# CA IDMS™

# DML Reference Guide for Assembler

## Release 18.5.00, 2nd Edition

## CA Technologies Product References

This document references the following CA products:

- CA IDMS™/DB
- CA ADS™
- CA IDMS™/DC
- DC/UCF
- CA IDMS™ UCF

## Contact CA Technologies

**Contact CA Support**

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

**Providing Feedback About Product Documentation**

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

# Documentation Changes

The following documentation updates were made for the 18.5.00, 2nd Edition release of this documentation:

- @COPY IDMS (see page 411)—Added the conditions which cause the DML to define record elements using the Assembler EQU instruction.
- Output from the Precompiler (see page 457)—The output from the DML precompiler has been updated.
- Output from the Assembler (see page 467)—The output from the Assembler has been updated.
- IDMS Communications Block (see page 34), 18-Byte Communications Blocks (see page 541)—Updated the tables and field descriptions.

The following documentation updates were made for the 18.5.00 release of this documentation:

- @ACCEPT DBKEY FROM CURRENCY (see page 85)—Added the PGINFO parameter to this statement.
- @ACCEPT DBKEY RELATIVE TO CURRENCY (see page 87)—Added the PGINFO parameter to this statement.
- @Ready (see page 308)—The description of the FORCE option was added.
- Online Debugger Syntax (see page 543)—This new appendix was previously available in the Programming Quick Reference Guide.

# Contents

# Chapter 1: Introduction

This guide presents navigational and LRF DML statements for use in CA IDMS/DB and CA IDMS/DC and CA IDMS UCF data communications environments.

Most *data communications* DML statements are applicable in both CA IDMS/DC and CA IDMS UCF environments. The acronym DC/UCF is used to represent this.

This guide is intended for Assembler language programmers who run programs against CA IDMS/DB databases and who want to use the DC/UCF system facilities.

This section contains the following topics:

## Syntax Diagram Conventions

The syntax diagrams presented in this guide use the following notation conventions:

`UPPERCASE OR SPECIAL CHARACTERS`

Represents a required keyword, partial keyword, character, or symbol that must be entered completely as shown.

`lowercase`

Represents an optional keyword or partial keyword that, if used, must be entered completely as shown.

*`italicized lowercase`*

Represents a value that you supply.

**`lowercase bold`**

Represents a portion of the syntax shown in greater detail at the end of the syntax or elsewhere in the document.

◄─

Points to the default in a list of choices.

►►────────────────────

Indicates the beginning of a complete piece of syntax.

────────────────────►◄

Indicates the end of a complete piece of syntax.

────────────────────►

Indicates that the syntax continues on the next line.

►────────────────────

Indicates that the syntax continues on this line.

```
──────────────────────►─
```

Indicates that the parameter continues on the next line.

```
─►──────────────────────
```

Indicates that a parameter continues on this line.

```
►── parameter ─────────►
```

Indicates a required parameter.

```
►───┬─ parameter ─┬─────►
    └─ parameter ─┘
```

Indicates a choice of required parameters. You must select one.

```
►───┬─────────────┬─────►
    └─ parameter ─┘
```

Indicates an optional parameter.

```
►───┬─────────────┬─────►
    ├─ parameter ─┤
    └─ parameter ─┘
```

Indicates a choice of optional parameters. Select one or none.

```
   ┌─────────────┐
►─▼─ parameter ─┴──────►
```

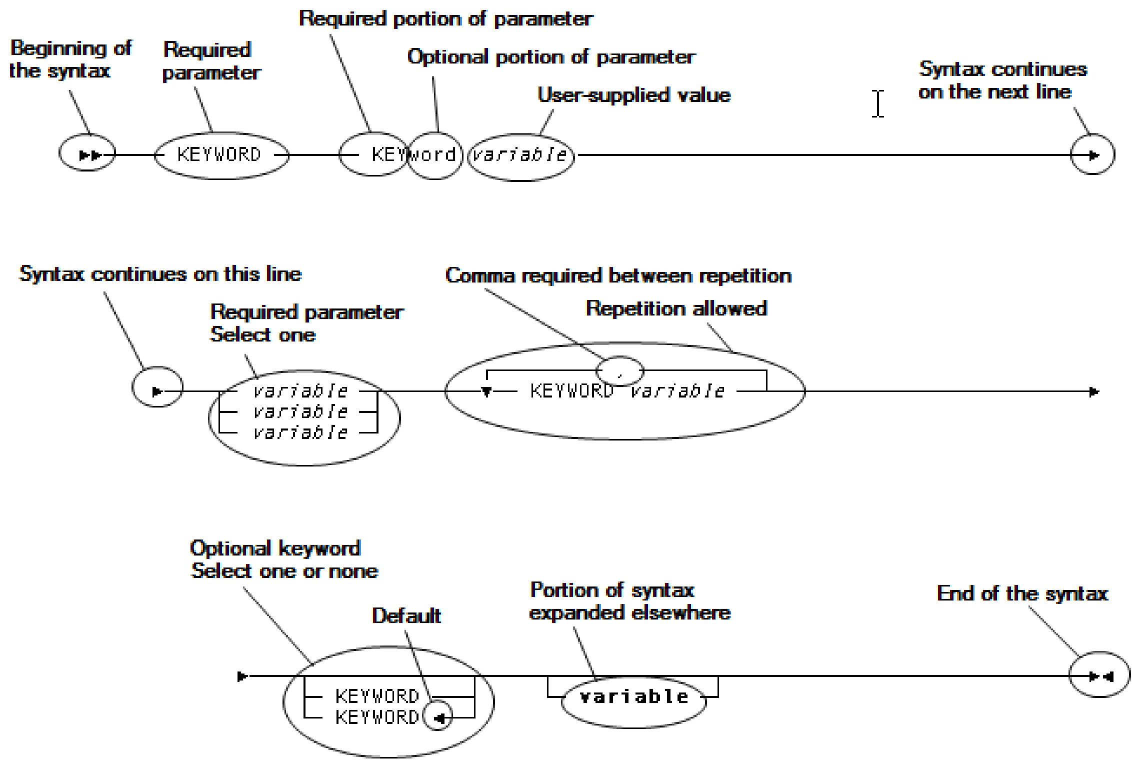
Indicates that you can repeat the parameter or specify more than one parameter.

```
   ┌─── , ───────┐
►─▼─ parameter ─┴──────►
```

Indicates that you must enter a comma between repetitions of the parameter.

## Sample Syntax Diagram

The following sample explains how the notation conventions are used:


Beginning of the syntax
Required parameter
Required portion of parameter
Optional portion of parameter
User-supplied value
Syntax continues on the next line
KEYWORD
KEYword
variable
Syntax continues on this line
Required parameter Select one
Comma required between repetition
Repetition allowed
Optional keyword Select one or none
Default
Portion of syntax expanded elsewhere
End of the syntax

# Chapter 2: Introduction to CA IDMS Data Manipulation Language

This guide discusses how to use Assembler **Data Manipulation Language (DML)** statements in your Assembler program to perform the following:

- Access a CA IDMS/DB database
- Perform data communications functions through CA IDMS/DC and CA IDMS UCF (DC/UCF)

Assembler DML statements are embedded in the program source as if they were part of the host language. During assembly, *most* **DML precompiler** statements are expanded into executable Assembler source code (whether or not the DML precompiler was executed), and source-level error checking is performed.

Depending on your operating environment, your Assembler program uses different sets of DML statements. For example, a batch program uses database DML statements; an online program can use both database and data communications DML statements.

This chapter discusses the following:

- When to use different sets of Assembler DML statements depending on your operating environment

How to use the DML precompiler to prepare your program for assembly and execution

This section contains the following topics:

## Operating Environments

This manual presents the following categories of Assembler DML statements:

- **Database** statements perform CA IDMS/DB database access functions in either a batch or an online environment. Database DML statements have an at sign (@) prefix; for example, @STORE.
- **Data communications**, also called **online** statements, perform data communications functions for CA IDMS/DC and CA IDMS UCF (DC/UCF) programs. Online DML statements have a pound sign (#) prefix; for example, #LINK.

- **DC-batch** statements are a subset of online DML statements that allow batch application programs to access DC/UCF facilities such as queues and printers. This category consists of the following DML statements: #DELQUE, #GETQUE, #PUTQUE, and #PRINT.

  **Note:** For more information about DC-batch programming, see the *Navigational DML Programming Guide*.

## Accessing the Database

Your program can access a CA IDMS/DB database by using either navigational or LRF (logical record) DML statements:

- **Navigational** statements access database records and sets one record at a time.
- **LRF** statements access predefined groups of database records using the Logical Record Facility (LRF).

Navigational and LRF DML statements are discussed separately below.

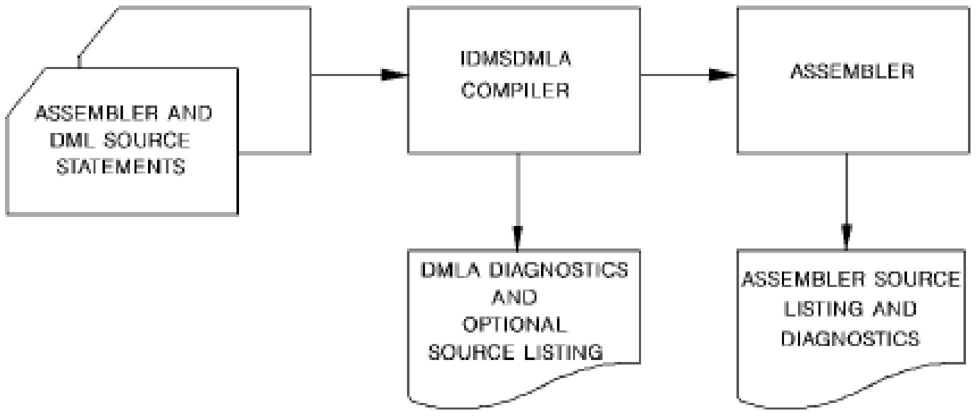
**Navigating the Database**

Navigational DML statements access database records and sets one record at a time, checking and maintaining currency in order to assure correct results. Navigational DML statements provide:

- **Control over error checking**—You can check the result of each navigational statement
- **Flexibility in choosing how you want to access the database**—For example, your program can access the database either sequentially (performing an area sweep), by using a symbolic key value (CALC), or by using a database key value (DIRECT)

To use navigational DML statements, you must have a thorough knowledge of the database structure. The database structure is illustrated in a data structure diagram. For an example of a data structure diagram, see the EMPLOYEE Data Structure Diagram.

The following figure illustrates a database structure that contains two owner records (EMPLOYEE and JOB) that share one member record (EMPOSITION). To obtain EMPLOYEE and JOB information, the program must retrieve an EMPLOYEE record, the first EMPOSITION record in the EMP-EMPOSITION set, and the owner record in the JOB-EMPOSITION set.


ASSEMBLER AND DML SOURCE STATEMENTS
IDMSDMLA COMPILER
ASSEMBLER
DMLA DIAGNOSTICS AND OPTIONAL SOURCE LISTING
ASSEMBLER SOURCE LISTING AND DIAGNOSTICS

Navigational DML statements are grouped into four categories:

- **Control** statements initiate and terminate processing, effect recovery, prevent concurrent updates, and evaluate set conditions
- **Retrieval** statements locate data in the database and make it available to the application program
- **Modification** statements update the database
- **Accept** statements pass database keys, storage address information, and statistics to the program

**Accessing the Database Through LRF**

LRF DML statements use the Logical Record Facility (LRF) to access database records. LRF accesses fields from multiple database records as if they were data fields in a single record. LRF DML statements allow your program to specify selection criteria (by using the WHERE clause) that enable your program to access only the logical records you need.

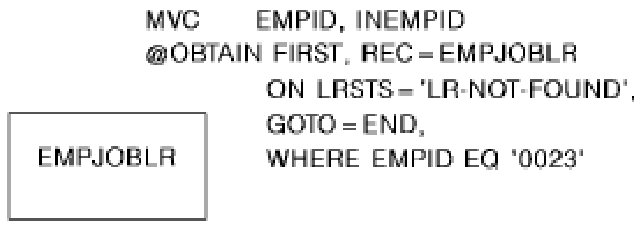
Note: For more information, see the *Logical Record Facility Guide*.

LRF DML statements provide:

- **Easy access to database records**—You need not be familiar with database structure, and your programs need not include database navigation logic.

- **Data flexibility**—You do not usually have to modify or recompile your LRF program when the database is changed.
- **Runtime efficiency**—LRF minimizes communication between the program and the database management system (DBMS).

The following figure illustrates how to use LRF DML statements to access the EMPJOBLR record. The EMPJOBLR record is a logical record that contains the EMPLOYEE record, the EMPOSITION record, the OFFICE record, and the JOB record. The EMPJOBLR logical record contains information from the EMPLOYEE, EMPOSITION, and JOB records.



```
MVC      EMPID, INEMPID
@OBTAIN FIRST, REC=EMPJOBLR
               ON LRSTS='LR-NOT-FOUND',
               GOTO=END,
               WHERE EMPID EQ '0023'
```

The LRF DML statements are:

- **@ERASE** deletes a logical record from the database.
- **@MODIFY** updates a logical record.
- **@OBTAIN** retrieves a logical record.
- **@STORE** adds a new logical record to the database.

## Programming in the DC/UCF Environment

DC/UCF application programs can use both database and online DML statements.

Online DML statements perform the following types of functions:

- **Program management** statements govern flow of control and abend processing
- **Storage management** statements allocate and release variable storage
- **Task management** statements provide runtime services that control task processing
- **Time management** statements obtain the time and date and define time-related events
- **Scratch management** statements create, delete, or retrieve records from the scratch area
- **Queue management** statements create, delete, or retrieve records in a queue area

- **Terminal management** statements transfer data between the application program and a terminal
- **Utility function** statements retrieve task-related information or statistics, send messages, and monitor access to database records
- **Recovery** statements perform functions relating to database, scratch, and queue area recovery in the event of a system failure

**Example**

The following example illustrates how online DML statements access the database and perform data communications functions. Specifically, this example maps in data entered from the terminal, retrieves and displays the specified information, and performs a DC return, naming TSK02 as the next task to be performed.

```
#MREQ IN,MRB=EMPMAP,INDATA=YES,COND=ALL,ERROR=ERRORTN
#MREQ OUT,MRB=EMPMAP,OUTDATA=YES,OPTNS=NEWPAGE
#RETURN NXTTASK=TSK02
```

## Assembling and Executing Programs

An Assembler source program that contains DML statements is processed by the DML precompiler (IDMSDMLA) before it is submitted to the assembler. The DML precompiler performs the following functions:

- Converts *most* DML statements into standard Assembler source statements.
- Ensures that all statements issued by the program are consistent with the logical structure of the database, the subschema view of the program, and the access restrictions defined in the subschema.
- Copies information maintained in the dictionary into program storage. Dictionary entities include database record descriptions, file definitions, map records, map definitions, logical records, and other predefined modules.
- Updates the dictionary with compile-time statistics used to monitor database activities for a given application program.
- Performs source level error checking.
- Generates an optional source statement listing of error conditions detected during DML processing.
- Supports the use of native VSAM files in conjunction with database access methods.
- Recognizes record, element, and file synonyms defined in the dictionary.
- Allows programs to be compiled for execution under various TP monitors without changing the source DML statements.

An Assembler program *must* be submitted to the DML precompiler if the program contains any of the following statements:

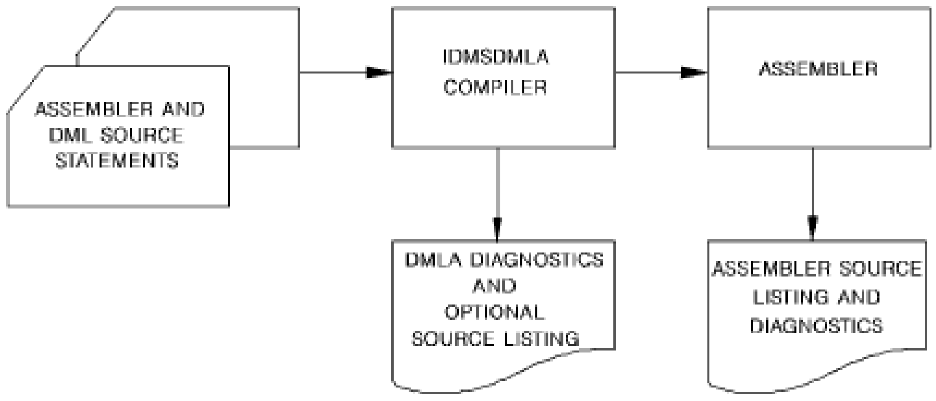
- An @COPY IDMS statement
- An @INVOKE statement
- Logical-record DML statement containing a WHERE clause

If none of these statements is included, the Assembler program can bypass the DML precompiler. The source can be submitted directly to the assembler because most Assembler DML statements are macro instructions that are expanded during assembly. It is recommended, however, that all programs accessing the database or running under a DC/UCF system use the DML precompiler. For a list of Assembler DML macros, see the Assembler DML Macros and Error Messages.

Output from the DML precompiler is a card-image source file that serves as input to the assembler. Output from the assembler consists of an object program and a source listing that includes any generated diagnostics. During assembly, *most* procedural DML verbs are expanded into executable Assembler source code, whether or not the DML precompiler was executed.

After the program is assembled, it is submitted to the linkage editor. The linkage editor link edits the object program into a specified load library. Output from the linkage editor consists of a load module and a link map.

The following figure illustrates the steps involved in assembling and executing an Assembler program containing DML statements.


ASSEMBLER AND DML SOURCE STATEMENTS
IDMSDMLA COMPILER
ASSEMBLER
DMLA DIAGNOSTICS AND OPTIONAL SOURCE LISTING
ASSEMBLER SOURCE LISTING AND DIAGNOSTICS

## Callable Services and Common Facilities

CA IDMS provides callable services and common facilities to use with your application programs.

### Callable Services

The callable services include:

- The IDMSCALC utility that lets you sort input into target page sequence.
- The IDMSIN01 facility that lets you perform miscellaneous CA IDMS functions.
- The TCP/IP socket program interface that lets you communicate with another TCP/IP application.

**Note:** For more information about using these callable services, see the *Callable Services Guide*.

### Common Facilities

The common facilities include:

- The Command Facility that lets you submit command statements in a batch or online environment.
- The Online Compiler Text Editor that lets you edit compiler output and resubmit it as input using the CA IDMS development tools.
- The Transfer Control Facility that lets you transfer between CA IDMS development tools.
- The SYSIDMS parameter file that contains parameters that you can add to a batch job running in local mode or under the central version. These parameters let you specify environment requirements, runtime directives, and operating system-dependent information.

**Note:** For more information about using these common facilities and the SYSIDMS parameter file, see the *Common Facilities Guide*.

# Chapter 3: DML Precompiler Options

This chapter contains syntax for the DML precompiler options. DML precompiler option statements are included in the input source code to the DML precompiler. These statements are used to:

- Override the default shared update usage mode for the DDLDML area of the dictionary and ready the area in either retrieval or protected update mode
- Print comment lines stored in the dictionary for subschema data items on the DML listing
- Generate a source statement listing of the output from the DML precompiler
- Suppress the logging of program activity statistics in the dictionary

These options are discussed separately below.

This section contains the following topics:

## Dictionary Usage Mode

When the main area (DDLDML area) of the dictionary accessed by the DML precompiler is readied, several options are available. The default usage mode, shared update usage, is defined at system generation. Shared update mode readies the DDLDML area for both retrieval and update and allows other concurrently executing run units to ready the DDLDML area in shared update or shared retrieval usage mode. You can override the default usage mode by specifying either retrieval or protected update usage mode in your application program.

**Syntax**

```
►─┬─ *RETRIEVAL ────────┬──────────────────────────────────────────────►
  └─ *PROTECTED-UPDATE ─┘
```

```
The asterisk (*) must be in column 1.
```

**Parameters**

***RETRIEVAL**

Readies the DDLDML area for retrieval only and allows other concurrently executing run units to open the DDLDML area in shared retrieval, shared update, protected retrieval, or protected update mode.

**Note:** If the DDLDML area is readied for retrieval only, no program activity statistics can be logged.

***PROTECTED-UPDATE**

Readies the DDLDML area for both retrieval and update and allows other concurrently executing run units to open the DDLDML area in retrieval usage mode only. The protected update usage mode prevents concurrent update of the area by run units executing under the same central version.

If included, the dictionary usage mode statement must precede all source statements.

## Comment Generation

The *SCHEMA-COMMENTS option causes schema-defined data item comments and IDD-defined record-element comments in the dictionary to be printed on the DML source listing. You can specify this option by including the following entry at the beginning of the input source code, after the dictionary usage mode statements (if present) and before any DML or Assembler statements.

**Syntax**

```
►── *SCHEMA-COMMENTS ──────────────────────────────────────────────►
```

```
The asterisk (*) must be in column 1.
```

If the input does not include a *SCHEMA-COMMENTS entry, comment lines are not generated.

## List Generation

You can turn on or off the source statement listing output by the DML precompiler by inserting a list generation option in the source program.

**Syntax**

```
►─┬─────────────┬──────────────────────────────────────────────────►
  ├─ *NODMLIST ◄ ┤
  └─ *DMLIST ────┘
```

```
The asterisk (*) must be in column 1.
```

**Parameters**

***NODMLIST**

Specifies that no source code listing is to be generated for the DML statements that follow.

***DMLIST**

Generates the source code listing for all the DML statements that follow.

In general, you would include one of these entries at the beginning of the input source code before any standard DML or Assembler statements. However, generation of the list can be turned on or off any number of times within one source program by inserting appropriate *DMLIST/*NODMLIST entries in the code.

Note: The DML precompiler always produces a listing of error messages. The *DMLIST option controls listing of the DML source code.

## Log Suppression

You can suppress the logging of program activity statistics in the dictionary by using the *NO-ACTIVITY-LOG option. This option, if included, is placed at the beginning of the DML source program. The DML precompiler generates and logs the following program activity statistics unless the *NO-ACTIVITY-LOG option is included in the program source code:

- Program name
- Language
- Date last compiled
- Number of lines
- Number of compilations
- Date created
- Subschema name (if any)
- File statistics
- Database access statistics (for example, records and modules copied from the dictionary; subprograms called; and records, sets, and areas accessed by DML verbs)

**Syntax**

```
►── *NO-ACTIVITY LOG ──────────────────────────────────────────────►
```

The asterisk (*) must be in column 1.

**Note:** Program activity statistics cannot be logged if you ready the dictionary DDLDML area for retrieval only.

# Chapter 4: Communications Blocks and Error Detection

This chapter describes the communication blocks and registers available under CA IDMS/DB and DC/UCF systems to return status information to an application program that requests database and data communication services.

CA IDMS/DB and DC/UCF systems use the following facilities to communicate with your application program:

- The **IDMS communications block** returns information from the database management system (DBMS) to your application program.

  The **ERRSTAT** field of the IDMS communications block receives a status code that indicates the successful or unsuccessful execution of a DML command. You can test for the content of the ERRSTAT field in your database program.

- The **logical-record request control (LRC) block** returns information from the Logical Record Facility (LRF) to your application program when you are accessing logical records that have been created by LRF.

  The **LRSTAT** field of the LRC block returns the path status for a logical-record DML request. You can test for the contents of the LRSTAT field in your program.

- **Register 15** is used by the DC/UCF system to return information regarding the successful or unsuccessful execution of DML commands that request data communication services. You can test for the content of register 15 to determine the outcome of a DC/UCF DML statement.

In addition to the above topics, this chapter lists the status codes returned by the DBMS for database requests and the return codes issued by DC/UCF system for data communications requests.

This section contains the following topics:

# IDMS Communications Block

The IDMS communications block passes information between the DBMS and the application program. Whenever a run unit issues a call to the DBMS for a database operation, the DBMS returns information about the outcome of the requested service to the ERRSTAT field in the application program's IDMS communications block.

To receive status information from the DBMS, an application program must define the IDMS communications block in variable storage. You must either copy the IDMS communications block from the dictionary into your program's variable storage by using the @COPY IDMS statement or generate the IDMS communications block by using the @SSCTRL statement. The following example illustrates the @COPY IDMS statement before and after it has been expanded by the DML precompiler:

```
          @COPY IDMS,SUBSCHEMA-CTRL      (Before DML expansion)
          @COPY IDMS,SUBSCHEMA-CTRL      (After DML expansion)
          DS    0D
SSCTRL    DS    0CL216
PGMNAME   DC    CL8' '
ERRSTAT   DC    CL4'1400'
DBKEY     DS    FL4
RECNAME   DC    CL16' '
AREANAME  DC    CL16' '
ERRORSET  DC    CL16' '
ERRORREC  DC    CL16' '
ERRAREA   DC    CL16' '
SSCIDBCM  DS    0CL100
IDBMSCOM  DS    100CL1
   ORG   SSCIDBCM
RDBMSCOM  DS    0CL100
PGINFO    DS    0CL4
PGINFGRP  DS    HL2
PGINFDBK  DS    HL2
          DS    CL96
DIRDBKEY  DC    FL4'0'
DBSTATUS  DS    0CL8
DBSTMTCD  DS    CL2
DBSTATCD  DS    CL5
          DS    CL1
RECOCCUR  DC    FL4'0'
DMLSEQ    DC    FL4'0'
```

The same expansion would result by using the @SSCTRL statement in your application program instead of the @COPY IDMS,SUBSCHEMA-CTRL statement. The @SSCTRL statement is a macro that generates the variable storage definitions of the IDMS communications block instead of copying the block from the dictionary.

**Note:** For more information about the differences between these statements, see the DML Precompiler Options (see page 29).

After every call to the DBMS, the DBMS issues an error-status code that indicates successful or unsuccessful completion of the requested service. This status code is returned to the ERRSTAT field in the IDMS communications block. You should examine the ERRSTAT field after every call to the DBMS. Depending on the error-status code, it may be useful to examine other fields and/or branch to a routine that responds to the condition indicated by the error-status code.

The following figure shows the layout of the 16-byte IDMS communications block; each field is described separately. Starting with offset 200, the layout of the block differs for application programs that run under CICS.

**Note:** For more information about the 18-byte IDMS communications block, see the 18-Byte Communications Blocks.

16-CHARACTER IDMS COMMUNICATIONS BLOCK

| | | Field | Data Type | Length (bytes) | Initial Value |
|---|---|---|---|---|---|
| * | 0 7 | PROGRAM-NAME | Alphanumeric | 8 | Program Name |
| | 8 11 | ERROR-STATUS | Alphanumeric | 4 | '1400' |
| | 12 15 | DBKEY | Binary | 4(Fullword) | 0000 |
| | 16 31 | RECORD-NAME | Alphanumeric | 16 | Spaces |
| | 32 47 | AREA-NAME | Alphanumeric | 16 | Spaces |
| | 48 63 | ERROR-SET | Alphanumeric | 16 | Spaces |
| | 64 79 | ERROR-RECORD | Alphanumeric | 16 | Spaces |
| | 80 95 | ERROR-AREA | Alphanumeric | 16 | Spaces |
| ** | 96 99 | PAGE-INFO | Binary | 4(Fullword) | 0000 |
| | 96...195 | IDBMSCOM-AREA | Alphanumeric | 100 | Low Values |
| | 196 199 | DIRECT-DBKEY | Binary | 4(Fullword) | 0000 |
| | 200 206 | DATABASE-STATUS | Alphanumeric | 7 | Spaces |
| | 207 | FILLER | ... | 1 | ... |
| | 208 211 | RECORD-OCCUR | Binary | 4(Fullword) | 0000 |
| | 212 215 | DML-SEQUENCE | Binary | 4(Fullword) | 0000 |

* word aligned
** PAGE-INFO-GROUP overlays bytes 97 and 98 and PAGE-INFO-DBK-FORMAT overlays bytes 99 and 100. Both of these fields are binary datatype, each with a length of two bytes. Suggested initial values for both are 00. Together these two fields represent PAGE-INFO.

## Field Descriptions

### Program Status Fields

The IDMS communications block contains the following fields that describe program status information:

- **PGMNAME** (offsets 0-7) is an 8-byte alphanumeric field that contains the name of the program being executed. This field is initialized automatically at the beginning of program execution if the program contains an @COPY IDMS SUBSCHEMA-BINDS statement. Otherwise, it must be initialized by the programmer.
- **ERRSTAT** (offsets 8-11) is a 4-byte alphanumeric field that contains a value indicating the outcome of the DML statement that calls the DBMS. The ERRSTAT field must be initialized to 1400 by the program. The DBMS updates this field immediately before returning control to the user program after performing (attempting) a requested database service.

The ERRSTAT field and its use are described under Testing for DML Error-Status Codes (see page 52) later in this chapter.

Note: A program that consists of two or more run units must reinitialize the ERRSTAT field to 1400 after finishing one run unit and before binding the next.

- **DBKEY** (offsets 12-15) is a 4-byte (fullword) binary field that contains the database key (db-key) of the last record accessed by the run unit. For example, after successful execution of an @FIND command, DBKEY is updated with the db-key of the located record. DBKEY is not changed if the call to the DBMS results in an error condition.
- **RECNAME** (offsets 16-31) is a 16-byte alphanumeric field that contains the name of the last record accessed successfully by the run unit. This field is left justified and padded with spaces on the right.
- **AREANAME** (offsets 32-47) is a 16-byte alphanumeric field that contains the name of the last area accessed successfully by the run unit. This field is left justified and padded with spaces on the right.
- **ERRORSET** (offsets 48-63) is a 16-byte alphanumeric field that contains the name of the set involved in the last operation to produce an error condition. This field is left justified and padded with spaces on the right.
- **ERRORREC** (offsets 64-79) is a 16-byte alphanumeric field that contains the name of the record involved in the last operation to produce an error condition. This field is left justified and padded with spaces on the right.
- **ERRAREA** (offsets 80-95) is a 16-byte alphanumeric field that contains the name of the area involved in the last operation to produce an error condition. This field is left justified and padded with spaces on the right.
- **IDBMSCOM** (offsets 96-195) is a 100-byte alphanumeric array that is used internally by CA IDMS/DB for specification of runtime function information.
- **PGINFO** (offsets 96-99) is a 4-byte binary field that represents the page information associated with the last record accessed by the rununit. For example, after successful execution of an @FIND command, PGINFO is updated with the page information of the located record.

  Page information is not changed if the call to the DBMS results in a nonzero status condition.

  Page information is a 4-byte field consisting of the following sub-fields:

  - Bytes 1-2: Page group number (PGINFGRP)
  - Bytes 3-4: Dbkey radix (PGINFDBK)

  The PGINFO field overlays part of the IDBMSCOM area in the subschema control.

The dbkey radix portion of the page information can be used in interpreting a dbkey for display purposes and in formatting a dbkey from page and line numbers. The dbkey radix represents the number of bits within a dbkey value that are reserved for the line number of a record. By default, this value is 8, meaning that up to 255 records can be stored on a single page of the area. Given a dbkey, you can separate its associated page number by dividing the dbkey by 2 raised to the power of the dbkey radix. For example, if the dbkey radix is 4, you would divide the dbkey value by 2**4. The resulting value is the page number of the dbkey. To separate the line number, you would multiply the page number by 2 raised to the power of the dbkey radix and subtract this value from the dbkey value. The result would be the line number of the dbkey. The following two formulas can be used to calculate the page and line numbers from a dbkey value:

```
Page-number = dbkey value / (2 ** dbkey radix)
Line-number = dbkey value - (page-number * ( 2 ** dbkey radix))
```

- **DIRDBKEY** (offsets 196-199) is a 4-byte (fullword binary) field that contains a user-specified db-key value or a null db-key value of -1. This field is used for storing a record with a location mode of direct. DIRDBKEY must be initialized by the user; it is not updated by the DBMS.

  Note: (native VSAM users) The DIRDBKEY field can be used only when storing a record in a native VSAM relative record data set (RRDS). This field must be initialized by the user to the relative record number of the record being stored.

- **Reserved for system** (offsets 200-206) is a 7-byte alphanumeric field reserved for CA IDMS/DB use.
- **FILLER** (offset 207) is a 1-byte field used to ensure fullword alignment.
- **RECOCCUR** (offsets 208-211) is a 4-byte (fullword) binary field that contains a record-occurrence sequence identifier used internally by the DBMS.
- **DMLSEQ** (offsets 212-215) is a 4-byte (fullword) binary field that contains the source-level sequence number generated by the DML macros, if DEBUG is specified. It is not used by the runtime system, with the exception of SYSIDMS DMLTRACE=ON tracing.

**Updating the Fields**

After a call to the DBMS, one or more of the fields described above may be updated, depending on the DML statement issued and whether or not the statement was executed successfully.

**Example of Updating Fields**

The following figure illustrates the updating process; only those fields accessed by the runtime system are shown. Fields used internally by the DBMS are not shown. Blank fields are not updated by DML statements.

Key for this figure:

| | |
|---|---|
| * | If true, field is set to zone decimal zeroes (0000); if false, field is set to 1601 |
| 0 | Field is set to zone decimal zeroes |
| Y | Field is updated |
| C | Field is cleared to spaces |
| N | Field is set to null db-key value (-1) |
| *nn* | Specific minor error code |

| | SUCCESSFUL | | | | | | | | | | UNSUCCESSFUL | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PGMNAME | ERRSTAT | DBKEY | RECNAME | AREANAME | ERRORSET | ERRORREC | ERRAREA | PGINFO | DIRDBKEY | PGMNAME | ERRSTAT | DBKEY | RECNAME | AREANAME | ERRORSET | ERRORREC | ERRAREA | PGINFO | DIRDBKEY |
| Control statements | | | | | | | | | | | | | | | | | | | | |
| BIND SUBSCH | | 0 | | | | | | | | | | 14nn | | | | | | | | |
| BINDREC | | 0 | | | | | | | | | | 14nn | | | | Y | Y | Y | | |
| BIND PROC | | 0 | | | | | | | | | | 14nn | | | | Y | Y | Y | | |
| READY | | 0 | | | | | | | | | | 09nn | | | | C | C | C | | |
| FINISH | | 0 | N | C | | C | C | C | | | | 01nn | | | | C | C | C | | |
| COMMIT (ALL) | | 0 | N | C | | C | C | C | | | | 18nn | | | | C | C | C | | |
| ROLLBAK (CONTINUE) | | 0 | N | C | | C | C | C | | | | 19nn | | | | C | C | C | | |
| KEEP (EXCLUSIVE) | | 0 | Y | Y | Y | C | C | C | Y | | | 06nn | | | | Y | Y | Y | | |
| IF set-name EMPTY | | * | Y | Y | Y | C | C | C | Y | | | 16nn | | | | Y | Y | Y | | |
| IF set-name MEMBER | | * | Y | Y | Y | C | C | C | Y | | | 16nn | | | | Y | Y | Y | | |
| Retrieval statements | | | | | | | | | | | | | | | | | | | | |
| FIND / OBTAIN | | 0 | Y | Y | Y | C | C | C | Y | | | 03nn | | | | Y | Y | Y | | |
| GET | | 0 | Y | Y | Y | C | C | C | Y | | | 05nn | | | | Y | Y | Y | | |
| RETURN | | 0 | Y | Y | Y | C | C | C | Y | | | 17nn | | | | Y | Y | Y | | |
| Modification statements | | | | | | | | | | | | | | | | | | | | |
| STORE record-name | | 0 | Y | Y | Y | C | C | C | Y | | | 12nn | | | | Y | Y | Y | | |
| CONNECT record-name | | 0 | Y | Y | Y | C | C | C | Y | | | 07nn | | | | Y | Y | Y | | |
| MODIFY record-name | | 0 | Y | Y | Y | C | C | C | Y | | | 08nn | | | | Y | Y | Y | | |
| DISCON record-name | | 0 | Y | Y | Y | C | C | C | Y | | | 11nn | | | | Y | Y | Y | | |
| ERASE record-name | | 0 | N | Y | Y | C | C | C | | | | 02nn | | | | Y | Y | Y | | |
| Accept statements | | | | | | | | | | | | | | | | | | | | |
| ACCEPT DBKEY FROM CURRENCY | | 0 | | | | C | C | C | | | | 15nn | | | | Y | Y | Y | | |
| ACCEPT DBKEY REL TO CURRENCY | | 0 | | | | C | C | C | | | | 15nn | | | | Y | Y | Y | | |
| ACCEPT STATS | | 0 | | | | C | C | C | | | | 15nn | | | | Y | Y | Y | | |

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ACCEPT BIND | | 0 | | | | C | C | C | | | | 15nn | | | | Y | Y | Y | | |
| ACCEPT PROC | | 0 | | | | C | C | C | | | | 15nn | | | | Y | Y | Y | | |
| ACCEPT PGINFO | | 0 | | | | C | C | C | | | | 15nn | | | | Y | Y | Y | | |

## ERRSTAT Field and Codes

You can use the ERRSTAT field of the IDMS communications block to determine if a DML request was processed successfully. The DBMS system returns a value to the ERRSTAT field indicating the result of each DML request. For more information about the ERRSTAT field, see Testing for DML Error-Status Codes (see page 52).

**LRF users**: You should check the LR-STATUS field of the LRC block before checking the ERRSTAT field.

**Major and Minor Codes**

The ERRSTAT field is a four-byte zoned decimal field. The first two bytes represent a major code; the second two bytes represent a minor code. Major codes identify the function performed; minor codes describe the status of that function.

**Value of Codes**

A value of 0000 indicates successful completion of the requested function. A value other than 0000 indicates completion of the function in a manner that may or may not be in error, depending on your expectations. For example, 0326 (DB-REC-NOT-FOUND) should be anticipated after FIND CALC retrieval; this allows you to trap the condition and continue processing.

DB status codes have a major code in the range 01 to 20. They occur during database access in batch or online processing. DC status codes have a major code in the range 30 to 51. They occur in online or DC-BATCH processing. Status codes with a major code of 00 apply to all DML functions. DB status codes and DC status codes are discussed separately below.

### DB Status Codes

The following tables list DB major and minor codes and their meanings.

## Major DB Status Codes

| Major Code | Database Function |
|---|---|
| 00 | Any DML statement |
| 01 | FINISH |
| 02 | ERASE |
| 03 | FIND/OBTAIN |
| 05 | GET |
| 06 | KEEP |
| 07 | CONNECT |
| 08 | MODIFY |
| 09 | READY |
| 11 | DISCONNECT |
| 12 | STORE |
| 14 | BIND |
| 15 | ACCEPT |
| 16 | IF |
| 17 | RETURN |
| 18 | COMMIT |
| 19 | ROLLBACK |
| 20 | LRF requests |

## Minor DB Status Codes

| Minor Code | Database Function Status |
|---|---|
| 00 | Combined with a major code of 00, this code indicates successful completion of the DML operation. Combined with a nonzero major code, this code indicates that the DML operation was not completed successfully due to central version causes, such as time-outs and program checks. |
| 01 | An area has not been readied. When this code is combined with a major code of 16, an IF operation has resulted in a valid false condition. |

| Minor Code | Database Function Status |
|---|---|
| 02 | Either the db-key used with a FIND/OBTAIN DB-KEY statement or the direct db-key suggested for a STORE is not within the page range for the specified record name. |
| 03 | Invalid currency for the named record, set, or area. This can only occur when a run unit is sharing a transaction with other database sessions. The 03 minor status is returned if the run unit tries to retrieve or update a record using a currency that has been invalidated because of changes made by another database session that is sharing the same transaction. |
| 04 | The occurrence count of a variably occurring element has been specified as either less than zero or greater than the maximum number of occurrences defined in the control element. |
| 05 | The specified DML function would have violated a duplicates-not-allowed option for a CALC, sorted, or index set. |
| 06 | No currency has been established for the named record, set, or area. |
| 07 | The end of a set, area, or index has been reached or the set is empty. |
| 08 | The specified record, set, procedure, or LR verb is not in the subschema or the specified record is not a member of the set. |
| 09 | The area has been readied with an incorrect usage mode. |
| 10 | An existing access restriction or subschema usage prohibits execution of the specified DML function. For LRF users, the subschema in use allows access to database records only. Combined with a major code of 00, this code means the program has attempted to access a database record, but the subschema in use allows access to logical records only. |
| 11 | The record cannot be stored in the specified area due to insufficient space. |
| 12 | There is no db-key for the record to be stored. This is a system internal error and should be reported. |
| 13 | A current record of run unit either has not been established or has been nullified by a previous ERASE statement. |
| 14 | The CONNECT statement cannot be executed because the requested record has been defined as a mandatory automatic member of the set. |
| 15 | The DISCONNECT statement cannot be executed because the requested record has been defined as a mandatory member of the set. |
| 16 | The record cannot be connected to a set of which it is already a member. |
| 17 | The transaction manager encountered an error. |
| 18 | The record has not been bound. |
| 19 | The run unit's transaction was forced to back out. |

| Minor Code | Database Function Status |
|---|---|
| 20 | The current record is not the same type as the specified record name. |
| 21 | Not all areas being used have been readied in the correct usage mode. |
| 22 | The record name specified is not currently a member of the set name specified. |
| 23 | The area name specified is either not in the subschema or not an extent area; or the record name specified has not been defined within the area name specified. |
| 25 | No currency has been established for the named set. |
| 26 | No duplicates exist for the named record or the record occurrences cannot be found. |
| 28 | The run unit has attempted to ready an area that has been readied previously. |
| 29 | The run unit has attempted to place a lock on a record that is locked already by another run unit. A deadlock results. Unless the run unit issued either a FIND/OBTAIN KEEP EXCLUSIVE or a KEEP EXCLUSIVE, the run unit is aborted. |
| 30 | An attempt has been made to erase the owner record of a nonempty set. |
| 31 | The retrieval statement format conflicts with the record's location mode. |
| 32 | An attempt to retrieve a CALC/DUPLICATE record was unsuccessful; the value of the CALC field in variable storage is not equal to the value of the CALC control element in the current record of run unit. |
| 33 | At least one set in which the record participates has not been included in the subschema. |
| 40 | The WHERE clause in an OBTAIN NEXT logical-record request is inconsistent with a previous OBTAIN FIRST or OBTAIN NEXT command for the same record. Previously specified criteria, such as reference to a key field, have been changed. A path status of LR-ERROR is returned to the LRC block. |
| 41 | The subschema contains no path that matches the WHERE clause in a logical-record request. A path status of LR-ERROR is returned to the LRC block. |
| 42 | An ON clause included in the path by the DBA specified return of the LR-ERROR path status to the LRC block; an error has occurred while processing the LRF request. |

| Minor Code | Database Function Status |
|---|---|
| 43 | A program check has been recognized during evaluation of a WHERE clause; the program check indicates that either a WHERE clause has specified comparison of a packed decimal field to an unpacked nonnumeric data field, or data in variable storage or a database record does not conform to its description. A path status of LR-ERROR is returned to the LRC block unless the DBA has included an ON clause to override this action in the path. |
| 44 | The WHERE clause in a logical-record request does not supply a key element (sort key, CALC key, or db-key) expected by the path. A path status of LR-ERROR is returned to the LRC block. |
| 45 | During evaluation of a WHERE clause, a program check has been recognized because a subscript value is neither greater than 0 nor less than its maximum allowed value plus 1. A path status of LR-ERROR is returned to the LRC block unless the DBA has included an ON clause to override this action in the path. |
| 46 | A program check has revealed an arithmetic exception (for example: overflow, underflow, significance, divide) during evaluation of a WHERE clause. A path status of LR-ERROR is returned to the LRC block unless the DBA has included an ON clause to override this action in the path. |
| 53 | The subschema definition of an indexed set does not match the indexed set's physical structure in the database. |
| 54 | Either the prefix length of an SR51 record is less than zero or the data length is less than or equal to zero. |
| 55 | An invalid length has been defined for a variable-length record. |
| 56 | An insufficient amount of memory to accommodate the CA IDMS compression/decompression routines is available. |
| 57 | A retrieval-only run unit has detected an inconsistency in an index that should cause an 1143 abend, but optional APAR bit 216 has been turned on. |
| 58 | An attempt was made to rollback updates in a local mode program. Updates made to an area during a local mode program's execution cannot be automatically rolled out. The area must be manually recovered. |
| 60 | A record occurrence type is inconsistent with the set named in the ERROR-SET field in the IDMS communications block. This code usually indicates a broken chain. |
| 61 | No record can be found for an internal db-key. This code usually indicates a broken chain. |
| 62 | A system-generated db-key points to a record occurrence, but no record with that db-key can be found. This code usually indicates a broken chain. |

| Minor Code | Database Function Status |
|---|---|
| 63 | The DBMS cannot interpret the DML function to be performed. When combined with a major code of 00, this code means invalid function parameters have been passed on the call to the DBMS. For LRF users, a WHERE clause includes a keyword that is longer than the 32 characters allowed. |
| 64 | The record cannot be found; the CALC control element has not been defined properly in the subschema. |
| 65 | The database page read was not the page requested. |
| 66 | The area specified is not available in the requested usage mode. |
| 67 | The subschema invoked does not match the subschema object tables. |
| 68 | The CICS interface was not started. |
| 69 | A BIND RUN-UNIT may not have been issued; the CV may be inactive or not accepting new run units; or the connection with the CV may have been broken due to time out or other factors. When combined with a major code of 00, this code means the program has been disconnected from the DBMS. |
| 70 | The database will not ready properly; a JCL error is the probable cause. |
| 71 | The page range or page group for the area being readied or the page requested cannot be found in the DMCL. |
| 72 | There is insufficient memory to dynamically load a subschema or database procedure. |
| 73 | A central version run unit will exceed the MAXERUS value specified at system generation. |
| 74 | The dynamic load of a module has failed. If operating under the central version, a subschema or database procedure module either was not found in the data dictionary or the load (core image) library or, if loaded, will exceed the number of subschema and database procedures provided for at system generation. |
| 75 | A read error has occurred. |
| 76 | A write error has occurred. |
| 77 | The run unit has not been bound or has been bound twice. When combined with a major code of 00, this code means either the program is no longer signed on to the subschema or the variable subschema tabl es have been overwritten. |
| 78 | An area wait deadlock has occurred. |
| 79 | The run unit has requested more db-key locks than are available to the system. |

| Minor Code | Database Function Status |
|---|---|
| 80 | The target node is either not active or has been disabled. |
| 81 | The converted subschema requires specified database name to be in the DBNAME table. |
| 82 | The subschema must be named in the DBNAME table. |
| 83 | An error has occurred in accessing native VSAM data sets. |
| 87 | The owner and member records for a set to be updated are not in the same page group or do not have the same db-key radix. |
| 91 | The subschema requires a DBNAME to do the bind run unit. |
| 92 | No subschema areas map to DMCL. |
| 93 | A subschema area symbolic was not found in DMCL. |
| 94 | The specified dbname is neither a dbname defined in the DBNAME table, nor a SEGMENT defined in the DMCL. |
| 95 | The specified subschema failed DBTABLE mapping using the specified dbname. |

**Note:** For a complete description of DB runtime status codes, see the chapter "CA IDMS Status Codes" in the *Messages and Codes Guide*.

## DC Status Codes

The following tables list the DC major and minor codes and their meanings.

### Major DC Status Codes

| Major Code | Function |
|---|---|
| 00 | Any DML statement |
| 30 | TRANSFER CONTROL |
| 31 | WAIT/POST |
| 32 | GET STORAGE/FREE STORAGE |
| 33 | SET ABEND EXIT/ABEND CODE |
| 34 | LOAD/DELETE TABLE |
| 35 | GET TIME/SET TIMER |

| Major Code | Function |
|---|---|
| 36 | WRITE LOG |
| 37 | ATTACH/CHANGE PRIORITY |
| 38 | BIND/ACCEPT/END TRANSACTION STATISTICS |
| 39 | ENQUEUE/DEQUEUE |
| 40 | SNAP |
| 43 | PUT/GET/DELETE SCRATCH |
| 44 | PUT/GET/DELETE QUEUE |
| 45 | BASIC MODE TERMINAL MANAGEMENT |
| 46 | MAPPING MODE TERMINAL MANAGEMENT |
| 47 | LINE MODE TERMINAL MANAGEMENT |
| 48 | ACCEPT/WRITE PRINTER |
| 49 | SEND MESSAGE |
| 50 | COMMIT TASK/ROLLBACK TASK/FINISH TASK/WRITE JOURNAL |
| 51 | KEEP LONGTERM |
| 58 | SVC SEND/RECEIVE |

## Minor DC Status Codes

| Minor Code | Function Status |
|---|---|
| 00 | Combined with a major code of 00, this code indicates either successful completion of the DML function or that all tested resources have been enqueued. |
| 01 | The requested operation cannot be performed immediately; waiting will cause a deadlock. |
| 02 | Either there is insufficient storage in the storage pool or the storage required for control blocks is unavailable. |
| 03 | The scratch area ID cannot be found. |
| 04 | Either the queue ID (header) cannot be found or a paging session was in progress when a second STARTPAGE command was received (that is, an implied ENDPAGE was processed before this STARTPAGE was executed successfully). |
| 05 | The specified scratch record ID or queue record cannot be found. |

| Minor Code | Function Status |
|---|---|
| 06 | No resource control element (RCE) exists for the queue record; currency has not been established. |
| 07 | Either an I/O error has occurred or the queue upper limit has been reached. |
| 08 | The requested resource is not available. |
| 09 | The requested resource is available. |
| 10 | New storage has been assigned. |
| 11 | A maximum task condition exists. |
| 12 | The named task code is invalid. |
| 13 | The named resource cannot be found. |
| 14 | The requested module is defined as nonconcurrent and is currently in use. |
| 15 | The named module has been overlaid and cannot be reloaded immediately. |
| 16 | The specified interval control element (ICE) address cannot be found. |
| 17 | The record has been replaced. |
| 18 | No printer terminals have been defined for the current DC system. |
| 19 | The return area is too small; data has been truncated. |
| 20 | An I/O, program-not-found, or potential-deadlock status condition exists. |
| 21 | The message destination is undefined, the long term ID cannot be found, or a KEEP LONGTERM request was issued by a nonterminal task. |
| 22 | A record already exists for the scratch area specified. |
| 23 | No storage or resource control element (RCE) could be allocated for the reply area. |
| 24 | The maximum number of outstanding replies has been exceeded. |
| 25 | An attention interrupt has been received. |
| 26 | There is a logical error in the output data stream. |
| 27 | A permanent I/O error has occurred. |
| 28 | The terminal dial-up line is disconnected. |
| 29 | An invalid parameter has been passed in the list set up by the DML processor. |
| 30 | The named function has not yet been implemented. |

| Minor Code | Function Status |
|---|---|
| 31 | An invalid parameter has been passed; the TRB, LRB, or MRB contains an invalid field; or the request is invalid because of a possible logic error in the application program. In a DC-BATCH environment, a possible cause is that the record length specified by the command exceeds the maximum length based on the packet size. |
| 32 | The derived length of the specified variable storage is negative or zero. |
| 33 | Either the named table or the named map cannot be found in the data dictionary load area. |
| 34 | The named variable-storage area must be an 01-level entry in the LINKAGE SECTION. |
| 35 | A GET STORAGE request is invalid because the LINKAGE SECTION variable has already been allocated. |
| 36 | The program either was not defined during system generation or is marked out-of-service. |
| 37 | A GET STORAGE operand is invalid because the specified variable storage area is in the WORKING-STORAGE SECTION instead of the LINKAGE SECTION. |
| 38 | Either no GET STORAGE operand was specified or the specified LINKAGE SECTION variable has not been allocated. |
| 39 | The terminal device being used is out of service. |
| 40 | NOIO has been specified but the datastream cannot be found. |
| 41 | An IF operation resulted in a valid true condition. |
| 42 | The named map does not support the terminal device in use. |
| 43 | A line I/O session has been cancelled by the terminal operator. |
| 44 | The referenced field does not participate in the specified map; a possible cause is an invalid subscript. |
| 45 | An invalid terminal type is associated with the issuing task. |
| 46 | A terminal I/O error has occurred. |
| 47 | The named area has not been readied. |
| 48 | The run unit has not been bound. |
| 49 | NOWAIT has been specified but WAIT is required. |
| 50 | Statistics are not being kept. |
| 51 | A lock manager error occurred during the processing of a KEEP LONGTERM request |
| 52 | The specified table is missing or invalid. |

| Minor Code | Function Status |
|---|---|
| 53 | An error occurred from a user-written edit routine. |
| 54 | Either there is invalid internal data or a data conversion error has occurred. |
| 55 | The user-written edit routine cannot be found. |
| 56 | No DFLDS have been defined for the map. |
| 57 | The ID cannot be found, is not a long-term permanent ID, or is being used by another run unit. |
| 58 | Either the LRID cannot be found, the maximum number of concurrent task threads was exceeded, or an attempt was made to rollback database changes in local mode. |
| 59 | An error occurred in transferring the KEEP LONGTERM request to IDMSKEEP |
| 60 | The requested KEEP LONGTERM lock id was already in use with a different page group |
| 63 | Invalid function parameters have been passed on the call to the DBMS. |
| 64 | No detail exists currently for update; no action has been taken. Alternatively, the requested node for a header or detail is either not present or not updated. |
| 68 | There are no more updated details to MAP IN or the amount of storage defined for pageable maps at sysgen is insufficient. In the latter case, subsequent MAP OUT DETAIL statements are ignored. |
| 72 | No detail occurrence, footer, or header fields exist to be mapped out by a MAP OUT RESUME command, or the scratch record that contains the requested detail could not be accessed. The latter case is a mapping internal error and should be reported. |
| 76 | The first screen page has been transmitted to the terminal. |
| 77 | Either the program is no longer signed on to the subschema or the variable subschema tables have been overwritten. |
| 80 | The target node is either not active or has been disabled. |
| 97 | An error was encountered processing a syncpoint request; check the log for details. |
| 98 | An unsupported COBOL compiler option (for example, DEBUG) has been specified for an online program or a program running in a batch region has issued a DML verb that is only valid when running online under CA IDMS/DC/UCF. |
| 99 | An unexpected internal return code has been received; the terminal device is out of service. |

**Note:** For a complete description of DC runtime status codes, see the chapter "CA IDMS Status Codes" in the *Messages and Codes Guide*.

## Testing for DML Error-Status Codes

Testing for the value of the ERRSTAT field in an Assembler program is a simple procedure. CA IDMS/DB places a value in the ERRSTAT field after each DML statement requesting database services is executed. This value can be compared to known error-status codes to determine whether execution was successful. For example, you can check for successful completion by comparing the ERRSTAT field to a working storage field defined as 0000. The program can then perform a conditional branch.

The following example demonstrates a test for the successful execution of the @OBTAIN statement. After completion of the @OBTAIN statement, the value returned to the ERRSTAT field is compared to the defined constant STATOK. If the @OBTAIN is successfully completed, processing continues. Otherwise, the program branches to routine OBERR2, which evaluates the ERRSTAT field and determines the next statement to be executed.

```
         @OBTAIN OWNER,SET='DEPT-EMPLOYEE'
         CLC   ERRSTAT,STATOK
         BNE   OBERR2
         MVC   DID,DEPTID
         .
         .
         .
STATOK   DC   CL4'0000'
```

In topic Data Manipulation Language Statements, the status codes that can be returned to the ERRSTAT field of the IDMS communications block are listed after the description of each database command. To determine test conditions based on error-status codes see Data Manipulation Language Statements (see page 73) .

## Logical-Record Request Control (LRC) Block

The logical-record request control (LRC) block passes information between the application program and LRF. It is used in conjunction with the IDMS communications block to pass information to LRF about a logical-record request and to return path status information about the processing of the request to the program.

To receive information about a logical-record request, the application program must define the LRC block in variable storage. You must either copy the LRC block from the dictionary into the program's variable storage by using the @COPY IDMS statement or generate the LRC block by using the @SSLRCTL statement. The following example illustrates the @COPY IDMS statement before and after expansion by the DML precompiler:

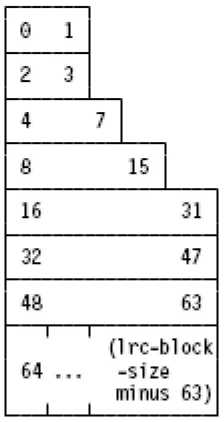
```
          @COPY IDMS,SUBSCHEMA-LR-CTRL   (before DML expansion)

*         @COPY IDMS,SUBSCHEMA-LR-CTRL   (after DML expansion)
          DS    0D
SSLRCTL   DS    0CL576
LRPXLN    DS    HL2
LRMVXP    DS    HL2
LRIDENT   DC    CL4'LRC '
LRVERB    DC    CL8' '
LRNAME    DC    CL16' '
LRSTAT    DC    CL16' '
LRFILL    DC    CL16' '
LRPXE     DS    512CL1
```

The same expansion would result by using the @SSLRCTL statement in your application program instead of the @COPY IDMS,SUBSCHEMA-LR-CTRL statement. The @SSLRCTL statement is a macro that generates the variable storage definitions of the LRC block instead of copying the block from the dictionary. For more information about the differences between these statements, see DML Precompiler-Directive Statements (see page 405) .

When a program issues a logical-record request, the LRC block stores the DML verb used by the program, the name of the logical-record, and the selection criteria of the request. LRF uses this information to select the appropriate path to handle the request.

After LRF has processed a request, it returns path status information in the LRC block. After issuing the path status, LRF returns an error-status code in the ERRSTAT field of the IDMS communications block. You can use this information to evaluate the result of the request and to determine further processing based on that result. The following figure shows the layout of the LRC block; each field is described separately following the figure.

LRC BLOCK



| Field | Description | Data Type | Length (bytes) |
|---|---|---|---|
| LRPXLN | Logical-record LRC length | BINARY | 2 |
| LRMVXP | Evaluation work-area-length | BINARY | 2 |
| LRIDENT | Constant 'LRC ' | ALPHANUMERIC | 4 |
| LRVERB | Logical-record verb | ALPHANUMERIC | 8 |
| LRNAME | Logical-record name | ALPHANUMERIC | 16 |
| LRSTAT | Logical-record error-status indicator | ALPHANUMERIC | 16 |
| LXFIL | Filler | ... | 16 |
| LRPXE | WHERE clause | Mixed | ... |

## Field Descriptions

The LRC block contains the following fields:

- **LRPXLN** (offsets 0-1) is a halfword field that describes the length of the LRC block for a logical record.
- **LRMVXP** (offsets 2-3) is a halfword field that describes the evaluation work area length used for processing the logical record.
- **LRIDENT** (offsets 4-7) is a 4-byte alphanumeric field used internally by LRF. (It contains the constant LRC followed by a space.)
- **LRVERB** (offsets 8-15) is an 8-byte alphanumeric field used to record the DML verb issued by the LRF program.
- **LRNAME** (offsets 16-31) is a 16-byte alphanumeric field that contains the name of the logical record being accessed.
- **LRSTAT** (offsets 32-47) is a 16-byte alphanumeric field that contains the path status of a logical-record request. The standard path statuses are LR-FOUND, LR-NOT-FOUND, and LR-ERROR. Path statuses can also be defined by the DBA. Testing for the value of the LRSTAT field is described below in "Testing for the logical-record path status."

- **LXFIL** (offsets 48-63) is a 16-byte filler.
- **LRPXE** (offset 64-end) is a variable length data area that contains information regarding the logical-record request's WHERE clause. This field is usually 512 bytes (default). You can code the SIZE option of the @BIND SUBSCH, @COPY IDMS,SUBSCHEMA-LR-CTRL, and @SSLCTRL statements to lengthen this field to accommodate a long, complex WHERE clause. (For more information about increasing the size of this field, see @COPY IDMS (see page 411).)

## Testing for the Logical-Record Path Status

Path statuses are issued during execution of the path selected to service a logical-record request. LRF returns a specific path status to the LRSTAT field of the program's LRC block to indicate the result of each logical-record request. You can examine this information to determine further processing.

**Path Statuses**

Path statuses are 1- to 16-byte character strings; they can either be standard or defined by the DBA in the subschema. The standard path statuses are:

- **LR-FOUND**—Indicates the logical-record request has been successfully executed. This status can be returned as the result of any LRF DML statement. When LR-FOUND is returned, the ERRSTAT field of the IDMS communications block contains 0000.
- **LR-NOT-FOUND**—Indicates the specified logical record cannot be found because either no such record exists or all such occurrences have already been retrieved. This status can be returned as the result of any LRF DML statement, provided that the path to which LRF is directed includes retrieval logic. When LR-NOT-FOUND is returned, the ERRSTAT field of the IDMS communications block contains 0000.
- **LR-ERROR**—Indicates that either a logical-record request is issued incorrectly or an error occurs in the processing of the path selected to service the request.

**Code Depends on Type of Error**

When LR-ERROR is returned, the type of status code returned to the program in the ERRSTAT field of the IDMS communications block differs according to the type of error. If the error occurs in the logical-record path, the ERRSTAT field contains a status code issued by CA IDMS/DB with a major code from 00 to 19.

When the error occurs in the request itself, LRF returns the path status LR-ERROR to the LRSTAT field of the LRC block and places one of the following codes in the ERRSTAT field of the IDMS communications block:

Note: Any of these error-status codes can result from any of the logical-record DML statements. The only exception is code 2040, which applies only to the @OBTAIN NEXT DML statement.

**2008**

Either the named logical record is not defined in the subschema or the specified DML verb is not permitted with the named logical record.

**2010**

The program has attempted to access a logical record, but the subschema in use allows access to database records only.

**2018**

A path command has attempted to access a database record that has not been bound.

**2040**

The WHERE clause included in an @OBTAIN NEXT statement has directed LRF to a different path than did the WHERE clause in the preceding @OBTAIN statement for the same logical record. Either an @OBTAIN FIRST should have been issued instead of @OBTAIN NEXT or the WHERE clause is incorrect.

**2041**

LRF was unable to match the request's WHERE clause to a path to service the request.

**2042**

An ON clause included in the path by the DBA specified return of the LR-ERROR path status to the program.

**2043**

During evaluation of a WHERE clause, a program check has been recognized for one of the following reasons:

- A WHERE clause has specified that a packed decimal field be compared to a field that is not packed and that cannot be converted to a packed field due to the presence of nonnumeric data.
- Data in either variable storage or a database record does not conform to its description.

A path status of LR-ERROR is returned to the program unless the DBA has included an ON clause in the path to override this action.

**2044**

The WHERE clause in a logical-record request does not include a field of information required by the path.

**2045**

During evaluation of a WHERE clause, a program check has been recognized because a subscript value is either less than zero or greater than its maximum allowed value plus 1. A path status of LR-ERROR is returned to the program unless the DBA has included an ON clause in the path to override this action.

**2046**

A program check has been recognized during evaluation of a WHERE clause for one of the following reasons:

- An arithmetic overflow would occur (fixed point, decimal, or exponent).
- An arithmetic underflow would occur (exponent).
- A divide exception would occur (fixed point, decimal, or floating point).
- A significance exception has occurred.

A path status of LR-ERROR is returned to the program unless the DBA has included an ON clause in the path to override this action.

**2063**

A logical-record request's WHERE clause includes a keyword that is longer than the 32 characters allowed.

**2064**

A path command has attempted to access a CALC data item that has not been described properly in the subschema.

**2072**

Storage is not available for the work areas required to evaluate the logical-record request's WHERE clause.

**Optional ONLRSTS Clause**

In addition to directly testing the value of the LRSTAT field, you can include an ON clause that tests for a specific standard or DBA-defined path status for each DML statement; for example:

```
@OBTAIN NEXT,REC='EMPJOBLR',ONLRSTS='LR-NOT-FOUND',GOTO=RECERROR
```

The ONLRSTS clause tests for the standard path status of LR-NOT-FOUND. If LR-NOT-FOUND is returned, the branch imperative GOTO=RECERROR will be executed and the program will branch to the label RECERROR.

**Syntax**

```
►─┬────────────────────────────────────────────┬──────────────────────►
  └─ ,ONLRSTS=path-status,GOTO=branch-location ─┘
```

**Parameters**

**ONLRSTS=*path-status***

Tests the LRSTAT field for a path status returned as the result of the logical-record request issued by the program. Path-status must be a quoted literal (1-16 bytes under z/OS and OS/390 or 1-6 bytes under VSE) or program variable.

**GOTO=*branch-location***

Specifies the program action to be taken if the specified path status is found in LRSTAT.

**Note:** For more information about LRF DML commands and clauses see Data Manipulation Language Statements (see page 73).

## DC/UCF General Registers

General registers 0, 1, and 15 pass information about data communication services from the DC/UCF system to the application program. The registers are used in the following manner:

- **Register 0** is used by several DC/UCF commands to return information regarding specific parameters of the DML statement.
- **Register 1** is sometimes used to either store the address of the IDMS communications block after an I/O error occurs during execution of a DML command, or to receive information from the DC/UCF system regarding certain status conditions that are associated with a return code.
- **Register 15** is used to receive the return code from the system after execution of a DML verb that requests a data communications service.

  The value of the return code in register 15 indicates whether a DML request for data communication services was successful. The return codes issued by the system after execution of a DML statement are listed on the following pages.

**Note:** If your program uses DML commands to request data communication services *and* to access the CA IDMS/DB database, you must check register 15 for return codes issued by the DC/UCF system, and the ERRSTAT field of the IDMS communications block for the status codes issued by CA IDMS/DB.

## DC/UCF Status Codes

Following each DML request for data communication services, the system places a return code in register 15 to indicate either an error or a specific condition that occurred during processing. Table 3-3 lists the runtime register 15 return codes for the DML statements associated with DC/UCF services. Specific return codes are listed for each command in Chapter 6.

For every DML verb, a register 15 value of X'00' indicates that the request has been serviced successfully.

The following table shows the DC/UCF Runtime Register 15 Return Codes.

| R15 Value | DML Verb | Return Condition |
|---|---|---|
| X'00' | All verbs | No error |
| | #ENQ | ■ ACQUIRE—All requested resources have been acquired.<br>■ TEST—All tested resources have already been enqueued by the issuing task with the EXCLUSIVE/SHARED option specified by the test request. |
| | #SETIME | The request to cancel a previously issued #SETIME has been serviced successfully. |
| X'04' | #ATTACH | The maximum number of tasks has already been attached; no new tasks can be attached at this time. |
| | #COMMIT | Internal run-unit table full; check the CA IDMS/DC log for details. |
| | #DELQUE | The parameter list is invalid. |
| | #DELSCR | The parameter list is invalid. |
| | #DEQ | At least one resource id (RSCID) cannot be found; all that were located have been dequeued. |

| R15 Value | DML Verb | Return Condition |
|---|---|---|
| | #ENQ | ■ ACQUIRE—At least one of the resources indicated is currently owned by another task and is not available for the EXCLUSIVE/SHARED option specified; no new resources have been acquired. ■ TEST—At least one of the tested resources is owned by another task and is not available to this task for the EXCLUSIVE/SHARED option specified. |
| | #FINISH | There are too many run units for the internal run-unit table. This is a system internal error and should be reported. |
| | #GETQUE | The parameter list is invalid. |
| | #GETSCR | The parameter list is invalid. |
| | #GETSTG | The request specified a storage id that did not previously exist; the indicated space has been allocated. |
| | #LINEIN | The input area specified for return of data to the issuing program is too small to accommodate the full data stream; the returned data has been truncated accordingly. |
| | #LINK | Either the request cannot be serviced because of an I/O, program-not-found, or potential deadlock error or no null program definition elements (PDEs) have been allocated. If the load fails, the link will fail and a minor code will be returned in register 1. |
| | #LOAD | There is not enough space in the program pool to load the program. |
| | #MREQ | The specified edit or code table cannot be found or is invalid for use with the named map. |
| | #PRINT | An I/O error occurred during processing. |
| | #PUTJRNL | The derived journal record length is zero or negative. |
| | #PUTQUE | Invalid #PUTQUE request. Check for proper queue-id specification and logical selection of options. |

| R15 Value | DML Verb | Return Condition |
|---|---|---|
| | #PUTSCR | Invalid request. Check for proper scratch-id specification and logical selection of options as specified in the #PUTSCR statement. |
| | #ROLLBAK | Internal run-unit table full; check the CA IDMS/DC log for details. |
| | #SENDMSG | An I/O error occurred during processing. |
| | #SETIME | For a #SETIME TYPE=CANCEL request, the internal control element (ICE) address specified cannot be found. |
| | #STRTPAG | A paging session was already in progress when another #STRTPAG command was issued. An implied #ENDPAG has been processed and the #STRTPAG has been executed successfully. |
| | #TREQ | For a #TREQ GET, #TREQ PUTGET, or #TREQ CHECK request, the input area specified for the return of data to the issuing program is too small to accommodate the full data stream; the returned data has been truncated accordingly. |
| | #TRNSTAT | A new transaction statistics block (TSB) has been allocated. |
| X'08' | #ATTACH | The requested task code is invalid. |
| | #COMMIT | An invalid request has been issued. #COMMIT is valid only if the program accesses CA IDMS/DB database or dictionary entities (that is, CA IDMS/DB records or DC/UCF scratch/queue records). Typically, #COMMIT need be specified only when CA IDMS/DB database or dictionary entities are accessed in an update usage mode. |
| | #DELQUE | The requested queue header record (QUEID) cannot be found. |
| | #DELSCR | The requested scratch area id (SAID) cannot be found. |
| | #ENQ | ■ ACQUIRE—Not applicable.<br>■ TEST—At least one of the tested resources is not already owned by any task and is available for the EXCLUSIVE/SHARED option specified. If both conditions described for return codes X'04' and X'08' exist, the register 15 value will be X'04'. |

| R15 Value | DML Verb | Return Condition |
|---|---|---|
| | #FINISH | An invalid request has been issued. #FINISH is only valid if the program accesses CA IDMS/DB database or dictionary entities (that is, CA IDMS/DB records or DC/UCF scratch/queue records). #FINISH need be specified only when the program performs database or dictionary accessing activities. |
| | #GETQUE | The requested queue header record (QUEID) cannot be found. |
| | #GETSCR | The requested scratch area id (SAID) cannot be found. |
| | #GETSTG | There is insufficient storage in the storage pool to process the request. |
| | #LINEIN | The I/O session has been canceled; the terminal operator has pressed the CLEAR (3270), ATTENTION (2741), or BREAK (teletype) key. |
| | #LINEOUT | The I/O session has been canceled; the terminal operator has pressed the CLEAR (3270), ATTENTION (2741), or BREAK (teletype) key. |
| | #LOAD | An I/O error occurred during a load from a load library. |
| | #MREQ | I/O has been interrupted; the terminal operator has pressed the ATTENTION (2741) or CLEAR (3270) key. |
| | #PRINT | The parameter list passed to #PRINT contains an invalid field. |
| | #PUTJRNL | The required storage is not available for the necessary control blocks. |
| | #ROLLBAK | An invalid request has been issued. There is a possible logic error in the program. Ensure that checkpoints are made (by means of #COMMIT) in the program logic before the #ROLLBAK request. |
| | #SENDMSG | The parameter list is invalid. |
| | #TREQ | For a #TREQ GET, #TREQ PUTGET, or #TREQ CHECK request, output has been interrupted; the terminal operator has pressed the ATTENTION (2741) or CLEAR (3270) key. |

| R15 Value | DML Verb | Return Condition |
|---|---|---|
| | #TRNSTAT | Storage for the transaction statistics block (TSB) is not available; waiting would cause a deadlock. |
| | #WAIT | Waiting on the specified ECBs would cause a deadlock. |
| X'0C' | #ATTACH | The request cannot be serviced due to a security violation. |
| | #COMMIT | An invalid status has been issued from DBIO/DBMS; check the CA IDMS/DC log for details. |
| | #DELQUE | The requested queue record cannot be found |
| | #DELSCR | The requested scratch record id (SRID) cannot be found within the named SAID. |
| | #ENQ | ■ ACQUIRE—A requested resource cannot be enqueued immediately and waiting would cause a deadlock; no new resources have been acquired.<br>■ TEST—Not applicable. |
| | #FINISH | An invalid status has been issued from DBIO/DBMS; check the CA IDMS/DC log for details. |
| | #GETQUE | The requested queue record cannot be found. |
| | #GETSCR | The requested scratch record id (SRID) cannot be found within the named SAID. |
| | #GETSTG | The parameter list is invalid. |
| | #LINEIN | A logical or permanent I/O error has been encountered in the input data stream. |
| | #LINEOUT | A logical or permanent I/O error has been encountered in the output data stream. |
| | #LOAD | The requested program is nonconcurrent and in use. |
| | #MREQ | A logical error (for example, invalid control character) has been encountered in the output data stream. |
| | #PRINT | No printer logical terminals have been defined in this DC/UCF system. |

| R15 Value | DML Verb | Return Condition |
|---|---|---|
| | #PUTJRNL | An invalid error status has been issued from DBIO/DBMS; check the IDMS/DC log for details. |
| | #ROLLBAK | An invalid error status has been issued from DBIO/DBMS; check the IDMS/DC log for details. |
| | #SENDMSG | The message destination is undefined. |
| | #TREQ | For a #TREQ GET, #TREQ PUTGET, or #TREQ CHECK request a logical error (for example, invalid control character) has been encountered in the output data stream. |
| | #TRNSTAT | No transaction statistics block (TSB) exists; #TRNSTAT TYPE=BIND has not been issued. This return code is valid only for #TRNSTAT TYPE=ACCEPT and #TRNSTAT TYPE=END statements. |
| X'10' | #DELQUE | No resource control element (RCE) exists for the queue record; currency has not been established. |
| | #GETSTG | The requested storage cannot be allocated immediately (insufficient storage) and waiting would cause a deadlock. |
| | #LINEIN | The line request block (LRB) contains an invalid field. |
| | #LINEOUT | The line request block (LRB) contains an invalid field. |
| | #LOAD | The requested program has been temporarily overlayed in the program pool, resulting in a storage conflict. |
| | #MREQ | A permanent I/O error occurred during processing. |
| | #PRINT | A print screen request has been made from a non-3270-type terminal or from a 3270-type terminal without read buffer support. |
| | #PUTSCR | The request to replace a scratch record has been serviced successfully. |
| | #TREQ | For a #TREQ GET, #TREQ PUTGET, or #TREQ CHECK request, a permanent I/O error occurred during processing. |
| | #TRNSTAT | Either the task in question is not associated with a terminal or the request is invalid. |

| R15 Value | DML Verb | Return Condition |
|---|---|---|
| X'14' | #LINEOUT | The name specified for DESTID, USERID, or LTERMID is unknown to this DC/UCF system. |
| | #LOAD | The requested program is not defined to the program definition table (PDT), the requested program is marked as out of service, or a null program definition element (PDE) could not be allocated for the program. |
| | #MREQ | The dial-up line for the terminal is disconnected. |
| | #PRINT | Either the specified printer destination is invalid or, for OPTNS=DIRECT, LTEID or LTEADDR is invalid. |
| | #PUTSCR | The request to add a new scratch record cannot be processed because the record id specified by the SRID operand already exists for the named scratch area. |
| | #TREQ | For a #TREQ GET, #TREQ PUTGET, or #TREQ CHECK request, the dial-up line for the terminal is disconnected. |
| | #TRNSTAT | Transaction statistics or task statistics are not enabled in this DC/UCF system. |
| X'18' | #GETQUE | The user area specified for the return of the queue record is too small; the returned record has been truncated to fit in the available storage space. |
| | #GETSCR | The user area specified for the return of the scratch record is too small; the returned record has been truncated to fit in the available storage space. |
| | #GETSTG | Allocated XA storage above the 16 megabyte line cannot be addressed by a 24-bit task. |
| | #LOAD | The requested program cannot be loaded immediately due to insufficient space; waiting would cause a deadlock. |
| | #MREQ | The terminal being used is out of service. |
| | #PRINT | A terminal I/O error occurred during a #PRINT request. |

| R15 Value | DML Verb | Return Condition |
|---|---|---|
| | #TREQ | For a #TREQ GET, #TREQ PUTGET, or #TREQ CHECK request, the terminal being used is out of service. |
| X'1C' | #DELQUE | An I/O error occurred during a delete queue operation. |
| | #DELSCR | An I/O error occurred during a delete scratch operation. |
| | #GETQUE | An I/O error occurred during get queue processing. |
| | #GETSCR | An I/O error occurred during get scratch processing. |
| | #PRINT | No printer can be found to satisfy the print-direct request and OPTNS=NOWAIT has been specified. |
| | #PUTSCR | An I/O error occurred during processing. |
| | #TREQ | For a #TREQ GET, #TREQ PUTGET, or #TREQ CHECK request, the terminal is closed or was never opened. |
| X'20' | #ATTACH | The maximum number of concurrent tasks has been reached. |
| | #LOAD | An I/O error occurred during a load from the dictionary DDLDCLOD area. |
| | #MREQ | The map request block (MRB) contains an invalid field, indicating a possible error in application program parameters. |
| | #PRINT | The print-direct request specified an LTEID or LTEADDR that is out of service. |
| | #TREQ | The terminal request block (TRB) contains an invalid field. |
| X'24' | #MREQ | The map load module requested by the map request block (MRB) cannot be found. |
| | #PRINT | The print-direct request specified a wait; waiting would cause a deadlock. |
| | #TREQ | The name specified for DESTID, LTERMID, or USERID is invalid. |
| X'28' | #MREQ | The requested map does not support the terminal device type being used. |

| R15 Value | DML Verb | Return Condition |
|---|---|---|
| | #PRINT | A DCMT VARY PRINTER CANCEL command has been issued in the DC/UCF system for this direct printer. |
| X'2C' | #MREQ | An error was detected upon return from a user-written edit module. An invalid pointer to the data stream has been returned to register 1. |
| | #PRINT | A DCMT VARY PRINTER REQUEUE command had been issued in the DC/UCF system for this direct printer. |
| X'30' | #MREQ | Invalid internal data has been encountered. Either the data in the record does not match the internal data or the internal data cannot be converted to the external format, as specified in the external picture. |
| X'34' | #MREQ | The named user-written edit module cannot be found. |
| X'38' | #MREQ | An invalid immediate write request to DESTID, LTERMID, or USERID has been issued. |
| X'3C' | #MREQ | The map load module is invalid. |
| X'40' | #MREQ | For an #MREQ IN request, the requested node for a header or detail was either not present or not updated. For an #MREQ OUT request, there is no current detail occurrence to be updated. No action is taken. |
| X'44' | #MREQ | No more modified detail occurrences require a mapin. For an #MREQ OUT request, the maximum amount of storage defined for pageable maps during system generation is insufficient. |
| X'48' | #MREQ | For an #MREQ IN request, the scratch record that contains the requested detail could not be accessed (internal error). For an #MREQ OUT,RESUME request, no detail occurrence, footer, or header fields exist. |
| X'4C' | #MREQ | For an #MREQ OUT request, the first screen page has been transmitted to the terminal. |

| R15 Value | DML Verb | Return Condition |
|---|---|---|
| X'50' | #MREQ | An #MREQ IN,COND=MPNS or #MREQ OUT,COND=MPNS request has been received when no map paging session is in progress. Either a #STRTPAG command was not issued prior to this #MREQ IN command or a #ROLLBAK was issued that rendered the scratch area for the pageable map (area id MPGPSCRA) unavailable. If the COND specification is not MPNS, this condition abends the map paging task. |

## Testing for DC/UCF Return Codes

Testing for the return code in register 15 is usually not necessary because most DML commands have options that take action based on the return code value.

**Specifying Conditions**

The COND (condition) parameter provides a conditional return to the issuing program should a specified error or special condition occur during processing. This return of control can be directed to one of the following locations:

- The next sequential instruction
- A specified exit routine

The options of a COND parameter consist of statement-specific conditions that can occur during the execution of a DML statement. Any number of conditions can be specified. For example, the following COND parameter requests a return of control in the event of an I/O error or deadlock condition:

```
COND=(IOER,DEAD)
```

If a condition associated with a specified parameter arises, control will be returned to the issuing program. If a condition occurs for which no COND parameter is coded, a default action will be taken. Typically, the default action either aborts the task or waits for the condition to change.

**Specifying COND with an Exit Routine**

When more than one conditional parameter is permitted, you can code the value ALL to indicate that all of the permitted COND parameters apply. If a condition corresponding to an available parameter occurs and ALL is specified, control will be returned to the issuing program.

Most DC/UCF DML statements provide the facility to associate an exit routine with each special condition. To return control to a specific exit when a condition occurs, you include both the appropriate condition (COND parameter) and the name of its associated exit routine.

For example, a DML statement may include a COND parameter of IOER and the IOERXIT parameter, which names the routine to which control will be returned in the event of an I/O error that occurs during execution of the DML command; for example:

```
COND=(IOER),IOERXIT=IOERROR
```

**Specifying COND Without an Exit Routine**

Specifying only the COND parameter without an exit routine causes a return of control to the next sequential instruction in the program that issues the DML statement. In this case, you can examine the contents of register 15 to determine which condition code was set as a result of the operation.

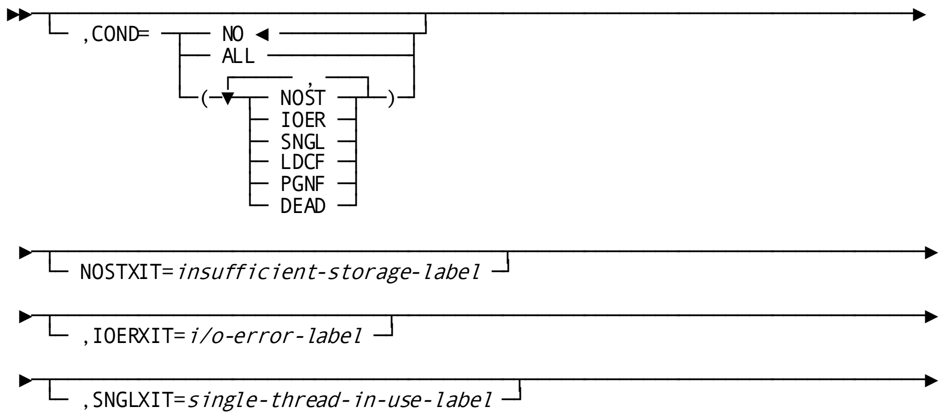
**Specifying a General Exit Routine**

You can specify a general exit routine by using the ERROR parameter. The system passes control to this routine when a condition occurs for which no specific exit routine was coded.

**Note:** If a condition occurs for which an associated exit routine and the ERROR parameter are specified, control will be returned to the routine named by the specific exit. If you have multiple exit routines containing the same logic, you should specify only the ERROR parameter to avoid redundant coding.

**Syntax**

The following syntax lists the COND parameter and exit routines found in the #LOAD statement. The NOSTXIT exit is associated with the NOST condition, the IOERXIT exit is associated with the IOER condition, and so forth.


,COND=
NO
ALL
(
,
NOST
IOER
SNGL
LDCF
PGNF
DEAD
)
NOSTXIT=*insufficient-storage-label*
,IOERXIT=*i/o-error-label*
,SNGLXIT=*single-thread-in-use-label*

```
►─┬──────────────────────────────────────────────┬──────────────────────►
  └─ ,LDCFXIT=storage-location-conflict-label ─┘

►─┬──────────────────────────────────────┬──────────────────────────────►
  └─ ,PGNFXIT=program-not-found-label ─┘

►─┬─────────────────────────────┬───────────────────────────────────────►
  └─ ,DEADXIT=deadlock-label ─┘

►─┬────────────────────────┬────────────────────────────────────────────►◄
  └─ ,ERROR=error-label ─┘
```

Some DML statements have only a single condition, as the following excerpt from the #LINK statement illustrates.

**Syntax**

```
►─┬───────────────────────┬─────────────────────────────────────────────►
  └─ ,COND= ─┬─ NO ◄ ──┬─┘
             └─ PGNA ──┘

►─┬──────────────────────────────────────────┬──────────────────────────►
  └─ ,PGNAXIT=program-not-available-label ─┘

►─┬────────────────────────┬────────────────────────────────────────────►
  └─ ,ERROR=error-label ─┘
```

In this case, the general ERROR parameter functions identically to the specific PGNAXIT parameter. It supplies the name of a routine to which the program will branch when a program-not-available error occurs.

**Note:** The COND parameter list is enclosed in parentheses. If multiple parameters are specified, each parameter is separated from the previous one by a comma.

**Example of COND in #LOAD**

The following example of the #LOAD statement attempts to load the program JOBMAP1 into the program pool. The COND parameter is set to PGNF, which will return control to the issuing program only if the requested program cannot be dynamically loaded or is marked as out of service. The return code for this condition is X'14'.

In this example, if the return code matches the PGNF condition, the system returns control to the issuing program at label ERRMSG, indicated by the ERROR parameter. If the system returns a code of X'00' the program JOBMAP1 will have been successfully loaded into the program pool. Return codes other then X'00' or X'14' will result in an abend and control will be returned to either a higher-level program or the system.

```
LOAD1    #LOAD PGM=JOBMAP1,COND=PGNF,ERROR=ERRMSG
         .
         .
         .
ERRMSG   EQU   *
         .
         .
```

**Testing for DML Statements that Request DC/UCF Services**

In addition to the COND parameter, you can test for the return code value in register 15 for each DML statement that requests DC/UCF services. Your program can compare the register 15 value to a literal or a defined constant, then execute a conditional branch.

In the following example, if the value in register 15 equals the numeric value 0000, the program branches to the label CONTINU. Any value other than zero causes a branch to the program label RCCOND.

```
         .
         .
         C     15,=F'0'
         BE    CONTINU
         B     RCCOND
         .
         .
CONTINU  EQU   *
         .
         .
         .
RCCOND   EQU   *
         .
         .
```

# Chapter 5: Data Manipulation Language Statements

This chapter describes each data manipulation language (DML) statement that requests CA IDMS/DB database access or online service. The DML commands are presented in two ways:

- The first table presents the commands by function.
- Each DML command is presented in alphabetical order. The discussion of each command includes:
  - A description of the DML statement
  - Syntax and syntax rules

- Currency considerations, where applicable
- An example of how to use the statement
- Error handling after a DML statement is issued

The WHERE and ON clauses that are used with DML statements to access logical records created by the Logical Record Facility (LRF) are described at the end of this chapter.

This section contains the following topics:

## Functions of DML Statements

The data manipulation language enables you to access the database management system (DBMS) and to request LRF and DC/UCF services from your Assembler program. The DML statements can be grouped into 14 categories by function:

- **Control** statements perform the following:
  - Initiate and terminate processing
  - Effect recovery
  - Prevent concurrent retrieval and update of database records
  - Evaluate set conditions
- **Retrieval** statements locate records in the database and make them available to the application program.
- **Modification** statements add new records to the database and modify and delete existing records.
- **Accept** statements allow you to move special information such as database keys, storage addresses, and statistics from the DBMS to the application program's variable storage.
- **Logical-record** statements allow you to retrieve, modify, store, and erase logical records created through Logical Record Facility.
- **Program management** statements perform the following:
  - Pass and return control from one program to another
  - Load and delete programs and tables
  - Define exit routines to be performed before an abnormal program termination (abend)
  - Force an abend condition
- **Storage management** statements allocate and release variable storage.
- **Task management** statements perform the following:
  - Initiate a new task
  - Change the dispatching priority of the issuing task
  - Enqueue and dequeue system resources
  - Signal that a task is to wait pending completion of an event
  - Post an event control block (ECB) indicating completion of an event
- **Time management** statements obtain the time and date and set up time-related events. These events include:
  - Placing the issuing task in a wait state for a specified time

- Posting a user-specified ECB after a specified interval
- Initiating a new task after a specified interval

- **Scratch management** statements create, delete, or retrieve records from the scratch area.
- **Queue management** statements create, delete, or retrieve records from the queue area.
- **Terminal management** statements transfer data between the application program and a terminal or printer.
- **Utility function** statements perform the following:
  - Request retrieval of task-related information
  - Request a memory dump of selected parts of storage
  - Retrieve and send a predefined message stored in the dictionary
  - Send a specified message to one or more users or logical terminals
  - Collect, retrieve, and write DC/UCF system statistics on a transaction basis
  - Establish long-term database locks and monitor access to database records used across tasks in a pseudo-conversational transaction
- **Recovery** statements perform functions relating to database, scratch, and queue area recovery in the event of a system failure. These functions perform the following:
  - Establish checkpoints on the journal file for database, scratch, and queue records used by the issuing task
  - Roll back user database, scratch, and queue record areas to the last checkpoint established
  - Establish an end-of-task checkpoint and relinquish control of all database, scratch, and queue record areas associated with the issuing task
  - Write user defined records to the journal file

The following table groups the DML statements by function and gives a brief description of each command.

**DML Statements Grouped by Function**

| Function | DML statement | Description |
|---|---|---|
| Control Statements | @BIND SUBSCH | Signs on the application program to the CA IDMS/DB database management system |

| Function | DML statement | Description |
|---|---|---|
| | @BIND REC | Establishes addressability in variable storage for one or more records included in the program's subschema |
| | @BIND PROC | Establishes communication between the application program and a DBA-defined database procedure |
| | @READY | Prepares database areas for processing |
| | @FINISH | Commits changes made to the database through an individual run unit and terminates the run unit |
| | @IF | Evaluates the presence of records in a set and specifies action based on the outcome |
| | @COMMIT | Commits changes made to the database by an individual run unit |
| | @ROLLBAK | Rolls back uncommitted changes made to the database through an individual run unit |
| | @KEEP | Places locks on record occurrences |
| Retrieval Statements | @FIND/OBTAIN DBKEY | Accesses a record by using a db-key previously saved by the program |
| | @FIND/OBTAIN CURRENT | Accesses a record by using established currencies |
| | @FIND/OBTAIN WITHIN SET/AREA | Accesses a record based on its logical location within a set or its physical location within an area |
| | @FIND/OBTAIN OWNER | Accesses the owner record of a set occurrence |
| | @FIND/OBTAIN CALC/DUPLICATE | Accesses a record by using its CALC-key value |
| | @FIND/OBTAIN USING SORT KEY | Accesses a record in a sorted set by using its sort-key value |

| Function | DML statement | Description |
|---|---|---|
| | @GET | Moves all data associated with a previously located record into program variable storage |
| Modification Statements | @STORE | Adds a new record to the database |
| | @MODIFY | Changes the contents of an existing record |
| | @CONNECT | Links a record to a set |
| | @DISCON | Removes a member record from a set |
| | @ERASE | Deletes a record from the database |
| Accept Statements | @ACCEPT DBKEY FROM CURRENCY | Saves the db-key of the current record of run unit, record type, set, or area |
| | @ACCEPT DBKEY RELATIVE TO CURRENCY | Saves the db-key of the next, prior, or owner record relative to the current record of a set |
| | @ACCEPT PAGE INFORMATION FOR A GIVEN RECORD | Saves the page information for a record current record of a set |
| | @ACCEPT STATS | Returns system runtime statistics to the program |
| | @ACCEPT BIND | Returns a record's bind address to the program |
| | @ACCEPT PGINFO | Returns page information for a given record to the program |
| | @ACCEPT PROC | Returns information in the application program information block associated with a database procedure to the program |
| | @RETURN | Retrieves a database key of a record entry that has been indexed under integrated indexing. |
| Logical Record Facility (LRF) Statements | @OBTAIN logical-record @MODIFY logical-record @STORE logical-record @ERASE logical-record | Retrieves a logical record Modifies a logical record Stores a new logical record Deletes a logical record |

| Function | DML statement | Description |
|---|---|---|
| Program Management Statements | #LINK | Passes control to another program with the expectation of receiving it back |
| | #RETURN | Returns control to the next higher level calling program |
| | #LOAD | Loads a program or table into the program pool |
| | #DELETE | Signals that the program has finished using a program or table in the program pool |
| | #STAE | Establishes linkage to a program or routine that will receive control in the event of an abend |
| | #ABEND | Abnormally terminates the issuing task |
| | #XCTL | Passes control to another program with no expectation of having it returned |
| Storage Management Statements | #GETSTG<br>#FREESTG | Allocates variable storage from a DC/UCF storage pool Frees all or part of a block of variable storage |
| Task Management Statements | #ATTACH | Attaches a new task within the DC/UCF system |
| | #CHAP | Changes the dispatching priority of the issuing task |
| | #ENQ | Acquires a resource or a list of resources |
| | #DEQ | Releases a resource |
| | #WAIT | Relinquishes control to the system while awaiting the completion of an event |
| | #POST | Posts an event control block |
| Time Management Statements | #GETIME | Obtains the time and date from the system |
| | #SETIME | Defines a timed event |

| Function | DML statement | Description |
|---|---|---|
| Scratch Management Statements | #PUTSCR #GETSCR #DELSCR | Stores a scratch record Retrieves a scratch record Deletes a scratch record |
| Queue Management Statements | #PUTQUE #GETQUE #DELQUE | Stores a queue record Retrieves a queue record Deletes a queue record |
| Terminal Management (Basic Mode) | #TREQ | Transfers data and device dependent information to or from the terminal, or establishes a terminal request block (TRB) for use by subsequent #TREQ operations. The #TREQ statement can be used to communicate in an SNA network environment |
| Terminal Management (Line Mode) | #LINEIN | Requests a synchronous data transfer from the terminal to the issuing program |
| | #LINEOUT | Requests a synchronous or asynchronous data transfer from the issuing program to the terminal |
| | #LINEEND | Terminates the current line I/O session |
| Terminal Management (Mapping Mode) | #MREQ | Requests a transfer of data from the issuing program to the terminal and/or vice versa |
| | #MAPINQ | Obtains information or tests conditions concerning the previous map operation |
| | #MAPMOD | Requests modifications of mapping options for a map |
| | #STRTPAG | Begins a map paging session and specifies options for that session |
| | #ENDPAG | Terminates a map paging session |
| Terminal Management (Print Mode) | #PRINT | Transfers data from a task to a terminal defined as a printer. |

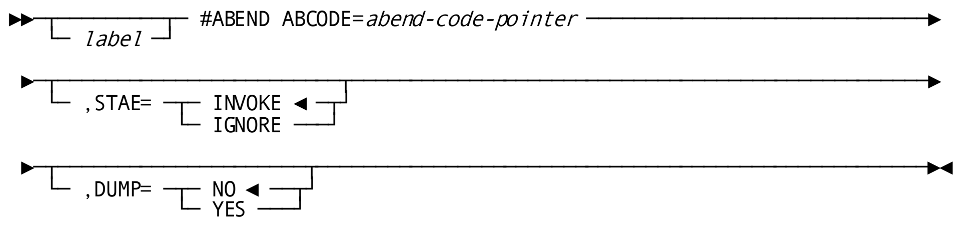
| Function | DML statement | Description |
|---|---|---|
| Utility Functions | #ACCEPT | Retrieves task-related information |
| | #SNAP | Requests a memory dump of selected parts of storage |
| | #SENDMSG | Sends a message to a user, logical terminal, list of users, or list of logical terminals |
| | #TRNSTAT | Requests or terminates statistics collection; retrieves transaction statistics into program storage |
| | #KEEP | Enables database locks or database monitoring for records, sets, or areas or terminates a prior #KEEP request |
| | #WTL | Retrieves a message from the dictionary and sends it to a predefined destination |
| Recovery Statements | #COMMIT | Commits changes made to the database through an individual run unit or through all database sessions associated with a task |
| | #FINISH | Commits changes made to the database through an individual run unit or through all database sessions associated with a task |
| | #ROLLBAK | Rolls back uncommitted changes made to the database through an individual run unit or through all database sessions associated with a task |
| | #PUTJRNL | Writes user-defined records to the journal file |

## #ABEND—terminates the issuing task abnormally

The #ABEND statement terminates the issuing task abnormally and specifies whether the system invokes previously established abend exits or writes a task dump to the log file.

After completion of the #ABEND function, control is returned to the system.

## #ABEND Syntax



## #ABEND Parameters

**ABCODE=**

Specifies a 4-character user-defined abend code.

***abend-code***

A register pointing to a field that contains the abend code, the symbol name of a user-defined field containing the code, or the abend-code literal enclosed in single quotation marks.

Note: Because the specified abend code appears in the system log and is displayed at the task's terminal, you should not use DC/UCF system abend codes.

**STAE=INVOKE/IGNORE**

Specifies whether the system invokes or ignores abend routines that were previously established by #STAE requests; the default is INVOKE.

**DUMP=NO/YES**

Specifies whether the system writes a formatted task dump to the DC/UCF log file. The default is NO.

## #ABEND Example

**Example: Terminating the issuing task**

The following example of the #ABEND statement terminates the issuing task abnormally and specifies the register that points to a field in the application program containing the abend code. This statement requests that the system ignore abend routines and to write a task dump to the DC/UCF log file. Control returns to the system after completion of the #ABEND statement.

```
#ABEND ABCODE=(R12),STAE=IGNORE,DUMP=YES
```

## #ABEND Status Codes

The #ABEND request is unconditional; control is passed to the DC/UCF program control module.

## @ACCEPT BIND—moves the bind address

The @ACCEPT BIND statement moves the bind address of a record to a location in program variable storage. The requesting program is usually a subprogram that requires the address of a record in order to access it.

**Currency**

Currency must be established for the record whose bind address will be returned to the application program.

A successful execution of the @ACCEPT BIND command does not update the currency of the record type or the run unit.

### @ACCEPT BIND Syntax

```
►►── @ACCEPT BIND=bind-address────────────────────────────────────────►
 ►── ,REC=record-name ────────────────────────────────────────────────►◄
```

### @ACCEPT BIND Parameters

**BIND=*bind-address***

Specifies the 4-byte (fullword) location in storage to which the system returns the record's bind address. Note that *bind-address* does not specify a database key.

**REC=**

Specifies the record whose bind address will be returned to the specified location in program variable storage.

***record-name***

Must be a record previously bound by the run unit.

## @ACCEPT BIND Status Codes

After completion of the @ACCEPT BIND statement, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**1508**

The specified record is not in the named subschema

## @ACCEPT BIND Example

The following @ACCEPT BIND statement moves the bind address for an EMPLOYEE record to register 1.

```
@ACCEPT BIND=(R1),REC='EMPLOYEE'
```

## @ACCEPT DBKEY FROM CURRENCY—moves the db-key of the current record

The @ACCEPT DBKEY FROM CURRENCY statement moves the db-key of the current record of run unit, record type, set, or area to a specified location in program variable storage. Use the PGINFO option to specify a location in program variable storage where the page information associated with the returned dbkey is moved. Records whose db-key are saved in this manner are available for subsequent direct access by using an @FIND/@OBTAIN DBKEY statement.

**Currency**

The record must be current of run unit, record type, set, or area before execution of the @ACCEPT DBKEY FROM CURRENCY statement. Currency is maintained but not updated after the statement is executed.

**Note:** You must establish set currency before using this statement. If no set currency has been established, the DBMS returns 0000 to the ERRSTAT field and -1 to the DB-KEY field.

For more information on page information fields, see @ACCEPT PGINFO (see page 90).

## @ACCEPT DBKEY FROM CURRENCY Syntax

```
►►─── @ACCEPT DBKEY=db-key ──┬──────────────────┬──────────────────►
                             └ ,PGINFO=pg-info-v ┘

►─┬────────────────────┬──────────────────────────────────────────►◄
  ├ ,REC=record-name  ─┤
  ├ ,SET=set-name     ─┤
  └ ,AREA=area-name   ─┘
```

## @ACCEPT DBKEY FROM CURRENCY Parameters

**DBKEY=*db-key***

Identifies the location in variable storage that will contain the db-key of the named record. Must identify a full-word binary field.

**PGINFO=*pg-info-v***

Specifies the name of a four-byte field that is made up of two halfword fields. Identifies the location in variable storage that contains page information for the specified record. Upon successful completion of this statement, the first two bytes of the field contain the page group number and the last two bytes contain a value that may be used for interpreting dbkeys.

**REC=*record-name*/SET=*set-name*/AREA=*area-name***

Specifies the record whose db-key will be placed in the location identified by *db-key*. If the record, set, or area qualifiers are omitted, the db-key of the current record of run unit is saved. Otherwise, db-keys are saved as follows:

- **REC=***record-name* saves the db-key of the record that is current of the specified record type.
- **SET=***set-name* saves the db-key of the record that is current of the specified set.
- **AREA=***area-name* saves the db-key of the record that is current of the specified area.

## @ACCEPT DBKEY FROM CURRENCY Status Codes

After completion of the @ACCEPT DBKEY FROM CURRENCY function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation.

The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**1508**

The specified record is not in the subschema. The program has probably invoked the wrong subschema.

## @ACCEPT DBKEY FROM CURRENCY Example

The following statements illustrate the use of the @ACCEPT DBKEY FROM CURRENCY statement. The program performs the following steps:

1. Establishes an EMPLOYEE record as current of run unit
2. Saves its db-key in location SAVEDKEY
3. Accesses the EMPLOYEE record occurrence by using the saved db-key, after further processing has changed currency

```
MVC    EMPID,=CL4'7690'
@FIND CALC,REC='EMPLOYEE'
@ACCEPT DBKEY=SAVEDKEY
.
.
@OBTAIN DBKEY=SAVEDKEY
```

## @ACCEPT DBKEY RELATIVE TO CURRENCY—moves the db-key

The @ACCEPT DBKEY RELATIVE TO CURRENCY statement moves the db-key of the next, prior, or owner record relative to the current record of set to a location in variable storage. Use the PGINFO option to specify a location in program variable storage where the page information associated with the returned dbkey is moved. The DBMS examines the current record of the named set and extracts the requested pointer from its prefix.

This statement allows you to save the db-key of a record within a set without actually having to access the record. Records whose db-keys are saved in this manner are available for subsequent direct access by an @FIND/@OBTAIN DBKEY statement.

**Note: Native VSAM users**—The @ACCEPT DBKEY RELATIVE TO CURRENCY statement is not valid for native VSAM data sets.

**Note:** You must establish set currency before using this statement. If no set currency has been established, the DBMS returns 0000 to the ERRSTAT field and -1 to the DB-KEY field.

**Currency**

Currency is not updated after execution of an @ACCEPT DBKEY RELATIVE TO CURRENCY statement. The record that is current of record type before the @ACCEPT statement will remain current immediately after the statement is executed.

For more information on page information fields, see @ACCEPT PGINFO (see page 90).

## @ACCEPT DBKEY RELATIVE TO CURRENCY Syntax

```
►►── @ACCEPT DBKEY=db-key ─┬─────────────────────┬──────────────────►
                           └ ,PGINFO=pg-info-v ──┘

 ►──┬─ ,SETN= ─┬── set-name ──────────────────────────────────────►◄
    ├─ ,SETP= ─┤
    └─ ,SETO= ─┘
```

## @ACCEPT DBKEY RELATIVE TO CURRENCY Parameters

**DBKEY=*db-key***

Identifies the location in variable storage that will contain the db-key of the requested record.

**PGINFO=pg-info-v**

Specifies the name of a four-byte field that is made up of two halfword fields. Identifies the location in variable storage that contains page information for the specified record. Upon successful completion of this statement, the first two bytes of the field contain the page group number and the last two bytes contain a value that may be used for interpreting dbkeys.

**SETN=/SETP=/SETO=*set-name***

Determines the record whose db-key will be placed in the location identified by *db-key. Set-name* must be a set included in the subschema. The saved db-key can belong to the next, prior, or owner record relative to the current record of the named set:

- **SETN=***set-name* saves the db-key of the next record relative to the record that is current of the specified set. A request for SETN currency cannot be specified unless the named set has prior pointers; prior pointers ensure that the next pointer in the prefix of the current record does not point to a logically deleted record.

- **SETP=***set-name* saves the db-key of the prior record relative to the record that is current of the specified set. A request for SETP currency cannot be specified unless the named set has prior pointers.

  Note: No indication of an end-of-set condition is possible for an @ACCEPT SETN or SETP. A retrieval statement must be issued to determine whether the next or prior record in the set occurrence is the owner record.

- **SETO=***set-name* saves the db-key of the owner of the current set. A request for SETO currency cannot be executed unless the named set has owner pointers. If the current record of the named set is the owner record occurrence, requests for SETO currency return the db-key of the record itself, even if this set does not have owner pointers.

  Note: When a record declared as an optional or manual member of a set is accessed, it is *not* established as current of set if it is not currently connected to the named set. A subsequent attempt to access the owner record will instead locate the owner of the current record of set. In such cases, determine whether the retrieved record is actually a set member before executing the @ACCEPT DBKEY=*db-key*, SETO=*set-name* statement. The @IF statement (see "@IF" later in this chapter) can be used for this purpose.

## @ACCEPT DBKEY RELATIVE TO CURRENCY Example

The following statements illustrate the use of the @ACCEPT DBKEY RELATIVE TO CURRENCY statement. The program performs the following steps:

1. Traverses the DEPT-EMPLOYEE set
2. Saves the db-key of the owner record of the OFFICE-EMPLOYEE set
3. Accesses the owner record of the OFFICE-EMPLOYEE set by using the saved db-key, after further processing has changed currency

```
MVC    DEPTID,=CL4'1234'

@FIND CALC,REC='DEPARTMENT'

@FIND NEXT,SET='DEPT-EMPLOYEE'

@ACCEPT DBKEY=SAVDKEY,SETO='OFFICE-EMPLOYEE'

.

.

@OBTAIN DBKEY=SAVEDKEY
```

### @ACCEPT DBKEY RELATIVE TO CURRENCY Status Codes

After completion of the @ACCEPT DBKEY RELATIVE TO CURRENCY function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation.

The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**1508**

The specified record is not in the subschema. The program has probably invoked the wrong subschema.

## @ACCEPT PGINFO—moves the page information

The @ACCEPT PGINFO statement moves the page information for a given record to a specified location in program variable storage. Page information that is saved in this manner is available for subsequent direct access by using a @FIND/@OBTAIN DBKEY statement.

The dbkey radix portion of the page information can be used in interpreting a dbkey for display purposes and in formatting a dbkey from page and line numbers. The dbkey radix represents the number of bits within a dbkey value that are reserved for the line number of a record. By default, this value is 8, meaning that up to 255 records can be stored on a single page of the area. Given a dbkey, you can separate its associated page number by dividing the dbkey by 2 raised to the power of the dbkey radix. For example, if the dbkey radix is 4, you would divide the dbkey value by 2**4. The resulting value is the page number of the dbkey. To separate the line number, you would multiply the page number by 2 raised to the power of the dbkey radix and subtract this value from the dbkey value. The result would be the line number of the dbkey. The following two formulas can be used to calculate the page and line numbers from a dbkey value:

- Page-number = dbkey value / (2 ** dbkey radix)
- Line-number = dbkey value - (page-number * ( 2 ** dbkey radix))

### @ACCEPT PGINFO Syntax

```
►►─ @ACCEPT PGINFO=pg-info-v,REC=record-name ─────────────────────────►◄
```

## @ACCEPT PGINFO Parameters

**PGINFO=*pg-info-v***

Specifies the name of a four-byte field that is made up of two halfword fields. Identifies the location in variable storage that contains page information for the specified record. Upon successful completion of this statement, the first two bytes of the field contain the page group number and the last two bytes contain a value that may be used for interpreting dbkeys.

**REC=*record-name***

Specifies the record whose page information will be placed in the specified location.

## @ACCEPT PGINFO Example

The following example retrieves the page information for the DEPARTMENT record.

```
PAGEINFO DS    0F
PGROUP   DS    H
RADIX    DS    H

         @ACCEPT PGINFO=PAGEINFO,REC='DEPARTMENT'
```

## @ACCEPT PGINFO Status Codes

**Status Codes**

After completion of the @ACCEPT PGINFO statement, the ERROR-STATUS field in the IDMS communications block indicates the outcome of the operation.

The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**1508**

The specified record is not in the subschema. The program has probably invoked the wrong subschema.

## @ACCEPT PROC—moves the information block

The @ACCEPT PROC statement moves the 256-byte application program information block associated with a previously defined database procedure to a specified location in program variable storage. Information is placed in this block by a previously issued @BIND PROC statement (discussed later in this chapter). This information may have subsequently been updated by the procedure. The @ACCEPT PROC statement can be used by programs running under, but in a different partition from, the central version.

### @ACCEPT PROC Syntax

```
►►── @ACCEPT PROC=procedure-name ──────────────────────────────────────►
 ►── ,COMAREA=procedure-control-location ───────────────────────────────►◄
```

### @ACCEPT PROC Parameters

**PROC=*procedure-name***

Specifies the name of the database procedure whose application program information block is to be moved to program variable storage. *Procedure-name* must identify a fullword-aligned 8-byte literal.

**COMAREA=*procedure-control-location***

Specifies the fullword-aligned 256-byte field in program variable storage to which the application program information block is to be moved.

### @ACCEPT PROC Example

The following statement moves the application program information block used by the CHECKALL procedure to the location identified as CHECKIT in the application program's variable storage:

```
@ACCEPT PROC='CHECKALL',COMAREA=CHECKIT
```

## @ACCEPT PROC Status Codes

After completion of the @ACCEPT PROC function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation.

The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**1508**

The specified record is not in the subschema. The program has probably invoked the wrong subschema.

## @ACCEPT STATS—moves system runtime statistics

The @ACCEPT STATS statement moves system runtime statistics located in the program's IDMS statistics block to program variable storage. You can issue this statement any number of times during the execution of a run unit. For example, you might request database statistics after storing a variable-length record to determine whether the entire record was stored in one place or if fragments were placed in an overflow area.

The @ACCEPT STATS statement does not reset any of the statistics fields to zero. The IDMS statistics block fields are reset when you issue an @FINISH statement.

### @ACCEPT STATS Syntax

```
►►── @ACCEPT STATS=db-statistics ───────────────────────────────────►

►──┬──────────────────────────────────┬──────────────────────────►◄
   └ STATX=extended-db-statistics ────┘
```

### @ACCEPT STATS Parameters

**STATS=**

Moves system runtime statistics to a location in program variable storage identified by *db-statistics*.

***db-statistics***

Identifies an aligned, 100-byte field. The dictionary contains a record, DBSTATS, for the system runtime statistics. You can copy this record into program variable storage by coding the following statement:

```
@COPY IDMS,DBSTATS

DBSTATS  DS   0D
DATE2DAY DS   CL8   TODAY'S DATE
TIME2DAY DS   CL8   CURRENT TIME OF DAY
PAGESRED DS   F     PHYSICAL PAGES READ
PAGESWRT DS   F     PHYSICAL PAGES WRITTEN
PAGESQST DS   F     LOGICAL PAGES READ
CALCTARG DS   F     NO. CALC STORES ON TARGET PAGE
CALCOVFL DS   F     NO. CALC OVERFLOWS
VIATARGT DS   F     NO. VIA STORES ON OWNER PAGE
VIAOVRFL DS   F     NO. VIA OVERFLOWS
LINERQST DS   F     RECORDS (LINES) REQUESTED
CURRECDS DS   F     RECORDS CURRENT
IDMSCALL DS   F     NO. CALLS TO IDMSDBMS
FRAGMTST DS   F     NO. VAR-LENGTH FRAGMENTS STORED
RELORECS DS   F     NO. RECORDS RELOCATED
LOCKREQS DS   F     TOTAL NO. RECORD LOCKS HELD
SELECLOK DS   F     TOTAL NO. SELECT LOCKS HELD
UPDATLOC DS   F     TOTAL NO. EXCLUSIVE LOCKS HELD
RUNUNIT# DS   F     RUN-UNIT ID NUMBER
TASK#ID  DS   F     TASK ID NUMBER
LOCAL#ID DS   CL8   LOCAL ID NUMBER
         DS   CL8   RESERVED
```

The LOCAL#ID field consists of the 4-byte identifier of the interface in which the run unit originated (for example, BATC, DBDC, CICS) and a unique identifier (a fullword binary value) assigned to the run unit by that interface. For batch and CMS run units, this identifier specifies the internal machine time. For CICS run units, this identifier specifies the CICS transaction number assigned to the run unit. To display the originating interface identifier and the run-unit identifier for a program, you can move the LOCAL#ID field to a work field:

```
WRKLCID  DS    0D
WRKLCORG DC    CL4' '
WRKLCNUM DC    F'0'
```

Note: The DBSTATS record can be modified by your DBA to define two subordinate fields for the LOCAL#ID field.

**STATX=**

Moves extended system runtime statistics to a location in program variable storage identified by extended-db-statistics.

***extended-db-statistics***

Identifies an aligned, 100-byte field. The dictionary contains a record, DBSTATX, for the system runtime extended statistics. You can copy this record into program variable storage by coding the following statement:

```
                 @COPY IDMS,DBSTATS
                 DS   OD
        DBSTATX  DS    0CL100
        SR8SPLIT DS    FL4               Number of SR8 splits
        SR8SPAWN DS    FL4               Number of SR8 spawns
        SR8STORE DS    FL4               Number of SR8 STOREs
        SR8ERASE DS    FL4               Number of SR8 ERASEs
        SR7STORE DS    FL4               Number of SR7 STOREs
        SR7ERASE DS    FL4               Number of SR7 ERASEs
        BSRCHTOT DS    FL4               Number of binary searches
        LSRCHTOT DS    FL4               Number of levels searched
        ORPHADPT DS    FL4               Number of orphans adopted
        LSRCHBST DS    HL2               Best number of levels searched
        LSRCHWST DS    HL2               Worst number of levels searched
                 DS    CL60
```

Most of these counters are self-explanatory. The BSRCHTOT field indicates the total number of binary searches performed during the course of the run unit. LSRCHTOT indicates the total number of index levels searched.

The LSRCHBST and LSRCHWST fields describe the best and worst cases, respectively, for all random searches (such as generic searches) of all indexes. In other words, these statistics indicate the smallest and largest number of levels searched for a single request.

## @ACCEPT STATS Status Codes

After completion of the @ACCEPT STATS function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully

**1518**

The database statistics location has not been bound properly.

## @ACCEPT STATS Example

The following statements establish currency for the sets in which a new EMPLOYEE record will participate as a member, store the EMPLOYEE record, and move statistics regarding the stored EMPLOYEE record to the DBSTATS location in main storage:

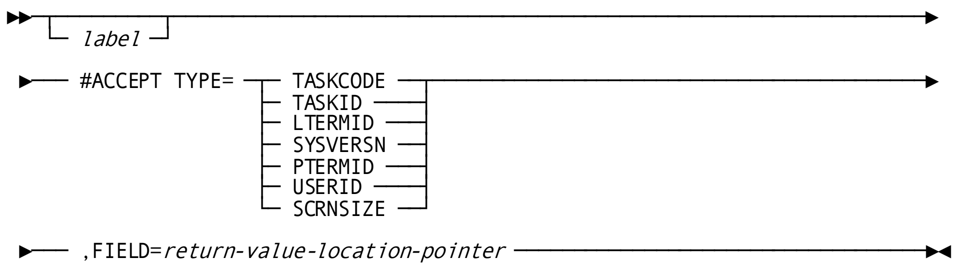
```
MVC      DEPTID,INDEPTID
@FIND    CALC,REC='DEPARTMENT'
MVC      OFFCODE,IOFFCODE
@FIND    CALC,REC='OFFICE'
@STORE   REC='EMPLOYEE'
@ACCEPT  STATS=DBSTATS
```

## #ACCEPT—retrieves system task-related information

The #ACCEPT statement retrieves the following system task-related information:

- Current task code
- Task identifier
- Logical terminal identifier
- Physical terminal identifier
- DC/UCF system version
- The ID of the user signed on to the task's logical terminal
- Physical terminal screen dimensions

### #ACCEPT Syntax



## #ACCEPT Parameters

**TYPE=**

Retrieves the requested information:

**TASKCODE**

Retrieves the 1- to 8-character code that invokes the current task.

**TASKID**

Retrieves the task identifier assigned by the system. The task identifier is a unique sequence number stored in a binary fullword numeric field. At system startup, the system sets the identifier to zero; each time a task is executed, the system increments the identifier by one.

**LTERMID**

Retrieves the 1- to 8-character identifier of the logical terminal associated with the current task.

**SYSVERSN**

Retrieves the version of the current DC/UCF system. The version number is an integer in the range 0 through 9999 stored in a binary halfword numeric field.

**PTERMID**

Retrieves the 1- to 8-character identifier of the physical terminal associated with the current task.

**USERID**

Retrieves the 32-character identifier of each user signed on to the logical terminal associated with the current task. If no user is signed on, the system returns blank.

**SCRNSIZE**

Retrieves the screen dimensions of the physical terminal associated with the current task. The screen size is returned in a field that is divided into two halfword fields: the first halfword contains the row, the second halfword contains the column. For example, a 24-line by 80-character screen is represented by a value of 24 in the first halfword and 80 in the second halfword. If the current task is not associated with a terminal, the system returns a null value of 0.

**FIELD=**

Specifies the location to which the system returns the requested task-related information.

***return-value-location***

A register that points to the field or the symbolic name of a user-defined field whose length is compatible with the length of the field containing the requested data.

## #ACCEPT Status Codes

After completion of the #ACCEPT statement, the value in register 15 indicates the outcome of the operation. The following is a list of the Register 15 value and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'04'**

An invalid return-value location address has been specified in the FIELD parameter.

**X'08'**

#ACCEPT TYPE=PTERM was specified but no PTERM exists.

## #ACCEPT Example

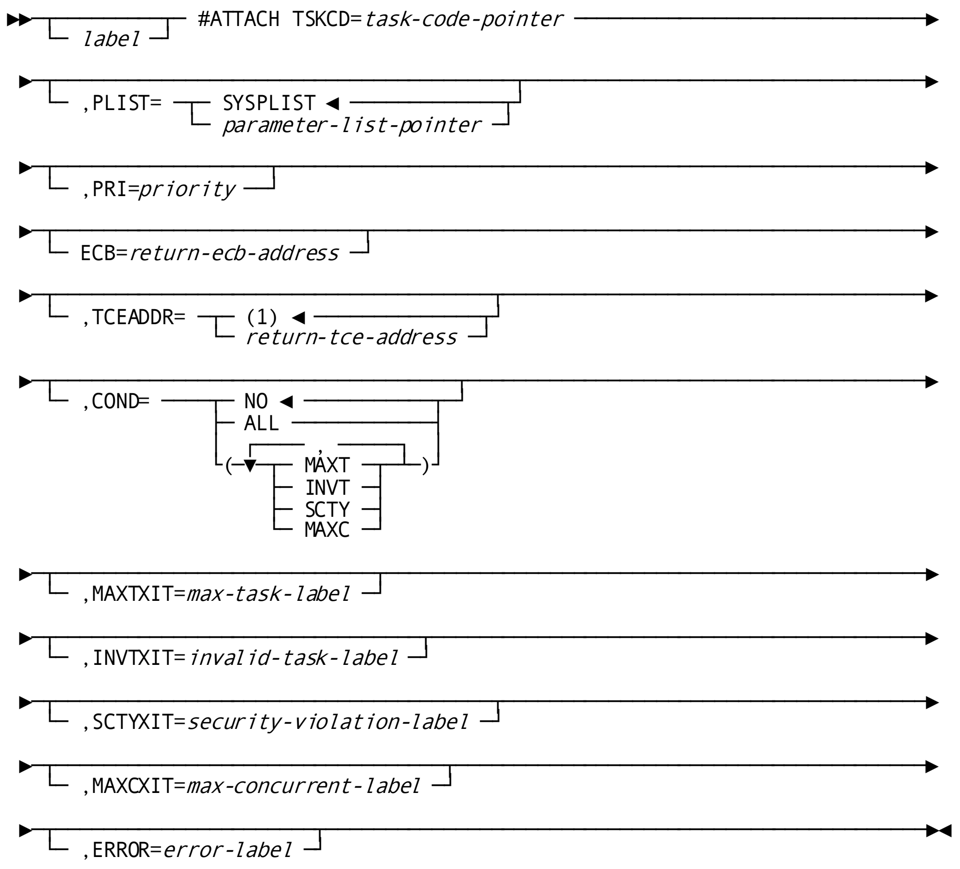
The following example of the #ACCEPT statement retrieves the user ID of each user signed on to the logical terminal associated with the current task. This information is placed into the field USERSL2, which is defined in the application program's variable storage.

```
#ACCEPT TYPE=USERID,FIELD=USERSL2
```

# #ATTACH—instructs the system to initiate a new task

The #ATTACH statement instructs the system to initiate a new task by acquiring the necessary task control elements (TCEs) and storage and by adding the task to its dispatching list. The issuing program retains processing control; the system simply initializes the attached task and gives it processor time according to its established priority. (Note that task code priorities established during system generation can be overridden by the #ATTACH or #CHAP statements.) The #ATTACH may optionally designate an ECB upon which initial execution of a new task will depend.

## #ATTACH Syntax



## #ATTACH Parameters

**TSKCD=**

Specifies the 1- to 8-character code of the task to be initiated.

***task-code***

A register pointing to a field that contains the task code, symbolic name of a user-defined field containing the task code, or the task-code literal enclosed in quotation marks. *Task-code* must have been defined either during system generation or dynamically by using the DCMT VARY DYNAMIC TASK command.

**PLIST=**

Specifies the location of the 5-fullword storage area that contains one or more parameters to be passed to the program receiving control.

**SYSPLIST**

(Default); the symbolic name of the storage area in which the system will build the #ATTACH parameter list.

***parameter-list***

A register that points to the area in which the system will build the #ATTACH parameter list or the symbolic name of that area.

**PRI=**

Specifies the dispatching priority of the attached task.

***priority***

A register containing the priority in the low-order byte or an absolute expression. Valid codes are 0 through 240; the default is the priority established during system generation for the specified task code, and the applicable terminal and user.

**ECB=**

Specifies the location to which the system will return the address of the event control block (ECB) for the initiated task. Use ECB to control execution of the attached task through the ECB; if ECB is not defined, the attached task will be set ready-to-run.

***return-ecb-address***

A register or the symbolic name of a fullword user-defined field.

**TCEADDR=(1)/*return-tce-address***

Specifies the location to which the system will return the address of the TCE for the initiated task. *return-tce-address*

A register or the symbolic name of a fullword user-defined field; the default is register 1.

**COND=**

Specifies whether this #ATTACH is conditional and under what conditions control should be returned to the issuing program.

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the attach cannot be serviced for one or more of the reasons listed below.

***condition***

Specifies under what conditions control is returned to the issuing program. Multiple *condition* values must be enclosed in parentheses and separated by commas.

**MAXT**

A maximum-task condition exists; that is, if the number of tasks specified as the maximum during system generation are currently active. If MAXT is not specified and a maximum-task condition exists, the attaching task will wait until the attach can be completed successfully.

**INVT**

The specified task code is invalid. If INVT is not specified and the specified task is not valid, the attaching task will be abended.

**SCTY**

The user signed on to the issuing task is denied access to the requested task because of a security violation. If SCTY is not specified and a security violation is detected, the attaching task will be abended.

**MAXC**

An attempt is being made to attach a task for which a MAXIMUM CONCURRENT value is specified in the system generation. The maximum number of occurrences of the task are already active. If MAXC is not specified and a maximum concurrent condition is detected, the attaching task will be abended.

**MAXTXIT=*max-task-label***

Specifies the symbolic name of a routine to which control is returned if the #ATTACH request cannot be serviced because of a maximum-task condition.

**INVTXIT=*invalid-task-label***

Specifies the symbolic name of a routine to which control is returned if the #ATTACH request cannot be serviced because the task code is invalid.

**SCTYXIT=*security-violation-label***

Specifies the symbolic name of a routine to which control is returned if the #ATTACH request cannot be serviced because of a security violation.

**MAXCXIT=*max-concurrent-label***

Specifies the symbolic name of a routine to which control is returned if the #ATTACH request cannot be serviced because of a maximum concurrent condition.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control is returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

## #ATTACH Status Codes

By default, the attach request is unconditional. Error conditions that can occur are described below:

- A maximum-task condition will result in a delay until another task terminates. The maximum number of active tasks is set during system generation.
- Any abnormal condition will result in an abend. Conditions in this category include:
  - Invalid task code specified

- The user signed on to the issuing task is denied access to the requested new task because of a security violation

The issuing program can request return of control to avoid a delay or an abend by using the COND parameter.

After completion of the #ATTACH request, the value returned to register 15 indicates the outcome of the operation. The following is a list of the Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'04'**

The request cannot be serviced because the maximum number of tasks have already been attached; no new tasks can currently be attached.

**X'08'**

The request cannot be serviced due to an invalid task code.

**X'0C'**

The request cannot be serviced due to a task security violation.

**X'14'**

The task cannot be attached because the maximum concurrent task limit (for that task code) has been exceeded.

Additionally, the values in two user-defined registers or fullwords contain information:

- Register **n** contains the address of the ECB of the initiated task is found in the register or fullword assigned by the ECB= parameter. If the task has been set ready-to-run, as described above for the ECB parameter, this register is not set.
- Register **m** contains the address of the TCE of the initiated task is placed in the register or fullword assigned by the TCEADDR parameter.

## #ATTACH Example

**Example**

The example shown below of the #ATTACH statement performs the following functions:

- Task MENU3 is initiated and added to the system dispatching list with a priority setting of 150.
- WPLIST is the work area where the system builds the parameter list.
- Register 3 is designated to receive the address of the ECB for the initiated task from the system.

- Control will be returned to the exit routine MENERR if the attach cannot be serviced for any of the optional conditions associated with the COND parameter.

```
#ATTACH TSKCD='MENU3',PLIST=WPLIST,PRI=150,ECB=(3),COND=ALL,   *
      ERROR=MENERR
```

## @BIND PROC—establishes communication

The @BIND PROC statement establishes communication between a program and a DBA-written database procedure (for example, a security routine). You should use this statement only when the application program is required to pass more information to the procedure than is provided by CA IDMS/DB itself. Such instances are unusual; in most cases, you will not be aware of which procedures gain control before or after the various DML functions.

### @BIND PROC Syntax

```
►►─── @BIND PROC=procedure-name ──────────────────────────────────────►
 ►─── ,COMAREA=procedure-control-location ────────────────────────────►◄
```

### IDMSDB--@BIND PROC Parameters

**PROC=**

Establishes addressability for the specified database procedure in program variable storage.

***procedure-name***

Must refer to an 8-character literal aligned on a fullword boundary.

**COMAREA=**

Identifies the program storage location to which the named procedure will be bound.

***procedure-control-location***

Must identify a 256-byte (fixed-length) area.

A program running in a different partition than the central version may need to pass certain information to the database procedure. When the DBMS invokes the database procedure, this information is copied from the program storage area, identified by *procedure-control-location,* into the CA IDMS/DB application program information block. The information passed is the information in the program storage location at the time the BIND PROC was performed; it is not the information in the program's storage at the time of the procedure call.

## @BIND PROC Status Codes

After completion of the BIND PROC function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**1400**

The @BIND PROC statement cannot be recognized. This code usually indicates that the IDMS communications block (SUBSCHEMA-CTRL) is not aligned on a fullword boundary.

**1418**

The procedure has been bound improperly to location 0.

**1472**

The memory available is insufficient to load dynamically the database procedure.

## @BIND PROC Example

The following example of the @BIND PROC statement specifies that register 8 contains the name of the database procedure to receive information from the program's variable storage area labeled DBPASS:

```
@BIND PROC=(R8),COMAREA=DBPASS
```

## @BIND REC—establishes addressability for a record

The @BIND REC statement establishes addressability for a record in variable storage. In most cases, you do not need to issue individual @BIND REC statements, since the necessary statements typically are generated as a group by the @COPY IDMS,SUBSCHEMA-BINDS statement see Assembler DML Coding Considerations (see page 399). However, you can issue these statements separately as necessary.

For example, since the @COPY IDMS,SUBSCHEMA-BINDS statement does not verify that each record is bound successfully, you may wish to issue an @BIND REC statement for each record and to check the ERRSTAT field in the IDMS communications block after each @BIND REC statement. You can also issue separate @BIND REC statements to bind several records to the same storage location. In any case, you must establish addressability for each subschema record to be used by the program.

Note: If program registration is in effect, you should code a @COPY IDMS,SUBSCHEMA-BINDS statement to properly assign the programs to the subschema control block. Otherwise your program must explicitly initialize the PGNAME field in the IDMS communications block before the @BIND SUBSCHEMA and @BIND REC statements are executed.

## @BIND REC Syntax

```
►►── @BIND REC=record-name ──────────────────────────────────────────────►

 ►── ,IOAREA=record-location ────────────────────────────────────────────►◄
```

## @BIND REC Parameters

**REC=*record-name***

Binds the named record to a location in variable storage that corresponds to the record description copied into the program. *Record-name* must specify a record included in the subschema.

**IOAREA=*record-location***

Identifies the specific location in the program's variable storage to which the record is bound.

Note: Use care with this option because source-object mismapping can result from improper use. In cases where the description of a given CA IDMS/DB record is present in more than one location in variable storage, you must ensure that the proper record description is bound at the proper time.

## @BIND REC Status Codes

After completion of the @BIND REC function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**1408**

The name record is not in the subschema. The program has probably invoked the wrong subschema.

**1418**

The record has been bound improperly to location 0.

## @BIND REC Example

The following example of the @BIND REC statement establishes addressability for the database record EMPLOYEE to the program's variable storage area labeled EMPLOYE:

```
@BIND REC='EMPLOYEE',IOAREA=EMPLOYE
```

## @BIND SUBSCH—helps the run unit

The @BIND SUBSCH statement performs the following:

- Signs on the run unit to the DBMS
- Identifies the location of optional user-specified IDMS and LRC communication blocks to the DBMS
- Names the subschema to be loaded for the run unit
- Names the Distributed Database System (DDS) node under which the run unit will execute
- Identifies the database to be accessed

You must code the @BIND SUBSCH statement as the first DML statement in the program that is passed to CA IDMS/DB at execution time. This statement must be first both logically and physically; you cannot branch to @BIND SUBSCH.

In most cases, specific designation of @BIND SUBSCH within an application program is not necessary since the @COPY IDMS,SUBSCHEMA-BINDS statement (see @COPY IDMS (see page 411)) automatically invokes the necessary @BIND statements.

Note: If program registration is in effect, the @COPY IDMS,SUBSCHEMA-BINDS statement is required to properly assign the programs to the subschema control block. Individual @BIND SUBSCH and @BIND REC statements should not be used if program registration was enabled during system generation.

## @BIND SUBSCH Syntax

```
►►── @BIND SUBSCH=subschema-name ──────────────────────────────────────►

►─┬─────────────────────────────────┬──────────────────────────────────►
  └─ ,PGMNAME=program-name ─┘

►─┬─────────────────────────────────┬──────────────────────────────────►
  └─ ,LRC=lr-control-block-location ─┘

►─┬─────────────────────────────────┬──────────────────────────────────►
  └─ ,LRSIZ=lr-control-block-size ─┘

►─┬─────────────────────────────────┬──────────────────────────────────►
  └─ ,DBNAME=database-name-pointer ─┘

►─┬─────────────────────────────────┬──────────────────────────────────►
  └─ ,DBNODE=nodename-pointer ─┘

►─┬─────────────────────────────────┬──────────────────────────────────►
  └─ ,DICTNAM=dictionary-name-pointer ─┘

►─┬─────────────────────────────────┬──────────────────────────────────►◄
  └─ ,DICTNOD=dictionary-nodename-pointer ─┘
```

## @BIND SUBSCH Parameters

**SUBSCH=**

Signs on the application program to CA IDMS/DB.

***subschema-name***

Identifies the subschema in use. The run unit uses the standard IDMS communications block brought previously into the program by compiler-directive statements.

**PGMNAME=*program-name***

Identifies the user program.

**LRC=*lrc-block-location***

Identifies the address of a logical-record request control (LRC) block other than that brought into the program by the DML precompiler. The definition of this user-specified subschema control area must be consistent with the standard SSLRCTL block as normally invoked and used.

**LRSIZ=*lrc-block-size***

Specifies the size of that portion of the LRC block that contains information about the request's WHERE clause. *Lrc-block-size* defaults to 576 bytes. For the algorithm for calculating *lrc-block-size,* see @COPY IDMS (see page 411).

**DBNAME=**

Identifies the database to be accessed by the program. If this parameter is specified, *database-name* may be overridden by IDMSOPTI module or SYSCTL file specifications.

***database-name***

Must specify a register that points to the name of the database, a 1- to 8-character field, or a quoted literal.

**DBNODE=**

Optionally names the node that will service database requests issued by the program. If this parameter is specified, *nodename* may or may not be overridden by IDMSOPTI module or SYSCTL file specifications (z/OS and OS/390 only).

***nodename-pointer***

Must be a register that points to the name of the node, a 1- to 8-character field, or a quoted literal.

**DICTNAM=**

The dictionary that contains the subschema.

***dictionary-name-pointer***

Either a register that points to the field that contains the dictionary name or a quoted literal.

**DICTNOD=**

The dictionary node that contains the subschema.

***dictionary-nodename-pointer***

Either a register that points to the field that contains the name of the dictionary or a quoted literal.

## @BIND SUBSCH Status Codes

**Status Codes**

After completion of the @BIND SUBSCH function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**1400**

The @BIND SUBSCH statement cannot be recognized. This code usually indicates that the IDMS communications block (SUBSCHEMA-CTRL) is not aligned on a fullword boundary.

**1467**

The subschema invoked does not match the subschema object tables.

**1469**

The run unit is not bound to the DBMS. This code indicates that the central version is not active or is not accepting new run units, or that the run unit's connection to the central version is broken due to timeout or other factors, as noted on the CV log.

**1470**

The journal file will not open (local mode only); under OS, the most probable cause is that a DD statement for the journal file is missing in the JCL.

**1472**

The available memory is insufficient to dynamically load a subschema or database procedure.

**1473**

The central version is not accepting new run units.

**1474**

The subschema was not found in the dictionary load area or in the load library.

**1477**

The run unit has been bound previously.

**1480**

The node specified in the NODENAME clause either is not active or has been disabled from the communications network.

**1481**

The database specified in the CA IDMS network clause is not known to CA IDMS/DB.

**1482**

The named subschema is not allowed under the database specified in the DBNAME clause.

**1483**

The available memory is insufficient to allocate native VSAM work areas.

## @BIND SUBSCH Example

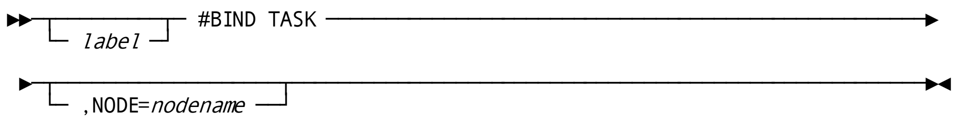
The following example of the @BIND SUBSCH statement signs on the application program EMPUPD to CA IDMS/DB, identifies the subschema EMPSS01, and identifies the address in program variable storage of the user-specified communications block EMPCTRL:

```
@BIND SUBSCH='EMPSS01',SCB=EMPCTRL,PGMNAME='EMPUPD'
```

## #BIND TASK—initiates a DC/UCF task

The #BIND TASK statement initiates a DC/UCF task when the operating mode is DC-BATCH. This statement establishes communication with the system and, if accessing DC/UCF queues and printers, allocates a packet-data movement buffer to contain the queue or printer data. Once a task is started, the program can issue any number of consecutive BIND-READY-FINISH sequences.

### #BIND TASK Syntax



### #BIND TASK Parameters

**,NODE=**

Specifies the 1- to 8-character name of the node to which the task will be bound.

***nodename***

Either the symbolic name of a user-defined field that contains the nodename or the nodename itself enclosed in quotation marks. The specified nodename must match the node named in the *nodename* statement at system generation.

### #BIND TASK Status Codes

After completion of the BIND TASK function, the status field in the IDMS communications block indicates the outcome of the operation.

## #BIND TASK Example

The following statement establishes communication with a DC/UCF system:

```
#BIND TASK.
```

## #CHAP—changes the dispatching priority

The #CHAP statement changes the dispatching priority of the issuing task. #CHAP does not relinquish control to another task and cannot be used to alter the priority of other tasks.

### #CHAP Syntax

```
►►─┬───────────┬─ #CHAP PRI=priority ─┬──────────────────────────────┬───────►◄
   └─ label ───┘                      └ACTION=─┬─ SET ◄ ─────┤
                                               ├─ ADD ───────┤
                                               └─ SUBTRACT ──┘
```

### #CHAP Parameters

**PRI=**

Specifies a new dispatching priority for the issuing task.

***priority***

A register that contains the priority in the low-order byte, the symbolic name of a user-defined field that contains the priority, or an absolute expression in the range 0 through 240.

**ACTION=**

Specifies the meaning of the *priority* value using one of the following options:

**SET**

The priority is an absolute value. SET is the default.

**ADD**

The priority is a relative value and is added to the task's current priority.

**SUBTRACT**

The priority is a relative value and is subtracted from the task's current priority.

## #CHAP Status Codes

The change-priority request is unconditional; any return code other than X'00' will result in an abend of the task.

## #CHAP Example

The following example of the #CHAP statement changes the dispatching priority to one less than the current dispatching priority:

```
#CHAP PRI=1,ACTION=SUBTRACT
```

## @COMMIT—commits changes made to the database

The @COMMIT statement commits changes made to the database by an individual run unit. @COMMIT simulates an @FINISH-@BIND-@READY sequence without relinquishing control of database resources.

If the run unit is sharing its transaction with another database session, the run unit's changes may not be committed at the time the @COMMIT statement is executed.

**Note:** For more information about the impact of transaction sharing, see the *Navigational DML Programming Guide*.

**Currency**

Specifying @COMMIT ALL sets all currencies to null.

### @COMMIT Syntax

```
►►── @COMMIT ─┬──────┬──────────────────────────────────────────►◄
              └ ,ALL ┘
```

### @COMMIT Parameters

**ALL**

Releases record locks and sets all currencies to null.

### @COMMIT Status Codes

The only acceptable status code returned for an @COMMIT function is 0000.

## #COMMIT—commits changes made to the database

The #COMMIT statement commits changes made to the database through an individual run unit or through all database sessions associated with a task. A task-level commit also commits all changes made in conjunction with scratch, queue and print activity.

All locks held on current records except for select locks are released. #COMMIT simulates an #FINISH/@BIND/@READY sequence but does not relinquish control of database resources.

If the commit applies to an individual run unit and the run unit is sharing its transaction with another database session, the run unit's changes may not be committed at the time the #COMMIT statement is executed.

**Note:** For more information about the impact of transaction sharing, see the *Navigational DML Programming Guide*.

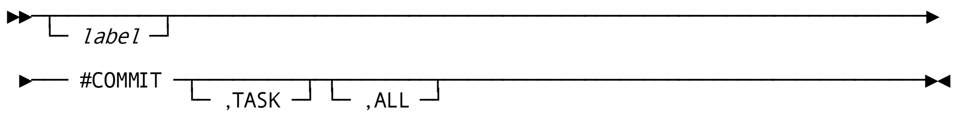
Run units (and SQL sessions) impacted by the COMMIT statement remain active after the operation is complete.

The #COMMIT statement is used in both the navigational and logical record facility environments. The #COMMIT TASK statement is also used in an SQL programming environment.

**Currency**

Specifying #COMMIT ALL sets all currencies to null.

### #COMMIT Syntax



### #COMMIT Parameters

### #COMMIT Status Codes

## @CONNECT—establishes a record occurrence

The @CONNECT statement establishes a record occurrence as a member of a set occurrence. The specified record must be defined as an optional automatic, optional manual, or mandatory manual member of the set.

**Note: Native VSAM users**—The @CONNECT statement is not valid since all sets in native VSAM data sets must be defined as mandatory automatic.

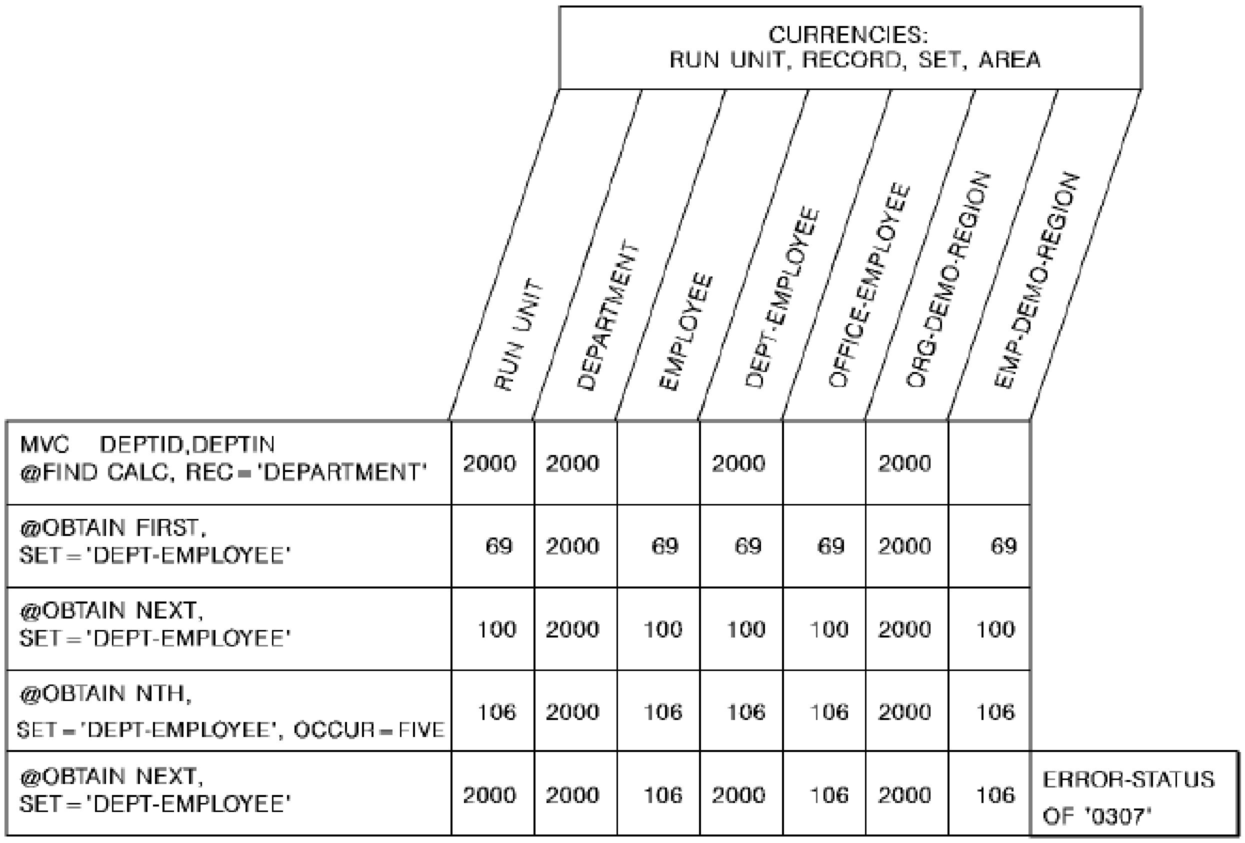
**Currency**

Before execution of the @CONNECT statement, you must satisfy the following conditions:

- All areas affected either explicitly or implicitly by the @CONNECT statement must be readied in one of the update usage modes (see @READY (see page 308) later in this chapter).
- The named record must be established as current of its record type.
- The appropriate occurrence of the set into which the named record will be connected must be established. The current record of set determines the set occurrence. If the set order is NEXT or PRIOR, this record determines the position of the new member within the set.

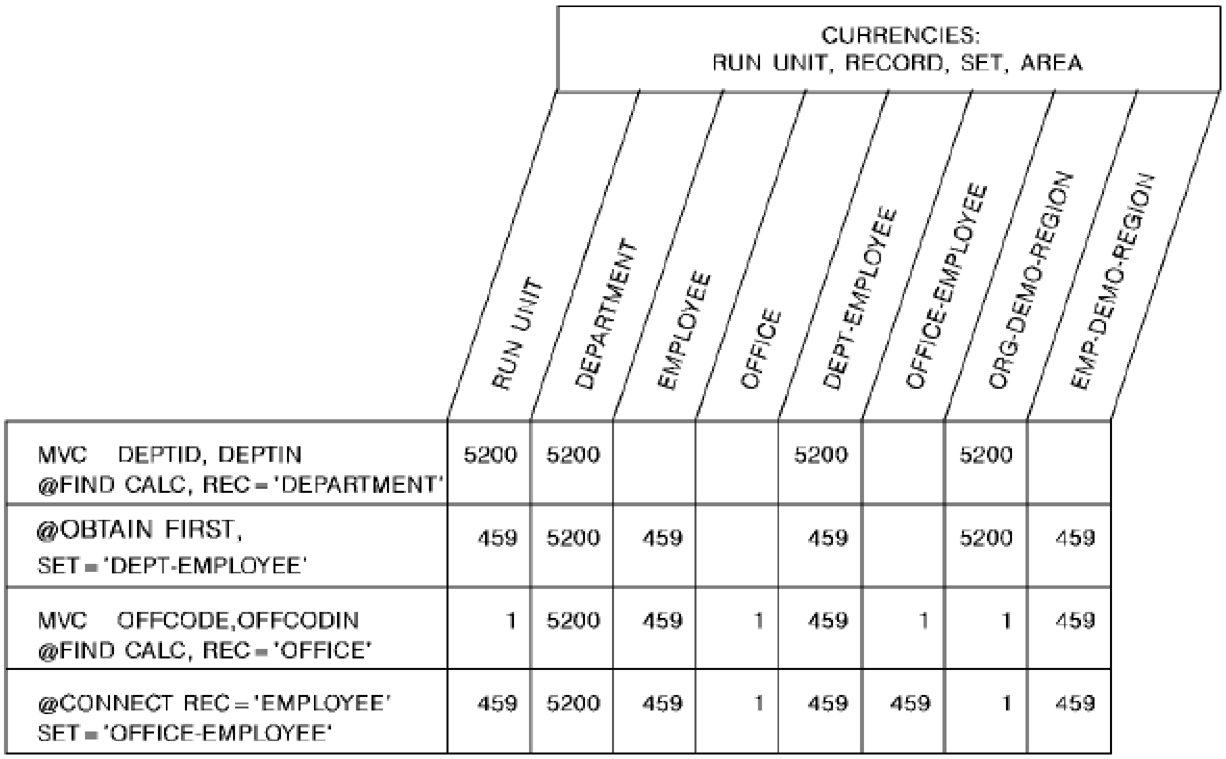
Following successful execution of the @CONNECT statement, the named record is current of run unit, its record type, its area, and all sets in which it currently participates. The following figure illustrates the steps required to connect an EMPLOYEE record to an occurrence of the OFFICE-EMPLOYEE set.

To connect EMPLOYEE 459 to the OFFICE 1 occurrence of the OFFICE- EMPLOYEE set, you must establish EMPLOYEE 459 as current of record type, locate the proper occurrence of the OFFICE record, and connect EMPLOYEE 459 to the OFFICE-EMPLOYEE set.



CURRENCIES:
RUN UNIT, RECORD, SET, AREA

| | RUN UNIT | DEPARTMENT | EMPLOYEE | DEPT-EMPLOYEE | OFFICE-EMPLOYEE | ORG-DEMO-REGION | EMP-DEMO-REGION | |
|---|---|---|---|---|---|---|---|---|
| MVC DEPTID,DEPTIN<br>@FIND CALC, REC='DEPARTMENT' | 2000 | 2000 | | 2000 | | 2000 | | |
| @OBTAIN FIRST,<br>SET='DEPT-EMPLOYEE' | 69 | 2000 | 69 | 69 | 69 | 2000 | 69 | |
| @OBTAIN NEXT,<br>SET='DEPT-EMPLOYEE' | 100 | 2000 | 100 | 100 | 100 | 2000 | 100 | |
| @OBTAIN NTH,<br>SET='DEPT-EMPLOYEE', OCCUR=FIVE | 106 | 2000 | 106 | 106 | 106 | 2000 | 106 | |
| @OBTAIN NEXT,<br>SET='DEPT-EMPLOYEE' | 2000 | 2000 | 106 | 2000 | 106 | 2000 | 106 | ERROR-STATUS OF '0307' |



| | CURRENCIES: RUN UNIT, RECORD, SET, AREA | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RUN UNIT | DEPARTMENT | EMPLOYEE | OFFICE | DEPT-EMPLOYEE | OFFICE-EMPLOYEE | ORG-DEMO-REGION | EMP-DEMO-REGION |
| MVC DEPTID, DEPTIN @FIND CALC, REC='DEPARTMENT' | 5200 | 5200 | | | 5200 | | 5200 | |
| @OBTAIN FIRST, SET='DEPT-EMPLOYEE' | 459 | 5200 | 459 | | 459 | | 5200 | 459 |
| MVC OFFCODE,OFFCODIN @FIND CALC, REC='OFFICE' | 1 | 5200 | 459 | 1 | 459 | 1 | 1 | 459 |
| @CONNECT REC='EMPLOYEE' SET='OFFICE-EMPLOYEE' | 459 | 5200 | 459 | 1 | 459 | 459 | 1 | 459 |

## @CONNECT Syntax

```
►►─── @CONNECT REC=record-name ───────────────────────────────────►
►─── ,SET=set-name ──────────────────────────────────────────────►◄
```

## @CONNECT Parameters

**REC=**

Connects the current occurrence of the named record to the current occurrence of the specified set.

***record-name***

Must be a record included in the subschema and must be defined as an optional automatic, optional manual, or mandatory manual member of the set to which it is being connected. *Record-name* may be specified as a register, a user-defined variable data field, or a user-supplied value in quotation marks.

**SET=**

Specifies the set to which the member record is to be connected.

***set-name***

Must specify a set included in the subschema. The record is connected to the set in accordance with the ordering rules defined for that set in the schema. *Set-name* may be specified as a register, a user-defined variable data field, or a user-supplied value in quotation marks.

## @CONNECT Status Codes

**Status Codes**

After completion of the @CONNECT function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**0705**

The @CONNECT would violate a duplicates-not-allowed option.

**0706**

Currency has not been established for the named record or set.

**0708**

The specified record is not in the subschema. The program has probably invoked the wrong subschema.

**0709**

The named record's area has not been readied in one of the three update usage modes.

**0710**

The subschema specifies an access restriction that prohibits connecting the named record in the named set.

**0714**

The @CONNECT statement cannot be executed because the named record has been defined as a mandatory automatic member of the set.

**0716**

The record cannot be connected to a set in which it is already a member.

**0721**

An area other than the area of the named record has been readied with an incorrect usage mode.

**0725**

Currency has not been established for the named set type.

### @CONNECT Example

The following statements connect an EMPLOYEE record from the DEPT-EMPLOYEE set to the OFFICE-EMPLOYEE set as a new member.

```
MVC      DEPTID,=C'5200'
@FIND    CALC,REC='DEPARTMENT'
@OBTAIN  FIRST,SET='DEPT-EMPLOYEE'
MVI      OFFCODE,C'1'
@FIND CALC,REC='OFFICE'
@CONNECT REC='EMPLOYEE',SET='OFFICE-EMPLOYEE'
```

## #DELETE—notifies the DC/UCF system

The #DELETE statement notifies the DC/UCF system that the issuing task has finished using a module from the program pool. This module is identified by the program name or entry-point address that was previously specified by the #LOAD request that placed the module into the program pool. If your site uses multiple dictionaries you can specify either the dictionary in which the program resides or the node that controls the dictionary. Other options for a multiple dictionary environment include specifying a parameter list and a program version number for the program you are requesting to delete.

#DELETE does not physically delete the module from the program pool unless the program has been defined as NONREUSABLE. Rather, it decrements the in-use count maintained by the DC/UCF system. An in-use count of 0 indicates to the system that the space occupied by the module can be reused.

## #DELETE Syntax

```
►►─┬───────────┬──────────────────────────────────────────────────────────►
   └─ label ───┘

►──── #DELETE─┬─ PGM=program-name-pointer ───┬───────────────────────────►
              └─ EPADDR=entry-point-address ─┘

►─┬──────────────────────────────────────────────┬───────────────────────►
  └─ ,PLIST= ─┬─ SYSPLIST ◄ ───────────────────┬─┘
              └─ parameter-list-pointer ───────┘

►─┬───────────────────────────────────┬──────────────────────────────────►
  └─ .DICTNOD=nodename-pointer ───────┘

►─┬───────────────────────────────────────────┬──────────────────────────►◄
  └─ ,DICTNAM=dictionary-name-pointer ────────┘
```

## #DELETE Parameters

**PGM=**

Specifies the 1- to 8-character name of the module being released from use.

***program-name-pointer***

A register that points to a field containing the program name, the symbolic name of a user-defined field containing the program name, or the program-name literal enclosed in quotation marks.

**EPADDR=**

Specifies the entry-point address of the module being released from use. This address was returned to the issuing program when the module was originally loaded.

***entry-point-address***

Either a register or the symbolic name of a fullword user-defined field containing the entry-point address.

**PLIST=**

Specifies the location of the storage area the system uses to build the parameter list. The PLIST parameter is required only if the DICTNAM or DICTNOD parameters are specified.

**SYSPLIST**

The symbolic name of the storage area in which the system will build the #DELETE parameter list.

***parameter-list-pointer***

A register that points to the area in which the system will build the #DELETE parameter list or the symbolic name of that area.

**DICTNOD=**

Identifies the node that controls the dictionary in which the program resides.

***nodename-pointer***

A register that points to a field that contains the name of the node, the symbolic name of a user-defined field containing the name of the node, or the nodename literal enclosed in quotation marks.

**DICTNAM=**

Identifies the dictionary in which the named program resides.

***dictionary-name-pointer***

A register that points to a field containing the dictionary name, the symbolic name of a user-defined field containing the dictionary name, or the dictionary name literal enclosed in quotation marks.

Note: The DICTNOD or DICTNAM parameters must correspond to those specified on a previously issued #LOAD statement. If either DICTNOD or DICTNAM or both are specified, the PLIST parameter must be included.

## #DELETE Status Codes

The #DELETE request is unconditional; any error detected during execution will result in an abend of the issuing task.

## #DELETE Example

The following example of the #DELETE statement notifies the system that the program or module whose entry-point address is contained in register 5 is no longer needed by the issuing task. The system can reuse this area in the program pool if space is needed.

```
#DELETE EPADDR=(R5)
```

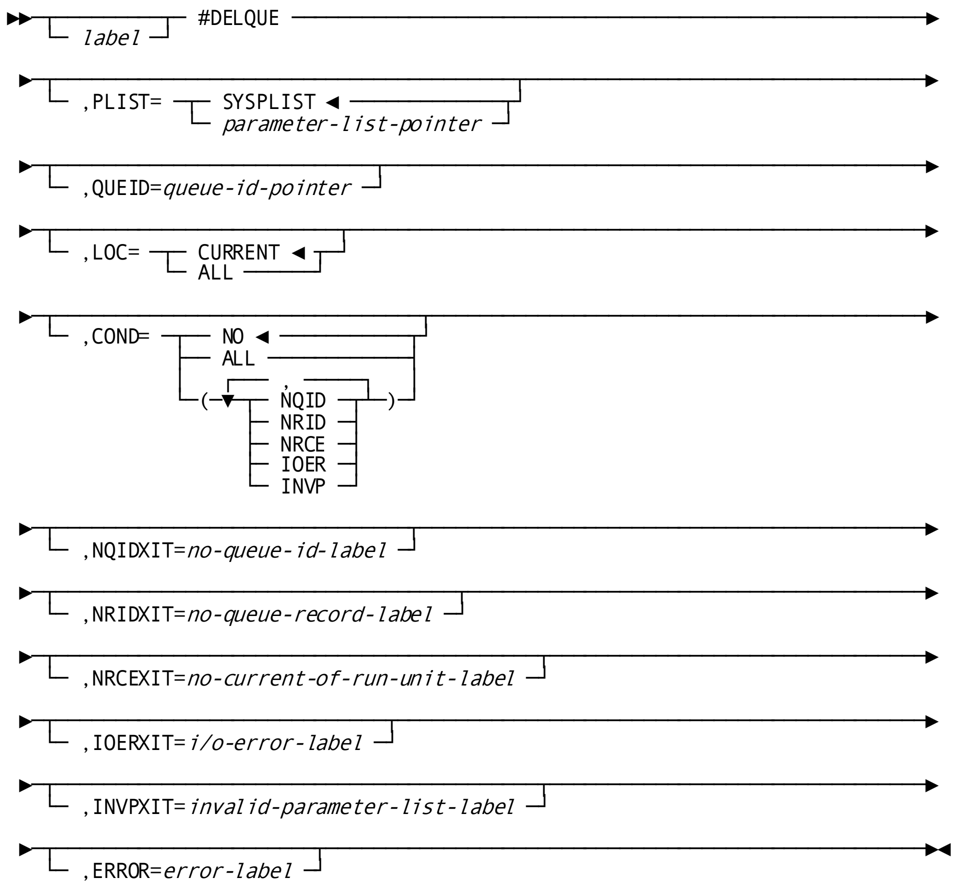
The example shown below illustrates the use of the #LOAD and the #DELETE statements in a multiple dictionary environment. After execution of the #DELETE statement the area in the program pool in which EMPMENU resides is released and can be reused by issuing a new #LOAD request statement.

```
#LOAD PGM='EMPMENU'


.

.

.

#DELETE PGM='EMPMENU'
```

## #DELQUE—deletes all or part of a queue

The #DELQUE statement deletes all or part of a queue. If only one queue record is deleted, the system maintains currency within the queue by using the next and prior pointers of the queue record.

### #DELQUE Syntax



### #DELQUE Parameters

**PLIST=**

Specifies the location of the 2-fullword storage area in which the system will build the #DELQUE parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system will build the #DELQUE parameter list.

***parameter-list-pointer***

Either a register that points to the area or the symbolic name of the area.

**QUEID=**

Specifies the 1- to 16-character queue header ID associated with the queue or queue record to be deleted.

***queue-id-pointer***

A register that points to a field containing the id, the symbolic name of a user-defined field containing the ID, or the ID literal enclosed in quotation marks. If the queue header ID is not specified, a blank ID is assumed.

**LOC=**

Indicates the portion of the queue to be deleted.

**CURRENT**

(Default); deletes only the current record of the queue associated with the requesting task.

**ALL**

Deletes all records in the queue and the queue header id.

**COND=**

Specifies whether this #DELQUE is conditional and under what conditions control should be returned to the issuing program:

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the delete cannot be serviced for one or more of the reasons listed below.

***condition***

Specifies under what conditions control should be returned to the issuing program. Multiple values must be enclosed in parentheses and separated by commas. *Condition* options are as follows:

- **NQID**—The queue header record cannot be found.

- **NRID**—LOC=CURRENT has been specified and the record previously established as current of queue cannot be found.
- **NRCE**—LOC=CURRENT has been specified and no resource control element (RCE) exists for the current record; that is, no record has been established as current of queue.
- **IOER**—An I/O error occurs while processing the delete.
- **INVP**—The parameter list built for the #DELQUE is invalid.

**NQIDXIT=*no-queue-id-label***

Specifies the symbolic name of the routine to which control should be returned if the #DELQUE request cannot be serviced because the queue header record cannot be found.

**NRIDXIT=*no-queue-record-label***

Specifies the symbolic name of the routine to which control should be returned if the #DELQUE request cannot be serviced because the record previously established as current of queue cannot be found.

**NRCEXIT=*no-current-of-run-unit-label***

Specifies the symbolic name of the routine to which control should be returned if the #DELQUE request cannot be serviced because no current of queue has been established (no resource control element exists for the queue record).

**IOERXIT=*i/o-error-label***

Specifies the symbolic name of the routine to which control should be returned if the #DELQUE request cannot be serviced because of an I/O error while processing the delete.

**INVPXIT=*invalid-parameter-list-label***

Specifies the symbolic name of the routine to which control should be returned if the #DELQUE request cannot be serviced because of an invalid parameter in the parameter list.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

## #DELQUE Status Codes

By default, the #DELQUE request is unconditional; any runtime error will result in an abend of the issuing task. To avoid an abend, you can request return of control to the issuing program by using the COND parameter.

After completion of the #DELQUE function, the value in register 15 indicates the outcome of the operation. The following is a list of the Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'04'**

The request cannot be serviced due to an invalid parameter list.

**X'08'**

The request cannot be serviced because the requested queue header record (identified by QUEID) cannot be found.

**X'0C'**

The request cannot be serviced because the requested queue record cannot be found.

**X'10'**

The request for a #DELQUE LOC=CURRENT cannot be serviced because no resource control element (RCE) exists for the queue record, indicating that currency has not been established.

**X'1C'**

A database error occurred during queue processing. A common cause is a DBKEY deadlock. For a PUT QUEUE operation, this code can also mean that the queue upper limit has been reached.

If a database error has occurred, there are usually be other messages in the CA-IDMS/DC/UCF log indicating a problem encountered in RHDCRUAL, the internal Run Unit Manager. If a deadlock has occurred, messages DC001000 and DC001002 are also produced.

If an I/O error occurs while processing a #DELQUE request, the system will return the address of the IDMS communications block to register 1. In this situation, you can check the status code in the ERRSTAT field (for more information, see ERRSTAT Field and Codes (see page 41)).

## #DELQUE Example

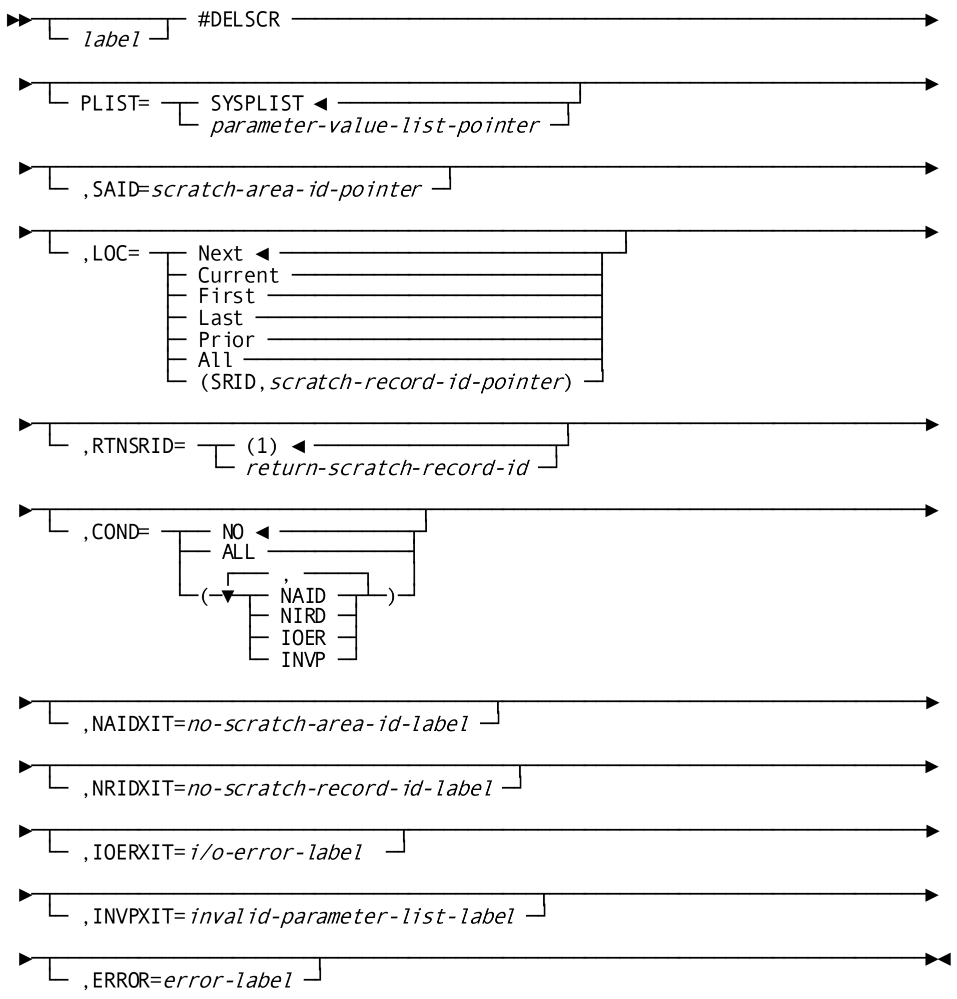
The following example of the #DELQUE statement deletes an entire queue area. The address of the queue header ID is contained in register 4. In the event of an I/O error, control will be returned to the ERROR5 routine of the issuing program; other error conditions will result in an abend of the issuing task.

```
#DELQUE QUEID=(R4),LOC=ALL,COND=IOER,IOERXIT=ERROR5
```

## #DELSCR—deletes scratch records

The #DELSCR statement deletes one or all scratch records in a scratch area.

### #DELSCR Syntax


label
#DELSCR
PLIST=
SYSPLIST
parameter-value-list-pointer
,SAID=scratch-area-id-pointer
,LOC=
Next
Current
First
Last
Prior
All
(SRID,scratch-record-id-pointer)
,RTNSRID=
(1)
return-scratch-record-id
,COND=
NO
ALL
,
NAID
NIRD
IOER
INVP
,NAIDXIT=no-scratch-area-id-label
,NRIDXIT=no-scratch-record-id-label
,IOERXIT=i/o-error-label
,INVPXIT=invalid-parameter-list-label
,ERROR=error-label

## #DELSCR Parameters

**PLIST=**

Specifies the location of the 3-fullword storage area in which the system will build the #DELSCR parameter list.

**SYSPLIST**

(Default); the symbolic name of the storage area in which the system will build the #DELSCR parameter list.

***parameter-list-pointer***

A register that points to the area or the symbolic name of the area in which the system will build the #DELSCR parameter list.

**SAID=**

Specifies the 1- to 8-character ID of the scratch area associated with the scratch record being deleted.

***scratch-area-id-pointer***

A register that points to a field containing the id, the symbolic name of a user-defined field containing the ID, or the ID literal enclosed in quotation marks. If the SAID parameter is not specified, a scratch area ID of 8 blanks is assumed.

**LOC=**

Specifies the scratch record to be deleted from the area associated with the specified scratch record id.

**NEXT**

(Default); deletes the next record. If currency has not been established, NEXT is equivalent to FIRST.

**CURRENT**

Deletes the current record, that record most recently referenced by another scratch function.

**FIRST**

Deletes the first record. (Records are always stored in ascending order by scratch record ID.)

**LAST**

Deletes the last record.

**PRIOR**

Deletes the prior record. If currency has not been established, PRIOR is equivalent to LAST.

**ALL**

Deletes all records.

**(SRID,*scratch-record-id*)**

Deletes the record identified by *scratch-record-id*. *Scratch-record-id* is a register that points to the 4-byte scratch record id, the symbolic name of a user-defined field containing the id, or an absolute expression of the id.

**RTNSRID=(1)/**

Specifies the location to which the system will return the scratch record ID of the last record deleted with a #DELSCR function.

***return-scratch-record-id***

A register or the symbolic name of a fullword user-defined field to which the system will return the scratch record ID of the last record deleted, the default is register 1.

**COND=**

Specifies whether this #DELSCR is conditional and under what conditions control should be returned to the issuing program, as follows.

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the delete cannot be serviced for any of the reasons listed below.

***condition***

Specifies conditions under which control is returned to the issuing program. Multiple *condition* options must be enclosed in parentheses and separated by commas. *Condition* options are as follows:

- **NAID** The scratch area ID cannot be found.
- **NRID** The scratch record ID cannot be found.
- **IOER** An I/O error occurs while processing the deletion.
- **INVP** The parameter list built for the #DELSCR is invalid.

**NAIDXIT=*no-scratch-area-id-label***

Specifies the symbolic name of the routine to which control should be returned if the #DELSCR request cannot be serviced because the scratch area ID cannot be found.

**NRIDXIT=*no-scratch-record-id-label***

Specifies the symbolic name of the routine to which control should be returned if the #DELSCR request cannot be serviced because the scratch record ID cannot be found.

**IOERXIT=*i/o-error-label***

Specifies the symbolic name of the routine to which control should be returned if the #DELSCR request cannot be serviced because of an I/O error while processing the #DELSCR request.

**INVPXIT=*invalid-parameter-list-label***

Specifies the symbolic name of the routine to which control should be returned if the #DELSCR request cannot be serviced because of an invalid parameter list.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

## #DELSCR Status Codes

By default, the #DELSCR request is unconditional; any runtime error will result in an abend of the issuing task. You can request return of control to the issuing program by using the COND parameter to avoid an abend.

After completion of the #DELSCR, the value in register 15 indicates the outcome of the operation. The following is a list of the Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'04'**

The request cannot be serviced due to an invalid parameter list

**X'08'**

The request cannot be serviced because the requested scratch area ID (SAID) cannot be found.

**X'0C'**

The request cannot be serviced because the requested scratch record ID (SRID) cannot be found within the named SAID.

**X'1C'**

The request cannot be serviced due to an I/O error during processing.

If an I/O error occurs while processing a #DELSCR request, the system will return the address of the IDMS communications block to register 1. In this situation, you can check the status code in the ERRSTAT field for more information (see ERRSTAT Field and Codes (see page 41)). If no error occurs during processing, a user-defined register assigned by the RTNSRID parameter will contain the SRID of the last scratch record deleted.

## #DELSCR Example

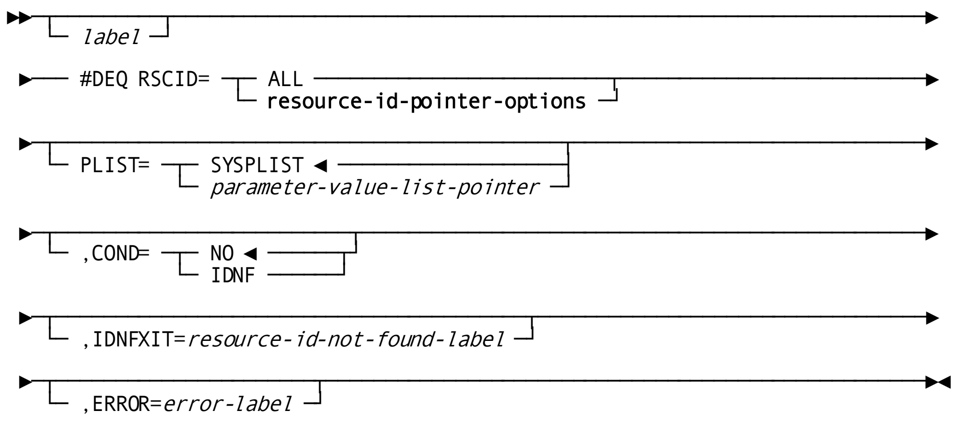
The following example of the #DELSCR statement deletes the current record within the scratch area labeled SCRAREA1. The ID of the deleted record will be placed in register 1. The request is not conditional; any error condition resulting from the execution of this statement will result in an abend of the issuing task.

```
#DELSCR SAID='SCRAREA1',LOC=CURRENT,RTNSRID=(R1),COND=NO
```

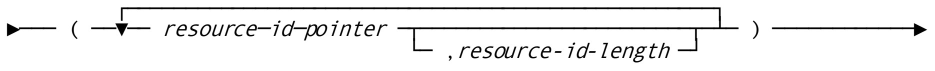
## #DEQ—releases resources acquired by the issuing task

The #DEQ statement releases resources acquired by the issuing task with an #ENQ request. All acquired resources will be released, either explicitly with a #DEQ request or automatically at task termination.

## #DEQ Syntax


label
#DEQ RSCID=
ALL
resource-id-pointer-options
PLIST=
SYSPLIST
parameter-value-list-pointer
,COND=
NO
IDNF
,IDNFXIT=resource-id-not-found-label
,ERROR=error-label

**Expansion of resource-id-pointer-options**



## #DEQ Parameters

**RSCID=**

Specifies the resources to be released.

**ALL**

Requests that the system release all resources acquired by the issuing task by means of the #ENQ requests.

**resource-id-pointer-options**

Specifies the ID associated with a specific resource to be dequeued. *Resource-id-pointer* is a register that points to a field containing the id, the symbolic name of a user-defined field containing the id, or the ID literal enclosed in quotation marks. *Resource-id-pointer* must be enclosed in parentheses.

The optional *resource-id-length* specifies the length of the resource ID named by *resource-id-pointer* (up to 256 bytes). *Resource-id-length* is a register that contains the length, the symbolic name of a fullword, halfword, or byte-length user-defined field containing the length, or an absolute expression. The length of the ID need not be specified if *resource-id-pointer* is provided as a literal enclosed in quotation marks.

Multiple RSCID parameters must be in successive order, separated by commas.

**PLIST=**

Specifies the location of the storage area in which the system will build the #DEQ parameter list, as follows.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system will build the #DEQ parameter list.

***parameter-value-list-pointer***

A register that points to the area or the symbolic name of the area in which the system will build the #DEQ parameter list.

The size, in fullwords, of the parameter-list area is equal to:

```
1 + 2P + ((R + 3)/4),
```

where:

- **P** is the number of *resource-id* specifications named for the RSCID parameter (described above).
- **R** is the number of *resource-id-length* specifications named in register notation for the RSCID parameter.

If RSCID=ALL is specified, the length of this storage area is one fullword; if five resource ids are specified and four have a length indicated in register notation, it is 13 fullwords. (Note that in this case the calculated value of 12.75 was rounded up to a whole number.)

**COND=**

Specifies whether this #DEQ is conditional and under what conditions control should be returned to the issuing program:

**NO**

(Default); specifies that the request is not conditional.

**IDNF**

Specifies that the request is conditional. Control is returned if one or more resource ids identified by the RSCID parameter cannot be found.

**IDNFXIT=*resource-id-not-found-label***

Specifies the symbolic name of the routine to which control should be returned if the #DEQ request cannot be completely serviced because one or more resource ids cannot be found.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded. In this case, the ERROR parameter functions the same as IDNFXIT.

## #DEQ Status Codes

By default, the #DEQ is unconditional. Error conditions that can occur are described below. If one or more resources cannot be found, the issuing task will abend. You can avoid an abend by specifying the COND parameter, requesting the DC/UCF system to return control to the issuing program.

After completion of the #DEQ request, the value in register 15 indicates the outcome of the operation. The following is a list of the Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'04'**

## #DEQ Example

The following example of the #DEQ statement releases the resource that is identified in the program variable storage field labeled RESOURC3. Register 4 contains the length of the resource. If the resource cannot be found, control will be returned to the routine NOTFOUND.

```
#DEQ RSCID=(RESOURC3,(4)),COND=IDNF,IDNFXIT=NOTFOUND
```

At least one resource ID (RSCID) could not be found; all that were located have been dequeued.

## @DISCON—cancels the current membership of a specified record

The @DISCON statement cancels the current membership of a specified record in a set occurrence. The specified record must be defined as an optional member of the named set.

**Note: Native VSAM users**—The @DISCON statement is not valid because all sets in native VSAM data sets must be defined as mandatory automatic.

The following consideration apply:

- All areas affected, either explicitly or implicitly, by the @DISCON statement must be readied with one of the update usage modes (see @READY (see page 308) later in this chapter).
- After successful execution of the @DISCON statement, you can no longer access the specified record through the set for which membership was canceled. However, you can access the disconnected record through all the other sets in which it participates as a member, or if it has a location mode of CALC. It is always accessible by means of a complete scan of the area in which it participates or directly through its db-key, if known.

**Currency**

Before execution of the @DISCON statement, the following currency-related conditions must be satisfied:

- The specified record must be established as current of its record type.
- The specified record must currently participate as a member in an occurrence of the named set.

A successfully executed @DISCON statement nullifies currency in the named set. However, the next of set and prior of set are maintained, thereby enabling continued access within the set. The disconnected record is current of run unit, its record type, and its area.

## @DISCON Syntax

```
►►── @DISCON REC=record-name ──────────────────────────────────────►
 ►── ,SET=set-name ───────────────────────────────────────────────►◄
```

## @DISCON Parameters

**REC=**

Disconnects the specified record from the named set.

***record-name***

Must be a record included in the subschema and must be defined as an optional member of the specified set.

**SET=**

Specifies the set from which the named record will be disconnected.

***set-name***

Must be a set included in the subschema.

## @DISCON Status Codes

After completion of the @DISCON function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**1106**

Currency has not been established for the named record.

**1108**

The named record is not in the subschema. The program has probably invoked the wrong subschema.

**1109**

The specified record's area has not been readied in one of the three update usage modes.

**1110**

The subschema specifies an access restriction that prohibits use of the @DISCON statement.

**1115**

The @DISCON statement cannot be executed because the specified record has been defined as a mandatory member of the set.

**1121**

An area other than the area of the specified record has been readied with an incorrect usage mode.

**1122**

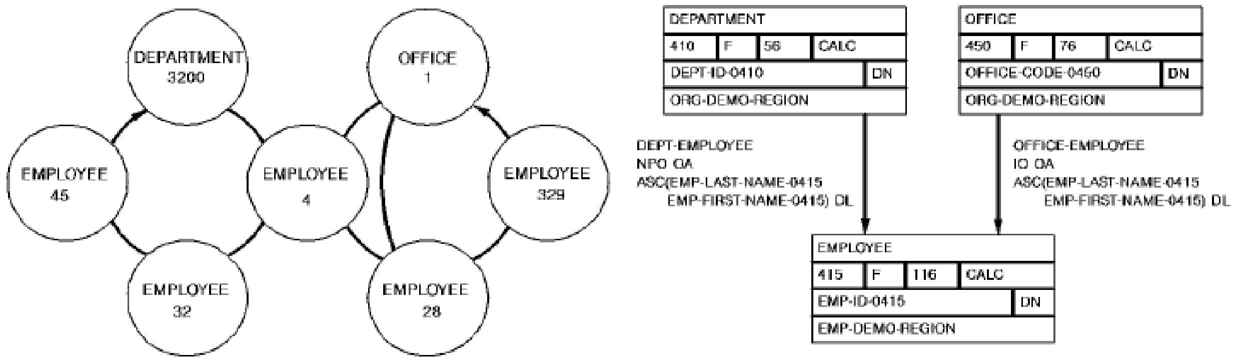
The specified record is not currently a member of the specified set.

## @DISCON Example

The following example demonstrates the use of the @DISCON statement to remove an EMPLOYEE record from the OFFICE-EMPLOYEE set occurrence. The EMPLOYEE record remains a member in the other set occurrences in which it participates:

```
MVC    OFFCODE,=CL4'3200'
@FIND CALC,REC='OFFICE'
@FIND FIRST,REC='EMPLOYEE',SET='OFFICE-EMPLOYEE'
@DISCON REC='EMPLOYEE',SET='OFFICE-EMPLOYEE'
```

The following figure illustrates the above example. To disconnect EMPLOYEE 4 from the OFFICE 1 occurrence of the OFFICE-EMPLOYEE set, enter the database on OFFICE 1, establish EMPLOYEE 4 as current of the EMPLOYEE record type, and disconnect it from the OFFICE-EMPLOYEE set.



| | CURRENCIES: RUN UNIT, RECORD, SET, AREA | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RUN UNIT | EMPLOYEE | DEPARTMENT | OFFICE | DEPT-EMPLOYEE | OFFICE-EMPLOYEE | ORG-DEMO-REGION | EMP-DEMO-REGION |
| MVC OFFCODE,OFFCODIN @FIND CALC, REC='OFFICE' | 1 | | | 1 | | 1 | 1 | |
| @FIND='OFFICE-EMPLOYEE', FIRST, REC='EMPLOYEE' | 4 | 4 | | 1 | 4 | 4 | 1 | 4 |
| @DISCON REC='EMPLOYEE', SET='OFFICE-EMPLOYEE' | 4 | 4 | | 1 | 4 | NPO | 1 | 4 |

## #ENQ—acquires resources or tests for availability

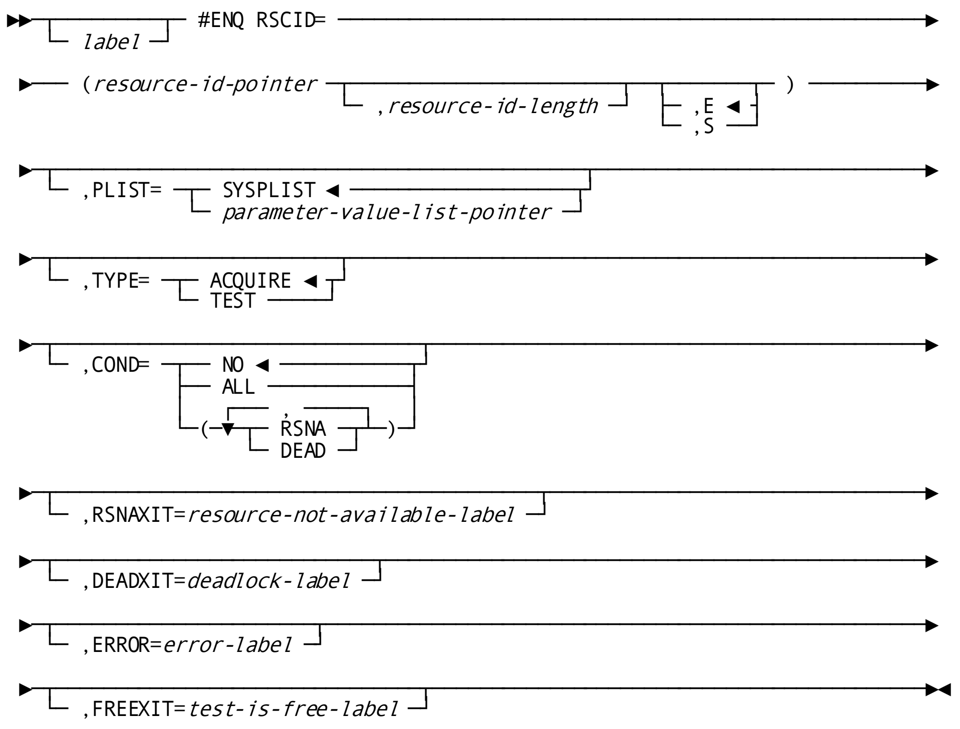
The #ENQ statement acquires resources or tests for availability of a resource or list of resources. Defined during installation, resources can be storage areas, common routines, queues, and processor time.

An enqueued resource can be exclusive or shared:

- **Exclusive** specifies that the resource is owned exclusively by the issuing task and is not available to any other tasks. The system prohibits other tasks from issuing #ENQ requests for exclusive resources.
- **Shared** specifies that the resource is available for use by all tasks. The system allows other tasks to issue nonexclusive #ENQ requests for the resources, permitting the resources to be shared.

An exclusive #ENQ request prohibits another task from enqueuing a resource by name; however, it does not prohibit the use of the resource by another task. Therefore, to effect queue resource protection, you must apply the enqueue/dequeue mechanism consistently, according to your site standards.

### #ENQ Syntax



## #ENQ Parameters

**RSCID=**

Names one or more resources to be acquired or tested, specifies the length of each resource, and designates the resource as exclusive or shared.

***resource-id-pointer***

Specifies the character ID associated with a resource. The resource-id-pointer can be a register that points to a field that contains the ID, he symbolic name of a user-defined field that contains the ID, or the ID literal enclosed in quotation marks. The source-id is a 1 to 256 byte character string used to identify the resource upon which an enqueue is to be set or tested. Any character string may be defined as long as all programs that access the resource use the same name and the name is unique relative to all other names used to identify other resources within the CV.

***resource-id-length***

Specifies the length of the resource id. *Resource-id-length* is a register that contains either the length, the symbolic name of a fullword, halfword, or byte-length user-defined field that contains the length, or an absolute expression. You need not specify the length of the ID if *resource-id-pointer* is provided as a literal enclosed in quotation marks.

**E/S**

Assigns the exclusive (E) (default) or shared (S) attribute to the named resource.

Note: Multiple RSCID parameters must be in successive order, separated by commas.

**PLIST=**

Specifies the location of the storage area in which the system will build the #ENQ parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system will build the #ENQ parameter list.

***parameter-value-list-pointer***

Either a register that points to the area or the symbolic name of the area in which the system will build the #ENQ parameter list.

The size of the parameter-list area, in fullwords, is equal to:

1 + 3P + ((R + 3)/4)

where:

- **P** is the number of *resource-id* specifications in the RSCID parameter (described above).
- **R** is the number of *resource-id-length* specifications named in register notation for the RSCID parameter.

Thus, if four resource IDs are specified and three are identified using register notation, the length of this storage area is 15 fullwords. In this case the calculated value of 14.5 was rounded up to a whole number. Calculated values are always rounded **up** to the nearest whole number, regardless of the remainder value.

**TYPE=**

Specifies whether the issuing task is to test a resource for availability or request acquisition of a resource:

**ACQUIRE**

(Default); requests that the system acquire the specified resources.

**TEST**

Requests that the system test the availability of the specified resource.

**COND=**

Specifies whether this #ENQ request is conditional and under what conditions control should be returned to the issuing program. Only acquire requests can be conditional; this parameter should *not* be specified when testing the enqueue status of a resource.

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the #ENQ cannot be serviced for any of the reasons listed below.

***condition***

Specifies specific conditions you can test for. Multiple conditions must be enclosed in parentheses and separated by commas.

**RSNA**

Specifies that control is returned if any of the requested resources is not available in the usage mode requested.

**DEAD**

Specifies that control is returned if a requested resource cannot be enqueued immediately because of an unavailable condition, and or to wait would cause a deadlock.

**RSNAXIT=*resource-not-available-label***

Specifies the symbolic name of a routine to which control should be returned if the #ENQ request cannot be serviced because at least one of the requested resources is not available.

**DEADXIT=*deadlock-label***

Specifies the symbolic name of a routine to which control should be returned if the #ENQ request cannot be serviced because one of the requested resources cannot be enqueued immediately, and if to wait on its availability would cause a deadlock.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

**FREEXIT=*test-is-free-label***

(Test requests only); specifies the symbolic name of a routine to which control should be returned if at least one of the resources is free.

## #ENQ Status Codes

By default, an *acquire* #ENQ is unconditional. Error conditions that can occur are described below:

- A resource-not-available condition, caused when at least one of the resources cannot be acquired by the issuing task, will result in a delay until the resource becomes available (unless such a wait would cause a deadlock).
- A potential deadlock condition, caused when a wait on a resource would cause a deadlock, will result in an abend of the issuing task.

You can request return of control with the COND parameter while processing an acquire #ENQ to avoid a delay or an abend.

By default, a *test* #ENQ is unconditional. The return code, contained in register 15, indicates the outcome of the test. Control is returned to the next instruction in the issuing program following the #ENQ. Through the FREEXIT parameter, however, you can request a return of control to a specific label or routine in the event that at least one of the resources tested is free.

After completion of the #ENQ request, the value in register 15 indicates the outcome of the operation.

**X'00'**

ACQUIRE - All requested resources have been acquired.

TEST - All test resources have already been enqueued by the issuing task with the exclusive/shared option indicated by the test request.

**X'04'**

ACQUIRE-At least one of the resources indicated is currently owned by another task and is not available for the exclusive/shared option specified; no new resources have been acquired.

TEST- At least one of the tested resources is owned by another task and is not available to this task for the exclusive/shared option specified.

**X'08'**

ACQUIRE -Not applicable.

TEST - At least one of the tested resources is not already owned by any task and is available for the exclusive/shared option specified.

**X'0C'**

ACQUIRE - A requested resource could not be enqueued immediately and to wait would cause a deadlock; no new resources have been acquired.

TEST - Not applicable.

### #ENQ Example

The following example of the #ENQ statement tests for the availability of a resource. Register 5 contains the address of the field that contains the resource id, the user-defined field LENGTH contains the length of the resource id, and if the test indicates the resource is free, control is returned to the routine labeled GETRTN:

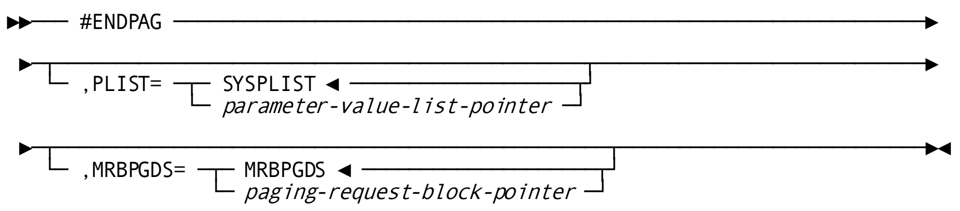
```
#ENQ RSCID=(R5),LENGTH,TYPE=TEST,FREEXIT=GETRTN
```

## #ENDPAG—terminates a map paging session

The #ENDPAG statement terminates a map paging session, clears the scratch record for the session, and clears the map paging options for the completed session. A #STRTPAG/#ENDPAG pair encloses commands that handle a pageable map at runtime.

Note: For more information about the #STRTPAG statement, see #STRTPAG (see page 340) later in this chapter.

### #ENDPAG Syntax



## #ENDPAG Parameters

**PLIST=**

Specifies the location of the storage area in which the system will build the #ENDPAG parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system will build the #ENDPAG parameter list.

***parameter-value-list-pointer***

A register that points to the area or the symbolic name of the area.

**MRBPGDS=**

Specifies the location of the 16-byte map paging request block.

**MRBPGDS**

(Default); is the symbolic name of the area in program variable storage in which the map paging request block was copied by an #MRB DML statement.

***paging-request-block-pointer***

A register that points to the area or the symbolic name of the area that contains the map paging request block.

## #ENDPAG Status Codes

The #ENDPAG statement is unconditional; any runtime error will result in an abend of the issuing task.

## #ENDPAG Example

The following example of the #ENDPAG statement terminates a map paging session that began with the #STRTPAG statement, clears the BACKPAG=YES and FLAG=UPDATE map paging options, and specifies the address of the #ENDPAG parameter list in register 3:

```
#STRTPAG MRB=(R4),BACKPAG=YES,FLAG=UPDATE

.

.  (*** MAP PAGING SESSION ***)

.

#ENDPAG PLIST=(R3)
```

## @ERASE—disconnects or erases records

The @ERASE statement performs the following functions:

- Disconnects the specified record from all set occurrences in which it participates as a member and physically deletes the record from the database
- Optionally erases all records that are mandatory members of set occurrences owned by the specified record
- Optionally disconnects or erases all records that are optional members of set occurrences owned by the specified record

Erasure is a two-step process that first cancels the existing membership of the specified record in specific set occurrences and then releases for reuse the space occupied by the named record and its db-key. Erased records are unavailable for further processing by any DML statement.

Before using the @ERASE statement, you must ready all the areas affected, either implicitly or explicitly, in one of the three update usage modes (see Dictionary Usage Mode (see page 29)).

**Currency**

Before execution of the @ERASE statement, the following currency-related conditions must be satisfied:

- All sets in which the specified record participates as owner either directly or indirectly (for example, as owner of a set with a member that is owner of another set) and all member record types in those sets must be included in the subschema in use.
- The named record must be established as current of run unit.

Following successful execution of an @ERASE statement, currency is nullified for all record types both explicitly and implicitly involved in the erase and for all sets in which erased records participate. Run unit and area currency remain unchanged.

**Note: Native VSAM users**—When the @ERASE statement is used against a native VSAM area, the area currency changes and reflects the next record in the native VSAM area.

An attempt to retrieve erased records results in an error condition. However, if the erased record was reached by walking the set occurrence of the erased record, the prior of set is maintained for the erased record, whether or not prior pointers were defined for that set. (The next of set is also maintained, as usual). Also, CA IDMS/DB maintains the next, prior, and owner pointers for the last erased record occurrence that participates as a member in any other set occurrence not the object of the @ERASE. In this case, you can retrieve the next or prior records in the area, or the next, prior, or owner records in the set in which the erased record participated.

## @ERASE Syntax

```
►►── @ERASE= ─┬─ REC ───────┬──────────────────────────────────────────►
              ├─ PERMANENT ─┤
              ├─ SELECTIVE ─┤
              └─ ALL ───────┘

 ►── ,REC=record-name ─────────────────────────────────────────────────►◄
```

## @ERASE Parameters

**REC/PERMANENT/SELECTIVE/ALL,REC=*record-name***

Erases a record from the database.

**REC**

Erases the specified record if it is not an owner of any member records. An error condition results if the named record is the owner of any nonempty set occurrences.

**Note: Native VSAM users**—@ERASE REC,REC=*record-name* is the only form of the @ERASE statement valid for records in a native VSAM KSDS or RRDS; no form of the @ERASE statement is allowed for a native VSAM entry-sequenced data set (ESDS).

**PERMANENT**

Erases the specified record and all mandatory member record occurrences owned by that record. Optional member records are disconnected. If any of the erased mandatory members are themselves the owners of any set occurrences, the @ERASE statement is executed on such records as if they were directly the named record of an @ERASE PERMANENT statement (that is, all mandatory members of such sets are also erased). This process continues until all (direct and indirect) members have been processed.

Note: The statement ERASE/PERMANENT/SELECTIVE/ALL cannot be used where there exists a cyclical relationship between two or more of the records that are to be erased. The following describes a cyclical set relationship:

```
REC-A owns REC-B in the A-B set
REC-B owns REC-C in the B-C set
REC-C owns REC-B in the C-B set

(cyclical relationship between REC-B and REC-C)
```

Junction records should be used to define the needed relationships.

**SELECTIVE**

Erases that record and all mandatory member record occurrences owned by the specified record. Optional member records are erased if they do not *currently participate* as members in other set occurrences. All erased records that are themselves the owners of any set occurrences are treated as if they were the object of an @ERASE SELECTIVE statement.

**ALL**

Erases the specified record and all mandatory member record occurrences owned by the specified record. All erased records that are themselves the owners of any set occurrences are treated as if they were the specified record of an @ERASE ALL statement.

**REC=*record-name***

A record included in the subschema. The current of *record-name* must be current of run unit.

## @ERASE Status Codes

After completion of the @ERASE function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**0208**

The named record is not in the specified subschema, or the record name has been misspelled.

**0209**

The specified record's area has not been readied in one of the three update usage modes.

**0210**

The subschema specifies an access restriction that prohibits use of the @ERASE statement. For integrated indexing users, this code can also indicate use of an invalid form of the @ERASE statement.

**0213**

A current record of run unit has not been established or has been nullified by a previous @ERASE statement.

**0217**

A db-key has been encountered that contains a long-term permanent lock.

**0220**

The current record of run unit is not the same type as the specified record.

**0221**

An area other than the area of the named record has been readied with an incorrect usage mode.

**0225**

Currency has not been established. For integrated indexing users, this usually indicates that an @FIND statement has been issued for an indexed record and followed by an @ERASE statement for the same record. Only an @OBTAIN statement updates index set currencies.

**0226**

A broken chain has been encountered in the process of executing an @ERASE ALL, PERMANENT, or SELECTIVE statement.

**0230**

An attempt has been made to erase the owner record of a nonempty set.

**0233**

Erasure of the record occurrence is not allowed in this subschema, or all sets in which the record participates have not been included in the subschema.

**0237**

There are cyclical set relationships present under the target record of the erase verb

**0260**

A record occurrence has been encountered whose type is inconsistent with the set named in the ERRORSET field of the IDMS communications block; probable causes could be a broken chain or improper database descriptions.

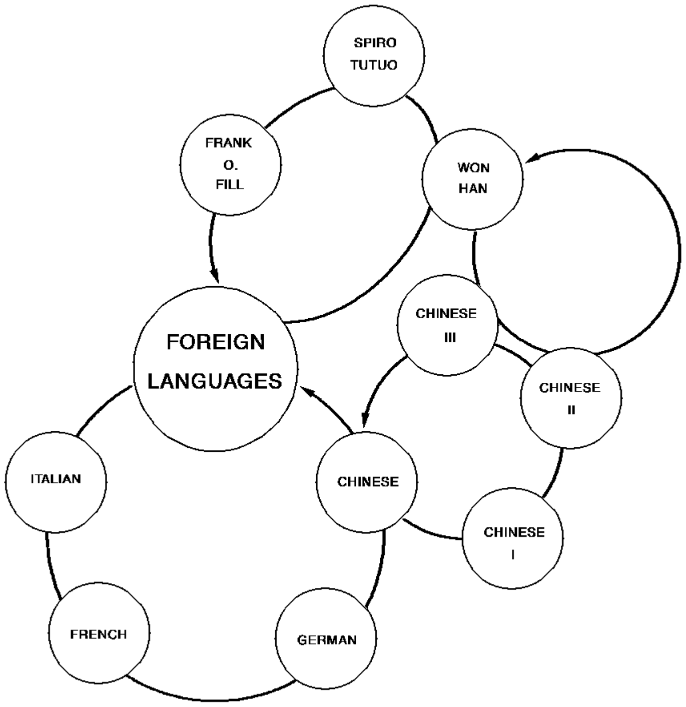
**0261**

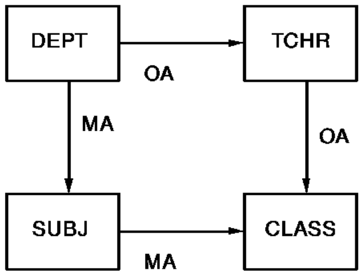
No record can be found for an internal db-key. This code usually indicates a broken chain.

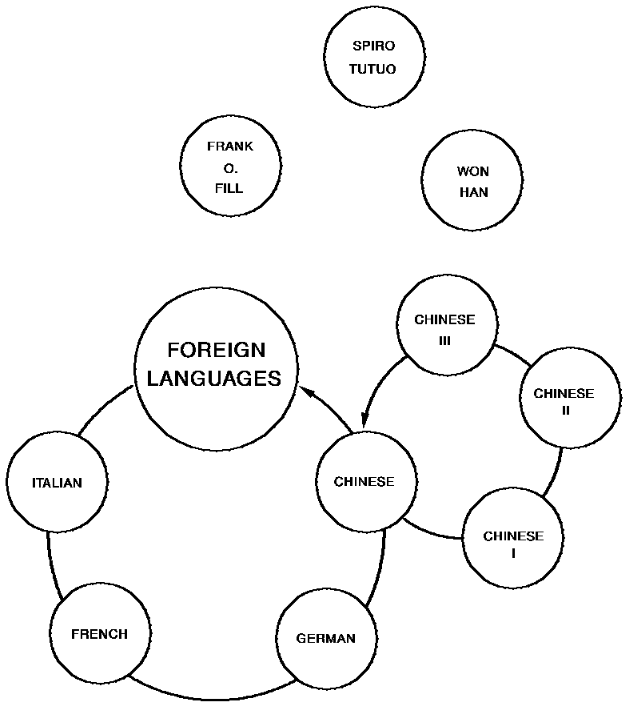
## @ERASE Example

```
@ERASE PERMANENT,REC='DEPT'

@ERASE SELECTIVE,REC='TCHR'

@ERASE ALL,REC='TCHR'
```

The sample employee database affords no appropriate examples of these parameters; a sample high school database is used instead. The outcome of the @ERASE statement varies, based on the qualifier specified (PERMANENT, SELECTIVE, or ALL). Although all three qualifiers cause all mandatory members owned by the specified record to be erased, they differ in their effect on optional members.







**@ERASE PERMANENT,REC='DEPT'**

(assuming that FOREIGN LANGUAGES is current of run unit)

The Foreign Languages Department can no longer be funded, so it is deleted from the database along with its subjects and classes. The teachers will be reassigned to other departments.

Erases the foreign language record and all mandatory members; disconnects optional members.

## @ERASE (LRF)—deletes logical record occurrences

The @ERASE statement can also be used to delete logical record occurrences. The @ERASE statement does not necessarily result in the deletion of all or any of the database records used to create the specified logical record; the path selected to service an @ERASE logical-record request performs whatever database access operations the DBA has specified to service the request.

LRF uses field values present in the variable-storage location reserved for the logical record to update the database. You can specify an alternative storage location from which LRF is to take field values to make the appropriate updates to the database.

### @ERASE (LRF) Syntax

```
►►─── @ERASE REC=logical-record-name ─────────────────────────────────►

►─┬────────────────────────────────────┬──────────────────────────────►
  └─ ,IOREA=alt-logical-record-location ┘

►─┬─────────────────────────────────────────────────┬─────────────────►
  └─ ,ONLRSTS=path-status,GOTO=branch-location ─────┘

►─┬───────────────────────────────┬───────────────────────────────────►◄
  └─ ,WHERE boolean-expression ───┘
```

## @ERASE (LRF) Parameters

**REC=*logical-record-name***

Deletes the named logical record. Unless the IOAREA clause (below) is included, LRF uses field values present in the variable-storage location reserved for the logical record to make any necessary updates to the database. *Logical-record-name* must specify a logical record defined in the subschema.

**IOAREA=*alt-logical-record-location***

Identifies an alternative variable-storage location from which LRF is to obtain field values to perform the appropriate database updates in response to this statement. When erasing a logical record that has previously been retrieved into an alternative storage location, you should use the IOAREA parameter to name the same location specified in the @OBTAIN request. If the IOAREA parameter is included in the @ERASE statement, *alt-logical-record-location* must identify a record location defined in the program.

**ONLRSTS=*path-status*,GOTO=*branch-location***

Tests for the indicated path status. If *path-status* results from this @ERASE statement, the action specified by GOTO=*branch-location* is performed. *Path-status* must be a literal (1-16 bytes) enclosed in quotation marks or a program variable.

**WHERE boolean-expression**

Specifies the selection criteria to be applied to the specified logical record.

**Note:** For more information about the WHERE clause, see WHERE Clause (see page 388) later in this chapter.

## @ERASE (LRF) Status Codes

When using LRF, the type of status code returned to the program in the ERRSTAT field of the IDMS communications block differs according to the type of error. If the error occurs in the *logical-record path*, the ERRSTAT field contains a status code issued by CA IDMS/DB with a major code from 00 to 19. For a list of these codes, see ERRSTAT Field and Codes (see page 41).

When the error occurs in the request itself, LRF returns the path status LR-ERROR to the LRSTAT field of the LRC block and places a status code with a major code of 20 in the ERRSTAT field of the IDMS communications block.

## @ERASE (LRF) Example

The example below illustrates a request to erase the OFFEMPLR logical record for office 012's employee ID 1234.

In this example, the DBA has designated the keyword DELETE-EMPLOYEE to direct the request to the path designed to retrieve the appropriate OFFEMPLR logical record and to delete the indicated employee information from the database.

```
@ERASE REC=OFFEMPLR,                                              *
      ONLRSTS='NO-OFFICE',GOTO=END,                               *
      WHERE OFFCODE EQ '012'                                      *
      AND EMPID EQ '1234'                                         *
      AND DELETE-EMPLOYEE
```

## @FIND/@OBTAIN Statements—accesses database records

The @FIND and @OBTAIN statements are used to access database records:

- **@FIND** locates a record occurrence in the database, but does not move it into program variable storage.
- **@OBTAIN** locates the record occurrence in the database and moves it into program variable storage.

### Six formats

@FIND and @OBTAIN have six different formats:

- **@FIND/@OBTAIN CALC/DUPLICATE** accesses a record occurrence using its CALC-key value.
- **@FIND/@OBTAIN CURRENT** accesses a record occurrence using previously established currencies.
- **@FIND/@OBTAIN DBKEY** accesses a record occurrence using a db-key that was previously saved by the program.
- **@FIND/@OBTAIN OWNER** accesses the owner of a set occurrence.
- **@FIND/@OBTAIN USING SORT KEY** accesses a record occurrence in a sorted set, using its sort-key value.
- **@FIND/@OBTAIN WITHIN SET/AREA** accesses a record occurrence based either on the record's logical location in a set or on its physical location in an area.

Each of these @FIND/@OBTAIN statements is discussed on the following pages.

## @FIND/@OBTAIN CALC/DUPLICATE

The @FIND/@OBTAIN CALC/DUPLICATE statement accesses a record based on the value of an element in the record defined as a CALC-key. The requested record must be stored in the database with a location mode of CALC. Before issuing the @FIND/@OBTAIN CALC/DUPLICATE statement, you must initialize a field in program variable storage with the CALC-key value.

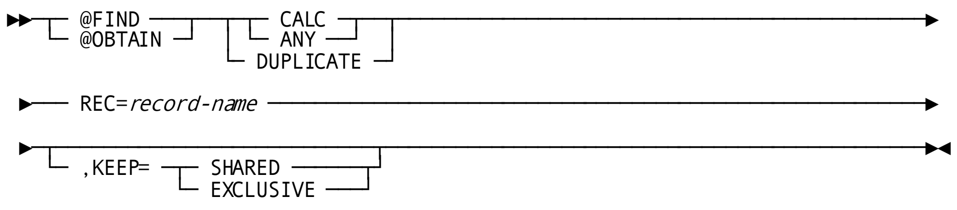
You can use the DUPLICATE option to access records with the same CALC-key value as the record that is current of record type, provided that an @FIND/@OBTAIN CALC statement has previously accessed an occurrence of the same record type.

**Currency**

You do not need to establish currency before executing a @FIND/@OBTAIN CALC statement. However, record currency must be established by a prior @FIND/@OBTAIN CALC statement before executing a @FIND/@OBTAIN DUPLICATE statement.

Following successful execution of an @FIND/@OBTAIN CALC/DUPLICATE statement, the accessed record becomes the current record of run unit, its area, its record type, and all sets in which it currently participates as member or owner.

**Syntax**



**Parameters**

**CALC/DUPLICATE,REC=*record-name***

Accesses the record specified by *record-name* using the value of its CALC-key.

**CALC**

Accesses the first or only occurrence of the designated record type whose CALC-key matches the value of the CALC data item in program variable storage. ANY is a synonym of CALC.

**DUPLICATE**

Accesses the next record with the same CALC-key value as the current record type. Use of the DUPLICATE option requires prior selection of an occurrence of the same record type with the CALC option. If the value of the CALC-key in variable storage is not equal to the CALC-key field of the current of record type, a status code of 0332 is returned.

**REC=*record-name***

Names the record being accessed. *Record-name* can be a register containing the name of the record or a user-supplied value enclosed in quotation marks.

**KEEP=**

Optionally places a shared or exclusive lock on the accessed record.

**SHARED**

Places a shared lock on the specified record.

**EXCLUSIVE**

Places an exclusive lock on the specified record.

### Example

To retrieve an occurrence of the EMPLOYEE record with the @FIND/@OBTAIN CALC/DUPLICATE statement, you must first initialize a field in program variable storage with the CALC-control element. The following statements initialize the CALC field EMPID and retrieve an occurrence of the EMPLOYEE record:

```
MVC    EMPID,INEMPID
@OBTAIN CALC,REC='EMPLOYEE'
```

### Status codes

After completion of the @FIND/@OBTAIN CALC/DUPLICATE function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**0301**

The area in which the named record participates has not been readied.

**0306**

A successful @FIND/@OBTAIN CALC has not yet been executed (applies to the DUPLICATE option only).

**0308**

The specified record is not in the subschema. The program has probably invoked the wrong subschema, or the record name has been misspelled.

**0310**

The subschema specifies an access restriction that prohibits retrieval of the named record.

**0318**

The record has not been bound.

**0326**

The record or integrated indexing entry cannot be found, or no more duplicates exist for the named record.

**0331**

The retrieval statements format conflicts with the record's location mode.

**0332**

The value of the CALC data item in program variable storage does not equal the value of the CALC data item in the current record (applies to the DUPLICATE option only).

**0364**

The CALC control element has not been described correctly either in the program or in the subschema.

**0370**

A database file will not open properly.

When the KEEP parameter is specified a major code of 06 will be returned if an error occurs during the KEEP processing. The major code of 03 states that an error has occurred in the @FIND/@OBTAIN processing.

## @FIND/@OBTAIN CURRENT

The @FIND/@OBTAIN CURRENT statement accesses the record that is current of its record type, set, or area. This form of the @FIND/@OBTAIN verb is an efficient means of establishing the proper record as current of run unit before executing a DML verb that utilizes run-unit currency (for example, @ACCEPT, @IF, @GET, @MODIFY, or @ERASE).

**Currency -** Following successful execution of an @FIND/@OBTAIN CURRENT statement, the accessed record is current of run unit, its area, its record type, and all sets in which it currently participates as member or owner.

**Syntax**

```
►►─┬─ @FIND ───┬────── CURRENT ──────────────────────────────────────────►
   └─ @OBTAIN ─┘

 ►─┬───────────────────┬──────────────────────────────────────────────────►
   ├─ ,REC=record-name ─┤
   ├─ ,SET=set-name ────┤
   └─ ,AREA=area-name ──┘

 ►─┬─────────────────────────────┬────────────────────────────────────────►◄
   └─ ,KEEP= ─┬─ SHARED ────────┤
              └─ EXCLUSIVE ────┘
```

**Parameters**

**@FIND/@OBTAIN CURRENT**

Accesses the record occurrence that is current of run unit.

**REC=*record-name*/SET=*set-name*/AREA=*area-name***

Specifies that the current record of the named record type, set, or area is to be accessed.

**REC=**

Accesses the record that is current of run unit.

***record-name***

A register containing the record name, a user-defined variable field, or a user-supplied value enclosed in quotation marks.

**SET=**

Accesses the set that is current of run unit.

***set-name***

A register containing the set name, a user-defined variable field, or a user-supplied value enclosed in quotation marks.

**AREA=**

Accesses the area that is current of run unit.

***area-name***

A register containing the area name, a user-defined variable field, or a user-supplied value enclosed in quotation marks.

**KEEP=**

Places a shared or exclusive lock on the accessed record.

**SHARED**

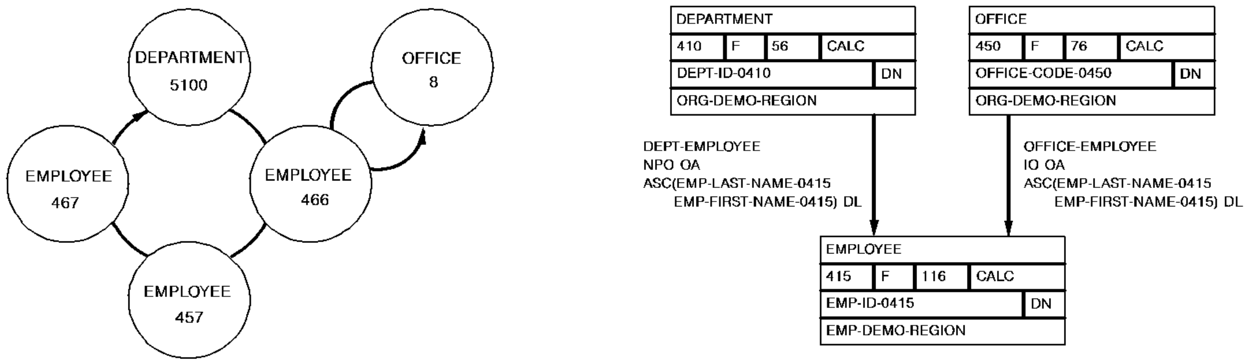
Places a shared lock on the specified record.

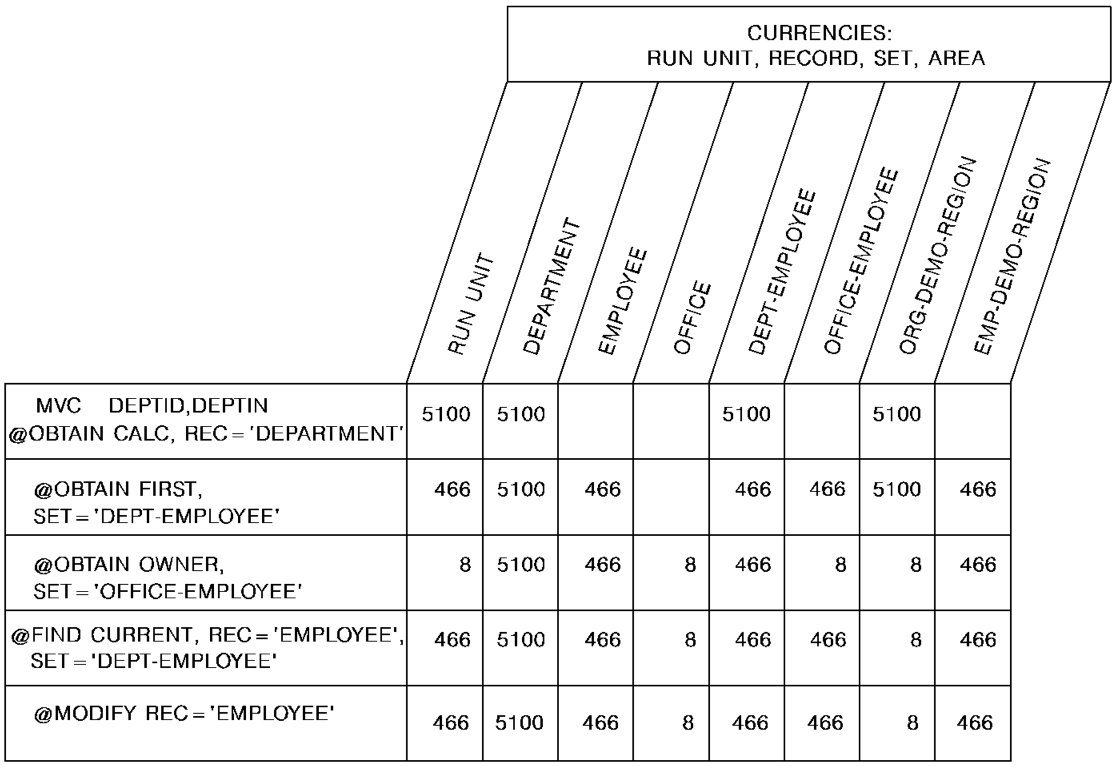
**EXCLUSIVE**

Places an exclusive lock on the specified record.

**Example**

The following figure illustrates the use of the @FIND/@OBTAIN CURRENT statement to establish a record as current of run unit before that record is modified. (See @MODIFY (see page 255) later in this chapter for a complete description of the @MODIFY verb and its use.) Enter the database on DEPARTMENT 5100 by using CALC retrieval. Then examine EMPLOYEE 466 and obtain further information from its owner OFFICE record. OFFICE 8 becomes current of run unit. Before modifying EMPLOYEE 466, you must issue the @FIND CURRENT statement to reestablish EMPLOYEE 466 as current of run unit.





| | CURRENCIES: RUN UNIT, RECORD, SET, AREA | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RUN UNIT | DEPARTMENT | EMPLOYEE | OFFICE | DEPT-EMPLOYEE | OFFICE-EMPLOYEE | ORG-DEMO-REGION | EMP-DEMO-REGION |
| MVC DEPTID,DEPTIN<br>@OBTAIN CALC, REC = 'DEPARTMENT' | 5100 | 5100 | | | 5100 | | 5100 | |
| @OBTAIN FIRST,<br>SET = 'DEPT-EMPLOYEE' | 466 | 5100 | 466 | | 466 | 466 | 5100 | 466 |
| @OBTAIN OWNER,<br>SET = 'OFFICE-EMPLOYEE' | 8 | 5100 | 466 | 8 | 466 | 8 | 8 | 466 |
| @FIND CURRENT, REC = 'EMPLOYEE',<br>SET = 'DEPT-EMPLOYEE' | 466 | 5100 | 466 | 8 | 466 | 466 | 8 | 466 |
| @MODIFY REC = 'EMPLOYEE' | 466 | 5100 | 466 | 8 | 466 | 466 | 8 | 466 |

### Status Codes

After completion of the @FIND/@OBTAIN CURRENT function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**0301**

The area in which the named record participates has not been readied.

**0306**

Currency has not been established for the named record, set, or area.

**0308**

The specified record is not in the subschema. The program has probably invoked the wrong subschema.

**0310**

The subschema specifies an access restriction that prohibits retrieval of the named record.

**0313**

A current record of run unit has not been established or has been nullified by a previous @ERASE statement.

**0323**

The area name specified has not been included in the subschema invoked.

When the KEEP parameter is specified, a major code of 06 will be returned if an error occurs during the KEEP processing. The major code of 03 states that an error has occurred in the @FIND/@OBTAIN processing.

## @FIND/@OBTAIN DBKEY

The @FIND/@OBTAIN DBKEY statement accesses a record occurrence directly by using a database key that has been stored previously by the program. You can use the DML @ACCEPT verb (see @ACCEPT DBKEY FROM CURRENCY (see page 85) and @ACCEPT DBKEY RELATIVE TO CURRENCY (see page 87)) or an Assembler assignment statement to save a db-key. In this manner, you can directly access any record in the program's subschema regardless of its location mode.

Additionally, the DML @ACCEPT PGINFO verb (see @ACCEPT PGINFO (see page 90), @ACCEPT DBKEY FROM CURRENCY (see page 85), and @ACCEPT DBKEY RELATIVE TO CURRENCY (see page 87)) can be used to save page information that can be used to directly access the record from a specific page group when the Mixed Page Binds Allowed feature is used.

For more information about the Mixed Page Group Binds Allowed feature, see the *Database Administration Guide.*

**Note: Native VSAM users**—This statement is not valid for accessing data records in a native VSAM key-sequenced data set (KSDS).

**Currency**

Currency is not used to determine the location of the record specified in the @FIND/@OBTAIN DBKEY statement; the record is identified by its db-key and, optionally, by its record name.

Following successful execution of an @FIND/@OBTAIN DBKEY statement, the accessed record becomes the current record of run unit, its area, its record type, and all sets in which it currently participates as member or owner. The RECNAME field of the IDMS communications block is updated with the name of the accessed record.

**Syntax**

```
►►─┬─ @FIND ───┬──────────────────────────────────────────────────────►
   └─ @OBTAIN ─┘

►─┬────────────────────────────────┬──────────────────────────────────►
  └─ ,KEEP= ─┬─ SHARED ────┬───────┘
             └─ EXCLUSIVE ─┘

►─┬─ DBKEY=db-key ─┬─────────────────────┬─┬──────────────────────────►◄
  │                └─ ,PGINFO=pg-info ───┘ │
  └─┬─────────────────────┬─ DBKEY=db-key ─┘
    └─ REC=record-name ───┘
```

**Parameters**

**@FIND/@OBTAIN DBKEY=*db-key***

Accesses a record directly by using a db-key value contained in program variable storage.

***db-key***

Identifies the location in program variable storage that contains a db-key previously saved by the program. If a record name is specified, *db-key* must contain the db-key of an occurrence of the named record type. If a record name is not specified, *db-key* can contain the db-key of an occurrence of any record type in the subschema. *Db-key* must identify a binary fullword synchronized field; it can be a register or a user-defined variable.

**KEEP=**

Places a shared or exclusive lock on the accessed record:

**SHARED**

Places a shared lock on the specified record.

**EXCLUSIVE**

Places an exclusive lock on the specified record.

**PGINFO=*pg-info***

Specifies page information that is used to determine the area with which the db-key is associated. If not specified, the page information associated with the record that is current of rununit is used.

**Note:** Page information is only used if the subschema includes areas that have mixed page groups; otherwise, it is ignored.

*Pg-info*, a four-byte field that is made up of two halfword fields, identifies the location in variable storage that contains the page information previously saved by the program.

Page information is returned in the PGINFO field in the subschema control area if the subschema includes areas in mixed page groups. Page information can also be returned using the @ACCEPT PGINFO, @ACCEPT DBKEY FROM CURRENCY, and @ACCEPT DBKEY RELATIVE TO CURRENCY statements.

**REC=*record-name***

Optionally identifies the record type of the requested record. *Record-name* must identify a record that is included in the subschema; it can be a register, a user-defined variable, or a user-supplied variable enclosed in quotes.

**Example**

The following @FIND statement locates an occurrence of the EMPLOYEE record whose db-key matches the value of a field in program variable storage called SAVEDKEY.

The located record becomes current of run unit, current of the EMPLOYEE record type, current of the DEPT-EMPLOYEE, OFFICE-EMPLOYEE, and all other sets in which it currently participates as member or owner, and current of the ORDER-REGION area.

```
@FIND DBKEY=SAVEDKEY,REC='EMPLOYEE'
```

**Status codes**

After completion of the @FIND/@OBTAIN DBKEY function, the ERRSTAT field in the IDMS communication block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

This request has been serviced successfully.

**0301**

The area in which the named record participates has not been readied.

**0302**

The db-key is inconsistent with the area in which the record is stored. The db-key has not been initialized properly, or the record name is incorrect.

**0308**

The requested record is not in the subschema. The program has probably invoked the wrong subschema.

**0310**

The subschema specifies an access restriction that prohibits retrieval of the na med record.

**0326**

The specified record cannot be found.

**0370**

A database file will not open properly.

When the KEEP parameter is specified as part of an @FIND/@OBTAIN statement, a major code of 06 will be returned if an error occurs during the KEEP processing (see @KEEP (see page 200) later in this chapter). The major code of 03 states that an error has occurred in the @FIND/@OBTAIN processing.

## @FIND/@OBTAIN OWNER

The @FIND/@OBTAIN OWNER statement accesses the owner record of the current set occurrence. You can use this statement to retrieve the owner record of any set whether or not that set has been assigned owner pointers.

**Note: Native VSAM users**—The @FIND/@OBTAIN OWNER statement is not valid since the owner records are not defined in native VSAM data sets.

**Currency**

To execute an @FIND/@OBTAIN OWNER statement, currency must be established for the specified set.

**Note:** When a record declared as an optional or manual member of a set is retrieved, it is *not* established as current of set if it is not currently connected to the named set. A subsequent attempt to retrieve the owner record will instead locate the owner of the current record of set. In such cases, you should determine whether the retrieved record is actually a member of the named set before issuing the @FIND/@OBTAIN OWNER statement. The @IF statement (see @IF (see page 197) in this chapter) can be used for this purpose.

Following successful execution of an @FIND/@OBTAIN OWNER statement, the accessed record becomes the current record of run unit, its area, its record type, and all sets in which it currently participates as member or owner. If the current record of set is the owner record when the statement is executed, currency in the specified set remains unchanged.

**Syntax**

```
►►─┬─ @FIND ───┬────── OWNER ────────────────────────────────────────────►
   └─ @OBTAIN ─┘

 ►── ,SET=set-name ──────────────────────────────────────────────────────►

 ►─┬────────────────────────────┬────────────────────────────────────────►◄
   └─ ,KEEP= ─┬─ SHARED ────────┤
              └─ EXCLUSIVE ─────┘
```

**Parameters**

**@FIND/@OBTAIN OWNER**

Accesses the owner record of the specified set occurrence.

**SET=*set-name***

Names the set whose owner record is to be retrieved. *Set-name* must be a set included in the subschema; it can be a register, a user-defined variable, or a user-supplied variable enclosed in quotes.

**KEEP=**

Places a shared or exclusive lock on the accessed record:

**SHARED**

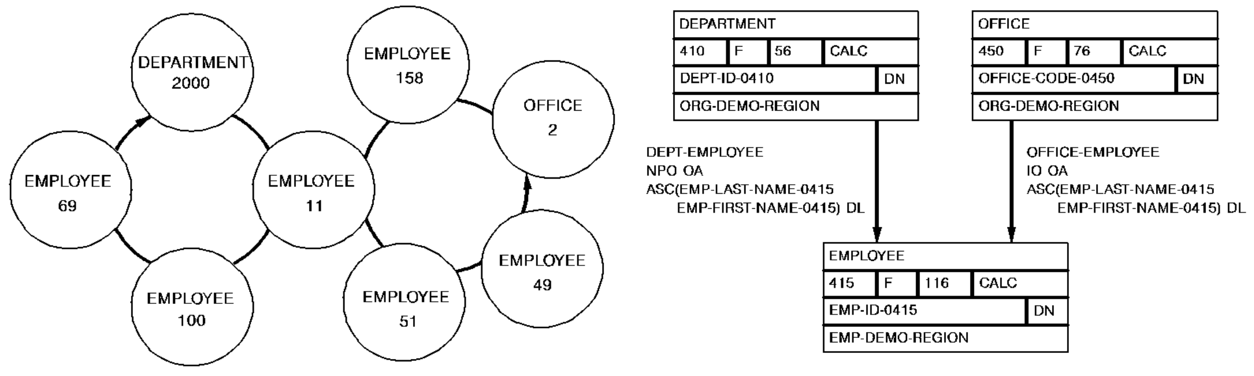
Places a shared lock on the accessed record.

**EXCLUSIVE**

Places an exclusive lock on the accessed record.

**Example**

The following figure provides an example of how you would use the @OBTAIN OWNER statement, in conjunction with other @OBTAIN statements, to navigate the database and access the owner record of the OFFICE-EMPLOYEE set from the owner record occurrence of the DEPT-EMPLOYEE set.



| | CURRENCIES: RUN UNIT, RECORD, SET, AREA | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RUN UNIT | DEPARTMENT | EMPLOYEE | OFFICE | DEPT-EMPLOYEE | OFFICE-EMPLOYEE | ORG-DEMO-REGION | EMP-DEMO-REGION |
| MVC DEPTID.DEPTIN<br>@OBTAIN CALC, REC = 'DEPARTMENT' | 2000 | 2000 | | | 2000 | | 2000 | |
| @OBTAIN FIRST, DEPT = 'EMPLOYEE' | 11 | 2000 | 11 | | 11 | 11 | 2000 | 11 |
| @OBTAIN OWNER,<br>SET = 'OFFICE-EMPLOYEE' | 2 | 2000 | 11 | 2 | 11 | 2 | 2 | 11 |

**Status codes**

After completion of the @FIND/@OBTAIN OWNER function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**0301**

The area in which the named record participates has not been readied.

**0306**

Currency has not been established for the named record, set, or area.

**0308**

The named record or the named set is not in the subschema, or the named record is not defined as a member of the named set. The program has probably invoked the wrong subschema. or the record name has been misspelled.

**0310**

The subschema specifies an access restriction that prohibits retrieval of the named record.

**0360**

A record occurrence has been encountered whose record type is not a member or owner of the set as it is defined in the subschema.

**0370**

A database file will not open properly

When the KEEP parameter is specified as part of an @FIND/@OBTAIN statement, a major code of 06 will be returned if an error occurs during the KEEP processing (see @KEEP (see page 200) in this chapter). The major code of 03 states that an error has occurred in the @FIND/@OBTAIN processing.

## @FIND/@OBTAIN USING SORT KEY

The @FIND/@OBTAIN USING SORT KEY statement accesses a member record in a sorted set. Sorted sets are ordered in ascending or descending sequence based on the value of a sort-control element in each member record. The search begins with the current of set *or* the owner of the current of set, and always proceeds through the set in the NEXT direction.

Before issuing this statement, you must initialize the sort-control element in program variable storage. The selected record occurrence will have a key value equal to the value of the sort-control element. If more than one record occurrence contains a sort key equal to the key value in variable storage, the first such record will be selected.

**Currency**

Before execution of an @FIND/@OBTAIN USING SORT KEY statement you have to establish currency for the specified set.

Following successful execution of an @FIND/@OBTAIN USING SORT KEY statement, the accessed record becomes current of run unit, its area, its record type, and all sets in which it currently participates as owner or member. If a member record with the requested sort-key value is not found, the current of set is nullified but the next of set and prior of set are maintained. The next of set is the member record with the next higher sort-key value (or next lower for descending sets) than the requested value; the prior of set is the member record with the next lower value (or higher for descending sets) than requested. Because these currencies are maintained, the program can walk the set to do a generic search on the sort-key value.

**Syntax**

```
►►─┬─ @FIND ───┬──────────────┬─ ,REC=record-name ──────────────────►
   └─ @OBTAIN ─┘   └─ CURRENT ─┘

►── ,SET=set-name ───────────────────────────────────────────────────►

►── USING=sort-field-name ───────────────────────────────────────────►

►─┬──────────────────────────────┬───────────────────────────────────►◄
  └─ ,KEEP= ─┬─ SHARED ────┬─────┘
             └─ EXCLUSIVE ─┘
```

**Parameters**

**@FIND/@OBTAIN,REC=*record-name*,SET=*set=name***

Accesses the named record in a sorted set. The search begins with the owner of the current record of the specified set. *Record-name* must be a record that is defined in the subschema and that participates in the specified set.

**CURRENT**

Current indicates that the search begins with the currencies already established for the specified set. If the key value for the record that is current of set is higher than the key value of the specified record (assuming ascending set order), an error condition results.

**USING=**

Specifies the sort-control element to be used in searching the sorted set.

***sort-field-name***

The name of the sort-control element in the record or the name of a field in program variable storage that contains the value of the sort-control element.

Note: The value coded for *sort-field-name* can only specify a single field name. If the sort key is composed of multiple fields, the value coded must point to an area of contiguous storage that contains the values of the various key components. These field values must be in the same sequence as the corresponding fields within the set's schema definition and their data formats must match the formats of the fields within the database record's definition.

**KEEP=**

Places a shared or exclusive lock on the accessed record.

**SHARED**

Places a shared lock on the specified record.

**EXCLUSIVE**

Places an exclusive lock on the specified record.

**Example**

The following example illustrates the use of an @FIND/@OBTAIN USING SORT KEY statement. Assume that the DEPT-EMPLOYEE set is ordered in ascending sequence, based on the value stored in EMPNAME in each EMPLOYEE record occurrence. The @FIND statement assumes that the user has previously selected an occurrence of a DEPARTMENT record to establish the set currency. Retrieval of an EMPLOYEE record with a name (last name, first name) equal to IANDOLI, LUIGI is accomplished by the following statements:

```
MVC   EMPNAME,=CL25'IANDOLI, LUIGI'
@FIND REC='EMPLOYEE',SET='DEPT-EMPLOYEE',USING=EMPNAME
```

**Status codes**

After completion of the @FIND/@OBTAIN USING SORT KEY function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**0301**

The area in which the named record participates has not been readied.

**0306**

Currency has not been established for the named set.

**0308**

The named record or the named set is not in the subschema, or the named record is not a member of the named set. The program has probably invoked the wrong subschema.

**0310**

The subschema specifies an access restriction that prohibits retrieval of the named record.

**0326**

The record cannot be found.

**0331**

The retrieval statement format conflicts with the record's location mode.

**0360**

A record occurrence has been encountered whose record type is not a member or owner of the set as it is defined in the subschema.

**0361**

A record cannot be found because of a broken chain in the database.

**0370**

A database file will not open properly.

When the KEEP parameter is specified as part of an @FIND/@OBTAIN statement, a major code of 06 will be returned if an error occurs during the KEEP processing (see @KEEP (see page 200) in this chapter). The major code of 03 states that an error has occurred in the @FIND/@OBTAIN processing.

## @FIND/@OBTAIN WITHIN SET/AREA

The @FIND/@OBTAIN WITHIN SET/AREA statement accesses records logically based on set relationships or physically based on database location. The formats of this statement allow you serial access to each record in a set or area, or selection of specific occurrences of a given record type in a set or area.

### Set currency

The following rules apply to currency and the selection of member records in a *set*:

- The set occurrence used as the basis for the operation is determined by the current record of the specified set. Set currency must be established before attempting to access records in a set.
- The next or prior record in a set is the subsequent or previous record, respectively, relative to the *current record of the named set* in the logical order of the set. The prior record in a set can be retrieved only if the set has been assigned prior pointers.
- The first or last record in a set is the first or last member occurrence in terms of the logical order of the set. The record selected is the same as would be selected if the current of set were the owner record and the next or prior record had been requested. The last record in a set can be retrieved only if the set has prior pointers.
- The *n*th occurrence of a record in a set can be retrieved by specifying a sequence number that identifies the position of the record in the set. CA IDMS/DB begins its search with the *owner of the current of set* for the specified set and continues until it locates the *n*th record or encounters an end-of-set condition. If the specified sequence number is negative, the search proceeds in the prior direction in the set. Note, however, that prior pointers are required to exercise this option.
- When an end-of-set condition occurs, the owner record occurrence of the set becomes the current record of run unit, current of its record type, current of its area, and current of only the set involved in this operation. Currency of other sets in which the specified record participates as owner or member remains unaffected.

**Note: Note 1**If @OBTAIN has been specified, the contents of the owner record are not moved to program variable storage (@OBTAIN under these circumstances is treated as an @FIND).

**Note: Note 2**(Native VSAM users): When an end-of-set condition occurs, all currencies remain the same.

### Area currency

The following rules apply to currency and the selection of records in an *area*:

- The first record occurrence in an area is the one with the lowest db-key; the last record is the one with the highest db-key.

- The next record in an area is the one with the next higher db-key relative to the *current record of the named area;* the prior record is the one with the next lower db-key relative to the current of area.
- The first, last, or *n*th occurrence of a record in an area must be retrieved to establish correct starting position before next or prior records are requested.

Following successful execution of an @FIND/@OBTAIN WITHIN SET/AREA statement, the accessed record becomes the current record of run unit, its area, its record type, and all sets in which it currently participates as member or owner.

**Syntax**

```
►►─┬─ @FIND ───┬──┬─ NEXT ──────────────────────────────────────────────────►
   └─ @OBTAIN ─┘  ├─ PRIOR ─┤
                  ├─ FIRST ─┤
                  ├─ LAST ──┤
                  └─ NTH ───┘

 ►─┬─ ,SET=set-name ────┬────────────────────────────────────────────────────►
   └─ ,AREA=area-name ──┘

 ►─┬─────────────────────┬───────────────────────────────────────────────────►
   └─ ,REC=record-name ──┘

 ►─┬─────────────────────┬───────────────────────────────────────────────────►
   └─ ,OCCUR=sequence ───┘

 ►─┬──────────────────────────────────┬──────────────────────────────────────►◄
   └─ ,KEEP= ──┬─ SHARED ─────────────┤
               └─ EXCLUSIVE ─────────┘
```

**Parameters**

**NEXT/PRIOR/FIRST/LAST/NTH**

Accesses a record based on its location in a set or area.

**NEXT**

Accesses the next record in the specified set or area relative to the current record of the set or area.

**PRIOR**

Accesses the prior record in the specified set or area relative to the current record of the set or area. The specified set must have prior pointers.

**FIRST**

Accesses the first record in the specified set or area.

**LAST**

Accesses the last record in the specified set or area. The specified set must have prior pointers.

**NTH**

Accesses the *n*th record in the specified set or area. NTH requires the use of the OCCUR parameter (see below) to specify which record is to be accessed.

**Note: Native VSAM users**—FIRST, LAST, and NTH options are not allowed for a native VSAM KSDS with spanned records.

**SET=*set-name*/AREA=*area-name***

Specifies the set or area to be searched.

**SET=*set-name***

Specifies the name of the set that contains the record to be accessed. *Set-name* must identify an set included in the subschema.

**AREA=*area-name***

Specifies the name of the area that contains the record to be accessed. *Area-name* must identify an area included in the subschema.

**REC=**

Specifies that in a set or area, only occurrences of the named record type will be accessed.

***record-name***

Must be defined as a member of the specified set or contained in the specified area.

**OCCUR=**

Identifies the position of the record in the set (that is, the numeric occurrence that is associated with the keyword NTH).

***sequence***

Must specify a positive or negative number that is stored in a numerical field used by CA IDMS/DB in searching for the *n*th record occurrence. If *sequence* specifies a negative number, the specified set must have prior pointers.

**KEEP=**

Places a shared or exclusive lock on the accessed record.

**SHARED**

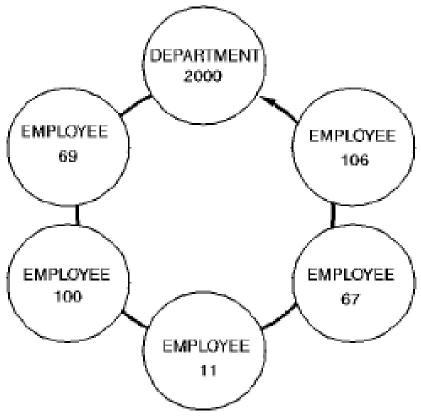
Places a shared lock on the specified record.

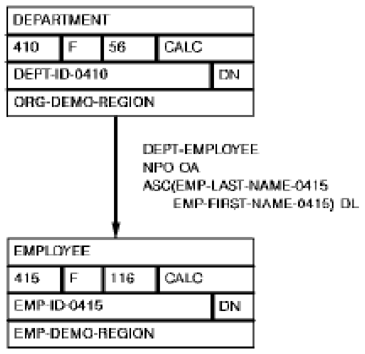
**EXCLUSIVE**

Places an exclusive lock on the specified record.

**Example**

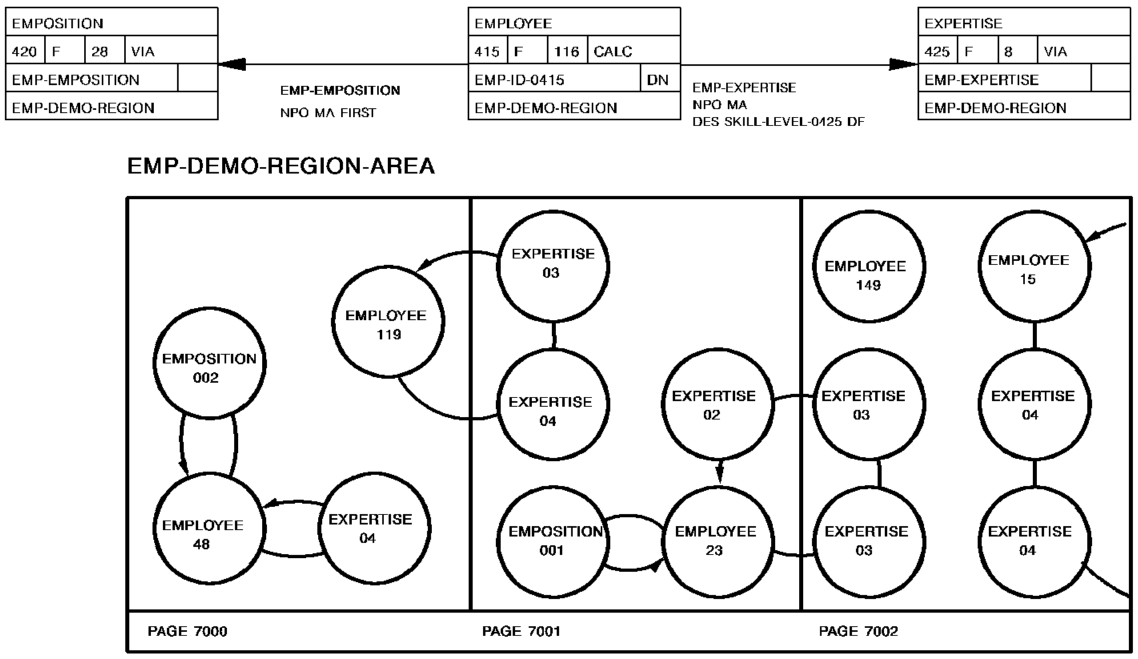
The following example illustrates the retrieval of records in an occurrence of the DEPT-EMPLOYEE set. The @FIND CALC statement establishes currency in the DEPT-EMPLOYEE set. Member EMPLOYEE records are then retrieved by a series of OBTAIN WITHIN SET statements. Note that when EMPLOYEE 106 is retrieved, the end of the set is reached and the next OBTAIN statement positions the program on the owner of the set, DEPARTMENT 2000.





| | CURRENCIES: RUN UNIT, RECORD, SET, AREA | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RUN UNIT | DEPARTMENT | EMPLOYEE | DEPT-EMPLOYEE | OFFICE-EMPLOYEE | ORG-DEMO-REGION | EMP-DEMO-REGION | |
| MVC DEPTID,DEPTIN<br>@FIND CALC, REC='DEPARTMENT' | 2000 | 2000 | | 2000 | | 2000 | | |
| @OBTAIN FIRST,<br>SET='DEPT-EMPLOYEE' | 69 | 2000 | 69 | 69 | 69 | 2000 | 69 | |
| @OBTAIN NEXT,<br>SET='DEPT-EMPLOYEE' | 100 | 2000 | 100 | 100 | 100 | 2000 | 100 | |
| @OBTAIN NTH,<br>SET='DEPT-EMPLOYEE', OCCUR=FIVE | 106 | 2000 | 106 | 106 | 106 | 2000 | 106 | |
| @OBTAIN NEXT,<br>SET='DEPT-EMPLOYEE' | 2000 | 2000 | 106 | 2000 | 106 | 2000 | 106 | ERROR-STATUS OF '0307' |

The following figure illustrates special considerations relating to the retrieval of records in an area that contains multiple record types. In this example, the user wishes to sweep the EMP-DEMO-REGION area, retrieving sequentially each EMPLOYEE record and all records in the associated EMP-EXPERTISE set. The first command retrieves EMPLOYEE 119. Subsequent @OBTAIN WITHIN SET statements retrieve the associated EXPERTISE records and establish currency on EXPERTISE 03. The @FIND DBKEY statement is used to reestablish the proper position before retrieving EMPLOYEE 48. Note that if @FIND DBKEY for the employee record is not specified, an attempt to retrieve the next EMPLOYEE record in the area would return EMPLOYEE 23.



| | RUN UNIT | EMPLOYEE | EXPERTISE | EMP-EXPERTISE | EMP-DEMO-REGION |
|---|---|---|---|---|---|
| @OBTAIN FIRST, AREA='EMP-DEMO-REGION' | 119 | 119 | | 119 | 119 |
| @OBTAIN FIRST, SET='EMP-EXPERTISE' | 04 | 119 | 04 | 04 | 04 |
| @OBTAIN NEXT, SET= 'EMP-EXPERTISE' | 03 | 119 | 03 | 03 | 03 |
| @FIND CURRENT, REC='EMPLOYEE' | 119 | 119 | 03 | 119 | 119 |
| @OBTAIN NEXT, AREA= 'EMP-DEMO-REGION' | 48 | 48 | 03 | 48 | 48 |

**Status codes**

After completion of the @FIND/@OBTAIN WITHIN SET/AREA function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

This request has been serviced successfully.

**0301**

The area in which the named record participates has not been readied.

**0304**

A sequence number of zero or a variable field that contains a value of zero was specified for the named record.

**0306**

Currency has not been established for the named record, set, or area.

**0307**

The end of the set or area has been reached, or the set is empty.

**0308**

Either the named record or the named set is not in the subschema, or the named record is not defined as a member of the named set. The program has probably invoked the wrong subschema, or has misspelled the record or set name.

**0310**

The subschema specifies an access restriction that prohibits retrieval of the named record.

**0323**

The area name specified has not been included in the subschema invoked, the record name specified has not been defined in the named area, or the area name has been misspelled.

**0326**

The record cannot be found.

**0360**

A record occurrence has been encountered whose record type is not a member or owner of the set as it is defined in the subschema.

**0361**

The record cannot be stored because of broken chains in the database.

**0370**

A database file will not open properly.

When the KEEP parameter is specified as part of the @FIND/@OBTAIN statement a major code of 06 will be returned if an error occurs during the KEEP processing (see @KEEP (see page 200) in this chapter). The major code of 03 states that an error has occurred in the @FIND/@OBTAIN processing.

## @FINISH—commits changes to database and terminates run unit

The @FINISH statement commits changes made to the database through an individual run unit and terminates the run unit. No further DML retrieval or modification statements can be executed until the appropriate BINDs have been issued and the necessary areas have been readied again.

If the run unit is sharing its transaction with another database session, the run unit's changes may not be committed at the time the @FINISH statement is executed.

Note: For more information about the impact of transaction sharing, see the *Navigational DML Programming Guide*.

**Currency**

Following the successful execution of an @FINISH, all currencies are set to null. You cannot perform database access activities until you issue an @BIND/@READY sequence.

### @FINISH Syntax

```
►►── @FINISH ─────────────────────────────────────────────────────────►◄
```

### @FINISH Status Codes

The only acceptable status code returned for an @FINISH function is 0000.

## #FINISH—commits changes to the database

\The #FINISH statement commits changes made to the database through an individual run unit or through all database sessions associated with a task. A task-level finish also commits all changes made in conjunction with scratch, queue, and print activity.

If the finish applies to an individual run unit and the run unit is sharing its transaction with another database session, the run unit's changes may not be committed at the time the #FINISH statement is executed.

Note: For more information about the impact of transaction sharing, see the *Navigational DML Programming Guide*.

Run units (and SQL sessions) impacted by the #FINISH statement end, and their access to the database is terminated.

The #FINISH statement is used in both the navigational and logical record facility environments. The #FINISH TASK statement is also used in an SQL programming environment.

**Currency**

Following the successful execution of a #FINISH request, all currencies are set to null and the issuing task cannot perform database access through an impacted run unit without executing an @BIND/@READY sequence.

## #FINISH Syntax

```
►►─┬───────────┬─ #FINISH ─┬────────┬──────────────────────────────────────►◄
   └─ label ───┘           └─ TASK ─┘
```

## #FINISH Parameters

**TASK**

Commits the changes made by all scratch, queue, and print activity and all top-level run units associated with the current task. Its impact on SQL sessions associated with the task depends on whether those sessions are suspended and whether their transactions are eligible to be shared.

**More information**:

For more information about the impact of a #FINISH TASK statement on SQL sessions, see the *SQL Programming Guide*.

For more information about run units and the impact of #FINISH TASK, see the *Navigational DML Programming Guide*.

## #FINISH Status Codes

After completion of the #FINISH statement, the value in register 15 indicates the outcome of the operation. The following is a list of the Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'08'**

The request cannot be serviced due to an invalid request.

**X'14'**

The request cannot be serviced because the transaction was backed out.

**X'0C'**

The request cannot be serviced because an internal error was detected. Check the DC/UCF log file for details.

## #FREESTG—requests that the system release variable storage

The #FREESTG statement requests that the system release all or a part of a block of variable storage. The storage to be released may have been acquired with a #GETSTG request in the issuing task or by another task running on the same terminal as the issuing task. A partial release is valid only for user storage; shared storage must be freed in its entirety.

The #FREESTG request is unconditional; any runtime error will result in an abend of the issuing task.

### IDMSDB--#FREESTG

```
►►─┬─────────┬──────────────────────────────────────────────────────►
   └─ label ─┘

►── #FREESTG ─┬─ ADDR=storage-address ─┬────────────────────────────►
              └─ STGID=storage-id ─────┘

►─┬──────────────────────┬──────────────────────────────────────────►◄
  └─ ,NEWLEN=newlength ──┘
```

## #FREESTG Parameters

**ADDR=*storage-address*/STGID=*storage-id***

Specifies the storage area to be released. One of these options must be specified.

***storage-address***

Specifies the address of the storage area to be released. *Storage-address* is a register or the symbolic name of a fullword user-defined field that contains the storage area address.

***storage-id***

Specifies the 4-byte identifier of the variable storage area to be released. *Storage-id* is a register that contains the ID, the symbolic name of a user-defined field aligned on a fullword boundary that contains the ID, or the ID literal enclosed in quotation marks.

**NEWLEN=**

Specifies the number of bytes to be retained in the storage pool, indicating a partial storage release (release of only part of the area originally allocated).

***new-length***

A register that contains the number of bytes, the symbolic name of a user-defined halfword or fullword field that contains the number of bytes, or an absolute expression.

When a release is partial, the low-address portion of storage will be retained and the high-address portion released.

## #FREESTG Status Codes

The #FREESTG request is unconditional; any runtime error will result in an abend of the issuing task.

## #FREESTG Example

The following example illustrates the use of the #FREESTG statement to release part of the user storage area that is identified by the value in register 7. The number of bytes to remain in the storage area is specified in the variable field SPACE1.

```
#FREESTG STGID=(R7),NEWLEN=SPACE1
```

# @GET—transfers the contents of an accessed record occurrence

The @GET statement transfers the contents of an accessed record occurrence into program variable storage. Elements in the accessed record are moved to their respective locations in variable storage according to the subschema view of the record. The transferred elements will appear in storage at the location to which the record has been bound. (For further details, see @BIND REC (see page 104) in this chapter.)

**Currency**

The @GET statement operates only on the record that is current of run unit.

Following successful execution of an @GET statement, the accessed record is current of run unit, its area, its record type, and all sets in which it participates as owner or member.

## @GET Syntax

```
►►─── @GET ─┬─────────────────────┬──────────────────────────────────────►◄
            └─ REC=record-name ───┘
```

## @GET Parameters

***REC=record-name***

Retrieves the record that is current of run unit. If the optional REC=*record-name* clause is used, the current of run unit must be an occurrence of the named record type.

## @GET Status Codes

After completion of the @GET function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**0508**

The requested record is not in the subschema. The program has probably invoked the wrong subschema or the record name is misspelled.

**0510**

The subschema specifies an access restriction that prohibits retrieval of the named record.

**0513**

A current record of run unit has not been established or has been nullified by a previous @ERASE statement.

**0518**

The record has not been bound.

**0520**

The current record is not the same type as the named record.

**0526**

The requested record has been erased.

**0555**

An invalid length has been returned for a variable-length field.

## @GET Example

The following statement moves the EMPLOYEE record that is current of run unit into program variable storage:

```
@GET REC='EMPLOYEE'
```

## #GETIME—gets time and date from the operating system

The #GETIME statement obtains the time and date from the operating system. The system time is returned to the issuing task in binary absolute, binary formatted, packed decimal, or edited format, as specified by the task. The date is returned to the program in packed decimal format.

After completion of the #GETIME request, a user-defined register and register 1 contain the following time and date information:

- **Register n** specifies system time (if requested in binary formatted or binary absolute format) or the address of a field that contain the system time (if requested in packed or edited format). The register number (*n*) is assigned by the FORMAT parameter; if not specified, the default is register 0.

  Note: The return-time location can be defined by the FORMAT parameter as a variable field name rather than a register number; in this instance, register 0 will still contain the time value or return-time address, as described above.

- **Register 1** contains the Julian date in packed format: 0*yyydddc* (padded zero, current year relative to 1900, days in year, sign). For example, 0099365C would represent December 31, 1999. 0100001C would represent January 1, 2000.

## #GETIME Syntax

```
►►─┬───────┬──────────────────────────────────────────────────────────────►
   └ label ┘

►── #GETIME FORMAT= ──────────────────────────────────────────────────────►

►── ( ─┬─ BINABS ─┬─ , ─┬─(0) ◄────────────────┬─ ) ───────────────────►◄
       ├─ BINFMT ─┤     └ return-time-pointer ─┘
       ├─ PACK ───┤
       └─ EDIT ───┘
```

## #GETIME Parameters

**FORMAT=**

Specifies how and where the time is returned by the operating system.

**BINABS/BINFMT/PACK/EDIT**

Specifies the format of the time which is returned. The returned value indicates the elapsed time since midnight.

**BINABS**

(Binary absolute) (default); returns time as a fullword binary integer representing elapsed time since midnight in intervals of ten-thousandths of a second.

Note: BINABS returns the most precise time.

**BINFMT**

(Binary formatted); returns time as a fullword binary value which, when translated to decimal form, is formatted as: *hhmmsstttt* (hours, minutes, seconds, and ten-thousandths seconds).

**PACK**

(Packed); returns time as a 6-byte packed decimal value, formatted as: 0*hhmmsstttc* (hours, minutes, seconds, ten-thousandths seconds, and sign).

**EDIT**

(Edited); returns time as an 11-byte edited value, formatted as: *hh:mm:ss:hh* (hours, minutes, seconds, and hundredths seconds).

**(0)/*return-time***

Specifies the location to which the time is returned.

**(0)**

(Default); is the register that contains the time or points to a field that contains the time.

***return-time***

A register that contains the time (FORMAT is BINABS or BINFMT), a register that points to the time (FORMAT is PACK or EDIT), or the symbolic name of a user-defined field (FORMAT is BINABS, BINFMT, PACK, or EDIT). The required size of the field is dependent on the format requested.

## #GETIME Status Codes

The #GETIME request is unconditional; any runtime error will result in an abend of the issuing task.

## #GETIME Example

The following example of the #GETIME statement obtains the time from the operating system into the variable field TIMECK and the Julian date is returned in register 1. The time is in an 11-byte edited format; the Julian date is in packed decimal format.

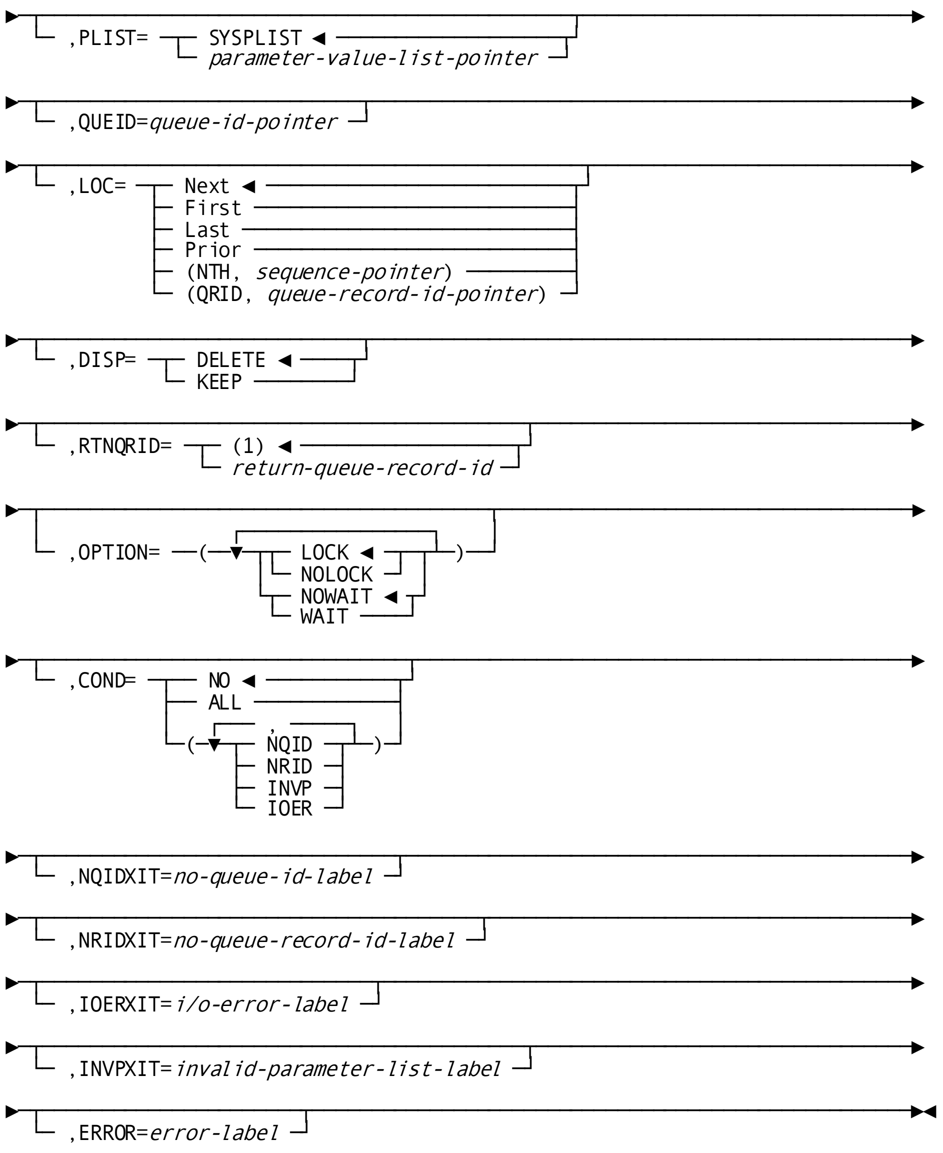
```
#GETIME FORMAT=(EDIT,TIMECK)
```

## #GETQUE—retrieves a queue record

The #GETQUE statement retrieves a queue record, places it in a storage area associated with the issuing program and optionally deletes it from the queue. If the queue record is larger than the designated storage area, the record is truncated as necessary.

## #GETQUE Syntax

```
►►─┬─────────┬──────────────────────────────────────────────────────────►
   └─ label ─┘

►── #GETQUE RECORD=return-queue-data-location-pointer ─────────────────►

►── ,RECLEN= ─┬─ queue-data-max-length ─┬──────────────────────────────►
              └─ queue-data-length ─────┘
```



## #GETQUE Parameters

**RECORD=**

Specifies the location to which the system will return the requested queue record.

**return-queue-data-location-pointer**

A register that points to the area or the symbolic name of the area.

**RECLEN=**

Specifies the length of the area defined by the RECORD parameter and, if provided in the form of a user-defined variable field name, assigns an area into which the system will place the actual length of the retrieved queue record.

**queue-data-max-length**

Specifies the length of the data area associated with the requested queue record. It is a register that contains the length or an absolute expression.

**queue-data-length**

A symbolic user-defined field, specifies a two-fullword area that is subdivided into two fullwords. The first fullword contains the length of the data area associated with the requested queue record. The system returns the actual length of the retrieved queue record to the second fullword. If the record length is provided in register notation or as an absolute expression, a two-fullword area as defined by queue- data-length will be built dynamically at runtime in the sixth and seventh fullwords of the parameter list.

**PLIST=**

Specifies the location of the seven-fullword storage area in which the system will build the #GETQUE parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system F builds the #GETQUE parameter list.

**parameter-value-list-pointer**

A register that points to the area or the symbolic name of the area.

**QUEID=**

Specifies the 1- to 16-character ID of the queue associated with the record to be retrieved.

**queue-id-pointer**

A register that points to a field that contains the ID, the symbolic name of a user-defined field that contains the ID, or the ID literal enclosed in quotation marks. If the queue ID is not specified, a null queue ID (16 blanks) is assumed.

**LOC=**

Specifies the queue record to be retrieved:

**NEXT**

(Default); retrieves the next record in the queue. If currency in the queue has not been established, NEXT is equivalent to FIRST.

**FIRST**

Retrieves the first record in the queue.

**LAST**

Retrieves the last record in the queue.

**PRIOR**

Retrieves the prior record in the queue. If currency in the queue has not been established, PRIOR is equivalent to LAST.

**(NTH,sequence)**

Retrieves the nth record in the queue as defined by sequence. Sequence is a register that points to a field that contains the record sequence number (n), the symbolic name of a user-defined field that contains the number, or an absolute expression. (Within each queue, records are assigned numbers beginning with 1, not 0.)

**(QRID,queue-record-id)**

Retrieves the record identified by queue-record-id. Queue-record-id is a register that points to a field that contains the queue record id, the symbolic name of a user-defined field that contains the id, or an absolute expression.

**DISP=**

Specifies the disposition of the queue record after it is passed to the requesting program.

**DELETE**

(Default); deletes the record from the queue. If DELETE is specified and the record is truncated, some data may be lost.

**KEEP**

Keeps the record in the queue.

**RTNQRID=**

Specifies the location in the program to which the system will return the system-assigned ID of the retrieved queue record. The returned ID can be saved and used to retrieve or delete the queue record.

**(1)**

(Default); the register to which the system will return the queue record ID.

**return-queue-record-id**

A register or the symbolic name of a fullword user-defined field to which the system will return the queue record ID.

**OPTION=**

Specifies whether to retain a lock on the current queue record and whether the issuing task suspends execution if the requested record cannot be accessed in the queue:

**LOCK/NOLOCK**

These parameters have been non-functional since CA IDMS Release 12.0. They are included as parameters for release compatibility. Queue record locking is performed as part of the standard database locking routines since CA IDMS Release 12.0.

**NOWAIT**

Continues task execution in the event of a nonexistent queue. The system returns a value of X'0C' to register 15 in the event that the requested queue does not currently exist.

**WAIT**

Suspends task execution until the requested queue exists.

**COND=**

Specifies whether the #GETQUE is conditional and under what conditions control should be returned to the issuing program:

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the request cannot be serviced for any of the reasons listed below.

**condition**

Specifies conditions under which the system returns control to the program. Multiple conditions must be enclosed in parentheses and separated by commas.

**NQID**

The queue ID cannot be found.

**NRID**

The queue record cannot be found.

**IOER**

An I/O error occurs while processing the request.

**INVP**

The parameter list built for the #GETQUE is invalid. A list of conditions must be enclosed in parentheses. If multiple conditions are specified, each is separated from the previous one by a comma.

**NQIDXIT=no-queue-id-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETQUE request cannot be serviced because the header record identified by the QUEID parameter cannot be found.

**NRIDXIT=no-queue-record-id-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETQUE request cannot be serviced because the queue record ID cannot be found.

**IOERXIT=i/o-error-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETQUE parameter cannot be serviced because of an I/O error.

**INVPXIT=invalid-parameter-list-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETQUE cannot be serviced because of an invalid parameter in the parameter list.

**ERROR=error-label**

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

## #GETQUE Status Codes

By default, the #GETQUE request is unconditional; any runtime error will result in an abend of the issuing task. The issuing program can request return of control with the COND parameter to avoid an abend.

After completion of the #GETQUE function, the value in register 15 indicates the outcome of the operation. The following is a list of the Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'04'**

The request cannot be serviced due to an invalid parameter list.

**X'08'**

The request cannot be serviced because the requested queue header record (identified by QUEID) cannot be found.

**X'0C'**

The request cannot be serviced because the requested queue record cannot be found.

**X'18'**

The program storage area specified for return of the queue record is too small; the returned record has been truncated to fit the available storage.

**X'1C'**

A database error occurred during queue processing. A common cause is a DBKEY deadlock. For a PUT QUEUE operation, this code can also mean that the queue upper limit has been reached.

If a database error has occurred, there are usually be other messages in the CA-IDMS/DC/UCF log indicating a problem encountered in RHDCRUAL, the internal Run Unit Manager. If a deadlock has occurred, messages DC001000 and DC001002 are also produced.

If an I/O error occurs while processing a #GETQUE request, the system will return the address of the IDMS communications block to register 1. If no error occurs during processing, a user-defined register, as assigned by the RTNQRID parameter, will contain the queue record ID (QRID) of the retrieved queue record.

## #GETQUE Example

The example of the #GETQUE statement shown below performs

the following functions:

- Specifies location QREC5 as the area in program variable storage to receive the requested queue record
- Specifies the length of area QREC5 in register 6
- Uses the default location to build the parameter list, SYSPLIST
- Specifies that register 7 will hold the address of the field that contains the ID of the queue associated with the record to be retrieved
- Specifies the next record (in regard to queue currency) in the queue as the record to be retrieved
- Specifies that the record will not be deleted from the queue after it has been passed to the requesting program
- Uses the register 1 default to receive the system-assigned ID of the retrieved scratch record
- Specifies the WAIT option to suspend task execution until the requested queue record is available
- Specifies that this request is not conditional; any runtime error will result in an abend of the issuing task

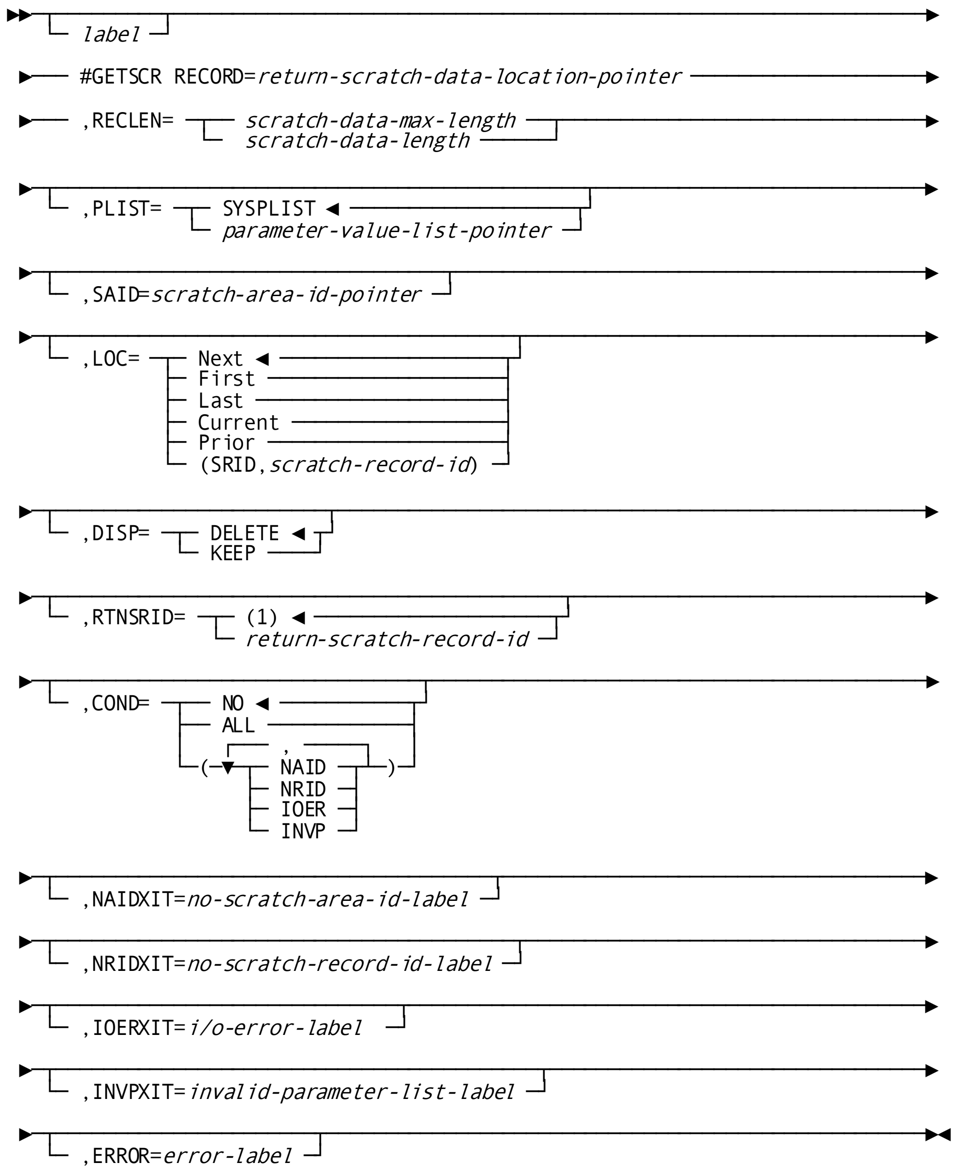
```
#GETQUE RECORD=QREC5,RECLEN=(6),QUEID=(7),LOC=NEXT,DISP=KEEP, _
OPTION=WAIT,COND=NO
```

# #GETSCR—retrieves a scratch record

The #GETSCR statement retrieves a scratch record and places it in a storage area associated with the issuing program. The storage area must already be allocated to the requesting task; no implicit #GETSTG function is performed during the #GETSCR operation. If the scratch record is larger than the designated storage area, the record is truncated as necessary.

By default, the #GETSCR request is unconditional; any runtime error will result in an abend of the issuing task. The issuing program can request return of control with the COND parameter to avoid an abend.

## #GETSCR Syntax


label
#GETSCR RECORD=return-scratch-data-location-pointer
,RECLEN= scratch-data-max-length
scratch-data-length
,PLIST= SYSPLIST ◄
parameter-value-list-pointer
,SAID=scratch-area-id-pointer
,LOC= Next ◄
First
Last
Current
Prior
(SRID,scratch-record-id)
,DISP= DELETE ◄
KEEP
,RTNSRID= (1) ◄
return-scratch-record-id
,COND= NO ◄
ALL
( , NAID NRID IOER INVP )
,NAIDXIT=no-scratch-area-id-label
,NRIDXIT=no-scratch-record-id-label
,IOERXIT=i/o-error-label
,INVPXIT=invalid-parameter-list-label
,ERROR=error-label

## #GETSCR Parameters

**RECORD=**

Specifies the location to which the system will return the scratch record.

**record-scratch-data-location-pointer**

A register that points to the variable storage area or the user-defined symbolic name of the area.

**RECLEN=**

Specifies the length of the area defined by the RECORD parameter and, if provided in the form of a user-defined variable field, assigns an area into which the system will place the actual length of the returned data.

**scratch-data-max-length**

Specifies the length of the data area associated with the requested scratch record. It is a register that contains the length or an absolute expression.

**scratch-data-length**

A symbolic user-defined field, specifies an area which is subdivided into two fullwords. The first fullword contains the length of the data area associated with the requested scratch record. The system returns the actual length of the requested scratch record to the second. If the record has been scratch-data-length will contain the length of the scratch record. If the record length is provided in register notation or as an absolute expression, an area composed of two fullwords, as defined by scratch-data-length, will be built dynamically at runtime in the sixth and seventh fullwords of the parameter list.

**PLIST=**

Specifies the location of the seven-fullword storage area in which the system will build the #GETSCR parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system will build the #GETSCR parameter list.

**parameter-list-pointer**

A register that points to the area in which the system will build the #GETSCR parameter list or the symbolic name of that area.

**SAID=**

Specifies the 1- to 8-character ID of the scratch area associated with the record being retrieved.

**scratch-area-id-pointer**

A register that points to a field that contains the id, the symbolic name of a user-defined field that contains the ID, or the ID literal enclosed in quotation marks. If the SAID parameter is not specified, a null scratch area ID of 8 blanks is assumed.

**LOC=**

Specifies the scratch record to be retrieved.

**NEXT**

(Default); retrieves the next record in the scratch area.

**FIRST**

Retrieves the first record in the scratch area. (Records are always stored in ascending order by scratch record id.)

**LAST**

Retrieves the last record in the scratch area.

**CURRENT**

Retrieves the current record; that is, that record most recently referenced by another scratch function.

**PRIOR**

Retrieves the prior record in the scratch area. If currency in the scratch area has not been established, PRIOR is equivalent to LAST.

**(SRID,scratch-record-id)**

Retrieves the scratch record identified by scratch-record-id. Scratch-record-id is a register that points to the 4-byte scratch record id, the symbolic name of a user-defined field that contains the id, or an absolute expression of the id.

**DISP=**

Specifies whether the scratch record is to be kept after it is passed to the requesting program.

**DELETE**

(Default); deletes the record from the scratch area. If DELETE is specified and the record has been truncated, some data may be lost. To maintain currency following a DELETE request, the system saves the next and prior pointers of the deleted record.

**KEEP**

Keeps the record in the scratch area.

**RTNSRID=**

Specifies the location to which the system will return the scratch record ID of the retrieved record.

**(1)**

(Default); is the register into which the system will place the ID of the scratch record.

**return-scratch-record-id**

A register or the symbolic name of a fullword user-defined field to which the system will return the ID of the retrieved scratch record.

**COND=**

Specifies whether this #GETSCR is conditional and under what conditions control should be returned to the issuing program:

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the request cannot be serviced for any of the reasons listed below.

**condition**

Specifies conditions under which the system returns control to the issuing task. Multiple conditions must be included in parentheses and separated by commas.

**NAID**

The scratch area ID cannot be found.

**NRID**

The scratch record ID cannot be found.

**IOER**

An I/O error occurs while processing the retrieval.

**INVP**

The parameter list built for the #GETSCR is invalid.

**NAIDXIT=no-scratch-area-id-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETSCR cannot be serviced because the scratch area ID cannot be found.

**NRIDXIT=no-scratch-record-id-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETSCR cannot be serviced because the scratch area record ID cannot be found.

**IOERXIT=i/o-error-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETSCR cannot be serviced because of an I/O error.

**INVPXIT=invalid-parameter-list-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETSCR request cannot be serviced because of an invalid parameter in the parameter list.

**ERROR=error-label**

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

## #GETSCR Status Codes

After completion of the #GETSCR function, the value in register 15 indicates the outcome of the operation.

**X'00'**

The request has been serviced successfully

**X'04'**

The request cannot be serviced due to an invalid parameter list.

**X'08'**

The request cannot be serviced because the requested scratch area ID (SAID) cannot be found

**X'0C'**

The request cannot be serviced because the requested scratch record ID (SRID) cannot be found in the named SAID.

**X'18'**

The request cannot be serviced because the program storage area specified for return of the scratch record is too small; the returned record has been truncated to fit the available space.

**X'1C'**

The request cannot be serviced due to an I/O error during processing.

If an I/O error occurs while processing a #GETSCR request, the system will return the address of the IDMS communications block to register 1. If no error occurs during processing, a user-defined register, assigned by the RTNSRID parameter, will contain the scratch record ID of the obtained record.

## #GETSCR Example

The example of the #GETSCR statement shown below performs the following functions:

- Specifies location SREC5 as the area in program variable storage to receive the requested scratch record.
- Specifies the length of area SREC5 in user-defined field SCRLENG.
- Uses the default location to build the parameter list, SYSPLIST.
- Specifies the literal SCR3 as the ID of the scratch area associated with the record to be retrieved.
- Specifies the first record in the scratch area as the record to be retrieved.
- Specifies that the record will be deleted from the scratch area after it has been passed to the requesting program.
- Specifies that register 4 will receive the system-assigned ID of the retrieved scratch record.
- Specifies that this request is conditional. If the scratch record id cannot be found control will be returned to the routine labeled NORECRTN.

```
#GETSCR RECORD=SREC5,RECLEN=SCRLENG,SAID='SCR3',LOC=FIRST, _
DISP=DELETE,COND=NRID,NRIDXIT=NORECRTN
```

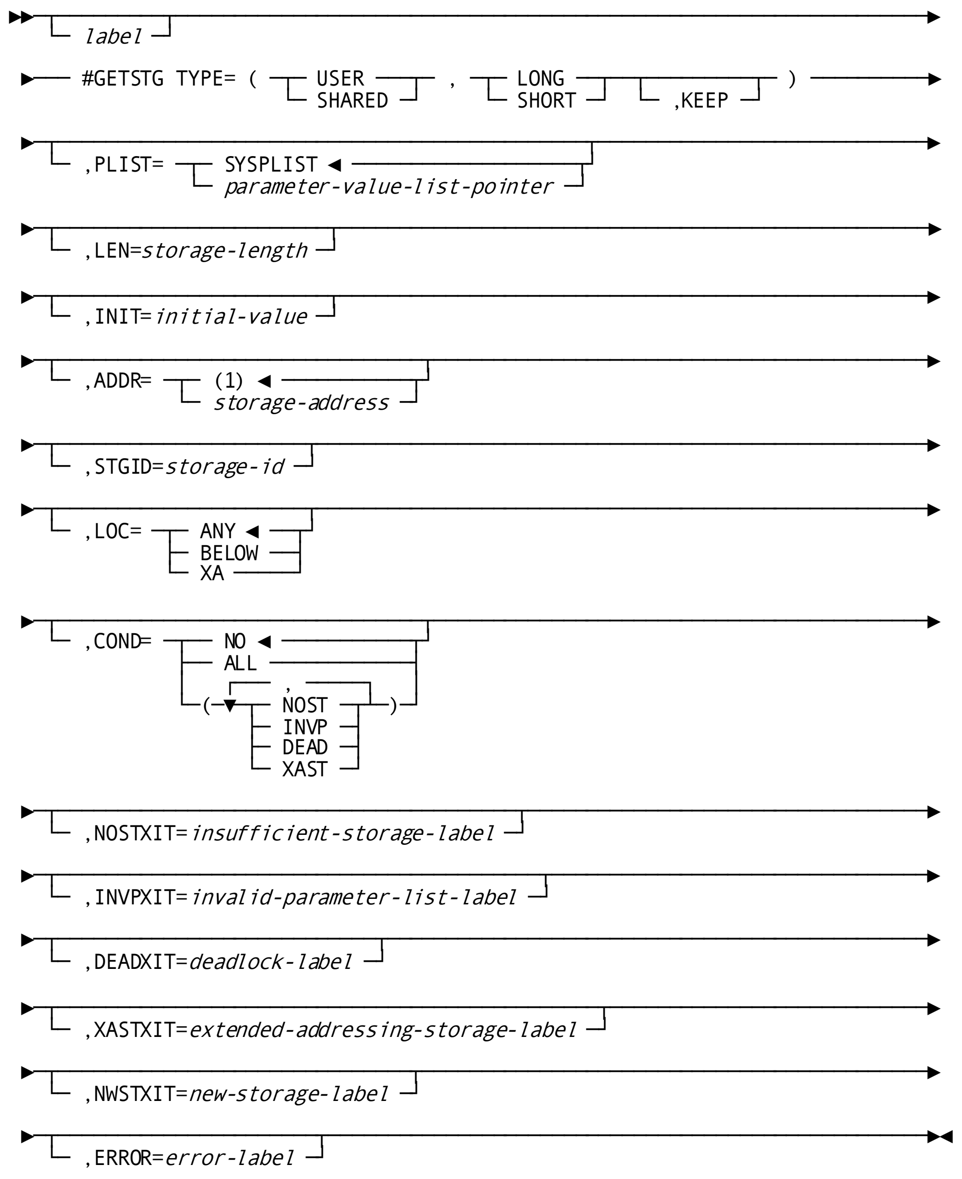
# #GETSTG—acquires variable storage from a storage pool

The #GETSTG statement acquires variable storage from a storage pool or obtains the address of a previously acquired storage area. Once acquired, the storage is available for use:

- By the issuing task only (user storage)
- By subsequent tasks running on the same logical terminal (user-kept storage)
- By all tasks in the system (shared or shared-kept storage)

Storage availability is governed by #GETSTG parameter specifications. The value stored in a user-defined register assigned by the ADDR parameter contains the address of acquired storage.

## #GETSTG Syntax



## #GETSTG Parameters

**TYPE=**

Required for all requests for storage, specifies three subparameters. Specified subparameters must be enclosed in parentheses.

**USER/SHARED**

Specifies whether access to the storage is to be restricted to the issuing task or is to be available to all tasks in the system.

**USER**

Specifies that access to the storage area is to be restricted to the issuing task or, if KEEP is specified, to subsequent tasks executing on the same terminal.

**Note:** During system generation, a program defined with the NOPROTECT option can access any storage area in the system, including an area associated exclusively with another task. Thus, the USER attribute may not protect the storage area being acquired. However, storage areas can be protected on a system-wide or program-by-program basis during system generation and by the modes specified when storage is allocated.

**SHARED**

Specifies that any task in the DC/UCF system can access and modify the acquired storage. Each task must establish addressability to the storage area by explicitly issuing a #GETSTG request.

**LONG/SHORT**

Specifies whether the system should allocate the storage from the bottom or the top of the storage pool.

**LONG**

Specifies that storage, used long-term, is allocated from the bottom of the storage pool.

**SHORT**

Specifies that storage, used short-term, is allocated from the top of the storage pool. An incorrect LONG/SHORT specification will not affect normal program execution; however, it may affect the overall performance of the DC/UCF system.

**Note**: For more information about the use of the LONG/SHORT option, see the *CA IDMS Navigational DML Programming Guide.*

**KEEP**

Optionally specifies whether the storage area will be used by subsequent tasks executing on the same logical terminal. When KEEP is specified, the storage area can be accessed by subsequent tasks; otherwise the storage area cannot be accessed by subsequent tasks.

**Note:** For more information about the KEEP parameter, see the *CA IDMS Navigational DML Programming Guide*.

**PLIST=**

Specifies whether the six-fullword #GETSTG parameter list will be built inline or in a variable storage area and, if built in a variable storage area, identifies the location of that area.

**SYSPLIST**

(Default); builds the list in a variable storage area identified by the symbolic name SYSPLIST.

*****

Builds the list inline. The generated parameter list will be reentrant; that is, no generated code will modify it. If PLIST=* is specified, other parameters of the #GETSTG statement cannot be identified with register notation.

**parameter-list**

Builds the list in a variable storage area associated with the task. Parameter-list is a register which points to the area or the symbolic name of that area.

**LEN=**

Specifies the size, in bytes, of a new storage area.

**storage-length**

A register or the symbolic name of a user-defined halfword or fullword field that contains the number of bytes, or an absolute expression.

Note: If the parameter list is being generated inline (PLIST=*), the LEN parameter must specify the symbolic name of a fullword field or an absolute expression; register notation and a halfword variable field name are invalid.

**INIT=**

Specifies an initial value for a new storage area.

**initial-value**

An absolute expression of the initial value. Each byte of the acquired storage area is initialized to the specified value.

**ADDR=**

Specifies the address of the acquired or previously acquired storage:

**(1)**

(Default); is a register or the symbolic name of a fullword user-defined field to which the system will return the address of the acquired storage.

**storage-address**

A register or the symbolic name of a fullword user-defined field to which the system returns the address of the acquired storage.

**STGID=**

Specifies the 4-character ID associated with the storage area. The STGID parameter must be specified with #GETSTG requests for previously allocated storage areas or areas to be reallocated.

**storage-id**

A register that contains the id, the symbolic name of a 4-byte user-defined field which is aligned on a fullword boundary and contains the ID, or the ID literal enclosed in single quotation marks.

**Note:** If the parameter list is being generated inline, the STGID parameter must specify the symbolic name of a variable field or a literal enclosed in quotation marks; register notation is invalid. When using the STGID option to request the address of an existing storage area, the #GETSTG statement must specify the same USER/SHARED option as the original #GETSTG request issued by the task to acquire the area.

**Note:** All storage ids owned by a task must be unique. While more than one variable storage area with the same storage ID can exist (for example, one shared and the other user) only one such area can be owned by a task at a time.

**LOC=**

Indicates where the system allocates storage.

**ANY**

(Default); indicates that storage can be allocated anywhere in the region.

**BELOW**

Requests that the system allocate storage below the 16-megabyte line.

**XA**

Requests that the system allocate storage above the 16-megabyte line. This option is ignored if the system has no XA storage pools defined or if it is not XA-enabled.

**COND=**

Specifies whether this #GETSTG statement is conditional and under what condition control should be returned to the issuing program:

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the request cannot be serviced for any of the reasons listed below.

**condition**

Specifies conditions under which the system returns control to the issuing task. Multiple conditions must be enclosed in parentheses and separated by commas.

**NOST**

Available space in the storage pool is insufficient to satisfy the request. Do not wait for additional storage to become available.

**INVP**

The parameter list built for the #GETSTG is invalid.

**DEAD**

The available space in the storage pool is insufficient to satisfy the request and if to wait would cause a deadlock

**XAST**

Allocated storage above the 16-megabyte line cannot be addressed by the 24-bit task.

**NOSTXIT=insufficient-storage-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETSTG cannot be serviced because the available storage is insufficient to satisfy the request.

**INVPXIT=invalid-parameter-list-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETSTG cannot be serviced because of an invalid parameter in the parameter list.

**DEADXIT=deadlock-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETSTG cannot be serviced because the available storage is insufficient to satisfy the request, and if to wait would cause a deadlock.

**NWSTXIT=new-storage-label**

Specifies the symbolic name of the routine to which control should be returned if the #GETSTG request names a STGID that does not exist in the system (TYPE=SHARED) or in the task (TYPE=USER).

**XASTXIT=extended-addressing-storage-label**

Specifies the symbolic name of the routine to which control is returned if the allocated storage above the 16-megabyte line cannot be addressed by the 24-bit task.

**ERROR=error-label**

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

## #GETSTG Status Codes

By default, the #GETSTG request is unconditional. Error conditions that can occur are:

- A short-on-storage condition, caused when the amount of storage in the storage pool is inadequate to accommodate the request, will result in a delay until sufficient storage becomes available (unless such a wait would cause a deadlock)
- Any abnormal condition will result in an abend. Conditions in this category include the following:
  - I/O error
  - A wait on storage (default action resulting from the short-on-storage condition) would result in a deadlock

The issuing program can request return of control with the COND to avoid a delay or an abend.

After completion of the #GETSTG request, the value in register 15 indicates the outcome of the operation:

| Register 15 Value | Meaning |
|---|---|
| X'00' | The request has been serviced successfully. |
| X'04' | The request has specified a storage ID which did not previously exist; the indicated space has been allocated. |
| X'08' | The request cannot be serviced due to insufficient storage in the storage pool. |
| X'0C' | The request cannot be serviced due to an invalid parameter list. |
| X'10' | The requested storage cannot be allocated immediately (insufficient storage), and to wait would cause a deadlock. |
| X'18' | Allocated XA storage cannot be accessed by a 24-bit task. This situation occurs if storage is requested by STGID and the storage was initially allocated by an XA task. |

## #GETSTG Example

The example of the #GETSTG statement shown below performs the following functions:

- Specifies that the requested storage area is to be shared by any task in the DC/UCF system, that it will contain short-term storage allocated from the top of the storage pool, and that it will not be available for use by subsequent tasks
- Builds the parameter list, SYSPLIST (default), in the variable storage area

- Specifies the length of the new storage area in register 2
- Specifies that every byte in the storage area be initialized to blanks
- Uses register 1 (default) to receive the address of the acquired storage from the system
- Specifies the ID of the storage area in register 9
- Specifies that control will be returned to the routine labeled NOSTGRTN if the amount of available storage is insufficient to satisfy the request, otherwise, any runtime error will result in an abend of the issuing task

```
#GETSTG TYPE=(SHARED,SHORT),LEN=(2),INIT=' ',STGID=(9), _
COND=NOST,NOSTXIT=NOSTGRTN
```

## @IF—tests for the presence of member record occurrences

The @IF statement allows you to test for the presence of member record occurrences in a set or to determine the membership status of a record occurrence in a specified set; once the set has been evaluated, the @IF statement specifies further action based on the outcome of the evaluation. For example, you might use an @IF statement to determine whether a set occurrence is empty and, if it is empty, to erase the owner record.

**Note: Native VSAM users**—This statement is not allowed for sets defined with member records that are stored in native VSAM data sets.

Each @IF statement contains a conditional phrase and a branch statement. When an @IF is issued, the DML precompiler first generates a call to CA IDMS/DB to execute the conditional phrase. CA IDMS/DB tests for a status code of 0000 or 1601, as requested in the conditional phrase; the results of the test determine whether or not the branch statement is executed.

**Currency**

Depending on its format, the @IF statement uses set or run-unit currency. The set occurrence of an @IF statement is determined by the current record of the named set; the named record occurrence is the record that is current of run unit.

Currency is not updated after execution of the @IF statement.

## @IF Syntax

```
►►─── @IF SET=set-name ───────────────────────────────────────────►

►─┬─ MEMBER= ─┬─┬─ YES ─┬─────────────────────────────────────────►
  └─ EMPTY= ──┘ └─ NO ──┘

►─── ,GOTO=branch-location ───────────────────────────────────────►

►─┬──────────────────────────────────┬────────────────────────────►◄
  └─ ,ERRSTAT=error-status-location ─┘
```

## @IF Parameters

**SET=set-name**

Identifies the set that is to be tested for existing member record occurrences. Set-name must specify a set included in the subschema.

**MEMBER=**

Determines whether the current record of run unit participates as a member in any occurrence of the named set and, depending on the outcome of the evaluation, executes the branch statement.

**YES**

Specifies that the branch statement is executed only if the record is a member of the set (that is, ERRSTAT is 0000).

**NO**

Specifies that the branch statement is executed only if the named record is not a member of the named set (that is, ERRSTAT is 1601).

**EMPTY=**

Evaluates the current occurrence of the named set for the presence of member record occurrences and, depending on the outcome of the evaluation, executes the branch statement.

**YES**

Specifies that the branch statement is executed only if the set is empty (that is, ERRSTAT is 0000).

**NO**

Specifies that the branch statement is executed only if the specified set has one or more member records (that is, ERRSTAT is 1601).

**GOTO=branch-location**

Identifies the next statement in the program be executed. Branch-location must be a statement label; register notation is not supported for this parameter.

**ERRSTAT=status-location**

Specifies the name of the status field in the IDMS communications block. If the status field is other than ERRSTAT, this clause is required. Status-location must be a statement label; register notation is not supported for this parameter

## @IF Status Codes

After completion of the @IF function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation:

| Status Code | Meaning |
|---|---|
| 0000 | The set is empty, or the current record of run unit is a member of the set. |
| 1601 | The set is not empty, or the current record of run unit is not a member of the set. |
| 1606 | Currency has not been established for the specified set. |
| 1608 | An invalid set name has been specified, or the current record of run unit is not a member of the named set. A misspelled set name can account for this message. |
| 1613 | A current record of run unit has not been established or has been nullified by a preceding @ERASE statement. |

## @IF Example

The following examples illustrate two uses of the @IF statement.

In the first example, the @IF statement tests the DEPT-EMPLOYEE set for existing EMPLOYEE members and, if no occurrences of the EMPLOYEE record are found (that is, ERRSTAT is 0000), moves a message to that effect to location EMPLSWS.

If the current occurrence of the DEPT-EMPLOYEE set contains one or more occurrences of the EMPLOYEE record (that is, ERRSTAT is 1601), the GOTO clause is ignored and the next statement in the program is executed.

```
         @IF SET='DEPT-EMPLOYEE',EMPTY=YES, _
         GOTO=NOEMPL
         .
         .
NOEMPL EQU _
         MVC EMPLSWS,=CL2_'NO EMPLOYEES IN SET'
```

In this next example, the @IF statement is used to verify that the EMPLOYEE record that is current of run unit is not a member of the current occurrence of the OFFICE-EMPLOYEE set before code is executed to connect the EMPLOYEE record to that set.

If the EMPLOYEE record is not a member of OFFICE-EMPLOYEE (that is, ERRSTAT is 1601), the program branches to the LINKSET paragraph. If the EMPLOYEE record is already a member of the OFFICE-EMPLOYEE set (that is, ERRSTAT is 0000), the GOTO clause is ignored and the next statement in the program is executed.

```
@IF SET='OFFICE-EMPLOYEE',MEMBER=NO,GOTO=LINKSET
```

## @KEEP—places an explicit shared or exclusive lock on a record

The @KEEP statement places an explicit shared or exclusive lock on a record that is current of run unit, record, set, or area. Explicit record locks are used to maintain record locks that would otherwise be released following a change in currency:

- **Explicit shared**—Other run units can retrieve the locked record but cannot update it as long as the lock is in effect. Any number of concurrently executing run units can place a shared lock on a record; however, no run unit can place a shared lock on a record on which another run unit has placed an exclusive lock.
- **Explicit exclusive**—No other run unit can access the record as long as the lock is in effect. Only one run unit at a time can place an exclusive lock on a record; that run unit has exclusive control of the record. In order for a run unit to place an exclusive lock or a record, that record cannot hold either an exclusive or a shared lock assigned by any other run unit.

Locks placed on records by the @KEEP function are maintained for the duration of the recovery unit or until explicitly released by means of the @COMMIT verb.

**Currency**

Currency on run unit, record, set, or area must be established before execution of the @KEEP statement.

Currency is not updated after execution of the @KEEP statement.

## @KEEP Syntax

```
►►─── @KEEP ─┬─ EXCLUSIVE ─┬─ , ─┬─ CURRENT ──────────────┬──────────────────►◄
             └─ SHARED ────┘     ├─ REC=record-name ─┤
                                 ├─ SET=set-name ────┤
                                 └─ AREA=area-name ──┘
```

## @KEEP Parameters

**EXCLUSIVE/SHARED**

Places an exclusive or shared lock on a current record.

**CURRENT/REC=**

Specifies which record to lock.

**CURRENT**

Specifies the current record of run unit.

**REC=record-name**

Specifies the current occurrence of the named record type.

**SET=set-name**

Specifies the current occurrence of the named set type.

**AREA=area-name**

Specifies the current occurrence of the named area

## @KEEP Status Codes

After completion of the @KEEP function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation:

| Status Code | Meaning |
|---|---|
| 0000 | This request has been serviced successfully. |
| 0606 | Currency has not been established for the named record, set, or area. |
| 0608 | The named record, set, or area is not in the subschema, or the current record of run unit is not a member of the named set or is misspelled. |

| | |
|---|---|
| 0610 | The program's subschema specifies an access restriction that prohibits execution of the @KEEP function. |
| 0623 | The named area is not in the subschema or has been misspelled. |
| 0626 | The record to be kept has been erased. |
| 0629 | Deadlock occurred during locking of target record. |

### @KEEP Example

The following example of the @KEEP statement places an exclusive lock on the current record occurrence of the set OFFICE-EMPLOYEE:

```
@KEEP EXCLUSIVE,SET='OFFICE-EMPLOYEE'
```

The currency of the set for this example would have to be established before this statement can be executed.

## #KEEP—establishes long-term record locks

The #KEEP statement is used in DC/UCF pseudo-conversational transactions to establish long-term record locks and to monitor access to records between tasks. Long-term database locks can be shared or exclusive:

- **Long-term shared locks** allow other run units to access the locked record but prevent run units from updating the record as long as the lock is maintained.
- **Long-term exclusive locks** prevent other run units from accessing the locked record. However, run units executing on the logical terminal associated with a task that establishes a long-term exclusive lock are not restricted from accessing the locked record. Therefore, subsequent tasks in a transaction can access the locked record and complete the database processing required by the transaction.

If a record has been locked with a #KEEP request, restrictions may exist on the type of lock that can be placed on that record by other run units, based on existing locks and whether the requesting run unit is executing on the same logical terminal as the run unit that originally placed the lock on the record. The following table illustrates these restrictions.

| Type of lock in effect | Type of lock allowed for other run units | Type of lock disallowed for other run units |
|---|---|---|
| Shared | Shared and longterm shared | Exclusive and longterm exclusive |

| Type of lock in effect | Type of lock allowed for other run units | Type of lock disallowed for other run units |
|---|---|---|
| Exclusive | None | Shared, exclusive, longterm, shared, and longterm exclusive |
| Longterm shared | For all run units: shared and long term shared For run units on the same terminal: exclusive and longterm exclusive | For run units on other terminals: exclusive and longterm exclusive |
| Longterm exclusive | For run units on the same terminal: shared exclusive, longterm shared, and longterm exclusive | For run units on other terminals: shared exclusive, longterm shared, longterm exclusive |

Tasks can monitor database activity associated with a specified record during a pseudo-converse and, if desired, can place a long-term lock on the record being monitored. A subsequent task can then make inquiries about that database activity for the record and take the appropriate action.

The system maintains information on database activity using five-bit flags, each of which is either turned on (binary 1) or turned off (binary 0). This information is returned from the system to the low-order byte of register 0 as a numeric value. The bit assignments, the corresponding numeric value returned to the program, and a description of the associated database activity follows:

**X'10'**

The record has been physically deleted.

**X'08'**

The record has been logically deleted.

**X'04'**

The record's prefix has been modified, that is, a set operation (for example, @CONNECT or @DISCON) occurred involving the record.

**X'02'**

The record's data has been modified.

**X'01'**

The record has been obtained.

Any combination of these bits may be set. To determine the action or combination of actions that has occurred, you can compare the numeric value returned to the program in register 0 with an appropriate constant; for example:

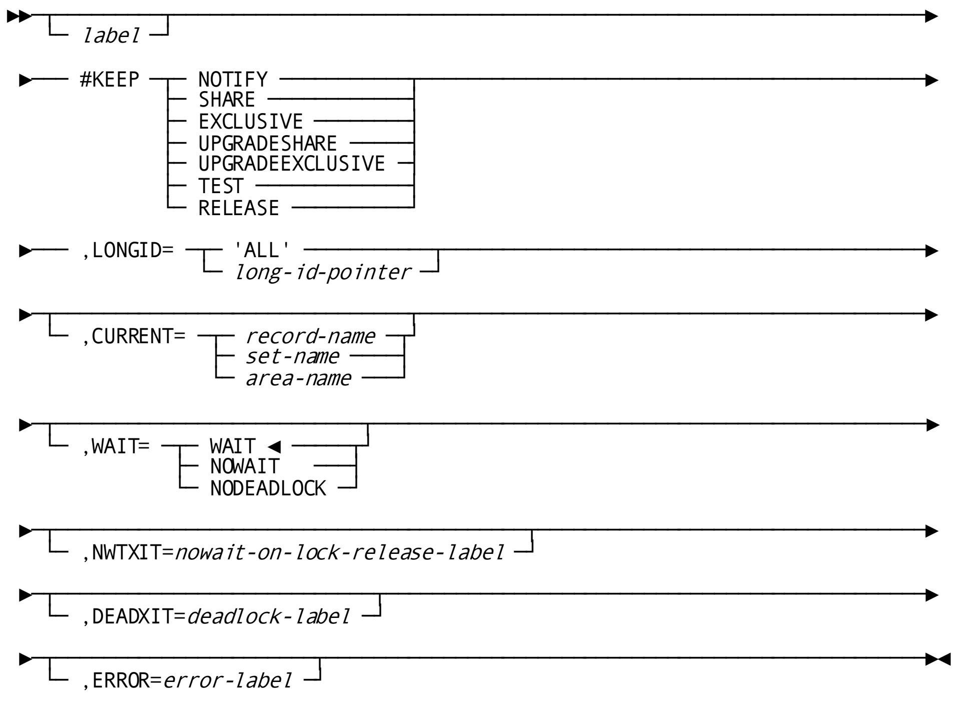
- If the returned value is 0, no database activity occurred for the monitored record.
- If the returned value is 2, the data in the record was modified.
- If the returned value is 3, the record has been obtained and modified.
- If the returned value is 8 or greater, the record was deleted.

The maximum possible value is 31 (X'1F'), indicating that all the above actions occurred for the monitored record. The example of the #KEEP statement, shown later in this topic, illustrates a test for the value of the five bit flags returned by the system to the low-order byte of register 0.

You may prefer to monitor database activity across a pseudo-converse rather than to set long-term locks. Long-term locks can prevent access to a record by other run units for an undesirably long time if, during a pseudo-converse, the terminal operator fails to enter a response. Monitoring does not restrict access to database records, sets, or areas by other run units; however, it does enable a program to test a record for alterations made by other run units. When long-term locks are used, it may be desirable to release those locks at specified timeout intervals.

**Note:** For more information about the use of timeout intervals, see the *System Generation Guide*.

## #KEEP Syntax



## #KEEP Parameters

**NOTIFY/SHARE/EXCLUSIVE/UPGRADESHARE/UPGRADEEXCLUSIVE/TEST/RELEASE**

Specifies the type of record lock or monitoring.

**NOTIFY**

Requests that the system monitor database activity associated with the current record type, set, or area specified in the CURRENT parameter, described following. When NOTIFY is specified, the system initializes register 0 to contain information on database activity for the specified record. Only the low-order byte of register 0 will actually contain the value of the five bit flags used to monitor database activity of the specified record.

**SHARE**

Specifies that the current occurrence of the record type, set, or area specified in the CURRENT parameter, described below, will receive a long-term shared lock.

**EXCLUSIVE**

Specifies that the current occurrence of the record type, set, or area specified in the CURRENT parameter, described below, will receive a long-term exclusive lock.

**UPGRADESHARE**

Upgrades a previous #KEEP NOTIFY request by placing a shared long-term lock on the record identified by the LONGID parameter, described below.

**UPGRADEEXCLUSIVE**

Upgrades a previous #KEEP NOTIFY request by placing an exclusive long-term lock on the record identified by the LONGID parameter, described below.

**TEST**

Requests that the system return information on database activity associated with the record identified by the LONGID parameter of a previously issued #KEEP NOTIFY statement. The system returns the information to the low-order byte of register 0 as a numeric value.

The TEST request must specify a long-term lock ID that matches the long-term lock ID specified in a previous #KEEP NOTIFY request.

**RELEASE**

Releases the long-term lock for the record identified by the LONGID parameter, described below. RELEASE also releases the statistics block allocated by a previous #KEEP NOTIFY request.

**LONGID=**

Specifies either the record locks to be upgraded or the records for which information about database activity is desired.

**'ALL'**

(#KEEP RELEASE requests only); requests that the system release all long-term locks kept for the logical terminal associated with the current task.

**long-id-pointer**

Specifies the 1- to 16-character identifier that will be used by subsequent #KEEP requests to upgrade a long-term lock or to make inquiries about database activity associated with the specified record. Long-id is a register that contains the address of the long-term id, the symbolic name of a user-defined field that contains the long-term id, or an absolute expression.

**CURRENT=record-name/set-name/area-name**

Specifies the record type, set, or area for which the system will monitor database activity or assign a long-term shared or exclusive lock. One of the keywords NOTIFY, SHARE, or EXCLUSIVE must also be specified with the CURRENT parameter. The value of the CURRENT parameter can be a register or the symbolic name of a user-defined field that contains the record name, set name, or area name or the name itself enclosed in quotation marks.

**WAIT=**

(#KEEP SHARE/EXCLUSIVE/UPGRADESHARE/ UPGRADEEXCLUSIVE requests only); specifies whether the issuing task is to wait if the requested lock cannot be set immediately because of an existing lock on the named

record.

**WAIT**

(Default); Requests that the system wait for the existing lock to be released in order to set the requested lock. If the wait would cause a deadlock, the system terminates the issuing task abnormally.

**NOWAIT**

Requests that the system not wait for the existing lock to be released.

**NODEADLOCK**

Requests that the system wait for the existing lock to be released, unless to do so would cause a deadlock. If the wait would cause a deadlock, the system returns control to the issuing task.

**NWTXIT=nowait-on-lock-release-label**

Specifies the symbolic name of a routine to which control should be returned if the #KEEP request that specified the NOWAIT option cannot be serviced because the requested lock cannot be set immediately.

**DEADXIT=deadlock-label**

(#KEEP requests specifying WAIT only);

Specifies the symbolic name of a routine to which control is returned if the requested lock cannot be set immediately, and if to wait would cause a deadlock.

**ERROR=error-label**

Specifies the symbolic name of a routine to which control should be returned if a condition occurs for which no other exit routine was coded.

## #KEEP Status Codes

After completion of the #KEEP request, the value in register 15 indicates the outcome of the operation:

| Register 15 Value | Meaning |
|---|---|
| X'00' | This request has been serviced successfully. |
| X'04' | Either the requested longterm ID cannot be found or the #KEEP request has been issued by a nonterminal task. |
| X'14' | The request cannot be serviced because a lock on the specified record already exists; NOWAIT has been specified. |
| X'18' | The request cannot be serviced because to wait for an existing lock to be released would cause a deadlock. |

## #KEEP Example

The following is an example of the #KEEP statement that requests that the system monitor the database activity of a record. The #KEEP NOTIFY statement selects an EMPLOYEE record that is current of the EMPLOYEE record type and assigns it a long-term lock ID of REC1. Use of the NOTIFY parameter causes the system to initialize register 0, which will receive the information regarding database activities.

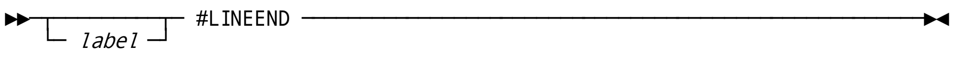
The #KEEP TEST statement calls on the system to return the database activity information for the record identified by a lock ID of REC1 to the low-order byte of register 0. The information is returned as a numeric value and is tested by comparing the value in register 0 to the numeric literal that contains the value 2. If the value in register 0 is greater than or equal to 2, the program will branch to location MODREC. If the value is less than the value of register 0 the program will proceed to the next statement.

```
#KEEP NOTIFY,LONGID='REC1',CURRENT='EMPLOYEE'
.
.
.
#KEEP TEST,LONGID='REC1'
C (R_),=F'2'
BNL MODREC
.
.
```

## #LINEEND—requests termination of the current line I/O session

The #LINEEND statement requests termination of the current line I/O session and deletes any outstanding buffered output lines and pages queued for asynchronous I/O. Unless NOBKPG is specified, all pages processed by the terminal operator during the I/O session remain available until the operator signals completion of the review by pressing ENTER with no request to see another page. At that time, all pages for the session are deleted, page header lines are cleared, and the current page number is set to 1.

### #LINEEND Syntax



### #LINEEND Parameters

**#LINEEND**

Requests that the system terminate the current line I/O session and to delete any remaining buffered output lines and pages queued for asynchronous I/O.

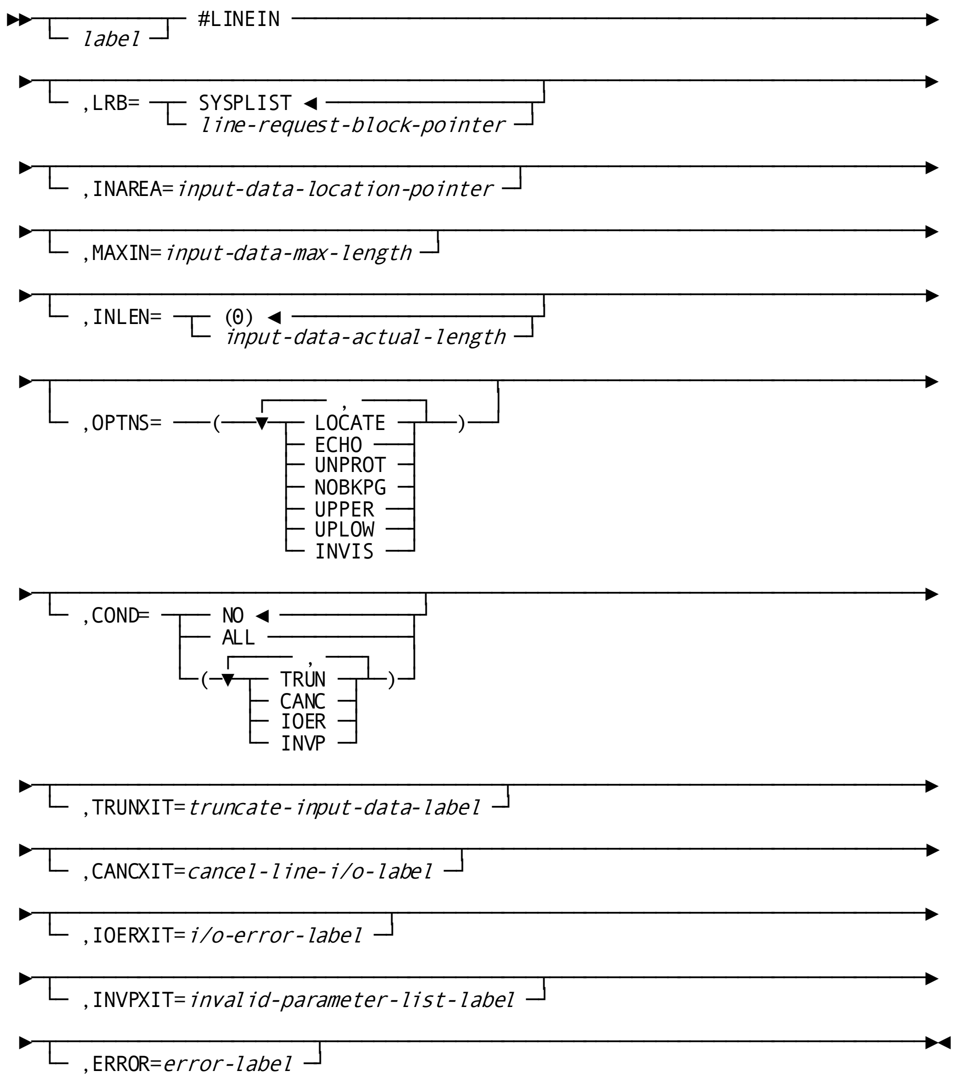
### #LINEEND Status Codes

The #LINEEND request is unconditional; any error detected during execution will result in an abend of the issuing task.

## #LINEIN—requests a synchronous transfer of data

The #LINEIN statement requests a synchronous transfer of data from the terminal to the issuing program.

## #LINEIN Syntax


label
#LINEIN
,LRB=
SYSPLIST
line-request-block-pointer
,INAREA=input-data-location-pointer
,MAXIN=input-data-max-length
,INLEN=
(0)
input-data-actual-length
,OPTNS=
(
,
LOCATE
ECHO
UNPROT
NOBKPG
UPPER
UPLOW
INVIS
)
,COND=
NO
ALL
(
,
TRUN
CANC
IOER
INVP
)
,TRUNXIT=truncate-input-data-label
,CANCXIT=cancel-line-i/o-label
,IOERXIT=i/o-error-label
,INVPXIT=invalid-parameter-list-label
,ERROR=error-label

## #LINEIN Parameters

**LRB=**

Specifies the three-fullword storage area in which the system will build the #LINEIN parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system will build the line request block (LRB).

**line-request-block-pointer**

A register that points to the area or the symbolic name of the area in which the system will build the LRB.

**INAREA=**

Specifies the storage area into which the data will be read.

**input-data-location-pointer**

A register that points to the area or the symbolic name of the area. When INAREA is specified, the LOCATE option should not be requested.

**MAXIN=**

Specifies the length, in bytes, of the data area, defined by INAREA, that is reserved for the input data stream.

**input-data-max-length**

A register that contains the length of the data area or an absolute expression. When MAXIN is specified, the LOCATE option should not be requested.

**INLEN=**

Specifies the location to which the system will return the actual length of the input data stream. If INAREA is too small to hold the entire input line, resulting in truncation, the returned length will indicate the original length of the data stream before truncation.

**(0)**

(Default); is the register to which the system will return the actual length of the input data stream.

**input-data-actual-length**

A register or the symbolic name of a halfword or fullword user-defined field to which the system will return the actual length of the input data stream.

**OPTNS=**

Specifies several options applicable to terminal input operations. This parameter is never required and should be specified only when appropriate. The OPTNS-parameter values must be enclosed in parentheses. If multiple values are specified, each is separated from the previous one by a comma.

**LOCATE**

Allocates a buffer area for the data being read into the program, rather than a user-defined area. The system allocates the buffer when the read operation is completed. Register 1 contains the address of this buffer on completion of the input operation. The issuing program is responsible for releasing the buffer area, using a #FREESTG command. When this option is requested, INAREA and MAXIN should not be specified.

**ECHO**

(3270 devices only); requests that the system save the line of input data as displayed on the screen in the current page. If OPTNS=ECHO is not specified, data entered will not be retained and will not be available for review by the terminal operator.

**UNPROT**

(3270 devices only); causes the first line of output that follows the #LINEIN to be unprotected. At runtime, the terminal operator can reuse the unprotected first line of an output display for input to a subsequent #LINEIN. The UNPROT option can be used with or without the ECHO parameter. For example, if the terminal operator has made an error in previous input data, the data that is retained by the ECHO option can be rekeyed and corrected. If UNPROT is not included, all lines of the following output display remain protected.

**NOBKPG**

(3270 devices only); requests the system not to keep pages that have been input in a scratch area. If NOBKPG is specified, the terminal operator can view only the current page of data. NOBKPG is valid only with the first request in a line mode session.

**UPPER**

Directs the system to translate all letters in a #LINEIN request into uppercase characters.

**UPLOW**

Specifies that no uppercase translation of characters in a #LINEIN request be performed.

**INVIS**

Specifies that the operator's response to the #LINEIN command will not appear on the screen as it is typed in. This option is useful when expecting a secret password to be entered.

**COND=**

Specifies whether this #LINEIN is conditional and under what conditions control should be returned to the issuing program.

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the request cannot be serviced for any of the reasons listed below.

**condition**

Specifies conditions under which the system returns control to the issuing task. Multiple conditions must be enclosed in parentheses and separated by commas.

**TRUN**

The input data is truncated due to insufficient storage in the specified INAREA.

**CANC**

The line I/O session is terminated by the terminal operator pressing CLEAR (3270), ATTENTION (2741), or BREAK (teletype).

**IOER**

A logical or permanent I/O error is encountered in the input data stream.

**INVP**

There is an invalid parameter in the LRB.

**TRUNXIT=truncate-input-data-label**

Specifies the symbolic name of the routine to which control should be returned if input data is truncated due to insufficient storage in the INAREA buffer.

**CANCXIT=cancel-line-i/o-label**

Specifies the symbolic name of the routine to which control should be returned if the line I/O session is terminated by the terminal operator.

**IOERXIT=i/o-error-label**

Specifies the symbolic name of the routine to which control should be returned if a permanent or logical error is detected in the input data stream.

**INVPXIT=invalid-parameter-list-label**

Specifies the symbolic name of the routine to which control should be returned in the event of an invalid parameter in the LRB.

**ERROR=error-label**

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

## #LINEIN Status Codes

By default, the #LINEIN request is unconditional; any runtime error will result in an abend of the issuing task. The issuing program can request return of control with the COND parameter to avoid an abend.

After completion of the #LINEIN, the value in register 15 indicates the outcome of the operation.

| Register 15 Value | Meaning |
|---|---|
| X'00' | The request has been serviced successfully. |

| | |
|---|---|
| X'04' | The input area specified for the return of data to the issuing program is too small; the returned data has been truncated to fit available space. |
| X'08' | The line I/O session has been canceled; the terminal operator has pressed CLEAR (3270), ATTENTION (2741), or BREAK (teletype). |
| X'0C' | A logical or permanent I/O error has been encountered in the input data stream. |
| X'10' | The line request block (LRB) contains an invalid field, indicating a possible error in the program parameters. |

Upon successful completion of a #LINEIN request, register 1 and a user-defined register will contain the following information:

- Register 1 (LOCATE option only) contains the address of the buffer into which the input data has been placed.
- Register n contains the actual length of returned data from the input operation; it can be a register or a user-defined field. The register number, n, is assigned by the INLEN parameter.

## #LINEIN Example

The example of the #LINEIN statement shown below performs the following functions:

- Uses the default storage area, SYSPLIST, to build the line request block
- Specifies that the data is to be read into an input storage area located at the address contained in register 5
- Specifies that register 6 contains the length of the data area, defined by the INAREA parameter, that is reserved for the input data stream
- Uses the default register 0 to receive the actual length of the input data stream from the system
- Specifies the conditional return of control if either the input data stream is truncated due to insufficient storage in the specified INAREA or the I/O session is terminated by the terminal operator
- Specifies the two routines to receive control in the event of a TRUN or CANC condition

```
#LINEIN INAREA=(R5),MAXIN=(R6),COND=(TRUN,CANC),TRUNXIT=TRUNRTN, _
CANCXIT=OPERTER
```

## #LINEOUT—requests a transfer of data

The #LINEOUT statement requests a transfer of data from the issuing program to the terminal, after appending line and device control characters appropriate to the physical terminal in use. #LINEOUT also establishes, modifies, and deletes page header lines.

A data transfer requested by the #LINEOUT statement can be synchronous or asynchronous; requests are asynchronous only when the NOWAIT option is specified:

- **Synchronous**—Following a synchronous request, control passes to the DC/UCF system. The system places the issuing task in an inactive state; when the #LINEOU request is completed, the task is redispatched according to its established priority. With 3270 terminals, a synchronous #LINEOUT request causes a processing delay immediately following the request while the system transfers the line to the page buffer. If the line of data fills the buffer, the system transfers the entire page of data to the terminal. Control does not return to the issuing program until the terminal operator has pressed the ENTER key. Thus, the program is made conversational.
- **Asynchronous**—Following an asynchronous request, control returns immediately to the issuing program. Thereafter, each time the program issues a line-mode I/O request, the system automatically checks to determine if the last asynchronous request has completed, and whether a new data transfer can be initiated.

  Asynchronous requests enable programs to buffer all required pages of output without suspending task execution during the actual data transmission. With an asynchronous request, the task can optionally terminate itself, freeing all its resources. The terminal operator can then review the buffered output, if desired.

The system processes I/O requests in the sequence received from the task; thus, if a program issues a synchronous #LINEOUT request after issuing one or more asynchronous requests, the system will complete all I/O requests before returning control to the issuing program.

The #LINEOUT request issued automatically by the system to empty partially-filled buffers on completion of a task is synchronous; therefore the terminal operator can view all screens and catch up with processing at that time.

If an application necessitates allowing the terminal operator to interrupt or terminate processing at some point in a task, a synchronous request must be issued to suspend processing while waiting for an operator response.

To transfer data immediately to a terminal, a write-direct-to-terminal #LINEOUT request (blast) can be issued. The system does not page multiple blast requests. The following #LINEOUT parameters are ignored during blast requests:

- HDR=
- OPTNS=(NOWAIT/NOBKPG/NEWPAGE)

(The NEWPAGE option is automatically forced during blast requests.)

Header lines can be defined for each new page of output to be transferred to a terminal. A maximum of three header lines can be established for each new page of output. The #LINEOUT statement specifies a header line and corresponding header-line number that can be used in subsequent new pages. The established header lines are sent to the terminal and written with each new page of output. The existing header lines may be overridden or deleted at any time during processing by issuing a #LINEOUT request specifying the appropriate line number and, for an override, the corresponding new header line.

## #LINEOUT Syntax

```
►►─┬───────┬──────────────────────────────────────────────────────────────►
   └ label ┘

►── #LINEOUT OUTLEN=output-data-length ────────────────────────────────────►

►─┬──────────────────────────────────────────┬─────────────────────────────►
  └ ,LRB= ─┬─ SYSPLIST ◄ ────────────────────┤
           └─ line-request-block-pointer ────┘

►─┬─────────────────────────────────────────┬──────────────────────────────►
  └ ,OUTAREA=output-data-location-pointer ──┘

►─┬──────────────────────────────────────────┬─────────────────────────────►
  │                   ┌────── , ───────┐     │
  └ ,OPTNS= ──(──▼─┬─ NEWPAGE ─┬──┴─)───┘
                   ├─ NOWAIT ──┤
                   ├─ NOBKPG ──┤
                   └─ SAVE ────┘

►─┬───────────────────────────┬────────────────────────────────────────────►
  └ ,HDR=header-number ───────┘

►─┬──────────────────────────────────────────────────┬─────────────────────►
  ├ ,DESTID=destination-id-pointer ──────────────────┤
  ├ ,USERID=user-id-pointer ─────────────────────────┤
  └ ,LTERMID=logical-terminal-id-pointer ────────────┘

►─┬─────────────────────────────────────┬──────────────────────────────────►
  └ ,COND= ─┬─ NO ◄ ────────────────────┤
            ├─ ALL ─────────────────────┤
            │    ┌──── , ────┐          │
            └─(─▼─┬─ CANC ─┬─┴─)────────┘
                  ├─ IOER ─┤
                  ├─ INVP ─┤
                  └─ UNDF ─┘

►─┬──────────────────────────────────┬─────────────────────────────────────►
  └ ,CANCXIT=cancel-line-i/o-label ──┘

►─┬──────────────────────────────┬─────────────────────────────────────────►
  └ ,IOERXIT=i/o-error-label ────┘

►─┬──────────────────────────────────────────┬─────────────────────────────►
  └ ,INVPXIT=invalid-parameter-list-label ───┘

►─┬──────────────────────────────────────────┬─────────────────────────────►
  └ ,UNDFXIT=invalid-destid-ltermid-label ───┘

►─┬─────────────────────────────┬─────────────────────────────────────────►◄
  └ ,ERROR=error-label ─────────┘
```

## #LINEOUT Parameters

**OUTLEN=**

Specifies the length, in bytes, of the data stream to be written to the terminal.

**output-data-length**

A register that contains the length or an absolute expression of the length. Output data lengths of 0 and 1 can be used in the following situations:

- OUTLEN=0 Specifies that no data is to be written to the terminal or that a header line is to be deleted:

  When the HDR parameter is not specified, OUTLEN=0 specifies a dummy write. No I/O is initiated by this request unless the NEWPAGE option, described below for the OPTNS parameter, is specified; if OPTNS=(NEWPAGE), this request writes a partially-filled buffer to the

  terminal.

  When the HDR parameter is specified, OUTLEN=0 specifies a deletion of a header line. The HDR parameter indicates the number of the header line to be deleted.

- OUTLEN=1 Specifies that a 1-byte data stream is to be written to the terminal. Typically, OUTLEN=1 is used to write a blank line to the screen. In this case, the OUTAREA parameter, described below, should designate a single blank character.

**LRB=**

Specifies the three-fullword storage area in which the system will build the #LINEOUT parameter list

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system will build the line request block.

**line-request-block**

A register that points to the area or the symbolic name of that area in which the system will build the LRB.

**OUTAREA=**

Specifies the storage area that contains data to be output. OUTAREA need not be defined if OUTLEN=0 has been specified.

**output-data-location**

A register that points to the area or the symbolic name of the area.

**OPTNS=terminal-option**

Specifies several options applicable to terminal output operations. This parameter is never required and should be specified only when appropriate. The OPTNS parameter values must be enclosed in parentheses. If multiple values are specified, each is separated from the previous one by a comma.

**NEWPAGE**

Requests that the system write the output data line beginning on a new page. For 3270 devices, the NEWPAGE option forces the system to output all lines stored in the current buffer, even if the buffer is not full.

**NOWAIT**

Requests an asynchronous transfer of data; the issuing task executes concurrently with the output operation.

**NOBKPG**

(3270 devices only); requests the system not to keep pages that have been output in a scratch area. If NOBKPG is specified, the terminal operator can view only the current page of data. NOBKPG is valid only with the first request in a line mode session.

**SAVE**

Directs the system to preserve the output from the #LINEOUT in the event that an unsolicited write-direct-to-terminal data stream is received at the issuing terminal while the #LINEOUT data stream is being displayed. This option overrides the task SAVE/NOSAVE option specified during system generation.

**HDR=**

Specifies the number of the page header line being defined, modified, or deleted.

**header-line-number**

An absolute expression of the line number. If OUTLEN is other than 0 the value stored in OUTAREA will be moved to the designated (first, second, or third) header line. If a header line with the same number has been previously defined for this I/O session, the system will replace it with the value stored in OUTAREA. If OUTLEN=0, the designated header line will be deleted.

**DESTID/USERID/LTERMID**

Specifies a write-direct-to-terminal request. The HDR= and OPTNS=(NOWAIT/NOBKPG/NEWPAGE) parameters are ignored during a blast request.

**DESTID=**

Specifies a write-direct-to-terminal request (blast) to the following destinations defined during system generation:

- List of logical terminals indicates that the system will send the #LINEOUT data stream specified in the OUTAREA parameter to all available terminals in the list.

- List of users indicates that the system will send the #LINEOUT data stream specified in the OUTAREA parameter to all users in the list who are currently signed on to the system.

**destination-id**

A register that points to the destination id, the symbolic name of a user-defined field that contains the destination ID, or the ID itself enclosed in quotation marks. The destination list can include both 3270 and TTY devices.

**USERID=**

Specifies a blast request to a specific signed-on user. The system will send the #LINEOUT data stream specified in the OUTAREA parameter to a specific signed-on user.

**user-id**

A register that points to the user id, the symbolic name of a user-defined field that contains the user ID, or the ID itself enclosed in quotation marks.

**LTERMID=**

(#LINEOUT only); specifies a blast request to a specific in-service terminal. The system will send the #LINEOUT data stream specified in the OUTAREA parameter to a specific in-service terminal.

**logical-terminal-id**

A register that points to the logical terminal id, the symbolic name of a user-defined field that contains the logical terminal ID, or the ID itself enclosed in quotation marks.

**COND=**

Specifies whether this #LINEOUT is conditional and under what conditions control should be returned to the issuing program.

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the request cannot be serviced for any of the reasons listed below.

**condition**

Specifies conditions under which the system returns control to the issuing task. Multiple conditions must be enclosed in parentheses and separated by commas.

**CANC**

The line I/O session is terminated by the terminal operator pressing CLEAR (3270), ATTENTION (2741), or BREAK (teletype).

**IOER**

A logical or permanent I/O error is encountered in the output data stream.

**INVP**

There is an invalid parameter in the LRB.

**UNDF**

An undefined DESTID or LTERMID is specified in a #LINEOUT blast request.

**CANCXIT=cancel-line-i/o-label**

Specifies the symbolic name of the routine to which control should be returned if the line I/O session is terminated by the terminal operator.

**IOERXIT=i/o-error-label**

Specifies the symbolic name of the routine to which control should be returned if a permanent or logical I/O error is detected in the output data stream.

**INVPXIT=invalid-parameter-list-label**

Specifies the symbolic name of the routine to which control should be returned in the event of an invalid parameter in the LRB.

**UNDFXIT=invalid-destid-ltermid-label**

Specifies the symbolic name of the routine to which control should be returned if an undefined DESTID or LTERMID is specified in a #LINEOUT blast request.

**ERROR=error-label**

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

## #LINEOUT Status Codes

By default, the #LINEOUT request is unconditional; any runtime error will result in an abend of the issuing task. The issuing program can request return of control with the COND parameter to avoid an abend.

After completion of the #LINEOUT, the value in register 15 indicates the outcome of the operation:

| Register 15 Value | Meaning |
|---|---|
| X'00' | The request has been serviced successfully. |

| | |
|---|---|
| X'08' | The line I/O session has been canceled by the operator pressing the CLEAR (3270), ATTENTION (2741), or BREAK (teletype) key. |
| X'0C' | A logical or permanent I/O error has been encountered in the output data stream. |
| X'10' | The line request block (LRB) contains an invalid field, indicating a possible error in the #LINEOUT parameters. |
| X'14' | The name specified for DESTID, USERID, or LTERMID is unknown to this DC/UCF system. |

## #LINEOUT Example

The example of the #LINEOUT statement shown below performs the following functions:

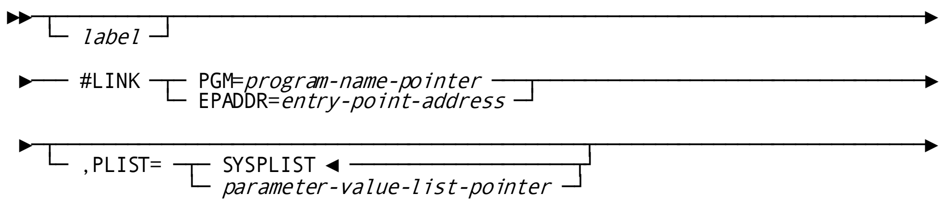
- Specifies that register 7 contains the length of the output data stream
- Uses the default storage area SYSPLIST to build the line request block (LRB)
- Identifies OUT1 as the storage area that contains the output data stream
- Specifies a write-direct-to-terminal request to a group of users defined during system generation as USERLIST
- Specifies a conditional return of control to the routine labeled LISTERR in the event that DESTID 'USERLIST' is not defined to the system

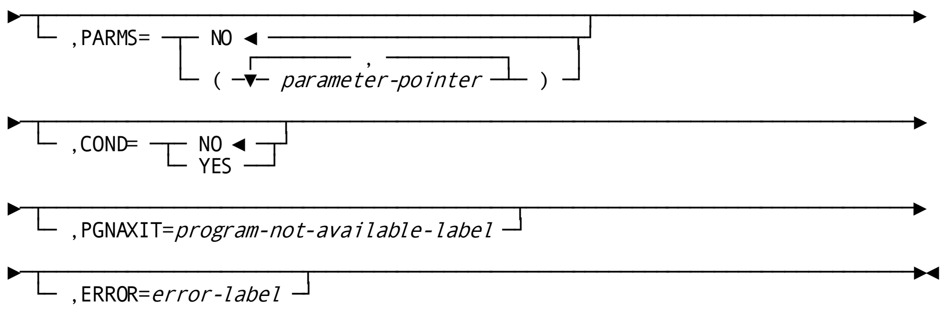
```
#LINEOUT OUTLEN=(R7),OUTAREA=OUT1,DESTID='USERLIST',COND=UNDF, _
UNDFXIT=LISTERR
```

## #LINK—establishes linkage with a program

The #LINK statement establishes linkage with, and passes control and an optional parameter list to, a specified program. When the linked program terminates or executes a #RETURN request, the program issuing the #LINK expects return of control to the instruction immediately following the #LINK statement.

### #LINK Syntax





## #LINK Parameters

**PGM=**

Specify the program and/or entry-point address of the program to which control is transferred.

**program-name-pointer**

Specifies the 1- to 8-character name of the program to which control is transferred. Program-name is a register that points to a field that contains the program name, the symbolic name of a user-defined field that contains the program name, or the program-name literal enclosed in quotation marks.

**entry-point-address**

Specifies the entry-point address of the program to which control is transferred. Entry-point-address is a register or symbolic name of a fullword user-defined field that contains the entry-point address.

**PLIST=**

Specifies the location of the storage area that contains one or more parameters to be passed to the program receiving control.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system will build the parameter list.

**parameter-value-list-pointer**

A register that points to the area in which the system will build the list or the symbolic name of that area.

The size of the parameter-list area, in fullwords, must be equal to 2 plus the number of parameters listed in the PARMS parameter described below.

Thus, if no parameters are specified (PARMS=NO), the length of this storage area is two fullwords; if one parameter is specified, the length is three fullwords.

**PARMS=**

Specifies the location of each parameter to be passed to the program receiving control.

**NO**

(Default); indicates that no parameters will be passed to the program.

**parameter-pointer**

Indicates that parameters are to be passed to the program. Parameter is a register that contains the address of the parameter or the symbolic name of a user-defined field that contains the parameter.

The parameter list must be enclosed in parentheses. If multiple parameters are specified, each is separated from the previous one by a comma.

**COND=**

Specifies whether this #LINK is conditional; that is, whether control should be returned to the issuing program in the event of an error:

**NO**

(Default); specifies that the request is not conditional.

**PGNA**

Specifies that the request is conditional. Control is returned if the #LINK cannot be serviced because the program is not available.

**PGNAXIT=program-not-available-label**

Specifies the symbolic name of the routine to which control should be returned if the #LINK request cannot be serviced because the program is not available.

**ERROR=error-label**

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded. In this case, the ERROR parameter functions the same as PGNAXIT.

## #LINK Status Codes

By default, the #LINK request is unconditional. Error conditions

that can occur are described below:

- A no-space-in-program-pool condition, caused when the amount of storage in the program pool is inadequate to accommodate the program, will result in a delay until sufficient storage space becomes available (unless such a wait would cause a deadlock, in which case an abort would occur).

- A nonconcurrent-program-in-use condition, caused when a copy of the program is already in use and is marked as nonconcurrent (indicating that this program can be used by a maximum of one task), will result in a delay until the program becomes available.
- A storage-conflict condition, caused when a copy of the program previously loaded is temporarily overlayed while in use by a waiting task, will result in a delay until the program is replaced in the program pool.
- Any abnormal condition will result in an abend. Conditions in this category include the following:
  - I/O error
  - Program not found in program definition table (PDT)
  - A wait on storage (default action resulting from the no-storage-in-program-pool condition) would result in a deadlock

The issuing program can request return of control with the COND parameter to avoid a delay or an abend.

After completion of the #LINK function, the value in register 15 indicates the outcome of the operation:

| **Register 15 Value** | **Meaning** |
|---|---|
| X'00' | The request has been serviced successfully. |
| X'04' | The request cannot be serviced because an I/O, program-not-found, or potential-deadlock error has occurred, or the program has not been defined in the program definition element (PDE). |

## #LINK Example

The example of the #LINK statement shown below performs the

following functions:

- Specifies that control will be transferred to the program HELPLK
- Uses the default storage area, SYSPLIST, in which the system builds the parameter list
- Identifies the parameters, PARM1 and PARM2, to be passed to the program HELPLK
- Specifies a conditional return of control if the program HELPLK is not available and identifies the routine NOPROG that will receive control in the event of a PGNA error condition

```
#LINK PGM='HELPLK',PARMS=(PARM1,PARM2),COND=PGNA,PGNAXIT=NOPROG
```

# #LOAD—loads a module into the program pool

The #LOAD statement loads a module (program or table) into the program pool. In response to a #LOAD, the system returns the entry-point address of the module and the address of the resource control element (RCE) to the issuing program.

## #LOAD Syntax

```
►►─┬─────────┬─ #LOAD PGM=program-name-pointer ──────────────────────►
   └─ label ─┘

►─┬──────────────────────────┬────────────────────────────────────────►
  └─ VERSION=version-number ─┘

►─┬─────────────────────────────┬─────────────────────────────────────►
  └─ ,DICTNOD=nodename-pointer ─┘

►─┬────────────────────────────────────┬──────────────────────────────►
  └─ ,DICTNAM=dictionary-name-pointer ─┘

►─┬────────────────────────────────────────┬──────────────────────────►
  └─ ,EPADDR= ─┬─ (0) ◄ ───────────────────┤
               └─ entry-point-address ─────┘

►─┬───────────────────────────┬───────────────────────────────────────►
  └─ ,TYPE= ─┬─ PROGRAM ──────┤
             └─ TABLE ────────┘

►─┬──────────────────────────────────────────────┬────────────────────►
  └─ ,PLIST= ─┬─ SYSPLIST ◄ ─────────────────────┤
              └─ parameter-value-list-pointer ───┘

►─┬──────────────────────────────────┬────────────────────────────────►
  └─ ,COND= ─┬─ NO ◄ ────────────────┤
             ├─ ALL ─────────────────┤
             │     ┌─────── , ───────┐ │
             └─(─▼─┬─ NOST ─┬─┴─)──┘
                   ├─ IOER ─┤
                   ├─ SNGL ─┤
                   ├─ LDCF ─┤
                   ├─ PGNF ─┤
                   └─ DEAD ─┘

►─┬───────────────────────────────────────┬───────────────────────────►
  └─ NOSTXIT=insufficient-storage-label ──┘

►─┬────────────────────────────┬──────────────────────────────────────►
  └─ ,IOERXIT=i/o-error-label ─┘

►─┬─────────────────────────────────────────┬─────────────────────────►
  └─ ,SNGLXIT=single-thread-in-use-label ───┘

►─┬────────────────────────────────────────────┬──────────────────────►
  └─ ,LDCFXIT=storage-location-conflict-label ─┘

►─┬────────────────────────────────────┬──────────────────────────────►
  └─ ,PGNFXIT=program-not-found-label ─┘

►─┬───────────────────────────┬───────────────────────────────────────►
  └─ ,DEADXIT=deadlock-label ─┘

►─┬─────────────────────┬────────────────────────────────────────────►◄
  └─ ,ERROR=error-label ┘
```

## #LOAD Parameters

**PGM=**

Specifies the 1- to 8-character name of the module to be loaded in the program pool.

**program-name-pointer**

A register that points to a field that contains the program name, the symbolic name of a user-defined field that contains the program name, or the program-name literal enclosed in quotation marks.

**VERSION=version-number**

Specifies a version number. Version-number can be an absolute value, a halfword or fullword value, or a register.

**DICTNOD=**

Identifies the node that controls the dictionary in which the program resides.

**nodename-pointer**

A register that points to a field that contains the name of the node, the symbolic name of a user-defined field containing the name of the node, or the nodename literal enclosed in quotation marks. A blank value refers to the local node.

**DICTNAM=**

Identifies the default dictionary in which the named program resides.

**dictionary-name-pointer**

A register that points to a field containing the dictionary name, the symbolic name of a user-defined field containing the dictionary name, or the dictionary name literal enclosed in quotation marks.

**Note**: If the DICTNAM and/or DICTNOD is specified, the system searches only the specified dictionary for the module. A program-not-found condition is returned if the module cannot be found in the specified dictionary.

**EPADDR=**

Specifies where the system will return the entry-point address of the loaded program.

**(0)**

(Default) specifies that the system will return the entry-point address to register 0.

**entry-point-address**

Specifies that the system will return the entry-point address to a user-defined Entry-point-address is a register location or the symbolic name of a fullword user-defined field that contains the entry-point address.

**,TYPE=**

Qualifies the type of load to perform.

**PROGRAM**

Has been pre-defined as a program at system generation or dynamically defined as a program via DCMT VARY DYNAMIC PROGRAM command.

**Note:** The program must reside in a load library. No attempt will be made to load the program from a dictionary load area.

**TABLE**

Has been pre-defined as a table at system generation or dynamically defined using a DCMT VARY DYNAMIC PROGRAM command.

**PLIST=**

Specifies the location of the storage area in which the system builds the #LOAD parameter list.

**SYSPLIST**

Is the symbolic name of the storage area in which the system builds the #LOAD parameter list.

**parameter-value-list-pointer**

A register that points to the area or the symbolic name of the area.

**Note:** The PLIST parameter is required only if the DICNAM or DICTNOD options are specified.

**COND=**

Specifies whether this #LOAD is conditional and under what conditions control should be returned to the issuing program:

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the load cannot be serviced for one or more of the reasons listed under condition.

**condition**

Specifies conditions under which control is returned to the program.

**NOST**

Available storage in the program pool is insufficient to load the requested program.

**IOER**

An I/O error occurs during the load.

**SNGL**

The requested program has been defined as nonconcurrent and is currently in use.

**LDCF**

The requested program is in use by another task but has been overlayed temporarily in the program pool, causing a storage location conflict.

**PGNF**

The requested program cannot be found in the program definition table (PDT), or is marked as out-of-service.

**DEAD**

The requested program cannot be loaded immediately because of a no-storage-in-program-pool condition and waiting would cause a deadlock.

**NOSTXIT=insufficient-storage-label**

Specifies the symbolic name of a routine to which control should be returned if the #LOAD request cannot be serviced due to insufficient storage in the program pool.

**IOERXIT=i/o-error-label**

Specifies the symbolic name of a routine to which control should be returned if the #LOAD request cannot be serviced due to an I/O error while processing the load.

**SNGLXIT=single-thread-in-use-label**

Specifies the symbolic name of a routine to which control should be returned if the #LOAD request is for a program marked nonconcurrent and the program is in use.

**LDCFXIT=storage-location-conflict-label**

Specifies the symbolic name of a routine to which control should be returned if the #LOAD request cannot be serviced due to a storage location conflict.

**PGNFXIT=program-not-found-label**

Specifies the symbolic name of a routine to which control should be returned if either the requested program cannot be found in the PDT or is out-of-service.

**DEADXIT=deadlock-label**

Specifies the symbolic name of a routine to which control should be returned if the requested program cannot be loaded immediately and to wait on its availability would cause a deadlock.

**ERROR=error-label**

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

## #LOAD Status Codes

By default, the #LOAD request is unconditional. Error conditions that can occur are:

- A no-storage-in-program-pool condition is caused when there is not enough storage in the program pool to accommodate the program. This conditions results in a delay until sufficient storage becomes available (unless such a wait would cause a deadlock).
- A nonconcurrent-program-in-use condition is caused when a copy of the program is already in use and is marked as nonconcurrent (indicating that this program can be used by a maximum of one task at a time). This conditions results in a delay until the program becomes available.
- A storage-conflict condition occurs when a previously loaded copy of the program is temporarily overlayed while in use by a waiting task. This condition results in a delay until the program is replaced in the program pool.
- Any abnormal condition will result in an abend. Conditions in this category include the following:
  - I/O error
  - Program not found in PDT, or marked as out-of-service
  - Waiting for storage-pool or program-pool memory, the default action resulting from the no-storage-in-program-pool condition, would cause a deadlock

The issuing program can request return of control with the COND parameter to

avoid a delay or an abend.

After completion of the #LOAD function, the value in register 15 indicates the outcome of the operation:

| Register 15 Value | Meaning |
|---|---|
| X'00' | The request has been serviced successfully. |
| X'04' | The request cannot be serviced due to insufficient storage in the program pool. |
| X'08' | The request cannot be serviced due to an I/O error during a load from a load library. |
| X'0C' | The requested program is nonconcurrent and in use. |
| X'10' | The requested program has been overlayed temporarily in the program pool, resulting in a storage conflict. |
| X'14' | The requested program is not defined to the PDT, is marked as out-of-service, or a null PDE could not be allocated for the program. |

| | |
|---|---|
| X'18' | The requested program cannot be loaded immediately (insufficient storage); to wait would cause a deadlock. |
| X'20' | The requested program cannot be loaded immediately due to an I/O error during a load from the dictionary DDLDCLOD area. |

The values in a user-defined register and register 1 also contain the following information:

- Register n specifies the entry-point address of the loaded program. The register number n is assigned by the EPADDR parameter of the #LOAD statement.
- Register 1 specifies the address of the RCE of the loaded program.

## #LOAD Example

The #LOAD statement shown below loads the program EMPMENU into the program pool:

```
#LOAD PGM='EMPMENU'
```

## #MAPINQ

The #MAPINQ statement is used after a map input request to accomplish one of the following actions related to the input operation:

- Move map-related information into variable storage
- Test for conditions relating to global map input operations
- Test specific map fields for the presence of the cursor
- Test for conditions relating to specific map fields

If you use the #MAPINQ statement to test for conditions, you must specify a routine that receives control if the condition is true.

Each of the four types of #MAPINQ statements is discussed in this chapter.

## Moving Map-Related Data

This version of the #MAPINQ statement moves the following information into variable storage:

- The cursor position (row and column).
- The attention ID (AID) key used. An AID key is the key that was last pressed during the input operation.
- The entered length of a specific input field.

**Syntax**

```
►►── #MAPINQ MRB=map-request-block-pointer ──────────────────────────────►

 ►─┬──────────────────────────────────────────────┬──────────────────────►
   └─ ,MRBLIST= ─┬─ MRBPLIST ◄ ──────────────────┬┘
                 └─ mrb-parameter-list-pointer ──┘

   ┌────────────────────────────┐
 ►─▼─┬─ ,CURSOR=cursor-position ─┬┴─────────────────────────────────────►◄
     ├─ ,AID=aid-indicator ──────┤
     └─ field-options ───────────┘
```

**Expansion of field-options**

```
►►── ,FIELD=field-name ─────────────────────────────────────────────────►

 ►─┬──────────────────────────┬─────────────────────────────────────────►
   └─ ,INDEX=index-register ──┘

 ►─┬────────────────────────────────┬───────────────────────────────────►◄
   └─ ,INLEN=field-length-register ─┘
```

**Parameters**

**MRB=**

Specifies the storage area associated with the MRB of the map about which the inquiry is being made.

***map-request-block***

A register that points to the MRB storage area or the symbolic name of that area.

**MRBLIST=**

Specifies the location of the 20-fullword storage area that is substituted for the DC/UCF portion of SUBSCHEMA-CTRL.

**MRBPLIST**

(Default); is the symbolic name of the storage area that will be substituted for the DC/UCF portion of SUBSCHEMA-CTRL.

***mrb-parameter-list***

A register that points to the area or the symbolic name of the area.

**CURSOR=**

Requests that the system return the cursor address from the last map in operation to the specified location in the issuing program.

***cursor-position***

The symbolic name of a 2-byte user-defined field. The system will set the value of *cursor-position* to the row and the column, each a 1-byte binary number, of the cursor position on the screen.

**AID=**

Requests that the system return the AID to the specified location in the issuing program.

***aid-indicator***

The symbolic name of a 1-byte user-defined field that will be set to the 3270 AID character received in the last map in request.

**FIELD=**

Requests that the system move the entered length of the specified map field for which information is required.

***field-name***

Specifies the name of the map field.

Note: For each #MAPINQ request to return map-related data, field-specific information can be requested for one map field; if information is needed for multiple fields, additional #MAPINQ commands must be issued.

**INDEX=**

Specifies the occurrence of the field if *field-name* is a multiply-occurring field.

***index-register***

Either a register or the symbolic name of a user-defined field that contains the subscript or an absolute expression.

**INLEN=**

Requests that the system return the entered length, in bytes, of the specified map field to the issuing program.

***field-length-register***

A register or the symbolic name of a halfword or fullword user-defined field to which the system will return the length.

**Example**

The following #MAPINQ statement moves the contents of map field EMPNUM to the area in the program labeled BLOCK1. The value of the 3270 AID character received in the last map in request is returned to the user-defined field AIDBYTE. This field is tested for the specific AID key value that indicates the operator is finished with this program.

```
         #MAPINQ MRB=BLOCK1,AID=AIDBYTE,FIELD=EMPNUM
         CLI   AIDBYTE,CLEAR
         BE    RETURN
         .
         .
CLEAR    EQU   X'6D'
```

The following table lists attention ID (AID) key values.

| Key | AID Character | Key | AID Character |
|---|---|---|---|
| ENTER | "'" (single quote) | PF14 | 'B' |
| CLEAR | '_' (underscore) | PF15 | 'C' |
| PF01 | '1' | PF16 | 'D' |
| PF02 | '2' | PF17 | 'E' |
| PF03 | '3' | PF18 | 'F' |
| PF04 | '4' | PF19 | 'G' |
| PF05 | '5' | PF20 | 'H' |
| PF06 | '6' | PF21 | 'I' |
| PF07 | '7' | PF22 | ¢ |
| PF08 | '8' | PF23 | '.' |
| PF09 | '9' | PF24 | '<' |
| PF10 | ':' | PA01 | '%' |
| PF11 | '#' | PA02 | '>' |
| PF12 | '@' | PA03 | ',' |
| PF13 | 'A' | | |

## Testing for Global Map Input Conditions

This version of the #MAPINQ statement tests for one of the following conditions related to map input operations:

- The screen was not previously formatted before the map in was performed.
- One or more input fields were truncated when transferred to program variable storage.
- One or more input fields were modified on the screen before being transferred.
- One or more fields, which were modified on the screen, are undefined in the map being used.

**Syntax**

```
►►── #MAPINQ MRB=map-request-block-pointer ──────────────────────────────►

 ►─┬───────────────────────────────────────────────────────────────────┬─►
   └─ ,MRBLIST= ─┬─ MRBPLIST ◄ ─────────────┬┘
                 └─ mrb-parameter-list-pointer ─┘

 ►── ,CURSOR=cursor-position ─────────────────────────────────────────────►

 ►── ,AID=aid-indicator ──────────────────────────────────────────────────►

 ►─┬───────────────────────────────────────────────────────────────────┬─►◄
   └─ ,IF= ( ─┬─ UNFORMAT,unformatted-screen-label ──┬─ ) ─┘
              ├─ TRUNCATE,truncated-data-label ──────┤
              ├─ CHANGED,updated-data-label ─────────┤
              └─ XTRNEOUS,extraneous-input-data-label ┘
```

**Parameters**

**MRB=**

Specifies the storage area associated with the MRB of the map about which the inquiry is being made.

***map-request-block-pointer***

A register that points to the MRB area or the symbolic name of that area.

**MRBLIST=**

Specifies the location of the 20-fullword storage area that is substituted for the DC/UCF portion of SUBSCHEMA-CTRL.

**MRBPLIST**

(Default); is the symbolic name of the storage area that will be substituted for the DC/UCF portion of SUBSCHEMA-CTRL.

***mrb-parameter-list-pointer***

A register that points to the area or the symbolic name of the area.

**CURSOR=**

Requests that the system return the cursor address from the last map in operation to the specified location in the issuing program.

***cursor-position***

The symbolic name of a 2-byte user-defined field. The system will set the value of *cursor-position* to the row and the column, each a 1-byte binary number, of the cursor position on the screen.

**AID=**

Requests that the system return the AID to the specified location in the issuing program.

***aid-indicator***

The symbolic name of a 1-byte user-defined field that will be set to the 3270 AID character received in the last map in request.

**IF=**

Tests the outcome of the last map in request for a condition relating to the data input as a whole. Map data fields that are in error are not transferred to program variable storage.

Note: For more information about testing map input conditions, see the *Mapping Facility Guide*.

For each condition, the associated label specifies the symbolic name of the routine in the issuing program to which the system will pass control if the tested condition is true. The IF-parameter condition and label must be enclosed in parentheses.

**UNFORMAT,*unformatted-screen-label***

Tests whether the screen had been formatted before the input operation was performed.

**TRUNCATE,*truncated-data-label***

Tests whether any of the screen fields had been truncated when transmitted to program variable storage.

**CHANGED,*updated-data-label***

Tests whether any of the screen fields actually had been mapped to program data fields when the map in operation was performed.

**XTRNEOUS,*extraneous-input-data-label***

Tests whether the data stream that had been read from the terminal contains any data from a field undefined to the map. If this condition occurs, the system does not move the undefined data field to program variable storage.

**Example**

The following example of the #MAPINQ statement tests if any of the screen fields have been updated to the program data fields of the map identified by BLOCK1 when the last map in operation was performed. If the test is true, the program branches to the label NEWINFO. A false condition causes the program to execute the next sequential instruction:

```
#MAPINQ MRB=BLOCK1,IF=(CHANGED,NEWINFO)
```

## Testing Cursor Position

This version of the #MAPINQ statement tests a specified map field for the presence of the cursor.

**Syntax**

```
►►─── #MAPINQ MRB=map-request-block-pointer ─────────────────────────────►

 ►─┬──────────────────────────────────────────────────┬──────────────────►
   └─ ,MRBLIST= ─┬─ MRBPLIST ◄ ──────────────────┬─┘
                 └─ mrb-parameter-list-pointer ─┘

 ►─┬──────────────────────────────┬──────────────────────────────────────►
   └─ ,CURSOR=cursor-position ─┘

 ►─┬────────────────────────┬────────────────────────────────────────────►
   └─ ,AID=aid-indicator ─┘

 ►─── ,FIELD=field-name ─────────────────────────────────────────────────►

 ►─┬────────────────────────────┬────────────────────────────────────────►
   └─ ,INDEX=index-register ─┘

 ►─┬───────────────────────────────────┬─────────────────────────────────►
   └─ ,INLEN=field-length-register ─┘

 ►─┬─────────────────────────────────────────────┬───────────────────────►◄
   └─ ,IF=(CURSOR,cursor-at-this-field-label) ─┘
```

**Parameters**

**MRB=**

Specifies the storage area associated with the MRB of the map about which the inquiry is being made.

***map-request-block-pointer***

A register that points to the MRB area or the symbolic name of that area.

**MRBLIST=**

Specifies the location of the 20-fullword storage area that is substituted for the DC/UCF portion of SUBSCHEMA-CTRL:

**MRBPLIST**

(Default); is the symbolic name of the storage area that will be substituted for the DC/UCF portion of SUBSCHEMA-CTRL.

***mrb-parameter-list-pointer***

A register that points to the area or the symbolic name of the area.

**CURSOR=**

Requests that the system return the cursor address from the last map in operation to the specified location in the issuing program.

***cursor-position***

The symbolic name of a 2-byte user-defined field. The system will set the value of *cursor-position* to the row and the column, each a 1-byte binary number, of the cursor position on the screen.

**AID=**

Requests that the system return the AID to the specified location in the issuing program.

***aid-indicator***

The symbolic name of a 1-byte user-defined field that will be set to the 3270 AID character received in the last map in request.

**FIELD=**

Requests that the system move field-related information to the issuing program.

***field-name***

Specifies the name of the map field being tested.

Note: For each #MAPINQ request to test for cursor position, field-specific information can be requested for one map field; if information is needed for multiple fields, additional #MAPINQ commands must be issued.

**INDEX=**

Specifies the occurrence of the field if *field-name* is a multiply-occurring field.

***index-register***

Either a register or the symbolic name of a user-defined field that contains the subscript or an absolute expression.

**INLEN=**

Requests that the system return the entered length, in bytes, of the specified map field to the issuing program.

***field-length-register***

Either a register or the symbolic name of a halfword or fullword user-defined field to which the system will return the length.

**IF=CURSOR,**

Tests the outcome of the last map in request to determine whether the cursor was in the named field during the last map in operation.

***cursor-at-this-field-label***

Specifies the symbolic name of the routine within the issuing program to which the system will pass control if the cursor is in the named field during the last map in operation.

**Example**

The #MAPINQ statement shown below moves information about the EMPNUM field to the issuing task. The IF statement tests the outcome of the last map in request; if the cursor was in that field during the last map in operation, the system passes control to the routine labeled CURATNUM.

```
#MAPINQ MRB=BLOCK1,FIELD=EMPNUM,IF=(CURSOR,CURATNUM)
```

## Testing for Identical Data

You can compare the contents of a mapped-in field with the map data that is currently in your program's record buffer.

You can use #MAPINQ when you want to reduce the number of database I/O operations performed for your programs, updating the database only when the user enters different data.

To test for identical data, use the DATAIDEN and DATADIFF options of the IF= clause (see Testing for Input Conditions (see page 238)).

**Example**

Use a #MAPINQ statement to test whether the user has entered identical data in the EMPNUM, EMPNAME, CONCODE *and* UPDFLAG.

- If the identical condition is *true* (the user enters identical data in these fields), the program branches to NEXPRO2.
- If the identical condition is *false* (the user has changed at least one of these fields), control continues with the next executable instruction.

Use a #MAPINQ statement to test whether the user has entered a new department ID. If the user enters a new ID (different is *true*), the program branches to label OBTDEPT.

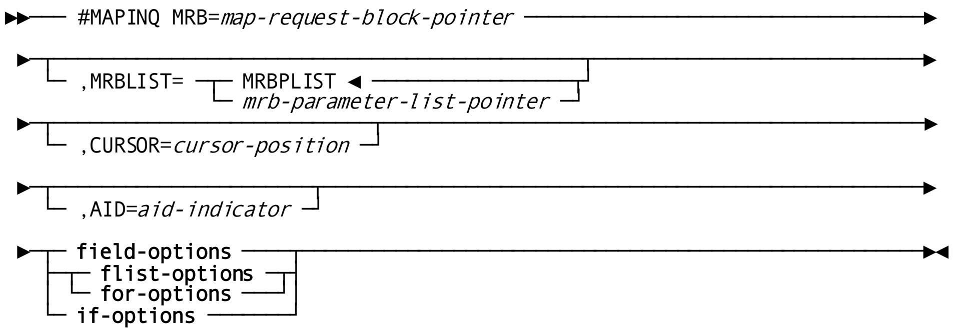
```
#MAPINQ MRB=BLOCK1,FLIST=(FIELD,DEPTID-0410),FOR=ANY,
       IF=(DATADIFF,OBTDEPT)
```

## Testing for Input Conditions

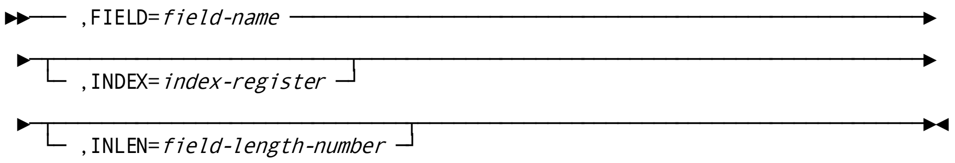
This version of the #MAPINQ statement tests for the following input conditions related to specific map fields:

- If map fields have been modified and the data fields in storage contain the new (changed) contents of that field.
- If map fields have not been modified and the data fields in storage remain unchanged.
- If map fields have been erased by operator action.
- If map fields have been truncated.
- If the specified map fields are either in error (the error flag has been set on) or the map fields are correct, (the error flag has been set off). This option applies only to those maps and map fields for which automatic editing is enabled.

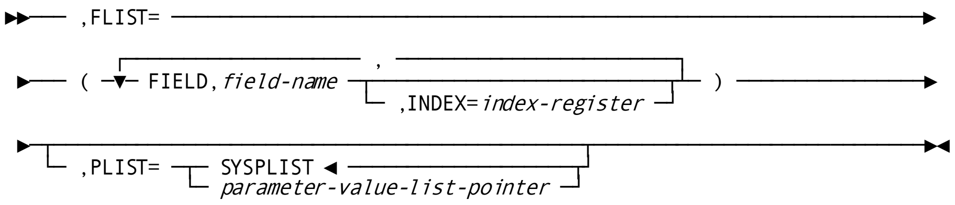
### Syntax


#MAPINQ MRB=*map-request-block-pointer*
,MRBLIST=
MRBPLIST
*mrb-parameter-list-pointer*
,CURSOR=*cursor-position*
,AID=*aid-indicator*
field-options
flist-options
for-options
if-options

### Expansion of field-options



### Expansion of flist-options



### Expansion of for-options

```
►►─── ,FOR= ──────────────────────────────────────────────────────────────►

►───┬─ CURRENT ─┬─────────────────────────────────────────────────────────►◄
    ├─ ALL ─────┤
    ├─ NONE ────┤
    ├─ SOME ────┤
    └─ ANY ─────┘
```

### Expansion of if-options

```
►►─── ,IF= ───────────────────────────────────────────────────────────────►

►─── ( ─┬─ DATANO,unchanged-field-label ──────┬─ ) ───────────────────────►◄
        ├─ DATAYES,updated-field-label ───────┤
        ├─ DATAERAS,erased-field-label ───────┤
        ├─ DATARUN,truncated-field-label ─────┤
        ├─ EDITERR,edit-error-field-label ────┤
        ├─ EDITCOR,edit-correct-field-label ──┤
        ├─ DATAIDEN,identical-data-label ─────┤
        └─ DATADIFF,different-data-label ─────┘
```

### Parameters

**MRB=**

Specifies the storage area associated with the MRB about which the inquiry is being made.

***map-request-block-pointer***

A register that points to the MRB area or the symbolic name of that area.

**MRBLIST=**

Specifies the location of the 20-fullword storage area that is substituted for the DC/UCF portion of SUBSCHEMA-CTRL.

**MRBPLIST**

(Default); is the symbolic name of the storage area that will be substituted for the DC/UCF portion of SUBSCHEMA-CTRL.

***mrb-parameter-list-pointer***

A register that points to the area or the symbolic name of the area.

**CURSOR=**

Requests that the system return the cursor address from the last map in operation to the specified location in the issuing program.

***cursor-position***

The symbolic name of a 2-byte user-defined field. The system will set the value of *cursor-position* to the row and the column, each a 1-byte binary number, of the cursor position on the screen.

**AID=**

Requests the system to return the AID to the specified location in the issuing program.

***aid-indicator***

The symbolic name of a 1-byte user-defined field that will be set to the 3270 AID character received in the last map in request.

**FIELD=**

Moves field-related information to the issuing program.

***field-name***

Specifies the name of the map field being tested. The following options can be used with FIELD:

- **INDEX=index** specifies the occurrence of the field if *field-name* is a multiply-occurring field. *Index* is either a register or the symbolic name of a user-defined field that contains the subscript or an absolute expression.
- **INLEN=field-length.** requests that the system return the entered length, in bytes, of the specified map field to the issuing program. *Field-length* is a register or the symbolic name of a halfword or fullword user-defined field to which the system will return the length.

**FLIST=**

Specifies a list of map fields to be tested, as indicated by the FOR parameter, described below. The FLIST-parameter values must be enclosed in parentheses. Each field specification must be coded on a separate line. The FLIST parameters are:

- *Field-name* is the name of the map data field to be tested.
- INDEX= specifies the occurrence of the field if *field-name* is a multiply-occurring field. *Index-register* is a register or the symbolic name of a user-define field that contains the subscript or an absolute expression.
- **PLIST=** (optional); indicates the location in which the system will build the field parameter list.
- **SYSPLIST** (default); is the symbolic name of the storage area in which the system will build the field parameter list.
- *Parameter-value-list-pointer* is a register that points to the area or the symbolic name of the area.

**FOR=**

Specifies the map data fields to which the test applies.

**CURRENT**

Specifies that the test applies only to the current data field; that is, the data field that was referenced in the last #MAPMOD or #MAPINQ statement issued by the program. If the last #MAPMOD or #MAPINQ statement specified a field list, no currency exists.

**ALL**

Specifies that the test is true if all map data fields meet the specified condition.

**NONE**

Specifies that the test is true if none of the map data fields meet the specified condition.

**SOME**

Specifies that the test is true if more than one, but not all of the map data fields meet the specified condition.

**ANY**

Specifies that the test is true if one or more of the map data fields meet the specified condition.

**ALLBUT**

Specifies that the test is true if all map fields except for the named field meet the specified condition.

**NTCURFLD**

Specifies that the test is true if all map fields except the current field meet the specified condition.

**IF=**

Specifies the input test condition. For each condition, the associated label specifies the symbolic name of the routine in the issuing program to which the system will pass control if the tested condition is true. The IF-parameter condition and label must be enclosed in parentheses.

**DATANO**

Determines if the terminal operator did not enter data in the named map fields. This condition is true if the field has not been modified or if it had been modified but the INDATA=NO option was in effect for the field during the last #MREQ IN request.

**DATAYES**

Determines if the terminal operator entered data in the named map fields.

**DATAERAS**

Determines if the data has been erased from the named map fields using 3270 local editing features. In this case, the data fields would remain unchanged unless a padding character had been specified, which would fill the field with that character.

**DATATRUN**

Determines if the data has been truncated in the named map fields. A field that has been truncated would also fulfill the condition DATAYES, described above.

**EDITERR**

Determines if the named map fields were found to be in error during automatic editing. To test for this condition, automatic editing must be enabled for the map and for each of the named map fields.

**EDITCOR**

Determines if the named map fields were found to be correct during automatic editing. To test for this condition, automatic editing must be enabled for the map and for each of the named map fields.

**DATAIDEN**

Tests whether input data is *identical* to map data currently in program variable storage. DATAIDEN is *true* in either of the following cases:

- The field's modified data tag (MDT) is off. On mapin, the MDT is off if the user did not type any characters in the field, a previous modify map did not set it, or the map specifies N to MDT on Y/N.
- The field's MDT is on, but each character that the user typed in is identical (including capitalization) to the data in variable storage.

**DATADIFF**

Tests whether input data is *different* from map data currently in program variable storage. DATADIFF is *true* if the field's MDT is on *and* at least one input character differs from the data in variable storage.

### Example

The following example of the #MAPINQ statement tests for whether the terminal operator entered data in more than one, but not all of the fields described in the FLIST parameter. If this condition is true the program branches to the label CHECFLDS. A false condition returns control to the next executable instruction.

```
#MAPINQ MRB=BLOCK1,FLIST=(FIELD,SCREENF2,                    *
      FIELD,SCREENF3,                                        *
      FIELD,SCREENF4,                                        *
      FIELD,SCREENF5),                                       *
      FOR=SOME,IF=(DATAYES,CHECFLDS)
```

**Status Codes**

The #MAPINQ request is unconditional; any return code other than X'00' will result in an abend of the issuing task.

## #MAPMOD—requests that the system modify options in the map request block

The #MAPMOD statement requests that the system modify options in the map request block (MRB) for a map; modifications can be designated as permanent or temporary. Requested revisions can be field-specific and/or non field-specific. Field-specific revisions apply to the map's variable data fields, not to literal fields.

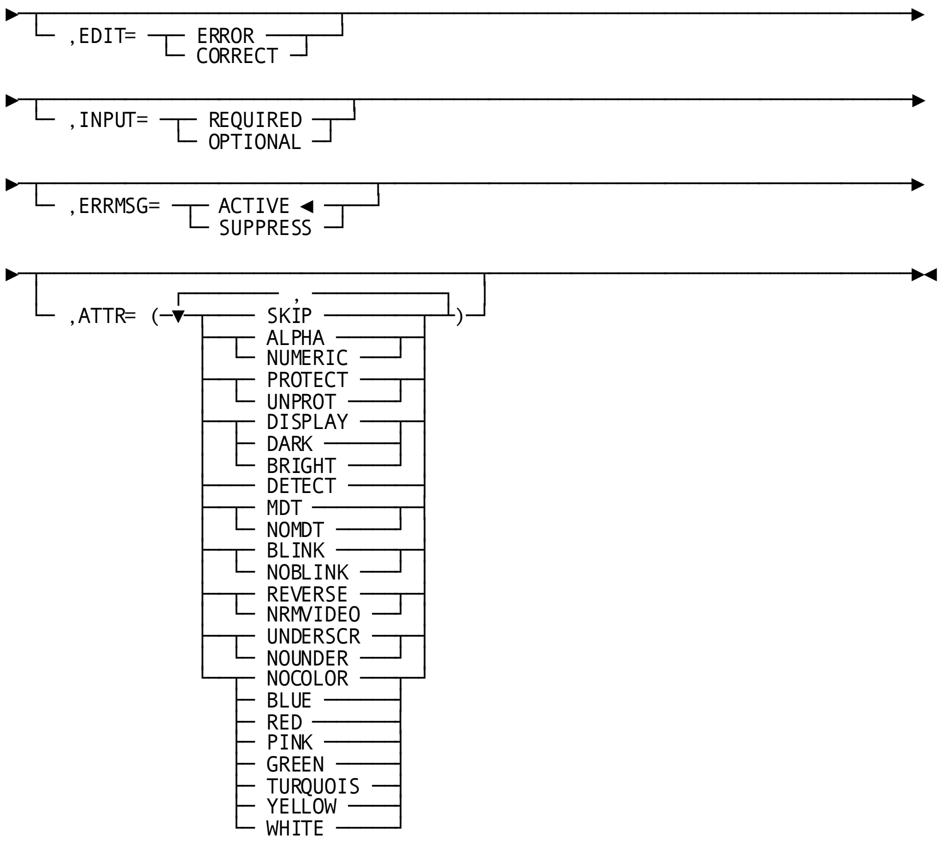
The following considerations apply:

- If modification of one field is necessary, the FIELD, MRB, and the optional PLIST parameters, described below, should be specified.
- If modification of more than one field is necessary, the FLIST, FOR, and MRBLIST parameters, described below, should be specified.
- The #MAPMOD attribute parameters revise predefined map and/or map data field attributes, and thus have no defaults. If a #MAPMOD attribute parameter is *not* specified, that parameter remains set to the value specified at map generation or to the value set with a previously issued #MAPMOD statement specifying TYPE=PERM. Conflicting attributes are resolved by runtime mapping.

### #MAPMOD Syntax

```
►►─── #MAPMOD ──────────────────────────────────────────────────────────►

►─┬───────────────────────────────┬─────────────────────────────────────►
  └─ TYPE= ──┬─ PERM ◄ ─┬─────────┘
             └─ TEMP ───┘

►─── ,MRB=map-request-block-pointer ────────────────────────────────────►
```

```
►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,PLIST= ──┬─ SYSPLIST ◄ ─────────────┬┘
               └─ parameter-list-pointer ─┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,MRBLIST= ──┬─ MRBPLIST ◄ ─────────────────┬┘
                 └─ mrb-parameter-list-pointer ─┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,CURSOR= ( ─┬─ cursor-row,cursor-column ─────────────────────┬─ ) ─┘
                 └─ FIELD,fieldname ─┬──────────────────────────┬─┘
                                     └─ ,INDEX,index-register ─┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,WCC= ─┬──┬─ RESETMDT ─┬──┬┘
            │  └─ NOMDT ────┘  │
            ├──┬─ RESETKBD ─┬──┤
            │  └─ NOKBD ────┘  │
            ├──┬─ ALARM ──────┤
            │  └─ NOALARM ─┘   │
            ├──┬─ STARTPRT ─┬──┤
            │  └─ NOPRT ────┘  │
            └──┬─ NLCR ─┬──────┘
               ├─ 40CR ─┤
               ├─ 64CR ─┤
               └─ 80CR ─┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,FIELD=field-name ─┬───────────────────────────┬┘
                        └─ ,INDEX=index-register ─┘

►─┬───────────────────────────────────────────────────────────────────────────►
  │                ┌──────────────────────────────────────────────┐           │
  └─ ,FLIST= (─▼─ FIELD,field-name ─┬──────────────────────────┬─┴─)─────────┘
                                    └─ ,INDEX,index-register ─┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,FOR= ─┬─ ALL ────────┬┘
            ├─ ERROR ─────┤
            ├─ CORRECT ───┤
            ├─ CURRENT ───┤
            ├─ NOTCURNT ──┤
            ├─ FLIST ─────┤
            └─ NOTFLIST ──┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,BACKSCN= ──┬─ YES ─┬┘
                 └─ NO ──┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,OUTDATA= ─┬─ YES ─────────┬┘
                ├─ NO ─────────┤
                ├─ ERASE ──────┤
                └─ ATTRibute ──┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,INDATA= ──┬─ YES ─┬┘
                └─ NO ──┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,JUSTIFY= ──┬─ RIGHT ─┬┘
                 └─ LEFT ──┘

►─┬───────────────────────────────────────────────────────────────────────────►
  └─ ,PAD= ─┬─ NO ─────────────────┬┘
            ├─ C'pad-character' ──┤
            └─ X'pad-character' ──┘
```



## #MAPMOD Parameters

**MRB=**

Specifies the storage area associated with the MRB of the map that is being modified. This storage area is of variable length according to the number of fields included in the map; it is copied into program variable storage by the #MRB statement.

***map-request-block-pointer***

A register that points to the MRB area or the symbolic name of that area.

*Note: Map-request-block* cannot be a register if the FIELD=*field-name* operand is also specified in the #MAPMOD statement.

**TYPE=**

Specifies whether the modifications are to be permanent or temporary.

**PERM**

(Default); specifies that modifications apply to all mapping mode I/O requests issued until the program terminates or until a subsequent #MAPMOD request overrides the requested revisions.

**TEMP**

Specifies that modifications will apply only to the next #MREQ request.

**PLIST=**

Indicates the location of the storage area in which the system will build the field parameter list specified by the FLIST parameter, described below.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system will build the field parameter list.

***parameter-value-list-pointer***

A register that points to the area or the symbolic name of the area.

**MRBLIST=**

Indicates the location of the 20-fullword storage area that is substituted for the DC/UCF portion of SUBSCHEMA-CTRL. It is generated at the bottom of the first map request block in the program.

**MRBPLIST**

(Default) is the symbolic name of the storage area that will be substituted for the DC/UCF portion of SUBSCHEMA-CTRL.

***mrb-parameter-list-pointer***

A register that points to the area or the symbolic name of the area.

**CURSOR=**

Identifies the screen location at which the cursor will be positioned during output operations.

***cursor-row,cursor-column***

Specifies the row and column on the terminal screen to which the cursor will be moved. *Cursor-row* is a numeric literal indicating the row value. *Cursor-column* is a numeric literal indicating the column value.

***field-name***

Specifies the field to which the cursor will be moved. *Field-name* is the name of a map data field.

***index-register***

Optionally specifies the occurrence of the field if *field-name* is a multiply-occurring field. *Index* is either a register or the symbolic name of a user-defined field that contains the subscript or an absolute expression.

**WCC=**

Specifies the write-control character (WCC) options requested for the output operation. The WCC is a single byte transmitted with a screen during a #MREQ OUT, that indicates the functions that the 3270 control unit is to perform as it displays the information on the screen.

If a #MAPMOD request alters any WCC option, the system resets unspecified options to the following values:

- NOMDT
- NOKBD
- NOALARM

Multiple WCC parameter values must be enclosed in parentheses and separated by commas.

**RESETMDT/NOMDT**

Specifies whether the modified data tags (MDTs) for the map fields will be reset to *off* automatically when the map is displayed. If RESETMDT is specified, the contents of variable fields are transmitted to storage only if the terminal operator modified the field or if the MDT has been set programmatically.

**RESETMDT**

States that the MDTs will be reset (turned off).

**NOMDT**

States that the MDTs will not be reset.

**RESETKBD/NOKBD**

Specifies whether the keyboard will be unlocked automatically when the map is displayed.

**RESETKBD**

States that the keyboard will be unlocked.

**NOKBD**

States that the keyboard will not be unlocked.

**ALARM/NOALARM**

Specifies whether the terminal audible alarm, if installed, will sound automatically when the map (for example, a screen that displays error messages), is displayed.

**ALARM**

States that the alarm will sound.

**NOALARM**

States that the alarm will not sound.

**STARTPRT/NOPRT**

(3280 printers only); specifies whether the contents of the terminal buffer will be printed automatically when the map is displayed.

**STARTPRT**

States that the contents of the terminal buffer will be printed.

**NOPRT**

States that the contents of the terminal buffer will not be printed.

**NLCR/40CR/64CR/80CR**

Specifies the characters-per-line formatting for 3280 printer output, meaningful only if the STARTPRT option, described above, has been specified.

**NLCR**

States that no line formatting will be performed on the printer output. Printing will begin on a new line only if the printer encounters new line (NL) and carriage control (CR) characters.

**40CR**

States that the contents of the 3280 print buffer will be printed at 40 characters per line.

**64CR**

States that the contents of the 3280 print buffer will be printed at 64 characters per line.

**80CR**

States that the contents of the 3280 print buffer will be printed at 80 characters per line.

**FIELD/FLIST**

Specifies one or more map fields to be modified. Choose one of these parameters to change field-specific options such as FOR, BACKSCN, OUTDATA, INDATA, JUSTIFY, PAD, EDIT, INPUT, and ATTR.

**FIELD=**

Specifies one map field to be modified.

**FIELD**

Specifies that one map field is to be modified.

***field-name***

Is the name of the map data field to be modified.

***index***

Specifies the occurrence of the field if *field-name* is a multiply-occurring field. *Index* is a register, the symbolic name of a user-defined field that contains the subscript, or an absolute expression.

**FLIST=**

Specifies a list of map fields to be modified or to be excluded from modification, as indicated by the FOR=FLIST and FOR=NOTFLIST parameters described below. The FLIST parameter values must be enclosed in parentheses. Each field specification must be coded on a separate line. Specify each field by using the following parameters.

***field-name***

Is the name of the map data field to be modified.

***index-register***

Specifies the occurrence of the field if *field-name* is a multiply-occurring field. *Index-register* is a register, the symbolic name of a user-defined field that contains the subscript, or an absolute expression.

**FOR=**

Specifies the map fields to be modified or excluded from modification:

**ALL**

Modifies all fields.

**ERROR**

Modifies those fields found to be in error during automatic editing.

**CORRECT**

Modifies those fields found to be correct during automatic editing.

**CURRENT**

Modifies only the field found to be current during automatic editing. The current field is the map field that was referenced in the last #MAPMOD or #MAPINQ request issued by the program. A #MAPMOD or #MAPINQ that specifies a field list does not establish currency.

**NOCURNT**

Modifies all the fields except the current field during automatic editing. The current field is the map field that was referenced in the last #MAPMOD or #MAPINQ request issued by the program. A #MAPMOD or #MAPINQ that specifies a field list does not establish currency.

**FLIST**

Modifies all the fields in the field list defined by the FLIST parameter above.

**NOTFLIST**

Modifies all fields except those in the field list defined by the FLIST parameter above.

**BACKSCN=**

Specifies whether the system is to backscan the specified field to remove trailing blanks before performing the map output operation.

**YES**

Requests that the system send all characters up to the last nonblank character to the terminal; fields remaining on the screen will contain whatever characters were present before the #MREQ request was issued. If the #MREQ request specifies the NEWPAGE option, the system erases the contents of all map data fields.

**NO**

Requests that the system leave in trailing blanks.

**OUTDATA=**

Indicates whether map fields will be set to the value of the corresponding variable-storage data fields.

**YES**

Specifies that the value of the variable storage field will be mapped out to the map field.

**NO**

Specifies that data from the record buffer as well as the attribute byte will not be mapped out.

**ERASE**

Requests that the system erase the map data fields.

**ATTRIBUTE**

Requests that the system transfer only the attribute byte from the record buffer to the map field.

**INDATA=YES/NO**

Indicates whether the map fields will be moved automatically to the corresponding variable-storage data fields (YES) or left unchanged (NO) during an input operation.

**JUSTIFY=RIGHT/LEFT**

Indicates whether the variable-storage field should be right or left justified on input.

**PAD=**

Indicates whether the alphanumeric variable-storage data field should be padded on input and defines the pad value or character:

**NO**

Does not pad the field.

***pad-character***

Pads the field with the specified pad character on the left if JUSTIFY=RIGHT is specified and on the right if JUSTIFY=LEFT is specified. *Pad-character* is a binary numeric literal pad-character value.

**EDIT=ERROR/CORRECT**

Explicitly sets the error flag on (ERROR) or off (CORRECT) for the specified map fields. If this parameter is specified, automatic editing must be enabled for the map and for the named map fields.

The ability to set the error flag enables programs to perform their own editing and validation in addition to that provided by the automatic editing feature.

**INPUT=**

Specifies whether the terminal operator will be required to add input in the specified map fields.

**REQUIRED**

Specifies that input is required. An error results if the terminal operator fails to enter data in a required field.

**OPTIONAL**

Specifies that input is optional. An error will not result if the terminal operator fails to enter data in an optional field.

**ERRMSG**

**ACTIVE**

(Default); enables display of the error message associated with the field.

**SUPPRESS**

Disables display of the error message associated with the field. If the map is redisplayed because of errors, the message defined for the map field will not be displayed even if the field contains edit errors. You typically enable display of a message only after specifying ERRMSG=SUPPRESS for the map in a previous #MAPMOD TYPE=PERM statement.

**ATTR=**

Specifies the 3270 and 3279 attributes for the named map fields. Multiple ATTR parameter values must be enclosed in parentheses and separated by commas. Only the named attributes will be modified in the MRB. ATTR options are.

**SKIP**

Requests that the system reposition the cursor automatically over the ma fields to the next unprotected field. When SKIP is specified, the named map fields are implicitly assigned the NUMERIC and PROTECT attributes (described below) automatically.

**ALPHA/NUMERIC**

Specifies whether the data input to the map fields by the terminal operator are alphanumeric (any character on the 3270 terminal keyboard) or numeric. ALPHA cannot be specified if SKIP has been specified.

**PROTECT/UNPROT**

Specifies whether or not map fields will be protected from data entry or modification by the terminal operator. UNPROT cannot be specified if SKIP has been specified.

**DISPLAY/DARK/BRIGHT**

Specifies how map fields are displayed.

**DISPLAY**

Specifies that the map fields will be displayed with normal intensity. DISPLAY cannot be specified if DETECT, described below, has been specified.

**DARK**

Specifies that the map fields will not be displayed. DARK cannot be specified if DETECT, described below, has been specified.

**BRIGHT**

Specifies that the map fields will be displayed with bright intensity. BRIGHT cannot be specified if DETECT, described below, has been specified.

**DETECT**

Specifies that the map fields will be light-pen-detectable. All fields assigned the BRIGHT attribute will automatically be detectable by a light pen.

**MDT/NOMDT**

Specifies whether MDTs are automatically set (turned on) for the map field when displayed.

**MDT**

Requests that the system automatically set the MDT for the map fields when displayed.

**NOMDT**

Requests that the system not automatically set the MDT for the map fields when displayed.

**BLINK/NOBLINK**

(3279 terminals only); specifies whether map fields will be displayed with blinking characters.

**BLINK**

Specifies that the fields characters will blink.

**NOBLINK**

Suppresses blinking.

**REVERSE/NRMVIDEO**

(3279 terminals only); specifies whether map fields will be displayed in reverse video; dark characters on a light background.

**REVERSE**

Indicates that map fields will be displayed in reverse video.

**NRMVIDEO**

Specifies that the map fields will be displayed in normal video; light characters on a dark background.

**UNDERSCR/NOUNDER**

(3279 terminals only); specifies whether the map fields are displayed with underlined characters.

**UNDERSCR**

Specifies that the map fields will be displayed with underlined characters.

**NOUNDER**

Specifies that the map fields will be displayed with nonunderlined characters.

**NOCOLOR/BLUE/RED/PINK/GREEN/TURQUOIS/YELLOW/WHITE**

(3279 terminals only); specifies that the map fields will be displayed with no color attribute or with one of the seven available color attributes.

Note: The BLINK/NOBLINK, REVERSE/NRMVIDEO, and UNDERSCR/NOUNDER options are mutually exclusive; the last attribute specified will override any previously specified attribute.

## #MAPMOD Status Codes

The #MAPMOD request is unconditional; any return code other then X'00' will result in an abend of the issuing task.

## #MAPMOD Example

The example of the #MAPMOD statement shown below performs the following functions:

- Identifies BLOCK1 as the storage area associated with the MRB of the map that is being modified
- Accepts the default of setting the modifications listed in this statement as permanent until the program terminates or another #MAPMOD statement is issued
- Accepts the default of MRBPLIST as the symbolic name of the storage area that will be substituted for the DC/UCF portion of SUBSCHEMA-CTRL
- Identifies the initial position of the cursor during a map out operation on the first position of the field SCREENF1
- Defines the WCC character options requested for output operations
- Specifies that all the fields listed in the FLIST parameter are to be modified
- Specifies that during an output operation the screen fields associated with the fields listed in the FLIST parameter are to be set to the value of the storage fields
- Specifies that during an input operation the storage fields are to be set to the value of the corresponding screen fields
- Specifies that the storage fields will be left justified on input
- Specifies that on input the storage fields will be padded on the right with blank spaces
- Specifies that input is optional
- Specifies the 3270 attributes for the specified map fields

```
#MAPMOD MRB=BLOCK1,CURSOR=(SCREENF1),WCC=(NOMDT,RESETKDB,      *
      NOALARM,NOPRT),FLIST=(FIELD,SCREENF1,                    *
      FIELD,SCREENF2                                           *

      FIELD,SCREENF3                                           *
      FIELD,SCREENF4),                                         *
      FOR=FLIST,OUTDATA=YES,INDATA=YES,                        *
      JUSTIFY=LEFT,PAD=C' ',INPUT=OPTIONAL,                    *
      ATTR=(SKIP,BRIGHT,UNDERSCR)
```

The following #MAPMOD statement shows how to suppress display of default error messages for fields EMPID and DEPTID on the current map.

```
#MAPMOD TYPE=TEMP,MRB=MAPMRB,                                           *
      FLIST=(FIELD,EMPID FIELD,DEPTID),                                 *
      FOR=FLIST,ERRMSG=SUPPRESS
```

Because this #MAPMOD statement specifies TEMP, error messages for these fields are suppressed for the next mapout only. If PERM (default) were used, the error messages would be suppressed until the program terminated or until the error message specifications were overridden by a subsequent #MAPMOD statement.

## @MODIFY—replaces element values of the database record

The @MODIFY statement replaces element values of the specified database record with new element values present in program variable storage.

Before execution of the @MODIFY statement, the following conditions must be met:

- All areas affected, either implicitly or explicitly, must be readied in one of the update usage modes (see @READY (see page 308) in this chapter).
- The named record must be established as current of run unit. If the record that is current of run unit is not an occurrence of the named record, an error condition results.
- The values of all elements defined for the named record in the subschema view must be in variable storage. If the @MODIFY statement is not preceded by an @OBTAIN statement, you must initialize the appropriate values. It is recommended that you issue an @OBTAIN statement to ensure that all the elements in the modified record are present in variable storage before you alter the values, then issue the @MODIFY statement.

**Modifying CALC- and Sort-Control Elements**

The following special considerations apply to the modification of CALC- and sort-control elements:

- If modification of a CALC- or sort-control element will violate a duplicates-not-allowed option, the record is not modified and an error condition results.
- If a CALC-control element is modified, successful execution of the @MODIFY statement enables the record to be accessed on the basis of its new CALC-key value. The db-key of the specified record is not changed.

- If a sort-control element is to be modified, the sorted set in which the named record participates must be included in the subschema invoked by the program. A record occurrence that is a member of a set not defined in the subschema can be modified if the undefined set is not sorted.
- If any of the modified elements in the specified record are defined as sort-control elements for any set occurrence in which that record is currently a member, the set occurrence is examined. If necessary, the specified record is automatically disconnected and reconnected in the set occurrence to maintain the set order specified in the schema.

**Native VSAM Considerations**

The following special considerations apply to the modification of records in native VSAM data sets:

- The length of a record in an entry-sequenced data set (ESDS) cannot be changed even if the record is variable length.
- The prime key for a key-sequenced data set (KSDS) cannot be modified.

**Currency**

Before execution of the @MODIFY statement:

- The specified record must be established as current of run unit. If the record that is current of run unit is not an occurrence of the specified record, an error condition results.
- The values of all elements defined for the named record in the program's subschema view must be in variable storage. If the @MODIFY statement is not preceded by an @OBTAIN statement, the programmer must initialize the appropriate values. The best practice is to issue an @OBTAIN statement to ensure that all the elements in the modified record are present in variable storage before altering the values as desired and then issue the @MODIFY statement.

Following a successfully executed @MODIFY statement, the modified record becomes current of its run unit, record type, area, and all sets in which in participates as owner or member.

## @MODIFY Syntax

```
►►─── @MODIFY REC=record-name ──────────────────────────────────────►◄
```

## @MODIFY Parameters

**REC=*record-name***

Defines the named record occurrence, as specified in program variable storage. *Record-name* must specify a record type included in the subschema.

## @MODIFY Status Codes

After completion of the @MODIFY function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0800**

The request has been serviced successfully.

**0804**

The OCCURS DEPENDING ON item is less than 0 or greater than the maximum number of occurrences of the control element.

**0805**

Modification of the record would violate a duplicates-not-allowed option for a CALC record, a sorted set, or an index set.

**0806**

Currency has not been established for the specified record.

**0808**

The specified record cannot be found. The record name has probably been misspelled.

**0809**

The specified record's area has not been readied in one of the three update usage modes.

**0810**

The subschema specifies an access restriction that prohibits modification of the named record.

**0811**

There is insufficient space to hold the modified variable-length record occurrence.

**0813**

A current record of run unit has not been established or has been nullified by a previous @ERASE statement.

**0818**

The record has not been bound.

**0820**

The current record of run unit is not the same type as the specified record.

**0821**

An area other than the area of the named record has been readied with an incorrect usage mode.

**0825**

No current record of set type has been established.

**0833**

All sorted sets in which the specified record participates have not been included in the subschema.

**0855**

An invalid length has been defined for a variable-length record.

**0860**

A record occurrence has been encountered whose type is inconsistent with the set named in the ERROR-SET field of the IDMS communications block. Probable causes are either a broken chain and improper database description.

**0861**

No record can be found for an internal db-key. This code usually indicates a broken chain.

**0883**

Either the length of a record in a native VSAM ESDS has been changed, or a prime key in native VSAM KSDS has been modified.

## @MODIFY Example

The following example illustrates the steps involved in modifying an occurrence of the EMPLOYEE record. Assume that the employee name is to be changed. The first step is to retrieve the desired EMPLOYEE record and move its contents to variable storage by using the statements shown below:

```
MVC    EMPID,INEMPID
@OBTAIN CALC,REC='EMPLOYEE'
```

The next step is to update the value of the EMPLOYEE field by moving the new employee name into the proper location in the EMPLOYEE record:

```
MVC    EMPNAME,NEWNAME
```

The final step is to issue an @MODIFY statement to return all data items in the EMPLOYEE record to the database:

```
@MODIFY REC='EMPLOYEE'
```

## @MODIFY (LRF)—changes field values of an existing logical-record occurrence

The @MODIFY statement changes field values of an existing logical-record occurrence. LRF uses the field values present in the variable-storage location reserved for the logical record to update the appropriate database records in the database. You can optionally specify an alternative variable-storage location from which the changed field values are to be taken.

### @MODIFY (LRF) Syntax

```
►►── @MODIFY REC=logical-record-name ────────────────────────────────►

►─┬──────────────────────────────────────┬──────────────────────────►
  └─ ,IOAREA=alt-logical-record-location ┘

►─┬─────────────────────────────────────────────────┬───────────────►
  └─ ,ONLRSTS=path-status,GOTO=branch-location ─────┘

►─┬────────────────────────────────┬────────────────────────────────►◄
  └─ ,WHERE boolean-expression ────┘
```

### @MODIFY (LRF) Parameters

**REC=*logical-record-name***

Defines the logical record. Unless the IOAREA clause is specified (see below), the field values used to update the database are taken from the area in program variable storage reserved for the specified logical record. *Logical-record-name* must specify a logical record defined in the subschema.

**IOAREA=*alt-logical-record-location***

Identifies an alternative variable-storage location from which the field values are to be obtained to perform the requested modification. When modifying a logical record that was retrieved into an alternative location in variable storage, you should use the IOAREA clause to name the same location specified in the @OBTAIN request. If the IOAREA clause is included in the @MODIFY statement, *alt-logical-record-location* must identify a record location defined in the program.

**ONLRSTS=*path-status*,GOTO=*branch-location***

Tests for the indicated path status. *Path-status* must be a quoted literal or program variable (1 to 16 bytes under z/OS and OS/390 or 1 to 6 bytes under the z/VSE operating system). If *path-status* results from this @MODIFY statement, the action specified by GOTO=*branch-location* is performed. See ON Clause (see page 393) in this chapter for details.

**WHERE boolean-expression**

Specifies the selection criteria to be applied to the named logical record. See WHERE Clause (see page 388), later in this chapter, for details.

## @MODIFY (LRF) Status Codes

When using LRF, the type of status code returned to the program in the ERRSTAT field of the IDMS communications block differs according to the type of error. If the error occurs in the logical-record path, the ERRSTAT field contains a status code issued by CA IDMS/DB with a major code from 00 to 19. For a list of these codes, see ERRSTAT Field and Codes (see page 41).

When the error occurs in the request itself, LRF returns the path status LR-ERROR to the LRSTAT field of the LRC block and places a status code with a major code of 20 in the ERRSTAT field of the IDMS communications block. For a list of these codes, see Testing for the Logical-Record Path Status (see page 55).

## @MODIFY (LRF) Example

The sample code shown below illustrates the steps taken to modify an occurrence of the EMPSKLLR logical record. Assume that the department name for department 1200 is to be changed, as well as the maximum salary for a specific job working in this department (job identification number 5051).

1. Retrieve the desired logical record:

```
@OBTAIN FIRST,REC=EMPSKLLR,                              *
         WHERE DEPTID EQ '1200'                          *
         AND JOBID EQ '5051'
```

2. Update the JOBNAME and MAXSAL fields by moving the new department name and the revised maximum salary to the proper fields in the obtained DEPJOBLR logical record:

```
MVC    JOBNAME,NEWNAME
MVC    MAXSAL,NEWSAL
```

3. Issue the @MODIFY statement for the update EMPSKLLR logical record:

```
@MODIFY REC=EMPSKLLR
```

# #MREQ—determines how data is transferred

The #MREQ statement determines how data is transferred between the terminal and program variable storage. There are three types of #MREQ statements, each performing a different type of I/O operation:

- **#MREQ IN** transfers data from the terminal device to program variable storage.
- **#MREQ OUT** transfers data from program variable storage to the terminal device.
- **#MREQ OUTIN** transfers data from program variable storage to the terminal device, followed by a transfer from the terminal device back to program variable storage.

**Native Mode Transfers**

You can also use the #MREQ statement to perform the following **native-mode data transfers**:

- Map in data from an area in variable storage that contains a 3270-like data stream to data fields defined for the map.
- Map out data to another area in variable storage.

**Synchronous and Asynchronous Requests**

All #MREQ input requests are synchronous; output requests can be either synchronous or asynchronous:

- For **synchronous requests**, control does not return to the issuing program until the I/O operation is completed. You specify a synchronous input request (the default for mapping output) by indicating YES in the CHECK parameter, as described below.
- For **asynchronous requests**, control is returned to the issuing program immediately after the requested I/O operation is initiated. The program continues to execute concurrently with the I/O operation. An ECB is established that will be posted after the I/O has been completed. The address of the ECB is contained in register 1.

  To ensure that the previous #MREQ processing has been completed before you issue an #MREQ request, your program must issue a #TREQ CHECK following asynchronous data transfer.

  Note: For more information about the #TREQ CHECK statement, see #TREQ (see page 343) later in this chapter.

To transfer data immediately from program variable storage to the terminal, your program can issue a write-direct-to-terminal #MREQ OUT request (blast). Blast requests must be directed to 3270 devices that support mapping-mode terminal I/O operations.

**Note:** For more information about mapping functions, see the *Mapping Facility Guide*.

## #MREQ Syntax

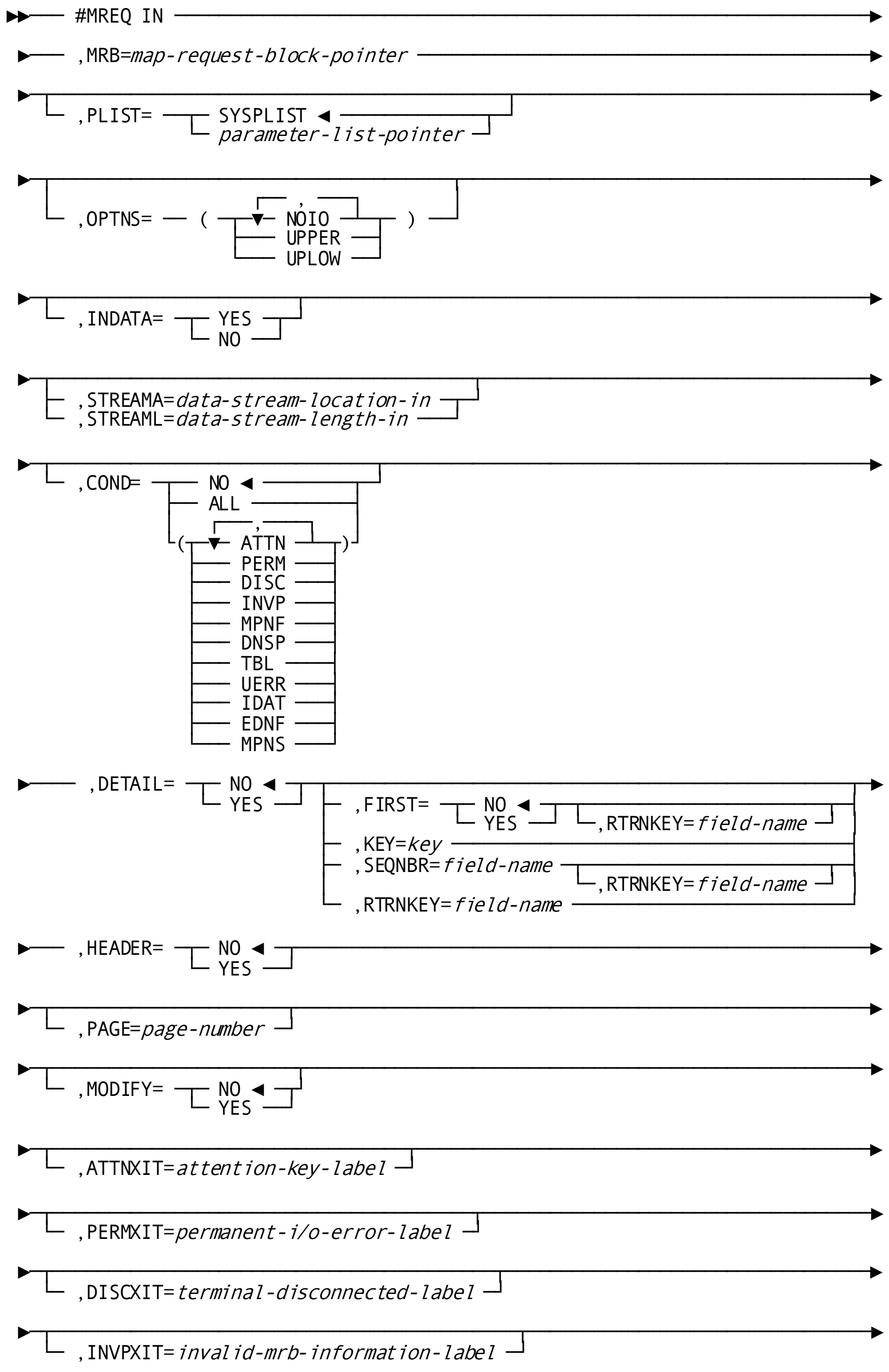
Syntax for each of the these #MREQ statements follows:

- #MREQ IN
- #MREQ OUT
- #MREQ OUTIN

Parameter descriptions follow the syntax diagrams.

**Syntax**

*#MREQ IN*



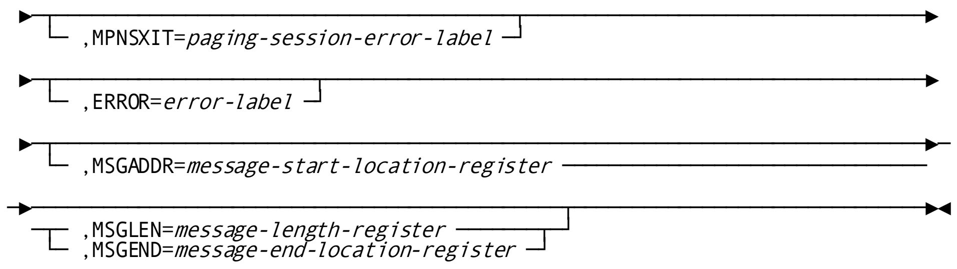
```
►─┬──────────────────────────────────────┬────────────────────────────────────►
  └─ ,MPNFXIT=map-not-found-label ───────┘

►─┬──────────────────────────────────────────────────────┬────────────────────►
  └─ ,DNSPXIT=terminal-device-not-supported-label ───────┘

►─┬──────────────────────────────────────┬────────────────────────────────────►
  └─ ,TBLXIT=error-in-table-label ───────┘

►─┬──────────────────────────────────────────────────────┬────────────────────►
  └─ ,UERRXIT=error-in-return-user-edit-mod-label ───────┘

►─┬────────────────────────────────────────────┬──────────────────────────────►
  └─ ,IDATXIT=internal-data-error-label ───────┘

►─┬───────────────────────────────────────────────┬───────────────────────────►
  └─ ,EDNFXIT=edit-module-not-found-label ────────┘

►─┬─────────────────────────────────────────────┬─────────────────────────────►
  └─ ,MPNSXIT=paging-session-error-label ───────┘

►─┬─────────────────────────────┬─────────────────────────────────────────────►◄
  └─ ,ERROR=error-label ────────┘
```

**Syntax**

*#MREQ OUT*

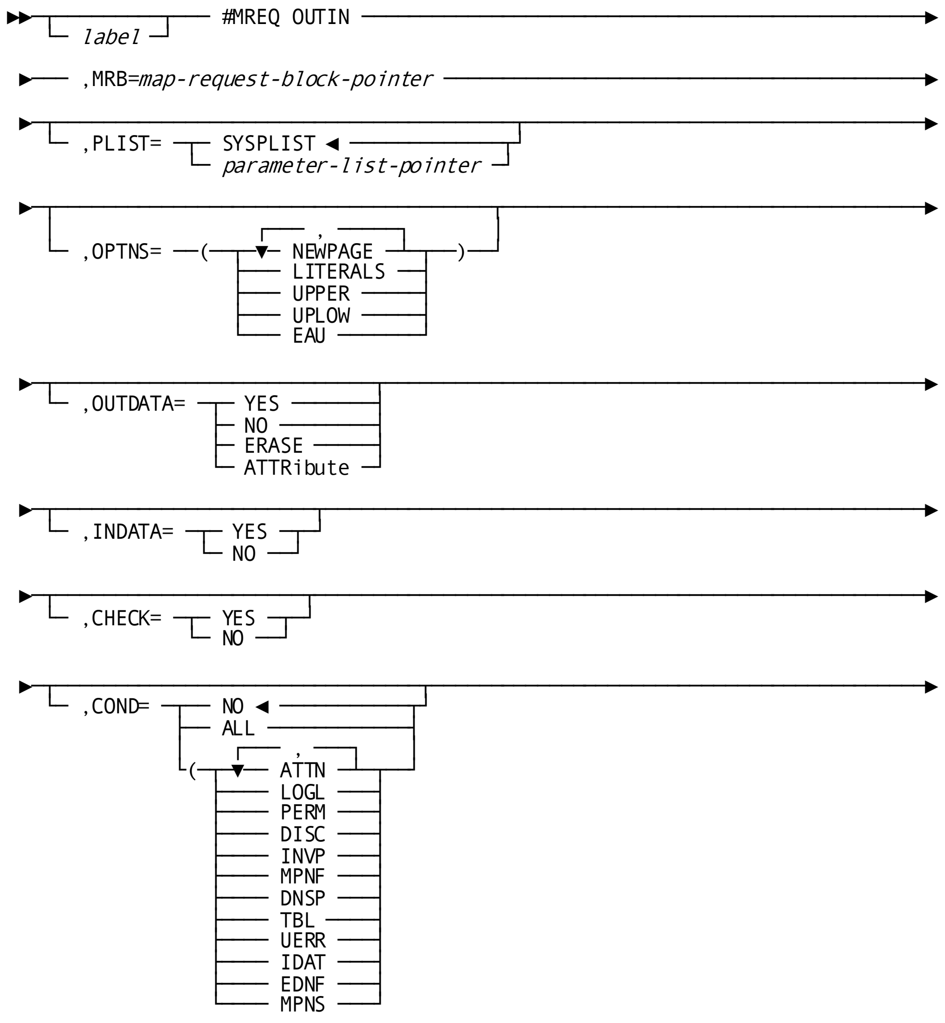
```
►►─┬─────────────┬─── #MREQ OUT ──────────────────────────────────────────────►
   └─ label ─────┘

►─── ,MRB=map-request-block-pointer ──────────────────────────────────────────►

►─┬─────────────────────────────────────────────────┬─────────────────────────►
  └─ ,PLIST= ──┬── SYSPLIST ◄ ──────────────────────┤
               └── parameter-list-pointer ──────────┘

►─┬─────────────────────────────────────────────────────────────────┬─────────►
  │                      ┌────────── , ───────────┐                 │
  └─ ,OPTNS= ── ( ───────▼──┬── NEWPAGE ──────┬───┴──── ) ──────────┘
                            ├── LITERALS ─────┤
                            ├── NOIO ─────────┤
                            ├── SAVE ─────────┤
                            └── EAU ──────────┘

►─┬─────────────────────────────────────────┬─────────────────────────────────►
  └─ ,OUTDATA= ──┬── YES ───────────────────┤
                 ├── NO ────────────────┤
                 ├── ERASE ─────────────┤
                 └── ATTRibute ─────────┘

►─┬────────────────────────────────────────────────────────────────────────┬──►
  └─ ,DETAIL= ──┬── NO ◄ ──┬─┬───────────────────────────────────────────┬─┘
                └── YES ───┘ └─ ,UPDATE= ──┬── NEW ◄ ──┬─┬──────────────┬─┘
                                           └── CURRENT ┘ └─ ,KEY=key ───┘

►─── ,RESUME= ──┬── NO ◄ ──┬─┬───────────────────────────────────────────┬────►
                └── YES ───┘ └─ ,PAGE= ──┬── CURRENT ◄ ──────────────────┤
                                         ├── NEXT ───────────────────┤
                                         ├── PRIOR ──────────────────┤
                                         ├── FIRST ──────────────────┤
                                         ├── LAST ───────────────────┤
                                         ├── page-number ────────────┤
                                         └── (page-number-pointer) ──┘
```

```
►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,CHECK= ─┬─ YES ─┬─┘
              └─ NO ──┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,STREAMA= ─┬─ (1) ◄ ──────────────────────────────────────┬─┘
                └─ return-data-stream-address-out-register ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,STREAML= ─┬─ (0) ◄ ─────────────────────────────────────┬─┘
                └─ return-data-stream-length-out-register ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  ├─ ,DESTID=destination-id-pointer ─────┤
  ├─ ,USERID=user-id-pointer ────────────┤
  └─ ,LTERMID=logical-terminal-id-pointer ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,COND= ─┬─── NO ◄ ───────────────┬─┘
             ├─── ALL ───────────┤
             │     ┌──── , ──┐   │
             └(──▼─ ATTN ──┬─)───┘
                 ├──── LOGL ──┤
                 ├──── PERM ──┤
                 ├──── DISC ──┤
                 ├──── INVP ──┤
                 ├──── MPNF ──┤
                 ├──── DNSP ──┤
                 ├──── TBL ───┤
                 ├──── UERR ──┤
                 ├──── IDAT ──┤
                 ├──── EDNF ──┤
                 ├──── UNDF ──┤
                 └──── MPNS ──┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,ATTNXIT=attention-key-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,LOGLXIT=logical-output-error-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,PERMXIT=permanent-i/o-error-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,DISCXIT=terminal-disconnected-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,INVPXIT=invalid-mrb-information-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,MPNFXIT=map-not-found-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,DNSPXIT=terminal-device-not-supported-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,TBLXIT=error-in-table-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,UERRXIT=error-in-return-user-edit-mod-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,IDATXIT=internal-data-error-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,EDNFXIT=edit-module-not-found-label ─┘

►─┬─────────────────────────────────────────────────────────────────────►
  └─ ,UNDFXIT=invalid-destid-ltermid-label ─┘
```


,MPNSXIT=paging-session-error-label
,ERROR=error-label
,MSGADDR=message-start-location-register
,MSGLEN=message-length-register
,MSGEND=message-end-location-register

**Syntax**

*#MREQ OUTIN*



```
►─┬──────────────────────────────────────────────────────┬─►
  └─ ,ATTNXIT=attention-key-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,LOGLXIT=logical-output-error-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,PERMXIT=permanent-i/o-error-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,DISCXIT=terminal-disconnected-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,INVPXIT=invalid-mrb-information-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,MPNFXIT=map-not-found-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,DNSPXIT=terminal-device-not-supported-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,TBLXIT=error-in-table-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,UERRXIT=error-in-return-user-edit-mod-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,IDATXIT=internal-data-error-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,EDNFXIT=edit-module-not-found-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,MPNSXIT=paging-session-error-label ─┘

►─┬──────────────────────────────────────────────────────┬─►
  └─ ,ERROR=error-label ─┘

►─┬──────────────────────────────────────────────────────┬─►─
  └─ ,MSGADDR=message-start-location-register ───────────┘

─►─┬─────────────────────────────────────────────────────┬─►◄
   ├─ ,MSGLEN=message-length-register ──────┤
   └─ ,MSGEND=message-end-location-register ─┘
```

## #MREQ Parameters

**MRB=*map-request-block-pointer***

Specifies the location of the MRB for the mapping operation, as copied into program variable storage by the #MRB statement. The #MRB statement is described under #MRB (see page 419). *Map-request-block-pointer* is either a register that points to the MRB area or the symbolic name of that area.

**PLIST=**

Specifies the location of the storage area in which the system builds the #MREQ parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area.

***parameter-list-pointer***

Is either a register that points to the area or the symbolic name of the area.

**OPTNS=**

Specifies several options applicable to terminal I/O operations. Multiple OPTNS parameter values must be enclosed in parentheses and separated by commas.

**NEWPAGE**

(#MREQ OUT and #MREQ OUTIN only); requests that the system activate the erase-write mechanism to clear the contents of a screen. If NEWPAGE is not specified, the system will write over any existing screen display without first erasing it.

You can mark individual fields to be erased by using the OUTDATA=ERASE option of the #MAPMOD statement, described earlier in this chapter.

**LITERALS**

(#MREQ OUT and #MREQ OUTIN only); requests that the system transmit literal fields as well as variable-storage data fields to the terminal. If LITERALS is not specified, the system writes literal fields to the map only if NEWPAGE is specified.

**NOIO**

(#MREQ IN and #MREQ OUT only); requests that the system transfer a native-mode data stream, a 3270-like data stream that consists of user data and all device-control characters, to program storage. No terminal I/O is associated with the request:

- For **IN** requests, the native-mode data stream replaces data that would normally be read from the terminal by the system.
- For **OUT** requests, the native-mode data stream replaces data that would normally be written out to the terminal by the system.

When OPTNS=(NOIO) is specified, the STREAMA= and STREAML= parameters must also be defined, as described below.

**SAVE**

(Non-write-direct-to-terminal #MREQ OUT only); requests that the system preserve the mapped output from the #MREQ OUT request in the event that an unsolicited write-direct-to-terminal data stream is received at the issuing terminal while the map is being displayed. This option overrides the task SAVE/NOSAVE option specified during system generation.

**UPPER**

(#MREQ IN and #MREQ OUTIN only); requests that the system translate all letters in a map in request into uppercase characters.

**UPLOW**

(#MREQ IN and #MREQ OUTIN only); requests that lowercase characters are not translated into uppercase characters in a map in request. This can also be accomplished by issuing a DCUF SET UPLOW statement before starting the mapping session.

**EAU**

(#MREQ OUT and #MREQ OUTIN only); allows you to request the 3270 erase all unprotected command. This command sets all unprotected character locations to nulls, resets the MDTs for all unprotected fields, unlocks the keyboard, resets the AID key, and places the cursor at the first unprotected field. This option cannot be used with OPTNS=(NEWPAGE).

**OUTDATA=**

(#MREQ OUT and #MREQ OUTIN only); specifies how the variable-storage data fields are to be transmitted to the terminal. This specification applies to all variable-storage data fields unless overridden by an OUTDATA= clause in a previously issued #MAPMOD request.

**YES**

Transfers the contents of variable-storage data fields to the corresponding map fields.

**NO**

Requests that map fields remain unchanged.

**ERASE**

Does not transfer the contents of variable-storage data fields to the screen.

**ATTRIBUTE**

Transmits only the attribute byte of each variable-storage field to the screen. Data in the variable-storage field is not transmitted.

**INDATA=**

(#MREQ IN and #MREQ OUTIN only); specifies whether the contents of the map fields are moved automatically into variable-storage data fields. This specification applies to all variable-storage data fields unless overridden by an INDATA= clause in a previously issued #MAPMOD request.

**YES**

Transfers the contents of map fields to the corresponding variable-storage data fields.

**NO**

Does not transfer the contents of map fields to the corresponding variable-storage data fields.

**DETAIL/HEADER**

(Pageable map #MREQ IN only); specifies whether the #MREQ IN operation is to retrieve data from a detail occurrence or from the header or footer area.

Note: For more information about pageable maps, see the *Mapping Facility Guide*.

**DETAIL=**

Specifies whether the #MREQ IN operation is to retrieve data from a modified detail occurrence (modified data tag set on):

**NO**

(Default); specifies that data is not to be retrieved from a detail occurrence.

**YES**

Specifies that data is to be retrieved from a modified detail occurrence (MDT set on). By default, the *next* sequential modified detail occurrence is retrieved; a different detail occurrence can be specified by using the FIRST/KEY/SEQNBR/RTRNKEY clause.

The contents of all map fields in the detail occurrence are retrieved unless MODIFY=YES is specified for the #MREQ IN,DETAIL statement. MODIFY=YES causes only modified fields to be retrieved.

**FIRST/KEY/SEQNBR/RTRNKEY**

Specifies the detail occurrence to be retrieved. Only one option can be specified.

**FIRST=**

Specifies whether the first available modified detail occurrence is to be retrieved.

**NO**

(Default); specifies that the FIRST clause is not used to determine the detail occurrence to be retrieved.

**YES**

Retrieves the first available modified detail occurrence. An end-of-data condition results if there are no more modified detail occurrences to be retrieved.

The optional RTRNKEY=*data-field-name* parameter specifies the name of a variable field in which the system stores the key value (if any) associated with the retrieved detail occurrence. If no value is associated with the detail occurrence, the system sets *data-field-name* to 0. *Data-field-name* must be a 4-byte value (not necessarily a binary fullword).

Note: A value is associated with a detail occurrence by using the KEY parameter in an #MREQ OUT,DETAIL=YES command for that occurrence.

**KEY=*key***

Retrieves a modified detail occurrence based on the value associated with the detail occurrence. *Key* is a 4-byte variable field.

Note: A value is associated with a detail occurrence by using the KEY parameter in an #MREQ OUT,DETAIL=YES command for that occurrence.

A detail-not-found condition is returned if the specified occurrence is not a modified detail occurrence or if no detail occurrence with the specified value is found.

**SEQNBR=*data-field-name***

Retrieves a detail occurrence by sequence number. Detail occurrences are built by the application program at run time and are stored in the sequence in which they are created. *Data-field-name* is a 4-byte binary fullword field.

**RTRNKEY=*data-field-name***

(Optional); names the variable field used to store the 4-byte value (if any) of the retrieved detail occurrence. If no value is associated with the detail occurrence, *data-field-name* is set to 0. (*Data-field-name* does not have to be a binary fullword).

Note: A value is associated with a detail occurrence by using the KEY parameter in an #MREQ OUT,DETAIL=YES command for that occurrence.

**RTRNKEY=*data-field-name***

Retrieves the next sequential modified detail occurrence, and specifies the name of the variable field in which the system stores the value (if any) associated with the retrieved detail occurrence. If no value is associated with the detail occurrence, *data-field-name* is set to 0. *Data-field-name* must be a 4-byte value (not necessarily a binary fullword).

Note: A value is associated with a detail occurrence by using the KEY parameter in an #MREQ OUT,DETAIL=YES command for that occurrence.

**HEADER=**

(Pageable map #MREQ IN only); specifies whether the map in operation is to retrieve the contents of data fields in the header and footer areas.

**NO**

(Default); specifies that data from the header and footer areas is not to be retrieved.

**YES**

Specifies that data from the header and footer areas is to be retrieved.

The contents of all data fields in the header and footer areas are retrieved unless MODIFY=YES is specified in the #MREQ IN,HEADER statement; MODIFY=YES causes only modified fields to be retrieved.

**PAGE=*page-number***

Specifies the name of a numeric variable field to store the current binary fullword value of the $PAGE field on map in.

**MODIFY=**

Specifies whether the contents of modified fields are to be retrieved.

**NO**

(Default); retrieves all fields from the header and footer areas when a modified field (MDT set on) is found in the occurrence or areas.

**YES**

Retrieves only the contents of modified fields from the header and footer areas; data in unmodified fields is not retrieved.

**DETAIL/RESUME**

(Pageable map #MREQ OUT only); specifies whether the #MREQ OUT command is to create or modify a detail occurrence, or to map out a page of existing detail occurrences.

**DETAIL=**

Specifies whether the #MREQ OUT command is to create or modify a detail occurrence.

**NO**

(Default); specifies that the #MREQ OUT command does not create or modify detail occurrences.

**YES**

Specifies that the #MREQ OUT command can either create or modify individual detail occurrences. You can optionally associate a numeric key value with each occurrence.

**UPDATE=NEW/CURRENT**

Specifies whether the detail occurrence is to be created or modified.

**NEW**

(Default); creates a detail occurrence in a pageable map. Occurrences are displayed in the order in which they are created by the application program.

**CURRENT**

Modifies the detail occurrence referenced by the most recent #MREQ OUT or #MREQ IN command.

**KEY=*key***

(Optional); specifies a value to be associated with the created or modified detail occurrence. The 4-byte numeric value is not displayed on the terminal screen. *Key* is the name of the variable field that contains the database key of the database record associated with the detail occurrence.

When the KEY parameter is used with the #MREQ OUT,HEADER=YES,UPDATE=CURRENT command, the specified value replaces the value (if any) previously associated with the detail occurrence.

**RESUME=**

Specifies whether a page of detail occurrences is to be displayed on the terminal screen.

**NO**

(Default); specifies that the #MREQ OUT command does not map out a page of detail occurrences to the terminal.

**YES**

Specifies that the #MREQ OUT command maps out a page of detail occurrences to the terminal.

**PAGE=**

(Optional); determines the page of occurrences to be displayed on the terminal screen.

**CURRENT**

(Default); redisplays the current page. If no page has been displayed, the first page of the pageable map is displayed.

**NEXT**

Displays the page that follows the current page. If no page follows the current page, the current page is redisplayed.

**PRIOR**

Displays the page that precedes the current page. If no page precedes the current page, the current page is redisplayed.

**FIRST**

Displays the first available page of detail occurrences.

**LAST**

Displays the page of detail occurrences with the highest available page number.

***page-number***

Displays the numeric variable field that contains the binary fullword number of the page. A page number is previously stored in the variable field by an #MREQ IN,HEADER=YES,PAGE=*page-number* statement that names the same numeric variable field.

**(*page-number*)**

Specifies the register that contains the address of a 4-byte binary fullword field in variable storage that contains the number of the page to be displayed. *Page-number* must be enclosed in single quotes.

**CHECK=**

(#MREQ OUT and #MREQ OUTIN only); specifies whether the data transfer is synchronous or asynchronous.

**YES**

Specifies that the data transfer is synchronous. the system places the issuing task in an inactive state. When the output operation is completed, the task resumes processing according to its established dispatching priority.

**NO**

Specifies that the data transfer is asynchronous. the system returns control to the issuing program immediately after initiating the output operation and establishing an ECB to be posted when the output operation is completed.

An asynchronous transfer must be followed by a CHECK #TREQ request before another #MREQ request is issued to ensure that the previous #MREQ processing has been completed.

Note: For more information about synchronous and asynchronous processing, see #TREQ (see page 343) later in this chapter.

Specifying CHECK=NO in a #MREQ OUT statement issued before task termination frees the task resources when the task terminates; the system automatically issues a #TREQ CHECK.

**STREAMA/STREAML**

(OPTNS=(NOIO only); specifies the location and the length of the input data stream to be transmitted.

**STREAMA=**

Specifies the location of the native-mode data stream to be transmitted.

***data-stream-location-in***

Either a register that points to the data stream or the symbolic name of the area that contains the data stream.

**STREAML=**

Specifies the length of the native-mode data stream to be transmitted.

***data-stream-length-in***

A register that contains either the length or an absolute expression of the length.

**STREAMA/STREAML**

Specifies the length of the output data stream and the location to which it is returned.

**STREAMA=(1)/*return-data-stream-address-out***

Specifies the location to which the system transfers the mapped data.

**(1)**

(Default); is the register that contains the address of the location to which the system transfers the mapped data.

***return-data-stream-address-out***

Specifies the location to which the system transfers the mapped data. *Return-data-stream-address-out* is either a register or the symbolic name of a fullword user-defined area.

**STREAML=**

Specifies the location to which the system returns the length of the output data stream.

**(0)**

(Default); is the register to which the system returns the length, in bytes, of the output data stream.

***return-data-stream-length-out***

Specifies the location to which the system returns the length, in bytes, of the output data stream. *Return-data-stream-length-out* is either a register or the symbolic name of a halfword or fullword user-defined field.

**DESTID/USERID/LTERMID**

(#MREQ OUT only); specifies a write-direct-to-terminal request (blast) to either a destination, user, or logical terminal.

**DESTID=*destination-id***

Specifies a write-direct-to-terminal request to one of the following destinations defined during system generation.

- **A list of logical terminals** indicates that the system sends the #MREQ data stream specified in the OUTAREA parameter to all available terminals in the list.
- **A list of users** indicates that the system sends the #MREQ data stream specified in the OUTAREA parameter to all users in the list who are currently signed on to the system.

Note: This works only if there is a valid OUTAREA parameter for line mode (#LINEOUT) as well as for mapping mode (#MREQ).

***destination-id***

A register that points to the destination id, the symbolic name of a user-defined field that contains the destination ID, or the ID itself enclosed in quotation marks.

Note: The destination list can include different 3270 models. If a map has been generated to support a specified terminal device, the system will write the map to that device. If the targeted terminal-device type is not in the map device list, the system will ignore that terminal device.

**USERID=**

Specifies a write-direct-to-terminal request to a specific signed-on user. The system sends the #MREQ data stream specified in the OUTAREA parameter to a specific signed-on user.

***user-id***

Either a register that points to the user ID, the symbolic name of a user-defined field that contains the user id, or the ID itself enclosed in quotation marks.

**LTERMID=**

Specifies a write-direct-to-terminal request to a specific in-service terminal. The system will send the #MREQ data stream specified in the OUTAREA parameter to a specific in-service terminal.

***logical-terminal-id***

Either a register that points to the logical terminal id, the symbolic name of a user-defined field that contains the logical terminal ID, or the ID itself enclosed in quotation marks.

**COND=**

Specifies whether this #MREQ is conditional and under what conditions control should be returned to the issuing program.

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the request cannot be serviced for any of the reasons listed under *condition*.

***condition***

Specifies one or more conditions under which the system returns control to the issuing program. Multiple conditions must be enclosed in parentheses and separated by commas. You can specify one or more of the following conditions.

- **ATTN**

  The I/O is interrupted by the terminal operator pressing the ATTENTION (2471) or BREAK (teletype) key during an output operation.

- **LOGL**

  A logical error is encountered in the output data stream.

- **PERM**

  A permanent I/O error has occurred.

- **DISC**

  The dial-up line is disconnected or the terminal goes out of service.

- **INVP**

  There is an invalid parameter in the MRB.

- **MPNF**

  The map load module requested by the MRB cannot be found in the load area of the dictionary.

- **NSP**

  The requested map does not support the terminal device type being used.

- **TBL**

  The named edit or code table cannot be found or is invalid for use with the requested map.

- **UERR**

  An error has occurred in a user-written edit module.

- **IDAT**

  A data conversion error occurs where the internal map data does not match the map data description.

- **EDNF**

  The user-written edit module cannot be found or is invalid for use with the requested map.

- **UNDF**

  (#MREQ OUT only); an undefined DESTID or LTERMID is specified in an #MREQ blast request.

- **MPNS**

  A map paging #MREQ is issued when no paging session is in progress.

**ATTNXIT=*attention-key-label***

Specifies the symbolic name of the routine to which control should be returned if the I/O operation is interrupted by the terminal operator.

**LOGLXIT=*logical-output-error-label***

Specifies the symbolic name of the routine to which control should be returned if a logical error is detected in the output data stream.

**PERMXIT=*permanent-i/o-error-label***

Specifies the symbolic name of the routine to which control should be returned if a permanent I/O error occurs.

**DISCXIT=*terminal-disconnected-label***

Specifies the symbolic name of the routine to which control should be returned if the terminal line or terminal goes out of service.

**INVPXIT=*invalid-mrb-information-label***

Specifies the symbolic name of the routine to which control should be returned if the #MREQ cannot be serviced because of an invalid parameter in the MRB.

**MPNFXIT=*map-not-found-label***

Specifies the symbolic name of the routine to which control should be returned if the #MREQ cannot be serviced because the map requested by MRB cannot be found.

**DNSPXIT=*terminal-device-not-supported-label***

Specifies the symbolic name of the routine to which control should be returned if the #MREQ cannot be serviced because the terminal device in use is not supported by the requested map.

**TBLXIT=*error-in-table-label***

Specifies the symbolic name of the routine to which control should be returned if an edit or code table cannot be found or is invalid for use with the requested map.

**UERRXIT=*error-in-return-user-edit-mod-label***

Specifies the symbolic name of the routine to which control should be returned if an error has occurred in a user-written edit module.

**IDATXIT=*internal-data-error-label***

Specifies the symbolic name of the routine to which control should be returned if the internal map data does not match the map data description.

**EDNFXIT=*edit-module-not-found-label***

Specifies the symbolic name of the routine to which control should be returned if a user-written edit module cannot be found or is invalid for use with the requested map.

**UNDFXIT=*invalid-destid-ltermid-label***

(#MREQ OUT only); specifies the symbolic name of the routine to which control should be returned if an undefined DESTID or LTERMID is specified in an #MREQ OUT blast request.

**MPNSXIT=*paging-session-error-label***

Specifies the symbolic name of the routine to which control should be returned if a map paging #MREQ specification is issued when a no paging session is in progress.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

**MSGADDR=*message-start-location*,MSGLEN=*message-length*/ MSGEND=*message-end-location***

(#MREQ OUT and #MREQ OUTIN only); specifies a program-supplied message to be displayed in the map message area. The message text is a 1- to 80-character alphanumeric value. *Message-start-location* is either a register that points to the message area or the symbolic name of that area. Specify the end of the message in one of the following ways.

**MSGLEN=*message-length***

Specifies the length, in bytes, of the message output data area. *Message-length* is a register that contains the length, the symbolic name of a user-defined field that contains the length, or the length itself expressed as a numeric constant.

**MSGEND=*message-end-location***

Specifies the end of the message by referencing the next data item following the message storage area. *Message-end-location* is a register or a fullword that points to the first data item following the message storage area. This data item may be a dummy byte, a data item not associated with the output data, or the symbolic name of that data item.

## #MREQ Status Codes

By default, the #MREQ request is unconditional; any return-code other than X'00' will result in an abend of the issuing task. The issuing program can request return of control with the COND parameter to avoid an abend.

The value returned to register 15 differs according to whether the #MREQ request is a paging or a nonpaging request. Status codes issued as a result of a nonpaging #MREQ request fall in the range of '00' to '38'; paging requests return values in the range of '40' to '50'.

After completion of an #MREQ statement that does not involve pageable maps, the value in register 15 indicates the outcome of the operation. The following status codes apply to nonpageable maps:

**X'00'**

The request has been serviced successfully.

**X'04'**

The specified edit or code table cannot be found or is invalid for use with the named map.

**X'08'**

The I/O has been interrupted; the terminal operator has pressed ATTENTION (2741) or BREAK (teletype).

**X'0C'**

A logical error (for example, an invalid control character) has been encountered in the output data stream.

**X'10'**

A permanent I/O error has occurred during processing.

**X'14'**

The dial-up line for the terminal is disconnected.

**X'18'**

The terminal being used is out of service.

**X'20'**

The map request block (MRB) contains an invalid field, indicating a possible error in program parameters.

**X'24'**

The map load module named in the MRB either cannot be found in the dictionary load area (DDLDCLOD) or is invalid.

**X'28'**

The requested map does not support the terminal device type being used.

**X'2C'**

An error has occurred in a user-written edit module. An invalid pointer to the data stream has been returned to register 1.

**X'30'**

A data conversion error has occurred; the internal map data does not match the map data description.

**X'34'**

The specified user-written edit module cannot be found or is invalid for use with the named map.

**X'38'**

Invalid blast request to DESTID, LTERMID, or USER ID.

**X'3C'**

Invalid map load module.

After completion of an #MREQ function that involves pageable maps, the value in register 15 indicates the outcome of the operation: The following status codes apply to pageable maps:

**X'40'**

(#MREQ IN) The requested node for a header or detail was either not present or not updated.

(#MREQ OUT) There is no current detail occurrence to be updated. No action is taken

**X'44'**

(#MREQ IN) No more modified detail occurrences require map in.

(#MREQ OUT) The maximum amount of storage defined for pageable maps at system generation has been reached. This and any ensuing map out detail occurrences are ignored.

**X'48'**

(#MREQ IN) The scratch record containing the requested detail could not be accessed (internal error).

(#MREQ OUT) No detail occurrence, footer, or header fields exist to be mapped out by an #MREQ OUT,RESUME command.

**x'4C'**

(#MREQ OUT) The first screen page has been transmitted to the terminal.

**X'50'**

(#MREQ IN) An #MREQ IN,COND=MPNS or an #MREQ OUT,COND=MPNS request was received when no map paging session is in progress. Either a #STRTPAG command was not received prior to this #MREQ IN command or a #ROLLBAK was issued so that the scratch area for the pageable map (area ID MPGPSCRA) is no longer available. Unless the COND=MPNS is specified for #MREQ, this condition abends the map paging task with the message DC242021.

(#MREQ OUT) A mapout command was received when no map paging session was in progress. Either the #STRTPAG command was not received prior to this mapout command or a #ROLLBAK was issued so that the scratch area for the pageable map (area ID MPGPSCRA) is no longer available. This return code is received only when COND=MPNS is specified for #MREQ; otherwise, this condition abends the map paging task.

**X'54'**

(#MREQ OUT) Value returned to register 15 when a pageable map page is built before the page is actually displayed. Test for the new map paging return code after each #MREQ OUT DETAIL=YES statement. This allows you to detect when the last detail that can fit on a page has been placed on that page.

Upon successful completion of certain #MREQ requests, four registers contain the following information:

- **Register 0,** for #MREQ OUT blast requests, contains the actual number of terminals to which the data stream has been routed.
- **Register 1,** for asynchronous output requests, contains the address of the ECB that the system posts on completion of the I/O operation.
- **Register n**, for non-I/O requests (OPTNS=(NOIO) parameter), contains the *address* of the native-mode data stream. The register number *n* is assigned by the STREAMA parameter. This register does not have to be assigned for non-I/O requests; the system can place the address of the native-mode data stream in a user-defined storage area rather than in a register.
- **Register m**, for non-I/O requests, contains the *length* of the native-mode data stream. The register number *m* is assigned by the STREAML parameter. This register does not have to be assigned for non-I/O requests. The following conditions apply:
  - For output requests, the system can place the length of the native-mode data stream in a user-defined storage area.
  - For input requests, the length can be defined as an absolute expression.

## #MREQ Example

The following examples illustrate how to use the #MREQ statement:

The #MREQ IN statement shown below requests that the system read the map associated with the map request block TESTMAP1. Data values are transferred from map fields to the corresponding variable-storage data fields. Subsequent commands can evaluate the input values and perform appropriate processing. For any error condition that can be specified by the COND=ALL parameter, control will be returned to the routine labeled ERRORTN.

```
#MREQ IN,MRB=TESTMAP1,INDATA=YES,COND=ALL,ERROR=ERRORTN
```

The #MREQ IN statement shown below requests that the system map in the next (default) modified detail occurrence of the pageable map associated with the map request block TESTPAG1.

```
#MREQ IN,MRB=TESTPAG1,DETAIL=YES,MODIFY=YES,COND=ALL,             *
      ERROR=ERRORTN
```

The #MREQ OUT statement shown below requests that the system map out all literal and data fields associated with the map request block TESTMAP1. The NEWPAGE option clears the screen before transferring the TESTMAP1 data fields to the screen.

```
#MREQ OUT,MRB=TESTMAP1,OUTDATA=YES,OPTNS=(NEWPAGE)
```

The #MREQ OUT statement shown below creates a new detail occurrence and maps out a page of detail occurrences to the terminal screen. The detail occurrence can be displayed in mixed uppercase and lowercase characters. Control is returned to the ERRRTN routine if the request cannot be serviced due to any of the conditions listed under the COND options. A program-supplied message is mapped out to the map message area. Register 7 points to where the message is stored; register 4 contains the message length

```
#MREQ OUT,MRB=TESTPAG1,OPTNS=(UPLOW),DETAIL=YES,RESUME=YES,   *
      COND=ALL,ERROR=ERRRTN,MSGADDR=(R7),MSGLEN=(R4)
```

## @OBTAIN (LRF)—retrieves the named logical record

The @OBTAIN statement retrieves the named logical record and places it in the variable-storage location reserved for that logical record. The @OBTAIN statement can perform the following functions:

- Retrieve an occurrence of a logical record that meets criteria specified in the WHERE clause.
- Specify that the retrieved logical record is to be placed into an alternative variable-storage location.

## @OBTAIN (LRF) Syntax

```
►►── @OBTAIN ─┬─ NEXT ◄ ─┬─ ,REC=logical-record-name ──────────────────────►
              └─ FIRST ──┘

►─┬─────────────────────────────────────────┬───────────────────────────────►
  └─ ,IOAREA=alt-logical-record-location ───┘

►─┬────────────────────────────────────────────────┬────────────────────────►
  └─ ,ONLRSTS=path-status,GOTO=branch-location ────┘

►─┬──────────────────────────────┬──────────────────────────────────────────►◄
  └─ ,WHERE boolean-expression ──┘
```

## @OBTAIN (LRF) Parameters

**NEXT/FIRST,REC=*logical-record-name***

Retrieves a logical record and places it in program variable storage. *Logical-record-name* must specify a logical record defined in the subschema.

**NEXT/FIRST**

Specifies which occurrence of the logical record is to be retrieved.

**NEXT**

(Default); retrieves a subsequent occurrence of the named logical record. @OBTAIN NEXT is generally used to serially retrieve logical-record occurrences.

When LRF receives repeated @OBTAIN NEXT commands, it replaces field values in program variable storage with new values obtained through repeated access to database records.

If the program issues an @OBTAIN NEXT statement without issuing an @OBTAIN FIRST, or if the last path status returned for the path was LR-NOT-FOUND, LRF interprets the @OBTAIN NEXT as @OBTAIN FIRST. After LR-ERROR or a DBA-defined path status, LRF does not interpret @OBTAIN NEXT as @OBTAIN FIRST.

**FIRST**

Retrieves the first occurrence of the logical record. @OBTAIN FIRST is generally used to retrieve the first in a series of logical-record occurrences.

If an @OBTAIN FIRST statement is followed by an @OBTAIN NEXT statement to retrieve a series of occurrences of the same logical record, the @OBTAIN statements must direct LRF to the same path. For this reason, you must ensure that the selection criteria specified in the WHERE clauses accompanying the @OBTAIN FIRST and @OBTAIN NEXT statements describe the same attributes of the desired logical record.

**IOAREA=*alt-logical-record-location***

Identifies an alternative location in variable storage into which LRF is to place the retrieved logical record.

Any subsequent @MODIFY, @STORE, or @ERASE statements for a logical record placed in the named location should name that area. LRF is to obtain the data to be used to update the logical record from the named area. *Alt-logical-record-location* must identify a record location defined in the program.

**ONLRSTS=*path-status*,GOTO= *branch-location***

Tests for the indicated path status. *Path-status* is a quoted literal program variable (1 to 16 bytes). If *path-status* results from this @OBTAIN statement, the action specified by GOTO=*branch-location* is performed.

Note: For more information about how to code this clause, see ON Clause (see page 393) later in this chapter.

**WHERE boolean-expression**

Specifies the selection criteria to be applied to the specified logical record.

Note: For details about how to code the WHERE clause, see WHERE Clause (see page 388) later in this chapter.

## @OBTAIN (LRF) Status Codes

When using LRF, the type of status code returned to the program in the ERRSTAT field of the IDMS communications block differs according to the type of error:

- **If the error occurs in the logical-record path**, the ERRSTAT field contains an status code issued by CA IDMS/DB with a major code from 00 to 19. For a list of these codes, see ERRSTAT Field and Codes (see page 41).
- **If the error occurs in the request itself**, LRF returns the path status LR-ERROR to the LRSTAT field of the LRC block and places an status code with a major code of 20 in the ERRSTAT field of the IDMS communications block.

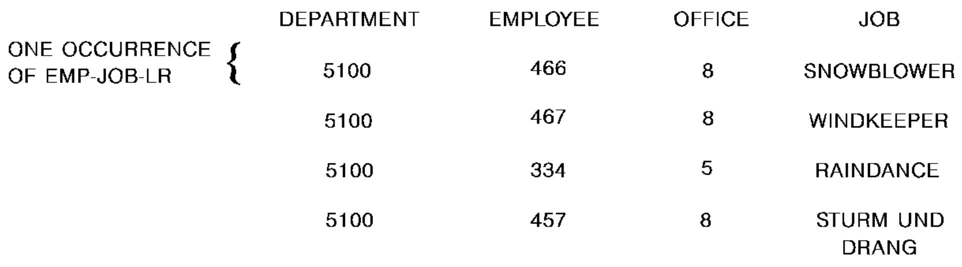
For a list of these codes, see Testing for the Logical-Record Path Status.

## @OBTAIN (LRF) Example

The @OBTAIN NEXT statement shown below retrieves a series of logical-record occurrences. The program issues the @OBTAIN NEXT statement iteratively to retrieve the first and all subsequent occurrences of the DEPEMPLR logical record for department 5100. Each @OBTAIN NEXT statement retrieves an employee ID and employee name for the department with an ID of 5100 (assuming that department 5100 has more than one employee).

```
GETEMPL  EQU   *
         @OBTAIN NEXT,REC=DEPEMPLR,                                      *
               ONLRSTS='LR-NOT-FOUND',GOTO=END,                          *
               WHERE DEPTID EQ '5100'
         .
         .
         .
         B     GETEMPL
```

The following figure illustrates how to use the @OBTAIN command in conjunction with the WHERE clause, described later in this chapter, to retrieve occurrences of the EMPJOBLR logical record. Only those detail occurrences with a department-id value equal to 5100 are retrieved. The EMPJOBLR logical record contains information from the employee, job, office, and department records. The WHERE clause is used to obtain only those employees in department 5100.



| | DEPARTMENT | EMPLOYEE | OFFICE | JOB |
|---|---|---|---|---|
| ONE OCCURRENCE OF EMP-JOB-LR | 5100 | 466 | 8 | SNOWBLOWER |
| | 5100 | 467 | 8 | WINDKEEPER |
| | 5100 | 334 | 5 | RAINDANCE |
| | 5100 | 457 | 8 | STURM UND DRANG |

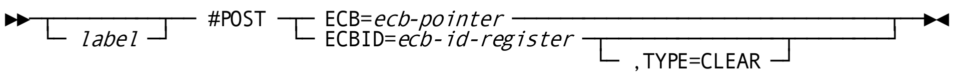
## #POST—modifies an event control block

The #POST statement modifies an event control block (ECB) in one of two ways:

- Posting an ECB to indicate completion of an event for which another task is waiting
- Clearing an ECB to an unposted status

The ECB wait must have been previously established by a #WAIT or #ATTACH request.

### #POST Syntax



### #POST Parameters

**ECB=**

Specifies the ECB to be posted.

***ecb***

Either a register that points to the ECB or the symbolic name of a user-defined fullword field that contains the ECB.

**ECBID=**

Specifies the 4-character ID of the ECB to be posted or to be cleared to an unposted status.

***ecb-id***

A register that contains the ECB ID, the symbolic name of a fullword field that contains the ID, or the ID literal enclosed in quotation marks.

**TYPE=CLEAR**

(Optional); clears the ECB to an unposted status. Programs that are posting and waiting for the posting of ECBs are responsible for clearing the ECB. An ECB must be cleared prior to issuing a subsequent #WAIT request.

### #POST Status Codes

The #POST request is unconditional; any runtime error will result in an abend of the issuing task.

### #POST Example

The following example of the #POST statement clears the event control block identified by the ID literal ECB4 to an unposted status.

```
#POST ECBID='ECB4',TYPE=CLEAR
```

## #PRINT—requests that the system transmit data

The #PRINT statement requests that the system transmit data from a task to a terminal defined as a printer device during system generation. The terminal designated as a printer is usually a hard-copy device. The following considerations apply to the use of the #PRINT statement:

- The DC/UCF system does not usually transmit data directly from program storage to the terminal in response to a #PRINT command. Data is passed to a queue maintained by the system, then from the queue to the printer terminal. The data stream passed to the queue by the #PRINT request contains pure data; the system inserts line and device control characters automatically when it writes the data to the printer.

- To bypass the queuing process described above and to transfer data immediately to a printer device, issue a **print-direct request** by specifying #PRINT OPTNS=(DIRECT).

- You can use a #PRINT request to transmit **native-mode data streams**, data streams that contain device-control information as well as user data. This capability is useful in formatting reports for 3280-type printers. To transmit native-mode data streams, you issue a #MREQ NOIO request, followed by a #PRINT request with OPTNS=(NATIVE).

- Each line of data transmitted by a #PRINT request is considered a **record**. Each record is associated with a **report** in the print queue. A report consists of one or more records. Each task can have up to 256 active print reports. A program can issue multiple #PRINT requests, each specifying a different report. The DC/UCF system maintains the status of each report individually.

- The #PRINT request transmits data or screen contents to print classes or to destinations:

  - **Print classes**—During system generation, one or more print classes are associated with each terminal designated as a printer. For each report, the first record transmitted to the print queue with a #PRINT request establishes the print class in the range of 1 to 64 for that report. The report is printed on the first available printer assigned the same print class.

  - **Destinations**—Destinations are groups of terminals, printers, or users. If destinations have been defined during system generation, the #PRINT request can direct a report to a destination. Reports sent to printer destinations are printed either on the first available printer for the destination or on all printers in that destination, regardless of print class.

- You can request that the system hold the report rather than print it immediately. You can explicitly release the report at a later time.

- The DC/UCF system prints a report only when that report is completed, either explicitly as part of a #PRINT request or implicitly when the issuing task terminates. If the task abends, all reports in the print queue that have not been ended explicitly are deleted without being printed.

- After completion of a #PRINT request, register 1 contains the address of a 10-character identifier that uniquely identifies the report in the DC/UCF system. This identifier is *not* the user-defined report ID described below for the RPTID parameter. It is a value assigned by the system primarily for internal use. This value appears on the master terminal when report statistics are requested from that terminal.

- A report can be printed several times by indicating to the system to keep the report after it has been printed, rather than automatically deleting it. The report can be manually released to be printed using a DCMT VARY REPORT RELEASE command.

## #PRINT Syntax

```
►►─┬─────────┬──────────────────────────────────────────────────────────────────►
   └─ label ─┘

►── #PRINT RECORD=message-location-pointer,RECLEN=message-length-register ──►

►─┬─────────────────────────────────────────┬───────────────────────────────────►
  └─ ,RPTID= ─┬─ 1 ◄ ───────────────────────┤
              └─ report-id-register ────────┘

►─┬──────────────────────────────────────────────┬──────────────────────────────►
  └─ ,CLASS= ─┬─ 1 ◄ ────────────────────────────┤
              └─ printer-class-register ─────────┘

►─┬─────────────────────────────────────────┬───────────────────────────────────►
  │                ┌──── , ────┐            │
  └─ ,OPTNS= ──(─▼─ option ───┴─ ) ─────────┘

►─┬──────────────────┬──────────────────────────────────────────────────────────►
  └─ ,MF= ─┬─ R ─┬───┘
           ├─ L ─┤
           └─ E ─┘

►─┬────────────────────────────────────────────────┬────────────────────────────►
  ├─ ,DEST=printer-destination-pointer ────────────┤
  ├─ ,LTEID=direct-printer-ltermid-pointer ────────┤
  └─ ,LTEADDR=direct-printer-lterm-address ────────┘

►─┬──────────────────────────────────────────────────┬──────────────────────────►
  └─ ,ECBADDR=direct-print-return-ecb-address ───────┘

►─┬─────────────────────────────────────────────┬───────────────────────────────►
  └─ ,JOBNAME=batch-request-jobname-pointer ────┘

►─┬───────────────────────────────────────────┬─────────────────────────────────►
  └─ ,COND= ─┬─── NO ◄ ───────────────────────┤
             ├─── ALL ────────────────────────┤
             │          ┌──── , ────┐         │
             └─(──┬──▼─── NOPR ───┬─┴──)──────┘
                  ├────── IOER ───┤
                  ├────── INVP ───┤
                  ├────── UNDF ───┤
                  ├────── SCRN ───┤
                  ├────── INVT ───┤
                  ├────── WAIT ───┤
                  ├────── OUTS ───┤
                  ├────── DEAD ───┤
                  ├────── CANC ───┤
                  └────── REQU ───┘

►─┬────────────────────────────────────────────────┬────────────────────────────►
  └─ ,PRB= ─┬─ SYSPLIST ◄ ─────────────────────────┤
            └─ print-request-block-pointer ────────┘

►─┬───────────────────────────────────┬─────────────────────────────────────────►
  └─ ,NOPRXIT=no-printer-label ───────┘

►─┬──────────────────────────────────────────────┬──────────────────────────────►
  └─ ,INVPXIT=invalid-parameter-list-label ──────┘

►─┬─────────────────────────────────┬───────────────────────────────────────────►
  └─ ,IOERXIT=i/o-error-label ──────┘

►─┬──────────────────────────────────────────┬──────────────────────────────────►
  └─ ,UNDFXIT=invalid-destid-list-label ─────┘
```

```
►─┬─────────────────────────────────────────┬──────────────────────────►
  └─ ,SCRNXIT=screen-term-i/o-error-label ──┘

►─┬─────────────────────────────────────┬──────────────────────────────►
  └─ ,INVTXIT=invalid-terminal-label ───┘

►─┬───────────────────────────────────────────┬────────────────────────►
  └─ ,WAITXIT=wait-for-direct-printer-label ──┘

►─┬─────────────────────────────────────────────────┬──────────────────►
  └─ ,OUTSXIT=direct-printer-out-of-service-label ──┘

►─┬────────────────────────────────────────────┬───────────────────────►
  └─ ,DEADXIT=deadlock-on-direct-print-label ──┘

►─┬──────────────────────────────────────┬─────────────────────────────►
  └─ ,CANCXIT=cancel-direct-report-label ┘

►─┬───────────────────────────────────────────┬────────────────────────►
  └─ ,REQUXIT=requeue-direct-report-label ────┘

►─┬────────────────────────┬───────────────────────────────────────────►◄
  └─ ,ERROR=error-label ───┘
```

## #PRINT Parameters

**RECORD=**

Specifies the storage area that contains data to be output.

***message-location-pointer***

Either a register that points to the area or the symbolic name of the area.

**RECLEN=**

Specifies the length, in bytes, of the data stream to be output.

***message-length-register***

A register that contains the length, the symbolic name of a user-defined halfword or fullword field that contains the length, or an absolute expression.

**RPTID=1/**

Specifies the identifier of the report to be printed. The report identifier must be an integer in the range 1 through 255; the default is 1.

***report-id-register***

A register that contains the ID, the symbolic name of a user-defined field that contains the ID, or an absolute expression.

**CLASS=1/**

Specifies the class of the printer to which the report is assigned. Valid print classes are 1 through 64; the default is 1.

***printer-class-register***

A register that contains the class, the symbolic name of a user-defined field that contains the class, or an absolute expression. This parameter should be specified only for the first line (record) of each report. If no printer class is specified, the default print class assigned to the issuing task's physical terminal during system generation is used.

**OPTNS=options**

Specifies several options available to print I/O. This parameter is never required and should be specified only when appropriate. The OPTNS parameter values must be enclosed in parentheses. Separate multiple values with commas.

**NATIVE**

Indicates that the data stream contains device control characters. If NATIVE is not specified, the system automatically inserts the necessary characters.

**NEWPAGE**

Requests that the system print the data stream beginning on a new page.

**ENDRPT**

Indicates that the data stream constitutes the last record in the specified report. When ENDRPT is specified, the report can be printed before the issuing task has terminated. To print the report immediately, the program must issue a #COMMIT TASK request. Reports not explicitly ended with an ENDRPT are automatically ended at task termination.

**SCREEN**

(3270-type devices only) transmits the contents of the currently displayed screen to the print queue. When SCREEN is specified, the system implicitly assigns the NATIVE option and ignores the RECORD= and RECLEN= clauses. The terminal operator can print screen contents by pressing the print key established during system generation. If the SCREEN option is specified for a non-3270 terminal or a remote 3270 terminal running under TCAM, an error results.

**ALL**

Causes the report to be printed on all printers associated with the destination specified in the DEST parameter. The report is printed on one printer at a time and saved until it has been printed on all of the printers. You can use a DCMT DISPLAY REPORT DESTINATION command to display the report name followed by a list of the printer names on which the report has yet to be printed.

**HOLD**

Requests that the system hold a report in the print queue before it is printed. The report is not printed until a DCMT VARY REPORT RELEASE command is issued.

**KEEP**

Keeps a report in the print queue after the report has printed. A report marked with the KEEP option can be manually released for printing with the DCMT VARY REPORT RELEASE command. The report can be deleted either manually by issuing a DCMT VARY REPORT DELETE command or automatically through the queue expiration date.

**DIRECT**

Indicates a print-direct request that will be routed directly to the destination specified. Specify the destination by using the CLASS parameter, as described above, or the DEST, LTEID, or LTEADDR parameters, described below. If LTEID or LTEADDR is specified, the system will acquire the specific printer. If CLASS or DEST is specified, the system will acquire the first available printer that satisfies the requested class or destination.

**NOWAIT**

(default) Requests that the DC/UCF system not wait for a printer to become available if the request cannot be immediately serviced; control is returned to the issuing program with a status code indicating that the printer device is unavailable.

**WAIT**

Requests that the system wait for a printer to become available if the request cannot be immediately serviced. If COND=OUTS or COND=ALL has been specified, the total wait time will be the product of the task's stall interval to a maximum of 60 seconds and the MAXIMUM ERRORS parameter of the PTE. Otherwise, the maximum wait time equals the stall interval.

**MF=**

Specifies the type of #PRINT request.

**R**

Identifies a **regular** #PRINT request. The DC/UCF system builds a new print request block (PRB) for each request and performs the requested operation.

**L**

Identifies a **list** #PRINT request. The DC/UCF system adds a predefined PRB in the data definition section of program storage. The PRB contains parameters whose values remain constant throughout the program. The #PRINT label used to identify the PRB is referenced by the PRB parameter in subsequent execute-type requests. Only the label and the MF parameter are required for list-type #PRINT requests; other parameters should be specified only when required to predefine PRB parameter values.

**E**

Identifies an **execute** #PRINT request. The DC/UCF system adds to or overrides the predefined PRB with the parameters defined in the request and performs the requested operation.

**DEST/LTEID/LTEADDR**

Identifies the printers to which a report is routed. These parameters can only be specified with OPTNS=DIRECT; you specify the destination.

**DEST=**

Specifies a destination defined during system generation. The destination can be one of the following:

- A **list of logical terminals** requesting that the system route the report to all available terminals in the list
- A **list of users** requesting that the system route the report to all listed users who are currently signed on to the system

***printer-destination-pointer***

A register that points to the destination ID, the symbolic name of a user-defined field that contains the destination ID, or the ID itself enclosed in quotation marks.

**LTEID=**

Specifies the logical terminal ID of a specific printer-terminal device.

***direct-printer-ltermid-pointer***

A register that points to the logical terminal ID, the symbolic name of a user-defined field that contains the logical terminal ID, or the ID itself enclosed in quotation marks.

**LTEADDR=**

Specifies the logical terminal element (LTE) address of a specific printer-terminal device.

***direct-printer-lterm-address***

A register that points to the address of the LTE, the symbolic name of a user-defined field that contains the address of the LTE, or the address itself enclosed in quotation marks.

**ECBADDR=**

Specifies the location to which the system returns the address of a list of event control blocks (ECBs) if the print-direct request cannot be serviced immediately. If OPTNS=(DIRECT,NOWAIT) has been specified and the system cannot immediately acquire the requested printer device, the system returns the address of a list of ECBs to the requesting task. One ECB from the list is posted when the requested printer becomes available. At that time, the print-direct request can be reissued.

Note: If you use the ECBADDR= parameter and specify OPTNS=(DIRECT,NOWAIT), the system will allocate storage for the ECBLIST. The program is responsible for freeing the storage space.

***direct-print-return-ecb-address***

Either a register that points to the ECB area or the symbolic name of a user-defined field that contains the address of the area.

**JOBNAME=**

Specifies the name of the system report to be associated with a print request from a batch program. The JOBNAME parameter is for informational use

only.

***batch-request-jobname-pointer***

A 1- to 8-character job name that is displayed as the original logical terminal ID when a DCMT DISPLAY REPORTS command is issued. *Batch-request-jobname* is a register that points to the job name, the symbolic name of a user-defined field that contains the job name, or the name itself enclosed in quotation marks.

**COND=**

Specifies the conditions under which control is to be returned to the issuing program.

**NO**

(Default); specifies that the request is not conditional. Control is not returned to your program under any circumstances.

**ALL**

Specifies that the request is conditional. Control is returned to your program if the #PRINT request cannot be serviced for one or more of the reasons listed below.

***condition***

Specifies under which conditions control is returned to your program. Multiple conditions must be enclosed in parentheses and separated by commas. *Conditions* can specify one or more of the following conditions:

- **NOPR**—No printer logical terminals were defined during system generation.
- **IOER**—An I/O error occurred during processing.
- **INVP**—There is an invalid parameter in the PRB.
- **UNDF**—An undefined destination is specified or, for a print-direct request, an invalid LTEID or LTEADDR is specified.
- **SCRN**—A print-screen type request results in a terminal I/O error.
- **INVT**—A print-screen request has been made from a non-3270-type terminal or from a 3270-type terminal without read-buffer support.

- **WAIT**—No printer can be found to service a print-direct request that specifies OPTNS=(DIRECT,NOWAIT).
- **OUTS**—The printer specified by the LTEID or LTEADDR parameters in a print-direct request is out of service.
- **DEAD**—A print-direct request has been issued with OPTNS=(DIRECT,WAIT) and a deadlock condition would otherwise occur.
- **CANC**—A DCMT VARY PRINTER CANCEL command has been issued for the printer in a print-direct request.
- **REQU**—A DCMT VARY PRINTER REQUEUE command has been issued for the printer specified in a print-direct request.

**PRB=**

Specifies the location of the storage area in which the system will build the PRB (MF=R) or has built the PRB (MF=E).

**SYSPLIST**

(Default for regular-type requests only); is the symbolic name of the storage area in which the system builds the PRB.

***print-request-block-pointer***

A register that points to the area or the symbolic name of the area in which the system will build the PRB. For execute-type requests (MF=E), this entry explicitly defines the area by identifying *label*, provided in a previously-issued list-type #PRINT that established the PRB.

**NOPRXIT=*no-printer-label***

Specifies the symbolic name of the routine to which control should be returned if the #PRINT request cannot be serviced because no printer terminal was defined during system generation.

**INVPXIT=*invalid-parameter-list-label***

Specifies the symbolic name of a routine to which control should be returned if the #PRINT request cannot be serviced because of an invalid parameter in the PRB.

**IOERXIT=*i/o-error-label***

Specifies the symbolic name of a routine to which control should be returned if the #PRINT request cannot be serviced because of an I/O error during processing.

**UNDFXIT=*invalid-dest-id-label***

Specifies the symbolic name of a routine to which control should be returned if the #PRINT request cannot be serviced because an invalid destination was specified or, for OPTNS=(DIRECT) type requests, an invalid LTEID or LTEADDR was specified.

**SCRNXIT=*screen-term-i/o-error-label***

Specifies the symbolic name of a routine to which control should be returned if the #PRINT request cannot be serviced because a terminal I/O error occurred in response to a #PRINT request to print the screen contents.

**INVTXIT=*invalid-terminal-label***

Specifies the symbolic name of a routine to which control should be returned if the screen #PRINT request cannot be serviced because an invalid terminal was specified.

**WAITXIT=*wait-for-direct-printer-label***

Specifies the symbolic name of a routine to which control should be returned if the #PRINT request cannot be serviced because OPTNS=(DIRECT,NOWAIT) was requested and no printer is available to service the request immediately.

**OUTSXIT=*direct-printer-out-of-service-label***

Specifies the symbolic name of a routine to which control should be returned if the #PRINT request cannot be serviced because the printer identified by LTEID or LTEADDR in a print-direct request is out of service.

**DEADXIT=*deadlock-on-direct-print-label***

Specifies the symbolic name of a routine to which control should be returned if the #PRINT request cannot be serviced because OPTNS=(DIRECT,WAIT) was specified and would otherwise cause a deadlock condition to occur.

**CANCXIT=*cancel-direct-report-label***

Specifies the symbolic name of a routine to which control should be returned if the #PRINT request cannot be serviced because a DCMT VARY PRINTER CANCEL has been issued for the specified printer while the print request is being serviced.

**REQUXIT=*requeue-direct-report-label***

Specifies the symbolic name of a routine to which control should be returned if the #PRINT request cannot be serviced because a DCMT VARY PRINTER REQUEUE has been issued for the specified printer while the print request is being serviced.

**ERROR=*error-label***

Specifies the symbolic name of a routine to which control should be returned if a condition in the COND parameter occurs for which no other exit routine was coded.

## #PRINT Status Codes

After completion of a #PRINT request, the value in register 15 indicates the outcome of the operation. The following is a list of the Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'04'**

The request cannot be serviced because an I/O error occurred during a #PUTQUE request or, for OPTNS=(DIRECT), a permanent I/O occurred on the direct printer.

**X'08'**

The request cannot be serviced because the parameter list passed to #PRINT contains an invalid field.

**X'0C'**

The request cannot be serviced because no printer logical terminals have been defined for the current system.

**X'10'**

The request cannot be serviced because a print screen request has been made from a non-3270-type terminal or from a 3270-type terminal without read-buffer support.

**X'14'**

The request cannot be serviced because the specified printer destination is invalid or, for OPTNS=(DIRECT), the LTEID or LTEADDR specification is invalid.

**X'18'**

The request cannot be serviced because a terminal I/O error occurred during a print-screen type #PRINT request.

**X'1C'**

The request cannot be serviced because no printer could be found to satisfy the print-direct request, and OPTNS=(NOWAIT) was specified.

**X'20'**

The request cannot be serviced because the print-direct request has specified an LTEID or LTEADDR that is out of service.

**X'24'**

The request cannot be serviced because the print-direct request specified a wait, and to wait would cause a deadlock.

**X'28'**

The request cannot be serviced because a DCMT VARY PRINTER CANCEL command has been issued in the DC/UCF system for this direct printer.

**X'2C'**

The request cannot be serviced because a DCMT VARY PRINTER REQUEUE command has been issued in the DC/UCF system for this direct printer.

## #PRINT Example

The #PRINT statement shown below performs the following functions:

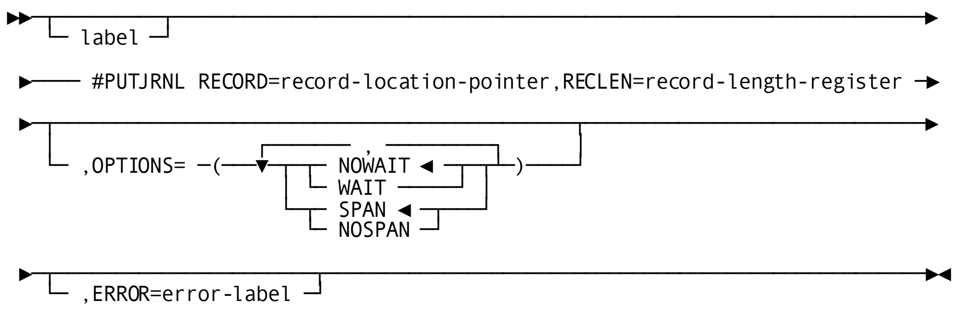
- Directs the system to transmit the data in storage area RECOUT to a terminal defined as a printer device.
- Specifies that the length of data transmitted is contained in the field OUTLEN.
- Directs the print request to a specific printer, bypassing the queuing process.
- Asks the system to wait until the named printer is able to service the request. If the wait time exceeds the stall interval defined at system generation, the program will abort.
- Names the printer by logical terminal ID.

```
#PRINT RECORD=RECOUT,RECLEN=OUTLEN,OPTNS=DIRECT,WAIT,LTEID='LV009'
```

## #PUTJRNL—writes a task-defined record to the journal file

The #PUTJRNL statement writes a task-defined record to the journal file. The records written to the journal file are available to user-defined exit routines during a task-initiated or system-initiated rollback.

## #PUTJRNL Syntax


label
#PUTJRNL RECORD=record-location-pointer,RECLEN=record-length-register
,OPTIONS= (
,
NOWAIT
WAIT
SPAN
NOSPAN
)
,ERROR=error-label

## #PUTJRNL Parameters

**RECORD=**

Specifies the location of the record to be written to the journal file.

***record-location-pointer***

Either a register that points to the record area or the symbolic name of the record area.

**RECLEN=**

Specifies the length, in bytes, of the record to be written to the journal file.

***record-length-register***

Either a register that contains the length of the record or the symbolic name of a fullword user-defined field that contains the length of the record.

**OPTIONS=**

Specifies whether the issuing task is to wait for completion of the #PUTJRNL function before resuming task execution and indicates how the system writes the named record to the journal file. Multiple options are enclosed in parentheses and separated by commas.

The following options determine whether the issuing task will wait for completion of the #PUTJRNL function.

**NOWAIT**

(Default); specifies that the issuing task will not wait for completion of the #PUTJRNL function; the journal record remains in a storage buffer until a future request necessitates writing the buffer to the journal file.

**WAIT**

Specifies that the issuing task will wait for completion of the #PUTJRNL operation before continuing. This option Requests that the system write a partially filled buffer to the journal file.

When a record is shorter than a journal file block, based on space available in the current journal block, the system either places the record in the block, splits it across multiple blocks (SPAN), or writes it to a new block after the current block is filled (NOSPAN). The following options determine how the system writes the named record to the journal file.

**SPAN**

(Default); specifies that the DC/UCF system will write the record across several journal blocks, if necessary. In general, the SPAN option provides better space utilization in the journal file because it increases the average fullness of each block.

**NOSPAN**

Specifies that the system will write the record into a single journal block, assuming that the record fits. If the record is longer than the journal block, it will be split.

The following considerations apply to using an exit routine to retrieve journal file records during recovery:

- If a #PUTJRNL statement issued before a failure specifies the SPAN option, records may have been written across several journal blocks. To retrieve these records, the program must invoke the exit routine once for each segment of each record to be retrieved.
- If a #PUTJRNL statement issued before a failure specified the NOSPAN option, and records written to the journal file are shorter than journal blocks, the exit routine need only be concerned with the complete records.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control is to be returned in the event of an error condition during the #PUTJRNL operation.

## #PUTJRNL Status Codes

After completion of the #PUTJRNL request, the value in register 15 indicates the outcome of the operation:

**X'00'**

The request has been serviced successfully.

**X'04'**

The request cannot be serviced because the journal record length is zero or negative

**X'08'**

The request cannot be serviced because the required storage is not available for necessary control blocks.

**X'0C'**

The request cannot be serviced because an invalid error status has been received from DBIO/DBMS. Check the DC/UCF log for details.

## #PUTJRNL Example

The following example of the #PUTJRNL statement writes a record to the journal file. The address of the record is contained in register 5, the length of the record is contained in register 7. The default SPAN and NOWAIT options are in effect.

```
#PUTJRNL RECORD=(R5),RECLEN=(R7)
```

## #PUTQUE—stores a queue record in the queue

The #PUTQUE statement stores a queue record in the queue (DDLDCRUN or DDLDCQUE) area of the dictionary, causing the system to place the record in the queue-header/queue-record set referenced by the QUEID parameter. A program does not assign an ID to a queue record; the #PUTQUE request stores the record at the beginning or end of the queue and the system automatically assigns the queue record ID.

### #PUTQUE Syntax

```
►►─┬───────┬──────────────────────────────────────────────────────────────►
   └ label ┘

►── #PUTQUE RECORD=queue-data-location,RECLEN=queue-data-length-register ──►

►─┬───────────────────────────────────────┬───────────────────────────────►
  └ ,PLIST= ─┬─ SYSPLIST ◄ ──────────────┬┘
             └─ parameter-list-pointer ──┘

►─┬───────────────────────────┬───────────────────────────────────────────►
  └ ,QUEID=queue-id-pointer ──┘

►─┬──────────────────────┬────────────────────────────────────────────────►
  └ ,LOC= ─┬─ LAST ◄ ─┬──┘
           └─ FIRST ──┘

►─┬────────────────────────────────────────────────────────┬──────────────►
  └ ,RTNQRID= ─┬─ (1) ◄ ──────────────────────────────────┬┘
               └─ return-queue-record-id-register ────────┘

►─┬──────────────────────┬────────────────────────────────────────────────►
  └ ,COND= ─┬─ NO ◄ ──┬──┘
            └─ IOER ──┘

►─┬──────────────────────────────┬────────────────────────────────────────►
  └ ,IOERXIT=i/o-error-label ────┘

►─┬───────────────────────┬───────────────────────────────────────────────►
  └ ,ERROR=error-label ───┘

►─┬──────────────────────────────────────┬────────────────────────────────►◄
  └ ,RETAIN=retention-period-register ───┘
```

### #PUTQUE Parameters

**RECORD=**

Specifies the location of the user area that contains data to be stored in the queue record.

***queue-data-location***

A register that points to the area or the user-defined symbolic name of the area.

**RECLEN=**

Specifies the length of the data area to be stored in the queue record.

***queue-data-length-register***

A register that contains the length, the symbolic name of a fullword user-defined field that contains the length, or an absolute expression.

**PLIST=SYSPLIST**

Specifies the location of the seven-fullword storage area in which the system builds the #PUTQUE parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system builds the #PUTQUE parameter list.

***parameter-list-pointer***

Either a register that points to the area or the symbolic name of the area.

**QUEID=**

Specifies the 1- to 16-character ID of the queue with which the record being stored is associated.

***queue-id-pointer***

A register that points to a field that contains the ID, the symbolic name of a user-defined field that contains the ID, or the ID literal enclosed in quotation marks. If a queue ID is not specified, 16 blanks are assumed.

**LOC=LAST/FIRST**

Specifies whether the queue record is to be placed at the beginning or end of the queue.

**LAST**

(Default); stores the record at the end of the queue.

**FIRST**

Stores the record at the beginning of the queue.

**RTNQRID=**

Specifies the location in the program to which the system returns the system-assigned ID of the stored queue record; the returned ID can be saved and used to retrieve or delete the queue record.

**(1)**

(Default); is the register to which the system returns the queue record ID.

***return-queue-record-id-register***

Either a register or the symbolic name of a fullword user-defined field to which the system returns the queue record ID.

**COND=**

Specifies whether this #PUTQUE is conditional and under what conditions control should be returned to the issuing program.

**NO**

(Default); specifies that the request is not conditional.

**IOER**

Specifies that the request is conditional. Control is returned if an I/O error occurs while processing the request.

**IOERXIT=*i/o-error-label***

Specifies the symbolic name of the routine to which control should be returned if the #PUTQUE cannot be serviced because of an I/O error.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control should be returned if a condition in the COND parameter occurs for which no other exit routine was coded. In this case, the ERROR parameter functions identically to IOERXIT.

**RETAIN=**

Specifies the amount of time, in days, that the system will retain the queue in the dictionary. At system startup, queues whose retention periods have expired are deleted automatically by the system. The retention period begins when the first record is stored in the queue.

If RETAIN is omitted, the default retention period for dynamic queues is taken.

**Note:** For more information on the default retention period for dynamic queues, see the *CA IDMS System Generation Guide*.

***retention-period-register***

A register that points to a field that contains the retention period, the symbolic name of a user-defined fixed-binary field that contains the retention period, or an absolute expression. The retention period must be a numeric constant in the range 0 through 255. A retention period of 255 indicates that the queue is never to be deleted automatically by the system.

## #PUTQUE Status Codes

By default, the #PUTQUE request is unconditional; a runtime I/O error results in an abend of the issuing task. The issuing program can request return of control with the COND parameter to avoid an abend.

After completion of a #PUTQUE request, the value in register 15 indicates the outcome of the operation:

**X'00'**

The request has been serviced successfully.

**X'04'**

The request cannot be serviced; check for proper queue-id specification (for example, a negative queue ID is an improper specification) and for logical selection of options.

**X'1C'**

A database error occurred during queue processing. A common cause is a DBKEY deadlock. For a PUT QUEUE operation, this code can also mean that the queue upper limit has been reached.

If a database error has occurred, there are usually be other messages in the CA-IDMS/DC/UCF log indicating a problem encountered in RHDCRUAL, the internal Run Unit Manager. If a deadlock has occurred, messages DC001000 and DC001002 are also produced.

If an I/O error occurs while processing a #PUTQUE request, the system returns the address of the communications block to register 1. If no error occurs during processing, a user-defined register, assigned by the RTNQRID parameter, contains the queue record ID of the stored queue record.

## #PUTQUE Example

The following example Requests that the system store the data contained in the field RECQ1 in the beginning of the RES-Q queue. The length of the data is contained in register 8. The DC/UCF system is requested to return the ID of the record to the QRECID field and to retain the queue for 45 days.

```
#PUTQUE RECORD=RECQ1,RECLEN=(R8),QUEID='RES-Q',LOC=FIRST,                *
      RTNQRID=QRECID,RETAIN=45
```

## #PUTSCR—stores or replaces a scratch record

The #PUTSCR statement stores or replaces a scratch record in the scratch area of the dictionary. For new records, #PUTSCR generates an index entry in a scratch area associated with the issuing task. If the scratch area does not already exist, the system allocates it dynamically in the storage pool.

After completion of the #PUTSCR function, control is returned to the issuing program at the next sequential instruction following the #PUTSCR request. Through the REPXIT, NEWXIT, and EREPXIT parameters, you can request return of control to a specified label after a successful replace or store, or after confirmation that the new record already exists for the task.

## #PUTSCR Syntax

```
►►─┬───────┬──────────────────────────────────────────────────────────────────►
   └ label ┘

►─ #PUTSCR RECORD=scratch-data-location,RECLEN=scratch-data-length-register ─►

►─┬──────────────────────────────────────────┬────────────────────────────────►
  └ ,PLIST= ─┬─ SYSPLIST ◄ ──────────────┬───┘
             └─ parameter-list-pointer ──┘

►─┬─────────────────────────────────┬─────────────────────────────────────────►
  └ ,SAID=scratch-area-id-pointer ──┘

►─┬───────────────────────────────────┬───────────────────────────────────────►
  └ ,SRID=scratch-record-id-pointer ──┘

►─┬──────────────────────────┬────────────────────────────────────────────────►
  └ ,REPLACE= ─┬─ NO ◄ ──┬───┘
               └─ YES ───┘

►─┬─────────────────────────────────────────────────────────┬─────────────────►
  └ ,RTNSRID= ─┬─ (1) ◄ ─────────────────────────────────┬──┘
               └─ return-scratch-record-id-register ─────┘

►─┬───────────────────────┬───────────────────────────────────────────────────►
  └ ,COND= ─┬─ NO ◄ ──┬───┘
            └─ IOER ──┘

►─┬──────────────────────────────┬────────────────────────────────────────────►
  └ ,IOERXIT=i/o-error-label ────┘

►─┬──────────────────────────┬────────────────────────────────────────────────►
  └ ERROR=error-label ───────┘

►─┬────────────────────────────────────────┬──────────────────────────────────►
  └ ,REPXIT=successful-replace-label ──────┘

►─┬────────────────────────────────────┬──────────────────────────────────────►
  └ ,NEWXIT=successful-store-label ────┘

►─┬──────────────────────────────────────────┬───────────────────────────────►◄
  └ ,EREPXIT=record-already-exists-label ────┘
```

## #PUTSCR Parameters

**RECORD=**

Specifies the location of the user area that contains the data area to be stored in the scratch record.

***scratch-data-location***

Either a register that points to the area or the user-defined symbolic name of the area.

**RECLEN=**

Specifies the length of the record to be stored.

***scratch-data-length-register***

A register that contains the length, the symbolic name of a fullword user-defined field that contains the length, or an absolute expression.

When replacing a scratch record, the RECLEN specified need not agree with that of the old record, because the replace is effected with a delete and an add. If a replace of a nonexistent record is requested, the system performs the request with an add, and an error status value of 0 is returned into register 15.

**PLIST=**

Specifies the location of the seven-fullword storage area in which the system builds the #PUTSCR parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system will build the #PUTSCR parameter list.

***parameter-list-pointer***

Either a register that points to the area in which the system will build the #PUTSCR parameter list or the symbolic name of that area.

**SAID=**

Specifies the 1- to 8-character ID of the scratch area associated with the record being allocated.

***scratch-area-id***

Either a register that points to a field that contains the ID, the symbolic name of a user-defined field that contains the ID, or the ID literal enclosed in quotation marks. If the SAID parameter is not specified, 8 blanks are assumed.

**SRID=**

Specifies the fullword ID of the scratch record being stored.

***scratch-record-id-pointer***

A register that points to the ID, the symbolic name of a user-defined field that contains the ID, or an absolute expression.

An SRID must be specified for all replace-type #PUTSCR requests or an I/O error will result. If not specified for add-type requests, the SRID is assigned automatically by the system and is returned in the register defined in the RTNSRID parameter.

**REPLACE=**

Indicates whether the scratch record is added or replaced.

**NO**

(Default); directs the system to add a new record to a scratch area.

**YES**

Directs the system to replace an existing record in the scratch area.

**RTNSRID=**

Specifies the location to which the system will return the automatically assigned scratch record ID of the stored record.

**(1)**

(Default); is the register into which the system will place the scratch record ID.

***return-scratch-record-id-register***

A register or the symbolic name of a fullword user-defined field into which the system will place the scratch record ID.

**COND=**

Specifies whether this #PUTSCR is conditional and under what conditions control should be returned to the issuing program.

**NO**

(Default); specifies that the request is not conditional.

**IOER**

Specifies that control is returned to the issuing program if an I/O error occurs while processing the request.

**IOERXIT=*i/o-error-label***

Specifies the symbolic name of the routine to which control should be returned if the #PUTSCR cannot be serviced because of an I/O error.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded. In this case, the ERROR and IOERXIT parameters function identically.

**REPXIT=*successful-replace-label***

(REPLACE=YES only); specifies the symbolic name of the routine to which control should be returned when the request is serviced successfully. If no REPXIT is defined in a successful replace-type #PUTSCR request, control will be returned to the next sequential instruction following the #PUTSCR.

**NEWXIT=*successful-store-label***

(Add requests only); specifies the symbolic name of the routine to which control should be returned when the request is successful. If no NEWXIT is defined in a successful add-type request, control will be returned to the next sequential instruction following the #PUTSCR.

**EREPXIT=*record-already-exists-label***

(Add requests only) specifies the symbolic name of the routine to which control should be returned when the scratch record ID specified by the SRID parameter already exists in the scratch area identified by the SAID parameter. If no EREPXIT is defined for an add-type request and the requested SRID already exists, control is returned to the next sequential instruction following the #PUTSCR.

## #PUTSCR Status Codes

By default, the #PUTSCR request is unconditional; a runtime I/O error will result in an abend of the issuing task. The issuing program can request return of control with the COND parameter to avoid an abend.

After completion of the #PUTSCR function, the value in register 15 indicates the outcome of the operation. The following is a list of the Register 15 values and the corresponding meaning:

**X'00'**

The request to add a new record has been serviced successfully.

**X'04'**

The request cannot be serviced; check for proper scratch-id specification (for example, a negative scratch ID is an improper specification) and for logical selection of options.

**X'10'**

The request to replace a scratch record has been serviced successfully.

**X'14'**

The request to add a new scratch record cannot be serviced because the scratch record ID specified by the SRID parameter already exists for the named scratch area and REPLACE=YES has not been specified.

**X'1C'**

The request cannot be serviced due to an I/O error during processing.

If an I/O error occurs while processing a #PUTSCR request, the system returns the address of the communications block to register 1. If no error occurs during processing, a user-defined register, assigned by the RTNSRID parameter, contains the SRID of the stored or replaced record.

### #PUTSCR Example

The following example of the #PUTSCR statement stores a scratch record containing the data in SCR605 in the dictionary. The length of the record is contained in the fullword field SCRLN1. SCRID1 is the ID of the scratch area into which the record will be stored.

```
#PUTSCR RECORD=SCR605,RECLEN=SCRLN1,SAID='SCRID1'
```

## @READY—prepares a database area for access by DML functions

The @READY statement prepares a database area for access by DML functions and specifies the usage mode of the area. @READY also defines and logs the initial checkpoint for a recovery unit to facilitate recovery operations.

The DBA can specify default usage modes in the subschema. A run-unit using a subschema with specified default usage modes need not issue any @READY statements; the areas are readied automatically in the predefined usage modes. However, if a run-unit issues an @READY statement for one area, it must issue @READY statements for all areas that it accesses unless the FORCE option was specified for the default usage mode. Areas using the default usage mode combined with the FORCE option are automatically readied even if the run-unit already issued @READY for other areas.

The usage mode specified in the @READY statement (or in the subschema) indicates the runtime operations that the readying run unit can or cannot perform against the database area. The following usage modes can be specified:

- **UPDATE=YES** indicates that the readying run unit is permitted to issue all DML functions for records in that area.
- **RDONLY=YES** indicates that the readying run unit is prohibited from issuing the STORE, ERASE, MODIFY, CONNECT, or DISCON functions for records in that area.

The specified usage mode can be qualified with a **PROTECTED** or **EXCLUSIVE** option to prevent update or use, respectively, of areas by other run units executing concurrently under the CA IDMS/DB central version. Each area can be readied in its own usage mode. Usage modes can be changed during a recovery unit by executing an @FINISH statement and readying the areas in a different usage mode. Note, however, that the appropriate BIND statements must also be issued.

When the run unit (rather than the subschema) readies database areas, all areas can be readied with a single @READY statement or each area to be accessed can be readied individually. You must ready all areas explicitly or implicitly affected by the DML statements issued by the run unit. Areas are affected implicitly, for example, when a set's owner and member records belong to different areas. Some areas included in the subschema may not need to be specified in an @READY statement, as only those areas that are explicitly or implicitly affected need to be readied.

The @READY statement can appear anywhere in an application program; however, to avoid runtime deadlock, the best practice is to ready all areas before issuing any other DML statements.

## @READY Syntax

```
►►── @READY ─┬─ ALL ◄ ───────────┬─ , ─┬─ UPDATE= ─┬─┬─ YES(SHARED) ──────────►◄
             └─ AREA=area-name ──┘     └─ RDONLY= ─┘ ├─ PROTECTED ───┤
                                                     └─ EXCLUSIVE ───┘
```

## @READY Parameters

**ALL/AREA=**

Opens the database areas.

**ALL**

(Default); opens all database areas in the subschema.

**AREA=*area-name***

Opens only the specified area. *Area-name* must be an area included in the subschema.

**UPDATE/RDONLY=YES/PROTECTED/EXCLUSIVE**

Specifies how the database areas are opened and qualify database area usage.

**UPDATE/RDONLY**

Specifies how the database areas are opened.

**UPDATE**

Specifies that the database areas are opened in both update and retrieval modes.

**RDONLY**

Specifies that the database areas are opened in retrieval mode only.

**YES/PROTECTED/EXCLUSIVE**

Qualifies database area usage.

**YES**

Allows other concurrently executing run units to open the same area in shared retrieval or shared update usage modes. Keywords YES and SHARED are synonymous.

**PROTECTED**

Prevents concurrent update of the areas by run units executing under the same central version. Once a run unit has readied an area with the protected option, no other run unit can ready that area in any update usage mode until the first run unit releases it by means of a FINISH statement. A run unit cannot ready an area with the protected option if another run unit has readied the area in update usage mode.

**EXCLUSIVE**

Prevents concurrent use of the areas by any other run unit executing under the central version. Once a run unit has readied an area with the exclusive option, no other run unit can ready that area in any usage mode until the first run unit releases it.

If, under the central version, an @READY statement would result in a mode usage conflict for an area, the run unit issuing the @READY is placed in a wait state on the first functional database call.

Modification statements involving areas opened in one of the update usage modes are not allowed if they affect sets that include records in an area opened in one of the retrieval usage modes.

## @READY Status Codes

After completion of the @READY function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0910**

The subschema specifies an access restriction that prohibits readying the area in the specified usage mode.

**0923**

The named area is not in the subschema.

**0928**

The run unit has attempted to ready an area that has been readied previously.

**0966**

The area specified is not available in the requested usage mode. Probable causes for the return of this status code are:

- If running in local mode, the area is locked against update.
- If running under the central version, either the area is offline to the central version, or an update usage mode was requested and the area is in retrieval mode to the central version.

**0970**

The database will not ready properly; a JCL error is the probable cause.

**0971**

The page group or page range for the area being readied could not be found in the DMCL.

**0978**

A wait for an area would cause a deadlock. Either you should ready all areas before the first functional call or all user programs should ready areas in the same order.

## @READY Example

The following example of the @READY statement prepares all database areas in the subschema for retrieval usage mode only (read only). YES is equivalent to SHARED usage mode, allowing other concurrently executing run units to open the same area in shared retrieval usage mode.

```
@READY ALL,RDONLY=YES
```

## @RETURN

The @RETURN statement retrieves the database key for an indexed record without retrieving the record itself, thus establishing currency in the index set. The record's symbolic key is moved into the data fields within the record in program variable storage. The contents of all non-key fields after the execution of the @RETURN verb are unpredictable. Alternatively, you can have the record's symbolic key moved into some other specified variable storage location.

Index currency is established by:

- Successful execution of the @RETURN statement, which sets current of index at the index entry from which the database key was retrieved.
- A status code 1707 (end of index), which sets currency on the index owner. The DBMS returns the owner's db-key.
- A status code 1726 (end of set), which sets current of index as follows:
  - Between the two entries that are higher and lower than the specified value
  - After the highest entry, if the specified value is higher than all index entries
  - Before the lowest entry, if the specified value is lower than all index entries

The @RETURN statement is used in both navigational and LRF environments.

## @RETURN Syntax

Navigational @RETURN

```
►►─── @RETURN ─┬─ CURRENT ─┬─ ,SET=index-set-name, DBKEY=db-key ──────────►◄
               ├─ FIRST ───┤
               ├─ LAST ────┤
               ├─ NEXT ────┤
               └─ PRIOR ───┘
```

LRF @RETURN

```
►►─── @RETURN SET=index-set-name,DBKEY=db-key,USING=index-key-value ──────►◄
```

## @RETURN Parameters

**CURRENT/FIRST/LAST/NEXT/PRIOR**

Indicates the record whose database key will be returned.

**CURRENT**

Retrieves the database key for the current index entry.

**FIRST**

Retrieves the database key for the first index entry.

**LAST**

Retrieves the database key for the last index entry.

**NEXT**

Retrieves the database key for the index entry following current of index. If the current of index is the last entry, an error status of 1707 (end of index) is returned.

**PRIOR**

Retrieves the database key for the index entry preceding current of index. If the current of index is the first entry, an error status of 1707 (end of index) is returned.

**SET=**

Identifies the indexed set from which the specified database key is to be returned.

***index-set-name***

Either a register containing the name of the indexed set or a quoted variable containing the name of the set.

**DBKEY=**

Where the database key is returned.

***db-key***

A register containing the database key or a user defined variable data field.

**USING=**

Saves the symbolic key (CALC, sort, or index) or the specified record.

***index-key-value***

A register containing the index key value or the name of the user-defined alphanumeric field into which the symbolic key of the specified record will be returned. *Index-key-value* must be large enough to accommodate the symbolic key. For example, if the set is indexed on employee last name (15 characters) and employee first name (10 characters) the *index-key-value* must be large enough to accommodate 25 characters.

## @RETURN Status Codes

After the @RETURN statement has been processed, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**1707**

Either the end of the indexed set has been reached or the indexed set is empty.

**1725**

Currency has not been established for the specified indexed set.

**1726**

The index entry cannot be found.

## @RETURN Example

The @RETURN statement shown below retrieves the database key for the first index entry in the EMPLNAMX set and moves the record's db-key into the LNAMXKEY field:

```
@RETURN FIRST,SET=EMPLNAMX,DBKEY=LNAMXKEY
```

## #RETURN—returns control to a program

The #RETURN statement performs the following functions:

- Returns control to a program at the next higher level in a task, optionally specifying the next task to be initiated on the same terminal.
- In abend routines established by #STAE functions, #RETURN specifies the recovery procedure to be initiated by the abend exit if the task terminates abnormally.

  Note: For more information about #STAE exits, see #STAE (see page 331) later in this chapter.
- Specifies the action the system takes when the terminal operator does not enter the response required to initiate the specified task.

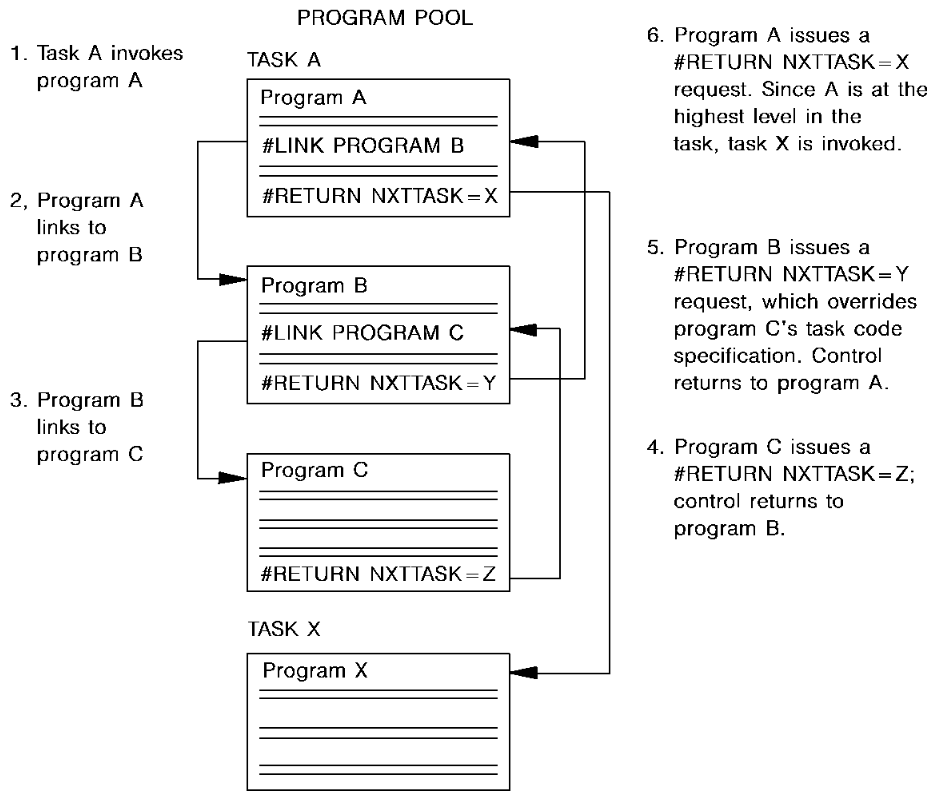
Following a #RETURN request, control returns to the program at the next higher level in the task. If the issuing program is the highest level program, control returns to the system. Any #RETURN statement can include a NXTTASK option to specify the next task to be initiated by the system. However, the position of the issuing program in the task governs whether the specified task will, in fact, receive control.

When the system receives control from the highest level program that issued a #RETURN NXTTASK request, the specified task is executed immediately if the specified task code has been assigned the NOINPUT attribute during system generation. If the task code has been assigned the INPUT attribute, the task executes only when the terminal operator enters the requested data.

You can define tasks that relinquish control to the system while awaiting completion of an event. This way, resources for the issuing task are freed during the time it takes for a particular event to finish and the next task to start.

The DC/UCF system gives control to the next task when a specific event control block (ECB) is posted, indicating that the event is completed.

When initiated, the next task is associated with the same logical terminal (LTERM) as the task that issued the #RETURN. An example of the flow of control between tasks is illustrated in the following figure.


PROGRAM POOL
1. Task A invokes program A
TASK A
Program A
#LINK PROGRAM B
#RETURN NXTTASK=X
2, Program A links to program B
Program B
#LINK PROGRAM C
#RETURN NXTTASK=Y
3. Program B links to program C
Program C
#RETURN NXTTASK=Z
TASK X
Program X
6. Program A issues a #RETURN NXTTASK=X request. Since A is at the highest level in the task, task X is invoked.
5. Program B issues a #RETURN NXTTASK=Y request, which overrides program C's task code specification. Control returns to program A.
4. Program C issues a #RETURN NXTTASK=Z; control returns to program B.

## @ROLLBAK—rolls back uncommitted changes made to the database

The @ROLLBAK statement rolls back uncommitted changes made to the database through an individual run unit.

Whether the changes are automatically backed out depends on the execution environment:

- If the changes were made under the control of a central version that is journaling to a disk file, they are backed out automatically. The central version continues to process other applications during recovery.

- The changes are not backed out automatically under the following circumstances:
  - If the changes were made under the control of a central version that is journaling to a tape file.
  - If the changes were made in local mode.

  In these cases, the @ROLLBAK statement causes the affected areas to remain locked against subsequent access by other database sessions. They must be manually recovered. If changes cannot be backed out and CONTINUE was specified on the rollback request, a non-zero error status is returned to the application and the run unit is terminated.

  Note: For more information about manual recovery, see the *Database Administration Guide*.

If CONTINUE is not specified, the run unit ends and its access to the database is terminated. If CONTINUE is specified, the run unit remains active after the operation is complete.

**Currency**

Following an @ROLLBAK statement, all currencies are set to null. Unless the CONTINUE parameter is specified, the issuing program cannot perform database access through the run unit without executing another @BIND/@READY sequence.

## @ROLLBACK Syntax

```
►►── @ROLLBAK ─┬────────────┬──────────────────────────────────────────►◄
               └─ CONTINUE ─┘
```

## @ROLLBACK Parameters

**CONTINUE**

Central version only. Causes the run unit to remain active after its changes are backed out. Database access can be resumed without reissuing @BIND and @READY statements.

Note: The CONTINUE option should not be used in local mode if database changes have been made.

## @ROLLBACK Status Codes

After completion of the @ROLLBAK function, the ERRSTAT field of the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**0000**

The request has been serviced successfully.

**1958**

CONTINUE was specified and database changes could not be backed out. The run unit has been terminated.

## @ROLLBACK Example

The @ROLLBAK statement shown below reverses the effects of the run unit through which it is issued but does not terminate it.

```
@ROLLBAK CONTINUE
```

## #ROLLBAK—rolls back uncommitted changes made to the database

The #ROLLBAK statement rolls back uncommitted changes made to the database through an individual run unit or through all database sessions associated with a task. A task-level rollback also backs out all uncommitted changes made in conjunction with scratch, queue, and print activity.

Whether the changes are automatically backed out depends on the execution environment:

- If the changes were made under the control of a central version that is journaling to a disk file, they are backed out automatically. The central version continues to process other applications during recovery.
- The changes are not backed out automatically under the following circumstances:
  - If the changes were made under the control of a central version that is journaling to a tape file.
  - If the changes were made in local mode.

  In these cases, the #ROLLBAK statement causes the affected areas to remain locked against subsequent access by other database sessions. They must be manually recovered. If changes cannot be backed out and CONTINUE was specified on the rollback request, a non-zero error status is returned to the application and if the request was for an individual run unit, that run unit is terminated.

Note: For more information about manual recovery, see the *Database Administration Guide*.

If CONTINUE is not specified, run units (and SQL sessions) impacted by the #ROLLBAK statement end, and their access to the database is terminated. If CONTINUE is specified, impacted database sessions remain active after the operation is complete.

The #ROLLBAK statement is used in both the navigational and logical record facility environments. The #ROLLBAK TASK statement is also used in an SQL programming environment.

**Currency**

Following a #ROLLBAK statement, all currencies are set to null. Unless the CONTINUE option is specified, the issuing program or task cannot perform database access through an impacted run unit without executing another @BIND/@READY sequence.

## #ROLLBAK Syntax

```
►►─┬─────────┬─ #ROLLBAK ─┬──────┬─┬────────────┬──────────────────────────►◄
   └─ label ─┘            └ TASK ┘ └ ,CONTINUE ─┘
```

## #ROLLBAK Parameters

**TASK**

Rolls back the uncommitted changes made by all scratch, queue, and print activity and all top-level run units associated with the current task and terminates those run units. Its impact on SQL sessions associated with the task depends on whether those sessions are suspended and whether their transactions are eligible to be shared.

More information:

For more information about the impact of a #ROLLBAK TASK statement on SQL sessions, see the *SQL Programming Guide*.

For more information about run units and the impact of #ROLLBAK TASK, see the *Navigational DML Programming Guide*.

**CONTINUE**

Central version only. Causes the affected run units and SQL sessions to remain active after their changes are backed out. Database access can be resumed without reissuing @BIND and @READY statements.

Note: The CONTINUE option should not be used in local mode if database changes have been made.

## #ROLLBAK Status Codes

After completion of the #ROLLBAK function, the value in register 15 indicates the outcome of the operation. The following is a list of the Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'08'**

The request cannot be serviced due to an invalid request.

**X'10'**

CONTINUE was specified and database changes could not be backed out.

**X'0C'**

An error was encountered processing a syncpoint request; check the log for details.

## #ROLLBAK Example

The following backs out the uncommitted effects of all non-suspended database sessions associated with the task and all changes associated with scratch, queue and report processing. The affected database sessions are terminated.

```
#ROLLBAK TASK
```

## #SENDMSG—sends a message to another terminal or user

The #SENDMSG statement sends a message to another terminal or user, or to a group of terminals or users defined as a destination during system generation. The #SENDMSG statement does not send messages directly from the message area of the dictionary. Rather, the system places each message in a queue and sends the message to the appropriate terminals when it can do so without disrupting executing tasks. Normally, the system sends queued messages to a terminal the next time the ENTER NEXT TASK CODE message is displayed.

**Note:** For more information about message destinations, see the *System Generation Guide*.

## #SENDMSG Syntax

```
►►─┬───────┬─ #SENDMSG RECORD=message-location-pointer ──────────────►
   └ label ┘

 ►── ,RECLEN=message-length-register ─────────────────────────────────►

 ►─┬ ,DESTID=destination-id-pointer ──────────────────────────────────►
   ├ ,USERID=user-id-pointer ──────────────┤
   └ ,LTERMID=logical-terminal-id-pointer ─┘

 ►─┬────────────────────────────┬─────────────────────────────────────►
   └ ,OPTNS= ──┬─ ONLY ◄ ──┬────┘
               └─ ALWAYS ──┘

 ►─┬─────────────────────────────────────────────────┬────────────────►
   └ ,SMRB= ─┬─ SYSPLIST ◄ ──────────────────────────┤
             └─ send-message-request-block-pointer ──┘

 ►─┬──────────────────────────────────────┬───────────────────────────►
   └ ,COND= ──┬─ NO ◄ ────────────────────┤
              ├─ ALL ─────────────────────┤
              │     ┌──── , ────┐         │
              └─(───▼─┬─ IOER ──┴─┬─)─────┘
                      ├─ INVP ────┤
                      └─ UNDF ────┘

 ►─┬──────────────────────────┬───────────────────────────────────────►
   └ ,IOERXIT=i/o-error-label ┘

 ►─┬───────────────────────────────────────────┬──────────────────────►
   └ ,INVPXIT=invalid-parameter-list-label ────┘

 ►─┬────────────────────────────────────────┬─────────────────────────►
   └ ,UNDFXIT=undefined-destination-label ──┘

 ►─┬───────────────────────┬──────────────────────────────────────────►◄
   └ ,ERROR=error-label ───┘
```

## #SENDMSG Parameters

**RECORD=**

Specifies the location in program storage that contains the message to be sent.

***message-location-pointer***

Either a register that points to the message text or the symbolic name of the area that contains the message text.

**RECLEN=**

Specifies the length, in bytes, of the message text.

***message-length-register***

A register that contains the length of the message, the symbolic name of a user-defined field that contains the length, or an absolute expression.

**DESTID=**

Specifies the destination receiving the message. The destination is a list of logical terminals or users defined during system generation.

***destination-id-pointer***

A register that points to the destination ID, the symbolic name of a user-defined field that contains the ID, or the ID literal enclosed in quotation marks.

**USERID=**

Specifies the user to receive the message. The user can be signed on to any terminal.

***user-id***

A register that points to the user ID or the symbolic name of a user-defined field that contains the ID.

**LTERMID=**

Specifies the logical terminal to receive the message.

***logical-terminal-id-pointer***

A register that points to the logical terminal ID, the symbolic name of a user-defined field that contains the ID, or the ID literal enclosed in quotation marks.

**OPTNS=**

Specifies whether the system is to queue the message if the specified destination, user, or terminal is not currently being used.

**ONLY**

(Default); The DC/UCF system sends the message immediately if the destination, user, or terminal is available, and does not queue the message for subsequent transmission if the destination, user, or terminal is not available.

Note: If ONLY is specified with the DESTID parameter, described above, the system sends the message to those users or terminals in the destination that are available. The sender is not aware of any unsuccessful transmissions.

**ALWAYS**

The DC/UCF system sends the message immediately if the destination, user, or terminal is available, and queues the message for later transmission if the destination, user, or terminal is not available.

**SMRB=**

Specifies the location of the storage area in which the system builds the #SENDMSG parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system builds the #SENDMSG parameter list.

***send-message-request-block***

A register that points to the area or the symbolic name of the area in which the system builds the #SENDMSG parameter list.

**COND=**

Specifies whether this #SENDMSG is conditional and under what conditions control should be returned to the issuing program.

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that control is returned if the #SENDMSG cannot be serviced for one or more of the reasons listed in *condition*.

***condition-option***

Specifies conditions under which control is returned to the program. Multiple conditions must be enclosed in parentheses and separated by commas.

**IOER**

An I/O error occurred during processing.

**INVP**

The parameter list is invalid.

**UNDF**

The specified message destination is not defined to the system.

**IOERXIT=*i/o-error-label***

Specifies the symbolic name of a routine to which control should be returned if the #SENDMSG request cannot be serviced because of an I/O error.

**INVPXIT=*invalid-parameter-list-label***

Specifies the symbolic name of a routine to which control should be returned if the #SENDMSG cannot be serviced because of an invalid parameter list.

**UNDFXIT=*undefined-destination-label***

Specifies the symbolic name of a routine to which control should be returned if the #SENDMSG cannot be serviced because the specified destination is undefined to the system.

**ERROR=*error-label***

Specifies the symbolic name of a routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

## #SENDMSG Status Codes

By default, the #SENDMSG statement is unconditional; any runtime error results in an abend of the issuing task. The issuing program can request return of control with the COND parameter to avoid an abend.

After completion of the #SENDMSG, the value in register 15 indicates the outcome of the operation. The following is a list of Register 15 values and the corresponding meaning:

**X'04'**

The request cannot be serviced due to an I/O error during processing.

**X'08'**

The request cannot be serviced due to an invalid parameter list.

**X'0C'**

The request cannot be serviced because the message destination is undefined.

## #SENDMSG Example

The #SENDMSG statement shown below sends the message labeled MESS01 in program storage to a group of logical terminals identified by RMT007. The length of the message is held in LEN01. The DC/UCF system transmits the message immediately if any of the logical terminals in the destination are available, and queues the message for later transmission if none of the logical terminals are available.

```
#SENDMSG RECORD=MESS01,RECLEN=LEN01,DESTID=RMT007,OPTNS=ALWAYS
```

## #SETIME

The #SETIME statement defines an event that is to occur after a specified time interval or cancels the effect of a previously issued #SETIME request. The following time-related events can be defined:

- Delay task processing for a specified period of time
- Post an event control block (ECB) at the end of a specified period of time
- Initiate a task at the end of a specified period of time

### Syntax

```
►►─┬─────────┬─ #SETIME TYPE= ─┬─ WAIT ────┬──────────────────────────────►
   └─ label ─┘                 ├─ POST ────┤
                               ├─ STRTASK ─┤
                               └─ CANCEL ──┘

►─┬────────────────────────────────────────────┬─────────────────────────►
  └─ ,PLIST= ─┬─ SYSPLIST ◄ ──────────────┬─┘
              └─ parameter-list-pointer ─┘

►─┬──────────────────────────────────────────────────┬───────────────────►
  └─ ,INTVL=time-before-action-taken-register ─┘

►─┬──────────────────────────┬───────────────────────────────────────────►
  └─ ,ECB=post-ecb-pointer ─┘

►─┬───────────────────────────────────┬──────────────────────────────────►
  └─ ,TSKCD=start-task-code-pointer ─┘

►─┬──────────────────────────────┬───────────────────────────────────────►
  └─ ,PRI=priority-register ─┘

►─┬──────────────────────────────────────────────────┬───────────────────►
  └─ ,DATADDR=start-task-data-location-register ─┘

►─┬────────────────────────────────────────────────┬─────────────────────►
  └─ ,DATALEN=start-task-data-length-register ─┘

►─┬──────────────────────────────────────┬───────────────────────────────►
  └─ ,ICEADDR=ice-address-register ─┘

►─┬─────────────────────────────────────┬───────────────────────────────►◄
  └─ ,ICNFXIT=ice-not-found-label ─┘
```

### Parameters

**TYPE=**

Requests that the system establish a time-related event or cancels a previously requested time-dependent action.

**WAIT**

Places the issuing task in a wait state and instructs the system to redispatch the issuing task after the specified time interval elapses. A subsequent #SETIME request cannot be used to cancel this event until the time interval has elapsed.

**POST**

Posts an ECB after the specified time interval elapses. The issuing task continues to run. The ECB is specified using the ECB parameter (described below).

Note: The POST instruction will only POST an ECB that is within storage owned by the TASK initiating the POST instruction. If the storage is not owned by the same task, it will not be executed.

**STRTASK**

Initiates a task after the specified time interval elapses. The task is specified using the TSKCD parameter (described below).

**CANCEL**

Cancels the effect of a previously issued #SETIME request. If CANCEL is specified, the ICEADDR parameter (described below) must also be specified.

PLIST= Specifies the location of the six-fullword storage area in which the system builds the #SETIME parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system builds the parameter list.

***parameter-list-pointer***

Is a register that points to the area in which the system builds the list or the symbolic name of that area.

INTVL= (WAIT, POST, STRTASK requests only); specifies when the event is to occur. The interval is the amount of time in seconds between when the #SETIME request is issued to when the requested event is to occur.

***time-before-action-taken-pointer***

A register that contains the time interval, the symbolic name of a user-defined field that contains the time interval, or an absolute expression.

Note: For efficiency reasons, the time when the event is to occur is calculated by adding the INTVL value to the time at which the last TICKER interval expired. Therefore, the actual interval before the event occurs may vary plus or minus from INTVL by an amount up to the TICKER interval. For more information about the TICKER interval, see the *System Generation Guide*.

ECB= (POST only); specifies the location of the ECB to be posted.

***post-ecb-pointer***

A register that points to the ECB or the symbolic name of a user-defined field that contains the ECB. The ECB is an internal ECB which is three (3) fullwords in length and should be initialized with nulls.

TSKCD= (STRTASK only); specifies the 1- to 8-character task code of the task to be initiated.

***start-task-code-pointer***

A register that points to the task code, the symbolic name of a user-defined field that contains the task code, or the task-code literal enclosed by single quotation marks. The specified task code must have been defined during system generation or defined dynamically using the DCMT VARY DYNAMIC TASK command.

PRI= (STRTASK only); specifies a dispatching priority for the task to be initiated.

***priority***

A register that contains the priority or an absolute expression Valid codes are 0 through 240. The task's priority defaults to the priority defined for the task either during system generation or at dynamic definition using the DCMT VARY DYNAMIC TASK command.

DATADDR= (STRTASK only); identifies the user data to be passed to the new task.

***start-task-data-location***

A register that points to the data or the symbolic name of a user-defined field that contains the data. A register that points to the data or the symbolic name of a user-defined field that contains the data. The DATALEN parameter must be specified with DATADDR.

When the new task is started, the first program receiving control can access the data area (parameter list) through register 1. Register 1 will contain the address of a halfword which contains the value specified in DATALEN. This halfword will be followed by the data.

DATALEN= (STRTASK,DATADDR only); specifies the length, in bytes, of the data area identified by *start-task-data-location*.

***start-task-data-length-register***

A register that contains the length, the symbolic name of a user-defined field that contains the length, or an absolute expression.

ICEADDR= (POST, STRTASK, CANCEL only); specifies the address of the interval control element (ICE) associated with the time event.

**POST or STRTASK**

The optional ICEADDR parameter specifies the location to which the system returns the ICE address.

***ice-address-register***

A register or the symbolic name of a fullword user-defined field. name of a fullword user-defined field.

Note: The ICEADDR parameter must be specified with POST and STRTASK requests if the program is to issue subsequent #SETIME TYPE=CANCEL requests.

**CANCEL**

The ICEADDR must be specified. The ICEADDR references the location that contains the ICE address following a previously issued POST or STRTASK request.

ICNFXIT=*ice-not-found-label* (CANCEL only); specifies the symbolic name of the routine to which control should be returned if the ICE referenced by the ICEADDR parameter cannot be found. If ICNFXIT is not specified, control returns to the next sequential instruction following the #SETIME statement.

**Examples**

The #SETIME statement shown below requests that the system initiate the task labeled TSK01 sixty seconds after the #SETIME request is issued:

```
#SETIME TYPE=STRTASK,TSKCD='TSK01',INTVL=60
```

**Status Codes**

The #SETIME request is unconditional. Error conditions that can occur are described below:

- For wait, post, and start-task requests, any runtime error results in an abend of the issuing task.
- For cancel requests, any runtime error other than an interval-control-element-not-found condition results in an abend of the issuing task.

  The interval-control-element-not-found condition, caused when the ICE cannot be located, results in a return of control to the issuing program, either at a defined routine (ICNFXIT, described above) or at the next sequential instruction after the #SETIME statement.

After completion of the #SETIME request, the value in register 15 indicates the outcome of the operation. Register 15 values are significant only for requests that cancel a previously issued #SETIME request.

The following is a list of Register 15 values and the corresponding meaning:

**X'00'**

The request to cancel a previously issued #SETIME has been serviced successfully.

**X'04'**

The request to cancel a #SETIME request cannot be serviced because the specified ICE address cannot be found.

**X'08'**

The specified task code is not known to the DC/UCF system.

# #SNAP

The #SNAP statement requests a memory snap of one or more of the following areas:

- **Specified locations in memory**— The snap includes one or more areas of memory specifically requested by location and length.
- **Task areas**—The snap includes all resources associated with the issuing task, as well as the task control element (TCE), dispatch control element (DCE), logical terminal element (LTE), and physical terminal element (PTE) for the task. Information displayed by the snap is formatted with headers.
- **System areas**—The snap includes areas for all tasks and DC/UCF internal control blocks. Task areas are not itemized separately. Information displayed by the snap is formatted with headers.

The information requested by the #SNAP is written to the DC/UCF log file. A user-supplied title can be displayed with any of these types of snaps.

## Syntax

```
►►─┬─────────┬─ #SNAP ─────────────────────────────────────────────►
   └─ label ─┘

 ►─┬───────────────────────────┬───────────────────────────────────►
   └─ FORMAT=( ─┬─ ALL ──┬─ ) ─┘
                ├─ SYS ──┤
                └─ TASK ─┘

 ►─┬───────────────────────────────────────────┬───────────────────►
   └─ ,PLIST= ─┬─ SYSPLIST ◄ ────────────────┬─┘
               └─ parameter-list-pointer ────┘

 ►─┬──────────────────────────┬────────────────────────────────────►
   └─ ,TITLE=title-pointer ───┘

 ►─┬─────────────────────────────────────────────────────────────┬─►
   │          ┌──────────────────── , ─────────────────┐          │
   └─ ,AREA=( ─▼─ data-area-pointer,data-length-register ─┴─ ) ──┘

 ►─┬────────────────────────┬──────────────────────────────────────►◄
   └─ ,REGS= ─┬─ YES ◄ ─┬───┘
              └─ NO ────┘
```

## Parameters

### FORMAT=

Requests a formatted snap of system and/or task areas.

#### ALL

Requests that the system write a snap of both task and system areas. Areas associated with the issuing task are itemized and formatted separately from the system areas. The entire task control area is included as one item with a system snap.

**SYS**

Requests that the system write a snap of system areas.

Note: In most systems, this is a very large amount of memory; system snaps will impede system performance and should be reserved for special use.

**TASK**

Requests that the system write a snap of task areas and resources associated with the issuing task.

**PLIST=**

Specifies the location of the storage area in which the system builds the #SNAP parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system builds the #SNAP parameter list.

***parameter-list-pointer***

A register that points to the area or the symbolic name of the area in which the system builds the #SNAP parameter list.

Calculate the size of the parameter-list area using this formula:

5 + 2P + T

where the following conditions are met:

- *P* is the number of *data-area-pointer,data-length-register* pairs coded for the AREA parameter, described below.
- *T* is equal to 0 if the TITLE parameter, described below, has not been specified, or 1 if the TITLE parameter has been specified.

For example, if four pairs are specified and the TITLE parameter is omitted, the length of this storage area is 13 fullwords.

**TITLE=**

Specifies the title to be printed at the beginning of the snap. If requested, the title can be, at most, 133 characters. The first character must be a valid ASA carriage control character (Δ, 0, 1, or +). In addition, there must be a 1-byte field defined prior to the ASA control character which designates the length of the title field. For example, this denotes a length of 133:

```
LEN DC AL1(133)
```

***title-pointer***

A register that points to the title, or the symbolic name of a user-defined field that contains the title.

**AREA=**

Requests a snap of the specified areas. The AREA parameter can be specified independently of or together with the FORMAT specification. The memory defined by the AREA parameter may or may not be included in the memory areas associated with task or system areas specified by the FORMAT parameter.

***data-area-pointer***

Specifies the area to be snapped. *Data-area-pointer* may be the symbolic name of the area, or a register that points to the area. Register 1 is reserved for internal use; any other register is valid.

***data-length-register***

Specifies the length, in bytes, of the area to be included in the snap. *Data-length-register* is a register that contains the length, the symbolic name of a user-defined halfword or fullword field that contains the length, or an absolute expression of the length of the data area.

**REGS=**

Specifies whether values contained in the register should be printed.

**YES**

(Default); specifies that the snap includes all register values.

**NO**

Specifies that the snap does not include register values.

**Examples**

The #SNAP statement shown below requests a snap of two specific task areas. The MAINSAVE area (80 bytes in length) is the area to be snapped. A title is printed at the top of each page of the snap.

```
#SNAP AREA=(MAINSAVE,80),TITLE=TITLE1
          .
          .
          .
     TITLE1   DC    AL1(L'TITLE+1)
     CC       DC    C'1'
     TITLE    DC    C'ABEND EXIT PROGRAM AND WORKAREA SAMPLE'
```

**Status Codes**

The #SNAP request is unconditional; any runtime error results in an abend of the issuing task.

## #STAE

The #STAE (system task abend exit) statement establishes or cancels linkage to an abend routine. Control passes to the abend routine if the issuing task terminates abnormally. Any program in a task can establish a #STAE exit; only one abend exit can be in effect at any given time for each task level. If more than one abend exit has been established, the system recognizes the last #STAE request issued.

A task can terminate abnormally following a processing error or on request by an #ABEND function. Abend exits for the program that is executing at the time of the abend and for all higher level programs are executed before the task is terminated. You can override the automatic execution of abend exits by including an #ABEND function in the program or by including a #RETURN function in the abend routine.

**Note:** A #STAE command issued with no parameters cancels any previously issued #STAE. For further information see STAE Exits.

### Syntax

```
►►─┬───────────┬─ #STAE ─────────────────────────────────────────────►
   └─ label ───┘

 ►─┬─────────────────────────────────────┬───────────────────────────►◄
   ├─ PGM=program-name-pointer ──────────┤
   └─ EPADDR=entry-point-address-register┘
```

### Parameters

**PGM=**

Specifies whether linkage is established to another program or to an abend routine in the issuing program.

***program-name-pointer***

Identifies the 1- to 8-character name of the program. *Program-name-pointer* is a register that points to a field that contains the program name, the symbolic name of a user-defined field that contains the program name, or the program-name literal enclosed in quotation marks.

**Note:** The DC/UCF system does not test whether the specified program name is valid when the #STAE request is issued. If the program is not found or is otherwise unloadable when the system attempts to execute it, the #STAE request will be ignored.

**EPADDR=**

Identifies the abend entry-point address of an abend routine in the issuing program. The named routine must have a separate entry point in the program.

***entry-point-address-register***

Either a register or the symbolic name of a fullword user-defined field that contains the entry-point address.

### Example

The #STAE statement shown below establishes a link to the abend routine ABRT02. The program ABRT02 receives control in the event of an abnormal termination of the issuing task.

```
#STAE PGM=ABRT02
```

### Status Codes

The #STAE instruction is unconditional; any error detected during execution results in an abend of the issuing task.

## @STORE

### Functions of @STORE

The @STORE statement performs the following functions:

- Acquires space and a database key for a new record occurrence in the database
- Transfers the values of the appropriate elements from program variable storage to the requested record occurrence in the database
- Connects the requested record into all sets for which it is defined as an automatic member

### Location Modes

A record is stored in the database according to the location mode specified in the schema definition of the record. The location modes are as follows:

- **CALC** places the record on or near a page calculated by CA IDMS/DB from a control element (the CALC key) in the record.
- **VIA** places the record as follows:
  - If the owner and member record occurrences share a common page range, the DBMS places the record as close possible to its owner record occurrence.
  - If the owner and member record occurrences do not share a common page range, the DBMS places the record in the same relative position in the member record's page range as the owner record occurrence is in its associated page range.
- **DIRECT** places the record on or near a user-specified page, as determined by the value in the DIRDBKEY field of the IDMS communications block:
  - If DIRDBKEY contains a valid db-key for the record being stored, the DBMS assigns a db-key to the new record occurrence on that page if space is available.

  - If DIRDBKEY does not contain a valid db-key for the record being stored, the DBMS assigns the next available db-key, subject to the page-range limits of the record being stored.
  - If DIRDBKEY contains a value of -1, the DBMS assigns the record the first db-key available in the page range in which the record is to be stored.

  In any case, the db-key of the stored record occurrence is returned to DBKEY (positions 13-16 in the IDMS communications block). The contents of DIRDBKEY remain unchanged.

**Before Executing @STORE**

Before execution of the @STORE statement, the following conditions must be met:

- All areas affected either implicitly or explicitly by the @STORE statement must be readied in one of the three update usage modes. Update usage modes are discussed along with the @READY statement earlier in this chapter.
- All control elements (CALC and sorted set control fields) must be initialized.
- If the record being stored has a location mode of DIRECT, the contents of DIRDBKEY (the direct db-key, positions 197-200 of the IDMS communications block) must be initialized with a db-key value or a null db-key value of -1.
- If the record is to be stored in a native VSAM relative-record data set (RRDS), the contents of DIRDBKEY must be initialized with the relative record number that represents the location in the data set where the record is to be stored.
- Every set in which the named record is defined as an automatic member, and the owner record of every such set, must be included in the subschema. Sets for which the named record is defined as a manual member need not be defined in the subschema since the @STORE statement does not access those sets. An automatic member is connected automatically to the selected set occurrence when the record is stored; a manual member is not connected automatically to the selected set occurrence.
- If the record being stored has a location mode of VIA, currency must be established for the set in which the record participates as a member; this is true whether the record being stored is an automatic or manual member of that set.

**Currency**

Currency must be established for all set occurrences in which the stored record will participate as an *automatic* member. The @STORE statement uses currency depending on how the set is ordered:

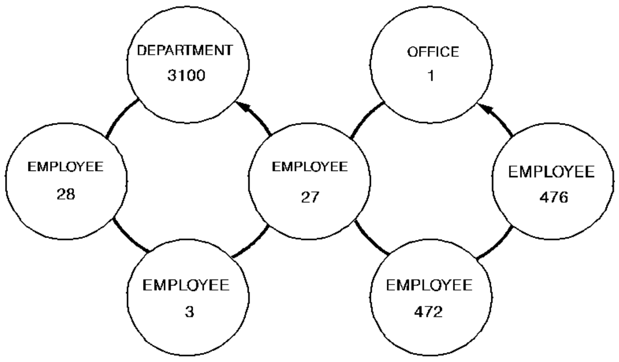
- If the stored record is defined as a **member of a set that is ordered FIRST or LAST**, the record that is current of set establishes the set occurrence to which the stored record will be connected.

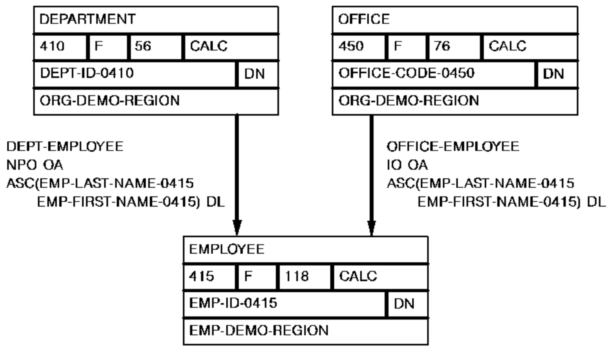
- If the stored record is defined as a **member of a set that is ordered NEXT or PRIOR**, the record that is current of set establishes the set occurrence into which the stored record will be connected and determines its position in the set.
- If the stored record is defined as a **member of a sorted set**, the record that is current of set establishes the set occurrence into which the stored record will be connected. IDMS compares the sort key of the stored record with the sort key of the current record of set to determine if the stored record can be inserted into the set by movement in the next direction:
  - If the record can be inserted by movement in the next direction, the set occurrence remains positioned at the record that is current of set and the stored record is inserted.
  - If the record cannot be inserted by movement in the next direction, the DBMS positions the set occurrence at the owner record occurrence (not necessarily the current occurrence of the owner record type) and moves as far forward in the next direction as is necessary to determine the logical insertion point for the stored record.

Following successful execution of an @STORE statement, the stored record becomes current of run unit, its record type, its area, and all sets in which it participates as owner or automatic member.

The following figure illustrates the currency issues involved in adding a new EMPLOYEE record to the database.

Since EMPLOYEE is defined as an automatic member of both the DEPT-EMPLOYEE and OFFICE-EMPLOYEE sets, currency must be established in each of those sets before issuing the @STORE statement. The first two DML commands establish DEPARTMENT-3100 and OFFICE-1 as current of the DEPT-EMPLOYEE and OFFICE-EMPLOYEE sets, respectively. When EMPLOYEE-27 is stored, it is connected automatically to each set.





| | CURRENCIES: RUN UNIT, RECORD, SET, AREA | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RUN UNIT | DEPARTMENT | EMPLOYEE | OFFICE | DEPT-EMPLOYEE | OFFICE-EMPLOYEE | ORG-DEMO-REGION | EMP-DEMO-REGION |
| MVC OFFCODE,OFFCODIN<br>@ FIND CALC, REC='OFFICE' | 1 | | | 1 | | 1 | 1 | |
| MVC DEPTID, DEPTIN<br>@FIND CALC, REC='DEPARTMENT' | 3100 | 3100 | | 1 | 3100 | 1 | 3100 | |
| @STORE, REC='EMPLOYEE' | 27 | 3100 | 27 | 1 | 27 | 27 | 3100 | 27 |

**Syntax**

```
►►── @STORE REC=record-name ─────────────────────────────────────────►◄
```

**Parameters**

**REC=*record-name***

Specifies the record occurrence to be moved from variable storage to the database. The @STORE statement connects the requested record to an occurrence of each set for which it is defined as an automatic member, and establishes it as the owner of a set. The @STORE statement also establishes the named record as the owner of a set occurrence for each set for which it is defined as an owner. The ordering rules for each set govern the insertion point of the named record in the set. *Record-name* must specify a record type included in the subschema.

**Example**

The @STORE statement shown below performs the following:

- Moves a single occurrence of the EMPLOYEE record from program variable storage to the database
- Connects this occurrence of EMPLOYEE to each set for which it is defined as an automatic member
- Establishes EMPLOYEE as the owner in each set occurrence in which it is defined as the owner

```
@STORE REC=EMPLOYEE
```

**Status Codes**

After the completion of the @STORE function, the ERRSTAT field in the IDMS communications block indicates the outcome of the operation. The following is a list of the acceptable status codes for this function and their corresponding meaning:

**1201**

The area in which the named record is to be stored has not been readied.

**1202**

The suggested DIRDBKEY value is not in the page range for the named record.

**1205**

Storage of the record would violate a duplicates-not-allowed option for a CALC record, a sorted set, or an index set.

**1208**

The named record is not in the subschema. The program has probably invoked the wrong subschema, or the record name has been misspelled.

**1209**

The named record's area has not been readied in one of the three update usage modes.

**1210**

The subschema specifies an access restriction that prohibits storage of the named record.

**1211**

The record cannot be stored in the area because of insufficient space.

**1212**

The record cannot be stored because no db-key is available. This is a system internal error.

**1218**

The record has not been bound.

**1221**

An area other than the area of the named record occurrence has been readied with an incorrect usage mode.

**1225**

A set occurrence has not been established for each set in which the named record is to be stored.

**1233**

All sets in which the record participates as an automatic member have not been included in the subschema.

**1253**

The subschema definition of an indexed set does not match the indexed set's physical structure in the database.

**1254**

Either the prefix length of an SR51 record is less than zero or the data length is less than or equal to zero.

**1255**

An invalid length has been defined for a variable-length record.

**1260**

A record occurrence encountered in the process of connecting automatic sets is inconsistent with the set named in the ERRORSET field of the IDMS communications block; probable causes include a broken chain or an improper database description.

**1261**

The record cannot be stored because of broken chains in the database.

# @STORE (LRF)

The @STORE statement can also update the database with field values for new logical record occurrences. The @STORE statement does not necessarily store new occurrences of all or any of the database records that participate in the logical record; the path selected to service an @STORE logical-record request performs whatever database access operations the DBA has specified to service the request.

LRF uses field values stored in the variable-storage location reserved for the logical record to make the appropriate updates to the database. You can optionally name an alternate storage location from which the new field values are to be obtained to perform the requested store operation.

**Syntax**

```
►►─── @STORE REC=logical-record-name ──────────────────────────────────►

►─┬──────────────────────────────────────┬───────────────────────────►
  └─ ,IOAREA=alt-logical-record-location ─┘

►─┬─────────────────────────────────────────────┬────────────────────►
  └─ ,ONLRSTS=path-status,GOTO=branch-location ─┘

►─┬───────────────────────────────┬──────────────────────────────────►◄
  └─ ,WHERE boolean-expression ───┘
```

**Parameters**

***REC=logical-record-name***

Names a new occurrence of the named logical record. Unless the IOAREA parameter (see below) is included, LRF updates the database by using field values stored in a variable-storage location reserved for the named logical record. *Logical-record-name* must specify a logical record defined in the subschema.

**IOAREA=*alt-logical-record-location***

Identifies an alternative variable-storage location that contains the field values to be used to update the database. When storing a logical record that has previously been retrieved into an alternative variable-storage location, you should use the IOAREA clause to name the same area specified in the @OBTAIN request. If the IOAREA clause is included in the @STORE statement, *alt-logical-record-location* must identify a record location defined in the program.

**ONLRSTS=*path-status*,GOTO=*branch-location***

Tests for the indicated path status. *Path-status* must be a quoted literal (1 to 16) or a program variable. If *path-status* results from this @STORE statement, the action specified by *branch-location* is performed. For more information about how to code the ONLRSTS clause, see the discussion of the ON clause later in this chapter.

**WHERE boolean-expression**

Specifies selection criteria to be applied to the named logical record. For details on how to code the WHERE clause, refer to the discussion of the WHERE clause later in this chapter.

**Example**

The example below illustrates how to add a new office by adding occurrences of the OFFEMPLR logical record. The program subsequently stores one occurrence of the OFFEMPLR logical record for each employee added to the office.

```
STOROFF  EQU   *
         MVC   OFFICE,NEWOFF
         @STORE REC=OFFEMPLR,WHERE ADD-OFFICE
         .
         .
STOREMP  EQU   *
         MVC   EMPL,NEWEMP
         @STORE REC=OFFEMPLR,WHERE ADD-EMP
         .
         .
         B     STOREMP
```

In the above example, the DBA has designated the keywords ADD-OFFICE and ADD-EMP to direct the request to a path designed to store new employee information for a new office. The path to which the first request is directed stores the appropriate new office information before storing the new employee information.

All input data concerning the new employee is contained in group fields called NEWOFF and NEWEMP, whose layouts correspond to those of the OFFICE and EMPLOYEE positions, respectively, of the OFFEMPLR logical record. The program moves the input field NEWOFF to the logical-record group field OFFICE and the input field NEWEMP to the logical-record group field EMPL.

**Status Codes**

After you issued an @STORE statement for a logical record, the type of status code returned to the program in the ERRSTAT field of the IDMS communications block depends on the type of error. If the error occurs in the *logical-record path*, the ERRSTAT field contains a status code issued by CA IDMS/DB with a major code from 00 to 19. For a list of these codes, see ERRSTAT Field and Codes (see page 41).

When the error occurs in the request itself, LRF returns the path status LR-ERROR to the LRSTAT field of the LRC block and places a status code with a major code of 20 in the ERRSTAT field of the IDMS communications block. These codes are listed in Testing for the Logical-Record Path Status (see page 55).

# #STRTPAG

The #STRTPAG statement initiates a map paging session, and specifies the map paging options in effect for that session. The paging session can contain any number of DML statements, including #MREQ IN and #MREQ OUT; the #STRTPAG statement must precede any of these mapping commands. The map paging session is terminated by an #ENDPAG statement, or by the next #STRTPAG statement if no #ENDPAG statement is coded.

**Note:** Only one pageable map can be handled by the statements enclosed by a given #STRTPAG/#ENDPAG pair.

**Syntax**

```
►►── #STRTPAG MRB=map-request-block-pointer ──────────────────────────────────►

►─┬──────────────────────────────────────────┬────────────────────────────────►
  └─ ,PLIST ─┬─ SYSPLIST ◄ ──────────────┬─┘
             └─ parameter-list-pointer ──┘

►─┬─────────────────────────────────────────────────┬─────────────────────────►
  └─ ,MRBPGDS= ─┬─ MRBPGDS ◄ ─────────────────────┬─┘
                └─ paging-request-block-pointer ──┘

►─┬──────────────────────────────┬────────────────────────────────────────────►
  └─ ,TYPE= ─┬─ NOWAIT ◄ ──┬─┘
             ├─ WAIT ──────┤
             └─ RETURN ────┘

►─┬─────────────────────────────┬─────────────────────────────────────────────►
  └─ ,BACKPAG= ─┬─ YES ◄ ─┬─┘
                └─ NO ────┘

►─┬──────────────────────────────┬────────────────────────────────────────────►
  └─ ,FLAG= ─┬─ UPDATE ◄ ─┬─┘
             └─ BROWSE ───┘

►─┬──────────────────────────┬────────────────────────────────────────────────►◄
  └─ ,AUTO= ─┬─ YES ◄ ─┬─┘
             └─ NO ────┘
```

**Parameters**

**MRB=*map-request-block-pointer***

Specifies the location of the map request block for the mapping operation, as copied into program variable storage by a previously issued #MRB statement.

***map-request-block-pointer***

Either a register that points to the MRB area or the symbolic name of that area.

**PLIST=**

The location of the storage area in which the system builds the #STRTPAG parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system builds the #STRTPAG parameter list.

***parameter-list-pointer***

Either a register that points to the area or the symbolic name of the area.

**MRBPGDS=**

Specifies the location of the 16-byte map paging request block.

**MRBPGDS**

(Default); is the symbolic name of the area in program variable storage that contains the map paging request block. The map paging request block is copied by a previously issued #MRB statement.

***paging-request-block-pointer***

Either a register pointing to the area that contains the map paging request block or the symbolic name of the area.

**TYPE=**

Specifies the runtime flow of control when the operator presses a control key.

**NOWAIT**

(Default); specifies that runtime mapping automatically handles all paging and update transactions. Control is passed to the program only when neither an update nor a paging request is made when the operator presses a control key.

**WAIT**

Specifies that runtime mapping automatically handles paging transactions that do not cause data to be updated. Control is passed to the program when the terminal operator presses a control key that requests an update or nonpaging operation.

**RETURN**

Specifies that runtime mapping does not handle any terminal transactions in the paging session. Control is passed to the program whenever the operator presses a control key.

Runtime mapping does not update program variable storage unless an #MREQ IN command is issued. In cases where the operator can update data (FLAG=UPDATE), it is recommended that WAIT and RETURN be specified for the session so that data can be retrieved as it is updated.

**BACKPAG=**

Specifies whether the terminal operator can display a previous map page.

**YES**

(Default); specifies that the operator can display previous pages of the map.

**NO**

Specifies that the operator cannot display any page of detail occurrences with a page number lower than the current page number. Modifications made on a given page of the map must be requested by #MREQ IN statements in the application program before an #MREQ OUT,RESUME=YES command is issued. The previous page of detail occurrences is deleted from the session scratch record when a new map page is displayed.

Note: BACKPAG=NO cannot be assigned if FLAG=UPDATE (discussed below) and TYPE=NOWAIT are specified for the session.

**FLAG=**

Specifies whether the terminal operator can modify map data fields.

**UPDATE**

(Default); specifies that the terminal operator can modify variable map fields, subject to restrictions specified for the map either at map definition time or by the statements in the program.

**BROWSE**

Specifies that the terminal operator can modify only the page and response fields of the map. At runtime, runtime mapping automatically protects all variable fields. The MDTs for variable fields on the map can be set only according to specifications made either in the map definition or by statements in the program.

**AUTO**

You can override the automatic mapout of a pageable map's first page. Overriding automatic display of a map's first page allows you to modify the map page and defined messages before the page is displayed. To determine when the first page of the map is built, you test the new map return code. By default, the first page of a pageable map is displayed as soon as the first detail occurrence of the second map page is written to scratch. You determine whether the first page of a pageable map is automatically displayed by using the **AUTO** parameter.

**YES**

(Default); enables automatic display of the pageable map's first page.

**NO**

Disables automatic display of the pageable map's first page. You manually display the page by using a #MREQ statement.

### Example

The following example of the #STRTPAG statement initiates a pageable map session with the following map paging options in effect:

- MRBPROG1 is the symbolic name of the location in variable storage that contains the map-request block for the mapping operation.

- WAIT indicates that runtime mapping passes control to the program when an update or a nonpaging request is made. Runtime mapping automatically handles all paging requests that do not involve field updates.
- Unless otherwise coded, the location of the map-paging request block is found in MRBPGDS in program variable storage. By default, the operator can display previous map pages and data fields.

  ```
  #STRTPAG MRB=MRBPROG1,TYPE=WAIT
  ```

The following example illustrates usage of the AUTO parameter:

```
#STRTPAG MRB=EMPMAPPG,AUTO=NO
```

### Status Codes

After completion of a #STRTPAG request, the value in register 15 indicates the outcome of the operation. The following is a list of Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'04'**

A paging session was already in progress when this #STRTPAG command was received. An implied #ENDPAG statement was processed before this #STRTPAG command was successfully executed.

## #TREQ

### Functions of #TREQ

The #TREQ statement allows your program to do the following:

- Transfer data between a terminal device and your application program in basic mode. Device-control characters appropriate to your terminal device are sent along with the data.
- Converse with SNA resources.
- Acquire and release storage areas used for I/O buffers. The following considerations apply:
  - In response to an *input request*, the input for data-item descriptions is acquired dynamically from the storage pool. Use the LOCATE option of the #TREQ GET or #TREQ READ statement to acquire the input buffer. When you specify LOCATE, your program is responsible for releasing the acquired storage with a #FREESTG statement. If the storage is not explicitly freed, the system releases all acquired input buffers when the task terminates.

- In response to an *output request*, a previously acquired storage area for the output buffer is released. To release the output buffer, use the FREEBUF option of the #TREQ PUT, #TREQ WRITE, #TREQ PUTGET, or #TREQ WRITREAD statement. The output buffer is released on completion of the output request.

**DC/UCF Response to #TREQ**

The DC/UCF system does the following in response to a #TREQ request:

- Automatically inserts the appropriate line control characters
- Builds and/or modifies a terminal request block (TRB), depending on the type of #TREQ request:
  - For *regular #TREQ requests* (MF=R), the system builds a new TRB for each request. Constant values are specified for each subsequent #TREQ request.
  - For *list #TREQ requests* (MF=L), the system builds a TRB in the data definition section of program storage. Subsequent #TREQ statements include parameters that add to or override this predefined TRB. The *list* #TREQ statement defines constant values; subsequent *execute* (MF=E) #TREQ statements modify the previously designate TRB. This technique saves coding time and storage space.
- Initiates the requested I/O operation and transfers the data

## Regular and Execute #TREQ Description

The regular and execute versions of the #TREQ statement request a transfer of data from the issuing program to the physical terminal and/or from the physical terminal to the program. The requested transfer is designated as synchronous or asynchronous:

- **Synchronous**—Control is not returned to the issuing program until the I/O operation is completed. Synchronous transfer is accomplished by using #TREQ GET, PUT, PUTGET, or ALLOC statements.
- **Asynchronous**—Control is returned to the issuing program immediately after the requested I/O operation is initiated; the program continues to execute concurrently with the I/O operation. An event control block (ECB) is established that will be posted after the I/O operation is completed. Asynchronous transfer is accomplished by using the #TREQ READ, WRITE, WRITREAD, or ALLOC statements.

  An asynchronous request must be followed by a #TREQ CHECK before continuing with further terminal I/O operations to ensure that the previous #TREQ processing is completed. Most error message codes associated with #TREQ READ, WRITE, WRITREAD, or ALLOC requests are returned when the #TREQ CHECK statement is issued.

To send a data stream immediately to a terminal or group of terminals, you can issue a #TREQ WRITE/PUT (blast) request, using the DESTID, USERID, and LTERMID parameters. For write-direct-to-terminal requests, the system ignores the SAVE, EOT, and TRANSPAR options. Write-direct-to-terminal requests are not supported for list #TREQ requests.

## Regular and Execute #TREQ Syntax

#TREQ syntax is presented alphabetically:

- #TREQ ALLOC
- #TREQ CHECK
- #TREQ DISC
- #TREQ GET
- #TREQ PUT
- #TREQ PUTGET
- #TREQ READ
- #TREQ UIOCB
- #TREQ WRITE
- #TREQ WRITEREAD

#TREQ syntax for list requests is presented in the next section.

**Syntax**

```
►►─┬───────────┬─ #TREQ ALLOC ──────────────────────────────────────►
   └─ label ───┘

 ►─── ,UIOCBA=user-io-control-block-pointer ─────────────────────────►

 ►─┬─────────────────────────┬───────────────────────────────────────►
   └─ ,MF= ──┬── R ◄ ──┬─────┘
             ├── E ────┤
             └── L ────┘

 ►─┬───────────────────────────────────────────────┬─────────────────►
   │                  ┌──────────── , ──────────┐   │
   └─,OPTNS= ─(──▼──┬──┬─ ANY ◄ ──┬──┬─────┴─)─┘
                    │  ├─ CONN ───┤  │
                    │  └─ IMM ────┘  │
                    ├─── WAIT ◄ ──┤  │
                    └─── NOWAIT ──┘──┘

 ►─┬──────────────────────────────────────────────┬──────────────────►◄
   └─ ,LTERMID=logical-terminal-element-pointer ─┘
```

**Parameters**

**#TREQ ALLOC**

Establishes a session and allocates an SNA conversation between your program and an SNA logical unit.

Note: For more information about using the #TREQ ALLOC statement, see Systems Network Architecture Considerations (SNA).

**Syntax**

```
►►─┬─────────┬─ #TREQ CHECK ──────────────────────────────────────────────►
   └─ label ─┘

►─┬───────────────────┬───────────────────────────────────────────────────►
  └─ ,MF= ─┬─ R ◄ ─┬─┘
           └─ E ───┘

►─┬──────────────────────────────────────────────────┬────────────────────►
  └─ ,TRB= ─┬─ SYSPLIST ◄ ─────────────────────────┬─┘
            └─ terminal-request-block-pointer ─────┘

►─┬────────────────────────────────────────────────────┬──────────────────►
  └─ ,INLEN= ─┬─ (0) ◄ ─────────────────────────────┬─┘
              └─ input-data-actual-length-register ─┘

►─┬──────────────────────────────────┬────────────────────────────────────►
  └─ ,COND= ─┬─ NO ◄ ──────────────┬─┘
             ├─ ALL ───────────────┤
             │      ┌──── , ────┐  │
             └─(─▼─┬─ ATTN ─┬──┴─)─┘
                   ├─ DISC ─┤
                   ├─ INVP ─┤
                   ├─ LOGL ─┤
                   ├─ PERM ─┤
                   └─ TRUN ─┘

►─┬─────────────────────────────────┬─────────────────────────────────────►
  └─ ,ATTNXIT=attention-key-label ──┘

►─┬───────────────────────────────────────┬───────────────────────────────►
  └─ ,DISCXIT=terminal-disconnected-label ┘

►─┬──────────────────────────────────────────┬────────────────────────────►
  └─ ,INVPXIT=invalid-trb-information-label ─┘

►─┬─────────────────────────────────────┬─────────────────────────────────►
  └─ ,LOGLXIT=logical-output-error-label┘

►─┬─────────────────────────────────────┬─────────────────────────────────►
  └─ ,PERMXIT=permanent-i/o-error-label ┘

►─┬─────────────────────────────────────┬─────────────────────────────────►
  └─ ,TRUNXIT=truncate-input-data-label ┘

►─┬──────────────────────┬───────────────────────────────────────────────►◄
  └─ ,ERROR=error-label ─┘
```

**Parameters**

#TREQ CHECK Delays task processing until a previously requested asynchronous I/O operation is completed. The DC/UCF system places the task in an inactive state if the I/O operation is incomplete. When the I/O operation is complete, the task resumes processing according to its established dispatching priority.

**Syntax**

```
►►─┬─────────┬─ #TREQ DISC ───────────────────────────────────────────────►
   └─ label ─┘

►─┬───────────────────────────────────┬──────────────────────────────────►◄
  └─ ,LTEADDR=lte-address-register ───┘
```

## Parameters

### #TREQ DISC

(SNA conversations only); terminates an SNA session between your program and another logical unit.

### Syntax

```
►►─┬───────┬─ #TREQ GET ──────────────────────────────────────────────►
   └ label ┘

 ►─┬─────────────────┬────────────────────────────────────────────────►
   └ ,MF= ─┬─ R ◄ ─┬─┘
           └─ E ───┘

 ►─┬──────────────────────────────────────────────────┬───────────────►
   └ ,TRB= ─┬─ SYSPLIST ◄ ───────────────────────────┬┘
            └─ terminal-request-block-pointer ───────┘

 ►─┬────────────────────────────────────┬─────────────────────────────►
   └ ,INAREA=input-data-location-pointer ┘

 ►─┬───────────────────────────────────────┬──────────────────────────►
   └ ,MAXIN=input-data-max-length-register ┘

 ►─┬───────────────────────────────────────────────────┬──────────────►
   └ ,INLEN= ──┬─ (0) ◄ ──────────────────────────────┬┘
               └─ input-data-actual-length-register ──┘

 ►─┬───────────────────────────────┬──────────────────────────────────►
   └ ,LTEADDR=lte-address-register ┘

 ►─┬────────────────────────────────────────┬─────────────────────────►
   │                      ┌─────── , ───────┐│
   └ ,OPTNS= ──(─▼─┬──── BUFFER ───────┬──┴─)┘
                   ├─┬── INFMHY ──┬────┤
                   │ └── INFMHN ──┘    │
                   ├─┬── LL ──────┬────┤
                   │ └── NOCHASM ─┘    │
                   ├──── LOCATE ───────┤
                   ├──── MODIFIED ─────┤
                   ├──── POSITION ─────┤
                   ├──── UPLOW ────────┤
                   └──── UPPER ────────┘

 ►─┬────────────────────────────┬─────────────────────────────────────►
   └ ,FROMPOS=screen-position ──┘

 ►─┬─────────────────────────────────────┬────────────────────────────►
   └ ,COND= ──┬──── NO ◄ ───────────────┬┘
              ├──── ALL ────────────────┤
              │       ┌──── , ────┐     │
              └(──▼─┬── DISC ──┬──┴─)───┘
                    ├── INVP ──┤
                    ├── PERM ──┤
                    └── TRUN ──┘

 ►─┬───────────────────────────────────────┬──────────────────────────►
   └ ,DISCXIT=terminal-disconnected-label ─┘

 ►─┬─────────────────────────────────────────┬────────────────────────►
   └ ,INVPXIT=invalid-trb-information-label ─┘
```

```
►─┬──────────────────────────────────────────┬────────────────────────►
  └─ ,PERMXIT=permanent-i/o-error-label ─────┘

►─┬──────────────────────────────────────────┬────────────────────────►
  └─ ,TRUNXIT=truncate-input-data-label ─────┘

►─┬───────────────────────────┬───────────────────────────────────────►◄
  └─ ,ERROR=error-label ──────┘
```

**Parameters**

**#TREQ GET**

Requests synchronous transfer of data from a device to program storage when the terminal operator signals completion of data entry by pressing ENTER or a special function key.

**Syntax**

```
►►─┬─────────┬─ #TREQ PUT ────────────────────────────────────────────►
   └─ label ─┘

►─┬──────────────────────┬────────────────────────────────────────────►
  └─ ,MF= ─┬─ R ◄ ─┬─────┘
           └─ E ───┘

►─┬──────────────────────────────────────────────────────┬────────────►
  └─ ,TRB= ─┬─ SYSPLIST ◄ ─────────────────────────────┬─┘
            └─ terminal-request-block-pointer ─────────┘

►─┬──────────────────────────────────────────────┬────────────────────►
  └─ ,OUTAREA=output-data-location-pointer ──────┘

►─┬────────────────────────────────────────────────────┬──────────────►
  └─ ,OUTLEN= ─┬─ output-data-length-register ───────┬─┘
               └─ log-data-length-register ──────────┘

►─┬──────────────────────────────────────────────┬────────────────────►
  │                  ┌─────────── , ───────────┐ │
  └─ ,OPTNS= ──(─▼─┬─── CHNCONT ──────────┬───┴─)┘
                   ├─── CONFIRM ──────────┤
                   ├─── CONFIRMED ────────┤
                   ├─── EOT ──────────────┤
                   ├─── ERASUNPR ─────────┤
                   ├─── ERROR ────────────┤
                   ├─── FREEBUF ──────────┤
                   ├─── INVITE ───────────┤
                   ├─── NEWPAGE ──────────┤
                   ├─── NOCR ─────────────┤
                   ├─┬─ OUTFMHY ─┬────────┤
                   │ └─ OUTFMHN ─┘        │
                   ├─── SAVE ─────────────┤
                   ├─── SIGNAL ───────────┤
                   └─── TRANSPAR ─────────┘

►─┬──────────────────────────────────────┬────────────────────────────►
  └─ ,LTEADDR=lte-address-register ──────┘

►─┬──────────────────────────────────────┬────────────────────────────►
  └─ ,SENSE=sna-sense-code-register ─────┘

►─┬──────────────────────────────────────────┬────────────────────────►
  └─ ,LOGDATA=log-data-address-register ─────┘
```

```
►─┬─────────────────────────────────────────────┬──────────────────────────────►
  ├─ ,DESTID=destination-id-pointer ────────────┤
  ├─ ,USERID=user-id-pointer ───────────────────┤
  └─ ,LTERMID=logical-terminal-id-pointer ──────┘

►─┬──────────────────────────────────────────┬─────────────────────────────────►
  └─ ,COND= ──┬── NO ◄ ───────────────────┬──┘
              ├── ALL ────────────────────┤
              │       ┌──── , ────┐       │
              └─(──┬──▼── ATTN ───┬──)────┘
                   ├───── DISC ───┤
                   ├───── INVP ───┤
                   ├───── LOGL ───┤
                   ├───── PERM ───┤
                   └───── UNDF ───┘

►─┬────────────────────────────────────┬───────────────────────────────────────►
  └─ ,ATTNXIT=attention-key-label ─────┘

►─┬──────────────────────────────────────────┬─────────────────────────────────►
  └─ ,DISCXIT=terminal-disconnected-label ───┘

►─┬─────────────────────────────────────────────┬──────────────────────────────►
  └─ ,INVPXIT=invalid-trb-information-label ────┘

►─┬────────────────────────────────────────┬───────────────────────────────────►
  └─ ,LOGLXIT=logical-output-error-label ──┘

►─┬──────────────────────────────────────┬─────────────────────────────────────►
  └─ ,PERMXIT=permanent-i/o-error-label ─┘

►─┬───────────────────────────────────────────┬────────────────────────────────►
  └─ ,UNDFXIT=invalid-destid-ltermid-label ───┘

►─┬──────────────────────────┬────────────────────────────────────────────────►◄
  └─ ,ERROR=error-label ─────┘
```

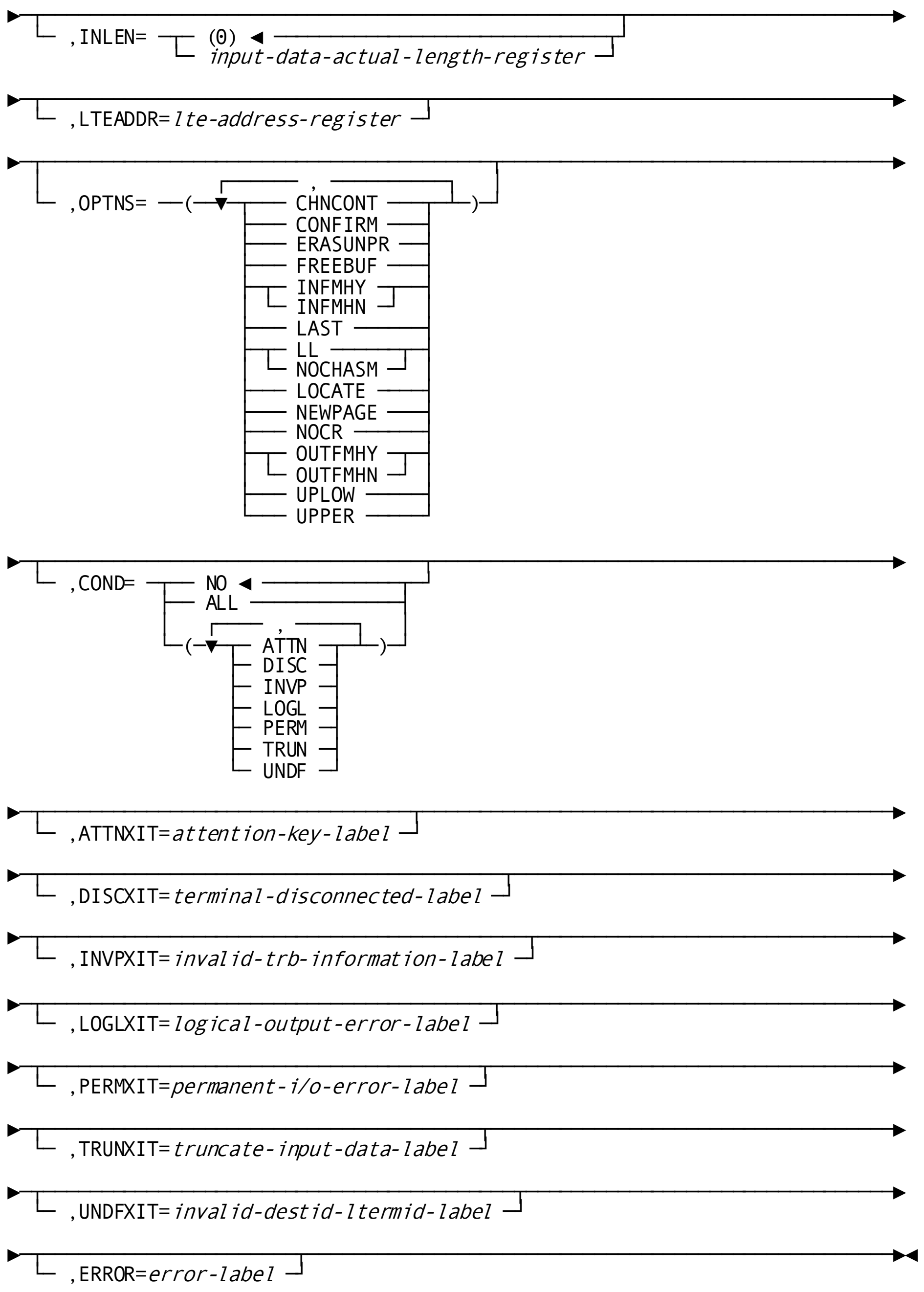
**Parameters**

**#TREQ PUT**

Requests synchronous transfer of data from program storage to a terminal or device.

**Syntax**

```
►►─┬───────────┬─ #TREQ PUTGET ────────────────────────────────────────────────►
   └─ label ───┘

►─┬──────────────────────┬─────────────────────────────────────────────────────►
  └─ ,MF= ──┬─ R ◄ ─┬────┘
            └─ E ───┘

►─┬──────────────────────────────────────────────────┬─────────────────────────►
  └─ ,TRB= ──┬── SYSPLIST ◄ ─────────────────────────┤
             └── terminal-request-block-pointer ─────┘

►─┬─────────────────────────────────────────────┬──────────────────────────────►
  └─ ,OUTAREA=output-data-location-pointer ─────┘

►─┬─────────────────────────────────────────────┬──────────────────────────────►
  └─ ,OUTLEN=output-data-length-register ───────┘

►─┬─────────────────────────────────────────────┬──────────────────────────────►
  └─ ,INAREA=input-data-location-pointer ───────┘

►─┬─────────────────────────────────────────────┬──────────────────────────────►
  └─ ,MAXIN=input-data-max-length-register ─────┘
```

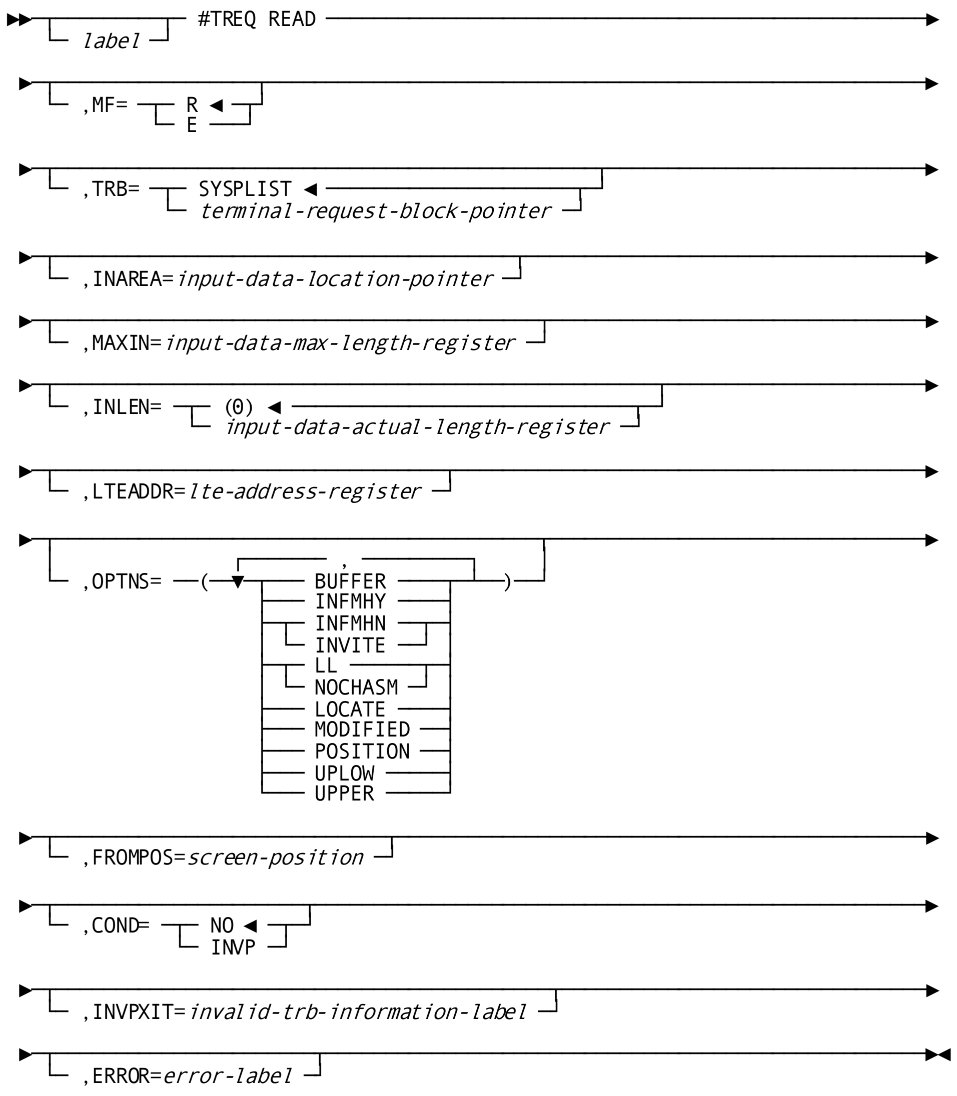

,INLEN=
(0)
input-data-actual-length-register
,LTEADDR=lte-address-register
,OPTNS=
(
,
CHNCONT
CONFIRM
ERASUNPR
FREEBUF
INFMHY
INFMHN
LAST
LL
NOCHASM
LOCATE
NEWPAGE
NOCR
OUTFMHY
OUTFMHN
UPLOW
UPPER
)
,COND=
NO
ALL
(
,
ATTN
DISC
INVP
LOGL
PERM
TRUN
UNDF
)
,ATTNXIT=attention-key-label
,DISCXIT=terminal-disconnected-label
,INVPXIT=invalid-trb-information-label
,LOGLXIT=logical-output-error-label
,PERMXIT=permanent-i/o-error-label
,TRUNXIT=truncate-input-data-label
,UNDFXIT=invalid-destid-ltermid-label
,ERROR=error-label

## Parameters

### #TREQ PUTGET

Requests a synchronous data transfer from program storage to a terminal, then back to the program when the terminal operator indicates completion of data entry.

**Syntax**


label
#TREQ READ
,MF=
R
E
,TRB=
SYSPLIST
terminal-request-block-pointer
,INAREA=input-data-location-pointer
,MAXIN=input-data-max-length-register
,INLEN=
(0)
input-data-actual-length-register
,LTEADDR=lte-address-register
,OPTNS=
(
,
BUFFER
INFMHY
INFMHN
INVITE
LL
NOCHASM
LOCATE
MODIFIED
POSITION
UPLOW
UPPER
)
,FROMPOS=screen-position
,COND=
NO
INVP
,INVPXIT=invalid-trb-information-label
,ERROR=error-label

**Parameters**

**#TREQ READ**

Requests asynchronous transfer of data from a terminal or device to program storage when the terminal operator signals completion of the data entry by pressing ENTER or a special function key.

#TREQ UIOCB

**Syntax**

```
►►─┬───────┬─ #TREQ UIOCB ────────────────────────────────────────────►
   └ label ┘

 ►─┬─────────────────────────────────────────────┬────────────────────►
   └ ,UIOCBA=user-i/o-control-block-pointer ─────┘

 ►─┬─────────────────────────────────────┬────────────────────────────►◄
   └ ,LTEADDR=lte-address-register ──────┘
```

**Parameters**

**#TREQ UIOCB**

Locates a user I/O communications block used to maintain the status of an SNA conversation and of the data being passed between logical units.

**Syntax**

```
►►─┬───────┬─ #TREQ WRITE ────────────────────────────────────────────►
   └ label ┘

 ►─┬─────────────────────┬────────────────────────────────────────────►
   └ ,MF= ─┬─ R ◄ ─┬─────┘
           └─ E ───┘

 ►─┬─────────────────────────────────────────────────┬────────────────►
   └ ,TRB= ─┬─ SYSPLIST ◄ ───────────────────────┬───┘
            └─ terminal-request-block-pointer ───┘

 ►─┬──────────────────────────────────────────┬───────────────────────►
   └ ,OUTAREA=output-data-location-pointer ───┘

 ►─┬───────────────────────────────────────────────┬──────────────────►
   └ ,OUTLEN= ─┬─ output-data-length-register ──┬──┘
               └─ log-data-length-register ─────┘

 ►─┬─────────────────────────────────────┬────────────────────────────►
   └ ,LTEADDR=lte-address-register ──────┘

 ►─┬──────────────────────────────────────────┬───────────────────────►
   │                 ┌───────── , ──────────┐ │
   └ ,OPTNS= ──(──▼─┬─── ABEND ───────┬─────┴─)─┘
                    ├─── CHNCONT ─────┤
                    ├─── CONFIRM ─────┤
                    ├─── CONFIRMED ───┤
                    ├─── EOT ─────────┤
                    ├─── ERASUNPR ────┤
                    ├─── ERROR ───────┤
                    ├─── FREEBUF ─────┤
                    ├─── INVITE ──────┤
                    ├─── LAST ────────┤
                    ├─── NEWPAGE ─────┤
                    ├─── NOCR ────────┤
                    ├─┬─ OUTFMHY ─┬───┤
                    │ └─ OUTFMHN ─┘   │
                    ├─── SAVE ────────┤
                    ├─── SIGNAL ──────┤
                    └─── TRANSPAR ────┘
```

```
►─┬───────────────────────────┬────────────────────────────────────────────►
  └─ ,SENSE=sna-sense-code ───┘

►─┬──────────────────────────────────────────┬─────────────────────────────►
  └─ ,LOGDATA=log-data-address-register ─────┘

►─┬──────────────────────────────────────────────┬─────────────────────────►
  ├─ ,DESTID=destination-id-pointer ─────────────┤
  ├─ ,USERID=user-id-pointer ────────────────────┤
  └─ ,LTERMID=logical-terminal-id-pointer ───────┘

►─┬───────────────────────────────────────┬────────────────────────────────►
  └─ ,COND= ─┬── NO ◄ ──────────────────┬─┘
             ├── ALL ───────────────────┤
             │      ┌───── , ────┐      │
             └─(─▼─┬── INVP ──┬─┴──)────┘
                   └── UNDF ──┘

►─┬────────────────────────────────────────────────┬───────────────────────►
  └─ ,INVPXIT=invalid-trb-information-label ───────┘

►─┬───────────────────────────────────────────────┬────────────────────────►
  └─ ,UNDFXIT=invalid-destid-ltermid-label ───────┘

►─┬─────────────────────────────┬──────────────────────────────────────────►◄
  └─ ,ERROR=error-label ────────┘
```

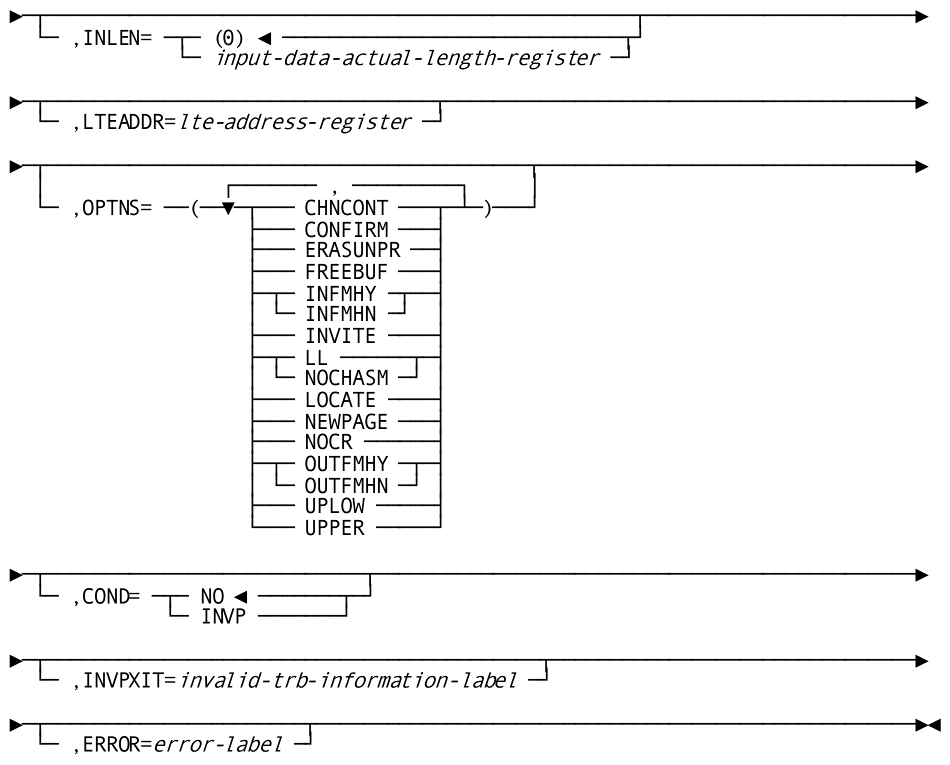
**Parameters**

**#TREQ WRITE**

Requests an asynchronous data transfer from program storage to a terminal or device.

**Syntax**

```
►►─┬─────────┬─ #TREQ WRITREAD ────────────────────────────────────────────►
   └─ label ─┘

►─┬─────────────────┬──────────────────────────────────────────────────────►
  └─ ,MF= ─┬─ R ◄ ─┬┘
           └─ E ───┘

►─┬─────────────────────────────────────────────────┬──────────────────────►
  └─ ,TRB= ─┬─ SYSPLIST ◄ ──────────────────────────┤
            └─ terminal-request-block-pointer ──────┘

►─┬──────────────────────────────────────────────┬─────────────────────────►
  └─ ,OUTAREA=output-data-location-pointer ──────┘

►─┬─────────────────────────────────────────────┬──────────────────────────►
  └─ ,OUTLEN=output-data-length-register ───────┘

►─┬─────────────────────────────────────────────┬──────────────────────────►
  └─ ,INAREA=input-data-location-pointer ───────┘

►─┬──────────────────────────────────────────────┬─────────────────────────►
  └─ ,MAXIN=input-data-max-length-register ──────┘
```



**Parameters**

**#TREQ WRITREAD**

Requests an asynchronous data transfer from program storage to a terminal or device, then back to program storage when the terminal operator indicates completion of data entry.

Syntax rules for the #TREQ statements are shown below. One complete set of syntax rules is provided; the explanation for each parameter indicates the applicable #TREQ statements. #TREQ options necessary to comply with SNA protocols are also indicated.

Note: For more information about SNA programming considerations, see Systems Network Architecture Considerations (SNA).

The discussion of syntax applies to regular and execute commands, with the following exceptions:

- The TRB parameter of an execute request identifies a terminal request block (TRB) previously established by a list #TREQ request.
- For an execute request, parameters already defined to the TRB need not be specified. If they are specified, the requested parameters will override the existing values in the TRB.

**Parameters**

**MF=**

Specifies the category of #TREQ.

**R**

(Default); specifies a regular #TREQ statement.

**E**

Specifies an execute #TREQ statement.

**TRB=**

Specifies the five-fullword storage area in which the system will build the TRB (MF=R) or has built the TRB (MF=E).

**SYSPLIST**

(Default for regular requests only); is the symbolic name of the storage area in which the system will build the TRB.

***terminal-request-block***

Either a register that contains the address of the area or the symbolic name of the area in which the system will build or has built the TRB. For execute requests, this entry explicitly defines the area by identifying *label*, provided in the list #TREQ that generated the TRB.

**OUTAREA=**

(PUT, WRITE, PUTGET, and WRITREAD only) specifies the storage area that contains data to be output. OUTAREA need not be defined if the OUTLEN parameter, described below, is 0.

***output-data-location***

Either a register that contains the address of the area or the symbolic name of the area.

**OUTLEN=**

(PUT, WRITE, PUTGET and WRITREAD only); specifies the length, in bytes, of the data stream to be transmitted.

***output-data-length***

Specifies the length of data being sent to a terminal. *Output-data-length* is either a register that contains the length or an absolute expression of the length of data sent in a normal exchange.

***log-data-length***

A register that contains the length or an absolute expression of the length of data to be sent along with error information in an SNA conversation.

**INAREA=**

(GET, READ, PUTGET, and WRITREAD only); specifies the storage area into which the data will be read. When INAREA is specified, the LOCATE option, described under the OPTNS parameter below, should not be requested.

***input-data-location***

Either a register that points to the area or the symbolic name of the area.

**MAXIN=**

(GET, READ, PUTGET, and WRITREAD only); specifies the maximum length in bytes of the data area defined by INAREA that is reserved for the input data stream. When MAXIN is specified, the LOCATE option, described under the OPTNS parameter below, should not be requested.

***input-data-max-length***

Either a register that contains the length of the data area or an absolute expression.

**INLEN=**

(GET, PUTGET, or CHECK following an asynchronous input request); specifies the location to which the system returns the actual length of the input data stream. If the input data stream has been truncated, the original length of the data stream before truncation is returned.

**(0)**

(Default); is the register to which the system returns the actual length of the input data stream.

***input-data-actual-length***

A register or the symbolic name of a halfword or fullword user-defined field to which the system will return the actual length of the input data stream.

**UIOCBA=**

(ALLOC and UIOCB only); specifies the location of the storage area that contains the user I/O control block (UIOCB) used for the conversation.

***user-i/o-control-block***

Either a register containing the address or the symbolic name of the area.

Note: For more information about the user I/O control block, see Systems Network Architecture Considerations (SNA).

**LTEADDR=**

(SNA only); specifies the address of the logical terminal element (LTE) of the remote task in the conversation.

***lte-address***

Either a register containing the address or the symbolic name of the area.

**OPTNS=(*treq-option*)**

Specifies several options applicable to the input or output operation. Multiple values must be enclosed in parentheses and separated by commas.

The BUFFER, MODIFIED, and POSITION options specify special purpose read operations for 3270 devices. They should not be confused with normal READ/GET requests that read modified fields when the operator presses the ENTER key or a special function key.

**ABEND (SNA WRITE only)**

Notifies the remote system that the task is terminating abnormally and that the conversation has ended.

**ANY/CONN/IMM (SNA ALLOC only)**

Specifies the type of session to be established:

- **ANY** (default) specifies that the system allocate a session in the following order:

  **1.** A session that is *immediately available* and currently unused

  **2.** A session that is *disconnected*; that is, the session has not yet been established

  **3.** A session that is *busy*; that is, the session is established and is allocated to another task

  If neither an immediately available nor a disconnected session is available, the system waits for a busy session to become available.

- **CONN** requests the system not to wait for a busy session. The system first attempts to allocate an immediately available session, then a disconnected session.

- **IMM** specifies that only immediately available sessions are acceptable for the allocation request.

**BUFFER**

(GET and READ with 3270 devices only) Indicates that the data will be transmitted to program storage automatically. BUFFER requests that the system execute an immediate READ BUFFER command; this reads the entire contents of the 3270 terminal buffer into the program storage specified by INAREA and MAXIN.

**CHNCONT**

(SNA non-LU6.2 PUT, PUTGET, WRITE, and WRITREAD only) Specifies that the user task is providing a chain of outbound messages, and that the current #TREQ output request is not the last message in the chain. Omitting OPTNS=CHNCONT after it has been specified once indicates that the current message is the final chain element.

**CONFIRM**

(SNA PUT, PUTGET, WRITE, and WRITREAD only) Sends a confirmation request to a remote SNA logical unit. For example, when your program specifies #TREQ WRITE,OPTNS=CONFIRM, the system requests that the remote logical unit confirm that the request has been sent.

**CONFIRMED**

(SNA WRITE and PUT only) Sends a positive response to a confirmation request.

**EOT**

(PUT or WRITE to 3741 or 3780 bisynchronous batch terminals only) Specifies that there is no more data to follow.

**ERASUNPR**

(PUT, PUTGET, WRITE, and WRITREAD with 3270 devices only) Causes the system to activate the erase-all-unprotected mechanism. Because no data is transferred for an ERASUNPR request, use of this option implies that OUTLEN=0; no output data need be defined with the OUTAREA parameter described above.

**ERROR**

(SNA WRITE and PUT only) Allows a task to send a negative response to a remote logical unit, or to reject a confirmation request that was found unacceptable. Do not use the ERROR option if the CONFIRMED option is specified.

**FREEBUF**

(PUT, PUTGET, WRITE, and WRITREAD only) Frees the storage area that contains the output data stream. The buffer area being freed must have been acquired by a #GETSTG statement or the LOCATE option of a previously issued input request.

If FREEBUF is not specified, the system does not release the output buffers associated with the output request until the issuing task terminates. When the task is terminated, all storage acquired by a #GETSTG or a LOCATE will be released.

**INFMHY/INFMHN**

(SNA non-LU6.2 GET, READ, PUTGET and WRITREAD only) Specifies whether a function management header (FMH) is passed to the program:

- **INFMHY** indicates that all FMHs on the inbound message are passed to the program.
- **INFMHN** requests that the system remove any FMHs before the data is passed to the program.

**INVITE**

(SNA WRITE, PUT, WRITREAD, and PUTGET) Allows a task to specify a change of direction from the send to the receive state.

**LAST**

(SNA WRITE and PUT only) Ends a conversation between two logical units.

**LL/NOCHASM**

(SNA GET, PUTGET, READ, WRITREAD only) Specifies the format of the data to be input to the program:

- **LL** indicates whether a generalized data stream (GDS) header is to be removed from an LU6.2 data record before it is received by a conversation. For a mapped conversation, LL specifies that one LU6.2 data record is received with the GDS header removed. For an unmapped conversation, LL specifies that one LU6.2 data record is received, without GDS removal.
- **NOCHASM** (SNA GET, PUTGET, READ, and WRITREAD only; not allowed for LU6.2 mapped conversations) specifies that inbound chains in a conversation are passed to the user task individually. The chains are passed a single chain element at a time, without assembling the entire chain into a buffer. A single chain element consists of one SNA request unit (RU).

**LOCATE**

(GET, PUTGET, READ, and WRITREAD only) Allocates a buffer area for the data being read into the program, rather than a user-specified area. The DC/UCF system allocates the buffer when the read operation is completed. Register 1 contains the address of the buffer that will contain the input data on completion of the input operation. The issuing program is responsible for using a #FREESTG to free the buffer area.

When this option is requested, do not specify INAREA and MAXIN.

**MODIFIED**

(GET and READ with 3270 devices only) Indicates that the data will be transmitted to program storage automatically, without waiting for a signal of completion of data entry from the terminal operator. MODIFIED requests that the system read all *modified* fields in the 3270 terminal buffer into the program storage specified by INAREA and MAXIN.

**NEWPAGE**

(PUT, PUTGET, WRITE, and WRITREAD with SYSINOUT or 3270 devices only) Requests that the system activate the page-eject (SYSINOUT) or erase-write (3270) mechanism to erase the contents of a screen. If NEWPAGE is not specified, the #TREQ request will write over any existing screen display without first erasing it.

**NOCR**

(PUT, PUTGET, WRITE, and WRITREAD with teletype terminals only) Specifies that carriage-control and line-feed characters should *not* be automatically appended to an output data stream.

**OUTFMHY/OUTFMHN**

(SNA non-LU6.2 PUT, PUTGET, WRITE, and WRITREAD only) Specifies whether a function management header (FMH) has been included in the beginning of the write buffer:

- **OUTFMHY** indicates that an FMH has been provided. The FMH overrides the default defined at system generation.
- **OUTFMHN** indicates that no FMH has been added to the outbound message.

**POSITION**

(GET and READ with 3270 devices only), used in conjunction with the BUFFER or MODIFIED options, indicates that the FROMPOS parameter, described below, will specify the position at which the read buffer contents will begin.

**UPPER**

(GET, PUTGET, READ, and WRITREAD only); Directs the system to translate all letters in an input data stream into uppercase characters.

**SAVE**

(PUT and WRITE non-write-direct-to-terminal only) Directs the system to preserve the output from the #TREQ request in the event that an unsolicited write-direct-to-terminal data stream is received at the issuing terminal while the #TREQ data stream is being displayed. This option overrides the task SAVE/NOSAVE option specified during system generation.

**SIGNAL**

(SNA WRITE and PUT only) Requests a change of direction from the receive to the send state. SIGNAL is used with the SENSE parameter, discussed below.

**TRANSPAR**

(PUT or WRITE to 3741 or 3780 bisynchronous batch terminals only) Specifies that the output may contain line control characters and must be written with a transparent write operation.

**UPLOW**

(GET, PUTGET, READ, and WRITREAD only) Specifies that no uppercase translation of characters in an input data stream is performed.

**UPPER**

(GET, PUTGET, READ, and WRITREAD only) Directs the system to translate all letters in an input data stream into uppercase characters.

**WAIT/NOWAIT**

(SNA ALLOC only) Specifies whether the allocation request is synchronous or asynchronous:

- **WAIT** (default) indicates that the allocation request is synchronous.
- **NOWAIT** indicates that the allocation request is asynchronous. After specifying #TREQ ALLOC with OPTNS=NOWAIT, you must code a #TREQ CHECK request before any other I/O requests are issued. The NOWAIT option cannot be specified with OPTNS=ANY.

**SENSE=**

(SNA WRITE and PUT only); specifies a sense code that describes errors that the system encounters in conversation processing.

***sna-sense-code***

Either a register containing the sense code or a 4-byte hexadecimal value enclosed in quotation marks. Sense codes supported by the system are listed in Systems Network Architecture Considerations (SNA).

**LTERMID=**

(ALLOC only); identifies the logical terminal element (LTE) of a remote logical unit in an SNA conversation, or a write-direct-to-terminal destination for a non-SNA #TREQ request.

***logical-terminal-element-name***

Either a register pointing to the area containing the LTE or the name of a user-supplied variable data field that holds the address.

**LOGDATA=**

(SNA LU6.2 WRITE only); specifies the address of a data buffer containing data that will be sent along with error information to the remote task.

***log-data-address***

Either a register containing the address of the data buffer or a user-defined variable field. When LOGDATA is specified, you must code the OUTLEN parameter to indicate the length of the data being sent.

**FROMPOS=**

(#TREQ GET and READ requests with 3270 devices; BUFFER or MODIFIED options only); specifies the 2-character EBCDIC buffer address at which the read will start.

***screen-position***

Either the symbolic name of a user-defined fixed binary field that contains the buffer address or the address itself enclosed in quotation marks.

**DESTID/USERID/LTERMID**

(PUT and WRITE only); specifies the destination of a write-direct-to-terminal request.

**DESTID=**

Specifies a write-direct-to-terminal request (blast) to one of the following destinations defined during system generation:

- **List of logical terminals** indicates that the system will send the #TREQ data stream specified in the OUTAREA parameter to all available terminals in the list.

- **List of users** indicates that the system will send the #TREQ data stream specified in the OUTAREA parameter to all users in the list who are currently signed on the system.

***destination-id***

A register that points to the destination id, the symbolic name of a user-defined field that contains the destination id, or the id itself enclosed in quotation marks.

Each destination should refer to terminal devices of the same type to ensure compatibility with program-supplied device control information. If a #TREQ blast request is routed to an incompatible device type, the system will reject the request and return control to the issuing program.

**USERID=**

Specifies a blast request to a specific signed-on user. The DC/UCF system will send the #TREQ data stream specified in the OUTAREA parameter to a specific signed-on user.

***user-id***

A register that points to the user id, the symbolic name of a user-defined field that contains the user id, or the id itself enclosed in quotation marks.

**LTERMID=**

Specifies a blast request to a specific in-service terminal. The DC/UCF system will send the #TREQ data stream specified in the OUTAREA parameter to a specific in-service terminal.

***logical-terminal-id***

A register that points to the logical terminal id, the symbolic name of a user-defined field that contains the logical terminal id, or the id itself enclosed in quotation marks.

**COND=**

Specifies whether the #TREQ is conditional and under what conditions control should be returned to the issuing program:

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned if the request cannot be serviced for any of the reasons listed under *condition*.

***condition***

Can be any of the following options. Multiple options must be enclosed in parentheses and separated by commas. *Condition* options are as follows:

- **ATTN** (PUT, PUTGET, or CHECK)—The I/O operation is interrupted by the terminal operator pressing ATTENTION or BREAK.
- **DISC** (PUT, GET, PUTGET, or CHECK)—The dial-up line is disconnected or the terminal goes out of service.
- **INVP**—There is an invalid parameter in the TRB.
- **LOGL** (PUT, PUTGET, or CHECK)—A logical error is encountered in the output data stream.
- **PERM** (PUT, GET, PUTGET, or CHECK)—A permanent I/O error occurs during processing.
- **TRUN** (GET, PUTGET, or CHECK)—The data has been truncated due to insufficient storage in the specified INAREA.
- **UNDF**—Control is returned if an undefined DESTID or LTERMID is specified in a #TREQ blast request.

The following parameters represent routines to which control is returned as a result of one of the preceding conditions:

**ATTNXIT=*attention-key-label***

Specifies the symbolic name of the routine to which control should be returned if the output is interrupted by the terminal operator.

**DISCXIT=*terminal-disconnected-label***

Specifies the symbolic name of the routine to which control should be returned if the terminal is disconnected or the terminal goes out of service.

**INVPXIT=*invalid-trb-information-label***

Specifies the symbolic name of the routine to which control should be returned if the #TREQ cannot be serviced because of an invalid parameter in the TRB.

**LOGLXIT=*logical-output-error-label***

Specifies the symbolic name of the routine to which control should be returned if a logical error is detected in the output data stream.

**PERMXIT=*permanent-i/o-error-label***

Specifies the symbolic name of the routine to which control should be returned if a permanent I/O error occurs.

**TRUNXIT=*truncate-input-data-label***

Specifies the symbolic name of the routine to which control should be returned if input data is truncated due to insufficient storage in the INAREA buffer.

**UNDFXIT=*invalid-destid-ltermid-label***

Specifies the symbolic name of the routine to which control should be returned if an undefined DESTID or LTERMID is specified in a #TREQ PUT or WRITE blast request.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control should be returned if a condition specified in the COND parameter occurs for which no other exit routine was coded.

**Examples**

The following examples illustrate how to use the #TREQ statement.

The following #TREQ ALLOC statement allocates a session between your LU and a remote LU that is identified in the user I/O control block. OPTNS=ANY specifies that the system will attempt to assign a currently unused session first; if one is not available it will attempt to assign a session that has not yet been established. If neither of these session types is possible, the system will wait for a busy session to become available. OPTNS=WAIT indicates synchronous processing. COND=ALL specifies that control is returned to the program request cannot be serviced due to any terminating conditions.

```
#TREQ ALLOC,UIOCBA=(R3),OPTNS=(ANY,WAIT),COND=ALL
```

The following #TREQ DISC statement terminates a session between your LU and the remote LU identified by the LTE address contained in register 8.

```
#TREQ DISC,LTEADDR=(R8)
```

The following #TREQ GET statement transfers data from a terminal to program variable storage after the terminal operator presses the ENTER key. #TREQ GET indicates synchronous data transfer. SYSPLIST is the symbolic name of the storage area in which the system builds the TRB. Input read from the terminal is moved to INPROG02; the maximum length of the input data is 40 bytes.

```
#TREQ GET,MF=R,TRB=SYSPLIST,INAREA=INPROG02,MAXIN=40
```

The following #TREQ PUT statement issues a write-direct-to-terminal request. The blast request transfers the 50 byte output data stream in OUTPGM9 directly to all users in the currently signed-on users in DEST09.

```
#TREQ PUT,TRB=SYSPLIST,OUTAREA=OUTPGM9,OUTLEN=50,DESTID=DEST09
```

The following #TREQ PUTGET statement is being used in a non-LU6.2 SNA conversation between the system task and a remote 3600 device. The remote LU is identified by the LTE address in LU3603 because your task may be having more than one conversation at a time. The data you are sending is held in the output buffer OUT09, and can be up to 60 bytes long. If the data returned by the remote LU exceeds the MAXIN specification (60 bytes), the system buffers the data so that it will be available to your next read request. OUTFMHN requests the system not to add any function management headers to the output data stream. INFMHN requests that the system remove any incoming FMH from the data before it is passed to your task.

```
#TREQ PUTGET,OUTAREA=OUT09,OUTLEN=60,INAREA=IN09,MAXIN=60,               *
      LTEADDR=LU3603,OPTNS=(OUTFMHN,INFMHN)
```

The following execute #TREQ READ statement reads the contents of the buffer INAREA. The MODIFIED option specifies that modified data is transmitted to program storage automatically, without waiting until the terminal operator has signaled completion of data entry. The NEWPAGE option requests that the system erase the contents of the screen before the new data is read in. Control is returned to the RTNINVP routine if there is an invalid parameter in the TRB.

```
#TREQ READ,MF=E,TRB=SYSPLIST,INAREA=INAREA,OPTNS=(MODIFIED,NEWPAGE),     *
      COND=INVP,INVPXIT=RTNINVP
```

The following #TREQ WRITE statement requests that the system initiate the erase-all-unprotected mechanism for output. No data is transferred with this request (OUTLEN=0); no output data has to be defined in OUTAREA.

```
#TREQ WRITE,OUTAREA=OUTPGM9,OPTNS=(ERASUNPR)
```

The following #TREQ WRITREAD statement sends the output data stream in the buffer OUTPGM08 to the terminal. FREEBUF releases the contents of OUTPGM08 after the WRITREAD request has been completed. OUTPGM08 must have been previously acquired by a #GETSTG statement or the LOCATE option of a previously issued input request. Data is sent from the terminal to the INPGM08 buffer.

```
#TREQ WRITREAD,TRB=SYSPLIST,OUTAREA=OUTPGM08,OUTLEN=60,        *
      INAREA=INPGM08,INLEN=60,OPTNS=FREEBUF
```

The following #TREQ UIOCB statement assigns a user I/O control block to an SNA conversation started by a remote task. The address of the UIOCB is in register 8.

```
#TREQ UIOCB,UIOCBA=(R8)
```

**Status Codes**

Upon successful completion of certain #TREQ requests, three registers contain information about the outcome of the request:

- **Register 0** contains the actual number of terminals to which the data stream has been routed for a blast request (PUT or WRITE).
- **Register 1** contains information related to the type of request:
  - For asynchronous requests, Register 1 contains the address of the ECB that will be posted by the system on completion of the I/O operation.
  - For LOCATE requests and after asynchronous CHECK requests, register 1 contains the address of the buffer into which the input data has been placed.
- **Register n** contains the actual length of returned data for an input operation (GET, PUTGET, READ, or WRITREAD). The register number *n* is assigned by the INLEN parameter.

By default, the #TREQ request is unconditional; any runtime error will result in an abend of the issuing task. The issuing program can request return of control with the COND parameter to avoid an abend.

After completion of the #TREQ, the value in register 15 indicates the outcome of the operation. The following is a list of the Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'04'**

For a GET, PUTGET, or CHECK request, the input area specified for the return of data to the issuing program is too small; the returned data has been truncated to fit the available space.

**X'08'**

For a GET, PUTGET, or CHECK request, the output has been interrupted; the terminal operator has pressed ATTENTION or BREAK.

**X'0C'**

For a GET, PUTGET, or CHECK request, a logical error (for example, an invalid control character) has been encountered in the output data stream.

**X'10'**

For a GET, PUT, PUTGET, or CHECK request, a permanent I/O error has occurred during processing.

**X'14'**

For a GET, PUT, PUTGET, or CHECK request, the dial-up line for the terminal has been disconnected.

**X'18'**

For a GET, PUT, PUTGET, or CHECK request, the terminal associated with the issuing task is out of service.

**X'1C'**

For a GET, PUTGET, or CHECK request, the terminal is closed, or was never opened.

**X'20'**

The TRB contains an invalid field, indicating a possible error in the program parameters.

**X'24'**

For a PUT or WRITE request, the requested logical terminal id or list of logical terminals or users identified by LTERMID, USERID, or DESTID cannot be found.

## List #TREQ

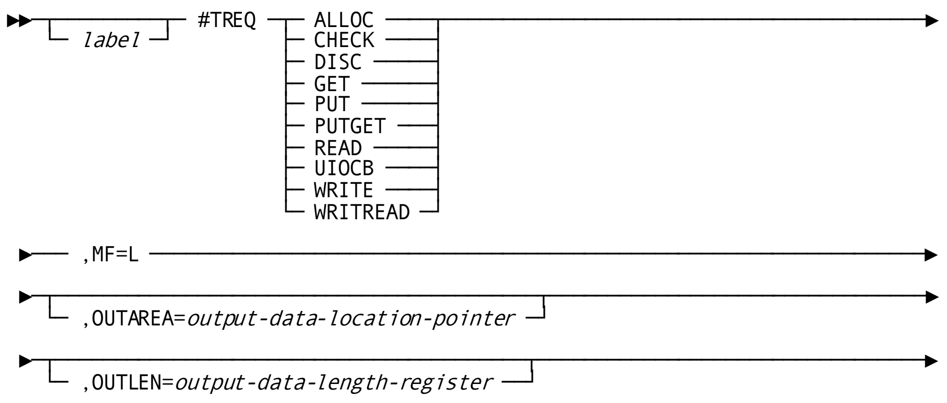
Using the list #TREQ you can build a terminal request block (TRB) in the data definition section of program storage, and assign constant values. After you have issued one list #TREQ statement, subsequent *execute* #TREQ statements override only the fields in the named TRB that need to be updated.

The TRB is identified by the list #TREQ *label*. This label is referenced by the TRB parameter in subsequent execute requests.

In the list #TREQ, only the *label* and the MF parameter are required; all other parameters should be specified only when required to predefine TRB parameter values.

In a list #TREQ request, parameter values *cannot* be specified by using register notation. The list #TREQ syntax presented here shows only those parameters that are affected by this restriction. Syntax for the list #TREQ statement is shown below:

**Syntax**


label
#TREQ
ALLOC
CHECK
DISC
GET
PUT
PUTGET
READ
UIOCB
WRITE
WRITREAD
,MF=L
,OUTAREA=output-data-location-pointer
,OUTLEN=output-data-length-register

```
►─┬─────────────────────────────────────┬──────────────────────────────►
  └─ ,INAREA=input-data-location-pointer ┘

►─┬───────────────────────────────────────┬────────────────────────────►
  └─ ,MAXIN=input-data-max-length-register ┘

►─┬──────────────────────────────┬─────────────────────────────────────►
  └─ ,UIOCBA=user-i/o-control-block ┘

►─┬──────────────────────┬─────────────────────────────────────────────►
  └─ ,LTEADDR=lte-address ┘

►─┬────────────────────────┬───────────────────────────────────────────►
  └─ ,SENSE=sna-sense-code ┘

►─┬─────────────────────────────┬──────────────────────────────────────►
  └─ ,LOGDATA=log-data-address ┘

►─┬────────────────────────────────────┬───────────────────────────────►◄
  ├─ ,DESTID=destination-id-pointer ───┤
  ├─ ,USERID=user-id-pointer ──────────┤
  └─ ,LTERMID=logical-terminal-id-pointer ┘
```

## Parameters

### ALLOC/CHECK/DISC/GET/PUT/PUTGET/READ/UIOCB/WRITE/WRITREAD

Specifies the type of #TREQ statement.

### MF=L

Specifies a list #TREQ.

Each parameter (other than MF=L) functions identically to the corresponding parameter in the regular and execute forms of #TREQ statements, described previously.

For example, the value specified for OUTAREA must be a symbolic name of a user-defined area, whereas in the regular and execute forms it could be either a register that points to the area or the symbolic name of the area.

## #TRNSTAT

The #TRNSTAT statement enables your program to access transaction statistics about task-related activities. The system allocates a block of storage, called a transaction statistics block (TSB), in which to accumulate these statistics.

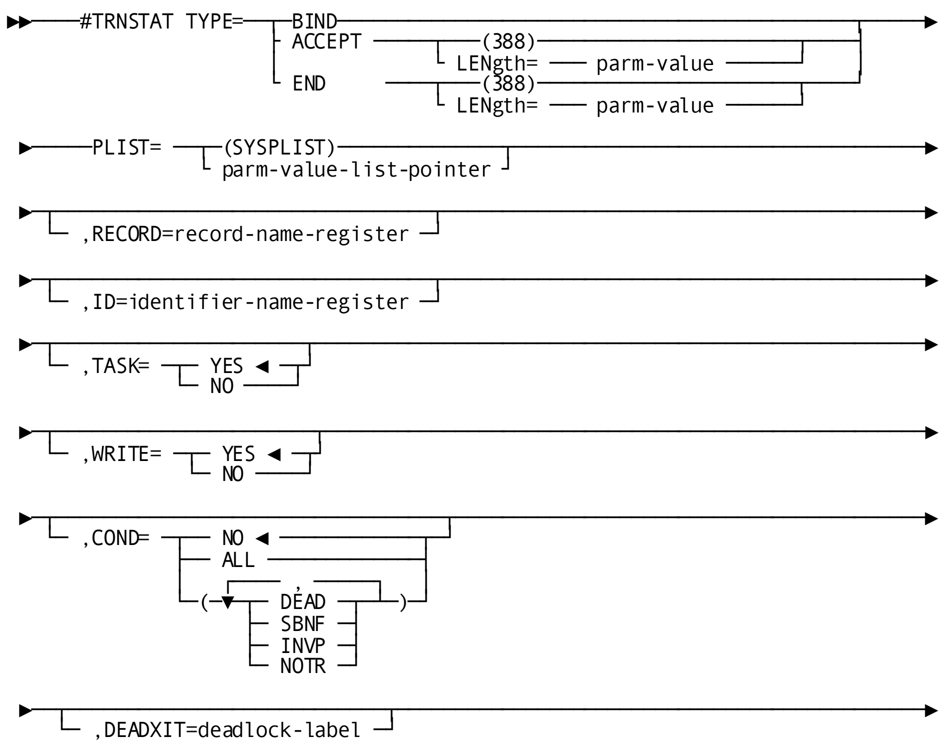
Three versions of the #TRNSTAT statement collect and write transaction statistics:

- **#TRNSTAT TYPE=BIND** starts recording transaction statistics for the requestor's logical terminal.
- **#TRNSTAT TYPE=ACCEPT** copies transaction statistics from the TSB and places them in a storage area associated with the issuing task and/or writes them to the DC/UCF log file.
- **#TRNSTAT TYPE=END** stops collecting transaction statistics for the requestor's logical terminal and optionally writes the statistics to a storage area associated with the issuing task and/or to the DC/UCF log file.

**Note:** Do not attempt to collect transaction statistics using the #TRNSTAT statement if your Assembler program is a subroutine to a CA ADS dialog.

For more information about the transaction statistics block (TSB), see the *DSECT Reference Guide*.

### Syntax



```
►─┬──────────────────────────────────────────────┬──────────────────────►
  └─ ,SBNFXIT=statistics-block-not-found-label ─┘

►─┬────────────────────────────────────────┬────────────────────────────►
  └─ ,INVPXIT=invalid-parameter-list-label ─┘

►─┬───────────────────────────────────────────────────────┬─────────────►
  └─ ,NOTRXIT=no-transaction-statistics-allowed-label ─┘

►─┬──────────────────────────┬────────────────────────────────────────►◄
  └─ ,ERROR=error-label ─┘
```

**Parameters**

**TYPE=**

Specifies the type of transaction statistics activity.

**BIND**

Defines the beginning of a transaction for the purposes of collecting transaction statistics. The system allocates a block of storage to collect these statistics. Because this block is owned by the logical terminal associated with the current task, the #TRNSTAT=BIND can only be used with terminal tasks.

**Note**: If a terminal statistics block (TSB) is already allocated for the logical terminal associated with the current task, the BIND request writes any existing statistics to the log and clears the TSB for new statistics.

When a #TRNSTAT TYPE=BIND request is issued, the system assigns the transaction a 40-character identifier. The first 32 characters are the identifier of the signed-on user, if any. The last 8 characters are the identifier of the logical terminal associated with the current task.

**ACCEPT**

Requests that the system return the contents of the TSB to a preallocated location in program storage and/or write the block to the DC/UCF log file by the WRITE option described below. The system does not delete the contents of the TSB as a result of the ACCEPT option; transaction statistics can accumulate between #TRNSTAT statements where the ACCEPT option is specified. To prevent the program from altering the contents of the TSB and to ensure integrity of the data, the system returns a copy of the TSB to the program.

**LENGTH (parm-value)**

Specifies the length of the TSB to be returned. Can be specified as a value, register or storage area.

**Default:** 388

**END**

Ends the transaction and frees the TSB. The system ends the transaction when the task issuing the #TRNSTAT TYPE=END request terminates. Optionally, END can write the TSB to a preallocated location in program storage by using the RECORD option described below. To prevent the program from altering the contents of the TSB and to ensure integrity of the data, the system returns a copy of the TSB to the program.

**LENGTH (parm-value)**

Specifies the length of the TSB to be returned. Can be specified as a value, register or storage area.

**Default:** 388

**PLIST (parm-value-list-pointer)**

Specifies the location of the storage area where he system builds the #TRNSTAT parameter list.

**RECORD=**

(#TRNSTAT TYPE=ACCEPT or END requests only); specifies the location of the storage area into which the system places the TSB.

***record-name-register***

A register that contains the location of the area, the symbolic name of the area, or an absolute expression.

**ID=**

(for #TRNSTAT TYPE=BIND requests only) Specifies the location of a storage area that contains an 8-byte identifier to be placed in the Transaction Statistics Block.

***identifier-name-register***

Specifies a register that contains the location of the identifier, a symbolic name of the identifier's location, or an absolute expression.

**TASK=**

(for #TRNSTAT TYPE=BIND or END requests only) Specifies the action that is taken relative to the current task.

**YES**

(Default)

(for #TRNSTAT TYPE=BIND requests only) Specifies that the collection of statistics starts at the beginning of the current task.

(for #TRNSTAT TYPE=END requests only) Specifies that if statistics are being written to the DC/UCF log file, they are written at the end of the current task.

**NO**

(for #TRNSTAT TYPE=BIND requests only) Specifies that the collection of statistics starts at the time of the execution of the #TRNSTAT macro.

(for #TRNSTAT TYPE=END requests only) Specifies that if statistics are being written to the DC/UCF log file, they are written immediately.

**WRITE=**

(for #TRNSTAT TYPE=ACCEPT or END requests only) Specifies that the system writes the contents of the TSB to the DC/UCF log file.

**YES**

(Default) Specifies that the system writes the TSB to the log file.

**NO**

Specifies that the system does not write the TSB to the log file.

**COND=**

Specifies whether the #TRNSTAT request is conditional and under what error conditions control should be returned to the issuing program.

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that control is returned to your program if the #TRNSTAT request cannot be serviced for any of the reasons described under *condition*.

***condition***

Specifies one or more conditions under which the system returns control to the issuing program. Multiple conditions must be enclosed in parentheses

and separated by commas. The following options can be specified:

- **DEAD** (TYPE=BIND only) specifies that storage for the TSB is not available; waiting would cause a deadlock.
- **SBNF** specifies that a TSB for the user terminal cannot be found for a #TRNSTAT TYPE=ACCEPT or END request. This condition is probably due to a #TRNSTAT BIND not having been issued.
- **INVP** specifies that the requested task is not associated with a logical terminal or that the request is invalid.

- **NOTR** specifies that transaction statistics or task statistics are not enabled in the DC/UCF system.

**DEADXIT=*deadlock-label***

Specifies the symbolic name of a routine to which the system returns control if storage for the TSB is not available, and waiting would cause a deadlock.

**SBNFXIT=*statistics-block-not-found-label***

Specifies the symbolic name of a routine to which the system returns control if a TSB for the terminal cannot be found for a #TRNSTAT TYPE=ACCEPT or END request.

**INVPXIT=*invalid-parameter-list-label***

Specifies the symbolic name of the routine to which the system returns control when the requested task is not associated with a logical terminal or when the request is invalid.

**NOTRXIT=*no-transaction-statistics-allowed-label***

Specifies the symbolic name of a routine to which the system returns control when transaction statistics or task statistics are not enabled in the DC/UCF system.

**ERROR=*error-label***

Specifies the symbolic name of a routine to which the system returns control if a condition specified in the COND parameter occurs for which no other exit routine was coded.

**Example**

The #TRNSTAT statement shown below requests that the system return the contents of the TSB to TSBAREA in program variable storage and to write the block to the DC/UCF log file. Control is returned to the program if this request would result in a deadlock or if the TSB cannot be found.

```
#TRNSTAT TYPE=ACCEPT,RECORD=TSBAREA,WRITE=YES,COND=(SBNF,DEAD)
```

**Status Codes**

By default, the #TRNSTAT statement is unconditional; any runtime error will result in an abend of the issuing task.

After completion of the #TRNSTAT request, the value in register 15 indicates the outcome of the operation. The following is a list of Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully. For TYPE=BIND only, an existing TSB has been written to the DC/UCF log.

**X'04'**

The request has been serviced; a new TSB has been allocated (TYPE=BIND only).

**X'08'**

Storage for the TSB is not available and waiting would cause a deadlock (TYPE=BIND only).

**X'0C'**

No TSB exists; a #TRNSTAT TYPE=BIND request has not been issued (TYPE=ACCEPT or END only).

**X'10'**

The task issuing the #TRNSTAT request is not associated with a logical terminal or the request is invalid.

**X'14'**

The collection of task statistics or transaction statistics was not enabled during system generation.

# #WAIT

The #WAIT statement relinquishes control to the system. Control is relinquished for one of the following reasons:

- To wait for the completion of one or more events
- To give other higher priority ready-to-run tasks a chance to be dispatched by the system.

If a task relinquishes control to await completion of an event, an event control block (ECB) must be defined for each event for which the task is waiting. If an ECB is already posted when the #WAIT is issued, the task is redispatched immediately and control does not pass to another task.

An ECB is a binary three-fullword field used to indicate the status of an event. If the ECB contains zeros, the event is not complete or has not been posted. If the ECB contains a nonzero value, the event has been posted. The ECB field can be allocated explicitly by individual programs or implicitly by the system:

- **Program allocation**—A three-fullword storage area must be defined in the variable storage of the associated programs. Programs using the ECB field are responsible for establishing addressability to the ECB as well as indicating the status of the event.
- **DC/UCF system allocation** — The three-fullword field associated with the ECB is allocated by the system. To wait on an event, the program specifies the 4-character ECB ID. The system associates the ECB ID with a fullword field and automatically sets the status of the ECB field.

**Syntax**

```
►►─┬─────────┬─ #WAIT ───────────────────────────────────────────────────►
   └─ label ─┘

 ►─┬────────────────────────┬────────────────────────────────────────────►
   └─ TYPE= ─┬─ LONG ───────┤
             ├─ SHORT ──┤
             └─ HICCUP ─┘

 ►─┬───────────────────────────────────┬──────────────────────────────────►
   └─ , ─┬─ ECB=ecb-pointer ───────────┤
         ├─ ECBID=ecb-id-register ───┤
         └─ ECBLIST=ecb-list-pointer ┘

 ►─┬────────────────────────┬────────────────────────────────────────────►
   └─ ,COND= ─┬─ NO ◄ ──────┤
              ├─ ALL ───┤
              ├─ INACT ─┤
              └─ DEAD ──┘

 ►─┬──────────────────────────┬──────────────────────────────────────────►
   └─ ,DEADXIT=deadlock-label ┘

 ►─┬───────────────────────────┬─────────────────────────────────────────►
   └─ ,INACTXIT=inactive-label ┘

 ►─┬──────────────────────┬──────────────────────────────────────────────►◄
   └─ ,ERROR=error-label ─┘
```

**Parameters**

**TYPE=**

Specifies whether the task is relinquishing control to await the completion of an event, or is giving other tasks the chance to be dispatched.

**LONG=/SHORT=**

TYPE=LONG and TYPE=SHORT have been obsolete since 10.2 releases of CA IDMS but are allowed to be specified for downward compatibility with existing client source code. Specifying TYPE=LONG or TYPE=SHORT will result in an assembler NOTE as follows:

"NOTE: TYPE=SHORT IS NO LONGER MEANINGFUL. IT WILL BE IGNORED."

"NOTE: TYPE=LONG IS NO LONGER MEANINGFUL. IT WILL BE IGNORED."

**HICCUP**

Relinquishes control to another ready-to-run task before being dispatched. HICCUP requests do not require an ECB.

**ECB=*ecb-pointer***

Defines the ECB for which the task will wait. *Ecb-pointer* is a register that points to the user-defined three-fullword field that contains the ECB or the symbolic name of the ECB field.

**ECBID=*ecb-id-register***

Specifies the 4-character ID of a previously defined ECB for which the task will wait. *Ecb-id-register* is a register that contains the ECB ID, the symbolic name of a fullword field that contains the ECB ID, or the ID literal enclosed in quotation marks.

**ECBLIST=*ecb-list-pointer***

Specifies that the wait is for more than one event. Each event in the list is represented by a pair of fullwords:

- The first fullword is a pointer to the ECB associated with the event
- The second fullword is zeros

Note: To identify the end of the list, the high-order bit of the last fullword in the parameter list must be turned **on**.

*Ecb-list-pointer* is a register that points to the list or the user-defined symbolic name of the fullword area containing the list of ECBs.

**COND=**

Specifies whether this #WAIT request is conditional and under what condition control should be returned to the issuing program.

**NO**

(Default); specifies that the request is not conditional.

**ALL**

Specifies that the request is conditional. Control is returned to the requesting program if the wait cannot be serviced for any reason.

**INACT**

Specifies that the request is conditional. Control is returned to the requesting program if the wait resulted in a task exceeded the STALL INTERVAL.

**DEAD**

Specifies that the request is conditional. Control is returned if waiting for the specified ECBs would cause a deadlock.

**DEADXIT=*deadlock-label***

Specifies the symbolic name of the routine to which control should be returned if waiting for the specified ECBs would cause a deadlock.

**INACTXIT=*inactive-label***

Specifies the symbolic name of the routine to which control should be returned if waiting for the specified ECBs would cause the task to surpass the STALL INTERVAL.

**ERROR=*error-label***

Specifies the symbolic name of the routine to which control is returned if a condition specified in the COND parameter occurs for which no other exit routine was coded. In this case, the ERROR parameter functions the same as DEADXIT.

**Example**

The #WAIT statement shown below passes control to the system while waiting for terminal input. Processing is suspended until the ECB for the task is posted, indicating that the terminal input operation is completed. If this #WAIT request would cause a deadlock, control is returned to the LOCKRTN9 routine.

```
#WAIT ECB=ECB_9,COND=DEAD,DEADXIT=LOCKRTN9
```

**Status Codes**

By default, the #WAIT request is unconditional; any runtime error results in an abend of the issuing task.

After completion of the #WAIT request, the value in register 15 indicates the outcome of the operation. The following is a list of Register 15 values and the corresponding meaning:

**X'00'**

The request has been serviced successfully.

**X'08'**

The request cannot be serviced because to wait for the specified ECBs would cause a deadlock.

**X'12'**

The request cannot be serviced because the task stalled waiting for the specified ECBs.

# #WTL

The #WTL (write to log) statement performs the following functions:

- Retrieves a predefined message from the message area of the dictionary
- Sends the message to selected destinations
- Optionally writes the message to a specified location in program storage

Messages are stored in the message area of the dictionary. Each message in the dictionary consists of the message text and the message destination. Typical destinations are the operator console and the DC/UCF log file. Messages are defined in the dictionary by using the IDD DDDL compiler.

Note: For more information about the IDD DDDL compiler, see the *IDD DDDL Reference*.

The message text can be dynamically changed by your program using symbolic parameters. You can also optionally request the system not to retrieve the message but to send only the message ID and symbolic parameter replacement values to the selected destinations.

The message ID specified in a #WTL statement is a 7-digit number. The first six digits contain the message identifier used to retrieve the message from the dictionary. The seventh digit is a severity code. When the program requests that the system retrieve the message from the dictionary (MSGDICT=YES), a predefined severity code is retrieved along with the message text.

When the dictionary lookup is bypassed (MSGDICT=NO), the system uses the severity code specified in the program. The severity level determines the action the system takes after the message is written to the log.

The dictionary severity may be overridden by using the OVRIDES parameter.

The possible severity codes and their resulting DC/UCF system responses are listed below:

| Severity code | DC/UCF system action |
|---|---|
| 0 | Returns control to the issuing program and continues processing |
| 1 | Snaps all task resources to the log and returns control to the issuing program |
| 2 | Snaps all system areas to the log and returns control to the issuing program |
| 3 | Snaps all task resources and abends the task with a task abend code of D002 |

| Severity code | DC/UCF system action |
|---|---|
| 4 | Snaps all system areas and abends the task with a task abend code of D002 |
| 5 | Abends the task with a task abend code of D002 |
| 6 | Undefined |
| 7 | Undefined |
| 8 | Snaps all system areas and abends the system with a system abend code of 3996 |
| 9 | Terminates the system with a system abend code of 3996 |

If a #WTL statement specifies a message ID that is not in the message dictionary, the system issues a prototype message with severity level 0. Messages should be defined in the message dictionary before they are issued by an executing program.

The message text can be dynamically altered by using symbolic parameters. Messages stored in the message dictionary can contain symbolic parameters, identified by an ampersand (&). followed by a 2-digit numeric identifier. Symbolic parameters can appear in any order in the message.

The #WTL statement can specify replacement values for one or more symbolic parameters by using the PARMS operand. The position of replacement values in the #WTL request must correspond exactly with the 2-digit numeric identifier in the message text. For example, the first value specified corresponds to &01., the second &02., and the third &03., as shown in the example below.

The stored message text reads:

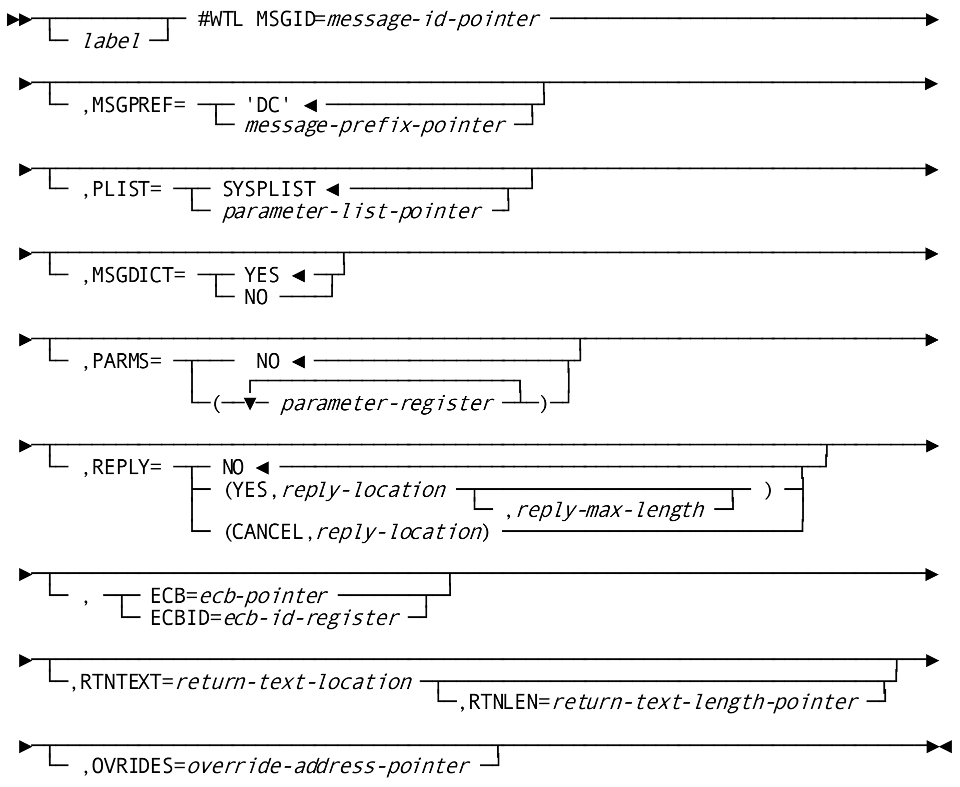
```
THIS IS TEXT &01. AND &03. OR &02.
```

The PARMS clause reads: PARMS=('A','B','C'). The resulting text would read:

```
THIS IS TEXT A AND C OR B
```

If the message destination is the operator console, the #WTL can optionally request a reply. An event control block (ECB) can be defined that will permit control to be returned immediately to the issuing task without waiting for the reply. The ECB will be posted by the system when the reply is sent. If no ECB is defined, control is not returned to the issuing task until the reply has been received.

**Syntax**



**Parameters**

**MSGID=*message-id***

Specifies the 7-digit message ID that is stored in the message dictionary. *Message-id* can be specified as follows:

- A register that points to the field containing the message ID
- The symbolic name of a user-defined message ID
- A message ID literal enclosed in quotation marks

A message ID must be a 4-byte packed decimal field (PL4), formatted as *nnnnnnS*, where *nnnnnn* is the 6-digit ID and *S* is the severity code. *Message-id* can specify any number in the range 900001 through 999999; id numbers 000001 through 900000 are reserved for use by the system.

**MSGPREF=DC/*message-prefix-pointer***

Specifies a 2-character alphanumeric prefix to the message ID. The default message prefix is 'DC'.

Note: It is important when using the MSGPREF option that you keep the message ID within the user range of 900001 through 999999. The system uses message prefixes which could cause a conflict with user message prefixes unless this restriction is observed.

***message-prefix-pointer***

A register that points to the prefix, the symbolic name of a user-defined field containing the prefix, or the prefix literal enclosed in quotation marks.

**PLIST=**

Specifies the area in which the system builds the #WTL parameter list.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system builds the #WTL parameter list.

***parameter-list-pointer***

A register that points to the area or the symbolic name of the area in which the system builds the #WTL parameter list.

If MSGID is the only operand specified on the #WTL request, you do not need to specify PLIST. If any additional operands are included, the following rules determine the size of the PLIST:

1 + P + X

where the following conditions are met:

- *P* is the number of parameters coded in the PARMS operand (described below).
- *X* is as follows:
  - At least 1 if either RTNTEXT or REPLY is specified
  - At least 3 if OVRIDES is specified
  - At least 4 if ECB or ECBID is specified
  - At least 5 if RTNLEN is specified

**MSGDICT=**

Specifies whether to retrieve the message from the message area of the dictionary.

**YES**

(Default); requests that the system locate the predefined message, apply substitution values, and send the message to the designated destinations.

**NO**

Requests that the system bypass the dictionary. The system writes a message to the console operator and log file that contains only the message ID and any replacement values specified in the PARMS parameter.

**OVRIDES=**

Override the default destination and/or severity code values.

***override-address-pointer***

A register that points to the address of the override values or the symbolic name of the field containing the override values.

Override values must be defined in the following manner:

| Bytes | Contents |
|---|---|
| 0 | X'80'—Destination is the DC log<br>X'40'—Destination is the console operator<br>X'20'—Destination is the terminal operator<br>X'10'—Destination is the ID of any terminal<br>X'08'—Override the severity with severity passed in message ID<br>X'01'—Null override |
| 1 - 2 | Overrides for MVS description in the format 00N0, where N is a valid MVS descriptor code. |
| 3 - 4 | Overrides for MVS route code in the format 00N0, where N is a valid MVS route code. |

**PARMS=**

Specifies replacement values for one or more symbolic parameters stored with the message text.

Note: If the text parameters contain any binary zeroes (x'00'),CA IDMS/DC automatically changes them to blanks (x'40') after copying the parameters to an internal work area.

**NO**

(Default); specifies that there are no symbolic parameters to be replaced, or requests that the system not replace any of the symbolic parameters.

***parameter-register***

Requests that the system replace the specified parameters. *Parameter-register* is a register that points to the replacement field, the symbolic name of a user-defined replacement field, or the replacement value literal enclosed in quotation marks.

When *parameter-register* is a register or user-defined field, each parameter field must begin with a 1-byte field from which the system obtains the length of the adjacent replacement field. The value in the length does not include the length byte.

**REPLY=**

Performs one of the following functions:

- Specifies that your program expects a reply to the message being sent

- Cancels a previously issued #WTL request for a reply to a message

The REPLY and RTNTEXT options are mutually exclusive; do not specify both options on a single #WTL request. The following options can be specified for the REPLY parameter:

**NO**

(Default); specifies that no reply is expected.

**(YES,*reply-location*,*reply-max-length*)**

Specifies that a reply is expected and should be returned to the area defined by *reply-location* and, optionally, *reply-max-length*.

***reply-location***

Specifies the location of the area reserved for a reply to the message issued by a #WTL request. *Reply-location* is either a register that points to the area or the symbolic name of that area.

***reply-max-length***

Specifies the maximum length, in bytes, of the area reserved for the reply. *Reply-max-length* is an absolute expression of the area length. If the maximum length is not specified by using the REPLY option, you must indicate the maximum length in the second halfword of the reply location.

Note: If **YES** is specified, the ECB or the ECBID parameters must be included to identify the ECB to be posted.

When the reply is sent, the reply area will be formatted by the system, as shown below:

| Bytes | Contents |
|---|---|
| 0 - 1 | Reserved for system use |
| 2 - 3 | Length of the reply text expressed as a halfword binary value. If the maximum reply length is not specified, you must set this maximum length before issuing the #WTL request. On completion of the #WTL request, this field will contain the actual length of the text. |
| 4 - n | Reply text |

**(CANCEL,*reply-location*)**

Cancels a request for a reply to a previously issued #WTL request. *Reply-location* specifies the area reserved for a reply to the message. *Reply-location* is either a register that points to the area or the symbolic name of the area.

**ECB=**

(#WTL requests with REPLY=YES only); identifies the ECB to be posted when the reply has been sent to its destination. Naming an ECB allows control to return immediately to the issuing task without waiting for a reply. The system will post the ECB when the reply is sent. If no ECB is defined, the system does not return control to the issuing task until the reply is received.

**ECB=**

Identifies the ECB that is posted when the reply is sent.

***ecb-pointer***

Either a register that points to the fullword ECB or the symbolic name of the ECB.

**ECBID=**

Identifies the 4-character symbolic ECB that is posted when the reply is sent.

***ecb-id-register***

Either a register that contains the ECB ID, the symbolic name of a fullword field that contains the ECB ID, or the ID literal enclosed in quotation marks.

**RTNTEXT=*return-text-location***

Specifies the location into which the system places the retrieved message text identified by *message-id*. Any replacement values specified in the PARMS parameter are included in the retrieved text.

If the length of the retrieved message text (RTNLEN) is not specified, the first byte of the return text receiving field *must* specify the length, in hexadecimal notation, of the returned string.

***return-text-location***

Either a register that points to the storage area reserved for the message text or the symbolic name of a user-defined field reserved for the message text.

Note: The RTNTEXT and REPLY options are mutually exclusive; only one of these operands can be specified in a single #WTL request.

**RTNLEN=**

Indicates the length of the return text receiving field.

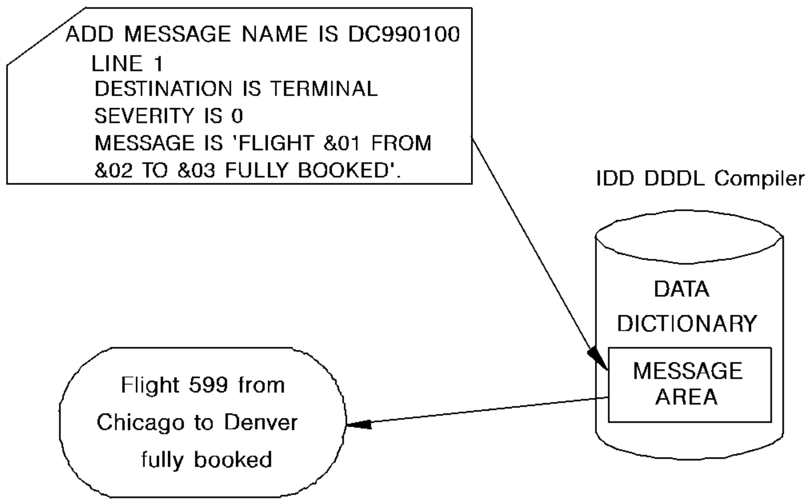
***return-text-length-pointer***

A register that points to the length of the field, a halfword or fullword field containing the length of the field, or an absolute expression of the length of the field enclosed in quotation marks.

If this parameter is included, the first byte of the RTNTEXT receiving field does not have to be a length indicator. If the length specified is not large enough to accommodate the entire message, register 1 will contain the number of lines that could not be sent.

### Example

The following figure illustrates a #WTL statement that supplies three replacement parameters and requests a reply. Program A issues a #WTL request for message 990100 with a prefix DC. The message text and severity are stored in the message area of the dictionary. Symbolic parameters are within the message text. The program specifies values to replace the symbolic parameters &01., &02., and &03. stored in the message area of the dictionary along with the message text. The system sends the message to terminal A, which is the logical terminal associated with the issuing task, and waits for a reply. The reply is returned to the area specified by REPLY; the length of the reply can be up to 20 bytes.



```
          #WTL MSGID=MFIELD,PLIST=WKPLIST,PARMS=(FLIGHT,SOURCE.DEST)
MFIELD          DC PL4'9901000
WKPLIST         DC
FLIGHT          DC
SOURCE          DC
DEST            DC
```

### Status Codes

The system returns the following values to register 15 during processing of a #WTL request. Any value greater than zero indicates that the request was not serviced, and no #WTL was performed. Register 15 values are as follows:

**X'00'**

The request has been serviced successfully.

**X'04'**

An invalid parameter or combination of parameters has been specified.

**X'08'**

A resource necessary for the processing of the request, for example, a resource control element, is not available.

**X'0C'**

The maximum number of outstanding replies was exceeded.

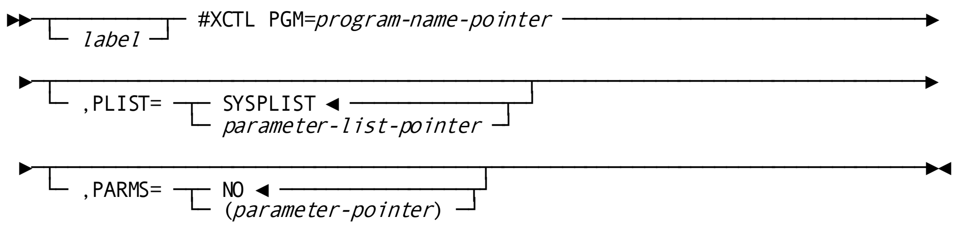
**X'10'**

The length of the return text area is not large enough to contain the entire message text.

## #XCTL

The #XCTL statement transfers control and sends an optional parameter list to a specified program. Control does not return to the issuing program when the specified program ends.

### Syntax



### Parameters

**PGM=**

Specifies the 1- to 8-character name of the program to which control is transferred.

***program-name***

A register that points to a field that contains the program name, the symbolic name of a user-defined field that contains the program name, or the program-name literal enclosed in quotation marks.

**PLIST=**

Specifies the location of the storage area that contains one or more parameters to be passed to the program receiving control.

**SYSPLIST**

(Default); is the symbolic name of the storage area in which the system builds the parameter list.

***parameter-list-pointer***

Either a register that points to the area in which the system builds the list or the symbolic name of the area.

The size of the parameter-list area is equal to two fullwords plus one fullword for each parameter listed. Thus, if no parameters are specified (PARMS=NO), the length of the storage area is two fullwords; if one parameter is specified, the length is three fullwords.

**PARMS=**

Specifies whether parameters will be passed to the program receiving control.

**NO**

(Default); specifies that no parameters will be passed to the program.

***parameter-pointer***

Specifies that parameters will be passed to the program. *Parameter-register* is either a register that contains the address of the parameter or the symbolic name of a user-defined field that contains the parameter.

### Example

The #XCTL statement shown below transfers control to the Cloud Airlines flight booking program and passes parameters that specify the flight, the city of departure, and the flight destination.

```
#XCTL PGM='CLBOOK',PARMS=(FLT,DEPART,DEST)
```

### Status Codes

By default, the #XCTL request is unconditional. Error conditions that can occur are described below:

- A no-space-available-in-program-pool condition is caused when there is not enough storage in the program pool to accommodate the program. The system delays processing until sufficient storage becomes available. If such a wait would cause a deadlock, the system aborts the program.
- A nonconcurrent-program-in-use condition is caused when a copy of the program is already in use and is marked as nonconcurrent (indicating that this program can be used by one task at a time). The system delays processing until the program becomes available.
- A storage-conflict condition is caused when a copy of the previously loaded program is temporarily overlayed while being used by a waiting task. The system delays processing until the program is replaced in the program pool.

- Any abnormal condition causes the system to terminate the program abnormally. Conditions in this category include:
  - An I/O error
  - A program not found in the PDT (program definition table) or marked out-of-service
  - A wait-on-storage (default action resulting from the no-space-available-in-program-pool condition) would result in a deadlock

## Logical Record Clauses

Logical record clauses are used with any of the four DML statements that access logical records: @OBTAIN, @MODIFY, @STORE, and @ERASE. The logical record clauses are as follows:

- **WHERE** specifies criteria used to select logical-record occurrences or to limit the selection of logical-record occurrences
- **ON** tests for a specific path status returned to indicate the result of a logical-record DML statement

The WHERE and ON clauses are explained in this section.

### WHERE Clause

#### Functions of the WHERE Clause

The WHERE clause has two major functions:

- **To direct the program to a predefined path** in the subschema. The path is defined by the DBA and is transparent to the application program. Predefined paths allow the program to access database records without issuing specific instructions for navigating the database.
- **To specify selection criteria to be applied to a logical record**. Selection criteria allow the program to specify attributes of the desired logical record, reducing the need for the program to inspect multiple logical record occurrences.

#### Two Elements in a WHERE Clause

The WHERE clause is constructed from two elements:

- A positional parameter that contains the key value WHERE
- An Assembler remark that encodes a Boolean expression that consists of comparisons and keywords connected by Boolean operators (AND, OR, and NOT)

An Assembler logical record DML statement that contains a WHERE clause consists of an Assembler macro parameter concatenated with a compiler-level expression. The remark is resolved by the DML precompiler, not by the assembler. Therefore, programs that contain logical record DML statements using WHERE clauses *must* be submitted to the DML precompiler before assembly.

**Coding WHERE**

Because the Boolean expression is treated as an Assembler remark, it can be written in a more readable form than conventional Assembler statements. WHERE clauses can span several lines in an Assembler program. The keyword WHERE must begin in column 16, continuation lines must be in column 16 or greater, and are marked by coding a nonblank character in column 72. Descriptive comments cannot be on the same line as the WHERE clause.

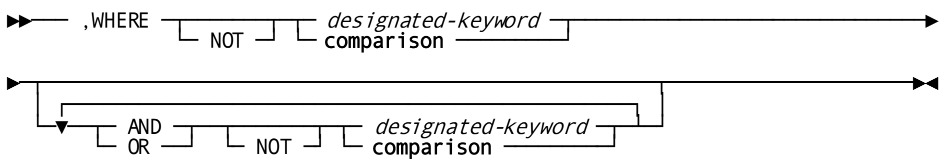
**Including Boolean Operators**

Individual comparisons and keywords must be connected by the Boolean operators AND, OR, and NOT. Parentheses can be used to clarify a multiple-comparison Boolean expression or to override preceding operators.

Operators in a WHERE clause are evaluated in the following order:

1. Comparisons enclosed in parentheses, in order of precedence within parentheses
2. Arithmetic, comparison, and Boolean operators in order of precedence, from highest to lowest:
   a. Unary plus or minus in an arithmetic expression
   b. Multiplication or division in an arithmetic expression
   c. Addition or subtraction in an arithmetic expression
   d. MATCHES or CONTAINS comparison operators
   e. EQ, NE, GT, LT, GE, LE comparison operators
   f. NOT Boolean operator
   g. AND Boolean operator
   h. OR Boolean operator
3. From left to right within operators of equal precedence

**Syntax**



**Expansion of comparison**

```
►►─┬─ literal ──────────────────────────────────────┬────────────────────────────►
   ├─ idd-defined-variable-field-name ──────────────┤
   ├─ logical-record-field-name ─┬──────────────┬───┤
   │                             └─ OF LR ──────┘   │
   └─ arithmetic-expression ────────────────────────┘

 ►─┬─── CONTAINS ─┬──────────────────────────────────────────────────────────────►
   ├─── MATCHES ──┤
   ├─┬─ EQ ─┬─────┤
   │ └─ = ──┘     │
   ├─── NE ───────┤
   ├─┬─ GT ─┬─────┤
   │ └─ > ──┘     │
   ├─┬─ LT ─┬─────┤
   │ └─ < ──┘     │
   ├─── GE ───────┤
   └─── LE ───────┘

 ►─┬─ literal ──────────────────────────────────────┬───────────────────────────►◄
   ├─ idd-defined-variable-field-name ──────────────┤
   ├─ logical-record-field-name ─┬──────────────┬───┤
   │                             └─ OF LR ──────┘   │
   └─ arithmetic-expression ────────────────────────┘
```

**Parameters**

***dba-designated-keyword*/comparison**

Specify selection criteria to be applied to the logical record.

***dba-designated-keyword***

Specifies a keyword that applies to the named logical record. The DBA has previously associated this keyword with the named logical record; the keyword routes the logical-record request to the appropriate predetermined path in the subschema. *Dba-designated-keyword* can be no longer than 32 characters.

Note: A path must exist to service a request that includes *dba-designated-keyword*. If no such path exists, the DML precompiler issues an error message.

**comparison**

Specifies the comparison operation to be performed, using the indicated operands and operators. *Comparison* also may direct the logical record request to a path in the subschema.

Syntax for *comparison* contains individual comparisons and keywords that are connected by the Boolean operators AND, OR, and NOT. Parentheses can be used to clarify a multiple-comparison Boolean expression or to override the precedence of operators.

***literal/idd-defined-variable-field-name/ logical-record-field-name/arithmetic-expression***

Identifies a left or right comparison operand.

***literal***

Specifies an alphanumeric or numeric literal. Alphanumeric literals must be enclosed in site-standard quotation marks.

***dd-defined-variable-field-name***

Specifies a program variable storage field predefined in the dictionary.

***logical-record-field-name***

Specifies a data field that participates in the named logical record. *Logical-record-field-name* uniquely identifies the named logical-record field.

The optional **OF LR** entry specifies that the value of the named field at the time the request is issued will be used throughout request processing. If the value of the field changes during request processing, LRF will continue to use the original value. If the OF LR entry is not included and the value of the field changes during request processing, the new field value in variable storage will be used.

***arithmetic-expression***

Specifies an arithmetic expression designated as a unary minus (-), unary plus (+), simple arithmetic operation, or compound arithmetic operation. Arithmetic operators permitted in an arithmetic expression are plus (+), minus (-), an asterisk (*), and a slash (/). These arithmetic operators must have a blank on either side. Operands can be the literals, variable fields, or the logical-record fields described above.

**CONTAINS/MATCHES/EQ/NE/GT/LT/GE/LE**

Specifies the comparison operator.

**CONTAINS**

Is true if the value of the right operand occurs in the value of the left operand. Both operands included with the CONTAINS parameter must be alphanumeric values.

**MATCHES**

Is true if each character in the left operand matches a corresponding character in the right operand (the mask). LRF compares the left operand with the mask, one character at a time, moving from left to right.

The result of the match is either true or false:

- The result is **true** if LRF reaches the end of the mask before encountering a character in the left operand that does not match a corresponding mask character.
- The result is **false** if LRF encounters a character in the left operand that does not match a mask character.

Three special characters can be used in the mask to perform pattern matching:

- **@** matches any alphabetic character
- **#** matches any numeric character
- ***** matches any alphabetic or numeric character

Both the left operand and the mask must be alphanumeric values.

**EQ**

Is true if the value of the left operand is equal to the value of the right operand.

**NE**

Is true if the value of the left operand is not equal to the value of the right operand.

**GT**

Is true if the value of the left operand is greater than the value of the right operand.

**LT**

Is true if the value of the left operand is less than the value of the right operand.

**GE**

Is true if the value of the left operand is greater than or equal to the value of the right operand.

**LE**

Is true if the value of the left operand is less than or equal to the value of the right operand.

The WHERE clause can contain as many comparisons and keywords as are required to specify the criteria you want. Processing efficiency is not affected by the composition of the WHERE clause (other than the logical order of the operators), since LRF automatically uses the most efficient path to process the logical-record request.

If necessary, the value of the SIZE parameter on the @COPY IDMS,SUBSCHEMA-LR-CTRL, @SSLRCTL, and @BIND SUBSCH statements can be increased to accommodate very large and complex WHERE clause specifications. For the algorithm to calculate *lrc-block-size,* see @COPY IDMS (see page 411).

**Examples**

The WHERE clause shown below uses Boolean selection criteria to obtain the requested EMPJOBLR occurrence. This statement retrieves any customer in Massachusetts who has an outstanding balance greater than $1500, or who has an outstanding balance less than $500 and has a questionable credit rating.

```
@OBTAIN EMPJOBLR WHERE MASSACHUSETTS AND ((UNITS * PRICE) -  *
      PAYMENT GT 1500 OR ((UNITS * PRICE) -                  *
      PAYMENT GT 500 AND (CREDRATE                           *
      EQ 'REF' OR CREDRATE EQ 'REJ')))
```

## ON Clause

The ON clause tests for a specific path status returned to indicate the result of a logical record request. If LRF returns the specified path status, the imperative statement included in the ON clause is executed. The imperative statement usually consists of a GOTO statement. If the path status is not returned, the imperative statement included in the ON clause is ignored.

Note: Only one ON clause can be coded per logical record DML statement; only one specific path status can be tested for.

**Standard Path Statuses**

Path statuses are issued during execution of the path selected to service the request. The following standard path statuses can be returned:

- **LR-FOUND** is returned when the logical-record request has executed successfully. LR-FOUND can be returned as the result of:
  - Any @OBTAIN LRF statement
  - Any of the other LRF statements containing a WHERE clause

  When LR-FOUND is returned, the ERRSTAT field of the IDMS communications block contains 0000.
- **LR-NOT-FOUND** is returned when the specified logical record cannot be found, either because no such record exists or because all such occurrences have already been retrieved. LR-NOT-FOUND can be returned as the result of any of the four LRF DML statements, provided that the path to which LRF is directed includes retrieval logic. When LR-NOT-FOUND is returned, the ERRSTAT field of the IDMS communications block contains 0000.
- **LR-ERROR** is returned when a logical record request is issued incorrectly or when an error occurs in the processing of the path selected to service the request. When LR-ERROR is returned, the type of status code returned to the program in the ERRSTAT field of the IDMS communications block differs according to the type of error:
  - When the error occurs in the **logical-record request**, the ERRSTAT field contains a status code issued by LRF (with a major code of 20). For a list of these codes, see Logical-Record Status Codes (see page 395).
  - When an error occurs in **logical-record path processing,** the ERRSTAT field contains a status code issued by the DBMS (with a major code from 00 to 19). For a list of these codes, see ERRSTAT Field and Codes (see page 41).

When accessing ASF-defined data tables, you should always check for all of the following path statuses:

- **INVALID-DATA** is returned when the data violates the definition-time selection criteria. For example, INVALID-DATA is returned when the selection criteria is WHERE STATE = 'MA' and the program tries to replace the state with 'NY'. When INVALID-DATA is returned, the ERRSTAT field in the IDMS communications block is set to 0000.
- **DEFN-MISSING** is returned when the record definition cannot be found. When DEFN-MISSING is returned, the ERRSTAT field in the IDMS communications block is set to 0000.
- **OOAK-MISSING** is returned when a one-of-a-kind record cannot be found. When OOAK-MISSING is returned, the ERRSTAT field in the IDMS communications block is set to 0000.
- **SYNC-ERROR** is returned when the time stamp in the catalog and the table definition do not match. When SYNC-ERROR is returned, the ERRSTAT field in the IDMS communications block is set to 0000.

The return of one or more of these path statuses indicates a fatal error. For more information, consult your DBA.

**Syntax**

```
►─┬───────────────────────────────────────────────┬──────────────────────────►
  └─ ,ONLRSTS=path-status,GOTO=branch-location ─┘
```

**Parameters**

**ONLRSTS=*path-status***

Tests for a path status returned as the result of the logical-record request issued by the program. *Path-status* must be a quoted literal (1 to 16 bytes under z/OS or 1 to 6 bytes under z/VSE) or a program variable.

Note: In addition to testing for a specific path status (using ONLRSTS), your program should check for standard path statuses (for example, LR-NOT-FOUND and LR-ERROR, and path statuses for ASF defined tables if applicable) whenever the program issues a logical record request.

**GOTO=*branch-location***

Specifies the program action to be taken if the indicated path status results from the logical-record request.

**Example**

The following ON clause causes the program to branch to the NOFFICE label when the path status specified in the variable NOOFF is met. NOOFF indicates a path status indicating that there are no offices that meet the criteria specified in the WHERE clause. Standard LRF path statuses are checked as well.

```
@OBTAIN REC=EMPJOBLR,                                              *
      ONLRSTS=NOOFF,GOTO=NOFFICE,                                  *
      WHERE OFFICE-CODE-0450 EQ '0980'
CLC   LRSTAT,=CL16'LR-FOUND'
BE    CRDITREF
CLC   LRSTAT,=CL16'LR-ERROR'
BE    LRERRTN
CLC   LRSTAT,=CL16'LR-NOT-FOUND'
BE    LRNTFND
```

## Logical-Record Status Codes

A path status of LR-ERROR signifies an error in the processing of a logical-record request. When the error occurs in the request itself, LRF returns a path of LR-ERROR to the LR-STATUS field of the logical-record request control (LRC) block and places one of the following codes in the ERRSTAT field of the IDMS communications block:

**2008**

The named logical record is not defined in the subschema, or the specified DML verb is not permitted with the named logical record. The logical record name may have been misspelled.

**2010**

The subschema prohibits access to logical records.

**2018**

A path command has attempted to access a database record that has not been bound.

**2040**

The WHERE clause in an @OBTAIN NEXT statement has directed LRF to a different processing path than did the WHERE clause in the preceding @OBTAIN statement for the same logical record. Either the WHERE clause is incorrect or an @OBTAIN FIRST should have been issued instead of @OBTAIN NEXT.

**2041**

LRF was unable to match the request's WHERE clause to a path in the subschema.

**2042**

The logical-record path for the request specifies return of the LR-ERROR path status to the program.

**2043**

Bad or inconsistent data was encountered in the logical-record buffer during evaluation of the request's WHERE clause:

- A WHERE clause has specified that a packed decimal field should be compared to a field that is not packed; the field that is not packed cannot be converted to packed because it contains nonnumeric data.
- Data in variable storage or in a database record does not conform to its description.

A path status of LR-ERROR is returned to the program unless the DBA has included an ON clause in the path to override this action.

**2044**

The request's WHERE clause does not include information required by the logical-record path.

**2045**

A subscript value in a WHERE clause is either less than 0 or greater than its maximum allowed value. A path status of LR-ERROR is returned to the program unless the DBA has included an ON clause in the path to override this action.

**2046**

A program check has been issued during evaluation of a WHERE clause for one of the following reasons:

- An arithmetic overflow would occur (fixed point, decimal, or exponent).
- An arithmetic underflow would occur (exponent).
- A divide exception would occur (fixed point, decimal, or floating point).
- A significance exception has occurred.

A path status of LR-ERROR is returned to the program unless the DBA has included an ON clause in the path to override this action.

**2063**

A request's WHERE clause contains a keyword that exceeds 32 characters.

**2064**

A path command has attempted to access a CALC data item that has not been defined properly in the subschema.

**2072**

LRF cannot acquire sufficient storage to evaluate the request.

These status codes can result from any of the logical-record DML statements with the exception of 2040, which applies to @OBTAIN NEXT only.

# Chapter 6: Assembler DML Coding Considerations

This chapter describes how to code Assembler DML statements. The following topics are discussed:

- Coding user-supplied operands
- Coding DML statement parameters
- Synonym processing

Logical Record Facility keywords

This section contains the following topics:

## Coding User-Supplied Operands

User-supplied operands in DML statements can be specified by name, in register notation, or in data field notation.

**By Name**

Record, set, or area names can be specified explicitly by name. Unless QUOTES=NO has been specified in the @MODE statement, the name must be enclosed in quotation marks; for example:

```
SUBSCH='DEMOSUBS'
```

The DML precompiler performs validity checking for explicitly specified names.

**Note: z/VSE USERS**—A quoted name operand in a **logical-record** DML statement cannot exceed 6 characters. A program variable can be used for a path status that exceeds 6 characters. An exception is a quoted operand in a WHERE clause, which can be up to 32 characters long.

**Note: ASSEMBLER G USERS**—A quoted name operand in a **logical record** DML statement cannot exceed 6 characters unless the maximum variable size is modified by the appropriate Assembler PARM. A maximum variable size of at least 18 characters is recommended. An exception is a quoted operand in a WHERE clause, which can be up to 32 characters long.

**Note: ASSEMBLER H USERS**—The DML precompiler (IDMSDMLA) supports 32-character names and converts hyphens to underscores.

### In Register Notation

A register can contain either the variable value or the variable address. The general register symbol or register reference must be enclosed in parentheses; for example:

```
#FREESTG STGID=(7)
```

The DML precompiler does not perform validity checking of operands specified by register notation.

**Note: z/VSE USERS**—A general register symbol or register reference in a logical record DML statement cannot exceed 6 characters.

**Note: ASSEMBLER G USERS**—A general register symbol or register reference in a logical record DML statement cannot exceed 6 characters, unless the maximum variable size is modified by the appropriate Assembler PARM. A maximum variable size of at least 18 characters is recommended.

### In Data Field Notation

Your program can specify the name of a variable field containing the desired data name; for example:

```
@OBTAIN CURRENT,REC=RECFLD
```

The DML precompiler does not perform validity checking of operands specified by data field notation.

## Coding Parameters

**Types of Parameters**

There are two types of parameters in DML statements:

- **Positional parameters**—Positional parameters appear in specific relative locations; for example:

  ```
  #GETSTG TYPE=(USER,LONG,KEEP)
  ```

- **Keyword parameters**—Keyword parameters are constructed from:

  1. **A keyword**—A character string that is predefined to the system
  2. **An equal sign (=)**
  3. **A variable-value parameter**—Containing one or more variable values

  For example:

  ```
  @OBTAIN NEXT,SET='CUSTOMER-ORDER',REC='ORDER'
  ```

  CA IDMS keywords are listed in Logical Record Facility Keywords (see page 403) later in this chapter.

**Coding Considerations**

The following considerations apply to coding DML parameters:

- All DML statements except for logical-record DML statements use keyword parameter notation. The DML precompiler generates positional-pair parameter notation.
- Logical-record DML statements that bypass the DML precompiler must be coded using positional-pair parameter notation. The assembler misinterprets or rejects logical-record DML statements that contain keyword parameters.
- Logical-record DML statements that are processed by the DML precompiler can be coded using either keyword parameter or positional-pair parameter notation.

## Synonym Processing

CA IDMS/DB allows alternative identification of records and elements in the dictionary. Synonyms are added to the dictionary by using DDDL statements. The DML precompiler automatically copies these language dependent synonyms in place of the primary names whenever an @COPY IDMS statement appears in the application program.

**Note: ASSEMBLER H USERS**—The DML precompiler supports 32-character field names and conversion of hyphens to underscores, in accordance with the new features of Assembler H. CA IDMS/DB record names remain restricted to 16 characters and CA IDMS/DB element names to 32 characters. Synonyms are therefore not required for user supplied names and for fields containing hyphens in Assembler H programs using the DML precompiler.

IDD record names can be up to 16 characters long, and IDD element names can be up to 32 characters long. Because Assembler versions F and G restrict names to 8 characters, alternative and unique 8 character names for use in Assembler F and Assembler G programs should be defined in the dictionary. Use of synonyms is recommended if @COPY IDMS and @INVOKE statements are to be included in Assembler programs.

Synonyms cannot be defined for logical record names. Assembler programs that access logical records must use a separate subschema in which logical records are defined according to Assembler restrictions.

### How the Precompiler Copies Synonyms

When the DML precompiler copies record descriptions from the dictionary into program variable storage, it copies synonyms according to the following rules:

- If a record is defined for the program's language, but the primary record name is not, the synonym is copied into the program.
- If more than one synonym for a given record is defined for Assembler, the first one found in the dictionary is copied.
- If the primary record name is defined for Assembler, the primary name is copied into the program.

For example, assume that the following record is defined in the dictionary with three synonyms:

```
RECORD JOB
RECORD NAME SYNONYM JOBSYN1 LANGUAGE ASSEMBLER
RECORD NAME SYNONYM JOB-SYN2
RECORD NAME SYNONYM JOBSYN3 LANGUAGE ASSEMBLER
```

Since the dictionary defines JOBSYN1 as the first synonym for Assembler, the DML precompiler copies it into the program. The DML precompiler would copy the primary record name (JOB) if it were defined for Assembler.

These rules apply regardless of the record name or synonym that appears in the schema and subschema invoked by the program.

**Synonyms are Recognized as Primary Records**

The DML precompiler treats a synonym as if it were the primary record. The expansion of a DML statement will include the record name of the primary record name, even if the synonym is copied into program variable storage.

For example, an @COPY IDMS,SUBSCHEMA-BINDS statement used in an Assembler program generates the following @BIND REC statement for the employee record:

```
@BIND REC='EMPLOYEE',IOAREA=EMPLOYE
```

This statement lists both the primary record name (EMPLOYEE) and the Assembler synonym (EMPLOYE).

Note: For more information about synonym facilities, see the *IDD DDDL Reference Guide*.

## Logical Record Facility Keywords

The following is a list of LRF keywords recognized by the Assembler DML precompiler. These keywords should not be used as labels in Assembler DML programs that use the Logical Record Facility:

- FIRST
- GOTO
- LR
- LRSTAT
- NEXT
- ONLRSTS
- REC
- WHERE

# Chapter 7: DML Precompiler-Directive Statements

This chapter presents syntax for precompiler-directive statements.

**Function of Precompiler Directives**

To use DML statements that request CA IDMS/DB and DC/UCF services, you must include precompiler-directive statements in your application program. Precompiler-directive statements:

- Ensure that the assembler performs the proper expansion of DML statements into calling sequences appropriate to the CA IDMS environment
- Identify the dictionary resources (subschema and/or maps) required by the program
- Cause predefined source modules to be copied into the program from the dictionary
- Generate source data description code

**Summary of Statements**

The DML precompiler-directive statements are summarized below:

- **@MODE** initializes all global SET symbols that control the expansion of subsequent macros and DML commands into calling sequences appropriate to the CA IDMS/DB environment. You must code the @MODE directive before all procedural statements in the program, including DML commands for CA IDMS/DB and DC/UCF requests.
- **@INVOKE** identifies all dictionary resources used by the application program. The @INVOKE statement must precede all procedural statements in the program, including DML commands for CA IDMS/DB and DC/UCF requests. This statement will generate non-executable source code when the MAP= operand is used for a map with multiple occurring fields.
- **@COPY IDMS** copies the source data description code associated with CA IDMS/DB database records, the IDMS communications block, map records, and the map request block, as well as other predefined source modules and records, into the program from the dictionary at the location of the @COPY IDMS statement.

- **#MRB** establishes a map request block (MRB), which is required for the mapping mode of terminal I/O operations. The MRB is a variable storage area in the application program and is used for communications between the program and the mapping compiler during a mapping I/O request.
- **#MAPBIND** initializes the MRB for mapping requests issued by the application program. #MAPBIND generates executable code.
- **@SSCTRL** generates the source data description code associated with the IDMS communications block in the program.
- **@SSLRCTL** generates the source data description code associated with the LRC block in the program.

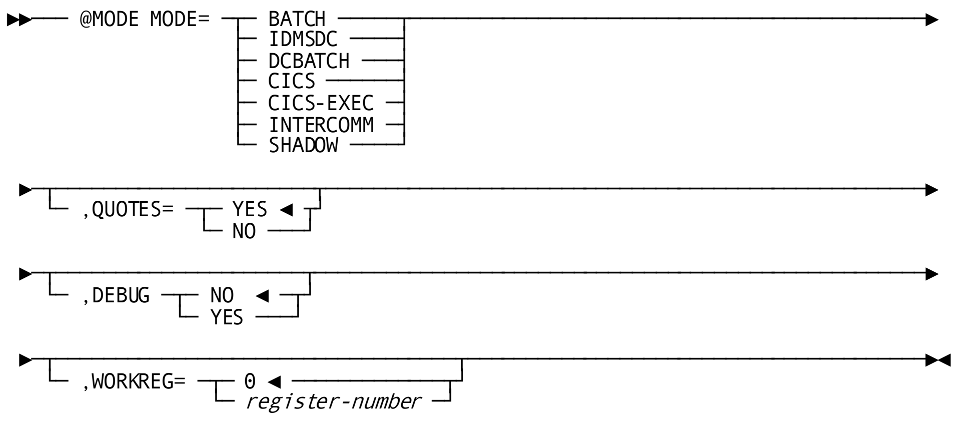
This section contains the following topics:

## @MODE—initializes global SET symbols

The @MODE statement initializes global SET symbols for the assembler; these symbols control the generation of macros associated with CA IDMS/DB requests. You must specify the operating mode for programs that access a CA IDMS/DB database. If you do not code an @MODE statement, you can specify the CA IDMS/DB environment by using the MODE parameter of the @INVOKE statement, described later in this chapter. For CA IDMS programs that do not require access to a CA IDMS/DB database, the function of the @MODE statement is to indicate the operating mode: batch or online. An online mode selection is made from one of the valid teleprocessing monitors.

The @MODE and the @INVOKE statement must precede all other DML statements in the program. Either statement can be placed before the other.

## @Mode Syntax



## @MODE Parameters

**MODE=**

Defines the operating environment for which the calling sequence will be generated. If the @MODE statement is not used, the CA IDMS/DB environment must be specified in the @INVOKE statement, which is discussed below.

**BATCH**

(Default); specifies to execute the program in batch mode. The IDMS communications block is copied into variable storage; standard CALL statements are generated.

**IDMSDC**

Specifies to execute the program in IDMS DC mode. The IDMS DC communications block is copied into variable storage; CA IDMS/DC CALL statements are generated for CA IDMS/DC requests.

**DCBATCH**

Specifies to execute the program in DC-BATCH mode. The IDMS DC communications block is copied into variable storage; DC-BATCH CALL statements are generated for CA IDMS/DC requests. Specify MODE=DCBATCH to access DC queues and printers from batch applications running under the CA IDMS central version.

**CICS/CICS-EXEC/INTERCOMM/SHADOW**

Specifies to execute the program in a special environment under the specified teleprocessing monitor. The appropriate communications block is copied into variable storage and operating-mode-specific CALL sequences are generated.

**QUOTES=**

Required for programs that access the CA IDMS/DB database; indicates whether names (such as record name or area name) coded in DML statements must be enclosed in site-standard quotation marks.

**YES**

(Default); specifies to enclose names specified in CA IDMS/DB database requests in site-standard quotation marks.

**NO**

Specifies to not enclose names specified in CA IDMS/DB database requests in site-standard quotation marks.

**DEBUG=**

Required for programs that access the CA IDMS/DB database; requests the DML precompiler to save sequence numbers associated with DML statements in the IDMS communications block, as follows:

**NO**

(Default); specifies not to save sequence numbers of DML statements.

**YES**

Generates the appropriate code for saving sequence numbers associated with DML statements. At runtime, the sequence number of each DML statement is moved to the IDMS communications block before program execution. These sequence numbers appear in the Assembler source statement listing in the form DML-SEQUENCE=*n*. Depending on the error routine defined by the DBA, the DML sequence number can be reported when errors occur and can be used to assist you in debugging your Assembler program.

Note: This option does not apply to DC/UCF requests. Statement numbers associated with DC/UCF requests cannot be saved because the system does not use the IDMS communications block.

**WORKREG=0/**

Required for programs that access the CA IDMS/DB database; specifies the general purpose register to be used for constructing the IDMS parameter list for calls to IDMS.

***register***

An integer in the range 0 through 15, or any valid symbolic or defining term for the general-purpose register (for example, R0). The default is general register 0.

# @INVOKE

The @INVOKE statement performs the following functions:

- Specifies the subschema and maps required by the program
- Defines the operating mode if not previously defined by an @MODE statement
- Identifies the program if program registration has been implemented
- Identifies the program for use during statistics collection

The @INVOKE statement and the @MODE statement must precede all other precompiler-directive and DML statements in the program. @INVOKE must be included if the DML precompiler will be used and if the program requests CA IDMS/DB services.

**Syntax**

```
►►─── @INVOKE ──┬──────────────────────────────────────────────────┬──────────►
                └─ PROGRAM=program-name ──┬──────────────────────────┬┘
                                          └─ ,VERSION=version-number ┘

►─┬────────────────────────────┬──────────────────────────────────────────────►
  └─ ,SUBSCH=subschema-name ───┘

►─┬────────────────────────────────────────────────────────────┬──────────────►
  └─,SCHEMA=schema-name ──┬──────────────────────────────┬─────┘
                          └─ ,VERSION=version-number ────┘

►─┬──────────────────────────────┬────────────────────────────────────────────►
  └─ ,MODE= ──┬── BATCH ─────────┤
              ├── IDMSDC ────────┤
              ├── DCBATCH ───────┤
              ├── CICS ──────────┤
              ├── CICS-EXEC ─────┤
              ├── INTERCOMM ─────┤
              └── SHADOW ────────┘

►─┬──────────────────────────────────────────────────────┬────────────────────►
  └─,MAP=map-name ──┬──────────────────────────────┬─────┘
                    └─ ,VERSION=version-number ────┘

►─┬──────────────────────────────────────┬────────────────────────────────────►
  └─ ,MRBTYPE= ──┬── STANDARD ◄ ──┬──────┘
                 └── EXTENDED ────┘

►─┬───────────────────────────────┬──────────────────────────────────────────►◄
  └─ ,PAGING = ──┬── NO ◄ ──┬─────┘
                 └── YES ───┘
```

**Parameters**

**PROGRAM=*program-name***

Required if program registration is in effect; specifies the 1- to 8-character name of the registered program. If in effect, subschema authorization specifies that programs must be registered with the named subschema in order to be compiled against it.

If the program has been previously defined in the dictionary using IDD, *program-name* must match the assigned name of the program; otherwise the DML precompiler will not recognize it as the same program.

**Version=*version-number***

Optional; indicates the version number of the program to distinguish multiple versions of the same program-name. *Version* is a numeric literal in the range 1 through 9999. If the version number is not specified, and *program-name* is found in the dictionary, the version number defaults to the highest value defined in the dictionary for the program. If *program-name* is unknown to the data dictionary, the version number defaults to 1.

**SUBSCH=**

Identifies the subschema to be used by the program.

***subschema-name***

Specifies a subschema defined in the dictionary.

**SCHEMA=*schema-name***

Identifies the schema with which the subschema is associated.

**Version=*version-number***

Optionally specifies the version of the schema as defined in the dictionary. It defaults to the highest version of the named schema.

**MODE=**

Defines the operating mode for the program. This clause is optional; it can replace the @MODE statement if @COPY is the only additional DML statement in use, but should be omitted in all other cases.

**BATCH**

Specifies to execute the program in batch mode. The IDMS communications block is copied into variable storage; standard CALL statements are generated.

**IDMSDC**

Specifies to execute the program in IDMS DC mode. The IDMS DC communications block is copied into variable storage; CA IDMS/DC CALL statements are generated for CA IDMS/DC requests.

**DCBATCH**

Specifies to execute the program in DC-BATCH mode. The IDMS DC communications block is copied into variable storage; DC-BATCH CALL statements are generated for CA IDMS/DC requests. Specify MODE=DCBATCH to access DC queues and printers from batch applications running under the CA IDMS central version.

**CICS/CICS-EXEC/INTERCOMM/SHADOW**

Specifies to execute the program in a special environment under the specified teleprocessing monitor. The appropriate communications block is copied into variable storage and operating-mode-specific CALL sequences are generated.

**MAP=**

Specifies that mapping mode terminal I/O is required by the program and identifies the maps stored in the dictionary. Multiple maps can be specified in a single @INVOKE statement by defining a separate MAP clause for each map.

***map-name***

Specifies the 1- to 8-character name of a map defined in the dictionary.

**Version=*version-number***

Optionally specifies the version of the map being used. It defaults to the highest version of the named schema.

**MRBTYPE=STANDARD/EXTEN DED**

Specifies the format of the map request block (MRB) built for the map:

- **STANDARD** (default) specifies that the map has standard 3270-type terminal attributes.
- **EXTENDED** specifies that the map has extended 3279-type terminal attributes, such as color, blinking fields, and reverse video.

**PAGING=NO/YES**

Specifies whether the program uses pageable maps. A pageable map is a single map that is associated with an unlimited number of map fields. You can use pageable maps when all the map fields cannot fit on a terminal operator's screen at one time. The default is NO.

The DML statements #MREQ, #STRTPAG, and #ENDPAG are used to control the pageable map option. For more information, see the descriptions of these commands in Data Manipulation Language Statements (see page 73).

## @COPY IDMS

The @COPY IDMS statement copies source data description code and modules from the dictionary into the program at the location of the @COPY IDMS statement. This statement copies CA IDMS/DB database record descriptions, the IDMS communications block, map record descriptions, or MRBs. However, any source module or record description stored in the dictionary can be copied into either a CSECT or DSECT, as specified by the DSECT parameter (discussed below).

Source code requirements differ according to the usage (DML, LR, or MIXED) defined in the program's subschema. The program should not copy components that conflict with its usage. These usages determine the types of records a program can access, as follows:

- **DML** allows a program that uses the named subschema to access database records only and requires the following source code components:
  - **SUBSCHEMA-CTRL**— The IDMS communications block through which the application program and the DBMS communicate (for further details, see IDMS Communications Block (see page 34))
  - **SUBSCHEMA-RECORDS**— The descriptions of all records to which the subschema permits access
- **LR** allows a program to access logical records only and requires the following source code components
  - **SUBSCHEMA-CTRL**— The IDMS communications block through which the LRF and the DBMS communicate
  - **SUBSCHEMA-LR-CTRL**— The logical-record request control (LRC) block through which the application program and the Logical Record Facility communicate (for further details, see Logical-Record Request Control (LRC) Block (see page 52))
  - **SUBSCHEMA-LR-RECORDS**— The descriptions of all logical records defined in the subschema
- **MIXED** allows a program to access both database records and logical records; this usage requires the following source code components:
  - **SUBSCHEMA-CTRL**— The IDMS communications block through which the application program and the LRF communicates with the DBMS, For further details, see IDMS Communications Block (see page 34).
  - **SUBSCHEMA-RECORDS**— The descriptions of all records to which the subschema permits access
  - **SUBSCHEMA-LR-CTRL**— The logical-record request control (LRC) block, through which the application program and the LRF communicate (for further details, see Logical-Record Request Control (LRC) Block (see page 52))
  - **SUBSCHEMA-LR-RECORDS**— The descriptions of all logical records defined in the subschema

The DML precompiler determines whether source record descriptions are copied into a CSECT or DSECT portion of the program, and applies the following rules:

- If the record is being copied into a CSECT, the DML precompiler defines record elements that have specified initial values by means of the Assembler DC (define constant) data definition instruction.

- If the record is being copied into a DSECT, DML defines record elements that have specified initial values by means of the Assembler DS (Define Storage) data definition instruction.

**Note:** The DML defines record elements using the Assembler EQU instruction if the record element is:

- created in the dictionary (IDD) with the USAGE CONDITION-NAME parameter (according to the COBOL 88-level convention)

  and

- copied into a CSECT or DSECT.

**Note:** If the optional keyword DSECT is coded in the @COPY IDMS statement, the record being copied is established as an individual DSECT named with the record name.

**Syntax**

```
►►─── @COPY IDMS ─────────────────────────────────────────────────────────►

 ►─┬─ ,SUBSCHEMA-DML-LR DESCRIPTION ─────────────────────────────────────────►
   ├─ ,SUBSCHEMA-DESCRIPTION ──────────────────────────────────┤
   ├─ ,SUBSCHEMA-CTRL ─────────────────────────────────────────┤
   ├─ ,SUBSCHEMA-RECORDS ──────────────────────────────────────┤
   ├─ ,RECORD=record-name ─┬───────────────────────────┬───────┤
   │                       └─ VERSION=version-number ──┘       │
   ├─ ,SUBSCHEMA-LR-DESCRIPTION ───────────────────────────────┤
   ├─ ,SUBSCHEMA-LR-CTRL ─┬─────────────────────────┬──────────┤
   │                      └─ ,SIZE=lrc-block-size ──┘          │
   ├─ ,SUBSCHEMA-LR-CONTROL ───────────────────────────────────┤
   ├─ ,SUBSCHEMA-LR-RECORDS ───────────────────────────────────┤
   ├─ ,LR=logical-record-name ─────────────────────────────────┤
   ├─ ,MAPS ───────────────────────────────────────────────────┤
   ├─ ,MAP=map-name ───────────────────────────────────────────┤
   ├─ ,MAP-CONTROLS ───────────────────────────────────────────┤
   ├─ ,MAP-CONTROL=map-name ───────────────────────────────────┤
   ├─ ,MAP-RECORDS ────────────────────────────────────────────┤
   ├─ ,MODULE=module-name ─┬───────────────────────────┬───────┤
   │                       └─ VERSION=version-number ──┘       │
   ├─ ,SUBSCHEMA-BINDS ────────────────────────────────────────┤
   └─ ,MAP-BINDS ──────────────────────────────────────────────┘

 ►─┬──────────┬─────────────────────────────────────────────────────────────►◄
   └─ ,DSECT ─┘
```

**Parameters**

**SUBSCHEMA-DML-LR-DESCRIPTION**

(Subschema usage is mixed); copies all components required to access both database and logical records: SUBSCHEMA-CTRL, SUBSCHEMA-RECORDS, SUBSCHEMA-LR-CTRL, and SUBSCHEMA-LR-RECORDS.

**SUBSCHEMA-DESCRIPTION**

(Subschema usage is DML); copies the source data description code for the IDMS communications block (SUBSCHEMA-CTRL) and for all records (SUBSCHEMA-RECORDS) defined in the subschema specified in the @INVOKE statement.

**SUBSCHEMA-CTRL**

Copies the IDMS communications block into the program.

**SUBSCHEMA-RECORDS**

Copies the source data description code for all records defined in the subschema into the program. You can copy Assembler synonyms defined for the subschema records in the data dictionary into the program according to the rules of synonym usage.

**RECORD=**

Copies the description of an individual record defined in the dictionary.

***record-name***

Can be the primary name or a synonym for a record or module stored in the dictionary.

A record that has been copied into a schema can only be copied into a program that uses a subschema associated with the schema. In other words, schema-owned records cannot be copied into non-IDMS programs (that is, programs that do not use a subschema and that do not access the database). However, a synonym defined for the schema-owned record *can* be copied into a non-IDMS program (use the VERSION clause to identify the synonym).

**VERSION=*version-number***

Optional; can be used to qualify IDD records (but not schema-owned records) with a version number. If no version number is specified, CA IDMS/DB first assumes that *record-name* identifies a record that is included in the subschema named in the @INVOKE statement, and looks for it in that subschema. If the record is not associated with a subschema, *version* defaults to the highest version number of the record defined in the dictionary for the operating mode under which the program is being compiled.

**SUBSCHEMA-LR-DESCRIPTION**

Copies all components required to access logical records: SUBSCHEMA-CTRL, SUBSCHEMA-LR-CTRL, and SUBSCHEMA-LR-RECORDS.

**SUBSCHEMA-LR-CTRL**

Copies the LRC block data description.

**SIZE=*lrc-block-size***

Optional; specifies the size of that portion of the LRC block that contains information about the logical-record request's WHERE clause. *Lrc-block-size* defaults to 576 bytes. If included, it should specify a size large enough to accommodate the most complex WHERE clause in the program. *Lrc-block-size* is calculated as follows:

1. Multiply the greatest number of operands and operators that will be included in a single WHERE clause by 16 bytes.
2. Add the number of bytes, rounded up to the nearest multiple of 8, associated with the data field for each operand; that is:
   - The number of characters in a keyword
   - The number of characters in a field described by a program variable or by a logical-record field named in the OF LR clause.
3. Add the length, rounded up to the nearest multiple of 8, of each operand that is a character literal.
4. Add 12 bytes for each operand that is a numeric literal.
5. Add 64 bytes for fixed logical-record request control (LRC) overhead.

*Lrc-block-size* must be a positive integer in the range 64 through 9999. Note that 64 can be specified if none of the logical-record requests issued by the program include WHERE clauses.

- **SUBSCHEMA-LR-CONTROL** copies the SUBSCHEMA-CTRL and SUBSCHEMA-LR-CTRL components. Do not include SUBSCHEMA-LR-CONTROL if the subschema's usage is DML.
- **SUBSCHEMA-LR-RECORDS** copies the descriptions of all logical records defined in the subschema.
- **LR=*logical-record-name*** copies the description of an individual logical record defined in the subschema.
- **MAPS** copies the #MRB statements required to establish the MRBs for all maps specified in the @INVOKE statement. Additionally, the @COPY IDMS,MAPS statement copies the source data description code for map records associated with all maps specified in the @INVOKE statement.
- **MAP=*map-name*** copies the #MRB statement and map records associated with the named map. *Map-name* is the 1- to 8-character name of the map. The version number of the map defaults to the version number specified for the map in the @INVOKE statement.
- **MAP-CONTROLS** copies the #MRB statements for all maps specified in the @INVOKE statement.

- **MAP-CONTROL=***map-name* copies the #MRB statement for the named map. *Map-name* is the 1- to 8-character name of the requested map. The version number of the map defaults to the version number specified in the @INVOKE statement.
- **MAP-RECORDS** copies the map records associated with all maps specified in the @INVOKE statement.
- **MODULE=***module-name*,VERSION=*version* copies a sequence of Assembler source statements stored in the dictionary. *Module-name* is the 1- to 8-character name of the requested module; it can be optionally qualified by *version*. The version number defaults to the highest version number defined in the dictionary for the requested module.

  The @COPY IDMS,MODULE statement copies a module from the dictionary into the source program. The DBA must have previously added this module to the data dictionary by means of the IDD DDDL compiler.

  The DML precompiler places the module into the program at the location of the request. The module may contain DML statements. If DML statements are present, they are treated as if the programmer had coded them directly. @COPY IDMS,MODULE statements can be nested (that is, code invoked by an @COPY IDMS,MODULE statement can itself contain a @COPY IDMS,MODULE statement). However, you must ensure that a copied module does not, in turn, copy itself.
- **SUBSCHEMA-BINDS** copies @BIND SUBSCH and @BIND REC statements for each CA IDMS/DB database record accessed by the program.

  The @COPY IDMS,SUBSCHEMA-BINDS statement instructs the precompiler to bring into the source program a standard @BIND SUBSCH statement and appropriate standard @BIND REC statements for each CA IDMS/DB subschema record explicitly copied into the program variable storage by means of @COPY IDMS statements. @COPY IDMS does not automatically generate BINDS for all subschema records; it also does not generate BINDS for logical records.

  All @COPY IDMS,RECORD statements must precede any @COPY IDMS,SUBSCHEMA-BINDS statement, because the DML precompiler is a one-pass precompiler. The DML precompiler will not generate BINDS for any record-type descriptions copied into the program after the @COPY IDMS,SUBSCHEMA-BINDS statement.

  Instead of issuing an @COPY IDMS,SUBSCHEMA-BINDS statement, you can issue @BIND SUBSCH and @BIND REC statements. Separately issued @BIND READY and @BIND REC statements allow the program to perform the following:

  – Check the ERRSTAT field after each @BIND REC statement

- Bind several records to the same location by including a DML @BIND statement for each record (see @BIND REC (see page 104))

Note: The subschema registration feature requires the @COPY IDMS,SUBSCHEMA-BINDS statement to properly assign the programs to the subschema control block. Individual @BIND SUBSCH and @BIND REC statements should not be used if program registration is in effect.

Note: If a record or a synonym of the record has been copied in twice, an @BIND REC statement will not be automatically generated for the record due to the ambiguity.

- **MAP-BINDS** copies appropriate #MAPBIND statements for all maps specified in the @INVOKE statement. (#MAPBIND statements are discussed later in this chapter.) The @COPY IDMS,MAPS statement must be coded before this statement in order to generate binds for the map records.
- **DSECT** copies the source data description code and source modules defined in any of the above @COPY IDMS statements into a DSECT. Records can be individually copied into a DSECT by including the DSECT parameter in each @COPY IDMS statement. Several records can be copied into a single DSECT by explicit use of the Assembler DSECT instruction followed by the individual @COPY IDMS statements; in this case, the DSECT parameter is not specified in the @COPY IDMS statements. When specifying a DSECT, the program is responsible for designating the end of the DSECT storage area.

The following example illustrates the use of the DSECT parameter to create individual dummy control sections for the IDMS communications block and for a map request block:

```
         @MODE MODE=IDMSDC
         @INVOKE SUBSCHEMA=XYZ,SCHEMA=ABC,
               PROGRAM=TESTXYZ,MAP=DEFMAP

*   THE FOLLOWING @COPY IDMS STATEMENT COPIES THE SOURCE DATA
*   DESCRIPTION CODE FOR THE IDMS COMMUNICATION BLOCK (SUBSCHEMA-CTRL):

         @COPY IDMS,SUBSCHEMA-CTRL,DSECT

*   THE DML PRECOMPILER GENERATES THE DSECT INSTRUCTION FOR THE DUMMY
*   CONTROL SECTION TO CONTAIN THE SOURCE DATA DESCRIPTION CODE OF THE
*   IDMS COMMUNICATIONS BLOCK:

         DSECT
SSCTRL   DS
         .
         .
         .

*   THE FOLLOWING @COPY IDMS STATEMENT COPIES THE SOURCE DATA
*  DESCRIPTION CODE FOR THE REQUIRED MAP REQUEST BLOCK (MAP-CONTROLS):

          @COPY IDMS,MAP-CONTROL=DEFMAP,DSECT

*  THE DML PRECOMPILER GENERATES THE DSECT INSTRUCTION FOR THE DUMMY
*  CONTROL SECTION TO CONTAIN THE SOURCE DATA DESCRIPTION CODE FOR
*  THE MRB:

         DSECT
         DS
         .
         .
         .

*  THE END OF EACH DSECT MUST BE DESIGNATED EITHER BY AN ASSEMBLER
*  END, CSECT, OR ANOTHER DSECT INSTRUCTION.
```

A single DSECT is created for the IDMS communications block, CA IDMS/DB record descriptions, MRB, and map record description.

```
         @MODE MODE=IDMSDC
         @INVOKE SUBSCHEMA=XYZ,SCHEMA=ABC,                              *
               PROGRAM=TESTXYZ,MAP=DEFMAP

*  THE FOLLOWING ASSEMBLER DSECT INSTRUCTION IS CODED BY THE
*  PROGRAMMER TO DEFINE THE BEGINNING OF A DUMMY CONTROL SECTION:

IDMSSTG  DSECT

*  COPY STATEMENTS WITHIN A DSECT ENABLE RECORD DESCRIPTIONS TO BE
*  COPIED INTO THE DUMMY CONTROL SECTION.  NOTE THAT THE DSECT
*  PARAMETER IS NOT INCLUDED IN THE @COPY IDMS STATEMENTS:

         @COPY IDMS,SUBSCHEMA-DESCRIPTION
SSCTRL   DS
         .
         .
         .
         DS
         .
         .
         @COPY IDMS,MAPS
         DS
         .
```

```
              .
              .
              DS
              .
              .
              .

*  THE END OF THE DSECT MUST BE DESIGNATED BY AN ASSEMBLER END,
*  CSECT, OR ANOTHER DSECT INSTRUCTION.
```

## #MRB

The #MRB statement establishes a map request block (MRB) in the program's variable storage area. It allocates storage, but does not initialize that storage. For each mapping request, the MRB communicates between the program and the mapping compiler. A separate MRB must be defined for each map used by a program. The DML precompiler uses map information stored in the dictionary to determine the actual size of the MRB, and generates the necessary Assembler DS instructions with macros.

One or more #MRB statements can be copied into the program by using the @COPY IDMS statement, discussed earlier in this chapter.

**Syntax**

```
►►── #MRB MAPNAME=map-name ──────────────────────────────────────►
 ►── ,FIELDS=field-count ────────────────────────────────────────►
 ►── ,RECORDS=record-count ──────────────────────────────────────►◄
```

**Parameters**

**MAPNAME=*map-name***

Specifies the 1- to 8-character name of an existing map.

**FIELDS=**

Specifies the number of data and response fields in the specified map.

***field-count***

Absolute expression of the number of fields.

**RECORDS=**

Specifies the number of records in the map.

***record-count***

Absolute expression of the number of records.

# #MAPBIND

For each map request block used by a program, a #MAPBIND request specifies the MRB location and initializes the fields of the MRB. #MAPBIND statements can be global or record-specific:

- **Global**—By specifying only the map name, the #MAPBIND statement applies to the map as a whole. It initializes the entire MRB and fills in fields that apply to the map in general.
- **Record-specific**—By specifying RECNAME and RECADDR parameters as well as the map name, the #MAPBIND statement applies only to the named map record. It initializes the variable storage address of the named record in the MRB.

A program typically issues a global #MAPBIND statement for each map, followed by #MAPBIND statements for each map record used by the program. The program can alter the storage address for a map record at any time by issuing another #MAPBIND statement for that record.

After the initial global bind, all records are considered unbound; map operations that use those records will not have any effect on storage. After binding a record to a storage address, subsequent map operations will use that address to access the record. To unbind a record, issue a record-specific #MAPBIND statement and specify a null (0) bind location using the RECADDR parameter.

All global and record-specific #MAPBIND statements for a map can be copied automatically into the program with the @COPY IDMS statement, discussed earlier in this chapter.

**Syntax**

```
►►── #MRB MAPNAME=map-name ──────────────────────────────────────►
 ►── ,FIELDS=field-count ────────────────────────────────────────►
 ►── ,RECORDS=record-count ──────────────────────────────────────►◄
```

**Parameters**

**MRB=**

Initializes the MRB associated with the named map.

***map-name***

Specifies the 1- to 8-character name of an existing map.

**RECNAME=*record-name***

Is the 1- to 32-character name of a record used by the map.

**RECADDR=**

Requests that the named record be unbound or specifies the storage address to which the record will be bound.

**0**

(Default); specifies that the named record is to be unbound.

***record-address***

Specifies a register that contains either the address of the area or the symbolic name of a user-defined field containing the address of the area. Subsequent I/O operations will use the specified area of storage for any operations dealing with the record.

## @SSCTRL

The @SSCTRL statement is an Assembler macro used to generate source data description code for the IDMS communications block. @SSCTRL must be used in place of the @COPY IDMS,SUBSCHEMA-CTRL statement when the DML precompiler is not used.

**Syntax**

```
►►── @SSCTRL ─────────────────────────────────────────────────────────►◄
```

**Note:** To use an IDMS communications block in which the RECORD, AREA, and ERROR-SET/RECORD/AREA fields are 18 bytes, specify @SSC120 instead.

## @SSLRCTL

The @SSLRCTL statement is an Assembler macro instruction that generates source data description code for the LRC block. @SSLRCTL must be used in place of the @COPY IDMS,SUBSCHEMA-LR-CTRL statement when the DML precompiler is not used.

**Syntax**

```
►►── @SSLRCTL ─┬────────────────────────────────┬─────────────────────►◄
               └─ LRSIZ=lr-control-block-size ──┘
```

**Parameters**

**LRSIZ=**

Specifies the size of that portion of the LRC block that contains information about the logical-record request's WHERE clause.

***lrc-block-size***

Defaults to 576 bytes; if included, it should specify a size large enough to accommodate the most complex WHERE clause in the program. (For the algorithm for calculating *lrc-block-size*, see @COPY IDMS (see page 411) earlier in this chapter.)

# Chapter 8: Considerations for Assembler Programs in a DC/UCF Online System

Certain coding conventions should be observed in Assembler programs which are to be used in a DC/UCF online system both for stand-alone programs and programs which are to be called from another online program. This chapter will discuss the following topics:

- SVC instructions in an online program
- Making your assembler program reentrant
- Methods of calling an assembler subprogram

Defining an assembler program which uses standard IBM calling conventions

This section contains the following topics:

## SVC Instructions in an Online Program

You should avoid coding any SVC instructions or macros that generate SVC instructions in an online DC/UCF assembler program. While an SVC is in control, no other online task can use the DC/UCF system. This prevents the system from allocating resources between tasks as it is designed to do. In addition, any error that occurs during the processing of an SVC instruction can cause a hang or abnormal termination of the entire DC/UCF system.

If it is absolutely necessary to code such an instruction in an online program, the program must not be called via a COBOL or PL/I CALL instruction. This restriction is explained further in later sections of this chapter.

## Making Your Assembler Program Reentrant

All programs that are designed to run in an online DC environment should be written using fully reentrant coding techniques. This means that the program should never update its own storage. Any variable storage that your program needs to update should be in an area reserved for the exclusive use of a single task. Typically, you would define a DSECT to map this area. Several techniques can be used to achieve this goal. Two or more of these techniques can be combined in a single program.

- Specify ISASIZE on the PROGRAM statement in the DC Sysgen. On entry to your program, register 11 will be set to point to an area of this size reserved for the use of your program.
- On entry to your program, code a #GETSTG ...PLIST=*,LENGTH=constant.... This form of the #GETSTG macro does not update any program variable storage. The sample program in Appendix C of this manual uses this technique.
- An assembler subprogram can use storage passed to it from its caller provided that storage is itself reentrant.
- If one of the above techniques is used to obtain enough storage for a PLIST, then a more generic form of the #GETSTG macro can be used to obtain further variable storage.
- Specify SAVEAREA on the PROGRAM statement in the DC Sysgen. See section 7.4 below for more information.

A non-reentrant assembler routine can be used in an online DC environment under certain limited circumstances, but this is not recommended for reasons explained below.

- A non-reentrant stand-alone assembler program, i.e., an assembler program which is linked as its own load module or phase, can be invoked directly from a TASK CODE or can be called via a high level language TRANSFER CONTROL, a #LINK from another assembler program or a LINK PROGRAM from an ADS dialog. Such a program must be defined in the DC/UCF Sysgen as non-reentrant. Note that this will cause separate copies of the load module to be loaded for every concurrent task using the program. This is generally highly inefficient.
- A non-reentrant assembler subprogram can be called via a dynamic or static COBOL CALL verb, or a PL/I CALL verb if all of the following conditions are met:
  - The subprogram does not issue any SVC or PC calls. Note that many IBM macros generate such calls.
  - The subprogram does not issue any DML calls.
  - Multitasking is not in effect on the DC/UCF system.

  This technique is not recommended because a small change in the program or online environment may cause the program to stop functioning correctly and potentially allow it to cause storage and data corruption.

## Methods of Calling an Online Assembler Subprogram

### TRANFER CONTROL, #LINK, or ADS LINK

The preferred method for calling an assembler subprogram is a TRANSFER CONTROL from COBOL or PL/I, a #LINK from another assembler program, or a LINK instruction from ADS. If this method of control is used, the DC/UCF system is in control of the calling process. This method provides the following advantages:

- If an error occurs in the subprogram, system error messages will reflect the correct program name.
- Any limits set for that program will be taken into account
- SVC screening will be turned off. This is imperative if the subprogram issues any SVC instructions and it is called from a COBOL or PL/I program.

### COBOL or PL/I CALL

It is valid to call a stand-alone assembler subprogram via a CALL IDENTIFIER from a COBOL program or CALL from a PL/I program provided that the program does not issue any SVC instructions. An assembler subprogram can also be link edited in the same load module with its caller provided that it does not issue any SVC instructions or DML calls. The DC/UCF system will not be aware that the subprogram has been called. So any limits or system-generated error messages will not reflect the call.

A COBOL or PL/I CALL may use somewhat less CPU than a TRANSFER CONROL DML verb. Therefore, it may be desirable to use this technique if a qualified subprogram is called many times in the same task.

### Assembler LINK macro

It is never valid to use the assembler LINK macro in an online assembler program. This macro generates an SVC that is incompatible with DC/UCF online processing. An abnormal termination of the DC/UCF system may occur.

## Standard IBM calling conventions

An assembler program that is written using standard IBM calling conventions can be used as a top-level program or a subprogram in a DC/UCF online system. Such a program will typically issue an instruction to save its registers in an area pointed to by General Register 13 on entry. The following conventions must be observed:

- If the program is a top-level program or is invoked via a TRANSFER CONTROL, #LINK or ADS LINK, then the SAVEAREA parameter must be specified on the PROGRAM statement in the DC/UCF Sysgen. SAVEREA is not needed if the program is invoked via a COBOL or PL/I CALL, but the SAVEAREA parameter will not cause a problem if it is specified.
- If the program sets register 13 to point at its own save area and stores into it, then the save area must be in reentrant storage obtained using one of the methods described in section 7.3.

# Appendix A: DML Precompile, Assembly, and Link-Edit JCL

This appendix describes processing for Assembler programs containing DML statements. It also provides samples of the z/OS, z/VSE and CMS, commands you use to prepare these programs.

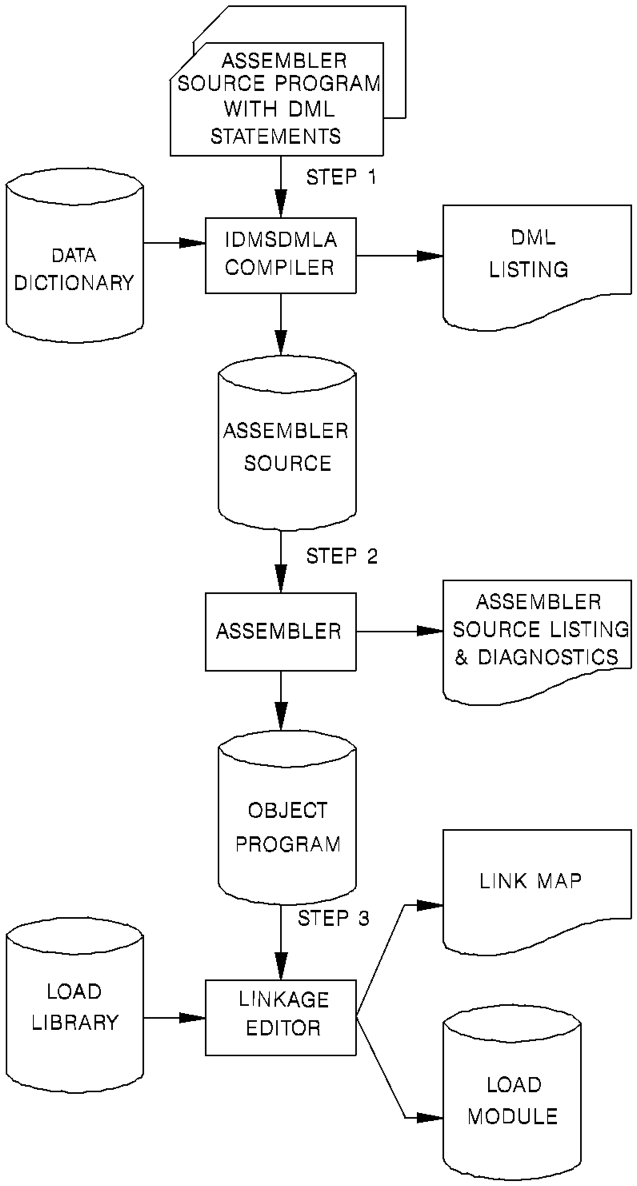
**Processing Assembler Programs Containing DML**

To prepare a DML program for execution, you first execute the DML precompiler (IDMSDMLA). After this, you assemble and link edit.

| Component | Input | Output |
|---|---|---|
| IDMSDMLA | ■ Assembler source program containing DML<br>■ Protocol/control information<br>■ Dictionary record descriptions | ■ Source Assembler program with DML-generated code<br>■ DML and source listing and diagnostics |
| Assembler | Source program produced by IDMSDMLA | ■ Object program<br>■ Assembler listing |
| Linkage Editor | Object program produced by assembler | ■ Load module<br>■ Link-edit map |

## Steps for Assembly

The following figure illustrates steps involved in assembling a DML Assembler program.



This section contains the following topics:

# IDMSDMLA Under z/OS

**Executing Under the Central Version IDMSDMLA (z/OS)**

```
//*****************************************************************
//**                  PRECOMPILE PROGRAM                         **
//*****************************************************************
//precomp  EXEC PGM=IDMSDMLA,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.custom.loadlib,DISP=SHR
//         DD   DSN=idms.cagjload,DISP=SHR
//sysctl   DD   DSN=idms.sysctl,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYSPCH   DD   DSN=&.&source.,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(TRK,(10,5),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSLST   DD   SYSOUT=A
//SYSIDMS  DD   *
DMCL=dmcl-name
DICTNAME=dictionary-name
Other SYSIDMS parameters, as appropriate
/*
//SYSIPT   DD   *
Assembler DML source statements
/*
//*****************************************************************
//**                  ASSEMBLE PROGRAM                           **
//*****************************************************************
//asm      EXEC PGM=assembler,REGION=1024K,PARM='DECK,LIST,NOLOAD'
//SYSPRINT DD   SYSOUT=A
//SYSLIB   DD   DSN=sys1.maclib,DISP=SHR
//         DD   DSN=yourHLQ.CAGJMAC,DISP=SHR
//SYSUT1   DD   UNIT=disk,SPACE=(CYL,(3,2))
//SYSUT2   DD   UNIT=disk,SPACE=(CYL,(3,2))
//SYSUT3   DD   UNIT=disk,SPACE=(CYL,(3,2))
//SYSPUNCH DD   DSN=&.&object.,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(TRK,(10,5),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
```

```
//SYSIN    DD   DSN=&.&source.,DISP=(OLD,DELETE)
//*****************************************************************
//**                 LINK PROGRAM MODULE                        **
//*****************************************************************
//link     EXEC PGM=IEWL,REGION=300K,PARM='LET,LIST,NCAL,XREF'
//SYSUT1   DD   UNIT=disk,SPACE=(TRK,(20,5))
//loadlib  DD   DSN=idms.loadlib,DISP=SHR
//SYSLMOD  DD   DSN=user.loadlib,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSLIN   DD   DSN=&.&object.,DISP=(OLD,DELETE)
//         DD   *
 INCLUDE loadlib(IDMS)      required for BATCH and DCBATCH, omit for CICS
 INCLUDE loadlib (IDMSCINT) for CICS only
 INCLUDE loadlib(IDMSCANC)  optional; BATCH and DCBATCH only
 INCLUDE loadlib(IDMSOPTI)  optional; BATCH and DCBATCH only
 ENTRY   userentry
 NAME    userprog(R)
/*
//*
```

***idms.dba.loadlib***

Data set name of the load library containing the DMCL and database name table load modules

***idms.custom.loadlib***

Data set name of the load library containing the customized CA IDMS executable modules

***idms.cagjload***

Data set name of the load library containing the vanilla CA IDMS executable modules

***sysctl***

DDname of SYSCTL file

***idms.sysctl***

Data set name of SYSCTL file

***dcmsg***

DDname of the system message (DDLDCMSG) area

***idms.sysmsg.ddldcmsg***

Data set name of the system message (DDLDCMSG) area

***&.&source***

Name of the temporary data set output from the precompiler

***disk***

Symbolic device name for work files

***dmcl-name***

specifies the name of the dictionary the DMLF precompiler should access

***dictionary-name***

Identifies the DC/UCF system to bind at runtime

***assembler***

Name of the assembler program

***sys1.maclib***

Vendor-supplied system macro library

***yourHLQ.CAGJMAC***

Vendor-supplied idms macro library, created at installation time

***&.&object.***

Name of temporary data set output from Assembler

***user.loadlib***

User application load library

***loadlib***

DDname of the *idms.loadlib*

***userentry***

Name of a program entry point

***userprog***

Name of program in load library

**Note:** Depending on the central version operating environment, an IDMSOPTI module link edited with IDMSDMLA can be used in place of or in addition to the SYSCTL file.

The link of CICS application programs that use IDMSCINT must incorporate JCL to resolve external reference DFHEI1. The particular JCL depends on the nature and language of your application. See the appropriate IBM CICS application programming documentation for details.

**Executing in Local Mode IDMSDMLA (z/OS)**

```
//******************************************************************
//**                    PRECOMPILE PROGRAM                        **
//******************************************************************
//precomp  EXEC PGM=IDMSDMLA,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.custom.loadlib,DISP=SHR
//         DD   DSN=idms.cagjload,DISP=SHR
//dictb    DD   DSN=idms.appldict.ddldml,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrnl  DD   DSN=idms.tapejrnl,DISP=(NEW,CATLG),UNIT=tape
//SYSPCH   DD   DSN=&.&source.,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(TRK,(10,5),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSLST   DD   SYSOUT=A
//SYSIDMS  DD   *
DMCL=dmcl-name
DICTNAME=dictionary-name
Other SYSIDMS parameters, as appropriate
/*
//SYSIPT   DD   *
Assembler DML source statements
/*
//******************************************************************
//**                    ASSEMBLE PROGRAM                          **
//******************************************************************
//asm      EXEC PGM=assembler,REGION=1024K,PARM='DECK,LIST,NOLOAD'
//SYSPRINT DD   SYSOUT=A
//SYSLIB   DD   DSN=sys1.maclib,DISP=SHR
//         DD   DSN=idms.cagjmac,DISP=SHR
//SYSUT1   DD   UNIT=disk,SPACE=(CYL,(3,2))
//SYSUT2   DD   UNIT=disk,SPACE=(CYL,(3,2))
//SYSUT3   DD   UNIT=disk,SPACE=(CYL,(3,2))
//SYSPUNCH DD   DSN=&.&object.,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(TRK,(10,5),RLSE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSIN    DD   DSN=&.&source.,DISP=(OLD,DELETE)
//******************************************************************
//**                    LINK PROGRAM MODULE                       **
//******************************************************************
```

```
//link     EXEC PGM=IEWL,REGION=300K,PARM='LET,LIST,NCAL,XREF'
//SYSUT1   DD   UNIT=disk,SPACE=(TRK,(20,5))
//VANILLA  DD   DSN=idms.cagjload,DISP=SHR
//CUSTOM   DD   DSN=idms.custom.loadlib,DISP=SHR
//SYSLMOD  DD   DSN=idms.custom.loadlib,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSLIN   DD   DSN=&.&object.,DISP=(OLD,DELETE)
//         DD   *
 INCLUDE VANILLA(IDMS)      required for BATCH and DCBATCH, omit for CICS
 INCLUDE CUSTOM(IDMSCINT) for CICS only
 INCLUDE CUSTOM(IDMSOPTI)  optional; BATCH and DCBATCH only
 ENTRY   userentry
 NAME    userprog(R)
/*
//*
```

***idms.dba.loadlib***

Data set name of the load library containing the DMCL and database name table load modules

***idms.custom.loadlib***

Data set name of the load library containing the customized CA IDMS executable modules

***idms.cagjload***

Data set name of the load library containing the vanilla CA IDMS executable modules

***dictb***

DDname of journal file

***idms.appldict.ddldml***

File-ID of the application dictionary definition (DDLDML) area

***dcmsg***

Filename of the system message (DDLDCMSG) area

***idms.sysmsg.ddldcmsg***

File-ID of the system message (DDLDCMSG) area

***sysjrnl***

DDname of the tape journal file

***idms.tapejrnl***

File ID of tape journal file

***tape***

Device name for the tape journal file

***&.&source.***

Name of the temporary data set output from the precompiler

***disk***

Symbolic device name for work files

***dmcl-name***

Specifies the name of the dictionary the DMLF precompiler should access

***dictionary-name***

Identifies the DC/UCF system to bind at runtime

***assembler***

Name of the assembler program

***sys1.maclib***

Vendor-supplied system macro library

***idms.cagjmac***

Vendor-supplied idms macro library, supplied at installation time

***&.&object.***

Name of temporary data set output from Assembler

***user.loadlib***

User application load library

***VANILLA***

DDname for the *loadlib* created during the SMP/E install

***CUSTOM***

DDname for the *loadlib* created during configuration

***userentry***

Name of a program entry point

***userprog***

Name of program in load library

# IDMSDMLA Under z/VSE

**Executing Under the Central Version IDMSDMLA (z/VSE)**

```
/*****************************************************************
/**                  PRECOMPILE PROGRAM                         **
/*****************************************************************
* step1
// EXEC PROC=IDMSLBLS
// UPSI b                 if specified in IDMSOPTI module
// DLBL      idmspch,'temp.dmla',0
// EXTENT    SYS020,nnnnnn,,,ssss,llll
// ASSGN     SYS020,DISK,VOL=nnnnnn,SHR
// EXEC      IDMSDMLA

Input SYSIDMS parameters here, as required

/*

Assembler/DML source statements

/*****************************************************************
/**                  COMPILE PROGRAM                            **
/*****************************************************************
/*
* step2
// DLBL      IJSYSIN,'temp.dmla',0
// EXTENT    SYSIPT,nnnnnn
   ASSGN     SYSIPT,DISK,VOL=nnnnnn,SHR
// OPTION    CATAL,NODECK,NOSYM
  PHASE userprog,*
// EXEC      ASSEMBLY
/*****************************************************************
/**                  LINK PROGRAM MODULE                        **
/*****************************************************************
* step3
   CLOSE     SYSIPT,SYSRDR
ENTRY (dmla)
// EXEC      LNKEDT
/*
```

***IDMSLBLS***

Name of the procedure provided at installation that contains the file definitions for CA IDMS dictionaries and databases.

**Note:** For a complete listing of IDMSLBLS, see "IDMSLBLS Procedure".

***b***

Appropriate UPSI switch, 1 through 8 characters, if specified in the IDMSOPTI module

***idmspch***

Filename of data set output from the IDMSDMLA precompiler

***temp.dmla***

File ID of data set output from the IDMSDMLA precompiler

***SYS020***

Logical unit assignment of the DMLA output

***nnnnnn***

Volume serial identifier of appropriate disk volume

***ssss***

Starting track (CKD) or block (FBA) of disk extent

***llll***

Number of tracks (CKD) or blocks (FBA) of disk extent

***userprog***

Name of program in the library

***dmla***

Name of Assembler/DML module

**Runtime Parameters**

You can use SYSIDMS parameters to specify information about your runtime environment.

**Note:** For more information about optional SYSIDMS parameters, see the *Common Facilities Guide*.

**INCLUDE Statements**

For programs that include an Assembler internal sort, place the following statements in the second step, before EXEC ASSEMBLY:

```
ACTION NOAUTO              prevents multiple inclusions of IDMS

INCLUDE IDMS               IDMS interface for use with COMRG

INCLUDE IDMSOPTI           IDMSOPTI module
                           (omit in local mode)

INCLUDE IDMSCANC           local mode abort entry point
                           (omit IDMSCANC if TP application)
```

**Note:** Assembler overlay programs must resolve references to IDMS within their root segment; care must be taken to prevent the overlaying of the IDMS interface. Use of IDMS and IDMSLDPT is recommended for these programs.

**Executing in Local Mode**

To execute the IDMSDMLA precompiler in local mode, remove the UPSI specification and add the following statements in step 1 (the IDMSDMLA step):

```
// TLBL     sysjrnl,'idms.tapejrnl',,nnnnnn,,f
// ASSGN    SYS009,TAPE,VOL=nnnnnn
```

***idms.tapejrnl***

File ID of tape journal file

***f***

File number of tape journal file

***sys009***

Logical unit assignment for journal file

**IDMSLBLS Procedure**

The IDMSLBLS procedure is provided during CA IDMS installation. It contains file definitions for the CA IDMS components, such as these:

- Dictionaries
- Sample databases

- Disk journal files
- SYSIDMS file

Tailor the IDMSLBLS procedure to reflect the filenames and definitions in use at your site and include this procedure in z/VSE JCL job streams.

The following is a listing of the IDMSLBLS procedure:

```
* ----------- LIBDEFS -----------
// LIBDEF  *,SEARCH=idmslib.sublib
// LIBDEF  *,CATALOG=user.sublib
/*  --------------------- LABELS -----------------------
// DLBL    idmslib,'idms.library',1999/365
// EXTENT  ,nnnnnn,,,ssss,1500
// DLBL    dcdml,'idms.system.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclod,'idms.system.ddldclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,21
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclog,'idms.system.ddldclog',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,401
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcrun,'idms.system.ddldcrun',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,68
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcscr,'idms.system.ddldcscr',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,135
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcmsg,'idms.sysmsg.ddldcmsg',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclscr,'idms.sysloc.ddlocscr',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dirldb,'idms.sysdirl.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dirllod,'idms.sysdirl.ddldclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,2
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    empdemo,'idms.empdemo1',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    insdemo,'idms.insdemo1',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    orgdemo,'idms.orgdemo1',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    empldem,'idms.sqldemo.empldemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR

// DLBL    infodem,'idms.sqldemo.infodemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    projdem,'idms.projseg.projdemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    indxdem,'idms.sqldemo.indxdemo',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sysctl,'idms.sysctl',1999/365,SD
// EXTENT  SYSnnn,nnnnnn,,,ssss,2
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    secdd,'idms.sysuser.ddlsec',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,26
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dictdb,'idms.appldict.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dloddb,'idms.appldict.ddldclod',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqldd,'idms.syssql.ddlcat',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqllod,'idms.syssql.ddlcatl',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqlxdd,'idms.syssql.ddlcatx',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,26
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    asfdml,'idms.asfdict.ddldml',1999/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL     asflod,'idms.asfdict.asflod',1999/365,DA
// EXTENT   SYSnnn,nnnnnn,,,ssss,401
// ASSGN    SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL     asfdata,'idms.asfdict.asfdata',1999/365,DA
// EXTENT   SYSnnn,nnnnnn,,,ssss,201
// ASSGN    SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL     ASFDEFN,'idms.asfdict.asfdefn',1999/365,DA
// EXTENT   SYSnnn,nnnnnn,,,ssss,101
// ASSGN    SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL     j1jrnl,'idms.j1jrnl',1999/365,DA
// EXTENT   SYSnnn,nnnnnn,,,ssss,54
// ASSGN    SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL     j2jrnl,'idms.j2jrnl',1999/365,DA
// EXTENT   SYSnnn,nnnnnn,,,ssss,54
// ASSGN    SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL     j3jrnl,'idms.j3jrnl',1999/365,DA
// EXTENT   SYSnnn,nnnnnn,,,ssss,54
// ASSGN    SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL     SYSIDMS,'#SYSIPT',0,SD
/+
/*
```

***idmslib.sublib***

Name of the sublibrary within the library containing CA IDMS modules

***user.sublib***

Name of the sublibrary within the library containing user modules

***idmslib***

Filename of the file containing CA IDMS modules

***idms.library***

File-ID associated with the file containing CA IDMS modules

***SYSnnn***

Logical unit of the volume for which the extent is effective

***nnnnnn***

Volume serial identifier of appropriate disk volume

***ssss***

Starting track (CKD) or block (FBA) of disk extent

***dccat***

Filename of the system dictionary catalog (DDLCAT) area

***idms.system.dccat***

File-ID of the system dictionary catalog (DDLCAT) area

***dccatl***

Filename of the system dictionary catalog load (DDLCATLOD) area

***idms.system.dccatlod***

File-ID of the system dictionary catalog load (DDLCATLOD) area

***dccatx***

Filename of the system dictionary catalog index (DDLCATX) area

***idms.system.dccatx***

File-ID of the system dictionary catalog index (DDLCATX) area

***dcdml***

Filename of the system dictionary definition (DDLDML) area

***idms.system.ddldml***

File-ID of the system dictionary definition (DDLDML) area

***dclod***

Filename of the system dictionary definition load (DDLDCLOD) area

***idms.system.ddldclod***

File-ID of the system dictionary definition load (DDLDCLOD) area

***dclog***

Filename of the system log area (DDLDCLOG) area

***idms.system.ddldclog***

File-ID of the system log (DDLDCLOG) area

***dcrun***

Filename of the system queue (DDLDCRUN) area

***idms.system.ddldcrun***

File-ID of the system queue (DDLDCRUN) area

***dcscr***

Filename of the system scratch (DDLDCSCR) area

***idms.system.ddldcscr***

File-ID of the system scratch (DDLDCSCR) area

***dcmsg***

Filename of the system message (DDLDCMSG) area

***idms.sysmsg.ddldcmsg***

File-ID of the system message (DDLDCMSG) area

***dclscr***

Filename of the local mode system scratch (DDLOCSCR) area

***idms.sysloc.ddlocscr***

File-ID of the local mode system scratch (DDLOCSCR) area

***dirldb***

Filename of the IDMSDIRL definition (DDLDML) area

***idms.sysdirl.ddldml***

File-ID of the IDMSDIRL definition (DDLDML) area

***dirllod***

Filename of the IDMSDIRL definition load (DDLDCLOD) area

***idms.sysdirl.dirllod***

File-ID of the IDMSDIRL definition load (DDLDCLOD) area

***empdemo***

Filename of the EMPDEMO area

***idms.empdemo1***

File-ID of the EMPDEMO area

***insdemo***

Filename of the INSDEMO area

***idms.insdemo1***

File-ID of the INSDEMO area

***orgdemo***

Filename of the ORGDEMO area

***idms.orgdemo1***

File-ID of the ORDDEMO area

***empldem***

Filename of the EMPLDEMO area

***idms.sqldemo.empldemo***

File-ID of the EMPLDEMO area

***infodem***

Filename of the INFODEMO area

***idms.sqldemo.infodemo***

File-ID of the INFODEMO area

***projdem***

Filename of the PROJDEMO area

***idms.projseg.projdemo***

File-ID of the PROJDEMO area

***indxdem***

Filename of the INDXDEMO area

***idms.sqldemo.indxdemo***

File-ID of the INDXDEMO area

***sysctl***

Filename of the SYSCTL file

***idms.sysctl***

File-ID of the SYSCTL file

***secdd***

Filename of the system user catalog (DDLSEC) area

***idms.sysuser.ddlsec***

File-ID of the system user catalog (DDLSEC) area

***dictdb***

Filename of the application dictionary definition area

***idms.appldict.ddldml***

File-ID of the application dictionary definition (DDLDML) area

***dloddb***

Filename of the application dictionary definition load area

***idms.appldict.ddldclod***

File-ID of the application dictionary definition load (DDLDCLOD) area

***sqldd***

Filename of the SQL catalog (DDLCAT) area

***idms.syssql.ddlcat***

File-ID of the SQL catalog (DDLCAT) area

***sqllod***

Filename of the SQL catalog load (DDLCATL) area

***idms.syssql.ddlcatl***

Filename of the SQL catalog index (DDLCATX) area

***sqlxdd***

File name of the SQL catalog index (DDLCATX) area

***idms.syssql.ddlcatx***

File-ID of the SQL catalog index (DDLCATX) area

***asfdml***

Filename of the asf dictionary definition (DDLDML) area

***idms.asfdict.ddldml***

File-ID of the asf dictionary definition (DDLDML) area

***asflod***

Filename of the asf dictionary definition load (ASFLOD) area

***idms.asfdict.asflod***

File-ID of the asf dictionary definition load (ASFLOD) area

***asfdata***

Filename of the asf data (ASFDATA) area

***idms.asfdict.asfdata***

File-ID of the asf data area (ASFDATA) area

***ASFDEFN***

Filename of the asf data definition (ASFDEFN) area

***idms.asfdict.asfdefn***

File-ID of the asf data definition area (ASFDEFN) area

***j1jrnl***

Filename of the first disk journal file

***idms.j1jrnl***

File-ID of the first disk journal file

***j2jrnl***

Filename of the second disk journal file

***idms.j2jrnl***

File-ID of the second disk journal file

***j3jrnl***

Filename of the third disk journal file

***idms.j3jrnl***

File-ID of the third disk journal file

***SYSIDMS***

Filename of the SYSIDMS parameter file

## IDMSDMLA Under CMS

**Executing Under the Central Version IDMSDMLA (CMS)**

```
FILEDEF SYSIPT DISK sysipt data a (RECFM F LRECL ppp BLKSIZE nnn
FILEDEF SYSPCH DISK prgnme assemble a
FILEDEF SYSIDMS DISK sysidms parms a (RECFM F LRECL ppp. BLKSIZE nnn
EXEC IDMSFD
OSRUN IDMSDMLA PARM='CVMACH=vmid'       Precompiler step
FILEDEF TEXT DISK prgnme text a
GLOBAL TXTLIB asmlibvs IDMSLIB1
ASSEMBLE prgnme (OSDECK APOST LIB       Assemble step
TXTLIB DEL utextlib prgnme
TXTLIB ADD utextlib prgnme
FILEDEF SYSLMOD uloadlib loadlib a (RECFM V LRECL 1024 BLKSIZE 10 24
FILEDEF objlib DISK utextlib txtlib a
FILEDEF SYSLIB DISK asmlibvs txtlib p
LKED linkctl data a (LIST XREF LET MAP RENT NOTERM PRINT SIZE 512K 64K
                                              Link edit step
```

***sysipt data a***

Filename, type, and mode of the file containing the Assembler/DML source statements

***ppp***

Record length of the data file

***nnn***

Block size of the data file

***prgnme assemble a***

Filename of the Assembler program

***sysidms parms a***

Filename, filetype, and filemode of the file that contains SYSIDMS parameters (parameters that define your runtime environment)

***vmid***

ID of the virtual machine running the CA IDMS/DB central version

***asmlibvs***

Filename of the library that contains Assembler logic modules

***utextlib***

Filename of the user text library

***uloadlib loadlib a***

Filename, filetype, and filemode of the user load library

***objlib1***

DDname of the first CA IDMS/DB object library

***objlib***

DDname of the user object library

***asmlibvs txtlib p***

Filename, filetype, and filemode of the library that contains Assembler logic modules

***linkctl***

Filename of the file that contains the linkage editor control statements

### How to Edit the SYSIDMS File

To edit the SYSIDMS file, enter these CMS commands:

```
XEDIT sysidms parms a (NOPROF
INPUT
 .
 .
 .
SYSIDMS parameters
 .
 .
 .
FILE
```

To run IDMSDMLA, you must include the NODENAME and DICTNAME SYSIDMS parameters.

**Note:** For more information on SYSIDMS, see the *Common Facilities Guide*.

### How to Create the SYSIPT File

To create the SYSIPT file, enter these CMS commands:

```
XEDIT sysipt data a (NOPROF
INPUT
 .
 .
 .
DML source statements
 .
 .
 .
FILE
```

### How to Create the LINKCTL File

To create the LINKCTL file, enter these CMS commands:

```
XEDIT linkctl data a (NOPROF
INPUT
 .
 .
 .
INCLUDE objlib(prgnme)
INCLUDE objlib1(IDMS)     IDMS is required, omit for CICS
INCLUDE objlib1(IDMSCINT) for CICS only
INCLUDE objlib1(IDMSCANC) IDMSCANC for BATCH and DCBATCH
ENTRY prgnme
NAME prgnme(R)
 .
 .
 .
FILE
```

### Executing in Local Mode

To execute the IDMSDMLA precompiler in local mode, remove the CVMACH parameter from OSRUN, and do *one* of the following:

- Link IDMSDMLA with an IDMSOPTI program that specifies local execution mode
- Specify *LOCAL* as the first input parameter in the file specified in the FILEDEF SYSIPT statement
- Modify the OSRUN statement, as follows:

  ```
  OSRUN IDMSDMLA PARM='*LOCAL*'
  ```

  Note: This option is valid only if the OSRUN command is issued from a System Product Interpreter or from an EXEC2 file.

## Link-Edit Considerations

The modules involved in the link edit of an application program contain six external references. Some must be resolved depending on the mode of operation. The following table lists and explains the external references; unresolved references should be checked against this table to ensure proper linkage to the program.

| Reference | Referenced by | Resolved by | Comments |
|---|---|---|---|
| ABORT | Application Program | IDMSCANC | Should be resolved |

| Reference | Referenced by | Resolved by | Comments |
|---|---|---|---|
| IDCSACON | Application Program | IDMSBALI | Must be resolved; alternatively, include the #BALI macro in the application program if you use the #RETURN macro |
| IDMS | Application Program | IDMS | Must be resolved |
| IDMSOPTI* | IDMS | IDMSOPTI module | Must be resolved under z/OS if using the central version without a SYSCTL file, and under z/VSE if using the central version |
| .IDMSWAIT* | IDMS | IDMSWAIT | Must be resolved if user-written wait program is desired; otherwise, system routine is used |

* Under z/OS , IDMSOPTI is a weak external reference (WXTRN)

# Appendix B: Sample CA IDMS/DB Batch Program

This appendix contains a sample batch Assembler program that accesses database records using navigational DML statements. The sample program shown performs the following:

Performs an area sweep of the ORG-DEMO region for office records

Walks the OFFICE-EMPLOYEE set

- Uses a junction record (EMPLOYEE)
- Walks the DEPT-EMPLOYEE set
- Tests database conditions

This section contains the following topics:

## Input to the Precompiler

The following illustrates a sample batch program as input to the DML precompiler.

```
*RETRIEVAL
*DMLIST
*NO-ACTIVITY-LOG
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         SPACE 1
*                                  ENTER FROM NEXT HIGHER LEVEL
         SPACE 1
         PRINT GEN                 ASSEMBLER PRINT OPTIONS
SYBPG2   CSECT
         LR    R12,R15             ESTABLISHES REGISTER 12 AS THE
         USING SYBPG2,R12           BASE REGISTER
         USING STORAGE,R10         ESTABLISH ADDRESSABILITY OF DSECT
         B     PROCESS             BRANCH TO FIND INVOKING TASKCODE
         EJECT
         @INVOKE MODE=IDMSDC,MAP=SYBMAP
*                                  OPERATING MODE: IDMS DC/MAPPING
         EJECT
         SPACE 1
RETURN   DS    0H
         #FREESTG STGID='SYB4'     FREE THE STORAGE ACQUIRED EARLIER
         #RETURN                   RETURN TO HIGHER LEVEL
         SPACE 1
RETURNXT DS    0H
         #RETURN NXTTASK=SYBTSK03  PASS CONTROL BACK TO ITSELF
         SPACE 1
*                                  MAINLINE PROGRAM
         SPACE 1
PROCESS  DS    0H
         #GETSTG TYPE=(USER,LONG,KEEP),PLIST=*,LEN=STORLGTH,           *
               STGID='SYB4',COND=(ALL),ERROR=ERRORTN,ADDR=(R10),       *
               INIT=X'40'
*                                  ACQUIRE VARIABLE STORAGE
         SPACE 1
         #MAPBIND MRB=SYBMAP       BIND MAP AND RECORDS
         #MAPBIND MRB=SYBMAP,RECNAME=SYBREC
         SPACE 1
ACCEPTSK #ACCEPT TYPE=TASKCODE,FIELD=TASKCODE
*                                  ACCEPT TASK CODE TO INVOKE TASK
         CLC   TASKCODE,SYBTSK2    FIRST TIME CALLED ?
         BNE   RECCUR              YES - OUTPUT FIRST SCREEN
*                                  NO  - INPUT DATA FROM SCREEN
FIRSTIME DS    0H
         MVC   SYBDEPID,=C'0000'   PRIME DATA FIELD
         SPACE
         #MREQ OUT,MRB=SYBMAP,OPTNS=(NEWPAGE),ERROR=ERRORTN,           *
               COND=(ALL)
*                                  MAP OUT PROMPT
         SPACE
         B     RETURNXT            EXIT & WAIT FOR OPERATOR RESPONSE
         SPACE 2
RECCUR   DS    0H
         #MREQ IN,MRB=SYBMAP,ERROR=ERRORTN,COND=(ALL)
*                                  MAP IN TERMINAL INPUT
         SPACE 1
         #MAPINQ MRB=SYBMAP,AID=AIDBYTE
*                                  MOVE MAP DATA TO PROG VARIABLE STG
         CLI   AIDBYTE,CLEAR       DID THE OPERATOR REQUEST FINISH?
         BE    RETURN              YES - EXIT PGM, BACK TO IDMS DC
         SPACE
         #MREQ OUT,MRB=SYBMAP,ERROR=ERRORTN,                            *
               COND=(ALL)
*                                     MAP OUT DATA
         SPACE
         B     RETURNXT               EXIT & WAIT FOR OPERATOR RESPONSE
*                                     NO  - MAPOUT, WAIT ON OPERATOR
ERRORTN  DS    0H                     HERE FOR NONZERO RETURN CODE
         #SNAP AREA=(SYBMAP,SYBMAPLN)
         B     RETURN                 EXIT
CLEAR    EQU   X'6D'                  CLEAR AIDBYTE VALUE
SYBTSK2  DC    CL8'SYBTSK2 '          DC TASK INVOKING VALUE (EXTERNAL)
SYBTSK03 DC    CL8'SYBTSK03'          DC TASK INVOKING VALUE (INTERNAL)
         LTORG
         #BALI
         SPACE 2
*********************************************************************
STORAGE  DSECT                        STORAGE DSECT
         @COPY IDMS,MAP-CONTROL=SYBMAP
SYBMAPLN EQU   *-SYBMAP               LENGTH OF #MRB FOR SNAP
         SPACE 1
         @COPY IDMS,MAP-RECORDS
         SPACE 1
SYSPLIST DS    20F                    MAP OUT PARAMETER LIST AREA
TASKCODE DS    CL8                    TASK CODE WHICH INVOKED PROGRAM
AIDBYTE  DS    X                      ATTENTION IDENTIFIER BYTE
         DS    3X                     RESERVED
STORLGTH EQU   *-STORAGE              TOTAL LENGTH OF STORAGE NEEDED
         SPACE 1
         END   SYBPG2
```

## Output from the Precompiler

The following illustrates the sample batch program as output from the DML precompiler.

```
*DMLIST
SAMPLE1  START         #REGEQU
         STM       R14,R12,12(R13)
         LR        R12,R15
         USING     SAMPLE1,R12,R11,R10
         LR        R11,R12
         LA        R11,4095(R11)
         LA        R11,1(R11)
         LA        R10,4095(R11)
         LA        R10,1(R10)
         ST        R13,SAVEAREA+4
         LA        R7,SAVEAREA
         ST        R7,8(R13)
         LA        R13,SAVEAREA
         B         BEGIN
         @MODE MODE=BATCH,DEBUG=YES
*        @INVOKE SUBSCH=EMPSS01,SCHEMA=EMPSCHM,VERSION=100
*        @COPY IDMS,SUBSCHEMA-CTRL
                                 DS    0D
SSCTRL                           DS    0CL216
PGMNAME                          DC    CL8' '
ERRSTAT                          DC    CL4'1400'
DBKEY                            DS    FL4
RECNAME                          DC    CL16' '
AREANAME                         DC    CL16' '
ERRORSET                         DC    CL16' '
ERRORREC                         DC    CL16' '
ERRAREA                          DC    CL16' '
SSCIDBCM                         DS    0CL100
IDBMSCOM                         DS    100CL1
         ORG   SSCIDBCM
RDBMSCOM                         DS    0CL100
PGINFO                           DS    0CL4
PGINFGRP                         DS    HL2
PGINFDBK                         DS    HL2
                                 DS    CL96
DIRDBKEY                         DC    FL4'0'
DBSTATUS                         DS    0CL8
DBSTMTCD                         DS    CL2
DBSTATCD                         DS    CL5
                                 DS    CL1
RECOCCUR                         DC    FL4'0'
DMLSEQ                           DC    FL4'0'
****************************************
*        @COPY IDMS,SUBSCHEMA-RECORDS
                                 DS    0D
STRUCTUR                         DS    0CL12
STRCODE                          DS    CL2
ADMIN                            EQU   C'A'
PROJECT                          EQU   C'P1'
STRDATE                          DS    0CL8
STRYEAR                          DS    CL4
STRMONTH                         DS    CL2
STRDAY                           DS    CL2
                                 DS    CL2
****************************************
                                 DS    CL4
                                 DS    0D
SKILLA                           DS    0CL76
SKILID                           DS    CL4
SKILNAME                         DS    CL12
SKILDESC                         DS    CL60
****************************************
                                 DS    CL4
                                 DS    0D
OFFIC                            DS    0CL76
OFFCODE                          DS    CL3
OFFADDR                          DS    0CL46
OFFSTRT                          DS    CL20
OFFCITY                          DS    CL15
OFFSTATE                         DS    CL2
OFFZIP                           DS    0CL9
OFFZIPF5                         DS    CL5
```

```
OFFZIPL4                           DS    CL4
OFFPHONE                           DS    3CL7
OFFAREA                            DS    CL3
OFFSPEED                           DS    CL3
******************************************
                                   DS    CL4
                                   DS    0D
NONHSPCL                           DS    0CL1052
NHCLMDT                            DS    0CL8
NHCLMYR                            DS    CL4
NHCLMMO                            DS    CL2
NHCLMDAY                           DS    CL2
NHPTNAME                           DS    0CL25
NHPTFNAM                           DS    CL10
NHPTLNAM                           DS    CL15
NHPTBDAT                           DS    0CL8
NHPTBYR                            DS    CL4
NHPTBMO                            DS    CL2
NHPTBDA                            DS    CL2
NHPTSEX                            DS    CL1
NHRELEMP                           DS    CL10
NHPHYNAM                           DS    0CL25
NHPHYFNM                           DS    CL10
NHPHYLNM                           DS    CL15
NHPHYADD                           DS    0CL46
NHPHYSTR                           DS    CL20
NHPHYCTY                           DS    CL15
NHPHYSTA                           DS    CL2
NHPHYZIP                           DS    0CL9
NHPHYZ5                            DS    CL5
NHPHYZ4                            DS    CL4
NHPHYSID                           DS    CL6
NHDIAGN                            DS    2CL60
NHNOPROC                           DS    HL2
                                   DS    CL1
NHPHYCHG                           DS    0CL800
NHSERVDT                           DS    0CL8
NHSERVYR                           DS    CL4
NHSERVMO                           DS    CL2
NHSERVDA                           DS    CL2
NHPROCCD                           DS    CL4
NHDESCSV                           DS    CL60
NHFEE                              DS    PL5
                                   DS    CL3
                                   DS    CL720
******************************************
                                   DS    CL4
                                   DS    0D
JOBA                               DS    0CL296
JOBID                              DS    CL4
JOBTITLE                           DS    CL20
JOBDESCR                           DS    0CL120
JOBDSCLN                           DS    2CL60
JOBRQMNT                           DS    0CL120
JOBREQLN                           DS    2CL60
JOBMNSAL                           DS    CL8
JOBMXSAL                           DS    CL8
JOBSALGR                           DS    4CL2
JOBNMPOS                           DS    CL3
JOBNMOPN                           DS    CL3
                                   DS    CL2
******************************************
                                   DS    0D
INSPLAN                            DS    0CL132
INPCODE                            DS    CL3
GROUPLIF                           EQU   C'001'
HMO                                EQU   C'002'
GRPHLTH                            EQU   C'003'
GROUPDNT                           EQU   C'004'
INPCNAME                           DS    CL45
INPCADDR                           DS    0CL46
INPCSTRT                           DS    CL20
INPCCITY                           DS    CL15
INPCSTAT                           DS    CL2
```

```
INPCZIP                         DS    0CL9
INPCZPF5                        DS    CL5
INPCZPL4                        DS    CL4
INPCPHON                        DS    CL10
INPGRPNO                        DS    CL6
INPDESCR                        DS    0CL20
INPDEDCT                        DS    PL5
INPMXLIF                        DS    PL5
INPFAMCS                        DS    PL5
INPDEPCS                        DS    PL5
                                DS    CL2
***************************************
                                DS    CL4
                                DS    0D
HOSPCLM                         DS    0CL300
HCCLMDT                         DS    0CL8
HCCLMYR                         DS    CL4
HCCLMMO                         DS    CL2
HCCLMDAY                        DS    CL2
HCPTNAME                        DS    0CL25
HCPTFNAM                        DS    CL10
HCPTLNAM                        DS    CL15
HCPTBDAT                        DS    0CL8
HCPTBYR                         DS    CL4
HCPTBMO                         DS    CL2
HCPTBDA                         DS    CL2
HCPTSEX                         DS    CL1
HCRELEMP                        DS    CL10
HCHSPNAM                        DS    CL25
HCHSPADD                        DS    0CL46
HCHSPSTR                        DS    CL20
HCHSPCTY                        DS    CL15
HCHSPSTA                        DS    CL2
HCHSPZIP                        DS    0CL9
HCHSPZF5                        DS    CL5
HCHSPZL4                        DS    CL4
HCADMTDT                        DS    0CL8
HCADMTYR                        DS    CL4
HCADMTMO                        DS    CL2
HCADMTDA                        DS    CL2
HCDSCGDT                        DS    0CL8
HCDSCGYR                        DS    CL4
HCDSCGMO                        DS    CL2
HCDSCGDA                        DS    CL2
HCDIAGN                         DS    2CL60
HCHSPCHG                        DS    0CL41
HCRMBRD                         DS    0CL26
HCWARD                          DS    0CL13
HCWDDAYS                        DS    PL3
HCWDRATE                        DS    PL5
HCWDTOTL                        DS    PL5
HCSPRIV                         DS    0CL13
HCSDAYS                         DS    PL3
HCSRATE                         DS    PL5
HCSTOTAL                        DS    PL5
HCOTHCHG                        DS    0CL15
HCDELVCH                        DS    PL5
HCANSTHC                        DS    PL5
HCLABCST                        DS    PL5
***************************************
                                DS    CL4
                                DS    0D
EXPRTISE                        DS    0CL12
EXPSKLVL                        DS    CL2
EXPERT                          EQU   C'04'
PROFICNT                        EQU   C'03'
COMPETNT                        EQU   C'02'
ELEMNTRY                        EQU   C'01'
EXPDATE                         DS    0CL8
EXPYEAR                         DS    CL4
EXPMONTH                        DS    CL2
EXPDAY                          DS    CL2
                                DS    CL2
***************************************
```

```
                                  DS    CL4
                                  DS    0D
EMPOSITN                          DS    0CL32
EPSTRTDT                          DS    0CL8
EPSTRTYR                          DS    CL4
EPSTRTMO                          DS    CL2
EPSTRTDA                          DS    CL2
EPFINIDT                          DS    0CL8
EPFINIYR                          DS    CL4
EPFINIMO                          DS    CL2
EPFINIDA                          DS    CL2
EPSALGRD                          DS    CL2
EPSALAMT                          DS    PL5
EPBONPCT                          DS    PL2
EPCMMPCT                          DS    PL2
EPOTRATE                          DS    PL2
                                  DS    CL3
*****************************************
                                  DS    0D
EMPLOYE                           DS    0CL120
EMPID                             DS    CL4
EMPNAME                           DS    0CL25
EMPFNAME                          DS    CL10
EMPLNAME                          DS    CL15
EMPADDR                           DS    0CL46
EMPSTRET                          DS    CL20
EMPCITY                           DS    CL15
EMPSTATE                          DS    CL2
EMPZIP                            DS    0CL9
EMPZIPF5                          DS    CL5
EMPZIPL4                          DS    CL4
EMPPHONE                          DS    CL10
EMPSTATU                          DS    CL2
ACTIVE                            EQU   C'01'
STDSBL                            EQU   C'02'
LTDSBL                            EQU   C'03'
LVOFAB                            EQU   C'04'
TRMINATD                          EQU   C'05'
EMPSSNUM                          DS    CL9
EMPSTDT                           DS    0CL8
EMPSTYR                           DS    CL4
EMPSTMO                           DS    CL2
EMPSTDA                           DS    CL2
EMPTRMDT                          DS    0CL8
EMPTRMYR                          DS    CL4
EMPTRMMO                          DS    CL2
EMPTRMDA                          DS    CL2
EMPBIRDT                          DS    0CL8
EMPBIRYR                          DS    CL4
EMPBIRMO                          DS    CL2
EMPBIRDA                          DS    CL2
*****************************************
                                  DS    0D
DEPARTMT                          DS    0CL56
DEPTID                            DS    CL4
DEPTNAME                          DS    CL45
DEPTHDID                          DS    CL4
                                  DS    CL3
*****************************************
                                  DS    0D
DENTCLM                           DS    0CL932
DCCLMDT                           DS    0CL8
DCCLMYR                           DS    CL4
DCCLMMO                           DS    CL2
DCCLMDA                           DS    CL2
DCPNAME                           DS    0CL25
DCPFNAME                          DS    CL10
DCPLNAME                          DS    CL15
DCPBIRDT                          DS    0CL8
DCPBIRYR                          DS    CL4
DCPBIRMO                          DS    CL2
DCPBIRDA                          DS    CL2
DCPSEX                            DS    CL1
DCRELEMP                          DS    CL10
```

```
DCDNNAME                          DS    0CL25
DCDNFNAM                          DS    CL10
DCDNLNAM                          DS    CL15
DCDNADDR                          DS    0CL46
DCDNSTR                           DS    CL20
DCDNCITY                          DS    CL15
DCDNSTAT                          DS    CL2
DCDNZIP                           DS    0CL9
DCDNZPF5                          DS    CL5
DCDNZPL4                          DS    CL4
DCDNLICN                          DS    CL6
DCNOPROC                          DS    HL2
                                  DS    CL1
DCDNCHGS                          DS    0CL800
DCTOTHNO                          DS    CL2
DCSERVDT                          DS    0CL8
DCSERVYR                          DS    CL4
DCSERVMO                          DS    CL2
DCSERVDA                          DS    CL2
DCPROCCD                          DS    CL4
DCDESCSV                          DS    CL60
DCFEE                             DS    PL5
                                  DS    CL1
                                  DS    CL720
****************************************
                                  DS    CL4
                                  DS    0D
COVERGE                           DS    0CL20
COVSELDT                          DS    0CL8
COVSELYR                          DS    CL4
COVSELMO                          DS    CL2
COVSELDA                          DS    CL2
COVTRMDT                          DS    0CL8
COVTRMYR                          DS    CL4
COVTRMMO                          DS    CL2
COVTRMDA                          DS    CL2
COVTYPE                           DS    CL1
COVMASTR                          EQU   C'M'
COVFAMLY                          EQU   C'F'
COVDPNDT                          EQU   C'D'
COVPLNCD                          DS    CL3
GROUP_LIFE                        EQU   C'001'
HMO                               EQU   C'002'
GROUP_HEALTH                      EQU   C'003'
GROUP_DENTAL                      EQU   C'004'
****************************************
                                  DS    CL4
BEGIN    DS        0F
*        @COPY IDMS,SUBSCHEMA-BINDS
         @BIND SUBSCH='EMPSS01 ',SCB=SSCTRL,DICTNAM='APPLDICT'
         @BIND REC='OFFICE',IOAREA=OFFIC
         @BIND REC='EMPLOYEE',IOAREA=EMPLOYE
         @BIND REC='DEPARTMENT',IOAREA=DEPARTMT
         OPEN  (OUTFILE,OUTPUT)
         MVC   EDSW,=C'N'          SET SWITCHES
         MVC   DSW,=C'N'
         MVC   ESW,=C'N'
         LA    R5,MAIN000          LOAD ADDRESS OF MAINLINE ROUTINE
         B     PRTHEAD
MAIN000  EQU   *
         @READY ALL,RDONLY=YES     READY ALL DATABASE AREAS
         CLC   ERRSTAT,STATOK      CHECK IF ERROR
         BNE   AREAERR             BRANCH TO ERROR ROUTINE
         @OBTAIN FIRST,AREA='ORG-DEMO-REGION',REC='OFFICE'
NEWOFFC  CLC   ERRSTAT,STATOK      CHECK IF NO OFFICE
         BNE   AREAERR
         MVC   OCODE,OFFCODE
         MVC   OCITY,OFFCITY
         @OBTAIN FIRST,SET='OFFICE-EMPLOYEE',REC='EMPLOYEE'
         CLC   ERRSTAT,STATOK      CHECK IF NO EMPLOYEE
         BNE   OBERR1
         MVC   EID,EMPID           MOVE EMPLOYEE ID
         MVC   FNAME,EMPFNAME      MOVE EMPLOYEE FIRST NAME
         MVC   LNAME,EMPLNAME      MOVE EMPLOYEE LAST NAME
         MVC   WALK,EMPID          SAVE ID
         MVC   STATNUM,EMPSTATU    MOVE EMPLOYEE STATUS
         LA    R6,NEWDPT           LOAD ADDRESS OF NEW DEPT ROUTINE
         B     CKSTAT              BRANCH TO STATUS-CHECK RTN
NEWDPT   EQU   *
         @OBTAIN OWNER,SET='DEPT-EMPLOYEE'
         CLC   ERRSTAT,STATOK      CHECK IF DEPARTMENT
         BNE   OBERR2
         MVC   DID,DEPTID
         MVC   DEPT,DEPTNAME
         LA    R5,MAIN020          LOAD ADDRESS OF SET-WALK RTN
         B     PRINTREC            PRINT DEPARTMENT INFORMATION
MAIN020  EQU   *                   *
         @OBTAIN NEXT,SET='DEPT-EMPLOYEE',REC='EMPLOYEE'
         CLC   ERRSTAT,0307        CHECK IF END OF SET
         BE    MAIN030             BRANCH IF END OF SET
         CLC   ERRSTAT,STATOK      CHECK IF ERROR
         BNE   OBERR3
         MVC   EID,EMPID           MOVE EMPLOYEE ID
         MVC   FNAME,EMPFNAME      MOVE EMPLOYEE FIRST NAME
         MVC   LNAME,EMPLNAME      MOVE EMPLOYEE LAST NAME
         MVC   STATNUM,EMPSTATU    MOVE EMPLOYEE STATUS
         LA    R6,MAIN025          LOAD ADDRESS OF PRINT LINK
         B     CKSTAT
MAIN025  EQU   *
         LA    R5,MAIN020
         B     PRINTREC
MAIN030  EQU   *
         MVC   EMPID,WALK
         @FIND CALC,REC='EMPLOYEE' FIND NEXT EMPLOYEE
         CLC   ERRSTAT,STATOK      CHECK IF ERROR
         BNE   CALCERR
REPEAT   EQU   *
         @OBTAIN NEXT,SET='OFFICE-EMPLOYEE',REC='EMPLOYEE'
         CLC   ERRSTAT,=C'0307'    END OF SET ?
         BE    MAIN040             BRANCH IF END OF SET
         CLC   ERRSTAT,STATOK
         BNE   OBERR1
         @IF   SET='DEPT-EMPLOYEE',MEMBER=YES,GOTO=REPEAT
         MVC   EID,EMPID           MOVE EMPLOYEE ID
         MVC   FNAME,EMPFNAME      MOVE EMPLOYEE FIRST NAME
         MVC   LNAME,EMPLNAME      MOVE EMPLOYEE LAST NAME
         MVC   WALK,EMPID
         MVC   STATNUM,EMPSTATU
         LA    R6,NEWDPT           ADDRESS OF DEPT ROUTINE
         B     CKSTAT
MAIN040  EQU   *
         @OBTAIN NEXT,AREA='ORG-DEMO-REGION',REC='OFFICE'
         B     NEWOFFC
EOF      EQU   *
         @FINISH                   *
         CLC   ERRSTAT,STATOK
         BNE   FINERR
         CLOSE (OUTFILE)
         L     R13,SAVEAREA+4
         LM    R14,R12,12(R13)
         BR    R14                 RETURN
*   ERROR ROUTINES                             *
BSERROR  EQU   *
         MVI   ERRMSG,C' '
         MVC   ERRMSG+1(19),ERRMSG
         MVI   ERRNUM,C' '
         MVC   ERRNUM+1(3),ERRNUM
         MVC   ERRNUM,ERRSTAT
         MVC   ERRMSG,BSMSG
         B     PRINTERR
BRERROR  EQU   *
         MVI   ERRMSG,C' '
         MVC   ERRMSG+1(19),ERRMSG
         MVI   ERRNUM,C' '
         MVC   ERRNUM+1(3),ERRNUM
         MVC   ERRNUM,ERRSTAT
         MVC   ERRMSG,BRMSG
         B     PRINTERR
```

```
AREAERR  EQU   *
         MVI   ERRMSG,C' '
         MVC   ERRMSG+1(19),ERRMSG
         MVI   ERRNUM,C' '
         MVC   ERRNUM+1(3),ERRNUM
         MVC   ERRNUM,ERRSTAT
         MVC   ERRMSG,AREAMSG
         B     PRINTERR
CALCERR  EQU   *
         MVI   ERRMSG,C' '
         MVC   ERRMSG+1(19),ERRMSG
         MVI   ERRNUM,C' '
         MVC   ERRNUM+1(3),ERRNUM
         MVC   ERRNUM,ERRSTAT
         MVC   ERRMSG,CALMSG
         B     PRINTERR
FINERR   EQU   *
         MVI   ERRMSG,C' '
         MVC   ERRMSG+1(19),ERRMSG
         MVI   ERRNUM,C' '
         MVC   ERRNUM+1(3),ERRNUM
         MVC   ERRNUM,ERRSTAT
         MVC   ERRMSG,FINMSG
         B     PRINTERR
OBERR1   EQU   *
         MVC   EDSW,=C'Y'
         LA    R5,MAIN040
         B     PRINTREC
OBERR2   EQU   *
         MVC   DSW,=C'Y'
         LA    R5,REPEAT
         B     PRINTREC
OBERR3   EQU   *
         MVC   ESW,=C'Y'
         LA    R5,MAIN030
         B     PRINTREC
*   PRINT ROUTINES
PRINTERR EQU   *
         MVC   ERRLINE,C' '
         MVC   ERRLINE+1(132),ERRLINE
         MVI   ERRLINE,C'0'
         PUT   OUTFILE,ERRLINE
         B     EOF
PRINTREC EQU   *
         MVI   LINE1,C' '
         MVC   LINE1+1(132),LINE1
         MVI   LINE1,C'0'
         MVI   LINE2,C' '
         MVC   LINE2+1(132),LINE2
         CLC   EDSW,=C'Y'
         BE    SKIPED
         CLC   DSW,=C'Y'
         BE    SKIPD
         MVC   LINE1+27(45),DEPT
         MVC   LINE2+27(4),DID
         CLC   DSW,=C'Y'
         BE    SKIPED
SKIPD    EQU   *
         MVC   LINE1+77(27),ENAME
         MVC   LINE2+77(4),EID
         MVC   LINE1+109(20),STAT
SKIPED   EQU   *
         MVC   LINE1+7(15),OCITY
         MVC   LINE2+7(4),OCODE
         PUT   OUTFILE,LINE1
         PUT   OUTFILE,LINE2
         MVC   EDSW,=C'N'
         MVC   DSW,=C'N'
         MVC   ESW,=C'N'
         BR    R5
*   CHECK STATUS ROUTINE           *
CKSTAT   EQU   *
         CLC   STATNUM,=C'01'
         BE    ACT
         CLC   STATNUM,=C'02'
         BE    STD
         CLC   STATNUM,=C'03'
         BE    LTD
         CLC   STATNUM,=C'04'
         BE    LVO
         CLC   STATNUM,=C'05'
         BE    TRM
         MVC   STAT,=C' STATUS CODE ERROR  '
         BR    R6
ACT      EQU   *
         MVC   STAT,=C' ACTIVE             '
         BR    R6
STD      EQU   *
         MVC   STAT,=C' SHORT TERM DISABLED'
         BR    R6
LTD      EQU   *
         MVC   STAT,=C' LONG TERM DISBALED '
         BR    R6
LVO      EQU   *
         MVC   STAT,=C' LEAVE OF ABSENCE   '
         BR    R6
TRM      EQU   *
         MVC   STAT,=C' TERMINATED         '
         BR    R6
*   PRINT REPORT HEADING ROUTINE   *
PRTHEAD  EQU   *
         MVI   LINE1,C' '
         MVC   LINE1+1(132),LINE1
         MVI   LINE1,C'1'
         MVC   LINE1+54(26),HEAD1
         PUT   OUTFILE,LINE1
         MVI   LINE1,C' '
         MVC   LINE1+1(132),LINE1
         MVI   LINE1,C'-'
         MVC   LINE1+6(18),HEAD2O
         MVC   LINE1+26(26),HEAD2D
         MVC   LINE1+76(20),HEAD2E
         MVC   LINE1+108(15),HEAD2S
         PUT   OUTFILE,LINE1
         BR    R5
*
WORKFLDS DC   C'WORK-FIELDS'
SAVEAREA DC   18F'0'
STATNUM  DS    CL2
STAT     DS    CL20
STATOK   DC    CL4'0000'
STATUS   DS    CL2
OCODE    DS    CL3
OCITY    DS    CL15
EID      DS    CL4
ENAME    DS    0CL27
FNAME    DS    CL10
         DS    CL2
LNAME    DS    CL15
WALK     DS    CL4
DID      DS    CL4
DEPT     DS    CL45
ERRLINE  DS    0CL133
         DS    CL1
         DC    CL48'* * * * * * * * * * * * * * * * * * * *   '
         DC    CL6'      '
ERRMSG   DS    CL20
ERRNUM   DS    CL4
         DC    CL6'      '
         DC    CL48'* * * * * * * * * * * * * * * * * * * *   '
         DC    CL5'     '
BSMSG    DC    CL20'BIND SUBSCH ERROR # '
BRMSG    DC    CL20'BIND RECORD ERROR # '
AREAMSG  DC    CL20'READY AREA  ERROR # '
CALMSG   DC    CL20'FIND CALC  ERROR  # '
FINMSG   DC    CL20'@FINISH ERROR     # '
EDSW     DS    CL1
DSW      DS    CL1
```

```
ESW      DS    CL1
LINE1    DS    CL133
LINE2    DS    CL133
HEAD1    DC    CL26'OFFICE  PERSONNEL  LISTING'
HEAD2O   DC    CL18'OFFICE/OFFICE CODE'
HEAD2D   DC    CL26'DEPARTMENT/DEPARTMENT CODE'
HEAD2E   DC    CL20'EMPLOYEE/EMPLOYEE ID'
HEAD2S   DC    CL15'EMPLOYEE STATUS'
*   OUTPUT FILE DCB INFO
OUTFILE  DCB   DDNAME=OUTFILE,MACRF=PM,BLKSIZE=133,LRECL=133,          X
               DSORG=PS
         LTORG
         END   SAMPLE1
```

## Output from the Assembler

The following illustrates the sample batch program as output from the assembler.

```
                    1 *DMLIST
000000                                2 SAMPLE1  START
                                      3          #REGEQU
                                      4+*
                                      5+*        REGISTER EQUATES
                                      6+*
                            00000     7+R0       EQU   0                                                       01-#REGE
                            00001     8+R1       EQU   1                                                       01-#REGE
                            00002     9+R2       EQU   2                                                       01-#REGE
                            00003    10+R3       EQU   3                                                       01-#REGE
                            00004    11+R4       EQU   4                                                       01-#REGE
                            00005    12+R5       EQU   5                                                       01-#REGE
                            00006    13+R6       EQU   6                                                       01-#REGE
                            00007    14+R7       EQU   7                                                       01-#REGE
                            00008    15+R8       EQU   8                                                       01-#REGE
                            00009    16+R9       EQU   9                                                       01-#REGE
                            0000A    17+R10      EQU   10                                                      01-#REGE
                            0000B    18+R11      EQU   11                                                      01-#REGE
                            0000C    19+R12      EQU   12                                                      01-#REGE
                            0000D    20+R13      EQU   13                                                      01-#REGE
                            0000E    21+R14      EQU   14                                                      01-#REGE
                            0000F    22+R15      EQU   15                                                      01-#REGE
000000 90EC D00C            0000C    23          STM      R14,R12,12(R13)
000004 18CF                          24          LR       R12,R15
               R:CBA  00000          25          USING    SAMPLE1,R12,R11,R10
000006 18BC                          26          LR       R11,R12
000008 41BB 0FFF            00FFF    27          LA       R11,4095(R11)
00000C 41BB 0001            00001    28          LA       R11,1(R11)
000010 41AB 0FFF            00FFF    29          LA       R10,4095(R11)
000014 41AA 0001            00001    30          LA       R10,1(R10)
000018 50D0 B410            01410    31          ST       R13,SAVEAREA+4
00001C 4170 B40C            0140C    32          LA       R7,SAVEAREA
000020 507D 0008            00008    33          ST       R7,8(R13)
000024 41D0 B40C            0140C    34          LA       R13,SAVEAREA
000028 47F0 CD58            00D58    35          B        BEGIN
                                     36          @MODE MODE=BATCH,DEBUG=YES
                                     37 *        @INVOKE SUBSCH=EMPSS01,SCHEMA=EMPSCHM,VERSION=100
                                     38 *        @COPY IDMS,SUBSCHEMA-CTRL
000030                               39                                  DS    0D
000030                               40 SSCTRL                           DS    0CL216
000030 4040404040404040              41 PGMNAME                          DC    CL8' '
000038 F1F4F0F0                      42 ERRSTAT                          DC    CL4'1400'
00003C                               43 DBKEY                            DS    FL4
000040 4040404040404040              44 RECNAME                          DC    CL16' '
000050 4040404040404040              45 AREANAME                         DC    CL16' '
000060 4040404040404040              46 ERRORSET                         DC    CL16' '
000070 4040404040404040              47 ERRORREC                         DC    CL16' '
000080 4040404040404040              48 ERRAREA                          DC    CL16' '
000090                               49 SSCIDBCM                         DS    0CL100
000090                               50 IDBMSCOM                         DS    100CL1
0000F4                      00090    51          ORG   SSCIDBCM
000090                               52 RDBMSCOM                         DS    0CL100
000090                               53 PGINFO                           DS    0CL4
000090                               54 PGINFGRP                         DS    HL2
000092                               55 PGINFDBK                         DS    HL2
000094                               56                                  DS    CL96
0000F4 00000000                      57 DIRDBKEY                         DC    FL4'0'
0000F8                               58 DBSTATUS                         DS    0CL8
0000F8                               59 DBSTMTCD                         DS    CL2
0000FA                               60 DBSTATCD                         DS    CL5
0000FF                               61                                  DS    CL1
000100 00000000                      62 RECOCCUR                         DC    FL4'0'
000104 00000000                      63 DMLSEQ                           DC    FL4'0'
                                     64 ****************************************
                                     65 *        @COPY IDMS,SUBSCHEMA-RECORDS
000108                               66                                  DS    0D
000108                               67 STRUCTUR                         DS    0CL12
000108                               68 STRCODE                          DS    CL2
                                     69 ADMIN                            EQU   C'A'
                                     70 PROJECT                          EQU   C'P1'
00010A                               71 STRDATE                          DS    0CL8
00010A                               72 STRYEAR                          DS    CL4
00010E                               73 STRMONTH                         DS    CL2
```

```
000110                         74 STRDAY                             DS    CL2
000112                         75                                    DS    CL2
                               76 *****************************************
000114                         77                                    DS    CL4
000118                         78                                    DS    0D
000118                         79 SKILLA                             DS    0CL76
000118                         80 SKILID                             DS    CL4
00011C                         81 SKILNAME                           DS    CL12
000128                         82 SKILDESC                           DS    CL60
                               83 *****************************************
000164                         84                                    DS    CL4
000168                         85                                    DS    0D
000168                         86 OFFIC                              DS    0CL76
000168                         87 OFFCODE                            DS    CL3
00016B                         88 OFFADDR                            DS    0CL46
00016B                         89 OFFSTRT                            DS    CL20
00017F                         90 OFFCITY                            DS    CL15
00018E                         91 OFFSTATE                           DS    CL2
000190                         92 OFFZIP                             DS    0CL9
000190                         93 OFFZIPF5                           DS    CL5
000195                         94 OFFZIPL4                           DS    CL4
000199                         95 OFFPHONE                           DS    3CL7
0001AE                         96 OFFAREA                            DS    CL3
0001B1                         97 OFFSPEED                           DS    CL3
                               98 *****************************************
0001B4                         99                                    DS    CL4
0001B8                        100                                    DS    0D
0001B8                        101 NONHSPCL                           DS    0CL1052
0001B8                        102 NHCLMDT                            DS    0CL8
0001B8                        103 NHCLMYR                            DS    CL4
0001BC                        104 NHCLMMO                            DS    CL2
0001BE                        105 NHCLMDAY                           DS    CL2
0001C0                        106 NHPTNAME                           DS    0CL25
0001C0                        107 NHPTFNAM                           DS    CL10
0001CA                        108 NHPTLNAM                           DS    CL15
0001D9                        109 NHPTBDAT                           DS    0CL8
0001D9                        110 NHPTBYR                            DS    CL4
0001DD                        111 NHPTBMO                            DS    CL2
0001DF                        112 NHPTBDA                            DS    CL2
0001E1                        113 NHPTSEX                            DS    CL1
0001E2                        114 NHRELEMP                           DS    CL10
0001EC                        115 NHPHYNAM                           DS    0CL25
0001EC                        116 NHPHYFNM                           DS    CL10
0001F6                        117 NHPHYLNM                           DS    CL15
000205                        118 NHPHYADD                           DS    0CL46
000205                        119 NHPHYSTR                           DS    CL20
000219                        120 NHPHYCTY                           DS    CL15
000228                        121 NHPHYSTA                           DS    CL2
00022A                        122 NHPHYZIP                           DS    0CL9
00022A                        123 NHPHYZ5                            DS    CL5
00022F                        124 NHPHYZ4                            DS    CL4
000233                        125 NHPHYSID                           DS    CL6
000239                        126 NHDIAGN                            DS    2CL60
0002B1                        127 NHNOPROC                           DS    HL2
0002B3                        128                                    DS    CL1
0002B4                        129 NHPHYCHG                           DS    0CL800
0002B4                        130 NHSERVDT                           DS    0CL8
0002B4                        131 NHSERVYR                           DS    CL4
0002B8                        132 NHSERVMO                           DS    CL2
0002BA                        133 NHSERVDA                           DS    CL2
0002BC                        134 NHPROCCD                           DS    CL4
0002C0                        135 NHDESCSV                           DS    CL60
0002FC                        136 NHFEE                              DS    PL5
000301                        137                                    DS    CL3
000304                        138                                    DS    CL720
                              139 *****************************************
0005D4                        140                                    DS    CL4
0005D8                        141                                    DS    0D
0005D8                        142 JOBA                               DS    0CL296
0005D8                        143 JOBID                              DS    CL4
0005DC                        144 JOBTITLE                           DS    CL20
0005F0                        145 JOBDESCR                           DS    0CL120
0005F0                        146 JOBDSCLN                           DS    2CL60
000668                        147 JOBRQMNT                           DS    0CL120
```

```
000668                          148 JOBREQLN                           DS    2CL60
0006E0                          149 JOBMNSAL                           DS    CL8
0006E8                          150 JOBMXSAL                           DS    CL8
0006F0                          151 JOBSALGR                           DS    4CL2
0006F8                          152 JOBNMPOS                           DS    CL3
0006FB                          153 JOBNMOPN                           DS    CL3
0006FE                          154                                    DS    CL2
                                155 ***********************************************
000700                          156                                    DS    0D
000700                          157 INSPLAN                            DS    0CL132
000700                          158 INPCODE                            DS    CL3
                                159 GROUPLIF                           EQU   C'001'
                                160 HMO                                EQU   C'002'
                                161 GRPHLTH                            EQU   C'003'
                                162 GROUPDNT                           EQU   C'004'
000703                          163 INPCNAME                           DS    CL45
000730                          164 INPCADDR                           DS    0CL46
000730                          165 INPCSTRT                           DS    CL20
000744                          166 INPCCITY                           DS    CL15
000753                          167 INPCSTAT                           DS    CL2
000755                          168 INPCZIP                            DS    0CL9
000755                          169 INPCZPF5                           DS    CL5
00075A                          170 INPCZPL4                           DS    CL4
00075E                          171 INPCPHON                           DS    CL10
000768                          172 INPGRPNO                           DS    CL6
00076E                          173 INPDESCR                           DS    0CL20
00076E                          174 INPDEDCT                           DS    PL5
000773                          175 INPMXLIF                           DS    PL5
000778                          176 INPFAMCS                           DS    PL5
00077D                          177 INPDEPCS                           DS    PL5
000782                          178                                    DS    CL2
                                179 ***********************************************
000784                          180                                    DS    CL4
000788                          181                                    DS    0D
000788                          182 HOSPCLM                            DS    0CL300
000788                          183 HCCLMDT                            DS    0CL8
000788                          184 HCCLMYR                            DS    CL4
00078C                          185 HCCLMMO                            DS    CL2
00078E                          186 HCCLMDAY                           DS    CL2
000790                          187 HCPTNAME                           DS    0CL25
000790                          188 HCPTFNAM                           DS    CL10
00079A                          189 HCPTLNAM                           DS    CL15
0007A9                          190 HCPTBDAT                           DS    0CL8
0007A9                          191 HCPTBYR                            DS    CL4
0007AD                          192 HCPTBMO                            DS    CL2
0007AF                          193 HCPTBDA                            DS    CL2
0007B1                          194 HCPTSEX                            DS    CL1
0007B2                          195 HCRELEMP                           DS    CL10
0007BC                          196 HCHSPNAM                           DS    CL25
0007D5                          197 HCHSPADD                           DS    0CL46
0007D5                          198 HCHSPSTR                           DS    CL20
0007E9                          199 HCHSPCTY                           DS    CL15
0007F8                          200 HCHSPSTA                           DS    CL2
0007FA                          201 HCHSPZIP                           DS    0CL9
0007FA                          202 HCHSPZF5                           DS    CL5
0007FF                          203 HCHSPZL4                           DS    CL4
000803                          204 HCADMTDT                           DS    0CL8
000803                          205 HCADMTYR                           DS    CL4
000807                          206 HCADMTMO                           DS    CL2
000809                          207 HCADMTDA                           DS    CL2
00080B                          208 HCDSCGDT                           DS    0CL8
00080B                          209 HCDSCGYR                           DS    CL4
00080F                          210 HCDSCGMO                           DS    CL2
000811                          211 HCDSCGDA                           DS    CL2
000813                          212 HCDIAGN                            DS    2CL60
00088B                          213 HCHSPCHG                           DS    0CL41
00088B                          214 HCRMBRD                            DS    0CL26
00088B                          215 HCWARD                             DS    0CL13
00088B                          216 HCWDDAYS                           DS    PL3
00088E                          217 HCWDRATE                           DS    PL5
000893                          218 HCWDTOTL                           DS    PL5
000898                          219 HCSPRIV                            DS    0CL13
000898                          220 HCSDAYS                            DS    PL3
00089B                          221 HCSRATE                            DS    PL5
```

```
0008A0                         222 HCSTOTAL                     DS    PL5
0008A5                         223 HCOTHCHG                     DS    0CL15
0008A5                         224 HCDELVCH                     DS    PL5
0008AA                         225 HCANSTHC                     DS    PL5
0008AF                         226 HCLABCST                     DS    PL5
                               227 ****************************************
0008B4                         228                              DS    CL4
0008B8                         229                              DS    0D
0008B8                         230 EXPRTISE                     DS    0CL12
0008B8                         231 EXPSKLVL                     DS    CL2
                               232 EXPERT                       EQU   C'04'
                               233 PROFICNT                     EQU   C'03'
                               234 COMPETNT                     EQU   C'02'
                               235 ELEMNTRY                     EQU   C'01'
0008BA                         236 EXPDATE                      DS    0CL8
0008BA                         237 EXPYEAR                      DS    CL4
0008BE                         238 EXPMONTH                     DS    CL2
0008C0                         239 EXPDAY                       DS    CL2
0008C2                         240                              DS    CL2
                               241 ****************************************
0008C4                         242                              DS    CL4
0008C8                         243                              DS    0D
0008C8                         244 EMPOSITN                     DS    0CL32
0008C8                         245 EPSTRTDT                     DS    0CL8
0008C8                         246 EPSTRTYR                     DS    CL4
0008CC                         247 EPSTRTMO                     DS    CL2
0008CE                         248 EPSTRTDA                     DS    CL2
0008D0                         249 EPFINIDT                     DS    0CL8
0008D0                         250 EPFINIYR                     DS    CL4
0008D4                         251 EPFINIMO                     DS    CL2
0008D6                         252 EPFINIDA                     DS    CL2
0008D8                         253 EPSALGRD                     DS    CL2
0008DA                         254 EPSALAMT                     DS    PL5
0008DF                         255 EPBONPCT                     DS    PL2
0008E1                         256 EPCMMPCT                     DS    PL2
0008E3                         257 EPOTRATE                     DS    PL2
0008E5                         258                              DS    CL3
                               259 ****************************************
0008E8                         260                              DS    0D
0008E8                         261 EMPLOYE                      DS    0CL120
0008E8                         262 EMPID                        DS    CL4
0008EC                         263 EMPNAME                      DS    0CL25
0008EC                         264 EMPFNAME                     DS    CL10
0008F6                         265 EMPLNAME                     DS    CL15
000905                         266 EMPADDR                      DS    0CL46
000905                         267 EMPSTRET                     DS    CL20
000919                         268 EMPCITY                      DS    CL15
000928                         269 EMPSTATE                     DS    CL2
00092A                         270 EMPZIP                       DS    0CL9
00092A                         271 EMPZIPF5                     DS    CL5
00092F                         272 EMPZIPL4                     DS    CL4
000933                         273 EMPPHONE                     DS    CL10
00093D                         274 EMPSTATU                     DS    CL2
                               275 ACTIVE                       EQU   C'01'
                               276 STDSBL                       EQU   C'02'
                               277 LTDSBL                       EQU   C'03'
                               278 LVOFAB                       EQU   C'04'
                               279 TRMINATD                     EQU   C'05'
00093F                         280 EMPSSNUM                     DS    CL9
000948                         281 EMPSTDT                      DS    0CL8
000948                         282 EMPSTYR                      DS    CL4
00094C                         283 EMPSTMO                      DS    CL2
00094E                         284 EMPSTDA                      DS    CL2
000950                         285 EMPTRMDT                     DS    0CL8
000950                         286 EMPTRMYR                     DS    CL4
000954                         287 EMPTRMMO                     DS    CL2
000956                         288 EMPTRMDA                     DS    CL2
000958                         289 EMPBIRDT                     DS    0CL8
000958                         290 EMPBIRYR                     DS    CL4
00095C                         291 EMPBIRMO                     DS    CL2
00095E                         292 EMPBIRDA                     DS    CL2
                               293 ****************************************
000960                         294                              DS    0D
000960                         295 DEPARTMT                     DS    0CL56
```

```
000960                              296 DEPTID                           DS    CL4
000964                              297 DEPTNAME                         DS    CL45
000991                              298 DEPTHDID                         DS    CL4
000995                              299                                  DS    CL3
                                    300 ****************************************
000998                              301                                  DS    0D
000998                              302 DENTCLM                          DS    0CL932
000998                              303 DCCLMDT                          DS    0CL8
000998                              304 DCCLMYR                          DS    CL4
00099C                              305 DCCLMMO                          DS    CL2
00099E                              306 DCCLMDA                          DS    CL2
0009A0                              307 DCPNAME                          DS    0CL25
0009A0                              308 DCPFNAME                         DS    CL10
0009AA                              309 DCPLNAME                         DS    CL15
0009B9                              310 DCPBIRDT                         DS    0CL8
0009B9                              311 DCPBIRYR                         DS    CL4
0009BD                              312 DCPBIRMO                         DS    CL2
0009BF                              313 DCPBIRDA                         DS    CL2
0009C1                              314 DCPSEX                           DS    CL1
0009C2                              315 DCRELEMP                         DS    CL10
0009CC                              316 DCDNNAME                         DS    0CL25
0009CC                              317 DCDNFNAM                         DS    CL10
0009D6                              318 DCDNLNAM                         DS    CL15
0009E5                              319 DCDNADDR                         DS    0CL46
0009E5                              320 DCDNSTR                          DS    CL20
0009F9                              321 DCDNCITY                         DS    CL15
000A08                              322 DCDNSTAT                         DS    CL2
000A0A                              323 DCDNZIP                          DS    0CL9
000A0A                              324 DCDNZPF5                         DS    CL5
000A0F                              325 DCDNZPL4                         DS    CL4
000A13                              326 DCDNLICN                         DS    CL6
000A19                              327 DCNOPROC                         DS    HL2
000A1B                              328                                  DS    CL1
000A1C                              329 DCDNCHGS                         DS    0CL800
000A1C                              330 DCTOTHNO                         DS    CL2
000A1E                              331 DCSERVDT                         DS    0CL8
000A1E                              332 DCSERVYR                         DS    CL4
000A22                              333 DCSERVMO                         DS    CL2
000A24                              334 DCSERVDA                         DS    CL2
000A26                              335 DCPROCCD                         DS    CL4
000A2A                              336 DCDESCSV                         DS    CL60
000A66                              337 DCFEE                            DS    PL5
000A6B                              338                                  DS    CL1
000A6C                              339                                  DS    CL720
                                    340 ****************************************
000D3C                              341                                  DS    CL4
000D40                              342                                  DS    0D
000D40                              343 COVERGE                          DS    0CL20
000D40                              344 COVSELDT                         DS    0CL8
000D40                              345 COVSELYR                         DS    CL4
000D44                              346 COVSELMO                         DS    CL2
000D46                              347 COVSELDA                         DS    CL2
000D48                              348 COVTRMDT                         DS    0CL8
000D48                              349 COVTRMYR                         DS    CL4
000D4C                              350 COVTRMMO                         DS    CL2
000D4E                              351 COVTRMDA                         DS    CL2
000D50                              352 COVTYPE                          DS    CL1
                                    353 COVMASTR                         EQU   C'M'
                                    354 COVFAMLY                         EQU   C'F'
                                    355 COVDPNDT                         EQU   C'D'
000D51                              356 COVPLNCD                         DS    CL3
                                    357 GROUP_LIFE                       EQU   C'001'
                                    358 HMO                              EQU   C'002'
                                    359 GROUP_HEALTH                     EQU   C'003'
                                    360 GROUP_DENTAL                     EQU   C'004'
                                    361 ****************************************
000D54                              362                                  DS    CL4
000D58                              363 BEGIN    DS        0F
                                    364 *        @COPY IDMS,SUBSCHEMA-BINDS
                                    365          @BIND SUBSCH='EMPSS01 ',SCB=SSCTRL,DICTNAM='APPLDICT'
                                    366+*                               *** BEGIN DML EXPANSION ***
000D58 4100 C030            00030   367+         LA    0,SSCTRL                                                  02-@IDMS
000D5C 5000 C094            00094   368+         ST    0,SSCIDBCM+4                                              02-@IDMS
000D60 4100 C0CA            000CA   369+         LA    0,SSCIDBCM+59-1                                           02-@IDMS
```

```
000D64 5000 C098           00098   370+         ST    0,SSCIDBCM+8                                          02-@IDMS
000D68 4100 C030           00030   371+         LA    0,SSCTRL                                              02-@IDMS
000D6C 5000 C09C           0009C   372+         ST    0,SSCIDBCM+12                                         02-@IDMS
000D70 4100 B834           01834   373+         LA    0,=CL18'EMPSS01 '                                     02-@IDMS
000D74 5000 C0A0           000A0   374+         ST    0,SSCIDBCM+16                                         02-@IDMS
000D78 4100 C030           00030   375+         LA    0,SSCTRL                                              02-@IDMS
000D7C 5000 C0A4           000A4   376+         ST    0,SSCIDBCM+20                                         02-@IDMS
000D80 4100 C030           00030   377+         LA    0,SSCTRL                                              02-@IDMS
000D84 5000 C0A8           000A8   378+         ST    0,SSCIDBCM+24                                         02-@IDMS
000D88 D207 C050 B7A0 00050 017A0  379+         MVC   AREANAME(8),=CL8' '                                   01-@BIND
000D8E D207 C058 B7A8 00058 017A8  380+         MVC   AREANAME+8(8),=CL8'APPLDICT'                          01-@BIND
000D94 4100 C050           00050   381+         LA    0,AREANAME                                            02-@IDMS
000D98 5000 C0AC           000AC   382+         ST    0,SSCIDBCM+28                                         02-@IDMS
000D9C 9680 C0AC      000AC        383+         OI    SSCIDBCM+28,X'80'                                     02-@IDMS
000DA0 4100 0001           00001   384+         LA    0,1                                                   02-@IDMS
000DA4 5000 C104           00104   385+         ST    0,DMLSEQ                                              02-@IDMS
                                   386+*,                         DML-SEQUENCE = 1                          02-@IDMS
000DA8 4110 C094           00094   387+         LA    1,SSCIDBCM+4                                          02-@IDMS
000DAC 58F0 B7B0           017B0   388+         L     15,=V(IDMS)                                           02-@IDMS
000DB0 05EF                        389+         BALR  14,15              *** CALL IDMS MODE=BATCH ***       02-@IDMS
                                   390+*                                 *** END DML EXPANSION ***
                                   391          @BIND REC='OFFICE',IOAREA=OFFIC
                                   392+*                                 *** BEGIN DML EXPANSION ***
000DB2 4100 C030           00030   393+         LA    0,SSCTRL                                              02-@IDMS
000DB6 5000 C094           00094   394+         ST    0,SSCIDBCM+4                                          02-@IDMS
000DBA 4100 C0BF           000BF   395+         LA    0,SSCIDBCM+48-1                                       02-@IDMS
000DBE 5000 C098           00098   396+         ST    0,SSCIDBCM+8                                          02-@IDMS
000DC2 4100 B846           01846   397+         LA    0,=CL18'OFFICE'                                       02-@IDMS
000DC6 5000 C09C           0009C   398+         ST    0,SSCIDBCM+12                                         02-@IDMS
000DCA 4100 C168           00168   399+         LA    0,OFFIC                                               02-@IDMS
000DCE 5000 C0A0           000A0   400+         ST    0,SSCIDBCM+16                                         02-@IDMS
000DD2 9680 C0A0      000A0        401+         OI    SSCIDBCM+16,X'80'                                     02-@IDMS
000DD6 4100 0002           00002   402+         LA    0,2                                                   02-@IDMS
000DDA 5000 C104           00104   403+         ST    0,DMLSEQ                                              02-@IDMS
                                   404+*,                         DML-SEQUENCE = 2                          02-@IDMS
000DDE 4110 C094           00094   405+         LA    1,SSCIDBCM+4                                          02-@IDMS
000DE2 58F0 B7B0           017B0   406+         L     15,=V(IDMS)                                           02-@IDMS
000DE6 05EF                        407+         BALR  14,15              *** CALL IDMS MODE=BATCH ***       02-@IDMS
                                   408+*                                 *** END DML EXPANSION ***
                                   409          @BIND REC='EMPLOYEE',IOAREA=EMPLOYE
                                   410+*                                 *** BEGIN DML EXPANSION ***
000DE8 4100 C030           00030   411+         LA    0,SSCTRL                                              02-@IDMS
000DEC 5000 C094           00094   412+         ST    0,SSCIDBCM+4                                          02-@IDMS
000DF0 4100 C0BF           000BF   413+         LA    0,SSCIDBCM+48-1                                       02-@IDMS
000DF4 5000 C098           00098   414+         ST    0,SSCIDBCM+8                                          02-@IDMS
000DF8 4100 B858           01858   415+         LA    0,=CL18'EMPLOYEE'                                     02-@IDMS
000DFC 5000 C09C           0009C   416+         ST    0,SSCIDBCM+12                                         02-@IDMS
000E00 4100 C8E8           008E8   417+         LA    0,EMPLOYE                                             02-@IDMS
000E04 5000 C0A0           000A0   418+         ST    0,SSCIDBCM+16                                         02-@IDMS
000E08 9680 C0A0      000A0        419+         OI    SSCIDBCM+16,X'80'                                     02-@IDMS
000E0C 4100 0003           00003   420+         LA    0,3                                                   02-@IDMS
000E10 5000 C104           00104   421+         ST    0,DMLSEQ                                              02-@IDMS
                                   422+*,                         DML-SEQUENCE = 3                          02-@IDMS
000E14 4110 C094           00094   423+         LA    1,SSCIDBCM+4                                          02-@IDMS
000E18 58F0 B7B0           017B0   424+         L     15,=V(IDMS)                                           02-@IDMS
000E1C 05EF                        425+         BALR  14,15              *** CALL IDMS MODE=BATCH ***       02-@IDMS
                                   426+*                                 *** END DML EXPANSION ***
                                   427          @BIND REC='DEPARTMENT',IOAREA=DEPARTMT
                                   428+*                                 *** BEGIN DML EXPANSION ***
000E1E 4100 C030           00030   429+         LA    0,SSCTRL                                              02-@IDMS
000E22 5000 C094           00094   430+         ST    0,SSCIDBCM+4                                          02-@IDMS
000E26 4100 C0BF           000BF   431+         LA    0,SSCIDBCM+48-1                                       02-@IDMS
000E2A 5000 C098           00098   432+         ST    0,SSCIDBCM+8                                          02-@IDMS
000E2E 4100 B86A           0186A   433+         LA    0,=CL18'DEPARTMENT'                                   02-@IDMS
000E32 5000 C09C           0009C   434+         ST    0,SSCIDBCM+12                                         02-@IDMS
000E36 4100 C960           00960   435+         LA    0,DEPARTMT                                            02-@IDMS
000E3A 5000 C0A0           000A0   436+         ST    0,SSCIDBCM+16                                         02-@IDMS
000E3E 9680 C0A0      000A0        437+         OI    SSCIDBCM+16,X'80'                                     02-@IDMS
000E42 4100 0004           00004   438+         LA    0,4                                                   02-@IDMS
000E46 5000 C104           00104   439+         ST    0,DMLSEQ                                              02-@IDMS
                                   440+*,                         DML-SEQUENCE = 4                          02-@IDMS
000E4A 4110 C094           00094   441+         LA    1,SSCIDBCM+4                                          02-@IDMS
000E4E 58F0 B7B0           017B0   442+         L     15,=V(IDMS)                                           02-@IDMS
000E52 05EF                        443+         BALR  14,15              *** CALL IDMS MODE=BATCH ***       02-@IDMS
```

```
                                  444+*                                 *** END DML EXPANSION ***
                                  445          OPEN  (OUTFILE,OUTPUT)
000E54                            446+         CNOP  0,4                          ALIGN LIST TO FULLWORD      01-OPEN
000E54 4510 CE5C           00E5C  447+         BAL   1,*+8                        LOAD REG1 W/LIST ADDR. @L2A 01-OPEN
000E58 8F                         448+         DC    AL1(143)                     OPTION BYTE                 01-OPEN
000E59 00173C                     449+         DC    AL3(OUTFILE)                 DCB ADDRESS                 01-OPEN
000E5C 0A13                       450+         SVC   19                           ISSUE OPEN SVC              01-OPEN
000E5E D200 B5C4 B8BC 015C4 018BC 451          MVC   EDSW,=C'N'          SET SWITCHES
000E64 D200 B5C5 B8BC 015C5 018BC 452          MVC   DSW,=C'N'
000E6A D200 B5C6 B8BC 015C6 018BC 453          MVC   ESW,=C'N'
000E70 4150 CE78           00E78  454          LA    R5,MAIN000          LOAD ADDRESS OF MAINLINE ROUTINE
000E74 47F0 B3A4           013A4  455          B     PRTHEAD
                           00E78  456 MAIN000  EQU   *
                                  457          @READY ALL,RDONLY=YES     READY ALL DATABASE AREAS
                                  458+*                                 *** BEGIN DML EXPANSION ***
000E78 4100 C030           00030  459+         LA    0,SSCTRL                                                 02-@IDMS
000E7C 5000 C094           00094  460+         ST    0,SSCIDBCM+4                                             02-@IDMS
000E80 4100 C0B4           000B4  461+         LA    0,SSCIDBCM+37-1                                          02-@IDMS
000E84 5000 C098           00098  462+         ST    0,SSCIDBCM+8                                             02-@IDMS
000E88 9680 C098     00098        463+         OI    SSCIDBCM+8,X'80'                                         02-@IDMS
000E8C 4100 0005           00005  464+         LA    0,5                                                      02-@IDMS
000E90 5000 C104           00104  465+         ST    0,DMLSEQ                                                 02-@IDMS
                                  466+*,                        DML-SEQUENCE = 5                              02-@IDMS
000E94 4110 C094           00094  467+         LA    1,SSCIDBCM+4                                             02-@IDMS
000E98 58F0 B7B0           017B0  468+         L     15,=V(IDMS)                                              02-@IDMS
000E9C 05EF                       469+         BALR  14,15              *** CALL IDMS MODE=BATCH ***          02-@IDMS
                                  470+*                                 *** END DML EXPANSION ***
000E9E D503 C038 B46A 00038 0146A 471          CLC   ERRSTAT,STATOK      CHECK IF ERROR
000EA4 4770 B1F4           011F4  472          BNE   AREAERR             BRANCH TO ERROR ROUTINE
                                  473          @OBTAIN FIRST,AREA='ORG-DEMO-REGION',REC='OFFICE'
                                  474+*                                 *** BEGIN DML EXPANSION ***
000EA8 4100 C030           00030  475+         LA    0,SSCTRL                                                 03-@IDMS
000EAC 5000 C094           00094  476+         ST    0,SSCIDBCM+4                                             03-@IDMS
000EB0 4100 C0A2           000A2  477+         LA    0,SSCIDBCM+18+1-1                                        03-@IDMS
000EB4 5000 C098           00098  478+         ST    0,SSCIDBCM+8                                             03-@IDMS
000EB8 4100 B846           01846  479+         LA    0,=CL18'OFFICE'                                          03-@IDMS
000EBC 5000 C09C           0009C  480+         ST    0,SSCIDBCM+12                                            03-@IDMS
000EC0 4100 B87C           0187C  481+         LA    0,=CL18'ORG-DEMO-REGION'                                 03-@IDMS
000EC4 5000 C0A0           000A0  482+         ST    0,SSCIDBCM+16                                            03-@IDMS
000EC8 4100 C0BA           000BA  483+         LA    0,SSCIDBCM+43-1                                          02-@IDMS
000ECC 5000 C0A4           000A4  484+         ST    0,SSCIDBCM+20                                            02-@IDMS
000ED0 9680 C0A4     000A4        485+         OI    SSCIDBCM+20,X'80'                                        02-@IDMS
000ED4 4100 0006           00006  486+         LA    0,6                                                      02-@IDMS
000ED8 5000 C104           00104  487+         ST    0,DMLSEQ                                                 02-@IDMS
                                  488+*,                        DML-SEQUENCE = 6                              02-@IDMS
000EDC 4110 C094           00094  489+         LA    1,SSCIDBCM+4                                             02-@IDMS
000EE0 58F0 B7B0           017B0  490+         L     15,=V(IDMS)                                              02-@IDMS
000EE4 05EF                       491+         BALR  14,15              *** CALL IDMS MODE=BATCH ***          02-@IDMS
                                  492+*                                 *** END DML EXPANSION ***
000EE6 D503 C038 B46A 00038 0146A 493 NEWOFFC  CLC   ERRSTAT,STATOK      CHECK IF NO OFFICE
000EEC 4770 B1F4           011F4  494          BNE   AREAERR
000EF0 D202 B470 C168 01470 00168 495          MVC   OCODE,OFFCODE
000EF6 D20E B473 C17F 01473 0017F 496          MVC   OCITY,OFFCITY
                                  497          @OBTAIN FIRST,SET='OFFICE-EMPLOYEE',REC='EMPLOYEE'
                                  498+*                                 *** BEGIN DML EXPANSION ***
000EFC 4100 C030           00030  499+         LA    0,SSCTRL                                                 03-@IDMS
000F00 5000 C094           00094  500+         ST    0,SSCIDBCM+4                                             03-@IDMS
000F04 4100 C0A1           000A1  501+         LA    0,SSCIDBCM+18+0-1                                        03-@IDMS
000F08 5000 C098           00098  502+         ST    0,SSCIDBCM+8                                             03-@IDMS
000F0C 4100 B858           01858  503+         LA    0,=CL18'EMPLOYEE'                                        03-@IDMS
000F10 5000 C09C           0009C  504+         ST    0,SSCIDBCM+12                                            03-@IDMS
000F14 4100 B88E           0188E  505+         LA    0,=CL18'OFFICE-EMPLOYEE'                                 03-@IDMS
000F18 5000 C0A0           000A0  506+         ST    0,SSCIDBCM+16                                            03-@IDMS
000F1C 4100 C0BA           000BA  507+         LA    0,SSCIDBCM+43-1                                          02-@IDMS
000F20 5000 C0A4           000A4  508+         ST    0,SSCIDBCM+20                                            02-@IDMS
000F24 9680 C0A4     000A4        509+         OI    SSCIDBCM+20,X'80'                                        02-@IDMS
000F28 4100 0007           00007  510+         LA    0,7                                                      02-@IDMS
000F2C 5000 C104           00104  511+         ST    0,DMLSEQ                                                 02-@IDMS
                                  512+*,                        DML-SEQUENCE = 7                              02-@IDMS
000F30 4110 C094           00094  513+         LA    1,SSCIDBCM+4                                             02-@IDMS
000F34 58F0 B7B0           017B0  514+         L     15,=V(IDMS)                                              02-@IDMS
000F38 05EF                       515+         BALR  14,15              *** CALL IDMS MODE=BATCH ***          02-@IDMS
                                  516+*                                 *** END DML EXPANSION ***
000F3A D503 C038 B46A 00038 0146A 517          CLC   ERRSTAT,STATOK      CHECK IF NO EMPLOYEE
```

```
000F40 4770 B260            01260   518          BNE   OBERR1
000F44 D203 B482 C8E8 01482 008E8   519          MVC   EID,EMPID           MOVE EMPLOYEE ID
000F4A D209 B486 C8EC 01486 008EC   520          MVC   FNAME,EMPFNAME      MOVE EMPLOYEE FIRST NAME
000F50 D20E B492 C8F6 01492 008F6   521          MVC   LNAME,EMPLNAME      MOVE EMPLOYEE LAST NAME
000F56 D203 B4A1 C8E8 014A1 008E8   522          MVC   WALK,EMPID          SAVE ID
000F5C D201 B454 C93D 01454 0093D   523          MVC   STATNUM,EMPSTATU    MOVE EMPLOYEE STATUS
000F62 4160 CF6A            00F6A   524          LA    R6,NEWDPT           LOAD ADDRESS OF NEW DEPT ROUTINE
000F66 47F0 B342            01342   525          B     CKSTAT              BRANCH TO STATUS-CHECK RTN
                            00F6A   526 NEWDPT   EQU   *
                                    527          @OBTAIN OWNER,SET='DEPT-EMPLOYEE'
                                    528+*                                 *** BEGIN DML EXPANSION ***
000F6A 4100 C030            00030   529+         LA    0,SSCTRL                                         03-@IDMS
000F6E 5000 C094            00094   530+         ST    0,SSCIDBCM+4                                     03-@IDMS
000F72 4100 C0AE            000AE   531+         LA    0,SSCIDBCM+31-1                                  03-@IDMS
000F76 5000 C098            00098   532+         ST    0,SSCIDBCM+8                                     03-@IDMS
000F7A 4100 B8A0            018A0   533+         LA    0,=CL18'DEPT-EMPLOYEE'                           03-@IDMS
000F7E 5000 C09C            0009C   534+         ST    0,SSCIDBCM+12                                    03-@IDMS
000F82 4100 C0BA            000BA   535+         LA    0,SSCIDBCM+43-1                                  02-@IDMS
000F86 5000 C0A0            000A0   536+         ST    0,SSCIDBCM+16                                    02-@IDMS
000F8A 9680 C0A0      000A0         537+         OI    SSCIDBCM+16,X'80'                                02-@IDMS
000F8E 4100 0008            00008   538+         LA    0,8                                              02-@IDMS
000F92 5000 C104            00104   539+         ST    0,DMLSEQ                                         02-@IDMS
                                    540+*,                         DML-SEQUENCE = 8                     02-@IDMS
000F96 4110 C094            00094   541+         LA    1,SSCIDBCM+4                                     02-@IDMS
000F9A 58F0 B7B0            017B0   542+         L     15,=V(IDMS)                                      02-@IDMS
000F9E 05EF                         543+         BALR  14,15              *** CALL IDMS MODE=BATCH ***          02-@IDMS
                                    544+*                                 *** END DML EXPANSION ***
000FA0 D503 C038 B46A 00038 0146A   545          CLC   ERRSTAT,STATOK      CHECK IF DEPARTMENT
000FA6 4770 B26E            0126E   546          BNE   OBERR2
000FAA D203 B4A5 C960 014A5 00960   547          MVC   DID,DEPTID
000FB0 D22C B4A9 C964 014A9 00964   548          MVC   DEPT,DEPTNAME
000FB6 4150 CFBE            00FBE   549          LA    R5,MAIN020          LOAD ADDRESS OF SET-WALK RTN
000FBA 47F0 B2AE            012AE   550          B     PRINTREC            PRINT DEPARTMENT INFORMATION
                            00FBE   551 MAIN020  EQU   *                   *
                                    552          @OBTAIN NEXT,SET='DEPT-EMPLOYEE',REC='EMPLOYEE'
                                    553+*                                 *** BEGIN DML EXPANSION ***
000FBE 4100 C030            00030   554+         LA    0,SSCTRL                                         03-@IDMS
000FC2 5000 C094            00094   555+         ST    0,SSCIDBCM+4                                     03-@IDMS
000FC6 4100 C099            00099   556+         LA    0,SSCIDBCM+10+0-1                                03-@IDMS
000FCA 5000 C098            00098   557+         ST    0,SSCIDBCM+8                                     03-@IDMS
000FCE 4100 B858            01858   558+         LA    0,=CL18'EMPLOYEE'                                03-@IDMS
000FD2 5000 C09C            0009C   559+         ST    0,SSCIDBCM+12                                    03-@IDMS
000FD6 4100 B8A0            018A0   560+         LA    0,=CL18'DEPT-EMPLOYEE'                           03-@IDMS
000FDA 5000 C0A0            000A0   561+         ST    0,SSCIDBCM+16                                    03-@IDMS
000FDE 4100 C0BA            000BA   562+         LA    0,SSCIDBCM+43-1                                  02-@IDMS
000FE2 5000 C0A4            000A4   563+         ST    0,SSCIDBCM+20                                    02-@IDMS
000FE6 9680 C0A4      000A4         564+         OI    SSCIDBCM+20,X'80'                                02-@IDMS
000FEA 4100 0009            00009   565+         LA    0,9                                              02-@IDMS
000FEE 5000 C104            00104   566+         ST    0,DMLSEQ                                         02-@IDMS
                                    567+*,                         DML-SEQUENCE = 9                     02-@IDMS
000FF2 4110 C094            00094   568+         LA    1,SSCIDBCM+4                                     02-@IDMS
000FF6 58F0 B7B0            017B0   569+         L     15,=V(IDMS)                                      02-@IDMS
000FFA 05EF                         570+         BALR  14,15              *** CALL IDMS MODE=BATCH ***          02-@IDMS
                                    571+*                                 *** END DML EXPANSION ***
000FFC D503 C038 0133 00038 00133   572          CLC   ERRSTAT,0307        CHECK IF END OF SET
001002 4780 B038            01038   573          BE    MAIN030             BRANCH IF END OF SET
001006 D503 C038 B46A 00038 0146A   574          CLC   ERRSTAT,STATOK      CHECK IF ERROR
00100C 4770 B27C            0127C   575          BNE   OBERR3
001010 D203 B482 C8E8 01482 008E8   576          MVC   EID,EMPID           MOVE EMPLOYEE ID
001016 D209 B486 C8EC 01486 008EC   577          MVC   FNAME,EMPFNAME      MOVE EMPLOYEE FIRST NAME
00101C D20E B492 C8F6 01492 008F6   578          MVC   LNAME,EMPLNAME      MOVE EMPLOYEE LAST NAME
001022 D201 B454 C93D 01454 0093D   579          MVC   STATNUM,EMPSTATU    MOVE EMPLOYEE STATUS
001028 4160 B030            01030   580          LA    R6,MAIN025          LOAD ADDRESS OF PRINT LINK
00102C 47F0 B342            01342   581          B     CKSTAT
                            01030   582 MAIN025  EQU   *
001030 4150 CFBE            00FBE   583          LA    R5,MAIN020
001034 47F0 B2AE            012AE   584          B     PRINTREC
                            01038   585 MAIN030  EQU   *
001038 D203 C8E8 B4A1 008E8 014A1   586          MVC   EMPID,WALK
                                    587          @FIND CALC,REC='EMPLOYEE' FIND NEXT EMPLOYEE
                                    588+*                                 *** BEGIN DML EXPANSION ***
00103E 4100 C030            00030   589+         LA    0,SSCTRL                                         02-@IDMS
001042 5000 C094            00094   590+         ST    0,SSCIDBCM+4                                     02-@IDMS
001046 4100 C0AF            000AF   591+         LA    0,SSCIDBCM+32-1                                  02-@IDMS
```

```
00104A 5000 C098           00098   592+         ST    0,SSCIDBCM+8                                                 02-@IDMS
00104E 4100 B858           01858   593+         LA    0,=CL18'EMPLOYEE'                                            02-@IDMS
001052 5000 C09C           0009C   594+         ST    0,SSCIDBCM+12                                                02-@IDMS
001056 9680 C09C      0009C        595+         OI    SSCIDBCM+12,X'80'                                            02-@IDMS
00105A 4100 000A           0000A   596+         LA    0,10                                                         02-@IDMS
00105E 5000 C104           00104   597+         ST    0,DMLSEQ                                                     02-@IDMS
                                   598+*,                          DML-SEQUENCE = 10                               02-@IDMS
001062 4110 C094           00094   599+         LA    1,SSCIDBCM+4                                                 02-@IDMS
001066 58F0 B7B0           017B0   600+         L     15,=V(IDMS)                                                  02-@IDMS
00106A 05EF                        601+         BALR  14,15              *** CALL IDMS MODE=BATCH ***          02-@IDMS
                                   602+*                                 *** END DML EXPANSION ***
00106C D503 C038 B46A 00038 0146A  603          CLC   ERRSTAT,STATOK      CHECK IF ERROR
001072 4770 B218           01218   604          BNE   CALCERR
                           01076   605 REPEAT   EQU   *
                                   606          @OBTAIN NEXT,SET='OFFICE-EMPLOYEE',REC='EMPLOYEE'
                                   607+*                                 *** BEGIN DML EXPANSION ***
001076 4100 C030           00030   608+         LA    0,SSCTRL                                                     03-@IDMS
00107A 5000 C094           00094   609+         ST    0,SSCIDBCM+4                                                 03-@IDMS
00107E 4100 C099           00099   610+         LA    0,SSCIDBCM+10+0-1                                            03-@IDMS
001082 5000 C098           00098   611+         ST    0,SSCIDBCM+8                                                 03-@IDMS
001086 4100 B858           01858   612+         LA    0,=CL18'EMPLOYEE'                                            03-@IDMS
00108A 5000 C09C           0009C   613+         ST    0,SSCIDBCM+12                                                03-@IDMS
00108E 4100 B88E           0188E   614+         LA    0,=CL18'OFFICE-EMPLOYEE'                                     03-@IDMS
001092 5000 C0A0           000A0   615+         ST    0,SSCIDBCM+16                                                03-@IDMS
001096 4100 C0BA           000BA   616+         LA    0,SSCIDBCM+43-1                                              02-@IDMS
00109A 5000 C0A4           000A4   617+         ST    0,SSCIDBCM+20                                                02-@IDMS
00109E 9680 C0A4      000A4        618+         OI    SSCIDBCM+20,X'80'                                            02-@IDMS
0010A2 4100 000B           0000B   619+         LA    0,11                                                         02-@IDMS
0010A6 5000 C104           00104   620+         ST    0,DMLSEQ                                                     02-@IDMS
                                   621+*,                          DML-SEQUENCE = 11                               02-@IDMS
0010AA 4110 C094           00094   622+         LA    1,SSCIDBCM+4                                                 02-@IDMS
0010AE 58F0 B7B0           017B0   623+         L     15,=V(IDMS)                                                  02-@IDMS
0010B2 05EF                        624+         BALR  14,15              *** CALL IDMS MODE=BATCH ***          02-@IDMS
                                   625+*                                 *** END DML EXPANSION ***
0010B4 D503 C038 B7B4 00038 017B4  626          CLC   ERRSTAT,=C'0307'    END OF SET ?
0010BA 4780 B126           01126   627          BE    MAIN040             BRANCH IF END OF SET
0010BE D503 C038 B46A 00038 0146A  628          CLC   ERRSTAT,STATOK
0010C4 4770 B260           01260   629          BNE   OBERR1
                                   630          @IF   SET='DEPT-EMPLOYEE',MEMBER=YES,GOTO=REPEAT
                                   631+*                                 *** BEGIN DML EXPANSION ***
0010C8 4100 C030           00030   632+         LA    0,SSCTRL                                                     02-@IDMS
0010CC 5000 C094           00094   633+         ST    0,SSCIDBCM+4                                                 02-@IDMS
0010D0 4100 C0CB           000CB   634+         LA    0,SSCIDBCM+60-1                                              02-@IDMS
0010D4 5000 C098           00098   635+         ST    0,SSCIDBCM+8                                                 02-@IDMS
0010D8 4100 B8A0           018A0   636+         LA    0,=CL18'DEPT-EMPLOYEE'                                       02-@IDMS
0010DC 5000 C09C           0009C   637+         ST    0,SSCIDBCM+12                                                02-@IDMS
0010E0 9680 C09C      0009C        638+         OI    SSCIDBCM+12,X'80'                                            02-@IDMS
0010E4 4100 000C           0000C   639+         LA    0,12                                                         02-@IDMS
0010E8 5000 C104           00104   640+         ST    0,DMLSEQ                                                     02-@IDMS
                                   641+*,                          DML-SEQUENCE = 12                               02-@IDMS
0010EC 4110 C094           00094   642+         LA    1,SSCIDBCM+4                                                 02-@IDMS
0010F0 58F0 B7B0           017B0   643+         L     15,=V(IDMS)                                                  02-@IDMS
0010F4 05EF                        644+         BALR  14,15              *** CALL IDMS MODE=BATCH ***          02-@IDMS
                                   645+*                                 *** END DML EXPANSION ***
0010F6 D503 C038 B7B8 00038 017B8  646+         CLC   ERRSTAT,=C'0000'                                             01-@IF
0010FC 4780 B076           01076   647+         BE    REPEAT                                                       01-@IF
001100 D203 B482 C8E8 01482 008E8  648          MVC   EID,EMPID           MOVE EMPLOYEE ID
001106 D209 B486 C8EC 01486 008EC  649          MVC   FNAME,EMPFNAME      MOVE EMPLOYEE FIRST NAME
00110C D20E B492 C8F6 01492 008F6  650          MVC   LNAME,EMPLNAME      MOVE EMPLOYEE LAST NAME
001112 D203 B4A1 C8E8 014A1 008E8  651          MVC   WALK,EMPID
001118 D201 B454 C93D 01454 0093D  652          MVC   STATNUM,EMPSTATU
00111E 4160 CF6A           00F6A   653          LA    R6,NEWDPT           ADDRESS OF DEPT ROUTINE
001122 47F0 B342           01342   654          B     CKSTAT
                           01126   655 MAIN040  EQU   *
                                   656          @OBTAIN NEXT,AREA='ORG-DEMO-REGION',REC='OFFICE'
                                   657+*                                 *** BEGIN DML EXPANSION ***
001126 4100 C030           00030   658+         LA    0,SSCTRL                                                     03-@IDMS
00112A 5000 C094           00094   659+         ST    0,SSCIDBCM+4                                                 03-@IDMS
00112E 4100 C09A           0009A   660+         LA    0,SSCIDBCM+10+1-1                                            03-@IDMS
001132 5000 C098           00098   661+         ST    0,SSCIDBCM+8                                                 03-@IDMS
001136 4100 B846           01846   662+         LA    0,=CL18'OFFICE'                                              03-@IDMS
00113A 5000 C09C           0009C   663+         ST    0,SSCIDBCM+12                                                03-@IDMS
00113E 4100 B87C           0187C   664+         LA    0,=CL18'ORG-DEMO-REGION'                                     03-@IDMS
001142 5000 C0A0           000A0   665+         ST    0,SSCIDBCM+16                                                03-@IDMS
```

```
001146 4100 C0BA           000BA   666+         LA    0,SSCIDBCM+43-1                                    02-@IDMS
00114A 5000 C0A4           000A4   667+         ST    0,SSCIDBCM+20                                      02-@IDMS
00114E 9680 C0A4      000A4        668+         OI    SSCIDBCM+20,X'80'                                  02-@IDMS
001152 4100 000D           0000D   669+         LA    0,13                                               02-@IDMS
001156 5000 C104           00104   670+         ST    0,DMLSEQ                                           02-@IDMS
                                   671+*,                          DML-SEQUENCE = 13                     02-@IDMS
00115A 4110 C094           00094   672+         LA    1,SSCIDBCM+4                                       02-@IDMS
00115E 58F0 B7B0           017B0   673+         L     15,=V(IDMS)                                        02-@IDMS
001162 05EF                        674+         BALR  14,15              *** CALL IDMS MODE=BATCH ***    02-@IDMS
                                   675+*                                 *** END DML EXPANSION ***
001164 47F0 CEE6           00EE6   676          B     NEWOFFC
                           01168   677 EOF      EQU   *
                                   678          @FINISH                   *
                                   679+*                                 *** BEGIN DML EXPANSION ***
001168 4100 C030           00030   680+         LA    0,SSCTRL                                           02-@IDMS
00116C 5000 C094           00094   681+         ST    0,SSCIDBCM+4                                       02-@IDMS
001170 4100 C091           00091   682+         LA    0,SSCIDBCM+2-1                                     02-@IDMS
001174 5000 C098           00098   683+         ST    0,SSCIDBCM+8                                       02-@IDMS
001178 9680 C098      00098        684+         OI    SSCIDBCM+8,X'80'                                   02-@IDMS
00117C 4100 000E           0000E   685+         LA    0,14                                               02-@IDMS
001180 5000 C104           00104   686+         ST    0,DMLSEQ                                           02-@IDMS
                                   687+*,                          DML-SEQUENCE = 14                     02-@IDMS
001184 4110 C094           00094   688+         LA    1,SSCIDBCM+4                                       02-@IDMS
001188 58F0 B7B0           017B0   689+         L     15,=V(IDMS)                                        02-@IDMS
00118C 05EF                        690+         BALR  14,15              *** CALL IDMS MODE=BATCH ***    02-@IDMS
                                   691+*                                 *** END DML EXPANSION ***
00118E D503 C038 B46A 00038 0146A  692          CLC   ERRSTAT,STATOK
001194 4770 B23C           0123C   693          BNE   FINERR
                                   694          CLOSE (OUTFILE)
001198                             695+         CNOP  0,4                              ALIGN LIST TO FULLWORD 01-CLOSE
001198 4510 B1A0           011A0   696+         BAL   1,*+8                       LOAD REG1 W/LIST ADDR. @L2A 01-CLOSE
00119C 80                          697+         DC    AL1(128)                         OPTION BYTE            01-CLOSE
00119D 00173C                      698+         DC    AL3(OUTFILE)                     DCB ADDRESS            01-CLOSE
0011A0 0A14                        699+         SVC   20                               ISSUE CLOSE SVC        01-CLOSE
0011A2 58D0 B410           01410   700          L     R13,SAVEAREA+4
0011A6 98EC D00C           0000C   701          LM    R14,R12,12(R13)
0011AA 07FE                        702          BR    R14                 RETURN
                                   703 *   ERROR ROUTINES                                *
                           011AC   704 BSERROR  EQU   *
0011AC 9240 B50D      0150D        705          MVI   ERRMSG,C' '
0011B0 D212 B50E B50D 0150E 0150D  706          MVC   ERRMSG+1(19),ERRMSG
0011B6 9240 B521      01521        707          MVI   ERRNUM,C' '
0011BA D202 B522 B521 01522 01521  708          MVC   ERRNUM+1(3),ERRNUM
0011C0 D203 B521 C038 01521 00038  709          MVC   ERRNUM,ERRSTAT
0011C6 D213 B50D B560 0150D 01560  710          MVC   ERRMSG,BSMSG
0011CC 47F0 B28A           0128A   711          B     PRINTERR
                           011D0   712 BRERROR  EQU   *
0011D0 9240 B50D      0150D        713          MVI   ERRMSG,C' '
0011D4 D212 B50E B50D 0150E 0150D  714          MVC   ERRMSG+1(19),ERRMSG
0011DA 9240 B521      01521        715          MVI   ERRNUM,C' '
0011DE D202 B522 B521 01522 01521  716          MVC   ERRNUM+1(3),ERRNUM
0011E4 D203 B521 C038 01521 00038  717          MVC   ERRNUM,ERRSTAT
0011EA D213 B50D B574 0150D 01574  718          MVC   ERRMSG,BRMSG
0011F0 47F0 B28A           0128A   719          B     PRINTERR
                           011F4   720 AREAERR  EQU   *
0011F4 9240 B50D      0150D        721          MVI   ERRMSG,C' '
0011F8 D212 B50E B50D 0150E 0150D  722          MVC   ERRMSG+1(19),ERRMSG
0011FE 9240 B521      01521        723          MVI   ERRNUM,C' '
001202 D202 B522 B521 01522 01521  724          MVC   ERRNUM+1(3),ERRNUM
001208 D203 B521 C038 01521 00038  725          MVC   ERRNUM,ERRSTAT
00120E D213 B50D B588 0150D 01588  726          MVC   ERRMSG,AREAMSG
001214 47F0 B28A           0128A   727          B     PRINTERR
                           01218   728 CALCERR  EQU   *
001218 9240 B50D      0150D        729          MVI   ERRMSG,C' '
00121C D212 B50E B50D 0150E 0150D  730          MVC   ERRMSG+1(19),ERRMSG
001222 9240 B521      01521        731          MVI   ERRNUM,C' '
001226 D202 B522 B521 01522 01521  732          MVC   ERRNUM+1(3),ERRNUM
00122C D203 B521 C038 01521 00038  733          MVC   ERRNUM,ERRSTAT
001232 D213 B50D B59C 0150D 0159C  734          MVC   ERRMSG,CALMSG
001238 47F0 B28A           0128A   735          B     PRINTERR
                           0123C   736 FINERR   EQU   *
00123C 9240 B50D      0150D        737          MVI   ERRMSG,C' '
001240 D212 B50E B50D 0150E 0150D  738          MVC   ERRMSG+1(19),ERRMSG
001246 9240 B521      01521        739          MVI   ERRNUM,C' '
```

```
00124A D202 B522 B521 01522 01521   740          MVC   ERRNUM+1(3),ERRNUM
001250 D203 B521 C038 01521 00038   741          MVC   ERRNUM,ERRSTAT
001256 D213 B50D B5B0 0150D 015B0   742          MVC   ERRMSG,FINMSG
00125C 47F0 B28A            0128A   743          B     PRINTERR
                            01260   744 OBERR1   EQU   *
001260 D200 B5C4 B8BD 015C4 018BD   745          MVC   EDSW,=C'Y'
001266 4150 B126            01126   746          LA    R5,MAIN040
00126A 47F0 B2AE            012AE   747          B     PRINTREC
                            0126E   748 OBERR2   EQU   *
00126E D200 B5C5 B8BD 015C5 018BD   749          MVC   DSW,=C'Y'
001274 4150 B076            01076   750          LA    R5,REPEAT
001278 47F0 B2AE            012AE   751          B     PRINTREC
                            0127C   752 OBERR3   EQU   *
00127C D200 B5C6 B8BD 015C6 018BD   753          MVC   ESW,=C'Y'
001282 4150 B038            01038   754          LA    R5,MAIN030
001286 47F0 B2AE            012AE   755          B     PRINTREC
                                    756 *   PRINT ROUTINES
                            0128A   757 PRINTERR EQU   *
00128A D284 B4D6 0040 014D6 00040   758          MVC   ERRLINE,C' '
001290 D283 B4D7 B4D6 014D7 014D6   759          MVC   ERRLINE+1(132),ERRLINE
001296 92F0 B4D6      014D6         760          MVI   ERRLINE,C'0'
                                    761          PUT   OUTFILE,ERRLINE
00129A 4110 B73C            0173C   762+         LA    1,OUTFILE                         LOAD PARAMETER REG 1   02-IHBIN
00129E 4100 B4D6            014D6   763+         LA    0,ERRLINE                         LOAD PARAMETER REG 0   02-IHBIN
0012A2 1FFF                         764+         SLR   15,15                    CLEAR REGISTER             @L1A 01-PUT
0012A4 BFF7 1031            00031   765+         ICM   15,7,49(1)               LOAD PUT ROUTINE ADDR      @L1C 01-PUT
0012A8 05EF                         766+         BALR  14,15                    LINK TO PUT ROUTINE             01-PUT
0012AA 47F0 B168            01168   767          B     EOF
                            012AE   768 PRINTREC EQU   *
0012AE 9240 B5C7      015C7         769          MVI   LINE1,C' '
0012B2 D283 B5C8 B5C7 015C8 015C7   770          MVC   LINE1+1(132),LINE1
0012B8 92F0 B5C7      015C7         771          MVI   LINE1,C'0'
0012BC 9240 B64C      0164C         772          MVI   LINE2,C' '
0012C0 D283 B64D B64C 0164D 0164C   773          MVC   LINE2+1(132),LINE2
0012C6 D500 B5C4 B8BD 015C4 018BD   774          CLC   EDSW,=C'Y'
0012CC 4780 B302            01302   775          BE    SKIPED
0012D0 D500 B5C5 B8BD 015C5 018BD   776          CLC   DSW,=C'Y'
0012D6 4780 B2F0            012F0   777          BE    SKIPD
0012DA D22C B5E2 B4A9 015E2 014A9   778          MVC   LINE1+27(45),DEPT
0012E0 D203 B667 B4A5 01667 014A5   779          MVC   LINE2+27(4),DID
0012E6 D500 B5C5 B8BD 015C5 018BD   780          CLC   DSW,=C'Y'
0012EC 4780 B302            01302   781          BE    SKIPED
                            012F0   782 SKIPD    EQU   *
0012F0 D21A B614 B486 01614 01486   783          MVC   LINE1+77(27),ENAME
0012F6 D203 B699 B482 01699 01482   784          MVC   LINE2+77(4),EID
0012FC D213 B634 B456 01634 01456   785          MVC   LINE1+109(20),STAT
                            01302   786 SKIPED   EQU   *
001302 D20E B5CE B473 015CE 01473   787          MVC   LINE1+7(15),OCITY
001308 D203 B653 B470 01653 01470   788          MVC   LINE2+7(4),OCODE
                                    789          PUT   OUTFILE,LINE1
00130E 4110 B73C            0173C   790+         LA    1,OUTFILE                         LOAD PARAMETER REG 1   02-IHBIN
001312 4100 B5C7            015C7   791+         LA    0,LINE1                           LOAD PARAMETER REG 0   02-IHBIN
001316 1FFF                         792+         SLR   15,15                    CLEAR REGISTER             @L1A 01-PUT
001318 BFF7 1031            00031   793+         ICM   15,7,49(1)               LOAD PUT ROUTINE ADDR      @L1C 01-PUT
00131C 05EF                         794+         BALR  14,15                    LINK TO PUT ROUTINE             01-PUT
                                    795          PUT   OUTFILE,LINE2
00131E 4110 B73C            0173C   796+         LA    1,OUTFILE                         LOAD PARAMETER REG 1   02-IHBIN
001322 4100 B64C            0164C   797+         LA    0,LINE2                           LOAD PARAMETER REG 0   02-IHBIN
001326 1FFF                         798+         SLR   15,15                    CLEAR REGISTER             @L1A 01-PUT
001328 BFF7 1031            00031   799+         ICM   15,7,49(1)               LOAD PUT ROUTINE ADDR      @L1C 01-PUT
00132C 05EF                         800+         BALR  14,15                    LINK TO PUT ROUTINE             01-PUT
00132E D200 B5C4 B8BC 015C4 018BC   801          MVC   EDSW,=C'N'
001334 D200 B5C5 B8BC 015C5 018BC   802          MVC   DSW,=C'N'
00133A D200 B5C6 B8BC 015C6 018BC   803          MVC   ESW,=C'N'
001340 07F5                         804          BR    R5
                                    805 *   CHECK STATUS ROUTINE           *
                            01342   806 CKSTAT   EQU   *
001342 D501 B454 B8B2 01454 018B2   807          CLC   STATNUM,=C'01'
001348 4780 B37C            0137C   808          BE    ACT
00134C D501 B454 B8B4 01454 018B4   809          CLC   STATNUM,=C'02'
001352 4780 B384            01384   810          BE    STD
001356 D501 B454 B8B6 01454 018B6   811          CLC   STATNUM,=C'03'
00135C 4780 B38C            0138C   812          BE    LTD
001360 D501 B454 B8B8 01454 018B8   813          CLC   STATNUM,=C'04'
```

```
001366 4780 B394            01394   814          BE    LVO
00136A D501 B454 B8BA 01454 018BA   815          CLC   STATNUM,=C'05'
001370 4780 B39C            0139C   816          BE    TRM
001374 D213 B456 B7BC 01456 017BC   817          MVC   STAT,=C' STATUS CODE ERROR  '
00137A 07F6                         818          BR    R6
                            0137C   819 ACT      EQU   *
00137C D213 B456 B7D0 01456 017D0   820          MVC   STAT,=C' ACTIVE             '
001382 07F6                         821          BR    R6
                            01384   822 STD      EQU   *
001384 D213 B456 B7E4 01456 017E4   823          MVC   STAT,=C' SHORT TERM DISABLED'
00138A 07F6                         824          BR    R6
                            0138C   825 LTD      EQU   *
00138C D213 B456 B7F8 01456 017F8   826          MVC   STAT,=C' LONG TERM DISBALED '
001392 07F6                         827          BR    R6
                            01394   828 LVO      EQU   *
001394 D213 B456 B80C 01456 0180C   829          MVC   STAT,=C' LEAVE OF ABSENCE   '
00139A 07F6                         830          BR    R6
                            0139C   831 TRM      EQU   *
00139C D213 B456 B820 01456 01820   832          MVC   STAT,=C' TERMINATED         '
0013A2 07F6                         833          BR    R6
                                    834 *   PRINT REPORT HEADING ROUTINE   *
                            013A4   835 PRTHEAD  EQU   *
0013A4 9240 B5C7      015C7         836          MVI   LINE1,C' '
0013A8 D283 B5C8 B5C7 015C8 015C7   837          MVC   LINE1+1(132),LINE1
0013AE 92F1 B5C7      015C7         838          MVI   LINE1,C'1'
0013B2 D219 B5FD B6D1 015FD 016D1   839          MVC   LINE1+54(26),HEAD1
                                    840          PUT   OUTFILE,LINE1
0013B8 4110 B73C            0173C   841+         LA    1,OUTFILE                         LOAD PARAMETER REG 1   02-IHBIN
0013BC 4100 B5C7            015C7   842+         LA    0,LINE1                           LOAD PARAMETER REG 0   02-IHBIN
0013C0 1FFF                         843+         SLR   15,15                    CLEAR REGISTER             @L1A 01-PUT
0013C2 BFF7 1031            00031   844+         ICM   15,7,49(1)               LOAD PUT ROUTINE ADDR      @L1C 01-PUT
0013C6 05EF                         845+         BALR  14,15                    LINK TO PUT ROUTINE             01-PUT
0013C8 9240 B5C7      015C7         846          MVI   LINE1,C' '
0013CC D283 B5C8 B5C7 015C8 015C7   847          MVC   LINE1+1(132),LINE1
0013D2 9260 B5C7      015C7         848          MVI   LINE1,C'-'
0013D6 D211 B5CD B6EB 015CD 016EB   849          MVC   LINE1+6(18),HEAD2O
0013DC D219 B5E1 B6FD 015E1 016FD   850          MVC   LINE1+26(26),HEAD2D
0013E2 D213 B613 B717 01613 01717   851          MVC   LINE1+76(20),HEAD2E
0013E8 D20E B633 B72B 01633 0172B   852          MVC   LINE1+108(15),HEAD2S
                                    853          PUT   OUTFILE,LINE1
0013EE 4110 B73C            0173C   854+         LA    1,OUTFILE                         LOAD PARAMETER REG 1   02-IHBIN
0013F2 4100 B5C7            015C7   855+         LA    0,LINE1                           LOAD PARAMETER REG 0   02-IHBIN
0013F6 1FFF                         856+         SLR   15,15                    CLEAR REGISTER             @L1A 01-PUT
0013F8 BFF7 1031            00031   857+         ICM   15,7,49(1)               LOAD PUT ROUTINE ADDR      @L1C 01-PUT
0013FC 05EF                         858+         BALR  14,15                    LINK TO PUT ROUTINE             01-PUT
0013FE 07F5                         859          BR    R5
                                    860 *
001400 E6D6D9D260C6C9C5             861 WORKFLDS DC   C'WORK-FIELDS'
00140B 00
00140C 0000000000000000             862 SAVEAREA DC   18F'0'
001454                              863 STATNUM  DS    CL2
001456                              864 STAT     DS    CL20
00146A F0F0F0F0                     865 STATOK   DC    CL4'0000'
00146E                              866 STATUS   DS    CL2
001470                              867 OCODE    DS    CL3
001473                              868 OCITY    DS    CL15
001482                              869 EID      DS    CL4
001486                              870 ENAME    DS    0CL27
001486                              871 FNAME    DS    CL10
001490                              872          DS    CL2
001492                              873 LNAME    DS    CL15
0014A1                              874 WALK     DS    CL4
0014A5                              875 DID      DS    CL4
0014A9                              876 DEPT     DS    CL45
0014D6                              877 ERRLINE  DS    0CL133
0014D6                              878          DS    CL1
0014D7 5C405C405C405C40             879          DC    CL48'* * * * * * * * * * * * * * * * * * * *   '
001507 404040404040                 880          DC    CL6'      '
00150D                              881 ERRMSG   DS    CL20
001521                              882 ERRNUM   DS    CL4
001525 404040404040                 883          DC    CL6'      '
00152B 5C405C405C405C40             884          DC    CL48'* * * * * * * * * * * * * * * * * * * *   '
00155B 4040404040                   885          DC    CL5'     '
001560 C2C9D5C440E2E4C2             886 BSMSG    DC    CL20'BIND SUBSCH ERROR # '
```

```
001574 C2C9D5C440D9C5C3            887 BRMSG    DC    CL20'BIND RECORD ERROR # '
001588 D9C5C1C4E840C1D9            888 AREAMSG  DC    CL20'READY AREA  ERROR # '
00159C C6C9D5C440C3C1D3            889 CALMSG   DC    CL20'FIND CALC  ERROR  # '
0015B0 7CC6C9D5C9E2C840            890 FINMSG   DC    CL20'@FINISH ERROR     # '
0015C4                             891 EDSW     DS    CL1
0015C5                             892 DSW      DS    CL1
0015C6                             893 ESW      DS    CL1
0015C7                             894 LINE1    DS    CL133
00164C                             895 LINE2    DS    CL133
0016D1 D6C6C6C9C3C54040            896 HEAD1    DC    CL26'OFFICE  PERSONNEL  LISTING'
0016EB D6C6C6C9C3C561D6            897 HEAD2O   DC    CL18'OFFICE/OFFICE CODE'
0016FD C4C5D7C1D9E3D4C5            898 HEAD2D   DC    CL26'DEPARTMENT/DEPARTMENT CODE'
001717 C5D4D7D3D6E8C5C5            899 HEAD2E   DC    CL20'EMPLOYEE/EMPLOYEE ID'
00172B C5D4D7D3D6E8C5C5            900 HEAD2S   DC    CL15'EMPLOYEE STATUS'
                                   901 *   OUTPUT FILE DCB INFO
                                   902 OUTFILE  DCB   DDNAME=OUTFILE,MACRF=PM,BLKSIZE=133,LRECL=133,          X
                                                      DSORG=PS
                                   904+*                        DATA CONTROL BLOCK
                                   905+*
00173A 0000
00173C                             906+OUTFILE  DC    0F'0'          ORIGIN ON WORD BOUNDARY                  01-DCB
                                   907+*                        DIRECT ACCESS DEVICE INTERFACE
00173C 0000000000000000            908+         DC    BL16'0'        FDAD, DVTBL                              01-DCB
00174C 00000000                    909+         DC    A(0)           KEYLEN, DEVT, TRBAL                      01-DCB
                                   910+*                        COMMON ACCESS METHOD INTERFACE
001750 00                          911+         DC    AL1(0)         BUFNO, NUMBER OF BUFFERS                 01-DCB
001751 000001                      912+         DC    AL3(1)         BUFCB, BUFFER POOL CONTROL BLOCK         01-DCB
001754 0000                        913+         DC    AL2(0)         BUFL, BUFFER LENGTH                      01-DCB
001756 4000                        914+         DC    BL2'0100000000000000' DSORG, DATA SET ORGANIZATION      01-DCB
001758 00000001                    915+         DC    A(1)           IOBAD FOR EXCP OR RESERVED               01-DCB
                                   916+*                        FOUNDATION EXTENSION
00175C 00                          917+         DC    BL1'00000000'  BFTEK, BFALN, DCBE INDICATORS            01-DCB
00175D 000001                      918+         DC    AL3(1)         EODAD (END OF DATA ROUTINE ADDRESS)      01-DCB
001760 00                          919+         DC    BL1'00000000'  RECFM (RECORD FORMAT)                    01-DCB
001761 000000                      920+         DC    AL3(0)         EXLST (EXIT LIST ADDRESS)                01-DCB
                                   921+*                        FOUNDATION BLOCK
001764 D6E4E3C6C9D3C540            922+         DC    CL8'OUTFILE'   DDNAME                                   01-DCB
00176C 02                          923+         DC    BL1'00000010'  OFLGS (OPEN FLAGS)                       01-DCB
00176D 00                          924+         DC    BL1'00000000'  IFLGS (IOS FLAGS)                        01-DCB
00176E 0050                        925+         DC    BL2'0000000001010000' MACR (MACRO FORMAT)               01-DCB
                                   926+*                        BSAM-BPAM-QSAM INTERFACE
001770 00                          927+         DC    BL1'00000000'  OPTCD, OPTION CODES                      01-DCB
001771 000001                      928+         DC    AL3(1)         CHECK OR INTERNAL QSAM SYNCHRONIZING RTN. 01-DCB
001774 00000001                    929+         DC    A(1)           SYNAD, SYNCHRONOUS ERROR RTN. (3 BYTES)  01-DCB
001778 0000                        930+         DC    H'0'           INTERNAL ACCESS METHOD FLAGS             01-DCB
00177A 0085                        931+         DC    AL2(133)       BLKSIZE, BLOCK SIZE                      01-DCB
00177C 00000000                    932+         DC    F'0'           INTERNAL ACCESS METHOD FLAGS             01-DCB
001780 00000001                    933+         DC    A(1)           INTERNAL ACCESS METHOD USE               01-DCB
                                   934+*                             QSAM INTERFACE
001784 00000001                    935+         DC    A(1)           EOBAD                                    01-DCB
001788 00000001                    936+         DC    A(1)           RECAD                                    01-DCB
00178C 0000                        937+         DC    H'0'           QSWS (FLAGS) AND EITHER DIRCT OR BUFOFF  01-DCB
00178E 0085                        938+         DC    AL2(133)       LRECL                                    01-DCB
001790 00                          939+         DC    BL1'00000000'  EROPT, ERROR OPTION                      01-DCB
001791 000001                      940+         DC    AL3(1)         CNTRL                                    01-DCB
001794 00000000                    941+         DC    H'0,0'         RESERVED AND PRECL                       01-DCB
001798 00000001                    942+         DC    A(1)           EOB, INTERNAL ACCESS METHOD FIELD        01-DCB
0017A0                             943          LTORG
0017A0 4040404040404040            944                =CL8' '
0017A8 C1D7D7D3C4C9C3E3            945                =CL8'APPLDICT'
0017B0 00000000                    946                =V(IDMS)
0017B4 F0F3F0F7                    947                =C'0307'
0017B8 F0F0F0F0                    948                =C'0000'
0017BC 40E2E3C1E3E4E240            949                =C' STATUS CODE ERROR  '
0017D0 40C1C3E3C9E5C540            950                =C' ACTIVE             '
0017E4 40E2C8D6D9E340E3            951                =C' SHORT TERM DISABLED'
0017F8 40D3D6D5C740E3C5            952                =C' LONG TERM DISBALED '
00180C 40D3C5C1E5C540D6            953                =C' LEAVE OF ABSENCE   '
001820 40E3C5D9D4C9D5C1            954                =C' TERMINATED         '
001834 C5D4D7E2E2F0F140            955                =CL18'EMPSS01 '
001846 D6C6C6C9C3C54040            956                =CL18'OFFICE'
001858 C5D4D7D3D6E8C5C5            957                =CL18'EMPLOYEE'
00186A C4C5D7C1D9E3D4C5            958                =CL18'DEPARTMENT'
00187C D6D9C760C4C5D4D6            959                =CL18'ORG-DEMO-REGION'
```

```
00188E D6C6C6C9C3C560C5            960               =CL18'OFFICE-EMPLOYEE'
0018A0 C4C5D7E360C5D4D7            961               =CL18'DEPT-EMPLOYEE'
0018B2 F0F1                        962               =C'01'
0018B4 F0F2                        963               =C'02'
0018B6 F0F3                        964               =C'03'
0018B8 F0F4                        965               =C'04'
0018BA F0F5                        966               =C'05'
0018BC D5                          967               =C'N'
0018BD E8                          968               =C'Y'
000000                             969         END   SAMPLE1
```

# Appendix C: Sample DC/UCF Online Program

This appendix contains a sample DC/UCF program that performs a map out operation, prompting the terminal operator for a department ID.

This section contains the following topics:

## Input to the DML Precompiler

The following illustrates a sample online program as input to the DML precompiler.

```
*RETRIEVAL
*DMLIST
*NO-ACTIVITY-LOG
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         SPACE 1


*                                   ENTER FROM NEXT HIGHER LEVEL
         SPACE 1
         PRINT GEN                  ASSEMBLER PRINT OPTIONS
SYBPG2   CSECT
         LR    R12,R15              ESTABLISHES REGISTER 12 AS THE
         USING SYBPG2,R12            BASE REGISTER
         USING STORAGE,R10          ESTABLISH ADDRESSABILITY OF DSECT
         B     PROCESS              BRANCH TO FIND INVOKING TASKCODE
         EJECT
         @INVOKE MODE=IDMSDC,MAP=SYBMAP
*                                   OPERATING MODE: IDMS DC/MAPPING
         EJECT
         SPACE 1
RETURN   DS    0H
         #FREESTG STGID='SYB4'      FREE THE STORAGE ACQUIRED EARLIER
         #RETURN                    RETURN TO HIGHER LEVEL
         SPACE 1
RETURNXT DS    0H
         #RETURN NXTTASK=SYBTSK03   PASS CONTROL BACK TO ITSELF
         SPACE 1
*                                   MAINLINE PROGRAM
         SPACE 1
PROCESS  DS    0H
         #GETSTG TYPE=(USER,LONG,KEEP),PLIST=*,LEN=STORLGTH,           *
               STGID='SYB4',COND=(ALL),ERROR=ERRORTN,ADDR=(R10),       *
               INIT=X'40'
*                                   ACQUIRE VARIABLE STORAGE
         SPACE 1
         #MAPBIND MRB=SYBMAP        BIND MAP AND RECORDS
         #MAPBIND MRB=SYBMAP,RECNAME=SYBREC
         SPACE 1
ACCEPTSK #ACCEPT TYPE=TASKCODE,FIELD=TASKCODE
*                                   ACCEPT TASK CODE TO INVOKE TASK
         CLC   TASKCODE,SYBTSK2     FIRST TIME CALLED ?
         BNE   RECCUR               YES - OUTPUT FIRST SCREEN
*                                   NO  - INPUT DATA FROM SCREEN
FIRSTIME DS    0H
         MVC   SYBDEPID,=C'0000'    PRIME DATA FIELD
         SPACE
         #MREQ OUT,MRB=SYBMAP,OPTNS=(NEWPAGE),ERROR=ERRORTN,           *


               COND=(ALL)
*                                   MAP OUT PROMPT
         SPACE
         B     RETURNXT             EXIT & WAIT FOR OPERATOR RESPONSE
         SPACE 2
RECCUR   DS    0H
         #MREQ IN,MRB=SYBMAP,ERROR=ERRORTN,COND=(ALL)
*                                   MAP IN TERMINAL INPUT
         SPACE 1
         #MAPINQ MRB=SYBMAP,AID=AIDBYTE
*                                     MOVE MAP DATA TO PROG VARIABLE STG
         CLI   AIDBYTE,CLEAR          DID THE OPERATOR REQUEST FINISH?
         BE    RETURN                 YES - EXIT PGM, BACK TO IDMS DC
         SPACE
         #MREQ OUT,MRB=SYBMAP,ERROR=ERRORTN,                           *
               COND=(ALL)
*                                     MAP OUT DATA
         SPACE
         B     RETURNXT               EXIT & WAIT FOR OPERATOR RESPONSE
*                                     NO  - MAPOUT, WAIT ON OPERATOR
ERRORTN  DS    0H                     HERE FOR NONZERO RETURN CODE
         #SNAP AREA=(SYBMAP,SYBMAPLN)
         B     RETURN                 EXIT
CLEAR    EQU   X'6D'                  CLEAR AIDBYTE VALUE
SYBTSK2  DC    CL8'SYBTSK2 '          DC TASK INVOKING VALUE (EXTERNAL)
SYBTSK03 DC    CL8'SYBTSK03'          DC TASK INVOKING VALUE (INTERNAL)
         LTORG
         #BALI
         SPACE 2
*********************************************************************
STORAGE  DSECT                        STORAGE DSECT
         @COPY IDMS,MAP-CONTROL=SYBMAP
SYBMAPLN EQU   *-SYBMAP               LENGTH OF #MRB FOR SNAP
         SPACE 1
         @COPY IDMS,MAP-RECORDS
         SPACE 1
SYSPLIST DS    20F                    MAP OUT PARAMETER LIST AREA
TASKCODE DS    CL8                    TASK CODE WHICH INVOKED PROGRAM
AIDBYTE  DS    X                      ATTENTION IDENTIFIER BYTE
         DS    3X                     RESERVED
STORLGTH EQU   *-STORAGE              TOTAL LENGTH OF STORAGE NEEDED
         SPACE 1
         END   SYBPG2
```

## Output from the DML Precompiler

The following illustrates the sample online program as output from the DML precompiler.

```
    00001  *RETRIEVAL
    00002  *DMLIST
    00003  *NO-ACTIVITY-LOG
    00004  R0       EQU   0
    00005  R1       EQU   1
    00006  R2       EQU   2
    00007  R3       EQU   3
    00008  R4       EQU   4
    00009  R5       EQU   5
    00010  R6       EQU   6
    00011  R7       EQU   7
    00012  R8       EQU   8
    00013  R9       EQU   9
    00014  R10      EQU   10
    00015  R11      EQU   11
    00016  R12      EQU   12
    00017  R13      EQU   13
    00018  R14      EQU   14
    00019  R15      EQU   15
    00020           SPACE 1
    00021  *                                    ENTER FROM NEXT HIGHER LEVEL
    00022           SPACE 1
    00023           PRINT GEN                   ASSEMBLER PRINT OPTIONS
    00024  SYBPG2   CSECT
    00025           LR    R12,R15               ESTABLISHES REGISTER 12 AS THE
    00026           USING SYBPG2,R12             BASE REGISTER
    00027           USING STORAGE,R10           ESTABLISH ADDRESSABILITY OF DSECT
    00028           B     PROCESS               BRANCH TO FIND INVOKING TASKCODE
    00029           EJECT
    00030           @INVOKE MODE=IDMSDC,MAP=SYBMAP
    00032  *                                    OPERATING MODE: IDMS DC/MAPPING
    00033           EJECT
    00034           SPACE 1
    00035  RETURN   DS    0H
    00036           #FREESTG STGID='SYB4'       FREE THE STORAGE ACQUIRED EARLIER
    00037           #RETURN                     RETURN TO HIGHER LEVEL
    00038           SPACE 1
    00039  RETURNXT DS    0H
    00040           #RETURN NXTTASK=SYBTSK03    PASS CONTROL BACK TO ITSELF
    00041           SPACE 1
    00042  *                                    MAINLINE PROGRAM
    00043           SPACE 1
    00044  PROCESS  DS    0H
    00045           #GETSTG TYPE=(USER,LONG,KEEP),PLIST=*,LEN=STORLGTH,           *
    00046                 STGID='SYB4',COND=(ALL),ERROR=ERRORTN,ADDR=(R10),       *
    00047                 INIT=X'40'
    00048  *                                    ACQUIRE VARIABLE STORAGE
    00049           SPACE 1
    00050           #MAPBIND MRB=SYBMAP         BIND MAP AND RECORDS
    00057           #MAPBIND MRB=SYBMAP,RECNAME=SYBREC
-   00061           SPACE 1
    00062  ACCEPTSK #ACCEPT TYPE=TASKCODE,FIELD=TASKCODE
    00063  *                                    ACCEPT TASK CODE TO INVOKE TASK
    00064           CLC   TASKCODE,SYBTSK2      FIRST TIME CALLED ?
    00065           BNE   RECCUR                YES - OUTPUT FIRST SCREEN
    00066  *                                    NO  - INPUT DATA FROM SCREEN
    00067  FIRSTIME DS    0H
    00068           MVC   SYBDEPID,=C'0000'     PRIME DATA FIELD
    00069           SPACE
    00070           #MREQ OUT,MRB=SYBMAP,OPTNS=(NEWPAGE),ERROR=ERRORTN,           *
    00071                 COND=(ALL)
    00072  *                                    MAP OUT PROMPT
    00073           SPACE
    00074           B     RETURNXT              EXIT & WAIT FOR OPERATOR RESPONSE
    00075           SPACE 2
    00076  RECCUR   DS    0H
    00077           #MREQ IN,MRB=SYBMAP,ERROR=ERRORTN,COND=(ALL)
    00078  *                                    MAP IN TERMINAL INPUT
    00079           SPACE 1
    00080           #MAPINQ MRB=SYBMAP,AID=AIDBYTE
    00082  *                                    MOVE MAP DATA TO PROG VARIABLE STG
    00083           CLI   AIDBYTE,CLEAR         DID THE OPERATOR REQUEST FINISH?
    00084           BE    RETURN                YES - EXIT PGM, BACK TO IDMS DC
```

```
                SPACE
                #MREQ OUT,MRB=SYBMAP,ERROR=ERRORTN,                        *
                      COND=(ALL)
       *                                    MAP OUT DATA
                SPACE
                B     RETURNXT              EXIT & WAIT FOR OPERATOR RESPONSE
       *                                    NO  - MAPOUT, WAIT ON OPERATOR
       ERRORTN  DS    0H                    HERE FOR NONZERO RETURN CODE
                #SNAP AREA=(SYBMAP,SYBMAPLN)
                B     RETURN                EXIT
       CLEAR    EQU   X'6D'                 CLEAR AIDBYTE VALUE
       SYBTSK2  DC    CL8'SYBTSK2 '         DC TASK INVOKING VALUE (EXTERNAL)
       SYBTSK03 DC    CL8'SYBTSK03'         DC TASK INVOKING VALUE (INTERNAL)
                LTORG
                #BALI
                SPACE 2
       *********************************************************************
       STORAGE  DSECT                       STORAGE DSECT
DMLA            @COPY IDMS,MAP-CONTROL=SYBMAP
                #MRB  MAPNAME=SYBMAP,FIELDS=0001,RECORDS=0001
       SYBMAPLN EQU   *-SYBMAP              LENGTH OF #MRB FOR SNAP
                SPACE 1
DMLA            @COPY IDMS,MAP-RECORDS
                                        DS    0D
       SYBREC                           DS    0CL4
       SYBDEPID                         DS    CL4
       *****************************************
-               SPACE 1
       SYSPLIST DS    20F                   MAP OUT PARAMETER LIST AREA
       TASKCODE DS    CL8                   TASK CODE WHICH INVOKED PROGRAM
       AIDBYTE  DS    X                     ATTENTION IDENTIFIER BYTE
                DS    3X                    RESERVED
       STORLGTH EQU   *-STORAGE             TOTAL LENGTH OF STORAGE NEEDED
                SPACE 1
                END   SYBPG2
```

## Output from the Assembler

The following illustrates the sample online program as output from the assembler.

```
1 *DMLIST
                                      2 *NO-ACTIVITY-LOG
                            00000     3 R0       EQU   0
                            00001     4 R1       EQU   1
                            00002     5 R2       EQU   2
                            00003     6 R3       EQU   3
                            00004     7 R4       EQU   4
                            00005     8 R5       EQU   5
                            00006     9 R6       EQU   6
                            00007    10 R7       EQU   7
                            00008    11 R8       EQU   8
                            00009    12 R9       EQU   9
                            0000A    13 R10      EQU   10
                            0000B    14 R11      EQU   11
                            0000C    15 R12      EQU   12
                            0000D    16 R13      EQU   13
                            0000E    17 R14      EQU   14
                            0000F    18 R15      EQU   15
                                     20 *                                   ENTER FROM NEXT HIGHER LEVEL
                                     22          PRINT GEN                  ASSEMBLER PRINT OPTIONS
000000                               23 SYBPG2   CSECT
000000 18CF                          24          LR    R12,R15              ESTABLISHES REGISTER 12 AS THE
                            00000    25          USING SYBPG2,R12            BASE REGISTER
                            00000    26          USING STORAGE,R10          ESTABLISH ADDRESSABILITY OF DSECT
000002 47F0 C03A            0003A    27          B     PROCESS              BRANCH TO FIND INVOKING TASKCODE
                                                                                                              PAGE    3
                                     29 *        @INVOKE MODE=IDMSDC,MAP=SYBMAP
                                     30          @INVOKE MRBTYPE=STANDARD,PAGING=NO
                                     31 *                                   OPERATING MODE: IDMS DC/MAPPING
                                                                                                              PAGE    4
000006                               34 RETURN   DS    0H
                                     35          #FREESTG STGID='SYB4'      FREE THE STORAGE ACQUIRED EARLIER
                                     36+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
000006 47F0 C010            00010    37+         B     $$LD0002                                              + 01-#FREE
00000A 0700                          38+         CNOP  0,4                                                   + 01-#FREE
00000C E2E8C2F4                      39+$$GC0002    DC    CL4'SYB4'                                          + 01-#FREE
                            00010    40+$$LD0002    EQU   *                                                  + 01-#FREE
000010 5810 C00C            0000C    41+         L     1,$$GC0002                                            + 01-#FREE
000014 4100 0012            00012    42+         LA    0,18                                                  + 01-#FREE
000018 58F0 C240            00240    43+         L     15,=V(IDCSACON)                                       + 02-#ENTE
00001C 05EF                          44+         BALR  14,15                                                 + 02-#ENTE
00001E 0002                          45+         DC    AL2(2)                                                + 02-#ENTE
                                     46+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                     47          #RETURN                    RETURN TO HIGHER LEVEL
                                     48+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
000020 1B00                          49+         SR    0,0                                                   + 01-#RETU
000022 1B11                          50+         SR    1,1                                                   + 01-#RETU
000024 58F0 C240            00240    51+         L     15,=V(IDCSACON)                                       + 02-#ENTE
000028 05EF                          52+         BALR  14,15                                                 + 02-#ENTE
00002A 0005                          53+         DC    AL2(5)                                                + 02-#ENTE
                                     54+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
00002C                               56 RETURNXT DS    0H
                                     57          #RETURN NXTTASK=SYBTSK03   PASS CONTROL BACK TO ITSELF
                                     58+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
00002C 1B00                          59+         SR    0,0                                                   + 01-#RETU
00002E 4110 C214            00214    60+         LA    1,SYBTSK03                                            + 01-#RETU
000032 58F0 C240            00240    61+         L     15,=V(IDCSACON)                                       + 02-#ENTE
000036 05EF                          62+         BALR  14,15                                                 + 02-#ENTE
000038 0005                          63+         DC    AL2(5)                                                + 02-#ENTE
                                     64+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                     66 *                                   MAINLINE PROGRAM
00003A                               68 PROCESS  DS    0H
                                     69          #GETSTG TYPE=(USER,LONG,KEEP),PLIST=*,LEN=STORLGTH,          *
                                                       STGID='SYB4',COND=(ALL),ERROR=ERRORTN,ADDR=(R10),      *
                                                       INIT=X'40'
                                     70+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
00003A 0700                          71+         CNOP  0,4                                                   + 01-#GETS
00003C 4510 C058            00058    72+         BAL   1,*+28                                                + 01-#GETS
000040 0000004C                      73+         DC    A(*+12)            ADDR OF PARM1                      + 01-#GETS
000044 00000054                      74+         DC    A(*+16)            ADDR OF PARM2                      + 01-#GETS
000048 00000050                      75+         DC    A(*+8)             ADDR OF PARM3                      + 01-#GETS
00004C 00000120                      76+         DC    A(STORLGTH)                                           + 01-#GETS
000050 E2E8C2F4                      77+         DC    CL4'SYB4'                                             + 01-#GETS
```

```
000054 40                            78+         DC    AL1(X'40')                                             + 01-#GETS
000055 41                            79+         DC    AL1(65)                                                + 01-#GETS
000056 ED                            80+         DC    AL1(237)                                               + 01-#GETS
000057 00                            81+         DC    AL1(0)                                                 + 01-#GETS
000058 58F0 C240            00240    82+         L     15,=V(IDCSACON)                                        + 02-#ENTE
00005C 05EF                          83+         BALR  14,15                                                  + 02-#ENTE
00005E 0001                          84+         DC    AL2(1)                                                 + 02-#ENTE
000060 49F0 C248            00248    85+         CH    15,=H'8'                                               + 01-#GETS
                                                                                                               PAGE    5
000064 47B0 C1DA            001DA    86+         BNL   ERRORTN                                                + 01-#GETS
000068 18A1                          87+         LR    R10,1                                                  + 01-#GETS
                                     88+*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                     89 *                                    ACQUIRE VARIABLE STORAGE
                                     91 *        #MAPBIND MRB=SYBMAP         BIND MAP AND RECORDS
                                     92          #MAPBIND MRB=SYBMAP,                                          *
                                                       TSTAMP='11/25/91171238R2',                              *
                                                       SSNAME='        ',                                      *
                                                       NFLDS=1,                                                *
                                                       NRECS=1,                                                *
                                                       SEG=NO
00006A                               93+         DS    0H ++++++++ BIND MAP +++++++++++++++++++++++++++++++++++ 01-#MAPB
00006A D207 A000 C220 00000 00220    94+         MVC   SYBMAP(8),=CL8'SYBMAP'                                  X01-#MAPB
                                       +                                    MAP NAME
000070 D743 A008 A008 00008 00008    95+         XC    SYBMAP+8(76-8),SYBMAP+8   CLEAR REST OF BASIC MRB        01-#MAPB
000076 D20F A008 C228 00008 00228    96+         MVC   SYBMAP+8(16),=CL16'11/25/91171238R2'                    X01-#MAPB
                                       +                                       COMPILE DATE/TIME
00007C D207 A018 C238 00018 00238    97+         MVC   SYBMAP+24(8),=CL8'        '                             X01-#MAPB
                                       +                                       SUBSCHEMA NAME
000082 4100 004C            0004C    98+         LA    0,76                                                     01-#MAPB
000086 4000 A03C            0003C    99+         STH   0,SYBMAP+60             MRE OFFSET                       01-#MAPB
00008A 4100 0001            00001   100+         LA    0,1                     NUMBER OF FIELDS                 01-#MAPB
00008E 4000 A02A            0002A   101+         STH   0,SYBMAP+42                                              01-#MAPB
000092 4110 0001            00001   102+         LA    1,1                     NUMBER OF RECORDS                01-#MAPB
000096 4010 A02C            0002C   103+         STH   1,SYBMAP+44                                              01-#MAPB
00009A 41F0 000E            0000E   104+         LA    15,14                   LENGTH OF ONE MAP REQ ELEMENT    01-#MAPB
00009E 4CF0 A02A            0002A   105+         MH    15,SYBMAP+42            TIMES NUMBER OF FIELDS           01-#MAPB
0000A2 41FF 0003            00003   106+         LA    15,3(15)                ROUND UP TO NEXT FULLWORD        01-#MAPB
0000A6 88F0 0002            00002   107+         SRL   15,2                    RECOF=((L'MRE*#FIELDS)+3)/4)*4   01-#MAPB
0000AA 89F0 0002            00002   108+         SLL   15,2                                                     01-#MAPB
0000AE 40F0 A02E            0002E   109+         STH   15,SYBMAP+46            EQUALS LENGTH OF ALL MRE'S       01-#MAPB
0000B2 92D5 A03A      0003A         110+         MVI   SYBMAP+58,C'N'          SUBSCHEMA VIEW NOT SEGMENTED     01-#MAPB
0000B6 41E0 A04C            0004C   111+         LA    14,SYBMAP+76            POINT TO END OF BASIC MRB        01-#MAPB
0000BA D70D E000 E000 00000 00000   112+         XC    0(1*14,14),0(14)              CLEAR MAP REQUEST ELEMENTS 01-#MAPB
0000C0 41E0 A04C            0004C   113+         LA    14,SYBMAP+76            POINT TO END OF MRB              01-#MAPB
0000C4 4AE0 A02E            0002E   114+         AH    14,SYBMAP+46            POINT TO RECORD ADDRESS SLOTS    01-#MAPB
0000C8 D703 E000 E000 00000 00000   115+         XC    0(1*4,14),0(14)         CLEAR DATA RECORD ADDRESS SLOTS  01-#MAPB
                                    116+*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                    118 *        #MAPBIND MRB=SYBMAP,RECNAME=SYBREC
                                    119          #MAPBIND MRB=SYBMAP,                                          *
                                                       RECNUM=1,                                               *
                                                       RECADDR=SYBREC
0000CE                              120+         DS    0H ++++++++ BIND MAP +++++++++++++++++++++++++++++++++++ 01-#MAPB
0000CE 41E0 A04C            0004C   121+         LA    14,SYBMAP+76            POINT TO END OF BASIC MRB        01-#MAPB
0000D2 4AE0 A02E            0002E   122+         AH    14,SYBMAP+46            PNT TO START OF DATA REC SLOTS   01-#MAPB
0000D6 41F0 A0C0            000C0   123+         LA    15,SYBREC               DATA RECORD ADDRESS              01-#MAPB
0000DA 50FE 0000            00000   124+         ST    15,4*(1-1)(14)          STORE IN MRB SLOT                01-#MAPB
                                    125+*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                    128 ACCEPTSK #ACCEPT TYPE=TASKCODE,FIELD=TASKCODE
                                    129+*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0000DE                              130+ACCEPTSK DS    0H                                                     + 01-#ACCE
                                                                                                               PAGE    6
0000DE 4100 0000            00000   131+         LA    0,1-1              SET RQST TYPE.                      + 01-#ACCE
0000E2 4110 A114            00114   132+         LA    1,TASKCODE         POINT TO RECEIVING FIELD            + 01-#ACCE
0000E6 58F0 C240            00240   133+         L     15,=V(IDCSACON)                                        + 02-#ENTE
0000EA 05EF                         134+         BALR  14,15                                                  + 02-#ENTE
0000EC 0033                         135+         DC    AL2(51)                                                + 02-#ENTE
                                    136+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                    137 *                                    ACCEPT TASK CODE TO INVOKE TASK
0000EE D507 A114 C20C 00114 0020C   138          CLC   TASKCODE,SYBTSK2      FIRST TIME CALLED ?
0000F4 4770 C144            00144   139          BNE   RECCUR                YES - OUTPUT FIRST SCREEN
                                    140 *                                    NO  - INPUT DATA FROM SCREEN
0000F8                              141 FIRSTIME DS    0H
0000F8 D203 A0C0 C244 000C0 00244   142          MVC   SYBDEPID,=C'0000'     PRIME DATA FIELD
                                    144          #MREQ OUT,MRB=SYBMAP,OPTNS=(NEWPAGE),ERROR=ERRORTN,           *
```

```
                                                     COND=(ALL)
0000FE                                145+        DS    0H +++++MAPPING REQUEST ++++++++++++++++++++++++++++ 01-#MREQ
0000FE 9205 A020      00020           146+        MVI   32+SYBMAP,B'00101'              REQUEST TYPE FLAGS   + 01-#MREQ
000102 9601 A021      00021           147+        OI    33+SYBMAP,B'00000001'                FRST OPTION BYTE+ 01-#MREQ
000106 9601 A022      00022           148+        OI    34+SYBMAP,B'00000001'             SECOND OPTION BYTE + 01-#MREQ
00010A 9600 A047      00047           149+        OI    71+SYBMAP,B'00000000'             THIRD  OPTION BYTE + 01-#MREQ
00010E 92FF A023      00023           150+        MVI   35+SYBMAP,B'11111111'                 COND FLAGS     + 01-#MREQ
000112 920F A03B      0003B           151+        MVI   59+SYBMAP,B'1111'            COND FLAGS              + 01-#MREQ
000116 41F0 A0C4            000C4     152+        LA    15,SYSPLIST                                          + 01-#MREQ
00011A D703 F000 F000 00000 00000     153+        XC    0(4,15),0(15)       INITIALIZE THIS FULLWORD       XA + 01-#MREQ
000120 4110 A000            00000     154+        LA    1,SYBMAP                                             + 01-#MREQ
000124 501F 0004            00004     155+        ST    1,4(15)                                           XA + 01-#MREQ
000128 927F F000      00000           156+        MVI   0(15),X'7F'         INDICATE RELEASE 2+ PARMLIST    XA + 01-#MREQ
00012C 9680 F004      00004           157+        OI    4(15),X'80'         INDICATE END OF PLIST           XA + 01-#MREQ
000130 181F                           158+        LR    1,15                                                 + 01-#MREQ
000132 58F0 C240            00240     159+        L     15,=V(IDCSACON)                                      + 02-#ENTE
000136 05EF                           160+        BALR  14,15                                                + 02-#ENTE
000138 002E                           161+        DC    AL2(46)                                              + 02-#ENTE
00013A 12FF                           162+        LTR   15,15               WERE THERE ANY ERRORS AT ALL?      + 01-#MREQ
00013C 4770 C1DA            001DA     163+        BNZ   ERRORTN              YES                             + 01-#MREQ
                                      164+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                      166 *                                 MAP OUT PROMPT
000140 47F0 C02C            0002C     168         B     RETURNXT              EXIT & WAIT FOR OPERATOR RESPONSE
000144                                170 RECCUR   DS    0H
                                      171         #MREQ IN,MRB=SYBMAP,ERROR=ERRORTN,COND=(ALL)
000144                                172+        DS    0H +++++MAPPING REQUEST ++++++++++++++++++++++++++++ 01-#MREQ
000144 9206 A020      00020           173+        MVI   32+SYBMAP,B'00110'              REQUEST TYPE FLAGS   + 01-#MREQ
000148 9600 A021      00021           174+        OI    33+SYBMAP,B'00000000'                FRST OPTION BYTE+ 01-#MREQ
00014C 9600 A022      00022           175+        OI    34+SYBMAP,B'00000000'             SECOND OPTION BYTE + 01-#MREQ
000150 9600 A047      00047           176+        OI    71+SYBMAP,B'00000000'             THIRD  OPTION BYTE + 01-#MREQ
000154 92FF A023      00023           177+        MVI   35+SYBMAP,B'11111111'                 COND FLAGS     + 01-#MREQ
000158 920F A03B      0003B           178+        MVI   59+SYBMAP,B'1111'            COND FLAGS              + 01-#MREQ
00015C 41F0 A0C4            000C4     179+        LA    15,SYSPLIST                                          + 01-#MREQ
000160 D703 F000 F000 00000 00000     180+        XC    0(4,15),0(15)       INITIALIZE THIS FULLWORD       XA + 01-#MREQ
000166 4110 A000            00000     181+        LA    1,SYBMAP                                             + 01-#MREQ
00016A 501F 0004            00004     182+        ST    1,4(15)                                           XA + 01-#MREQ
00016E 927F F000      00000           183+        MVI   0(15),X'7F'         INDICATE RELEASE 2+ PARMLIST    XA + 01-#MREQ
                                                                                                          PAGE    7
000172 9680 F004      00004           184+        OI    4(15),X'80'         INDICATE END OF PLIST           XA + 01-#MREQ
000176 181F                           185+        LR    1,15                                                 + 01-#MREQ
000178 58F0 C240            00240     186+        L     15,=V(IDCSACON)                                      + 02-#ENTE
00017C 05EF                           187+        BALR  14,15                                                + 02-#ENTE
00017E 002E                           188+        DC    AL2(46)                                              + 02-#ENTE
000180 12FF                           189+        LTR   15,15               WERE THERE ANY ERRORS AT ALL?      + 01-#MREQ
000182 4770 C1DA            001DA     190+        BNZ   ERRORTN              YES                             + 01-#MREQ
                                      191+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                      193 *                                 MAP IN TERMINAL INPUT
                                      195 *       #MAPINQ MRB=SYBMAP,AID=AIDBYTE
                                      196         #MAPINQ MRB=SYBMAP,AID=AIDBYTE
000186                                197+        DS    0H ++++++++++ INQUIRE ABOUT LAST MAP OPERATION +++++++ 01-#MAPI
000186 D200 A11C A038 0011C 00038     198+        MVC   AIDBYTE(1),56+SYBMAP                                  X01-#MAPI
                                        +                               ATTENTION IDENTIFIER
                                      199+*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                      201 *                                 MOVE MAP DATA TO PROG VARIABLE STG
00018C 956D A11C      0011C           202         CLI   AIDBYTE,CLEAR         DID THE OPERATOR REQUEST FINISH?
000190 4780 C006            00006     203         BE    RETURN                YES - EXIT PGM, BACK TO IDMS DC
                                      205         #MREQ OUT,MRB=SYBMAP,ERROR=ERRORTN,                          *
                                                     COND=(ALL)
000194                                206+        DS    0H +++++MAPPING REQUEST ++++++++++++++++++++++++++++ 01-#MREQ
000194 9205 A020      00020           207+        MVI   32+SYBMAP,B'00101'              REQUEST TYPE FLAGS   + 01-#MREQ
000198 9600 A021      00021           208+        OI    33+SYBMAP,B'00000000'                FRST OPTION BYTE+ 01-#MREQ
00019C 9600 A022      00022           209+        OI    34+SYBMAP,B'00000000'             SECOND OPTION BYTE + 01-#MREQ
0001A0 9600 A047      00047           210+        OI    71+SYBMAP,B'00000000'             THIRD  OPTION BYTE + 01-#MREQ
0001A4 92FF A023      00023           211+        MVI   35+SYBMAP,B'11111111'                 COND FLAGS     + 01-#MREQ
0001A8 920F A03B      0003B           212+        MVI   59+SYBMAP,B'1111'            COND FLAGS              + 01-#MREQ
0001AC 41F0 A0C4            000C4     213+        LA    15,SYSPLIST                                          + 01-#MREQ
0001B0 D703 F000 F000 00000 00000     214+        XC    0(4,15),0(15)       INITIALIZE THIS FULLWORD       XA + 01-#MREQ
0001B6 4110 A000            00000     215+        LA    1,SYBMAP                                             + 01-#MREQ
0001BA 501F 0004            00004     216+        ST    1,4(15)                                           XA + 01-#MREQ
0001BE 927F F000      00000           217+        MVI   0(15),X'7F'         INDICATE RELEASE 2+ PARMLIST    XA + 01-#MREQ
0001C2 9680 F004      00004           218+        OI    4(15),X'80'         INDICATE END OF PLIST           XA + 01-#MREQ
0001C6 181F                           219+        LR    1,15                                                 + 01-#MREQ
0001C8 58F0 C240            00240     220+        L     15,=V(IDCSACON)                                      + 02-#ENTE
0001CC 05EF                           221+        BALR  14,15                                                + 02-#ENTE
```

```
0001CE 002E                         222+         DC    AL2(46)                                               + 02-#ENTE
0001D0 12FF                         223+         LTR   15,15              WERE THERE ANY ERRORS AT ALL?      + 01-#MREQ
0001D2 4770 C1DA            001DA   224+         BNZ   ERRORTN             YES                               + 01-#MREQ
                                    225+*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                    227 *                                   MAP OUT DATA
0001D6 47F0 C02C            0002C   229          B     RETURNXT             EXIT & WAIT FOR OPERATOR RESPONSE
                                    230 *                                   NO  - MAPOUT, WAIT ON OPERATOR
0001DA                              231 ERRORTN  DS    0H                   HERE FOR NONZERO RETURN CODE
                                    232          #SNAP AREA=(SYBMAP,SYBMAPLN)
                                    233+*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0001DA 90E1 A0C4            000C4   234+         STM   14,1,SYSPLIST                                         + 01-#SNAP
0001DE 4110 A0C4            000C4   235+         LA    1,SYSPLIST                                            + 01-#SNAP
0001E2 9268 1010      00010         236+         MVI   16(1),96+8                                            + 01-#SNAP
                                                                                                              PAGE    8
0001E6 D702 1011 1011 00011 00011   237+         XC    17(3,1),17(1)                                         + 01-#SNAP
0001EC 41E0 A000            00000   238+         LA    14,SYBMAP                                             + 01-#SNAP
0001F0 50E0 1014            00014   239+         ST    14,20(,1)                                             + 01-#SNAP
0001F4 41E0 00C0            000C0   240+         LA    14,SYBMAPLN                                           + 01-#SNAP
0001F8 50E0 1018            00018   241+         ST    14,20+4(,1)                                           + 01-#SNAP
0001FC 9680 1018      00018         242+         OI    28-4(1),X'80'                                         + 01-#SNAP
000200 58F0 C240            00240   243+         L     15,=V(IDCSACON)                                       + 02-#ENTE
000204 05EF                         244+         BALR  14,15                                                 + 02-#ENTE
000206 001D                         245+         DC    AL2(29)                                               + 02-#ENTE
                                    247+*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
000208 47F0 C006            00006   248          B     RETURN               EXIT
                            0006D   249 CLEAR    EQU   X'6D'                CLEAR AIDBYTE VALUE
00020C E2E8C2E3E2D2F240             250 SYBTSK2  DC    CL8'SYBTSK2 '        DC TASK INVOKING VALUE (EXTERNAL)
000214 E2E8C2E3E2D2F0F3             251 SYBTSK03 DC    CL8'SYBTSK03'        DC TASK INVOKING VALUE (INTERNAL)
000220                              252          LTORG
000220 E2E8C2D4C1D74040             253                =CL8'SYBMAP'
000228 F1F161F2F561F9F1             254                =CL16'11/25/91171238R2'
000238 4040404040404040             255                =CL8'        '
000240 00000000                     256                =V(IDCSACON)
000244 F0F0F0F0                     257                =C'0000'
000248 0008                         258                =H'8'
                                    259          #BALI
000250                              260+IDCSACON CSECT ,                  IDMS DC ASSEMBLER PROGRAM INTERFACE  01-#BALI
000250 58FF 0008            00008   261+         L     15,8(15)           ADDRESS OF DC'S COMMON STORAGE AREA  01-#BALI
000254 07FF                         262+         BR    15                                                      01-#BALI
000258                              264+         DS    0F                 FORCE ALIGNMENT                      01-#BALI
                                    265+* THE FOLLOWING AD-CON IS FILLED IN BY THE DC PROGRAM LOADER.
000258 00000258C35BC15B             266+         DC    A(*),C'C$A$'                                            01-#BALI
00024A                              268+SYBPG2   CSECT                                                         01-#BALI
                                    270 *******************************************************************
000000                              271 STORAGE  DSECT                      STORAGE DSECT
                                    272 *        @COPY IDMS,MAP-CONTROL=SYBMAP
                                    273          #MRB  MAPNAME=SYBMAP,FIELDS=0001,RECORDS=0001
000000                              274+         DS    0A                 FORCE FULL-WORD ALIGNMENT            01-#MRB
000000 0000000000000000             275+SYBMAP   DC    XL76'0'            BASIC MAP REQUEST BLOCK              01-#MRB
00004C 0000000000000000             276+         DC    (0001)XL14'0'            MAP REQUEST ELEMENTS           01-#MRB
00005A 0000
00005C 00000000                     277+         DC    (0001)A(0)         DATA RECORD ADDRESS SLOTS            01-#MRB
000060 0000000000000000             278+MRBPLIST DC    20A(0)                                                  01-#MRB
0000B0 0000000000000000             279+MRBPGDS  DC    4A(0)              #STRTPAG, #ENDPAG PARM LIST          01-#MRB
                            000C0   280 SYBMAPLN EQU   *-SYBMAP             LENGTH OF #MRB FOR SNAP
                                    282 *        @COPY IDMS,MAP-RECORDS
0000C0                              283                                 DS    0D
0000C0                              284 SYBREC                          DS    0CL4
0000C0                              285 SYBDEPID                        DS    CL4
                                    286 ***************************************
0000C4                              288 SYSPLIST DS    20F                  MAP OUT PARAMETER LIST AREA
000114                              289 TASKCODE DS    CL8                  TASK CODE WHICH INVOKED PROGRAM
                                                                                                              PAGE    9
00011C                              290 AIDBYTE  DS    X                    ATTENTION IDENTIFIER BYTE
00011D                              291          DS    3X                   RESERVED
                            00120   292 STORLGTH EQU   *-STORAGE            TOTAL LENGTH OF STORAGE NEEDED
000000                              294          END   SYBPG2
```

# Appendix D: Assembler DML Macros and Error Messages

This appendix lists the following:

- Assembler DML macros in alphabetical order
- The error messages generated upon assembly of these macros

This section contains the following topics:

## DML Macros

**Types of Macros**

There are three types of Assembler DML macros, as follows:

- **Statement**—The macro instruction is coded in the application program as a DML statement.
- **Generated**—The macro instruction is generated from a DML statement by the DML precompiler.
- **Invoked**—The macro instruction is invoked by a DML statement macro during assembly.

**List of Macros**

The following table lists DML macros alphabetically.

| Macro | Type | Function |
|---|---|---|
| @ACCEPT | Statement | Encodes the #ACCEPT statement |
| @BIND | Statement | Encodes the @BIND statement |
| @COMMIT | Statement | Encodes the #COMMIT statement |
| @CONNECT | Statement | Encodes the @CONNECT statement |
| @DISCON | Statement | Encodes the @DISCON statement |
| @ERASE | Statement | Encodes the @ERASE statement |
| @FIND | Statement | Encodes the @FIND statement |

| Macro | Type | Function |
|---|---|---|
| @FINISH | Statement | Encodes the @FINISH statement |
| @GET | Statement | Encodes the @GET statement |
| @IDMSGSS | Invoked | Defines the IDMS global variables |
| @IDMSINR | Invoked | Generates the IDMS calling sequence |
| @IF | Statement | Encodes the @IF statement |
| @INVOKE | Statement | Encodes the @INVOKE compiler-directive statement |
| @KEEP | Statement | Encodes the @KEEP statement |
| @LRF | Invoked | Generates the logical record request sequences |
| @MODE | Statement | Encodes the @MODE compiler-directive statement |
| @MODIFY | Statement | Encodes the @MODIFY statement |
| @OBTAIN | Statement | Encodes the @OBTAIN statement |
| @PXE | Generated | Encodes a WHERE clause element |
| @READY | Statement | Encodes the @READY statement |
| @RETURN | Statement | Encodes the @RETURN statement |
| @ROLLBAK | Statement | Encodes the @ROLLBAK statement |
| @SSCTRL | Statement | Copies the IDMS communications block |
| @SSLRCTL | Statement | Copies the LRC block |
| @STORE | Statement | Encodes the @STORE statement |

**Note:** @COPY is a DMLA source statement, not an Assembler macro.

## Error Messages

The remainder of this appendix lists and describes error messages that are generated during macro assembly.

Note: For error messages generated by the DML precompiler and returned to the ERRSTAT field of the IDMS communications block following DML requests, see the *Messages and Codes Guide*.

**@ACCEPT**

**INDECIPHERABLE COMBINATION OF OPERANDS**

Excessive or conflicting operands prevent interpretation of the macro.

Severity: 08

**@BIND**

**LRC MUST BE SPECIFIED OR LRSIZ MUST BE OMITTED**

The LRC parameter was omitted but the LRSIZ parameter was specified. If the LRSIZ parameter is specified, the LRC parameter must also be specified.

Severity: 08

**INDECIPHERABLE COMBINATION OF OPERANDS**

Excessive or conflicting operands prevent interpretation of the macro.

Severity: 08

**TOO MANY OPERANDS SPECIFIED**

Parameters were specified that are not allowed for the BIND statement being issued.

Severity: 08

**@COMMIT**

**'ALL' PARAMETER MUST BE BLANK OR 'ALL'**

The first positional parameter ALL must be specified as ALL or must be omitted.

Severity: 08

**@CONNECT**

**BOTH RECORD NAME AND SET NAME ARE REQUIRED FOR CONNECT**

Either the REC or SET parameter was omitted.

Severity: 08

**@DISCON**

**BOTH RECORD NAME AND SET NAME ARE REQUIRED FOR DISCONNECT**

Either the REC or SET parameter was omitted.

Severity: 08

**@ERASE**

**TYPE OF ERASE IS MISSING OR INCORRECT**

The (required) parameter for type of erase was omitted or invalid. Valid parameters are REC, PERMANENT, SELECTIVE, or ALL.

Severity: 08

**RECORD NAME IS REQUIRED FOR ERASE**

The required REC parameter was not specified.

Severity: 08

**@FIND**

**TYPE OPERAND IS MISSING OR INVALID**

This type of FIND/OBTAIN was not specified (NEXT, FIRST, PRIOR, LAST, NTH).

Severity: 08

**SET,AREA, USING, AND OCCUR ARE NOT ALLOWED FOR FORMAT 1**

SET, AREA, USING, or OCCUR parameters were specified. These parameters are not allowed on FIND DBKEY statements.

Severity: 08

**DBKEY, USING, AND OCCUR ARE NOT ALLOWED FOR FORMAT 2**

DBKEY, USING, or OCCUR parameters were specified. These parameters are not allowed on FIND CURRENT statements.

Severity: 08

**REC, SET, AND AREA ARE MUTUALLY EXCLUSIVE FOR FORMAT 2**

Two or more of the REC, SET, or AREA parameters were specified. Only one of these parameters can be specified on FIND CURRENT statements.

Severity: 08

**DBKEY AND USING ARE NOT ALLOWED FOR FORMAT 3**

Either a DBKEY or a USING parameter was specified. These parameters are not allowed on FIND WITHIN SET/AREA statements.

Severity: 08

**EITHER SET OR AREA MUST BE SPECIFIED**

Neither SET nor AREA parameters were specified; one of these parameters must be specified.

Severity: 08

**OCCUR IS USED ONLY WITH FORMAT 3 FIND NTH**

OCCUR parameters were specified. These parameters are only allowed with FIND NTH WITHIN SET/AREA statements.

Severity: 08

**REC, AREA, DBKEY, USING, AND OCCUR NOT ALLOWED FOR FORMAT 4**

REC, AREA, DBKEY, USING, or OCCUR parameters were specified. These parameters are not allowed on FIND OWNER statements.

Severity: 08

**SET OPERAND IS REQUIRED FOR FORMAT 4**

The required SET parameter was not specified on a FIND OWNER statement.

Severity: 08

**SET, AREA, DBKEY, USING, AND OCCUR ARE NOT ALLOWED FOR FORMAT 5**

SET, AREA, DBKEY, USING, or OCCUR parameters were specified. These parameters are not allowed on FIND CALC/DUPLICATE statements.

Severity: 08

**REC OPERAND IS REQUIRED FOR FORMAT 5**

The required REC parameter was not specified on a FIND CALC/DUPLICATE statement.

Severity: 08

**AREA, DBKEY, AND OCCUR ARE NOT ALLOWED FOR FORMAT 6**

AREA, DBKEY, or OCCUR parameters were specified and are not allowed for FIND WITHIN SET USING SORT KEY statements.

Severity: 08

**REC, SET, AND USING ARE REQUIRED FOR FORMAT 6**

REC, SET, or USING parameters were not specified and are required for FIND WITHIN SET USING SORT KEY statements.

Severity: 08

**KEEP OPERAND NOT SPECIFIED AS SHARED OR EXCLUSIVE**

The KEEP parameter was specified without either the SHARED or EXCLUSIVE parameter.

Severity: 08

**UNEXPECTED ERROR IN FORMAT 3 FIND**

This is a system internal error.

Severity: 20

**@IDMSINR**

**@MODE MACRO DOES NOT PRECEDE THIS DML MACRO**

The @MODE macro was not specified before this macro. The @MODE macro must precede all other macros and must occur only once.

Severity: 16

**@MODE CONTAINED ERRORS, OR WAS NOT FIRST MACRO**

The @MODE macro was coded incorrectly or was not specified before this macro. The @MODE macro must precede all other macros and must occur only once.

Severity: 16

**INVALID TYPE 'type' SPECIFIED IN @IDMSINR MACRO**

This is a system internal error.

Severity: 20

**INVALID OPERAND 'operand' IN @IDMSINR MACRO DML-SEQUENCE = 9999**

This is a system internal error.

Severity: 20

**INVALID MODE 'mode' IN @IDMSINR MACRO**

This is a system error.

Severity: 20

**@IF**

**INVALID SPECIFICATION FOR MEMBER OPERAND OR EMPTY OPERAND**

The MEMBER/EMPTY parameter was specified incorrectly. Either MEMBER or EMPTY must be specified.

Severity: 08

**SET, GOTO, AND EITHER MEMBER OR EMPTY ARE REQUIRED FOR IF**

A required MEMBER or EMPTY, SET, or GOTO parameter was not specified. SET, GOTO and either MEMBER or EMPTY parameters are required in an IF statement.

Severity: 08

**@KEEP**

**TYPE NEITHER SHARED NOR EXCLUSIVE**

Either the SHARED or EXCLUSIVE parameter is required on KEEP statements.

Severity: 08

**INDECIPHERABLE COMBINATION OF OPERANDS**

Excessive or conflicting operands prevent interpretation of the macro.

Severity: 08

**@LRF**

**Key-value MISPLACED**

The named *key-value* parameter was coded in an incorrect position. Parameters must be coded in the following sequence: FIRST/NEXT, REC, IOAREA, ONLRSTS, GOTO, and WHERE.

Severity: 08

**KEY-value,variable-value MISPLACED**

The named positional pair of parameters was coded more than once or in incorrect sequence within a logical-record DML statement. Parameters must be coded in the following sequence: FIRST/NEXT, REC, IOAREA, ONLRSTS, GOTO, and WHERE.

Severity: 08

**KEYWORD PARAMETERS INVALID FOR LRF ACCESS MACROS**

Logical-record DML statements must be coded using positional-pair parameter notation at assembly time.

Severity: 08

**Key-value NOT PAIRED**

The named key-value positional parameter was coded without a corresponding variable value parameter.

Severity: 08

**Key-value IS AN INVALID PARAMETER**

The named key-value parameter is invalid. Valid key-value parameters in logical-record DML statements are: FIRST/NEXT, REC, IOAREA, ONLRSTS, GOTO and WHERE.

Severity: 08

**"REC" MUST BE SPECIFIED**

The REC parameter was not specified on a logical record request. All LRF access macros must specify a logical record name.

Severity: 08

**"WHERE" INVALID WHEN LRSIZ OMITTED ON @BIND**

The LRSIZ parameter (specified on @BIND SUBSCH) was not specified; therefore, no storage was made available for WHERE clause resolution. Add the LRSIZ parameter to @BIND SUBSCH and reassemble.

Severity: 08

**SPECIFY BOTH "ONLRSTS" AND "GOTO" OR NEITHER**

Either the ONLRSTS or the GOTO parameter was omitted from a logical record request; both parameters are needed to encode a DMLA ON clause. Either remove the parameter specified or add the omitted parameter.

Severity: 08

**MACRO ERROR: PASSED VERB verb**

The named verb was incorrectly passed. This is a system internal error.

Severity: 08

**@MODE**

**MULTIPLE OCCURRENCES OF @MODE MACRO**

The @MODE macro was specified more than one time; the @MODE macro must occur only once.

Severity: 08

**INVALID SPECIFICATION mode FOR MODE OPERAND**

The named mode is not valid. Valid mode parameters are BATCH, CICS, CICS-EXEC, IDMSDC, INTERCOMM, SHADOW, and DCBATCH.

Severity: 16

**INVALID SPECIFICATION debug FOR DEBUG OPERAND**

The named debug parameter is not valid. Valid debug parameters are YES or NO.

Severity: 16

**INVALID SPECIFICATION quotes FOR QUOTES OPERAND**

The named quotes parameter is invalid. Valid parameters for quotes are YES or NO.

Severity: 16

**@MODIFY**

**RECORD NAME IS REQUIRED FOR MODIFY**

The record REC parameter was not specified.

Severity: 08

**@OBTAIN**

**KEEP OPERAND NOT SPECIFIED AS SHARED OR EXCLUSIVE**

The KEEP parameter was specified without either the SHARED or EXCLUSIVE parameter. If the KEEP parameter is specified, either SHARED or EXCLUSIVE must also be specified.

Severity: 08

**@PXE**

**LRPXE 9999 CHARACTERS TOO SHORT**

*Lrc-block-size* (specified on @BIND SUBSCH) is too small to contain the specified WHERE clause. Increase *lrc-block-size* and reassemble.

Severity: 08

**DMLA/@PXE INCONSISTENCY: PXE TYPE=type**

This is a system internal error.

Severity: 12

**DMLA/@PXE INCONSISTENCY: V-TYPE=type**

This is a system internal error.

Severity: 12

**DMLA/@PXE INCONSISTENCY: MAX=9999/9999**

This is a system internal error.

Severity: 12

**DMLA/@PXE INCONSISTENCY: LEN=9999/9999**

This is a system internal error.

Severity: 12

**@READY**

**EITHER AREA OPERAND OR ALL OPERAND MUST BE SPECIFIED**

Neither the AREA nor ALL parameter was specified. One of these parameters is required for this macro.

Severity: 08

**EITHER RDONLY OPERAND OR UPDATE OPERAND MUST BE SPECIFIED**

Neither the RDONLY nor UPDATE parameter was specified. One of these parameters is required for this macro.

Severity: 08

**INVALID SPECIFICATION FOR UPDATE OR RDONLY OPERAND**

The UPDATE/RDONLY parameter was specified incorrectly. Valid parameters for UPDATE/RDONLY are: YES or SHARED, PROTECTED or PROTECT, or EXCLUSIVE.

Severity: 08

**ALL OPERAND IS NOT SPECIFIED AS 'ALL'**

The parameter ALL must be specified as ALL.

Severity: 08

**@RETURN**

**BOTH SET AND DBKEY ARE REQUIRED OPERANDS FOR RETURN**

The SET and/or DBKEY parameters were not specified. Both of these parameters are required for this macro.

Severity: 08

**EITHER TYPE OPERAND OR USING OPERAND MUST BE SPECIFIED**

Neither the type-of-return parameter (i.e., CURRENT, FIRST, LAST, NEXT, or PRIOR) nor the USING parameter was specified. RETURN macros must include one of these parameters.

Severity: 08

**INVALID SPECIFICATION FOR TYPE OPERAND**

The type operand was not specified correctly. Valid parameters are CURRENT, FIRST, LAST, NEXT, or PRIOR.

Severity: 08

**@ROLLBAK**

**POSITIONAL PARAMETER 'CONTINUE' INVALID**

The CONTINUE parameter must be specified as CONTINUE or must be omitted.

Severity: 08

**@STORE**

**RECORD NAME IS REQUIRED FOR STORE**

The required REC parameter was not specified.

Severity: 08

# Appendix E: STAE Exits

This section contains the following topics:

## Overview

**What are STAE Exits?**

STAE exits (system task abend exits) are user-written recovery modules supported by DC/UCF systems. STAE exits can be invoked in the event of a program interrupt or an abnormal condition encountered by the task. The user-written module can attempt to recover the task by correcting the abnormal condition. If the abnormal condition cannot be resolved, the STAE program can request abnormal termination of the task.

**How STAE Exits Work**

For each task level, a program can designate a STAE routine by issuing a #STAE request. A task abnormally terminates due to a processing error or an #ABEND command. When a task terminates abnormally, STAE routines for the abended program and for all higher-level programs are executed. #STAE routines can be overridden by a #RETURN statement or excluded explicitly by an #ABEND request from the program that failed.

**Note:** For more information about how to issue a #STAE request, see "#STAE" in Chapter 7.

STAE routines determine the cause of the abnormal condition or program interrupt by checking the abend control element (ACE). When control is transferred to the STAE routine, DC/UCF automatically sets the value in register 1 to the address of a fullword parameter list that contains the address of the ACE. When program execution is interrupted, DC/UCF saves the contents of all registers from the abended program in the ACE.

Note: #ACEDS is a DSECT provided in the DC/UCF macro library that defines the fields of the ACE. #ACEDS can be copied into the program using the @COPY IDMS #ACEDS statement.

For more information about the abend control element (ACE) DSECT, see the *DSECT Reference Guide*.

**Programming Considerations**

Programming considerations for STAE routines are as follows:

- STAE programs must be defined at system generation.
- Resources held by the task remain intact when the STAE routine is invoked.
- STAE routines can issue DC/UCF requests. However, if an error occurs which would normally abort the task, the DC/UCF system will abnormally terminate.
- STAE routines must end with a #RETURN statement. The #RETURN statement can request further action to be taken by specifying the TYPE=NORMAL/ABORT/CONTINUE parameter. If TYPE=CONTINUE is specified, the STAE routine must load the address of the instruction from where processing is to continue in the ACE.

**Beginning Register Values**

At the start of execution of a STAE routine, the DC/UCF system sets registers 1, 13, and 15 to the following values:

- **Register 1** holds the address of a 1-fullword parameter list that contains the address of the ACE.
- **Register 13** holds the address of the STAE routine save area if the SAVAREA option has been defined for the STAE program at system generation.
- **Register 15** holds the entry-point address of the STAE routine.

| Displacement-decimal (hex) | Label in #ACEDS DSECT | Contents | Field Size |
|---|---|---|---|
| 0(0) | ACEPSW | PSW at the time of the interrupt | 8 bytes |
| 8(8) | ACEGPRS | General registers from abended program 0-15 | 64 bytes |
| 72(48) | ACEFPRS | Floating point registers from the abended program 0-6 | 32 bytes |
| 112(70) | ACEFLG | ACE flag (see table below) | 1 byte |
| 115(73) | ACEABCOD | Abend code set by DC/UCF | 4 bytes |
| 120(78) | ACEPGMNM | Name of the abended program | 4 bytes |
| 129(81) | ACEEPSW | PSW in EBCDIC form | 17 bytes |

| Displacement-decimal (hex) | Label in #ACEDS DSECT | Contents | Field Size |
|---|---|---|---|
| 148(94) | ACEOFFST | Displacement of instruction that failed in the abended program | 6 bytes |
| 160(A0) | ACEILC | XA program interrupt length counter | 1 byte |
| 161(A1) | ACEINTC | XA interruption code | 2 bytes |
| ACEPSWDA | Data at PSW Start 16 bytes before and after PSW | 32 bytes | |

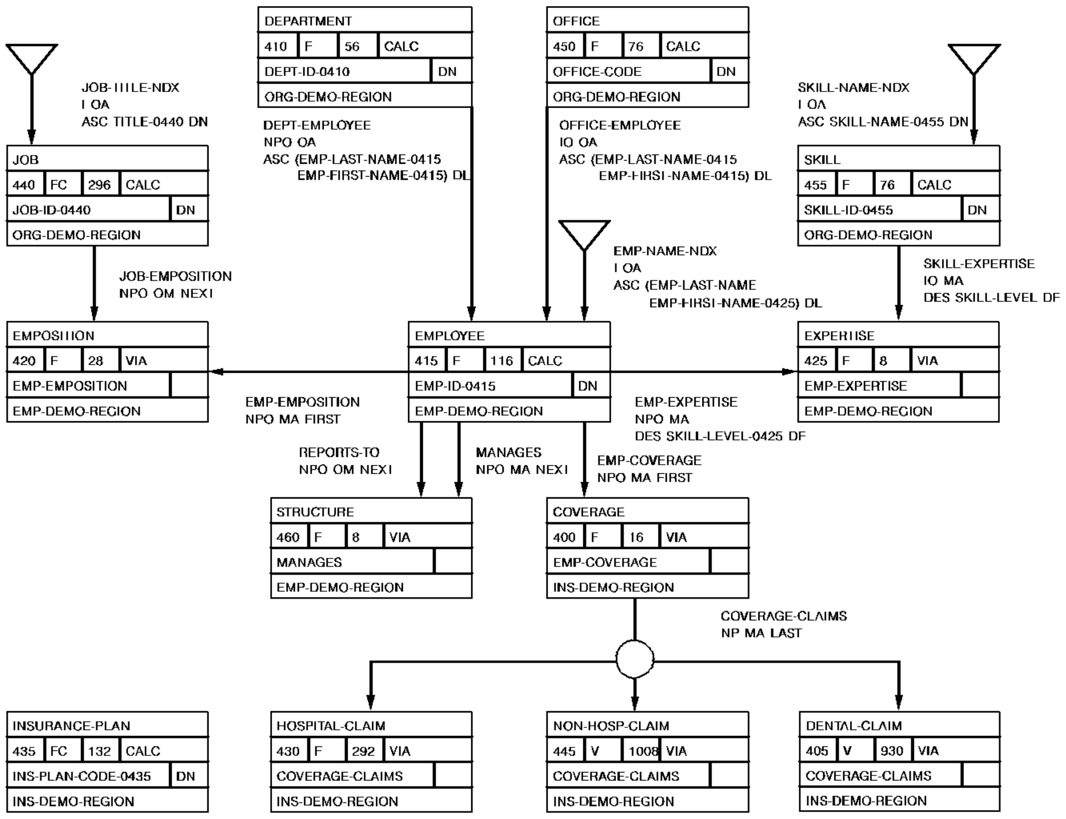
| Value | Meaning | Comments |
|---|---|---|
| X'80' | Abort was in user mode | Set by DC/UCF |
| X'40' | Program check | Set by DC/UCF |
| X'20' | No message is wanted | Set by STAE routine |
| X'10' | No SNAP is wanted | Set by STAE routine |
| X'08' | Abort task immediately | Set by #RETURN,TYPE=ABORT |
| X'01' | Continue processing at R14 address | Set by #RETURN,TYPE= CONTINUE |

# Appendix F: EMPLOYEE Data Structure Diagram

This section contains the following topics:

## Overview

The following figure is the data structure diagram for the EMPLOYEE database. This database is used for most of the examples in this document.



# Appendix G: Systems Network Architecture Considerations (SNA)

This appendix describes how to make your CA IDMS/DC Assembler program compatible with SNA protocols, allowing you to exchange information with other SNA-compatible products. The discussion will include information on:

- General SNA programming considerations in the CA IDMS/DC environment
- Allocating a session
- Starting a system task from a remote system
- Asynchronous and synchronous processing
- Sending data
- Requesting a confirmation
- Responding to a confirmation request
- Sending error information
- Changing direction: send to receive
- Receiving data
- Changing direction: receive to send
- Terminating a conversation

**What is SNA?**

Systems Network Architecture (SNA) is a set of protocols and formats that enable different types of communications products to function together in a network environment. There are no specific SNA hardware or software products. Rather, SNA is a set of rules, an architecture, to which a wide variety of products can conform.

**SNA/VTAM Line Driver**

The CA IDMS/DC SNA/VTAM line driver (VTAMLU) is a task running under CA IDMS/DC that allows your task to communicate with other SNA-compatible devices. Many SNA logical units, for example, 3270 terminals and printers, can communicate using the standard CA IDMS/DC VTAM line driver (VTAMLIN). VTAMLIN handles most SNA protocols automatically, and should be used when possible for greater operating efficiency.

The CA IDMS/DC SNA/VTAM driver, and the material covered in this appendix, should be used with logical unit configurations that require special protocol control; for example, LU6.2 logical units, and IBM 4700 or 3700 devices. The SNA protocols enabled for a specific logical unit are defined through VTAM by bind parameters in a VTAM MODENT table. These bind parameters are the only way the CA IDMS/DC SNA/VTAM driver can determine which specific SNA protocols have been established for a logical unit; care should be taken to ensure that the bind parameters accurately reflect the capabilities of the logical unit.

**Determining Compatibility and Need for Special Support**

To determine whether SNA protocols for a given logical unit are compatible with those handled by the standard VTAM driver, or if they need special protocol support from the CA IDMS/DC SNA/VTAM driver, compare the bind parameter values in the MODENT table for your logical unit to those for a 3270 device. If the MODENT values are comparable to those for a 3270, it is probable that the standard VTAM driver can handle any SNA protocols for that logical unit.

**Note:** For more information about establishing bind parameters for a logical unit, see the *System Generation Guide*.

**Support Offered by the SNA/VTAM Line Driver**

The following table lists the LU types, function management profiles, and transmission service profiles that are supported by the CA IDMS/DC SNA/VTAM driver (LU 6 is not supported).

| SNA Protocol | Types Supported by CA IDMS/DC |
|---|---|
| LU Types | 0, 1, 2, 3, 4, 6.2 |
| Function Management Profiles | 2, 3, 4, 7, 18, 19 |
| Transmission Service Profiles | 2, 3, 4, 7 |

This section contains the following topics:

## General Considerations

Before you start to write your SNA program, you should familiarize yourself with the following:

- SNA terms and their specific meanings in the CA IDMS/DC environment
- The CA IDMS/DC facilities your program needs to communicate in the SNA environment
- How SNA messages and error information are handled in the CA IDMS/DC environment

Each of these considerations is discussed on the following pages.

### SNA Terminology

The following SNA terms are used in this appendix. Special CA IDMS/DC considerations are included along with their definitions:

- A **logical unit (LU)** is a port through which you access the SNA network, a single network addressable unit (NAU). For example, an LU can be an end-user terminal, a program such as CICS or CA IDMS/DC, or a device such as a display writer.

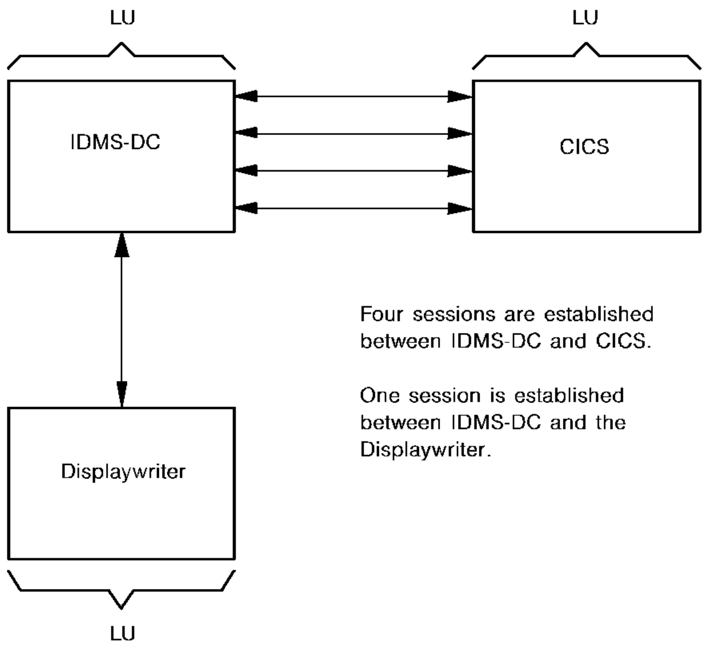
  Note: Unless otherwise specified, the discussions in this appendix apply to all LU types. Special LU6.2 considerations will be noted.

- A **session** is a logical connection between two logical units that enables the exchange of messages. Two logical units that share a single physical connection can have one or more sessions between them. Each session is represented in the CA IDMS/DC environment by a single physical terminal element (PTE)/logical terminal element (LTE) pair.

- A **conversation** is equivalent to one complete transaction between logical units. A conversation is delineated by a begin bracket and an end bracket. In the CA IDMS/DC environment, a conversation is requested by a #TREQ ALLOC statement, or is started by the remote LU, and is terminated by the LAST option on a #TREQ WRITE statement. Data is exchanged by two logical units in a conversation by using various forms of the #TREQ READ and WRITE statements.

**Multiple LU-LU Sessions**

The following figure illustrates how the SNA driver, functioning as an LU, takes part in multiple sessions. There are four sessions established between the SNA driver and CICS, and one session established between the SNA driver and a display writer. Each session can support only one conversation at a time. This configuration can support up to five simultaneous conversations: four between CA IDMS/DC and CICS, and one between CA IDMS/DC and the display writer.


LU
LU
IDMS-DC
CICS
Four sessions are established between IDMS-DC and CICS.
One session is established between IDMS-DC and the Displaywriter.
Displaywriter
LU

## Program Communications in the SNA Environment

Your program converses with other SNA network resources through #TREQ statements, in conjunction with the user I/O control block (UIOCB). CA IDMS/DC supports only basic-mode access to other SNA devices; line-mode and mapping-mode are not currently supported.

**#TREQ Command**

You use the **#TREQ command** to:

- Establish LU-LU sessions
- Initiate conversations between logical units
- Exchange data and error information between logical units
- Terminate conversations and sessions

Syntax and syntax rules for the #TREQ statement are discussed in #TREQ (see page 343).

**User Control Block**

The user I/O control block (UIOCB) contains LU-LU session information:

- Session attributes
- Conversation attributes
- Information about the data being sent and received
- Error information

**Establishing Sessions**

Sessions in the CA IDMS/DC environment can be established in three ways:

- CA IDMS/DC can automatically establish the session at system startup.
- A remote LU can establish the session.
- Your program can establish a session using the #TREQ ALLOC statement, as described later in this appendix.

When you issue a #TREQ ALLOC statement to allocate a conversation, before CA IDMS/DC can select a session for you, you must establish the UIOCB and initialize UIOCB fields with session attributes, such as which LU you want to talk to.

You also use the UIOCB to establish conversation attributes, for example, the maximum sync level that you will need (LU6.2 only).

After the conversation has begun, you use the UIOCB to obtain information about the conversation. For example, session and conversation information in the UIOCB is updated following #TREQ ALLOC or #TREQ UIOCB statements, and return codes, sense codes, data-information fields, and VTAM-information fields are updated following read requests.

### Sample User Control Block

The following figure illustrates a sample user I/O control block (UIOCB).

For the layout of the UIOCB, refer to the *DSECT Reference Guide*.

```
UOICB     DS     OF
************************************************************************
**                                                                    **
**     UIOCB      USER I/O COMMUNICATIONS BLOCK                       **
**                                                                    **
**               ## - LU6.2 ONLY                                      **
**               $$ - FOR FUTURE USE                                  **
************************************************************************

UIOLTEA   DS   A                    ADDR OF LOGICAL TERMINAL ELEMENT
                                    (CONVERSATION IDENTIFIER)

****  SESSION ATTRIBUTES *********
UIOBIND  DS   A                     ADDRESS OF BIND PARAMETERS
UIOLLU   DS   CL8                   LOCAL LU NAME (OWN_LU_NAME)
UIORLU   DS   CL8                   REMOTE LU NAME (PARTNER_LU_NAME)
UIOMODE  DS   CL8                   MODEENT NAME (MODE_NAME)
UIOSYNC  DS   X             ##      SYNC_LEVEL
UIOSYNCN EQU  X'00'                 SYNC_LEVEL = NONE
UIOSYNCC EQU  X'01'                 SYNC_LEVEL = CONFIRM
UIOSYNCS EQU  X'02'                 SYNC_LEVEL = SYNCPOINT
UIOCONV  DS   X             ##      CONVERSATION TYPE
UIOCONVB EQU  X'00'                 CONVERSATION TYPE = BASIC
UIOCONVM EQU  X'01'                 CONVERSATION TYPE = MAPPED
UIOMAPN  DS   CL24          $$      LU6.2 MAP NAME
UIOTASK  DS   CL8                   REMOTE TASK TO BE ALLOCATED (TPN)
UIOUSER  DS   CL8           ##      USER ID TO BE PASSED WITH ALLOCATE
UIOPASS  DS   CL8           ##      PASSWORD TO BE SENT WITH ALLOCATE
UIOPROFL DS   CL8           ##      PROFILE ID TO BE SENT W/ ALLOCATE
UIOINRU  DS   H                     MAX RU SIZE ON INPUT
UIOUTRU  DS   H                     MAX RU SIZE ON OUTPUT
UIORSV  DS   4H                    RESERVED
**** WHAT RECEIVED ***************
UIODAT   #FLAG X'80'                DATA
UIOERR   #FLAG X'40'        ##      ERROR  (SEND_ERROR RECEIVED)
UIOLST   #FLAG X'20'                DEALLOCATE   (SEND LAST RECEIVED)
UIOCD    #FLAG X'10'                CHANGE DIRECTION  (TIME TO SEND)
UIOCFM   #FLAG X'08'                CONFIRM  (CONFIRMATION REQUESTED)
UIOSIG   #FLAG X'04'                SIGNAL  (REQUEST_TO_SEND RECEIVED)
UIOSPT   #FLAG X'02'        $$      SYNCPOINT (TAKE_SYNCPOINT)
UIOROL   #FLAG X'01'        $$      SYNCPOINT ROLLBACK REQUIRED
UIOWREC  DS    X                    WHAT_RECEIVED
************************************************************************
UIOFMH   #FLAG X'80'                DATA CONTAINS FMH
```

```
UIODTC   #FLAG X'40'               DATA_COMPLETE (OFF = INCOMPLETE)
UIODATF  DS    X                   DATA TYPE FLAG
********* ERROR INFORMATION FIELDS ************************************
UIOURA   DS    0X
UIOUCOM  DS    XL1                 CA IDMS/DC ERROR CODE
UIOCOMPG EQU 0   GOOD COMPLETION - I/O SUCCESSFUL
UIOCOMPA EQU 8   TERMINAL OPERATOR HIT ATTN OR BREAK DURING OUTPUT
UIOCOMPL EQU 12  LOGICAL ERRORS - INVALID COMMAND SEQUENCE
UIOCOMPP EQU 16  PERMANENT I/O ERROR COMMAND SEQUENCE
UIOCOMPD EQU 20  SESSION WAS DISCONNECTED OR INTERVENTION REQ.
UIOCOMPO EQU 24  SESSION IS OUT-OF-SERVICE
UIOCOMPC EQU 28  SESSION IS CLOSED (OPEN DIDN'T WORK)
UIOCOMPI EQU 32  INVALID TRB PARAMETER LIST
UIOUCM2  DS  XL1                   SECONDARY DC ERR-CODE
UIOLGNR  EQU X'01'      ERR - LOGON ROUTINE
UIOPROF  EQU X'02'      ERR - PRIOR OPEN FAILURE
UIORTEX  EQU X'03'      ERR - RETRIES EXHAUSTED (MAX ERRS EXCEEDED)
UIONEGR  EQU X'04'      ERR - NEGATIVE RESP TO SEND DATA
UIOSRPF  EQU X'05'      ERR - SEND RESPONSE FAILED
UIONUIO  EQU X'06'      ERR - NO UIOCB ADDRESS AVAILABLE
UIOUNKI  EQU X'07'      ERR - UNKNOWN INPUT RECEIVED
UIOBBFL  EQU X'08'      ERR - BRACKET BID FAILURE
UIOWQUR  EQU X'09'      ERR - WAITING ON QUIESCE RELEASE
UIOSTSN  EQU X'0A'      ERR - MSG RESYNC FAILURE (ON SEND CHAIN)
UIOSTSR  EQU X'0B'      ERR - MSG RESYNC FAILURE REPETITIVELY
UIOPLEC  EQU X'0C'      ERR - PIPELINE EXCEEDED MAX EXCP RESPONSES
UIOPLRD  EQU X'0D'      ERR - PIPELINE READ RQST IS NOT SUPPORTED
UIOUCD   EQU X'0E'      ERR - UNIDENTIFIED NORMAL FLOW CMD RECEIVED
UIORCAF  EQU X'0F'      ERR - RESET TO CONT-ANY FAILED
UIOLUSR  EQU X'10'      ERR - UNKNOWN LUSTAT RECEIVED
UIOCNNA  EQU X'11'      ERR - CHAINED-INPUT NOT ALLOWED ON THIS PTE TYPE
UIOUNXC  EQU X'12'      ERR - UNEXPECTED COMMAND RECEIVED
UIOCNCR  EQU X'13'      ERR - CANCEL COMMAND RECEIVED
UIOCHRC  EQU X'14'      ERR - CHASE COMMAND RECEIVED
UIORCVF  EQU X'15'      ERR - RECEIVE  FAILED
UIOFMHG  EQU X'16'      ERR - FMH DEFAULT IN SYSGEN CAN'T BE USED
UIOVMMT  EQU X'17'      ERR - GENCB/MODCB FAILURE
UIOSNDF  EQU X'18'      ERR - SEND CMD FAILURE
UIOWBMS  EQU X'19'      ERR - WRITE BUFFER MISSING
UIOFMHS  EQU X'1A'      ERR - FMH OR FMH-OPTION SPECIFICATION ERROR
UIOQECR  EQU X'1B'      ERR - QEC RECV'D, USER CONTROLS OUTB CHAINING
UIOPUNK  EQU X'1C'      ERR - PTE TYPE UNKNOWN
UIOLTNA  EQU X'1D'      ERR - LAST OPTION DISALLOWED
UIOPMXW  EQU X'1E'      ERR - PIPELINE MAX NBR WRITES (1) EXCEEDED
UIOOPNS  EQU X'1F'      ERR - OPT/RQST NOT SUPPTD THIS PTE OR LU TYPE
UIOWSZX  EQU X'20'      ERR - WRT SIZ GTR PRUSZ, & CHAIN NOT ALLOWED
UIOSLUF  EQU X'21'      ERR - SEND LUS (IN LIEU NEG RESP) FAILED
UIORBKF  EQU X'22'      ERR - RESET BRACKET (SEND EB) FAILURE
UIORQRA  EQU X'23'      ERR - RQR ATTEMPTED
```

```
UIORBNS  EQU X'24'      ERR - READ BUFFER NOT SUPPORTED
UIOUCNR  EQU X'25'      ERR - OUTB USER CHANGING - NEG RESPONSE
UIONEGC  EQU X'26'      ERR - NEG RESP TO SEND COMMAND
UIONRNR  EQU X'27'      ERR - NEG RESP, SEND CHAIN, NO RECOVERY POSS
UIOLURS  EQU X'28'      ERR - LU RQST'D SHUTDOWN
UIORCCE  EQU X'29'      ERR - REQUEST CANCELLED, CONVERSTAION ENDED
UIOSIGR  EQU X'2A'      ERR - SIGNAL RECEIVED NOT RECOGNIZED
UIOIGDS  EQU X'2B'      ERR - INVALID LU6.2 GDS ID
UIOSCRM  EQU X'2C'      ERR - SEND CANCELLED, WE ARE IN RECV-MODE
UIOZLMR  EQU X'2D'      ERR - ZERO-LNG MSG RECEIVED
UIONMRT  EQU X'2E'      ERR - INVALID/MISSING REQUEST TYPE
UIOALFR  EQU X'2F'      ERR - ALLOCATE FAILED, SESSION BUSY, RETRY OK
UIOALFN  EQU X'30'      ERR - ALLOCATE FAILED, NO RETRY
UIOALFS  EQU X'31'      ERR - ALLOCATE FAILED, SYNCLEVEL NOT SUPPORTED
UIOUNBD  EQU X'32'      ERR - UNBIND RECEIVED
UIOSNDE  EQU X'33'      ERR - LU6.2 SEND ERROR RECEIVED
UIOABND  EQU X'34'      ERR - LU6.2 SEND ABEND RECEIVED
UIOXLIM  EQU X'35'      ERR - LIMIT ON INPUT EXCEEDED, READ FAILED
UIOEBR   EQU X'36'      END BRACKET RECEIVED - DEALLOCATE NORMAL
UIOURTC  DS  XLI                   VTAM RTNCD
UIOUFDB  DS  XLI                   VTAM FDBK2
UIOUSEI  DS  XLI                   VTAM SENSE INFO
UIOUSMI  DS  XLI                   VTAM SENSE MODIFIER
UIOUUSI  DS  XL2                   VTAM USER SENSE INFO
UIORSV1  DS  XL4                   RESERVED
UIOUSIG  DS  XL4                   SIGNAL DATA - EXPD-FLOW-CMD
UIOURAL EQU *-UIOURA     LENGTH OF ERROR INFO FIELDS
UIORSV2  DS  XL27                  RESERVED
UIODWORK DS  XL1                   WORK BYTE RESERVED FOR CA IDMS/DC
UIOCBL  EQU *-UIOCB     LENGTH OF UIOCB
```

## Error Handling

Information about the outcome of your request is returned to your program in several different ways:

- The outcome of any request is indicated in register 15. In most cases, register 15 is all that needs to be checked.
- For debugging purposes, the following fields in the UIOCB contain additional information:
  - The UIOUCM2 field of the UIOCB contains CA IDMS/DC secondary error codes.
  - The UIOURTC field of the UIOCB contains VTAM return code and feedback information.
  - The UIOUSEI, UIOUSMI, and UIOUUSI fields of the UIOCB contain SNA sense codes.

The following table lists the sense codes CA IDMS/DC sends to the remote LU to inform the remote system of errors encountered in conversation processing. Sense codes are specified with a 4-byte hexadecimal value. CA IDMS/DC sends the following SNA sense codes to inform the remote system of errors encountered in conversation processing.

| Sense code | CA IDMS/DC definition | SNA meaning |
|---|---|---|
| 10086021 | Task not defined to CA IDMS/DC | Allocation error, TPN not recognized |
| 084C0000 | Task out of service | Allocation error, TPN not available |
| 10086041 | Sync-level not supported | Sync-level not supported |
| 080F6051 | Security violation | Security not valid |
| 08640000 | Task abended | Deallocate abend |
| 08890000 | #TREQ WRITE,OPTNS=ERROR sent | Send error request |
| 08890101 | Invalid LU6.2 GDS-ID | Invalid GDS-ID |
| 08460000 | ERP message forthcoming | ERP message forthcoming |
| 08240000 | Rollback requested | Syncpoint rollback |
| 08130000 | Bracket bid reject (no RTR) | Allocate failure |
| 08010000 | Resource unavailable (busy) | Allocate failure |
| 08060000 | Resource unknown (LU not defined) | Allocate failure |
| 08210000 | Invalid session parameters | Allocate failure |

## SNA Functions in a CA IDMS/DC Environment

The remainder of this appendix will discuss how to perform SNA functions in a CA IDMS/DC environment. Each SNA function, for example, ALLOCATE, will be accompanied by a discussion of how to implement the specific protocols using the #TREQ statement and the UIOCB.

The following table lists the SNA functions supported by the CA IDMS/DC SNA/VTAM driver and their corresponding #TREQ statements.

**Note:** For more information about the #TREQ statement, see #TREQ (see page 343).

| SNA function | CA IDMS/DC #TREQ statement |
|---|---|
| ALLOCATE<br>LU_NAME<br>MODE_NAME<br>TPN<br>SECURITY<br>(PROGRAM<br>(USER ID,<br>PASSWORD))<br>TYPE (CONVERSATION)<br>RETURN_CONTROL | #TREQ ALLOC<br>UIOCBA<br>OPTNS=<br>IMM/CONN/ANY<br>WAIT/NOWAIT<br>LTERMID |
| CONFIRM<br>RESOURCE<br>RETURN_CODE | #TREQ WRITE<br>OPTNS=CONFIRM<br>LTEADDR |
| CONFIRMED RESOURCE<br>RESOURCE | #TREQ WRITE<br>OPTNS=CONFIRM<br>LTEADDR<br>#TREQ<br>(any request except<br>#TREQ WRITE, OPTNS=ERROR) |
| DEALLOCATE RESOURCE<br>TYPE (SYNC_LEVEL)<br>TYPE<br>LOG_DATA<br>TYPE (LOCAL)<br>RESOURCE | #TREQ WRITE<br>OPTNS=LAST<br>LTEADDR<br>#TREQ WRITE<br>OPTNS=ABEND<br>LTEADDR<br>SENSE<br>LOGDATA<br>OUTLEN<br>LTEADDR<br>#TREQ DISC<br>LTEADDR |

| SNA function | CA IDMS/DC #TREQ statement |
|---|---|
| GET_ATTRIBUTES<br>RESOURCE<br>GET_TYPE<br>RESOURCE | #TREQ UIOCB<br>UIOCBA<br>LTEADDR |
| POST_ON RECEIPT<br>RESOURCE<br>WAIT RESOURCE_LIST<br>RESOURCE | All #TREQ requests<br>#WAIT |
| PREPARE_TO_RECEIVE<br>RESOURCE | #TREQ WRITE<br>OPTNS=INVITE |
| RECEIVE_AND_WAIT<br>DATA<br>LENGTH<br>FILL<br>WHAT_RECEIVED<br>RESOURCE<br>RETURN_CODE | #TREQ GET<br>INAREA<br>MAXIN<br>INLEN<br>OPTNS=<br>LL<br>NOCHASM<br>LTEADDR<br>OPTNS=<br>INFMHY<br>INFMHN |
| REQUEST_TO_SEND<br>RESOURCE | #TREQ WRITE<br>OPTNS=SIGNAL<br>LTEADDR |

| SNA function | CA IDMS/DC #TREQ statement |
|---|---|
| SEND_DATA | #TREQ WRITE |
| DATA | OUTAREA |
| LENGTH | OUTLEN |
| RESOURCE | LTEADDR |
| RETURN_CODE | OPTNS= |
| | OUTFMHY |
| | OUTFMHN |
| | OPTNS=CHNCONT |
| SEND_ERROR | #TREQ WRITE |
| TYPE (PROGRAM) (SVC) | OPTNS=ERROR |
| LOG_DATA | SENSE |
| RESOURCE | LOGDATA |
| RESOURCE_CODE | OUTLEN |
| | LTEADDR |

## Allocating a Session

```
ALLOCATE LU_NAME
         MODE_NAME
         SYNC_LEVEL
         TPN
         SECURITY (PROGRAM (USER_ID, PASSWORD))
         TYPE (CONVERSATION)
         RETURN_CONTROL (WHEN_SESSION_ALLOCATED)
         RETURN_CONTROL (IMMEDIATE)
         RESOURCE
         RETURN_CODE
```

The #TREQ ALLOC statement allows you to allocate a conversation with another logical unit. In most cases, CA IDMS/DC selects a session for you from sessions defined at system generation. The system bases its selection on session *attributes* you have established in the UIOCB. You should initialize the following UIOCB fields before you allocate a session:

- The name of the LU (UIORLU) with which your program will be communicating.
- In some special cases your program may need to specify the name of a MODEENT table (UIOMODE), requesting a specific session for the conversation. Most programs do not have to specify UIOMODE.

Note: For more information about session modes, see the *System Generation Guide*.

- The maximum sync level (UIOSYNC) your task will need (LU6.2 only).

Instead of coding these parameters and letting CA IDMS/DC select a session for you, you can use the LTERMID parameter of the #TREQ ALLOC statement to allocate a specific session, identified by the logical terminal name of the other LU. For example:

```
#TREQ ALLOC,LTERMID=LTERMIDA
```

When LTERMID is specified, the UIORLU and UIOMODE fields in the UIOCB are ignored.

## Establishing Conversation Attributes

For *LU6.2 conversations only*, you also use the UIOCB to establish conversation attributes. Conversation attributes include:

- Security information to be passed to the remote system, for example, user id (UIOUSER), and user password (UIOPASS). These fields are valid only if security is enforced on the remote system.
- Whether the conversation is basic (UIOCONVB) or mapped (UIOCONVM). In most situations, your conversation will be in mapped mode. Unmapped (basic) mode is used with remote LU6.2 logical units that do not have an application programming interface (for example, an IBM display writer), or for system level service manager programs.
- The (optional) name of the remote task (UIOTASK).

  Note: For more information about the UIOTASK field, see the Starting a Task on a Remote Logical Unit (see page 529) later in this appendix.

## Issuing the #TREQ ALLOC Statement

After you have set the session and conversation attributes in the UIOCB, you must issue a #TREQ ALLOC statement to allocate the session.

**Coding Considerations**

You should consider the following parameters when coding your #TREQ ALLOC statement:

- The OPTNS=ANY/CONN/IMM parameter of the #TREQ ALLOC statement establishes criteria for choosing a session. The session you need can be in one of three states:
  - **Immediately available**—The session has already been established with the requested LU and is not currently in use.

    Note: (LU6.2 only); the session must be a contention winner to be considered immediately available.

    For more information about contention winners, see the *System Generation Guide*.
  - **Disconnected**—The session has not yet been established.
  - **Busy**—The session has been established, but is currently allocated to another logical unit. The session will become *immediately available* when that logical unit ends its conversation.

  The options on the #TREQ ALLOC statement are as follows:
  - **ANY** (default) specifies that CA IDMS/DC tries to allocate a session in the following order:

    A session that is *immediately available* and currently unused.

    A session that is *disconnected*.

    A session that is *busy*; CA IDMS/DC will wait for a busy session and return control to your program once the session is allocated.
  - **CONN** requests CA IDMS/DC not to wait for a busy session. CA IDMS/DC will first attempt to allocate an immediately available session, then a disconnected session.
  - **IMM** specifies that only immediately available sessions are acceptable for the allocation request.
- You can specify whether your #TREQ ALLOC request is made synchronous (default) by specifying OPTNS=WAIT or asynchronous by specifying OPTNS=NOWAIT.

  Note: If you specify OPTNS=ANY, do not request asynchronous processing with OPTNS=NOWAIT. OPTNS=ANY implies that the request may wait for a busy session.
- The UIOCB parameter of the #TREQ ALLOC statement establishes a UIOCB for the conversation.

### Example of LU-LU Session Allocation

The following example illustrates how you would allocate an LU-LU session, establishing the UIOCB, and setting session and conversation attributes:

- The first statement obtains storage for the UIOCB.
- The next statement establishes the remote logical unit.
- The next four statements establish LU6.2 conversation attributes.
- The #TREQ ALLOC statement allocates the session, initiates the conversation, and names the UIOCB.

```
UIOSTG   #GETSTG TYPE=(USER,LONG),PLIST=*,LEN=UIOLEN,INIT=X'00',       *
               STGID=UIOCBD,ADDR=(R1)
*                                         SESSION ATTRIBUTES
ATTR     MVC   UIORLU,=C'VTMF0178'     REMOTE LU
*                                         CONVERSATION ATTRIBUTES
         MVC   UIOUSER,=C'BRANCH01'    USER ID: DENVER BRANCH
         MVC   UIOPASS,=C'DENPR  '     USER PASSWORD: DENVER
         MVI   UIOCONV,UIOCONVM        MAPPED MODE
         MVI   UIOSYNC,UIOSYNCC        MAXIMUM SYNC-LEVEL
         #TREQ ALLOC,UIOCBA=UIOCB,COND=ALL
```

### After Issuing #TREQ ALLOC

After you have issued your #TREQ ALLOC request, you need to perform the following:

- Check the value in register 15:
  - **If register 15 contains a nonzero value**, the allocation request failed. The UIOUCM2 field in the UIOCB indicates whether the problem is permanent or temporary:
    - If CA IDMS/DC returns UIOALFR to the UIOUCM2 field, the allocate request was denied due to a temporary problem; for example, CA IDMS/DC was unable to wait for a busy session. In this case, you should issue the #TREQ ALLOC request again.
    - If CA IDMS/DC returns UIOALFN to the UIOUCM2 field, a permanent error was encountered.
    - If CA IDMS/DC returns UIOALFS to the UIOUCM2 field, the specified sync level for the conversation is not supported. This is a permanent error.

  - **If register 15 contains 0**, the session has been successfully established. Register 1 contains the logical terminal address (LTEADDR) of the remote LU. The logical terminal address (also stored in UIOLTEA) must be specified on all subsequent #TREQ requests in that session because a single task can have conversations with many logical units.

- If the #TREQ request was asynchronous (OPTNS=NOWAIT), you must issue a #TREQ CHECK statement before you make any further I/O requests. Your program must specify the LTE address of the remote LU (UIOLTEA) to identify the conversation.

## Starting a Task on a Remote Logical Unit

**Non-LU6.2 Sessions**

**For non-LU6.2 sessions**, if the UIOTASK field in the UIOCB contains a task name (is nonzero and nonblank) when a #TREQ ALLOC is issued, CA IDMS/DC will automatically send the task code to the remote system immediately after the session is established.

**LU6.2 Sessions**

**For LU6.2 sessions**, if the UIOTASK field in the UIOCB contains a task name (is nonzero and nonblank) when a #TREQ ALLOC is issued, CA IDMS/DC will automatically send the LU6.2 allocate request to the remote system, requesting the remote system to start the named task.

**Requests from Remote Units**

When CA IDMS/DC receives an allocate request from a remote LU6.2, it does the following:

- If the allocate request contains a nonzero and nonblank value in the user id (UIOUSER) or password (UIOPASS) fields, CA IDMS/DC will run the signon task for that session.
- The task identified in the allocate request is then attached.
- The conversation type (UIOCONV) and sync-level (UIOSYNC) are also passed by the allocate request and moved into the UIOCB.

Any errors encountered while processing a remote allocation request for example, task-not-defined or security violations, are reported to the remote system through an SNA sense code.

Note: For more information about sense codes, see Error Handling (see page 521) in this appendix.

## Starting a Task from a Remote System

GET_ATTRIBUTES

GET_TYPE

When your conversation is started from a remote LU, you must issue a #TREQ UIOCB statement before issuing any other #TREQ statements. The #TREQ UIOCB statement establishes a UIOCB for CA IDMS/DC to maintain session attributes and status information.

If the conversation was started from a remote system, the LTEADDR parameter can be left off, since the LTE address defaults to the LTE that started the task (that of the remote system).

CA IDMS/DC fills all session attribute fields upon completion of a #TREQ UIOCB or #TREQ ALLOC request.

## Synchronous and Asynchronous Processing

POST_ON_RECEIPT

WAIT RESOURCE_LIST

The statements used to establish SNA sessions and to exchange data can be issued as either synchronous or asynchronous requests.

Note: For more information about synchronous and asynchronous processing, see the #TREQ (see page 343).

When establishing a conversation you can request:

- **Synchronous** processing by using the OPTNS=WAIT parameter of the #TREQ ALLOC statement.
- **Asynchronous** processing by using #TREQ ALLOC,OPTNS=NOWAIT. You must issue a #TREQ CHECK, specifying the LTE address of the remote LU, prior to any other I/O requests for that conversation.

Note: For more information about the #TREQ ALLOC statement, the Allocating a Session (see page 525) in this appendix.

When you are issuing #TREQ input and output statements, you can request:

- **Synchronous** processing by using #TREQ GET, PUT, and PUTGET.
- **Asynchronous** processing by using #TREQ WRITE, READ, and WRITREAD. The #WAIT statement is used to wait on an ECB list. All asynchronous requests must be followed by a #TREQ CHECK statement before any other I/O requests can be made for that session.

## Sending Data

```
SEND_DATA DATA
          LENGTH
          RESOURCE
          RETURN_CODE
```

You can use any #TREQ WRITE, PUT, PUTGET, or WRITREAD request to send data to another LU in a conversation.

If the length of the data you are sending (OUTLEN) is larger than the SNA maximum request unit size (UIOTRU), CA IDMS/DC will chain the output automatically.

### LU6.2 Considerations for Sending Data

For *LU6.2-mapped* conversations, CA IDMS/DC appends a generalized data stream ID (GDS ID) to the data.

For *LU6.2 unmapped* conversations, you must supply the correct GDS ID and attach it to the data.

**Note:** For more information about GDS IDs, see the IBM SNA documentation.

## Non-LU6.2 Considerations for Sending Data

For *non-LU6.2* conversations, specifying OUTFMHY or OUTFMHN indicates whether or not a function management header (FMH) has been added to the outbound message:

- **OUTFMHY** specifies that you have included an FMH at the beginning of the write buffer that should be used instead of any sysgen defaults.
- **OUTFMHN** specifies that no default FMH should be added to the outbound message and that you have not provided an FMH.

The CHNCONT parameter (non-LU6.2 conversations only) specifies that your task is sending a chain of outbound messages and that the current message

is not the last in the chain. Not specifying CHNCONT after it has been specified once indicates the final chain element.

## Requesting a Confirmation

```
CONFIRM RESOURCE
        RETURN_CODE
```

If you want to request a confirmation, for any application-defined reason, you can include the CONFIRM option of the #TREQ WRITE, PUT, PUTGET, or WRITREAD statements. Your program must specify the LTE address of the remote logical unit to identify the conversation.

Specifying OPTNS=CONFIRM sends a confirmation request to the remote LU. The request is posted as complete as soon as it is received; a separate read statement is not necessary to get the confirmation. CA IDMS/DC sets the send-error received flag (UIOERR) on if the reply is negative.

The CONFIRM option can be specified with or without data (OUTLEN=0). Syntax and syntax rules for OPTNS=CONFIRM are described in Data Manipulation Language Statements (see page 73).

You can request a change of direction with the confirmation request by specifying OPTNS=(INVITE,CONFIRM).

You can also request confirmation before a conversation is terminated by specifying OPTNS=(LAST,CONFIRM).

Note: For more information about terminating a conversation, see Terminating a Conversation (see page 538) in this appendix.

For *non-LU6.2 sessions*, the following considerations apply:

- If the bind parameters issued at system generation indicate that the definite response protocol is supported, CA IDMS/DC will always request a definite response type1 (RDR1) on the last or only elements.
- If your program specifies OPTNS=CONFIRM, CA IDMS/DC will request a definite response type2 instead of type1.

## Responding to a Confirmation Request

```
CONFIRMED_RESOURCE

SEND_ERROR
```

After your program has received a confirmation request (UIOCFM is set on), your program can:

- Send a positive response by specifying OPTNS=CONFIRMED on a write request
- Allow CA IDMS/DC to send a positive response automatically the next time you make a request (with the exception of write requests specifying OPTNS=ERROR)
- Send a negative response by specifying OPTNS=ERROR on a write request

## Sending Error Information

```
SEND_ERROR TYPE (PROGRAM) (SVC)
           LOG_DATA
           RESOURCE
           RETURN_CODE
```

Your program can send error information to an LU by specifying OPTNS=ERROR on a WRITE or PUT request. You cannot issue a #TREQ PUTGET or WRITREAD request because the program remains in the send state after the error request is issued. Your program must specify the LTE address of the remote LU (UIOLTEA) to identify the conversation.

The error information is sent in the form of an 8 character hexadecimal SNA sense code, specified by the SENSE parameter on a write request. The default sense code is X'08890000'.

Note: For more information about sense codes, see Error Handling (see page 521) in this appendix.

Upon receipt of an error, CA IDMS/DC moves the sense code to the UIOCB. CA IDMS/DC indicates that an error has been received by setting an error flag (UIOERR) in the UIOWREC (what-received) field of the UIOCB and provides more specific information about the error in the secondary codes (UIOUCM2).

Note: Register 15 is not set in response to a SEND_ERROR verb. Therefore, UIOWREC should be examined if the possibility of UIOERR exist.

**LU6.2 Sessions**

For an LU6.2 session, if you send the error request while you are in the receive state, all input is purged until a change-direction indicator is received, and then CA IDMS/DC sends the error information.

Note: For more information about changing direction, see Changing Direction: Send to Receive (see page 534) in this appendix.

After sending error information, your program will be in the send state. You can then send additional data to the remote LU.

You can send log data along with the error information by using the LOGDATA parameter. If the remote system supports log data, the data will be logged onto the remote system when it receives the send-error request. LOGDATA specifies the address of the data buffer. You must also specify the OUTLEN parameter to indicate the length of the data.

**Non-LU6.2 Sessions**

For a non-LU6.2 session, CA IDMS/DC sends the sense code in a negative response if your task is in the receive state and in an LUSTAT command if your task is in the send state. If your program issues an error request while your task is in the receive state, all input is purged until a change-direction indicator is received. Your program must specify the LTE address of the remote LU (UIOLTEA) to identify the conversation.

## Changing Direction: Send to Receive

PREPARE_TO_RECEIVE

Your program can change from the send state to the receive state in either of the following ways:

- Implicitly, by issuing any type of read request (#TREQ READ, GET, WRITREAD, PUTGET). CA IDMS/DC automatically sends a change-direction indicator to the remote system before it issues the read request.

- Explicitly, by using the OPTNS=INVITE parameter on any write request.

The change-direction indicator is sent with data for all #TREQ PUTGET and WRITREAD requests, and without data for all GET and READ requests.

Your program must specify the LTE address of the remote LU (UIOLTEA) to identify the conversation.

## Receiving Data

```
RECEIVE_AND_WAIT DATA
                 LENGTH
                 FILL
                 WHAT_RECEIVED
                 RESOURCE
                 RETURN_CODE
```

To read data sent from another LU, your program must issue some form of read request (#TREQ READ, GET, PUTGET, or WRITREAD). CA IDMS/DC buffers all input received from a logical unit. Your program can issue multiple read statements until all of the data in the buffer has been transferred to your program.

**Parameters Applying to Incoming Data**

The following parameters apply to incoming data:

- The INAREA parameter specifies the location of the input data stream.
- The INLEN parameter specifies the actual length of the input data stream.
- The MAXIN parameter specifies the maximum length of data your program can receive. CA IDMS/DC never truncates data; if the length of the input data stream exceeds the MAXIN parameter in your READ statement, CA IDMS/DC will buffer the data so that it will be available for your next read request.
- The LOCATE parameter requests CA IDMS/DC to allocate a buffer the exact size of the input data stream. Register 1 contains the address of the buffer that will contain the input data. The INLEN parameter can be used to indicate the actual amount of data received. The LOCATE parameter and the INAREA and MAXIN parameters are mutually exclusive.

If all of the input has been transferred from the data buffer to your program on completion of a read request, the data-complete-flag (UIODC) will be set on. In general, you should always continue issuing read requests until the change-direction (UIOCD) or last (UIOLST) flag has been set.

### LU6.2 Conversations

For LU6.2 conversations, CA IDMS/DC can receive only one type of input with each request. For example, if CA IDMS/DC receives input that contains data, a change of direction indicator, and a confirm request, you must issue two read requests in order to get all the information you need:

- **First read request**—Reads the data. The data (UIODAT) and data-complete (UIODTC) flags in the UIOCB are set on to indicate that all of the data has been received and given to your program (assuming the buffer was large enough to hold all of the data). If the input buffer is not large enough to hold all of the data, CA IDMS/DC will buffer the data so that it will be available to your next read request.
- **Second read request**—Processes the change of direction and confirmation requests by setting on the change-direction (UIOCD) and confirmation-requested (UIOCFM) flags in the UIOCB.

LU6.2 data is always passed in *LU6.2 logical records*, made up of a header and the user data. The header consists of a 2-byte length field and a 2-byte generalized data stream ID (GDS ID).

#### LU6.2 Mapped Conversations

During LU6.2 *mapped* conversations, CA IDMS/DC removes the header from the logical record (OPTNS=LL).

#### LU6.2 Unmapped Conversations

During LU6.2 *unmapped* (basic) conversations, a read request can specify the following options:

- **LL** specifies that CA IDMS/DC will pass one LU6.2 logical record, without removing the header.
- **NOCHASM** requests CA IDMS/DC to pass single chain elements (RUs) to your task one at a time, regardless of logical record, without assembling the chain into a buffer area. The last (or only) chain element is indicated by the UIODTC flag.
- **Not specifying either option** requests CA IDMS/DC to read an input data stream of the length specified by the MAXIN operand, regardless of whether an entire logical record is sent. The read is complete when the amount of data specified by MAXIN has been read, or when the end-of-chain has been indicated.

**Non-LU6.2 Conversations**

For non-LU6.2 conversations, the following considerations apply:

- All currently available data and read information is passed to your program in one read, unless the buffer is not large enough to hold all of the data.
- A read request can specify either OPTNS=NOCHASM or leave this parameter unspecified:
  - **Specifying OPTNS=NOCHASM** indicates that an inbound chain is passed to your task a single chain element (RU) at a time, without assembling the chain into a buffer. The last (or only) chain element is indicated by the UIODTC flag.
  - **Not specifying this option** requests a read of a single buffer of the length specified in MAXIN. All SNA chains are assembled into a single buffer; the read is completed when either the specified length of data or the RU marked as the end of the chain is received. The data-complete (UIODTC) flag is set when the end of the chain is received.
- Your program can indicate how function management headers (FMH) are handled on input by specifying INFMHY or INFMHN:
  - **INFMHY** indicates that function management headers are passed to your task along with the input data stream. The UIOFMH flag in the UIOCB is set on to indicate the presence of an FMH in the data stream.
  - **INFMHN** requests CA IDMS/DC to remove any incoming FMH from the input data stream before the data is passed to your task.

## Changing Direction: Receive to Send

REQUEST_TO_SEND

Normally, your program remains in the receive state until the remote LU sends a change-direction indicator.

Your program can request a change of direction from the receive state to the send state by specifying OPTNS=SIGNAL on a write request. The SIGNAL option sends a change-direction signal code of X'00010000'.

If your program issues a write request while it is in the receive state, CA IDMS/DC sends the signal command, requesting change of direction, to the remote LU. CA IDMS/DC posts your program's write request as successfully completed with a logical error (R15 = 0C) and an error code in the UIOUCM2 field of the UIOCB.

The UIOUCM2 field indicates that your program tried to send data while in the receive state (UIOSCRM), and that CA IDMS/DC sent the change-direction signal for you. You must continue to send read requests until the remote LU sends a change-direction signal (UIOCD).

## Terminating a Conversation

```
DEALLOCATE RESOURCE TYPE (SYNC_LEVEL)
                    TYPE (ABEND_PROGRAM)
                    LOG_DATA (VARIABLE)
                    TYPE (LOCAL)
```

A conversation between CA IDMS/DC and another LU can be terminated in the following ways:

- Your program can request a **normal termination** of the conversation by specifying OPTNS=LAST on a write request.
- Your program can notify the remote LU that it is **terminating abnormally** by specifying OPTNS=ABEND on a write request.
- **The remote LU can terminate the conversation**. CA IDMS/DC sets the UIOLST (send last received) flag in the UIOCB.

Your program must specify the LTE address of the remote task (UIOLTEA) to identify the conversation.

The session that is being maintained between CA IDMS/DC and the remote LU is not closed, but remains available to be allocated to another conversation. This eliminates the overhead of reestablishing another session.

If you want to start another conversation after you have ended the current one, you must allocate a new conversation to the session.

Note: For more information about allocating a conversation, see Allocating a Session (see page 525) in this appendix.

### Normal Termination

To end a conversation between two logical units normally, specify OPTNS=LAST on a write request. CA IDMS/DC notifies the remote system, frees the session to make it available for other conversations, and, for LU6.2 conversations, performs a signoff for the remote LU.

The request to terminate a conversation can be made with or without data (OUTLEN=0).

You can request confirmation of the termination request by specifying OPTNS=(LAST,CONFIRM) on a #TREQ WRITE or PUT request. CA IDMS/DC notifies the remote system and will wait to free the session and perform the signoff until a positive confirmation is received.

If your task ends or abends before the conversation terminates normally, CA IDMS/DC performs the ABEND operation.

## Abnormal Termination

You can notify the remote system that your task is abending and that the conversation has ended by using the ABEND option of the #TREQ WRITE or PUT statements. CA IDMS/DC notifies the remote system, terminates the conversation, frees the session, and, for LU6.2 conversations only, signs off the remote LU.

**LU6.2 Conversations**

For LU6.2 conversations, CA IDMS/DC can pass log data along with the ABEND notification. The LOGDATA parameter locates the buffer containing the data. If the remote LU6.2 system supports LOGDATA, the data will be logged on to the remote system when the ABEND notification is received. If you specify LOGDATA, you must also include the OULEN parameter to indicate the length of the data.

**Non-LU6.2 Conversations**

For non-LU6.2 conversations, the SENSE option overrides the default sense code (X'08640000'; task abended).

## Terminating a Session

You can terminate a *non-LU6.2 session* between CA IDMS/DC and another LU by using the #TREQ DISC (disconnect) statement. The #TREQ DISC request must be followed by a #TREQ CHECK request. Your program must specify the LTE address of the remote LU (UIOLTEA) to identify the conversation.

# Appendix H: 18-Byte Communications Blocks

This section contains the following topics:

## Overview

As an alternative to using the 16-byte IDMS DB communications blocks, you can specify 18-byte blocks. The difference between 16-byte blocks and 18-byte blocks is that an 18-byte block contains an additional 18-byte filler field, and the following fields are 18 bytes instead of 16 bytes:

- RECNAME
- AREANAME
- ERRORSET
- ERRORREC
- ERRAREA

This appendix describes where to specify an 18-byte communications block and contains figures showing these blocks.

**Note:** For more information about the fields in IDMS DB communications blocks, see Communications Blocks and Error Detection (see page 33).

**Where to Specify the 18-Byte Block**

For Assembler, you specify an 18-byte communications block by using the @SSC120 statement in place of the @SSCTRL statement.

**Note:** For more information, see @SSCTRL (see page 421).

**18-Byte IDMS-DB Block**

The following figure shows the 18-byte IDMS DB communications block:

IDMS COMMUNICATIONS BLOCK

| | | Field | Data Type | Length (bytes) | Initial Value |
|---|---|---|---|---|---|
| * | 0 7 | PROGRAM-NAME | Alphanumeric | 8 | Program Name |
| | 8 11 | ERROR-STATUS | Alphanumeric | 4 | '1400' |
| | 12 15 | DBKEY | Binary | 4(Fullword) | 0000 |
| | 16 33 | RECORD-NAME | Alphanumeric | 18 | Spaces |
| | 34 51 | AREA-NAME | Alphanumeric | 18 | Spaces |
| | 52 69 | FILLER | Alphanumeric | 18 | Spaces |
| | 70 87 | ERROR-SET | Alphanumeric | 18 | Spaces |
| | 88 105 | ERROR-RECORD | Alphanumeric | 18 | Spaces |
| | 106 123 | ERROR-AREA | Alphanumeric | 18 | Spaces |
| ** | 124 127 | PAGE-INFO | Binary | 4(Fullword) | 0000 |
| | 124 ... 223 | IDBMSCOM | Alphanumeric | 100 | Spaces |
| | 224 227 | DIRECT-DBKEY | Binary | 4(Fullword) | 0000 |
| | 228 234 | DATABASE-STATUS | Alphanumeric | 7 | Spaces |
| | 235 | FILLER | ... | 1 | ... |
| | 236 239 | RECORD-OCCUR | Binary | 4(Fullword) | 0000 |
| | 240 243 | DML-SEQUENCE | Binary | 4(Fullword) | 0000 |
| | 244 299 | FILLER | Alphanumeric | 56 | Spaces |

* word aligned

** PGINFGRP overlays bytes 124 and 125 and PGINFDBK overlays bytes 126 and 127. Both of these fields are binary datatype each having a length of two bytes. Suggested initial values for both are 00. Together these two fields represent PGINFO.

# Appendix I: Online Debugger Syntax

This section contains the following topics:

## General Registers Symbols

**General registers** include the registers used by the program at the time of execution and the registers used by the DC/UCF system. The program status word (PSW) and register definitions are always preceded by a colon (:) and are specified by these symbols:

- **:PSW** for the current program status word
- **:Rn** for the user program register at the time of interrupt, where *n* represents the number of the register and can have a value of 0 through 15
- **:REGS** for all user program registers at the time of interrupt
- **:SRn** for a DC/UCF system register at the time of interrupt, where *n* represents the number of the register and can have a value of 0 through 15
- **:SREGS** for all DC/UCF system registers at the time of interrupt

**Important!** A single debug expression can reference only one general register.

## DC/UCF System Symbols

Certain DC/UCF system symbols also function as debugger entities, and you can refer to them during a debugging session. A colon (:) must precede each symbol. These are the valid symbols:

**:BAT**

Specifies the base address table for session.

**:CSA**

Specifies the DC/UCF common storage area.

**:DLB**

Specifies the debug local block, control block required for debugging session.

**:LTE**

Specifies the current logical terminal element.

**:PTE**

Specifies the current physical terminal element.

**:TCE**

Specifies the current task control element.

**:VECT**

Specifies the vector table for debugger.

**Important!** A single debug expression can reference only one system entity.

## Address Symbols and Markers

| Symbol | Symbol Name | Designated Location |
|---|---|---|
| @ | At sign | Absolute address |
| $ | Dollar sign | Load address |
| ¢ | Cent sign | Address of current dialog process |

## User Symbols

- **:DRn** for a debugger general register, where *n* represents the number of the register and can have a value of 0 through 15
- **:DREGS** for all debugger registers
- **:H1** and **:H2** for halfword 1 and halfword 2
- **:F1** and **:F2** for fullword 1 and fullword 2
- **:UCHR** for a 48-byte character area

  You can also refer to specified sections of this area:

  - **:UC0**, the first 16 bytes
  - **:UC16**, the next 16 bytes
  - **:UC32**, the last 16 bytes

## Program Symbols

### Syntax: Data Field Names

```
►►── data-field-name ─┬──────────────────────┬──────────────►◄
                      └ IN ─┬─ record-name ──┘
                        OF ─┘
```

### Syntax: Line Numbers

```
►►── # line-number ─────────────────────────────────────────►

►─┬──────────────────────────────────────────────────────────┬─►◄
  └ IN ─┬─┬─ current-process-name ──────────────────────────┤
    OF ─┘ └─ included-module-name ─┬─────────────────────────┤
                                   └ OCCurrence occurrence-number ┘
```

### Syntax: Qualifying Program Symbols

```
►►── process-name - . - program-symbol ─────────────────────►◄
```

## Expression Operators

| Operator | Meaning |
|---|---|
| + | Addition |
| - | Subtraction |

| Operator | Meaning |
|---|---|
| * | Multiplication |
| / | Division |

## Delimiters

| Delimiter | Meaning |
|---|---|
| * | Asterisk |
|  | Blank |
| , | Comma |
| = | Equal sign |
| ! | Exclamation point |
| - | Hyphen |
| % | Percent sign |
| . | Period |
| + | Plus sign |
| / | Slash |

## Debugger Commands

### Syntax: AT

**ADD Format**

```
►►── AT debug-expression ──────────────────────────────────────────────────►

 ►─┬─────────────────────────────────────┬─┬──────────────────────────────┬─►
   └─ BEFore ─┬─ MAXimum ◄ ──────────┬───┘ └─ AFTer ─┬─ 0 ◄ ──────────────┬┘
              └─ execution-count ────┘               └─ execution-count ──┘

 ►─┬────────────────────────────────────┬─┬─────────────┬─────────────────►◄
   └─ EVEry ─┬─ 1 ◄ ─────────────────┬──┘ ├─ ON ◄ ──────┤
             └─ execution-count ─────┘    └─ IGNore ────┘
```

**INQUIRE Format**

```
►►── AT ─┬─ ALL ──────────────┬─┬─ INQuire ─┬──────────────────────────────►◄
         └─ debug-expression ─┘ ├─ ON ──────┤
                                ├─ IGNore ──┤
                                └─ OFF ─────┘
```

## Syntax: DEBUG

**ADD format**

```
►►── DEBug ─┬─ PROgram ◄ ─┬─ entity-name ─┬──────────────────────────┬──►◄
            ├─ DIAlog ────┤               └─ VERsion version-number ─┘
            ├─ MAP ───────┤
            ├─ SS ────────┤
            └─ TABle ─────┘
```

**INQUIRE format**

```
►►── DEBug ─┬─ entity-name ─┬──────────────────────────┬─┬─┬─ INQuire ─┬──►◄
            │               └─ VERsion version-number ─┘ │ └─ OFF ─────┘
            └─ ALL ──────────────────────────────────────┘
```

## Syntax: EXIT

```
►►── EXIt ──────────────────────────────────────────────────────────────►◄
```

## Syntax: IOUSER

```
►►── IOUser ────────────────────────────────────────────────────────────►◄
```

## Syntax: LIST

**MEMORY Format**

```
►►─┬─ List ────┬─┬──────────┬─ begin-debug-expression ──────────────────►
   └─ Display ─┘ └─ Memory ─┘

 ►─┬───────────────────────────────────────────┬─┬──────┬────────────────►◄
   ├─ TO end-debug-expression ─────────────────┤ ├─ C ──┤
   └─┬────────────┬─ byte-count-number ────────┘ ├─ X ──┤
     └─ LENgth ───┘                              └─ XC ─┘
```

**ATTRIBUTES Format**

```
►►─┬─ List ────┬─ SESsion ATTributes ───────────────────────────────────►◄
   └─ Display ─┘
```

## Syntax: MENU

```
►►── MENu ─┬─────────────────┬──────────────────────────────────────────►◄
           └─ screen-name ───┘
```

## Syntax: PROMPT

```
►►── PROmpt ────────────────────────────────────────────────────────────►◄
```

## Syntax: QUALIFY

**RESET Format**

```
►►── QUAlify ─┬──────────────────────┬─ PROCess process-name ──────────►
              └─ DIAlog dialog-name ─┘

 ►─┬─────────────────────────┬────────────────────────────────────────►◄
   └─ VERsion version-number ─┘
```

**INQUIRE Format**

```
►►── QUAlify INQuire ─────────────────────────────────────────────────►◄
```

## Syntax: QUIT

```
►►── QUIt ────────────────────────────────────────────────────────────►◄
```

## Syntax: RESUME

```
►►── RESume ─┬───────────────────────────────────┬────────────────────►◄
             └─┬──────┬─┬─ debug-expression ─┬─┘
               └─ AT ─┘ └─ ABEnd ────────────┘
```

## Syntax: SET

**MEMORY Format**

```
►►─┬─ Set ──┬─┬──────────┬─ debug-expression ─┬────────────┬──────────►
   └─ Vary ─┘ └─ Memory ─┘                    ├─ EQUals ─┤
                                              └─ = ──────┘

 ►─┬─ data-field-name ────┬─┬──────┬─┬──────────────┬─────────────────►◄
   ├─ H halfword ─────────┤ ├─ C ──┤ ├─ RESEt ──────┤
   ├─ F fullword ─────────┤ ├─ X ──┤ └─ NOReset ◄ ──┘
   ├─ X hex-value ────────┤ └─ XC ─┘
   ├─ C character-string ─┤
   └─ P packed-value ─────┘
```

**ATTRIBUTES Format**

```
►►── Set ─┬─ CHAr ─┬──────────────────────────────────────────────────►◄
          ├─ HEX ──┤
          └─ BOTh ─┘
```

## Syntax: SNAP

```
►►── SNAp ─┬─ TASk ─────────────────────────────────────────────────┬─►
           └─ begin-debug-expression ─┬──────────────────────────┬─┘
                                      ├─ TO end-debug-expression ─┤
                                      └─┬──────────┬─ byte-count-number ─┘
                                        └─ LENgth ─┘

 ►─┬───────────────┬──────────────────────────────────────────────────►◄
   └─ TITle title ─┘
```

## Syntax: WHERE

```
►►── WHEre ──────────────────────────────────────────────────────────►◄
```

# Index