OF TTR TO BE RETURNED
*                    WORDS 1-3 DUPLICATED WITH THE HIGH ORDER BIT OF
*                    WORD 3 SET TO ONE FOR LAST ENTRY.
*
*************************************************************
         USING LISTMAP,RLIST        ADDRESSABILITY TO PARMLIST
TOPLOOP  EQU   *                    LOOP FOR EACH DSCB
         XC    BUFLIST(BFLHLN+BFLELN),BUFLIST ZERO BUFFER LIST
         OI    BFLHFL,BFLHDSCB      DSCBS TO BE READ WITH BUFFER LIST
         MVI   BFLHNOE,1            ONE BUFFER LIST ENTRY
         LA    RWORK,LISTDSCB       ADDRESS OF DSCB BUFFER
         ST    RWORK,BFLEBUF        PLACE IN BUFFER LIST
         OI    BFLEFL,BFLETTR       TTR OF DSCB RETURNED BY CVAF
         MVI   BFLELTH,DSCBLTH      DATA PORTION OF DSCB READ - DSN      *
                                    SUPPLIED IN CVPL
         L     RDSN,LISTDSN         ADDRESS OF DATA SET NAME
         CVAFDIR DSN=(RDSN),UCB=(RUCB),MF=(E,CVPL),BRANCH=YES
+        LA    1,CVPL                            LOAD PARAMETER REG 1
+        ST    RUCB,12(,1)               STORE UCB PTR IN PARM LIST
+        ST    RDSN,16(,1)               STORE DSN PTR IN PARM LIST
+        L     15,16                     LOAD THE CVT
+        L     15,328(,15)               LOAD VS1/VS2 COMMON EXTENSION2
+        L     15,12(,15)                LOAD THE CVT CVAF TABLE
+        L     15,0(,15)                 LOAD THE CVAF ADDRESS
+        BALR  14,15                     BRANCH AND LINK TO CVAF
         L     RTTR,LISTTTR         ADDRESS OF TTR TO BE RETURNED
         USING TTRMAP,RTTR          MAP OF TTR
         LTR   REG15,REG15          ANY ERROR
         BZ    NOERROR              BRANCH IF NOT
         XC    TTR,TTR              ZERO TTR INDICATING NO DSCB
         B     RELOOP               GET NEXT ENTRY
NOERROR  EQU   *                    DSCB READ
         MVC   TTR,BFLEARG          RETURN TTR OF DSCB
RELOOP   EQU   *                    GET NEXT ENTRY
         TM    LASTLIST,LASTBIT     IS IT LAST ENTRY IN LIST?
         LA    RLIST,NEXTLIST       GET NEXT ENTRY
         BZ    TOPLOOP              PROCESS NEXT LIST
         CVAFDIR ACCESS=RLSE,       RELEASE CVAF OBTAINED AREAS          *
               IOAREA=NOKEEP,       RELEASE IOAREA                       *
               IXRCDS=NOKEEP,       RELEASE VIER BUFFER LIST             *
               BUFLIST=0,           NO USER BUFFER LIST SUPPLIED TO RLSE*
               BRANCH=YES,          BRANCH ENTER CVAF                    *
               MF=(E,CVPL)
+        LA    1,CVPL                            LOAD PARAMETER REG 1
+        MVI   6(1),X'03'                SET FUNCTION CODE
+        NI    8(1),B'10110111'          RESET CVAF FLAGS OFF
+        LA    15,0                      GET BUFLIST ADDRESS AND
+        ST    15,20(,1)                 STORE BUFLIST PTR IN PARM LIST
+        L     15,16                     LOAD THE CVT
+        L     15,328(,15)               LOAD VS1/VS2 COMMON EXTENSION2
+        L     15,12(,15)                LOAD THE CVT CVAF TABLE
+        L     15,0(,15)                 LOAD THE CVAF ADDRESS
+        BALR  14,15                     BRANCH AND LINK TO CVAF
         L     13,SAVEAREA+4
         RETURN (14,12)
```

```
+          LM     14,12,12(13)                        RESTORE THE REGISTERS
+          BR     14                                  RETURN

 BUFLIST  ICVAFBFL DSECT=NO         BUFFER LIST

+***********************************************************************
+*         BUFFER LIST HEADER
+***********************************************************************

+BUFLIST  DS     0F                      BUFFER LIST HEADER
+BFLHNOE  DS     XL1                     NUMBER OF ENTRIES
+BFLHFL   DS     XL1                     KEY AND FLAG BYTE
+         ORG    BFLHFL
+BFLHKEY  DS     XL1                     PROTECT KEY (FIRST 4 BITS)
+BFLHVIR  EQU    X'08'                   BUF. LIST ENTRIES DESCRIBE VIRS
+BFLHDSCB EQU    X'04'                   BUF. LIST ENTRIES DESCRIBE DSCBS
+         DS     XL1                     RESERVED
+BFLHSP   DS     XL1                     SUBPOOL OF BUF. LIST/BUFFERS
+BFLHFCHN DS     A                       FORWARD CHAIN PTR TO NEXT BUF.
+*                                       LIST
+BFLHLN   EQU    *-BUFLIST               LENGTH OF BUFFER LIST HEADER
+***********************************************************************
+*         BUFFER LIST ENTRY
+***********************************************************************

+BFLE     DS     0F                      BUFFER LIST ENTRY
+BFLEFL   DS     XL1                     BUFFER LIST ENTRY FLAG
+BFLERBA  EQU    X'80'                   ARGUMENT IS RBA
+BFLECHR  EQU    X'40'                   ARGUMENT IS CCHHR
+BFLETTR  EQU    X'20'                   ARGUMENT IS TTR
+BFLEAUPD EQU    X'10'                   CVAF UPDATED ARGUMENT FIELD
+BFLEMOD  EQU    X'08'                   DATA IN BUF. HAS BEEN MODIFIED
+BFLESKIP EQU    X'04'                   SKIP THIS ENTRY
+BFLEIOER EQU    X'02'                   I/O ERROR
+         DS     XL1                     RESERVED
+BFLELTH  DS     XL1                     LENGTH OF DSCB BUFFER OR
+*                                       LENGTH OF VIR DIVIDED BY 256
+BFLEARG  DS     XL5                     ARGUMENT OF VIR OR DSCB (CCHHR)
+         ORG    BFLEARG+1
+BFLEATTR DS     XL3                     'TTR' OF ARGUMENT
+         ORG    BFLEARG+1
+BFLEARBA DS     XL4                     'RBA'  OF ARGUMENT
+BFLEBUF  DS     A                       BUFFER ADDRESS
+BFLELN   EQU    *-BFLE                  LENGTH OF A BUFFER LIST ENTRY
 SAVEAREA DS     18F                 REGISTER SAVE AREA
 LISTMAP  DSECT
 LISTDSN  DS     F                   ADDRESS OF DATA SET NAME
 LISTDSCB DS     F                   ADDRESS OF BUFFER FOR DSCB TO BE      *
                                     RETURNED
 LISTTTR  DS     0F                  ADDRESS OF TTR OF DSCB TO BE          *
                                     RETURNED
 LASTLIST DS     X                   FIRST BYTE
 LASTBIT  EQU    X'80'               LAST ENTRY IN LIST
          DS     XL3                 REMAINDER OF TTR ADDRESS
 NEXTLIST EQU    *                   NEXT LIST
 DSCB     DSECT
          IECSDSL1 (1)
```

```
+IECSDSL1 EQU   *                           FORMAT 1 DSCB
+IECSDSF1 EQU   IECSDSL1
+DS1DSNAM DS    CL44                        DATA SET NAME
+DS1FMTID DS    CL1                         FORMAT IDENTIFIER
+DS1DSSN  DS    CL6                         DATA SET SERIAL NUMBER
+DS1VOLSQ DS    XL2                         VOLUME SEQUENCE NUMBER
+DS1CREDT DS    XL3                         CREATION DATE
+DS1EXPDT DS    XL3                         EXPIRATION DATE
+DS1NOEPV DS    XL1                         NUMBER OF EXTENTS ON VOLUME
+DS1NOBDB DS    XL1                         NUMBER OF BYTES USED IN LAST
+*                                              DIRECTORY BLOCK
+         DS    XL1                         RESERVED
+DS1SYSCD DS    CL13                        SYSTEM CODE
+         DS    XL7                         RESERVED
+DS1DSORG DS    XL2                         DATA SET ORGANIZATION
+*                     FIRST BYTE OF DS1DSORG
+DS1DSGIS EQU   X'80'                       IS - INDEXED SEQUENTIAL         @01A
+*                                           ORGANIZATION
+DS1DSGPS EQU   X'40'                       PS - PHYSICAL SEQUENTIAL        @01A
+*                                           ORGANIZATION
+DS1DSGDA EQU   X'20'                       DA - DIRECT ORGANIZATION        @01A
+DS1DSGCX EQU   X'10'                       CX - BTAM OR QTAM LINE GROUP    @01A
+*        EQU   X'08'                       RESERVED                        @01A
+*        EQU   X'04'                       RESERVED                        @01A
+DS1DSGPO EQU   X'02'                       PO - PARTITIONED ORGANIZATION   @01A
+DS1DSGU  EQU   X'01'                       U - UNMOVABLE, THE DATA         @01A
+*                                           CONTAINS LOCATION DEPENDENT
+*                                           INFORMATION
+*
+*                      SECOND BYTE OF DS1DSORG
+DS1DSGGS EQU   X'80'                       GS - GRAPHICS ORGANIZATION      @01A
+DS1DSGTX EQU   X'40'                       TX - TCAM LINE GROUP            @01A
+DS1DSGTQ EQU   X'20'                       TQ - TCAM MESSAGE QUEUE         @01A
+*        EQU   X'10'                       RESERVED                        @01A
+DS1ACBM  EQU   X'08'                       ACCESS METHOD CONTROL BLOCK     @01A
+DS1DSGTR EQU   X'04'                       TR - TCAM 3705                  @01A
+*        EQU   X'02'                       RESERVED                        @01A
+*        EQU   X'01'                       RESERVED                        @01A
+DS1RECFM DS    XL1                         RECORD FORMAT
+DS1OPTCD DS    XL1                         OPTION CODE
+DS1BLKL  DS    XL2                         BLOCK LENGTH
+DS1LRECL DS    XL2                         RECORD LENGTH
+DS1KEYL  DS    XL1                         KEY LENGTH
+DS1RKP   DS    XL2                         RELATIVE KEY POSITION
+DS1DSIND DS    XL1                         DATA SET INDICATORS
+DS1SCALO DS    XL4                         SECONDARY ALLOCATION
+DS1LSTAR DS    XL3                         LAST USED TRACK AND BLOCK ON TRACK
+DS1TRBAL DS    XL2                         BYTES REMAINING ON LAST TRACK USED
+         DS    XL2                         RESERVED
+DS1EXT1  DS    XL10                        FIRST EXTENT DESCRIPTION
+*        FIRST BYTE                        EXTENT TYPE INDICATOR
+*        SECOND BYTE                       EXTENT SEQUENCE NUMBER
+*        THIRD - SIXTH BYTES               LOWER LIMIT
+*        SEVENTH - TENTH BYTES             UPPER LIMIT
+DS1EXT2  DS    XL10                        SECOND EXTENT DESCRIPTION
+DS1EXT3  DS    XL10                        THIRD EXTENT DESCRIPTION
+DS1PTRDS DS    XL5                         POSSIBLE PTR TO A FORMAT 2 OR 3 DSCB
+DS1END   EQU   *
 DSCBLTH  EQU   *-DSCB-L'DS1DSNAM           LENGTH OF DATA PORTION OF DSCB
 TTRMAP   DSECT
 TTR      DS    XL3                         TTR TO BE RETURNED
 EXAMPLE2 CSECT
 CVPL     CVAFDIR ACCESS=READ,BUFLIST=BUFLIST,MF=L,                    *
                IOAREA=KEEP,        KEEP IOAREA TO AVOID OVERHEAD      *
                IXRCDS=KEEP      KEEP VIERS FOR 2ND AND SUBSEQUENT CALLS*
                                    CALLED IN PROGRAM STATE BUT APF    *
                                    AUTHORIZED SO UCB IS SUPPLIED
+         CNOP  0,4
+CVPL     EQU   *
+         DC    CL4'CVPL'                EBCDIC 'CVPL'
+         DC    AL2(ICV8E-CVPL)          LENGTH OF CVPL
+         DC    XL1'01'                  FUNCTION CODE
+         DC    XL1'00'                  STATUS INFORMATION
+         DC    B'01001000'              FIRST FLAG BYTE
+         DC    B'00000000'              SECOND FLAG BYTE
+         DC    H'0'                     RESERVED
+         DC    A(0)                     UCB ADDRESS
+         DC    A(0)                     DATA SET NAME ADDRESS
+         DC    A(BUFLIST)               BUFFER LIST ADDRESS
+         DC    A(0)                     INDEX VIR'S BUFFER LIST ADDRESS
+         DC    A(0)                     MAP VIR'S BUFFER LIST ADDRESS
+         DC    A(0)                     I/O AREA ADDRESS
+         DC    A(0)                     DEB ADDRESS
+         DC    A(0)                     ARGUMENT ADDRESS
+         DC    A(0)                     SPACE PARAMETER LIST ADDRESS
+         DC    A(0)                     EXTENT TABLE ADDRESS
+         DC    A(0)                     NEW VRF VIXM BUFFER LIST ADDR
+         DC    A(0)                     VRF DATA ADDRESS
+         DC    A(0)                     COUNT AREA ADDRESS
+ICV8E    EQU   *                        END OF CVPL
          ORG   CVPL                OVERLAY CVPL WITH EXPANSION OF MAP
 CVPLMAP  ICVAFPL DSECT=NO

+***********************************************************************
+*        CVAF PARAMETER LIST
+***********************************************************************
+CVPLMAP  DS    0F                      CVAF PARAMETER LIST
+CVLBL    DS    CL4                     EBCDIC 'CVPL'
+CVLTH    DS    H                       LENGTH OF CVPL
+CVFCTN   DS    XL1                     FUNCTION BYTE
+CVDIRD   EQU   X'01'                   CVAFDIR ACCESS=READ
+CVDIWR   EQU   X'02'                   CVAFDIR ACCESS=WRITE
+CVDIRLS  EQU   X'03'                   CVAFDIR ACCESS=RLSE
+CVSEQGT  EQU   X'04'                   CVAFSEQ ACCESS=GT
+CVSEQGTE EQU   X'05'                   CVAFSEQ ACCESS=GTEQ
+CVDMIXA  EQU   X'06'                   CVAFDSM ACCESS=IXADD
+CVDMIXD  EQU   X'07'                   CVAFDSM ACCESS=IXDLT
+CVDMALC  EQU   X'08'                   CVAFDSM ACCESS=ALLOC
+CVDMRLS  EQU   X'09'                   CVAFDSM ACCESS=RLSE
+CVDMMAP  EQU   X'0A'                   CVAFDSM ACCESS=MAPDATA
+CVVOL    EQU   X'0B'                   CVAFVOL ACCESS=VIBBLD
+CVVRFRD  EQU   X'0C'                   CVAFVRF ACCESS=READ
+CVVRFWR  EQU   X'0D'                   CVAFVRF ACCESS=WRITE
+CVSTAT   DS    XL1                     STATUS INFORMATION (SEE LIST    *
+                                       BELOW)
```

```
+CVFL1    DS    XL1                        FIRST FLAG BYTE
+CV1IVT   EQU   X'80'                      INDEXED VTOC ACCESSED
+CV1IOAR  EQU   X'40'                      IOAREA=KEEP
+CV1PGM   EQU   X'20'                      BRANCH=(YES,PGM)
+CV1MRCDS EQU   X'10'                      MAPRCDS=YES
+CV1IRCDS EQU   X'08'                      IXRCDS=KEEP
+CV1MAPIX EQU   X'04'                      MAP=INDEX
+CV1MAPVT EQU   X'02'                      MAP=VTOC
+CV1MAPVL EQU   X'01'                      MAP=VOLUME
+CVFL2    DS    XL1                        SECOND FLAG BYTE
+CV2HIVIE EQU   X'80'                      HIVIER=YES
+CV2VRF   EQU   X'40'                      VRF DATA EXISTS
+CV2CNT   EQU   X'20'                      COUNT=YES
+CV2RCVR  EQU   X'10'                      RECOVER=YES
+CV2SRCH  EQU   X'08'                      SEARCH=YES
+CV2DSNLY EQU   X'04'                      DSNONLY=YES
+CV2VER   EQU   X'02'                      VERIFY=YES
+CV2NLEVL EQU   X'01'                      OUTPUT-NEW HIGHEST LEVEL VIER
+*                                          CREATED
+         DS    H                          RESERVED
+CVUCB    DS    A                          UCB ADDRESS
+CVDSN    DS    A                          DATA SET NAME ADDRESS
+CVBUFL   DS    A                          BUFFER LIST ADDRESS
+CVIRCDS  DS    A                          INDEX VIR'S BUFFER LIST ADDRESS
+CVMRCDS  DS    A                          MAP VIR'S BUFFER LIST ADDRESS
+CVIOAR   DS    A                          I/O AREA ADDRESS
+CVDEB    DS    A                          DEB ADDRESS
+CVARG    DS    A                          ARGUMENT ADDRESS
+CVSPACE  DS    A                          SPACE PARAMETER LIST ADDRESS
+CVEXTS   DS    A                          EXTENT TABLE ADDRESS
+CVBUFL2  DS    A                          NEW VRF VIXM BUFFER LIST ADDR
+CVVRFDA  DS    A                          VRF DATA ADDRESS
+CVCTAR   DS    A                          COUNT AREA ADDRESS
+CVPLNGTH EQU   *-CVPLMAP

+*              VALUES OF CVSTAT
+*(THIS PART OF THE ICVAFPL MACRO EXPANSION IS NOT SHOWN)
          END
```

Example 3: Using the CVAFFILT Macro

This example uses the CVAFFILT macro to read all format-1, format-2, and format-3 DSCBS from a given VTOC, calculates the total number of DSCBs by format type, and returns the totals to the calling program (the caller of this example program, not the caller of CVAF). The address of a DEB opened to the VTOC is passed to the example program in register 1 (labeled R1 in the example).

The buffer list and filter criteria list are defined in the program. The ICVAFBFL macro generates the buffer list, and the ICVFCL macro generates the filter criteria list.

```
EXAMPLE3 CSECT
 *
 * INPUT ... REGISTER 1  - ADDRESS OF A DEB OPENED TO THE VTOC.
 *           REGISTER 13 - ADDRESS OF A STANDARD REGISTER SAVE AREA
 *           REGISTER 14 - ADDRESS OF THE RETURN POINT WITHIN CALLER
 *
 * PROCESS . USE THE CVAFFILT MACRO (ACCESS=READ, ACCESS=RESUME, AND
 *           ACCESS=RLSE) TO READ ALL FORMAT 1, 2, AND 3 DSCBS FROM A
 *           GIVEN VTOC. IF FILTER SERVICE DETECTS AN ERROR CONDITION,
 *           IT RETURNS DIAGNOSTIC INFORMATION FOR DEBUGGING ANALYSIS.
 *
 * OUTPUT .. REGISTER 1  = ADDRESS OF THE DATA RETURN AREA (SEE LABEL
 *                         RET$AREA AT THE END OF THIS LISTING).
 *           REGISTER 15 = ZERO IF NO ERRORS WERE ENCOUNTERED.
 *                         OTHERWISE, ERROR INFORMATION IS PROVIDED
 *                         IN THE DATA RETURN AREA (SEE LABEL RET$ERR).
 *
 * THE CVAF PARAMETER LIST (CVPL), FILTER CRITERIA LIST (FCL), BUFFER
 * LIST, AND DSCB BUFFERS ARE DEFINED WITHIN THIS CSECT.
 *
 ********************************************************************
 *        EQUATES FOR ASSEMBLY CONSTANTS AND REGISTERS
 ********************************************************************
 BFLE$N   EQU   11  NUMBER OF BUFFER LIST ELEMENTS AND BUFFERS DESIRED
 R0       EQU   0
 R1       EQU   1
 R2       EQU   2
 R3       EQU   3
 R4       EQU   4
 R5       EQU   5
 R11      EQU   11
 R12      EQU   12
 R13      EQU   13
 R14      EQU   14
 R15      EQU   15
 *
 ********************************************************************
 *    SAVE CALLER'S REGISTERS AND ESTABLISH A NEW REGISTER SAVE AREA
 ********************************************************************
          STM   R14,R12,12(R13)       SAVE CALLER'S REGISTERS
          BALR  R12,0                 ESTABLISH THIS PROGRAM'S
          USING *,R12                    BASE REGISTER
          ST    R13,SAVEAREA+4        SAVE ADDRESS OF CALLER'S SAVE AREA
          LA    R15,SAVEAREA          GET ADDRESS OF THE NEW SAVE AREA
          ST    R15,8(,R13)           CHAIN CALLER'S AREA TO OURS
          LR    R13,R15               ESTABLISH THE NEW SAVE AREA
 *


 ********************************************************************
```

```
* ESTABLISH ADDRESSABILITY TO THE CVPL. PLACE GIVEN DEB ADDRESS
* IN THE CVPL, INITIALIZE THE FLAG BYTE AND THE RETURN DATA AREA.
*********************************************************************
         LA    R11,CVPL$DEF            ESTABLISH ADDRESSABILITY
         USING CVPL$MAP,R11               TO THE CVPL
         ST    R1,CVDEB                PLACE GIVEN ADDR(DEB) IN CVPL
         MVI   FLAGS,F$RSET            RESET THE LOCAL FLAG BYTE
         XC    RET$AREA,RET$AREA       INIT. DATA RETURN AREA TO ZERO
*
*********************************************************************
* INITIALIZE THE BUFFER LIST HEADER (BFLH) AND ELEMENTS (BFLE)
*********************************************************************
         XC    BFLH$DEF(BFL$SIZE),BFLH$DEF  SET BUFR LIST AREA TO ZERO
         LA    R1,BFLH$DEF             R1 -> BUFFER LIST HEADER
         USING BFL$MAP,R1              ESTABLISH ADDRESSABILITY
         MVI   BFLHNOE,BFLE$N          SET NUMBER OF BUFFER ELEMENTS
         OI    BFLHFL,BFLHDSCB         IDENTIFY AS DSCB BUFR ELEMNT LIST
         LA    R2,BFLH$DEF+BFLHLN      R2 -> FIRST BUFFER LIST ELEMENT
         USING BFLE,R2                 ESTABLISH ADDRESSABILITY
         LA    R3,DSCB$DEF             R3 -> FIRST DSCB BUFFER
         LA    R4,BFLE$N               R4 = NUMBER OF ELEMENTS AND BUFRS
BFLE$INT OI    BFLEFL,BFLECHR          REQUEST CCHHR ON RETURN
         MVI   BFLELTH,DSCB$SIZ        SET BUFR LNGTH TO FULL DSCB SIZE
         ST    R3,BFLEBUF              SET ADDR(DSCB BUFFER)
         LA    R2,BFLELN(R2)           R2 -> NEXT BUFFER LIST ELEMENT
         LA    R3,DSCB$SIZ(R3)         R3 -> NEXT DSCB BUFFER
         BCT   R4,BFLE$INT             LOOP THROUGH ALL ELEMENTS
         DROP  R1,R2                   DROP TEMP USING
*
*********************************************************************
* INITIALIZE THE FILTER CRITERIA LIST (FCL) HEADER AND ELEMENT
*********************************************************************
         XC    FCL$DEF(FCL$SIZE),FCL$DEF  SET FCL AREA TO ZERO
         LA    R1,FCL$DEF              R1 -> FCL HEADER
         USING FCL$MAP,R1              ESTABLISH ADDRESSABILITY
         MVC   FCLID,FCL$ID            SET THE EYECATCHER 'FCL '
         MVC   FCLCOUNT,=H'1'          SET NUMBER OF FCL ELEMENTS
         LA    R2,FCLHDEND             R2 -> FIRST (ONLY) FCL ELEMENT
         USING FCLDSN,R2               ESTABLISH ADDRESSABILITY
         MVI   FCLDSNLG,X'02'          SET LENGTH(DSN PATTERN)
         LA    R3,=C'**'               R3 -> C'**'
         ST    R3,FCLDSNA              SET ADDR(DSN PATTERN)
         DROP  R1,R2                   DROP TEMP USING
*
*********************************************************************
* ISSUE CVAFFILT ACCESS=READ REQUEST
*********************************************************************
         MVI  RET$FTN,RET$READ         IDENTIFY THE CURRENT FUNCTION
         CVAFFILT ACCESS=READ,FCL=FCL$DEF,BUFLIST=BFLH$DEF,            X
               MF=(E,CVPL$DEF)
+        LA    1,CVPL$DEF              LOAD PARAMETER REG 1
+        MVI   6(1),X'0E'              SET FUNCTION CODE
+        LA    15,FCL$DEF              GET FCL ADDRESS AND
+        ST    15,16(,1)               STORE FCL PTR IN PARM LIST
+        LA    15,BFLH$DEF             GET BUFLIST ADDRESS AND
+        ST    15,20(,1)               STORE BUFLIST PTR IN PARM LIST
+        SVC   139
*
*********************************************************************
* TEST THE RETURN FROM CVAFFILT ACCESS=READ OR ACCESS=RESUME
*********************************************************************
TEST$RET LTR   R15,R15                 IF DSCB RETURN IS COMPLETE
         BZ    COUNTLST                   GO COUNT LAST SET OF DSCBS
         CH    R15,=H'4'               IF RETURN CODE OTHER THAN FOUR
         BNE   ERR$RET                    GO RETURN THE ERROR CONDITION
         CLI   CVSTAT,STAT064          IF OTHER THAN RESUME RECOMMENDED
         BNE   ERR$RET                   GO RETURN THE ERROR CONDITION
         B     COUNTCUR               ELSE GO COUNT CURRENT DSCB SET
*
***********************************************************************
* COUNT THE NUMBER OF FORMAT 1, 2, AND 3 DSCBS RETURNED
***********************************************************************
COUNTLST OI    FLAGS,F$LAST           INDICATE LAST SET OF DSCBS
COUNTCUR LM    R1,R3,RET$F1           GET PRIOR DSCB COUNTS
         LA    R4,FCL$DEF             R4 -> FCL HEADER
         USING FCL$MAP,R4             ESTABLISH ADDRESSABILITY
         SLR   R5,R5                  R5 = ZERO FOR FOLLOWING ICM
         ICM   R5,B'0011',FCLDSCBR    R5 = NUMBER OF DSCBS RETURNED
         BZ    TST$RSUM               IF ZERO, GO TEST FOR RESUME
         LA    R4,DSCB$DEF            R4 -> FIRST DSCB BUFFER
         USING DSCB$MAP,R4            ESTABLISH ADDRESSABILITY
COUNTNXT CLI   DS1FMTID,X'F1'         IF FORMAT 1 DSCB
         BE    COUNT$F1                  GO INCREMENT ITS COUNTER
         CLI   DS1FMTID,X'F2'         IF FORMAT 2 DSCB
         BE    COUNT$F2                  GO INCREMENT ITS COUNTER
         LA    R3,1(R3)               ELSE INCREMENT FORMAT 3 DSCB COUNT
         B     COUNTTST               GO TEST FOR MORE DSCBS
*
COUNT$F2 LA    R2,1(R2)               INCREMENT COUNT OF FORMAT 2 DSCBS
         B     COUNTTST               GO TEST FOR MORE DSCBS
*
COUNT$F1 LA    R1,1(R1)               INCREMENT COUNT OF FORMAT 1 DSCBS
COUNTTST LA    R4,DSCB$SIZ(R4)        R4 -> NEXT DSCB BUFFER
         BCT   R5,COUNTNXT            LOOP THROUGH ALL RETURNED DSCBS
         STM   R1,R3,RET$F1           SAVE UPDATED DSCB COUNTS
         DROP  R4                     FINISHED COUNTING CURR DSCB SET
*
***********************************************************************
* CONDITIONALLY ISSUE CVAFFILT ACCESS=RESUME REQUEST
***********************************************************************
TST$RSUM TM    FLAGS,F$LAST           IF LAST SET OF DSCBS COUNTED
         BO    RLSE$REQ                  GO REQUEST A RELEASE
         MVI   RET$FTN,RET$RSUM       IDENTIFY THE CURRENT FUNCTION
         CVAFFILT ACCESS=RESUME,MF=(E,CVPL$DEF)  RESUME REQUEST
+        LA    1,CVPL$DEF             LOAD PARAMETER REG 1
+        MVI   6(1),X'0F'             SET FUNCTION CODE
+        SVC   139
         B     TEST$RET               GO TEST THE RETURN CODES
*
***********************************************************************
* ISSUE CVAFFILT ACCESS=RLSE REQUEST
***********************************************************************
RLSE$REQ MVI   RET$FTN,RET$RLSE       IDENTIFY THE CURRENT FUNCTION
         CVAFFILT ACCESS=RLSE,FLC=0,BUFLIST=0,FLTAREA=NOKEEP,          X
               MF=(E,CVPL$DEF)
+        LA    1,CVPL$DEF             LOAD PARAMETER REG 1
+        MVI   6(1),X'10'             SET FUNCTION CODE
+        NI    10(1),B'01111111'      RESET CVAF FLAGS OFF
+        LA    15,0                   GET FCL ADDRESS AND
+        ST    15,16(,1)                 STORE FCL PTR IN PARM LIST
+        LA    15,0                   GET BUFLIST ADDRESS AND
+        ST    15,20(,1)                 STORE BUFLIST PTR IN PARM LIST
+        SVC   139
         LTR   R15,R15                IF NO ERROR ON RELEASE REQUEST
         BZ    RETURN                    GO RETURN TO CALLER
         LA    R1,RET$RLSE            ELSE INDICATE ERROR IN RELEASE
*
***********************************************************************
* PLACE ERROR INFORMATION IN THE OUTPUT DATA RETURN AREA
***********************************************************************
ERR$RET  MVC   RET$STAT,CVSTAT        COPY CVSTAT TO RETURN AREA
         ST    R15,RET$RC             COPY CVAFFILT'S RETURN CODE
```

```
*
*********************************************************************
* ASSUME R15 = DESIRED RETURN CODE, SET R1 -> DATA RETURN AREA,
* RESTORE CALLER'S REGISTERS 0, 2-14, AND RETURN TO CALLER
*********************************************************************
RETURN   LA    R1,RET$AREA           R1 -> OUTPUT DATA RETURN AREA
         L     R13,SAVEAREA+4        R13 -> CALLER'S REG SAVE AREA
         L     R14,12(R13)           RESTORE CALLER'S REGISTER 14
         L     R0,20(R13)            RESTORE CALLER'S REGISTER 0
         LM    R2,R12,28(R13)        RESTORE CALLER'S REGISTERS 2-12
         BR    R14                   RETURN TO CALLER
FLAGS    DC    X'00'                 LOCAL FLAG BYTE
F$RSET   EQU   X'00'                 FLAG RESET VALUE
F$LAST   EQU   X'80'                 LAST DSCB SET ENCOUNTERED
*
         LTORG
               =H'1'
               =C'**'
               =H'4'
*
SAVEAREA DC    18F'0'                REGISTER SAVE AREA
*
*********************************************************************
* MAPPING MACROS
*********************************************************************
CVPL$MAP ICVAFPL CVPLFSA=YES
+*********************************************************************
+*        CVAF PARAMETER LIST
+*********************************************************************
+CVPL$MAP DSECT                       CVAF PARAMETER LIST
+         DS    0F
+CVLBL    DS    CL4                     EBCDIC 'CVPL'
+CVLTH    DS    H                       LENGTH OF CVPL
+CVFCTN   DS    XL1                     FUNCTION BYTE
+CVDIRD   EQU   X'01'                   CVAFDIR ACCESS=READ
+CVDIWR   EQU   X'02'                   CVAFDIR ACCESS=WRITE
+CVDIRLS  EQU   X'03'                   CVAFDIR ACCESS=RLSE
+CVSEQGT  EQU   X'04'                   CVAFSEQ ACCESS=GT
+CVSEQGTE EQU   X'05'                   CVAFSEQ ACCESS=GTEQ
+CVDMIXA  EQU   X'06'                   CVAFDSM ACCESS=IXADD
+CVDMIXD  EQU   X'07'                   CVAFDSM ACCESS=IXDLT
+CVDMALC  EQU   X'08'                   CVAFDSM ACCESS=ALLOC
+CVDMRLS  EQU   X'09'                   CVAFDSM ACCESS=RLSE
+CVDMMAP  EQU   X'0A'                   CVAFDSM ACCESS=MAPDATA
+CVVOL    EQU   X'0B'                   CVAFVOL ACCESS=VIBBLD
+CVVRFRD  EQU   X'0C'                   CVAFVRF ACCESS=READ
+CVVRFWR  EQU   X'0D'                   CVAFVRF ACCESS=WRITE
+CVFIRD   EQU   X'0E'                   CVAFFILT ACCESS=READ
+CVFIRES  EQU   X'0F'                   CVAFFILT ACCESS=RESUME
+CVFIRLS  EQU   X'10'                   CVAFFILT ACCESS=RLSE
+CVSTAT   DS    XL1                     STATUS INFORMATION (SEE LIST    X
+                                       BELOW)
+CVFL1    DS    XL1                     FIRST FLAG BYTE
+CV1IVT   EQU   X'80'                   INDEXED VTOC ACCESSED
+CV1IOAR  EQU   X'40'                   IOAREA=KEEP
+CV1PGM   EQU   X'20'                   BRANCH=(YES,PGM)
+CV1MRCDS EQU   X'10'                   MAPRCDS=YES
+CV1IRCDS EQU   X'08'                   IXRCDS=KEEP
+CV1MAPIX EQU   X'04'                   MAP=INDEX
+CV1MAPVT EQU   X'02'                   MAP=VTOC
+CV1MAPVL EQU   X'01'                   MAP=VOLUME
+CVFL2    DS    XL1                     SECOND FLAG BYTE
+CV2HIVIE EQU   X'80'                   HIVIER=YES
+CV2VRF   EQU   X'40'                   VRF DATA EXISTS
+CV2CNT   EQU   X'20'                   COUNT=YES
+CV2RCVR  EQU   X'10'                   RECOVER=YES
```

```
+CV2SRCH  EQU    X'08'                       SEARCH=YES
+CV2DSNLY EQU    X'04'                       DSNONLY=YES
+CV2VER   EQU    X'02'                       VERIFY=YES
+CV2NLEVL EQU    X'01'                       OUTPUT-NEW HIGHEST LEVEL VIER
+*                                           CREATED
+CVFL3    DS     XL1                         THIRD FLAG BYTE
+CV3FILT  EQU    X'80'                       FLTAREA=KEEP
+CV3IXERR EQU    X'40'                       INDEX ERROR FOUND
+         DS     XL1                         RESERVED
+CVUCB    DS     A                           UCB ADDRESS
+CVDSN    DS     A                           DATA SET NAME ADDRESS
+         ORG    CVDSN
+CVFCL    DS     A                           FILTER CRITERIA LIST ADDRESS
+CVBUFL   DS     A                           BUFFER LIST ADDRESS
+CVIRCDS  DS     A                           INDEX VIR'S BUFFER LIST ADDRESS
+CVMRCDS  DS     A                           MAP VIR'S BUFFER LIST ADDRESS
+CVIOAR   DS     A                           I/O AREA ADDRESS
+CVDEB    DS     A                           DEB ADDRESS
+CVARG    DS     A                           ARGUMENT ADDRESS
+CVSPACE  DS     A                           SPACE PARAMETER LIST ADDRESS
+CVEXTS   DS     A                           EXTENT TABLE ADDRESS
+CVBUFL2  DS     A                           NEW VRF VIXM BUFFER LIST ADDR
+CVVRFDA  DS     A                           VRF DATA ADDRESS
+CVCTAR   DS     A                           COUNT AREA ADDRESS
+CVFSA    DS     A                           FILTER SAVE AREA ADDRESS
+CVPLNGTH EQU    *-CVPL$MAP

+*               VALUES OF CVSTAT
+*(THIS PART OF THE ICVAFPL MACRO EXPANSION IS NOT SHOWN)

 FCL$MAP  ICVFCL
+*********************************************************************
+*        FILTER CRITERIA LIST
+*********************************************************************
+FCL$MAP  DSECT                              CVAF FILTER CRITERIA LIST
+FCLID    DS     CL4                         EYE CATCHER 'FCL '
+FCLCOUNT DS     H                           NUMBER OF DSN'S IN LIST
+FCLDSCBR DS     H                           NUMBER OF DSCB'S RETURNED
+FCL1FLAG DS     X                           REQUEST FLAG BYTE
+FCL1LIST EQU    X'80'                       LIST CONTAINS FULLY QUALIFIED
+*                                           DATA SET NAMES
+FCL1ORDR EQU    X'40'                       CALLER REQUESTS FCL ORDER
+FCL2FLAG DS     X                           STATUS FLAG BYTE
+FCL2SEQ  EQU    X'80'                       SEQUENTIAL VTOC ACCESS SELECTED
+FCL2SDIR EQU    X'40'                       SEQUENTIAL VTOC ACCESS REQUIRED
+*                                              AT LEAST ONE DIRECT DSCB READ
+FCLDRSV  DS     XL6                         RESERVED
+FCLHDEND EQU    *                           DSN ENTRIES DIRECTLY FOLLOW
+FCLHDLEN EQU    *-FCLID                     LENGTH OF FCL HEADER
+*
+         DSECT                              MAP OF EACH DSN ELEMENT
+FCLDSN   CL8                                DATA SET NAME INFORMATION
+         ORG    FCLDSN
+FCLDSNST DS     X                           STATUS OF THIS DATA SET NAME
+FCLST00  EQU    X'00'                       DSN REMAINS TO BE PROCESSED
+FCLST01  EQU    X'01'                       DSCB(S) HAVE BEEN RETURNED
+* FOR FCLST02 THROUGH FCLST05 -- CVAFFILT WILL NOT RETURN ANY
+* DSCB(S) FOR THE ASSOCIATED DSN ------------------
+FCLST02  EQU    X'02'                       DSN NOT FOUND
+FCLST03  EQU    X'03'                       ANOMALY FOUND IN DSCB(S)
+FCLST04  EQU    X'04'                       ANOMALY FOUND IN CVAFFILT
+FCLST05  EQU    X'05'                       NUMBER OF DSCBS FOR THIS DSN
+*                                              IS GREATER THAN THE TOTAL
+*                                              NUMBER OF BFLE PROVIDED
+*
+FCLDSNLG DS     X                           DATA SET NAME LENGTH
```

```
+FCL3FLAG DS     X                          FLAG BYTE
+FCL3UPDT EQU    X'80'                      THIS ELEMENT WAS PROCESSED
+*                                             BY THIS INVOCATION
+FCLDSNRV DS     X                          RESERVED
+FCLDSNA  DS     F                          DATA SET NAME ADDRESS
+FCLDSNEL EQU    *-FCLDSN                   LENGTH OF DSN INFO ENTRY

 BFL$MAP  ICVAFBFL
+*******************************************************************
+*        BUFFER LIST HEADER
+*******************************************************************
+BFL$MAP  DSECT                             BUFFER LIST HEADER
+         DS     0F
+BFLHNOE  DS     XL1                        NUMBER OF ENTRIES
+BFLHFL   DS     XL1                        KEY AND FLAG BYTE
+         ORG    BFLHFL
+BFLHKEY  DS     XL1                        PROTECT KEY (FIRST 4 BITS)
+BFLHVIR  EQU    X'08'                      BUF. LIST ENTRIES DESCRIBE VIRS
+BFLHDSCB EQU    X'04'                      BUF. LIST ENTRIES DESCRIBE DSCBS
+         DS     XL1                        RESERVED
+BFLHSP   DS     XL1                        SUBPOOL OF BUF. LIST/BUFFERS
+BFLHFCHN DS     A                          FORWARD CHAIN PTR TO NEXT BUF.
+*                                          LIST
+BFLHLN   EQU    *-BFL$MAP                  LENGTH OF BUFFER LIST HEADER
+**********************************************************************
+*        BUFFER LIST ENTRY
+**********************************************************************
+BFLE     DSECT                             BUFFER LIST ENTRY
+         DS     0F
+BFLEFL   DS     XL1                        BUFFER LIST ENTRY FLAG
+BFLERBA  EQU    X'80'                      ARGUMENT IS RBA
+BFLECHR  EQU    X'40'                      ARGUMENT IS CCHHR
+BFLETTR  EQU    X'20'                      ARGUMENT IS TTR
+BFLEAUPD EQU    X'10'                      CVAF UPDATED ARGUMENT FIELD
+BFLEMOD  EQU    X'08'                      DATA IN BUF. HAS BEEN MODIFIED
+BFLESKIP EQU    X'04'                      SKIP THIS ENTRY
+BFLEIOER EQU    X'02'                      I/O ERROR
+         DS     XL1                        RESERVED
+BFLELTH  DS     XL1                        LENGTH OF DSCB BUFFER OR
+*                                          LENGTH OF VIR DIVIDED BY 256
+BFLEARG  DS     XL5                        ARGUMENT OF VIR OR DSCB (CCHHR)
+         ORG    BFLEARG+1
+BFLEATTR DS     XL3                        'TTR' OF ARGUMENT
+         ORG    BFLEARG+1
+BFLEARBA DS     XL4                        'RBA'  OF ARGUMENT
+BFLEBUF  DS     A                          BUFFER ADDRESS
+BFLELN   EQU    *-BFLE                     LENGTH OF A BUFFER LIST ENTRY

          PUSH PRINT
          PRINT NOGEN
 DSCB$MAP DSECT
          IECSDSL1 (1)     USE FORMAT 1 DSCB MAPPING TO GET BUFFER SIZE
 DSCB$SIZ EQU    *-IECSDSL1     LENGTH OF FULL DSCB
          POP PRINT

 **********************************************************************
 *        SPACE ALLOCATION FOR CVPL, FCL, BFL, AND DSCB BUFFERS
 **********************************************************************
 EXAMPLE3 CSECT , CONTINUATION OF CSECT
 CVPL$DEF CVAFFILT MF=L,BRANCH=NO,FLTAREA=KEEP
+         CNOP   0,4
+CVPL$DEF EQU    *
+         DC     CL4'CVPL'                  EBCDIC 'CVPL'
+         DC     AL2(ICV11E-CVPL$DEF)       LENGTH OF CVPL
+         DC     XL1'0E'                    FUNCTION CODE
+         DC     XL1'00'                    STATUS INFORMATION
+         DC    B'00000000'            FIRST FLAG BYTE
+         DC    B'00000000'            SECOND FLAG BYTE
+         DC    B'10000000'            THIRD FLAG BYTE
+         DC    X'0'                   RESERVED
+         DC    A(0)                   UCB ADDRESS
+         DC    A(0)                   FILTER CRITERIA LIST ADDRESS
+         DC    A(0)                   BUFFER LIST ADDRESS
+         DC    A(0)                   INDEX VIR'S BUFFER LIST ADDRESS
+         DC    A(0)                   MAP VIR'S BUFFER LIST ADDRESS
+         DC    A(0)                   I/O AREA ADDRESS
+         DC    A(0)                   DEB ADDRESS
+         DC    A(0)                   ARGUMENT ADDRESS
+         DC    A(0)                   SPACE PARAMETER LIST ADDRESS
+         DC    A(0)                   EXTENT TABLE ADDRESS
+         DC    A(0)                   NEW VRF VIXM BUFFER LIST ADDRESS
+         DC    A(0)                   VRF DATA ADDRESS
+         DC    A(0)                   COUNT AREA ADDRESS
+         DC    A(0)                   FILTER SAVE AREA ADDRESS
+ICV11E   EQU   *                      END OF CVPL

 FCL$ID   DC    CL4'FCL '
 FCL$DEF  DS    (FCLHDLEN+FCLDSNEL)X   FCL HEADER AND ONE FCL ELEMENT
 FCL$SIZE EQU   *-FCL$DEF
 * DEFINE A CVAF BUFFER LIST WITH N BUFFER LIST ELEMENTS
 BFLH$DEF DS    (BFLHLN)X          BUFFER LIST HEADER
 BFLE$DEF DS    (BFLE$N*BFLELN)X   N BUFFER LIST ELEMENTS
 BFL$SIZE EQU   *-BFLH$DEF
 * DEFINE N FULL DSCB BUFFERS
 DSCB$DEF DS    (BFLE$N*DSCB$SIZ)X

 ***********************************************************************
 *        OUTPUT DATA RETURN AREA
 ***********************************************************************
          DS    0F
 RET$AREA DS    0XL20     OUTPUT DATA RETURN AREA
 RET$F1   DC    F'0'      COUNT OF FORMAT 1 DSCBS RETURNED BY CVAFFILT
 RET$F2   DC    F'0'      COUNT OF FORMAT 2 DSCBS RETURNED BY CVAFFILT
 RET$F3   DC    F'0'      COUNT OF FORMAT 3 DSCBS RETURNED BY CVAFFILT
 *
 RET$ERR  DS    0XL8      IF NO ERROR CONDITION WAS ENCOUNTERED THEN
 *                           REGISTER 15 = ZERO ON RETURN
 *                        OTHERWISE THE FOLLOWING INFORMATION IS PROVIDED
 RET$FTN  DC    X'00'     IDENTIFICATION OF CVAFFILT SUBFUNCTION
 RET$READ EQU   X'01'     CVAFFILT ACCESS=READ
 RET$RSUM EQU   X'02'     CVAFFILT ACCESS=RESUME
 RET$RLSE EQU   X'03'     CVAFFILT ACCESS=RLSE
          DC    XL2'00'   UNUSED
 RET$STAT DC    X'00'     COPY OF CVSTAT FROM CVAFFILT
 RET$RC   DC    F'0'      COPY OF THE RETURN CODE FROM CVAFFILT

          END
```

Example 4: Using the CVAFSEQ Macro with an Indexed VTOC

This example uses the CVAFSEQ macro to count the number of ISAM data sets whose data set names are within the range defined by two supplied data set names. The addresses of the two data set names are supplied to the program in registers 6 and 7, labeled RDSN1 and RDSN2, respectively. The address of a DEB open to the VTOC is supplied in register 4, labeled RDEB.

The CVAF parameter list is expanded by a list form of the CVAFSEQ macro. ACCESS=GTEQ is specified on the list form of the macro and is, therefore, not coded in the first execution of the CVPL. Subsequent executions of the CVPL (at label RELOOP) specify ACCESS=GT.

End of data is tested by comparing the CVSTAT field to the value of STAT032, which is an equate in the ICVAFPL mapping macro.

The count of ISAM DSCBs matching the data set name criterion is returned in register 15, unless an error is encountered, in which case a negative 1 is returned in register 15.

```
EXAMPLE4 CSECT
         STM   14,12,12(13)
         BALR  12,0
         USING *,12
         ST    13,SAVEAREA+4
         LA    RWORK,SAVEAREA
         ST    RWORK,8(,13)
         LR    13,RWORK
************************************************************
*
*        REGISTERS
*
************************************************************
REG1     EQU   1                  REGISTER 1
RWORK    EQU   3                  WORK REGISTER
RDEB     EQU   4                  DEB ADDRESS
RDSN1    EQU   6                  ADDRESS OF DATA SET NAME 1
RDSN2    EQU   7                  ADDRESS OF DATA SET NAME 2
REG15    EQU   15                 RETURN CODE REGISTER 15
************************************************************
*
*        COUNT THE NUMBER OF ISAM DATA SETS WHOSE DATA SET NAMES ARE
*           BETWEEN DSN1 AND DSN2 INCLUSIVELY.
*           RDSN1 CONTAINS ADDRESS OF DSN1.
*           RDSN2 CONTAINS ADDRESS OF DSN2.
*        ADDRESS OF DEB OPEN TO VTOC SUPPLIED IN RDEB.
*
************************************************************
         XC   BUFLIST(BFLHLN+BFLELN),BUFLIST ZERO BUFFER LIST
         OI   BFLHFL,BFLHDSCB      DSCBS TO BE READ WITH BUFFER LIST
         MVI  BFLHNOE,1            ONE BUFFER LIST ENTRY
         LA   RWORK,DS1FMTID       ADDRESS OF DSCB BUFFER
         ST   RWORK,BFLEBUF        PLACE IN BUFFER LIST
         MVI  BFLELTH,DSCBLTH      DATA PORTION OF DSCB READ - DSN      *
                                   SUPPLIED IN CVPL
         MVC  DS1DSNAM,0(RDSN1)    MOVE IN STARTING DATA SET NAME TO    *
                                   WORKAREA
         XR   RWORK,RWORK          ZERO COUNT
         CVAFSEQ DEB=(RDEB),       FIND FIRST DATA SET WHOSE DATA SET   *
               BUFLIST=BUFLIST,    NAME IS GREATER THAN OR EQUAL TO     *
               MF=(E,CVPL)         THAT OF DSN1
+        LA    1,CVPL                          LOAD PARAMETER REG 1
+        ST    RDEB,36(,1)             STORE DEB PTR IN PARM LIST
+        SVC   139
LOOP     EQU  *                    LOOP UNTIL END OF DATA OR DATA SET   *
                                   NAME GREATER THAN DSN2
         USING CVPL,REG1           ADDRESSABILITY TO CVPL
         LTR  REG15,REG15          ANY ERROR
         BZ   TESTDSN              BRANCH IF NOT-CHECK DSN LIMIT
************************************************************
*
*        DETERMINE WHAT ERROR IS
*
************************************************************
         C    REG15,ERROR4         IS RETURN CODE 4
         BNE  OTHERERR             BRANCH IF NOT 4
         CLI  CVSTAT,STAT032       IS IT END OF DATA?
         BNE  OTHERERR             BRANCH IF NOT
         DROP REG1                 ADDRESSABILITY TO CVPL NOT NEEDED
************************************************************
*
*        END OF DATA
*
************************************************************
```

```
         B    RELEASE              RELEASE CVAF RESOURCES AND RETURN
TESTDSN  EQU  *                    IS DATA SET NAME GREATER THAN DSN2
         CLI  DS1FMTID,C'1'        IS THIS A FORMAT 1 DSCB?
         BNE  CKLAST               BRANCH IF NO. CAN NOT BE ISAM.
         CLC  DS1DSNAM,0(RDSN2)    HAS LIMIT BEEN REACHED?
         BNH  TESTIS               BRANCH IF NO-TEST FOR ISAM
         B    RELEASE              RELEASE CVAF RESOURCES AND RETURN
TESTIS   EQU  *                    ONLY COUNT ISAM
         TM   DS1DSORG,DS1DSGIS    IS DATA SET ISAM
         BZ   CKLAST               BRANCH IF NO-DO NOT COUNT IT
         LA   RWORK,1(,RWORK)      INCREMENT COUNT BY ONE
CKLAST   EQU  *                    CHECK IF LAST DATA SET NAME (DSN2)
         CLC  DS1DSNAM,0(RDSN2)    HAS LIMIT BEEN REACHED?
         BNH  RELOOP               BRANCH IF NO-READ NEXT ONE
         B    RELEASE              RELEASE CVAF RESOURCES AND RETURN
RELOOP   EQU  *                    READ NEXT DSCB
         CVAFSEQ ACCESS=GT,MF=(E,CVPL) GET DSCB WITH DATA SET NAME     *
                                   GREATER THAN THE ONE LAST READ
+        LA    1,CVPL                             LOAD PARAMETER REG 1
+        MVI   6(1),X'04'               SET FUNCTION CODE
+        SVC   139
         B    LOOP                 CHECK RESULTS OF CVAFSEQ
OTHERERR EQU  *                    UNEXPECTED ERROR
*************************************************************
*
*        UNEXPECTED ERROR PROCESSING
*
*************************************************************
         LA   RWORK,1(0,0)         ONE IN RWORK
         LNR  RWORK,RWORK          SET NEGATIVE COUNT INDICATING ERROR
RELEASE  CVAFDIR ACCESS=RLSE,      RELEASE CVAF BUFFERS/IOAREA         *
               BUFLIST=0,          DO NOT RELEASE USER BUFFER LIST     *
               IXRCDS=NOKEEP,      RELEASE CVAF VIER BUFFERS           *
               MF=(E,CVPL)         RELEASE CVAF I/O AREA
+RELEASE  EQU   *
+        LA    1,CVPL                             LOAD PARAMETER REG 1
+        MVI   6(1),X'03'               SET FUNCTION CODE
+        NI    8(1),B'11110111'         RESET CVAF FLAGS OFF
+        LA    15,0                     GET BUFLIST ADDRESS AND
+        ST    15,20(,1)                STORE BUFLIST PTR IN PARM LIST
+        SVC   139
         LR   REG15,RWORK          CURRENT COUNT IS RETURN CODE
         L    13,SAVEAREA+4
         RETURN (14,12),RC=(15)    RETURN CURRENT COUNT
+        L     14,12(13,0)                        RESTORE REGISTER 14
+        LM    0,12,20(13)                        RESTORE THE REGISTERS
+        BR    14                                 RETURN
ERROR4   DC   F'4'                 ERROR RETURN CODE 4
BUFLIST  ICVAFBFL DSECT=NO         BUFFER LIST
```

```
+***********************************************************************
+*       BUFFER LIST HEADER
+***********************************************************************

+BUFLIST  DS    0F                           BUFFER LIST HEADER
+BFLHNOE  DS    XL1                          NUMBER OF ENTRIES
+BFLHFL   DS    XL1                          KEY AND FLAG BYTE
+         ORG   BFLHFL
+BFLHKEY  DS    XL1                          PROTECT KEY (FIRST 4 BITS)
+BFLHVIR  EQU   X'08'                        BUF. LIST ENTRIES DESCRIBE VIRS
+BFLHDSCB EQU   X'04'                        BUF. LIST ENTRIES DESCRIBE DSCBS
+         DS    XL1                          RESERVED
+BFLHSP   DS    XL1                          SUBPOOL OF BUF. LIST/BUFFERS
+BFLHFCHN DS    A                            FORWARD CHAIN PTR TO NEXT BUF.
+*                                           LIST
+BFLHLN   EQU   *-BUFLIST                    LENGTH OF BUFFER LIST HEADER

+***********************************************************************
+*       BUFFER LIST ENTRY
+***********************************************************************

+BFLE     DS    0F                           BUFFER LIST ENTRY
+BFLEFL   DS    XL1                          BUFFER LIST ENTRY FLAG
+BFLERBA  EQU   X'80'                        ARGUMENT IS RBA
+BFLECHR  EQU   X'40'                        ARGUMENT IS CCHHR
+BFLETTR  EQU   X'20'                        ARGUMENT IS TTR
+BFLEAUPD EQU   X'10'                        CVAF UPDATED ARGUMENT FIELD
+BFLEMOD  EQU   X'08'                        DATA IN BUF. HAS BEEN MODIFIED
+BFLESKIP EQU   X'04'                        SKIP THIS ENTRY
+BFLEIOER EQU   X'02'                        I/O ERROR
+         DS    XL1                          RESERVED
+BFLELTH  DS    XL1                          LENGTH OF DSCB BUFFER OR
+*                                           LENGTH OF VIR DIVIDED BY 256
+BFLEARG  DS    XL5                          ARGUMENT OF VIR OR DSCB (CCHHR)
+         ORG   BFLEARG+1
+BFLEATTR DS    XL3                          'TTR' OF ARGUMENT
+         ORG   BFLEARG+1
+BFLEARBA DS    XL4                          'RBA'  OF ARGUMENT
+BFLEBUF  DS    A                            BUFFER ADDRESS
+BFLELN   EQU   *-BFLE                       LENGTH OF A BUFFER LIST ENTRY
         IECSDSL1 (1)                   FORMAT 1 DSCB DATA SET NAME AND     *
                                        BUFFER
+IECSDSL1 EQU   *                       FORMAT 1 DSCB
+IECSDSF1 EQU   IECSDSL1
+DS1DSNAM DS    CL44                    DATA SET NAME
+DS1FMTID DS    CL1                     FORMAT IDENTIFIER
+DS1DSSN  DS    CL6                     DATA SET SERIAL NUMBER
+DS1VOLSQ DS    XL2                     VOLUME SEQUENCE NUMBER
+DS1CREDT DS    XL3                     CREATION DATE
+DS1EXPDT DS    XL3                     EXPIRATION DATE
+DS1NOEPV DS    XL1                     NUMBER OF EXTENTS ON VOLUME
+DS1NOBDB DS    XL1                     NUMBER OF BYTES USED IN LAST
+*                                          DIRECTORY BLOCK
+         DS    XL1                     RESERVED
+DS1SYSCD DS    CL13                    SYSTEM CODE
+         DS    XL7                     RESERVED
+DS1DSORG DS    XL2                     DATA SET ORGANIZATION
```

```
+*                    FIRST BYTE OF DS1DSORG
+DS1DSGIS EQU   X'80'                IS - INDEXED SEQUENTIAL         @01A
+*                                   ORGANIZATION
+DS1DSGPS EQU   X'40'                PS - PHYSICAL SEQUENTIAL        @01A
+*                                   ORGANIZATION
+DS1DSGDA EQU   X'20'                DA - DIRECT ORGANIZATION        @01A
+DS1DSGCX EQU   X'10'                CX - BTAM OR QTAM LINE GROUP    @01A
+*        EQU   X'08'                RESERVED                        @01A
+*        EQU   X'04'                RESERVED                        @01A
+DS1DSGPO EQU   X'02'                PO - PARTITIONED ORGANIZATION   @01A
+DS1DSGU  EQU   X'01'                U - UNMOVABLE, THE DATA         @01A
+*                                   CONTAINS LOCATION DEPENDENT
+*                                   INFORMATION
+*
+*                     SECOND BYTE OF DS1DSORG
+DS1DSGGS EQU   X'80'                GS - GRAPHICS ORGANIZATION      @01A
+DS1DSGTX EQU   X'40'                TX - TCAM LINE GROUP            @01A
+DS1DSGTQ EQU   X'20'                TQ - TCAM MESSAGE QUEUE         @01A
+*        EQU   X'10'                RESERVED                        @01A
+DS1ACBM  EQU   X'08'                ACCESS METHOD CONTROL BLOCK     @01A
+DS1DSGTR EQU   X'04'                TR - TCAM 3705                  @01A
+*        EQU   X'02'                RESERVED                        @01A
+*        EQU   X'01'                RESERVED                        @01A
+DS1RECFM DS    XL1                  RECORD FORMAT
+DS1OPTCD DS    XL1                  OPTION CODE
+DS1BLKL  DS    XL2                  BLOCK LENGTH
+DS1LRECL DS    XL2                  RECORD LENGTH
+DS1KEYL  DS    XL1                  KEY LENGTH
+DS1RKP   DS    XL2                  RELATIVE KEY POSITION
+DS1DSIND DS    XL1                  DATA SET INDICATORS
+DS1SCALO DS    XL4                  SECONDARY ALLOCATION
+DS1LSTAR DS    XL3                  LAST USED TRACK AND BLOCK ON TRACK
+DS1TRBAL DS    XL2                  BYTES REMAINING ON LAST TRACK USED
+         DS    XL2                  RESERVED
+DS1EXT1  DS    XL10                 FIRST EXTENT DESCRIPTION
+*        FIRST BYTE                 EXTENT TYPE INDICATOR
+*        SECOND BYTE                EXTENT SEQUENCE NUMBER
+*        THIRD - SIXTH BYTES        LOWER LIMIT
+*        SEVENTH - TENTH BYTES      UPPER LIMIT
+DS1EXT2  DS    XL10                 SECOND EXTENT DESCRIPTION
+DS1EXT3  DS    XL10                 THIRD EXTENT DESCRIPTION
+DS1PTRDS DS    XL5                  POSSIBLE PTR TO A FORMAT 2 OR 3 DSCB
+DS1END   EQU   *
 DSCBLTH  EQU   *-IECSDSL1-L'DS1DSNAM LENGTH OF DATA PORTION OF DSCB
 SAVEAREA DS    18F                  SAVE AREA
 CVPL     CVAFSEQ ACCESS=GTEQ,       READ DSCB WITH DSN >= SUPPLIED DSN  *
                IXRCDS=KEEP,         KEEP VIERS IN STORAGE DURING CALLS  *
                DSN=DS1DSNAM,        SUPPLIED DATA SET NAME              *
                BUFLIST=BUFLIST,
                MF=L
+         CNOP  0,4
+CVPL     EQU   *
+         DC    CL4'CVPL'                 EBCDIC 'CVPL'
+         DC    AL2(ICV10E-CVPL)          LENGTH OF CVPL
+         DC    XL1'05'                   FUNCTION CODE
+         DC    XL1'00'                   STATUS INFORMATION
+         DC    B'00001000'               FIRST FLAG BYTE
+         DC    B'00000000'               SECOND FLAG BYTE
+          DC    H'0'                          RESERVED
+          DC    A(0)                          UCB ADDRESS
+          DC    A(DS1DSNAM)                   DATA SET NAME ADDRESS
+          DC    A(0)                          BUFFER LIST ADDRESS
+          DC    A(0)                          INDEX VIR'S BUFFER LIST ADDRESS
+          DC    A(0)                          MAP VIR'S BUFFER LIST ADDRESS
+          DC    A(0)                          I/O AREA ADDRESS
+          DC    A(0)                          DEB ADDRESS
+          DC    A(0)                          ARGUMENT ADDRESS
+          DC    A(0)                          SPACE PARAMETER LIST ADDRESS
+          DC    A(0)                          EXTENT TABLE ADDRESS
+          DC    A(0)                          NEW VRF VIXM BUFFER LIST ADDR
+          DC    A(0)                          VRF DATA ADDRESS
+          DC    A(0)                          COUNT AREA ADDRESS
+ICV10E    EQU   *                             END OF CVPL
           ORG   CVPL                     EXPAND MAP OVER LIST
 CVPLMAP   ICVAFPL DSECT=NO               CVPL MAP
+***********************************************************************
+*         CVAF PARAMETER LIST
+***********************************************************************

+CVPLMAP   DS    0F                           CVAF PARAMETER LIST
+CVLBL     DS    CL4                          EBCDIC 'CVPL'
+CVLTH     DS    H                            LENGTH OF CVPL
+CVFCTN    DS    XL1                          FUNCTION BYTE
+CVDIRD    EQU   X'01'                        CVAFDIR ACCESS=READ
+CVDIWR    EQU   X'02'                        CVAFDIR ACCESS=WRITE
+CVDIRLS   EQU   X'03'                        CVAFDIR ACCESS=RLSE
+CVSEQGT   EQU   X'04'                        CVAFSEQ ACCESS=GT
+CVSEQGTE  EQU   X'05'                        CVAFSEQ ACCESS=GTEQ
+CVDMIXA   EQU   X'06'                        CVAFDSM ACCESS=IXADD
+CVDMIXD   EQU   X'07'                        CVAFDSM ACCESS=IXDLT
+CVDMALC   EQU   X'08'                        CVAFDSM ACCESS=ALLOC
+CVDMRLS   EQU   X'09'                        CVAFDSM ACCESS=RLSE
+CVDMMAP   EQU   X'0A'                        CVAFDSM ACCESS=MAPDATA
+CVVOL     EQU   X'0B'                        CVAFVOL ACCESS=VIBBLD
+CVVRFRD   EQU   X'0C'                        CVAFVRF ACCESS=READ
+CVVRFWR   EQU   X'0D'                        CVAFVRF ACCESS=WRITE
+CVSTAT    DS    XL1                          STATUS INFORMATION (SEE LIST    *
+                                             BELOW)
+CVFL1     DS    XL1                          FIRST FLAG BYTE
+CV1IVT    EQU   X'80'                        INDEXED VTOC ACCESSED
+CV1IOAR   EQU   X'40'                        IOAREA=KEEP
+CV1PGM    EQU   X'20'                        BRANCH=(YES,PGM)
+CV1MRCDS  EQU   X'10'                        MAPRCDS=YES
+CV1IRCDS  EQU   X'08'                        IXRCDS=KEEP
+CV1MAPIX  EQU   X'04'                        MAP=INDEX
+CV1MAPVT  EQU   X'02'                        MAP=VTOC
+CV1MAPVL  EQU   X'01'                        MAP=VOLUME
+CVFL2     DS    XL1                          SECOND FLAG BYTE
+CV2HIVIE  EQU   X'80'                        HIVIER=YES
+CV2VRF    EQU   X'40'                        VRF DATA EXISTS
+CV2CNT    EQU   X'20'                        COUNT=YES
+CV2RCVR   EQU   X'10'                        RECOVER=YES
+CV2SRCH   EQU   X'08'                        SEARCH=YES
+CV2DSNLY  EQU   X'04'                        DSNONLY=YES
+CV2VER    EQU   X'02'                        VERIFY=YES
+CV2NLEVL  EQU   X'01'                        OUTPUT-NEW HIGHEST LEVEL VIER
+*                                            CREATED
+          DS    H                            RESERVED
```

```
+CVUCB    DS    A                        UCB ADDRESS
+CVDSN    DS    A                        DATA SET NAME ADDRESS
+CVBUFL   DS    A                        BUFFER LIST ADDRESS
+CVIRCDS  DS    A                        INDEX VIR'S BUFFER LIST ADDRESS
+CVMRCDS  DS    A                        MAP VIR'S BUFFER LIST ADDRESS
+CVIOAR   DS    A                        I/O AREA ADDRESS
+CVDEB    DS    A                        DEB ADDRESS
+CVARG    DS    A                        ARGUMENT ADDRESS
+CVSPACE  DS    A                        SPACE PARAMETER LIST ADDRESS
+CVEXTS   DS    A                        EXTENT TABLE ADDRESS
+CVBUFL2  DS    A                        NEW VRF VIXM BUFFER LIST ADDR
+CVVRFDA  DS    A                        VRF DATA ADDRESS
+CVCTAR   DS    A                        COUNT AREA ADDRESS
+CVPLNGTH EQU   *-CVPLMAP

+*              VALUES OF CVSTAT
+*(THIS PART OF THE ICVAFPL MACRO EXPANSION IS NOT SHOWN)
          END
```

Example 5: Using the CVAFSEQ Macro with a Nonindexed VTOC

This example reads as many as five DSCBs in physical-sequential order. The address of the UCB is supplied to the program in register 5 (labeled RUCB). The address of a parameter list is supplied in register 4 (labeled RLIST). The first word of the parameter list contains the address of a 5-byte field. On entry, this field is set to zero if no previous DSCBs have been read; otherwise, the field is set to the CCHHR of the last DSCB read. This 5-byte field is supplied by the caller of this program and is not modified by this program.

The remainder of the parameter list consists of one or more 2-word entries, to a maximum of five 2-word entries. The first word of each entry contains the address of a 140-byte DSCB buffer. The second word contains the address of a 5-byte field that is to contain the CCHHR of the DSCB.

A buffer list with five buffer list entries is contained in the program. The ICVAFBFL macro generates the buffer list header and one buffer list entry. The remaining buffer list entries are generated following the ICVAFBFL macro.

The CVAFSEQ macro is used once in the program to read as many DSCBs as there are 2-word entries in the parameter list. The buffer list header field BFLHNOE is initialized with the number of buffer list entries that CVAFSEQ is to process. The number matches the number of 2-word entries in the parameter list supplied to this program.

After the CVAFSEQ call, the CCHHR for each DSCB read is moved from the buffer list entry field BFLEARG to the field whose address is supplied by the caller of the program. If the BFLEARG field is zero, the previous DSCB read was the last in the VTOC.

The BFLEARG in the first buffer list entry is initialized with the CCHHR supplied by the caller: its address is the third word in the parameter list. This CCHHR serves as the starting place for the CVAFSEQ call. DSCBs with a CCHHR greater than the supplied CCHHR are read.

This program must be APF authorized.

```
EXAMPLE5 CSECT
         STM   14,12,12(13)
         BALR  12,0
         USING *,12
         ST    13,SAVEAREA+4
         LA    RWORK,SAVEAREA
         ST    RWORK,8(,13)
         LR    13,RWORK
```

```
************************************************************
*
*        REGISTERS
*
************************************************************
REG1     EQU   1                   REGISTER 1
RWORK    EQU   3                   WORK REGISTER
RLIST    EQU   4                   ADDRESS OF PARM LIST
RUCB     EQU   5                   UCB ADDRESS
RCURRENT EQU   6                   CURRENT ENTRY IN PARM LIST
RBLE     EQU   7                   CURRENT BUFFER LIST ENTRY
RCOUNT   EQU   8                   COUNT OF ENTRIES IN BUFFER LIST
REG15    EQU   15                  RETURN CODE REGISTER 15
************************************************************
*
*        READ UP TO 5 DSCBS.
*        RUCB CONTAINS ADDRESS OF UCB.
*        RLIST CONTAINS ADDRESS OF PARAMETER LIST.
*          WORD 0 = ADDRESS OF CCHHR OF LAST DSCB READ. THIS DSCB IS
*                   NOT TO BE READ
*          WORD 1 = ADDRESS OF DSCB BUFFER.
*          WORD 2 = ADDRESS OF CCHHR OF DSCB READ.
*            WORD1 AND WORD2 REPEATED UP TO 4 TIMES.
*            HIGH ORDER BIT OF WORD 2 SET TO ONE FOR LAST ENTRY.
*
************************************************************
         USING LIST,RLIST           ADDRESSABILITY TO PARM LIST
         XC    BFLHDR(BFLHLN+5*BFLELN),BFLHDR ZERO BUFFER LIST WITH      *
                                    5 BUFFER LIST ENTRIES
         OI    BFLHFL,BFLHDSCB      DSCBS TO BE READ WITH BUFFER LIST
         LA    RCURRENT,LISTPRMS    FIRST DOUBLEWORD ENTRY IN PARM LIST
         USING LISTPRMS,RCURRENT    USING ON DOUBLEWORDS
         LA    RBLE,BFLE            FIRST BUFFER LIST ENTRY
         USING BFLE,RBLE
         L     RWORK,LISTSTRT       ADDRESS OF STARTING CCHHR
         MVC   BFLEARG,0(RWORK)     MOVE STARTING CCHHR INTO FIRST       *
                                    BUFFER LIST ENTRY
         XR    RCOUNT,RCOUNT        ZERO COUNT
BUFLOOP  EQU   *                    PUT BUFFER ADDRESSES IN BUFFER LIST *
                                    ENTRIES
         LA    RCOUNT,1(,RCOUNT)    INCREMENT COUNT
         L     RWORK,LISTBUF        ADDRESS OF DSCB BUFFER
         ST    RWORK,BFLEBUF-BFLE(,RBLE) PLACE IN BUFFER LIST
         MVI   BFLELTH-BFLE(RBLE),DSCBLTH FULL DSCB READ
         TM    LISTLAST,LASTBIT     IS IT LAST ENTRY IN LIST
         LA    RCURRENT,LISTNEXT    INCREMENT TO NEXT ENTRY IN LIST
         LA    RBLE,BFLELN(,RBLE)   INCREMENT TO NEXT BUFFER LIST ENTRY
         BZ    BUFLOOP              LOOP TO PUT NEXT BUFFER IN BFLE
         STC   RCOUNT,BFLHNOE       SET NUMBER OF ENTRIES IN BUFFER      *
                                    LIST HEADER
         DROP RCURRENT,RBLE
************************************************************
*
*        READ UP TO 5 DSCBS WHOSE CCHHR IS GREATER THAN THE CCHHR IN
*        THE FIRST BUFFER LIST ENTRY
*
************************************************************
```

```
         CVAFSEQ UCB=(RUCB),        CALL CVAF                               *
               BRANCH=YES,          BRANCH ENTER                            *
               MF=(E,CVPL)
+        LA    1,CVPL                              LOAD PARAMETER REG 1

+        ST    RUCB,12(,1)               STORE UCB PTR IN PARM LIST
+        L     15,16                     LOAD THE CVT
+        L     15,328(,15)               LOAD VS1/VS2 COMMON EXTENSION2
+        L     15,12(,15)                LOAD THE CVAF TABLE ADDRESS
+        L     15,0(,15)                 LOAD THE CVAF ADDRESS
+        BALR  14,15                     BRANCH AND LINK TO CVAF
         USING CVPL,REG1            ADDRESSABILITY TO CVPL
         LTR  REG15,REG15           ANY ERROR
         BZ   MOVECHR               BRANCH IF MOVE IN CCHHRS
************************************************************
*
*        DETERMINE WHAT ERROR IS
*
************************************************************
         C    REG15,ERROR4          IS RETURN CODE 4
         BNE  OTHERERR              BRANCH IF NOT 4
         CLI  CVSTAT,STAT032        IS IT END OF DATA?
         BNE  OTHERERR              BRANCH IF NOT
         DROP REG1                  ADDRESSABILITY TO CVPL NOT NEEDED
************************************************************
*
*        DETERMINE IF ANY DSCBS HAVE BEEN READ. BFLEARG IS NON-ZERO
*        IN EACH BUFFER LIST ENTRY FOR WHICH A DSCB HAS BEEN READ
*
************************************************************
MOVECHR  EQU  *                     IS DATA SET NAME GREATER THAN DSN2
         LA   RCURRENT,LISTPRMS     FIRST ENTRY IN PARM LIST
         USING LISTPRMS,RCURRENT
         LA   RBLE,BFLE             FIRST BUFFER LIST ENTRY
         USING BFLE,RBLE
CHRLOOP  EQU  *                     MOVE CCHHR ARGUMENT TO CALLER AREA
         L    RWORK,LISTCHR         ADDRESS OF CCHHR OF CALLER
         XC   0(L'BFLEARG,RWORK),0(RWORK) ZERO CALLER CCHHR AREA
         NC   BFLEARG,BFLEARG       IS CCHHR ZERO
         BZ   EXIT                  BRANCH IF YES-NO MORE DSCBS
         MVC  0(L'BFLEARG,RWORK),BFLEARG MOVE CCHHR TO CALLER AREA
         TM   LISTLAST,LASTBIT      LAST ENTRY IN PARM LIST?
         BO   EXIT                  BRANCH IF YES
         LA   RCURRENT,LISTNEXT     NEXT ENTRY IN LIST
         LA   RBLE,BFLELN(,RBLE)    NEXT BUFFER LIST ENTRY
         B    CHRLOOP               TEST NEXT BFLE
EXIT     EQU  *                     RETURN TO CALLER
         L    13,SAVEAREA+4
         RETURN (14,12)
+        LM    14,12,12(13)                        RESTORE THE REGISTERS
+        BR    14                                  RETURN

OTHERERR EQU  *                     ERROR PROCESSING
*
*
*
         B    EXIT                  RETURN
ERROR4   DC   F'4'                  RETURN CODE 4
         ICVAFBFL DSECT=NO          BUFFER LIST WITH ONE BUFFER LIST       *
                                    ENTRY
```

```
+***********************************************************************
+*        BUFFER LIST HEADER
+***********************************************************************

+BFLHDR   DS    0F                           BUFFER LIST HEADER
+BFLHNOE  DS    XL1                          NUMBER OF ENTRIES
+BFLHFL   DS    XL1                          KEY AND FLAG BYTE
+         ORG   BFLHFL
+BFLHKEY  DS    XL1                          PROTECT KEY (FIRST 4 BITS)
+BFLHVIR  EQU   X'08'                        BUF. LIST ENTRIES DESCRIBE VIRS
+BFLHDSCB EQU   X'04'                        BUF. LIST ENTRIES DESCRIBE DSCBS
+         DS    XL1                          RESERVED
+BFLHSP   DS    XL1                          SUBPOOL OF BUF. LIST/BUFFERS
+BFLHFCHN DS    A                            FORWARD CHAIN PTR TO NEXT BUF.
+*                                           LIST
+BFLHLN   EQU   *-BFLHDR                     LENGTH OF BUFFER LIST HEADER

+***********************************************************************
+*        BUFFER LIST ENTRY
+***********************************************************************

+BFLE     DS    0F                           BUFFER LIST ENTRY
+BFLEFL   DS    XL1                          BUFFER LIST ENTRY FLAG
+BFLERBA  EQU   X'80'                        ARGUMENT IS RBA
+BFLECHR  EQU   X'40'                        ARGUMENT IS CCHHR
+BFLETTR  EQU   X'20'                        ARGUMENT IS TTR
+BFLEAUPD EQU   X'10'                        CVAF UPDATED ARGUMENT FIELD
+BFLEMOD  EQU   X'08'                        DATA IN BUF. HAS BEEN MODIFIED
+BFLESKIP EQU   X'04'                        SKIP THIS ENTRY
+BFLEIOER EQU   X'02'                        I/O ERROR
+         DS    XL1                          RESERVED
+BFLELTH  DS    XL1                          LENGTH OF DSCB BUFFER OR
+*                                           LENGTH OF VIR DIVIDED BY 256
+BFLEARG  DS    XL5                          ARGUMENT OF VIR OR DSCB (CCHHR)
+         ORG   BFLEARG+1
+BFLEATTR DS    XL3                          'TTR' OF ARGUMENT
+         ORG   BFLEARG+1
+BFLEARBA DS    XL4                          'RBA'  OF ARGUMENT
+BFLEBUF  DS    A                            BUFFER ADDRESS
+BFLELN   EQU   *-BFLE                       LENGTH OF A BUFFER LIST ENTRY
          DS    CL(4*BFLELN)             FOUR BUFFER LIST ENTRIES
 SAVEAREA DS    18F                      SAVE AREA
 DSCB     DSECT
          IECSDSL1 (1)                   FORMAT 1 DSCB DATASET NAME AND      *
                                         DATA
+IECSDSL1 EQU   *                        FORMAT 1 DSCB
+IECSDSF1 EQU   IECSDSL1
+DS1DSNAM DS    CL44                     DATA SET NAME
+DS1FMTID DS    CL1                      FORMAT IDENTIFIER
+DS1DSSN  DS    CL6                      DATA SET SERIAL NUMBER
+DS1VOLSQ DS    XL2                      VOLUME SEQUENCE NUMBER
+DS1CREDT DS    XL3                      CREATION DATE
+DS1EXPDT DS    XL3                      EXPIRATION DATE
+DS1NOEPV DS    XL1                      NUMBER OF EXTENTS ON VOLUME
+DS1NOBDB DS    XL1                      NUMBER OF BYTES USED IN LAST
+*                                           DIRECTORY BLOCK
+         DS    XL1                      RESERVED
+DS1SYSCD DS    CL13                     SYSTEM CODE
+         DS    XL7                      RESERVED
+DS1DSORG DS    XL2                      DATA SET ORGANIZATION
```

```
+*                    FIRST BYTE OF DS1DSORG
+DS1DSGIS EQU   X'80'              IS - INDEXED SEQUENTIAL         @01A
+*                                 ORGANIZATION
+DS1DSGPS EQU   X'40'              PS - PHYSICAL SEQUENTIAL        @01A
+*                                 ORGANIZATION
+DS1DSGDA EQU   X'20'              DA - DIRECT ORGANIZATION        @01A
+DS1DSGCX EQU   X'10'              CX - BTAM OR QTAM LINE GROUP    @01A
+*        EQU   X'08'              RESERVED                        @01A
+*        EQU   X'04'              RESERVED                        @01A
+DS1DSGPO EQU   X'02'              PO - PARTITIONED ORGANIZATION   @01A
+DS1DSGU  EQU   X'01'              U - UNMOVABLE, THE DATA         @01A
+*                                 CONTAINS LOCATION DEPENDENT
+*                                 INFORMATION
+*
+*                     SECOND BYTE OF DS1DSORG
+DS1DSGGS EQU   X'80'              GS - GRAPHICS ORGANIZATION      @01A
+DS1DSGTX EQU   X'40'              TX - TCAM LINE GROUP            @01A
+DS1DSGTQ EQU   X'20'              TQ - TCAM MESSAGE QUEUE         @01A
+*        EQU   X'10'              RESERVED                        @01A
+DS1ACBM  EQU   X'08'              ACCESS METHOD CONTROL BLOCK     @01A
+DS1DSGTR EQU   X'04'              TR - TCAM 3705                  @01A
+*        EQU   X'02'              RESERVED                        @01A
+*        EQU   X'01'              RESERVED                        @01A
+DS1RECFM DS    XL1                RECORD FORMAT
+DS1OPTCD DS    XL1                OPTION CODE
+DS1BLKL  DS    XL2                BLOCK LENGTH
+DS1LRECL DS    XL2                RECORD LENGTH
+DS1KEYL  DS    XL1                KEY LENGTH
+DS1RKP   DS    XL2                RELATIVE KEY POSITION
+DS1DSIND DS    XL1                DATA SET INDICATORS
+DS1SCALO DS    XL4                SECONDARY ALLOCATION
+DS1LSTAR DS    XL3                LAST USED TRACK AND BLOCK ON TRACK
+DS1TRBAL DS    XL2                BYTES REMAINING ON LAST TRACK USED
+         DS    XL2                RESERVED
+DS1EXT1  DS    XL10               FIRST EXTENT DESCRIPTION
+*        FIRST BYTE               EXTENT TYPE INDICATOR
+*        SECOND BYTE              EXTENT SEQUENCE NUMBER
+*        THIRD - SIXTH BYTES      LOWER LIMIT
+*        SEVENTH - TENTH BYTES    UPPER LIMIT
+DS1EXT2  DS    XL10               SECOND EXTENT DESCRIPTION
+DS1EXT3  DS    XL10               THIRD EXTENT DESCRIPTION
+DS1PTRDS DS    XL5                POSSIBLE PTR TO A FORMAT 2 OR 3 DSCB
+DS1END   EQU   *
 DSCBLTH  EQU   *-IECSDSL1         LENGTH OF DSCB
 LIST     DSECT                    PARAMETER LIST
 LISTSTRT DS    F                  ADDRESS OF CCHHR TO START SEARCH
 LISTPRMS EQU   *
 LISTBUF  DS    F                  BUFFER ADDRESS
 LISTCHR  DS    0F                 ADDRESS OF CCHHR FIELD
 LISTLAST DS    X                  BYTE
 LASTBIT  EQU   X'80'              LAST DOUBLE WORD
          DS    AL3                3 BYTE ADDRESS OF CCHHR
 LISTNEXT EQU   *                  NEXT DOUBLEWORD
 EXAMPLE5 CSECT
 ***********************************************************
 *
 *        READ DSCBS WITH CCHHR GREATER THAN THE CCHHR IN THE FIRST
 *        BUFFER LIST ENTRY.
 *
 ***********************************************************
 CVPL      CVAFSEQ ACCESS=GT,                                             *
                 BUFLIST=BFLHDR,    ADDRESS OF BUFFER LIST                *
                 MF=L
+         CNOP  0,4
+CVPL     EQU   *
+         DC    CL4'CVPL'                EBCDIC 'CVPL'
+         DC    AL2(ICV6E-CVPL)          LENGTH OF CVPL
+         DC    XL1'04'                  FUNCTION CODE
+         DC    XL1'00'                  STATUS INFORMATION
+         DC    B'00100000'              FIRST FLAG BYTE
+         DC    B'00000000'              SECOND FLAG BYTE
+         DC    H'0'                     RESERVED
+         DC    A(0)                     UCB ADDRESS
+         DC    A(0)                     DATA SET NAME ADDRESS
+         DC    A(BFLHDR)                BUFFER LIST ADDRESS
+         DC    A(0)                     INDEX VIR'S BUFFER LIST ADDRESS
+         DC    A(0)                     MAP VIR'S BUFFER LIST ADDRESS
+         DC    A(0)                     I/O AREA ADDRESS
+         DC    A(0)                     DEB ADDRESS
+         DC    A(0)                     ARGUMENT ADDRESS
+         DC    A(0)                     SPACE PARAMETER LIST ADDRESS
+         DC    A(0)                     EXTENT TABLE ADDRESS
+         DC    A(0)                     NEW VRF VIXM BUFFER LIST ADDR
+         DC    A(0)                     VRF DATA ADDRESS
+         DC    A(0)                     COUNT AREA ADDRESS
+ICV6E    EQU   *                        END OF CVPL
          ORG   CVPL                EXPAND MAP OVER LIST
 CVPLMAP  ICVAFPL DSECT=NO          CVPL MAP

+***********************************************************************
+*        CVAF PARAMETER LIST
+***********************************************************************

+CVPLMAP  DS    0F                      CVAF PARAMETER LIST
+CVLBL    DS    CL4                     EBCDIC 'CVPL'
+CVLTH    DS    H                       LENGTH OF CVPL
+CVFCTN   DS    XL1                     FUNCTION BYTE
+CVDIRD   EQU   X'01'                   CVAFDIR ACCESS=READ
+CVDIWR   EQU   X'02'                   CVAFDIR ACCESS=WRITE
+CVDIRLS  EQU   X'03'                   CVAFDIR ACCESS=RLSE
+CVSEQGT  EQU   X'04'                   CVAFSEQ ACCESS=GT
+CVSEQGTE EQU   X'05'                   CVAFSEQ ACCESS=GTEQ
+CVDMIXA  EQU   X'06'                   CVAFDSM ACCESS=IXADD
+CVDMIXD  EQU   X'07'                   CVAFDSM ACCESS=IXDLT
+CVDMALC  EQU   X'08'                   CVAFDSM ACCESS=ALLOC
+CVDMRLS  EQU   X'09'                   CVAFDSM ACCESS=RLSE
+CVDMMAP  EQU   X'0A'                   CVAFDSM ACCESS=MAPDATA
+CVVOL    EQU   X'0B'                   CVAFVOL ACCESS=VIBBLD
+CVVRFRD  EQU   X'0C'                   CVAFVRF ACCESS=READ
+CVVRFWR  EQU   X'0D'                   CVAFVRF ACCESS=WRITE
+CVSTAT   DS    XL1                     STATUS INFORMATION (SEE LIST      *
+                                       BELOW)
+CVFL1    DS    XL1                     FIRST FLAG BYTE
+CV1IVT   EQU   X'80'                   INDEXED VTOC ACCESSED
+CV1IOAR  EQU   X'40'                   IOAREA=KEEP
+CV1PGM   EQU   X'20'                   BRANCH=(YES,PGM)
+CV1MRCDS EQU   X'10'                   MAPRCDS=YES
```

```
+CV1IRCDS EQU   X'08'                    IXRCDS=KEEP
+CV1MAPIX EQU   X'04'                    MAP=INDEX
+CV1MAPVT EQU   X'02'                    MAP=VTOC
+CV1MAPVL EQU   X'01'                    MAP=VOLUME
+CVFL2    DS    XL1                      SECOND FLAG BYTE
+CV2HIVIE EQU   X'80'                    HIVIER=YES
+CV2VRF   EQU   X'40'                    VRF DATA EXISTS
+CV2CNT   EQU   X'20'                    COUNT=YES
+CV2RCVR  EQU   X'10'                    RECOVER=YES
+CV2SRCH  EQU   X'08'                    SEARCH=YES
+CV2DSNLY EQU   X'04'                    DSNONLY=YES
+CV2VER   EQU   X'02'                    VERIFY=YES
+CV2NLEVL EQU   X'01'                    OUTPUT-NEW HIGHEST LEVEL VIER
+*                                       CREATED
+         DS    H                        RESERVED
+CVUCB    DS    A                        UCB ADDRESS
+CVDSN    DS    A                        DATA SET NAME ADDRESS
+CVBUFL   DS    A                        BUFFER LIST ADDRESS
+CVIRCDS  DS    A                        INDEX VIR'S BUFFER LIST ADDRESS
+CVMRCDS  DS    A                        MAP VIR'S BUFFER LIST ADDRESS
+CVIOAR   DS    A                        I/O AREA ADDRESS
+CVDEB    DS    A                        DEB ADDRESS
+CVARG    DS    A                        ARGUMENT ADDRESS
+CVSPACE  DS    A                        SPACE PARAMETER LIST ADDRESS
+CVEXTS   DS    A                        EXTENT TABLE ADDRESS
+CVBUFL2  DS    A                        NEW VRF VIXM BUFFER LIST ADDR
+CVVRFDA  DS    A                        VRF DATA ADDRESS
+CVCTAR   DS    A                        COUNT AREA ADDRESS
+CVPLNGTH EQU   *-CVPLMAP

+*              VALUES OF CVSTAT
+*(THIS PART OF THE ICVAFPL MACRO EXPANSION IS NOT SHOWN)
          END
```

Example 6: Using the CVAFTST and CVAFDSM Macros

This example returns a format-5 DSCB to the caller. The format-5 DSCB is constructed by this program if the volume contains an indexed VTOC. The format-5 DSCB is read by another program, F5RTN (not described in the example), if the volume contains a nonindexed VTOC.

The CVAFTST macro is used to determine if a nonindexed VTOC is on the volume.

If the CVAFTST return code is neither 0 nor 4 (a nonindexed VTOC is on the volume), the CVAFDSM macro is issued to obtain up to 27 extents from the VPSM in the VTOC index. The program does not determine whether the CVAFTST return code is 8 (volume contains indexed VTOC) or 12 (it cannot be determined what type of VTOC is on the volume). In either case, the CVAFDSM macro is issued. If the CVAFTST return code is 12, the CVAFDSM macro call will cause CVAF to determine whether an indexed or a nonindexed VTOC is on the volume, and the CV1IVT bit will be set to one or zero, accordingly.

The extent table (at label EXTABL) is initialized to request 27 extents from the CVAFDSM macro, which is one more than the number of extents that fit in a format-5 DSCB. The format-5 DSCB is constructed from the first 26 extents returned from the CVAFDSM call.

The first extent in the extent table is initialized from the last extent in the format-5 DSCB area supplied by the caller of the program. If this is the first call, the program assumes that the format-5 area is initialized to zero. Thus, the first extent in the extent table has a value of zero to serve as the starting place for the extent search. If this is the second or subsequent call, the last extent in the format-5 area would be the last extent obtained from the previous CVAFDSM call.

The format-5 chain pointer field (DS5PTRDS) is set to a nonzero value if CVAFDSM returned a 27th extent. In this case, the program will be called again to obtain another format-5 DSCB.

The program's return code is 0 if no errors were encountered and 4 if an error was encountered.

This program must be APF authorized.

```
EXAMPLE6 CSECT
         STM   14,12,12(13)
         BALR  12,0
         USING *,12
         ST    13,SAVEAREA+4
         LA    RWORK,SAVEAREA
         ST    RWORK,8(,13)
         LR    13,RWORK
```

```
***********************************************************
*
*        REGISTERS
*
***********************************************************
RDEB     EQU   3                  DEB ADDRESS SUPPLIED BY CALLER
RUCB     EQU   4                  UCB ADDRESS SUPPLIED BY CALLER
RF5      EQU   5                  ADDRESS OF FORMAT 5 BUFFER SUPPLIED  *
                                  BY CALLER
RWORK    EQU   6                  WORK REGISTER
REG15    EQU   15                 RETURN CODE REGISTER 15
*
KF5      EQU   26                 NUMBER OF FORMAT 5 EXTENTS
***********************************************************
*
*        READ FORMAT 5 DSCB OR BUILD A FORMAT 5 DSCB IF
*          AN INDEXED VTOC
*        UCB ADDRESS SUPPLIED IN RUCB.
*        RF5 CONTAINS THE ADDRESS OF THE FORMAT 5 DSCB BUFFER. IT
*          CONTAINS THE LAST FORMAT 5 DSCB READ OR BUILT. THE FORMAT 5
*          BUFFER IS ZERO IF THIS IS THE FIRST CALL
*        IF THE FORMAT 5 DSCB BUFFER RETURNED TO THE CALLER HAS A
*          NONZERO VALUE IN DS5PTRDS, THIS ROUTINE WILL BE CALLED
*          AGAIN TO OBTAIN THE NEXT FORMAT 5 DSCB.
*
***********************************************************
         USING IECSDSF5,RF5        ADDRESSABILITY TO FORMAT 5 BUFFER
         CVAFTST UCB=(RUCB)        TEST VTOC
+        CNOP  0,4                      START OF CVAFTST MACRO
+        LR    1,RUCB                           LOAD PARAMETER REG 1
+        L     15,16                    LOAD THE CVT
+        L     15,328(,15)              LOAD VS1/VS2 COMMON EXTENSION2
+        L     15,12(,15)               LOAD THE CVAF TABLE ADDRESS
+        LTR   15,15                    TEST FOR ZERO VALUE
+        BZ    ICV1E                    CVAF IS NOT ON THE SYSTEM
+        L     15,4(,15)                LOAD POINTER TO CVAF TEST E.P.
+        BALR  14,15                    BRANCH AND LINK TO CVAF TEST
+ICV1E   EQU   *                        END OF CVAFTST
         LTR   REG15,REG15
         BZ    UNINDXD             READ NEXT FORMAT 5
         C     REG15,NOTIXRC       UNINDEXED VTOC?
         BE    UNINDXD             READ NEXT FORMAT 5
***********************************************************
*
*        ASSUME INDEXED VTOC UNLESS CVAFDSM CALL INDICATES UNINDEXED
*
***********************************************************
         MVC   EXTS(L'DS5AVEXT),DS5MAVET+L'DS5MAVET-L'DS5AVEXT MOVE THE *
                                   LAST EXTENT FROM FORMAT 5 TO FIRST  *
                                   ENTRY IN THE EXTENT TABLE
         CVAFDSM MF=(E,CVPL),      GET 27 EXTENTS FROM CVPL            *
               UCB=(RUCB),         RUCB ADDRESS REQUIRED               *
               DEB=(RDEB),         RDEB ADDRESS REQUIRED BY            *
                                    UNAUTHORIZED PROGRAMS CALLING CVAF *
               BRANCH=YES          BRANCH ENTRY CALL                   *
```

```
+         LA     1,CVPL                               LOAD PARAMETER REG 1
+         L      15,16                       LOAD THE CVT
+         L      15,328(,15)                 LOAD VS1/VS2 COMMON EXTENSION2
+         L      15,12(,15)                  LOAD THE CVAF TABLE ADDRESS
+         L      15,0(,15)                   LOAD THE CVAF ADDRESS
+         BALR   14,15                       BRANCH AND LINK TO CVAF
          TM    CVFL1,CV1IVT         IS THIS INDEXED VTOC
          BZ    UNINDXD              READ FORMAT 5 IF NOT
          LTR   REG15,REG15          ANY ERROR
          BZ    NOERROR
          C     REG15,RC04
          BNE   OTHERERR             UNEXPECTED ERROR
          CLI   CVSTAT,STAT032       END OF DATA
          BNE   OTHERERR             UNEXPECTED ERROR
 NOERROR  EQU   *                    BUILD FORMAT 5
          MVC   DS5KEYID,F5ID
          MVC   DS5AVEXT(L'DS5AVEXT+L'DS5EXTAV),EXTS MOVE IN EXTENTS      *
                                     TO DS5FMTID
          MVI   DS5FMTID,C'5'
          MVC   DS5MAVET,EXTS+L'DS5AVEXT+L'DS5EXTAV MOVE REMAINING        *
                                      EXTENTS
          XR    REG15,REG15           RETURN CODE ZERO
          XC    DS5PTRDS,DS5PTRDS     ZERO CHAIN POINTER
          NC    EXTS+L'EXTS-L'DS5AVEXT(L'DS5AVEXT),EXTS+L'EXTS-L'DS5AVEXT*
                                      IS LAST(27TH) EXTENT FROM CVAF      *
                                      ZERO?
          BZ    RETURN                BRANCH IF YES-LEAVE DS5PTRDS ZERO
          MVI   DS5PTRDS+L'DS5PTRDS-1,1 SET DS5PTRDS NONZERO TO SIMULATE *
                                      THERE BEING ANOTHER FORMAT 5
          B     RETURN
 UNINDXD  EQU   *                     CALL ROUTINE TO READ NEXT FORMAT 5
          LINK EP=F5RTN               LINK TO FORMAT 5 ROUTINE. RETURN    *
                                      CODE PASSED BACK IN REG15
+         CNOP  0,4
+         BAL    15,*+20                              LOAD SUP.PARAMLIST ADR
+         DC     A(*+8)                               ADDR OF EP PARAMETER
+         DC     A(0)                     DCB ADDRESS PARAMETER        LC0A
+         DC     CL8'F5RTN'                           EP PARAMETER
+         SVC    6                                    ISSUE LINK SVC
 RETURN   EQU    *                    RETURN TO CALLER
          L      13,SAVEAREA+4
          RETURN (14,12),RC=(15)
+         L      14,12(13,0)                          RESTORE REGISTER 14
+         LM     0,12,20(13)                          RESTORE THE REGISTERS
+         BR     14                                   RETURN
 OTHERERR EQU    *                    ERROR
          L      REG15,RC04           ERROR RETURN CODE
          B      RETURN
 DSCB     DSECT
          IECSDSL1 (5)
+IECSDSL5 EQU    *                   FORMAT 5 DSCB
+IECSDSF5 EQU    IECSDSL5
+DS5KEYID DS     XL4                 KEY IDENTIFIER
+DS5AVEXT DS     XL5                 AVAILABLE EXTENT
+*        BYTES 1 - 2     RELATIVE TRACK ADDRESS OF THE FIRST TRACK
+*                        IN THE EXTENT
+*        BYTES 3 - 4     NUMBER OF UNUSED CYLINDERS IN THE EXTENT
+*        BYTE  5         NUMBER OF ADDITIONAL UNUSED TRACKS
```

```
+DS5EXTAV DS     XL35                  SEVEN AVAILABLE EXTENTS
+DS5FMTID DS     CL1                   FORMAT IDENTIFIER
+DS5MAVET DS     XL90                  EIGHTEEN AVAILABLE EXTENTS
+DS5PTRDS DS     XL5                   POINTER TO NEXT FORMAT 5 DSCB
+DS5END   EQU    *
 EXAMPLE6 CSECT
 NOTIXRC  DC     F'4'                  CVAFTST RETURN CODE-UNINDEXED
 RC04     DC     F'4'                  RETURN CODE 4
 F5ID     DC     XL4'0505050505'       FORMAT 5 FIELD, DS5KEYID
 SAVEAREA DS     18F                   REGISTER SAVE AREA
 EXTABL   DS     0CL(1+(KF5+1)*L'DS5AVEXT) EXTENT TABLE
 EXTNO    DC     AL1(KF5+1)            NUMBER OF EXTENTS IN TABLE
 EXTS     DS     CL((KF5+1)*L'DS5AVEXT) EXTENTS
 CVPL     CVAFDSM ACCESS=MAPDATA,                                      *
                 COUNT=NO,             DO NOT COUNT EXTENTS            *
                 MAP=VOLUME,           ACCESS VOLUME SPACE MAP         *
                 EXTENTS=EXTABL,       EXTENT TABLE ADDRESS            *
                 MF=L                  LIST FORM OF MACRO
+         CNOP   0,4
+CVPL     EQU    *
+         DC     CL4'CVPL'                  EBCDIC 'CVPL'
+         DC     AL2(ICV9E-CVPL)            LENGTH OF CVPL
+         DC     XL1'0A'                    FUNCTION CODE
+         DC     XL1'00'                    STATUS INFORMATION
+         DC     B'00100001'                FIRST FLAG BYTE
+         DC     B'00000000'                SECOND FLAG BYTE
+         DC     H'0'                       RESERVED
+         DC     A(0)                       UCB ADDRESS
+         DC     A(0)                       DATA SET NAME ADDRESS
+         DC     A(0)                       BUFFER LIST ADDRESS
+         DC     A(0)                       INDEX VIR'S BUFFER LIST ADDRESS
+         DC     A(0)                       MAP VIR'S BUFFER LIST ADDRESS
+         DC     A(0)                       I/O AREA ADDRESS
+         DC     A(0)                       DEB ADDRESS
+         DC     A(0)                       ARGUMENT ADDRESS
+         DC     A(0)                       SPACE PARAMETER LIST ADDRESS
+         DC     A(EXTABL)                  EXTENTS TABLE ADDRESS
+         DC     A(0)                       NEW VRF VIXM BUFFER LIST ADDR
+         DC     A(0)                       VRF DATA ADDRESS
+         DC     A(0)                       COUNT AREA ADDRESS
+ICV9E    EQU    *                          END OF CVPL
          ORG   CVPL                   OVERLAY CVPL WITH EXPANSION OF MAP
 CVPLMAP  ICVAFPL DSECT=NO
+***********************************************************************
+*        CVAF PARAMETER LIST
+***********************************************************************
+CVPLMAP  DS     0F                       CVAF PARAMETER LIST
+CVLBL    DS     CL4                      EBCDIC 'CVPL'
+CVLTH    DS     H                        LENGTH OF CVPL
+CVFCTN   DS     XL1                      FUNCTION BYTE
+CVDIRD   EQU    X'01'                    CVAFDIR ACCESS=READ
+CVDIWR   EQU    X'02'                    CVAFDIR ACCESS=WRITE
+CVDIRLS  EQU    X'03'                    CVAFDIR ACCESS=RLSE
+CVSEQGT  EQU    X'04'                    CVAFSEQ ACCESS=GT
+CVSEQGTE EQU    X'05'                    CVAFSEQ ACCESS=GTEQ
+CVDMIXA  EQU    X'06'                    CVAFDSM ACCESS=IXADD
+CVDMIXD  EQU    X'07'                    CVAFDSM ACCESS=IXDLT
+CVDMALC  EQU    X'08'                    CVAFDSM ACCESS=ALLOC
+CVDMRLS  EQU    X'09'                    CVAFDSM ACCESS=RLSE
+CVDMMAP  EQU    X'0A'                    CVAFDSM ACCESS=MAPDATA
```

```
+CVVOL    EQU   X'0B'                    CVAFVOL ACCESS=VIBBLD
+CVVRFRD  EQU   X'0C'                    CVAFVRF ACCESS=READ
+CVVRFWR  EQU   X'0D'                    CVAFVRF ACCESS=WRITE
+CVSTAT   DS    XL1                      STATUS INFORMATION (SEE LIST    X
+                                        BELOW)
+CVFL1    DS    XL1                      FIRST FLAG BYTE
+CV1IVT   EQU   X'80'                    INDEXED VTOC ACCESSED
+CV1IOAR  EQU   X'40'                    IOAREA=KEEP
+CV1PGM   EQU   X'20'                    BRANCH=(YES,PGM)
+CV1MRCDS EQU   X'10'                    MAPRCDS=YES
+CV1IRCDS EQU   X'08'                    IXRCDS=KEEP
+CV1MAPIX EQU   X'04'                    MAP=INDEX
+CV1MAPVT EQU   X'02'                    MAP=VTOC
+CV1MAPVL EQU   X'01'                    MAP=VOLUME
+CVFL2    DS    XL1                      SECOND FLAG BYTE
+CV2HIVIE EQU   X'80'                    HIVIER=YES
+CV2VRF   EQU   X'40'                    VRF DATA EXISTS
+CV2CNT   EQU   X'20'                    COUNT=YES
+CV2RCVR  EQU   X'10'                    RECOVER=YES
+CV2SRCH  EQU   X'08'                    SEARCH=YES
+CV2DSNLY EQU   X'04'                    DSNONLY=YES
+CV2VER   EQU   X'02'                    VERIFY=YES
+CV2NLEVL EQU   X'01'                    OUTPUT-NEW HIGHEST LEVEL VIER
+*                                       CREATED
+         DS    H                        RESERVED
+CVUCB    DS    A                        UCB ADDRESS
+CVDSN    DS    A                        DATA SET NAME ADDRESS
+CVBUFL   DS    A                        BUFFER LIST ADDRESS
+CVIRCDS  DS    A                        INDEX VIR'S BUFFER LIST ADDRESS
+CVMRCDS  DS    A                        MAP VIR'S BUFFER LIST ADDRESS
+CVIOAR   DS    A                        I/O AREA ADDRESS
+CVDEB    DS    A                        DEB ADDRESS
+CVARG    DS    A                        ARGUMENT ADDRESS
+CVSPACE  DS    A                        SPACE PARAMETER LIST ADDRESS
+CVEXTS   DS    A                        EXTENT TABLE ADDRESS
+CVBUFL2  DS    A                        NEW VRF VIXM BUFFER LIST ADDR
+CVVRFDA  DS    A                        VRF DATA ADDRESS
+CVCTAR   DS    A                        COUNT AREA ADDRESS
+CVPLNGTH EQU   *-CVPLMAP
+*             VALUES OF CVSTAT
+*(THIS PART OF THE ICVAFPL MACRO EXPANSION IS NOT SHOWN)
          END
```

Appendix C. VTOC Index Error Message and Associated Codes

Error Message

When CVAF finds an error in a VTOC index, it issues this message:

```
IEC606I VTOC INDEX DISABLED ON dev,volser,
 code,[rba[,secno,offset]]
```

In addition, CVAF puts a return code in the CVSTAT field of the CVPL.

Explanation

The Common VTOC Access Facility (CVAF) detected a VTOC index error on the device "dev" with volume serial number "volser." "code" is a number that represents the kind of VTOC index error encountered. "rba" is the RBA of the VIR in the VTOC index that contains a structure error indicated by "code." If the VIR is a VIER, the section number in the VIER containing the VTOC index entry is supplied in "secno," and the offset into the section of that VTOC index entry is supplied in "offset."

System Action

The VTOC index is disabled by zeroing the index bit in the format-4 DSCB and setting the bit in the first high-level VIER that indicates invalid VTOC index structure. The VTOC will be converted to nonindexed format when DADSM next allocates space on the volume. A system dump is written to the SYS1.DUMP data set, and an entry is made in the SYS1.LOGREC data set. The message IEC604I (which indicates that the VTOC convert routines have been used) will be issued later.

Programmer Response

Examine the system dump and a print of the VTOC index, and use the information in message IEC606I to determine the cause of the VTOC index structure error.

Routing and Descriptor Codes

The routing codes are 4 (direct access pool) and 10 (system/error maintenance), and the descriptor code is 4 (system status).

Codes Put in the CVSTAT Field

Code	Meaning
0(X'00')	No error.
1(X'01')	Data set name not found, or VIER is empty.
2(X'02')	Argument is outside VTOC extents or RBA range of VTOC index.
4(X'04')	Invalid parameter supplied (wrong key), or VRFAREA too small.
5(X'05')	DSN keyword omitted.
6(X'06')	Not authorized to perform this function.
7(X'07')	Buffer list omitted.
8(X'08')	DEB invalid or omitted or not open to VTOC.
9(X'09')	IOAREA=KEEP and user not authorized, or I/O area supplied and either user not authorized or CVAFVOL function.
10(X'0A')	Function not supported on OS VTOC.
11(X'0B')	DSCB is not format-0 DSCB and VERIFY=YES.
12(X'0C')	MAPRCDS=YES and/or IXRCDS=KEEP but VTOC is nonindexed.

Code	Meaning
13(X'0D')	IXRCDS=KEEP not specified for CVAFDSM ACCESS=IXADD or IXDLT.
14(X'0E')	CTAREA keyword omitted.
15(X'0F')	UCB invalid, volume not mounted; VIO unit, not DASD.
17(X'11')	DSCB length invalid for the function requested: 96 bytes for CVAFDIR ACCESS=WRITE,VERIFY=YES; 140 bytes for CVAFSEQ reading in data set name sequence; 96 bytes for CVAFSEQ reading in physical sequence.
19(X'13')	UCB omitted and CVAF I/O area not supplied.
22(X'16')	Data set name already supplied in index.
23(X'17')	Invalid DSN supplied (44 X'FF' is a reserved data set name).
24(X'18')	ARG keyword not supplied.
25(X'19')	Conflicting or incomplete information specified in the space table for a CVAFDSM ACCESS=ALLOC, MAP=VOLUME request.
27(X'1B')	VTOC index full. No free VIRs available and a VIER split is required.
28(X'1C')	Space keyword omitted (CVSPACE field zero in CVPL).
29(X'1D')	CVAFDSM ACCESS=ALLOC: No format 0 DSCB available (MAP=VTOC), or VTOC index full (MAP=INDEX), or volume space not available (MAP=VOLUME).
30(X'1E')	CVAFDSM ACCESS=ALLOC: CCHHR (MAP=VTOC) or RBA MAP=INDEX or volume space extent (MAP=VOLUME) already allocated.
31(X'1F')	CVAFDSM ACCESS=ALLOC or ACCESS=MAPDATA: CCHHR supplied outside VTOC extents (MAP=VTOC), or RBA outside VTOC index extents (MAP=INDEX), or volume space extent invalid or outside volume (MAP=VOLUME).
32(X'20')	End of data. CVAFDSM ACCESS=MAPDATA: no more free extents in VPSM. CVAFSEQ: no more names in index or DSCBs in VTOC. For indexed access, no DSN in VTOC index with higher or higher-or-equal key than that supplied. For physical-sequential access, no DSCB in the VTOC has a higher argument than that supplied. For a multiple DSCB request, the last DSCB in the VTOC was read and more DSCBs were requested.

Code	Meaning
33(X'21')	EXTENTS keyword omitted, or supplied number of extents is zero.
34(X'22')	CVAFDSM ACCESS=RLSE and format-0 DSCB already free (MAP=VTOC), or VIER already unallocated (MAP=INDEX) or volume space extent already unallocated (MAP=VOLUME).
42(X'2A')	VRF data supplied for write too long.
43(X'2B')	Buffer list is for VIRs, but a DSCB buffer list is required.
44(X'2C')	No buffer list entry found.
45(X'2D')	Invalid DSCB buffer length (neither 96 nor 140) in buffer list entry, or VIR buffer length not equal to VIB VIR size.
46(X'2E')	Neither TTR nor CCHHR bits set in buffer list entry to be used in writing or reading a 140-byte DSCB.
47(X'2F')	More than one of the TTR, CCHHR, and RBA bits set in the buffer list entry.
48(X'30')	Both the DSCB and VIR bits set in the buffer list header.
49(X'31')	RBA bit set in a buffer list entry for a DSCB buffer list.
50(X'32')	TTR or CCHHR bit set in buffer list entry but buffer list header indicates buffer list is for a VIR.
52(X'34')	Combination of MAP and COUNT not supported.
53(X'35')	MAP omitted.
54(X'36')	Buffer list for a VIR chained to or from a buffer list for a DSCB.
55(X'37')	Unauthorized caller and VIB not initialized.
56(X'38')	MAPRCDS=YES not specified but required.
57(X'39')	Buffer list for a DSCB supplied, but buffer list for a VIR is required (in MAPRCDS or IXRCDS buffer list address in CVAF parameter list).
58(X'3A')	Neither the VIR nor DSCB bit set in a buffer list header.

Code	Meaning
60(X'3C')	Invalid or conflicting setting of allocate option byte in space parameter
61(X'3D')	Filter criteria list address omitted (CVFCL=0) or ID not "FCL " (CVAFFILT).
62(X'3E')	FCLCOUNT field is zero or no elements to process (CVAFFILT).
63(X'3F')	FCLCOUNT field is greater than 1 and FCL1LIST indicates a generic data set name (CVAFFILT).
64(X'40')	Insufficient number of user buffer list entries to complete this request. All data set names in the filter criteria list up to this point processed successfully. Use RESUME function to continue processing (CVAFFILT).
65(X'41')	Buffer list entry error, for example, buffer length not 140 bytes (CVAFFILT).
66(X'42')	ACCESS=RESUME and filter save address (FSA) omitted (CVFSA=0), or FSA not in CVAF protect key (CVAFFILT).
67(X'43')	Invalid partially qualified data set name for generic access (CVAFFILT).
68(X'44')	Filter criteria list not in user key (CVAFFILT).
69(X'45')	Internal GETMAIN error (CVAFFILT).
70(X'46')	Direct VTOC access I/O error (CVAFFILT).
71(X'47')	Sequential VTOC access I/O error (CVAFFILT).
72(X'48')	Error in CVAFFILT processing. RESUME function not recommended (CVAFFILT).
73(X'49')	Insufficient number of user buffer list entries to complete this request. Error in DSCB chain. One or more names in the filter criteria list not processed successfully; however, RESUME function recommended for remaining DSCBs (CVAFFILT).
74(X'4A')	Data set name information is invalid (FCLDSNLG is either "0" or greater than '44', or FCLDSNA is either "0" or not in user key) (CVAFFILT).
127(X'7F')	I/O error occurred.
128(X'80')	Reserved.

Code	Meaning
129(X'81')	The first high-level VIER as indicated in the VIXM does not have the flag bit set indicating it is the first high-level VIER.
130(X'82')	A horizontal or vertical VIER pointer is outside the RBA range of the VTOC index.
131(X'83')	A vertical VIER pointer points to a VIR that is not a VIER (invalid ID in header).
132(X'84')	A level n vertical index entry pointer points to a VIER that is not at level n - 1.
133(X'85')	Level n horizontal index entry pointer points to VIER that is not at level n.
134(X'86')	Horizontal VIER/map pointer points to a VIR that is not a VIER/map (invalid ID in header).
135(X'87')	Horizontal map pointer points to VIR that is not one of the first n VTOC index records (n is recorded in VIXM field VIMRCDS), or the first record in the VTOC index is not a VIXM.
136(X'88')	A level-1 index entry contains a CCHHR pointer that is outside the VTOC extent.
137(X'89')	The first high-level VIER, as indicated in the VIB, does not have the flag bit set indicating it is the first high-level VIER. (This error is either recovered from by updating the VIB from the VIXM, or the error is changed to 129.)
138(X'8A')	The RBA of the VTOC index VIR does not match the RBA recorded in the header of the record.
139(X'8B')	The first record of a map (VIXM, VPSM, or VMDS) is not one of the first n VTOC index records (n is recorded in the VIXM field, VIMRCDS).
140(X'8C')	The data set name in a level n + 1 VIER entry is lower than the high key of the level n VIER that the level n + 1 VIER entry points to.
141(X'8D')	First high-level VIER structure error bit is on.
142(X'8E')	I/O error indicating the VTOC index is not formatted correctly.
143(X'8F')	Either the index bit is zero, or the DOS bit is zero in the format-4 DSCB of a VTOC previously found to be an indexed VTOC.
144(X'90')	No SYS1.VTOCIX.nnn data set name in a VTOC whose format-4 DSCB has the index bit on, indicating the VTOC has an index.

Code	Meaning
145(X'91')	The data set name in a level n + 1 VIER entry is higher than the high key of the level n VIER that the level n + 1 VIER entry points to
146(X'92')	Four or more high-level VIERs were encountered.
147(X'93')	Too many levels in the VTOC index. The length of the search list was exceeded.
148(X'94')	VIER invalid, because offset to last section is invalid.
149(X'95')	VIER invalid, because offset to last entry in a section is invalid.
150(X'96')	Media Manager initialization failed.
151(X'97')	Level-2 or higher VIER contains fewer than two entries.
152(X'98')	RECOVER=YES specified, but the static text module (ICVIXST0) indicates recovery is not permitted.
153(X'99')	The format-4 DSCB on an indexed VTOC is written with either the index- or DOS-bit zeroed on an indexed VTOC.
154(X'9A')	A space map extends over more than 10 VTOC index records.
155(X'9B')	Data set name not found in section with key greater than or equal to the name being searched for. The VIER section containing the name is invalid.
156(X'9C')	Invalid VIER horizontal pointer. Horizontal pointer of VIER1 points to VIER2 whose high key is lower than or equal to the high key of VIER1.
157(X'9D')	Could not find entry in level-2 or higher VIER that matches the high key of the VIER.
158(X'9E')	Invalid section length or invalid number of sections in a VIER header.
159(X'9F')	The first high-level VIER pointed to by the VIB has an invalid ID in the header.

Appendix D. Example of an OPEN Installation Exit Module

The description and example of IFG0EX0B (an installation-written OPEN exit module that takes control during OPEN for a DCB) has been moved to *Data Facility Product: Customization.*

Appendix E. DFP ISMF Services

Introduction

This appendix describes services of the ISMF component of DFP available for use by external application programs that are executing as ISMF line operators or commands. See *Data Facility Product: Customization* for details related to creating new line operators or commands. This appendix is organized into the following sections:

- DFP ISMF service descriptions
- DFP ISMF messages available to external applications
- DFP ISMF control blocks available to external applications

Standard Linkage and Error Handling

ISMF uses registers as described below. Services are accessed by creating required parameter lists, loading the entry point address of the desired routine into register 15, then branching to the address in register 15. The first four letters of all entry point names identify associated control blocks.

Input Register Usage

The following table describes registers on input to a line operator or command running in a DFP ISMF environment.

Register	Value on Input to Line Operator or Command
0	Unused
1	LPPL on input to line operator; CPPL on input to command
2-9	Unused
10	Points to GDRB
11, 12	Unused
13	Points to standard save area of calling routine
14	Address of return point in calling routine

Register	Value on Input to Line Operator or Command
15	Address of called routine

The following table describes registers on input to a DFP ISMF service from a line operator or command.

Register	Value on Input to DFP ISMF Service
0	Unused unless otherwise specified below
1	Points to input parameter list
2-9	Unused unless otherwise specified below
10	Points to GDRB
11, 12	Unused unless otherwise specified below
13	Points to standard save area of calling routine
14	Address of return point in calling routine
15	Address of called routine

Output Register Usage

The following table describes registers on output from a DFP ISMF service to a line operator or command, or on output from a line operator or command to ISMF.

Register	Value on Output from DFP ISMF Service, Line Operator, or Command
0-14	Unused unless otherwise specified below
15	Return code

Building Parameter Lists and Finding Control Blocks

Parameter lists must be constructed in one of two ways:

- The list consists of 31-bit pointers to required inputs, with the last entry of the list marked by having the high-order bit set to 1.
- The list consists of a set of flag bits followed by 31-bit pointers. The flag bits indicate which parameters are present and which are omitted.

Most control blocks are pointed to by fields in other control blocks. To find a control block, use the entry point name provided and find the control block whose name is equal to the first four letters of the given entry point name. The rest of the given entry point name identifies the field in the control block (identified by the first four letters) that contains the entry point.

For example, the control block ARVT identifies the field GDRBARVT as its entry point. Following the rules stated above, GDRB is the control block that contains the entry point address for the ARVT in the field labeled GDRBARVT (offset 8). The GDRB lists register 10 as its entry point; therefore, you find the entry point for the ARVT in the field at location GDRB + 8. Some fields in parameter lists refer to fields in control blocks; you may locate these fields in a similar manner.

Error Logging

If an error occurs, all services write information to the ERNT (see service descriptions for details). Typical error logging includes the failing module name, return code, reason code, and other diagnostic information. DFP ISMF creates a chain of ERNT control blocks to log errors. The ERTBCURR field in the ERTB always points to the current ERNT.

DFP ISMF Services

DGTCDT01 Decrement Use Count

Function: DGTCDT01 decrements the use count of modules loaded by DGTCLD01.

Entry Point Address: SRVTDT01

Input: R1 points to the parameter list described below.

Parameter list

Offset Hex	Parameter
0	31-bit pointer to an 8-byte character variable set to the name of the module to be deleted

Output: DGTCDT01 provides a return code, as follows.

Register states: R15 contains a return code, and the ERNT is updated as described below.

Return Code	ERNT Field Name
0	
8	ERNTMODN ERNTPD ERNTPROC ERNTRC ERNTRSNC

DGTCDV01 Data Set Name Syntax Verification

Function: DGTCDV01 verifies data set name syntax. If the variable pointed to by the last parameter in the following list is set to a blank (X'40'), unquoted input data set names are returned unquoted with the TSO prefix appended. Quoted input data set names are returned unchanged, but without quotation marks.

Entry Point Address: SRVTDV01

Input: R1 points to the parameter list described below.

Parameter list

Offset Hex (Dec)	Parameter
0	31-bit pointer to a 60-byte character variable set to input data set name
4	31-bit pointer to a 1-byte character variable set to Y (allow) or blank (disallow) full data set name
8	31-bit pointer to a 1-byte character variable set to Y (allow) or blank (disallow) partial data set name
C (12)	31-bit pointer to a 1-byte character variable set to Y (allow) or blank (disallow) member of partitioned data set
10 (16)	31-bit pointer to a 1-byte character variable set to Y (allow) or blank (disallow) relative generation reference data set
14 (20)	31-bit pointer to a 60-byte character variable to be set to output data set name.
18 (24)	31-bit pointer to a 1-byte character variable set to Y (allow) or blank (disallow) the following: TSO prefix should not be appended to data set name and data set name must be unquoted

Output: The output data set name is updated.

Register states: R15 contains a return code, and the ERNT fields listed below are updated.

Return Code	ERNT Field Name
0	
8	ERNTLMSG ERNTMODN ERNTPROC ERNTRSNC ERNTRC ERNTSMSG

Return Code	ERNT Field Name
12	ERNTLMSG ERNTMODN ERNTPROC ERNTRSNC ERNTRC ERNTSERV ERNTSMSG

DGTCEP01 Free Storage and Exit

Function: DGTCEP01 frees storage gotten by DGTCPR01 and exits to the caller of the routine that called DGTCEP01.

Entry Point Address: SRVTEP01

Input: R1 points to the parameter list described below.

Parameter list

Register	Parameter
0	31-bit pointer to a 15-bit variable set to the ISMF exit return code
1	31-bit pointer to an 8-byte character variable set to the name of the calling module
2	31-bit pointer to storage to be freed

Output: DGTCEP01 provides a return code, as follows.

Register states: R15 contains a return code, and the following ERNT fields are updated. R0 contains a pointer to the return code to be passed back to the original caller (caller of the caller of DGTCEP01). R1 contains a pointer to the size of the invoking module's automatic data area.

Return Code	ERNT Field Name
0	
12	ERNTMODN ERNTPROC ERNTRSNC ERNTRC

DGTCFM01 Free Memory

Function: DGTCFM01 frees storage acquired through DGTCGM01.

Entry Point Address: SRVTFM01

Input: R1 points to the parameter list described below.

Parameter list

Offset Hex (Decimal)	Parameter
0	31-bit pointer to a 4-byte character variable set to the storage identifier of the caller
4	31-bit pointer to a 31-bit pointer to the address of the area to be freed

Output: DGTCFM01 provides a return code, as follows.

Register states: R15 contains a return code and the following ERNT fields are updated.

Return Code	ERNT Field Name
0	
12	
	ERNTMODN
	ERNTPD
	ERNTPROC
	ERNTRSNC
	ERNTRC
	ERNTSERV

DGTCGM01 Get Storage

Function: DGTCGM01 gets storage (which is freed by DGTCFM01). If the first attempt to obtain storage fails, DGTCGM01 deletes modules (loaded by DGTCLD01) whose use count is zero and will retry the operation.

Entry Point Address: SRVTGM01

Input: R1 points to the parameter list described below.

Parameter list

Offset Hex (Decimal)	Parameter
0	31-bit pointer to a 31-bit variable set to number of storage bytes to be obtained
4	31-bit pointer to a 5-byte character variable set to BLANK or ZERO, specifying whether to initialize obtained storage to zeros or blanks
8	31-bit pointer to a 4-byte character variable set to the storage identifier of the obtained storage
C (12)	31-bit pointer to a 5-byte character variable set to BELOW or ANY, specifying storage to be obtained from BELOW the 16 megabyte boundary or from ANY location
10 (16)	31-bit pointer to a 5-byte character variable set to DBLWD or PAGE, specifying storage to be aligned on a doubleword or page boundary
14 (20)	31-bit pointer to a 31-bit variable to be set to the address of the obtained storage

Output: The last field of the parameter list (*address of obtained storage*) is updated to contain a pointer to the new storage.

Register states: R15 contains a return code and the following ERNT fields are updated.

Return Code	ERNT Field Name
0	
12	ERNTMODN ERNTPD ERNTPROC ERNTRSNC ERNTRC ERNTSERV

DGTCLD01 Load a Module

Function: DGTCLD01 loads a requested module into storage; if the module is already loaded, DGTCLD01 increments the module's use count.

Entry Point Address: SRVTLD01

Input: Register states: R1 points to the parameter list described below.

Parameter list

Offset Hex (Decimal)	Parameter
0	31-bit pointer to an 8-byte character variable set to the name of the requested module
4	31-bit pointer to a 31-bit variable that will be updated to contain the entry point address of the loaded module

Output: DGTCLD01 provides a return code, as follows, and the variable pointed to by the *load entry point address* field of the parameter list is updated.

Register states: R15 contains a return code and the ERNT is updated as described below.

Return Code	ERNT Field Name
0	
12	ERNTMODN ERNTPROC ERNTRC ERNTRSNC SRVC ERNTPD

DGTCLG01 Place Information in the ISPF Log

Function: DGTCLG01 creates log entries based on information from the current ERNT, the current IMNT, or the log buffer. If the first parameter in the following list is set to ERROR, the ERNT is used as the information source; if the first parameter is set to IMTT, the IMNT is used as the source, and if the first parameter is set to STATUS, the log buffer is used as the source.

Entry Point Address: SRVTLG01

Input: R1 points to the parameter list described below.

Parameter list

Offset Hex	**Parameter**
0	31-bit pointer to a 6-byte character variable set to the type of entry to be generated. Possible entries: ERROR, STATUS, IMTT
4	31-bit pointer to an 8-byte character variable set to the address of a message to be written to the log if type was set to STATUS, set to blanks otherwise
8	31-bit pointer to a 31-bit variable set to zeros

Output: DGTCLG01 provides a return code, as follows.

Register states: R15 contains a return code and the following ERNT fields are updated.

Return Code	**ERNT Field Name**
0	
12	ERNTMODN ERNTPD ERNTPROC ERNTRSNC ERNTRC ERNTSERV

DGTCPR01 Obtain Automatic Data Area

Function: If tracing is on, DGTCPR01 gets a module's automatic data area, adds an entry to the *Inter-Module Trace Table*, and writes an entry to the ISMF log.

Entry Point Address: SRVTPR01

Input: R1 points to the parameter list described below.

Parameter list

Register	Parameter
0	31-bit pointer to a 31-bit variable set to the amount of storage needed
1	31-bit pointer to an 8-byte character variable set to calling module name

Output: DGTEPR01 provides a return code, as follows.

Register states: R15 contains a return code, R1 contains the address of the new storage, and the following ERNT fields are updated.

Return Code	ERNT Field Name
0	
12	ERNTMODN ERNTPROC ERNTRSNC ERNTRC

DGTCVV01 Volume Serial Number Syntax Verification

Function: DGTCVV01 verifies volume serial number syntax. DGTCVV01 does not support quoted volume serial numbers.

Entry Point Address: SRVTVV01

Input: R1 points to the parameter list described below.

Parameter list

Offset Hex	Parameter
0	31-bit pointer to a 6-byte character variable set to the volume serial number to be checked
4	31-bit pointer to a 1-byte character variable set to Y (allow) or blank (disallow) full volume serial number
8	31-bit pointer to a 1-byte character variable set to Y (allow) or blank (disallow) partial volume serial number

Output: DGTCVV01 provides a return code, as follows.

Register states: R15 contains a return code and the following ERNT fields are updated.

Return Code	ERNT Field Name
0	
8	ERNTLMSG ERNTMODN ERNTPROC ERNTRSNC ERNTRC ERNTSMSG

DGTCWO01 Word Finder

Function: DGTCWO01 scans a specified section of storage for a character string delimited by blanks.

Entry Point Address: SRVTWO01

Input: R1 points to the parameter list described below.

Parameter list

Offset Hex (Decimal)	Parameter
0	31-bit pointer to a 31-bit variable set to the start address of storage to be scanned
4	31-bit pointer to a 31-bit variable set to the end address of storage to be scanned
8	31-bit pointer to a 31-bit variable that will contain the address of the first word found
C (12)	31-bit pointer to a 31-bit variable that will contain the size of the first word found

Output: The *word* and *size* fields of the parameter list will be updated.

Register states: R15 contains a return code and the following ERNT fields are updated.

Return Code	ERNT Field Name
0	
12	ERNTMODN ERNTPROC ERNTRSNC ERNTRC

DGTFARF1 Find an Entry in the Data Set List Array

Function: DGTFARF1 finds information in the data set list array and moves the current row pointer to the new position.

Entry Point Address: ARVTFIND

Input: Register states: R1 points to the parameter list described below.

Parameter list

Offset Hex	Parameter
0	31-bit pointer to a 32-bit parameter set to X'3', indicating which parameters are present
4	31-bit pointer to the SELB
8	31-bit pointer to a 3-byte character variable set to YES, specifying retrieve information, or NO, specifying find information only.

Output If the *retrieve information* field of the parameter list specified YES, DGTFARF1 updates variables addressed by the DABVAR fields. DGTFARF1 provides a return code, as follows:

Register states: R15 contains a return code and the ERNT is updated as described below.

Return Code	ERNT Field Name
0	
8	ERNTLMSG ERNTMODN ERNTPD ERNTPROC ERNTRC ERNTRSNC ERNTSERV ERNTSMSG
12	ERNTLMSG ERNTMODN ERNTPD ERNTPROC ERNTRC ERNTRSNC ERNTSERV ERNTSMSG

DGTFARP1 Position Current Row Pointer at Top of List

Function: DGTFARP1 positions the current row pointer at the top of the data set list array. The data set list array is an internal representation of data set names.

Entry Point Address: ARVTPOS

Input: Register states: R1 points to the parameter list described below.

Parameter list

Offset Hex (Decimal)	Parameter
0	31-bit pointer to a 32-bit parameter set to X'19', indicating which parameters are present
4	31-bit pointer set to the value of CPPLARNM
8	31-bit pointer to a 31-bit variable set to zeros
C (12)	31-bit pointer to a 31-bit variable set to zeros
10 (16)	31-bit pointer to a 3-byte character variable set to 'TOP'
14 (20)	31-bit pointer to a 2-byte character variable set to 'NO'
18 (24)	31-bit pointer to a 31-bit variable set to zeros
1C (28)	31-bit pointer to a 31-bit variable set to zeros

Output:: DGTFARP1 provides a return code, as follows.

Register states: R15 contains a return code and the ERNT is updated as described below.

Return Code	ERNT Field Name
0	
8	ERNTLMSG ERNTMODN ERNTPROC ERNTRC ERNTRSNC ERNTSERV ERNTSMSG
12	ERNTLMSG ERNTMODN ERNTPROC ERNTRC ERNTRSNC ERNTSERV ERNTSMSG

DGTFARS1 Obtain Count of Data Sets

Function: DGTFARS1 obtains the total number of displayable data sets from the data set list array.

Entry Point Address: ARVTSTAT

Input: Register states: R1 points to the parameter list described below.

Parameter list

Offset Hex (Decimal)	Parameter
0	31-bit pointer to a 32-bit string set to X'F', indicating which parameters are present
4	31-bit pointer set to the value in CPPLARNM
8	31-bit pointer to a 6-byte character variable set to 'RETURN'
C (12)	31-bit pointer to a 7-byte character variable set to 'DISPROW'
10 (16)	31-bit pointer to a 31-bit variable where DGTFARS1 will pass back count of displayable data sets
1D (20)	31-bit pointer to a 31-bit variable set to zeros

Output: DGTFARS1 updates the variable pointed to by the *count* field, and provides a return code as follows:

Register states: R15 contains a return code and the ERNT is updated as described below.

Return Code	ERNT Field Name
0	
8	ERNTLMSG ERNTMODN ERNTPD ERNTPROC ERNTRC ERNTRSNC ERNTSERV ERNTSMSG
12	ERNTLMSG ERNTMODN ERNTPD ERNTPROC ERNTRC ERNTRSNC ERNTSERV ERNTSMSG

DGTFARU1 Update Data Set List Array

Function: DGTFARU1 updates columns for the entry in the data set list array pointed to by the current row pointer.

Entry Point Address: ARVTUPDT

Input: Register states: R1 points to the parameter list described below.

Parameter list

Offset Hex (Decimal)	Parameter
0	31-bit pointer to a 32-bit parameter set to X'3', indicating which parameters are present
4	31-bit pointer set to the value of CPPLARNM
8	31-bit pointer to the DABL
C (12)	31-bit pointer to a 31-bit variable set to zeros

Output: DGTFARU1 updates the data set list array and provides a return code as follows:

Register states: R15 contains a return code and the ERNT is updated as described below.

Return Code	ERNT Field Name
0	
8	ERNTLMSG ERNTMODN ERNTPD ERNTPROC ERNTRC ERNTRSNC ERNTSERV ERNTSMSG
12	ERNTLMSG ERNTMODN ERNTPD ERNTPROC ERNTRC ERNTRSNC ERNTSERV ERNTSMSG

DGTFCTCK Verify Commands

Function: DGTFCTCK checks the validity of commands in the CTAP. Commands in the CTAP must be enabled by DGTFCTSE before use.

Entry Point Address: CTVTCTCK

Input: R1 points to the parameter list described below.

Parameter list

Offset Hex (Decimal)	Parameter
0	31-bit pointer to a 31-bit pointer to the command line field
4	31-bit pointer to a 31-bit variable set to the command line length
8	31-bit pointer to a 1-byte character variable set to a blank (X'40')
C (12)	31-bit pointer to an 8-byte character variable where DGTFCTCK will pass back the name of the load module which processes the named command
10 (16)	31-bit pointer to an 8-byte character variable where DGTFCTCK will pass back the name of the command

Output: DGFFCTCK updates the *command routine (load module)* and *command name* fields, and provides a return code as follows:

Register states: R15 contains a return code and the following ERNT fields are updated.

Return Code	ERNT Field Name
0	
8	ERNTLMSG ERNTRSNC ERNTSMSG
12	ERNTLMSG ERNTRSNC ERNTSMSG

DGTFCTPR Process commands

Function: DGTFCTPR gives control to the load module that processes input commands. You must create the CTPL in order to use DGTFCTPR.

Entry Point Address: CTVTCTPR

Input: R1 points to the parameter list described below.

Parameter list

Offset Hex	Parameter
0	31-bit pointer to the CTPL

Output: DGTFCTPR provides a return code, as follows.

Register states: R15 contains a return code and the following ERNT fields are updated.

Return Code	ERNT Field Name
0	
8	ERNTLMSG ERNTRSNC ERNTSMSG
12	ERNTLMSG ERNTRSNC ERNTSMSG

DGTFCTSE Enable Valid Commands

Function: DGTFCTSE, by updating the CTAP, enables all commands listed in the CTFU and disables all others. others by updating the CTAP. You must create the CTFU to use DGTFCTSE.

Entry Point Address: CTVTCSE

Input: Register states: R1 points to the parameter list described below.

Parameter list

Offset Hex	Parameter
0	31-bit pointer to the CTFU

Output: DGTFCTSE provides a return code, as follows.

Register states: R15 contains a return code and the ERNT is updated as described below.

Return Code	ERNT Field Name
0	
8	ERNTLMSG ERNTRSNC ERNTSMSG
12	ERNTLMSG ERNTRSNC ERNTSMSG

DGTFFOE1 Obtain Input Information From the Screen Area Image

Function: DGTFFOE1 obtains input information from the specified entry in the list of data sets displayed in the screen area image.

Entry Point Address: FOVTGTVL

Input: Register states: R1 points to the parameter list described below.

Parameter list

Offset Hex (Decimal)	Parameter
0	31-bit pointer to a 32-bit parameter set to X'F', indicating which parameters are present
4	31-bit variable set to the value of LPPLFOAD
8	31-bit variable set to the value of LPPLARAD
C (12)	31-bit pointer to the address of LPPLENT
10 (16)	31-bit pointer to the DABL
14 (20)	31-bit pointer to a 31-bit variable set to zeros.
18 (24)	31-bit pointer to a 31-bit variable set to zeros.
1C (28)	31-bit pointer to a 31-bit variable set to zeros.

Output: DGTFFOE1 updates the variables addressed by the DABVAR, and provides a return code as follows:

Register states: R15 contains a return code and the ERNT is updated as described below.

Return Code	ERNT Field Name
0	
12	ERNTMODN ERNTPROC ERNTRC ERNTRSNC

DGTFFOL1 Refresh the Screen Image Area From the Data Set List Array

Function: DGTFFOL1 uses the data set list array to refresh the screen image area.

Entry Point Address: FOVTLOAD

Input: Register states: R1 points to the parameter list described below.

Parameter list

Offset Hex	Parameter
0	31-bit pointer to a 32-bit parameter set to X'3', indicating which parameters are present
4	31-bit pointer set to the value of CPPLFONM
8	31-bit pointer set to the value of CPPLARNM

Output: DGTFFOL1 provides a return code, as follows.

Register states: R15 contains a return code and the ERNT is updated as described below.

Return Code	ERNT Field Name
0	
8	ERNTMODN ERNTPROC ERNTRC ERNTRSNC
12	ERNTMODN ERNTPROC ERNTRC ERNTRSNC

DFP Common Services

IGBDIS00 Call Device Information Services for UCB Address

Function: IGBDIS00 returns the address of the first UCB found for the generic device type passed as input. Whenever IGBDIS00 provides return code 8, you must re-call the function, passing the same parameter list you used on the first call. Do not initialize any fields as indicated in the following instructions for making the first call to the service; make the second call with the parameter *exactly* as it is returned from the first call. The second call to IGBDIS00 frees storage acquired by the function on the first call.

Entry Point Address: CVTEXT2 + 12; resulting address + 36.

Input: R0 contains zeros and R1 contains a pointer to a pointer to the parameter list described below. You must create the parameter list described below; it must be 60 bytes long and you must fill in only the indicated fields; you must set the rest to zeros. You must also create a 32-byte *problem determination area* and a 25-byte *return area*, both initialized to zeros.

Parameter list

Offset Hex (Decimal)	Parameter
0	X'003C0100'
6	X'10'
7	X'00' (first call); X'02' (second call)
8	31-bit pointer to return area
C (12)	31-bit variable set to the return area length
10 (16)	31-bit pointer to the problem determination area
14 (20)	31-bit variable set to the problem determination area length
28 (40)	8-byte character variable set to the generic device type name

Output: IGBDIS00 provides the UCB address in the *return area* + 20, and, if an error occurred, updates the *problem determination* area. It also provides a return code as follows.

Register states: R15 contains one of the following hex return codes: 0, 8 (successful), 10, 1C, 20, 3C (unsuccessful). See *DFP Diagnosis* for a description of the problem determination area and details about the valid return codes and reason codes.

DFP ISMF Messages Available to External Applications

The following messages are in the data set 'SYS1.DGTMLIB', with member names identified by the full message name minus the last character of the message name. For example, message DGTUV005 is in 'SYS1.DGTMLIB(DGTUV00)' Likewise, message DGTUV016 is in 'SYS1.DGTMLIB(DGTUV01)' and so on. Each member contains 10 messages, numbered 0 through 9.

```
DGTUV005   'ENTER Y OR N'              .HELP= DGTMUV05   .ALARM= YES
'ENTER Y OR N AT THE CURSOR POSITION

DGTUV006   'INVALID COMBINATION'       .HELP= DGTMUV06   .ALARM= YES
'IF MAXIMUM NUMBER OF RETRIES=0 THEN SECONDS BETWEEN RETRIES MUST BE 0

DGTUV007   'INVALID COMBINATION'       .HELP= DGTMUV07   .ALARM= YES
'IF SECONDS BETWEEN RETRIES=0 THEN MAXIMUM NUMBER OF RETRIES MUST BE 0

DGTUV016   'MUST BE BLANK'             .HELP= DGTMUV16   .ALARM= YES
'AMOUNT OF I/O BUFFERING MUST NOT BE SPECIFIED IF DUMP IN COMPRESSED FORM = Y

DGTUV021   'SPECIFY ONE OR MORE'       .HELP= DGTMUV21   .ALARM= YES
'YOU MUST SPECIFY ONE OR MORE OF THE FIELDS BELOW

DGTUV024   'ISMF INTERNAL ERROR'       .HELP= DGTMUV24   .ALARM= YES
'ISMF INTERNAL PROCESSING ERROR

DGTUV031   'ENTER 1 OR 2'              .HELP= DGTMUV31   .ALARM= YES
'ENTER 1 OR 2 AT THE CURSOR POSITION

DGTUV032   'RESET MUST EQUAL N'        .HELP= DGTMUV32   .ALARM= YES
'RESET = Y IS INVALID IF ACCESS SOURCE DATA SET IN SHARED MODE = Y

DGTUV037   ''                                            .ALARM= NO
'&LOGBJBNM(&LOGBJBNO) SUBMITTED

DGTUV038   ''                                            .ALARM= NO
'&LOGBJBNM PLACED IN &LOGBJBDS

DGTUV040   ''                                            .ALARM= NO
'&LOGBJBNM ADDED TO &LOGBJBDS

DGTUV045   '&CURLIOP FAILED'           .HELP= DGTMUV45   .ALARM= YES
'&CURLIOP LINE OPERATOR FAILED DUE TO AN UNEXPECTED ISMF INTERNAL ERROR

DGTUV046   '&CURCMD FAILED'            .HELP= DGTMUV46   .ALARM= YES
'&CURCMD COMMAND FAILED DUE TO AN UNEXPECTED ISMF INTERNAL ERROR

DGTUV048   'COMMAND FAILED'            .HELP= DGTMUV48   .ALARM= YES
'EITHER THE COMMAND FAILED OR AN UNEXPECTED INTERNAL ISMF ERROR OCCURRED

DGTUV049   'MISSING VOLSER'            .HELP= DGTMUV49   .ALARM= YES
'AT LEAST ONE VOLUME SERIAL NUMBER MUST BE SPECIFIED

DGTUV050   'MISSING UNIT TYPE'         .HELP= DGTMUV50   .ALARM= YES
'UNIT TYPE MUST BE SPECIFIED FOR VOLUME SERIAL(S)
```

Figure 46 (Part 1 of 3). DFP ISMF Messages Available to External Applications

```
DGTUV051   '&CURCMD INVALID'           .HELP= DGTMUV51   .ALARM= YES
'&CURCMD IS ONLY VALID WHEN ENTERED FROM THE LIST PANEL

DGTUV052   'UNABLE TO FIND VOLSER'     .HELP= DGTMUV52   .ALARM= YES
'THE ACTUAL VOLUME SERIAL FOR THE INDIRECT VOLSER COULD NOT BE FOUND

DGTUV053   'UPDATE OF LIST FAILED'     .HELP= DGTMUV53   .ALARM= YES
'BACKGROUND JOB SUCCESSFULLY CREATED BUT UPDATE OF LIST FAILED

DGTUV054   'INVALID STATUS'            .HELP= DGTMUV54   .ALARM= YES
'IF NO OUTPUT VOLUMES OR UNIT SPECIFIED, STATUS MUST BE OLD

DGTUV055   'MISSING DEVICE TYPE'       .HELP= DGTMUV55   .ALARM= YES
'WHEN STATUS OF DATA SET IS NEW, AT LEAST A VALUE FOR UNIT MUST BE SPECIFIED

DGTUV056   'MUST BE 3 OR BLANK'        .HELP= DGTMUV56   .ALARM= YES
'WHEN REPLACE IF DUPLICATE REQUESTED, DO NOT RENAME DATA SET

DGTUV057   'DUPLICATE DATA SET NAME'   .HELP= DGTMUV57   .ALARM= YES
'DUPLICATE DATA SET NAME INVALID FOR LIST COMMAND

DGTUV062   'PANEL PRIMING FAILED'      .HELP= DGTMUV62   .ALARM= YES
'UNABLE TO RETRIEVE SAVED PANEL VALUES - LAST-USE MODE IGNORED

DGTUV063   'PANEL PRIMING FAILED'      .HELP= DGTMUV63   .ALARM= YES
'UNABLE TO RETRIEVE SAVED PANEL VALUES

DGTUV067   '&CURLIOP INVALID'          .HELP= DGTMUV67   .ALARM= YES
'&CURLIOP INVALID FOR A DFHSM MIGRATED DATA SET

DGTUV068   'MIGRATED DATASET INVALID' .HELP= DGTMUV68   .ALARM= YES
'DFHSM MIGRATED DATA SETS ARE INVALID FOR THE &CURCMD LIST COMMAND

DGTUV069   'TOO MANY VOLSERS'          .HELP= DGTMUV69   .ALARM= YES
'MORE THAN 255 DIFFERENT VOLSERS IN THE DATA SET LIST

DGTUV070   '&CURCMD JOB CREATED'       .HELP= DGTMUV70   .ALARM= NO
'BACKGROUND JOB SUCCESSFULLY CREATED FOR THE &CURCMD COMMAND

DGTUV071   'INVALID VOLSER'            .HELP= DGTMUV71   .ALARM= YES
'VALID VOLSER REQUIRED TO PERFORM THE REQUESTED FUNCTION

DGTUV072   'ENTER REQUIRED FIELD'      .HELP= DGTMUV72   .ALARM= YES
'WHEN SPACE UNITS IS BLKS, A VALUE FOR BLOCKSIZE IS REQUIRED

DGTUV073   'ENTER REQUIRED FIELD'      .HELP= DGTMUV73   .ALARM= YES
'WHEN STATUS IS NEW, SPACE QUANTITIES ARE REQUIRED

DGTUV080   'INVALID DATA SET NAME'     .HELP= DGTMUV80   .ALARM= YES
'THE LINE OPERATOR FAILED BECAUSE IT WAS ISSUED AGAINST A BLANK DATA SET NAME

DGTUV082   'BLANK NAME INVALID'        .HELP= DGTMUV82   .ALARM= YES
'BLANK DATA SET NAMES ARE INVALID FOR LIST COMMANDS

DGTUV083   'INVALID DEVICE TYPE'       .HELP= DGTMUV83   .ALARM= YES
'VALID DEVICE TYPE REQUIRED TO PERFORM THE REQUESTED FUNCTION
```

Figure 46 (Part 2 of 3). DFP ISMF Messages Available to External Applications

```
DFQHA001   'RANGE 0 TO 999'              .HELP= DFQHA001   .ALARM= YES
'SPECIFY A NUMBER IN THE RANGE OF 0 TO 999

DFQHA002   'RANGE 0 TO 13'               .HELP= DFQHA002   .ALARM= YES
'SPECIFY A NUMBER IN THE RANGE OF 0 TO 13

DFQHA003   'MUTUALLY EXCLUSIVE FIELDS'   .HELP= DFQHA003   .ALARM= YES
'CANNOT SPECIFY NO. OF DAYS OR NO. OF BACKUP VERSIONS FOR SYSTEM DEFAULT

DFQHR001   'INVALID VOLSER'              .HELP= DFQHR001   .ALARM= YES
'CONTAINS INVALID CHARACTERS

DFQHR002   'ENTER REQUIRED INPUT'        .HELP= DFQHR002   .ALARM= YES
'VOLUME AND DEVICE TYPE MUST BOTH BE SPECIFIED, OR SPECIFY NEITHER

DFQHR003   'INVALID DEVICE TYPE'         .HELP= DFQHR003   .ALARM= YES
'THE DEVICE TYPE MUST BE 3330, 3330-1, 3330v, 3350, 3375, or 3380

DFQHC001   'RANGE 0 TO 12'               .HELP= DFQHC001   .ALARM= YES
'SPECIFY A NUMBER IN THE RANGE OF 0 TO 12

DFQHM001   'INVALID HMIGRATE COMMAND     .HELP= DFQHM001   .ALARM= YES
'CANNOT BE MIGRATED TO LEVEL-1 FOR A TAPE ONLY MIGRATE SYSTEM

DFQHM002   'INVALID HMIGRATE COMMAND     .HELP= DFQHM002   .ALARM= YES
'HMIGRATE CANNOT BE ISSUED FOR A LEVEL-2 TO A LEVEL-1

DFQHM003   'INVALID HMIGRATE COMMAND     .HELP= DFQHM003   .ALARM= YES
'HMIGRATE CANNOT BE ISSUED FOR A LEVEL-1 TO A LEVEL-1

DFQHD001  'INVALID HBDELETE COMMAND'     .HELP= DFQHD001   .ALARM= YES
'THERE ARE NO BACKUP VERSIONS TO DELETE

DFQCN001  'INVALID CONDENSE COMMAND'     .HELP= DFQCN001   .ALARM= YES
'CONDENSE CANNOT BE ISSUED FOR A MIGRATED DATA SET

DFQCN002  'INVALID CONDENSE COMMAND'     .HELP= DFQCN002   .ALARM= YES
'CONDENSE CANNOT BE ISSUED FOR A TAPE ONLY MIGRATE SYSTEM

DFQCN003  'INVALID CONDENSE COMMAND'     .HELP= DFQCN003   .ALARM= YES
'CONDENSE CAN ONLY BE ISSUED FOR A DATA SET WITH DSORG OF PS OR PO

DFQCN004  'DSORG CANNOT BE OBTAINED'     .HELP= DFQCN004   .ALARM= YES
'ERROR DURING THE OBTAIN OF THE FORMAT1 DSCB TO DETERMINE DSORG
```

Figure 46 (Part 3 of 3). DFP ISMF Messages Available to External Applications

The following messages are not displayed on a line operator entry panel, but are written in the ISPF log as problem determination aids.

```
DFQLP001  'READ FOR DFHSM RECORD FAILED' .HELP= DFQLP001   .ALARM= YES
'THE READ FOR THE DFHSM CDS RECORD FAILED, SVC ERROR

DFQLP002  'LINE OPERATOR FAILED'         .HELP= DFQLP002   .ALARM= YES
'LINE OPERATOR FAILED

DFQLP003  'READ FOR DFHSM RECORD FAILED' .HELP= DFQLP003   .ALARM= YES
'THE READ FOR THE DFHSM CDS RECORD FAILED, DFHSM ERROR

DFQLP004  'SUCCESSFUL SUBMISSION'        .HELP= DFQLP004   .ALARM= YES
'lineop LINE OPERATOR SUBMITTED SUCCESSFULLY
```

Figure 47. DFP ISMF Problem Determination Messages

DFP ISMF Control Blocks Available to External Applications

This section describes DFP ISMF control blocks available for use by external applications. Most control blocks are pointed to by fields in other control blocks. To find a control block, use the entry point name provided and find the control block whose name is equal to the first four letters of the given entry point name. The rest of the given entry point name identifies the field in the control block (identified by the first four letters) which contains the entry point.

For example, the control block ARVT identifies the field GDRBARVT as its entry point. Following the rules stated above, GDRB is the control block that contains the entry point address for the ARVT in the field labeled GDRBARVT (offset 8). The GDRB lists register 10 as its entry point; therefore, you find the entry point for the ARVT in the field at location GDRB + 8.

ARVT

Entry Point: GDRBARVT

```
ARVT
Offsets      Type        Length Name            Description
============================================================================
      ARRAY SERVICE VECTOR TABLE (ARVT)
============================================================================
    0      (0) CHARACTER    56  ARVT
    0      (0) CHARACTER     4  ARVTVID         VISUAL ID: 'ARVT'
    4      (4) FIXED         2  ARVTLEN         LENGTH OF ARVT
    6      (6) FIXED         2  ARVTUCNT        USE COUNT
    8      (8) ADDRESS       4  ARVTBEG         ADDRESS OF ARBEGIN MODULE
   12      (C) ADDRESS       4  ARVTDONE        ADDRESS OF ARDONE MODULE
   16     (10) ADDRESS       4  ARVTADD         ADDRESS OF ARADD MODULE
   20     (14) ADDRESS       4  ARVTCRET        ADDR. OF ARCREATE MODULE
   24     (18) ADDRESS       4  ARVTDELR        ADDR. OF ARDELROW MODULE
   28     (1C) ADDRESS       4  ARVTENDM        ADDRESS OF AREND MODULE
   32     (20) ADDRESS       4  ARVTFIND        ADDRESS OF ARFIND MODULE
   36     (24) ADDRESS       4  ARVTPOS         ADDRESS OF ARPOSIT MODULE
   40     (28) ADDRESS       4  ARVTSORT        ADDRESS OF ARSORT MODULE
   44     (2C) ADDRESS       4  ARVTSTAT        ADDRESS OF ARSTATS MODULE
   48     (30) ADDRESS       4  ARVTUPDT        ADDR. OF ARUPDATE MODULE
   52     (34) ADDRESS       4  ARVTTBLS        ADDR. OF ROWID CNTRL BLK
   56     (38) CHARACTER        ARVTEND         END OF ARVT
```

CLCB

Entry Point: LPPLCLCB/CPPLCLCB (LPPLCLCB for line operators, CPPLCLCB for commands).

CLCB

COMMAND/LINE OPERATOR CONTROL BLOCK

Offsets		Type	Length	Name	Description
0	(0)	CHARACTER	*	CLCB	
0	(0)	CHARACTER	4	CLCBVID	VISUAL ID: 'CLCB'
4	(4)	FIXED	2	CLCBLEN	LENGTH OF CLCB
6	(6)	BITSTRING	1	CLCBFLG1	FLAG FIELD
7	(7)	BITSTRING	1	CLCBFLG2	FLAG FIELD
8	(8)	ADDRESS	4	CLCBNEXT	ADDR OF ENXT CLCB
12	(C)	ADDRESS	4	CLCBPREV	ADDR. PREV CLCB
16	(10)	CHARACTER	*	CLCBENT	CLCB ENTRY

CONH

Entry Point: SELBCONH

If you are using DGTFARF1, the CONHCNT field must be a 15-bit variable set to the number of DABs whose column values are used to search the data set list array. The CONHCONE must be a 31-bit pointer to each CONE associated with the CONH.

CONH

CONDITIONAL CONTROL BLOCK HEADER (CONH)

Offsets		Type	Length	Name	Description
0	(0)	CHARACTER	*	CONH	
0	(0)	CHARACTER	8	CONHMAIN	
0	(0)	CHARACTER	4	CONHVID	VISUAL ID: 'CONH'
4	(4)	FIXED	2	CONHLEN	LENGTH OF CONH
6	(6)	FIXED	2	CONHCNT	NUMBER OF ENTRIES
8	(8)	ADDRESS	4	CONHCONE(*)	CONE ADDRESSES

CONH

Entry Point: CONHCONE

If you are using DGTFARF1, you must set the CONENAME to a column name as specified in the DAB associated with the CONE; you must set the CONECOND to either 'EQ' or 'NE', used when comparing the value in the variable pointed to by the DABVAR in the DAB associated with this CONE to the column values in the data set list array.

CONH

END OF CONDITIONAL CONTROL BLOCK HEADER (CONH) CONDITIONAL CONTROL BLOCK ENTRY (CONE)

Offsets		Type	Length	Name	Description
0	(0)	CHARACTER	12	CONE	
0	(0)	CHARACTER	8	CONENAME	NAME OF COLUMN
8	(8)	CHARACTER	2	*	RESERVED UNUSED
10	(A)	CHARACTER	2	CONECOND	CONDITION
12	(C)	CHARACTER		CONEEND	END OF CONE

CPPL

Entry Point: R1 on entry to command routine.

```
CPPL
Offsets      Type      Length Name            Description
===========================================================================
      COMMAND PROCESSOR PARAMETER LIST (CPPL) PASSES TO THE COMMAND
      ROUTINE
===========================================================================
   0     (0) CHARACTER    88  CPPL
   0     (0) CHARACTER     4  CPPLVID         VISUAL ID: 'CPPL'
   4     (4) FIXED         2  CPPLLEN         LENGTH OF CPPL
   6     (6) BITSTRING     1  CPPLFLG1        FLAG FIELD
         1... ....            CPPLSTD         STD INVOCATION
         .1.. ....            CPPLTEOL        INVOKE AT END OF LIST
         ..1. ....            CPPLTEOA        INVOKE AT END OF APPLICAT
         ...1 ....            CPPLLOFL        ACTIVE LINE OPERATOR?
         .... 1...            CPPLSMSG        SET SHORT MSG DONE?
         .... .1..            CPPLSCNM        LIST GEN'D FRM CATALOG
         .... ..1.            CPPLSORT        SORT FLAG
         .... ...1            CPPLFILT        FILTER FLAG
   7     (7) CHARACTER     1  CPPLPNTP        PANEL TYPE
   8     (8) ADDRESS       4  CPPLARNM        ADDRESS OF FIELD W/ ARRAY
  12     (C) ADDRESS       4  CPPLFONM        ADDR. OF FIELD W/ FORMAT
  16    (10) ADDRESS       4  CPPLCLAD        ADDR. OF THE COMMAND LINE
  20    (14) FIXED         4  CPPLCMDL        COMMAND LINE LENGTH
  24    (18) CHARACTER     8  CPPLCMD         COMMAND
  32    (20) ADDRESS       4  CPPLCMAD        ADDR. OF COMMAND ON LINE
  36    (24) ADDRESS       4  CPPLPMAD        ADDRESS OF COMMAND PARAM.
  40    (28) ADDRESS       4  CPPLCSCB        ADDR. OF CURSOR CNTL BLK IT
                                              MAY BE EITHER THE ADDR OF PCCB
                                              OR LPCB
  44    (2C) ADDRESS       4  CPPLCSCR        CURRENT SCROLL AMNT ADDR.
  48    (30) ADDRESS       4  CPPLPSCR        PREVIOUS SCROLL AMNT ADDR
  52    (34) ADDRESS       4  CPPLCLCB        COMMAND/LINE OPERATOR CONTROL
                                              BLOCK CHAIN PTR
  56    (38) FIXED         4  CPPLERRW        ROWID NAME OF ROW W/ERROR
  60    (3C) ADDRESS       4  CPPLCNAM        ADDR OF CATALOG NAME
  64    (40) ADDRESS       4  CPPLDABL        ADDR OF DABL
  68    (44) ADDRESS       4  CPPLMODA        ADDR OF FILTER MOD NAME
  72    (48) CHARACTER     2  CPPLAPPL        APPLICATION ID
  74    (4A) BITSTRING     1  CPPLFLG2        FLAG FIELD
         1... ....            CPPLMCDF        HSM = YES ?
         .1.. ....            CPPLVOLF        VOL = YES ?
         ..11 1111            *               RESERVED, UNUSED
  75    (4B) CHARACTER     1  *               RESERVED, UNUSED
  76    (4C) ADDRESS       4  CPPLCNTN        ADDR OF THE NAME OF THE
                                              CATALOG NAME TABLE
  80    (50) ADDRESS       4  CPPLLMSG        ADDR OF THE LONG MSG
  84    (54) CHARACTER     4  *               RESERVED UNUSED
  88    (58) CHARACTER        CPPLEND         END OF CPPL
```

CTAP

Entry Point: CTABCTAP

CTAP Offsets		Type	Length	Name	Description
		COMMAND TABLE - APPLICATION TABLE (CTAP)			
0	(0)	CHARACTER	*	CTAP	
0	(0)	CHARACTER	8	CTAPMAIN	
0	(0)	CHARACTER	4	CTAPVID	VISUAL ID: 'CTAP'
4	(4)	FIXED	2	CTAPLEN	LENGTH OF CTAP
6	(6)	FIXED	2	CTAPCNT	# OF COMMAND ENTRIES
8	(8)	CHARACTER	28	CTAPENT(*)	
8	(8)	CHARACTER	8	CTAPNAME	COMMAND NAME
16	(10)	FIXED	1	CTAPTRUN	MIN. # OF CHARACTERS USED IN TRUNCATION
17	(11)	BITSTRING	1	CTAPFLAG	FLAG FIELD
	1...			CTAPST	COMMAND STATUS
	.1..			CTAPIMED	IMMEDIATE COMMAND
	..1.			CTAPLIST	LIST COMMAND
	...1 1111			CTAPRSVD	RESERVED
18	(12)	CHARACTER	8	CTAPRTNM	COMMAND ROUTINE NAME
26	(1A)	CHARACTER	8	CTAPTENM	CMD TERMINATION ROUTINE
34	(22)	BITSTRING	2	*	FILL UP END OF WORD

Constants

Length	Type	Value	Name	Description
	END OF COMMAND TABLE - APPLICATION TABLE (CTAP) DEFINED			
	COMMAND STATUS			
	BIT	1	CMDENABL	COMMAND STATUS IS ENABLE
	BIT	1	CMDDSABL	COMMAND STATUS IS DISABL

CTFU

Entry Point: Created by invoker of DGTFCTSE

CTFU Offsets		Type	Length	Name	Description
		COMMAND TABLE - FUNCTION TABLE (CTFU)			
0	(0)	CHARACTER	168	CTFU	
0	(0)	CHARACTER	8	CTFUMAIN	
0	(0)	CHARACTER	4	CTFUVID	VISUAL ID: 'CTFU'
4	(4)	FIXED	2	CTFULEN	LENGTH OF CTFU
6	(6)	FIXED	2	CTFUCNT	# OF COMMAND ENTRIES
8	(8)	CHARACTER	160	CTFUNMS	
8	(8)	CHARACTER	8	CTFUNAME(20)	COMMAND NAMES

CTPL

Entry Point: Created by invoker of DGTFCTPR

```
CTPL
Offsets      Type      Length Name            Description
============================================================================
      COMMAND TABLE PROCESSOR PARAMETER LIST (CTPL)
============================================================================
     0     (0) CHARACTER    80  CTPL
     0     (0) CHARACTER     4  CTPLVID         VISUAL ID: 'CTPL'
     4     (4) FIXED         2  CTPLLEN         LENGTH OF CTPL
     6     (6) BITSTRING     1  CTPLFLG1        FLAG BYTE 1
           1... ....            CTPLLOFL        LINEOP FLAG
           .1.. ....            CTPLSCNM        SOURCE OF THE GENERATED LIST
                                                1 CATG,0 VTOC
           ..1. ....            CTPLVOLF        VOLUME DATA REQUEST FLG
                                                1 VOLUME DATA REQUESTED
           ...1 ....            CTPLMCDF        MCDS DATA REQUEST FLG 1 MCDS
                                                DATA REQUESTED
           .... 1...            CTPLSORT        SORT COMMAND FLAG
           .... .1..            CTPLFILT        FILTER COMMAND FLAG
           .... ..11            CTPLFLGU        FLAG UNUSED
     7     (7) CHARACTER     1  CTPLPANL        PANEL TYPE
     8     (8) ADDRESS       4  CTPLARNM        ADDRESS OF ARRAY NAME
    12     (C) ADDRESS       4  CTPLFONM        ADDRESS OF FORMAT NAME
    16    (10) CHARACTER     8  CTPLCMDN        NAME OF COMMAND
    24    (18) CHARACTER     8  CTPLCMDR        NAME OF COMMAND ROUTINE
    32    (20) ADDRESS       4  CTPLCLAD        ADDR OF COMMAND LINE
    36    (24) FIXED         2  CTPLCMDL        LEN OF COMMAND LINE
    38    (26) FIXED         2  CTPLERLN        ERROR LINE NUMBER FIELD
    40    (28) ADDRESS       4  CTPLCSCB        ADDR OF CURSOR CNTL BLOCK
    44    (2C) ADDRESS       4  CTPLCSCR        CURRENT SCROLL AMNT ADDR
    48    (30) ADDRESS       4  CTPLPSCR        PREVIOUS SCROLL AMNT ADDR
    52    (34) ADDRESS       4  CTPLCNAM        ADDRESS OF THE VDEFINED
                                                CATALOG NAME
    56    (38) ADDRESS       4  CTPLCNTN        ADDRESS OF THE NAME OF THE
                                                CATALOG NAME TABLE
    60    (3C) ADDRESS       4  CTPLDABL        ADDRESS OF DABL
    64    (40) ADDRESS       4  CTPLMODA        ADDRESS OF FILTER MODULE NAME
    68    (44) CHARACTER     2  CTPLAPPL        APPLICATION ID 'DS', 'VO'
    70    (46) CHARACTER     2  *               RESERVED
    72    (48) ADDRESS       4  CTPLLMSG        ADDRESS OF LONG MESSAGE
    76    (4C) CHARACTER     4  *               RESERVED
    80    (50) CHARACTER        CTPLEND         END OF CTPL
```

CTVT

Entry Point: GDRBCTVT

CTVT Offsets		Type	Length	Name	Description
COMMAND TABLE PROCESSOR VECTOR TABLE (CTVT)					
0	(0)	CHARACTER	36	CTVT	
0	(0)	CHARACTER	4	CTVTVID	VISUAL ID: 'CTVT'
4	(4)	FIXED	2	CTVTLEN	LENGTH OF CTVT
6	(6)	FIXED	2	CTVTUCNT	USE COUNT
8	(8)	ADDRESS	4	CTVTCBE	ADDRESS OF PGM DGTFCTB1
12	(C)	ADDRESS	4	CTVTCEN	ADDRESS OF PGM DGTFCTE1
16	(10)	ADDRESS	4	CTVTCIN	ADDRESS OF PGM DGTFCTIN
20	(14)	ADDRESS	4	CTVTCTE	ADDRESS OF PGM DGTFCTTE
24	(18)	ADDRESS	4	CTVTCSE	ADDRESS OF PGM DGTFCTSE
28	(1C)	ADDRESS	4	CTVTCCK	ADDRESS OF PGM DGTFCTCK
32	(20)	ADDRESS	4	CTVTCPR	ADDR. OF PGM DGTFCTPR
36	(24)	CHARACTER		CTVTEND	END OF CTVT

DAB

Entry Point: DABLDAB

The DAB must be created by the external application that uses it. Follow the rules in the tables below when using a DAB.

Note: The DABRQST field of the DAB must always be set to X'80'

Service Using DAB	Field Name	Field Value
DGTFARF1	DABVAR	31-bit pointer to a variable containing the column value to use for the search (if this DAB's address is contained in the DABL pointed to by SELBDBLI) or a 31-bit pointer a variable where retrieved column contents will be stored (if this DAB's address is contained in the DABL pointed to by SELBDBLO). Variables must be the same length as the value of DABCLEN.
DGTFARU1	DABVAR	31-bit pointer to a variable containing information which is to be written to the data set list array in the column specified by DABCNAME. Variables must be the same length as the value of DABCLEN.
DGTFFOE1	DABVAR	31-bit pointer to a variable where retrieved column value will be stored. Variables must be the same length as the value of DABCLEN.

DABCNAME	Description	DABTYPE	DABCLEN	DABVAR Restrictions
DDISPFLG	Undisplayed column, represents whether entry is displayable	X'01'	X'0001'	Must be X'F0' or X'F1'
DLINEOP	Line operator history column	X'01'	X'000A'	
DOBJ	Data set name	X'01'	X'002C'	
DALLOCUS	Allocated used space	X'11'	X'0007'	
DDSORG	Data set organization	X'07'	X'0003'	
DRECFMT	Data set record format	X'09'	X'0005'	
DVOLSER	Volume serial number	X'01'	X'0006'	
DDEVTYPE	Generic device type	X'0E'	X'0007'	
DENTYP	Catalog entry type	X'01'	X'0001'	
DUCAT	Indicates user catalog or non-user catalog	X'13'	X'0001'	Must be 'Y' or 'N'
DMULTV	Indicates multi-volume data set	X'01'	X'0003'	Must be 'YES' or 'NO '

```
DAB
Offsets      Type       Length Name              Description
============================================================================
      DATA ATTRIBUTE BLOCK (DAB)
============================================================================
    0    (0) CHARACTER    24  DAB
    0    (0) CHARACTER     4  DABVID             VISUAL ID: 'DAB'
    4    (4) FIXED         2  DABLEN             LENGTH OF DAB
    6    (6) FIXED         1  DABRS              RETURN STATUS OF COLUMN NAME
                                                 DAB
    7    (7) FIXED         1  DABFS              FILTER STATUS OF DSN DAB
    8    (8) CHARACTER     8  DABCNAME           COLUMN NAME
   16   (10) CHARACTER     1  DABFLAG1
         1... ....            DABRQST            REQUEST STATUS FLAG
         .1.. ....            DABDISP            DISPLAYABLE STATUS FLAG
         ..1. ....            DABEXT             EXTENSION DAB INDICATOR
         ...1 1111            *                  RESERVED, UNUSED
   17   (11) BITSTRING     1  DABTYPE            DATA ATTRIBUTES FOR COL
   18   (12) FIXED         2  DABCLEN            DATA LENGTH OF COLUMN
   20   (14) ADDRESS       4  DABVAR             ADDRESS OF VARIABLE
   24   (18) CHARACTER        DABEND             END OF DAB
Constants
Length  Type        Value       Name             Description
============================================================================
      VALUES FOR DABRS
============================================================================
    1    DECIMAL   0          DABRS0             DATA RETURNED SUCCESSFULLY
    1    DECIMAL   8          DABRS8             DATA RETURNED AS NULLS
    1    DECIMAL   12         DABRS12            DATA NOT RETURNED
============================================================================
      VALUES FOR DABFS
============================================================================
    1    DECIMAL   0          DABFS0             DATA NOT PROCESSED
    1    DECIMAL   4          DABFS4             DATA PASSED FILTER CRITERIA
    1    DECIMAL   8          DABFS8             DATA DID NOT PASS FILTER
                                                 CRITERIA
    1    DECIMAL   12         DABFS12            DSN NOT FOUND BY CVAFFILT
```

DABL

Entry Point: **The DABL must be created by the external application that uses the service that requires the DABL. For DGTFARF1, SELBDBLI points to the entry point for the DABL associated with column information used as the search criteria to locate entries in the data set list array. SELBDBLO points to the DABL associated with the column information to be retrieved from the entries in the data set list array that meet the search criteria.**

```
DABL
Offsets      Type       Length Name              Description
============================================================================
      DATA ATTRIBUTE BLOCK LIST (DABL)
============================================================================
    0    (0) CHARACTER     *  DABL
    0    (0) CHARACTER     8  DABLMAIN
    0    (0) CHARACTER     4  DABLVID            VISUAL ID: 'DABL'
    4    (4) FIXED         2  DABLLEN            LENGTH OF DABL
    6    (6) FIXED         2  DABLNUM            # OF ENTRIES IN DABL
    8    (8) ADDRESS       4  DABLDAB(*)         LIST OF DAB'S
```

ERTB

Entry Point: GDRBERTB

```
ERTB
Offsets     Type      Length Name            Description
===========================================================================
      ISMF ERROR TABLE (ERTB) - 08/15/84
===========================================================================
    0    (0) CHARACTER  1404  ERTB
    0    (0) CHARACTER    84  ERTBHDR         ERROR TABLE HEADER
    0    (0) CHARACTER     4  ERTBVID         VISUAL ID CONTAINS 'ERTB'
    4    (4) FIXED         2  ERTBLEN         LENGTH OF ERTB
    6    (6) CHARACTER     2  *               RESERVED UNUSED
    8    (8) ADDRESS       4  ERTBBUFR        PTR TO LOG BUFFER
   12    (C) ADDRESS       4  ERTBCURR        PTR TO CURRENT ERTB ENTRY
   16   (10) CHARACTER     8  ERTBAPPL        ISMF APPLICATION ID
   24   (18) CHARACTER     8  ERTBFUNC        FUNCTION/DIALOG NAME
   32   (20) CHARACTER    44  ERTBOBJ         LIST PANEL FUNCTION OBJ
   76   (4C) CHARACTER     8  ERTBPNL         LAST PANEL DISPLAYED ERTB
                                              ENTRIES
   84   (54) CHARACTER  1320  ERTBENT
```

ERNT

Entry Point: Before completing the ERNT, you must set the ERTBCURR field to the value of ERNTPTR (found in the current ERNT). The ERTBCURR then contains the entry point value for the ERNT.

```
ERTB
Offsets     Type      Length Name            Description
===========================================================================
            ERROR TABLE ENTRIES - 08/15/84
===========================================================================
    0    (0) CHARACTER    88  ERNT
    0    (0) ADDRESS       4  ERNTPTR         PTR TO NEXT ERTB ENTRY
    4    (4) CHARACTER    84  ERNTINFO        ERNT INFORMATION
    4    (4) CHARACTER     8  ERNTMODN        MODULE NAME
   12    (C) CHARACTER     8  ERNTPROC        PROCEDURE NAME
   20   (14) FIXED         2  *               RESERVED UNUSED
   22   (16) FIXED         2  ERNTRC          RETURN CODE
   24   (18) CHARACTER     4  ERNTRSNC        REASON CODE
   28   (1C) CHARACTER     8  ERNTSMSG        SHORT MESSAGE ID
   36   (24) CHARACTER     8  ERNTLMSG        LONG MESSAGE ID
   44   (2C) CHARACTER     8  ERNTSERV        FAILING EXTERNAL SRVC
   52   (34) CHARACTER    35  ERNTPD          PROBLEM DATA
   87   (57) CHARACTER     1  *               RESERVED, UNUSED
Constants
Length  Type      Value      Name            Description
===========================================================================
        END OF ISMF ERROR TABLE (ERTB)
===========================================================================
    2    DECIMAL   15         CNTERNT         NUMBER OF ERTB ENTRIES
    2    DECIMAL   1320       LENERNT
```

ET

Entry Point: GDRBET

```
ET
Offsets      Type      Length Name             Description
=======================================================================
      ISMF ENVIRONMENT TABLE (ET)
=======================================================================
    0     (0) CHARACTER    52  ET
    0     (0) CHARACTER     4  ETVID           VISUAL ID CONTAINS 'ET '
    4     (4) FIXED         2  ETLEN           LENGTH OF ET
    6     (6) FIXED         2  *               RESERVED UNUSED
    8     (8) FIXED         4  ETDSS           LEVEL OF DFDSS INSTALLED ON
                                               SYSTEM 'NNVVRRMM'X WHERE NN
                                               IS 04 IF LEVEL COULD NOT BE
                                               DETERMINED VV IS VERSION RR
                                               IS RELEASE MM IS MODIFICATION
   12     (C) FIXED         4  ETHSM           LEVEL OF DFHSM INSTALLED ON
                                               SYSTEM 'NNVVRRMM'X WHERE NN
                                               IS 04 IF LEVEL COULD NOT BE
                                               DETERMINED VV IS VERSION RR
                                               IS RELEASE MM IS MODIFICATION
   16    (10) CHARACTER     6  ETHSMV          6 CHARACTER VOLUME SERIAL
                                               NUMBER USED BY DFHSM TO SIGNAL
                                               MIGRATED NORMALLY 'MIGRAT'
   22    (16) CHARACTER     8  ETFMID          FMID OF ISMF
   30    (1E) CHARACTER     9  ETCOMPID        ISMF COMPONENT ID
   39    (27) CHARACTER     6  ETSYSRES        VOLSER OF SYSTEM RESIDENCE
                                               VOLUME
   45    (2D) CHARACTER     4  ETDEVTYP        DEVICE TYPE OF SYSRES
   49    (31) CHARACTER     3  *               RESERVED, UNUSED
```

FOVT

Entry Point: GDRBFOVT

```
FOVT
Offsets      Type      Length Name             Description
=======================================================================
      FORMAT SERVICE VECTOR TABLE (FOVT)
=======================================================================
    0     (0) CHARACTER    84  FOVT
    0     (0) CHARACTER     4  FOVTVID         VISUAL ID: 'FOVT'
    4     (4) FIXED         2  FOVTLEN         LENGTH OF FOVT
    6     (6) FIXED         2  FOVTUCNT        USE COUNT
    8     (8) ADDRESS       4  FOVTBEG         ADDRESS OF FOBEGIN MODULE
   12     (C) ADDRESS       4  FOVTDONE        ADDRESS OF FODONE MODULE
   16    (10) ADDRESS       4  FOVTFIND        ADDRESS OF FOFIND MODULE
   20    (14) ADDRESS       4  FOVTGTLA        ADDRESS OF FOGETLA MODULE
   24    (18) ADDRESS       4  FOVTGTVL        ADDR. OF FOGETVLA MODULE
   28    (1C) ADDRESS       4  FOVTHIDE        ADDRESS OF FOHIDE MODULE
   32    (20) ADDRESS       4  FOVTHILT        ADDRESS OF FOHILT MODULE
   36    (24) ADDRESS       4  FOVTHRD         ADDRESS OF FOHRD MODULE
   40    (28) ADDRESS       4  FOVTINIT        ADDRESS OF FOINIT MODULE
   44    (2C) ADDRESS       4  FOVTLAI         ADDRESS OF FOLAI MODULE
   48    (30) ADDRESS       4  FOVTLOAD        ADDRESS OF FOLOAD MODULE
   52    (34) ADDRESS       4  FOVTMOVE        ADDRESS OF FOMOVE MODULE
   56    (38) ADDRESS       4  FOVTPCSR        ADDRESS OF FOPCSR MODULE
```

```
      60    (3C) ADDRESS         4  FOVTPTLA        ADDRESS OF FOPUTLA MODULE
      64    (40) ADDRESS         4  FOVTPTVL        ADDR. OF FOPUTVLA MODULE
      68    (44) ADDRESS         4  FOVTQCSR        ADDRESS OF FOQCSR MODULE
      72    (48) ADDRESS         4  FOVTQURY        ADDRESS OF FOQUERY MODULE
      76    (4C) ADDRESS         4  FOVTTERM        ADDRESS OF FOTERM MODULE
      80    (50) ADDRESS         4  FOVTAREA        ADDR OF FOFXAREA MODULE
      84    (54) CHARACTER          FOVTEND         END OF FOVT
```

GDRB

Entry Point: Address is in R10 at all times.

```
    GDRB
    Offsets      Type      Length Name            Description
    ============================================================================
                 ISMF GLOBAL DATA REPOSITORY BLOCK (GDRB)
    ============================================================================
       0     (0) CHARACTER    96  GDRB
       0     (0) CHARACTER     4  GDRBVID         VISUAL ID CONTAINS 'GDRB'
       4     (4) FIXED         2  GDRBLEN         LENGTH OF GDRB
       6     (6) FIXED         2  *               RESERVED
       8     (8) ADDRESS       4  GDRBARVT        ARRAY SERVICES VECTOR TABLE
      12     (C) ADDRESS       4  GDRBASAB        ARRAY SERVICES ANCHOR BLOCK
      16    (10) ADDRESS       4  GDRBCTAB        COMMAND TABLE ANCHOR BLOCK
      20    (14) ADDRESS       4  GDRBDAAB        DATA ACQUISITION ANCHOR BLK
      24    (18) ADDRESS       4  GDRBDSAB        DATA SET ANCHOR BLOCK
      28    (1C) ADDRESS       4  GDRBERTB        ERROR TABLE ADDRESS
      32    (20) ADDRESS       4  GDRBET          ENVIRONMENT TABLE
      36    (24) ADDRESS       4  GDRBFOVT        FORMAT SERVICE VECTOR TABLE
      40    (28) ADDRESS       4  GDRBFSAB        FORMAT SERVICE ANCHOR BLOCK
      44    (2C) ADDRESS       4  GDRBFST         FROZEN STORAGE TABLE PTR
      48    (30) ADDRESS       4  GDRBIMTT        INTER MODULE TRACE TABLE
      52    (34) ADDRESS       4  GDRBLLBL        LOAD LIST BLOCK POINTER
      56    (38) ADDRESS       4  GDRBLPAB        LINE OPERATOR ANCHOR BLOCK
      60    (3C) ADDRESS       4  GDRBMDAB        MAIN DIALOG ANCHOR BLOCK
      64    (40) ADDRESS       4  GDRBPVT         PROFILE VARIABLE TABLE
      68    (44) ADDRESS       4  GDRBSCT         STORAGE CONTROL TABLE PTR
      72    (48) ADDRESS       4  GDRBSRVT        SERVICE ROUTINE VECTOR TBL
      76    (4C) ADDRESS       4  GDRBTPTT        TRACE POINT TRACE TABLE PTR
      80    (50) ADDRESS       4  GDRBISPF        POINTER TO ISPF
      84    (54) ADDRESS       4  GDRBCTVT        POINTER TO CTVT
      88    (58) ADDRESS       4  GDRBDCMD        PTR TO ZTRAIL TRUNC REMAIN
      92    (5C) ADDRESS       4  GDRBDEVT        PTR TO DEVICE TYPE TABLE
```

IMTT

Entry Point: Before completing the IMTT, you must set the IMTTCURR field to the value of IMNTPTR (found in the current ERNT). The IMTTCURR then contains the entry point value for the IMTT.

```
    IMTT
    Offsets      Type      Length Name            Description
    ============================================================================
               ISMF INTER MODULE TRACE TABLE (IMTT)
    ============================================================================
       0     (0) CHARACTER  6288  IMTT
       0     (0) CHARACTER    16  IMTTHDR         INTER MODULE TRACE TABLE HDR
       0     (0) CHARACTER     4  IMTTVID         VISUAL ID CONTAINS 'IMTT'
       4     (4) FIXED         2  IMTTLEN         LENGTH OF IMTT
       6     (6) FIXED         2  *               RESERVED UNUSED
       8     (8) ADDRESS       4  IMTTCUR         POINTER TO THE CURRENT ENTRY
      12     (C) CHARACTER     4  *               RESERVED UNUSED IMTT ENTRIES
      16    (10) CHARACTER  6272  IMNTENT
```

IMNT

Entry Point: IMTTCURR

```
IMTT
Offsets      Type      Length Name            Description
===========================================================================
               IMTT ENTRIES
===========================================================================
    0     (0) CHARACTER    56   IMNT
    0     (0) ADDRESS       4   IMNTPTR         PTR TO NEXT IMTT ENTRY
    4     (4) CHARACTER    52   IMNTINFO        IMNT INFORMATION
    4     (4) CHARACTER     8   IMNTEMID        MODULE ID
   12     (C) ADDRESS       4   IMNTREG1        PARM LIST REG REG 1
   16    (10) ADDRESS       4   IMNTREGB        CURRENT WORK AREA ADDR
   20    (14) CHARACTER    35   IMNTUSER        USER AREA
   55    (37) CHARACTER     1   *               RESERVED, UNUSED
Constants
Length  Type       Value      Name            Description
===========================================================================
        END OF ISMF INTER-MODULE TRACE TABLE (IMTT)
===========================================================================
    2    DECIMAL   112        CNTIMNT         NUMBER OF IMNT ENTRIES
    2    DECIMAL   6272       LENIMNT
```

LOGB

Entry Point: ERTBBUFR

```
LOGB
Offsets      Type      Length Name            Description
===========================================================================
      ISMF LOG BUFFER (LOGB)
===========================================================================
    0     (0) CHARACTER   236   LOGB
    0     (0) CHARACTER     4   LOGBVID         VISUAL ID CONTAINS 'LOGB'
    4     (4) FIXED         2   LOGBLEN         LENGTH OF LOGB
    6     (6) CHARACTER     2   *               RESERVED
    8     (8) CHARACTER    72   PGMLOG08        VARS WITH LENGTH OF 8
    8     (8) CHARACTER     8   LOGBAPPL        ISMF APPLICATION ID
   16    (10) CHARACTER     8   LOGBMODN        MODULE ID
   24    (18) CHARACTER     8   LOGBPROC        PROCEDURE ID
   32    (20) CHARACTER     8   LOGBPNL         LAST PANEL DISPLAYED
   40    (28) CHARACTER     8   LOGBSERV        NAME OF FAILING SERVICE
   48    (30) CHARACTER     8   LOGBJBNM        SUBMIT JOB NAME
   56    (38) CHARACTER     8   LOGBJBNO        SUBMIT JOB NUMBER
   64    (40) CHARACTER     8   LOGBSMSG        SHORT MESSAGE
   72    (48) CHARACTER     8   LOGBLMSG        LONG MESSAGE
   80    (50) CHARACTER    11   PGMLOG11        VARS WITH LENGTH OF 44
   80    (50) CHARACTER    11   LOGBFUNC        FUNCTION/DIALOG NAME
   91    (5B) CHARACTER     1   *               RESERVED UNUSED
   92    (5C) CHARACTER    44   PGMLOG44        VARS WITH LENGTH OF 44
   92    (5C) CHARACTER    44   LOGBOBJ         LIST PANEL FUNCTION OBJ
  136    (88) CHARACTER    54   PGMLOG54        VARS WITH LENGTH OF 54
  136    (88) CHARACTER    54   LOGBJBDS        SUBMIT DATA SET NAME
  190    (BE) CHARACTER     2   *               RESERVED UNUSED
  192    (C0) CHARACTER     8   PGMLOG04        VARS WITH LENGTH OF 4
  192    (C0) CHARACTER     4   LOGBRC          RETURN CODE
  196    (C4) CHARACTER     4   LOGBRSNC        REASON CODE
  200    (C8) CHARACTER    35   PGMLOG35        VARS WITH LENGTH OF 35
  200    (C8) CHARACTER    35   LOGBPD          PROBLEM DATA
  235    (EB) CHARACTER     1   *               FILL UP REST OF WORD
```

LPAP

Entry Point: LPABLPAP

```
LPAP
Offsets      Type       Length Name             Description
============================================================================
      LINE OPERATOR TABLE - APPLICATION TABLE (LPAP)
============================================================================
    0     (0) CHARACTER      *  LPAP
    0     (0) CHARACTER      8  LPAPMAIN
    0     (0) CHARACTER      4  LPAPVID          VISUAL ID: 'LPAP'
    4     (4) FIXED          2  LPAPLEN          LENGTH OF LPAP
    6     (6) FIXED          2  LPAPCNT          # OF LINE OPERATORS
    8     (8) CHARACTER     28  LPAPENT(*)
    8     (8) CHARACTER      8  LPAPLONM         LINE OPERATOR NAME
   16    (10) FIXED          1  LPAPTRUN         MIN. # OF CHARACTERS USED IN
                                                 TRUNCATION
   17    (11) CHARACTER      3  *                RESERVED, UNUSED
   20    (14) CHARACTER      8  LPAPRTNM         LINE OP ROUTINE NAME
   28    (1C) CHARACTER      8  LPAPTENM         TERMINATION ROUTINE
```

LPCB

Entry Point: LPPLLPCB (line operator is invoker)/CPPLCSCB (command is invoker)

```
LPCB
Offsets      Type       Length Name             Description
============================================================================
      LIST PANEL CURSOR CONTROL BLOCK (LPCB)
============================================================================
    0     (0) CHARACTER     30  LPCB
    0     (0) CHARACTER      4  LPCBVID          VISUAL ID: 'LPCB'
    4     (4) FIXED          2  LPCBLEN          LENGTH OF LPCB
    6     (6) FIXED          2  LPCBENO          ENTRY NUMBER
    8     (8) CHARACTER      1  LPCBMFG          MULTI LINE ENTRY FLAG
    9     (9) CHARACTER      1  *                RESERVED, UNUSED
   10     (A) FIXED          2  LPCBMLN          MULTI LNE ENTRY INDICATOR
   12     (C) CHARACTER      2  LPCBCTG          COLUMN TAG
   14     (E) CHARACTER      2  *                RESERVED, UNUSED
   16    (10) CHARACTER      8  LPCBAREA         AREA NAME
   24    (18) FIXED          4  LPCBRND          ROW ID NAME
   28    (1C) FIXED          2  LPCBOFF          OFFSET WITHIN AREA
   30    (1E) CHARACTER         LPCBEND          END OF LPCB
```

LPPL

Entry Point: R1 on entry to line operator routine

LPPL Offsets	Type	Length	Name	Description
LINE OPERATOR PROCESSOR PARAMETER LIST (LPPL) PASSES TO THE LINE OPERATOR ROUTINES				
0 (0)	CHARACTER	68	LPPL	
0 (0)	CHARACTER	4	LPPLVID	VISUAL ID: 'LPPL'
4 (4)	FIXED	2	LPPLLEN	LENGTH OF LPPL
6 (6)	BITSTRING	1	LPPLFLG1	FLAG FIELD
	1...		LPPLSTD	STD INVOCATION
	.1..		LPPLTEOA	INVOKE AT END OF APPL
	..1.		LPPLTEOL	INVOKE AT END OF LIST?
	...1		LPPLLOFL	ACTIVE LINE OPERATOR?
	.... 1...		LPPLLAST	LAST USED MODE?
	.... .1..		LPPLSMSG	MSG ID SET IN ERTB ?
	.... ..1.		LPPLSCNM	LIST GEN'D FR CATALOG
	1		LPPLHIDE	HIDE LINE OP
7 (7)	BITSTRING	1	LPPLFLG2	FLAG FIELD , UNUSED
8 (8)	ADDRESS	4	LPPLARAD	ARRAY NAME ADDRESS
12 (C)	ADDRESS	4	LPPLFOAD	FORMAT NAME ADDRESS
16 (10)	ADDRESS	4	LPPLCLCB	COMMAND/LINE OP CNTL BLK
20 (14)	CHARACTER	8	LPPLLO	LINE OPERATOR IN PROGRESS
28 (1C)	FIXED	4	LPPLROWI	ROWID # OF THE LAI ENTRY
32 (20)	FIXED	2	LPPLENT	LAI ENTRY #
34 (22)	CHARACTER	1	LPPLCMDF	COMMAND FLAG
35 (23)	CHARACTER	1	*	RESERVED
36 (24)	ADDRESS	4	LPPLCNAM	ADDRESS OF CATALOG NAME FOR DATA SET APPLICATION
40 (28)	ADDRESS	4	LPPLLAIA	ADDRESS OF LAI
44 (2C)	ADDRESS	4	LPPLROWA	ADDRESS OF ROW IDS
48 (30)	FIXED	4	LPPLLAIT	TOTAL ENTRIES IN LAI
52 (34)	ADDRESS	4	LPPLLPCB	LIST PNL CURSOR CTL BLK
56 (38)	ADDRESS	4	LPPLCNTN	ADDR OF THE NAME OF THE CATALOG NAME TABLE FOR DATA SET APPLICATION
60 (3C)	ADDRESS	4	LPPLLMSG	ADDR OF THE LONG MSG
64 (40)	CHARACTER	4	*	RESERVED UNUSED
68 (44)	CHARACTER		LPPLEND	END OF LPPL

PCCB

Entry Point: CPPLCSCB

```
PCCB
Offsets      Type       Length Name              Description
===========================================================================
      PANEL CURSOR CONTROL BLOCK (PCCB)
===========================================================================
    0     (0) CHARACTER    16  PCCB
    0     (0) CHARACTER     4  PCCBVID           VISUAL ID: 'PCCB'
    4     (4) FIXED         2  PCCBLEN           LENGTH OF PCCB
    6     (6) FIXED         2  *                 RESERVED, UNUSED
    8     (8) ADDRESS       4  PCCBCFAD          CURSOR FIELD ADDRESS
   12     (C) ADDRESS       4  PCCBCOAD          CURSOR OFFSET ADDRESS
   16    (10) CHARACTER        PCCBEND           END OF PCCB
```

PVT

Entry Point: GDRBPVT

```
PVT
Offsets      Type       Length Name              Description
===========================================================================
      ISMF PROFILE VARIABLE TABLE (PVT)
===========================================================================
    0     (0) CHARACTER    16  PVT
    0     (0) CHARACTER     4  PVTVID            VISUAL ID CONTAINS 'PVT '
    4     (4) FIXED         2  PVTLEN            LENGTH OF PVT
    6     (6) FIXED         2  *                 RESERVED
    8     (8) ADDRESS       4  PVTL72            PTR TO VARS WITH LEN 72
   12     (C) ADDRESS       4  PVTL1             PTR TO VARS WITH LEN 1
```

PVTV

Entry Point: PVTL72 (field PVTL72 in control block PVT—for structure PGMPVT72)

PVTL1 (field PVTL1 in control block PVT—for structure PGMPVT01)

PVTV Offsets		Type	Length	Name	Description
		ISMF PROFILE VARIABLE TABLE VARIABLES (PVTV)			
0	(0)	CHARACTER	1512	PGMPVT72	PVT VARS WITH LEN OF 72
0	(0)	CHARACTER	72	APPFDE11	DFDSS EXECUTE STATEMENT VARIABLE
72	(48)	CHARACTER	72	APPFDE12	DFDSS EXECUTE STATEMENT VARIABLE
144	(90)	CHARACTER	72	APPFDE13	DFDSS EXECUTE STATEMENT VARIABLE
216	(D8)	CHARACTER	72	APPFDE14	DFDSS EXECUTE STATEMENT VARIABLE
288	(120)	CHARACTER	72	APPFDE15	DFDSS EXECUTE STATEMENT VARIABLE
360	(168)	CHARACTER	72	APPFDE16	DFDSS EXECUTE STATEMENT VARIABLE
432	(1B0)	CHARACTER	72	APPFDE17	DFDSS EXECUTE STATEMENT VARIABLE
504	(1F8)	CHARACTER	72	APPFJOB1	JOB STATEMENT VARIABLE
576	(240)	CHARACTER	72	APPFJOB2	JOB STATEMENT VARIABLE
648	(288)	CHARACTER	72	APPFJOB3	JOB STATEMENT VARIABLE
720	(2D0)	CHARACTER	72	APPFJOB4	JOB STATEMENT VARIABLE
792	(318)	CHARACTER	72	APPFJOB5	JOB STATEMENT VARIABLE
864	(360)	CHARACTER	72	APPFJOB6	JOB STATEMENT VARIABLE
936	(3A8)	CHARACTER	72	APPFJOB7	JOB STATEMENT VARIABLE
1008	(3F0)	CHARACTER	72	APPFDE21	DFDSS EXECUTE STATEMENT VARIABLE
1080	(438)	CHARACTER	72	APPFDE22	DFDSS EXECUTE STATEMENT VARIABLE
1152	(480)	CHARACTER	72	APPFDE23	DFDSS EXECUTE STATEMENT VARIABLE
1224	(4C8)	CHARACTER	72	APPFDE24	DFDSS EXECUTE STATEMENT VARIABLE
1296	(510)	CHARACTER	72	APPFDE25	DFDSS EXECUTE STATEMENT VARIABLE
1368	(558)	CHARACTER	72	APPFDE26	DFDSS EXECUTE STATEMENT VARIABLE
1440	(5A0)	CHARACTER	72	APPFDE27	DFDSS EXECUTE STATEMENT VARIABLE

PVTV Offsets		Type	Length	Name	Description
0	(0)	CHARACTER	4	PGMPVT01	PVT VARS WITH LEN OF 01
0	(0)	CHARACTER	1	APPFLDED	LOGGING/ABEND CONTROL VARIABLE
1	(1)	CHARACTER	1	APPFLIMT	LOGGING/ABEND CONTROL VARIABLE
2	(2)	CHARACTER	1	APPFLTPT	LOGGING/ABEND CONTROL VARIABLE
3	(3)	CHARACTER	1	APPFLRFA	LOGGING/ABEND CONTROL VARIABLE

SELB

Entry Point: This control block and all control blocks addressed by it must be created by the external application that uses the service requiring the control block.

When using the SELB, observe the requirements in the following table.

Service Using SELB	Field Name	Field Value
DGTFARF1	SELBARNM	Value pointed to by CPPLARNM
	SELBNP	'N'
	SELBNO	15-bit value set to X'1'

```
SELB
Offsets      Type      Length Name             Description
===========================================================================
      SELECT BLOCK (SELB)
===========================================================================
     0     (0) CHARACTER    40  SELB
     0     (0) CHARACTER     4  SELBVID          VISUAL ID: 'SELB'
     4     (4) FIXED         2  SELBLEN          LENGTH OF SELB
     6     (6) FIXED         2  *                RESERVED, UNUSED
     8     (8) CHARACTER     8  SELBARNM         ARRAY NAME
    16    (10) CHARACTER     1  SELBNP           DIRECTION OF SEARCH
    17    (11) CHARACTER     1  *                RESERVED, UNUSED
    18    (12) FIXED         2  SELBLNO          NTH LINE TO BE RETRIEVED
    20    (14) BITSTRING     1  SELBFLG1         FLAG FIELD
           1... ....            SELBKEEP         SAVE SEARCH CRITERIA
           .1.. ....            SELBUSE          USE OLD SEARCH CRITERIA
           ..1. ....            SELBONE          USE NEW SEARCH CRITERIA
    21    (15) CHARACTER     3  *                RESERVED, UNUSED
    24    (18) ADDRESS       4  SELBDBLO         ADDR OF DABL FOR OUTPUT
    28    (1C) ADDRESS       4  SELBCNDL         ADDRESS OF CONDITION LIST
    32    (20) ADDRESS       4  SELBDBLI         ADDR OF DABL FOR INPUT
    36    (24) ADDRESS       4  SELBRID          ADDR OF VAR TO PUT ROWID
    40    (28) CHARACTER        SELBEND          END OF SELB
```

SRVT

Entry Point: GDRBSRVT

```
SRVT
Offsets      Type       Length Name             Description
    0     (0) CHARACTER    112  SRVT
    0     (0) CHARACTER      8  SRVTHDR         SRVT HEADER
    0     (0) CHARACTER      4  SRVTVID         SRVT VISUAL ID IS 'SRVT'
    4     (4) FIXED          2  SRVTLEN         LENGTH OF SRVT
    6     (6) FIXED          2  *               UNUSED, RESERVED
    8     (8) ADDRESS        4  SRVTPR01        PROLOG ENTRY POINT ADDRESS
   12     (C) ADDRESS        4  SRVTPR02        PROLOG ENTRY POINT ADDRESS
   16    (10) ADDRESS        4  SRVTEP01        EPILOG ENTRY POINT ADDRESS
   20    (14) ADDRESS        4  SRVTGM01        GETMEM ENTRY POINT ADDRESS
   24    (18) ADDRESS        4  SRVTFM01        FREEMEM ENTRY POINT ADDRESS
   28    (1C) ADDRESS        4  SRVTLD01        LOAD ENTRY POINT ADDRESS
   32    (20) ADDRESS        4  SRVTDT01        DELETE ENTRY POINT ADDRESS
   36    (24) ADDRESS        4  SRVTDV01        DSN VERIFICATION ENTRY POINT
                                                ADR
   40    (28) ADDRESS        4  SRVTVV01        VOL VERIFICATION ENTRY POINT
                                                ADR
   44    (2C) ADDRESS        4  SRVTWO01        WORD PARSER ENTRY POINT
                                                ADDRESS
   48    (30) ADDRESS        4  SRVTARB1        ARRAY SERVICE INITIALIZATION
   52    (34) ADDRESS        4  SRVTARN1        ARRAY SERVICE TERMINATION
   56    (38) ADDRESS        4  SRVTFOB1        FORMAT SERVICE INITIALIZATION
   60    (3C) ADDRESS        4  SRVTFON1        FORMAT SERVICE TERMINATION
   64    (40) ADDRESS        4  SRVTAS01        RACROUTE SERVICE ROUTINE
   68    (44) ADDRESS        4  SRVTCD01        CONVERSION SERVICE ROUTINE
   72    (48) ADDRESS        4  SRVTDA00        DATA ACQUISITION EPA
   76    (4C) ADDRESS        4  SRVTLG01        LOG SERVICE ROUTINE EPA
   80    (50) ADDRESS        4  SRVTCC01        SELECT FILTER VERIFICATION
   84    (54) ADDRESS        4  SRVTCTB1        COMMAND TABLE BEGIN
   88    (58) ADDRESS        4  SRVTCTE1        COMMAND TABLE END
   92    (5C) ADDRESS        4  SRVTCD02        CONVERSION SERVICE ROUTINE
   96    (60) ADDRESS        4  SRVTCD03        CONVERSION SERVICE ROUTINE
  100    (64) ADDRESS        4  SRVTCD04        CONVERSION SERVICE ROUTINE
  104    (68) ADDRESS        4  SRVTCD05        CONVERSION SERVICE ROUTINE
  108    (6C) ADDRESS        4  SRVTCDQ1        CONVERSION TABLE QUERY ROUTINE
Constants
Length  Type      Value      Name             Description
============================================================================
      ISMF COMMON SERVICE ROUTINE VECTOR TABLE (SRVT)
============================================================================
    2    DECIMAL   18         SRVTCNT
```

Index

Reader's Comment Form

MVS/XA System-Data Administration
GC26-4149-2

GC26-4149-2

Reader's Comment Form

and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape | Please do not staple | Fold and tape

IBM®

[illegible]

[illegible]

[illegible]

[illegible] ATMS

[illegible] ATM series [illegible]
[illegible] these devices [illegible]
as requirements.

[illegible] units have a [illegible]
[illegible]
[illegible]
Surcharge/Rebate. This rule requires [illegible]
[illegible] an on-screen notice of [illegible]
surcharge and an option to [illegible]

[illegible] of this interactive [illegible]
[illegible] to proceed [illegible]
the IBM 3500 [illegible] with a single [illegible]

- [illegible] screen that notifies the [illegible] of a surcharge [illegible]
  the consumer the option [illegible] is a must.

- The "Fee Notice" as described [illegible] the surcharge amount.

- The screen must state "Read [illegible]

The second issue with the IBM [illegible]
receipts. IBM 3500 receipt must have [illegible]
the ATM to place an asterisk (*) by the [illegible]
would state:

Amount of [illegible]
[illegible]

If the [illegible] message is on the [illegible]
the [illegible] see [illegible]