MVS

Programming: Hiperbatch Guide

GC28-1470-00

MVS

Programming: Hiperbatch Guide

GC28-1470-00

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page vii.

First Edition, June 1994

This edition applies to Version 5 Release 1 of MVS/ESA System Product (5655-068 or 5655-069) and to all subsequent versions, releases, and modifications until otherwise indicated in new editions or technical newsletters.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 914+432-9405
FAX (Other Countries):
Your International Access Code +1+914+432-9405

IBMLink (United States customers only): KGNVMC(D58PUBS)
IBM Mail Exchange: USIB2NZL at IBMMAIL
Internet: d58pubs@vnet.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594
USA

Programming Interface Information

This book is intended to help the customer to determine whether Hiperbatch can work at a particular installation, and to learn how to use Hiperbatch. It contains information for those who manage the system and its workload, those who manage MVS, and those who understand the requirements for the installation's applications.

This book also documents Product-sensitive Programming Interface and Associated Guidance Information provided by MVS/ESA System Product, as of Version 5.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of MVS/ESA System Product, as of Version 5. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Product-sensitive programming interface

Product-sensitive Programming Interface and Associated Guidance Information...

End of Product-sensitive programming interface

Trademarks

The following terms, **DENOTED BY AN ASTERISK (*)** on first use in the text of this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

- DB2
- DFSMS
- DFSMS/MVS
- Hiperbatch
- Hipersorting
- IBM
- IBMLink
- MVS/DFP
- MVS/ESA
- OPC
- RACF

About This Book

This book describes the batch processing problem that Hiperbatch can resolve, how to determine whether Hiperbatch can work at a particular installation, and how to use Hiperbatch.

Who This Book Is For

This book is for the people at an installation who make decisions about managing the system and its workload to meet the needs of the business. It also contains information for the system programmer who manages MVS and the application programmer who understands the requirements for the applications that the installation uses.

How This Book Is Organized

This book has four chapters:

- Chapter 1, “Batch Processing” describes batch processing problems in general and how Hiperbatch might resolve these problems.
- Chapter 2, “Evaluating Batch Processing” describes specific batch processing concerns in relation to Hiperbatch and alternative ways of improving batch processing performance.
- Chapter 3, “Using Hiperbatch” describes how to set up system controls and plan operator procedures for Hiperbatch.
- Chapter 4, “Hiperbatch Monitor” describes how to use the Hiperbatch Monitor to determine if Hiperbatch is working effectively.

Where useful, a specific “Questions and Answers” section follows a topic.

How To Use This Book

Use this book to make decisions about your installation's use of Hiperbatch and to understand the system control mechanisms that Hiperbatch requires. This book also includes information about using the Hiperbatch monitor and the availability of the sample Hiperbatch installation exit, both of which IBM supplies.

This book is one of the set of programming books for MVS. This set describes how to write programs in assembler language or high-level languages, such as C, FORTRAN, and COBOL. The following tables show how this book fits in with the others in the set.

Note: The book titles and order numbers listed in the following tables pertain to MVS/ESA SP 5.1.0. If you are using a different release of MVS, please refer to the corresponding library guide to find the correct book titles and order numbers for your release.

Programming Books for Assembler Language Programs

Book Title	Use this book to:	Order Number
MVS/ESA SP V5 Assembler Services Guide	Find out how to use system services provided by macros and callable services available to all assembler language programs. If you are relatively new to assembler language programming, this book is a good place to start.	GC28-1466
MVS/ESA SP V5 Assembler Services Reference	Learn how to code macros and callable services that are available to all assembler language programs. This book is for all assembler language programmers.	GC28-1474
MVS/ESA SP V5 Auth Assembler Services Guide	Find out how to use system services provided by macros and callable services that are available to programs running in supervisor state or with PSW key 0-7 or that are APF-authorized programs. This book is for experienced assembler language programmers; it assumes, for example, that you are familiar with the information in *MVS/ESA SP V5 Assembler Services Guide*.	GC28-1467
MVS/ESA SP V5 Auth Assembler Services Reference ALE-DYN *MVS/ESA SP V5 Auth Assembler Services Reference ENF-ITT* *MVS/ESA SP V5 Auth Assembler Services Reference LLA-SDU* *MVS/ESA SP V5 Auth Assembler Services Reference SET-WTO*	Learn how to code macros and callable services that are available to programs running in supervisor state or with PSW key 0-7 or that are APF-authorized programs. This book is for experienced assembler language programmers.	GC28-1475 GC28-1476 GC28-1477 GC28-1478

Book Title	Use this book to:	Order Number
MVS/ESA SP V5 Extended Addressability Guide	Find out how to extend or restrict the storage available to programs through the use of access registers, cross memory services, data spaces, subspaces, and hiperspaces. This book is for experienced assembler language programmers.	GC28-1468
MVS/ESA SP V5 JES Common Coupling Services	Learn how to use the JES common coupling services and exits to affect JES communication processing in an MVS sysplex environment. This book is for experienced JES2, JES3, and BCP assembler language system programmers.	GC28-1497
MVS/ESA SP V5 Sysplex Services Guide	Find out how to use authorized system services to enable your multisystem application or subsystem to run in a sysplex, benefit from automatic restarts, and share data using the coupling facility. This book is for experienced assembler language programmers.	GC28-1495
MVS/ESA SP V5 Sysplex Services Reference	Learn how to code authorized system services to enable your multisystem application or subsystem to run in a sysplex, benefit from automatic restarts, and share data using the coupling facility. This book is for experienced assembler language programmers.	GC28-1496
MVS/ESA SP V5 Workload Management Services	Find out how to use workload management services that are intended for subsystem work manager and performance monitoring programs.	GC28-1494
MVS Batch Local Shared Resources	Find out whether your installation's batch applications can benefit from batch LSR subsystem and how to use batch LSR subsystem. Using this book does not require a knowledge of assembler language programming.	GC28-1469

Book Title	Use this book to:	Order Number
MVS Hiperbatch Guide	Find out whether your installation's batch applications can benefit from Hiperbatch and how to use Hiperbatch. Using this book does not require a knowledge of assembler language programming.	GC28-1470

Programming Books for High-Level Language Programs

Book Title	Use this book to:	Order Number
MVS/ESA SP V5 Callable Services for HLL	Find out how to use and code program CALLs for specific MVS services in high-level language (HLL) programs.	GC28-1464

Programming Books for Both Assembler and High-Level Language Programs

Book Title	Use this book to:	Order Number
&ieaq200t.	Learn how to use registration services to register a product or to prepare an optional OS/390 element dynamic enablement. This book is for assembler or C language programmers.	&ieaq200n.
MVS/ESA SP V5 Writing TPs for APPC/MVS	Find out how to use and code program CALLs for APPC/MVS communication services that are intended for both APPC/MVS transaction programs and servers. This book is a companion to *MVS/ESA SP V5 Writing Servers for APPC/MVS*, which documents APPC/MVS services for servers only.	GC28-1471
MVS/ESA SP V5 Writing Servers for APPC/MVS	Find out how to use and code program CALLs for APPC/MVS communication services that are intended for servers only. This book is a companion to *MVS/ESA SP V5 Writing TPs for APPC/MVS*, which documents the APPC/MVS services for both transaction programs and servers.	GC28-1472
MVS/ESA SP V5 Writing Transaction Schedulers for APPC/MVS	Find out how to use and code program CALLs for APPC/MVS system services that are intended for APPC/MVS transaction schedulers. This book is for programmers who write transaction schedulers other than the one APPC/MVS provides.	GC28-1465

Related Information

Where necessary, this book references information in other books, using shortened versions of the book title. The following table shows the full titles and the order numbers.

Note: The book titles and order numbers listed in the following table pertain to MVS/ESA SP 5.1.0. If you are using a different release of MVS, please refer to the corresponding library guide to find the correct book titles and order numbers for your release.

Short Title Used in This Book	Title	Order Number
MVS/DFP Access Method Services for the Integrated Catalog Facility	*MVS/DFP V3 Access Method Services for the Integrated Catalog Facility*	SC26-4562
	DFSMS/MVS Version 1 Release 1 Access Method Services for the Integrated Catalog Facility	SC26-4906
MVS/DFP Checkpoint/Restart	*MVS/DFP V3 Checkpoint/Restart*	SC26-4556
	DFSMS/MVS Version 1 Release 1 Checkpoint/Restart	SC26-4907
MVS/DFP Linkage Editor and Loader	*MVS/DFP V3 Linkage Editor and Loader*	SC26-4564
	DFSMS/MVS Version 1 Release 1 Program Management	SC26-4916
MVS/DFP Macro Instructions for Data Sets	*MVS/DFP V3 Macro Instructions for Data Sets*	SC26-4747
	DFSMS/MVS Version 1 Release 1 Macro Instructions for Data Sets	SC26-4913
MVS/DFP Using Data Sets	*MVS/DFP V3 Using Data Sets*	SC26-4749
	DFSMS/MVS Version 1 Release 1 Using Data Sets	SC26-4922
Assembler H Version 2 Application Programming: Language Reference	*Assembler H Version 2 Application Programming: Language Reference*	GC26-4037
DFSORT Application Programming Guide	*DFSORT Application Programming Guide*	SC33-4035
MVS/ESA SP V5 Auth Assembler Services Reference ALE-DYN	*MVS/ESA Programming: Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC)*	GC28-1475
MVS/ESA SP V5 Auth Assembler Services Reference ENF-ITT	*MVS/ESA Programming: Authorized Assembler Services Reference, Volume 2 (ENFREQ-ITTFMTB)*	GC28-1476
MVS/ESA SP V5 Auth Assembler Services Reference LLA-SDU	*MVS/ESA Programming: Authorized Assembler Services Reference, Volume 3 (LLACOPY-SDUMPX)*	GC28-1477
MVS/ESA SP V5 Auth Assembler Services Reference SET-WTO	*MVS/ESA Programming: Authorized Assembler Services Reference, Volume 4 (SETFRR-WTOR)*	GC28-1478

Short Title Used in This Book	Title	Order Number
MVS/ESA SP V5 Extended Addressability Guide	*MVS/ESA Programming: Extended Addressability Guide*	GC28-1468
MVS/ESA SP V5 Assembler Services Guide	*MVS/ESA Programming: Assembler Services Guide*	GC28-1466
MVS/ESA SP V5 Assembler Services Reference	*MVS/ESA Programming: Assembler Services Reference*	GC28-1474
MVS/ESA SP V5 Auth Assembler Services Guide	*MVS/ESA Programming: Authorized Assembler Services Guide*	GC28-1467
MVS/ESA SP V5 Callable Services for HLL	*MVS/ESA Programming: Callable Services for High-Level Languages*	GC28-1464
MVS/ESA SP V5 Sysplex Services Guide	*MVS/ESA Programming: Sysplex Services Guide*	GC28-1495
MVS/ESA SP V5 Sysplex Services Reference	*MVS/ESA Programming: Sysplex Services Reference*	GC28-1496
MVS/ESA SP V5 Writing TPs for APPC/MVS	*MVS/ESA Programming: Writing Transaction Programs for APPC/MVS*	GC28-1471
MVS/ESA SP V5 Writing Servers for APPC/MVS	*MVS/ESA Programming: Writing Servers for APPC/MVS*	GC28-1472
MVS/ESA SP V5 Writing Transaction Schedulers for APPC/MVS	*MVS/ESA Programming: Writing Transaction Schedulers for APPC/MVS*	GC28-1465
MVS/ESA SP V5 Workload Management Services	*MVS/ESA Programming: Workload Management Services*	GC28-1494
MVS Hiperbatch Guide	*MVS Programming: Hiperbatch Guide*	GC28-1470
MVS Batch Local Shared Resources	*MVS Programming: Batch Local Shared Resources Subsystem Guide*	GC28-1469
MVS/ESA SP V5 Conversion Notebook	*MVS/ESA Conversion Notebook*	GC28-1436
MVS/ESA SP V5 Initialization and Tuning Guide	*MVS/ESA Initialization and Tuning Guide*	SC28-1451
MVS/ESA SP V5 Initialization and Tuning Reference	*MVS/ESA Initialization and Tuning Reference*	SC28-1452
MVS/ESA SP V5 Installation Exits	*MVS/ESA Installation Exits*	SC28-1459
MVS/ESA SP V5 JCL User's Guide	*MVS/ESA JCL User's Guide*	GC28-1473
MVS/ESA SP V5 JES Common Coupling Services	*MVS/ESA Programming: JES Common Coupling Services*	GC28-1497
MVS/ESA SP V5 Program Directory: JES2	*MVS/ESA Program Directory: JES2*	None

Short Title Used in This Book	Title	Order Number
MVS/ESA SP V5 Program Directory: JES3	*MVS/ESA Program Directory: JES3*	None
MVS/ESA SP V5 System Commands	*MVS/ESA System Commands*	GC28-1442
RACF V2 Security Administrator's Guide[1]	*RACF V2 Security Administrator's Guide*	SC23-3726
	RACF V1 R9.2 Security Administrator's Guide	SC28-1340

[1] Select the book that supports the version of the product your installation is using.

Summary of Changes

Summary of Changes
for CG28-1470-00
as updated June, 1994

This book contains information previously presented in *MVS/ESA Application Development Guide: Hiperbatch*, GC28-1673-03, which supports MVS/ESA System Product Version 4 Release 3.

This revision includes maintenance and editorial changes.

Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Chapter 1. Batch Processing

The workload at a data processing installation typically falls into three categories: online processing, batch processing, and housekeeping (work, such as maintenance, that is not part of the production workload.)

Online processing traditionally peaks in the morning and afternoon. Unless online processing has been designed to tolerate significant amounts of concurrent batch processing, an installation concentrates its batch processing in the time that exists when there is limited online processing. This time is often called the overnight batch window, because of the traditional contrast between the high use of online services during the day.

Figure 1-1 shows how the three activities might occur in a typical installation over a 24-hour day:

- Online, or interactive, processing (A)
- Batch processing (B)
- Housekeeping (C)

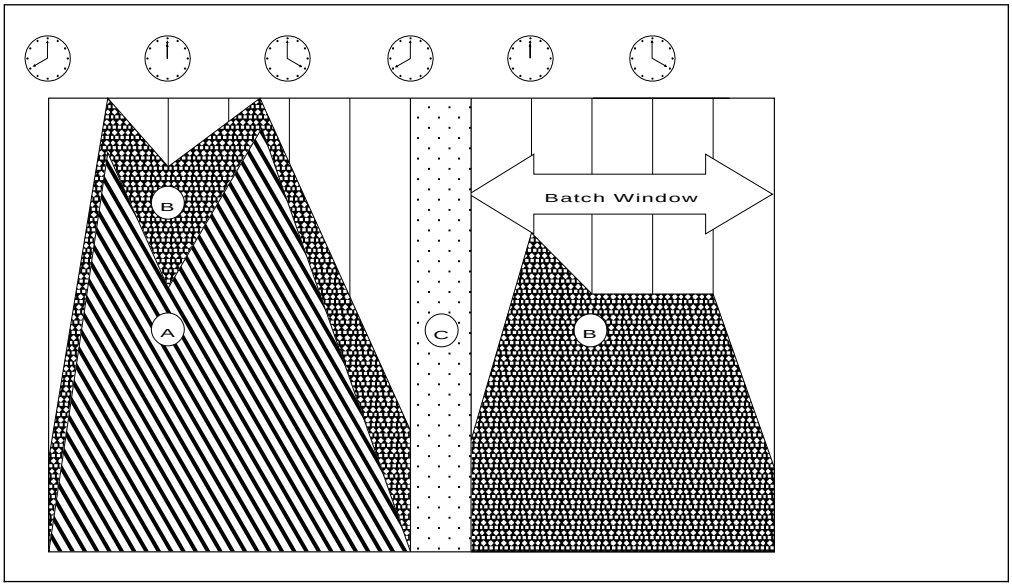

Figure 1-1. Overnight Batch Window

The need for online services to be available throughout the 24-hour day is, however, increasing, and the time required to run batch processing is also increasing. Taken together, these trends have created the following problems:

1. **A shrinking batch window**

   The need for online services to be available throughout the full 24-hour day has caused the batch window — the time available to do batch processing — to shrink. One reason is an increased number of online services. Another reason is improving customer service by expanding the hours of online services. For example, automatic teller machines (ATMs) must be available for most, if not all, of the 24-hour day. Another factor is online processing across several time zones. Dependencies on work from other time zones often means that local batch processing cannot start until all online work in all time zones has completed.

2. **Increased batch processing**

   At the same time, batch processing is taking longer to run. Increased business volume means the batch jobs must process more data. New applications, new services, and new reporting requirements also contribute to the increased time that batch processing requires.

The increased demands for 24-hour online services as well as the increased volume of batch processing is, in effect, squeezing the overnight batch window. If one day's batch work must complete before the next day's online services can begin, the problem of how to complete the batch workload is particularly acute.

Before attempting to solve the problem of how to complete a larger amount of batch processing in a shrinking batch window, let's first look at some characteristics of batch processing. Jobs that run during the batch cycle:

- Are typically I/O bound.
- Frequently access all of a given data set sequentially.
- Access the same data set concurrently with other jobs.

As a result, batch jobs can cause significant contention for a given device and require much time to do repetitive I/O operations to the same data set, tying up both processor and I/O resources, as well as causing other jobs to wait, either for the active job to complete or for the active job to free a device.

Some possible solutions include:

- Redesigning and rewriting batch applications.
- Using the fastest possible DASD.
- Using cached DASD control units.
- Making multiple copies of data sets on multiple devices.
- Restricting the number of jobs that can run concurrently.
- Scheduling jobs to avoid contention.
- Traditional tuning activities, such as increasing the block size or buffering for a data set.

These solutions, however, are either labor-intensive, expensive, or both, and might still not solve the problem.

One solution is to maintain the data in processor storage (processor storage = central storage + expanded storage).

When data resides in processor storage, data transfers can take place in microseconds as compared to the milliseconds required to transfer the data from I/O devices. Data in processor storage eliminates I/O operations to DASD. Thus, jobs run faster, there is less data set contention, and the DASD holding the data set is free to serve other jobs wanting to access other data sets on the same device.

Hiperbatch* is designed to maintain data in expanded storage.

Hiperbatch

Hiperbatch builds on existing MVS/ESA* architecture and provides new function that allows more data to remain in expanded storage for a longer period of time. This approach reduces I/O operations and I/O device contention. Reducing I/O time reduces batch processing time.

Hiperbatch is designed to eliminate the problems caused by:

1. **Multiple jobs requesting data from the same data set simultaneously.** Each job request causes I/O operations to the device holding the data; the more jobs that access the data set concurrently, the more I/O operations, the more contention for the device, and the longer the wait time for each request.
2. **Jobs passing temporary or short-lived data sets to subsequent jobs.** As a job completes, it puts the data it used back onto DASD. Subsequent jobs must perform I/O operations to once again retrieve that data from DASD.

Hiperbatch is an extension to both QSAM and VSAM designed to improve the elapsed time required for batch processing by keeping as much as possible of the QSAM or VSAM data in expanded storage and managing shared access to that data. Once the data resides in expanded storage, QSAM or VSAM application programs can access the data without actual I/O operations. This strategy can significantly reduce the elapsed time the application programs require by reducing both contention and the number of I/O operations.

One important aspect of Hiperbatch is that installations can take advantage of these performance benefits without having to change existing application programs or the JCL required to run them.

Hiperbatch depends on the data lookaside facility (DLF), to control access to expanded storage. When DLF is active, the first attempt to access a QSAM or VSAM data set defined to DLF creates a DLF object. A **DLF object** contains data from a single data set managed by Hiperbatch. The user (an application program) is connected to the DLF object, and the connected user can then access the data in the object through normal QSAM or VSAM macro instructions.

When subsequent users access the data set, they are connected to the object. The system manages shared access to the DLF object in the same way it would manage shared access to the data set. When a user relinquishes access, DLF disconnects that user from the object.

A DLF object exists until there are no users of the data set, at which time DLF deletes the object. As long as there is at least one user of a data set the access pattern means that the DLF object exists.

However, if a batch job or job step creates a data set and passes it as input to another job or job step, there is not always one user of the data set, and the system would delete the DLF object. To prevent this automatic deletion of an object, define the data set to DLF as a retained DLF object. A **retained DLF object** is one that the system does not automatically delete when there are no users of the data set.

How well Hiperbatch works in a particular installation depends primarily on the characteristics of the batch processing performed. To decide whether or not Hiperbatch can resolve batch processing concerns at your installation, see Chapter 2, “Evaluating Batch Processing” on page 2-1.

Hiperbatch Restrictions

Before considering the use of Hiperbatch, note the following restrictions:

- Hiperbatch does not run on the 4381 9XE models.
- Hiperbatch works with checkpoint/restart when the data set is accessed by VSAM, as updated in DFSMS* 1.1.
- Hiperbatch does not work with DB2* tables.

Hiperbatch Requirements

Hiperbatch requires a processor that supports the Move Page facility. See the appropriate *MVS/ESA SP V5 Program Directory: JES3* for a list of processors and the corresponding SEC numbers for the Move Page facility.

Questions and Answers

Does Hiperbatch require RACF 1.9?*

No. If your installation installs RACF 1.9, you can use RACF profiles to name the data sets that are eligible for Hiperbatch and to provide an additional level of security for the data while it is in storage. (All releases of RACF provide security for the data sets.) If your installation does not include RACF 1.9, the DLF installation exit can name the data sets that are eligible for Hiperbatch and verify that a particular user is allowed to access the data while it is in storage.

Does Hiperbatch require any particular JES release?

No. To use Hiperbatch, you do not need a particular release of JES2 or JES3.

Our installation currently charges users based on CPU use and I/O use. Hiperbatch reduces I/O use; how does it affect CPU use?

When Hiperbatch is working effectively, CPU use does go up significantly, but it is more concentrated. For example, a batch job stream that uses 25% of the processor cycles over a period of 4 hours might, with Hiperbatch, use 37% of the processor cycles for a period of three hours.

The EXCP counts in SMF records used to record I/O use do not change; the counts reflect I/O operations requested regardless of whether a request is satisfied by a physical I/O operation or from a DLF object in expanded storage. New fields in SMF record types 14, 15, and 64 provide data about Hiperbatch processing. Thus, existing accounting techniques should work, though you might want to explore alternative methods.

How does Hiperbatch affect DASD use?

Hiperbatch can reduce I/O activity as well as the need to use DASD space for multiple copies of the same data set. In some cases, these advantages might reduce demand for DASD space. Using Hiperbatch might also reduce the need to tune the I/O configuration by carefully spreading I/O activity over several devices to avoid contention.

What is the difference between using Hiperbatch and the batch LSR subsystem?

Refer to “Hiperbatch and Other Techniques” on page 2-5 for a discussion of the differences.

Chapter 2. Evaluating Batch Processing

If your installation has batch processing concerns that you think Hiperbatch might solve, the first step is to evaluate your batch processing. This process involves determining if your data sets are good candidates for Hiperbatch and comparing Hiperbatch to other techniques for reducing batch processing time.

Data Sets and Hiperbatch

One key factor in evaluating the value of Hiperbatch is the data sets in your batch workload. Data sets that are candidates for Hiperbatch are:

1. Data sets that reside on DASD.
2. Data sets accessed by QSAM or VSAM non-shared resource (NSR) access methods. All VSAM data sets (KSDS, ESDS, and RRDS) with a control interval of 4096 bytes, or a multiple of 4096 bytes, are eligible.

   Note: If the data set is a VSAM key sequenced data set (KSDS), the DLF object contains only the data component, not the index component. For optimum performance in this case, place the volume containing the index component on a device connected to a cached control unit.
3. Physical sequential data sets.
4. Data sets that are initially read or written sequentially. Only data sets that are initially written sequentially can place data in a retained DLF object.

Note: Hiperbatch is designed for batch processing. It is possible, however, that TSO/E users, like batch jobs, might also benefit from Hiperbatch. If many TSO/E users read from or write to and then read again from the same data set, defining this data set to Hiperbatch could reduce the time required for I/O activity.

In contrast, data sets that are not candidates for Hiperbatch are:

1. Data sets that two or more systems might access in the same general time frame. Hiperbatch depends on normal QSAM and VSAM serialization mechanisms to provide serialization. These mechanisms (shareoptions 1 or 2 for VSAM) provide data sharing within a single system. There is no mechanism for sharing data in expanded storage across multiple systems. To avoid data integrity exposures, Hiperbatch does not process a VSAM data set with shareoptions 3 or 4, even if that data set has been defined as eligible for Hiperbatch.
2. Data sets that reside on tape. (If there are multiple jobs that later read the tape data set, a possible solution might be to write the tape data set to DASD and define the data set to Hiperbatch as a retained object to keep the data for subsequent jobs).
3. Data sets not accessed by QSAM or VSAM NSR.
4. Partitioned data sets or members of partitioned data sets.
5. Data sets that are initially read from or written to in a random manner.

6. Data sets accessed by QSAM that contain only a single block.

   Note: A data set that is to be processed by QSAM but is open for MOD will not have the data being 'MODed' populate the cache. This data may be placed in the cache if read in a later step or by another job.

7. VSAM data sets with the number of strings (ACBSTRNO) value greater than 1.
8. VIO data sets.
9. VSAM catalog data sets.
10. Data sets allocated as dummy data sets (DD DUMMY).
11. VSAM data sets opened by any name other than the name of the base cluster. If any errors are encountered while trying to determine this, message IEC516I is displayed on the operator console. All caching for VSAM data sets will be disabled for the duration of the IPL.
12. VSAM data sets accessed through local shared resources (LSR) access methods are ineligible for Hiperbatch if the ACB is opened for input, or create-load. When updating the data set through LSR, the dataset is still eligible as long as other jobs are accessing the data set through NSR.
13. VSAM data sets cataloged in VSAM (non-ICF) catalogs.

To help you determine which data sets might be good candidates for Hiperbatch, answer questions like:

How much of the batch processing time, in general, comes from DASD delay?

What are the most frequently-used data sets? Are they VSAM NSR or QSAM (and thus candidates for Hiperbatch)?

How many jobs read these data sets during the batch run?

What is the maximum number of jobs that read these data sets concurrently?

How much time does DASD contention add to the average I/O request?

Once you have identified the data sets that are likely to benefit from Hiperbatch, you must then decide if Hiperbatch is to process each data set as a retained DLF object or as a non-retained DLF object.

Retained DLF Objects: Retained DLF objects are especially useful for data sets that are created by one job or job step and subsequently read by one or more other jobs or job steps. A retained object remains in even when there is no connected user. You can define a DLF object as a retained object only when the data set is opened for writing (that is, when the data set is being created). In other words, only jobs writing to DASD can place data in a retained DLF object. These jobs can be sequential QSAM writers or jobs that use VSAM load processing.

Once data exists in the retained DLF object, VSAM random readers or random updaters can access it, but neither random readers nor random updaters can place data in the object. If a VSAM random updater changes data in the data set, that change is also made to the DLF object.

Defining a data set that is shared among multiple systems as a retained DLF object can create a data integrity exposure. If a job on one system places the data in a retained object, the object exists even when there are no users of the data set. When there are no users, a job on another system could access the data set and change the data in the data set, but Hiperbatch does not know about the change. It does not update the data in the retained DLF object. Subsequent users of the data in the DLF object thus access data that does not match the data in the physical data set.

See “Keeping Data in Expanded Storage” on page 2-4 for related information.

If Hiperbatch detects an error during VSAM load processing for a retained DLF object and the error causes the cache to be invalidated and released, the object will be treated as non-retained.

Non-Retained DLF Objects: Non-retained DLF objects work best when data is accessed sequentially by multiple concurrent readers. Non-retained objects exist only as long as there is at least one user of the data set. That is, the object is deleted when the last user closes the data set, thus freeing the expanded storage frames occupied by the object. Jobs reading data sequentially from DASD can place data in the object. Both sequential and random readers can obtain data from the object. If a VSAM random updater changes data in the data set, that change is also made to the DLF object. If a subsequent user opens a QSAM data set for random update (using BSAM or EXCPVR, for example), Hiperbatch deletes the DLF object even if there are other users connected to the object.

Deleting DLF Objects: Deleting DLF objects that are no longer required for jobs returns essential system resources to the system. Non-retained objects are automatically deleted when the last user closes the data set. Retained DLF objects continue to exist after the last user closes the data set and must be deleted in one of the following ways:

- When the data set is recreated.
- When the data set is deleted or renamed.
- When the data set is updated by an access method (such as BSAM) that does not support Hiperbatch.
- When the object is explicitly deleted (See “Explicitly Deleting DLF Objects” on page 3-7).

Changing the Batch Workload

The benefits of Hiperbatch depend on the nature of the data sets that batch jobs process, and these benefits do not require changes to the batch workload. You might find, however, that it is also a good idea to analyze the batch workload.

Analyzing your batch workload might lead you to identify situations where you can, with some small JCL modifications or run procedure changes, use Hiperbatch or realize even greater benefits from Hiperbatch. Some examples of these situations are:

1. Jobs that share data sets but that do not run concurrently. Hiperbatch normally discards the data from expanded storage when no jobs are using the data set.

If a subsequent job accesses the data set, physical I/O operations are required to read the data set again. See “Keeping Data in Expanded Storage.”

2. Jobs that read data sets in a random manner. When a data set is initially read randomly, Hiperbatch cannot place the data in expanded storage. See “Random Access Data Sets.”
3. Jobs that do not use QSAM or VSAM. Hiperbatch works only with QSAM and VSAM. See “Copying and Sorting” on page 2-5.

Keeping Data in Expanded Storage

There are two ways to keep data in expanded storage even when there are no users, which is particularly useful for read-only data sets.

One way is to define the data set as a retained DLF object, described under “Retained DLF Objects” on page 2-2. With this technique, the data remains in expanded storage until the retained object is explicitly deleted. Using this technique means that the expanded storage is not available for other uses, and it requires explicit action to delete the object. See “Explicitly Deleting DLF Objects” on page 3-7.

Another way is to modify the batch job stream to ensure that the data set is always in use by at least one job. For example, create a job that opens the read-only data set and then enters a wait state (“goes to sleep”), perhaps by issuing a long STIMER wait. This sleeper job ensures that the DLF object exists, and it requires little overhead. Because the DLF object is a non-retained object, the expanded storage is available if needed for other Hiperbatch uses. When all the jobs that use the data set finish, stopping the sleeper job causes Hiperbatch to delete the object, freeing its expanded storage for other use.

Random Access Data Sets

Hiperbatch can place data in a DLF object only when the job reads or writes the data set sequentially. Once data is in the DLF object, then jobs can read the data from the object both randomly and sequentially.

If your batch workload includes a data set that might benefit from Hiperbatch but is initially read in a random manner, first make sure that you are looking at the job that actually creates the data set. If the creating job uses VSAM or QSAM, you might be able to process the data set as a retained DLF object. Otherwise, consider using the following process to make the data set available to Hiperbatch:

1. Rename the data set.
2. Allocate a new data set with the original name.
3. Run a job that reads the data set renamed in step 1 and copies it into the newly-allocated dataset (with the original name). During the process, Hiperbatch places the data in expanded storage. Use the Access Method Services REPRO command to read the data.

If the data set is very large, the time it takes to create the copy might outweigh the potential benefits.

Copying and Sorting

Key elements of many batch jobs are copying and sorting. Two programs that are typically used to accomplish these are IEBGENER and ICEMAN (DFSORT). However, IEBGENER and DFSORT do not support Hiperbatch.

For copying, the DFSORT routine, ICEGENER, or the Access Method Services REPRO command are alternatives to IEBGENER; both support Hiperbatch.

For sorting, many batch jobs use a sort package, such as DFSORT, to sort the contents of data sets. DFSORT does not use QSAM. Thus, if a previous step used Hiperbatch to place the data in expanded storage, the sort step would still need to do I/O to read the data set. Similarly, if a subsequent step needs the SORTOUT data set, it must do I/O to read the data set. With Hiperbatch, elapsed time could be saved if the SORTIN and SORTOUT operations were done with QSAM.

DFSORT provides two user exits that you can code to cause DFSORT to use QSAM, thus avoiding physical I/O for sort steps. The E15 exit can be used to read in the input data set. The E35 exit can be used to write the output data set.

These DFSORT user exits can be used to manipulate or print the input data before it is sorted or the output data after it is sorted. They use QSAM as an access method and therefore enable the use of Hiperbatch. While JCL changes are required in addition to the effort of coding the exit routine, the results might well be worth the effort. See *DFSORT Application Programming Guide* for more information.

Note: Hipersorting* and Hiperbatch perform two different functions. Hipersorting uses expanded storage for temporary space to save intermediate results while sorting a data set. Hiperbatch uses expanded storage to pass SORTIN and SORTOUT data sets from one step or job to another.

Hiperbatch and Other Techniques

While Hiperbatch can provide significant benefits, it is not the only technique available to reduce batch processing time. Three such techniques, which this book briefly describes, are cached DASD, VIO, and the batch LSR subsystem.

Cached DASD

The IBM* 3990 Model 3 Storage Control Unit comes with up to 256 megabytes of cache, including 4 megabytes of non-volatile cache to support the DASD Fast Write feature. You might ask “Why would I want to use Hiperbatch if I have a 3990-3?” The answer is that they address different problems.

In general, the best performance for reading data sequentially from DASD has been achieved by defining large buffer pools and block sizes, so that data is retrieved from DASD into a buffer in processor storage before it is needed. Cached DASD has been used primarily to improve the performance of random data reads, and reads from data sets with small block sizes. With 3990-3 Extended Function, cached controllers can also improve application performance with DASD writes.

The kinds of things cached DASD does well are:

- Provide faster response times for data sets that are read randomly. The required data might be in the 3990 cache from a previous read.
- Provide faster response times for data sets that are sequentially read with blocks of less than half a track in size. Data sets are cached a track at a time.
- For 3990-3s with the DASD Fast Write feature, provide faster response time for DASD writes from the point of view of an individual job; the job does not have to wait until the physical I/O is completed.
- Provide faster access to data for concurrent users who are on separate MVS systems.

In other situations, cached DASD is not as useful:

- For sequential reads from a data set, the system bypasses the 3990 cache if the data transfer involves more than about two tracks of data. In this situation, using the cache would not improve performance — the data is already in the buffer in processor storage by the time it is needed.

  For sequential reads that do not bypass the cache, only three to four tracks of data are kept in the 3990 cache. This fact is important in batch processing because most data sets are read sequentially. If a data set is read by multiple jobs, it is highly unlikely that the jobs will be reading the data set within three tracks of each other. Thus, the DASD cache does not help; each job must perform physical I/O to the data set.
- When a data set is written sequentially, as mentioned before, the DASD Fast Write feature does make the I/O go faster, but, again, only three to four tracks of data are kept in the 3990 cache. When another step tries to read the data set, it requires physical I/O operations to the DASD; the information will not be in the 3990 cache.

Hiperbatch can help in both of these situations.

Combining DASD Fast Write with Hiperbatch can improve data set accesses:

1. If the data set is created in one step and read in a following step, then DASD Fast Write speeds up the I/O when the data set is written to DASD, and Hiperbatch speeds up the subsequent reads.
2. If the data set is opened only for input, the cached 3990 can speed up the initial I/O if the block size is small, and Hiperbatch can speed up the subsequent reads. Because there is only one reader, using Hiperbatch can avoid having many copies of parts of the same data set in the 3990 cache, thus increasing its efficiency.
3. Hiperbatch does not place the indexes of VSAM data sets in expanded storage. Put these indexes on DASD backed by cache to improve the elapsed time of jobs reading the indexes.

Note: Do not use the IDCAMS SETCACHE command to deactivate DASD Fast Write or read cache while Hiperbatch jobs are running. Issuing IDCAMS SETCACHE stops Hiperbatch from writing to expanded storage and causes the job to run longer.

VIO

Both VIO and Hiperbatch can be used to pass data sets from one step of a job to another and avoid I/O operations, but there are several differences that might make Hiperbatch the best choice. These are:

- Hiperbatch works with VSAM data sets. VIO does not.
- Hiperbatch allows jobs or job steps to both pass and share data. VIO allows job steps to pass data but not to share data.
- Hiperbatch performs physical I/Os for all updates. VIO does no physical I/O. Thus, if a job fails in the middle of its execution, with Hiperbatch it can be restarted from the step that failed; with VIO, it might need to be restarted from the beginning, though VIO journaling can minimize the restart impact.
- Hiperbatch does not require virtual storage in the private address space. VIO, prior to MVS/DFP* 3.2, does.
- Hiperbatch can handle very large data sets. For example, Hiperbatch can process a QSAM data set of up to 6 gigabytes and the largest possible VSAM data set (4 gigabytes). VIO can process only a single-volume data set that fits on a logical 3380 base model (up to approximately .6 gigabytes).
- Hiperbatch requires an installation exit or RACF profile to authorize access to the data in the DLF object. VIO is specified through JCL by using a generic name and UNIT parameter on the DD statement or through use of DFSMS.

With Hiperbatch, there is central control of how expanded storage is used and less possibility of flooding expanded storage and auxiliary storage (the page data sets). Hiperbatch allows you to control the maximum amount of expanded storage used. Also, it does not use any auxiliary paging space; Hiperbatch uses the actual data set rather than auxiliary paging space.

The advantage of VIO is that it is easier to use, in that it is specified by users and does not require system-related activities, such as coding an installation exit routine or setting up a parmlib member. When you want to maintain central control of expanded storage use, or when there is a danger of overtaxing the paging subsystem, Hiperbatch is preferable. For situations where individual end user control is acceptable or desirable, VIO is an alternative.

Batch LSR

The batch LSR subsystem allows a batch job, which can normally access VSAM data sets only as VSAM NSR (non-shared resources), to access the data as if it were using VSAM LSR (local shared resources). All high-level languages currently use NSR for batch processing; LSR has been used only by data base managers.

The batch LSR subsystem has two advantages:

1. Using batch LSR requires only a simple JCL change.
2. Batch LSR provides better performance and elapsed time savings when the same control interval is referenced once and then referenced again later on in the processing and there are other requests for data between the first reference and the second reference. Batch LSR is particularly useful when, for example, a batch job makes semi-random accesses to a VSAM data set. More information about batch LSR can be found in *MVS Batch Local Shared Resources*.

There are cases, which often depend on the buffer hit ratio for the data set, where Hiperbatch might be a better alternative than batch LSR. The buffer hit ratio defines, for a given number of I/O operations, how often the data is found in storage. For example, sometimes the locality of reference to the data set is such that, to get a reasonable buffer hit ratio, the size of the buffer pool that must be defined approaches the size of the data set itself. Hiperbatch might also be better when the data set is used in multiple jobs or job steps or when it is possible to load the VSAM data set into expanded storage before running the job or job step. See "Random Access Data Sets" on page 2-4.

In summary, batch LSR and Hiperbatch are designed to solve two different types of data access problems:

- Batch LSR is designed to assist jobs that randomly access VSAM data and need a bigger buffer pool to reuse data.
- Hiperbatch is designed to assist jobs that concurrently and sequentially access the same data.

There is very little to be gained from using both on the same data set at the same time. Also, both can tolerate some write activity but provide the most benefits when the I/O activity is read-intensive.

Questions and Answers

Can Hiperbatch process a VSAM data set opened as skip sequential?

Yes.

What are the specific Hiperbatch requirements for a VSAM data set?

A VSAM data set is eligible for Hiperbatch processing when all of the following are true:

1. The data set is not accessed as VSAM LSR or GSR.
2. The data set is not accessed with shareoptions 3 or 4.
3. The control interval size is 4096 or a multiple of 4096.
4. The data set does not have user-managed buffers.
5. The data set is not a catalog.
6. The data set is not accessed through control interval processing.

Once a batch job stream has been set up to make effective use of Hiperbatch, will the job stream run on a system that does not have the hardware Hiperbatch requires?

A batch job stream "tuned" for Hiperbatch will run on a system that does not have the hardware Hiperbatch requires, but it will require more elapsed time, particularly if an installation changes the job stream significantly, such as changing job steps to allow more concurrent readers of a data set. These steps will run on the system without the required hardware, but there will be no performance benefits from Hiperbatch.

If a job or job stream that is using Hiperbatch abends, what happens to the data in retained DLF objects?

What happens to the data in a retained object depends on the disposition specified for the data set. If DISP=(NEW,KEEP,DELETE), the data is discarded. If DISP=(,KEEP,KEEP), then the data is kept. The data, however, is overwritten when the data set is opened again for output. As the job writes the data set, the data is placed in the DLF object.

When a VSAM user is making random updates to a data set accessed through Hiperbatch, when is the data set updated?

When a VSAM user makes random updates, the system updates the data set on DASD, then replaces the control interval in expanded storage if that control interval is currently in expanded storage. The updates are synchronous; that is, there is no delay between the update to the data set and the update to the data in expanded storage.

My job stream includes multiple files that are very large (larger than the amount of expanded storage installed) and passed from one job to another. Does Hiperbatch provide any potential benefits?

The benefits from Hiperbatch come from jobs being able to access data in expanded storage rather than perform physical I/O operations. If a data set passed from one job to another is defined to Hiperbatch as a retained DLF object, Hiperbatch can place in expanded storage only the portion of the data set that fits in the amount of expanded storage available for retained objects. Thus, the benefits of Hiperbatch in the situation you describe are not clearly predictable because I/O operations are avoided only for the portion of the data set that fits in expanded storage.

Hiperbatch, in contrast, might very well provide benefits for very large data sets that have multiple readers and are thus processed as non-retained DLF objects. “Monitoring Hiperbatch Activity” on page 3-4 describes techniques for evaluating the effectiveness of Hiperbatch once it is active on your system.

What happens if Hiperbatch detects an error?

If Hiperbatch encounters a system error, its recovery routine increments an error counter and attempts to recover. Once this counter reaches a system-defined limit of errors, Hiperbatch disables itself. This condition will not cause any data integrity problems, because all of the data in expanded storage is backed up by the physical DASD, and the system performs the I/O when Hiperbatch becomes disabled. Once Hiperbatch becomes disabled, it cannot be re-enabled until an IPL is done.

Chapter 3. Using Hiperbatch

Because Hiperbatch depends on the data lookaside facility (DLF) to create DLF objects and to control access to expanded storage, DLF must be active when applications are using Hiperbatch. Before you can activate DLF, you must:

- Identify the data sets you want Hiperbatch to process
- Set up system controls
- Plan operator procedures.

You might also need to set up procedures for explicitly deleting retained DLF objects. If a step in the batch job stream deletes the QSAM or VSAM data set, the system explicitly deletes the retained DLF object that corresponds to the data set. If no job step deletes the data set, your installation must delete the retained DLF object. See “Explicitly Deleting DLF Objects” on page 3-7.

Identify Eligible Data Sets

Once you have decided to use Hiperbatch for an application, DLF needs a data set name list. The data set name list contains the following information about data sets and users:

- The names of the eligible data sets — the QSAM or VSAM data sets that you want processed as DLF objects.
- The names of the eligible data sets that you want Hiperbatch to process as retained DLF objects.
- The users and/or jobnames allowed to access each DLF object.

For each data set in the data set name list, DLF creates a corresponding object with a name consisting of a 6-character volume serial number concatenated with the data set name. For example, the DLF object name for a data set named ACCOUNT.MASTER that resides on volume 222222 would be:

```
222222ACCOUNT.MASTER
```

If the volume serial number is less than six characters, the number is padded on the right with blanks in the DLF object name. For example, the DLF object name for a data set named ACCOUNT.MASTER that resides on volume 12345 would be:

```
12345 ACCOUNT.MASTER
```

There are several ways to provide the data set name list to DLF.

- If your system includes RACF 1.9, you can provide the information by defining RACF profiles in the DLFCLASS general resource class. See “Establishing RACF Profiles” on page 3-4 for more information.
- You can provide the information by coding a DLF installation exit routine. In the DLF installation exit routine, you can also set the maximum number of DLF objects that your installation allows at any one time. The default is 50.

  To change this value, you must code the DLF installation exit routine. It is important that this value not be too small. If DLF has created the maximum number of objects and is asked to create another, it rejects the request; an existing object must be deleted before DLF can create a new object.

See “Planning the DLF Connect/Disconnect Installation Exit Routine” on page 3-4 for more information.

- If your system includes Operations Planning and Control/ESA (OPC*/ESA), you can use OPC/ESA to identify eligible data sets. To define an eligible data set, you use the OPC/ESA special resource dialog, and OPC/ESA provides a DLF installation exit that verifies the data set name and the connection to a particular job.

  OPC/ESA uses job scheduling information to determine when to delete a DLF object; it deletes the object when there are no current users of the data set and no more scheduled jobs ready to use the data set.

  If you also use RACF DLFCLASS profiles with OPC/ESA, the DLF installation exit that OPC/ESA provides does not override the RACF decisions.

Changing the Data Set Name List: Once Hiperbatch is active, it is possible to change the data set name list. How to make the change depends on how you defined the list. Be aware, however, that dynamically changing the list can create a data integrity exposure. The exposure can occur under the following conditions:

- If a job is updating a data set that is dynamically added to the eligible list, Hiperbatch does not know about the updater and might then create a DLF object that contains back-level data.
- If a job is accessing data in a DLF object and the data set is dynamically removed from the eligible list, another job might then update the data set on DASD, but any job still accessing the data in the DLF object does not know about the updates.

Set up System Controls

To set up system controls, you must make other decisions about your use of Hiperbatch. Also, you must provide the COFDLFxx member of SYS1.PARMLIB and, because DLF runs as a started task, place the procedure for starting DLF in SYS1.PROCLIB.

Setting up the COFDLFxx Parmlib Member

Before you can start DLF, you must provide the COFDLFxx parmlib member. This member provides, through the CONEXIT keyword, the name of the DLF installation exit. Through the MAXEXPB and PCTRETB keywords, it sets limits on the amount of expanded storage that the system can use for Hiperbatch:

- MAXEXPB sets the maximum amount of expanded storage that Hiperbatch can use. IBM recommends that you set this number to a value that is less than the amount of expanded storage installed. When DLF is active, Hiperbatch use of expanded storage takes precedence over other uses.

  As a starting point for setting the value, you could add the sizes of the largest data sets that you expect Hiperbatch to process at any one time. The actual amount of expanded storage used is often much less than the size of the data sets, especially when multiple users move through the data concurrently. In this context, the size of a VSAM data set is a good indicator of the maximum amount of expanded storage the data set might require.

For QSAM data sets, however, Hiperbatch needs additional control information. How much expanded storage this control information requires depends on the block size of the data set. The ideal block size is slightly less than a multiple of 4K. Large block sizes also require minimal expanded storage for control information. In the worst case (a block size equal to or slightly larger than 4K), the DLF object might be as much as twice the size of the data set.

Regardless of the value you specify for MAXEXPB, Hiperbatch does not use any expanded storage until it actually places data in a DLF object. If all the expanded storage available to Hiperbatch contains data and a user tries to place additional data in a DLF object, Hiperbatch tries to satisfy the request by discarding data from non-retained objects. From each existing DLF object, it discards the data that is least likely to be used again soon.

- PCTRETB is the percentage of the expanded storage amount (defined in MAXEXPB) that is available for retained DLF objects, generally used to pass the output of one job or job step to another job or job step.

  Expanded storage used by a retained DLF object is not available for other uses until the retained object is deleted. If all the expanded storage defined for retained objects is in use, additional data requires physical I/O operations. If all the expanded storage defined for retained DLF objects is not in use, Hiperbatch can use the available expanded storage for non-retained DLF objects.

You cannot replace a DLF installation exit while DLF is active. The exit requested when DLF is started remains in effect for the duration of the DLF address space. To replace the exit, you must:

1. Stop DLF, as described in “Stopping DLF” on page 3-6.
2. Do either one of the following:
   - Replace the exit. For information about installing a new DLF exit, see the description of the DLF connect/disconnect exit in *MVS/ESA SP V5 Installation Exits*.
   - Specify a different exit on the CONEXIT parameter in the COFDLFxx parmlib member. For information about the use of the COFDLFxx parmlib member, refer to *MVS/ESA SP V5 Initialization and Tuning Reference*.
3. Start DLF again, as described in “Setting up the DLF Start Procedure” on page 3-4

Although only one COFDLFxx member can be active at a time, you can create several members with different limit values for the storage definition parameters, MAXEXPB and PCTRETB, then use the MODIFY system command to change the active parmlib member. One reason for setting up several COFDLFxx parmlib members is to allow you to switch to a member with different limits as necessary.

See *MVS/ESA SP V5 Initialization and Tuning Reference* for detailed information about creating COFDLFxx.

Monitoring Hiperbatch Activity

Once your installation is using Hiperbatch, there are several ways to obtain information about how well it is working:

- You can monitor the effectiveness of the values you specify for MAXEXPB and PCTRETB through the MODIFY DLF command with the STATUS operand. The resulting display shows the maximum amounts of expanded storage available and how much of this available storage DLF is currently using. The current use is shown in either megabytes or 4K blocks and as a percentage of the maximum.

  You can display a list of the current DLF objects and connected users through the DISPLAY DLF command.

- RMF Monitor II can report the amount of expanded storage that the system uses for DLF objects.
- The online monitor that IBM provides for use with Hiperbatch can track Hiperbatch activity on your system. See Chapter 4, “Hiperbatch Monitor” on page 4-1 for information about the availability of the monitor and how to use it.

Setting up the DLF Start Procedure

Starting DLF requires a DLF procedure. The following procedure is distributed with the system in SYS1.PROCLIB:

```
//DLF      PROC  NN=00
//DLFEXEC  EXEC  PGM=COFMISDO,PARM='NN=&NN'
//IEFPARM  DD    DISP=SHR,DSN=SYS1.PARMLIB
```

Once this procedure is available, the operator can issue the START command to activate DLF, or your installation can place the START command for DLF in the COMMNDxx parmlib member.

Planning the DLF Connect/Disconnect Installation Exit Routine

Before starting DLF, your installation can code the DLF connect/disconnect installation exit routine to provide the control information that DLF needs. If you define DLFCLASS profiles for RACF, you still might want to code the exit routine; its decisions can then override the information in the RACF profiles.

You also use the installation exit routine to specify the maximum number of DLF objects that can exist at any one time. The default is 50. SYS1.SAMPLIB contains two sample DLF installation exit routines: COFXDLF2, COFXUPDT, COFXMACS, that combined, make up one example, and COFXDLF1.

For more information about coding your own routine, see the description of the DLF connect/disconnect exit routine in *MVS/ESA SP V5 Installation Exits*.

Establishing RACF Profiles

Your existing security procedures for restricting access to data sets restrict access to the QSAM or VSAM data sets that are eligible for DLF processing. Thus, only users who have the authority to access the data sets can attempt to connect to a DLF object. If your system includes RACF 1.9, you can establish profiles in the DLFCLASS general resource class.

To RACF, the data set and the DLF object associated with it are separate resources; thus, a user allowed to access the data set needs separate authorization, either through a DLFCLASS profile or the DLF installation exit routine, to connect to the DLF object. Otherwise, the system might deny access to a DLF object but grant access to its associated data set.

DLFCLASS profiles provide the data set name list that DLF needs:

- The existence of a DLFCLASS profile for a QSAM or VSAM data set identifies that data set as one that is eligible to be processed as a DLF object.
- An optional field in the DLFDATA segment of the profile allows you to indicate to DLF whether the object is to be a retained DLF object.
- The access list for the profile identifies the users who are allowed to access the DLF object. An optional field in the DLFDATA segment of the profile allows you to further restrict access to the DLF object to specific jobnames:
  - When you specify a list of jobnames, access to the DLF object is allowed only when the userid appears in the access list and the specific jobname appears in the jobnames list.
  - When you do not specify a list of jobnames, all jobs submitted by authorized users (users identified in the access list of the profile) can access the DLF object.

If you use RACF DLFCLASS profiles to provide control information to DLF, time any changes to the profiles very carefully to avoid the data integrity exposure described under “Changing the Data Set Name List” on page 3-2.

If you do not provide control information in DLFCLASS profiles, your DLF installation exit routine must identify the eligible data sets and retained DLF objects and verify user and/or jobname access to a DLF object.

For more information about using DLFCLASS profiles with RACF 1.9, see *RACF V2 Security Administrator's Guide*.

Plan Operator Procedures

DLF runs as a started task. You can place the START command for DLF in the COMMNDxx parmlib member or have the operator issue the command. Note that the START command for DLF must always include the SUB=MSTR operand. If your installation is using parmlib member COFDLF00, the START command need not specify the name of the DLF parmlib member. IBM recommends that you start DLF before you start JES2 or JES3.

Once DLF is started, its services are available, and the operator can issue additional commands to control and monitor DLF processing. These commands allow the operator to change the active COFDLFxx member of SYS1.PARMLIB, stop DLF in an orderly fashion, and display DLF status or a list of DLF objects and their users.

MVS/ESA SP V5 System Commands contains detailed information about using the START, STOP, MODIFY, and DISPLAY commands to control and monitor DLF. Stopping DLF, however, is a process that requires some special considerations.

Stopping DLF

If you need to stop DLF, either because DLF itself has detected errors or for any other reason, ensure that DLF stops in an orderly fashion. DLF will not stop as long as any users are connected to any DLF objects. Establish a procedure to stop DLF based on the following considerations:

1. Establish procedures to prevent the initiation of any new work units that require DLF objects. Possible actions include, for example, altering JES initiators. Note that the QSAM and VSAM applications can continue to run without DLF, but stopping DLF can significantly affect the time required to run the batch workload.

2. Determine the DLF processing mode your installation needs to prepare to stop DLF. DLF has three processing modes: normal, drain, and quiesce. To change the processing mode, the operator issues the MODIFY DLF command with the MODE=NORMAL, MODE=DRAIN, or MODE=QUIESCE operand.

   After DLF is started, normal mode is in effect.
   In normal mode, DLF:

   - Connects users to new DLF objects
   - Connects users to existing DLF objects
   - Disconnects users from DLF objects

   Either drain mode or quiesce mode prepares DLF to stop. When the operator specifies drain or quiesce mode, the change in processing mode does not take effect until the operator issues the STOP DLF command.

   In drain mode, DLF:

   - Connects users to existing DLF objects
   - Disconnects users from DLF objects

   In quiesce mode, DLF:

   - Disconnects users from DLF objects

   As a general rule, select drain mode to prepare DLF to stop. Select quiesce mode only when you know that it is acceptable for some of the jobs that have not yet connected to existing DLF objects to run without connecting to these DLF objects. To use quiesce mode, you must understand the workload well enough to be sure that rejecting connections to existing DLF objects will not cause problems for the application.

3. Determine what action to take if DLF does not stop as expected.

   Issuing the STOP DLF command begins the process of stopping DLF. Using the stop mode in effect (either DRAIN or QUIESCE), DLF continues processing until there are no users connected to DLF objects. If there are no connected users, DLF stops immediately. If there are users connected to DLF objects, DLF issues an eventual action message to inform the operator that there are connections and to indicate the stop mode in effect. When there are no users connected to DLF objects, DLF stops automatically and deletes the message from the console.

If DLF does not stop within a reasonable amount of time, the operator can issue the DISPLAY DLF command to display a list of connected DLF objects and see what jobs are currently using the DLF objects. Based on the display, the operator could take several possible actions:

- Prevent connections to existing DLF objects by issuing MODIFY DLF,MODE=QUIESCE, if quiesce mode is not already in effect.
- Cancel jobs that are currently connected to DLF objects.
- Restore normal DLF processing by issuing MODIFY DLF,MODE=NORMAL. This command has the effect of cancelling the attempt to stop DLF.
- Take additional steps to explicitly delete DLF objects, as described in the following section.

To restart DLF once it has stopped, the operator issues the START DLF command.

Explicitly Deleting DLF Objects

If your installation uses retained DLF objects and if a DLF object created by one job or job step has not, because of a failure or some other consideration, been deleted by another job or job step, your installation might need to explicitly delete one or more objects before stopping DLF. In the display of connected DLF objects, a DLF object that exists but is no longer connected to a user appears as a connected DLF object, but the only user is DLF.

These retained DLF objects take up system resources. If either the operator or an installation-written program detects that a DLF object will not be accessed in the current batch window, it is good practice to delete the object. There are several ways to delete a DLF object that is no longer needed:

1. Add a step to the batch job stream to explicitly delete any retained DLF object. See “Setting Up the Required Procedure” on page 3-8. Once the procedure is in place, you can delete a retained DLF object with one of the following JCL EXEC statements:

   ```
   //        EXEC  PROC=DELOBJ,PARM='OBJ=volserdataset.name'
   //        EXEC  PROC=DELOBJ,PARM='V=volser,D=dataset.name'
   //        EXEC  PROC=DELOBJ,PARM='D=dataset.name,V=volser'
   ```

   If the data set is cataloged, you can delete a retained DLF object with the following JCL EXEC statement:

   ```
   //        EXEC  PROC=DELOBJ,PARM='D=dataset.name'
   ```

   In this case, you do not need to supply the volume serial number to delete the DLF object.

2. The operator can explicitly delete any retained DLF object. See “Setting Up the Required Procedure.” Once the procedure is in place, the operator can supply the name of the object to be deleted and invoke the procedure through one of the following examples of the START command:

   ```
   START DELOBJ,OBJ='volserdataset.name'
   START DELOBJ,V='volser',D='dataset.name'
   START DELOBJ,D='dataset.name',V='volser'
   ```

If the data set is cataloged, the operator can delete a retained DLF object with the following example of the START command:

```
START DELOBJ,D='dataset.name'
```

In this case, the operator does not need to supply the volume serial number to delete the DLF object.

3. If your application has an authorized termination routine that performs tasks such as cleaning up resources, you can modify that routine to issue the COFSDONO macro, which explicitly deletes retained DLF objects. To use COFSDONO, you must know the complete DLF object name (the volume serial number concatenated with the data set name) to delete the object. For more information about COFSDONO, see *MVS/ESA SP V5 Auth Assembler Services Reference ALE-DYN*.

Before trying to delete a DLF object, verify that there are no connected users. The deletion process does not complete until there are no connected users. For example, if a program issues COFSDONO to delete a DLF object that has at least one connected user, control does not return to the program until the user has been disconnected from the object, which must occur before the object can be deleted.

Setting Up the Required Procedure: COFMSTCN is an IBM-supplied program that deletes a named DLF object using the COFSDONO macro. COFMSTCN deletes only the DLF object; it does not affect the data set on DASD. To make this program available to batch jobs or to operators, create a procedure in SYS1.PROCLIB that invokes the program with the appropriate parameters. The procedure you create depends on the parameters you want to use to identify the DLF object:

- If you use the OBJ parameter to specify the DLF object name, model your procedure after the following sample:

```
//DELOBJ  PROC  OBJ=''
//        EXEC  PGM=COFMSTCN,PARM='OBJ=&OBJ'
```

- If you use the V or D parameter, or a combination of both, model your procedure after the following sample:

```
//DELOBJ  PROC  OBJ='',V='',D=''
//        EXEC  PGM=COFMSTCN,PARM='OBJ=&OBJ,V=&V,D=&D'
```

Note: It is not a good idea to invoke COFMSTCN from a TSO/E terminal, through a CLIST, for example. If the named object has a connected user, control does not return to the terminal until the user has been disconnected.

Interpreting COFMSTCN Return Codes: COFMSTCN returns one of the following hexadecimal return and reason code combinations:

Return Code	Reason Code	Meaning and Action
00	00	**Meaning**: Successful completion. The DLF object has been deleted. **Action**: None.
08	00	**Meaning**: Program error. The system was not able to determine the object name. **Action**: Verify the syntax of the invocation.
08	04	**Meaning**: Program error. The specified data set is not in the system catalog. **Action**: Verify that the specified data set exists and is cataloged.
28	00	**Meaning**: Environmental error. DLF is not active. **Action**: Issue the START DLF command and rerun the job.

In some cases, the return and reason code combination comes from the COFSDONO macro. If you receive codes that are not described in the preceding table, see the description of COFSDONO in *MVS/ESA SP V5 Auth Assembler Services Reference ALE-DYN*.

Chapter 4. Hiperbatch Monitor

IBM provides an online monitor that you can use to track Hiperbatch activity on your system. Through the monitor, you can evaluate:

How Hiperbatch is using expanded storage

The use patterns for individual data sets

The I/O operation savings for individual jobs

For information about how to install the Hiperbatch monitor, see “Installing the Hiperbatch Monitor” on page 4-14. Once it is installed, you can invoke the monitor under TSO/E by issuing the following command:

```
COFDMON
```

For information about how to invoke the data collection services the monitor uses, see “Hiperbatch Monitor Data Collection” on page 4-15.

Using the Hiperbatch Monitor

Once the Hiperbatch monitor is installed, it provides four data displays:

Global status display — provides an overview of how Hiperbatch is using expanded storage. See “Global Status Display” on page 4-4.

Bird's-eye view — provides a high-level view of the use pattern for up to ten individual data sets. See “Bird's-Eye View Display” on page 4-5.

Zoom display — provides a more detailed view of the use pattern for one data set. See “Zoom Display” on page 4-8.

Job (SMF) data display — provides information about the users of a single data set and how effectively Hiperbatch is reducing physical I/O operations to that data set. See “Jobs Display” on page 4-11.

There are two menus that allow you to control the processing of the Hiperbatch monitor. See“Main Menu” on page 4-2 and “Freeze Menu” on page 4-3.

If you want to get started quickly, use the following summary procedures.

Getting Started

The following procedures describe the actions you will need most frequently as you use the Hiperbatch monitor.

Starting the Monitor

To begin using the monitor:

1. Issue COFDMON under TSO/E. The main menu appears.
2. Optionally, change the default automatic update interval (5 seconds). You can change this interval only from the main menu.
3. Select a display. Type the letter that corresponds to the display you want, then press the enter key.

Freezing a Display

To suspend automatic updating to examine a particular data display more closely:

1. Press the attention key from the data display. The freeze menu appears.
2. Type F from the freeze menu, then press the enter key.

To resume automatic updating:

1. Press the attention key from the data display. The freeze menu appears.
2. Select a display from the freeze menu, then press the enter key.

Changing the Display

To change from one display to another:

1. Press the attention key from the current display. The freeze menu appears.
2. Select the display you want from the freeze menu, then press the enter key.

Stopping the Monitor

To end the monitor from a data display:

1. Press the attention key. The freeze menu appears.
2. Select X from the freeze menu.

To end the monitor from the main menu, select X or press PFK 3 or PFK 15.

Main Menu

When you issue COFDMON to invoke the monitor, the first screen is the main menu, shown in Figure 4-1.

```
----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
  ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
    ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
      ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
        ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
          ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
        ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
      ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
    ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
  ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>

SELECT THE DESIRED DISPLAY:

G ===> GLOBAL STATUS DISPLAY              **********************
B ===> BIRD'S-EYE VIEW                    * PRESS ATTN FROM ANY *
Z ===> ZOOM DISPLAY                       *  DISPLAY TO RETURN  *
J ===> JOB (SMF DATA) DISPLAY             *    TO  THIS MENU    *
                                          **********************
X ===> EXIT (OR PFK 3 / PFK 15)

                           AUTOMATIC UPDATE INTERVAL (IN SECONDS) ===> 005
```

Figure 4-1. Main Menu

The main menu allows you to select a particular display. It also allows you to specify the automatic update interval. The monitor assumes that you are most interested in tracking a particular display over time. Once you select a display, the monitor will refresh that display (update it automatically), using the interval you choose or the default interval, which is 5 seconds. Whenever the main menu is on your screen, you can select a different interval simply by typing the value you want over the value displayed.

To select a particular display from the main menu, type the letter that corresponds to the display you want, then press the enter key.

Freeze Menu

While you are viewing data displays, the monitor updates them automatically. To suspend automatic updating — freeze — a display, press the attention key when the display is on the screen.

In response, the monitor presents the freeze menu, shown in Figure 4-2.

```
|
----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
  ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
    ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
      ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
        ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
          ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
        ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
      ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
    ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
  ----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>
----------=====>>>>  HIPERBATCH STATUS MONITOR  ----------=====>>>>>

SELECT THE DESIRED DISPLAY:

G ===> GLOBAL STATUS DISPLAY              **********************
B ===> BIRD'S-EYE VIEW                    * PRESS ATTN FROM ANY *
Z ===> ZOOM DISPLAY                       *  DISPLAY TO RETURN  *
J ===> JOB (SMF DATA) DISPLAY             *    TO  THIS MENU    *
F ===> FREEZE THE PREVIOUS DISPLAY        **********************
X ===> EXIT
```

Figure 4-2. Freeze Menu

The freeze menu looks very much like the main menu, but, depending on your terminal, it might appear with different colors or highlighting. There are two main differences between the main menu and the freeze menu:

1. You cannot change the automatic update interval from the freeze menu.
2. There is an option that allows you to freeze the previous display, to suspend automatic updating so that you can examine the display more carefully.

To freeze a display, press the attention key from the display you want to examine more carefully. The monitor presents the freeze menu. Select F, then press the enter key to return to the previous display.

To resume automatic updating, press the attention key from the display. The monitor presents the freeze menu. If you select the display you want, the monitor presents the display and resumes automatic updating. If you press the enter key from the freeze menu, the monitor presents the main menu, which allows you to change the automatic update interval.

Global Status Display

Select the global status display by typing G from the main menu or the freeze menu.

The data display enables you to evaluate how effectively Hiperbatch is using the expanded storage available for its processing. Figure 4-3 shows a sample display.

```
89354 12:01:17 -*-*-*- HIPERBATCH STATUS MONITOR DISPLAY *-*-*-*-*-*-*-*-* GLBL

CURRENT / MAXIMUM NUMBER OF DATA SET ENTRIES          4 /      24
CURRENT NUMBER OF RETAINED DLF OBJECTS                2

TOTAL AMOUNT OF ESTORE ONLINE (K-BYTES)          32,768
MAXIMUM ESTORE AVAILABLE TO HIPERBATCH            6,144
AMOUNT OF ESTORE IN USE BY HIPERBATCH             5,972  OR   97%

MAXIMUM AMOUNT OF RETAINED STORAGE                2,456
IN-USE AMOUNT OF RETAINED STORAGE                 2,444  OR   99%

MAX ESTORE AVAIL FOR NON-RETAIN STORAGE           3,700
IN-USE AMOUNT OF NON-RETAINED STORAGE             3,528  OR   95%

DLF ADDRESS SPACE IDENTIFIER (ASID)                0041 (HEX)  /   65 (DEC)
```

Figure 4-3. Global Status Data Display

The display includes the following information:

- Information about the DLF objects that Hiperbatch is using, including the current number of DLF objects, the maximum possible number, and the number of current objects that are retained.
- Information about how Hiperbatch is using expanded storage, including both numeric values (all of which are expressed in kilobytes) and percentages that show how much of the expanded storage available to Hiperbatch is actually in use, as a total and broken down into storage for retained objects and storage for non-retained objects.
- The address space identifier for DLF.

Bird's-Eye View Display

Select the bird's-eye view display by typing B from the main menu or the freeze menu.

The bird's-eye view display allows you to examine the use pattern for several data sets.

The monitor first supplies a selection screen that allows you to select the data sets you want to display. The selection screen shows the maximum number of data sets you can select, which depends on the size of your terminal screen, and lists the data sets that Hiperbatch is currently processing. Figure 4-4 shows a sample selection screen.

```
 *-*-*-*-*-*-*-*-* HIPERBATCH STATUS MONITOR DISPLAY -*-*-*-*-*-*-*-*-* SLCT

TYPE 'S' TO SELECT (UP TO)  6 OF THE FOLLOWING DATA SET(S):

 s QSAMTEST.RETAIN                              PRESS ENTER  TO CONTINUE
 _ VSAMTEST.RETAIN                                    PF3/15 TO EXIT
 s QSAMTEST.NORETAIN                                  PF1/13 TO IDENTIFY OTHER
 s VSAMTEST.NORETAIN                                         DATA SETS
```

Figure 4-4. Bird's-Eye View Selection Screen

To select one or more data sets from the list, type S in front of each data set name, then press the enter key. To select one or more data sets not included in the list, press PFK 1 or PFK 13 and type the name of each data set, then press the enter key. When you select a data set not included in the list, the monitor begins to display data about the data set and its use as soon as Hiperbatch begins to process the data set.

To select one or more data sets from the list and one or more data sets not included in the list, type S to select each listed data set, then press PFK 1 or PFK 13. Type the name of each additional data set, then press the enter key to view the display or press PFK 7 to return to the selection menu. Returning to the selection menu allows you to see the entire list before viewing the display.

The monitor displays information about the data sets you selected. Figure 4-5 shows a sample display.

```
89354 12:06:41   QSAMTEST.RETAIN                                                   53
A R   BUFSIZE    #USERS   #RANGES     #ES-ALLOC /  #ES-HBATCH     EACH-POS
Q Y    24,576         0         1         1,248 /       6,144            1
 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>




89354 12:06:41   QSAMTEST.NORETAIN                                                561
A R   BUFSIZE    #USERS   #RANGES     #ES-ALLOC /  #ES-HBATCH     EACH-POS
Q      24,576         4         3         2,640 /       6,144            8
>>>>>>>>>>>                      >11>2




89354 12:06:41   VSAMTEST.NORETAIN                                         16,588,800
A R   CI-SIZE    #USERS   #RANGES     #ES-ALLOC /  #ES-HBATCH     EACH-POS
V       4,096         4         6         1,056 /       6,144      209,985
>>                     >>   >  > >>4
```

Figure 4-5. Bird's-Eye View Data Display

For each data set selected, the monitor displays four lines of information.

The first line includes:

Date The current date in Julian form, such as 89354 (December 20, 1989)

Time The current time on a 24-hour clock, in the form *hh.mm.ss*, such as 12:06:41

Data set name The name of the data set.

Current size The current size of the data set.

For a QSAM data set, this value is the high relative block number (RBN), where each unit is equal to the block size for the data set (shown in BUFSIZE). For example, QSAMTEST.RETAIN has a high RBN of 53.

For a VSAM data set, this value is the high relative byte address (RBA), in units of bytes. For example, VSAMTEST.NORETAIN has a high RBA of 16,588,800.

The next two lines present additional information about the data set:

A The access method for the data set:

Q QSAM
V VSAM

R An indicator of whether the data set is defined to Hiperbatch as a retained object. If **Y** appears, the data set is retained. Otherwise, the data set is not retained.

BUFSIZE For a QSAM data set, the buffer size, in bytes. The buffer size is the same as the block size.

CI-SIZE For a VSAM data set, the control interval (CI) size, in bytes.

#USERS The number of users currently accessing the data set through Hiperbatch. This number includes all users that have opened the data set, even though they might not be currently accessing the DLF object.

#RANGES Number of contiguous ranges of expanded storage that represent the current copy of the data set in Hiperbatch storage.

#ES-ALLOC Amount of expanded storage (in kilobytes) allocated to this data set.

#ES-HBATCH Amount of expanded storage (in kilobytes) allocated to Hiperbatch. This number is the same for all data sets.

EACH-POS Defines the value of each position in the following graphic data line.

For a QSAM data set, the position value is in units equal to the block size for the data set.

For a VSAM data set, the position value is in units of bytes.

The fourth line, the graphic data line, shows the current use pattern for the data set. Each position contains one of the following characters:

Blank This part of the DLF object contains no data.

Arrow (>) This part of the DLF object contains data.

Number The number (1 to 9) of users accessing the data at this location in the data set.

Asterisk (*) An indication that there are at least 10 users accessing the data at this location in the data set.

The information in the bird's-eye view display can help you determine whether the data sets Hiperbatch is currently processing are good choices. For example, in Figure 4-5 on page 4-6, the data set QSAM.NORETAIN shows four users in the same general location. In the figure, each 1 indicates a single user; 2 indicates there are two users processing the data at the same location. A group of users in the same location means that Hiperbatch is reducing the number of physical I/O operations.

In contrast, if you saw a data set with many ranges but no groups of users, the condition might indicate either that there is not enough expanded storage available for non-retained objects or that the data set itself is not a good candidate for Hiperbatch. Initiator restrictions or jobs forced to start at different times because of inter-job dependencies can also cause the condition.

Retained objects require a slightly different evaluation. For example, Figure 4-5 on page 4-6 shows QSAMTEST.NORETAIN as having no current users. Watch such a data set over time; you would expect to see one or more users appear and work through the data set sequentially. If no users appear, the data set is taking up expanded storage but providing no benefit; it might not be a good candidate for Hiperbatch.

Especially for large data sets, you might need more detail to make a complete evaluation. To examine a data set in more detail, press the attention key to display the freeze menu, then select the zoom display.

Zoom Display

Select the zoom display by typing Z from the main menu or the freeze menu.

The zoom display allows you to examine the use pattern for a single data set at a detailed level. You normally use the zoom display when the bird's-eye view indicates a condition that you need to investigate.

The monitor first supplies a selection screen that allows you to select the data set you want to investigate. The selection screen lists the data sets that Hiperbatch is currently processing. Figure 4-6 shows a sample selection screen.

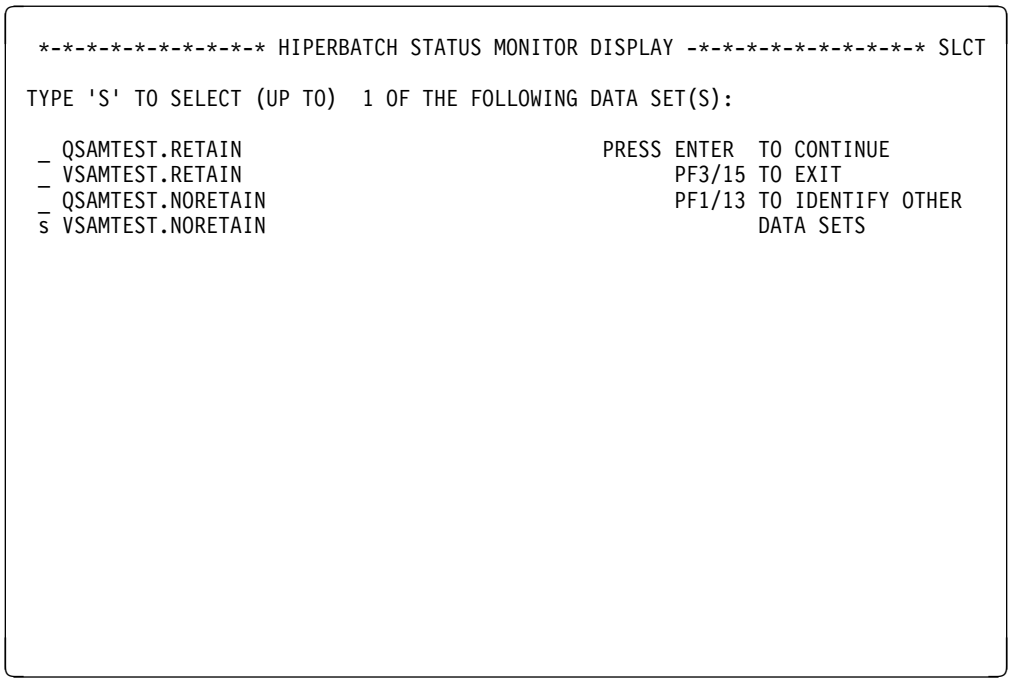

Figure 4-6. Zoom Selection Screen

To select a data set from the list, type S in front of each data set name, then press the enter key. To select a data set not included in the list, press PFK 1 or PFK 13 and type the name of the data set, then press the enter key. When you select a data set not included in the list, the monitor begins to display data about the data set and its use as soon as Hiperbatch begins to process the data set.

The monitor displays information about the data set you selected. Figure 4-7 shows a sample display.

```
89354 12:11:12 -*- HIPERBATCH STATUS MONITOR DISPLAY OF DATA SET *-*-*-*-* ZOOM
                              VSAMTEST.NORETAIN                       ON KREPAK
A R   CI-SIZE    #USERS   #RANGES     #ES-ALLOC /  #ES-HBATCH     EACH-POS
V       4,096         4         4           968 /       6,144       20,999
----+----+----+----+----+----+----+----+----+----+----+----+----------1,658,880
>>>>>>>>>>>>>>>>
----+----+----+----+----+----+----+----+----+----+----+----+----------3,317,760

----+----+----+----+----+----+----+----+----+----+----+----+----------4,976,640
                                                                      >>>
----+----+----+----+----+----+----+----+----+----+----+----+----------6,635,520
>>>>>>>>>>>     >>>>1>>  1>>>>2>>>
----+----+----+----+----+----+----+----+----+----+----+----+----------8,294,400

----+----+----+----+----+----+----+----+----+----+----+----+----------9,953,280

----+----+----+----+----+----+----+----+----+----+----+----+---------11,612,160

----+----+----+----+----+----+----+----+----+----+----+----+---------13,271,040

----+----+----+----+----+----+----+----+----+----+----+----+---------14,929,920

----+----+----+----+----+----+----+----+----+----+----+----+---------16,588,800
```

Figure 4-7. Zoom Data Display

For the selected data set, the monitor displays information about the data set and ten lines of graphic data that describe the current use of the data set.

The first two lines identify:

Date The current date in Julian form, such as 89354 (December 20, 1989)

Time The current time on a 24-hour clock, in the form *hh.mm.ss*, such as 12:06:41

Data set name The name of the data set. In Figure 4-7, the data set name is VSAMTEST.NORETAIN.

Volume The volume serial number of the volume on which the data set resides. In Figure 4-7, the volume serial number is KREPAK.

The next two lines present additional information about the data set:

A The access method for the data set:

- **Q** QSAM
- **V** VSAM

R An indicator of whether the data set is defined to Hiperbatch as a retained object. If **Y** appears, the data set is retained. Otherwise, the data set is not retained.

BUFSIZE For a QSAM data set, the buffer size, in bytes. The buffer size is the same as the block size.

CI-SIZE For a VSAM data set, the control interval (CI) size, in bytes.

#USERS The number of users currently accessing the data set through Hiperbatch. This number includes all users that have opened the data set, even though they might not be currently accessing the DLF object.

#RANGES Number of contiguous ranges of expanded storage that represent the current copy of the data set in Hiperbatch storage.

#ES-ALLOC Amount of expanded storage (in kilobytes) allocated to this data set.

#ES-HBATCH Amount of expanded storage (in kilobytes) allocated to Hiperbatch.

EACH-POS Defines the value of each position in the following graphic data lines. Because the zoom display is more detailed, the value is smaller than the value in the bird's-eye view display for the same data set.

For a QSAM data set, the position value is in units equal to the block size for the data set (shown in BUFSIZE).

For a VSAM data set, the position value is in units of bytes.

Ten lines of graphic data show the current use pattern for the data set. Each position in each line contains one of the following characters:

Blank This part of the DLF object contains no data.

Arrow (>) This part of the DLF object contains data.

Number The number (1 to 9) of users accessing the data at this location in the data set.

Asterisk (*) An indication that there are at least 10 users accessing the data at this location in the data set.

At the end of each line, the monitor provides the RBN (for a QSAM data set) or the RBA (for a VSAM data set) that corresponds to the data displayed in the line.

Jobs Display

Select the jobs display by typing J from the main menu or the freeze menu.

Select the jobs display when you need information about the particular jobs that are accessing a data set or want information about how effectively Hiperbatch is reducing physical I/O operations for a particular data set. The information about a job is similar to the information about I/O operations that SMF collects for a job.

The monitor first supplies a selection screen that allows you to select the data set you want to investigate. The selection screen lists the data sets that Hiperbatch is currently processing. Figure 4-8 shows a sample selection screen.

```
 *-*-*-*-*-*-*-*-*-* HIPERBATCH STATUS MONITOR DISPLAY -*-*-*-*-*-*-*-* SLCT

TYPE 'S' TO SELECT (UP TO)  1 OF THE FOLLOWING DATA SET(S):

 _ QSAMTEST.RETAIN                              PRESS ENTER TO CONTINUE
 _ VSAMTEST.RETAIN                                    PF3/15 TO EXIT
 _ QSAMTEST.NORETAIN                                  PF1/13 TO IDENTIFY OTHER
 s VSAMTEST.NORETAIN                                         DATA SETS
```

Figure 4-8. Jobs Selection Screen

To select a data set from the list, type S in front of the data set name, then press the enter key. To select a data set not included in the list, press PFK 1 or PFK 13 and type the name of the data set, then press the enter key. When you select a data set not included in the list, the monitor begins to display data about the data set and its use as soon as Hiperbatch begins to process the data set.

The monitor displays information about the data set you selected. Figure 4-9 shows a sample display.

```
89354 12:14:14 -*- HIPERBATCH STATUS MONITOR DISPLAY OF DATA SET *-*-*-*-* JOBS
                              VSAMTEST.NORETAIN                        ON KREPAK
A R   CI-SIZE    #USERS   #RANGES     #ES-ALLOC /  #ES-HBATCH   HIGH-USED-RBA
V       4,096         4         4           968 /       6,144      16,588,800
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
JOBNAME     CURRENT-RBA READ-RQTS  READ-OBJ-%%% NUM-WAITS PHYS-I/OS P-I/O-OBJ
VN$$R404      6,324,224        73        41  56         3        32        31
VN$$R402      6,324,224        73        66  90         0         7         7
VN$$R403      6,324,224        73        57  78         0        16        16
VN$$R401      6,324,224        73        55  75         0        18        18
```

Figure 4-9. Jobs Data Display

The jobs data display has two parts; header information about the data set and detailed information about each of the jobs currently accessing the data set.

There are four lines of header information that describe the data set.

The first two lines identify:

Date The current date in Julian form, such as 89354 (December 20, 1989)

Time The current time on a 24-hour clock, in the form *hh.mm.ss*, such as 12:06:41

Data set name The name of the data set. In Figure 4-9, the data set name is VSAMTEST.NORETAIN..

Volume The volume serial number of the volume on which the data set resides. In Figure 4-9, the volume serial number is KREPAK.

The next two lines present additional information about the data set:

A The access method for the data set:

Q QSAM
V VSAM

R An indicator of whether the data set is defined to Hiperbatch as a retained object. If **Y** appears, the data set is retained. Otherwise, the data set is not retained.

BUFSIZE For a QSAM data set, the buffer size, in bytes. The buffer size is the same as the block size.

CI-SIZE For a VSAM data set, the control interval (CI) size, in bytes.

#USERS The number of users currently accessing the data set through Hiperbatch. This number includes all users that have opened the data set, even though they might not be currently accessing the DLF object.

#RANGES Number of contiguous ranges of expanded storage that represent the current copy of the data set in Hiperbatch storage.

#ES-ALLOC Amount of expanded storage (in kilobytes) allocated to this data set.

#ES-HBATCH Amount of expanded storage (in kilobytes) allocated to Hiperbatch.

HIGH-USED-RBA The current size of a VSAM data set. This value is the high relative byte address (RBA), shown in units of bytes. In Figure 4-9 on page 4-12, VSAMTEST.NORETAIN has a high RBA of 16,588,800.

HIGH-USED-RBN The current size of a QSAM data set. This value is the high relative block number (RBN), in units equal to the block size of the data set (shown in BUFSIZE).

The detailed information includes the following, for each of the jobs currently accessing the DLF object:

JOBNAME The job name of the user. Any job that has closed the data set or started to process the data set randomly is not included.

CURRENT-RBA The user's current location within the data set. For a VSAM data set, the location is expressed as the current RBA, in units of bytes.

CURRENT-RBN The user's current location within the data set. For a QSAM data set, the location is expressed as the current RBN, in units equal to the block size of the data set (shown in BUFSIZE).

READ-RQTS The number of times this job has requested a read operation from the data set.

READ-OBJ The number of read requests for this job that were satisfied by accessing the data in the DLF object rather than accessing the data on DASD.

%%% The percentage of the total number of read requests that were satisfied by accessing data in the DLF object. This percentage indicates how successfully Hiperbatch processing is reducing I/O operations for this job.

NUM-WAITS The number of times that Hiperbatch detected concurrent I/O operations and caused this job to wait while another user completed the physical I/O operation to or from DASD.

PHYS_I/OS The number of times that this job performed the physical I/O operation to or from DASD.

P-I/O-OBJ The number of times that physical I/O operations caused Hiperbatch to place data in the DLF object. This number reflects the I/O operations done by the first job that placed data in the DLF object.

Installing the Hiperbatch Monitor

The Hiperbatch Monitor is an IBM supplied example of using the Hiperbatch data collection services. The Hiperbatch monitor consists of several source modules contained in SYS1.SAMPLIB. The modules are:

```
COFDMON
COFDBIRD
COFDJOBS
COFDZOOM
COFDSLCT
COFDGLBL
```

These modules are the data display modules for the monitor.

To make the Hiperbatch monitor available on your system, you must perform the following steps:

Step 1: Assemble and link-edit the data display modules. You must use Assembler H to assemble and link-edit each of the six data display modules separately. Figure 4-10 shows sample JCL for the assemble and link-edit steps for one module, COFDMON. In the sample, underscores identify the fields that you must change to make the sample JCL fit each of the modules.

```
//JOBNAME  JOB job-card-info
//*********************************************************************
//*               ASSEMBLE  STEP                                      *
//*********************************************************************
//ASM     EXEC PGM=IEV90,REGION=2048K,
//  PARM='RENT,SYSPARM(JBB3313)'
//SYSIN    DD  DSN=SYS1.SAMPLIB(COFDMON),DISP=SHR
//SYSPUNCH DD  DSN=COFDMON.OBJ,DISP=(NEW,PASS),
//  SPACE=(TRK,(10,10)),
//  DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//*********************************************************************
//*               LINKEDIT  STEP                                      *
//*********************************************************************
//LKED    EXEC PGM=IEWL,COND=(8,LE,ASM),
//  PARM='NCAL,LIST,XREF,LET,RENT,AC=1,RMODE=24,AMODE=24',REGION=256K
//SYSPRINT DD SYSOUT=*
//SYSLMOD  DD DSN=ANY.AUTHLIB.LOAD,DISP=(OLD,KEEP)
//SYSUT1   DD UNIT=SYSDA,SPACE=(1024,(200,20))
//SYSUT2   DD UNIT=SYSDA,SPACE=(1024,(200,20))
//OBJ1     DD DSN=COFDMON.OBJ,DISP=SHR
//SYSLIN   DD *
     INCLUDE OBJ1
     NAME    COFDMON(R)
/*
//
```

Figure 4-10. Sample JCL to Assemble and Link-edit Data Display Modules

Notes:

1. Ensure that the macro library, shown as SYS1.MACLIB in the sample JCL, includes the Hiperbatch monitor mapping macros for the SYSLIB DD statement.
2. Place the output of the link-edit step in an authorized program library, shown as ANY.AUTHLIB.LOAD in the sample.
3. As shown in the sample, link-edit each of the modules with RMODE=24 and AC=1.

Step 2: Make COFDMON available to TSO/E users. To make the Hiperbatch monitor available to users under TSO/E, you must add COFDMON to the TSO/E authorized commands table. Add the following statement to the IKJTSOxx member of SYS1.PARMLIB:

```
AUTHCMD NAME(COFDMON)
```

Product-Sensitive Programming Interface

Hiperbatch Monitor Data Collection

To meet the needs of your installation, you can modify the data display modules of the Hiperbatch monitor. The data display modules, which reside in SYS1.SAMPLIB, invoke the DLF data collection services to collect requested data. Modified data display modules must do the following to use the data collection function:

1. Complete the Hiperbatch monitor parameter list, mapped by the COFZHMPL mapping macro. Figure 4-11 shows key fields in the parameter list and their relationship to the answer area you supply and the data areas defined in the parameter list for each anticipated data set.
2. Obtain pageable private area storage for the answer area, mapped by the COFZHMAA mapping macro. Figure 4-12 shows the format of the answer area for each type of request and the relationship between data areas within the answer area.
3. Obtain storage for the high-used-RBA/RBN block. Figure 4-11 shows the format and relationship of this block, which corresponds to DSECT HMPHR in the COFZHMPL mapping macro. Note that this area is optional; providing this area, however, improves the performance of the monitor.
4. Place the address of the parameter list mapped by HMPL in register 1.
5. Using standard linkage conventions, pass control to COFDSTAT, using either the LINK macro or the LOAD macro followed by a BASSM instruction (to pass control to COFDSTAT).

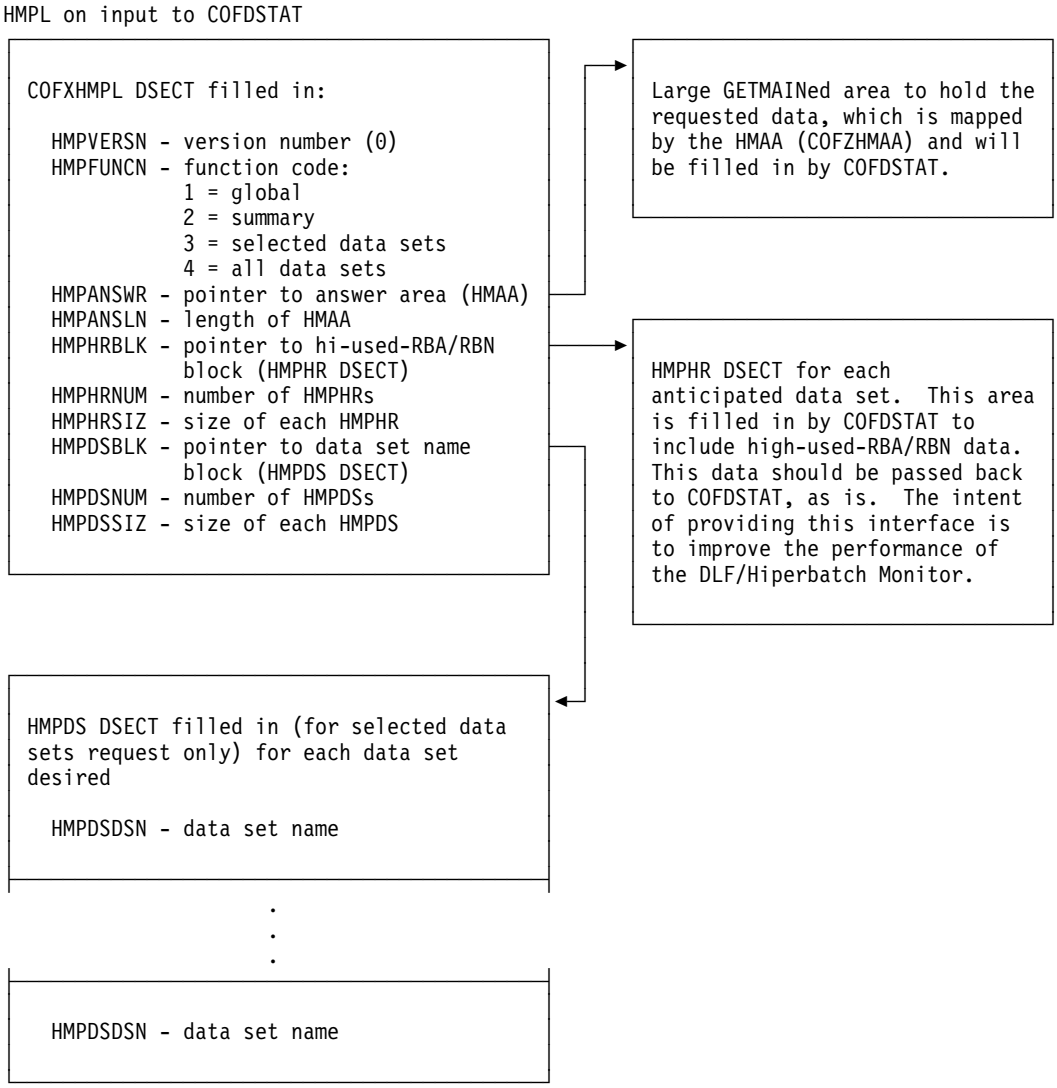

Figure 4-11. Hiperbatch Monitor Parameter List (HMPL)

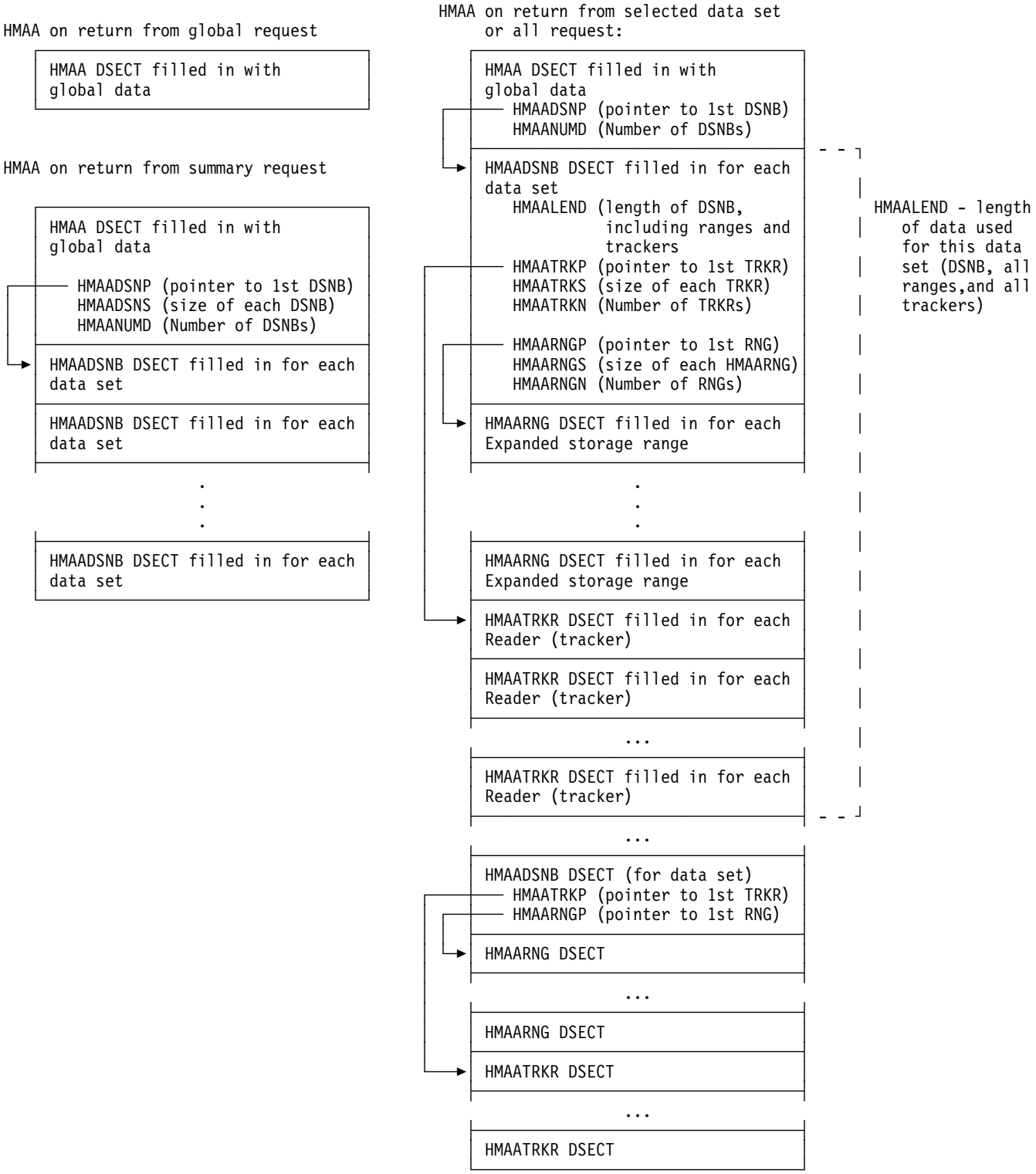

Figure 4-12. Hiperbatch Monitor Answer Area (HMAA)

End of Product-Sensitive Programming Interface

Index

I

J

M

N

O

Q

R

S

T

V

Z

Communicating Your Comments to IBM

MVS
Programming: Hiperbatch Guide

Publication No. GC28-1470-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing an RCF from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
  - FAX: (International Access Code)+1+914+432-9405
- If you prefer to send comments electronically, use this network ID:
  - IBMLink: (United States customers only): KGNVMC(MHVRCFS)
  - IBM Mail Exchange: USIB6TC9 at IBMMAIL
  - Internet e-mail: mhvrcfs@us.ibm.com
  - World Wide Web: http://www.s390.ibm.com/os390

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies

Optionally, if you include your telephone number, we will be able to respond to your comments by phone.

Reader's Comments — We'd Like to Hear from You

MVS
Programming: Hiperbatch Guide
Publication No. GC28-1470-00

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Today's date: _____

What is your occupation?

Newsletter number of latest Technical Newsletter (if any) concerning this publication:

How did you use this publication?

[ ]	As an introduction	[ ]	As a text (student)
[ ]	As a reference manual	[ ]	As a text (instructor)
[ ]	For another purpose (explain)		

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

Name

Address

Company or Organization

Phone No.

Reader's Comments — We'd Like to Hear from You
GC28-1470-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie NY 12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

File Number: S370/S390-40
Program Number: 5655-068
5655-069

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC28-1470-00