CA IDMS™/DC Sort

User Guide

Release 18.5.00

CA Technologies Product References

This guide references to the following CA products:

- CA ADS™ For CA IDMS™
- CA ADS™ Option for APPC
- CA ADS™ Batch Option
- CA ADS™ Alive Option
- CA ADS™ Trace Option
- CA IDMS™ Database Dictionary Module Editor Option
- CA IDMS™ Database Dictionary Migrator Option

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.

Contents

Chapter 1: Introduction 9

Chapter 2: General Information 11

Overview 11
Flexible Online Sorting with CA IDMS/DC Sort 11
Easy Selection of Criteria 12
Multiple Sorts at One Terminal 12
Multiple Sort Keys in Each Sequence 12
Optional Online Criteria at Runtime 12
Quick and Easy Sorts 12
Parameter Statements Make CA IDMS/DC Sort Easy to Use 13
CA IDMS Dictionary Access 14
Preprocessor Support 14
Flexible Retrieval of Sorted Data 14
Session Kept Active Across Pseudo Converses 14
Processing Environment 15

Chapter 3: Parameters 17

Overview 17
CA IDMS/DC Sort Parameter Statements 17
- Parameter Options 18
- SETSORT Statement 18
- PUTSORT Statement 18
- GETSORT Statement 18
- ENDSORT Statement 18
- SETLIMIT Statement 19

Notation Conventions and Syntax Rules 20
SETSORT Statement 24
- SESSION Parameter 25
- USER/PROGRAM Parameter 26
- IDMS Parameter 26
- record-name Parameter 26
- LENGTH Parameter 27
- VERSION Parameter 27
- DICTIONARY Parameter 27
- NODE Parameter 28

FIELDS Parameter 28
PUTSORT Statement 29
SESSION Parameter 30
GETSORT Statement 30
SESSION Parameter 30
NEXT/PRIOR/FIRST/LAST Parameters 31
record-name Parameter 31
ENDSORT Statement 32
SESSION Parameter 32
SETLIMIT Statement 32

Chapter 4: Examples 35

Overview 35
CA IDMS/DC Sort Examples 35
TPSEXPL1 36
TPSEXPL2 36
TPSEXPL3 36
TPSEXPL4 36
Selecting Sort Criteria on a User Screen 89
Specifying Sequence and Sort Order 90
Sample Sort Selection Screen 90
Field Error 91
Expanding Short Form Field Error Messages 92
Processing Errors 92

Chapter 5: Operations 95

Overview 95
Operational Considerations 95
CA IDMS/DC Sort System Flow 96
System Limits 98
Storage Requirements 98
Online Program Storage 98
COBOL/Assembler/PLI 98
CA ADS 100
Demonstration 104
CA IDMS Environment 104
CICS Environment 104
Customizing CA IDMS/DC Sort 104
Customization Considerations 105
Sample CA IDMS/DC Sort Customization 105

Chapter 6: CA ADS Preprocessor 109

Overview 109
CA ADS Preprocessor 109
- Step 1—Add CA IDMS/DC Sort Statements to Modules 109
- Step 2—Execute the Preprocessor for Each Module 110
- Step 3—Execute the Dialog Generator 111
- Step 4—Execute the Dialog 111

Preprocess Multiple Modules 111
System Flow 112
EDITOR 115
Editing Commands 115
- Scroll Options 115
- Primary Commands 115
- Line Commands 116
- Program Function Keys 116
- Entering Commands 116
- Conventions, Rules and Syntax 117
- Functions 117
- Command Syntax Summary 119

Primary Commands 121
- UP and DOWN Commands 121
- RIGHT and LEFT Commands 122
- LOCATE Command 123
- FIND and RFIND Commands 123
- CHANGE and RCHANGE Commands 124
- RESET Command 127
- CANCEL Command 127
- END/RETRY Command 127
- NULLS Command 128
- CAPS Command 128

Scroll Options 129
Line Commands 130
- Entering Line Commands 131
- B (before) and A (after) Commands 132
- C (copy) Command 132
- M (move) Command 133
- R (repeat) Command 134
- D (delete) Command 135
- I (insert) Command 136
- COLS (columns) Command 136

Key Settings 137

ENTER Function 137
RESHOW Function 138

Chapter 7: Messages 139

Overview 139
Non-Standard Message Codes--Batch 140
Non-Standard Message Codes--Batch or Online 143

Index 165

Chapter 1: Introduction

The CA IDMS/DC Transaction Server Sort User Guide is a reference tool that provides complete information on how to use CA IDMS /DC Transaction Server Sort.

Chapter 2: General Information

This section contains the following topics:

Overview (see page 11)
Flexible Online Sorting with CA IDMS/DC Sort (see page 11)
Easy Selection of Criteria (see page 12)
Multiple Sorts at One Terminal (see page 12)
Multiple Sort Keys in Each Sequence (see page 12)
Optional Online Criteria at Runtime (see page 12)
Quick and Easy Sorts (see page 12)
Parameter Statements Make CA IDMS/DC Sort Easy to Use (see page 13)
CA IDMS Dictionary Access (see page 14)
Preprocessor Support (see page 14)
Flexible Retrieval of Sorted Data (see page 14)
Session Kept Active Across Pseudo Converses (see page 14)
Processing Environment (see page 15)

Overview

CA IDMS/DC Sort provides fast, flexible sorting of data online. Sort criteria are specific to each application program, and can even be selected at runtime. You can run up to 10 concurrent sort sessions at one terminal, with each screen using up to 16 sort keys. With CA IDMS/DC Sort, you no longer need to store data in many different sorted sequences.

Flexible Online Sorting with CA IDMS/DC Sort

CA IDMS/DC Sort is a fast and efficient online sort utility for both the CA IDMS/DC Sort and CICS environments. It provides a versatile means to sort any information online, regardless of the file structure or original sequence.

CA IDMS/DC Sort eliminates the need to design and maintain specific sequences in your files or on your database. In addition, fewer separate sorting and reporting jobs need to be maintained at your site. Therefore, the use of CA IDMS/DC Sort reduces design and maintenance time and improves performance.

Easy Selection of Criteria

For each sort to be performed, you can select criteria easily, either before or during execution of a program. To specify the criteria before execution, you enter the sort criteria within the application program. As an alternative, in the program you can specify that each user define necessary sort criteria whenever a sort is required during execution.

Multiple Sorts at One Terminal

CA IDMS/DC Sort accommodates complex sequence requests. Within an application it can sort as many as 10 different data structures concurrently. It can also sort a single data structure into many different sequences.

Multiple Sort Keys in Each Sequence

When specifying a sort of a particular data structure, you can specify up to 16 elements as sort keys. These elements can be:

- Selected and sequenced when setting up the sort in the program or
- Specified by the user when a program is run.

Optional Online Criteria at Runtime

In the application program, the system designer can set a user option (USER) that allows the user to specify sort criteria at processing time.

When USER is specified in the program, at processing time CA IDMS/DC Sort displays a screen for defining sort criteria. At this time the user can select the sequence of key elements and specify whether each element is to be sorted in ascending or descending order.

Quick and Easy Sorts

CA IDMS/DC Sort makes it possible to respond quickly to a request for a new sequence of data.

- Sort sequences can be specified immediately online.
- There is no need to spend extra time restructuring either a file, a database, or an index.
- Permanent changes to files or to the database are not necessary.

CA IDMS/DC Sort increases flexibility.

- Future sort requirements are taken care of. If the USER function is specified, the user can even specify sequence and sort order at runtime.
- Easy-to-use parameter statements allow ordering of data from one file in many different sequences.

CA IDMS/DC Sort reduces time spent by program designers, because they do not have to decide in advance which is the best sequence to store the data; how many sequences are needed to support processing requirements; which sequences are most important; or whether the sequence will meet future processing requirements. Each program uses whatever sequence is necessary, no matter how the data is stored.

CA IDMS/DC Sort reduces time spent reorganizing and restructuring.

- It is not necessary to reorganize a data structure when a new sort sequence is proposed.

CA IDMS/DC Sort eliminates the need to create and maintain redundant files, databases, or indexes simply to satisfy new sort requests.

- System overhead is not required to maintain a variety of sorted sequences.
- Personnel time is not required to create and maintain programs to keep redundant data synchronized.
- Additional DASD storage space is not required.

Parameter Statements Make CA IDMS/DC Sort Easy to Use

CA IDMS/DC Sort is easily controlled by five parameter statements:

- The SETSORT statement initiates the CA IDMS/DC Sort process. In addition, the SETSORT statement either explicitly states the criteria for a sort to be performed or invokes the user option, so that the user can specify the sort criteria.
- The PUTSORT statement transfers a record to CA IDMS/DC Sort for processing.
- The GETSORT statement retrieves a record after CA IDMS/DC Sort processing, (FIRST, LAST, NEXT or PRIOR).
- The ENDSORT statement terminates the CA IDMS/DC Sort process.
- The SETLIMIT statement overrides internal system storage limits.

By simply entering these five statements in the online program, you can have CA IDMS/DC Sort efficiently perform the sorts in whatever order you specify.

Then all you have to do is enter the code that directs the system to display data or use it in some other way.

CA IDMS Dictionary Access

If your system operates under CA IDMS, CA IDMS/DC Sort allows you to specify a CA IDMS dictionary. Use of the dictionary eliminates the need to specify various parameters in the SETSORT statement, since CA IDMS/DC Sort can extract those values from the dictionary.

If you are specifying the sort criteria at runtime, the elements extracted from the dictionary are automatically displayed by CA IDMS/DC Sort on the Sort Selection Display, where you can easily select the sequence and sort order. See sequence and sort order. See Chapter 4, Examples for more information on the display.

Preprocessor Support

CA IDMS/DC Sort provides a preprocessor for use with COBOL, Assembler, PLI, and CA ADS. The preprocessor uses the CA IDMS/DC Sort parameter statements to generate programming statements that fit your sorting requirements. The preprocessor also identifies errors.

Flexible Retrieval of Sorted Data

CA IDMS/DC Sort offers several alternatives for retrieving sorted records. These alternatives are specified in GETSORT parameter statements. The sorted records can be retrieved:

- From the beginning of the list of the sorted records (FIRST).
- From the end of the list of the sorted records (LAST).
- By moving forward (NEXT) or backward (PRIOR) within the list of sorted records.

Multiple GETSORT statements with different retrieval parameters can be issued. For example, CA IDMS/DC Sort may have sorted a list of salespersons in descending order by sales. By specifying FIRST and NEXT in GETSORT statements, you can obtain the top five from the list. Similarly, by specifying LAST and PRIOR, you can obtain the lowest five from the same ordered list.

Session Kept Active Across Pseudo Converses

CA IDMS/DC Sort automatically keeps track of your location within the sorted file. For example, if the sorted file contains more than one screenful of records, CA IDMS/DC Sort allows you to move from screen to screen without additional programming.

Processing Environment

CA IDMS/DC Sort supports these languages:

- COBOL
- Assembler
- PLI
- CA ADS

CA IDMS/DC Sort sorts information stored in any data structure.

Chapter 3: Parameters

This section contains the following topics:

Overview (see page 17)
CA IDMS/DC Sort Parameter Statements (see page 17)
Notation Conventions and Syntax Rules (see page 20)
SETSORT Statement (see page 24)
PUTSORT Statement (see page 29)
GETSORT Statement (see page 30)
ENDSORT Statement (see page 32)
SETLIMIT Statement (see page 32)

Overview

This chapter is a guide to the CA IDMS/DC Sort parameters. It begins with an overview of the five parameter statements that can be entered in your application program. The overview includes a complete parameter summary chart, notation conventions, and syntax rules. The overview is followed by a description of each parameter, showing its appropriate syntax, rules for use, and defaults. The parameters are presented in the order shown in the parameter summary chart.

CA IDMS/DC Sort Parameter Statements

Five parameter statements are entered in the application programs: SETSORT, PUTSORT, GETSORT, ENDSORT and SETLIMIT. Sample application programs including these statements are shown in Examples (see page 35).

CA IDMS/DC Sort has a work-saving preprocessor which generates programming statements appropriate for your sorting requirements. This chapter explains the parameter statements that you must include in your program.

Parameter Options

Within the parameter statements, CA IDMS/DC Sort provides a variety of options for tailoring your sort session to meet your needs. These options allow you to select a record and define sort criteria.

Your choice of options entered in the parameter statements depends on the programming language, the operating environment, and the type of sort you want generated.

In a CA IDMS environment, you can direct CA IDMS/DC Sort to extract some of the control information from the dictionary.

SETSORT Statement

The SETSORT statement initiates a CA IDMS/DC Sort session. From this statement, values are initialized in the CA IDMS/DC Sort control blocks. These values are in effect until an ENDSORT statement is issued for the session.

In the SETSORT statement, you can specify the record name and sort criteria, or you can indicate that the user is to define criteria at execution time. The SETSORT statement can also indicate that some of the values are to be extracted from a dictionary.

PUTSORT Statement

The PUTSORT statement takes a record defined in the SETSORT statement for the session and transfers it to CA IDMS/DC Sort processing.

GETSORT Statement

The GETSORT statement retrieves a record defined in the SETSORT statement for the session. You can specify one of four different sequences for retrieval: NEXT, PRIOR, FIRST, or LAST.

ENDSORT Statement

The ENDSORT parameter statement is used to terminate a CA IDMS/DC Sort session and to release the resources used by CA IDMS/DC Sort.

SETLIMIT Statement

The SETLIMIT statement is used to override runtime storage limits and functional page organization. The use of SETLIMIT is enabled or disabled through the TPSPARM tuning macro described in Operations (see page 95).

Notation Conventions and Syntax Rules

The rest of this chapter explains in detail how to use the parameter statements. Be sure to review the following Exhibits.

- Exhibit 3.1 — CA IDMS/DC Sort Parameter Summary
- Exhibit 3.2 — CA IDMS/DC Sort Parameter Summary with the IDMS Extension
- Exhibit 3.3 — Notation Conventions
- Exhibit 3.4 — Parameter Syntax Rules

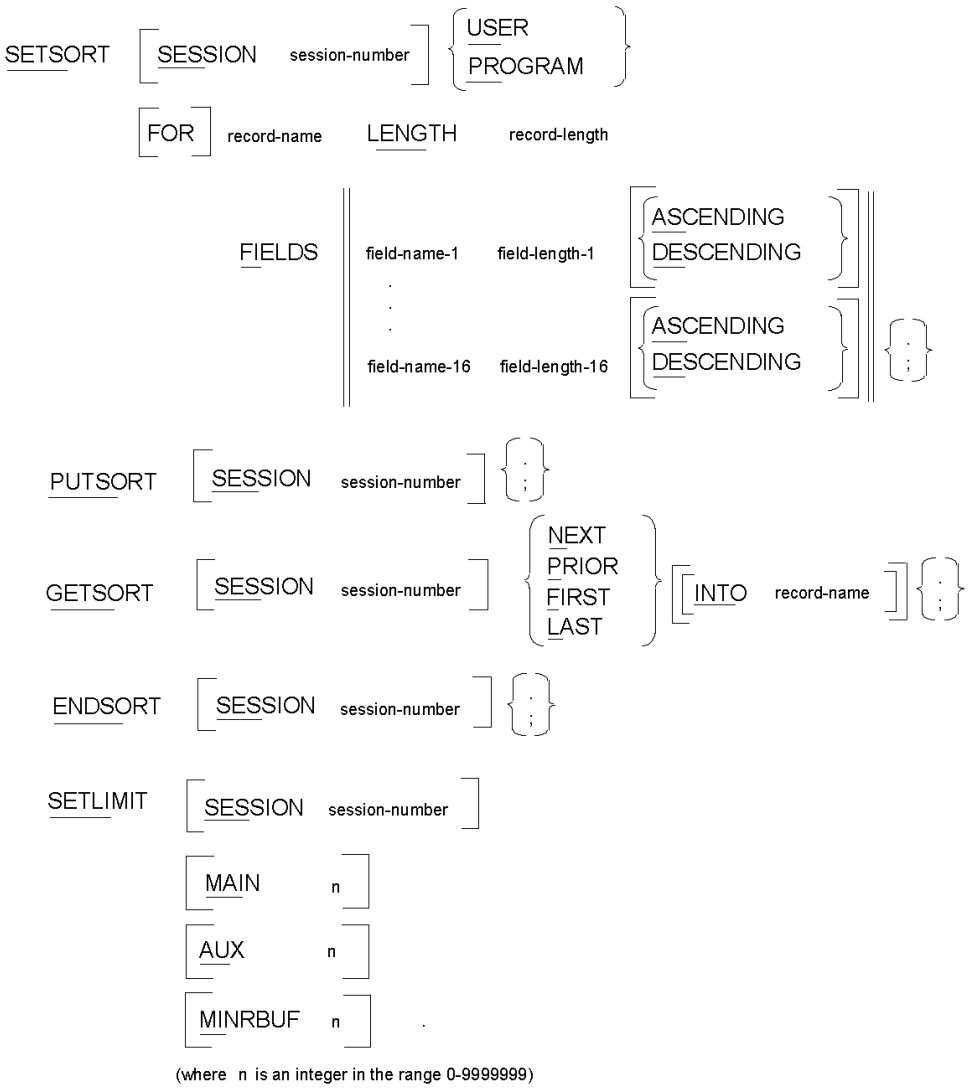

Exhibit 3.1: CA IDMS/DC Sort Parameter Summary

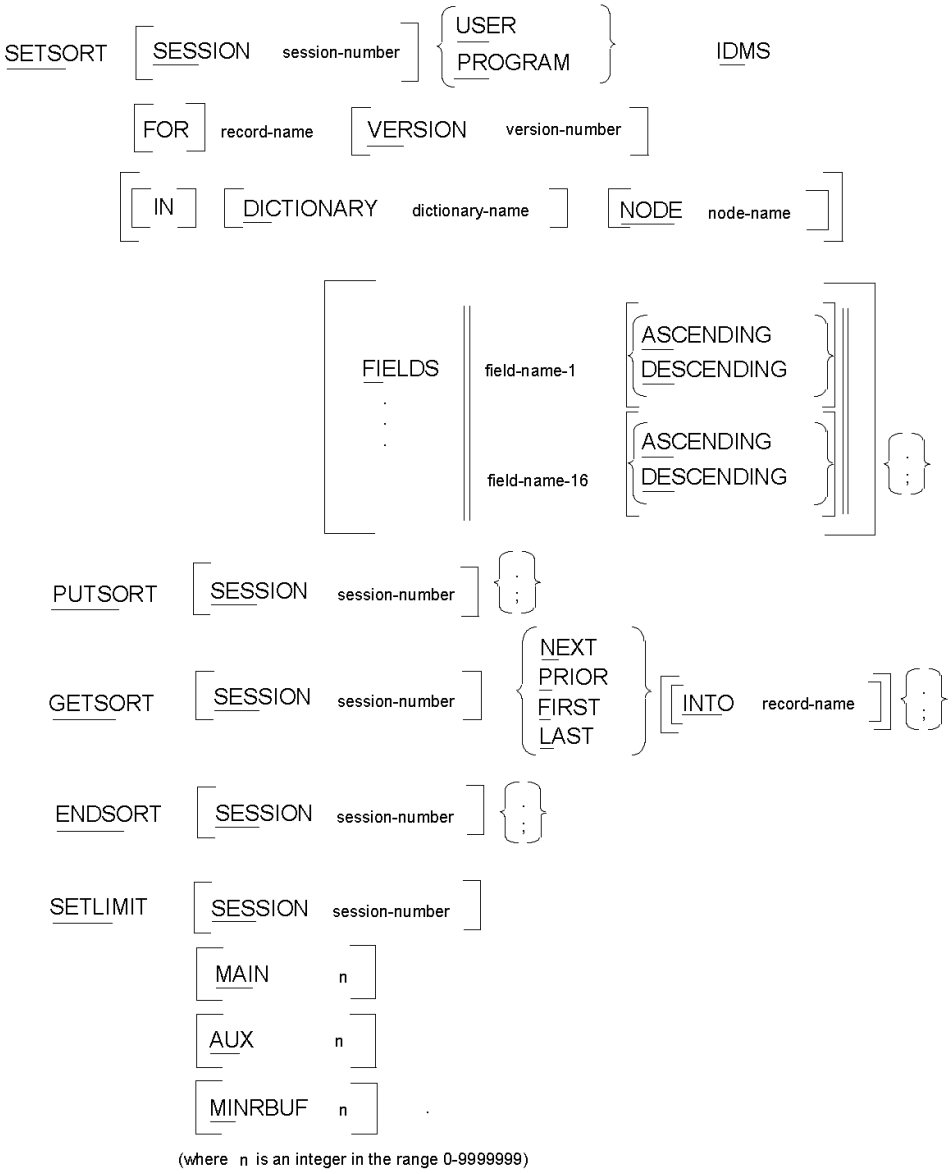

Exhibit 3.2: CA IDMS/DC Sort Parameter Summary with IDMS Extension

Example	Function
SETSORT	Keywords appear in UPPERCASE.
SESSION	The minimum required portion of each keyword is UNDERSCORED. You can omit the portion of a keyword that is not underscored without altering the meaning.
record-name	Variables appear in lowercase. You must substitute an appropriate value for a variable.
[field-length]	Brackets indicate optional clauses.
/PRIOR\ \FIRST/	Braces enclose two or more options. You must select one of them.
|| field-name-1 || || . || || . || || . || || field-name-16 ||	A pair of double bars encloses two or more options. You must select one or more of the options.

Exhibit 3.3: Notation Conventions

Item	Rule
Use of Delimiters	Use one or more blanks as a delimiter between keywords. Use a period or semicolon to end each parameter statement.
Coding Conventions	When inserting the CA IDMS/DC Sort parameters into your application program, follow the coding conventions of the application program language: PLI, Assembler, COBOL or CA ADS.
Parameter Statement Limits	Parameter statements can be continued on more than one line. However, you cannot exceed 50 lines of syntax for a single statement.

Exhibit 3.4: Parameter Syntax Rules

SETSORT Statement

A single SETSORT statement is required for each CA IDMS/DC Sort session. The SETSORT statement must be the first of the four statements which are coded into the application program.

The SETSORT statement identifies the particular session and indicates to CA IDMS/DC Sort the requirements of this session.

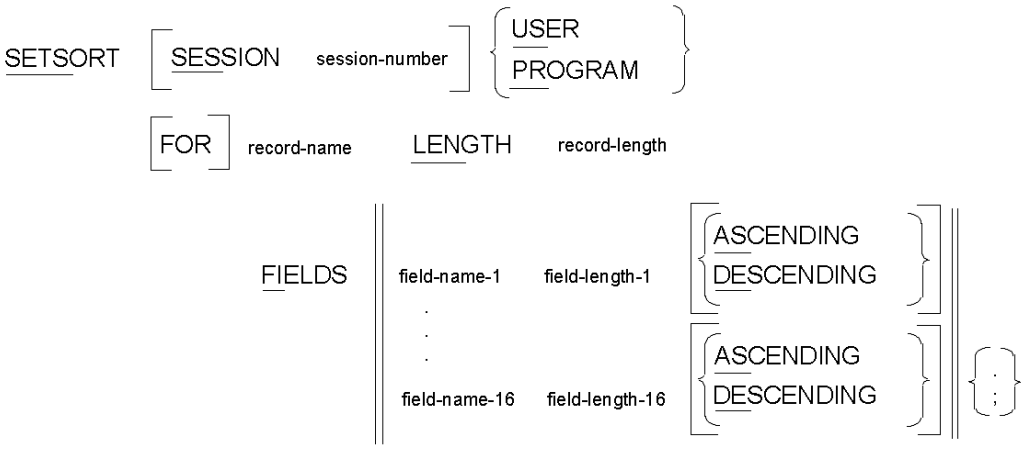

Exhibit 3.5: SETSORT Syntax

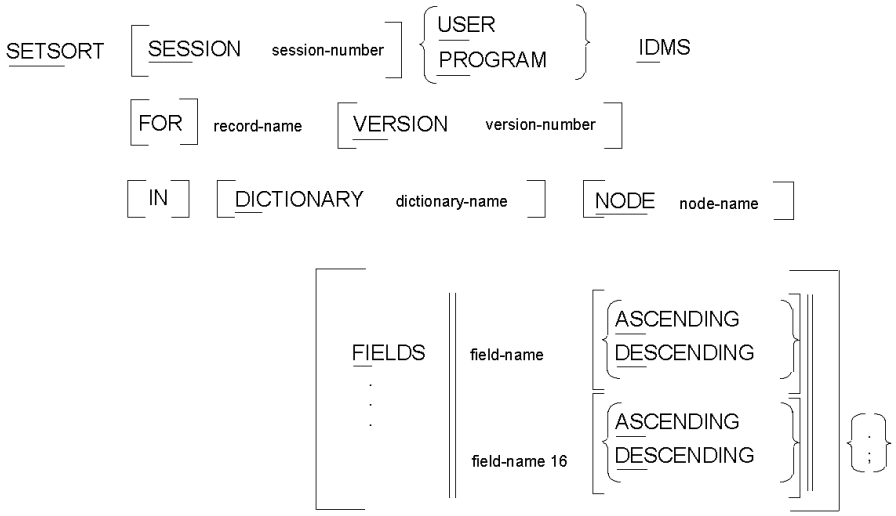

Exhibit 3.6: SETSORT Syntax with IDMS Extension

SESSION Parameter

*SES***SION session-number**

SESSION is an optional parameter that identifies a sorting process for a single terminal user. The session number can be a value from 0 to 9. This number allows you to differentiate concurrent sorting of various lists or of one list using different sort keys.

Default: The default value is 0.

Once the SESSION parameter has been set in the SETSORT statement, all sort criteria remain intact until the session is terminated with an ENDSORT statement. Within a particular task, a second SETSORT statement for the same session number cannot be issued until an ENDSORT statement has been issued. The same session number may then be reused, with a different set of sort criteria.

If a new task is begun and no ENDSORT statement for a given session number was issued in the old task, you can use a SETSORT statement with the same session number in the new task. In that case, CA IDMS/DC Sort automatically issues an ENDSORT for the session in the old task.

USER/PROGRAM Parameter

*/US***ER**\
*PR***OGRAM**/

is a required parameter which indicates to CA IDMS/DC Sort whether sort criteria will be defined dynamically by the user at processing time, or within the program.

Note: Blank spaces are not valid; only options USER and PROGRAM are valid.

*US***ER**

indicates that CA IDMS/DC Sort will prompt the terminal user for sort criteria on a screen at processing time. See Chapter 4, Examples for a description of selection screen.

*PR***OGRAM**

indicates that the sort criteria are defined in the program and the user cannot change them at processing time. When PROGRAM is specified, all sort criteria must be included in the SETSORT statement within the application program.

IDMS Parameter

*ID***MS**

is an optional parameter which indicates to CA IDMS/DC Sort that it must access a CA IDMS dictionary for information about the specified record. If the application program is written in CA ADS, IDMS is assumed automatically.

DICTIONARY and NODE, which are explained on a later page, may be specified to further identify a CA IDMS dictionary.

record-name Parameter

record-name

is a required parameter and specifies the name of the record that CA IDMS/DC Sort will use for put and get requests. There is a limitation in the manner DCSORT navigates the Integrated Data Dictionary to obtain the record element list when the IDMS option is used. This limitation means that only the last NAMESYN-083 within the SET SDR-NAMESYN will be recognized by DCSORT as a valid field name.

The IDD is navigated as follows to determine the element name:

- OBTAIN CALC SR-036 to obtain the correct record name and version.
- Repeat the DML command for all elements.
- OBTAIN NEXT SDR-042 WITHIN SET SR-SDT to obtain the SDR-042 junction record between record and element.
- OBTAIN LAST NAMESYN-083 WITHIN SET SDR-NAMESYN to obtain the element name.

For any SETSORT statement where IDMS is specified, or if your application is written in CA ADS, the record name must be the 01-level name of the record as it resides in the dictionary. For all other applications, it may be any symbolic item.

LENGTH Parameter

*LEN***GTH record-length**

is required if IDMS is not specified. It specifies the length of the record to be sorted. It may be specified as a numeric integer or as a symbolic data name that will satisfy an assignment to a halfword field.

If IDMS is specified, CA IDMS/DC Sort extracts the record length from the dictionary. If IDMS is specified and a length is specified, CA IDMS/DC Sort returns an error message.

VERSION Parameter

*VE***RSION version-number**

is an optional parameter used to further qualify a CA IDMS record if the IDMS keyword was used previously. If you specify VERSION without IDMS, CA IDMS/DC Sort returns an error message.

Version-number must be an unsigned integer from 0 to 9999.

Default: The default value is 1.

DICTIONARY Parameter

*DI***CTIONARY dictionary-name**

is an optional parameter which indicates to CA IDMS/DC Sort an alternate dictionary in which the CA IDMS record resides. If you use the DICTIONARY parameter without the IDMS parameter, CA IDMS/DC Sort returns an error message.

Dictionary-name must be a 1- to 8-character alphanumeric name.

Default: The primary dictionary.

NODE Parameter

***NODE* node-name**

is an optional parameter which represents a DDS node in which a CA IDMS record resides. If you use the NODE parameter without the IDMS parameter, CA IDMS/DC Sort returns an error message.

Node-name must be a 1- to 8- character alphanumeric name.

Defaults: For CA IDMS DME and TPSG interfaces, the default is the dictionary/node combination in which the current MODULE-SOURCE is stored.

For batch preprocessors, the default may be specified in a DDDL format "signon" statement. For CA ADS batch processing, the signon statement is left in place. For other languages, the signon statement is removed.

FIELDS Parameter

The FIELDS parameter is required if you specified any of the following parameters:

- USER without IDMS
- PROGRAM

```
FIELDS|| field-name-1  field-length-1  [sort-order] ||
      ||    .                                      ||
      ||    .                                      ||
      ||    .                                      ||
      || field-name-16 field-length-16 [sort-order] ||
```

Format 1 (USER without IDMS)

field-name field-length

You can define 1 to 16 sets of field-name and field-length.

Format 2 (PROGRAM without IDMS)

field-name field-length sort-order

You can define 1 to 16 sets of field-name, field-length, and sort-order.

Format 3 (PROGRAM with IDMS)

field-name sort-order

You can define 1 to 16 sets of field-name and sort-order.

field-name

indicates the symbolic name of a data item that is subordinate to the record-name specified in the SETSORT statement. This field cannot be subscripted.

field-length

indicates the length, in bytes, of the data item indicated by the field name.

sort-order

is either *AS*CENDING or *DE*SCENDING.

Examples

Format 1 - USER without IDMS

```
FIELDS     SALES-ITEM-NAME        25
           SALES-DATE              8
```

Format 2- PROGRAM without IDMS

```
FIELDS     EMPLOYEE-SICK-LEAVE     3   DE
           EMPLOYEE-NAME          40   AS
```

Format 3 - PROGRAM with IDMS (field-length as in dictionary)

```
FIELDS     EMPLOYEE-VACATION      AS
           EMPLOYEE-NAME          AS
```

PUTSORT Statement

The PUTSORT statement transfers a record to CA IDMS/DC Sort.

```
                                  /.\
PUTSORT [ SESSION session-number] <;>
```

SESSION Parameter

*SES***SION session-number**

is an optional parameter that identifies a sorting process for a single terminal user. The session-number can be a number from 0 to 9.

Default: The default value is 0.

The session-number in the PUTSORT statement must have the same value as the session-number in a corresponding SETSORT statement.

GETSORT Statement

The GETSORT statement lets you define a retrieval location for the sorted record. Once you issue a GETSORT statement, you cannot issue another PUTSORT statement for the same session until that session is ended with an ENDSORT.

```
                                    /NEXT\                          /.\
GETSORT [ SESSIONsession-number ] < PRIOR > [ [INTO] record-name] <;>
                                  | FIRST |
                                   \LAST/
```

SESSION Parameter

*SES***SION** session-number

is an optional parameter which identifies a sorting process for a single terminal user. The session-number can be a number from 0 to 9.

Default: The default value is 0.

The SESSION parameter in the GETSORT statement must have the same value as the SESSION parameter in the corresponding SETSORT statement.

NEXT/PRIOR/FIRST/LAST Parameters

One of these parameters is required and indicates to CA IDMS/DC Sort how assorted record should be retrieved.

***N*EXT**

returns next sequential record in the sort queue.

***P*RIOR**

returns previous sequential record in the queue.

***F*IRST**

returns the first record in the sort queue.

***L*AST**

returns the last record in the sort queue.

record-name Parameter

record-name

is the name of the record into which CA IDMS/DC Sort is to place each sorted record when it is retrieved.

If record-name is not specified, the record named in the SETSORT statement will be used.

The 'INTO<record-name>' clause must be specified whenever the GETSORT command is separated from the SETSORT command by a pseudo- converse.

Note that in CA ADS, a pseudo-converse occurs whenever a DISPLAY command that has no continue option is encountered.

ENDSORT Statement

The ENDSORT statement terminates a SETSORT sequence. Although optional, its use is recommended in order to free up resources.

A sort session can be terminated at any time with an ENDSORT statement. It need not follow a PUTSORT and GETSORT statement.

```
                                    /.\
ENDSORT [ SESSION session-number ] <;>
```

SESSION Parameter

SESSION session-number

is an optional parameter which identifies a sorting process for a single terminal user. The session-number can be an integer from 0 to 9.

Default: default is 0.

The SESSION parameter in the ENDSORT statement must have the same value as the SESSION parameter in a corresponding SETSORT statement.

SETLIMIT Statement

The SETLIMIT statement, **if enabled**, allows a developer to override system installed defaults for the amount of main and/or auxiliary storage per sort session, and for the minimum number of records to be placed in each sort buffer. These values are fully described under "Tuning Considerations" in Operations (see page 95).

The SETLIMIT statement that applies to a particular session must appear *after* the SETSORT statement that identifies the session and *before* any PUTSORT statements for the session.

```
SETLIMIT  [SESSION session-number]
          [MAIN   n]

          [AUX   n]

          [MINRBUF   n]
```

*SES***SION session-number**

is an optional parameter which identifies a sorting process for a single terminal user. The session number can be an integer from 0 to 9.

*MA***INn**

is an optional parameter which allows you to specify the amount of main storage to be made available to CA IDMS/DC Sort where **n** is an integer from 0 to 9999999.

*AU***Xn**

is an optional parameter which allows you to specify the amount of auxiliary storage to be made available to CA IDMS/DC Sort where **n** is an integer from 0 to 9999999.

*MI***NRBUFn**

is an optional parameter which indicates how space is to be allocated to buffers at runtime where **n** is an integer from 0 to 9999999. The allocation of buffers also depends on the record length in a particular sort.

Chapter 4: Examples

This section contains the following topics:

Overview (see page 35)
CA IDMS/DC Sort Examples (see page 35)

Overview

This chapter provides examples of application programs that use CA IDMS/DC Sort. Each example is presented in four formats—COBOL, Assembler, PLI, and CA ADS. The selection screens that appear when the USER parameter is selected in the SETSORT statement are also illustrated.

CA IDMS/DC Sort Examples

CA IDMS/DC Sort can be used in several ways. It can:

- Perform a single sort
- Perform multiple sorts
- Use predefined criteria
- Use criteria set by the user at runtime
- Operate within a single task
- Operate within multiple tasks (pseudo-conversational)

The examples in this chapter illustrate some of the ways that you can use CA IDMS/DC Sort. Each example is presented in the four languages supported by CA IDMS/DC Sort - COBOL Assembler, PLI, and CA ADS. Highlighted in the examples are the statements used to:

- Copy the control block
- Specify CA IDMS/DC Sort parameters: SETSORT, PUTSORT, GETSORT, ENDSORT, and SETLIMIT
- Check the CA IDMS/DC Sort return code (TPSRETN)
- Issue error messages when appropriate (TPSMSG)

TPSEXPL1

The first example uses CA IDMS/DC Sort to sort salespersons by sales volume and then to display the top five and bottom five salespersons.

Exhibit 4.2 COB is the COBOL version, Exhibit 4.3 ASM is the Assembler version, Exhibit 4.4 PLI is the PLI version, and Exhibit 4.5 ADS, the CA ADS version.

TPSEXPL2

The second example uses CA IDMS/DC Sort to sort accumulated employee sick leave in descending order and employee personal leave in descending order.

Exhibit 4.6 COB is the COBOL version, which also demonstrates the use of the IDMS option. Exhibit 4.7 ASM is the Assembler version, Exhibit 4.8 PLI is the PLI version, and Exhibit 4.9 CA ADS is the CA ADS version.

TPSEXPL3

The third example shows the use of CA IDMS/DC Sort in a pseudo-conversational mode. It sorts a given salesperson's sales by sales item and date.

Exhibit 4.10 COB is the COBOL version, Exhibit 4.11 ASM is the Assembler version, Exhibit 4.12 PLI is the PLI version, and Exhibit 4.13 ADS, the CA ADS version.

TPSEXPL4

The fourth example is similar to the third. In this example, the sort criteria are defined by the user at runtime. The user selection screens are illustrated with this example.

See Exhibit 4.1 for a table showing the features used in each example.

Name	Sort What?	By What?	Display What?	Exhibit Number	Language	Using These Features
TPSEXPL1	salesperson	sales volume ascending	top 5 and bottom 5	3.2 COB 3.3 ASM 3.4 PLI 3.5 ADS	COBOL Assembler PLI CA-ADS	NEXT, PRIOR, FIRST, LAST
TPSEXPL2	employees	accumulated sick leave descending	top 10	3.6 COB 3.7 ASM 3.8 PLI 3.9 ADS	COBOL Assembler PLI CA-ADS	IDMS multiple sessions
		personal leave descending	top 10			FIRST, NEXT
TPSEXPL3	sales data for a salesperson	item-name and date sold	20 items at a time	3.10 COB 3.11 ASM 3.12 PLI 3.13 ADS	COBOL Assembler PLI CA-ADS	pseudo converse
TPSEXPL4	sales data for a salesperson	sales item data User selection screens are illustrated with this example.	20 items at a time	3.14 COB 3.15 ASM 3.16 PLI 3.17 ADS	COBOL Assembler PLI CA-ADS	pseudo converse USER

Exhibit 4.1: Table of Sort Examples

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    TPSEXPL1.
 REMARKS.   THIS COBOL EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO
 DISPLAY THE TOP 5 AND BOTTOM 5 SALES PEOPLE IN A COMPANY USING A SINGLE
 SORT WITHOUT READING THE SALES PEOPLE IN THE MIDDLE OF THE SORTED FILE.

 ENVIRONMENT DIVISION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77    SALES-COUNT                          PIC S9(9) COMP.
 77    END-OF-SALES                              PIC X.

 01    SALES-DATA.
         05    SALES-PERSON                      PIC X(25).
         05    SALES-YTD                         PIC S(9)V99  COMP-3.
```

```
COPY SALESREC
.
COPY TPSCOMMC
.
.

PROCEDURE DIVISION.

        PERFORM 0100-SORT-SALES.
        PERFORM 0200-DISPLAY-TOP-5-BOTTOM-5.

        ...return to CA IDMS/DC.

*****************************************************************************
*          SORT SALES PEOPLE IN ASCENDING ORDER BY YEAR TO DATE             *
*          SALES. NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE       *
*          SALES DATA NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A        *
*          WORK RECORD FOR SORTING EFFICIENCY.                              *
*****************************************************************************

0100-SORT-SALES    SECTION.

           SETSORT PROGRAM
                  FOR SALES-DATA LENGTH 31
                  FIELD SALES-YTD 6
                  ASCENDING.
           IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

           MOVE 'N' END-OF SALES.
           PERFORM 0150-PUT-SORT UNTIL END-OF-SALES = 'Y'.

SECTION-EXIT.
           EXIT.

0150-PUT-SORT    SECTION.
           .
           .
           ...read a sales record, set END-OF-SALES to 'Y' at end.
.
IF END-OF-SALES = 'N'
THEN
           MOVE SALESREC-SALES-PERSON TO SALES-PERSON
           MOVE SALESREC-SALES-YTD    TO SALES-YTD
           PUTSORT;
           IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
```

```
SECTION-EXIT.
           EXIT.

*****************************************************************************
*          GET THE TOP 5 AND BOTTOM 5 SALES PEOPLE AND DISPLAY              *
*          THEIR NAME AND YEAR TO DATE SALES.                               *
*****************************************************************************

0200-DISPLAY-TOP-5-BOTTOM-5    SECTION.

           GETSORT LAST.
           IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
           PERFORM 0220-GET-TOP-SALES VARYING SALES-COUNT
                  FROM 1 BY 1 UNTIL SALES-COUNT > 5.

           GETSORT FIRST.
           IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
           PERFORM 0240-GET-BOTTOM-SALES VARYING SALES-COUNT
                  FROM 1 BY 1 UNTIL SALES-COUNT > 5.

           ENDSORT.
           IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

           .
           .
           ...display map
           .
           .

SECTION-EXIT.
           EXIT.

0220-GET-TOP-SALES    SECTION.

           MOVE SALES-PERSON TO ...map.
           MOVE SALES-YTD    TO ...map.

           GETSORT PRIOR.
           IF TPSRETN - '0000'
           THEN
                  NEXT SENTENCE
           ELSE
                  IF TPSRETN = '7020'
                  THEN
                      MOVE 5 TO SALES-COUNT
```

```
                ELSE
                    PERFORM 9999-SORT-ERROR.
SECTION-EXIT.
          EXIT.

0220-GET-BOTTOM-SALES    SECTION.

          MOVE SALES-PERSON TO ...map.
          MOVE SALES-YTD    TO ...map>

          GETSORT NEXT.
          IF TPSRETN = '0000'
          THEN
                 NEXT SENTENCE
          ELSE
                 IF TPSRETN = '7020'
                 THEN
                    MOVE 5 TO SALES-COUNT
                 ELSE
                    PERFORM 9999-SORT-ERROR

SECTION-EXIT.
          EXIT.

*****************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT      *
*          IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE  *
*          FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.              *
*****************************************************************************

9999-SORT-ERROR    SECTION.

          MOVE TPSMSG TO ...message line in map
          .
          ...display map
          .
          ...return to CA IDMS/DC

SECTION-EXIT.
          EXIT.
```

Exhibit 4.2: COB FIRST, NEXT, LAST, PRIOR--COBOL

```
         TITLE 'TPSEXPL1 -- 5 TOP AND BOTTOM SALES PERSONS'
*************************************************************************
*  THIS ASSEMBLER EXAMPLE ILLUSTRATES THE USE OF TP/SORT TO DISPLAY   *
*  THE TOP 5 AND BOTTOM 5 SALES PERSONS IN A COMPANY USING A SINGLE   *
*  SORT WITHOUT READING THE SALES PERSONS IN THE MIDDLE OF THE        *
*  SORTED FILE.                                                       *
*************************************************************************
name...  DSECT
SLDATA   DS    0XL31             EXTRACTED DATA FROM SALES RECORD
SLPERSON DS    CL25              NAME OF SALES PERSON
SLYTD    DS    PL6               YTD SALES FOR SALES PERSON
         .
END      DS    C                 END OF SALES INDICATOR
         .
         COPY  SALESREC          SALES RECORD
         .
         COPY  TPSCOMMA          TP/SORT COMMUNICATIONS BLOCK
         .
R3       EQU   3                 BAL - SUBROUTINE LINKAGE
R4       EQU   4                 BCT - LOOP COUNTER
         .
TPSEXPL1 CSECT
         .
         .
         BAL   R3,SORTSALE       EXTRACT AND SORT SALES DATA
         BAL   R3,DISPLAY        DISPLAY 5 TOP AND BOTTOM SALES PERSON
         .
         return to CICS or IDMS-DC
         SPACE 2
*************************************************************************
*  SORT SALES PEOPLE IN ASCENDING ORDER BY YEAR TO DATE SALES.        *
*  NOTE:  SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA        *
*  NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK RECORD FOR     *
*  SORTING EFFICIENCY.                                                *
*************************************************************************
SORTSALE EQU   *
         SETSORT PROGRAM
               FOR SLDATA LENGTH 31
               FIELD SLYTD 6 ASCENDING.
         CLC   TPSRETN,=CL4'0000'      SUCCESSFUL SETSORT ?
         BNE   BADSORT                 NO, REPORT ERROR AND ABORT
         MVI   END,C'N'                INITIALIZE FOR LOOP
```

```
         SPACE
PUTLOOP  EQU   *
         .
         read a sales record, set END to 'Y' at end
         .
         CLI   END,'Y'                ANY MORE SALES PERSONS ?
         BER   R3                     NO, RETURN
         MVC   SLPERSON,...           SAVE SALES PERSON FOR SORT/DISPL
         ZAP   SLYTD,...              SAVE YTD SALES FOR SORT/DISPLAY
         PUTSORT.
         CLC   TPSRETN,=CL4'0000'     SALES DATA ACCEPTED BY SORT ?
         BE    PUTLOOP                YES, CONTINUE EXTRACTION
         B     BADSORT                NO, REPORT ERROR AND ABORT
         SPACE 2
*************************************************************************
*  GET TOP 5 AND BOTTOM 5 SALES PERSONS AND DISPLAY THEIR NAME        *
*  AND YEAR TO DATE SALES.                                            *
*************************************************************************
DISPLAY  EQU   *
*-------------------- GET TOP 5 SALES PERSONS ----------------------*
         GETSORT LAST.
         LA    R4,5                   NBR PERSONS TO GET FROM TP/SORT
         B     MAPTOP
TOPLOOP  EQU   *
         GETSORT PRIOR.
MAPTOP   EQU   *
         CLC   TPSRETN,=CL4'7020'     END OF SORTED DATA ?
         BE    BOTTOM5                YES, GO GET BOTTOM 5
         CLC   TPSRETN,=CL4'0000'     SORTED SALES DATA RETRIEVED ?
         BNE   BADSORT                NO, REPORT ERROR AND ABORT
         MVC   ..MAP..,SLPERSON       PUT SALES PERSON NAME IN MAP
         UNPK  ..MAP..,SLYTD          PUT YTD SALES IN MAP
         BCT   R4,TOPLOOP
         SPACE
*-------------------- GET BOTTOM 5 SALES PERSONS -------------------*
BOTTOM5  EQU   *
         GETSORT FIRST.
         LA    R4,5                   NBR PERSONS TO GET FROM TP/SORT
         B     MAPBOT
BOTLOOP  EQU   *
         GETSORT NEXT.
MAPBOT   EQU   *
         CLC   TPSRETN,=CL4'7020'     END OF SORTED DATA ?
         BE    TERMSORT               YES, GO END THE SORT SESSION
         CLC   TPSRETN,=CL4'0000'     SORTED SALES DATA RETRIEVED ?
```

```
         BNE   BADSORT                NO, REPORT ERROR AND ABORT
         MVC   ..MAP..,SLPERSON       PUT SALES PERSON NAME IN MAP
         MVC   ..MAP..,SLYTD          PUT YTD SALES IN MAP
         BCT   R4,BOTLOOP
         SPACE
*------------------- END SORT SESSION -----------------------------*
TERMSORT EQU   *
         ENDSORT.
         CLC   TPSRETN,=CL4'0000'     SORT SESSION ENDED OK ?
         BNE   BADSORT                NO, REPORT ERROR AND ABORT
         SPACE
*------------------- DISPLAY MAP ----------------------------------*
DSPLYMAP EQU   *
         .
         display map
         .
         BR    R3
         SPACE 2
*************************************************************************
*  AN UNANTICIPATED RETURN CODE WAS RETURNED BY TP/SORT IN THE        *
*  TPSRETN FIELD.  TPSMSG FIELD CONTAINS A 79 CHARACTER MESSAGE       *
*  FROM TP/SORT DESCRIBING THE BAD RETURN CODE.                       *
*************************************************************************
BADSORT  EQU   *
         MVC   ..MAP..,TPSMSG         USE MESSAGE FROM TP/SORT
         .
         display map
         .
         return to CICS or IDMS-DC
```

Exhibit 4.3: ASM FIRST, NEXT, LAST--Assembler

```
 TPSEXPL1: PROC OPTIONS(MAIN) REORDER;
 /*      REMARKS.  THIS PLI EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO
         DISPLAY THE TOP 5 AND BOTTOM 5 SALES PEOPLE IN A COMPANY USING
         A SINGLE SORT WITHOUT READING THE SALES PEOPLE IN THE MIDDLE
         OF THE SORTED FILE.

   CA IDMS/DC SORT REQUIRES COMPILE OPTION "MARGINS(2,72)".
 */

 /*REQUIRED FOR IDMS*/
 DCL MODE (IDMS_DC) DEBUG;
 DCL IDMS ENTRY OPTIONS( INTER,ASM);
 INCLUDE IDMS(SUBSCHEMA_CTRL);
 /*END OF IDMS REQUIREMENT*/
```

```
DCL ADDR BUILTIN;

DCL SALES_COUNT                                          FIXED   BIN(31);
DCL END_OF_SALES                                                 CHAR(1);

DCL  1  SALES_DATA,
     2  SALES_PERSON                                             CHAR(25),
     2  SALES_YTD                                                PIC'S99999V99';

%INCLUDE SALESREC;
.
%INCLUDE TPSCOMMP;
.
CALL  SORT_SALES_0100;
CALL DISPLAY_TOP_5_BOTTOM_5_0200;

...return to CA IDMS/DC;

/*****************************************************************************
*          SORT SALES PEOPLE IN ASCENDING ORDER BY YEAR TO DATE SALES.       *
*          NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA        *
*          NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK RECORD        *
*          FOR SORTING EFFICIENCY.                                       *
*****************************************************************************

SORT_SALES_0100: PROC;

SETSORT PROGRAM
        FOR SALES_DATALENGTH 31
        FIELD SALES_YTD 7
        ASCENDING;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;

END_OF_SALES = 'N';

DO UNTIL (END_OF_SALES = 'Y');
        CALL PUT_SORT_0150;
END;

END SORT_SALES_0100;

PUT_SORT_0150: PROC;
.
.
...read a sales record, set END_OF_SALE to 'Y' at end.
```

```
.
.
IF (END_OF_SALES = 'N')
THEN DO;
        SALES_PERSON = SALESREC_SALES_PERSON;
        SALES_YTD    = SALESREC_SALES _YTD;
        PUTSORT;
        IF (TPSRETN  = '0000') THEN CALL SORT_ERROR_9999;
END;

END PUT_SORT_0150;

/*****************************************************************************
*          GET THE TOP 5 AND BOTTOM 5 SALES PEOPLE AND DISPLAY THEIR         *
*          NAME AND YEAR TO DATE SALES.                                      *
*****************************************************************************

DISPLAY_TOP_5_BOTTOM_5_0200: PROC;

GETSORT LAST;]
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
DO SALES_COUNT = 1 TO 5 BY 1;
        CALL GET_TOP_SALES_0220;
END;

GETSORT FIRST;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
DO SALES_COUNT - 1 TO 5 BY 1;
        CALL GET_BOTTOM_SALES_0240;
END;

ENDSORT;
IF(TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
```

```
.
.
...display map
.
.
END DISPLAY_TOP_5_BOTTOM_5_0200;

GET_TOP_SALES_0220: PROC;

...map = SALES_PERSON;
...map = SALES_YTD;

GETSORT PRIOR;
IF (TPSRETN = '0000')
THEN
        ;
ELSE
        IF (TPSRETN = '7020')
        THEN
                SALES_COUNT = 5;     /*  TERMINATE DO_LOOP */
        ELSE
                CALL SORT_ERROR_9999;

END GET_TOP_SALES_0220;

GET_BOTTOM_SALES_0240: PROC;

...map = SALES_PERSON;
...map = SALES_YTD;

GETSORT NEXT;
IF (TPSRETN = '0000')
THEN
        ;
ELSE
        IF (TPSRETN = '7020')
        THEN
                SALES_COUNT = 5;     /*TERMINATE DO_LOOP*/
        ELSE
                CALL SORT_ERROR_9999;

END GET_BOTTOM_SALES_0240;

/*****************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT      *
*          IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE  *
*          FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.              *
*****************************************************************************
```

```
SORT_ERROR_9999: PROC;

...message line in map = TPSMSG;
.
...display map
.
...return to CA IDMS/DC

END SORT_ERROR_9999;

END TPSEXPL1;
```

Exhibit 4.4: PLI FIRST, NEXT, LAST, PRIOR--PLI

```
!       ************************************************************************
!       *     THIS ADS EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT          *
!       *     TO DISPLAY THE TOP 5 AND BOTTOM 5 SALES PEOPLE IN A COMPANY      *
!       *     USING A SINGLE SORT WITHOUT READING THE SALES PEOPLE IN          *
!       *     THE MIDDLE OF THE SORTED FILE.                                   *
!       ************************************************************************

CALL SORT-SLS.
CALL DISPLAY-TOP-5-BOTTOM-5.

DISPLAY.
!       RETURN TO TOP.

!       ************************************************************************
!       *     SORT SALES PEOPLE IN ASCENDING ORDER BY YEAR TO DATE SALES.      *
!       *     NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA       *
!       *     NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO WORK RECORD         *
!       *     FOR SORTING EFFICIENCY.                                          *
!       ************************************************************************

DEFINE SUBROUTINE SORT-SLS.
SETSORT PROGRAM FOR SALES-DATA
        FIELD SALES-YTD 6 ASCENDING
IF TPSRETN NE ZERO
        CALL ERROR
MOVE 'N' TO END-OF-SALES.
WHILE END-OF-SALES NE 'Y'
REPEAT.
        CALL PUT-SORT.
```

```
END
GOBACK.

DEFINE SUBROUTINE PUT-SORT.
.
.
....obtain a sales record, set END-OF-SALES to 'Y' at end.
.
.
IF END-OF SALES = 'N'
DO.
        MOVE SALESREC-SALES-PERSON     TO SALES-PERSON.
        MOVE SALESREC-SALES-YTD        TO SALES-YTD.

        PUTSORT.
        IF TPSRETN NE ZERO
                DO.
                   CALL ERROR.

!       ************************************************************************
!       *DISPLAY-TOP-5-BOTTOM-5                                                *
!       *     GET THE TOP 5 AND BOTTOM 5 SALES PEOPLE AND DISPLAY              *
!       *     THEIR NAMES AND YEAR TO DATE SALES.                              *
!       ************************************************************************

DEFINE SUBROUTINE DTOPBOT.

GETSORT LAST.
IF TPSRETN NE ZERO
        CALL ERROR.

MOVE 1 TO SALES-COUNT.
WHILE SALES-COUNT < 6
REPEAT.
        CALL TOP-SLS.
        ADD 1 TO SALES-COUNT.
```

```
END.
GETSORT FIRST.
IF TPSRETN NE ZERO
        CALL ERROR.

MOVE 1 TO SALES-COUNT.
WHILE SALES-COUNT < 6
REPEAT.
        CALL BOTSLS.
        ADD 1 TO SALES-COUNT.
END.

ENDSORT.
IF TPSRETN NE ZERO
        CALL ERROR.

GOBACK.


DEFINE SUBROUTINE TOPSLS.

MOVE SALES-PERSON   TO   ...map.
MOVE SALES-YTD      TO   ...map.

GETSORT PRIOR.
IF TPSRETN = '7020'
        MOVE 5 TO SALES-COUNT

ELSE
        IF TPSRETN NE ZERO
                CALL ERROR.
GOBACK.

DEFINE SUBROUTINE BOTSLS.

MOVE SALES-PERSON    TO   ...map.
MOVE SALES-YTD       TO   ...map.

GETSORT NEXT.
IF TPSRETN = '7020'
        MOVE 5 TO SALES-COUNT
ELSE
        IF TPSRETN NE ZERO
                CALL ERROR.
GOBACK.
```

```
!       ************************************************************************
!       *SORT-ERROR                                                            *
!       *                                                                      *
!       *     AN UNANTICIPATED RETURN CODE WAS RETURNED BY                     *
!       *     CA IDMS/DC SORT IN THE TPSRETN FIELD.     TPSMSG CONTAINS        *
!       *     A 79 CHARACTER MESSAGE FROM CA IDMS/DC SORT DESCRIBING           *
!       *     THE BAD RETURN CODE.                                          *
!       ************************************************************************
 DEFINE SUBROUTINE ERROR.

 DISPLAY MESSAGE TEXT TPSMSG.
 !       RETURN TO TOP.
 GOBACK.
```

Exhibit 4.5: ADS FIRST, NEXT, LAST, PRIOR--ADS

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    TPSEXPL2.
 REMARKS.   THIS COBOL EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT
 TO EXECUTE TWO SORTS CONCURRENTLY. ONE ON ACCUMULATED SICK LEAVE
 IN DESCENDING ORDER AND ONE ON PERSONAL TIME USED IN DESCENDING
 ORDER. THE TOP TEN IN EACH CATEGORY ARE DISPLAYED.

 ENVIRONMENT DIVISION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77   END-OF-EMPLOYEES                          PIC X.
 77   EMPLOYEE-COUNT                            PIC S9(2) COMP-3.
 77   MAX-EMPLY-IN-MAP                          PIC S9(2) COMP-3 VALUE TO.
```

```
COPY IDMS EMPLYREC VER 22.
.
COPY TPSCOMMC
.
PROCEDURE DIVISION.

PERFORM 0100-SORT-EMPLOYEES.
PERFORM 0200-DISPLAY-TOP-TEN.

...return to CA IDMS/DC.

*****************************************************************************
*          EXECUTE TWO SORTS, BOTH IN DESCENDING ORDER--ONE ON ACCUMULATED     *
*          SICK LEAVE AND THE OTHER ON ACCUMULATED PERSONAL TIME.     NOTE: THE*
*          RECORD BEING SORTED IS AN IDMS RECORD.     THE ELEMENT ATTRIBUTES DO*
*          NOT HAVE TO BE CODED IN THE SETSORT STATEMENT, THEY WILL BE         *
*          EXTRACTED BY CA IDMS/DC SORT FROM THE DICTIONARY.                   *
*****************************************************************************

0100-SORT-EMPLOYEES    SECTION.

SETSORT SESSION 1 PROGRAM IDMS
        FOR EMPLYREC VER 22
        IN DICT TEST
        FIELD EMPLY-SICK-DAYS DESCENDING
                  EMPLY-NAME      ASCENDING.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

SETSORT SESSION 2 PROGRAM IDMS
        FOR EMPLYREC VER 22
        IN DICT TEST
        FIELD EMPLY-PERSONAL-DAYS DESCENDING
                  EMPLY-NAME          ASCENDING.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

MOVE 'N' END-OF-EMPLOYEES.
PERFORM 0150-PUT-SORT UNTIL END-OF-EMPLOYEES = 'Y'.

SECTION-EXIT.
EXIT.

0150-PUT-SORT    SECTION.
.
.
...read an employee record, set END-OF-EMPLOYEES to 'Y' at end.
.
.
```

```
IF END-OF-EMPLOYEES = 'N'
THEN
        PUTSORT SESSION 1.
        IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
        PUTSORT SESSION 2.
        IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

SECTION-EXIT.
EXIT.

*****************************************************************************
*          GET THE TOP TEN FROM THE SICK LEAVE AND PERSONAL TIME SORTS.     *
*          DISPLAY THE EMPLOYEE NAME AND TIME TAKEN.                        *
*****************************************************************************

0200-DISPLAY-TOP-TEN    SECTION.

PERFORM 0220-GET -EMPLOYEE VARYING EMPLOYEE-COUNT
        FROM 1 BY 1 UNTIL EMPLOYEE-COUNT > MAX-EMPLY-IN-MAP.

ENDSORT SESSION 1.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
ENDSORT SESSION 2.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

.
.
...display map
.
.

SECTION-EXIT.
EXIT.

0200-GET-EMPLOYEE    SECTION.

GETSORT SESSION 1 NEXT.
IF TPSRETN - '7020'
THEN
        MOVE MAX-EMPLY-IN-MAY TO EMPLOYEE-COUNT
ELSE
        IF TPSRETN NOT = '0000'
        THEN
                PERFORM 9999-SORT-ERROR
        ELSE
                MOVE EMPLY-NAME           TO ...map
                MOVE EMPLY-PERSONAL-DAYS  TO ...map.
```

```
SECTION-EXIT.
EXIT.

*************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT IN   *
*          THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE FROM*
*          CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.                   *
*************************************************************************

9999-SORT-ERROR    SECTION.

MOVE TPSMSG TO ...message line in map
.
...display map
.
...return to CA IDMS/DC

SECTION-EXIT.
EXIT.
```

Exhibit 4.6: COB Multiple Sessions--COBOL

```
          TITLE 'TPSEXPL2 —— 2 CONCURRENT SORTS'
*************************************************************************
*          THIS ASSEMBLER EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT     *
*          TO EXECUTE TWO SORTS CONCURRENTLY.     ONE ON ACCUMULATED SICK    *
*          LEAVE IN DESCENDING ORDER AND ONE ON PERSONAL TIME USED IN        *
*          DESCENDING ORDER. THE TOP TEN IN EACH CATEGORY ARE DISPLAYED. *
*************************************************************************
name...         DSECT
@COPY IDMS,RECORD=EMPLYREC,VERSION=22
.
END             DS        C                          END OF SALES INDICATOR
.
.
                COPY TPSCOMMA            CA IDMS/DC SORT COMMUNICATIONS BLOCK
```

```
 .
 R3             EQU       3                          BAL - SUBROUTINE LINKAGE
 R4             EQU       4                          BCT - LOOP COUNTER
 .
 TPSEXPL1       CSECT
                .
                .
                BAL           R3,SORTEMPL       EXTRACT AND SORT SICK & PERSONAL
TIME
                BAL           R3,DISPLAY        DISPLAY TOP TEN IN EACH CATEGORY
                .
                return to CA IDMS/DC
                SPACE          2
 ********************************************************************************
 *          EXECUTE TWO SORTS, BOTH IN DESCENDING ORDER--ONE ON ACCUMULATED     *
 *          SICK LEAVE AND THE OTHER ON ACCUMULATED PERSONAL TIME.     NOTE: THE*
 *          RECORD BEING SORTED IS AN IDMS RECORD.     THE ELEMENT ATTRIBUTES DO*
 *          NOT HAVE TO BE CODED IN THE SETSORT STATEMENT, THEY WILL BE         *
 *          EXTRACTED BY CA IDMS/DC SORT FROM THE DICTIONARY.                   *
 ********************************************************************************
 SORTEMPL       EQU       *
                SETSORT SESSION 1 PROGRAM IDMS
                          FOR EMPLYREC VER 22
                          IN DICT TEST
                          FIELD EMPSICK   DESCENDING
                                EMPNAME   ASCENDING.
                CLC             TPSRETN,=CL4'0000'   SUCCESSFUL SETSORT ?
                BNE             BADSORT              NO, REPORT ERROR AND ABORT
                SETSORT SESSION 2 PROGRAM IDMS
                          FOR EMPLYREC VER 22
                          IN DICT TEST
                          FIELD EMPPRSNL  DESCENDING
                                 EMPNAME   ASCENDING.
                CLC       TPSRETN,=CL4'0000'        SUCCESSFUL SETSORT ?
                BNE             BADSORT             NO, REPORT ERROR AND ABORT
                MVI             END,C'N'            INITIALIZE FOR LOOP
                SPACE

PUTLOOP         EQU                         *
                .
                read an employee record, set END to 'Y' at end
                .
                CLI             END,'Y'             ANY MORE EMPLOYEES ?
                BER             R3       NO, RETURN
                PUTSORT SESSION 1.
                CLC            TPSRETN,=CL4'0000'  SICK LEAVE ACCEPTED BY SORT ?
```

```
                BNE             BADSORT             NO, REPORT ERROR AND ABORT
                PUTSORT SESSION 2.
                CLC           TPSRETN,=CL4'0000'  PERSONAL TIME ACCEPTED BY SORT?
                BE              PUTLOOP             YES, CONTINUE EXTRACTION
                B               BADSORT             NO, REPORT ERROR AND ABORT
                SPACE           2
*********************************************************************************
*          GET THE TOP TEN FROM THE SICK LEAVE AND PERSONAL TIME SORTS.      *
*          DISPLAY THE EMPLOYEE NAME AND TIME TAKEN.                         *
*********************************************************************************
DISPLAY         EQU     *
*--------------------GET TOP TEN EMPLOYEES IN EACH CATEGORY-----------------*
                LA              R4,10    NBR EMPLOYEES FOR DISPLAY
                SPACE
LOOP            EQU     *
                GETSORT SESSION 1 NEXT.
                CLC             TPSRETN,=CL4'7020'    END OF SORTED DATA ?
                BE              TERMSORT            YES, GO END THE SORT SESSION
                CLC             TPSRETN,=CL4'0000'    SORTED SALES DATA RETRIEVED
?
                BNE             BADSORT               NO, REPORT ERROR AND ABORT
                MVC           ..MAP..,EMPNAME           PUT EMPLOYEE NAME IN MAP
                UNPK           ..MAP..,EMPSICK            PUT SICK TIME IN MAP
                SPACE
                GETSORT SESSION 2 NEXT.
                CLC             TPSRETN,=CL4'0000'    SORTED SALES DATA RETRIEVED
?
                BNE             BADSORT               NO, REPORT ERROR AND ABORT
                MVC           ..MAP..,EMPNAME           PUT EMPLOYEE NAME IN MAP
                UNPK          ..MAP..,EMPRSNL           PUT PERSONAL TIME IN MAP
                BCT             R4,LOOP
                SPACE

*---------------------------- END SORT SESSION ------------------------------*
TERMSORT        EQU     *
                ENDSORT SESSION 1.
                CLC             TPSRETN,=CL4'0000'       SORT SESSION ENDED OK ?
                BNE           BADSORT                 NO, REPORT ERROR AND ABORT
                ENDSORT SESSION 2.
                CLC             TPSRETN,=CL4'0000'       SORT SESSION ENDED OK ?
                BNE           BADSORT                 NO, REPORT ERROR AND ABORT
                SPACE
*------------------------------- DISPLAY MAP --------------------------------*
DSPLYMAP        EQU     *
                .
                display may
                .
                BR              R3
                SPACE           2
```

```
*************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT IN    *
*          THE TPSRETN FIELD. TPSMSG FIELD CONTAINS A 79 CHARACTER MESSAGE*
*          FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.               *
*************************************************************************
BADSORT         EQU      *
                MVC           ..MAP..,TPSMSG         USE MESSAGE FROM CA IDMS/DC
SORT
                .
                display map
                .
                return to CA IDMS/DC
```

Exhibit 4.7: ASM Multiple Sessions--Assembler

```
TPSEXPL2: PROC OPTIONS(MAIN) REORDER;

/*      REMARKS.  THIS PL1 EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT
    TO EXECUTE TWO SORTS CONCURRENTLY. ONE ON ACCUMULATED SICK LEAVE
    IN DESCENDING ORDER AND ONE ON PERSONAL TIME USED IN DESCENDING
    ORDER. THE TOP TEN IN EACH CATEGORY ARE DISPLAYED.

   TPSORT REQUIRED COMPILE OPTION "MARGINS(2,72)".
*/

/*REQUIRED FOR IDMS*/
DCL (subschema_name SUBSCHEMA,schema_name SCHEMA)
MODE (IDMS_DC) DEBUG;
DCL IDMS ENTRY OPTIONS(INTER,ASM);
INCLUDE IDMS(SUBSCHEMA_CTRL);
/*END OF IDMS REQUIREMENT*/

DCL ADDR BUILTIN;

DCL END_OF_EMPLOYEES                                  CHAR(1);
DCL EMPLOYEE_COUNT                                    PIC'S99';
DCL MAX_EMPLY_IN_MAP                                  PIC'S99' INIT(10);

%INCLUDE TPSCOMMP;
.
.

        CALL SORT_EMPLOYEES_0100;
        CALL DISPLAY_TOP_TEN_0200;

        ...return to CA IDMS/DC.
```

```
/*************************************************************************
*          EXECUTE TWO SORTS, BOTH IN DESCENDING ORDER--ONE ON           *
*          ACCUMULATED SICK LEAVE AND THE OTHER ON ACCUMULATED PERSONAL  *
*          TIME. NOTE: THE RECORD BEING SORTED IS AN IDMS RECORD.    THE*
*          ELEMENT ATTRIBUTES DO NOT HAVE TO BE CODED IN THE SETSORT     *
*          STATEMENT, THEY WILL BE EXTRACTED BY CA IDMS/DC SORT FROM THE *
*          DICTIONARY. THE RECORD DEFINITION "EMPLYREC" WILL BE INSERTED *
*          BY THE PLI_IDMS PREPROCESSOR THROUGH THE IDMS DCL.            *
*************************************************************************

SORT_EMPLOYEES_0100: PROC;

        SETSORT SESSION 1 PROGRAM IDMS
                 FOR EMPLYREC VER 22
                 IN DICT TEST
        FIELD EMPLY_SICK_DAYS  DESCENDING
              EMPLY_NAME       ASCENDING;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
        SETSORT SESSION 2 PROGRAM IDMS
        FOR EMPLYREC VER 22
        IN DICT TEST
        FIELD EMPLY_PERSONAL_DAYS DESCENDING
              EMPLY_NAME          ASCENDING.
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_999;

END_OF_EMPLOYEES = 'N';

DO UNTIL (END_OF_EMPLOYEES = 'Y');
        CALL PUT_SORT_0150;
END;

END SORT_EMPLOYEES_0100;

PUT_SORT_0150: PROC;
.
.
...read an employee record, set END_OF_EMPLOYEES to 'Y' at end.
.
.
IF (END_OF_EMPLOYEES = 'N')
THEN DO;
        PUTSORT SESSION 1;
        IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
        PUTSORT SESSION 2;
        IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
END;

END PUT_SORT_0150;
```

```
*****************************************************************************
*          GET THE TOP TEN FROM THE SICK LEAVE AND PERSONAL TIME SORTS.     *
*          DISPLAY THE EMPLOYEE NAME AND TIME TAKEN.                        *
*****************************************************************************

DISPLAY_TOP_TEN_0200: PROC;

DO EMPLOYEE_COUNT = 1 TO MAX_EMPLY_IN_MAP BY 1;
        CALL GET_EMPLOYEE_0220
END;

ENDSORT SESSION 1;
IF (TPSRETN - '0000') THEN CALL SORT_ERROR_9999;
ENDSORT SESSION 2;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;

.
.
...display map
.
.

END DISPLAY_TO_TEN_0200;

GET_EMPLOYEE_0220: PROC;

GETSORT SESSION 1 NEXT;
IF (TPSRETN = '7020')
THEN
        EMPLOYEE_COUNT = MAX_EMPLY_IN_MAP; /*CLOSE DO_LOOP*/
ELSE
        IF (TPSRETN = '0000')
        THEN
              CALL SORT_ERROR_9999;
        ELSE
          DO;
              ...map = EMPLY_NAME;
              ...map = EMPLY_SICK_DAYS;
              GETSORT SESSION 2 NEXT;
              IF (TPSRETN = '0000'0
              THEN
                  CALL SORT_ERROR_9999;
```

```
            ELSE
              DO;
                ...map = EMPLY_NAME;
                ...map = EMPLY_PERSONAL_DAYS;
              END;
        END;

END GET_EMPLOYEE_0220;

/*****************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT      *
*          IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE  *
*          FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.              *
*****************************************************************************

SORT_ERROR_9999: PROC;

...message line in map = TPSMSG;
.
...display map
.
...return to CA IDMS/DC

END SORT_ERROR_9999;

END TPSEXPL2;
```

Exhibit 4.8: PLI Multiple Sessions--PLI

```
!*****************************************************************************
!*         THIS MODULE PERFORMS TWO SORTS - ONE ON ACCUMULATED SICK LEAVE     *
!*         IN DESCENDING ORDER AND ONE ON PERSONAL TIME IN DESCENDING         *
!*         ORDER. THE TOP TEN IN EACH CATEGORY ARE DISPLAYED ON ONE MAP.  *
!*****************************************************************************
SETSORT SESSION 1 PROGRAM IDMS FOR EMPLOYEE-SICK PERSONAL
FIELDS                  EMPLOYEE-SICK-LEAVE          DESCENDING
                EMPLOYEE-NAME                        ASCENDING.
WHILE (NOT DB-END-OF-SET) AND
      (TPSRETN EQUAL ZERO)
REPEAT.
                OBTAIN NEXT EMPLOYEE WITHIN EMPLOYEE-MASTER.
```

```
                PUTSORT SESSION 1.
                IF TPSRETN NE ZERO
                        DO.
                           DISPLAY MESSAGE TEXT TPSMSG.
!                                RETURN TO TOP.
                        END.
                PUTSORT SESSION 2.
                IF TPSRETN NE ZERO
                        DO.
                           DISPLAY MESSAGE TEXT TPSMSG.
!                                RETURN TO TOP.
                        END.
END.

GETSORT SESSION 1 FIRST.
IF TPSRETN NE ZERO
                DO.
                        DISPLAY MESSAGE TEXT TPSMSG.
!                            RETURN  TO TOP.
                END.

MOVE ZERO TO MAP-FIELD-SUBSCRIPT.

WHILE (END-OF-SICK-LEAVE EQUAL 'N') AND
      (MAP-FIELD-SUBSCRIPT LE 10)
REPEAT.
                MOVE SICK-LEAVE-MSG TO MAP -SICK-LEAVE-MSG (MAP-FIELD-SUBSCRIPT).
                MOVE EMPLOYEE-NAME  TO MAP-EMPLOYEE-NAME   (MAP-FIELD-SUBSCRIPT).
                MOVE EMPLOYEE-NUM   TO MAP-EMPLOYEE-NUM    (MAP-FIELD-SUBSCRIPT).
                MOVE EMPLOYEE-SICK-LEAVE
                     TO            MAP-EMPLOYEE-SICK-MSG  (MAP-FIELD-SUBSCRIPT).
                ADD 1 TO MAP-FIELD-SUBSCRIPT.

                GETSORT SESSION 1 NEXT.
                IF TPSRETN EQUAL '7020'
                        DO.
                           MOVE 'Y' TO END-OF-SICK-LEAVE.
                        END.
                ELSE
                        IF TPSRETN NE ZERO
                              DO.
                                 DISPLAY MESSAGE TEXT TPSMSG.
!                                      RETURN TO TOP.

!               IF TPSRETN = ZERO CONTINUE IN ITERATION.
```

```
END.

          GETSORT SESSION 2 FIRST.
IF TPSRETN NE ZERO
        DO.
                DISPLAY MESSAGE TEXT TPSMSG.
!                       RETURN TO TOP.
        END.

MOVE ZERO TO MAP-FIELD-SUBSCRIPT.

WHILE (END-OF PERSONAL-LEAVE EQUAL 'N') AND
        (MAP-FIELD-SUBSCRIPT LE 10)
REPEAT.
  MOVE PERSONAL-LEAVE-MSG TO
         MAP-PERSONAL-LEAVE-MSG (MAP-FIELD-SUBSCRIPT).
  MOVE EMPLOYEE-NAME    TO MAP-EMPLOYEE-NAME    (MAP-FIELD-SUBSCRIPT).
  MOVE EMPLOYEE-NUM     TO MAP-EMPLOYEE-NUM     (MAP-FIELD-SUBSCRIPT).
  MOVE EMPLOYEE-PERSONAL-LEAVE TO
         MAP-EMPLOYEE-PERSONAL-LEAVE              (MAP-FIELD-SUBSCRIPT).
  ADD 1 TO MAP-FIELD-SUBSCRIPT.
  GETSORT SESSION 2 NEXT.
  IF TPSRETN EQUAL '7020'
        DO.
           MOVE 'Y' TO END-OF-PERSONAL-LEAVE.
        END.
ELSE
        IF TPSRETN NE ZERO
                DO.
                        DISPLAY MESSAGE TEXT TPSMSG.
!                               RETURN TO TOP.
                END.
!       IF TPSRETN = ZERO CONTINUE IN ITERATION.
END.

ENDSORT SESSION 1.
IF TPSRETN NE ZERO
        DO.
           DISPLAY MESSAGE TEXT TPSMSG.
!                  RETURN TO TOP.
        END.
```

```
ENDSORT SESSION 2.
IF TPSRETN NE ZERO
        DO.
           DISPLAY MESSAGE TEXT TPSMSG.
!                  RETURN TO TOP.
        END.

DISPLAY.
```

Exhibit 4.9: ADS Multiple Sessions--ADS

```
IDENTIFICATION DIVISION.
PROGRAM-ID.   TPSEXPL3.
REMARKS.  THIS COBOL EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO
    DISPLAY THE SALES DATA FOR A GIVEN SALES PERSON. THIS PROGRAM IS
    PSEUDO CONVERSATIONAL.

ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
77    DISPLAY-COUNT                                  PIC S9(9) COMP.
77    END-OF-DISPLAY                                 PIC X.
77    END-OF-SALES                                   PIC X.

01    SALES-DATA.
05    SALES-ITEM-NAME          PIC X(25).
05    SALES-AMOUNT       PIC S9(9)V99 COMP-3.
05    SALES-QTY                PIC S9(9)    COMP-3.
05    SALES-DATE               PIC X(08).

COPY SALESREC
.
COPY TPSCOMMC
.
PROCEDURE DIVISION.

IF ...first time
```

```
THEN
   ...set first time off
   PERFORM 0100-GET-SORTED-SALES-DATA.

MOVE 'N' TO END-OF-DISPLAY.
PERFORM DISPLAY-SALES-DATA.

IF END-OF-DISPLAY = 'Y'
THEN
   PERFORM 9000-END-SORT
   ...return to CA IDMS/DC
ELSE
   ...return to CA IDMS/DC with next task code
      for this program.

*****************************************************************************
*          SORT SALES DATA FOR A GIVEN SALES PERSON BY ITEM AND DATE        *
*          SOLD. NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES   *
*          DATA NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK         *
*          RECORD FOR SORTING EFFICIENCY.                                   *
*****************************************************************************

0100-GET-SORTED-SALES-DATA    SECTION.

SETSORT PROGRAM
      FOR SALES-DATA LENGTH 44
      FIELD     SALES-ITEM-NAME                    25 ASCENDING
                SALES-DATE      8 DESCENDING.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

        MOVE 'N' END-OF-SALES.
PERFORM 0150-PUT-SORT UNTIL END-OF-SALES = 'Y'.

SECTION-EXIT.
EXIT.

0150-PUT-SORT    SECTION.
.
.
...read a sales record for the sales person,
   when all records have been read for sales person
   move 'Y' to END-OF-SALES
```

```
.
.
IF END-OF-SALES = 'N'
THEN
        MOVE  SALESREC-SALES-ITEM                    TO  SALES-ITEM
        MOVE  SALESREC-SALES-AMOUNT                  TO  SALES-AMOUNT
        MOVE  SALESREC-SALES-QTY                     TO  SALES-QTY
        MOVE  SALESREC-SALES-DATE                    TO  SALES-DATE
        PUTSORT.
        IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

SECTION-EXIT.
EXIT.

*****************************************************************************
*          DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY SALES PERSON.         *
*****************************************************************************

0200-DISPLAY-SALES-DATA    SECTION.

PERFORM 0220-GET-SALES-ITEM
        VARYING DISPLAY-COUNT
        FROM 1 BY 1 UNTIL   (DISPLAY-COUNT > 20)
                     OR        (END-OF-DISPLAY = 'Y')

IF END-OF-DISPLAY = 'Y'
THEN
        MOVE 'NO MORE ITEMS FOR SALES PERSON'
              TO ...message in map.
ELSE
        MOVE 'MORE ITEMS FOLLOW FOR SALES PERSON'
              TO ...message in map.
.
.
...display map
.
.
SECTION-EXIT.
EXIT.
   0220-GET-SALES-ITEM    SECTION.

**      CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE BETWEEN
**      PSEUDO CONVERSES. THEREFORE, NO REPOSITIONING IS REQUIRED.

GETSORT NEXT.
IF TPSRETN = '0000'
```

```
THEN
        MOVE  SALES-ITEM-NAME                          TO ...map
        MOVE  SALES-AMOUNT                             TO ...map
        MOVE  SALES-QTY                                TO ...map
        MOVE  SALES-DATE                               TO ...map
ELSE
        IF TPSRETN = '7020'
        THEN
              MOVE 'Y' TO END-OF-DISPLAY
        ELSE
              PERFORM 9999-SORT-ERROR.

 SECTION-EXIT.
EXIT.

*****************************************************************************
*          END CURRENT SESSION OF CA IDMS/DC SORT.                          *
*****************************************************************************

 9000-END-SORT    SECTION.

ENDSORT.
IF TPSRETN NOT = '0000' THEN PERFORM 9999-SORT-ERROR.

 SECTION-EXIT.
EXIT.

*****************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT     *
*          IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE  *
*          FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.             *
*****************************************************************************

 9999-SORT-ERROR    SECTION.

MOVE TPSMSG TO ...message line in map
.
...display map
.
...return to CA IDMS/DC

 SECTION-EXIT.
EXIT.
```

Exhibit 4.10: COB Pseudo Conversational--COBOL

```
      TITLE 'TPSEXPL3 -- PSEUDO CONVERSATIONAL'
*************************************************************************
*          THIS ASSEMBLER EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO    *
*          DISPLAY THE SALES DATA FOR A GIVEN SALES PERSON.     THIS PROGRAM IS*
*          PSEUDO CONVERSATIONAL.                                              *
*************************************************************************
name...     DSECT
SLDATA     DS      0XL44                        EXTRACTED DATA FROM SALES RECORD
SLITEM      DS       CL25                             NAME OF ITEM SOLD
SLAMT       DS       PL6                              AMOUNT ITEM SOLD FOR
SLAMT       DS       PL5                              NUMBER OF ITEMS SOLD
SLDATE      DS       CL8                              DATE ITEM WAS SOLD
.
ENDSALE     DS       C                                END OF SALES INDICATOR
ENDDSPLY            DS    C                          END OF DISPLAY INDICATOR
.
            COPY SALESREC        SALES RECORD
            .
            COPY TPSCOMMA        CA IDMS/DC SORT COMMUNICATIONS BLOCK
            .
R3          EQU      3                               BAL - SUBROUTINE LINKAGE
R4          EQU      4                               BCT - LOOP COUNTER
.
TPSEXPL3 CSECT
.
            CLI         ...first time     IS THIS FIRST TIME ?
            BNE        MAIN0100                   NO, SKIP EXTRACT AND SORT THEN
            MVI         ...first time                          TURN FIRST TIME
INDICATOR
OFF
```

```
            BAL         R3,SORTSALE                  EXTRACT AND SORT SALES DATA
            SPACE
            MAIN0100            EQU
            MVI         ENDDSPLY,C'N'
            BAL         R3,DISPLAY                    DISPLAY SALES DATA
            SPACE
            CLI         ENDDSPLY,C'Y'                HAVE ALL SALES BEEN DISPLAYED
?
            BE          NOMORE                       YES, END SORT PSEUDO CONVERSE
            ...return to CA IDMS/DC with next code for this program
            SPACE
            NOMORE      EQU      *
            BAL         R3,TERMSORT                   RELEASE SORT
            ...return to CA IDMS/DC
            SPACE       2
 ***************************************************************************
 *          SORT SALES DATA FOR A GIVEN SALES PERSON BY ITEM AND DATE SOLD.   *
 *          NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA        *
 *          NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK RECORD FOR    *
 *          SORTING EFFICIENCY.                                                *
 ***************************************************************************
 SORTSALE            EQU   *
            SETSORT PROGRAM
                        FOR SLDATA LENGTH 44
                        FIELD SLITEM 25 ASCENDING
                              SLDATE  8 DESCENDING.
            CLC         TPSRETN,=CL4'0000'            SUCCESSFUL SETSORT ?
            BNE         BADSORT                      NO, REPORT ERROR AND ABORT
            MVI         ENDSALE,C'N'                  INITIALIZE FOR LOOP

            SPACE
 PUTLOOP    EQU          *
            .
            ...read a sales record, set ENDSALE to 'Y' at end
            .

            CLI           ENDSALE,'Y'                  ANY MORE SALES PERSONS ?
            BER           R3           NO, RETURN
            MVC           SLITEM,...      SAVE NAME OF ITEM SOLD
            ZAP           SLAMT,...       SAVE AMOUNT OF SALE
            ZAP           SLQTY,...       SAVE QUANTITY SOLD
            MVC           SLDATE,...      SAVE DATE ITEM SOLD
            CLC           TPSRETN,=CL4'0000'          SALES DATA ACCEPT
            ED BY SORT ?
            BE            PUTLOOP                     YES, CONTINUE EXTRACT
            B             BADSORT                     NO, REPORT ERROR AND ABORT
            SPACE         2
 ***************************************************************************
```

```
*          DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY THE SALES PERSON.      *
*          NOTE: CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE   *
*          BETWEEN PSEUDO CONVERSES, THEREFORE, NO REPOSITIONING IS          *
*          REQUIRED.                                                         *
*****************************************************************************
DISPLAY     EQU          *
            LA           R4,20        NUMBER OF ITEMS PER SCREEN
            SPACE
GETLOOP     EQU          *
            GETSORT NEXT.
            CLC          TPSRETN,=CL4'7020'          END OF SORTED DATA ?
            BE           LAST                       YES, INDICATE NO MORE ITEMS
            CLC          TPSRETN,=CL4'0000'          SORTED SALES DATA RETRIEVED
?
            BNE          BADSORT                     NO, REPORT ERROR AND ABORT
            MVC          ..map..,SLITEM  NAME OF ITEM SOLD
            UNPK         ..map..,SLAMT   AMOUNT ITEM SOLD FOR
            UNPK         ..map..,SLQTY   NUMBER OF ITEMS SOLD
            MVC          ..map..,SLDATE  DATE ITEM SOLD
            BCT          R4,GETLOOP
            MVC          ..message in map..,MSGMORE
            SPACE
DSPLYMAP    EQU          *
            .
            ...display map
            .
            BR           R3
            SPACE
            LAST         EQU          *
            MVI          ENDDSPLY,C'Y'             INDICATE LAST SCREEN OF DISPLAY
            MVC          ..message in map..,MSGLAST
            B            DSPLYMAP


            SPACE        2
*****************************************************************************
*          END CURRENT SESSION OF CA IDMS/DC SORT.                          *
*****************************************************************************
TERMSORT    EQU          *
            ENDSORT.
            CLC          TPSRETN,=CL4'0000'          SESSION ENDED OK ?
            BER          R3           YES
            B            BADSORT      NO
            SPACE        2
*****************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT IN    *
*          THE TPSRETN FIELD. TPSMSG FIELD CONTAINS A 79 CHARACTER MESSAGE*
*          FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.              *
*****************************************************************************
```

```
BADSORT     EQU          *
            MVC           ..map..,TPSMSG   USE MESSAGE FROM CA IDMS/DC SORT
            .
            ...display map
            .
            ...return to CA IDMS/DC
            SPACE         2
            MSGMORE       DC        CL79'MORE ITEMS FOLLOW FOR SALESPERSON'
            MSGLAST       DC        CL79'NO MORE ITEMS FOR SALES PERSON'
            .
            .
            .
```

Exhibit 4.11: ASM Pseudo Conversational--Assembler

```
TPSEXPL3: PROC OPTIONS(MAIN) REORDER;

/*      REMARKS.  THIS PL1 EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT

        TO DISPLAY THE SALES DATA FOR A GIVEN SALES PERSON. THIS PROGRAM
        IS PSEUDO CONVERSATIONAL.

        CA IDMS/DC SORT REQUIRES COMPILE OPTION "MARGINS(2,72)".
*/

/*REQUIRED FOR IDMS*/
DCL MODE (IDMS_DC) DEBUG;
DCL IDMS ENTRY OPTIONS(INTER,ASM);
INCLUDE IDMS(SUBSCHEMA_CTRL);
/*END OF IDMS REQUIREMENT*/

DCL ADDR BUILTIN;

DCL DISPLAY_COUNT                                    FIXED    BIN(31);
DCL END_OF_DISPLAY                                            CHAR(1);
DCL END_OF_SALES                                              CHAR(1);

DCL 1 SALES_DATA,
    2 SALES_ITEM_NAME                                          CHAR(25),
    2 SALES_AMOUNT                                           PIC'S999999999V99',
```

```
    2 SALES_QTY                                                  PIC'S999999999',
    2 SALES_DATE                                                 CHAR(8);

%INCLUDE SALESREC;
.
%INCLUDE TPSCOMMP;
.
.
        IF (...first time)
        THEN DO;
              ...set first time off;
              CALL GET_SORTED_SALES_DATA_0100;
        END;

        END_OF_DISPLAY = 'N';
        CALL DISPLAY_SALES_DATA_0200;

        IF (END-OF-DISPLAY = 'Y')
          THEN DO;
              END_SORT_9000;
              ...return to CA IDMS/DC;
          END;
        ELSE
              ...return to CA IDMS/DC with next task code
                 for this program;

/*************************************************************************
*          SORT SALES DATA FOR A GIVEN SALES PERSON BY ITEM AND DATE         *
*          SOLD. NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES   *
*          DATA NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK          *
*          RECORD FOR SORTING EFFICIENCY.                                    *
*************************************************************************/

GET_SORTED_SALES_DATA_0100: PROC;

SETSORT PROGRAM
        FOR SALES_DATA LENGTH 53
        FIELD SALES_ITEM_NAME   25 ASCENDING
              SALES_DATE         8 DESCENDING.
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;

END_OF_SALES = 'N';
DO UNTIL (END_OF_SALES = 'Y');
```

```
        CALL PUT_SORT_0150;
END;

END GET_SORTED_SALES_DATA_0100;

PUT_SORT_0150: PROC;
.
.
...read sales record for the sales person;
   when all records have been read for sales person
   END_OF_SALES = 'Y';
.
.
IF (END_OF_SALES = 'N')
THEN DO;
        SALES_ITEM     = SALESREC_SALES_ITEM;
        SALES_AMOUNT   = SALESREC_SALES_AMOUNT;
        SALES_QTY      = SALESREC_SALES_QTY;
        SALES_DATE     = SALESREC_SALES_DATE;
        PUTSORT;
        IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
END;

END PUT_SORT_0150;;

/*****************************************************************************
*          DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY THE SALES PERSON.      *
*****************************************************************************

DISPLAY_SALES_DATA_0200: PROC;

DO DISPLAY_COUNT = 1 TO 20 BY 1
        UNTIL (END_OF_DISPLAY = 'Y');
          CALL GET_SALES_ITEM_0220;
END;

IF (END-OF-DISPLAY = 'Y')
THEN
        ...message in map = 'NO MORE ITEMS FOR SALES PERSON';
ELSE
        ...message in map = 'MORE ITEMS FOLLOW FOR SALES PERSON';
.
...display map
```

```
.
END DISPLAY_SALES_DATA_0200;

       GET_SALES_ITEM_0220: PROC;
/*
**      CA IDMS/DC SORT keeps its currency within the sorted file between
**      pseudo converses, therefore, no repositioning is required.
*/
GETSORT NEXT INTO SALES_DATA;
IF (TPSRETN = '0000')
  THEN DO;
        ...map = SALES_ITEM_NAME;
        ...map = SALES_AMOUNT;
        ...map = SALES_QTY;
        ...map = SALES_DATE;
  END;
ELSE
        IF (TPSRETN = '7020')
        THEN
              END_OF_DISPLAY = 'Y';
        ELSE
              CALL SORT_ERROR_9999;

 END GET_SALES_ITEM_0220;

/*************************************************************************
*          END CURRENT SESSION OF CA IDMS/DC SORT.                       *
*************************************************************************

 END_SORT_9000: PROC;

ENDSORT;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;

 END END_SORT_9000;

/*************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT   *
*          IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE  *
*          FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.          *
*************************************************************************
```

```
 SORT_ERROR_9999: PROC;

...message line in map = TPSMSG;
.
...display map
.
...return to CA IDMS/DC

 END SORT_ERROR_9999;

 END TPSEXPL3;
```

Exhibit 4.12: PLI Pseudo Conversational--PLI

```
!       ************************************************************************
!       *     THIS ADS EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO       *
!       *     DISPLAY THE SALES DATA FOR A GIVEN SALES PERSON.     THE ENTIRE  *
!       *     SORTED DETAILS CANNOT BE DISPLAYED ON A SINGLE SCREEN.     HENCE,*
!       *     THE DETAILS ARE KEPT IN SORTED ORDER ACROSS MAP DISPLAYS.        *
!       ************************************************************************
IF      ...first time
        DO.
              ...set first time off
              CALL GET-SORTED-SALES-DATA.
        END.

MOVE 'N' TO END-OF-DISPLAY.
WHILE (DISPLAY-COUNT < 21) AND
         (END-OF_DISPLAY NE 'Y')
REPEAT.
!       ************************************************************************
!       *     CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE        *
!       *     BETWEEN PSEUDO CONVERSES; THEREFORE, NO REPOSITIONING IS         *
!       *     REQUIRED.                                                      *
!       ************************************************************************
        GETSORT NEXT.
        IF TPSRETN EQUAL ZERO
          DO.
              MOVE SALES-ITEM-NAME TO MAP-SALES-ITEM NAME   (DISPLAY-COUNT).
              MOVE SALES-AMOUNT    TO MAP-SALES-AMOUNT      (DISPLAY-COUNT).
              MOVE SALES-QTY       TO MAP-SALES-QTY         (DISPLAY-COUNT).
              MOVE SALES-DATE      TO MAP-SALES-DATE        (DISPLAY-COUNT).
```

```
            ADD 1 TO DISPLAY-COUNT
          END.
        ELSE
          IF TPSRETN EQUAL '7020'
            MOVE 'Y'   TO   END-OF-DISPLAY.
          ELSE
            CALL SORT-ERROR.
END.

IF END-OF-DISPLAY = 'Y'
        DO.
            ENDSORT.
            IF TPSRETN NOT EQUAL ZERO
              DO.
                CALL SORT-ERROR.
              END.
            DISPLAY MESSAGE TEXT 'NO MORE ITEMS FOR SALES PERSON'.
        END.
ELSE
        DISPLAY CONTINUE
            MESSAGE TEXT 'MORE ITEMS FOLLOW FOR SALES PERSON'.

!       ************************************************************************
!       *GET-SORTED-SALES-DATA                                                 *
!       *                                                                      *
!       *     SORT SALES DATA FOR A GIVEN SALES PERSON BY ITEM AND DATE SOLD.  *
!       *     NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA       *
!       *     NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK RECORD       *
!       *     FOR SORTING EFFICIENCY.                                          *
!       ************************************************************************
DEFINE SUBROUTINE GET-SORTED-SALES-DATA.

SETSORT PROGRAM IDMS FOR SALES-DATA
        FIELD SALES-ITEM-NAME   ASCENDING
              SALES-DATE        DESCENDING.
IF TPSRETN NOT EQUAL ZERO
        CALL SORT-ERROR.

MOVE 'N' TO END-OF-SALES
WHILE END-OF-SALES NOT EQUAL 'Y'
REPEAT.
  .
  .
```

```
  ...obtain a sales record for the sales person, when all records
  have been processed for this sales person, move 'Y' to
  END-OF-SALES.
  .
  .
  IF END-OF SALES = 'N'
        DO.
              MOVE SALESREC-SALES-ITEM     TO SALES-ITEM.
              MOVE SALESREC-SALES-AMOUNT   TO SALES-AMOUNT.
              MOVE SALESREC-SALES-QTY      TO SALES-QTY.
              MOVE SALESREC-SALES-DATE     TO SALES-DATE.
              PUTSORT.
              IF TPSRETN NOT EQUAL ZERO
                    CALL SORT-ERROR.
        END.
END.
GOBACK.


!       ************************************************************************
!       *SORT-ERROR                                                            *
!       *                                                                      *
!       *     AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT     *
!       *     IN THE TPSRETN FIELD. TPMSG CONTAINS A 79 CHARACTER MESSAGE  *
!       *     FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.             *
!       ************************************************************************
DEFINE SUBROUTINE SORT-ERROR.

DISPLAY MESSAGE TEXT TPMSG.
!       RETURN TO TOP.
GOBACK.
```

Exhibit 4.13: ADS Pseudo Conversational--ADS

```
IDENTIFICATION DIVISION.
PROGRAM-ID.   TPSEXPL4
REMARKS.   THIS COBOL EXAMPLE IS THE SAME AS EXAMPLE 3, EXCEPT A "USER"
SORT HAS BEEN SPECIFIED INSTEAD OF A "PROGRAM" SORT. THE PROGRAM
IS PSEUDO CONVERSATIONAL AND CAN SORT ANY OR ALL OF THE SALES DATA
FIELDS IN EITHER ASCENDING OR DESCENDING ORDER AT THE USERS
DISCRETION AT EXECUTION TIME.

ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
77      DISPLAY-COUNT                              PIC S9(9) COMP.
```

```
77      END-OF-DISPLAY                              PIC X.
77      END-OF-SALES                                PIC X.

01      SALES-DATA.
05      SALES-ITEM-NAME                             PIC X(25).
05      SALES-AMOUNT       PIC S9(9)V99 COMP-3.
05      SALES-QTY                                   PIC S9(9)    COMP-3.
05      SALES-DATE         PIC X(08).

COPY SALESREC
.
COPY TPSCOMMC
.
.

PROCEDURE DIVISION.

IF ...first time
THEN
        ...set first time off
        PERFORM 0100-GET-SORTED-SALES-DATA.

MOVE 'N' TO END-OF-DISPLAY.
PERFORM DISPLAY-SALES-DATA.

IF END-OF-DISPLAY - 'Y'
THEN
        PERFORM 9000-END-SORT
        ...return to CA IDMS/DC
ELSE
        ...return to CA IDMS/DC with next task code
           for this program.

*****************************************************************************
*          SORT ORDER WILL BE CONTROLLED BY THE USER.     THE USER CAN       *
*          SELECT ANY OR ALL OF THE FIELDS IN THE SALES-DATA-WORK            *
*          RECORD AS A SORT KEY. EACH SELECTED SORT KEY CAN BE ORDERED   *
*          EITHER IN ASCENDING OR DESCENDING SEQUENCE.                       *
*****************************************************************************
```

```
0100-GET-SORTED-SALES-DATA    SECTION.

SETSORT USER
        FOR SALES-DATA   LENGTH 44
        FIELD SALES-ITEM-NAME   25
              SALES-AMOUNT       6
              SALES-QTY          5
              SALES-DATE         8
IF TPSRETN NOT - '0000' PERFORM 9999-SORT-ERROR.

MOVE 'N' END-OF-SALES.
PERFORM  0150-PUT-SORT UNTIL END-OS-SALES = 'Y'.

SECTION-EXIT.
EXIT.

0150-PUT-SORT    SECTION.
.
...read a sales record for the sales person,
   when all records have been read for sales person
   move 'y' to END-OF-SALES
.
IF END-OF-SALES = 'N'
THEN
        MOVE SALESREC-SALES-ITEM                    TO SALES-ITEM
        MOVE SALESREC-SALES-AMOUNT                  TO SALES-AMOUNT
        MOVE SALES-REC-SALES-QTY                    TO SALES-QTY
        MOVE SALESREC-SALES-DATE                    TO SALES-DATE
        PUTSORT.
        IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

SECTION-EXIT.
EXIT.

*************************************************************************
*          DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY SALES PERSON      *
*************************************************************************

0200-DISPLAY-SALES-DATA    SECTION.

PERFORM 0220-GET-SALES-ITEM
        VARYING DISPLAY-COUNT
        FROM 1 BY 1 UNTIL  (DISPLAY-COUNT  >  20)
                        OR   (END-OF-DISPLAY = 'Y')
IF END-OF-DISPLAY = 'Y'
THEN
        MOVE 'NO MORE ITEMS FOR SALES PERSON'
```

```
              TO ...message in map.
ELSE
        MOVE 'MORE ITEMS FOLLOW FOR SALES PERSON'
              TO ...message in map.
.
...display map
.
SECTION-EXIT.
EXIT.
 0220-GET-SALES-ITEM    SECTION.

**      CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE BETWEEN
**      PSEUDO CONVERSES. THEREFORE, NO REPOSITIONING IS REQUIRED.

GETSORT NEXT.
IF TPSRETN = '0000'
THEN
        MOVE SALES-ITEM-NAME                          TO ...map
        MOVE SALES-AMOUNT                             TO ...map
        MOVE SALES-QTY                                TO ...map
        MOVE SALES-DATE                               TO ...map
ELSE
        IF TPSRETN = '7020'
        THEN
              MOVE 'Y' TO END-OF-DISPLAY
        ELSE
              PERFORM 9999-SORT-ERROR.

 SECTION-EXIT.
EXIT.

*****************************************************************************
*          END CURRENT SESSION OF CA IDMS/DC SORT.                          *
*****************************************************************************

 9000-END-SORT    SECTION.

ENDSORT.
IF TPSRETN NOT = '0000' THEN PERFORM 9999-SORT-ERROR.

 SECTION-EXIT.
EXIT.
```

```
*****************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT IN   *
*          THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE FROM*
*          CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.                   *
*****************************************************************************

 9999-SORT-ERROR    SECTION.

MOVE TPSMSG TO ...message line in map
.
...display map
.
...return to CA IDMS/DC

 SECTION-EXIT.
EXIT.
```

Exhibit 4.14: COB Pseudo Conversational, USER Option--COBOL

```
               TITLE 'TPSEXPL4 -- USER SORT -- PSEUDO CONVERSATIONAL'
*****************************************************************************
*          THIS ASSEMBLER EXAMPLE IS THE SAME AS EXAMPLE 3, EXCEPT A "USER"    *
*          SORT HAS BEEN SPECIFIED INSTEAD OF A "PROGRAM" SORT. THE PROGRAM    *
*          IS PSEUDO CONVERSATIONAL AND CAN SORT ANY OR ALL OF THE SALES       *
*          DATA FIELDS IN EITHER ASCENDING OR DESCENDING ORDER AT THE          *
*          USER'S DISCRETION AT EXECUTION TIME.                                *
*****************************************************************************
name...       DSECT
SLDATA       DS    0XL44                      EXTRACTED DATA FROM SALES RECORD
SLITEM        DS     CL25                          NAME OF ITEM SOLD
SLAMT                DS    PL6                     AMOUNT ITEM SOLD FOR
SLAMT                DS    PL5                     NUMBER OF ITEMS SOLD
SLDATE        DS     CL8                           DATE ITEM WAS SOLD
        .
ENDSALE       DS     C                             END OF SALES INDICATOR
ENDDSPLY      DS     C                             END OF DISPLAY INDICATOR
        .
        COPY         SALESREC                      SALES RECORD
        .
        COPY        TPSCOMMA                    CA IDMS/DC SORT COMMUNICATIONS
BLOCK
        .
R3            EQU    3                           BAL - SUBROUTINE LINKAGE
R4            EQU    4                           BCT - LOOP COUNTER
```

```
         .
TPSEXPL4      CSECT
         .
         .
         CLI   ...first time                          IS THIS FIRST TIME ?
         BNE   MAIN0100     NO, SKIP EXTRACT AND SORT THEN
         MVI   ...first time                        TURN FIRST TIME INDICATOR OFF
         BAL   R3,SORTSALE                           EXTRACT AND SORT SALES DATA
         SPACE
MAIN0100      EQU    *
         MVI   ENDDSPLY,C'N'
         BAL   R3,DISPLAY                            DISPLAY SALES DATA
         SPACE
         CLI   ENDDSPLY,C'Y'                       HAVE ALL SALES BEEN DISPLAYED
?
         BE    NOMORE       YES, END SORT PSEUDO CONVERSE
         ...return to CA IDMS/DC with next code for this program
         SPACE
NOMORE        EQU    *
         BAL   R3,TERMSORT                           RELEASE SORT
         ...return to CA IDMS/DC
         SPACE        2

*****************************************************************************
*          SORT ORDER WILL BE CONTROLLED BY THE USER.     THE USER CAN SELECT*
*          ANY OR ALL OF THE FIELDS IN THE SLDATA WORK RECORD AS A SORT KEY. *
*          EACH SELECTED SORT KEY CAN BE ORDERED EITHER IN ASCENDING OR      *
*          DESCENDING SEQUENCE.                                               *
*****************************************************************************
SORTSALE      EQU    *
         SETSORT USER
              FOR SLDATA LENGTH 44
              FIELD SLITEM  25
                      SLAMT    6
                      SLQTY    5
                      SLDATE   8.
         CLC   TPSRETN,=CL4'0000'                    SUCCESSFUL SETSORT ?
         BNE   BADSORT      NO, REPORT ERROR AND ABORT
         MVI   ENDSALE,C'N'                          INITIALIZE FOR LOOP
         SPACE
PUTLOOP       EQU    *
         .
         ...read a sales record, set ENDSALE to 'Y' at end
```

```
         .
         CLI   ENDSALE, 'Y'                          ANY MORE SALES PERSONS ?
         BER   R3          NO, RETURN
         MVC   SLITEM,...                            SAVE NAME OF ITEM SOLD
         ZAP   SLAMT,...                             SAVE AMOUNT OF SALE
         ZAP   SLQTY,...                             SAVE QUANTITY SOLD
         MVC   SLDATE,...                            SAVE DATE ITEM SOLD
         PUTSORT.
         CLC   TPSRETN,=CL4'0000'                    SALES DATA ACCEPTED BY SORT
?
         BE    PUTLOOP     YES, CONTINUE EXTRACT
         B     BADSORT     NO, REPORT ERROR AND ABORT
         SPACE 2
 ***************************************************************************
 *          DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY THE SALES PERSON.   *
 *          NOTE: CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE *
 *          BETWEEN PSEUDO CONVERSES.     THEREFORE, NO REPOSITIONING IS    *
 *          REQUIRED.                                                       *
 ***************************************************************************
 DISPLAY      EQU   *
         LA    R4,20       NUMBER OF ITEMS PER SCREEN
         SPACE
 GETLOOP      EQU   *
         GETSORT NEXT.
         CLC   TPSRETN,=CL4'7020'                    END OF SORTED DATA ?
         BE    LAST                                  YES, INDICATE NO MORE ITEMS
         CLC   TPSRETN,=CL4'0000'                    SORTED SALES DATA RETRIEVED
?
         BNE   BADSORT     NO, REPORT ERROR AND ABORT
         MVC   ..map..,SLITEM                        NAME OF ITEM SOLD
         UNPK        ..map..,SLAMT                   AMOUNT ITEM SOLD FOR
         UNPK        ..map..,SLQTY                   NUMBER OF ITEMS SOLD
         MVC   ..map..,SLDATE                        DATE ITEM SOLD
         BCT   R4,GETLOOP
         MVC   ..message in map..,MSGMORE
```

```
         SPACE
DSPLYMAP       EQU                              *
         .
         ..display map
         .
         BR           R3
         SPACE
LAST           EQU          *
         MVI          ENDDSPLY,C'Y'                  INDICATE LAST SCREEN OF DISPLAY
         MVC           ..message in map..,MSGLAST
         B             DSPLYMAP
         SPACE         2
*************************************************************************
*          END CURRENT SESSION OF CA IDMS/DC SORT.                      *
*************************************************************************
TERMSORT       EQU          *
         ENDSORT.
         CLC           TPSRETN,=CL4'0000'              SESSION ENDED OK ?
         BER           R3     YES
         B             BADSORT                         NO
         SPACE         2
*************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT  *
*          IN THE TPSRETN FIELD. TPSMSG FIELD CONTAINS A 79 CHARACTER    *
*          MESSAGE FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.  *
*************************************************************************
BADSORT        EQU          *
         MVC           ..map..,TPSMSG                  USE MESSAGE FROM CA IDMS/DC
SORT
         .
         ...display map
         .
         ...return to CA IDMS/DC
         SPACE         2
MSGMORE        DC            CL79'MORE ITEMS FOLLOW FOR SALES PERSON'
MSGLAST        DC            CL79'NO MORE ITEMS FOR SALES PERSON'
         .
         .
         .
```

Exhibit 4.15: ASM Pseudo Conversational, USER Option--Assembler

```
TPSEXPL4: PROC OPTIONS(MAIN) REORDER;

/*      REMARKS.  THIS PLI EXAMPLE IS THE SAME AS EXAMPLE 3, EXCEPT A
        "USER" SORT HAS BEEN SPECIFIED INSTEAD OF A "PROGRAM" SORT.
        THE PROGRAM IS PSEUDO CONVERSATIONAL AND CAN SORT ANY OR
        ALL OF THE SALES DATA FIELDS IN EITHER ASCENDING OR
        DESCENDING ORDER AT THE USERS DISCRETION AT EXECUTION TIME.

  CA IDMS/DC SORT REQUIRES COMPILE OPTION "MARGINS(2,72)".
*/

/*REQUIRED FOR IDMS*/
DCL MODE (IDMS_DC) DEBUG;
DCL IDMS ENTRY OPTIONS(INTER,ASM);
INCLUDE IDMS(SUBSCHEMA_CTRL);
/*END OF IDMS REQUIREMENT*/

DCL ADDR BUILTIN;

DCL DISPLAY_COUNT                                    FIXED  BIN(31);
DCL END_OF_DISPLAY                                          CHAR(1);
DCL END_OF_SALES                                            CHAR(1);

DCL     1 SALES_DATA,
2 SALES_ITEM_NAME               CHAR(25),
2 SALES_AMOUNT                  PIC'S999999999V99'.
2 SALES_QTY                     PIC'S999999999',
2 SALES_DATE                    CHAR(8);

%INCLUDE SALESREC;
.
%INCLUDE TPSCOMMP;
.
.


        IF (..first time)
              THEN DO:
```

```
                ...set first time off;
                CALL GET_SORTED_SALES_DATA_0100
              END;

        END_OF_DISPLAY = 'N';
        CALL DISPLAY_SALES_DATA_0200;

        IF (END_OF_DISPLAY = 'Y')
              THEN DO;
                ...return to CA IDMS/DC;
              END;
        ELSE
                ...return to CA IDMS/DC with next task code
                     for this program;

/*************************************************************************
*          SORT ORDER WILL BE CONTROLLED BY THE USER.     THE USER CAN SELECT*
*          ANY OR ALL OF THE FIELDS IN THE SALES-DATA WORK RECORD AS A       *
*          SORT KEY. EACH SELECTED SORT KEY CAN BE ORDERED EITHER IN     *
*          ASCENDING OR DESCENDING SEQUENCE.                                 *
*************************************************************************

GET_SORTED_SALES_DATA_0100: PROC;

SETSORT USER
        FOR SALES_DATA     LENGTH 53
        FIELDS  SALES_ITEM_NAME   25
                 SALES_AMOUNT      11
                 SALES_QTY          9
                 SALES_DATE         8;
IF (TPSRETN = '0000) CALL SORT_ERROR_9999;

END_OF_SALES = 'N';
DO UNTIL (END_OF_SALES = 'Y');
        CALL PUT_SORT_0150;
END;

END GET_SORTED_SALES_0100;

PUT_SORT_0150: PROC;
```

```
.
.
...read a sales record for the sales person;
   when all records have been read for sales person
   END_OF_SALES = 'Y';
.
.
IF (END_OF_SALES = 'N')
        THEN DO;
          SALES_ITEM = SALESREC_SALES_ITEM;
          SALES_AMOUNT = SALESREC_SALES_AMOUNT;
          SALES_QTY = SALESREC_SALES_QTY;
          SALES_DATE = SALESREC_SALES_DATE;
          PUTSORT;
          IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
        END;

END PUT_SORT_0150;

/*****************************************************************************
*          DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY THE SALES PERSON.      *
*****************************************************************************

 DISPLAY_SALES_DATA_0200: PROC;

DO DISPLAY_COUNT = 1 TO 20 BY 1
        UNTIL (END_OF_DISPLAY - 'Y');
         CALL GET_SALES_ITEM_0220;
END;

IF (END_OF_DISPLAY = 'Y')
        THEN
          ...message in map = 'NO MORE ITEMS FOR SALES PERSON';
ELSE
          ...message in map = 'MORE ITEMS FOLLOW FOR SALES PERSON';
.
.
...display map
.
.

 END DISPLAY_SALES_DATA_0200;

 GET_SALES_ITEM_0220: PROC;
```

```
/*
**      CA IDMS/DC SORT keeps its currency within the sorted file between
**      pseudo converses, therefore, no repositioning is required.
*/
GETSORT NEXT INTO SALES_DATA;
IF (TPSRETN = '0000')
        THEN DO;
          ...map = SALES_ITEM_NAME;
          ...map = SALES_AMOUNT;
          ...map = SALES_QTY;
          ...map = SALES_DATE;
        END;
ELSE
          IF (TPSRETN = '7020')
          THEN
              END_OF_DISPLAY = 'Y';
          ELSE
              CALL SORT_ERROR_9999;

 END GET_SALES_ITEM_0220;

/*****************************************************************************
*          END CURRENT SESSION OF CA IDMS/DC SORT.                           *
*****************************************************************************

 END_SORT_9000: PROC;

ENDSORT;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;

 END END_SORT_9000;

/*****************************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT      *
*          IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE  *
*          FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.              *
*****************************************************************************

 SORT_ERROR_9999: PROC;

...message line in map = TPSMSG;
```

```
.
...display map;
.
...return to CA IDMS/DC;

 END SORT_ERROR_9999;

 END TPSEXPL4;
```

Exhibit 4.16: PLI Pseudo Conversational, USER Option--PLI

```
!      ************************************************************************
!       *     THIS ADS EXAMPLE IS THE SAME AS EXAMPLE 3, EXCEPT A "USER" SORT    *
!       *     HAS BEEN SPECIFIED INSTEAD OF A "PROGRAM" SORT.     THE DIALOGUE IS*
!       *     PSEUDO CONVERSATIONAL AND CAN SORT ANY OR ALL OF THE SALES DATA    *
!       *     FIELDS IN EITHER ASCENDING OR DESCENDING ORDER AT THE USER'S       *
!       *     DISCRETION AT EXECUTION TIME.                                      *
!      ************************************************************************
IF      ...first time
        DO.
              ...set first time off
              CALL GET-SORTED-SALES-DATA.
        END.

MOVE 'N' TO END-OF-DISPLAY.
WHILE (DISPLAY-COUNT < 21) AND
         (END-OF-DISPLAY NE 'Y')
REPEAT.
!       ************************************************************************
!       *     CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE        *
!       *     BETWEEN PSEUDO CONVERSES; THEREFORE, NO REPOSITIONING IS         *
!       *     REQUIRED.                                                      *
!       ************************************************************************
GETSORT NEXT.
IF TPSRETN EQUAL ZERO
        DO.
```

```
                MOVE SALES-ITEM-NAME TO MAP-SALES-ITEM-NAME  (DISPLAY-COUNT).
                MOVE SALES-AMOUNT    TO MAP-SALES-AMOUNT     (DISPLAY-COUNT).
                MOVE SALES-QTY       TO MAP-SALES-QTY        (DISPLAY-COUNT).
                MOVE SALES-DATE      TO MAP-SALES-DATE       (DISPLAY-COUNT).
                ADD 1 TO DISPLAY-COUNT.
        END.
ELSE
        IF TPSRETN EQUAL '7020'
              MOVE 'Y' TO END-OF-DISPLAY.
        ELSE
              CALL SORT-ERROR.
END.

IF END-OF-DISPLAY = 'Y'
        DO.
              ENDSORT.
              IF TPSRETN NE ZERO
                    DO.
                       CALL ERROR.
                    END.
              DISPLAY MESSAGE TEXT 'NO MORE ITEMS FOR SALES PERSON'.
        END.
ELSE
        DISPLAY CONTINUE
              MESSAGE TEXT 'MORE ITEMS FOLLOW FOR SALES PERSON'.

!       ************************************************************************
!       *GET-SORTED-SALES-DATA                                                 *
!       *                                                                      *
!       *     SORT ORDER WILL BE CONTROLLED BY THE USER.     THE USER CAN SELECT*
!       *     ANY OR ALL OF THE FIELDS IN THE SALES-DATA WORK RECORD AS A      *
!       *     SORT KEY. EACH SELECTED SORT KEY CAN BE ORDERED EITHER IN     *
!!      *     ASCENDING OR DESCENDING SEQUENCE.                                *
!       ************************************************************************
DEFINE SUBROUTINE GET-SORTED-SALES-DATA.

SETSORT USER IDMS FOR SALES-DATA
IF TPSRETN NE ZERO
        CALL ERROR.

MOVE 'N' TO END-OF-SALES
WHILE END-OF-SALES NE 'Y'
```

```
REPEAT.
  .
  .
  .
  ...obtain a sales record for the sales person,
  when all records have been processed for this sales
  person, move 'y' to END-OF-SALES.
  .
  .
  IF END-OF-SALES - 'N'
        DO.
              MOVE SALESREC-SALES-ITEM     TO SALES-ITEM.
              MOVE SALESREC-SALES-AMOUNT   TO SALES-AMOUNT.
              MOVE SALESREC-SALES-QTY      TO SALES-QTY.
              MOVE SALESREC-SALES-DATE     TO SALES-DATE.
              PUTSORT.
              IF TPSRETN NE ZERO
                    CALL ERROR.
        END.
END.
GOBACK.


!       ************************************************************************
!       *SORT-ERROR                                                            *
!       *                                                                      *
!       *     AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT     *
!       *     IN THE TPSRETN FIELD. TPMSG CONTAINS A 79 CHARACTER MESSAGE  *
!       *     FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.             *
!       ************************************************************************
DEFINE SUBROUTINE ERROR.

DISPLAY MESSAGE TEXT TPSMSG.
!       RETURN TO TOP.
GOBACK.
```

Exhibit 4.17: ADS Pseudo Conversational, USER Option--ADS

Selecting Sort Criteria on a User Screen

When you specify USER in the SETSORT statement, at processing time CA IDMS/DC Sort displays a screen for selecting sort criteria. The screen shows the fields in the record specified in the SETSORT statement. You can make entries for 1 to 16 fields in the columns headed Sequence and Sort Order. To cancel the sort at any time, press PA2.

A sort selection screen is illustrated in Exhibit 4.17 SCR1.

Specifying Sequence and Sort Order

You can easily specify sequence and sort order by following these steps:

1. On the line containing the element that is to be the first sort key, in the sequence column enter 1.
2. On the same line, in the Sort Order column, you must enter either A for ascending or D for descending.
3. On the line of the element that is to be the second sort key, in the Sequence column enter 2 and in the Sort Order column enter either A or D.
4. Continue entering sequence and sort order for up to 16 elements. Do not skip any sequence numbers. Any element for which you assign a sequence number must also be assigned a sort order.

   If a record has more elements that can fit on one page, you can page backwards or forwards by using the PF keys indicated at the bottom of the screen.
5. For elements that are not sort keys, leave the Sequence and Sort Order columns blank.
6. When you are finished specifying sequence and sort order, press PF3 to execute the sort.
7. If you correct an error, press ENTER to validate the corrections before you execute the sort again.

Sample Sort Selection Screen

A sample Sort Selection screen is shown in Exhibit 3.18 SCR1. Here are descriptions of the fields on the screen:

- **SORT KEY DESCRIPTION**—Names of the fields or elements in the record.
- **SEQUENCE**—Column in which you can enter a number from 01 to 16 to indicate the sequence in which the fields are to be sorted.
- **SORT ORDER**—Column in which you enter A for ascending or D for descending for each element given a sequence number.
- **PF Keys**—List of PF key assignments.

```
CA IDMS/DC SORT Rnn.nn ——  USER SORT KEY SPECIFICATION —— hh:mm:ss  mm/dd/yy

Sort Key Description                Sequence         Sort Order
SORT-NAME
SORT-LANGUAGE




   HELP: PF1-(Expand Error Message)
CONTROL: ENTER-(Validate Screen) PA2-(Cancel Sort) PF3-(Execute Sort)
 PAGING: PF6-(Page First) PF7-(Page Prior) PF8-(Page Next)
```

Exhibit 4.18: SCR1--Sample Sort Selection Screen

Field Error

When an invalid value is entered in one of the columns on the Sort Selection Screen, a "field error" occurs. Such an error is shown in Exhibit 4.19 SCR2, below. In the detail line for SORT-LANGUAGE, the user entered a sequence value outside the range 1-16.

- **Generalized Error Message**—A generalized error message appears in the message area, line 2 of the screen, indicating that one or more field errors have been detected.
- **Specific Error Message, Short Form**—At the same time, short forms of more specific error messages appear next to the items in error. When necessary, these messages can be expanded to provide further information. The next example illustrates how to expand a short form error message.

```
CA IDMS/DC SORT  Rnn.nn  —— USER SORT KEY SPECIFICATION —— hh:mm:ss mm/dd/yy
TPU7066E-ONE OR MORE DETAIL FIELDS ARE IN ERROR
Sort Key Description                Sequence         Sort Order
SORT-NAME                             01                A
SORT-LANGUAGE                         31 7057-BAD SEQ #   A



   HELP: PF1-(Expand Error Message)
CONTROL: ENTER-(Validate Screen) PA2-(Cancel Sort) PF3-(Execute Sort)
 PAGING: PF6-(Page First) PF7-(Page Prior) PF8-(Page Next)
```

Exhibit 4.19: SCR2 Sort Selection Screen--Field Related Error

Expanding Short Form Field Error Messages

To expand a short form field error message and obtain more information, follow these steps:

1. **Position the cursor** on the detail line item that precedes the short form message to be expanded.
2. **Press the PF1 key.**
3. **Look at the Message Area** (line 2 of the screen). It now contains the long form of the message in the detail line, including the message code (first eight characters).

If the long form of the message still does not provide enough information, use the message code to look up a detailed explanation in Messages (see page 139).

```
CA IDMS/DC SORT  Rnn.nn  —— USER SORT KEY SPECIFICATION —— hh:mm:ss mm/dd/yy
TPU7057E-SEQUENCE NUMBER MUST BE BETWEEN 1 AND 16
Sort Key Description                Sequence         Sort Order
SORT-NAME                             01                 A
SORT-LANGUAGE                         31 7057-BAD SEQ #  A




   HELP: PF1-(Expand Error Message)
CONTROL: ENTER-(Validate Screen) PA2-(Cancel Sort) PF3-(Execute Sort)
 PAGING: PF6-(Page First) PF7-(Page Prior) PF8-(Page Next)
```

Exhibit 4.20: SCR3 Sort Selection Screen--Expanding Short Error Messages

Processing Errors

Processing errors can occur when the USER option is specified in the SETSORT statement. A screen illustrating a processing error message is shown in Exhibit 4.21 SCR4.

- **Message code and text.** For a detailed explanation of the message, see Messages (see page 139).

```
CA IDMS/DC SORT  Rnn.nn  —— USER SORT KEY SPECIFICATION —— hh:mm:ss mm/dd/yy
TPU7061E-AN IMPROPER PFKEY WAS PRESSED
Sort Key Description                Sequence         Sort Order
FIELD_1                               01                 A
FIELD_2




   HELP: PF1-(Expand Error Message)
CONTROL: ENTER-(Validate Screen) PA2-(Cancel Sort) PF3-(Execute Sort)
 PAGING: PF6-(Page First) PF7-(Page Prior) PF8-(Page Next)
```

Exhibit 4.21: SRC4 Sort Selection Screen--Processing Error

Chapter 5: Operations

This section contains the following topics:

Overview (see page 95)
Operational Considerations (see page 95)
CA IDMS/DC Sort System Flow (see page 96)
System Limits (see page 98)
Storage Requirements (see page 98)
COBOL/Assembler/PLI (see page 98)
CA ADS (see page 100)
Demonstration (see page 104)
Customizing CA IDMS/DC Sort (see page 104)

Overview

This chapter describes operational procedures for CA IDMS/DC Sort. It begins with operational considerations, system flow, and system limits. Next are the steps (including model JCL) necessary to use CA IDMS/DC Sort with COBOL, Assembler, or PLI applications. Finally, this chapter discusses tuning CA IDMS/DC Sort for your environment. For information on using CA IDMS/DC Sort with CA ADS, see CA ADS Preprocessor (see page 109).

Operational Considerations

When you first install CA IDMS/DC Sort, note the following:

- The limits in effect for CA IDMS/DC Sort are described under System Limits (see page 98).
- CA IDMS/DC Sort has a macro, TPSPARM, with which you can adjust the size of the sort buffers and the amounts of main and auxiliary storage available to hold the sort buffers. You can find more information about this macro under Customizing CA IDMS/DC Sort (see page 104).

CA IDMS/DC Sort System Flow

CA IDMS/DC Sort provides preprocessors for use with COBOL, Assembler, PLI, and CA ADS. The preprocessors use SETSORT, PUTSORT, GETSORT, ENDSORT, and SETLIMIT statements to generate required programming logic for CA IDMS/DC Sort.

A diagram of system flow is shown in Exhibit 5.1. The application program, including CA IDMS/DC Sort parameter statements, is fed into the CA IDMS/DC Sort precompiler. Then the program is compiled and linked.

At execution time, if the PROGRAM option was selected in the SETSORT statement, the program issues calls to CA IDMS/DC Sort. Then the sorts are done by CA IDMS/DC Sort, using main and auxiliary sort-work areas as necessary. After the sorts are completed, the results are displayed as directed by the application.

If the USER option was selected in the SETSORT statement, before the sorting is done CA IDMS/DC Sort presents the user with a sequence selection screen, where the user can designate up to 16 sort keys and the sort order (ascending or descending) for each key.

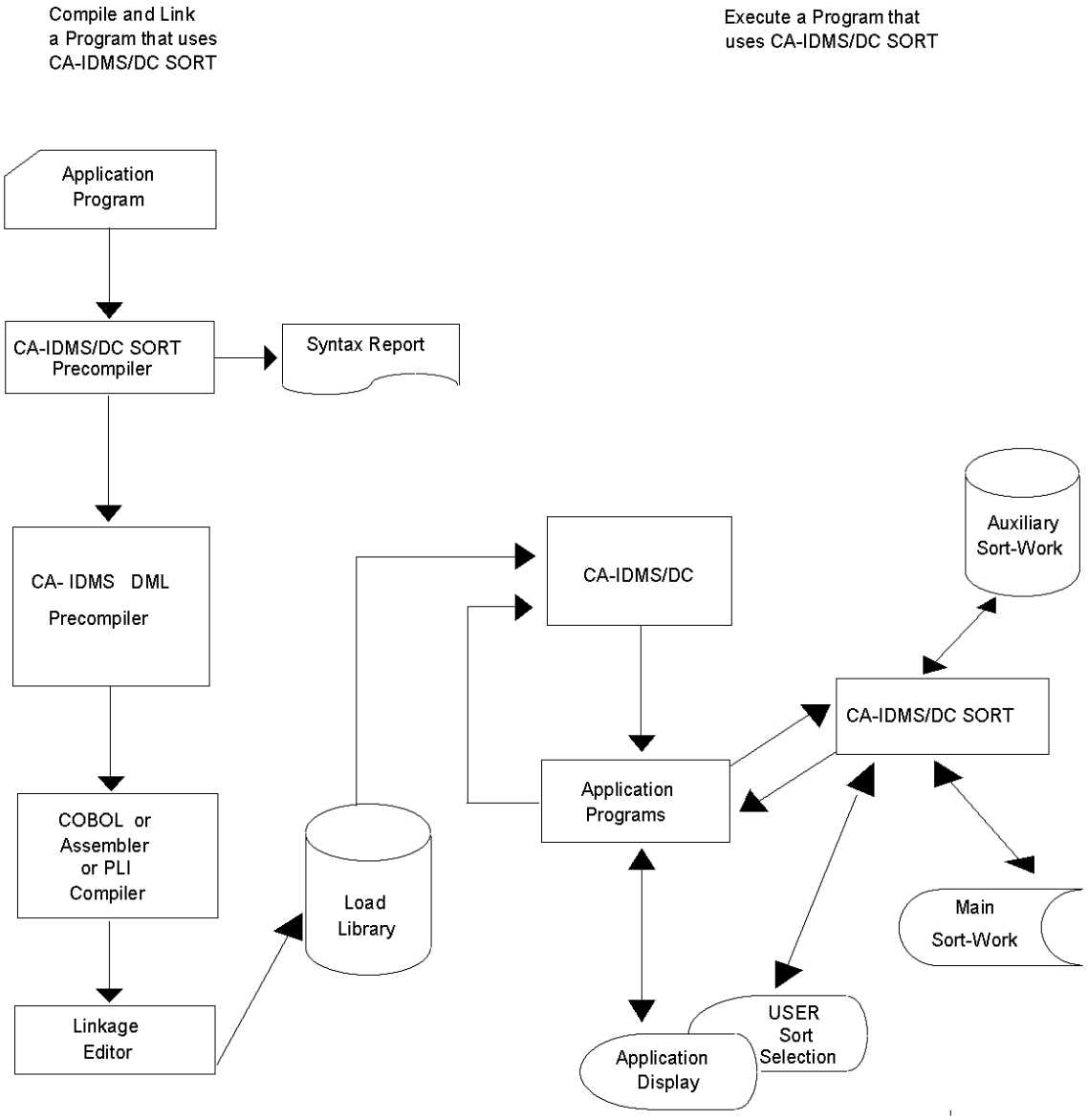
Compile and Link
a Program that uses
CA-IDMS/DC SORT
Execute a Program that
uses CA-IDMS/DC SORT
Application
Program
CA-IDMS/DC SORT
Precompiler
Syntax Report
CA- IDMS DML
Precompiler
COBOL or
Assembler
or PLI
Compiler
Linkage
Editor
Load
Library
CA-IDMS/DC
Application
Programs
Auxiliary
Sort-Work
CA-IDMS/DC SORT
Main
Sort-Work
USER
Sort
Selection
Application
Display

Exhibit 5.1: CA IDMS/DC Sort System Flow

System Limits

The following limits are in effect for CA IDMS/DC Sort:

- Record size can be no greater than 32000 bytes for CA IDMS and 16000 bytes for CICS.
- Element size can be no greater than 256 bytes.
- Sort buffer size can be no greater than 32000 bytes for CA IDMS and 16000 bytes for CICS (See MINRBUF parameter in this chapter.)
- The maximum number of sort keys is 16.
- Up to ten sessions may operate concurrently at one terminal.
- All sorting is performed on a binary basis, in EBCDIC collating sequence.

Storage Requirements

For each sort session CA IDMS/DC Sort requires main and auxiliary storage.

- The MAIN storage default at installation time is 10000 bytes, maximum.
- The AUX (auxiliary) storage default at installation is 10000 bytes, maximum.

Online Program Storage

- For COBOL and Assembler, runtime program storage requires a maximum of 38K.
- For CA ADS, the preprocessor requires a maximum of 147K (including the EDITOR).

Runtime program storage for CA ADS requires a maximum of 43K.

COBOL/Assembler/PLI

The steps required to use CA IDMS/DC Sort with COBOL, Assembler, or PLI programs are listed below. Sample application programs are shown in Examples (see page 35).

1. Add to your program the statements necessary to accomplish these tasks:
   a. Copy the appropriate CA IDMS/DC Sort control block.

      COBOL: TPSCOMMC

      Assembler: TPSCOMMA

      PLI: TPSCOMMP

b. Initiate the sort and establish criteria (SETSORT).

   Optionally, issue a SETLIMIT to alter the runtime environment.

   - Pass records to CA IDMS/DC Sort (PUTSORT).
   - Retrieve sorted records (GETSORT).
   - Terminate a session (ENDSORT).

     The details of using these four parameter statements are described in Parameters (see page 17).

c. Check the return code (TPSRETN). Do not ignore a non-zero return code. After a GETSORT, the content of the sorted record is unpredictable if the return code is non-zero.

d. Issue error messages when appropriate (TPSMSG).

2. Execute the preprocessor as shown in Exhibit 5.2 (z/OS), Exhibit 5.5 (Z/VSE), or Exhibit 5.8 (Z/VM).

   Note: CA IDMS 16.0 supports z/OS V2R10 as well as z/OS 1.1 and above. However, we refer to z/OS in this document.

3. Compile your program.

4. Link your program as shown in Exhibit 5.3 (z/OS, COBOL or Assembler), Exhibit 5.4 (z/OS, PLI), Exhibit 5.6 (Z/VSE, COBOL or Assembler), or Exhibit 5.7 (Z/VSE, PLI).

5. Execute your program.

6. If USER was specified in the SETSORT statement, respond to the user screens. These screens are illustrated in Parameters (see page 17).

CA ADS

To use the preprocessor with CA ADS, see Parameters (see page 17).

```
*************************************************************************
*       MODEL JCL FOR EXECUTION OF CA IDMS/DC SORT PREPROCESSORS FOR     *
*       ASSEMBLER, COBOL, AND PLI PROGRAMS IN CA IDMS/DC                *
*       ENVIRONMENT.                                                    *
*                                                                       *
*       CHANGE PGM STATEMENT TO EXECUTE ONE OF THE FOLLOWING:           *
*                                                                       *
*                TPSBCOBI  --  COBOL, CA IDMS/DC                        *
*                TPSBASMI  --  ASSEMBLER, CA IDMS/DC                    *
*                TPSBPLII  --  PLI, CA IDMS/DC                          *
*                TPSBADSI  --  ADS, CA IDMS/DC                          *
*                                                                       *
*        SYSCTL DD STATEMENT MAY, OPTIONALLY, BE USED FOR               *
*        CA IDMS DICTIONARY USAGE.                                      *
*************************************************************************

//TPSBXXXX      EXEC          PGM=TPSBXXXX
//STEPLIB       DD            DSN=your.tpsort.loadlib,DISP=SHR
//              DD            DSN=your.idms.loadlib,DISP=SHR
//SYSCTL        DD            DSN=your.idms.sysctl,DISP=SHR
//AUDIT         DD            SYSOUT=a
//INPUT         DD            DSN=your.source.code,DISP=SHR
//OUTPUT        DD            DSN=your.expanded.code,DISP=SHR
```

Exhibit 5.2: Modelz/OS JCL for Execution of CA IDMS/DC Sort

```
*************************************************************************
*       Sample Z/OS link-edit control cards for inclusion of             *
*       CA IDMS/DC SORT with Assembler or COBOL programs.                *
*************************************************************************

INCLUDE       SYSLIB(CICS module)       * Include CICS command-level language
INCLUDE       SYSLIB(your program)      * interface module.
INCLUDE       SYSLIB(TPSET)             * Include CA IDMS modules as
INCLUDE       SYSLIB(IDMS module)       * appropriate.
              .
              .
ENTRY         DFHEI1                    * Include for CA IDMS/DC programs.

NAME          your program(R)
```

Exhibit 5.3: Sample Z/OS Link Edit Control Statements Assembler or COBOL

```
*************************************************************************
*        Sample Z/OS link-edit control cards for inclusion of           *
*        CA IDMS/DC SORT with PLI programs.                             *
*************************************************************************

INCLUDE        SYSLIB(your program)       *
INCLUDE        SYSLIB(TPSORT)             *
INCLUDE        SYSLIB(TPSET)              * Enter these control statements
INCLUDE        SSSYSLIB(IDMS)             * for CA IDMS.
ENTRY          PLISTART                   *
NAME           your program(R)            *
```

Exhibit 5.4: Sample Z/OS Link Edit Control Statements--PLI

```
// OPTION PARTDUMP
// UPSI   00000001
*  **** PRIVATE CORE IMAGE LIBRARY WHERE TP/SORT INSTALLED
// DLBL    DBMS,'your.loadlib'
// EXTENT  ,volser
*  FOR DOS/SP USE THE FOLLOWING:
// LIBDEF  PHASE,SEARCH=(DBMS.sublibrary,IDMS.sublibrary)
*  FOR Z/VSE USE THE FOLLOWING:
// LIBDEF  CL,SEARCH=(DBMS,IDMS)
*
*  ********************** REPORT FILE ******************************
*
// ASSGN   SYS013,SYSLST
*
*  ******* SOURCE CODE INPUT TO PREPROCESSOR *************************
*
// ASSGN   SYS014,SYSRDR                        SYNTAX FILE
*
*  ******* PREPROCESSED SOURCE CODE FROM PREPROCESSOR *****************
*  ******* USED AS INPUT BY NEXT JOB STEP             *****************
*
// DLBL    OUTPUT,'work.file.output',0,SD
// EXTENT         SYS015,volser,,,00250,003
// ASSGN          SYS015,DISK,VOL=volser,SHR
*
// EXEC    TPSBXXXX,SIZE=(TPSBXXXX,400K)
PLACE LANGUAGE SOURCE CODE HERE
/*
/&
SS EOJ
```

Exhibit 5.5: Model Z/VSE JCL for Execution of CA IDMS/DC Sort

```
*************************************************************************
*        Sample Z/VSE link-edit control cards for inclusion of          *
*        CA IDMS/DC SORT with Assembler or COBOL programs.              *
*************************************************************************
 PHASE your-program,*
 INCLUDE CICS-module                  * Include CICS command level
 INCLUDE your-program                 * language interface module.
 INCLUDE TPSETC
 INCLUDE DFHEAI
 INCLUDE DFHEAI0
 INCLUDE IDMS-module                  * Include CA IDMS modules as
 ENTRY   DFHEI1                       * as appropriate.
```

Exhibit 5.6: Sample Z/VSE Link Edit Control Statements Assembler or COBOL

```
*************************************************************************
*        Sample Z/VSE link-edit control cards for inclusion of          *
*        CA IDMS/DC SORT with PL/I programs.                            *
*************************************************************************
 PHASE your-program,*
 INCLUDE your-program
 INCLUDE TPSETI
 INCLUDE IDMS
 ENTRY PLISTART
```

Exhibit 5.7: Sample Z/VSE Link Edit Control Statements--PLI

```
/* TPSBEXEC                                                          */
/*                                                                   */
/* KEY TO VARIABLES                                                  */
/* ----------------                                                  */
/*                                                                   */
/* ca.loadlib         The filename of the load library into which    */
/*                    you downloaded CA IDMS Tools.                  */
/*                                                                   */
/* idms.loadlib       The filename of the load library containing    */
/*                    your IDMS SUBSCHEMA and DMCL modules.          */
/*                                                                   */
/* tpsprogn           Change to: TPSBASMI for IDMS/DC-Assembler      */
/*                               TPSBCOBI for IDMS/DC-COBOL          */
/*                               TPSBPLII for IDMS/DC-PLI            */
/*  */
TRACE OFF; SIGNAL ON ERROR
/*                                                                   */
```

```
CA_LOADLIB_FN       = 'ca.loadlib'
IDMS_LOADLIB_FN     = 'idms.loadlib'
INPUT_FN            = 'input-source-fn'
INPUT_FT            = 'input-source-ft'
INPUT_FM            = '*'
OUTPUT_FN           = 'output-source-fn'
OUTPUT_FT           = 'output-source-ft'
OUTPUT_FM           = '*'
/*                                                                  */
/*  LINK AND ACCESS MINIDISKS CONTAINING REQUIRED LIBRARIES         */
/*                                                                  */
 'CP SPOOL PRINTER NOCONT CLOSE'
 'CP SPOOL PRINTER TO * NOHOLD CONT FORM OFF DIST OFF'
 'GLOBAL  LOADLIB ' CA_LOADLIB_FN IDMS_LOADLIB_FN
 'FILEDEF AUDIT   PRINTER'
 'FILEDEF INPUT   DISK ' INPUT_FN INPUT_FT INPUT_FM
 'FILEDEF OUTPUT  DISK ' OUTPUT_FN OUTPUT_FT OUTPUT_FM
 SIGNAL OFF ERROR
 SAY 'STARTING CA IDMS/DC SORT PREPROCESSOR'
 'EXECOS OSRUN tpsprogn'
 TPSBXEC_RC = RC
 'CP SPOOL PRINTER NOCONT'
 'CP CLOSE PRINTER NAME TPSBEXEC LISTING'
 'CP SPOOL PRINTER OFF'
 SAY 'TPSBEXEC FINISHED WITH A RETURN CODE OF' TPSBXEC_RC
 'GLOBAL LOADLIB'
 'FILEDEF * CLEAR'
 EXIT TPSBXEC_RC
/*                                                                  */
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
ERROR:
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
 ERROR_RC = RC
 TRACE OFF; SIGNAL OFF ERROR
 SAY 'NON-ZERO RETURN CODE ENCOUNTERED IN EXEC AT LINE' SIGL
'CP SPOOL PRINTER NOCONT'
'CP CLOSE PRINTER NAME TPSBEXEC LISTING'
'CP SPOOL PRINTER OFF'
 'GLOBAL LOADLIB'
 'FILEDEF * CLEAR'
 EXIT ERROR_RC
/*                                                                  */
```

Exhibit 5.8: Z/VM EXEC for CA IDMS/DC Sort Execution — Asm, COBOL, PLI

Demonstration

The CA IDMS tape contains a member that has information on how to run demonstrations. The member was downloaded to your source library during installation.

CA IDMS Environment

Read the instructions in TPSDEMO to find out how to run these CA IDMS demonstrations:

1. Test of CA IDMS/DC Sort runtime facilities. If the task TPS1 is specified, 50 random records are sorted on a key selected on the USER screen.
2. CA ADS prototype. An intentional syntax error is embedded in a SETSORT statement, in order to invoke the EDITOR.

CICS Environment

Read the instructions in TPSDEMOC to find out how to run the CICS demonstration:

- Test of CA IDMS/DC Sort run-time facilities, including the USER screen. After you select a sort field and sort order, a display of 50 random records should appear in the proper sequence.

Customizing CA IDMS/DC Sort

CA IDMS/DC Sort provides a customization macro that gives you the ability to:

- Specify the amount of main storage and auxiliary storage to be made available to CA IDMS/DC Sort.
- Indicate how space is to be allocated to buffers at runtime. The allocation of buffers also depends on the record length in a particular sort.
- Indicate whether or not developers are allowed to alter MAIN, AUX, or MINRBUF at runtime. It does this by enabling or disabling the SETLIMIT parameter.
- Indicate the CA IDMS/DC Sort CA ADS Preprocessor termination key.

These runtime options can be changed at anytime after initial product installation, either before or after SMP/E ACCEPT processing. See the CA IDMS installation guides for detailed instructions on processing customization macro changes under SMP/E. Additional customization considerations and examples for CA IDMS/DC Sort are shown below.

Customization Considerations

At installation time, the MAIN and AUX parameters are each assigned a value of 10000 bytes, unless you changed the default values during the SMP/E installation process. During each sort session in an application, CA IDMS/DC Sort acquires the main and auxiliary storage as necessary, up to the value assigned. (A session is defined by the session number in a SETSORT statement.) If you want to run the most efficient sorts possible, you may want to consider the following points:

The **most efficient** sort is one in which

- There are many small records in a buffer
- All of the buffers reside in main storage

To **increase efficiency** in a given sort session, use a work record that contains only the fields necessary for sorting. With only those fields, the work record is as small as possible to meet the requirements.

In an ideal situation,

- Main storage is slightly larger than the space needed for an average sort
- Auxiliary storage adds the extra space needed for large sorts

Increasing the proportion of auxiliary storage to main storage may affect response time.

Sample CA IDMS/DC Sort Customization

At execution time, CA IDMS/DC Sort allocates sort buffers in multiples of 2000 bytes. To determine the size of a sort buffer:

1. Multiply the MINRBUF value times the record size.
2. Round the result up to the next multiple of 2000 bytes.
3. Add 12 bytes for CA IDMS/DC Sort overhead.

Maximum: Sort buffer size can be no greater than 32K.

Note: CA IDMS/DC Sort will not split a buffer between main and auxiliary storage. Therefore it is necessary to make efficient use of main and auxiliary storage.

The product of the MINRBUF value and the record length cannot exceed either the MAIN value or the AUX value, whichever is larger, because there would not be enough space to store one sort buffer.

In the following four examples, The MAIN and AUX parameters are not changed. The default for each is 10000 bytes.

Example 1

```
MINRBUF=20
record-length=100
```

The sort buffer used by CA IDMS/DC Sort will be 2012 bytes:

```
20 * 100 = 2000
2000 is a multiple of 2000
2000 + 12 = 2012
```

CA IDMS/DC Sort can store four sort buffers (80 records) in main storage and four sort buffers (80 records) in auxiliary storage.

Example 2

```
MINRBUF=20
record-length=150
```

The sort buffer used by CA IDMS/DC Sort will be 4012 bytes:

```
20 * 150 = 3000
The next multiple of 2000 is 4000
4000 + 12 = 4012
```

CA IDMS/DC Sort can store two sort buffers (40 records) in main storage and two sort buffers (40 records) in auxiliary storage.

Example 3

```
MINRBUF=100  (default)
record-length=31
```

The sort buffer used by CA IDMS/DC Sort will be 4012 bytes:

```
31 * 100 = 3100
next multiple of 2000 is 4000
sort buffer is 4012
```

CA IDMS/DC Sort can store two sort buffers (200 records) in main storage and two sort buffers (200 records) in auxiliary storage.

Example 4

```
MINRBUF=100  (default)
record-length=51
```

The sort buffer used by CA IDMS/DC Sort will be 6012 bytes:

```
51 * 100 = 5100
next multiple of 2000 is 6000
sort buffer is 6012
```

CA IDMS/DC Sort can store one sort buffer (100 records) in main storage and one sort buffer (100 records) in auxiliary storage.

Chapter 6: CA ADS Preprocessor

This section contains the following topics:

Overview (see page 109)
CA ADS Preprocessor (see page 109)
Preprocess Multiple Modules (see page 111)
System Flow (see page 112)
EDITOR (see page 115)
Editing Commands (see page 115)
Primary Commands (see page 121)
Scroll Options (see page 129)
Line Commands (see page 130)
Key Settings (see page 137)

Overview

This chapter describes the steps necessary to use CA IDMS/DC Sort with CA ADS. It also describes how to use the EDITOR to correct syntax errors encountered by the CA ADS Preprocessor in CA IDMS/DC Sort.

CA ADS Preprocessor

To use the CA IDMS/DC Sort preprocessor for CA ADS, follow these four steps:

Step 1—Add CA IDMS/DC Sort Statements to Modules

1. Add the necessary SETSORT, PUTSORT, GETSORT, ENDSORT and SETLIMIT statements to the appropriate dictionary modules. See Chapter 3, Examples for sample dialogs.
2. Add a statement to check the return code (TPSRETN).
3. Add statements to issue error messages when appropriate.

Step 2—Execute the Preprocessor for Each Module

For each dictionary module containing CA IDMS/DC Sort syntax, execute the CA ADS preprocessor of CA IDMS/DC Sort (TPSG), as shown in Exhibit 6.1.

If an error is detected in the user-supplied parameters, CA IDMS/DC Sort automatically invokes an online EDITOR (see page 115) which allows you to correct syntax errors.

If you have installed the CA IDMS DME, the module source can be preprocessed at any time during a CA IDMS DME session by entering *TPSG* on the command line.

Note: For more information, see the *CA IDMS Dictionary Module Editor User Guide*.

If you are using CA IDMS Dictionary Migrator to move module source to another dictionary, the CA ADS Batch preprocessor, TPSBADSI, can be used to preprocess the DDDLUPD file.

To execute the preprocessor, enter the task code for CA IDMS/DC Sort (TPSG) on the system prompt screen. The system will display a screen on which you can enter parameters identifying the module. The screen is illustrated in Exhibit 6.1. If you use the PA2 key to cancel the preprocessor, only the current TPSG command is affected.

```
     CCCCCCCCC
   CCCCCCCCCCC
 CCC                                    CA IDMS/DC SORT
CCC      AAAA                         Syntax Preprocessor
CCC     AAAAA
CCC    AAAAAA                               Rnn.nn
CCC   AAA AAA
 CCC AAA  AAA
  CCAAACCCCCC
   AAACCCCCCCC
  AAA     AAA
 AAA      AAA
AAA       AAA    Module Name ===>

              Module Version ===> 0001              (Optional)

                 Dictionary ===>                    (Optional)

                       Node ===>                    (Optional)

   PA2   Terminate Preprocessor
   TPP7030E-MINIMUM ENTRY OF MODULE NAME IS REQUIRED
       Copyright (C)  2003 CA International, Inc.
```

Exhibit 6.1: CA ADS Module Preprocessor Screen

Another method of executing the preprocessor is to enter all of the preprocessor parameters when invoking the preprocessor:

```
TPSG  module-name  module-version  alternate-dictionary  node
```

where:

module-name

indicates the name of the module.

module-version

indicates the version number of the module.

alternate-dictionary

indicates the dictionary in which the module resides.

node

indicates the DDS node in which the alternate dictionary is found.

Each of these parameters is positional, with each default represented by a comma (,). A single space delimiter between parameters is required.

Example

```
V10 ENTER NEXT TASK CODE:
TPSG TEST-MOD , DICTB
```

Step 3—Execute the Dialog Generator

Execute the dialog compiler, ADSC, for the dialog containing the altered modules. Specify TPSCOMM as a work record.

Step 4—Execute the Dialog

Execute the dialog.

Preprocess Multiple Modules

CA IDMS users can create a CLIST consisting of a number of CA IDMS/DC Sort preprocessor task statements and the CA ADS compiler task statement. Once the CLIST is set up, you need execute only one instruction instead of several.

The following example illustrates the creation and execution of a CLIST to preprocess multiple CA ADS modules.

Example

Create a CLIST to preprocess multiple ADS modules and invoke the ADSC compiler.

```
ADD MODULE PRESORT VER 1 LANG DC
MODULE SOURCE FOLLOWS
TPSG MODULE-1 , DICTB
TPSG MODULE-4 2 DICTB
TPSG MODULE-6 99 DICTB
ADSC
MSEND.
```

Execute the CLIST.

```
V10 ENTER NEXT TASK CODE:
CLIST PRESORT
```

In this way, each of the modules containing CA IDMS/DC Sort syntax will be preprocessed prior to executing the CA ADS Compiler, ADSC.

System Flow

The preprocessor uses SETSORT, PUTSORT, GETSORT, and ENDSORT statements to generate required programming logic for CA IDMS/DC Sort.

A diagram of system flow is shown in Exhibit 6.2. The dialogs are placed in the dictionary (IDD), and from there read into the CA IDMS/DC Sort precompiler. If there are errors, the CA EDITOR is automatically invoked and you can make syntax corrections, replace the module in the dictionary, and re-execute the preprocessor. Then the dialogs are generated. After they have been preprocessed, CA IDMS/DC Sort statements appear as comments.

At execution time, if the PROGRAM option was selected in the SETSORT statement, the dialogs issue calls to CA IDMS/DC Sort. Then the sorts are done by CA IDMS/DC Sort, using main and auxiliary sort-work areas as necessary. After the sorts are completed, the results are displayed as directed by the application.

If the USER option was selected in the SETSORT statement, before the sorting is done CA IDMS/DC Sort presents the user with a sequence selection screen, where the user can designate up to 16 sort keys and the sort order (ascending or descending) for each key.

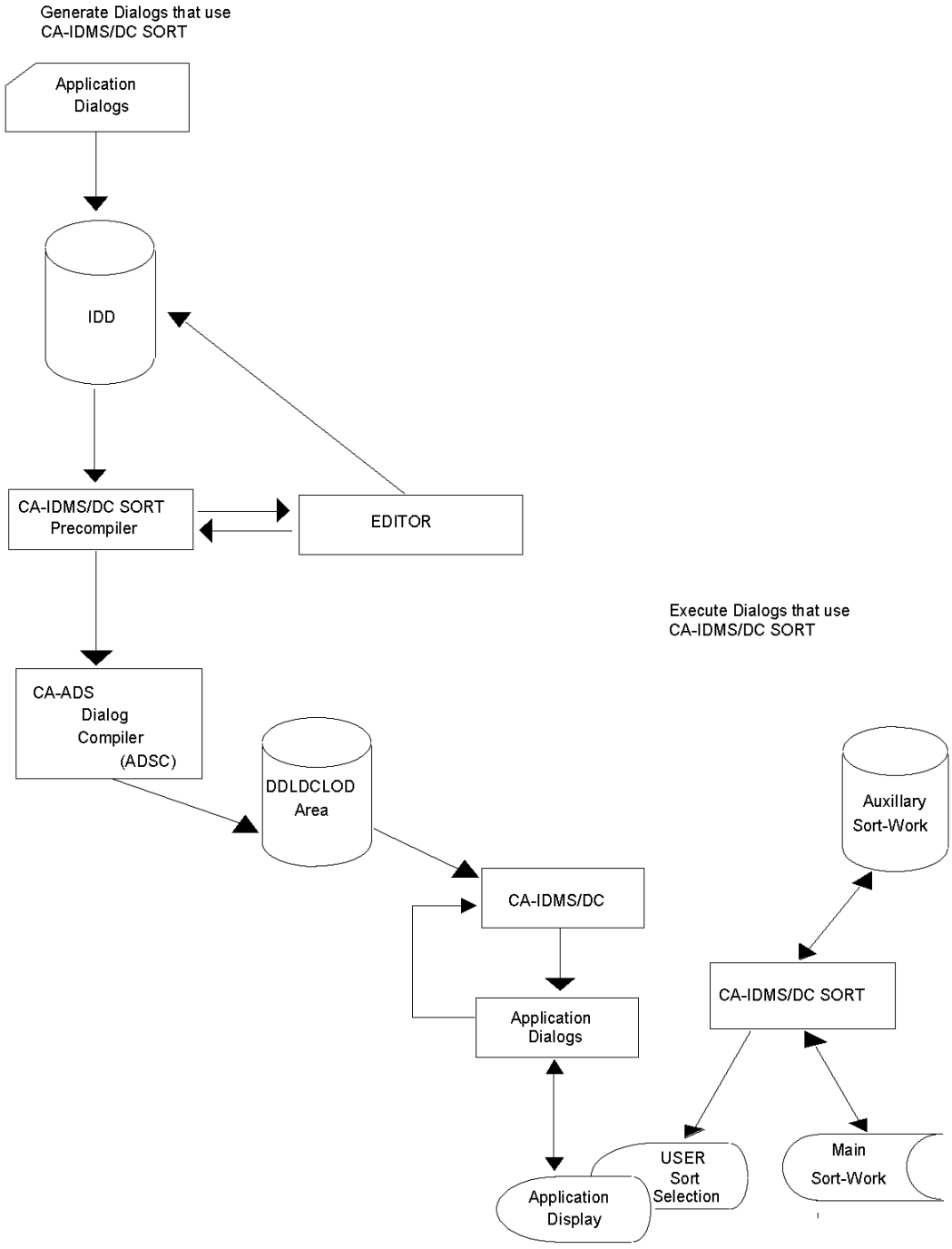

Exhibit 6.2: CA ADS System Flow

EDITOR

If the CA IDMS/DC Sort CA ADS Preprocessor detects sort control or formatting errors then the CA EDITOR is invoked during the preprocess step.

The EDITOR is conveniently invoked directly from the preprocessor. When a syntax error is detected, the EDITOR displays the module source with highlighted error lines describing the problem. The programmer can correct the syntax and RETRY the preprocessor. If the syntax is correct, it is replaced in the dictionary and processing continues.

If syntax errors remain, the EDITOR is reinvoked until the syntax is correct or the user cancels the preprocessor session.

Editing Commands

Editing commands are entered from the Edit Screen (see Exhibit 5.3). These are the three types of editing commands:

- Scroll Options
- Primary Commands
- Line Commands

Scroll Options

Scroll options are used to determine how many lines of the module to scroll up or down when using a Primary Command or a PF key.

Primary Commands

Primary Commands are used to:

- Scroll up, down, right or left through a module.
- Locate the desired line of a module.
- Find the next occurrence of a string.

- Change the next occurrence of a string.
- Reset the screen display to remove all Line Commands, column markers, and extraneous messages.
- Cancel changes made (with the editor) to a module.
- End/Retry the edit session. That is, save any changes made and reexecute the preprocessor.
- Replace blank characters at the end of a field with null characters.
- Turn the CAPS mode on or off.
- Redefine the PF key functions.

Line Commands

Line Commands are used to:

- Move source lines within the module. Specify the location at which source lines are to be copied or moved.
- Repeat source lines in the module.
- Delete source lines.
- Insert blank source lines.
- Display a line with column markings across the screen.

Program Function Keys

PF keys are set to many frequently-used Primary Commands. Also, the PA1, PA2 and CLEAR keys are set to redisplay the screen.

Entering Commands

Here are the descriptions of where commands are entered on the Edit Screen (see Exhibit 6.3).

- **Primary Commands**--Enter these commands at the left side of the second line, after the word COMMAND. This field is called the Command Line.
- **Scroll Options**--Enter these options at the far-right side of the second line on the Edit Screen, after the word SCROLL.
- **Line Commands**--Enter these commands with the cursor positioned in the line number fields.

After you key in a command, press the ENTER key to execute the command. You *can* key in more than one command before pressing the ENTER key to execute the commands.

PF keys are set for most Primary Commands. This allows you to enter a command from any position on the Edit Screen with one keystroke.

To execute the command set for a PF key, you simply press the key. (Settings are covered at the end of this section.) You *can* key in commands, then press a PF key. The PF key function and the commands will be executed when you press the PF key.

```
EDIT ——TIME-ENTER                                          COLUMNS 001 072
COMMAND ===>                                                SCROLL ===> PAGE
****** *** TOP OF DATA ********CA IDMS/DC SORT*********************************
I      !SETSORT IDMS.
==MSG> TPP7033E-INVALID WORD IDMS IN STATEMENT/WORD 0002
000003 OBTAIN FIRST ROOM-RECORD WITHIN LIBRARY-AREA2.
000004 !PUTSORT.
000005 OBTAIN NEXT ROOM-RECORD WITHIN LIBRARY-AREA2.
****** *** BOTTOM OF DATA ******CA IDMS/DC SORT*********************************
```

Exhibit 6.3: Where to Enter Commands on the Edit Screen

Conventions, Rules and Syntax

Be sure to review these exhibits before you begin your first edit session:

- **Exhibit 6.4** — Notation Conventions
- **Exhibit 6.5** — Command Syntax Rules
- **Exhibit 6.6** — Command Syntax Summary
- **Exhibit 6.7** — Command Syntax Summary

Scroll options, Primary Commands, Line Commands and Key Settings are covered in detail in the rest of this section.

Functions

The command and key functions are summarized in these exhibits:

- **Exhibit 6.8** — Summary of Primary Commands
- **Exhibit 6.9** — Summary of Scroll Options
- **Exhibit 6.10** — Summary of Line Commands
- **Exhibit 6.11** — Summary of Key Settings

Example	Function
FIND string	Variables appear in lowercase. you substitute an appropriate value for each variable.
UP [number-of-lines]	Brackets indicate optional clauses.
/ UP \ < DOWN > \ /	Braces enclose two or more options. You select one of them.

Exhibit 6.4: Notation Conventions

Item	Rule
Order of Commands	You must enter a B (before) or an A (after) line command in conjunction with the C (copy) and M (move) line commands to indicate where to copy or move the lines.
Entering Blanks in Commands	Blanks (character spaces) are ignored in line command sequences, so you can enter blanks between a command and a value without affecting processing. You must enter at least one blank (character space) between a Primary Command and a primary command value field. You cannot embed blanks in a keyword.

Exhibit 6.5: Command Syntax Rules

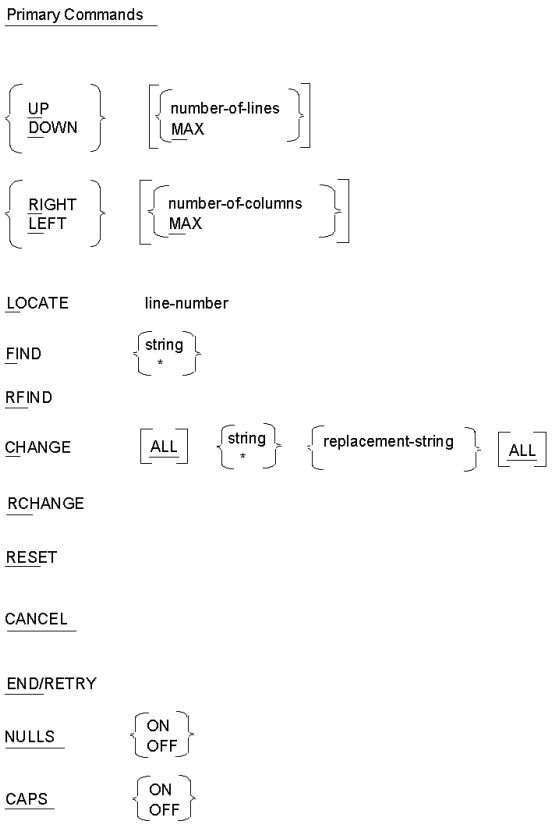

Exhibit 6.6: Command Syntax Summary

Command Syntax Summary

Scroll Options

```
/ PAGE          \
< HALF          >
| CSR           |
\ number-of-lines /
```

Line Commands

```
/ B \
< A > number-of-times
\   /

  ┌/               \┐
C |< number-of-lines >|
  └\ C             /┘

  ┌/               \┐
M |< number-of-lines >|
  └\ M             /┘

  ┌/                 \┐
R |< number-of-times   >|
  └\ R number-of-times /┘

  ┌/               \┐
D |< number-of-lines >|
  └\ D             /┘

I    number-of-lines

COLS
```

Key Settings

```
PF1/PF13    HELP   (currently unavailable)
PF2/PF14    END/RETRY
PF3/PF15    END/RETRY
PF4/PF16    CANCEL
PF5/PF17    RFIND
PF6/PF18    RCHANGE
PF7/PF19    UP
PF8/PF20    DOWN
PF9/PF21    RESET
PF10/PF22   LEFT
PF11/PF23   RIGHT
PF12/PF24   ENTER
PA1         RESHOW
PA2         RESHOW
CLEAR       CLEAR
```

Primary Commands

Primary Commands are entered on the second line of the Edit Screen after the word COMMAND (see Exhibit 5.1). Exhibit 5.8 summarizes the functions of the Primary Commands. You can enter more than one primary command at a time. Use the following syntax:

```
command ; command
```

Note: Blanks are not a valid option; ensure there are no blank spaces in the command line.

UP and DOWN Commands

The UP (scroll up) and DOWN (scroll down) commands are used to display source lines above or below your current view. The amount you scroll is determined by the Scroll Option setting. The setting can be overridden at any time. To override the existing setting, use the following syntax:

```
/ UP   \ / number-oflines \
< DOWN > < MAX            >
\      / \                /
```

Where:

number-of-lines specifies that you want to scroll up or down a specific number of lines.

MAX specifies that you want to scroll to the first or last full page of text.

Here are the three basic formats you can use:

```
/ UP   \
< DOWN >
\      /
```

Use either UP or DOWN alone to scroll the current scroll setting.

```
/ UP   \
< DOWN >  number-of-lines
\      /
```

Use UP or DOWN with a number to override the current scroll setting.

```
/ UP   \
< DOWN >  MAX
\      /
```

Use UP or DOWN with MAX to scroll to either the beginning (UP) or the end (DOWN) of the module.

Default UP keys: PF7, PF19

Default DOWN keys: PF8, PF20.

RIGHT and LEFT Commands

The RIGHT (scroll right) and the LEFT (scroll left) commands are used to display source lines to the right or left of your current view. The amount you scroll is determined by the scroll option setting. The setting can be overridden at any time. To override the existing setting, use the following syntax:

```
/ RIGHT \ / number-oflines \
< LEFT  > < MAX            >
\       / \                /
```

where:

number of columns specifies that you want to scroll a specific number of columns to the right or to the left.

MAX specifies that you want to scroll to either the far right or far left of the text.

Here are the three basic formats you can use:

```
/ RIGHT \
< LEFT  >
\       /
```

Use RIGHT or LEFT alone to scroll the current scroll option setting.

```
/ RIGHT \
< LEFT  >  number-of-columns
\       /
```

Use RIGHT or LEFT with a number to override the current scroll setting.

```
/ RIGHT \
< LEFT  >  MAX
\       /
```

Use RIGHT or LEFT with MAX to scroll to either the far right or far left of the module.

Default RIGHT keys: PF11, PF23

Default LEFT keys: PF10 or PF22.

LOCATE Command

Use the LOCATE command to scroll the display to a specific source line or to the beginning or the end of the module. The syntax is:

```
LOCATE  line-number
```

where:

line-number specifies the number of the line to which you want to scroll. The line you specify will be the top line displayed for the module.

How to Use the LOCATE Command

To scroll to a specific line, you specify the line number of the line you want displayed.

To scroll to the beginning of the module, you can specify 0 as a line number, and the first line of the module will be the top line displayed.

To scroll to the end of the module, you can specify the last line number or any larger number, and the last line of the module will be the top line displayed. For example, if the last line of the module is numbered 307 and you use 999, line number 307 will be the top line displayed.

FIND and RFIND Commands

Use the FIND command or the RFIND (repeat find) command to search for a string in the module.

The editor begins searching at the position of the cursor when you ENTER the command, and it searches downward until the string is found. If the cursor is on the Command Line when you ENTER the command, the editor begins searching at the top line displayed.

The cursor appears at the beginning of the string, if the string is found. Here is the FIND command syntax:

```
FIND  string
```

Rules for FIND Command

1. If the string has embedded blanks, enclose the string in either single or double quotation marks. For example:

```
 FIND
program name'
 FIND "program name"
```

2. If the string consists of a single asterisk (*), enclose the asterisk in quotation marks. A double asterisk does not require any quotation marks. For example:

```
 FIND
*'
 FIND **
```

3. Quotation marks are optional for all other strings. If you decide to enclose a string which contains a *single* quotation mark, enclose the string with double quotation marks. If you decide to enclose a string which contains *double* quotation marks, enclose the string with single quotation marks. For example:

```
FIND "They're"
```

The RFIND command repeats the last FIND command that was entered. Here is the syntax:

*RF***IND**

Default RFIND keys: PF5, PF17.

If the cursor is on the top line when you enter the RFIND command, the editor will search the entire file. When it reaches the end of the file after several finds, the message line will state BOTTOM OF DATA REACHED. Entering RFIND again will return you to the top of the file. If, however, the string does not exist in the file when you enter the RFIND command, the message line will state NO CHARACTERS "string" FOUND. Entering the RFIND command again has no effect.

To selectively change strings, use the RFIND PF key in conjunction with the RCHANGE PF key. See the next subsection on CHANGE and RCHANGE commands for a description.

CHANGE and RCHANGE Commands

Use the CHANGE command or the RCHANGE (repeat change) command to search for and change the next occurrences of a string in the module.

The editor begins searching at the position of the cursor when you ENTER the command, and it searches downward until the string is found. If the cursor is on the Command Line when you ENTER the command, the editor begins searching at the top line displayed.

If the string is found, it is changed to the replacement string. Here is the CHANGE command syntax:

```
            /       \ /                   \
CHANGE [ALL] < string > < replacement-string > [ALL]
            \ *     / \ *                 /
```

where:

ALL specifies that the string is to be replaced with the replacement string throughout the module (from beginning to end). ALL can be used in either position shown above. It can be used only once in the CHANGE command.

string specifies the string to be found and replaced.

*** (asterisk)**, when used as a string value, specifies the string value from the last FIND or CHANGE command entered. When an asterisk is used as a replacement string value, it specifies the replacement string value from the last CHANGE command entered.

replacement-string specifies the string to replace the first string specified.

The rules regarding the CHANGE and RCHANGE commands are on the following pages.

Rules for CHANGE Command

1. If a string has embedded blanks, enclose the string in either single or double quotation marks. For example:

```
 CHANGE
program name'
program name'
 CHANGE "program name" "program name"
```

2. If the string consists of a single asterisk (*), enclose the asterisk in single quotation marks. A double asterisk does not require any quotation marks. For example:

```
 CHANGE
*'
comments'
 CHANGE **
comments'
```

3. If a string is the word ALL, enclose the string in quotation marks.
4. Quotation marks are optional for all other strings. If you enclose a string which contains a *single* quotation mark, enclose the string with double quotation marks. If you enclose a string which contains *double* quotation marks, enclose the string with single quotation marks. For example:

```
CHANGE "They're" "He's"
```

The RCHANGE command repeats the last CHANGE command that was entered. The syntax is:

*RC***HANGE**

Default keys: PF6, PF18.

Using the RCHANGE and RFIND PF keys to Selectively Change Strings

You can use the RFIND PF key in conjunction with the RCHANGE PF key to selectively change strings. Here are two sample sequences:

First Sample Sequence

1. CHANGE WORK-NAME-1 WORK-NAME-2

   ENTER key

2. RFIND key
3. RCHANGE key
4. RFIND key

   RFIND key

In Step 1, you ENTER the CHANGE command to change the next occurrence of WORK-NAME-1 to WORK-NAME-2.

In Step 2, you want to find the next occurrence of WORK-NAME-1, but you are not sure if you will want to change the string. By pressing the RFIND key, the occurrence of WORK-NAME-1, which was specified in the CHANGE command during Step 1, will be found.

In Step 3, you want to change the occurrence of WORK-NAME-1 that was found during Step 2 to WORK-NAME-2. By pressing the RCHANGE key, the occurrence will be changed.

In Step 4, you press the RFIND key to find the next occurrence of WORK-NAME-1. This time you do not want to change the string, so instead of pressing the RCHANGE key, you press the RFIND key again. The next occurrence of WORD-NAME-1 will be found.

Second Sample Sequence

1. CHANGE WORK-NAME-1 WORK-NAME-2

   RFIND key

2. RFIND key
3. RCHANGE key

In Step 1, you want to find the next occurrence of WORK-NAME-1, but you are not sure if you will want to change it to WORK-NAME-2. By keying in the CHANGE command and pressing the RFIND key instead of the ENTER key, the RFIND will be executed. The next occurrence of WORK-NAME-1, which was specified in the CHANGE command, will be found.

In Step 2, you decide that you do not want to change the string that was found during Step 1, so you press the RFIND key. The next occurrence of WORK-NAME-1 will be found.

In Step 3, you want to change the string that was found during Step 2, so you press the RCHANGE key. This changes WORK-NAME-1 to WORK-NAME-2.

RESET Command

The RESET command clears the module display of all Line Commands, column markers, and extraneous messages--even those commands, markers and messages that are not currently displayed on the screen. Here is the syntax:

*RES***ET**

Default keys: PF9, PF21.

CANCEL Command

Use the CANCEL command to cancel all changes made to the module since the last END/RETRY and to exit from the Edit Screen (return to the Specification Screen). Here is the syntax:

CANCEL

Default keys: PF4, PF16

END/RETRY Command

The END/RETRY command saves the module and returns to the preprocessor for another syntax conversion attempt. This command saves the module, as currently modified, by updating the module in the dictionary. The syntax is:

END/**RETRY**

Default keys: PF2, PF3, PF14, PF15.

The END/RETRY function is performed only if the module has actually been changed. If the module has not been changed, END/RETRY simply refreshes the screen.

NULLS Command

The NULLS command is used to replace blank characters with null characters at the end of a field. The null characters replace all but the first blank in a field. If the field is completely blank, the blank characters do not change to null.

Turn the NULLS mode on or off by using the following syntax:

```
      / ON  \
NULLS < OFF >
      \     /
```

The NULLS ON mode is the default.

One practical application is to combine the NULLS ON command with the INSERT mode. This combination enables you to insert additional characters in the middle of a field. Each character you insert causes the rest of the field to move right.

An alternative to the NULLS command is to use either the DELETE or the ERASE EOF key. If you use the DELETE key, all the characters to the left of the deleted character will move left. A null character will automatically be inserted at the end of the field. If you use the ERASE EOF key, null characters will automatically be inserted from the point of erasure to the end of the field.

CAPS Command

The CAPS command is used to turn the CAPS mode on and off. With the CAPS mode on, all new alpha data is translated into upper case. With the CAPS mode off, the data remains unaffected. Data that was initially entered with the CAPS mode off will remain in lower case unless you edit the field. To override the existing mode, use the following syntax:

```
     / ON  \
CAPS < OFF >
     \     /
```

The CAPS ON mode is the default.

Command	Function
UP	Scrolls to display lines that are above your current view.
DOWN	Scrolls to display lines that are below your current view.
RIGHT	Scrolls to display lines that are right of your current view.
LEFT	Scrolls to display lines that are left of your current view.

Command	Function
LOCATE	Scrolls to a specific line or to the start or end of the module.
FIND	Finds the next occurrence of a string.
RFIND	Repeats the last FIND command entered.
CHANGE	Finds and changes the next occurrence of a string.
RCHANGE	Repeats the last CHANGE command entered.
RESET	Removes Line Commands, column markers, extraneous messages.
CANCEL	Cancels changes made to the module and exit the Edit Screen.
END/RETRY	Saves the module and reinvokes the preprocessor.
NULLS	Replaces blanks at the end of a field with nulls.
CAPS	Turns the CAPS mode on or off.

Exhibit 6.7: Summary of Primary Commands

Scroll Options

Scroll Options are used to specify how much of the screen is scrolled when you use either the UP or DOWN Primary Command (or corresponding PF key) by itself. Exhibit 5.9 summarizes the functions of these options.

At the far-right side of the second line on the Edit Screen, the word SCROLL appears, followed by one of the Scroll Options (see Exhibit 5.1). To change the current setting, key in one of the other options over the current setting. The Scroll Option you set will remain in effect until you enter a different setting. The syntax is:

```
/ PAGE            \
< HALF            >
| CSR             |
\ number-of-lines /
```

where:

***P*AGE** specifies that a whole screen is to be scrolled whenever the UP or DOWN command is used.

***H*ALF** specifies that a half screen is to be scrolled whenever the UP or DOWN command is used.

*C***SR** specifies that the line with the cursor on it is to become the bottom line displayed whenever the UP command is used or the top line whenever the DOWN command is used.

number-of-lines specifies that this number of lines is to be scrolled whenever the UP or DOWN command is used. The number you enter can be one to three digits in length.

Default value: CSR.

Option	Function
PAGE	Set scroll amount for a full screen at a time.
HALF	Set scroll amount for a half screen at a time.
CSR	Set scroll amount for all lines above or below the cursor.
number-of-lines	Set scroll amount for the specified number of lines.

Exhibit 6.8: Summary of Scroll Options

Line Commands

Line Commands are entered with the cursor positioned to the left of the source lines, in the line number fields (see Exhibit 5.1). To use a Line Command, you key over the line numbers. Exhibit 6.10 summarizes the functions of the Line Commands.

Command	Function
B (before)	Specifies that lines being copied or moved are to be placed before this source line.
A (after)	Specifies that lines being copied or moved are to be placed after this source line.
C (copy)	Copies source lines within the module. Blanks are not a valid option; ensure the command line contains no blank spaces.
M (move)	Moves source lines within the module. Blanks are not a valid option; ensure the command line contains no blank spaces.
R (repeat)	Repeats source lines directly after themselves in the module. Blanks are not a valid option; ensure the command line contains no blank spaces.
D (delete)	Deletes source lines. Blanks are not a valid option; ensure the command line contains no blank spaces.
I (insert)	Inserts blank source lines.

Command	Function
COLS (columns)	Displays a line with column markings.

Exhibit 6.9: Summary of Line Commands

Entering Line Commands

When you enter a line command, the editor looks for the right-most value that you changed in the line number field, and it looks at the position of the cursor. For example, here is how the line number field looks before and after specifying with the R (repeat) command that line 3 is to be repeated 12 times.

Before entering R command 000003 *After* entering R command R12003

In this example, the editor sees that the right-most changed value is the 2, and everything to the left of and including the 2 is read as the line command (R12).

If you wanted to repeat the line ten times instead of 12 times, here is how the line number field would appear:

Before entering R command 000003 *After* entering R command R10*0*03

For the editor to read the command as R10 (and not, for example, R1 or R100), the cursor must be positioned immediately after R10 in the line number field (at the position underlined above) when you press the ENTER key.

If you want to press the ENTER key with the cursor positioned elsewhere on the screen, you can key in a blank character space after the R10 command. The line number field would look like this:

After entering R command R10 03

Or, you could press the ERASE EOF (erase end of field) key after keying in the R10.

Note that the value you substitute for a numeric variable in a line command can be up to five digits in length.

B (before) and A (after) Commands

The B (before) and A (after) commands are used in conjunction with the C (copy) and M (move) line commands. Use either the B command or the A command in the line number field to indicate the location before or after the source line where lines are to be copied or moved. The syntax of the two commands is:

```
/ B \
< A >  [number-of-lines]
\   /
```

where:

number-of-lines specifies that you want a number of copies of the line(s) or the module copied or moved.

Note: Blanks are not a valid option; ensure that the command line contains no blanks.

C (copy) Command

The C (copy) line command is used in conjunction with either the B (before) command or the A (after) command to copy lines from one location to another within a module. The C command indicates the line(s) to be copied. The B or A command indicates the location to which the lines are to be copied (see preceding description of B and A commands for more information). Here is the syntax of the command:

```
  ┌/ number-of-lines \┐
C |< C              >|
  └\                 /┘
```

Note: Blanks are not a valid option; ensure that the command line contains no blanks.

How to Use the C (copy) Command

Using the C command, you can specify:

- A single line to be copied (that line marked).
- A number of lines to be copied (the first line marked).
- A block of lines to be copied (the first and last lines marked).

Here is the syntax for copying a single line:

C

The syntax for copying a specified number of lines is:

*C*number-of-lines

To copy a block of lines, mark both the first line and the last line of the block with:

CC

Rules for C (copy) Command

1. When using the Cnumber-of-lines or the CC form of the command, you cannot enter any other commands on the lines being copied.
2. A CC must be paired with another CC.
3. You must pair a B (before) or an A (after) command with every C or pair of CC commands.

M (move) Command

The M (move) command is used in conjunction with either the B (before) command or the A (after) command to move lines from one location to another within a module. The M command indicates the line(s) to be moved. The A or B command indicates the location to which lines are to be moved (see preceding description of B and A commands for more information). Here is the syntax of the command:

```
   ┌/ number-of-lines \┐
M  |< M               >|
   └\                 /┘
```

Note: Blanks are not a valid option; ensure that the command line contains no blanks.

How to Use the M (move) Command

Using the M command, you can specify:

- A single line to be moved (that line marked).
- A number of lines to be moved (the first line marked).
- A block of lines to be moved (the first and last lines marked).

Here is the syntax for moving a single line:

M

The syntax for moving a specified number of lines is:

*M*number-of-lines

To move a block of lines, mark both the first line and the last line of the block with:

MM

Rules for M (move) Command

1. When using the Mnumber-of-lines or the MM form of the command, you cannot enter any other commands on the lines being moved.
2. An MM must be paired with another MM.
3. You must pair a B (before) or an A (after) command with every M or pair of MM commands.

R (repeat) Command

Use the R (repeat) command to repeat lines directly after themselves. You do *not* use the B and A commands with the R command. Here is the syntax of the R command:

```
   ┌/ number-of-times    \┐
R  |< R [number-of-times >|
   └\                   /┘
```

Note: Blanks are not a valid option; ensure that the command line contains no blanks.

How to Use the R (repeat) Command

You can specify:

- A single line to be repeated (that line marked).
- A single line to be repeated a number of times (that line marked).
- A block of lines to be repeated (the first and last lines marked).
- A block of lines to be repeated a number of times (the first and last lines marked).

Here is the syntax for repeating a single line:

R

The syntax for repeating a single line a specified number of times is:

*R*number-of-times

To repeat a block of lines once, mark both the first line and the last line of the block with:

RR

To repeat a block of lines a number of times, mark both the first line and the last line of the block with:

*RR*number-of-times

You can enter the number of times the block is to be repeated on either the first line or the last line of the block. If you specify a number of times on both lines, the number on the last line will be used.

Rules for R (repeat) Command

- When using the Rnumber-of-times, the RR, or the RRnumber-of-times form of the command, you cannot enter any other commands on the lines being repeated.
- An RR must be paired with another.

D (delete) Command

The D (delete) command is used to delete source lines from the module. Here is the syntax:

```
   ┌/ number-of-lines \┐
D  |< D               >|
   └\                 /┘
```

Note: Blanks are not a valid option; ensure that the command line contains no blanks.

How to Use the D (delete) Command

Using the D command, you can specify:

- A single line to be deleted (that line marked).
- A number of lines to be deleted (the first line marked).
- A block of lines to be deleted (the first and last lines marked).

Here is the syntax for deleting a single line:

D

The syntax for deleting a specified number of lines is:

*D*number-of-lines

To delete a block of lines, mark both the first line and the last line of the block with:

DD

Rules for D (delete) Command

- When using the Dnumber-of-lines or the DD form of the command, you cannot enter any other commands on the lines being deleted.
- A DD must be paired with another DD.

I (insert) Command

Use the I (insert) command to insert blank source lines in the module. You do *not* use the B and A commands with the I command. After inserting blank lines, you can key in source code to be added to the module on the blank lines. If you do not key in anything on an inserted line, that line will be deleted the next time you press the ENTER key or a PF key. Here is the syntax:

I **[number-of-lines]**

where:

I used alone specifies that a single blank line is to be inserted after the source line at which you ENTER the command.

number-of-lines specifies that this number of blank lines are to be inserted after the source line at which you ENTER the command.

COLS (columns) Command

The COLS (columns) command displays a line with column markings for your reference. The line is an aid for editing. It is not given a line number, and it is not written to the dictionary when you use the END/RETRY command. The line will be displayed before the source line at which you ENTER the COLS command. Here is the syntax:

*COL***S**

Key Settings

PF keys are set for most Primary Commands. This allows you to enter a Primary Command from any position on the screen (not from just the Command Line). PF keys also reduce keystrokes. You do *not* need to press the ENTER key after pressing a PF key. Exhibit 6.11 summarizes the functions of the key settings.

ENTER Function

The PF keys set for the ENTER function work in the same way as the ENTER key: they execute Primary or Line Commands.

RESHOW Function

The RESHOW function on the PA1, PA2 and CLEAR keys cannot be performed with a Primary Command. Use RESHOW to view the last screen display.

Key	Command	Function
PF1/13	HELP	(currently unavailable).
PF2/14 PF3/15	END/RETRY	Saves the module and reinvokes the preprocessor.
PF4/16	CANCEL	Saves the module and exits the Edit Screen.
PF5/17	RFIND	Repeats the last FIND command.
PF6/18	RCHANGE	Repeats the last CHANGE command.
PF7/19	UP	Scrolls to display the lines above your current view.
PF8/20	DOWN	Scrolls to display the lines below your current view.
PF9/21	RESET	Removes Line Commands, column markers and extraneous messages.
PF10/22	LEFT	Scrolls to display the lines to the left of your current view.
PF11/23	RIGHT	Scrolls to display the lines to the right of your current view.
PF12/24	ENTER	Functions as the ENTER key.
PA1/PA2 CLEAR	RESHOW	Redisplays previous screen.

Exhibit 6.10: Summary of Key Settings

Chapter 7: Messages

This section contains the following topics:

Overview (see page 139)
Non-Standard Message Codes--Batch (see page 140)
Non-Standard Message Codes--Batch or Online (see page 143)

Overview

This chapter lists all messages generated by CA IDMS/DC Sort. Included are the unique error message numbers, reasons for the each error's occurrence, and suggestions for appropriate remedial actions.

Each message generated by CA IDMS/DC Sort is preceded by a unique eight-character code. The code is in the format TP*xnnnns* where *x* indicates the message type; *nnnn* is the actual message number; and *s* is the severity code for the message.

E (execution)—A message type code that accompanies online runtime messages. These messages are invoked when the application program is invoking CA IDMS/DC Sort. Error messages are displayed within the application program only if the appropriate error routine is coded into the program. See Chapter 3, Examples for sample programs.

The message number *nnnn* is returned in TPSRETN and the message code is in TPSMSG in the communication block.

These three return codes supply information to the application program and do not necessarily indicate error conditions: 7019, 7020, 7055.

P (preprocessor)—A message type code that accompanies messages generated by the preprocessor (the preprocessor translates CA IDMS/DC Sort syntax into CALL statements). These messages are found on the Syntax Report only.

U (user)—These message type codes are issued when the USER option has been specified in the SETSORT statement.

Informative—A severity code ending with the letter I indicates an informative message. Informative messages need no remedial action.

Error—A severity code ending with the letter E indicates an error. Error messages provide more information about problems which have caused processing to terminate.

Non-Standard Message Codes--Batch

The message codes associated with the following four messages do not conform to the standard described on the first page of this chapter. However, these messages may appear in reports generated by CA IDMS/DC Sort batch preprocessors.

FILE905E

GSSFILE RETURNED AN ERROR DURING file-function, FILE= file-name, CODES n1, n2, n3, n4

Reason:

The file handler is unable to perform the file function with the indicated file.

Action:

See Exhibit 7.1 for an explanation and appropriate action for the return codes indicated.

GSFL999I

file-id IS NOT VSAM - WILL TRY QSAM

Reason:

In Z/VSE, the indicated file is not a VSAM file. The message is preceded by a system message indicating an open error for a VSAM file.

Action:

None. If the attempt to open the file for QSAM processing is successful, CA IDMS/DC Sort will continue with normal processing.

IDMS0001E

PROGRAM program-name ABORTED WITH STATUS OF idms-status

Reason:

A non-zero return code was encountered in a CA IDMS/DB call.

Action:

See the CA IDMS Messages and Codes Guide.

SGEN001I

SUCCESSFUL PREPROCESS FOR LANGUAGE language-name ENVIRONMENT CICS/IDMS

Reason:

The preprocessor was successfully executed.

Action:

None.

Two types of errors can be reported by the return codes of n1, n2, n3, and n4--non-VSAM file errors and VSAM file errors. The error is described by n2 and n4. For VSAM file errors, n4 is always equal to 28. The error is described by n1, n2, and n3. A general return code is given by n4 for both non-VSAM and VSAM errors. All return codes are decimal values.

n4	Reason	Action
4	End-of-file	Call Product Support.
8	Open error or file is not open	Look for JCL errors or for the use of improper files.
12	An I/O error has occurred	Find cause for I/O error.
16	Request not recognized	Call Product Support.
20	File was already opened	Call Product Support.
24	Parameter list error	Call Product Support.
28	VSAM error n1=R15 return code from VSAM n2=low order byte from R0 GENCB/MODCB type of error n3=VSAM feedback byte error in I/O request	Use n1, n2, and n3 to check for possible user errors. If there are no user errors, call Product Support.
32	Insufficient storage	Increase storage for job step.
36	SYNAD error occurred n1=byte 1 of DECB n2=byte 2 of DECB N3=byte 3 of DECB	For BDAM files.
40	BPAM FIND error n1=R15 n2=R0	Use n1 and n2 (as described in Data Managment Macro Instructions) to check for errors.

n4	Reason	Action
44	BPAM STOW error n1=R15 n2=0	Use n1 and n2 (as described in Data Management Macro Instructions) to check for errors.
n2	Reason	Action
0	n4=8, use of unopened file n4=24, parameter list error	Call Product Support. Call Product Support.
1	JCL/label overrode parm list	Remove DCB information from JCL and ensure that the correct files are referenced.
2	Parm list overrode JCL/label	Remove DCB information from JCL and ensure that the correct files are referenced.
3	Unrecognized request	Call Product Support.
4	Z/OS x13 ABEND trapped at open	Fix cause for x13 ABEND.
5	Tried to update seq. file	Call Product Support.
6	VSAM write at other than load	Call Product Support.
7	SOS table could not expand	Call Product Support.
8	Z/OS DCB open failed	Call Product Support.
9	SOS table buffer pointer lost	Call Product Support.
10	SOS table file CB not built	Call Product Support.
11	Z/OS DD statement Missing	Supply missing DD statement.
12	VSAM ACB open failed	Call Product Support.
13	Record format invalid	Call Product Support.
14	Macro format invalid	Call Product Support.
15	Record length not numeric	Call Product Support.
16	Record length too large	Call Product Support.
17	Block size not numeric	Call Product Support.
18	Block size too large	Call Product Support.
19	Invalid Z/VSE sysname table	Assemble a valid sysname table.
20	Z/VSE sysname table entry missing	Assemble a sysname table with an entry for the missing one.
21	Z/VSE LU number too large	Use an LU number within range.

n4	Reason	Action
22	Z/VSE sysname is not numeric or is misspelled	Correct to a valid sysname.
23	Z/VSE sysname blank	Do not use blank sysname.
24	Z/VSE LU not assigned	Call Product Support.
25	Z/VSE DTF prototype missing	Call Product Support.
26	Z/VSE logic module missing	Generate missing logic module.
27	Z/VSE CCW mismatch	Call Product Support.
28	File is not a PDS	Allocate file to a PDS.

Exhibit 7.1: Return Codes

Non-Standard Message Codes--Batch or Online

The message code associated with the following message does not conform to the standard described on the first page of this chapter. However, the message may appear in reports generated by CA IDMS/DC Sort batch preprocessors or on CA IDMS/DC Sort screens.

LMSG900E

GSILMSG FAILURE CC = n

Reason:

A severe error occurred during an attempt to format another error message.

Action:

Condition code (CC) **n** indicates the specific problem and the required Action:

- **4**—An attempt was made to generate the message associated with a message code which is not in the message table. Reinstall load module TPSMSGT. If the problem persists, contact CA product support.
- **8**—The message table could not be loaded. Check for a proper sysgen and load module for TPSMSGT.
- **12**—Not enough storage is available for message processing. Check the amount of storage allocated to the TP monitor and increase it, if necessary.
- **16**—Incompatible parameters were passed to the message handler. Ensure that the most recent versions of CA IDMS/DC Sort and GSILMSG are installed. Contact CA product support if the problem persists.

TPE7001E

INVALID PARM LIST FOR process-name - PARM NUMBER = parm-number

Reason:

An invalid parameter sequence was specified during CA IDMS/DC Sort processing, where x represents the TPSPROC value, and y represents the number of the invalid parameter.

Action:

Review the CA IDMS/DC Sort parameters for errors. If the syntax is correct, contact CA product support.

TPE7002E

INVALID TPS-REQUEST VALUE OF request-type

Reason:

CA IDMS/DC Sort found a value other than 'U' or 'S' in TPSRQST.

Action:

Review the CA IDMS/DC Sort parameters for errors. If the syntax is correct, contact CA product support.

TPE7003E

INVALID TPS-ELEMENTS TYPE OF element-type

Reason:

CA IDMS/DC Sort found a value other than 'I', 'C' or 'P' in TPSELEM.

Action:

Review the CA IDMS/DC Sort parameters for errors. If the syntax is correct, contact CA product support.

TPE7004E

REQUEST OF request-type INVALID WITH ELEMENTS OF element-type

Reason:

The value in TPSRQST conflicts with the value in TPSELEM.

Action:

If TPSRQST has a value of 'U', TPSELEM must have a value of 'I' or 'C'. If TPSRQST has a value of 'S', TPSELEM must have a value of 'P' or blank.

Review the CA IDMS/DC Sort parameters for errors. If the syntax is correct, contact CA product support.

TPE7005E

INVALID VALUE FOR PROCESS process-type - PARM NUMBER = parm-number

Reason:

The parameter number is in the format xy. A value in the TPSPROC parameter list, represented by x, is invalid. Y represents the sequential number of the parameter.

Action:

Review the CA IDMS/DC Sort parameters for errors. If the syntax is correct, contact CA product support.

TPE7006E

INVALID PROCESS VALUE OF process-type

Reason:

TPSPROC contains a value other than 'SETSORT', 'PUTSORT', 'GETSORT' or 'ENDSORT'.

Action:

Review the CA IDMS/DC Sort parameters for errors. If the syntax is correct, contact CA product support.

TPE7007E

NO SETSORT PERFORMED FOR SESSION session-number

Reason:

A TPSPROC value of 'PUTSORT', 'GETSORT' or 'ENDSORT' was specified, but no CA IDMS/DC Sort controls were set up for this session.

Action:

Enter the necessary syntax to establish a SETSORT for this session.

TPE7008E

DUPLICATE SETSORTS ISSUED FOR SESSION session-number

Reason:

The indicated sort session contains two SETSORT requests without an intervening ENDSORT within a single task invocation.

Action:

Include an ENDSORT prior to the second SETSORT for this session.

TPE7009E

INVALID NUMBER OF SORT KEYS SPECIFIED FOR SESSION session-number

Reason:

The number of keys is either less than 1 or greater than 16.

Action:

Review the CA IDMS/DC Sort parameters for errors. If the syntax is correct, contact CA product support.

TPE7010E

RECORD LENGTH IS 0 OR BEYOND MAXIMUM FOR SESSION session-number

Reason:

The TPSRLEN field is less than 1 or greater than 32000 for CA IDMS.

Action:

Contact CA product support.

TPE7011E

PUTSORT BUFFER NOT SPECIFIED FOR SESSION session-number

Reason:

In the indicated session-number, a SETSORT request was made without a record-name parameter.

Action:

Review the CA IDMS/DC Sort parameters for errors. If the syntax is correct, contact CA product support.

TPE7012E

INVALID DISPLACEMENT FOUND IN KEY OCCURRENCES FOR SESSION session-number

Reason:

The sort-control record for the indicated session contains incorrect values. An element which is not in the session record has been specified in the FIELDS statement.

Action:

Correct the invalid FIELDS statement.

TPE7013E

KEY LENGTH EXCEEDS RECORD BOUNDARY FOR SESSION session-number

Reason:

The field-length specified for a field-name in a FIELDS statement exceeds the record boundary.

Action:

Correct the invalid FIELDS statement.

TPE7014E

INVALID KEY ORDER OF sort-order FOR SESSION session-number

Reason:

An invalid sort order was specified in the keys section.

Action:

Correct the value to either A or D in the sort keys table.

TPE7015E

MAIN AND AUX EXCEEDED - UNABLE TO CONTINUE

Reason:

In the current sort session, the maximum number of bytes allowed in MAIN and AUX has been exceeded.

Action:

In the user program, check for program loops which may be causing excessive PUT requests. If the program logic is correct, have your system programmer review the MAIN and AUX values in the TPSPARM macro to determine whether they need to be increased. See Operations (see page 95) for more information.

TPE7016E

PUTSORT DISALLOWED AFTER GETSORTS FOR SESSION session-number

Reason:

The applications program has attempted to write another record after one or more GET requests have been issued.

Action:

Either remove the PUTSORT from the program logic, or close and open the session with an ENDSORT/SETSORT sequence.

TPE7017E

STORAGE FAILURE DURING SORT PROCESSING

Reason:

A required storage block allocation failed.

Action:

Retry the application. If this message is frequently issued, review storage pool definitions for your online regions.

TPE7018E

THE RETURN ADDRESS FOR SORTED RECORDS WAS NOT SPECIFIED FOR GETSORT

Reason:

The buffer address of the area into which sorted records are returned has been overlaid.

Action:

Review program logic to ensure that a loop has not overlaid CA IDMS/DC Sort control blocks. If there is no apparent cause for the control block alteration, contact CA product support.

TPE7019E

NO RECORDS WERE SORTED FOR SESSION session-number

Reason:

The sort queue for the indicated session-number was empty. A GETSORT request was issued, but no records were sent to CA IDMS/DC Sort through PUTSORT requests.

Action:

Review your program to determine if the condition is appropriate.

TPE7020E

END OF SET ENCOUNTERED FOR SESSION session-number DURING process-type PROCESS

Reason:

The top or bottom of the sorted queue for the indicated session has been reached. Process-type indicates if the condition occurred during NEXT or PRIOR processing.

Action:

If you wish to take advantage of this condition and execute special processing at the end of the queue, add program logic to trap the TPSRETN value 7020. The content of the sorted record is unpredictable until another GETSORT request is successfully executed.

TPE7040E

INVALID SESSION VALUE OF session-number

Reason:

The session-number specified in the SESSION statement is not an integer between 0 and 9.

Action:

Correct the SESSION statement, and retry the preprocessor.

TPE7041E

MISMATCH ON TPSKNUM AND ACTUAL PARAMETERS FOR SESSION session-number

Reason:

The number in TPSKNUM and the number of parameters in the interface call to TPSET do not agree.

Action:

Contact CA product support.

TPE7044E

SETLIMIT OCCURRED AFTER PUTSORT FOR SESSION n

Reason:

A SETLIMIT statement for session n appears in the program *after* one or more PUTSORT statements for session n. For a given session, SETLIMIT must appear *before* any PUTSORTs.

Action:

Correct the program by moving the SETLIMIT statement for session n to a position after the SESSION statement for session n and before any PUTSORTs for session n.

TPE7045E

SETLIMIT ATTEMPTED, BUT INSTALLATION PROHIBITS USE

Reason:

The program contains a SETLIMIT statement, but your installation prohibits its use.

Action:

Correct the program by removing the SETLIMIT statement. Or, contact your systems programmer to reassemble the tailoring macro TPSPARM. This macro currently specifies LIMLOCK=Y, which prohibits use of the SETLIMIT statement. To allow use of SETLIMIT, TPSPARM must be reassembled with LIMLOCK=N, and CA IDMS/DC Sort must be relinked with the reassembled TPSPARM object deck.

TPP7021E

MODULE module-name NOT FOUND

Reason:

The user has attempted to execute the CA IDMS/DC Sort CA ADS preprocessor. The module name parameter from the datastream or the user display cannot be located in the dictionary/node/version specified.

Action:

Correct module name, dictionary, node, and/or version.

TPP7022E

DATABASE BIND FAILED FOR INDICATED DICT AND NODE--RECHECK THESE VALUES

Reason:

The dictionary and/or node specified to the CA IDMS/DC Sort CA ADS preprocessor does not exist under the current CV.

Action:

Correct dictionary and/or node names.

TPP7023E

INVALID DATA LINE ON TPSG

Reason:

An invalid or missing datastream has been entered as part of the execution of the TPSG task.

Action:

Enter the required information in the screen display.

TPP7024E

INVALID MODULE NAME IN INPUT DATA LINE

Reason:

An invalid module-name format was specified as part of the TPSG task data stream. The module name must be between 1 and 32 alphanumeric, non-space characters.

Action:

Correct the module name in the user display.

TPP7025E

INVALID PARAMETER AFTER DEFAULT VERSION

Reason:

Following the version-number default, the next fields must be a 1- to 8-character dictionary name or dictionary name default (represented by a comma), followed by a 1- to 8-character node name or node name default (represented by a comma).

Action:

Correct dictionary and/or node values in the user display.

TPP7026E

INVALID VERSION NUMBER IN INPUT DATA LINE

Reason:

The parameter after the module name in the TPSG task data stream must be a display integer between 1 and 9999, or the version number default (represented by a comma).

Action:

Enter a valid version number in the user display.

TPP7027E

INVALID ALTERNATE DICTIONARY NAME

Reason:

The dictionary name in the TPSG task datastream is not either a 1- to 8-character alphanumeric field, or the dictionary default (represented by a comma).

Action:

Enter a valid dictionary name in the user display.

TPP7028E

INVALID ALTERNATE NODE NAME

Reason:

The node name in the TPSG task datastream is not either a 1- to 8-character alphanumeric field, or the node default (represented by a comma).

Action:

Enter a valid node name in the user display.

TPP7029E

MODULE TEXT ENDED WITH IMPROPERLY TERMINATED TPSORT SYNTAX

Reason:

The end of the CA ADS process source was reached, but a CA IDMS/DC Sort syntax set was still in progress. CA IDMS/DC Sort syntax must be terminated with a period (.) or a semi-colon (;), and must wholly reside within a single, non-included module.

Action:

Either correct the syntax using the EDITOR, issuing a RETRY in the EDITOR command line, or CANCEL the preprocessor section.

TPP7030E

MINIMUM ENTRY OF MODULE NAME IS REQUIRED

Reason:

To initiate a CA IDMS/DC Sort CA ADS Preprocessor session, a minimum entry of module name is required.

Action:

Enter a module name in the user display.

TPP7031I

PREPROCESSING TERMINATED BY USER REQUEST

Reason:

The user has requested the termination of the current CA IDMS/DC Sort preprocessing.

Action:

None.

TPP7032E

SYNTAX OVERFLOW - TOO MANY CONTIGUOUS LINES IN A SINGLE TPSORT STATEMENT

Reason:

A single CA IDMS/DC Sort syntax statement can only occupy 50 lines of user source.

Action:

Reduce the number of lines in the indicated statement to 50.

TPP7033E

INVALID WORD word IN STATEMENT/WORD word-position

Reason:

An invalid or misplaced word has been detected in the CA IDMS/DC Sort syntax, where word-position represents the sequential position in the CA IDMS/DC Sort statement.

Action:

Correct the CA IDMS/DC Sort syntax, and enter RETRY in the EDITOR command line.

TPP7034E

INCOMPLETE OR INVALID STATEMENT AT WORD word

Reason:

A CA IDMS/DC Sort statement has been incorrectly specified. Usually this error occurs because the statement terminator was encountered before a substatement was fully qualified.

Action:

Correct the CA IDMS/DC Sort syntax, and enter RETRY in the EDITOR command line.

TPP7035E

SUBPARAM parameter-number SEQUENTIAL POSITION OF WORD IS OF INCORRECT LENGTH

Reason:

A parameter substatement has failed a length edit at the specified word.

Action:

Correct the CA IDMS/DC Sort syntax, and enter RETRY in the EDITOR command line.

TPP7036E

WORD word WAS FOUND WHEN TERMINATION WAS EXPECTED

Reason:

Instead of the expected terminator, the specified word was encountered.

Action:

Correct the CA IDMS/DC Sort syntax, and enter RETRY in the EDITOR command line.

TPP7037E

AT LEAST ONE SET OF FIELDS MUST BE SPECIFIED

Reason:

The current SETSORT request requires a FIELDS keyword and one or more sets of FIELDS sub parameters.

Action:

Correct the SETSORT syntax and retry the preprocessor.

TPP7038E

RECORD record-name NOT FOUND

Reason:

The SETSORT statement specified a record which could not be located in the indicated dictionary/node/version.

Action:

Correct the FOR statement, and retry the preprocessor.

TPP7039E

ELEMENT element-name NOT IN INDICATED RECORD

Reason:

An element name in the FIELDS statement could not be located in the record specified in the FOR statement.

Action:

Correct the statement in error and retry the preprocessor.

TPP7042I

SUCCESSFUL UPDATE OF MODULE module-name

Reason:

The indicated module has been successfully updated.

Action:

None.

TPP7043E

DICTIONARY UNABLE TO BE READIED IN UPDATE MODE

Reason:

An attempt has been made to update the indicated dictionary in update mode. This error is associated with the CA ADS Preprocessor only.

Action:

Check the status of the area or specify another dictionary.

TPP7046I

SUCCESSFUL PREPROCESS OF TPSORT STATEMENTS

Reason:

The TPSG verb was entered in CA IDMS DME, and the syntax was preprocessed without errors.

Action:

None.

TPP7047E

ERROR(S) DETECTED IN TPSORT STATEMENT(S)

Reason:

The TPSG verb was entered in CA IDMS DME, and errors were encountered in the CA IDMS/DC Sort syntax.

Action:

Review the embedded error messages following each CA IDMS/DC Sort statement in error, and reenter the TPSG verb after all errors are corrected.

TPP7048E

ELEMENT element-name IS A CONDITIONAL NAME (88 LEVEL)

Reason:

The element-name is an 88-level condition name and cannot be used as a sort key. A field name is required.

Action:

Replace element-name with a valid field name.

TPP7091E

ESAMcode END OF FILE REACHED (BEYOND BOTTOM)

Reason:

An error occurred during the CA IDMS/DC Sort CA ADS preprocessor interface to the EDITOR.

Action:

Contact CA product support.

TPP7092E

ESAMcode INVALID PARAMETER LIST

Reason:

An error occurred during the CA IDMS/DC Sort CA ADS preprocessor interface to the EDITOR.

Action:

Contact CA product support.

TPP7093E

ESAMcode ILLEGAL CALL (PUT BEFORE OPEN)

Reason:

An error occurred during the CA IDMS/DC Sort CA ADS preprocessor interface to the EDITOR.

Action:

Contact CA product support.

TPP7094E

ESAMcode AN I/O ERROR OCCURRED. I/O ERROR CODE: error-code

Reason:

An error occurred during the CA IDMS/DC Sort CA ADS preprocessor interface to the EDITOR.

Action:

Contact CA product support.

TPP7095E

ESAMcode UNEXPECTED RETURN CODE WHILE CREATING THE SOURCE TEXT AREA

Reason:

An error occurred during the CA IDMS/DC Sort CA ADS preprocessor interface to the EDITOR.

Action:

Contact CA product support.

TPP7096E

GSIUPLOW MODULE ERROR MESSAGE: error-message

Reason:

An error occurred during the case change to CAPS ON for the CA IDMS/DC Sort CA ADS Preprocessor.

Action:

Contact CA product support.

TPU7050E

NO RECORD NAME FOUND IN SORT-CONTROL BLOCK

Reason:

The current request requires a record name in TPSRECN.

Action:

Contact CA product support.

TPU7051E

NON-IDMS INTERFACE REQUIRES ELEMENT NAMES

Reason:

A non-CA IDMS USER request requires element names in TPSKOCCS.

Action:

If the call was formatted by a preprocessor, contact CA product support.

TPU7052E

RECORD NOT FOUND IN DICTIONARY

Reason:

A CA IDMS USER request is in progress, but the value specified in TPSRECN cannot be located using the values in TPSRVRS, TPSDICT, and TPSNODE.

Action:

Correct the values which are in error, recompile the dialog or program, and reexecute.

TPU7053E

NUMBER OF ELEMENTS IN RECORD EXCEEDS 720

Reason:

This release of CA IDMS/DC Sort does not support a USER request for records which contain more than 720 elements.

Action:

Create a new record equivalent containing 720 or fewer elements, recompile the dialog or program, and reexecute.

TPU7054E

MAIN STORAGE NOT AVAILABLE

Reason:

Sufficient storage was not available to complete the USER request.

Action:

Increase the available user storage for the TP monitor.

TPU7055I

USER CANCELLED SORT

Reason:

A USER sort request was terminated before a key was specified.

Action:

None. The user program must contain logic to recognize this condition.

TPU7056E

NUMBER OF SORT KEYS EXCEEDS 16

Reason:

During a USER sort request, the user specified more than 16 keys.

Action:

Reinvoke the USER sort request, specifying 16 or fewer keys.

TPU7057E

SEQUENCE NUMBER MUST BE BETWEEN 1 AND 16

Reason:

A value other than 1 through 16 was specified in the SEQUENCE field on the CA IDMS/DC Sort USER screen.

Action:

Correct the indicated SEQUENCE field.

TPU7058E

SEQUENCE NUMBER IS A DUPLICATE

Reason:

More than one SEQUENCE field contains the same value on a USER sort screen.

Action:

Correct the appropriate SEQUENCE entry.

TPU7059E

SORT ORDER IS MISSING

Reason:

A SEQUENCE value was specified for an element in the CA IDMS/DC Sort USER screen, but the ORDER value is missing.

Action:

Add a related ORDER value or remove the indicated SEQUENCE value.

TPU7060E

SORT ORDER MUST BE (A) OR (D)

Reason:

An ORDER value other than 'A' or 'D' was specified on the USER screen.

Action:

Specify 'A' to sort in ascending order, or 'D' to sort in descending order, or remove the SEQUENCE and ORDER entries.

TPU7061E

AN IMPROPER PF KEY WAS PRESSED

Reason:

During USER screen processing, an undefined PF key was pressed.

Action:

Press the appropriate key.

TPU7062E

EXECUTION REQUIRES ALL ERRORS CORRECTED

Reason:

The PF3 key was pressed to execute the sort from the USER screen, but errors remain in the SEQUENCE and ORDER fields.

Action:

Correct the indicated errors, press Enter to validate, and then retry PF3.

TPU7063E

IDMS INTERFACE ABEND - IDMS STATUS idms-status

Reason:

During CA IDMS/DC Sort USER processing, an unidentified CA IDMS/DB abend occurred. The CA IDMS status code for the error can be found in TPSRETN.

Action:

See the CA IDMS Messages and Codes Guide.

TPU7064E

SEQUENCE NUMBER MISSING

Reason:

The sequence numbers specified on the USER screen must begin with 1 and proceed sequentially.

Action:

Correct the sequence number order, press Enter to validate, and retry PF3.

TPU7065E

CURSOR NOT POSITIONED ON SEQUENCE OR ORDER FIELD IN ERROR

Reason:

PF1 was pressed during a USER session to expand a short error message on a detail line. There are two possible reasons for this error:

1. The cursor is not positioned on a detail line item in error. The cursor must be on an item in either the SEQUENCE or ORDER column that has a short error message following it.
2. The indicated detail line has no error message(s).

Action:

Move the cursor to the detail line item for which a message expansion is needed, and press PF1 again.

TPU7066E

ONE OR MORE DETAIL FIELDS ARE IN ERROR

Reason:

ORDER or SEQUENCE fields have errors. Each field in error has its own associated short error message.

Action:

Move the cursor to a detail line item for which a message has been issued. Press PF1 to see an expanded message. Correct the fields in error, and press Enter to reedit the screen values.

Index

C

CA ADS • 100
CA ADS Preprocessor • 109
CA IDMS Dictionary Access • 14
CA IDMS/DC Sort Examples • 35
CA IDMS/DC Sort Parameter Statements • 17
CA IDMS/DC Sort System Flow • 96
COBOL/Assembler/PLI • 98
Customizing CA IDMS/DC Sort • 104

D

Demonstration • 104

E

Easy Selection of Criteria • 12
Editing Commands • 115
EDITOR • 115
ENDSORT Statement • 18

F

Flexible Online Sorting with CA IDMS/DC Sort • 11
Flexible Retrieval of Sorted Data • 14

G

GETSORT Statement • 18

K

Key Settings • 137

L

Line Commands • 130

M

Multiple Sort Keys in Each Sequence • 12
Multiple Sorts at One Terminal • 12

N

Non-Standard Message Codes--Batch • 140
Non-Standard Message Codes--Batch or Online • 143
Notation Conventions and Syntax Rules • 20

O

Operational Considerations • 95
Optional Online Criteria at Runtime • 12

P

Parameter Statements Make CA IDMS/DC Sort Easy to Use • 13
Preprocess Multiple Modules • 111
Preprocessor Support • 14
Primary Commands • 115
Processing Environment • 15
PUTSORT Statement • 18

Q

Quick and Easy Sorts • 12

S

Scroll Options • 115
Session Kept Active Across Pseudo Converses • 14
SETLIMIT Statement • 19
SETSORT Statement • 24
Storage Requirements • 98
System Flow • 112
System Limits • 98