DIL User Reference Manual

AA–M581B–TK

November 1984

The DIL is a set of callable subroutines that enable you to transfer data between TOPS–10, TOPS–20 and VMS systems.

This revised document supersedes the *DIL User Reference Manual*, order numbers AA–M581A–TK and AD–M581A–T1.

OPERATING SYSTEM: TOPS–20 V4.1 and later
for 2020 (KS) Systems
TOPS–20 V5.1 and later
for KL Systems
VAX/VMS V3.1 and later
TOPS–10 V7.02 and later

SOFTWARE: TOPS–10/TOPS–20 COBOL V12B or later
TOPS–10/TOPS–20 FORTRAN V7 or later
VAX/VMS COBOL V3.1 or later
VAX/VMS FORTRAN V3.0 or later

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

Northeast/Mid–Atlantic Region

Digital Equipment Corporation
PO Box CS2008
Nashua, New Hampshire 03061
Telephone:(603)884–6660

Central Region

Digital Equipment Corporation
Accessories and Supplies Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone:(312)640–5612

Western Region

Digital Equipment Corporation
Accessories and Supplies Center
632 Caribbean Drive
Sunnyvale, California 94086
Telephone:(408)734–4915

digital equipment corporation • marlboro, massachusetts

First Printing, February 1983
Updated, January 1984
Revised, November 1984

© Digital Equipment Corporation 1983, 1984. All Rights Reserved.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The following are trademarks of Digital Equipment Corporation:

digital™

DEC	MASSBUS	RSX
DECmate	PDP	RT
DECsystem–10	P/OS	UNIBUS
DECSYSTEM–20	Professional	VAX
DECUS	Q–BUS	VMS
DECwriter	Rainbow	VT
DIBOL	RSTS	Work Processor

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

CONTENTS

PREFACE

PART I DIL USAGE MATERIAL

CHAPTER 1 DIL CONCEPTS AND CAPABILITIES

1.1 INTRODUCTION 1-1
1.1.1 Data Interchange Library Status Codes 1-4
1.1.2 The DIL Interface Support Files 1-6

CHAPTER 2 DATA CONVERSION CONCEPTS AND CAPABILITIES

2.1 CONVERTING FIELDS 2-1
2.1.1 Conversion Using a Foreign Field Descriptor . . 2-2
2.1.2 Conversion Using a Detailed Description 2-4
2.1.3 Record Layouts and Byte Offsets 2-4

CHAPTER 3 TASK-TO-TASK CONCEPTS AND CAPABILITIES

CHAPTER 4 REMOTE FILE ACCESS CONCEPTS AND CAPABILITIES

4.1 OPENING A REMOTE FILE 4-2

PART II USING THE DIL FROM TOPS-10 AND TOPS-20

CHAPTER 5 TOPS-10 AND TOPS-20 DATA CONVERSION

5.1 DATA CONVERSION FROM TOPS-10/TOPS-20 COBOL 5-1
5.1.1 Compiling Programs 5-1
5.1.2 Including the Interface Support Files 5-1
5.1.3 Storing an FFD 5-2
5.1.4 Passing a Record to the XDESCR Routine 5-2
5.1.5 Checking For Errors 5-3
5.2 DATA CONVERSION FROM TOPS-10/TOPS-20 FORTRAN . . . 5-4
5.2.1 Including the Interface Support Files 5-4
5.2.2 Storing an FFD 5-4
5.2.3 Passing a Record to the XDESCR Routine 5-5
5.2.4 Checking For Errors 5-6
5.3 TOPS-10/TOPS-20 DATA CONVERSION REFERENCE 5-7
5.3.1 DILINI - Allow the DIL to Recognize Status Codes 5-7
5.3.2 XDESCR - To Create an FFD 5-8
5.3.3 XCGEN - To Perform General Purpose Conversion 5-10
5.3.4 XCVST - Convert String Fields 5-12
5.3.5 XCVFB - Convert Fixed-Point Binary Fields . . 5-13
5.3.6 XCVFP - Convert Floating-Point Fields 5-14
5.3.7 XCVPD - Convert Packed Decimal Fields 5-15
5.3.8 XCVDN - Convert Display Numeric Fields 5-17
5.3.9 XCFBDN - Convert Fixed-Point Binary Fields to Display Numeric Fields 5-19
5.3.10 XCFBPD - Convert Fixed-Point Binary Fields to Packed Decimal Fields 5-21
5.3.11 XCPDDN - Convert Packed Decimal Fields to Display Numeric Fields 5-22

5.3.12 XCPDFB - Convert Packed Decimal Fields to Fixed-Point Binary Fields 5-24
5.3.13 XCDNPD - Convert Display Numeric Fields to Packed Decimal Fields 5-26
5.3.14 XCDNFB - Convert Display Numeric Fields to Fixed-Point Binary Fields 5-28
5.3.15 CVGEN - Perform Conversion Without an FFD . . 5-30
5.4 TOPS-10/TOPS-20 DATA CONVERSION EXAMPLES 5-35
5.4.1 TOPS-10/TOPS-20 COBOL Data Conversion Example 5-35
5.4.2 TOPS-10/TOPS-20 FORTRAN Data Conversion Example 5-38

CHAPTER 6 TOPS-20 AND TOPS-10 TASK-TO-TASK

6.1 TASK-TO-TASK FROM TOPS-20 OR TOPS-10 COBOL 6-1
6.1.1 Compiling Programs 6-1
6.1.2 Including the Interface Support Files 6-1
6.1.3 Storing a Network Logical Name (NLN) 6-2
6.1.4 Storing Task and User Attributes 6-2
6.1.5 Checking the Status of a Task-to-Task Routine . 6-3
6.1.6 The TOPS-10 Software Interrupt System 6-3
6.2 TASK-TO-TASK FROM TOPS-20 AND TOPS-10 FORTRAN . . 6-3
6.2.1 Including the Interface Support Files 6-4
6.2.2 Storing a Network Logical Name (NLN) 6-4
6.2.3 Storing Task and User Attributes 6-4
6.2.4 Checking the Status of a Task-to-Task Routine . 6-5
6.2.5 The TOPS-10 Software Interrupt System 6-5
6.3 TOPS-10 AND TOPS-20 TASK-TO-TASK REFERENCE 6-6
6.3.1 NFGND - Return the Status of a Link 6-6
6.3.2 NFINF - Get Information About the Other End of a Logical Link. 6-8
6.3.3 NFOPA - Open a Link From an Active Task (ASCII) 6-11
6.3.4 NFOPB - Open a Link From an Active Task (Binary) 6-14
6.3.5 NFOP8 - Open a Link From an Active Task (8-bit) 6-17
6.3.6 NFOPP - Open a Link From a Passive Task . . . 6-20
6.3.7 NFACC - Accept a Connection 6-22
6.3.8 NFREJ - Reject a Connection 6-24
6.3.9 NFRCV - Receive Data Over a Link 6-25
6.3.10 NFSND - Send Data Over a Link 6-28
6.3.11 NFINT - Send an Interrupt Data Message Over a Link 6-30
6.3.12 NFRCI - Receive an Interrupt Data Message Over a Link 6-31
6.3.13 NFCLS - Close a Link 6-32
6.4 TOPS-10/TOPS-20 TASK-TO-TASK EXAMPLES 6-34
6.4.1 TOPS-10/TOPS-20 COBOL Task-to-Task Examples . 6-34
6.4.2 TOPS-10/TOPS-20 FORTRAN Task-to-Task Examples 6-40

CHAPTER 7 TOPS-10 AND TOPS-20 REMOTE FILE ACCESS

7.1 REMOTE FILE ACCESS FROM TOPS-10 OR TOPS-20 COBOL . 7-1
7.1.1 Compiling Programs 7-1
7.1.2 Including the Interface Support Files 7-1
7.1.3 Storing a File Number 7-2
7.1.4 Storing Accounting Information 7-2
7.1.5 Reading and Writing Remote Data 7-2
7.1.6 Checking the Status of a Remote File Access Routine 7-3
7.1.7 The TOPS-10 Software Interrupt System 7-3
7.2 REMOTE FILE ACCESS FROM TOPS-20 OR TOPS-10 FORTRAN 7-3
7.2.1 Including the Interface Support Files 7-4
7.2.2 Storing a File Number 7-4

7.2.3 Storing Account Information 7-4
7.2.4 Reading and Writing Remote Data 7-4
7.2.5 Checking the Status of a Remote File Access Routine 7-5
7.2.6 The TOPS-10 Software Interrupt System 7-5
7.3 TOPS-20 AND TOPS-10 REMOTE FILE ACCESS REFERENCE . 7-6
7.3.1 ROPEN - Open a Remote File 7-6
7.3.2 RREAD - Read Data From a Remote File 7-9
7.3.3 RWRITE - Write Data To a Remote File 7-10
7.3.4 RCLOSE - Close a Remote File 7-11
7.3.5 RDEL - Delete a File 7-13
7.3.6 RSUB - Submit a File For Batch Processing . . 7-15
7.3.7 RPRINT - Print a File 7-17
7.4 TOPS-20 REMOTE FILE ACCESS EXAMPLES 7-19
7.4.1 TOPS-20 COBOL Remote File Access Example . . . 7-19
7.4.2 TOPS-20 FORTRAN Remote File Access Example . . 7-21
7.5 TOPS-10 REMOTE FILE ACCESS EXAMPLES 7-24
7.5.1 TOPS-10 COBOL Remote File Access Example . . . 7-24
7.5.2 TOPS-10 FORTRAN Remote File Access Example . . 7-26

CHAPTER 8 LINKING A TOPS-10/TOPS-20 PROGRAM

8.1 DECsystem-10 LINKAGE INSTRUCTIONS 8-1
8.2 DECSYSTEM-20 LINKAGE INSTRUCTIONS 8-1
8.3 DECSYSTEM-10 AND DECSYSTEM-20 OVERLAY INSTRUCTIONS 8-1

PART III USING THE DIL FROM VMS

CHAPTER 9 VMS DATA CONVERSION

9.1 DATA CONVERSION FROM VMS COBOL 9-1
9.1.1 Including the Interface Support Files 9-1
9.1.2 Storing an FFD 9-3
9.1.3 Passing a Record to the DIX$MAK_DES_DET Routine 9-3
9.1.4 Checking for Errors 9-3
9.2 DATA CONVERSION FROM VMS FORTRAN 9-4
9.2.1 Including the Interface Support Files 9-4
9.2.2 Storing an FFD 9-5
9.2.3 Passing a Record to the Data Conversion Routines 9-6
9.2.4 Checking for Errors 9-6
9.3 VMS DATA CONVERSION REFERENCE 9-7
9.3.1 DIX$MAK_DES_DET - Create an FFD 9-7
9.3.2 DIX$BY_DIX_DES - Perform General Conversion . . 9-9
9.3.3 DIX$BY_DET - Convert a Field Without an FFD . 9-11
9.4 VMS DATA CONVERSION EXAMPLES 9-15
9.4.1 VMS COBOL Data Conversion Example 9-15
9.4.2 VMS FORTRAN Data Conversion Example 9-17

CHAPTER 10 VMS TASK-TO-TASK

10.1 TASK-TO-TASK FROM VMS COBOL 10-1
10.1.1 Including the Interface Support Files 10-1
10.1.2 Storing a Network Logical Name 10-3
10.1.3 Storing Task and User Attributes 10-3
10.1.4 Checking the Status of a Task-to-Task Routine 10-3
10.2 TASK-TO-TASK FROM VMS FORTRAN 10-4
10.2.1 Including the Interface Support Files 10-4
10.2.2 Storing a Network Logical Name 10-5
10.2.3 Storing Task and User Attributes 10-6
10.2.4 Checking the Status of a Task-to-Task Routine 10-6

10.3 VMS TASK-TO-TASK REFERENCE 10-7
10.3.1 DIT$NFGND - Return the Status of Links 10-7
10.3.2 DIT$NFINF - Get Information About the Other End of a Link. 10-9
10.3.3 DIT$NFOPA - Open an ASCII Link From an Active Task 10-12
10.3.4 DIT$NFOPB - Open a Binary Link From an Active Task 10-15
10.3.5 DIT$NFOP8 - Open an 8-Bit Link From an Active Task 10-18
10.3.6 DIT$NFOPP - Open a Link From a Passive Task . 10-21
10.3.7 DIT$NFACC - Accept a Connection 10-23
10.3.8 DIT$NFREJ - Reject a Connection 10-25
10.3.9 DIT$NFRCV - Receive Data 10-26
10.3.10 DIT$NFSND - Send Data 10-28
10.3.11 DIT$NFRCI - Receive an Interrupt Data Message Over a Link 10-30
10.3.12 DIT$NFINT - Send an Interrupt Data Message Over a Link 10-31
10.3.13 DIT$NFCLS - Close a Link 10-32
10.4 VMS TASK-TO-TASK EXAMPLES 10-33
10.4.1 VMS COBOL Task-to-Task Examples 10-33
10.4.2 VMS FORTRAN Task-to-Task Examples 10-39

CHAPTER 11 VMS REMOTE FILE ACCESS

11.1 REMOTE FILE ACCESS FROM VMS COBOL 11-1
11.1.1 Including the Interface Support Files 11-1
11.1.2 Storing a File Number 11-3
11.1.3 Storing Account Information 11-3
11.1.4 Reading and Writing Remote Data 11-3
11.1.5 Checking the Status of a Remote File Access Routine 11-3
11.2 REMOTE FILE ACCESS FROM VMS FORTRAN 11-4
11.2.1 Including the Interface Support Files 11-4
11.2.2 Storing a File Number 11-6
11.2.3 Storing Account Information 11-6
11.2.4 Reading and Writing Remote Data 11-6
11.2.5 Checking the Status of a Remote File Access Routine 11-6
11.3 VMS REMOTE FILE ACCESS REFERENCE 11-7
11.3.1 DIT$ROPEN - Open a Remote File 11-7
11.3.2 DIT$RREAD - Read Data From a Remote File . . . 11-10
11.3.3 DIT$RWRITE - Write Data to a Remote File . . . 11-11
11.3.4 DIT$RCLOSE - Close a Remote File 11-12
11.3.5 DIT$RDEL - Delete a File 11-13
11.3.6 DIT$RSUB - Submit a File For Batch Processing 11-15
11.3.7 DIT$RPRINT - Print a File 11-17
11.4 VMS REMOTE FILE ACCESS EXAMPLES 11-19
11.4.1 VMS COBOL Remote File Access Examples 11-19
11.4.2 VMS FORTRAN Remote File Access Example 11-22

CHAPTER 12 LINKING A VMS PROGRAM

PART IV APPENDIXES

APPENDIX A LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

A.1 SPECIFYING DATA NAMES A-6

APPENDIX B DIL DATA FORMATS

B.1 ALPHANUMERIC STRING DATA TYPES B-1
B.2 BINARY FIXED-POINT DATA TYPES B-3
B.3 FLOATING-POINT DATA TYPES B-7
B.4 DISPLAY NUMERIC DATA TYPES B-11
B.5 PACKED DECIMAL DATA TYPES B-19

APPENDIX C THE DIL SAMPLE APPLICATION

APPENDIX D TASK IDENTIFICATION

D.1 ACCESSING A TASK BY NAME D-1
D.2 ACCESSING A TASK THAT PROVIDES A CLASS OF SERVICE D-2
D.3 ACCESSING SPECIFIC TASK WHICH PROVIDES SERVICE (TOPS-20 ONLY) D-2
D.4 VMS PASSIVE TASKS D-2
D.4.1 Tasks that Wait for a Connect Request D-3
D.4.2 Tasks Started as a Result of a Request D-3
D.4.3 Example of a Task Started as a Result of a Request D-4

APPENDIX E DIL STATUS CODES

APPENDIX F BIT TRANSPORT

F.1 NFOPB LINKS F-1
F.2 REMOTE FILE ACCESS IN ASCII MODE AND NFOPA LINKS . F-3
F.3 NFOP8 LINKS F-4

APPENDIX G TASK-TO-TASK AND REMOTE-FILE-ACCESS ERROR MESSAGES

G.1 REMOTE-FILE-ACCESS ERROR CODES AND THEIR MEANINGS: G-1
G.1.1 ROPEN/DIT$ROPEN Routine: G-1
G.1.2 RREAD/DIT$RREAD: G-3
G.1.3 RWRITE/DIT$RWRITE: G-4
G.1.4 RCLOSE/DIT$RCLOSE: G-5
G.1.5 RDEL/DIT$RDEL: G-6
G.1.6 RSUB/DIT$RUB: G-7
G.1.7 RPRINT/DIT$RPRINT: G-8
G.2 TASK-TO-TASK ERRORS AND THEIR MEANINGS: G-10
G.2.1 NFOPA, NFOPB, NFOP8, NFOPP/DIT$NFOPA, DIT$NFOPB, DIT$NFOP8, DIT$NFOPP: G-10
G.2.2 NFGND/DIT$NFGND: G-11
G.2.3 NFACC/DIT$NFACC: G-12
G.2.4 NFRCV/DIT$NFRCV: G-13
G.2.5 NFSND/DIT$NFSND: G-14
G.2.6 NFREJ/DIT$NFREJ: G-14
G.2.7 NFINT/DIT$NFINT: G-15
G.2.8 NFRCI/DIT$NFRCI G-15
G.2.9 NFCLS/DIT$NFCLS: G-16
G.2.10 NFINF/DIT$NFINF: G-17

INDEX

FIGURES

1-1	A Sample DIL Application	1-3
1-2	Format of a VMS Status Code	1-4
1-3	Format of a TOPS-10 or TOPS-20 Status Code	1-4
2-1	TOPS-10 or TOPS-20 Record Layout	2-5
2-2	VMS Record Layout	2-5
3-1	Simple Task-to-Task Application	3-3
3-2	Task-to-Task Application	3-4
C-1	Sample Application Flowchart	C-1

PREFACE

The Data Interchange Library (DIL) is a set of callable subroutines that can be used by COBOL and FORTRAN programmers on TOPS-20, TOPS-10 and VMS systems.

The DIL User Reference Manual describes the DIL and explains how to use it. The manual consists of three basic parts: introductory material, reference sections and appendices. The introductory chapters (Chapters 1-4) present the concepts and capabilities of each part of the Data Interchange Library. The reference portion of the manual is divided first into TOPS-10/TOPS-20 (Chapters 5-8) and VMS (Chapters 9-12) sections. Each of these system-specific sections has four subsections that explain:

- The Data Conversion Routines
- The Task-to-Task Routines
- The Remote File Access Routines
- Linking a program

The reference sections give specific instructions for using the DIL subroutines from your operating system.

Appendix A shows the language-specific values for DIL Names. Whenever the manual refers to the DIL Name of a status code, argument value or data type, check Appendix A to find its value for your language and system. Appendix D indicates the numerical value, DIL Name and meaning for status codes returned by the Data Interchange Library Routines. Other appendices present more in-depth information about the DIL.

Examples in this manual use lower case to indicate that you can supply your own name for this variable. In the following call, you can use any names permitted by your system to represent the "nln" and "wait" data items.

```
ENTER MACRO NFGND USING nln, wait.
```

CHAPTER 1
DIL CONCEPTS AND CAPABILITIES

CHAPTER 1

DIL CONCEPTS AND CAPABILITIES

1.1 INTRODUCTION

The Data Interchange Library (DIL) is a set of callable subroutines that enable you, the COBOL or FORTRAN programmer, to access and use data that resides on another computer system. The DIL allows you to pass data between programs on different systems or to directly access records in files on other systems. You can use the DIL to access a single record within a file and avoid having to transfer an entire file to your system. If the accessed data is not of the proper format or data type, DIL provides the necessary data conversion facilities.

Transporting an entire file can be costly and time consuming, particularly when you are dealing with a large file. It is probably not cost effective, for example, to spend hours transferring the entire personnel file to your system when you only need the data for one employee. If you have a small system, you probably don't even have the capability to store such a big file.

To use the DIL in a multiple computer environment, the computers must be connected by DECnet to form a network. The network can be homogeneous or heterogeneous. A homogeneous network consists of two or more systems of the same type. A heterogeneous network is made up of computers of different types. The DIL can be used in a network that supports any combination of VAX, DECSYSTEM-20 or DECsystem-10 computers.

The DIL consists of three types of routines:

1. Data Conversion Routines
   The Data Conversion Routines provide you with the ability to translate fields from one data type to another. These routines are especially useful when you move data through a heterogeneous network using the Task-to-Task Routines. A record written in VMS binary format, for example, cannot be used, as is, by a TOPS-20 system, because the two systems have different methods of internal data representation.

This release of the DIL supports "like-to-like" translation of some COBOL and FORTRAN data types. A like-to-like translation is a translation between data types in a single class. The Data Conversion Routines support translation of the following TOPS-10, TOPS-20 and VMS data types:

- Alphanumeric string
- Fixed-point data (includes fixed-point binary, packed decimal and display numeric)
- Floating-point binary

You can, for example, use the Data Conversion Routines to convert a VMS COBOL string data item to a TOPS-10/TOPS-20 COBOL, VMS FORTRAN or TOPS-10/TOPS-20 FORTRAN string data item. You cannot, however, use the routines to convert that same data item to any non-string data item.

2. Task-to-Task Routines

The Task-to-Task Routines allow you to move data between programs on different systems. To use the Task-to-Task Routines, you establish a network connection between two programs, rather than between a program and a file. This network connection is called a logical link. Once you have successfully established a logical link between programs, you can transfer data over that link. A program can use the Task-to-Task Routines to perform the following functions:

- Open a logical link to another program.
- Wait for another program to request its services.
- Get information about the status of network connections.
- Get information about the other end of a logical link.
- Accept a connection request from another program.
- Reject a connection request from another program.
- Receive data from another program over a logical link.
- Send data to another program over a logical link.
- Receive an interrupt data message from another program over a logical link.
- Send an interrupt data message to another program over a logical link.
- Close a logical link.

3. Remote File Access Routines
   The Remote File Access routines allow you to access and use data that resides in a sequential ASCII file on another computer. The Remote File Access routines perform the following functions:

   - Open a file for reading, writing or appending
   - Read a record from a file
   - Write a record to a file
   - Close a file
   - Delete a closed file
   - Submit a closed file for batch processing
   - Print a closed file

A practical DIL application might be used by a company that wants to eliminate the collection of paper employee labor tickets. Each week, every employee of the hypothetical company fills out a paper labor ticket (containing accounting information) and submits it to his or her secretary. By using the DIL in a heterogeneous network, the company could easily automate the labor ticket process. Figure 1-1 illustrates one method of using the DIL to collect weekly labor ticket information.

A. The user submits labor ticket information from a remote terminal connected to either a VAX-11 or DECSYSTEM-20.

B. The Task-to-Task Routines are used to ship data to a program on the DECSYSTEM-20.

C. The Data Conversion Routines are used to convert data, if necessary, to DECSYSTEM-20 format. The data is then stored on the DECSYSTEM-20.

D. The Remote File Access Routines are used by a program on the VAX-11 to access data in the file on the DECSYSTEM-20 in order to write a summary report.

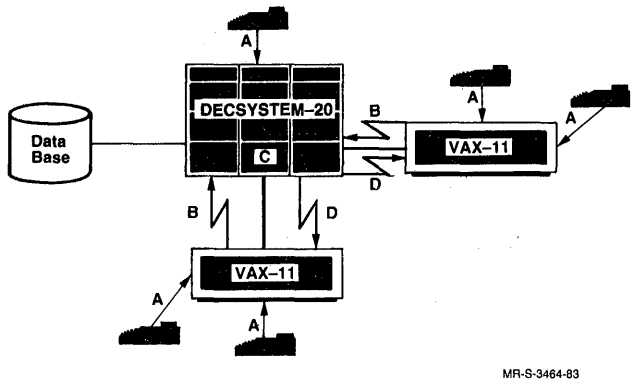


Figure 1-1: A Sample DIL Application

1.1.1 Data Interchange Library Status Codes

Each DIL routine returns a status code or condition value when it finishes processing. The status code lets you know whether the routine ran successfully. It alerts you to a problem by issuing success, warning, error or fatal severity codes. See Appendix D for a description of specific DIL status codes.

The DIL on VMS uses the standard VMS condition value. A condition value (or status code) is a word containing several fields that describe the type of error and its severity. Besides indicating the success or failure of the called subroutine, the condition value can provide the following information:

- Severity of the failure
- Error identification
- Associated message text (VMS only)
- Facility detecting the error
- Control of error message printing (VMS only)

Figures 1-2 and 1-3, below, illustrate the format of the status code returned by the DIL routines.

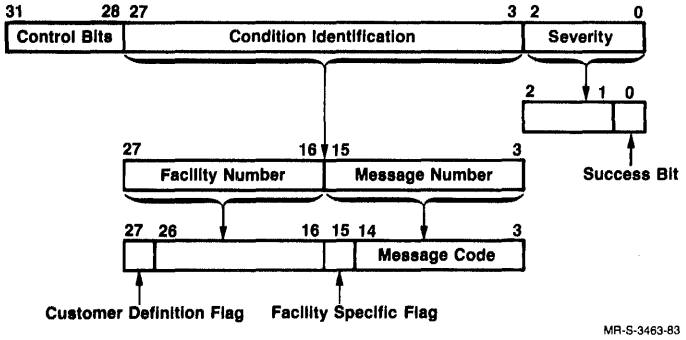


Figure 1-2: Format of a VMS Status Code

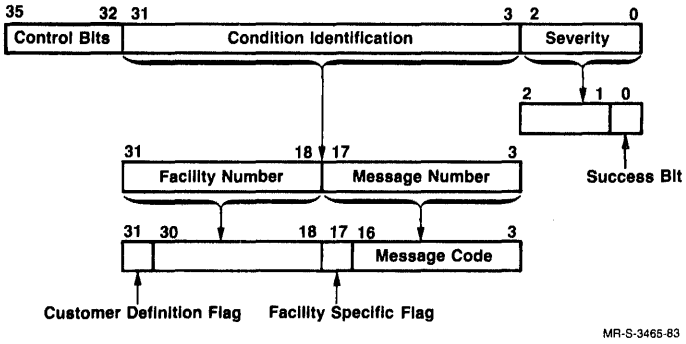


Figure 1-3: Format of a TOPS-10 or TOPS-20 Status Code

A status code is composed of the following parts:

Severity Code
The severity code indicates the seriousness of the error. The DIL returns the following severity codes:

0	warning
1	success
2	error
3	information (considered a success code)
4	severe error
5-7	reserved for future use

Condition Identification
The condition identification uniquely identifies the condition on a system-wide basis. Condition identification is composed of the facility number and the message number (described on the following page).

Facility Number
The facility number identifies the subsystem that generates the status code.

- Data Conversion routines return a facility number of 232 decimal.
- Remote File Access and Task-to-Task routines return a facility number of 233 decimal.

Message Number
The message number identifies the specific error that you made in your call to the routine.

Control Bits
A condition value normally contains four control bits. VMS status codes contain four zeroes as the value for this field. TOPS-10 and TOPS-20 status codes do not use this field. These bits are reserved for Digital use.

On VMS systems, the LIB$MATCH_COND function is available to compare two status codes and determine if they refer to the same condition. See the most recent version of the VAX-11 Run-Time Library Reference Manual for further information.

1.1.2 The DIL Interface Support Files

The DIL requires that you pass numeric values as parameters to the routines. These parameters can have different meanings, depending upon the specific position of the parameter in a call to a DIL routine. As a programmer, you may find it difficult to remember which numeric values you will need to make the different DIL functions occur. You may also prefer to specify symbolic names rather than to directly use the required numeric values. To simplify your use of the DIL, several text files, called Interface Support files, are provided with the DIL. These files define symbolic names for the parameters for each of the DIL routines. The names found in the Interface Support files will make it easier for you to communicate with the DIL routines.

There are two classes of Interface Support files, the "native" class and the "compatible" class. The native class Interface Support files define names that are native to the system for which the program is written. The compatible class of support files, define names that are compatible with both TOPS-10/TOPS-20 and VMS language names. TOPS-10/TOPS-20 systems do not have compatible class support files, since the TOPS-10/TOPS-20 native class names are already compatible with VMS language names. A set of native class support files exists for both COBOL and FORTRAN on TOPS-10/TOPS-20 systems. On VMS systems both the native and compatible classes of support files exist for both FORTRAN and COBOL. For VMS FORTRAN, these compatible names are also ANSI Standard names. If you want to write a program that can be easily transported to a TOPS-10 or TOPS-20 system, you may want to use the compatible class Interface Support files.

For each language/system/class combination, the DIL has two types of Interface Support files:

1. General DIL files that define terms applicable to all of the DIL routines from the language and the operating system that you plan to use.

2. Function-specific files which describe terms applicable to the type of DIL routines (Data Conversion, Remote File Access, or Task-to-Task), the language and the operating system that you plan to use. There are two types of function-specific files:

   - DIX files, which include the terms for the Data Conversion Routines

   - DIT files, which include the terms for the Remote File Access routines and the terms for the Task-to-Task Routines

You must include both general and function-specific files when you compile your program. If, for example, you want to use the Task-to-Task Routines, you should include the general DIL Interface Support file and the function-specific DIT file.

The Interface Support files for TOPS-10/TOPS-20 systems are provided in an appropriate manner for the languages they support. On VMS systems, the Interface Support files are provided as a single text library called DIL.TLB. You can use language-specific features to extract the information from the VMS Text library for inclusion in programs.

Appendix A contains a listing of the names and values included in the Interface Support files. You can use the LIBARY system utility on TOPS-10 or CPYLIB on TOPS-20 to either extract or modify individual modules in the COBOL library file. You can simply TYPE or PRINT the individual TOPS-10 and TOPS-20 FORTRAN Interface Support files. On VMS systems you can use the LIBRARY DCL command to either extract the individual modules in the Interface Support files as text files or to modify the files.

For more information about using the DIL Interface Support files, see the language-specific sections of this manual. For information about the names defined in the Interface Support files, refer to Appendix A.

Further Information:

To learn more about DECnet, see the most recent version of the DECnet User's Guide for your operating system.

An overview of DECnet can be found in the most recent version of Introduction to DECnet.

To learn more about Status Codes, see the most recent version of The VAX-11 Guide to Creating Modular Library Procedures and the system specific reference sections of this manual.

CHAPTER 2
DATA CONVERSION CONCEPTS AND CAPABILITIES

CHAPTER 2

DATA CONVERSION CONCEPTS AND CAPABILITIES

When you transfer records between heterogeneous systems, the data formats will not be understandable to both computers. Data representation in VMS memory differs from data representation in TOPS-10 or TOPS-20 memory. This means that a VMS COBOL record (and its fields), cannot be used, as is, by a TOPS-10 or TOPS-20 COBOL program. The Data Conversion Routines solve this problem by allowing you to translate a field from one data type to another.

This release of the DIL supports "like-to-like" translation within the following classes of data types:

- Alphanumeric string
- Fixed-point data (includes fixed-point binary, packed decimal and display numeric)
- Floating-point binary

A like-to-like translation is a translation from a particular data type to another data type within the same class. You can, for example, use the Data Conversion Routines to convert a VMS COBOL string data item to a TOPS-20 COBOL, TOPS-20 FORTRAN or VMS FORTRAN string data item. You cannot, however, use the routines to convert that same string data item to any type of non-string data item, such as a floating-point number.

2.1 CONVERTING FIELDS

You can use the Data Conversion Routines to convert a field while it still resides on its native system, before you transfer the field to another system. You can also wait and perform any necessary data conversion after you transfer the field. No matter where the conversion is done, the Data Conversion Routines require that you describe two fields: the source field and the destination field. The Data Conversion Routines provide two basic ways of describing a field.

2.1.1 Conversion Using a Foreign Field Descriptor

The Data Conversion Routines provide a convenient way to describe a field in a record that requires conversion: the Foreign Field Descriptor (FFD). An FFD contains descriptive information about a field, including:

- the name of the record that contains the field
- the record's native system
- the position of the field within the record
- the address of the record

This information allows the DIL to locate the field for conversion.

To build an FFD for a field, you must use one of the Data Conversion Routines. If you are working on a VMS system, the DIX$MAK_DES_DET routine builds the FFD. If you are working on a TOPS-10 or TOPS-20 system, the XDESCR routine builds the FFD.

To use the FFD method of data conversion you make two calls to the XDESCR (or DIX$MAK_DES_DET) routine; one call is used to create an FFD for the source field, the other to create an FFD for the destination field. The source field is the field that you want to convert. The destination field is the field after it has been converted. The source and destination FFDs must be different fields. The FFDs are data items that you define in your program and use when you call the XDESCR routine. When the routine successfully finishes processing, it returns an FFD value in the data item.

Once you have FFDs for the source and destination fields, you can use one of the other Data Conversion Routines to convert the field. These conversion routines require only two arguments: an FFD for the source field and an FFD for the destination field.

If you are working on a TOPS-10 or TOPS-20 system, use one of the following single-function conversion routines to convert the field using FFDs.

XCVST
: Use the XCVST routine if you want to convert a string field.

XCVFB
: Use the XCVFB routine if you want to convert a fixed-point binary field.

XCVFP
: Use the XCVFP routine if you want to convert a floating-point field.

XCVPD
: Use the XCVPD routine if you want to convert a packed decimal field.

XCVDN
: Use the XCVDN routine if you want to convert a display numeric field.

XCFBDN
: Use the XCFBDN routine if you want to convert a fixed-point binary source field to a display numeric destination field.

XCDNFB
: Use the XCDNFB routine if you want to convert a display numeric source field to a fixed-point binary destination field.

XCPDDN
: Use the XCPDDN routine if you want to convert a packed decimal source field to a display numeric destination field.

XCDNPD
: Use the XCDNPD routine if you want to convert a display numeric source field to a packed decimal destination field.

XCPDFB
: Use the XCPDFB routine if you want to convert a packed decimal source field to a fixed-point binary destination field.

XCFBPD
: Use the XCFBPD routine if you want to convert a fixed-point binary source field to a packed decimal destination field.

XCGEN
: Use the XCGEN routine to perform any type of conversion allowed by the DIL. (When you link a program that calls XCGEN with the DIL, the LINKER loads all of the conversion routines. When you use one of the single-function routines described above, only the routines that convert the specific data types are loaded.)

If you are working on a VMS system, use the DIX$BY_DCR_DES routine to perform string, fixed-point binary, packed decimal, display numeric or floating-point conversions.

When you convert a field using an FFD, the DIL creates an FFD for the field only once, no matter how many times you plan to convert the field. After creating the FFDs, you only have to call the proper single-function (or DIX$BY_DIX_DES on VMS) routine whenever you want to convert the field. The "Detailed Description" method of data conversion, described in the next section, processes location and description information about the field each time you convert the field. If you plan to convert a field many times, your program will run more efficiently if you convert the field using a Foreign Field Descriptor. See Section 2.1.2, below.

Further Information:

TOPS-10 and TOPS-20 users should read Sections 5.1, 5.2 and 5.3.2 for further information.

VMS users should read Sections 9.1, 9.2 and 9.3.1 for further information.

2.1.2 Conversion Using a Detailed Description

The detailed description method of data conversion allows you to convert a field without first making a Foreign Field Descriptor for the field. If you are working on a TOPS-10 or TOPS-20 system, you can use the CVGEN routine to convert a field without an FFD. If you are working on a VMS system, the DIX$BY_DET routine converts the field without an FFD.

CVGEN (DIX$BY_DET on VMS) is a general purpose conversion routine; it accepts location and description arguments for both the source and destination fields, eliminating the need to explicitly create a Foreign Field Descriptor.

When you call CVGEN (DIX$BY_DET on VMS), you uniquely identify the source and destination fields and supply the data type of the field to be converted. Because you describe both the source and destination fields in one routine, you don't have to use the single-function (or DIX$BY_DIX_DES on VMS) routines to perform the actual conversion. The CVGEN routine itself performs the conversion.

Using the CVGEN routine to perfom data conversion is inappropriate if you plan to perform a particular conversion more than a few times. CVGEN calculates source and destination field locations each time it is called by your program. If you convert a field using the CVGEN routine eighty times during a program run, the routine must read the same location and description arguments eighty times. Since an FFD is created only once, it would be faster to describe the field with an FFD. You could then use one of the other routines to perform the conversion. If, on the other hand, you plan to convert the field only once, or the program is table-driven, it may better to use the CVGEN routine.

Further Information:

TOPS-10 and TOPS-20 users should read Section 5.3.15 for information about CVGEN.

VMS users should read Section 9.3.3 for information about DIX$BY_DET.

2.1.3 Record Layouts and Byte Offsets

To convert a field, the conversion routine must know exactly where the field resides in local memory. It must have this information for both the source and destination fields.

To describe the location of a field in a record to the Data Conversion Routines, you must supply values for byte size, byte offset and bit offset. The bit offset is always zero. The byte offset is the number of bytes in a record that precede the field that you want to convert. You must also know the data type of the field. You can find these values by checking or creating a detailed layout for the record. See Appendix A of this manual for a list of data type names.

Since TOPS-10/TOPS-20 systems represent data in 36 bit words and VMS systems represent data in 8 bit bytes, the byte size, byte offset and data type of a field will differ on the two systems. Figure 2-1 shows a sample TOPS-10 or TOPS-20 COBOL record. Figure 2-2 shows a very similar record written in VMS COBOL format. You use the same COBOL picture clause for both records regardless of what type of system you are using. The internal format of the record is controlled by the system where it is used. Both examples show the byte size, byte offset, bit offset and data type for each field within the record. The "(0)" shown in the Data Type column represents the scale factor of the field. The scale factor is listed only for data types for which a scale factor is valid. Note that the byte size of the fields in the VMS record is always 8, while the byte size of the fields in the TOPS-10/TOPS-20 record can vary. The byte offset is calculated relative to the byte size.

			Byte Size	Byte Offset	Bit Offset	Data Type
01	JOB-TICKET.					
05	NAME	PIC X(30).	6	0	0	Sixbit
05	COST-CENTER	PIC X(5).	6	30	0	Sixbit
05	WD-END-DATE	PIC 9(6).	6	35	0	Sixbit
05	TOTAL-HOURS	COMP-1.	36	7	0	Float-36
05	ACTIV-CODE	PIC XXX.	6	48	0	Sixbit
05	PL-NUM	PIC XXX	6	51	0	Sixbit
05	DIS-NUM	PIC 9(5) COMP.	36	9	0	Sbf36(0)
05	MFG-NUM	PIC 9(5) COMP.	36	10	0	Sbf36(0)
05	HOURS	COMP-1.	36	11	0	Float-36
05	OP-CD	PIC X(5).	6	72	0	Sixbit

Figure 2-1: TOPS-10 or TOPS-20 Record Layout

			Byte Size	Byte Offset	Bit Offset	Data Type
01	JOB-TICKET.					
05	NAME	PIC X(30).	8	0	0	ASCII-8
05	COST-CENTER	PIC X(5).	8	30	0	ASCII-8
05	WD-END-DATE	PIC 9(6).	8	35	0	ASCII-8
05	TOTAL-HOURS	COMP-1.	8	41	0	F-Float
05	ACTIV-CODE	PIC XXX.	8	45	0	ASCII-8
05	PL-NUM	PIC XXX.	8	48	0	ASCII-8
05	DIS-NUM	PIC 9(5) COMP.	8	51	0	Sbf32(0)
05	MFG-NUM	PIC 9(5) COMP.	8	55	0	Sbf32(0)
05	HOURS	COMP-1.	8	59	0	F-Float
05	OP-CD	PIC X(5).	8	63	0	ASCII-8

Figure 2-2: VMS Record Layout

Further information:

Data representation and record layout information can be found in the most recent version of the following manuals:

VAX-11 COBOL Language Reference Manual

COBOL-74 Language Manual

TOPS-10/TOPS-20 FORTRAN Language Manual

VAX-11 FORTRAN Language Reference Manual

CHAPTER 3
TASK–TO–TASK CONCEPTS AND CAPABILITIES

CHAPTER 3

TASK-TO-TASK CONCEPTS AND CAPABILITIES

The Task-to-Task routines permit you to move data between programs on different systems in a network. These routines differ from the Remote File Access Routines because Task-to-Task allows communication between programs whereas Remote File Access permits a program to access a file on another system. The Task-to-Task routines are especially helpful in a distributed data base environment, where a corporate data base resides on a large computer, but you also have remote, satellite systems using the data base.

A simple Task-to-Task application, for example, might be used by a wholesale greengrocer and his company's central warehouse. The DECSYSTEM-20 at the warehouse contains information about the price and quantity of all products on hand at the warehouse. The grocer, in his shop, has a VAX-11 connected by DECnet to the central machine at the warehouse. If a restauranteur walks into the store demanding 100 crates of mangoes for immediate delivery, the shopkeeper runs a program on his VAX-11 which queries a program on the main DECSYSTEM-20: "Do we have 100 crates of mangoes?" The program at the warehouse sends a message to the grocer: "OK, we have the mangoes," and updates its inventory to reflect the sale. (See Figure 3-1, at the end of this chapter.)

The Task-to-Task Routines become even more useful if we expand this example to include a chain of greengroceries, each connected, by DECnet, to the central data base. A program using the Task-to-Task routines can connect with several other programs. If another restauranteur simultaneously appears at a different grocery store demanding 200 crates of mangoes, the owner of this store can run a program on his computer querying the data base: "Do we have 200 crates of mangoes?" The program at the warehouse processes the first request, for 100 cases. It then reads the next request, from shopkeeper number two, and sends a message: "Sorry, we're all sold out." The program at the warehouse remains open, waiting for requests from other grocers on the network. (See Figure 3-2, at the end of this chapter.)

The programs are able to transfer information via a network connection. This connection is called a logical link. A logical link is established with one of the four "open link" Task-to-Task Routines, described below. Every program that uses the Task-to-Task Routines to transfer data must establish a logical link. The program on one side of the link is called a "passive task;" the other program is an "active task." The passive task, also known as the server program, waits for other programs to link to it and request its services. The active task connects to the server program and requests its services. A passive task can simultaneously serve many active tasks, but each open link is between exactly two tasks: the passive task and one active task. A single task can, at the same time, be the active task of one link, and the passive task of another link.

In the examples described above, the computer at the warehouse runs the passive task. This task always has an open link, waiting for connections from the active tasks, run by the greengrocers. Once you link the two programs, however, there ceases to be any difference between active and passive tasks. Both active and passive tasks can send data, receive data or close the link.

These routines identify a logical link by its Network Logical Name (NLN). The NLN uniquely identifies the task to the routines. NLN is set automatically by the "open link" routine when it successfully finishes processing. When you call any of the other Task-to-Task routines: to send data, check the link or disconnect the link, you refer to the link by its Network Logical Name.

The Task-to-Task routines offer four open link routines:

- NFOPA (DIT$NFOPA on VMS)
  This routine opens an active link. It establishes a connection which permits you to transfer ASCII data between systems.

- NFOPB (DIT$NFOPB on VMS)
  This routine opens an active link. It establishes a connection which permits you to transfer binary data between systems.

- NFOP8 (DIT$NFOP8 on VMS)
  This routine opens an active link. It establishes a connection which permits you to transfer data that is stored in 8-bit bytes.

- NFOPP (DIT$NFOPP on VMS)
  This routine opens a passive link. NFOPP is always used in combination with the NFACC (DIT$NFACC) routine. First, the NFOPP routine opens the link. NFACC then accepts a network connection and specifies the type of data that you plan to move over the link.

The open link routine used by the active task requires you to pass information on the location of the target passive task (system name, object type and taskname of the passive task). The open link routine used by the passive task identifies the passive link and indicates that it is ready to accept network connections from active tasks. See Appendix D for further information on task identification.

The NFACC (DIT$NFACC on VMS) routine, called from the passive task, accepts the network connection requested by the active task.

To check the status of one or more network connections, use the NFGND (DIT$NFGND on VMS) routine. NFGND returns a value indicating the most recent event for the link, for example, a connect request has arrived, data is available or the link has aborted.

Once you have opened a link and connected the active and passive tasks, you can use the NFSND (DIT$NFSND on VMS) routine to send data and the NFRCV (DIT$NFRCV on VMS) routine to receive data. You can also use the NFINT (DIT$NFINT on VMS) to send an interrupt data message and NFRCI (DIT$NFRCI on VMS) to receive an interrupt data message. When the tasks have performed the necessary data transfer, close the link by issuing a call to the NFCLS (DIT$NFCLS on VMS) routine. The task that receives the last piece of data should be the first to close the link. When the other task notices the disconnect (by calling NFGND) it aborts its end of the link. This method of ending the communication between two tasks ensures that the last piece of data is processed.

The following figures show the order of calls to the Task-to-Task Routines that would be made by the shopkeeper(s) and the warehouse in the sample application discussed above.

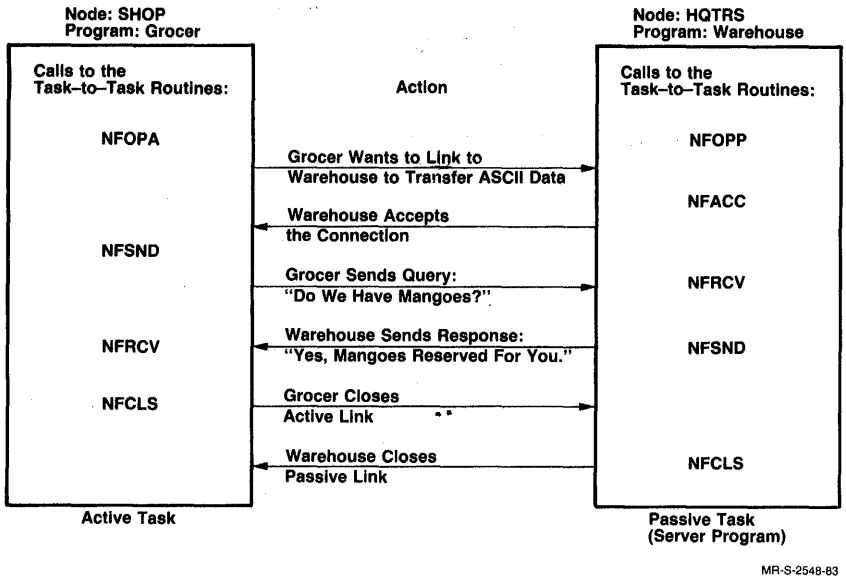


Figure 3-1: Simple Task-to-Task Application

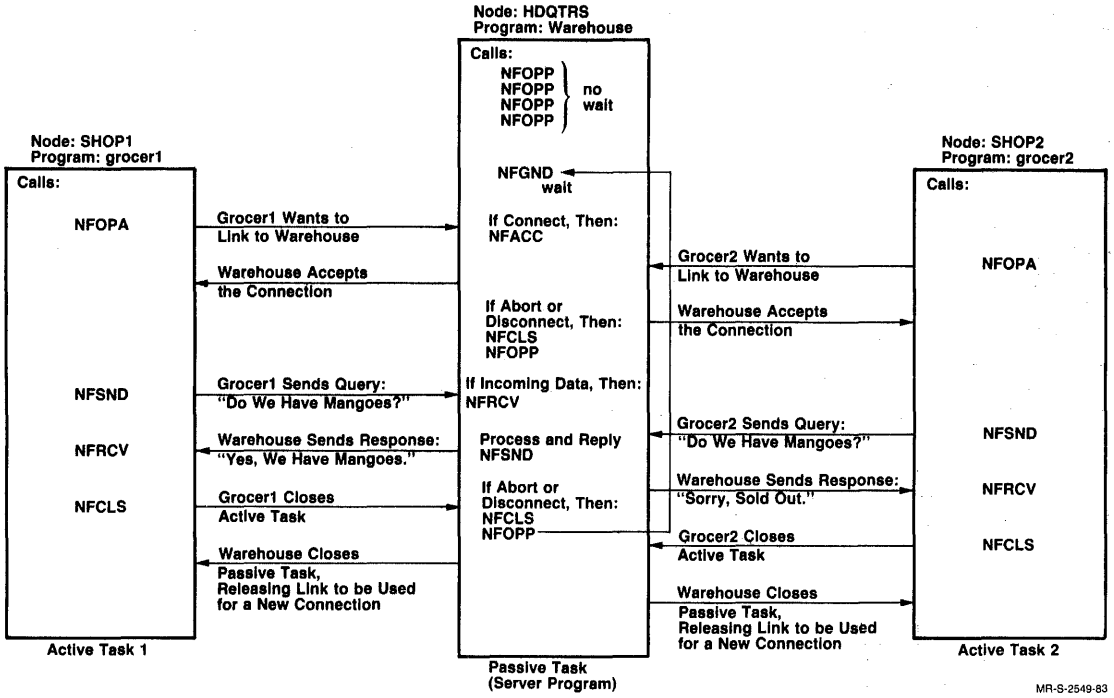


Figure 3-2: Task-to-Task Application

CHAPTER 4
REMOTE FILE ACCESS CONCEPTS AND CAPABILITIES

CHAPTER 4

REMOTE FILE ACCESS CONCEPTS AND CAPABILITIES

The Remote File Access (RFA) routines allow you to access and use records that reside in an ASCII sequential file on another computer. The RFA portion of the DIL consists of the following routines:

- ROPEN (DIT$ROPEN on VMS)
  The ROPEN routine opens an ASCII sequential file for reading, writing or appending

- RREAD (DIT$RREAD on VMS)
  The RREAD routine reads a record from an ASCII sequential file

- RWRITE (DIT$RWRITE on VMS)
  The RWRITE routine writes a record to an ASCII sequential file

- RCLOSE (DIT$RCLOSE on VMS)
  The RCLOSE routine closes an ASCII sequential file

- RDEL (DIT$RDEL on VMS)
  The RDEL routine deletes a closed ASCII sequential file

- RSUB (DIT$RSUB on VMS)
  The RSUB routine submits a closed ASCII sequential file

- RPRINT (DIT$RPRINT on VMS)
  The RPRINT routine prints a closed ASCII sequential file

If you are working on a VMS system, for example, you can access records in a sequential ASCII file on a TOPS-20 system. Since the file actually resides on another system, you have access to more data than would ordinarily be available to you as a VMS user. This means that if you have a large file on a TOPS-20 system, you can write reports from the file on your VMS system using records that you access with the RFA routines. When you use the RFA routines to access ASCII records, you don't have to worry about format translation. The RFA routines automatically perform any necessary data conversion between TOPS-10/TOPS-20 ASCII data and VMS ASCII data.

4.1 OPENING A REMOTE FILE

The ROPEN (DIT$ROPEN) routine opens a file on another system. ROPEN finds the proper system on the network and opens the correct file by using location information that you supply in your call to the ROPEN routine.

This location information is contained in the filename argument of the ROPEN routine. A valid network file name contains the following information:

- The node name of the system
- The file structure or device which contains the file
- The directory name
- The file name, type and optional version number

KL2116::PS:<MORRILL>IN.DAT is an example of a valid TOPS-20 file name. KL2116 is the node name. PS is the structure name. MORRILL is the directory name. IN.DAT is the file name and extension.

When you call any of the other RFA routines (to read a record, to write a record, or to close a file) you refer to the file by its file number, a unique value assigned by the ROPEN routine. You define a data item to store the file number, and use this data item as an argument in your call to the ROPEN routine. When ROPEN successfully finishes processing, it generates a file number value and places it in the data item.

The RFA routines work by talking to the File Access Listener (FAL) program on the remote system. To learn more about the FAL, see the DECnet manual for the remote system. You cannot use the RFA routines unless the FAL is available on the remote file's system. Before you attempt to access records with the RFA, check with the remote system manager to make sure that the FAL program is available and that it supports the operation that you want to perform.

CHAPTER 5
TOPS–10 AND TOPS–20 DATA CONVERSION

CHAPTER 5

TOPS-10 AND TOPS-20 DATA CONVERSION

5.1 DATA CONVERSION FROM TOPS-10/TOPS-20 COBOL

The information included in this section assumes

- You are writing a COBOL program
- You plan to use the program on a TOPS-10 or TOPS-20 system

To store a Foreign Field Descriptor, pass a record or perform an error check, you must represent several data items in your program. Users generally allocate space for foreign fields and records in WORKING-STORAGE.

5.1.1 Compiling Programs

To use the DIL Data Conversion Routines on TOPS-20 from a COBOL program, you may need to compile your program with the /STACK compiler switch to insure that you have an adequate pushdown list size. If your program gets a stack overflow, compile the program with /STACK:2000. TOPS-10 programmers should use the /D compiler switch. If a TOPS-10 COBOL program gets a stack overflow, compile the program with /D:2000.

5.1.2 Including the Interface Support Files

The Interface Support file provided for TOPS-10 and TOPS-20 COBOL is a copy library called DIL.LIB. You can use the COBOL COPY verb to retrieve the information at compilation time. There are three library elements in DIL.LIB. These elements are called DIL, DIT, and DIX.

The library element DIL defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task routines and the Remote File Access routines. The general success status code (DIL name SS-NORMAL) is defined in element DIL. Severity codes and system codes are defined in element DIL. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration:

```
COPY DIL OF "SYS:DIL.LIB".
```

In the following example, the DIL element of the library is retrieved and included in your program:

```
01  interface-files.
    COPY DIL OF "SYS:DIL.LIB".
```

The library element DIX defines codes specific to the data conversion routines. This includes the DIX status codes as well as data type names for each supported data type. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration, such as that shown above for the DIL element:

```
COPY DIX OF "SYS:DIL.LIB".
```

For programs which use the Data Conversion Routines, you must include both the DIL and DIX library elements.

5.1.3 Storing an FFD

An FFD occupies three full words of TOPS-10 or TOPS-20 memory. To store an FFD, you must define a data item with the following format:

```
01 your-ffd PIC S9(10) USAGE COMPUTATIONAL OCCURS 3 TIMES.
```

When you call the XDESCR routine to build the FFD, use your-ffd as the FFD to be returned. To pass the FFD to the routine, pass your-ffd with the subscript 1: your-ffd (1).

5.1.4 Passing a Record to the XDESCR Routine

To pass a record to the XDESCR routine, you must know how the record (containing the field that you want to convert) would be declared on its native system. You must then represent the record in your program on the local system. The record can be represented as any word-aligned group level data item. To pass the record to XDESCR, specify this group item as the record name ("rec") in the call to XDESCR.

To figure the size of the record, count the number of bits on its native system; make the record name on the foreign system at least that large. In the following example, the TOPS-20 record REC contains 720 bits of information.

```
01   rec   PIC S9(10)   USAGE COMPUTATIONAL OCCURS 20 TIMES.
```

To pass "rec" to the XDESCR routine simply include it in your call to the routine.

```
ENTER MACRO XDESCR USING ffd (1), rec, sysor, bysiz, byoff, bioff,

    type, lngth, scale.
```

5.1.5 Checking For Errors

To check for errors in a COBOL program on TOPS-10 or TOPS-20, you should define the following data items in WORKING-STORAGE:

```
01  ini-stat        PIC S9(10) USAGE COMPUTATIONAL.
01  dil-stat        PIC S9(10) USAGE COMPUTATIONAL.
01  dil-msg         PIC S9(10) USAGE COMPUTATIONAL.
01  dil-sev         PIC S9(10) USAGE COMPUTATIONAL.
    88  dil-warning         VALUE 0.
    88  dil-success         VALUE 1.
    88  dil-error           VALUE 2.
    88  dil-info            VALUE 3.
    88  dil-severe          VALUE 4.
    88  dil-ok              VALUES 1, 3.
```

You can choose your own names for the data items pictured above.

For the DIL to use these data items, you must first pass them to the DILINI routine. Before calling any other DIL routine, call the initialization routine, DILINI. DILINI tells the DIL what data items to use when it returns status information about a call to one of the other routines. Call the DILINI routine using the following format:

```
ENTER MACRO DILINI USING ini-stat, dil-stat, dil-sev,
   dil-msg.
```

The routine returns a value in ini-stat. If successful, the return value is 1. Any other value indicates either an error in the call to DILINI, or incorrect definition of dil-stat, dil-severity or dil-msg. If you plan to use the DIL from a COBOL subroutine, you must also pass the status data items to the subroutine: otherwise the subroutine cannot check for errors.

You only make this call to DILINI once in your program. All other calls to the conversion routines return their status codes in dil-stat. The severity portion of the status code is placed in dil-severity. A unique identifier for the condition is placed in dil-msg.

A call to the Conversion Routines with a simple check for success might look like this:

```
ENTER MACRO XDESCR USING ffd (1), rec, sysor, bysiz, byoff,
   bioff, type, lngth, scale.
IF NOT dil-ok
   DISPLAY "error status returned from XDESCR".
```

A call to the same routine with provisions for handling a specific type of error might look like this:

```
ENTER MACRO XDESCR USING ffd (1), rec, sysor, bysiz, byoff,
   bioff, type, lngth, scale.
IF NOT dil-ok
IF dil-msg = DIX-C-INVDATTYP
   DISPLAY "invalid data type specified"
ELSE
   DISPLAY "other error".
```

To determine which error occurred, compare dil-msg with the DIX condition identifiers defined in the TOPS-10/TOPS-20 COBOL Interface Support file. In the example above, DIX-C-INVDATTYP is the value (defined in the Interface Support file) indicating an invalid data type was specified.

5.2 DATA CONVERSION FROM TOPS-10/TOPS-20 FORTRAN

The information included in this section assumes

- You are writing a FORTRAN program
- You plan to use the program on a TOPS-10 or TOPS-20 system

The section explains methods to store a Foreign Field Descriptor, pass a record to the conversion routines and perform an error check.

5.2.1 Including the Interface Support Files

The Interface Support files provided for TOPS-10 and TOPS-20 FORTRAN are text files called DILV7.FOR, DITV7.FOR and DIXV7.FOR. The information from these files may be included into your source programs at compilation time using the FORTRAN INCLUDE statement.

The file DILV7 defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (DIL name SS-NORMAL) is defined in DILV7. Severity codes and system codes are defined in DILV7. To define these names in your program, include the following statement in your program:

```
INCLUDE 'SYS:DILV7'
```

The file DIXV7 defines codes specific to the Data Conversion Routines. These codes include the DIX status codes and data type names for each supported data type. To define these names in your program, include the following statement in your program:

```
INCLUDE 'SYS:DIXV7'
```

For programs which use the Data Conversion Routines, you must include both the DILV7 and DIXV7 files.

5.2.2 Storing an FFD

An FFD occupies three full words of TOPS-10 or TOPS-20 memory. To store an FFD, you must first dimension an array of type integer with length 3.

```
INTEGER    dilffd (3)
```

When you call the XDESCR routine to build an FFD, use dilffd as the FFD to be returned. To pass this value to the routine, pass the entire array as dilffd, not as dilffd (1).

5.2.3 Passing a Record to the XDESCR Routine

To convert a field in a record, you must know how the record would be declared on its native system. You must then declare the record in your program on the local system. You create a space for the record on the foreign system by declaring an integer array big enough to contain the record. To figure the array size, count the number of bits used by the record on its native system; make the array on the foreign system at least that large. In the following example, the DECSYSTEM-20 record contains 720 bits of information.

```
INTEGER    rec  (20)
```

To pass the record to one of the Data Conversion Routines, pass the entire array. The following example shows "rec" being passed to the XDESCR routine:

```
status = XDESCR (ffd, rec, sysor, bysiz, byoff, bioff, type,
1   lngth, scale)
```

5.2.4 Checking For Errors

Each Data Conversion Routine returns a one-word integer status value when it finishes processing. To perform an error check on a Data Conversion Routine, first declare an integer where the routine can place the status value.

```
      INTEGER     status
```

A simple call with an error check might then be:

```
         status = XDESCR (ffd, rec, sysor, bysiz, byoff, bioff, type,
         1   lngth, scale)
         IF ((status.AND.1).EQ.0) GO TO 20
         .
         .
         .
      20 WRITE (5,10)
      10 FORMAT ('error')
```

A call to the same routine with provisions for handling a specific type of error might look like this:

```
         status = XDESCR (ffd, rec, sysor, bysiz, byoff, bioff, dattyp,
         1   lngth, scale)
         IF (status.AND.1) GO TO 30
         IF (status.EQ.DATTYP) GO TO 20
      C  other error
         .
         .
         .
      20 WRITE (5,10)
      10 FORMAT ('invalid data type specified')
         .
         .
         .
      30 .
         .
      C  Success
         .
         .
         .
         .
```

To determine which error occurred, compare "status" to the DIX status codes defined in the TOPS-10/TOPS-20 FORTRAN Interface Support file. In the example above, DATTYP is the value (defined in the DIX Interface Support file) indicating that an invalid data type was specified.

5.3 TOPS-10/TOPS-20 DATA CONVERSION REFERENCE

5.3.1 DILINI - Allow the DIL to Recognize Status Codes

PURPOSE:

The DILINI routine allows status code data items in a TOPS-10 or TOPS-20 COBOL program to be recognized by the DIL. You do not need to use the DILINI routine if you plan to write a FORTRAN program.

You only call the DILINI routine once in your program, before you call any of the other DIL subroutines.

CALL FORMAT:

```
COBOL:    ENTER MACRO DILINI USING ini-stat, dil-stat, dil-msg,
              dil-sev.
```

where:

ini-stat is where the routine returns the results of the call to the DILINI routine. A return value of 1 indicates success. Any other value indicates an incorrect call or an incorrect definition of dil-stat, dil-msg or dil-sev.

dil-stat is a one-word integer that contains the status code for a call to any other DIL routine.

dil-msg is a one-word integer that contains the error message for a call to any other DIL routine.

dil-sev is a one-word integer that contains the severity portion of a DIL status code.

5.3.2 XDESCR - To Create an FFD

PURPOSE:

The XDESCR routine accepts the detailed description information which you supply and builds a Foreign Field Descriptor for a native or foreign field.

CALL FORMAT:

COBOL: ENTER MACRO XDESCR USING ffd (1), rec, sysor, bysiz, byoff, bioff, type, lngth, scale.

FORTRAN: status = XDESCR (ffd, rec, sysor, bysiz, byoff, bioff, 1 type, lngth, scale)

where:

ffd — is the Foreign Field Descriptor (FFD) to be returned. The argument represents the data item where the routine places the resultant FFD. An FFD consists of three one-word integers.

rec — is the record that contains the field to be described.

COBOL: This argument can be any word-aligned data item.

FORTRAN: This argument is usually an integer array.

sysor — is a one-word integer showing the system of origin of the record to be converted. Possible DIL Names for this argument are:

SYS-10-20 — for a record defined for TOPS-10 or TOPS-20.

SYS-VAX — for a record defined for VMS.

bysiz — is a one-word integer giving the byte size of the field to be described. All VMS fields have a byte size of 8. A TOPS-10 or TOPS-20 COBOL field can have a byte size of 6, 7, 9 or 36. A TOPS-10 or TOPS-20 FORTRAN field can have a byte size of 7 or 36. See Appendix A for further information.

byoff — is a one-word integer that gives the byte offset to the field within the record. The byte offset is the number of bytes in the record (of byte size specified in "bysiz") that precede the field that you want to convert.

bioff — is a one-word integer that gives the bit offset. This argument is not currently used; it should always be zero.

type — is a one-word integer that gives the data type of the field that is being converted. See Appendix A for a list of valid data type codes.

lngth — is a one-word integer that shows the length of the field: in characters for string fields. This argument is required for some data types; it must be zero for all other data types. See Appendix A for further information.

scale — is a one-word integer that gives the scale factor of the field. Scale factor indicates the number of decimal digits to move the decimal point to the left. A negative scale factor means that the decimal point will be moved to the right. You must specify a scale factor if you want to convert a fixed-point field. Specify a scale factor of zero for any other type of field.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-INVBYTSIZ	Invalid byte size specified.
DIX-ALIGN	Invalid alignment for data type.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

5.3.3 XCGEN - To Perform General Purpose Conversion

PURPOSE:

The XCGEN routine performs any type of data conversion that can be done by the DIL.

XCGEN is a general purpose routine. XCGEN accepts Foreign Field Descriptors for the source and destination fields and decides if it can perform that conversion. If the specified conversion is allowable, XCGEN converts the field.

NOTE

When you link a program that calls XCGEN with the DIL, this causes all of the conversion routines to be loaded. If you use one of the single function conversion routines (like XCVST), only the routines which perform the specified conversion are loaded.

CALL FORMAT:

```
COBOL:   ENTER MACRO XCGEN USING sffd (1), dffd (1).

FORTRAN: status = XCGEN (sffd, dffd)
```

where:

sffd — is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd — is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-INVALCHAR	Invalid character in source field or conversion table.
DIX-GRAPHIC	Graphic character changed in conversion.
DIX-FMTLOST	Format effector gained or lost in conversion.
DIX-NONPRINT	Non-printing character gained or lost in conversion.
DIX-TRUNC	String too long for destination -- truncated.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-UNSIGNED	Negative value moved to unsigned field.
DIX-ROUNDED	Result is rounded.
DIX-UNNORM	Floating-point number improperly normalized.
DIX-INVDNUMCHR	Invalid display numeric character in source field.
DIX-INVDNUMSGN	Invalid display numeric sign in source field.
DIX-INVPDDGT	Invalid packed decimal digit in source field.
DIX-INVPDSGN	Invalid packed decimal sign in source field.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

5.3.4 XCVST - Convert String Fields

PURPOSE:

The XCVST routine converts string fields.

CALL FORMAT:

COBOL: ENTER MACRO XCVST USING sffd (1), dffd (1).

FORTRAN: status = XCVST (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVALCHAR	Invalid character in source field or conversion table.
DIX-GRAPHIC	Graphic character changed in conversion.
DIX-FMTLOST	Format effector gained or lost in conversion.
DIX-NONPRINT	Non-printing character gained or lost in conversion.
DIX-TRUNC	String too long for destination -- truncated.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.5 XCVFB - Convert Fixed-Point Binary Fields

PURPOSE

The XCVFB routine converts fixed-point binary fields.

CALL FORMAT:

COBOL: ENTER MACRO XCVFB USING sffd (1), dffd (1).

FORTRAN: status = XCVFB (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-UNSIGNED	Negative value moved to unsigned field.
DIX-ROUNDED	Result is rounded.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.6 XCVFP - Convert Floating-Point Fields

PURPOSE:

The XCVFP routine converts floating-point fields.

CALL FORMAT:

COBOL: ENTER MACRO XCVFP USING sffd (1), dffd (1).

FORTRAN: status = XCVFP (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-ROUNDED	Result is rounded.
DIX-UNNORM	Floating-point number improperly normalized.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.7 XCVPD - Convert Packed Decimal Fields

PURPOSE:

The XCVPD routine converts packed decimal fields.

CALL FORMAT:

COBOL: ENTER MACRO XCVPD USING sffd (1), dffd (1).

FORTRAN: status = XCVPD (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routines. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-ROUNDED	Result is rounded.
DIX-INVPDDGT	Invalid packed decimal digit in source field.
DIX-INVPDSGN	Invalid packed decimal sign in source field.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.8 XCVDN - Convert Display Numeric Fields

PURPOSE:

The XCVDN routine converts display numeric fields.

CALL FORMAT:

COBOL: ENTER MACRO XCVDN USING sffd (1), dffd (1).

FORTRAN: status = XCVDN (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-ROUNDED	Result is rounded.
DIX-INVDNUMSGN	Invalid display numeric sign in source field.
DIX-INVDNUMCHR	Invalid display numeric character in source field.
DIX-UNSIGNED	Negative value moved to unsigned field.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.9 XCFBDN - Convert Fixed-Point Binary Fields to Display Numeric Fields

PURPOSE:

The XCFBDN routine converts Fixed-Point Binary fields to Display Numeric Fields.

CALL FORMAT:

COBOL: ENTER MACRO XCFBDN USING sffd (1), dffd (1).

FORTRAN: status = XCFBDN (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid argument for data type.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-ROUNDED	Result is rounded.
DIX-UNSIGNED	Negative value moved to unsigned field.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.10 XCFBPD - Convert Fixed-Point Binary Fields to Packed Decimal Fields

PURPOSE:

The XCFBPD routine converts Fixed-Point Binary fields to Packed Decimal Fields.

CALL FORMAT:

COBOL: ENTER MACRO XCFBPD USING sffd (1), dffd (1).

FORTRAN: status = XCFBPD (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-ROUNDED	Result is rounded.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.11 XCPDDN - Convert Packed Decimal Fields to Display Numeric Fields

PURPOSE:

The XCPDDN routine converts Packed Decimal fields to Display Numeric Fields.

CALL FORMAT:

COBOL: ENTER MACRO XCPDDN USING sffd (1), dffd (1).

FORTRAN: status = XCPDDN (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-ROUNDED	Result is rounded.
DIX-INVPDDGT	Invalid packed decimal digit in source field.
DIX-INVPDSGN	Invalid packed decimal sign in source field.
DIX-UNSIGNED	Negative value moved to unsigned field.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.12 XCPDFB - Convert Packed Decimal Fields to Fixed-Point Binary Fields

PURPOSE:

The XCPDFB routine converts Packed Decimal fields to Fixed-Point Binary Fields.

CALL FORMAT:

COBOL: ENTER MACRO XCPDFB USING sffd (1), dffd (1).

FORTRAN: status = XCPDFB (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-ROUNDED	Result is rounded.
DIX-INVPDDGT	Invalid packed decimal digit in source field.
DIX-INVPDSGN	Invalid packed decimal sign in source field.
DIX-UNSIGNED	Negative value moved to unsigned field.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.13 XCDNPD - Convert Display Numeric Fields to Packed Decimal Fields

PURPOSE:

The XCDNPD routine converts Display Numeric fields to Packed Decimal Fields.

CALL FORMAT:

COBOL: ENTER MACRO XCDNPD USING sffd (1), dffd (1).

FORTRAN: status = XCDNPD (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-ROUNDED	Result is rounded.
DIX-INVDNUMSGN	Invalid display numeric sign in source field.
DIX-INVDNUMCHR	Invalid display numeric character in source field.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.14 XCDNFB - Convert Display Numeric Fields to Fixed-Point Binary Fields

PURPOSE:

The XCDNFB routine converts Display Numeric fields to Fixed-Point Binary Fields.

CALL FORMAT:

COBOL: ENTER MACRO XCDNFB USING sffd (1), dffd (1).

FORTRAN: status = XCDNFB (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. This argument consists of three one-word integers.

dffd is a Foreign Field Descriptor describing the destination field. This argument consists of three one-word integers.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-ROUNDED	Result is rounded.
DIX-INVDNUMSGN	Invalid display numeric sign in source field.
DIX-INVDNUMCHR	Invalid display numeric character in source field.
DIX-UNSIGNED	Negative value moved to unsigned field.

RELATED ROUTINES:

CVGEN: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 5.3.15 for a description of CVGEN.

XCGEN: This general purpose routine performs any type of conversion allowed by the Data Conversion Routines. See Section 5.3.3 for a description of XCGEN.

5.3.15 CVGEN - Perform Conversion Without an FFD

PURPOSE:

The CVGEN routine allows you to convert a field without making an FFD for the field. CVGEN requires a detailed series of arguments; you must specify parameters for both the source and destination fields.

You should only use CVGEN in cases where the field will be converted a limited number of times or the program is table-driven. CVGEN creates a description for the field each time it processes the field. If you plan to convert a field many times during a run, it is quicker to create an FFD for the field and convert it with one of the other single function conversion routines. Whereas CVGEN creates an FFD internally each time it processes, if you perform the conversion with an FFD (built using the XDESCR routine) you make an FFD only once.

CALL FORMAT:

COBOL:
```
ENTER MACRO CVGEN USING srec, ssysor, sbysiz, sbyoff,
    sbioff, stype, slngth, sscale, drec, dsysor,
    dbysiz, dbyoff, dbioff, dtype, dlngth, dscale.
```

FORTRAN:
```
status = CVGEN (srec, ssysor, sbysiz, sbyoff, sbioff,
1   stype, slngth, sscale, drec, dsysor, dbysiz,
2   dbyoff, dbioff, dtype, dlngth, dscale)
```

where:

srec — is the source record that contains the field to be described.

COBOL: This field can be any word-aligned data item.

FORTRAN: This argument will usually be an integer array.

ssysor — is a one-word integer giving the system of origin of the source record. Possible DIL Names for this argument are:

SYS-10-20 — for a record defined for TOPS-10 or TOPS-20.

SYS-VAX — for a record defined for VMS.

sbysiz — is a one-word integer giving the byte size of the source field to be described. All VMS fields have a byte size of 8. A TOPS-10 or TOPS-20 COBOL field can have a byte size or 6, 7, 9 or 36. A TOPS-10 or TOPS-20 FORTRAN field can have a byte size of 7 or 36. See Appendix A for further information.

sbyoff — is a one-word integer giving the byte offset to the field within the source record. The byte offset is the number of bytes in the source record (of byte size specified in "sbysiz") that precede the field that you want to convert.

sbioff is a one-word integer giving the bit offset. This argument is not currently necessary; it should be zero.

stype is a one-word integer giving the data type of the source field. See Appendix A for a list of valid data type codes.

slngth is a one-word integer giving the length of the source field: in characters for string fields. This argument is required for some data types; it must be zero for all other data types. See Appendix A for further information.

sscale is a one-word integer giving the scale factor of the source field. It indicates the number of decimal digits to move the decimal point to the left. A negative scale factor means that the decimal point will be moved to the right. You must specify a scale factor if you want to convert a fixed-point field. Specify a scale factor of zero for any other type of field.

drec is the destination record which contains the field to be described.

COBOL: This argument can be any word-aligned data item.

FORTRAN: This argument will usually be an integer array.

dsysor is a one-word integer showing the system of origin of the destination record. Possible DIL Names for this argument are:

SYS-10-20 for a record defined TOPS-10 or TOPS-20.

SYS-VAX for a record defined for VMS.

dbysiz is a one-word integer giving the byte size of the destination field to be described. All VMS fields have a byte size of 8. A TOPS-10 or TOPS-20 COBOL field can have a byte size of 6, 7, 9 or 36. A TOPS-10 or TOPS-20 FORTRAN field can have a byte size of 7 or 36. See Appendix A for further information.

dbyoff is a one-word integer giving the byte offset of the destination record. The byte offset is the number of bytes in the record (of byte size specified in "dbysiz") that precede the field that you want to convert.

dbioff is a one-word integer that shows the bit offset. This argument is not currently used; it is always zero.

dtype is a one-word integer giving the data type of the destination field. See Appendix A for a list of valid data type codes.

dlngth is a one-word integer giving the length of the destination field: in characters for string fields. This argument is required for some data types, it must be zero for all other data types. See Appendix A for further information.

dscale is a one-word integer giving the scale factor of the destination field. It indicates the number of decimal digits to move the decimal point to the left. A negative scale factor means that the decimal point will be moved to the right. You must specify a scale factor if you want to convert a fixed-point field. Specify a scale factor of zero for any other type of field.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-INVBYSIZ	Invalid byte size specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-INVALCHAR	Invalid character in source field or conversion table.
DIX-GRAPHIC	Graphic character changed in conversion.
DIX-FMTLOST	Format effector gained or lost in conversion.
DIX-NONPRINT	Non-printing character gained or lost in conversion.
DIX-TRUNC	String too long for destination -- truncated.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-UNSIGNED	Negative value moved to unsigned field.
DIX-ROUNDED	Result is rounded.
DIX-UNNORM	Floating-point number improperly normalized.
DIX-INVDNUMCHR	Invalid display numeric character in source field.
DIX-INVDNUMSGN	Invalid display numeric sign in source field.
DIX-INVPDDGT	Invalid packed decimal digit in source field.
DIX-INVPDSGN	Invalid packed decimal sign in source field.

RELATED ROUTINES:

XDESCR: This routine builds a Foreign Field Descriptor for the field you wish to convert. See Section 5.3.2 for a description of XDESCR.

5.4 TOPS-10/TOPS-20 DATA CONVERSION EXAMPLES

5.4.1 TOPS-10/TOPS-20 COBOL Data Conversion Example

```
IDENTIFICATION DIVISION.

PROGRAM-ID.

        CDCR36.

        This program performs  a single string  data conversion.  The
        ASCII-7 string value  "ABCDE" will  be converted  to the  same
        ASCII-8 value.


ENVIRONMENT DIVISION.

.

.

.

DATA DIVISION.

WORKING-STORAGE SECTION.

* source data field

01 SRCDAT PIC X(5) USAGE DISPLAY-7 VALUE "ABCDE".

* The destination data field must be large enough to hold the ASCII-8
* equivalent of the source data field:
*   ASCII-7 PIC X(5) = 7 bits/character * 5 characters = 35 bits
*   ASCII-8 PIC X(5) = 8 bits/character * 5 characters = 40 bits

01  DSTDAT PIC S9(10) COMP OCCURS 2.

* The value of the destination data, when considered as a numeric  bit
* pattern,  contains  a  large  number  (22620095041)  which  is  not
* acceptable to COBOL.  Therefore we will  use the SIXBIT  equivalents
* for the destination data fields:

01  SIXBIT-EQUIVALENTS REDEFINES DSTDAT.
    05  SIXBIT-EQUIV1 PIC X(6) USAGE DISPLAY-6.
    05  SIXBIT-EQUIV2 PIC X(6) USAGE DISPLAY-6.

01  EXPECTED-VALUES.
    05  EXPECTED-VAL1 PIC X(6) USAGE DISPLAY-6 VALUE "5$0T)!".
    05  EXPECTED-VAL2 PIC X(6) USAGE DISPLAY-6 VALUE "     $".
```

```
* foreign field descriptors

01  FFDS.
    05  SRCFFD PIC S9(10) COMP OCCURS 3.
    05  DSTFFD PIC S9(10) COMP OCCURS 3.

01  INTERFACE-FILES.
    COPY DIL OF "SYS:DIL.LIB".
    COPY DIX OF "SYS:DIL.LIB".

01  DILINI-PARAMS.
    05  DIL-INIT-STATUS PIC S9(10) COMP.
    05  DIL-STATUS PIC S9(10) COMP.
    05  DIL-MESSAGE PIC S9(10) COMP.
    05  DIL-SEVERITY PIC S9(10) COMP.


PROCEDURE DIVISION.

INITIALIZE-STUFF.
* Set up for status code values, using DILINI routine

    ENTER MACRO DILINI USING DIL-INIT-STATUS, DIL-STATUS,
                             DIL-MESSAGE, DIL-SEVERITY.

    IF DIL-INIT-STATUS NOT = 1
        DISPLAY "? Failure in DILINI. Dil-status = " DIL-STATUS.

* initialize destination data field to zeros

    MOVE 0 TO DSTDAT(1).
    MOVE 0 TO DSTDAT(2).

MAKE-FFDS.
* make the foreign field descriptors for use by XCVST

    ENTER MACRO XDESCR USING SRCFFD(1), SRCDAT, DIX-SYS-10-20, 7, 0, 0,
                             DIX-DT-ASCII-7, 5, 0.

    IF DIL-SEVERITY NOT = STS-K-SUCCESS AND
       DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? Failure in XDESCR.  Dil-Status = " DIL-STATUS
        STOP RUN.

    ENTER MACRO XDESCR USING DSTFFD(1), DSTDAT(1), DIX-SYS-VAX, 8, 0, 0,
                             DIX-DT-ASCII-8, 5, 0.

    IF DIL-SEVERITY NOT = STS-K-SUCCESS AND
       DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? Failure in XDESCR.  Dil-status = " DIL-STATUS
        STOP RUN.

DO-CONVERSION.
* Convert ASCII-7 value "ABCDE" to ASCII-8 value "ABCDE".

    ENTER MACRO XCVST USING SRCFFD(1), DSTFFD(1).

    IF DIL-SEVERITY NOT = STS-K-SUCCESS AND
       DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? Failure in XCVST.  Dil-status = " DIL-STATUS
        STOP RUN.
```

```
CHECK-RESULTS.
* What should have been created is the VAX ASCII-8 form of the source
* value "ABCDE".
*
* In VAX memory, this is represented as follows:
*
*
*       symbolic representation:        numeric (binary) representation:
*               |AAAAAAAA| :m                    |01000001| :m
*               |BBBBBBBB| :m+1                  |01000010| :m+1
*               |CCCCCCCC| :m+2                  |01000011| :m+2
*               |DDDDDDDD| :m+3                  |01000100| :m+3
*               |EEEEEEEE| :m+4                  |01000101| :m+4
*
* Transposing this into DEC-20 memory we have:
*
*       symbolic representation:
* |EEEEDDDDDDDDCCCCCCCCBBBBBBBBAAAAAAAA| : n
* |                                EEEE| : n+1
*
*       numeric (binary) representation:
* |01010100010001000011010000100100000l| n
* |                                0100| n+1
*
* in octal this is:             and in decimal:
*
* |250420641101| n              | 22620095041 | :n
* |000000000004| n+1            |       4     | :n+1
*
* Since this large number (22620095041) is not acceptable to COBOL, we will
* use the SIXBIT equivalents for the destination data fields:
*       5$0T)!  :n
*            $  :n+1
*
    IF SIXBIT-EQUIV1 NOT EQUAL EXPECTED-VAL1
        DISPLAY "? Error in conversion:  "
        DISPLAY " expected converted value not returned from conversion"
        STOP RUN.

    IF SIXBIT-EQUIV2 NOT EQUAL EXPECTED-VAL2
        DISPLAY "? Error in conversion:  "
        DISPLAY " expected converted value not returned from conversion"
        STOP RUN.

    DISPLAY " CDCR36 successfully completed.".

    STOP RUN.
```

5.4.2 TOPS-10/TOPS-20 FORTRAN Data Conversion Example

```
C FDCR36

C       This program performs  a single string  data conversion.   The
C       ASCII-7 string value  "ABCDE" will  be converted  to the  same
C       ASCII-8 value.

C Include interface files
        INCLUDE 'SYS:DIXV7'
        INCLUDE 'SYS:DILV7'

C Source and destination data fields.
C NOTE: The destination data field must be large enough to hold the
C ASCII-8 equivalent of the source data field:
C   ASCII-7 (5 chars) = 7 bits/character * 5 characters = 35 bits
C   ASCII-8 (5 chars) = 8 bits/character * 5 characters = 40 bits

        INTEGER SRCDAT (1), DSTDAT (2)

C Foreign field descriptors (ffds)

        INTEGER SRCFFD (3), DSTFFD (3)

C Status return code
        INTEGER DILSTS

C Data for source and destination data fields
        DATA SRCDAT /'ABCDE'/
        DATA DSTDAT /0, 0/

C make the foreign field descriptors for use by XCVST

        DILSTS = XDESCR (SRCFFD, SRCDAT, SYS36, 7, 0, 0, ASCII7, 5, 0)
        IF (DILSTS.NE.NORMAL) GOTO 100

        DILSTS = XDESCR (DSTFFD, DSTDAT, SYSVAX, 8, 0, 0, ASCII8, 5, 0)
        IF (DILSTS.NE.NORMAL) GOTO 100

C Do conversions:  convert ASCII-7 value "ABCDE" to ASCII-8 value "ABCDE"

        DILSTS = XCVST (SRCFFD, DSTFFD)
        IF (DILSTS.NE.NORMAL) GOTO 102

C Check results:

C What should have been created is the VAX ASCII-8 form of the source
C value "ABCDE".
C
C In VAX memory, this is represented as follows:
C
C
C       symbolic representation:        numeric (binary) representation:
C               |AAAAAAAA| :m                    |01000001| :m
C               |BBBBBBBB| :m+1                  |01000010| :m+1
C               |CCCCCCCC| :m+2                  |01000011| :m+2
C               |DDDDDDDD| :m+3                  |01000100| :m+3
C               |EEEEEEEE| :m+4                  |01000101| :m+4
C
```

```
C Transposing this into DEC-20 memory we have:
C
C       symbolic representation:
C |EEEEDDDDDDDDCCCCCCCCBBBBBBBBAAAAAAAA| : n
C |                                EEEE| : n+1
C
C       numeric (binary) representation:
C |01010100010001000011010000100100000l| n
C |                                0100| n+1
C
C in octal this is:              and in decimal:
C
C |250420641101| n                | 22620095041 | :n
C |000000000004| n+1              |       4     | :n+1
C

        IF (DSTDAT (1) .NEQ. 22620095041) GOTO 104
        IF (DSTDAT (2) .NEQ. 4) GOTO 104
200     FORMAT (' FDCR36 successfully completed.')
        WRITE (5, 200)
        STOP

C Print error information
100     WRITE (5, 101) DILSTS
101     FORMAT ('? Failure in XDESCR.  Dil-status = ', I10)
        STOP

102     WRITE (5, 103) DILSTS
103     FORMAT ('? Failure in XCVST.  Dil-status = ', I10)
        STOP

104     WRITE (5, 105)
105     FORMAT ('? Error in conversion:')
        WRITE (5, 106)
106     FORMAT (' expected converted value not retured from conversion')
        STOP

        END
```

CHAPTER 6
TOPS–20 AND TOPS–10 TASK–TO–TASK

CHAPTER 6

TOPS-20 AND TOPS-10 TASK-TO-TASK

6.1 TASK-TO-TASK FROM TOPS-20 OR TOPS-10 COBOL

The information included in this section assumes

-You are writing a COBOL program
-You plan to use the program on a TOPS-20 or TOPS-10 system

To store a Network Logical Name (NLN), task characteristics or user attributes, to send data or to perform a status check you must represent several data items in your program. Users generally allocate space for these data items in WORKING-STORAGE.

6.1.1 Compiling Programs

To use the DIL Data Conversion Routines on TOPS-20 from a COBOL program, you may need to compile your program with the /STACK compiler switch to insure that you have an adequate pushdown list size. If your program gets a stack overflow, compile the program with /STACK:2000. On TOPS-10, you may need to compile your COBOL program with the /D compiler switch. If your program gets a stack overflow, compile it with /D:2000.

6.1.2 Including the Interface Support Files

The Interface Support file provided for TOPS-10 and TOPS-20 COBOL is a copy library called DIL.LIB. The COBOL COPY verb can be used to retrieve the information in the library at compilation time. There are three library elements in DIL.LIB. The elements are DIL, DIT, and DIX.

The library element DIL defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL. Severity codes and system codes are defined in element DIL. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration:

```
COPY DIL OF "SYS:DIL.LIB".
```

In the following example, the DIL element of the library is retrieved and included in your program:

```
01  interface-files.
    COPY DIL OF "SYS:DIL.LIB".
```

The library element DIT defines codes specific to the Task-to-Task and Remote File Access Routines. These codes include the DIT condition values Task-to-Task wait codes, Task-to-Task link types, Task-to-Task message modes and VMS task fire-up codes. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration, such as that shown above for the DIL element:

```
COPY DIT OF "SYS:DIL.LIB".
```

For programs which use the Task-to-Task routines, you must include both the DIL and DIT library elements.

6.1.3 Storing a Network Logical Name (NLN)

The NLN consists of one word of TOPS-10/TOPS-20 memory. To store an NLN, you must define a data item with the following format:

```
01 your-nln PIC     S9(10) USAGE COMPUTATIONAL.
```

When you call the NFOPA, NFOPB, NFOP8 or NFOPP routine to create the NLN, use your-nln as the NLN to be returned. When the routine successfully finishes processing, it returns a value to your-nln.

6.1.4 Storing Task and User Attributes

To include task and user attributes in a call to NFOPA, NFOPB or NFOP8 or task attributes in a call to NFOPP you must describe these attributes as data items in WORKING-STORAGE.

Task attributes are always DISPLAY-7 fields; they always have the picture clause PIC X(16). The format for the task attributes is as follows:

```
01  target-name     PIC X(16) DISPLAY-7.
01  object-type     PIC X(16) DISPLAY-7.
01  desc-name       PIC X(16) DISPLAY-7.
01  task-name       PIC X(16) DISPLAY-7.
```

User attributes are always DISPLAY-7 fields; they always have the picture clause PIC X(39). The format for the user attributes is as follows:

```
01  userid          PIC X(39) DISPLAY-7.
01  passwd          PIC X(39) DISPLAY-7.
01  acct            PIC X(39) DISPLAY-7.
```

6.1.5 Checking the Status of a Task-to-Task Routine

Section 5.1.5 of this manual presents a method for checking the status of any TOPS-10/TOPS-20 COBOL DIL Routine. Section 5.3.1 describes the DILINI initialization routine.

A call to the Task-to-Task Routines with a simple check for success might look like this:

```
ENTER MACRO NFOPA USING nln, trgsys, objtyp, desc, tsknam,
   userid, passwd, acct, usdat, wait.
IF NOT dil-ok
   DISPLAY "fatal error".
```

A call to the same routine with provisions for handling a specific type of error might look like this:

```
ENTER MACRO NFOPA USING nln, trgsys, objtyp,
   desc, tsknam, userid, passwd, acct, usdat,
   wait
IF NOT dil-ok
   IF dil-msg = DIT-C-INVARG
      DISPLAY "invalid data type error".
   ELSE
      DISPLAY "other error".
```

To determine which error occurred, compare dil-msg to the DIT condition identifier defined in the TOPS-10/TOPS-20 COBOL Interface Support files. In the example above, DIT-C-INVARG is the value (defined in the Interface Support file) indicating an invalid argument was specified.

6.1.6 The TOPS-10 Software Interrupt System

The DIL task-to-task and remote file access routines need the services of the TOPS-10 software interrupt system (PSI). Because the FORTRAN and COBOL runtime systems do not provide a facility to share the use of the software interrupt system between non-cooperating routines in the same program, the user cannot use the software interrupt system in a COBOL or FORTRAN program that uses the task-to-task or remote file access routines.

6.2 TASK-TO-TASK FROM TOPS-20 AND TOPS-10 FORTRAN

The information included in this section assumes

- -You are writing a FORTRAN program
- -You plan to use the program on a TOPS-20 or TOPS-10 system

This section explains how to store a Network Logical Name, task characteristics or user attributes, to send data or to perform a status check.

6.2.1 Including the Interface Support Files

The Interface Support files provided for TOPS-10 and TOPS-20 FORTRAN are text files called DILV7.FOR, DITV7.FOR and DIXV7.FOR. You can include the information from these files into your source programs at compilation time using the FORTRAN INCLUDE statement.

The file DILV7 defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in DILV7. Severity codes and system codes are defined in DILV7. To define these names in your program, include the following statement in your program:

```
INCLUDE 'SYS:DILV7'
```

The file DITV7 defines codes specific to the Task-to-Task and Remote File Access Routines. These codes include the DIT status codes, Task-to-Task wait codes, Task-to-Task link types, Task-to-Task message modes and VMS task fire-up codes. To define these names in your program, include the following statement:

```
INCLUDE 'SYS:DITV7'
```

For programs which use the Task-to-Task routines, you must include both the DILV7 and DITV7 files.

6.2.2 Storing a Network Logical Name (NLN)

The NLN consists of one word of TOPS-10/TOPS-20 memory. To store an NLN, you must declare an integer, as shown below:

```
INTEGER nln
```

6.2.3 Storing Task and User Attributes

To include task and user attributes in a call to NFOPA, NFOPB or NFOP8 or task attributes in a call to NFOPP you must describe these attributes in your program as follows:

```
INTEGER trgsys (4)    or  CHARACTER*16 trgsys
INTEGER objtyp (4)    or  CHARACTER*16 objtyp
INTEGER desc (4)      or  CHARACTER*16 desc
INTEGER tsknam (4)    or  CHARACTER*16 tsknam
INTEGER userid (8)    or  CHARACTER*39 userid
INTEGER passwd (8)    or  CHARACTER*39 passwd
INTEGER acct (8)      or  CHARACTER*39 acct
INTEGER usdat (4)     or  CHARACTER*16 usdat
```

6.2.4 Checking the Status of a Task-to-Task Routine

To check the status of a call to one of the task-to task routines from a TOPS-20 or TOPS-10 FORTRAN program, you should first declare an integer (implicitly or explicitly) where the routine can place a status value.

```
      INTEGER    status
```

A normal call to one of the Task-to-Task Routines (using the proper Interface Support File), including provisions for status checking might be:

```
            status = NFOP8 (nln, trgsys, objtyp, desc, tsknam, userid,
     1                      passwd, acct, usdat, wait)
            IF (status.EQ.NORMAL) GOTO 100
            IF (status.EQ.INVARG) GOTO 50
            TYPE 10
     10     FORMAT (' error occurred')
      .
      .
      .
     50     TYPE 51
     51     FORMAT (' invalid data type error')

     100
      .
      .
      .
```

To determine which error occurred, compare status to the DIT status codes defined in the TOPS-10/TOPS-20 FORTRAN Interface Support file. In the example above, NORMAL and INVARG are status code values defined in the Interface Support files.

6.2.5 The TOPS-10 Software Interrupt System

The DIL task-to-task and remote file access routines need the services of the TOPS-10 software interrupt system (PSI). Because the FORTRAN and COBOL runtime systems do not provide a facility to share the use of the software interrupt system between non-cooperating routines in the same program, the user cannot use the software interrupt system in a COBOL or FORTRAN program that uses the task-to-task or remote file access routines.

6.3 TOPS-10 AND TOPS-20 TASK-TO-TASK REFERENCE

6.3.1 NFGND - Return the Status of a Link

PURPOSE:

The NFGND routine returns the status of a specific network connection or the next network event (such as a connection, a disconnect or an abort).

CALL FORMAT:

COBOL: ENTER MACRO NFGND USING nln, wait.

FORTRAN: status = NFGND (nln, wait)

where:

nln is the Network Logical Name of the link that you want information about. The NLN is set by the NFOPA, NFOPB, NFOP8 or NFOPP routines. The Network Logical Name is a one-word integer.

If you want information about any event occurring on any logical link, use -1 as the value for this argument. NFGND finds the next network event and returns the NLN of that link. If NFGND cannot find an event, it returns an undefined value.

wait is a one-word integer that gives the wait code.

Set the wait code to "no" if you want the routine to return only the current status of events on the specified link. The DIL Name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want the routine to wait for a network event to occur involving the specified link. When an event occurs the routine reports it. Waiting uses minimal CPU time. The DIL name for this argument is:

WAIT-YES

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value of status.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing. You receive this code only if you don't wait for a new event and no events have occured since the last reported event. If an event has taken place, (connect, data, abort, disconnect, interrupt data) you receive the code for that event.
DIT-CONNECTEVENT	This code is returned for a connect event. If you are checking the status of a passive task, this code indicates that the task has received a connect request. If you are checking the status of an active task, this code indicates that a connect request issued by the active task has been accepted by the passive task.
DIT-ABREJEVENT	The routine returns this code if the link is aborted or rejected. You should call NFCLS to do an abort and release the resources of this link, so it can be used again.
DIT-DATAEVENT	The routine returns this code when data is available over the specified link. You should call NFRCV to receive the data.
DIT-DISCONNECTEVENT	The routine returns this code when the specified link has been disconnected. You should call NFCLS to do an abort to release the resources of this link, so it can be used again.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.
DIT-INTERRUPT	The routine returns this code when an interrupt data message is available. You should call NFRCI to read the interrupt data message.

6.3.2 NFINF - Get Information About the Other End of a Logical Link.

PURPOSE:

The NFINF routine returns information about the remote node, the remote process or the remote DECnet object associated with a specific network connection.

You can also use the NFINF routine to read optional data sent by a remote process. If the cooperating process is on a VMS system, you can use NFINF to read optional data associated with a disconnection or rejection of a process. If the cooperating process is a TOPS-10 or TOPS-20 system, you can read any type of optional data.

CALL FORMAT:

COBOL: ENTER MACRO NFINF USING nln, inftyp, length, buffer.

FORTRAN: status = NFINF (nln, inftyp, length, buffer)

where:

nln is the Network Logical Name of the link that you want information about. The NLN is set by the NFOPA, NFOPB, NFOP8, or NFOPP routine. The Network Logical Name is a one-word integer.

inftyp is a one-word integer that specifies the type of information wanted.

Refer to the DECnet User's Guide for your system for more information.

Information Type	DIL Name
Remote node name of the cooperating task.	INF-NODE
Remote object type. This information is only available to the passive task.	INF-OBJECT
Remote object descriptor format (0 if the task only requires an object id, 1 if the task only requires a taskname, or 2 if the task requires a project-programmer number). This information is only available to the passive task.	INF-DESCF
Remote DECnet object descriptor. This information is only available to the passive task.	INF-DESC
Remote process user id. This information is only available to the passive task.	INF-USERID
Remote process password. This information is only available to the passive task.	INF-PASSWD
Remote process account. This information is only available to the passive task.	INF-ACCT
Remote process optional data or disconnect optional data or reject optional data. If the cooperating task is running on a VMS system, only disconnect and reject optional data are available.	INF-OPT
Maximum segment size for the link in bytes. This is not available for VMS systems. The information can be used to determine the optimum size of records to be transmitted over the link.	INF-SEG
Abort code or reject code. You can find the meaning of the abort or reject code in the DECnet manual for the remote system.	INF-ABTCOD

length is a one word integer in which the length of ASCII data returned is specified.

buffer is the area in which to place returned ASCII data. This area must be at least 16 ASCII-7 characters long to accomodate optional data, a taskname, or a DECnet descriptor. It must be at least 39 ASCII-7 characters long to accommodate a userid, a password, or an account.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-INFONOTAVAIL	The information you have requested is not available to this task.
DIT-INFOOUTOFRANGE	The information you have requested is not in the range of valid values.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

6.3.3 NFOPA - Open a Link From an Active Task (ASCII)

PURPOSE:

The NFOPA routine opens a logical link to a program on another system. You use this routine when you intend to transmit or receive ASCII data.

CALL FORMAT:

COBOL: ENTER MACRO NFOPA USING nln, trgsys, objtyp, desc, tsknam, userid, passwd, acct, usdat, wait.

FORTRAN: status = NFOPA (nln, trgsys, objtyp, desc, tsknam, 1 userid, passwd, acct, usdat, wait)

where:

nln is the Network Logical Name (NLN) to be returned when this routine successfully finishes processing. You use the NLN to identify this link when you call other Task-to-Task routines. The Network Logical Name is a one-word integer.

trgsys is the node name of the target system. The target system, in this case, is the system which runs the passive task that you want to access with this link.

The node name has a length of sixteen ASCII-7 characters. If your node name is less than sixteen characters, left-justify the field. If you give this argument a value of spaces, it defaults to the local system's node name.

objtyp is the object type of the passive task. The object type specifies the kind of service performed by the passive task. This argument has a length of sixteen ASCII-7 characters. If the object type has less than sixteen characters, left-justify the field.

The object type can be expressed as either a number or a name. Most programs use an object type of 0 or TASK. Server programs which perform a generic service (MAIL, for example) have a non-zero numeric object type. You can find a list of valid DECnet object types and their meanings in the appropriate DECnet User's Guide.

desc is the DECnet descriptor. You must use a descriptor when you plan to access a task with object type 0 or object name TASK. The descriptor must contain the DECnet taskname of the passive task on the remote system. For TOPS-20 to TOPS-20 communication when the object type is not zero, you can provide a descriptor. See Appendix D for further information. The descriptor has a length of sixteen ASCII-7 characters. If your descriptor is less than sixteen characters, left-justify the field.

NOTE

For a TOPS-20 system, the name of the target system, the object type and the DECnet descriptor cannot have a combined total of more than thirty-seven non-blank characters.

tsknam is the DECnet taskname. Taskname is a unique sixteen character ASCII-7 string that identifies this process to the network. An active task never has to specify a taskname. If you pass a value of spaces for this argument, the operating system assigns a unique name.

NOTE

The following three arguments are optional user attributes. The passive task may use these attributes to validate a network connection, or to perform any other recognition function agreed to by both tasks. These arguments are optional if you want to connect to a passive TOPS-20 or TOPS-10 task, or a passive VMS task that is already running. The arguments are required if you want to connect to a passive VMS task that is to be started as a result of your request.

userid is your userid. Userid has a length of thirty-nine ASCII-7 characters. If your userid is less than thirty-nine characters, left-justify the field. If you don't want to specify a userid, pass a value of spaces for this argument.

passwd is your password. Password has a length of thirty-nine ASCII-7 characters. If your password is less than thirty-nine characters, left-justify the field. If you don't want to specify a password, pass a value of spaces for this argument.

acct is your account number or charge code. This field has a length thirty-nine ASCII-7 characters. If this information is less than thirty-nine characters, left-justify the field. Give this argument a value of spaces if you plan to connect to a passive task on a VMS system or you don't want to specify an account number.

usdat is sixteen optional ASCII-7 characters of user data. See the NFINF routine for more information.

wait is a one-word integer that gives the wait code.

Set the wait code to "no" if you do not want your program to wait until it establishes a connection to the passive task. Using this code enables your program to perform other duties while waiting for the network connection. To find out if the passive task has accepted your connection, periodically call the NGFND routine to check status. The DIL Name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want your program to wait until the passive task has accepted your link. The routine does not return to your program until it establishes the specified link. While it waits, you can not use the active task. Waiting uses minimal CPU time. The DIL Name for this argument is:

WAIT-YES

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value of status.

DIL Name	Meaning
DIT-TOOMANY	You attempted too many links. The DIL allows a maximum of 20 open links. On TOPS-20 systems, however, a non-privileged user can only open a maximum of four links.
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-ABORTREJECT	The link was aborted or rejected.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

6.3.4 NFOPB - Open a Link From an Active Task (Binary)

PURPOSE:

The NFOPB routine opens a logical link to a program on another system. You use this routine when you intend to transmit records or blocks of data.

Data moved through the network on a link opened with NFOPB is transmitted as a string of bits, sent eight bits at a time. This format allows the data to be used by the Data Conversion Routines. To learn more about bit transport, see Appendix F.

CALL FORMAT:

COBOL: ENTER MACRO NFOPB USING nln, trgsys, objtyp, desc, tsknam, userid, passwd, acct, usdat, wait.

FORTRAN: status = NFOPB (nln, trgsys, objtyp, desc, tsknam, 1 userid, passwd, acct, usdat, wait)

where:

nln is the Network Logical Name (NLN) to be returned when this routine successfully finishes processing. You use the NLN to identify this link when you call other Task-to-Task routines. The NLN is a one-word integer.

trgsys is the node name of the target system. The target system, in this case, is the system which runs the passive task that you want to access with this link.

The node name has a length of sixteen ASCII-7 characters. If your node name is less than sixteen characters, left-justify the field. If you give this argument a value of spaces, it defaults to the local system's node name.

objtyp is the object type of the passive task. The object type tells the kind of service performed by the passive task. This argument has a length of sixteen ASCII-7 characters. If the object type has less than sixteen characters, left-justify the field.

The object type can be expressed as either a number or name. Most programs use an object type of 0 or TASK. Server programs which perform a generic service (MAIL, for example) have a non-zero numeric object type. You can find a list of valid DECnet object types and their meanings in the appropriate DECnet User's Guide.

desc is the DECnet descriptor. You must use a descriptor when you plan to access a task with object type 0 or object name TASK. The descriptor must contain the DECnet taskname of the passive task on the remote system. For TOPS-20 to TOPS-20 communication when the object type is not zero, you can provide a descriptor. See Appendix D for further information. The descriptor has a length of sixteen ASCII-7 characters. If your descriptor is less than sixteen characters, left-justify the field.

NOTE

For a TOPS-20 system, the name of the target system, the object type and the DECnet descriptor cannot have a combined total of more than thirty-seven non-blank characters.

tsknam is the DECnet taskname. Taskname is a unique sixteen character ASCII-7 string that identifies this process to the network. An active task never has to specify a taskname. If you pass a value of spaces for this argument, the operating system assigns a unique name.

NOTE

The following three arguments are optional user attributes. The passive task may use these attributes to validate a network connection, or to perform any other recognition function agreed to by both tasks. These arguments are optional if you want to connect to a passive VMS task or a passive VMS task that is already running. The arguments are required if you want to connect to a passive VMS task that is to be started as a result of your request. See Appendix D for further information.

userid is your userid. Userid has a length of thirty-nine ASCII-7 characters. If your userid is less than thirty-nine characters, left-justify the field. If you don't want to specify a userid, pass a value of spaces for this argument.

passwd is your password. Password has a length of thirty-nine ASCII-7 characters. If your password is less than thirty-nine characters, left-justify the field. If you don't want to specify a password, pass a value of spaces for this argument.

acct is your account number or charge code. This field has a length thirty-nine ASCII-7 characters. If this information is less than thirty-nine characters, left-justify the field. Give this argument a value of spaces if you plan to connect to a VMS system or you don't want to specify an account number.

usdat is sixteen optional ASCII-7 characters of user data. See the NFINF routine for more information.

wait is a one-word integer that gives the wait code.

Set the wait code to "no" if you do not want your program to wait until it establishes a connection to the passive task. Using this code enables your program to perform other duties while waiting for the network connection. To find out if the passive task has accepted your connection, periodically call the NGFND routine to check status. The DIL Name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want your program to wait until the passive task has accepted your link. The routine does not return to your program until it establishes the specified link. While it waits, you can not use the active task. Waiting uses minimal CPU time. The DIL Name for this argument is:

WAIT-YES

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-TOOMANY	You attempted too many links. The DIL allows a maximum of 20 open links. On TOPS-20 systems, however, a non-privileged user can only open a maximum of four links.
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-ABORTREJECT	The link was aborted or rejected.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

6.3.5 NFOP8 - Open a Link From an Active Task (8-bit)

PURPOSE:

The NFOP8 routine opens a logical link to a program on another system. You use this routine when you intend to transmit 8-bit bytes of data. This type of link is used primarily to transmit data between two VMS systems or to perform special operations.

Data is moved through a link opened with NFOP8 as a string of 8-bit bytes. The routine does not transmit any unused bits. The target system stores the data in whatever way it normally stores 8-bit bytes of data. To learn more about bit transport, see Appendix F.

CALL FORMAT:

COBOL: ENTER MACRO NFOP8 USING nln, trgsys, objtyp, desc, tsknam, userid, passwd, acct, usdat, wait.

FORTRAN: status = NFOP8 (nln, trgsys, objtyp, desc, tsknam, 1 userid, passwd, acct, usdat, wait)

where:

nln is the Network Logical Name (NLN) to be returned when this routine successfully finishes processing. You use the NLN to identify this link when you call other Task-to-Task routines. The NLN is a one-word integer.

trgsys is the node name of the target system. The target system, in this case, is the system which runs the passive task that you want to access with this link.

The node name has a length of sixteen ASCII-7 characters. If your node name is less than sixteen characters, left-justify the field. If you give this argument a value of spaces, it defaults to the local system's node name.

objtyp is the object type of the passive task. The object type tells the kind of service performed by the passive task. This argument has a length of sixteen ASCII characters. If the object type has less than sixteen characters, left-justify the field.

The object type can be expressed as either a number or name. General, one-purpose programs use an object type of 0 or TASK. Server programs which perform a generic service (MAIL, for example) have a non-zero numeric object type. You can find a list of valid DECnet object types and their meanings in the appropriate DECnet User's Guide.

desc is the DECnet descriptor. You must use a descriptor when you plan to access a task with object type 0 or object name TASK. The descriptor must contain the DECnet taskname of the passive task on the remote system. For TOPS-20 to TOPS-20 communication when the object type is not zero, you can provide a descriptor. See Appendix D for further information. The descriptor has a length of sixteen ASCII-7 characters. If your descriptor is less than sixteen characters, left-justify the field.

NOTE

For a TOPS-20 system, the name of the target system, the object type and the DECnet descriptor cannot have a combined total of more than thirty-seven non-blank characters.

tsknam is the DECnet taskname. Taskname is a unique sixteen character ASCII-7 string that identifies this process to the network. An active task never has to specify a taskname. If you pass a value of spaces for this argument the operating system assigns a unique taskname.

NOTE

The following three arguments are optional user attributes. The passive task may use these attributes to validate a network connection, or to perform any other recognition function agreed to by both tasks. These arguments are optional if you want to connect to a passive TOPS-20 or TOPS-10 task, or a passive VMS task that is already running. The arguments are required if you want to connect to a passive VMS task that starts up as a result of your request.

userid is your userid. Userid has a length of thirty-nine ASCII-7 characters. If your userid is less than thirty-nine characters, left-justify the field. If you don't want to specify a userid, pass a value of spaces for this argument.

passwd is your password. Password has a length of thirty-nine ASCII-7 characters. If your password is less than thirty-nine characters, left-justify the field. If you don't want to specify a password, pass a value of spaces for this argument.

acct is your account number or charge code. This field has a length thirty-nine ASCII-7 characaters. If this information is less than thirty-nine characters, left-justify the field. Give this argument a value of spaces if you plan to connect to a VMS system or you don't want to specify an account number.

usdat is sixteen optional ASCII-7 characters of user data. See the NFINF routine for more information.

wait is a one-word integer that gives the wait code.

Set the wait code to "no" if you do not want your program to wait until it establishes a connection to the passive task. Using this code enables your program to perform other duties while waiting for the network connection. To find out if the passive task has accepted your connection, periodically call the NFGND routine to check status. The DIL Name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want your program to wait until the passive task has accepted your link. The routine does not return to your program until it establishes the specified link. While it waits, you can not use the active task. Waiting uses minimal CPU time. The DIL Name for this argument is:

WAIT-YES

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-TOOMANY	You attempted too many links. The DIL allows a maximum of 20 open links. On TOPS-20 systems, however, a non-privileged user can only open a maximum of four links.
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-ABORTREJECT	The link was aborted or rejected.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

6.3.6 NFOPP - Open a Link From a Passive Task

PURPOSE:

The NFOPP routine opens a logical link from a passive task. It indicates that the server program is ready to accept connections from active programs.

Data can be moved through the network in different ways. You must make sure that both the active and passive tasks specify the same method of data transfer. The active task which connects to NFOPP establishes the way data actually moves through the network. (NFOPA moves ASCII data, NFOPB moves binary data, NFOP8 moves 8-bit bytes of data.) When it receives a connect request, the passive task calls the NFACC routine to accept the request. The "lnktyp" argument in the call to NFACC specifies the type of data to be transfered over the connection. If the active task and the NFACC routine specify different data types, the results are undefined. To learn more about bit transport, see Appendix F.

CALL FORMAT:

COBOL: ENTER MACRO NFOPP USING nln, objtyp, desc, tsknam, wait.

FORTRAN: status = NFOPP (nln, objtyp, desc, tasknam, wait)

where:

nln is the Network Logical Name (NLN) to be returned when this routine successfully finishes processing. You use the NLN to identify this link when you call other Task-to-Task routines. The NLN is a one-word integer.

objtyp is the object type of the passive task. The object type tells the kind of service performed by this task. This argument has a length of sixteen ASCII-7 characters. If the object type has less than sixteen characters, left-justify the field.

The object type can be expressed as either a number or name. General, one-purpose programs use an object type of 0 or TASK. Server programs which perform a generic service (MAIL, for example) have a non-zero numeric object type. You can find a list of valid DECnet object types and their meanings in the appropriate DECnet User's Guide.

desc is the DECnet descriptor. A passive task can only specify a descriptor for TOPS-20 to TOPS-20 communication when the object type is not zero. For a TOPS-10 system, this argument is ignored. See Appendix D for examples of proper task identification.

A descriptor has a length of sixteen ASCII-7 characters. If your descriptor is less than sixteen characters, left-justify the field. If you don't want to specify a descriptor, pass a value of spaces for this argument.

tsknam is the DECnet taskname. Taskname is a unique sixteen character ASCII-7 string that identifies this process to the network. Normally, a passive task must specify a taskname. Active tasks use the taskname to identify the passive task. See Appendix D for further information. If you don't specify a taskname, the operating system assigns a unique taskname.

wait is a one-word integer that gives the wait code.

Set the wait code to "no" if you you want to set up a server task, but you do not want to wait until an active task requests its services. Using this code leaves your program free to perform other duties. To find out if the an active wants to connect to the server, call the NFGND routine to check status. The DIL Name of this argument is:

WAIT-NO

Set the wait code to "yes" if you want to set up a server task that waits until an active task requests its services. The routine returns a status value when an active task tries to connect to the server. While it waits, you cannot use the server program to perform any other processing functions. Waiting uses minimal CPU time. The DIL Name of this argument is:

WAIT-YES

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-TOOMANY	You attempted too many links. The DIL allows a maximum of 20 open links. On TOPS-20 systems, however, a non-privileged user can only open a maximum of four links.
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-ABORTREJECT	The link was aborted or rejected.
DIT-HORRIBLE	This code is returned in the event of system or unexpected error.

6.3.7 NFACC - Accept a Connection

PURPOSE:

The NFACC routine is used by the passive task to accept a connection from an active task and to specify how data will be moved through the network. The NFACC routine is always used following the NFOPP routine, which opens the logical link.

CALL FORMAT:

COBOL: ENTER MACRO NFACC USING nln, lnktyp, char, opdat.

FORTRAN: status = NFACC (nln, lnktyp, char, opdat)

where:

nln is the Network Logical Name set by the NFOPP routine when it successfully finishes processing. NLN is a one-word integer.

lnktyp is a one-word intger that specifies the type of data that you want to transfer over the link. If the active task specifies a different type of data, the results will be undefined. You must specify one of the following values:

Link Data Type	DIL Name
ASCII. To transfer ASCII data, use the NFOPA routine to open the link.	LTYPE-ASCII
Binary data. To transfer binary data, use the NFOPB routine to open the link.	LTYPE-BINARY
8-bit bytes. To transfer 8-bit bytes of data, use the NFOP8 rou- to open the link.	LTYPE-8BIT

char is a one word-integer that specifies the number of characters (0-16) of optional data that you plan to send (see below).

opdat is sixteen optional ASCII-7 characters of user data. See the NFINF routine for more information.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

6.3.8 NFREJ - Reject a Connection

PURPOSE:

The NFREJ routine is used by the passive task to reject a connection request from an active task.

CALL FORMAT:

COBOL: ENTER MACRO NFREJ USING nln, rejcod, char, opdat.

FORTRAN: status = NFREJ (nln, rejcod, char, opdat)

where:

nln is the Network Logical Name set by the NFOPP routine when it successfully finishes processing. NLN is a one word integer.

rejcod is a one-word integer that specifies the type of reject requested. You should use nine for this code, which means "User Program Abort." See your DECnet User's Guide for a list of abort/reject codes. This parameter is not used on either VMS or TOPS-10 systems.

char is a one-word integer that specifies the number of characters (0-16) of optional data that you plan to send (see below.)

opdat is sixteen ASCII-7 characters of optional user data. For more information, see the NFINF routine.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

6.3.9 NFRCV - Receive Data Over a Link

PURPOSE:

The NFRCV routine receives data (DECnet messages) over a logical link.

If you specify message mode, this routine reads one DECnet message. Message mode is the recommended mode.

If you specify stream mode, NFRCV reads a stream of as many characters as you request (in mxunit, below). It reads these characters from as many DECnet messages as are required to equal the numerical value shown in mxunit. The routine stops reading in the middle of a message once it has read the correct number of characters.

VMS users cannot specify stream mode; VMS systems always send and receive messages. You must use message mode to communicate with tasks that run on VMS systems. If you use a DECSYSTEM-20 or a DECsystem-10, you have the capability to specify stream mode.

On a DECSYSTEM-20, using message mode, if the user's buffer is not large enough to receive the message, DIT-OVERRUN will be returned. You may call NFRCV again with a larger buffer to receive the message with no loss of data.

On a DECsystem-10, using message mode, if the user buffer is not large enough to receive the message, the message is truncated to the size of the buffer and a status of DIT-OVERRUN is returned to indicate the loss of data. Normally, this should not happen because cooperating tasks can agree on the message size. To create a task that can receive messages without loss of data, you should use stream mode. After each call to NFRCV in stream mode, the task can inspect the value returned for msmode to determine if an end-of-message was detected while filling the user buffer.

CALL FORMAT:

```
COBOL:    ENTER MACRO NFRCV USING nln, msunit, mxunit, bufloc,
              msmode, wait.

FORTRAN:  status = NFRCV (nln, msunit, mxunit, bufloc, msmode,
          1 wait)
```

where:

nln — is the Network Logical Name set by the NFOPA, NFOPB, NFOP8 or NFOPP routine when the routine successfully finished processing. The NLN is a one-word integer.

msunit — is the message unit size. It tells the byte size, in bits, of messages written in binary format (with links opened through NFOPB). A value of zero for this argument indicates the data is to be transferred as words. If you open the active side of the link with NFOPA or NFOP8, the routine ignores this argument.

mxunit is a one-word integer that gives the maximum number of units to be read by the routine. When NFRCV returns, it replaces this value with the actual number of units that it read.

- For links opened with NFOPA, this is the maximum number of ASCII characters in the message.
- For links opened with NFOP8, this is the maximum number of sequential 8-bit bytes in the message.
- For links opened with NFOPB, this is the maximum number of bytes (of the unit size shown in msunit) or words in the message. NFRCV pads the last byte or word sent with zero bits if it does not divide evenly into bytes of the specified size.

If you use stream mode, set the value of this argument to the actual number of characters, bytes or words to be read.

bufloc is the location of the user buffer where the message will be placed after it is read. This buffer must be at least as large as the number of bytes, characters or words specified in mxunit.

msmode is a one-word integer that indicates the message-mode flag. You can set this argument to message mode or stream mode.

If you want to read the data in message mode, the DIL Name for this argument is:

MSG-MSG

If you want to read the required number of characters or bytes in stream mode, the DIL Name for this argument is:

MSG-STM

On a DECsystem-10, if the user specifies msmode as MSG-STM and the end-of-message is detected while filling the user's buffer, msmode will be returned with a value of MSG-MSG.

wait is a one-word integer that gives the wait code.

Set the wait code to "no" if you want the routine to return whatever data is currently available. If no data is available the routine returns and you can use the program to perform other duties. The DIL name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want the routine to wait until data is received or the read fails. Waiting uses minimal CPU time. The DIL name for this argument is:

WAIT-YES

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-ABORTREJECT	This code is returned if the link is disconnected or aborted. The character count area contains the number of units, bytes or words, that the routine reads before the disconnect.
DIT-OVERRUN	This code is returned if you try to send too much data to the routine.
DIT-NOTENOUGH	This code is returned if the amount of data you requested is not available. This error occurs only when you use stream mode with a no-wait code.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.
DIT-INTERRUPT	Interrupt data must be read first using NFRCI.

6.3.10 NFSND - Send Data Over a Link

PURPOSE:

The NFSND routine sends data over a logical link.

If you open the logical link with NFOPA, the routine sends the data in ASCII format. The receiving task can directly read ASCII data from another system, even if the tasks are on heterogeneous systems.

If you open the logical link with NFOP8, the routine sends the data as sequential 8-bit bytes. The local system treats the data as it normally treats 8-bit bytes of data. The remote system treats the data it receives in the way it normally stores 8-bit bytes of data.

If you open the logical link with NFOPB, the routine sends the data in binary format. If the sending and receiving tasks are on homogeneous systems, the receiving system can directly read the data. If the sending and receiving tasks are on heterogeneous systems, you must convert the data using the Data Conversion Routines. The sending task can perform the conversion before it sends the data or the receiving task can convert the data it receives.

If you specify message mode, this routine sends one DECnet message. To send more than one message, set up a loop to perform this routine until it has sent all of the messages.

If you specify stream mode, NFSND sends the data in a stream of as many characters as you request (in length, below). It sends the data in as many DECnet messages as are required to equal the numerical value shown in length. The routine stops sending in the middle of a message if it has sent the correct number of characters, bytes or words.

VMS users cannot specify stream mode; VMS systems always send and receive messages. You must use message mode to communicate with tasks that run on VMS systems. If you use a DECSYSTEM-20 or a DECsystem-10, you have the capability to specify stream mode.

CALL FORMAT:

COBOL: ENTER MACRO NFSND USING nln, msunit, length, buffer, msmode.

FORTRAN: status = NFSND (nln, msunit, length, buffer, msmode)

where:

nln is the Network Logical Name set by the NFOPA, NFOPB, NFOP8 or NFOPP routine when the routine successfully finished processing. The NLN is a one-word integer.

msunit is a one-word integer that specifies the message unit size. It tells the byte size, in bits, of messages written in binary format (with links opened through NFOPB). A value of zero for this argument indicates that the data is to be sent as words. If you open the active side of the link with NFOPA or NFOP8, the routine ignores this argument.

length is a one-word integer that gives the length of the data that you want to send. This argument must have a value that is greater than zero.

- For links opened with NFOPA, length is given in ASCII characters.
- For links opened with NFOP8, length is given in sequential 8-bit bytes.
- For links opened with NFOPB, length is given in bytes (of the unit size shown in msunit) or words.

buffer is the buffer containing the data that you want to send.

msmode is a one word integer that gives the message-mode flag. You can set this argument to send data in message mode or in stream mode.

If you want to send data in message mode, the DIL Name for this argument is:

MSG-MSG

If you want to send the required number of characters or bytes in stream mode, the DIL Name for this argument is:

MSG-STM

A TOPS-20 or TOPS-10 system permits you to send data in message mode or stream mode. It is advised, however, that you always send data in message mode. VMS users must send data in message mode.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

6.3.11 NFINT - Send an Interrupt Data Message Over a Link

PURPOSE

The NFINT routine sends a single interrupt data message over a logical link.

Unlike NFSND, NFINT data is always send in message mode, so a prompt attempt to send the data is guaranteed. Data sent in this mode is not sent in synchronization with data sent by NFSND. Only one interrupt data message can be outstanding from each end of a logical link at one time. If an interrupt data message is sent over a logical link by one process, a second interrupt data message cannot be sent by that process until the first one has been received at the other end of the logical link. If a second interrupt data message is sent before the first one has been received at the other side of the link, then the first interrupt data message may be lost at the receiving end of the link.

CALL FORMAT:

COBOL: ENTER MARCO NFINT USING nln, char, buffer

FORTRAN: status = NFINT (nln, char, buffer)

where:

nln is the Network Logical Name set by the NFOPP routine when it successfully finishes processing. NLN is a one word integer.

char is a one-word integer that specifies the number of ASCII-7 characters (1-16) of interrupt data to send.

buffer is the buffer that contains the 1-16 characters of ASCII-7 data that you want to send.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

6.3.12 NFRCI - Receive an Interrupt Data Message Over a Link

PURPOSE:

The NFRCI routine receives a single interrupt data message over a logical link.

Receipt of an interrupt data message is an asynchronous event. You should check for asynchronous events by using NFGND which will announce interrupt data messages before it will announce "regular" data messages (sent by NFSND). Interrupt data messages must be read before NFGND will announce any lower-level events (regular data messages or disconnections). NFRCV will return the DIT-INTERRUPT error and refuse to return data if an interrupt message is available which has not yet been read by NFRCI.

CALL FORMAT:

COBOL: ENTER MACRO NFRCI USING nln, char, buffer.

FORTRAN: status = NFRCI (nln, char, buffer)

where:

nln is the Network Logical Name set by the NFOPP routine when it successfully finishes processing. NLN is a one word integer.

char is a one-word integer into which the number of ASCII-7 characters (1-16) of interrupt data read is returned.

buffer is the location of the user buffer where the message will be placed after it is read. The length of this buffer must be at least 16 ASCII-7 characters.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NODATAAVAILABLE	No interrupt data is available to be read at this time.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

6.3.13 NFCLS - Close a Link

PURPOSE:

The NFCLS routine disconnects or aborts a logical link, releasing its resources to be used by another link.

A "synchronous disconnect" is the normal way to close a link; it disconnects the link after it performs all outstanding data transmission. An abort would be used on a logical link if the remote program terminated abnormally, or if fatal errors occurred on the link. An abort instantaneously disconnects the link.

You can call the NFCLS routine to disconnect the link before or after receiving a disconnect from the other end of the link. To preserve data integrity, the recipient of the last piece of data should be the first to disconnect the link (using a synchronous disconnect). The program that sent the data recognizes that its data was read when it receives the disconnect; it then aborts its end of the link.

CALL FORMAT:

COBOL: ENTER MACRO NFCLS USING nln, disc, char, opdat.

FORTRAN: status = NFCLS (nln, disc, char, opdat)

where:

nln is the Network Logical Name, set by the NFOPA, NFOPB, NFOP8 or the NFOPP routine when the routine successfully finished processing. The NLN is a one-word integer.

disc is a one-word integer that specifies the type of disconnect requested. Use zero for this argument if you want a synchronous disconnect. A non-zero value for this argument indicates an abort. The DECnet User's Guide for your system shows a list of abort codes. Code 9 ("User Program Abort") signifies a normal abort.

char is a one-word integer that gives the number of characters (0 to 16) of optional data to be sent (see opdat, below).

opdat is sixteen optional ASCII-7 characters of user data. See the NFINF routine for more information.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

6.4 TOPS-10/TOPS-20 TASK-TO-TASK EXAMPLES

6.4.1 TOPS-10/TOPS-20 COBOL Task-to-Task Examples

```
IDENTIFICATION DIVISION.

PROGRAM-ID.

        PASC36.

AUTHOR.

        SOFTWARE ENGINEERING.

        This program opens a passive link and then waits for a
        connection from an active task (created by the program
        ACTC36). Once a link is established, user specified messages
        are sent in both directions across the link. The link is
        closed by the program ACTC36 and this program waits for a
        confirmation of the close.
INSTALLATION.

        DEC MARLBOROUGH.

ENVIRONMENT DIVISION.

       .

       .

       .

DATA DIVISION.

WORKING-STORAGE SECTION.

01  INTERFACE FILES.
    COPY DIT OF "SYS:DIL.LIB".
    COPY DIL OF "SYS:DIL.LIB".

01  DILINI-PARAMETERS.
* Dilini is necessary for DECsystem-10 and DECSYSTEM-20 Cobol only.
    05  DIL-INIT-STATUS PIC S9(10) COMP.
    05  DIL-STATUS PIC S9(10) COMP.
    05  DIL-MESSAGE PIC S9(10) COMP.
    05  DIL-SEVERITY PIC S9(10) COMP.

01  DATA-RECORDS.
    05  SEND-DATA PIC X(100) USAGE DISPLAY-7.
    05  RECEIVE-DATA PIC X(100) USAGE DISPLAY-7.

01  COUNT-OPT-DATA PIC S9(10) COMP VALUE 0.
01  OPT-DATA PIC X(16) DISPLAY-7 VALUE SPACES.
01  NETLN PIC S9(10) COMP.
01  OBJECT-ID PIC X(16) DISPLAY-7 VALUE SPACES.
01  DESCRIPTOR PIC X(16) DISPLAY-7 VALUE SPACES.
01  TASKNAME PIC X(16) DISPLAY-7.
01  MESSAGE-UNITS-SIZE PIC S9(10) COMP VALUE 7.
01  MESSAGE-SIZE PIC S9(10) COMP VALUE 100.
```

```
PROCEDURE DIVISION.

SETUP-RETURN-CODES.
* Set up for status code values, using DILINI routine

    ENTER MACRO DILINI USING DIL-INIT-STATUS, DIL-STATUS,
                             DIL-MESSAGE, DIL-SEVERITY.

    IF DIL-INIT-STATUS NOT = 1
        DISPLAY "? Invalid return code from DILINI routine = " DIL-INIT-STATUS.


OPEN-PASSIVE
* Open a passive link.

    MOVE "SERVER" TO TASKNAME.
    MOVE "0" TO OBJECT-ID.

    ENTER MACRO NFOPP USING NETLN, OBJECT-ID, DESCRIPTOR,
                            TASKNAME, DIT-WAIT-NO.

    DISPLAY " NFOPP Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? NFOPP: unsuccessful status return "
        STOP RUN.

CHECK-FOR-CONNECT.
* Wait for a connect request

    ENTER MACRO NFGND USING NETLN, DIT-WAIT-YES.

    DISPLAY " NFGND Status return: " DIL-STATUS.
    IF DIL-MESSAGE = DIT-C-CONNECTEVENT NEXT SENTENCE
    ELSE
        DISPLAY "? NFGND: Unexpected or invalid status returned: " DIL-STATUS
        STOP RUN.

ACCEPT-LINK
* Accept link

    ENTER MACRO NFACC USING NETLN, DIT-LTYPE-ASCII, COUNT-OPT-DATA, OPT-DATA.

    DISPLAY " NFACC Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? NFACC: unsuccessful status return "
        STOP RUN.

CHECK-FOR-DATA
* Wait for a data event on the link

    ENTER MACRO NFGND USING NETLN, DIT-WAIT-YES.

    DISPLAY " NFGND Status return: " DIL-STATUS.
    IF DIL-MESSAGE = DIT-C-DATAEVENT NEXT SENTENCE
    ELSE
        DISPLAY "? NFGND: Unexpected or invalid status returned: " DIL-STATUS
        STOP RUN.
```

```
READ-THE-DATA.
* Read the data received over the link

    MOVE 100 TO MESSAGE-SIZE.

    ENTER MACRO NFRCV USING NETLN, MESSAGE-UNITS-SIZE, MESSAGE-SIZE,
                            RECEIVE-DATA, DIT-MSG-MSG, DIT-WAIT-YES.

    DISPLAY " NFRCV Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? NFRCV: unsuccessful status return "
        STOP RUN.

    DISPLAY " Data received: ".
    DISPLAY RECEIVE-DATA.

SEND-SOME-DATA.
* Send some data over the link

    MOVE 100 TO MESSAGE-SIZE

    DISPLAY " Enter some data to be sent over the link: ".
    ACCEPT SEND-DATA.

    ENTER MACRO NFSND USING NETLN, MESSAGE-UNITS-SIZE, MESSAGE-SIZE,
                            SEND-DATA, DIT-MSG-MSG.

    DISPLAY " NFSND Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? NFSND: unsuccessful status return "
        STOP RUN.

CHECK-FOR-CLOSE.

    ENTER MACRO NFGND USING NETLN, DIT-WAIT-YES.

    DISPLAY " NFGND Status return: " DIL-STATUS.
    IF DIL-MESSAGE NOT = DIT-C-ABREJEVENT AND
       DIL-MESSAGE NOT = DIT-C-DISCONNECTEVENT AND
       DIL-SEVERITY NOT = STS-K-SUCCESS AND
       DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? NFGND: Invalid status returned"
        STOP RUN.

    DISPLAY " PASC36 successful. ".
    STOP RUN.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.

        ACTC36.

AUTHOR.

        SOFTWARE ENGINEERING.

        This program opens an active link by connecting to the passive task set up
        by the program PASC36.  Once the link is established, user specified
        messages are sent in both directions across the link.  Then the link is
        closed.

INSTALLATION.

        DEC MARLBOROUGH.

ENVIRONMENT DIVISION.

        .

        .

        .

DATA DIVISION.

WORKING-STORAGE SECTION.

01  INTERFACE-FILES.
    COPY DIT OF "SYS:DIL.LIB".
    COPY DIL OF "SYS"DIL.LIB".

01  DILINI-PARAMETERS.
* Dilini is necessary for DECsystem-10 and DECSYSTEM-20 COBOL only.
    05  DIL-INIT-STATUS PIC S9(10) COMP.
    05  DIL-STATUS-PIC S9(10) COMP.
    05  DIL-MESSAGE PIC S9(10) COMP.
    05  DIL-SEVERITY PIC S9(10) COMP.

01  DATA-RECORDS.
    05  SEND-DATA PIC X(100) USAGE DISPLAY-7.
    05  RECEIVE-DATA PIC X(100) USAGE DISPLAY-7.

01  COUNT-OPT-DATA PIC S9(10) COMP VALUE 0.
01  OPT-DATA PIC X(16) DISPLAY-7 VALUE SPACES.
01  NETLN PIC S9(10) COMP.
01  HOSTNAME PIC X(06) DISPLAY-7 VALUE SPACES.
01  OBJECT-ID PIC X(16) DISPLAY-7 VALUE SPACES.
01  DESCRIPTOR PIC X(16) DISPLAY-7 VALUE SPACES.
01  TASKNAME PIC X(16) DISPLAY-7.
01  USERID PIC X(39) DISPLAY-7 VALUE SPACES.
01  PASSWD PIC X(39) DISPLAY-7.
01  ACCT PIC X(39) DISPLAY-7 VALUE SPACES.
01  MESSAGE-UNITS-SIZE PIC S9(10) COMP VALUE 7.
01  MESSAGE-SIZE PIC S9(10) COMP VALUE 100.

01  SYNCH-DISCONN PIC S9(10) COMP VALUE 0.
```

```
PROCEDURE DIVISION.

SETUP-RETURN-CODES.
* Set up for status code values, using DILINI routine

    ENTER MACRO DILINI USING DIL-INIT-STATUS, DIL-STATUS
                             DIL-MESSAGE, DIL-SEVERITY.

    IF DIL-INIT-STATUS NOT = 1
        DISPLAY "? Invalid return code from DILINI routine = " DIL-INIT-STATUS.

CONNECT-TO-PASSIVE
* Ask for a connection to the passive link

    MOVE "0" TO OBJECT-ID.
    MOVE "SERVER" TO DESCRIPTOR.
    MOVE SPACES TO TASKNAME.

    ENTER MACRO NFOPA USING NETLN, HOSTNAME, OBJECT-ID, DESCRIPTOR, TASKNAME,
                            USERID, PASSWD, ACCT, OPT-DATA, DIT-WAIT-NO.

    DISPLAY " NFOPA Status return: ", DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? NFOPA: unsuccessful status return "
        STOP RUN.

CHECK-FOR-CONNECT.
* Wait for confirmation of the connection

    ENTER MACRO NFGND USING NETLN, DIT-WAIT-YES.

    DISPLAY " NFGND Status return: " DIL-STATUS.
    IF DIL-MESSAGE = DIT-C-CONNECTEVENT NEXT SENTENCE
    ELSE
        DISPLAY "? NFGND: Unexpected or invalid status returned: " DIL-STATUS
        STOP RUN.

SEND-SOME-DATA
* Send some data over the link

    MOVE 100 TO MESSAGE-SIZE.

    DISPLAY " Enter some data to be sent over the link: ".
    ACCEPT SEND-DATA.

    ENTER MACRO NFSND USING NETLN, MESSAGE-UNITS-SIZE, MESSAGE-SIZE,
                            SEND-DATA, DIT-MSG-MSG.

    DISPLAY " NFSND Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? NFSND: unsuccessful status return "
        STOP RUN.
```

```
CHECK-FOR-DATA.
*Wait for a data event on the link

   ENTER MACRO NFGND USING NETLN, DIT-WAIT-YES.

   DISPLAY " NFGND Status return: " DIL-STATUS
   IF DIL-MESSAGE = DIT-C-DATAEVENT NEXT SENTENCE
   ELSE
       DISPLAY "? NFGND: Unexpected or invalid status returned: " DIL-STATUS
       STOP RUN.

READ-THE-DATA.
* Read the data received over the link

    MOVE 100 TO MESSAGE-SIZE

    ENTER MACRO NFRCV USING NETLN, MESSAGE-UNITS-SIZE, MESSAGE-SIZE,
                            RECEIVE-DATA, DIT-MSG-MSG, DIT-WAIT-YES.

    DISPLAY " NFRCV Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? NFRCV: unsuccessful status return "
        STOP RUN.

    DISPLAY " Data received: ".
    DISPLAY RECEIVE-DATA.

CLOSE-LINK.
* Close the link

    ENTER MACRO NFCLS USING NETLN, SYNCH-DISCONN, COUNT-OPT-DATA, OPT-DATA.

    DISPLAY " NFCLS Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? NFCLS: unsuccessful status return "
        STOP RUN.

    DISPLAY " CTTT36 successful. ".
    STOP RUN.
```

6.4.2 TOPS-10/TOPS-20 FORTRAN Task-to-Task Examples

```
C PASF36

C       This program opens a passive link and then waits for a
C       connection from an active task (created by the program
C       ACTC36).  Once a link is established, user specified messages
C       are sent in both directions across the link.  The link is
C       closed by the program ACTC36 and this program waits for a
C       confirmation of the close.

C Use the DIL interface files.

        INCLUDE 'SYS:DITV7'
        INCLUDE 'SYS:DILV7'

C Data records

        DIMENSION SENDD (20), RECD (20)

C DIL task to task routine parameters

        DIMENSION OPTDAT (4), OBJID (4), DESCR (4), TASKN (4)

        INTEGER NETLN, DILSTS, MSGSIZ, MUNTSZ, CNTOPD

C       Link description fields -- passive end

        DATA OBJID /'     ', '     ', '     ', '     '/
        DATA DESCR /'     ', '     ', '     ', '     '/
        DATA TASKN /'SERVE', 'R    ', '     ', '     '/

        DATA OPTDAT /'     ', '     ', '     ', '     '/


C Program messages

777     FORMAT (' Invalid status returned... ')
778     FORMAT (' Enter some data to be sent over the link: ')
779     FORMAT (' Data received: ')
200     FORMAT (' NFOPP Status return: ', I12)
202     FORMAT (' NFGND Status return: ', I12)
203     FORMAT (' NFACC Status return: ', I12)
204     FORMAT (' NFSND Status return: ', I12)
205     FORMAT (' NFRCV Status return: ', I12)

C       initialize sending and receiving message data fields

        DO 100 I = 1, 20
        SENDD (I) = 0
100     RECD (I) = 0

C       Open a passive link

        DILSTS = NFOPP (NETLN, OBJID, DESCR, TASKN, WAITLN)

        WRITE (5, 200) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 110
        WRITE (5, 777)
        STOP
```

```
C       Wait for a connect request

110     DILSTS = NFGND (NETLN, WAITLY)

        WRITE (5, 202) DILSTS
        IF (DILSTS .EQ. CONEVT) GO TO 120
        WRITE (5, 777)
        STOP

C       Accept link

120     CNTOPD = 0
        DILSTS = NFACC (NETLN, LASCII, CNTOPD, OPTDAT)

        WRITE (5, 203) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 130
        WRITE (5, 777)
        STOP

C       Wait for a data event on the link

130     DILSTS = NFGND (NETLN, WAITLY)

        WRITE (5, 202) DILSTS
        IF (DILSTS .EQ. DATEVT) GO TO 140
        WRITE (5, 777)
        STOP

C       Read the data received over the link

140     MSGSIZ = 100

        DILSTS = NFRCV (NETLN, MUNTSZ, MSGSIZ, RECD, MSGMSG, WAITLY)

        WRITE (5, 205) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 150
        WRITE (5, 777)
        STOP

150     WRITE (5, 779)
155     FORMAT (' ' 20A5)
        WRITE (5, 155) RECD

C       Send some data over the link

        WRITE (5, 778)
157     FORMAT (20A5)
        ACCEPT 157, SENDD

        MSGSIZ = 100

        DILSTS = NFSND (NETLN, MUNTSZ, MSGSIZ, SENDD, MSGMSG)

        WRITE (5, 204) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 160
        WRITE (5, 777)
        STOP
```

```
C       Check for the link being closed

160     DILSTS = NFGND (NETLN, WAITLY)

        WRITE (5, 202) DILSTS
        IF (DILSTS .EQ. ARJEVT) GO TO 170
        IF (DILSTS .EQ. DSCEVT) GO TO 170
        IF (DILSTS .EQ. NORMAL) GO TO 170
        WRITE (5, 777)
        STOP

170     WRITE (5, 175)
175     FORMAT (' PASF36 test successful ')
        STOP
        END

C ACTF36

C       This program opens an active link by connecting to the passive
C       task set up by the program PASC36.  Once the link is
C       established, user specified messages are sent in both
C       directions across the link.  Then the link is closed.

C Use the DIL interface files

              INCLUDE 'SYS:DITV7'
              INCLUDE 'SYS:DILV7'

C Data records

              DIMENSION SENDD (20), RECD (20)

C DIL task to task routine parameters

              DIMENSION HSTNAM (4), OPTDAT (4), OBJID (4), DESCR (4)
              DIMENSION PASSWD (8), ACCT (8), USERID (8), TASKN (4)

              INTEGER NETLN, DILSTS, MSGSIZ, MUNTSZ, CNTOPD, SYNCDS

C       Link description fields

        DATA OBJID /'TASK ', '     ', '     ', '     '/
        DATA DESCR /'SERVE', 'R    ', '     ', '     '/
        DATA TASKN /'     ', '     ', '     ', '     '/
        DATA HSTNAM /'     ', '     ', '     ', '     '/
        DATA PASSWD /'     ', '     ', '     ', '     ',
     1              '     ', '     ', '     ', '     '/
        DATA USERID /'     ', '     ', '     ', '     ',
     1              '     ', '     ', '     ', '     '/
        DATA ACCT /'     ', '     ', '     ', '     ',
     1            '     ', '     ', '     ', '     '/


        DATA OPTDAT /'     ', '     ', '     ', '     '
```

```
C Program messages

777     FORMAT (' Invalid status returned... ')
778     FORMAT (' Enter some data to be sent over the link: ')
779     FORMAT (' Data received: ')
201     FORMAT (' NFOPA Status return: ', I12)
202     FORMAT (, NFGND Status return: ', I12)
204     FORMAT (, NFSND Status return: ', I12)
205     FORMAT (, NFRCV Status return: ', I12)
206     FORMAT (, NFCLS Status return: ', I12)

C       initialize sending and receiving message data fields

        DO 100 I = 1, 20
        SENDO (I) = 0
100     RECD (I) = 0

C       Ask for a connection to the passive link

        DILSTS = NFOPA (NETLN, HSTNAM, OBJID, DESCR, TASKN,
     1                  USERID, PASSWD, ACCT, OPTDAT, WAITLN)

        WRITE (5, 201) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 110
        WRITE (5, 777)
        STOP

C       Wait for confirmation of the connection

110     DILSTS = NFGND (NETLN, WAITLY)

        WRITE (5, 202) DILSTS
        IF (DILSTS .EQ. CONEVT) GO TO 120
        WRITE (5, 777)
        STOP

C       Send some data over the link

120     WRITE (5, 778)
125     FORMAT (20A5)
        ACCEPT 125, SENDD

C       Initialize number of bytes

        MSGSIZ = 1

        DILSTS = NFSND (NETLN, MUNTSZ, MSGSIZ, SENDD, MSGMSG)

        WRITE (5, 204) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 130
        WRITE (5, 777)
        STOP

C       Wait for a data event on the link

130     DILSTS = NFGND (NETLN, WAITLY)

        WRITE (5, 202) DILSTS
        IF (DILSTS .EQ. DATEVT) GO TO 140
        WRITE (5, 777)
        STOP
```

```
C       Read the data received over the link

140     MSGSIZ = 100

        DILSTS = NFRCV (NETLN, MUNTSZ, MSGSIZ, RECD, MSGMSG, WAITLY)

        WRITE (5, 205) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 150
        WRITE (5, 777)
        STOP

150     WRITE (5, 779)
155     FORMAT (' '20A5)
        WRITE (5, 155) RECD

C       Close the link

        SYNCDS = 0
        DILSTS = NFCLS (NETLN, SYNCDS, CNTOPD, OPTDAT)
        WRITE (5, 206) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 160
        WRITE (5, 777)
        STOP

160     WRITE (5, 165)
165     FORMAT (' ACTF36 test successful ')
        STOP
        END
```

CHAPTER 7
TOPS–10 AND TOPS–20 REMOTE FILE ACCESS

CHAPTER 7

TOPS-10 AND TOPS-20 REMOTE FILE ACCESS

7.1 REMOTE FILE ACCESS FROM TOPS-10 OR TOPS-20 COBOL

The information included in this section assumes

-You are writing a COBOL program
-You plan to use the program on a TOPS-20 or TOPS-10 system

To store a file number, file name and user attributes, to read or write a record or to perform a status check you must represent several data items in your program. Users generally, but not necessarily, allocate space for these data items in Working-Storage.

7.1.1 Compiling Programs

To use the DIL Remote File Access Routines on TOPS-20 from a COBOL program, you may need to compile your program with the /STACK compiler switch to insure that you have an adequate pushdown list size. If your program gets a stack overflow, compile the program with /STACK:2000. On TOPS-10 you may need to compile your COBOL program with the /D compiler switch. If your program gets a stack overflow, compile it with /D:2000.

7.1.2 Including the Interface Support Files

The Interface Support file provided for TOPS-10 and TOPS-20 COBOL is a copy library called DIL.LIB. The COBOL COPY verb can be used to retrieve the information in the library at compilation time. There are three library elements in DIL.LIB. The elements are DIL, DIT, and DIX.

The library element DIL defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task routines and the remote file access routines. The general success status code (SS-NORMAL) is defined in element DIL. Severity codes and system codes are defined in element DIL. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration:

```
COPY DIL OF "SYS:DIL.LIB".
```

In the following example, the DIL element of the library is retrieved and included in your program.

```
01  Interface-files.
    COPY DIL OF "SYS:DIL.LIB".
```

The library element DIT defines codes specific to the Task-to-Task and Remote File Access Routines. This includes the DIT status codes, Remote File Access file types, Remote File Access open modes, Remote File Access record formats, Remote File Access record attributes and Remote File Access close options. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration, as shown above for element DIL:

```
COPY DIT OF "SYS:DIL.LIB".
```

For programs which use the remote file access routines, you must include both the DIL and DIT library elements.

7.1.3 Storing a File Number

A file number consists of one word of TOPS-10/TOPS-20 memory. To store a file number, you must define a data item with the following format:

```
01  your-fnum       PIC S9(10) USAGE COMPUTATIONAL.
```

When you call the ROPEN routine use your-fnum as the value for fnum, file number to be returned. When the routine successfully finishes processing, it returns a value in your-fnum.

7.1.4 Storing Accounting Information

To include your user code, password and account or charge code in a call to the ROPEN routine, you must describe these attributes as data items in WORKING-STORAGE. These arguments are always DISPLAY-7 data items; they must always have a length of PIC X(39). The format for these data items is as follows:

```
01  userid          PIC X(39)       USAGE DISPLAY-7.
01  passwd          PIC X(39)       USAGE DISPLAY-7.
01  acct            PIC X(39)       USAGE DISPLAY-7.
```

7.1.5 Reading and Writing Remote Data

To read or write data in a file on a remote system, the program on the local system must define an area to accept the data and a one-word computational item for the length of the data.

```
01  the-record      PIC X(100)      USAGE DISPLAY-7.
01  dlength         PIC S9(10)      USAGE COMPUTATIONAL VALUE 100.
```

7.1.6 Checking the Status of a Remote File Access Routine

Section 5.1.3 of this manual presents a method for checking the status of a TOPS-10/TOPS-20 COBOL DIL Routine. Section 5.3.1 describes the DILINI initialization routine.

A call to one of the Remote File Access Routines with a simple check for success might look like this:

```
ENTER MACRO RCLOSE USING fnum, close.
IF NOT dil-ok THEN
     DISPLAY "fatal error".
```

A call to the same routine with provisions for handling a specific type of error might look like this:

```
ENTER MACRO RCLOSE USING fnum, close.
IF NOT dil-ok
   IF dil-stat = DIT-C-INVARG
     DISPLAY "invalid argument error"
   ELSE
     DISPLAY "other error".
```

To determine which error occurred, compare dil-stat to the DIT condition identifier defined in the TOPS-10/TOPS-20 COBOL Interface Support files. In the example above, DIT-C-INVARG is the value (defined in the Interface Support file) indicating an invalid argument was specified.

7.1.7 The TOPS-10 Software Interrupt System

The DIL task-to-task and remote file access routines need the services of the TOPS-10 software interrupt system (PSI). Because the FORTRAN and COBOL runtime systems do not provide a facility to share the use of the software interrupt system between non-cooperating routines in the same program, the user cannot use the software interrupt system in a COBOL or FORTRAN program that uses the task-to-task or remote file access routines.

7.2 REMOTE FILE ACCESS FROM TOPS-20 OR TOPS-10 FORTRAN

The information included in this section assumes

-You are writing a FORTRAN program
-You plan to use the program on a TOPS-20 or TOPS-10 system

This section explains how to store a file number, file name and user attributes. It also tells how to read or write a record or to perform a status check.

7.2.1 Including the Interface Support Files

The Interface Support files provided for TOPS-10 and TOPS-20 FORTRAN are text files called DILV7.FOR, DITV7.FOR and DIXV7.FOR. You can include the information from these files into your source programs at compilation time using the FORTRAN INCLUDE statement.

The file DILV7 defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access routines. The general success status code (SS-NORMAL) is defined in DILV7. Severity codes and system codes are defined in DILV7. To define these names in your program, include the following statement in your program:

```
INCLUDE 'SYS:DILV7'
```

The file DITV7 defines codes specific to the Task-to-Task and Remote File Access Routines. This includes the DIT status codes, Remote File Access file types, Remote File Access open modes, Remote File Access record formats, Remote File Access record attributes and Remote File Access close options. To define these names in your program, include the following statement:

```
INCLUDE 'SYS:DITV7'
```

For programs which use the remote file access routines, you must include both the DILV7 and DITV7 files.

7.2.2 Storing a File Number

To store a file number, you must implicitly or explicitly declare an integer with the following format:

```
INTEGER    fnum
```

Use fnum as the value for the file number argument in the call to the ROPEN routine.

7.2.3 Storing Account Information

To include your user code, password, account number (or charge code) in a call to the ROPEN routine, you must first declare these values in your program. The format for these data items is as follows:

```
INTEGER   userid (8)  or  CHARACTER*39  userid
INTEGER   passwd (8)  or  CHARACTER*39  passwd
INTEGER   acct   (8)  or  CHARACTER*39  acct
```

7.2.4 Reading and Writing Remote Data

To read or write data from a file on a remote system, you must declare an area on the local system to accept the data, and an integer for the length of the data.

```
INTEGER  therec (20), recl
DATA     recl/100/
```

7.2.5 Checking the Status of a Remote File Access Routine

The Remote File Access Routines return status values as their function values. To perform an error check on a Remote File Access Routine, first declare an integer where the routine can place a status value.

```
     INTEGER    status
```

A simple call with an error check might then be:

```
        status = RREAD (fnum, dunit, maxsiz, bufnam)
        IF (status .EQ. NORMAL) GOTO 100
        IF (status .EQ. INVARG) GOTO 50
        TYPE 10
10      FORMAT (' error occurred')
  .
  .
  .
50      TYPE 51
51      FORMAT (' invalid data type error')
100
  .
  .
  .
```

To determine which error occurred, compare status with the DIT status codes defined in the TOPS-10/TOPS-20 FORTRAN Interface Support files. In the example above, NORMAL and INVARG are status code values defined in the Interface Support file.

7.2.6 The TOPS-10 Software Interrupt System

The DIL task-to-task and remote file access routines need the services of the TOPS-10 software interrupt system (PSI). Because the FORTRAN and COBOL runtime systems do not provide a facility to share the use of the software interrupt system between non-cooperating routines in the same program, the user cannot use the software interrupt system in a COBOL or FORTRAN program that uses the task-to-task or remote file access routines.

7.3 TOPS-20 AND TOPS-10 REMOTE FILE ACCESS REFERENCE

7.3.1 ROPEN - Open a Remote File

PURPOSE:

This routine opens a remote, sequential ASCII file for processing. It also assigns a file number to the opened file.

CALL FORMAT:

COBOL: ENTER MACRO ROPEN USING fnum, fnam, userid, passwd, acct, mode, dattyp, recfor, recatt, recsiz, runit.

FORTRAN: status = ROPEN (fnum, fnam, userid, passwd, acct, 1 mode, dattyp, recfor, recatt, recsiz, runit)

where:

fnum is a one-word integer that gives the file number of the file that you want to open. This number is assigned by the subroutine.

fnam is an ASCII-7 character string that gives the name of the file to be opened. The file name must also include the node name of the file's system of origin.

NOTE

The next three arguments supply accounting information. If the remote node is a DECSYSTEM-20 or a DECsystem-10, you must supply a userid and password. You must also specify an account unless the remote system sets a default account. If the remote node is a VAX, these parameters are optional. If you do not specify accounting information, the VMS system uses the default DECnet directory.

userid is your user code. This field contains thirty-nine ASCII-7 characters. If your user code is less than thirty-nine characters, left-justify the field.

passwd is your password. This field contains thirty-nine ASCII-7 characters. If your password is less than thirty-nine characters, left-justify the field.

acct is your account or charge code. This field contains thirty-nine ASCII-7 characters. If your account is less than thirty-nine characters, left-justify the field. If you don't plan to specify an account, pass spaces for this argument.

mode is a one-word integer that indicates the mode in which you plan to use the file after it is opened. This version of the DIL allows you to read the file, write the file or append data to the file. Check with your system manager to make sure that the FAL on the file's computer allows the access mode that you have in mind.

Mode	DIL Name
Read	MODE-READ
Write	MODE-WRITE
Append data to this file	MODE-APPEND

dattyp is a one-word integer that indicates the data type of the file. The DIL name for this argument is:

TYPE-ASCII

recfor is a one-word integer that gives the record format. This argument, and the next three arguments, refer to the record which you plan to process after opening the file with the ROPEN routine.

Record Format	DIL Name
Undefined	RFM-UNDEFINED
Fixed length	RFM-FIXED
Variable length	RFM-VARIABLE
Variable length with fixed length control (VFC)	RFM-VFC
Stream	RFM-STREAM

recatt is a one-word integer that indicates the record attributes of the file you plan to process.

Record Attribute	DIL Name
Unspecified	RAT-UNSPECIFIED
Implied <LF><CR> envelope	RAT-ENVELOPE
VMS printer carriage control	RAT-PRINT
FORTRAN carriage control	RAT-FORTRAN

NOTE

If you plan to use the RFA Routines to read a record from another system, you can use RFM-UNDEFINED or RAT-UNSPECIFIED values for the "recfor" and "recatt" arguments. You only have to specify values for these arguments if you plan to write a file on a VMS system.

recsiz is a one-word integer that gives the record size, measured in bytes of the size listed as the record size unit (below). If you plan to write the file, this is its maximum record size. If you plan to read or append data to the file or if you don't specify a maximum record size, this argument must be zero.

runit is a one-word integer that gives the record size unit. An ASCII record ignores this argument.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-TOOMANY	You attempted too many links. The DIL allows a maximum of 20 open links. On TOPS-20 systems, however, a non-privileged user can only open a maximum of four links.
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-UNSFILETYPE	You attempted to write a file whose file type is unsupported on the remote system.
DIT-FILEINUSE	The file is already being processed by another program.
DIT-NOFILE	The file does not exist or is not available to you.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

7.3.2 RREAD - Read Data From a Remote File

PURPOSE:

The RREAD routine reads a record from a file opened with the ROPEN routine.

CALL FORMAT:

COBOL: ENTER MACRO RREAD USING fnum, dunit, maxsiz, bufnam.

FORTRAN: status = RREAD (fnum, dunit, maxsiz, bufnam)

where:

fnum is a one-word integer that gives the number of the file that you want to read. File number is assigned by the ROPEN routine.

dunit is a one-word integer that gives the data unit size. This argument is always zero.

maxsiz is a one-word integer that specifies the maximum record size, in characters. When RREAD finishes processing, this argument contains the number of characters actually read by the routine.

bufnam is an ASCII-7 character string. This argument tells where the routine places the data that it reads.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-EOF	The routine reached the end of the file it was reading.
DIT-OVERRUN	The record being read is too large to fit into the user buffer area.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

7.3.3 RWRITE - Write Data To a Remote File

PURPOSE:

The RWRITE routine writes a record into a file opened by ROPEN.

CALL FORMAT:

COBOL: ENTER MACRO RWRITE USING fnum, dunit, length, data.

FORTRAN: status = RWRITE (fnum, dunit, length, data)

where:

fnum is a one-word integer that gives the file number of the file that you want to write. File number is assigned by the ROPEN routine.

dunit is a one-word integer that gives the data unit size. This argument is always zero.

length is a one-word integer that specifies the data length, in characters.

data is an ASCII-7 character string. This argument is the data to write.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-NOFILE	The file does not exist or is not available to you.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

7.3.4 RCLOSE - Close a Remote File

PURPOSE:

The RCLOSE routine closes the file that you opened with the ROPEN routine.

CALL FORMAT:

COBOL: ENTER MACRO RCLOSE USING fnum, clopt.

FORTRAN: status = RCLOSE (fnum, clopt)

where:

fnum is a one-word integer that gives the file number of the file that you want to close. File number is assigned by the ROPEN routine.

clopt is a one-word integer that specifies the close option. This argument tells what to do with the file once it's closed.

Close Option	DIL Name
No special action	OPT-NOTHING
Submit file for remote batch processing	OPT-SUBMIT
Submit file for remote printing	OPT-PRINT
Delete the remote file	OPT-DELETE

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines returns status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

7.3.5 RDEL - Delete a File

PURPOSE:

The RDEL routine deletes a file. You can only delete a closed file. If you want to delete an open file, delete it using the RCLOSE routine with the OPT-DELETE close option.

CALL FORMAT:

COBOL: ENTER MACRO RDEL USING fnam, userid, passwd, acct.

FORTRAN: STATUS = RDEL (fnam, userid, passwd, acct)

where:

fnam is an ASCII-7 character string that gives the name of the file to be deleted. The file name must also include the node name of the file's system of origin.

NOTE

The next three arguments supply accounting information. If the remote node is a DECSYSTEM-20 or DECsystem-10, you must supply a userid and password. You must also specify an account unless the remote system sets a default account. If the remote node is a VAX, these parameters are optional. If you do not specify accounting information, the VMS system uses the default DECnet directory.

userid is your user code. This field contains thirty-nine ASCII-7 characters. If your user code is less than thirty-nine characters, left-justify the field.

passwd is your password. This field contains thirty-nine ASCII-7 characters. If your password is less than thirty-nine characters, left-justify the field.

acct is your account. This field contains thirty-nine ASCII-7 characters. If your account is less than thirty-nine characters, left-justify the field.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-NOFILE	The file does not exist or is not available to you.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

7.3.6 RSUB - Submit a File For Batch Processing

PURPOSE:

The RSUB routine submits a remote file for batch processing on the remote system. You may only submit closed files for processing. If you want to submit an open file for batch processing, use the RCLOSE routine with the OPT-SUBMIT close option.

CALL FORMAT:

COBOL: ENTER MACRO RSUB USING fnam, userid, passwd, acct.

FORTRAN: status = RSUB (fnam, userid, passwd, acct)

where:

fnam is an ASCII-7 character string that gives the name of the file to be submitted. The file name must also include the node name of the file's system of origin.

NOTE

The next three arguments supply accounting information. If the remote node is a DECSYSTEM-20 or DECsystem-10, you must supply a userid and password. You must also specify an account unless the remote system sets a default account. If the remote node is a VAX, these parameters are optional. If you do not specify accounting information, the VMS system uses the default DECnet directory.

userid is your user code. This field contains thirty-nine ASCII-7 characters. If your user code is less than thirty-nine characters, left-justify the field.

passwd is your password. This field contains thirty-nine ASCII-7 characters. If your password is less than thirty-nine characters, left-justify the field.

acct is your account. This field contains thirty-nine ASCII-7 characters. If your account is less than thirty-nine characters, left-justify the field.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-NOFILE	The file does not exist or is not available to you.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

7.3.7 RPRINT - Print a File

PURPOSE:

The RPRINT routine prints a remote file at the remote system. You can only print closed files. If you want to print an open file use the RCLOSE routine with the OPT-PRINT close option.

CALL FORMAT:

COBOL: ENTER MACRO RPRINT USING fnam, userid, passwd, acct.

FORTRAN: status = RPRINT (fnam, userid, passwd, acct)

where:

fnam is an ASCII-7 character string that gives the name of the file to be printed. The file name must also include the node name of the file's system of origin.

NOTE

The next three arguments supply accounting information. If the remote node is a DECSYSTEM-20 or DECsystem-10, you must supply a userid and password. You must also specify an account unless the remote system sets a default account. If the remote node is a VAX, these parameters are optional. If you do not specify accounting information, the VMS system uses the default DECnet directory.

userid is your user code. This field contains thirty-nine ASCII-7 characters. If your user code is less than thirty-nine characters, left-justify the field.

passwd is your password. This field contains thirty-nine ASCII-7 characters. If your password is less than thirty-nine characters, left-justify the field.

acct is your account. This field contains thirty-nine ASCII-7 characters. If your account is less than thirty-nine characters, left-justify the field.

STATUS CODES:

For COBOL programs, the DIL returns status codes in the dil-stat variable defined by the DILINI routine. For FORTRAN programs, the DIL routines return status codes as an integer function value.

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-NOFILE	The file does not exist or is not available to you.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

7.4 TOPS-20 REMOTE FILE ACCESS EXAMPLES

7.4.1 TOPS-20 COBOL Remote File Access Example

```
IDENTIFICATION DIVISION.

PROGRAM-ID.

        CDAP36.

        This program opens a remote  file named DAP.TST and writes  an
        ASCII record into it,  closes the file,  reopens the file  and
        reads the record back and  then closes the file again.   Note,
        this program tries to write and read the file DAP.TST using  a
        directory called PS:<DIL-TEST>.   If this  directory does  not
        exist, it must be created as a valid LOGIN directory.


ENVIRONMENT DIVISION.


CONFIGURATION SECTION.

SOURCE-COMPUTER.

        DECSYSTEM-20.

OBJECT-COMPUTER.

        DECSYSTEM-20.

DATA DIVISION.

WORKING-STORAGE SECTION.

01  INTERFACE-FILES.
    COPY DIT OF "SYS:DIL.LIB".
    COPY DIL OF "SYS:DIL.LIB".

01  DILINI-PARAMETERS.
* Dilini is necessary for DECsystem-10 and DECSYSTEM-20 COBOL only.
    05  DIL-INIT-STATUS PIC S9(10) COMP.
    05  DIL-STATUS PIC S9(10) COMP.
    05  DIL-MESSAGE PIC S9(10) COMP.
    05  DIL-SEVERITY PIC S9(10) COMP.

* File and directory description fields

01  FILE-NAME PIC X(39) VALUE 'PS:<DIL-TEST>DAP.TST' DISPLAY-7.
01  USERID USAGE DISPLAY-7 PIC X(39) VALUE 'DIL-TEST'.
01  PASSWD USAGE DISPLAY-7 PIC X(39) VALUE SPACES.
01  ACCT USAGE DISPLAY-7 PIC X(39) VALUE SPACES.

* Record and file description fields

01  FILE-NUMBER USAGE COMP PIC S9(10).
01  RECORD-UNIT-SIZE USAGE COMP PIC S9(10) VALUE 0.
01  RECORD-SIZE USAGE COMP PIC S9(10) VALUE 100.

01  DATA-RECORD USAGE DISPLAY-7 PIC X(100).
```

```
PROCEDURE DIVISION.

* Set up for status code values, using DILINI routine

    ENTER MACRO DILINI USING DIL-INIT-STATUS, DIL-STATUS,
                             DIL-MESSAGE, DIL-SEVERITY.

    IF DIL-INIT-STATUS NOT = 1
        DISPLAY "? Invalid return code from DILINI routine = " DIL-INIT-STATUS.

* Request the password

    DISPLAY " Enter the password: " WITH NO ADVANCING ACCEPT PASSWD.

* Open file DAP.TST for output

    ENTER MACRO ROPEN USING FILE-NUMBER, FILE-NAME, USERID, PASSWD, ACCT,
                            DIT-MODE-WRITE, DIT-TYPE-ASCII, DIT-RFM-STREAM,
                            DIT-RAT-UNSPECIFIED, RECORD-SIZE, RECORD-UNIT-SIZE.

    DISPLAY " ROPEN Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? ROPEN: unsuccessful status return "
        STOP RUN.

* Accept a record and write it to the file

    DISPLAY " Enter data for the record for the remote file: ".
    ACCEPT DATA-RECORD.

    ENTER MACRO RWRITE USING FILE-NUMBER, RECORD-UNIT-SIZE,
                             RECORD-SIZE, DATA-RECORD.

    DISPLAY " RWRITE Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? RWRITE: unsuccessful status return. "
        STOP RUN.

* Close the file

    ENTER MACRO RCLOSE USING FILE-NUMBER, DIT-OPT-NOTHING.

    DISPLAY " RCLOSE Status return: ", DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? RCLOSE: unsuccessful status return."
        STOP RUN.

* Open the file to read the record

    ENTER MACRO ROPEN USING FILE-NUMBER, FILE-NAME, USERID, PASSWD, ACCT,
                            DIT-MODE-READ, DIT-TYPE-ASCII, DIT-RFM-STREAM,
                            DIT-RAT-UNSPECIFIED, RECORD-SIZE, RECORD-UNIT-SIZE.

    DISPLAY " ROPEN Status return: ", DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? ROPEN: unsuccessful status return."
        STOP RUN.
```

```
* Read the record

    MOVE SPACES TO DATA-RECORD.

    ENTER MACRO RREAD USING FILE-NUMBER, RECORD-UNIT-SIZE,
                            RECORD-SIZE, DATA-RECORD.

    DISPLAY " RREAD returned ", DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? RREAD: unsuccesful status return."
        STOP RUN.

    DISPLAY " The record was: ".
    DISPLAY DATA-RECORD.

* Close the file

    ENTER MACRO RCLOSE USING FILE-NUMBER, DIT-OPT-NOTHING.

    DISPLAY " RCLOSE Status return: ", DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? RCLOSE: unsuccessful status return."
        STOP RUN.

    DISPLAY " CDAP36 successful. ".

    STOP RUN.
```

7.4.2 TOPS-20 FORTRAN Remote File Access Example

```
C FDAP36

C       This program opens a remote  file named DAP.TST and writes  an
C       ASCII record into it,  closes the file,  reopens the file  and
C       reads the record back and  then closes the file again.  Note,
C       this program tries to write and  read the file DAP.TST from  a
C       directory called PS:<DIL-TEST>.  If this  directory does  not
C       exist, it must be created as a LOGIN directory.

C Use the DIL interface support files.

        INCLUDE 'SYS:DITV7'
        INCLUDE 'SYS:DILV7'

C File and directory description fields
        INTEGER FILNAM (8), USERID (8), PASSWD (8), ACCT (8), FILNUM
C Sending and receiving data records
        INTEGER DATA1 (20), DATA2 (20)
C DIL Status code
        INTEGER DILSTS
```

```
C Record size and record unit size
        INTEGER RECSIZ, RUNTSZ
        DATA FILNAM /'PS:<D', 'IL-TE', 'ST>DA', 'P.TST', '     ',
     1  '    ', '     ', '     '/
        DATA USERID /'DIL-T','EST  ', '     ', '     ',
     1  '     ', '     ', '     ', '     '/
        DATA PASSWD /'     ', '     ', '     ', '     ', '     ',
     1  '     ', '     ', '     '/
        DATA ACCT /'     ', '     ', '     ', '     ',
     1  '     ', '     ', '     ', '     '/

C Program messages

200     FORMAT (' ROPEN status return: ', I12)
201     FORMAT (' RWRITE status return: ', I12)
202     FORMAT (' RCLOSE status return: ', I12)
203     FORMAT (' RREAD status return: ', I12)
700     FORMAT (' ? Invalid status returned... ')

C Request the password

100     FORMAT (' Enter the password: ')
        WRITE (5,100)
105     FORMAT (8A5)
        ACCEPT 105, PASSWD

C Open file DAP.TST for output

        RUNTSZ = 0
        RECSIZ = 100

        DILSTS = ROPEN (FILNUM, FILNAM, USERID, PASSWD, ACCT, MWRITE,
     1                  TASCII, FSTM, AUNSPC, RECSIZ, RUNTSZ)

        WRITE (5,200) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 115
        WRITE (5,700)
        STOP

C Accept a record and write it to the file

110     FORMAT (' Enter data for the record: ')
115     WRITE (5,110)
120     FORMAT (20A5)
        ACCEPT 120, DATA1

        DILSTS = RWRITE (FILNUM, RUNTSZ, RECSIZ, DATA1)

        WRITE (5,201) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 125
        WRITE (5,700)
        STOP
```

```
C Close the file

125     DILSTS = RCLOSE (FILNUM, ONTHNG)

        WRITE (5,202) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 130
        WRITE (5,700)
        STOP

C Open the file to read the record

130     DILSTS = ROPEN (FILNUM, FILNAM, USERID, PASSWD, ACCT, MREAD,
     1                  TASCII, FSTM, AUNSPC, RECSIZ, RUNTSZ)

        WRITE (5,200) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 135
        WRITE (5,700)
        STOP

C Read the record

135     DILSTS = RREAD (FILNUM, RUNTSZ, RECSIZ, DATA2)

        WRITE (5,203) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 145
        WRITE (5,700)
        STOP

140     FORMAT (' The record read was: ')
145     WRITE (5, 140)
146     FORMAT (' ', 20A5)
        WRITE (5,146) DATA2

C Close the file

        DILSTS = RCLOSE (FILNUM, ONTHNG)

        WRITE (5,202) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 155
        WRITE (5,700)
        STOP

150     FORMAT (' FDAP36 successful ')
155     WRITE (5,150)
        STOP
        END
```

7.5 TOPS-10 REMOTE FILE ACCESS EXAMPLES

7.5.1 TOPS-10 COBOL Remote File Access Example

```
IDENTIFICATION DIVISION.

PROGRAM-ID.

        CDAP36.

        This program opens a remote file named DAP.TST and writes an
        ASCII record into it, closes the file, reopens the file and
        reads the record back and then closes the file again.  Note,
        this program tries to write and read the file DAP.TST using a
        directory called MYNODE::[10,105].  If this directory does not
        exist, it must be created as a valid LOGIN directory.


ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER.

        DECSYSTEM-10.

OBJECT-COMPUTER.

        DECSYSTEM-10.

DATA DIVISION.

WORKING-STORAGE SECTION.

01  INTERFACE-FILES.
    COPY DIT OF "SYS:DIL,LIB".
    COPY DIL OF "SYS:DIL,LIB".

01  DILINI-PARAMETERS.
* Dilini is necessary for DECsystem-10 and DECSYSTEM-20 COBOL only.
    05  DIL-INIT-STATUS PIC S9(10) COMP.
    05  DIL-STATUS PIC S9(10) COMP.
    05  DIL-MESSAGE PIC S9(10) COMP.
    05  DIL-SEVERITY PIC S9(10) COMP.

* File and directory description fields

01  FILE-NAME PIC X(39) VALUE 'MYNODE::DAP.TST' DISPLAY-7.
01  USERID USAGE DISPLAY-7 PIC X(39) VALUE '[10,105]'.
01  PASSWD USAGE DISPLAY-7 PIC X(39) VALUE SPACES.
01  ACCT USAGE DISPLAY-7 PIC X(39) VALUE SPACES.

* Record and file description fields

01  FILE-NUMBER USAGE COMP PIC S9(10).
01  RECORD-UNIT-SIZE USAGE COMP PIC S9(10) VALUE 0.
01  RECORD-SIZE USAGE COMP PIC S9(10) VALUE 95.

01  DATA-RECORD USAGE DISPLAY-7 PIC X(100).
```

```
PROCEDURE DIVISION.

* Set up for status code values, using DILINI routine

    ENTER MACRO DILINI USING DIL-INIT-STATUS, DIL-STATUS,
                             DIL-MESSAGE, DIL-SEVERITY.

    IF DIL-INIT-STATUS NOT = 1
        DISPLAY "? Invalid return code from DILINI routine = " DIL-INIT-STATUS.

* Request the password

    DISPLAY " Enter the password: " WITH NO ADVANCING ACCEPT PASSWD.

* Open file DAP.TST for output

    ENTER MACRO ROPEN USING FILE-NUMBER, FILE-NAME, USERID, PASSWD, ACCT,
                            DIT-MODE-WRITE, DIT-TYPE-ASCII, DIT-RFM-STREAM,
                            DIT-RAT-UNSPECIFIED, RECORD-SIZE, RECORD-UNIT-SIZE.

    DISPLAY " ROPEN Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? ROPEN: unsuccessful status return "
        STOP RUN.

* Accept a record and write it to the file

    DISPLAY " Enter data for the record for the remote file: ".
    ACCEPT DATA-RECORD.

    ENTER MACRO REWRITE USING FILE-NUMBER, RECORD-UNIT-SIZE,
                          RECORD-SIZE, DATA-RECORD.

    DISPLAY " RWRITE Status return: " DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? RWRITE: unsuccessful status return, "
        STOP RUN.

* Close the file

    ENTER MACRO RCLOSE USING FILE-NUMBER, DIT-OPT-NOTHING.

    DISPLAY " RCLOSE Status return: ", DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? RCLOSE: unsuccessful status return."
        STOP RUN.

* Open the file to read the record

    MOVE 100 TO RECORD-SIZE.

    ENTER MACRO ROPEN USING FILE-NUMBER, FILE-NAME, USERID, PASSWD, ACCT,
                      DIL-MODE-READ, DIT-TYPE-ASCII, DIT-RFM-STREAM,
                      DIT-RAT-UNSPECIFIED, RECORD-SIZE, RECORD-UNIT-SIZE.

    DISPLAY " ROPEN Status return: ", DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
       DISPLAY "? ROPEN: unsuccessful status return."
       STOP RUN.
```

```
* Read the record

    MOVE SPACES TO DATA-RECORD.

    ENTER MACRO RREAD USING FILE-NUMBER, RECORD-UNIT-SIZE,
                            RECORD-SIZE, DATA-RECORD.

    DISPLAY " RREAD returned ", DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? RREAD: unsuccessful status return."
        STOP RUN.

    DISPLAY " The record was: ".
    DISPLAY DATA-RECORD.

* Close the file

    ENTER MACRO RCLOSE USING FILE-NUMBER, DIT-OPT-NOTHING.

    DISPLAY " RCLOSE status return: ", DIL-STATUS.
    IF DIL-SEVERITY NOT = STS-K-SUCCESS
       AND DIL-SEVERITY NOT = STS-K-INFO
        DISPLAY "? RCLOSE: unsuccessful status return."
        STOP RUN.

    DISPLAY " CDAP36 successful. ".

    STOP RUN.
```

7.5.2 TOPS-10 FORTRAN Remote File Access Example

```
C FDAP36

C       This program opens a remote file named DAP.TST and writes an
C       ASCII record onto it.  Closes the file, reopens the file and
C       reads the record back and then closes the file again.  Note,
C       this program tries to write and read the file DAP.TST from a
C       directory called MYNODE::[10,105].  If this directory does not
C       exist, it must be created as a LOGIN directory.


C Use the DIL interface support files.

        INCLUDE 'SYS:DITV7'
        INCLUDE 'SYS:DILV7'

C File and directory description fields
        INTEGER FILNAM (8), USERID (8), PASSWD (8), ACCT (8), FILNUM
C Sending and receiving data records
        INTEGER DATA1 (20), DATA2 (20)
C DIL Status code
        INTEGER DILSTS
```

```
C Record size and record unit size
        INTEGER RECSIZ, RUNTSZ

        DATA FILNAM /'MYNOD', 'E::DA', 'P.TST','    ','    ',
     1  '     ', '     ', '     '/
        DATA USERID /'10,10','5    ', '     ', '  ',
     1  '     ', '     ', '     '/
        DATA PASSWD /'     ', '     ', '   ', '   ', '   ', '   ',
     1  '     ', '     ', '     '/
        DATA ACCT /'   ', '   ', '   ', '   ',
     1  '    ', '    ', '    ', '    '/

C Program messages

200     FORMAT (' ROPEN status return: ', I12)
201     FORMAT (' RWRITE status return: ', I12)
202     FORMAT (' RCLOSE status return: ', I12)
203     FORMAT (' RREAD status return: ', I12)
700     FORMAT (' ? Invalid status returned... ')

C Request the password

100     FORMAT (' Enter the password: ')
        WRITE (5,100)
105     FORMAT (8A5)
        ACCEPT 105, PASSWD

C Open file DAP.TST for output

        RUNTSZ = 0
        RECSIZ = 95

        DILSTS = ROPEN (FILNUM, FILNAM, USERID, PASSWD, ACCT, MWRITE,
    1                   TASCII, FSTM, AUNSPC, RECSIZ, RUNTSZ)



        WRITE (5,200) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 115
        WRITE (5,700)
        STOP

C Accept a record and write it to the file

110     FORMAT (' Enter data for the record: ')
115     WRITE (5,110)
120     FORMAT (20A5)
        ACCEPT 120, DATA1

        DILSTS = RWRITE (FILNUM, RUNTSZ, RECSIZ, DATA1)

        WRITE (5,201) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 125
        WRITE (5,700)
        STOP
```

```
C Close the file

125     DILSTS = RCLOSE (FILNUM, ONTHNG)

        WRITE (5,202) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 130
        WRITE (5,700)
        STOP

C Open the file to read the record

130     RECSIZ = 100

        DILSTS = ROPEN (FILNUM, FILNAM, USERID, PASSWD, ACCT, MREAD,
     1                  TASCII, FSTM, AUNSPC, RECSIZ, RUNTSZ)

        WRITE (5,200) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 135
        WRITE (5,700)
        STOP

C Read the record

135     DILSTS = RREAD (FILNUM, RUNTSZ, RECSIZ, DATA2)

        WRITE (5,203) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 145
        WRITE (5,700)
        STOP

140     FORMAT (' The record read was: ')
145     WRITE (5, 140)
146     FORMAT (' ', 20A5)
        WRITE (5,146) DATA2

C Close the file

        DILSTS = RECLOSE (FILNUM, ONTHNG)

        WRITE (5,202) DILSTS
        IF (DILSTS .EQ. NORMAL) GO TO 155
        WRITE (5,700)
        STOP

150     FORMAT (' FDAP36 successful ')
155     WRITE (5,150)
        STOP
        END
```

CHAPTER 8
LINKING A TOPS–10/TOPS–20 PROGRAM

CHAPTER 8

LINKING A TOPS-10/TOPS-20 PROGRAM

8.1 DECsystem-10 LINKAGE INSTRUCTIONS

The linker associates your program with the Data Interchange Library routines called by the program. When you have successfully compiled your program, link it by typing the following command sequence:

```
.R LINK
*PROG
*/SEGMENT:LOW/SEARCH SYS:DIL,-
#SYS:B361LB, -
#SYS:XPORT/EXCLUDE:XFUNCT/GO

EXIT
.SAVE PROG
```

where: "PROG" is the name of your program.

8.2 DECSYSTEM-20 LINKAGE INSTRUCTIONS

The linker associates your program with the Data Interchange Library routines that your program calls. When you have successfully compiled your program, link it by typing the following command string:

```
@LINK
*USRPRG, SYS:DIL/SEARCH, SYS:XPORT/SEARCH/EXCLUDE:XFUNCT/GO
```

where: "USRPRG" is the name of your program.

8.3 DECSYSTEM-10 AND DECSYSTEM-20 OVERLAY INSTRUCTIONS

If your program is very large, you may want to overlay the program. For general information on overlaying programs, refer to the Link Manual and the language specific manuals for your system. When building an overlay tree, you must include the following modules in the root segment:

1. .LOPEN (for Task-to-Task Routines)
2. .UPEVT (for Task-to-Task Routines)
3. XPNUTL (for Remote File Access Routines)
4. XPNPSI (for Remote File Access Routines)

CHAPTER 9
VMS DATA CONVERSION

CHAPTER 9

VMS DATA CONVERSION

9.1 DATA CONVERSION FROM VMS COBOL

The information included in this section assumes:

-You are writing a COBOL program
-You plan to use the program on a VMS system

To store an FFD, pass a record or perform an error check, you must represent several data items in your program. Users generally allocate space for foreign fields and records in WORKING-STORAGE.

9.1.1 Including the Interface Support Files

On VMS systems, there are two different classes of Interface Support files for each supported language. The first class of files includes native VMS-type names for each of the various codes for VMS COBOL and VMS FORTRAN. The second class includes files with TOPS-10/TOPS-20 COBOL compatible names for VMS COBOL, and files which include ANSI Standard names for the interface to VMS FORTRAN. The Interface Support files for VMS are provided as a text library called DIL.TLB. This library includes the support files for both VMS COBOL and VMS FORTRAN.

Native VMS COBOL

You can use the COBOL COPY verb to retrieve information from DIL.TLB at compilation time. There are three library elements for Native VMS COBOL in DIL.TLB. The elements are DIL$COBOL, DIT$COBOL, and DIX$COBOL.

The library element DIL$COBOL defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$COBOL. Severity codes and system codes are defined in element DIL$COBOL. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration:

```
COPY DIL$COBOL OF "SYS$LIBRARY:DIL.TLB".
```

In the following example the DIL$COBOL element of the library is retrieved and included in your program:

```
01  Interface-files.
    COPY DIL$COBOL OF "SYS$LIBRARY:DIL.TLB".
```

The library element DIX$COBOL defines codes specific to the Data Conversion Routines. These codes include the DIX status codes which are standard VMS condition values. Also included are data type names for each supported data type. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration as described above for the DIL$COBOL element:

```
COPY DIX$COBOL OF "SYS$LIBRARY:DIL.TLB".
```

For programs which use the Data Conversion Routines with native VMS names, you must include both the DIL$COBOL and DIX$COBOL library elements.

TOPS-10/TOPS-20 Compatible COBOL

If you want to write a program which can be easily transported to a DECsystem-10 or a DECSYSTEM-20, you may want to include the TOPS-10/TOPS-20 compatible names in your program rather than the native VMS names.

You can use the COBOL COPY verb to retrieve information from DIL.TLB at compilation time. There are three library elements for TOPS-10/TOPS-20 compatible COBOL in DIL.TLB. The elements are DIL$COBOL_20, DIT$COBOL_20, and DIX$COBOL_20.

The library element DIL$COBOL_20 defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL)is defined in element DIL$COBOL_20. Severity codes and system codes are defined in element DIL$COBOL_20. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration:

```
COPY DIL$COBOL_20 OF "SYS$LIBRARY:DIL.TLB".
```

In the following example, the DIL$COBOL_20 element of the library is retrieved and included in your program:

```
01  Interface-files.
    COPY DIL$COBOL_20 OF "SYS$LIBRARY:DIL.TLB".
```

The library element DIX$COBOL_20 defines codes specific to the data conversion routines. The codes include the DIX status codes in the compatible COBOL format which provides only the condition identification portion of the status code. Also included are data type names for each supported data type. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration as shown above for the DIL$COBOL_20 element:

```
COPY DIX$COBOL_20 OF "SYS$LIBRARY:DIL.TLB".
```

For programs which use the data conversion routines with the compatible names, you must include both the DIL$COBOL_20 and DIX$COBOL_20 library elements.

9.1.2 Storing an FFD

The FFD occupies three full longwords of VMS memory. To store an FFD you must define a data item with the following format:

```
01  your-ffd.
    03  your-ffd-tbl   PIC S9(9) USAGE COMPUTATIONAL OCCURS 3.
```

When you call the DIX$MAK_DES_DET routine to build the FFD, use your-ffd-tbl as the FFD to be returned. To pass the FFD to the routine, pass your-ffd-tbl with the subscript 1: your-ffd-tbl (1).

9.1.3 Passing a Record to the DIX$MAK_DES_DET Routine

To pass a record to the DIX$MAK_DES_DET routine, you must know how the record (containing the field that you want to convert) would be declared on its native system. You must then represent the record in your program on the local system. The record can be represented as any word-aligned group level data item. To pass the record to DIX$MAK_DES_DET, specify this group item as the record name ("rec") in the call to DIX$MAK_DES_DET.

To figure the size of the record, count the number of bits on its native system; make the record name on the foreign system at least that large. In the following example, the VAX/VMS record "rec" contains 640 bits of information.

```
01  rec   PIC S9(9)   USAGE COMPUTATIONAL OCCURS 20 TIMES.
```

To pass "rec" to the DIX$MAK_DES_DET routine simply include it in your call to the routine.

```
CALL "DIX$MAK_DES_DET" USING ffd (1), rec, sysor, bysiz, byoff,
     bioff, type, lngth, scale GIVING stat.
```

9.1.4 Checking for Errors

A call to the Data Conversion Routines from VMS COBOL looks like the following example:

```
CALL "DIX$MAK_DES_DET" USING ffd (1), rec, sysor, bysiz, byoff,
     bioff, type, lngth, scale GIVING stat.
```

"Stat", used above, is a standard VMS condition value and should be defined as:

```
01  stat   PIC S9(9) USAGE COMPUTATIONAL.
```

The routine performs the error check and places the resultant status value into the data item stat. The following example shows a simple error check:

```
CALL "DIX$MAK_DES_DET" USING ffd (1), rec, sysor, bysiz, byoff,
     bioff, type, lngth, scale GIVING stat.
IF stat IS FAILURE
     THEN DISPLAY "failure".
```

On VMS systems, the LIB$MATCH_COND function is available to compare two status codes and determine if they refer to the same condition. See the most recent version of the VAX-11 Run-Time Library Reference Manual for further information.

Using LIB$MATCH_COND, you can compare "stat" to the DIX status values defined in the DIL Interface Support files to identify which error occurred.

NOTE

If you are using the TOPS-10/TOPS-20 Compatible COBOL Interface Support files, you cannot use LIB$MATCH_COND to compare the status codes. Use the error checking procedure described in the TOPS-10/TOPS-20 COBOL section of this manual.

9.2 DATA CONVERSION FROM VMS FORTRAN

The information included in this section assumes

-You are writing a FORTRAN program
-You plan to use the program on a VMS system

The section explains methods to store a Foreign Field Descriptor, pass a record to the conversion routines and perform an error check.

9.2.1 Including the Interface Support Files

VMS systems have two different classes of Interface Support files for each supported language. The first class of files includes native VMS-type names for each of the various codes for VMS COBOL and VMS FORTRAN. The second class includes files with TOPS-10/TOPS-20 COBOL compatible names for VMS COBOL, and files which include ANSI Standard names for the interface to VMS FORTRAN. The Interface Support files for VMS are provided as a text library called DIL.TLB. This library includes the support files for both VMS COBOL and VMS FORTRAN.

Native VMS FORTRAN

You can use the FORTRAN INCLUDE statement to retrieve information from DIL.TLB at compilation time. There are three library elements for native VMS FORTRAN in DIL.TLB. The elements are DIL$FORTRAN, DIT$FORTRAN, and DIX$FORTRAN.

The library element DIL$FORTRAN defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$FORTRAN. Severity codes and system codes are defined in element DIL$FORTRAN. To define these names in your program, include the statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIL$FORTRAN)'
```

This retrieves the DIL$FORTRAN element of the library file and includes it in your program.

The library element DIX$FORTRAN defines codes specific to the Data Conversion Routines. The codes include the DIX status codes which are standard VMS condition values. Also included are data type names for each supported data type. To define these names in your program, include the statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIX$FORTRAN)'
```

For programs which use the Data Conversion routines with the native VMS names, you must include both the DIL$FORTRAN and DIX$FORTRAN library elements.

ANSI Standard Compatible FORTRAN

If you want to write a program which can be easily transported to another system, you may want to include the ANSI Standard FORTRAN names in your program rather than the native VMS names.

You can use the FORTRAN INCLUDE statement to retrieve information from DIL.TLB at compilation time. There are three library elements for ANSI Standard FORTRAN in DIL.TLB. The elements are DIL$ANSI_FORTRAN, DIT$ANSI_FORTRAN, and DIX$ANSI_FORTRAN.

The library element DIL$ANSI_FORTRAN defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$ANSI_FORTRAN. Severity codes and system codes are defined in element DIL$ANSI_FORTRAN. To define these names in your program, include the statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIL$ANSI_FORTRAN)'
```

This retrieves the DIL$ANSI_FORTRAN element of the library file and includes it in your program.

The library element DIX$ANSI_FORTRAN defines codes specific to the Data Conversion Routines. The codes include the DIX status codes which are standard VMS condition values. Also included are data type names for each supported data type. To define these names in your program, include the following statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIX$ANSI_FORTRAN)
```

For programs which use the data conversion routines with the compatible names, you must include both the DIL$ANSI_FORTRAN and DIX$ANSI_FORTRAN library elements.

9.2.2 Storing an FFD

The FFD occupies three full longwords of VMS memory. To store an FFD, you must first dimension an array of type integer with length 3.

```
INTEGER    dilffd  (3)
```

When you call the DIX$MAK_DES_DET routine to build an FFD, use dilffd as the value for the FFD to be returned. To pass this value to the routine, always pass the entire array, specifying just the variable name, dilffd, not the array entry dilffd (1).

9.2.3 Passing a Record to the Data Conversion Routines

To convert a field in a record, you must know how that field is represented on both its own system and the system where you plan to use the converted field. You create a space for the record on the local system by declaring an integer array big enough to contain the record. To figure the array size, count the number of bits used by the record on its native system; make the array on the local system at least that large. In the following example, this VAX/VMS record contains 640 bits of information:

```
INTEGER    rec  (20)
```

To pass the record to one of the Data Conversion Routines, pass the entire array. The example, below, shows rec being passed to the DIX$MAK_DES_DET routine.

```
status = DIX$MAK_DES_DET (ffd, rec, sysor, bysiz, byoff, bioff,
1        type, lngth, scale)
```

9.2.4 Checking for Errors

The Data Conversion Routines return a status value when called. To perform an error check on a Data Conversion Routine, first declare an integer where the routine can place a status value.

```
INTEGER    status
```

A simple call with an error check might then be:

```
status = DIX$MAK_DES_DET (ffd, rec, sysor, bysiz, byoff, bioff
1        type, lngth, scale)
IF (status .NE. SS$_NORMAL) GOTO 100
.
.
.
100  TYPE 101
101  FORMAT (' error occurred')
.
.
.
```

On VMS systems, the LIB$MATCH_COND function is available to compare two status codes and determine if they refer to the same condition. See the most recent version of the VAX-11 Run-Time Library Reference Manual for further information.

Using LIB$MATCH_COND you can compare status to the DIX status values defined in the VMS FORTRAN Interface Support file, to determine which error occurred. Note that SS$_NORMAL, used above, is the success status defined in the VMS FORTRAN Interface Support files.

9.3 VMS DATA CONVERSION REFERENCE

9.3.1 DIX$MAK_DES_DET - Create an FFD

PURPOSE:

The DIX$MAK_DES_DET routine reads the detailed description information which you supply and builds a Foreign Field Descriptor for a native or foreign field.

CALL FORMAT:

COBOL: CALL "DIX$MAK_DES_DET" USING ffd (1), rec, sysor, bysiz, byoff, bioff, type, lngth, scale GIVING stat.

FORTRAN: status = DIX$MAK_DES_DET (ffd, rec, sysor, bysiz, 1 byoff, bioff, type, lngth, scale)

where:

ffd — is the Foreign Field Descriptor to be returned. This argument is the data item where the routine places the resultant FFD. A Foreign Field Descriptor consists of three one-word integers.

rec — is the record that contains the field being described.

COBOL: This argument can be any word-aligned data item.

FORTRAN: This argument is usually an integer array.

sysor — is a long integer that gives the system of origin of the field to be converted. DIL Names that can be used for this argument are:

SYS-10-20 — for a record defined for TOPS-10 or TOPS-20.

SYS-VMS — for a record defined for VMS.

bysiz — is a long integer that gives the byte size of the record for this call to DIX$MAK_DES_DET. All VMS records have a byte size of 8. A TOPS-10 or TOPS-20 COBOL record can have a byte size of 6, 7, 9 or 36. A TOPS-10 or TOPS-20 FORTRAN record can have a byte size of 7 or 36. See Appendix A for further information.

byoff — is a long integer that gives the byte offset to the field within the record. The byte offset is the number of bytes in the record (of the size specified in "bysiz") that precede the field that you want to convert.

bioff — is a long integer that gives the bit offset. This argument is not currently necessary; it should always be zero.

type — is a long integer that gives the data type of the field that is being converted. See Appendix A for a list of valid data type codes.

lngth — is a long integer that gives the length of the field: in characters for string fields. This argument is required for some data types, it must be zero for all other data types. See Appendix A for further information.

scale — is a long integer that gives the scale factor of a fixed-point binary field. This argument tells how many decimal places the decimal point should be moved to the left. A negative scale factor means that you want to move the decimal point to the right. You must specify a scale factor for fixed-point conversions. Do not specify scale factor for any other type of field.

STATUS CODES:

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-INVBYTSIZ	Invalid byte size specified.

RELATED ROUTINES:

DIX$BY_DET: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 9.3.3 for a description of DIX$BY_DET.

9.3.2 DIX$BY_DIX_DES - Perform General Conversion

PURPOSE:

The DIX$BY_DIX_DES routine performs any type of data conversion that can be done by the DIL.

DIX$BY_DIX_DES is a general purpose routine. DIX$BY_DIX_DES accepts a Foreign Field Descriptor for the source and destination fields. If the DIL has the capability to convert that type of field, DIX$BY_DIX_DES converts the field.

CALL FORMAT:

COBOL: CALL "DIX$BY_DIX_DES USING" sffd (1), dffd (1)
GIVING stat.

FORTRAN: status = DIX$BY_DIX_DES (sffd, dffd)

where:

sffd is a Foreign Field Descriptor describing the source field. A Foreign Field Descriptor consists of three long integers.

dffd is a Foreign Field Descriptor describing the destination field. A Foreign Field Descriptor consists of three long integers.

STATUS CODES:

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-INVALCHAR	Invalid character in source field or conversion table.
DIX-GRAPHIC	Graphic character changed in conversion.
DIX-FMTLOST	Format effector gained or lost in conversion.
DIX-NONPRINT	Non-printing character gained or lost in conversion.
DIX-TRUNC	String too long for destination -- truncated.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-UNSIGNED	Negative value moved to unsigned field.
DIX-ROUNDED	Result is rounded.
DIX-UNNORM	Floating-point number improperly normalized.
DIX-INVDNUMCHR	Invalid display numeric character in source field.
DIX-INVDNUMSGN	Invalid display numeric sign in source field.
DIX-INVPDDGT	Invalid packed decimal digit in source field.
DIX-INVPDSGN	Invalid packed decimal sign in source field.

RELATED ROUTINES:

DIX$BY_DET: This routine allows you to convert a field without making a Foreign Field Descriptor for the field. See Section 9.3.3 for a description of DIX$BY_DET.

9.3.3 DIX$BY_DET - Convert a Field Without an FFD

PURPOSE:

The DIX$BY_DET routine allows you to convert a field without making an FFD for the field. DIX$BY_DET requires a detailed series of arguments; you must specify parameters for both the source and destination fields.

You should only use DIX$BY_DET in cases where the field will be converted a limited number of times or the program is table-driven. DIX$BY_DET creates a description for the field each time it processes the field. If you plan to convert a field many times during a run, it is quicker to create an FFD for the field and convert it with one of the other single function conversion routines. Whereas DIX$BY_DET creates an FFD internally each time it processes, if you perform the conversion with an FFD (built using the DIX$MAK_DES_DET routine) you make the FFD only once.

CALL FORMAT:

COBOL:
```
CALL "DIX$BY_DET" USING srec, ssysor, sbysiz, sbyoff,
     sbioff, stype, slength, sscale, drec, dsysor,
     dbysiz, dbyoff, dbioff, dtype, dlngth, dscale
     GIVING stat.
```

FORTRAN:
```
status = DIX$BY_DET (srec, ssysor, sbysiz, sbyoff,
1  sbioff, stype, slength, sscale, drec, dsysor,
2  dbysiz, dbyoff, dbioff, dtype, dlngth, dscale)
```

where:

srec — is the source record that contains the field being described.

COBOL: This argument can be any word-aligned data item.

FORTRAN: This argument is usually an integer array.

ssysor — is a long integer that gives the system of origin of the source record. Possible DIL Names for this argument are:

SYS-10-20 — for a record defined for TOPS-10 or TOPS-20.

SYS-VMS — for a record defined for VMS.

sbysiz — is a long integer that gives the byte size of the source record for this call to the DIX$BY_DET routine. All VMS records have a byte size of 8. A TOPS-10 or TOPS-20 COBOL record can have a byte size of 6, 7, 9 or 36. A TOPS-10 or TOPS-20 FORTRAN record can have a byte size of 7 or 36. See Appendix A for further information.

sbyoff is a long integer that gives the byte offset to the field within the source record. The byte offset is the number of bytes in the source record (of the size specified in "sbysiz") that precede the field which you want to convert.

sbioff is a long integer that gives the bit offset. This argument is not currently used; it should always be zero.

stype is a long integer that gives the data type of the source field. See Appendix A for a list of valid data type codes.

slngth is a long integer that indicates the length of the source field: in characters for string fields. This argument is required for some data types, it must be zero for all other data types. See Appendix A for further information.

sscale is a long integer that gives the scale factor of the source field. It indicates the number of decimal places to move the decimal point to the left. A negative scale factor means that the decimal point will be moved to the right. You must specify a scale factor if you want to convert a fixed-point field. Specify zero for all other classes of data.

drec is the record that contains the record to be described. This argument is a long integer.

COBOL: This argument can be any word-aligned data item.

FORTRAN: This argument will usually be an integer array. You can pass either the whole array or an element of the array.

dsysor is a long integer that gives the system of origin of the destination record. Possible DIL Names for this argument are:

SYS-10-20 for a record defined for TOPS-10 or TOPS-20.

SYS-VMS for a record defined for VMS.

dbysiz is a long integer that gives the byte size of the destination record for this call to DIX$BY_DET. All VMS records have a byte size of 8. A TOPS-10 or TOPS-20 COBOL record can have a byte size of 6, 7, 9 or 36. A TOPS-10 or TOPS-20 FORTRAN record can have a byte size of 7 or 36. See Appendix A for further information.

dbyoff is a long integer that gives the byte offset of the destination record. The byte offset is the number of bytes in the record (of the size specified in "dbysiz") that precede the field that you want to convert.

dbioff is a long integer that gives the bit offset. This argument is not currently necessary; it should always be zero.

dtype is a long integer that gives the data type of the destination field. See Appendix A for a list of valid data type codes.

dlngth is a long integer that gives the length of the destination field: in characters for string fields. This argument is required for some data types, it must be zero for all other data types. See Appendix A for further information.

dscale is a long integer that gives the scale factor of the destination field. It indicates the number of decimal digits to move the decimal point to the left. A negative scale factor means that the decimal point will be moved to the right. You must specify a scale factor if you want to convert a fixed-point field. Specify zero for all other classes of data.

STATUS CODES:

DIL Name	Meaning
DIX-INVDATTYP	Invalid data type code.
DIX-INVLNG	Length invalid or unspecified.
DIX-INVSCAL	Scale factor invalid or unspecified.
DIX-UNKSYS	Unknown system of origin specified.
DIX-ALIGN	Invalid alignment for data type.
DIX-INVBYSIZ	Invalid byte size specified.
DIX-INVALCHAR	Invalid character in source field or conversion table.
DIX-GRAPHIC	Graphic character changed in conversion.
DIX-FMTLOST	Format effector gained or lost in conversion.
DIX-NONPRINT	Non-printing character gained or lost in conversion.
DIX-TRUNC	String too long for destination -- truncated.
DIX-TOOBIG	Converted source field too large for destination field.
DIX-UNSIGNED	Negative value moved to unsigned field.
DIX-ROUNDED	Result is rounded.
DIX-UNNORM	Floating-point number improperly normalized.
DIX-INVDNUMCHR	Invalid display numeric character in source field.
DIX-INVDNUMSGN	Invalid display numeric sign in source field.
DIX-INVPDDGT	Invalid packed decimal digit in source field.
DIX-INVPDSGN	Invalid packed decimal sign in source field.

RELATED ROUTINES:

DIX$MAK_DES_DET: This routine builds an FFD for the field that you wish to convert. See Section 9.3.1 for a description of DIX$MAK_DES_DET.

9.4 VMS DATA CONVERSION EXAMPLES

9.4.1 VMS COBOL Data Conversion Example

```
IDENTIFICATION DIVISION.

PROGRAM-ID.

        CDCR32.

        This program  performs a  single string  data conversion.  The
        ASCII-8 string value  "ABCDE" will  be converted  to the  same
        ASCII-7 value.


ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER.

        VAX-11.

OBJECT-COMPUTER.

        VAX-11.

DATA DIVISION.

WORKING-STORAGE SECTION.

* source data field

01  SRCDAT PIC X(5) USAGE DISPLAY VALUE "ABCDE".

* The destination data field must be large enough to hold the ASCII-8
* equivalent of the source data field:
*   ASCII-8 PIC X(5) = 8 bits/character * 5 characters = 40 bits
*   ASCII-7 PIC X(5) = 7 bits/character * 5 characters = 35 bits

01  DESTINATION-DATA.
    02 DSTDAT PIC S9(9) COMP OCCURS 2.

* foreign field descriptors
01  FFDS.
    05  SRCFFD PIC S9(9) COMP OCCURS 3.
    05  DSTFFD PIC S9(9) COMP OCCURS 3.

01  INTERFACE-FILES.
    COPY DIL$COBOL OF "SYS$LIBRARY:DIL.TLB".
    COPY DIX$COBOL OF "SYS$LIBRARY:DIL.TLB".

* Dix call parameters.

01  DIX-SYS PIC S9(9) COMP.
01  DIX-DT PIC S9(9) COMP.
01  DIX-BYSZ PIC S9(9) COMP.
01  DIX-BYO PIC S9(9) COMP.
01  DIX-BTO PIC S9(9) COMP VALUE 0.
01  DIX-LEN PIC S9(9) COMP.
01  DIX-SCAL PIC S9(9) COMP.
01  DIL-DISPLAY PIC X(9).
01  DIL-STATUS PIC S9(9) COMP.
```

```
PROCEDURE DIVISION.

INITIALIZE-STUFF.
* initialize destination buffer to zeros

    MOVE 0 TO DSTDAT(1).
    MOVE 0 TO DSTDAT(2).

MAKE-FFDS.

    MOVE 7 TO DIX-BYSZ.
    MOVE 0 TO DIX-BYO.
    MOVE 5 TO DIX-LEN.
    MOVE 0 TO DIX-SCAL.

    CALL "DIX$MAK_DES_DET" USING DSTFFD(1), DSTDAT(1), DIX$K_SYS_10_20,
                                 DIX-BYSZ, DIX-BYO, DIX-BTO,
                                 DIX$K_DT_ASCII_7, DIX-LEN, DIX-SCAL
                           GIVING DIL-STATUS.

    IF DIL-STATUS IS NOT SUCCESS
        MOVE DIL-STATUS TO DIL-DISPLAY
        DISPLAY "? Failure in DIX$MAK_DES_DET.  Dil-status = " DIL-DISPLAY.

    MOVE 8 TO DIX-BYSZ.
    MOVE 0 TO DIX-BYO.
    MOVE 5 TO DIX-LEN.
    MOVE 0 TO DIX-SCAL.

    CALL "DIX$MAK_DES_DET" USING SRCFFD(1), SRCDAT, DIX$K_SYS_VAX,
                                 DIX-BYSZ, DIX-BYO, DIX-BTO,
                                 DIX$K_DT_ASCII_8, DIX-LEN, DIX-SCAL
                           GIVING DIL-STATUS.

    IF DIL-STATUS IS NOT SUCCESS
        MOVE DIL-STATUS TO DIL-DISPLAY
        DISPLAY "? Failure in DIX$MAK_DES_DET.  Dil-status = " DIL-DISPLAY.

DO-CONVERSION.
* Convert ASCII-8 value "ABCDE" to ASCII-7 value "ABCDE".

    CALL "DIX$BY_DIX_DES" USING SRCFFD(1), DSTFFD(1)
                          GIVING DIL-STATUS.

    IF DIL-STATUS IS NOT SUCCESS
        MOVE DIL-STATUS TO DIL-DISPLAY
        DISPLAY "? Failure in DIX$BY_DIX_DES.  Dil-status = " DIL-DISPLAY.

CHECK-RESULTS.
```

```
* What should have been created is the TOPS-10/TOPS-20 form of the
* source value "ABCDE".
*
* In DEC-10/DEC-20 memory this looks like the following:
*
*
*     symbolic representation:          numeric (binary) representation:
* |AAAAABBBBBCCCCCDDDDDEEEEE | :m     |10000011000010100001110001001000101 | :m
*
* Transposing this into VAX memory we have:
*     symbolic representation:          numeric (binary) representation:
*       |EEEEEEE | :n                   |1000101 | :n
*       |CDDDDDDD| :n+1                 |11000100| :n+1
*       |BBCCCCCC| :n+2                 |10100001| :n+2
*       |AAABBBBB| :n+3                 |00110000| :n+3
*       |    AAAA| :n+4                 |    1000| :n+4
*
* if you consider the numeric values as longwords in VAX memory:
* |00110000101000011100010010001010| :n
* |00000000000000000000000000001000| :n+4
*
*        decimal equivalent:
*        | 815907978 | :n
*        |      8    | :n+4
*
```

```
    IF DSTDAT(1) NOT = 815907978
        DISPLAY "? Error in conversion:  "
        DISPLAY " expected converted value not returned from conversion"
        STOP RUN.

    IF DSTDAT(2) NOT = 8
        DISPLAY "? Error in conversion:  "
        DISPLAY " expected converted value not returned from conversion"
        STOP RUN.

    DISPLAY " CDCR32 successfully completed.".

    STOP RUN.
```

9.4.2 VMS FORTRAN Data Conversion Example

```
C FDCR32

C       This program performs  a single string  data conversion.   The
C       ASCII-8 string value  "ABCDE" will  be converted  to the  same
C       ASCII-7 value.
C Include interface support files
        INCLUDE 'SYS$LIBRARY:DIL.TLB (DIL$ANSI_FORTRAN)'
        INCLUDE 'SYS$LIBRARY:DIL.TLB (DIX$ANSI_FORTRAN)'

C Source and destination data fields.
C NOTE: The destination data field must be large enough to hold the
C ASCII-8 equivalent of the source data field:
C   ASCII-8 (5 chars) = 8 bits/character * 5 characters = 40 bits
C   ASCII-7 (5 chars) = 7 bits/character * 5 characters = 35 bits

        INTEGER SRCDAT (2), DSTDAT (2)
```

```
C Foreign field descriptors (ffds)

        INTEGER SRCFFD (3), DSTFFD (3)

C Status return code
        INTEGER DILSTS

C Data for source and destination data fields
        DATA SRCDAT /'ABCD', 'E   '/
        DATA DSTDAT /0, 0/

C make the foreign field descriptors for use by DIX$BY_DIX_DES

        DILSTS = DIX$MAK_DES_DET (SRCFFD, SRCDAT, SYSVAX, 8, 0, 0,
        1       ASCII8, 5, 0)
        IF (DILSTS.NE.NORMAL) GOTO 100

        DILSTS = DIX$MAK_DES_DET (DSTFFD, DSTDAT, SYS36, 7, 0, 0,
        1       ASCII7, 5, 0)
        IF (DILSTS.NE.NORMAL) GOTO 100

C Do conversions: convert ASCII-8 value "ABCDE" to ASCII-7 value "ABCDE"

        DILSTS = DIX$BY_DIX_DES (SRCFFD, DSTFFD)
        IF (DILSTS.NE.NORMAL) GOTO 102

C Check results:

C What should have been created is the TOPS-10/TOPS-20 form of the
C source value "ABCDE".
C
C In DEC-10/DEC-20 memory this looks like the following:
C
C
C     symbolic representation:          numeric (binary) representation:
C |AAAAABBBBBCCCCCDDDDDEEEEE | :m     |10000011000010100001110001001000101 | :m
C
C Transposing this into VAX memory we have:
C     symbolic representation:          numeric (binary) representation:
C       |EEEEEEE | :n                   |1000101 | :n
C       |CDDDDDDD| :n+1                 |11000100| :n+1
C       |BBCCCCCC| :n+2                 |10100001| :n+2
C       |AAABBBBB| :n+3                 |00110000| :n+3
C       |    AAAA| :n+4                 |    1000| :n+4
C
C if you consider the binary values as longwords in VAX memory:
C |00110000101000011100010010001010| :n
C |00000000000000000000000000001000| :n+4
C
C        decimal equivalent:
C        | 815907978 | :n
C        |      8    | :n+4
C

        IF (DSTDAT (1) .NE. 815907978) GOTO 104
        IF (DSTDAT (2) .NE. 8) GOTO 104
200     FORMAT (' FDCR32 successfully completed.')
        WRITE (6, 200)
        STOP
```

```
C Print error information
100     WRITE (6, 101) DILSTS
101     FORMAT (' ? Failure in XDESCR. Dil-status = ', I10)
        STOP

102     WRITE (6, 103) DILSTS
103     FORMAT ('? Failure in XCVST.  Dil-status = ', I10)
        STOP

104     WRITE (6, 105)
105     FORMAT ('? Error in conversion:')
        WRITE (6, 106)
106     FORMAT (' expected converted value not returned from conversion')
        STOP

        END
```

CHAPTER 10
VMS TASK–TO–TASK

CHAPTER 10

VMS TASK-TO-TASK

10.1 TASK-TO-TASK FROM VMS COBOL

The information included in this section assumes

-You are writing a COBOL program
-You plan to use the program on a VMS system

To store a Network Logical Name, task characteristics or user attributes, to send data or to perform a status check you must represent several data items in your program. Users generally, but not necessarily, allocate space for these data items in WORKING-STORAGE.

10.1.1 Including the Interface Support Files

VMS systems have two different classes of Interface Support files for each supported language. The first class of files include native VMS-type names for each of the various codes for VMS COBOL and VMS FORTRAN. The second class includes files with TOPS-10/TOPS-20 COBOL compatible names for VMS COBOL, and files which include ANSI Standard names for the interface to VMS FORTRAN. The Interface Support files for VMS are provided as a text library called DIL.TLB. This library includes the support files for both VMS COBOL and VMS FORTRAN.

Native VMS COBOL

You can use the COBOL COPY verb to retrieve information from DIL.TLB at compilation time. There are three library elements for native VMS COBOL in DIL.TLB. The elements are DIL$COBOL, DIT$COBOL, and DIX$COBOL.

The library element DIL$COBOL defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$COBOL. Severity codes and system codes are defined in element DIL$COBOL. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration:

```
COPY DIL$COBOL OF "SYS$LIBRARY:DIL.TLB".
```

In the following example, the DIL$COBOL element of the library is retrieved and included in your program:

```
01  interface-files.
    COPY DIL$COBOL OF "SYS$LIBRARY:DIL.TLB".
```

The library element DIT$COBOL defines codes specific to the Task-to-Task and Remote File Access Routines. The codes include the DIT status codes, which are standard VMS condition values. Also included are Task-to-Task wait codes, Task-to-Task link types, Task-to-Task message modes and VMS task fire-up codes. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration as described above for the DIL$COBOL element:

```
COPY DIT$COBOL OF "SYS$LIBRARY:DIL.TLB".
```

For programs which use the Task-to-Task routines with the native VMS names, you must include both the DIL$COBOL and DIT$COBOL library elements.

TOPS-10/TOPS-20 Compatible COBOL

If you want to write a program that can be easily transported to a DECsystem-10 or a DECSYSTEM-20, you may want to include the TOPS-10/TOPS-20 compatible names in your program rather than the native VMS names.

You can use the COBOL COPY verb to retrieve information from DIL.TLB at compilation time. There are three library elements for TOPS-10/TOPS-20 compatible COBOL in DIL.TLB. The elements are DIL$COBOL_20, DIT$COBOL_20, and DIX$COBOL_20.

The library element DIL$COBOL_20 defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$COBOL_20. Severity codes and system codes are defined in element DIL$COBOL_20. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration:

```
COPY DIL$COBOL_20 OF "SYS$LIBRARY:DIL.TLB".
```

n the following example, the DIL$COBOL_20 element of the library is retrieved and included in your program:

```
1  interface-files.
    COPY DIL$COBOL_20 OF "SYS$LIBRARY:DIL.TLB".
```

The library element DIT$COBOL_20 defines codes specific to the Task-to-Task and Remote File Access Routines. This includes the DIT status codes, in the compatible COBOL format which provides only the condition identification portion of the status code. Also included are Task-to-Task wait codes, Task-to-Task link types, Task-to-Task message modes and VMS task fire-up codes. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration as shown above for the DIL$COBOL_20 element:

```
COPY DIT$COBOL_20 OF "SYS$LIBRARY:DIL.TLB".
```

For programs which use the Task-to-Task routines with the compatible names, you must include both the DIL$COBOL_20 and DIT$COBOL_20 library elements.

10.1.2 Storing a Network Logical Name

The NLN occupies one longword of VMS memory. To store an NLN, you must define a data item with the following format:

```
01  your-nln   PIC S9(9) COMPUTATIONAL.
```

When you call the DIT$NFOPA, DIT$NFOPB, DIT$NFOP8 or DIT$NFOPP routine to create the NLN, use your-nln as the value for NLN to be returned. When the routine successfully finishes processing, it returns a value to your-nln.

10.1.3 Storing Task and User Attributes

To include task and user attributes in a call to DIT$NFOPA, DIT$NFOPB or DIT$NFOP8 or task attributes in a call to DIT$NFOPP you must describe these attributes as data items in WORKING-STORAGE. These attributes must always be DISPLAY items with a length of PIC X(16) or PIC X(39), as shown below:

```
01  target-name      PIC X(16) USAGE DISPLAY.
01  object-type      PIC X(16) USAGE DISPLAY.
01  desc-name        PIC X(16) USAGE DISPLAY.
01  task-name        PIC X(16) USAGE DISPLAY.
01  userid           PIC X(39) USAGE DISPLAY.
01  passwd           PIC X(39) USAGE DISPLAY.
01  acct             PIC X(39) USAGE DISPLAY.
```

10.1.4 Checking the Status of a Task-to-Task Routine

Section 9.1.4 explains how to check the status of a DIL Routine from VMS COBOL.

A simple call to the Task-to-Task Routines from VMS COBOL, with an error check, looks like the following example:

```
CALL "DIT$NFOPA" USING nln, trgsys, objtyp, desc, tsknam, userid,
     passwd, acct, usdat, wait GIVING status.
IF status is FAILURE
     THEN DISPLAY "error occurred".
```

10.2 TASK-TO-TASK FROM VMS FORTRAN

The information included in this section assumes

-You are writing a FORTRAN program
-You plan to use the program on a VMS system

This section explains how to store a Network Logical Name, task characteristics or user attributes, to send data or to perform a status check.

10.2.1 Including the Interface Support Files

VMS systems have two different classes of Interface Support files for each supported language. The first class of files include native VMS type names for each of the various codes for VMS COBOL and VMS FORTRAN. The second class includes files with TOPS-10/TOPS-20 COBOL compatible names for VMS COBOL, and files that include ANSI Standard names for the interface to VMS FORTRAN. The Interface Support files for the VMS are provided as a text library called DIL.TLB. The library includes the support files for both VMS COBOL and VMS FORTRAN.

Native VMS FORTRAN

You can use the FORTRAN INCLUDE statement to retrieve information from DIL.TLB at compilation time. There are three library elements for native VMS FORTRAN in DIL.TLB. The elements are DIL$FORTRAN, DIT$FORTRAN, and DIX$FORTRAN.

The library element DIL$FORTRAN defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$FORTRAN. Severity codes and system codes are defined in element DIL$FORTRAN. To define these names in your program, include the following statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIL$FORTRAN)'
```

This retrieves the DIL$FORTRAN element of the library file from the system area SYS$LIBRARY and includes it in your program.

The library element DIT$FORTRAN defines codes specific to the Task-to-Task and Remote File Access Routines. This includes the DIT status codes, which are standard VMS condition values. Also included are Task-to-Task wait codes, Task-to-Task link types, Task-to-Task message modes and VMS task fire-up codes. To define these names in your program, include the following statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIT$FORTRAN)'
```

For programs which use the Task-to-Task routines with the native VMS names, you must include both the DIL$FORTRAN and DIT$FORTRAN library elements.

ANSI Standard Compatible FORTRAN

If you want to write a program that can be easily transported to another system, you may want to include the ANSI Standard FORTRAN names in your program rather than the native VMS names.

You can use the FORTRAN INCLUDE statement to retrieve information from DIL.TLB at compilation time. There are three library elements for ANSI Standard FORTRAN in DIL.TLB. The elements are DIL$ANSI_FORTRAN, DIT$ANSI_FORTRAN, and DIX$ANSI_FORTRAN.

The library element DIL$ANSI_FORTRAN defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$ANSI_FORTRAN. Severity codes and system codes are defined in element DIL$ANSI_FORTRAN. To define these names in your program, include the statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIL$ANSI_FORTRAN)'
```

This retrieves the DIL$ANSI_FORTRAN element of the library file and includes it in your program.

The library element DIT$ANSI_FORTRAN defines codes specific to the Task-to-Task and Remote File Access Routines. The codes include the DIT status codes, which are standard VMS condition values. Also included are Task-to-Task wait codes, Task-to-Task link types, Task-to-Task message modes and VMS task fire-up codes. To define these names in your program, include the following statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIT$ANSI_FORTRAN)
```

For programs which use the Task-to-Task routines with the compatible names, you must include both the DIL$ANSI_FORTRAN and DIT$ANSI_FORTRAN library elements.

10.2.2 Storing a Network Logical Name

The NLN occupies one longword of VMS memory. To store an NLN, you must declare an integer, as shown below:

```
INTEGER nln
```

10.2.3 Storing Task and User Attributes

To include task and user attributes in a call to DIT$NFOPA, DIT$NFOPB or DIT$NFOP8 or task attributes in a call to DIT$NFOPP you must describe these attributes in your program as follows:

```
INTEGER trgsys (4)    or   CHARACTER*16   trgsys
INTEGER objtyp (4)    or   CHARACTER*16   objtyp
INTEGER desc (4)      or   CHARACTER*16   desc
INTEGER tsknam (4)    or   CHARACTER*16   tsknam
INTEGER userid (10)   or   CHARACTER*39   userid
INTEGER passwd (10)   or   CHARACTER*39   passwd
INTEGER acct (10)     or   CHARACTER*39   acct
INTEGER usdat (4)     or   CHARACTER*16   usdat
```

NOTE

By default, VAX-11 FORTRAN passes arguments of type CHARACTER by descriptor. In order to use CHARACTER type variables as arguments to DIL (on VMS), the %REF built-in function must be used because DIL string arguments must be passed by reference.

10.2.4 Checking the Status of a Task-to-Task Routine

Section 9.2.4 explains how to check the status of a DIL Routine from VMS FORTRAN.

A simple call to the Task-to-Task routines from VMS FORTRAN, with an error check, might be:

```
        status = DIT$NFOPA (nin, trgsys, objtyp, desc, tsknam,
        1  userid, passwd, acct, usdat, wait)
        IF (status .NE. SS$_NORMAL) GOTO 100
        .
        .
        .
100     TYPE 101
101     FORMAT (' error occurred')
        .
        .
        .
```

10.3 VMS TASK-TO-TASK REFERENCE

10.3.1 DIT$NFGND - Return the Status of Links

PURPOSE:

The DIT$NFGND routine returns the status of network events such as connect requests, available data or aborted links.

CALL FORMAT:

COBOL: CALL "DIT$NFGND" USING nln, wait GIVING stat.

FORTRAN: status = DIT$NFGND (nln, wait)

where:

nln — is the Network Logical Name of the link that you want information about. An NLN is a long integer. It is set by the DIT$NFOPA, DIT$NFOPB, DIT$NFOP8 or DIT$NFOPP routine.

If you want information about any event occuring on any logical link, use -1 as the value for this argument. DIT$NFGND finds the next network event and returns the NLN of that link. If DIT$NFGND cannot find an event the value of this argument is undefined.

wait — is a long integer that gives the wait code.

Set the wait code to "no" if you want the routine to return only the current status of events on the specified link. The DIL Name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want the routine to wait for a network event to occur involving the specified link. When an event occurs the routine reports it. Waiting uses minimal CPU time. The DIL Name for this argument is:

WAIT-YES

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing. You receive this code only if you don't wait for a new event and no events have occured since the last reported event. If an event has taken place, (connect, abort, disconnect, data event) you receive the code for that event.
DIT-CONNECTEVENT	This code is returned for a connect event. If you are checking the status of a passive task, this code indicates that the task has received a connect request. If you are checking the status of an active task, this code indicates that a connect request issued by the active task has been accepted by the passive task.
DIT-ABREJEVENT	The routine returns this code if the link is aborted or rejected. You should call NFCLS to do an abort and release the resources of this link, so it can be used again.
DIT-DATAEVENT	The routine returns this code when data is available over the specified link. You should call NFRCV to receive the data.
DIT-DISCONNECTEVENT	The routine returns this code when the specified link has been disconnected. You should call NFCLS to do an abort to release the resources of this link, so it can be used again.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.
DIT-INTDATAEVENT	The routine returns this code when an interrupt data message is available. You should call NFRCI to read the interrupt data message.

10.3.2 DIT$NFINF - Get Information About the Other End of a Link.

PURPOSE:

The DIT$NFINF routine returns information about the remote node, the remote process of the remote DECnet object associated with a specific network connection.

You can also use the DIT$NFINF routine to read optional data sent by a remote process. If the cooperating process is on a VMS system, you can use DIT$NFINF to read optional data associated with a disconnection or rejection of a process. If the cooperating process is a TOPS-10 or TOPS-20 system, you can read any type of optional data.

CALL FORMAT:

COBOL: CALL "DIT$NFINF" USING nln, inftyp, length, buffer, GIVING stat.

FORTRAN: status = DIT$NFINF (nln, inftyp, length, buffer)

where:

nln is the Network Logical Name of the link that you want information about. The NLN is set by the DIT$NFOPA, DIT$NFOBPB, DIT$NFOP8, or DIT$NFOPP routine. The Network Logical Name is a long integer.

inftyp is a long integer that specifies the type of information wanted.

Refer to the DECnet User's Guide for your system for more information.

Information Type	DIL Name
Remote node name of the cooperating task.	INF-NODE
Remote object type. This information is only available to the passive task.	INF-OBJECT
Remote object descriptor format (0 if the task only requires an object id, 1 if the task only requires a taskname, or 2 if the task requires a project-programmer number). This information is only available to the passive task.	INF-DESCF
Remote DECnet object descriptor. This information is only available to the passive task.	INF-DESC
Remote process user id. This information is only available to the passive task.	INF-USERID
Remote process password. This information is only available to the passive task.	INF-PASSWD
Remote process account. This information is only available to the passive task.	INF-ACCT
Remote process optional data or disconnect optional data or reject optional data. If the cooperating task is running on a VMS system, only disconnect and reject optional data are available.	INF-OPT
Maximum segment size for the link in bytes. This is not available for VMS systems. The information can be used to determine the optimum size of records to be transmitted over the link.	INF-SEG
Abort code or reject code. You can find the meaning of the abort or reject code in the DECnet manual for your system.	INF-ABTCOD

length is a long integer in which the length of ASCII data returned is specified.

buffer is the area in which to place returned ASCII data. This area must be at least 16 ASCII-8 characters long to accomodate optional data, a taskname, or a DECnet descriptor. It must be at least 39 ASCII-8 characters long to accommodate a userid, a password, or an account.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-INFONOTAVAIL	The information you have requested is not available to this task.
DIT-INFOOUTOFRANGE	The information you have requested is not in the range of valid values.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

10.3.3 DIT$NFOPA - Open an ASCII Link From an Active Task

PURPOSE:

The DIT$NFOPA routine opens a logical link to a program on another system. You use this routine when you intend to transmit or receive ASCII data.

CALL FORMAT:

COBOL: CALL "DIT$NFOPA" USING nln, trgsys, objtyp, desc, tsknam,userid, passwd, acct, usdat, wait GIVING stat.

FORTRAN: status = DIT$NFOPA (nln, trgsys, objtyp, desc, tsknam, 1 userid, passwd, acct, usdat, wait)

where:

nln is the Network Logical Name (NLN) to be returned when this routine successfully finishes processing. You use the NLN to identify this link when you call other Task-to-Task routines. The NLN is a long integer.

trgsys is the node name of the target system. The target system, in this case, is the system which runs the passive task that you want to access with this link.

The node name has a length of sixteen ASCII-8 characters. If your node name is less than sixteen characters, left-justify the field. If you give this argument a value of spaces, it defaults to the local system's node name.

objtyp is the object type of the passive task. The object type specifies the kind of service performed by the passive task. This argument has a length of sixteen ASCII-8 characters. If the object type has less than sixteen characters, left-justify the field.

The object type can be expressed as either a number or name. Most programs use an object type of 0 or TASK. Server programs which perform a generic service (MAIL, for example) have a non-zero object type. You can find a list of valid DECnet object types and their meanings in the appropriate DECnet User's Guide.

desc is the DECnet descriptor. You must use a descriptor when you plan to access a task with object type 0 or object name TASK. The descriptor must contain the DECnet taskname of the passive task on the remote system. For TOPS-20 to TOPS-20 communication, when the object type is not 0, you can provide a descriptor. See Appendix D for further information. The descriptor has a length of sixteen ASCII-8 characters. If your descriptor is less than sixteen characters, left-justify the field.

tsknam is the DECnet taskname. Taskname is a unique sixteen character ASCII-8 string that identifies this process to the network. VMS ignores this argument for active tasks. It generates a unique name. If you don't want to specify a taskname, pass spaces as the value for taskname.

NOTE

The following three arguments are optional user attributes. The passive task may use these attributes to validate a network connection, or to perform any other recognition function agreed to by both tasks. These arguments may be given a value of spaces if you want to connect to a passive task on a TOPS-20 or TOPS-10 system; they are required if you want to connect to a passive task on a VMS system.

userid is your userid. Userid has a length of thirty-nine ASCII-8 characters. If your userid is less than thirty-nine characters, left-justify the field. If you don't want to specify a userid, pass spaces as the value of this argument.

passwd is your password. Password has a length of thirty-nine ASCII-8 characters. If your password is less than thirty-nine characters, left-justify justify the field. If you don't want to specify a password, pass spaces as the value of the argument.

acct is your account number or charge code. This field has a length of thirty-nine ASCII-8 characters. If the account has less than thirty-nine characters, left-justify the field. If you don't want to specify an account number, pass spaces as the value for this argument.

usdat is reserved for sixteen characters of ASCII-8 user data. Not currently usedon VMS systems; this argument must be spaces.

wait is a long integer that gives the wait code.

Set the wait code to "no" if you do not want your program to wait until it establishes a connection to the passive task. Using this code enables your program to perform other duties while waiting for the network connection. To find out if the passive task has accepted your connection, periodically call the DIT$NGFND routine to check status. The DIL Name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want your program to wait until the passive task has accepted your link. The routine does not return to your program until it establishes the specified link. While it waits, you can not use the active task. Waiting uses minimal CPU time. The DIL Name for this argument is:

WAIT-YES

STATUS CODES:

DIL Name	Meaning
DIT-TOOMANY	You attempted too many links. The DIL allows a maximum of 20 open links.
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-ABORTREJECT	The link was aborted or rejected.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

10.3.4 DIT$NFOPB - Open a Binary Link From an Active Task

PURPOSE:

The DIT$NFOPB routine opens a logical link to a program on another system. You use this routine when you intend to transmit records or blocks of data.

Data moved through the network on a link opened with DIT$NFOPB is transmitted as a string of bits, sent eight bits at a time. This format allows the data to be used by the Data Conversion Routines. To learn more about bit transport, see Appendix F.

CALL FORMAT:

COBOL: CALL "DIT$NFOPB" USING nln, trgsys, objtyp, desc, tsknam, userid, passwd, acct, usdat, wait GIVING stat.

FORTRAN: status = DIT$NFOPB (nln, trgsys, objtyp, desc, tsknam, 1 userid, passwd, acct, usdat, wait)

where:

nln is the Network Logical Name (NLN) to be returned when this routine successfully finishes processing. You use the NLN to identify this link when you call other Task-to-Task routines. The NLN is a long integer.

trgsys is the node name of the target system. The target system, in this case, is the system which runs the passive task that you want to access with this link.

The node name has a length of sixteen ASCII-8 characters. If your node name is less than sixteen characters, left-justify the field. If you give this argument a value of spaces, it defaults to the local system's node name.

objtyp is the object type of the passive task. The object type specifies the kind of service performed by the passive task. This argument has a length of sixteen ASCII-8 characters. If the object type has less than sixteen characters, left-justify the field.

The object type can be expressed as either number or name. Most programs use an object type of 0 or TASK. Server programs which perform a generic service (MAIL, for example) have a non-zero object type. You can find a list of valid DECnet object types and their meanings in the appropriate DECnet Guide.

desc is the DECnet descriptor. You must use a descriptor when you plan to access a task with object type 0 or object name TASK. The descriptor must contain the DECnet taskname of the passive task on the remote system. See Appendix D for further information. The descriptor has a length of sixteen ASCII-8 characters. If your descriptor is less than sixteen characters, left-justify the field.

tsknam is the DECnet taskname. Taskname is a unique string of sixteen ASCII-8 characters. The taskname identifies this process to the network. VMS ignores this argument for active tasks. It generates a unique name. If you don't want to specify a taskname, pass spaces as the value for this argument.

NOTE

The following three arguments are optional user attributes. The passive task may use these attributes to validate a network connection, or to perform any other recognition function agreed to by both tasks. These arguments may be given a value name of spaces if you want to connect to a passive task on a TOPS-20 or TOPS-10 system; they are required if you want to connect to a passive task on a VMS system.

userid is your userid. Userid has a length of thirty-nine ASCII-8 characters. If your userid is less than thirty-nine characters, left-justify the field. If you don't want to specify a userid, pass spaces as the value for this argument.

passwd is your password. Password has a length of thirty-nine ASCII-8 characters. If your password is less than thirty-nine characters, left-justify the field. If you don't want to specify a password, pass spaces as the value for this argument.

acct is your account number or charge code. This field has a length of thirty-nine ASCII-8 characters. If this information is less than thirty-nine characters, left-justify the field. If you don't want to specify an account number, pass spaces as the value for this argument.

usdat is reserved for sixteen ASCII-8 characters of user data. Not currently used on VMS systems; this argument must be spaces.

wait is a long integer that gives the wait code.

Set the wait code to "no" if you do not want your program to wait until it establishes a connection to the passive task. Using this code enables your program to perform other duties while waiting for the network connection. To find out if the passive task has accepted your connection, periodically call the DIT$NGFND routine to check status. The DIL Name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want your program to wait until the passive task has accepted your link. The routine does not return to your program until it establishes the specified link. While it waits, you can not use the active task. Waiting uses minimal CPU time. The DIL Name for this argument is:

WAIT-YES

STATUS CODES:

DIL Name	Meaning
DIT-TOOMANY	You attempted too many links. The DIL allows a maximum of 20 open links.
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-ABORTREJECT	The link was aborted or rejected.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

10.3.5 DIT$NFOP8 - Open an 8-Bit Link From an Active Task

PURPOSE:

The DIT$NFOP8 routine opens a logical link to a program on another system. You use this routine to transmit 8-bit bytes of data. This routine is especially useful for transmitting data between VMS systems and for performing special operations.

Data is moved through a link opened with DIT$NFOP8 as a string of 8-bit bytes. The routine does not transmit any unused bits. The target system stores the data in whatever way it normally stores 8-bit bytes of data. To learn more about bit transport, see Appendix F.

CALL FORMAT:

COBOL:
```
CALL "DIT$NFOP8" USING nln, trgsys, objtyp, desc,
     tsknam, userid, passwd, acct, usdat, wait GIVING
     stat.
```

FORTRAN:
```
status = DIT$NFOP8 (nln, trgsys, objtyp, desc, tsknam,
1 userid, passwd, acct, usdat, wait)
```

where:

nln — is the Network Logical Name (NLN) to be returned when this routine successfully finishes processing. You use the NLN to identify this link when you call other Task-to-Task routines. The NLN is a long integer.

trgsys — is the node name of the target system. The target system, in this case, is the system which runs the passive task that you want to access with this link.

The node name has a length of sixteen ASCII-8 characters. If your node name is less than sixteen characters, left-justify the field. If you give this argument a value of spaces, it defaults to the local system's node name.

objtyp — is the object type of the passive task. The object type specifies the kind of service performed by the passive task. This argument has a length of sixteen ASCII-8 characters. If the object type has less than sixteen characters, left-justify the field.

The object type can be expressed as either a number or name. Most programs use an object type of 0 or TASK. Server programs which perform a generic service (MAIL, for example) have a non-zero object type. You can find a list of valid DECnet object types and their meanings in the appropriate DECnet User's Guide.

desc — is the DECnet descriptor. You must use a descriptor when you plan to access a task with object type 0 or object name TASK. The descriptor must contain the DECnet taskname of the passive task on the remote system. See Appendix D for further information. The descriptor has a length of sixteen ASCII-8 characters. If your descriptor is less than sixteen characters, left-justify the field.

tsknam is the DECnet taskname. Taskname is a unique sixteen character ASCII-8 string that identifies this process to the network. VMS ignores this argument for active tasks. It generates a unique name. If you don't want to specify a taskname, pass spaces as the value for this argument.

NOTE

The following three arguments are optional user attributes. The passive task may use these attributes to validate a network connection, or to perform any other recognition function agreed to by both tasks. These arguments may be given a value name of spaces if you want to connect to a passive task on a TOPS-20 or TOPS-10 system; they are required if you want to connect to a passive task on a VMS system.

userid is your userid. Userid has a length of thirty-nine ASCII-8 characters. If your userid is less than thirty-nine characters, left-justify the field. If you don't want to specify a userid, pass spaces as the value for this argument.

passwd is your password. Password has a length of thirty-nine ASCII-8 characters. If your password is less than thirty-nine characters, left-justify the field. If you don't want to specify a password, pass spaces as the value for this argument.

acct is your account number or charge code. This field has a length of thirty-nine ASCII-8 characters. If this information is less than thirty-nine characters, left-justify the field. If you don't want to specify an account number, pass spaces as the value for this argument.

usdat is reserved for sixteen characters of ASCII-8 user data. Not currently used on VMS systems; this argument must be spaces.

wait is a long integer that gives the wait code.

Set the wait code to "no" if you do not want your program to wait until it establishes a connection to the passive task. Using this code enables your program to perform other duties while waiting for the network connection. To find out if the passive task has accepted your connection, periodically call the DIT$NGFND routine to check status. The DIL Name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want your program to wait until the passive task has accepted your link. The routine does not return to your program until it establishes the specified link. While it waits, you can not use the active task. Waiting uses minimal CPU time. The DIL Name for this argument is:

WAIT-YES

STATUS CODES:

DIL Name	Meaning
DIT-TOOMANY	You attempted too many links. The DIL allows a maximum of 20 open links.
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-ABORTREJECT	The link was aborted or rejected.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

10.3.6 DIT$NFOPP - Open a Link From a Passive Task

PURPOSE:

The DIT$NFOPP routine opens a logical link from a passive task. It indicates that the server program is ready to accept connections from active programs.

There are different ways to move data through the network. The cooperating tasks must both specify the same method of data transfer. The active task which connects to DIT$NFOPP establishes the way data actually moves through the network. (DIT$NFOPA moves ASCII-8 data, DIT$NFOPB moves binary data and DIT$NFOP8 moves 8-bit bytes of data.) When it receives a connect request, The passive task calls the DIT$NFACC routine to accept the request. The "lnktyp" argument in the call to DIT$NFACC specifies the type of data to be transferred over the connection. If the active task and the DIT$NFACC routine specify different data types, results are undefined. To learn more about bit transport, see Appendix F.

CALL FORMAT:

COBOL: CALL "DIT$NFOPP" USING nln, objtyp, desc, tsknam, wait GIVING stat.

FORTRAN: status = DIT$NFOPP (nln, objtyp, desc, tsknam, wait)

where:

nln is the Network Logical Name (NLN) to be returned when this routine successfully finishes processing. You use the NLN to identify this link when you call other Task-to-Task routines. The NLN is a long integer.

Possible DIL Names for this argument are:

PAS-FIREUP if the task can accept only one connection on this link.

PAS-NFIREUP if the task can accept more than one connection on this link. You must have SYSNAM privileges to specify this parameter.

See Appendix D for further information.

objtyp is the DECnet object type of the passive task. The object type tells the kind of service performed by this task. This argument has a length of sixteen ASCII-8 characters. If the object type has less than sixteen characters, left-justify the field.

The object type can be expressed as either a a number or name. Most programs use an object type of 0 or TASK. Server programs which perform a generic service (MAIL, for example) have a non-zero object type. You can find a list of valid DECnet object types and their meanings in the appropriate DECnet User's Guide.

desc is the DECnet descriptor. See Appendix D for examples of task identification. The descriptor has a length of sixteen ASCII-8 characters. If your descriptor is less than sixteen characters, left-justify the field.

tsknam is the DECnet taskname. Taskname is a unique sixteen character ASCII-8 string that identifies this process to the network. An active task may identify the passive task it wants to connect to by specifying the passive task's taskname. For this reason, most passive tasks are given specific task names. If you pass spaces for this argument, the operating system assigns a unique task name. See Appendix D for further information.

wait is a long integer that gives the wait code.

Set the wait code to "no" if you you want to set up a server task, but you do not want to wait until an active task requests its services. Using this code leaves your program free to perform other duties. To find out if the an active task wants to connect to the server, call the DIT$NFGND routine to check status. The DIL Name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want to set up a server task that waits until an active task requests its services. The routine returns a status value when an active task tries to connect to the server. While it waits, you can not use the server program to perform any other processing functions. Waiting uses minimal CPU time. The DIL Name for this argument is:

WAIT-YES

STATUS CODES:

DIL Name	Meaning
DIT-TOOMANY	You attempted too many links. The DIL allows a maximum of 20 open links.
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-ABORTREJECT	The link was aborted or rejected.
DIT-HORRIBLE	This code is returned in the event of system or unexpected error.

10.3.7 DIT$NFACC - Accept a Connection

PURPOSE:

The DIT$NFACC routine accepts a connection from an active task and specifies the type of data to be transferred over the link. It is used in conjunction with the DIT$NFOPP routine, which opens the logical link.

CALL FORMAT:

COBOL: CALL "DIT$NFACC" USING nln, lnktyp, char, opdat GIVING stat.

FORTRAN: status = DIT$NFACC (nln, lnktyp, char, opdat)

where:

nln is the Network Logical Name set by the DIT$NFOPP routine when it successfully finishes processing. The NLN is a long integer.

lnktyp is a long integer that indicates the type of data that you want to transfer over the link. You must specify one of the following values:

LINK Data Type	DIL Name
ASCII-8 data. To transfer ASCII data, use the DIT$NFOPA routine to open the link.	LTYPE-ASCII
Binary data. To transfer binary data, use the DIT$NFOPB routine to open the link.	LTYPE-BINARY
8-bit bytes. To transfer 8-bit bytes of data, use the DIT$NFOP8 routine to open the link.	LTYPE-8BIT

char is the number of characters of optional data that you plan to send (see below).

opdat sixteen ASCII-8 characters of optional user data. See the DIT$NFINF routine for more information.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

10.3.8 DIT$NFREJ - Reject a Connection

PURPOSE:

The DIT$NFREJ routine is used by the passive task to reject a connection request from an active task.

CALL FORMAT:

COBOL: CALL "DIT$NFREJ" USING nln, rejcod, char, opdat GIVING stat.

FORTRAN: status = DIT$NFREJ (nln, rejcod, char, opdat)

where:

nln is the Network Logical Name set by the DIT$NFOPP routine when it successfully finishes processing. NLN is a long integer.

rejcod is a long integer that specifies the type of reject requested. You should not use nine for this code, which means "User Program Abort." See your DECnet User's Guide for a list of abort/reject codes. This parameter is not used either VMS OR TOPS-10 systems.

char is a long integer that specifies the number of characters (0-16) of optional data that you plan to send (see below).

opdat is 16 ASCII-8 characters of optional user data. See the DIT$NFINF routine for more information.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

10.3.9 DIT$NFRCV - Receive Data

PURPOSE:

The DIT$NFRCV routine receives data (DECnet messages) sent over a logical link.

The routine reads one DECnet message each time it is sucessfully called.

CALL FORMAT:

COBOL: CALL "DIT$NFRCV" USING nln, msunit, mxunit, bufloc, msmode, wait GIVING stat.

FORTRAN: status = DIT$NFRCV (nln, msunit, mxunit, bufloc, 1 msmode, wait)

where:

nln is the Network Logical Name set by the DIT$NFOPA, DIT$NFOPB, DIT$NFOP8 or DIT$NFOPP routine when the routine successfully finished processing. The NLN is a long integer.

msunit is a long integer that gives the message unit size. It tells the byte size, in bits, of messages written in binary format (with links opened through DIT$NFOPB). A value of zero for this argument indicates that you plan to transfer data as words. If you open the active side of the link with DIT$NFOPA or DIT$NFOP8, the routine ignores this argument (both routines send 8-bit bytes of data).

mxunit is a long integer that specifies the maximum number of units to be read by the routine.

- For links opened with DIT$NFOPA, this is the maximum number of ASCII-8 characters in the message.
- For links opened with DIT$NFOP8, this is the maximum number of sequential 8-bit bytes in the message.
- For links opened with DIT$NFOPB, this is the maximum number of bytes (of the unit size shown in msunit) or words in the message. DIT$NFRCV pads the last byte or word sent with zero bits if it does not divide evenly into bytes of the specified size.

bufloc is the location of the user buffer where the message will be placed after it is read. This buffer must be at least as large as the number of bytes, characters or words specified in the mxunit argument, above.

msmode is the message-mode flag. The message-mode is a long integer. The DIL Name for this argument is:

MSG-MSG

wait is a long integer that gives the wait code.

Set the wait code to "no" if you want the routine to return whatever data is currently available. If data is not available, the routine returns and you can use the program to perform other duties. The DIL Name for this argument is:

WAIT-NO

Set the wait code to "yes" if you want the routine to wait until data is received or the read fails. The DIL Name for this argument is:

WAIT-YES

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-ABORTREJECT	This code is returned if the link is disconnected or aborted. The character count area contains the number of units, bytes or words that the routine read before the disconnect.
DIT-OVERRUN	This code is returned if you try to send too much data to the routine.
DIT-NOTENOUGH	This code is returned if the amount of data you requested is not available.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.
DIT-INTERRUPT	Interrupt data must be read first using NFRCI.

10.3.10 DIT$NFSND - Send Data

PURPOSE:

The DIT$NFSND routine sends data over a logical link.

If you open the logical link with DIT$NFOPA, the routine sends the data in ASCII format. The receiving task can directly read ASCII data from another system, even if the tasks are on heterogeneous systems.

If you open the logical link with DIT$NFOP8, the routine sends the data as sequential 8-bit bytes. The local system treats the data as it normally treats 8-bit bytes of data. The remote system treats the data it receives in the way it normally stores 8-bit bytes of data.

If you open the logical link with DIT$NFOPB, the routine sends the data in binary format. If the sending and receiving tasks are on homogeneous systems, the receiving system can directly read the data. If the sending and receiving tasks are on heterogeneous systems, you must convert the data using the Data Conversion Routines. The sending task can perform the conversion before it sends the data or the receiving task can convert the data it receives.

This routine sends one DECnet message each time it is sucessfully called.

CALL FORMAT:

```
COBOL:    CALL "DIT$NFSND" USING nln, msunit, length, buffer,
               msmode GIVING stat.

FORTRAN:  status = DIT$NFSND (nln, msunit, length, buffer,
          1     msmode)
```

where:

nln — is the Network Logical Name set by the DIT$NFOPA, DIT$NFOPB, DIT$NFOP8 or DIT$NFOPP routine when the routine successfully finished processing. The NLN is a long integer.

msunit — is a long integer that gives the message unit size. It tells the byte size, in bits, of messages written in binary format (with links opened through DIT$NFOPB). A value of zero for this argument indicates that the data is to be sent as words. If you open the active side of the link with DIT$NFOPA or DIT$NFOP8, the routine ignores this argument.

length — is a long integer that specifies the length of the data that you want to send. This argument must have a value that is greater than zero.

- For links opened with DIT$NFOPA, length is given in ASCII-8 characters.

- For links opened with DIT$NFOP8, length is given in sequential 8-bit bytes.
- For links opened with DIT$NFOPB, length is given in bytes (of the unit size shown in the msunit argument, above) or in words.

buffer is the buffer containing the data you want to send.

msmode is the message-mode flag. The message-mode is a one-word integer. The DIL Name for this argument is:

MSG-MSG

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

10.3.11 DIT$NFRCI - Receive an Interrupt Data Message Over a Link

PURPOSE:

The DIT$NFRCI routine receives a single interrupt data message over a logical link.

Receipt of an interrupt data message is an asynchronous event. You should check for asynchronous events by using DIT$NFGND which will announce interrupt data messages before it will announce "regular" data messages (sent by DIT$NFSND). Interrupt data messages must be read before DIT$NFGND will announce any lower-level events (regular data messages or disconnections). DIT$NFRCV will return the DIT-INTERRUPT error and refuse to return data if an interrupt message is available which has not yet been read by DIT$NFRCI.

CALL FORMAT:

COBOL: CALL "DIT$NFRCI" USING nln, char, buffer GIVING stat.

FORTRAN: status = DIT$NFRCI (nln, char, buffer)

where:

nln is the Network Logical Name set by the DIT$NFOPP routine when it successfully finishes processing. NLN is a long integer.

char is a long integer into which the number of ASCII-8 characters (1-16) of interrupt data read is returned.

buffer is the location of the user buffer where the message will be placed after it is read. The length of this buffer must be 16 ASCII-8 characters.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NODATAAVAILABLE	No interrupt data is available to be read at this time.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

10.3.12 DIT$NFINT - Send an Interrupt Data Message Over a Link

PURPOSE:

The DIT$NFINT routine sends a single interrupt data message over a logical link.

Unlike DIT$NFSND, DIT$NFINT data is always sent in message mode, so a prompt attempt to send the data is guaranteed. Data sent in this mode is not sent in synchronization with data sent by DIT$NFSND. Only one interrupt data message can be outstanding from each end of a logical link at one time. If an interrupt data message is sent over a logical link by one process, a second interrupt data message cannot be sent by that process until the first one has been received at the other end of the logical link. If a second interrupt data message is sent before the first one has been received at the other side of the link, then the first interrupt data message may be lost at the receiving end of the link.

CALL FORMAT:

COBOL: CALL "DIT$NFINT" USING nln, char, buffer GIVING stat.

FORTRAN: status = DIT$NFINT (nln, char, buffer)

where:

nln is the Network Logical Name set by the DIT$NFOPP routine when it successfully finishes processing. The NLN is a long integer.

char is a long integer that specifies the number of ASCII-8 characters (1-16) of interrupt data to send.

buffer is the buffer that contains the data that you want to send.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.
DIT-INTERRUPT	Available interrupt data must be read first. See the DIT$NFRCI routine.

10.3.13 DIT$NFCLS - Close a Link

PURPOSE:

The DIT$NFCLS routine disconnects or aborts a logical link, releasing its resources to be used by another link.

A "synchronous disconnect" is the normal way to close a link; it disconnects the link after it performs all outstanding data transmission. An abort instantaneously disconnects the link.

You can call the DIT$NFCLS routine to disconnect the link before or after receiving a disconnect from the other end of the link. To preserve data integrity, the recipient of the last piece of data should be the first to disconnect the link (using a synchronous disconnect). The program that sent the data recognizes that its data was read when it receives the other program's disconnect. It should then abort its end of the link, to free the resources for another use.

CALL FORMAT:

COBOL: CALL "DIT$NFCLS" USING nln, disc, char, opdat
GIVING stat.

FORTRAN: status = DIT$NFCLS (nln, disc, char, opdat)

where:

nln is the Network Logical Name, set by the DIT$NFOPA, DIT$NFOPB, DIT$NFOP8 or the DIT$NFOPP routine when the routine successfully finished processing. The NLN is a long integer.

disc is a long integer that indicates the type of disconnect requested. Use zero for this argument if you want a synchronous disconnect. A non-zero value for this argument indicates an abort. "9" is the normal value for an abort. The DECnet User's Guide for your system gives a list of possible abort codes.

char is a long integer that specifies the number of characters (0 to 16) of optional data to be sent (see the opdat argument, below).

opdat is sixteen ASCII-8 characters of optional user data. See the DIT$NFINF routine for more information.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

10.4 VMS TASK-TO-TASK EXAMPLES

10.4.1 VMS COBOL Task-to-Task Examples

```
IDENTIFICATION DIVISION.

PROGRAM-ID.

        PASC32.

AUTHOR.

        SOFTWARE ENGINEERING.

        This program opens a passive link and then waits for a
        connection from an active task (created by the program
        ACTC32). Once a link is established, user specified messages
        are sent in both directions across the link. The link is
        closed by the program ACTC32 and this program waits for a
        confirmation of the close.

INSTALLATION

        DEC MARLBORO.

ENVIRONMENT DIVISION.

.

.

.

DATA DIVISION.

WORKING-STORAGE SECTION.

01  INTERFACE-FILES.
    COPY DIT$COBOL OF "SYS$LIBRARY:DIL.TLB".
    COPY DIL$COBOL OF "SYS$LIBRARY:DIL.TLB".

* DIL status return or condition value
01  DIL-STATUS PIC S9(9) COMP.

01  DATA-RECORDS.
    05  SEND-DATA PIC X(100).
    05  RECEIVE-DATA PIC X(100).

01  COUNT-OPT-DATA PIC S9(9) COMP VALUE 0.
01  OPT-DATA PIC X(16) VALUE SPACES.
01  NETLN PIC S9(9) COMP.
01  OBJECT-ID PIC X(16).
01  DESCRIPT PIC X(16) VALUE SPACES.
01  TASKNAME PIC X(16).
01  MESSAGE-UNITS-SIZE PIC S9(9) COMP VALUE 8.
01  MESSAGE-SIZE PIC S9(9) COMP VALUE 100.

01  DIL-DISPLAY PIC X(12).

01  DIL-MATCH-COND PIC S9(9) COMP.
    88  NO-MATCH VALUE 0.
    88  MATCH-1 VALUE 1.
```

```
PROCEDURE DIVISION.

OPEN-PASSIVE.
* Open a passive link.

    MOVE "SERVER" TO TASKNAME.
    MOVE "0" TO OBJECT-ID.
    MOVE DIT$K_PAS_NFIREUP TO NETLN.

    CALL "DIT$NFOPP" USING NETLN, OBJECT-ID, DESCRIPT,
                           TASKNAME, DIT$K_WAIT_NO
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFOPP Status return: " DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? NFOPP: unsuccessful status return "
        STOP RUN.

CHECK-FOR-CONNECT
* Wait for a connect request

    CALL "DIT$NFGND" USING NETLN, DIT$K_WAIT_YES
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFGND Status return: " DIL-DISPLAY.

    CALL "LIB$MATCH_COND" USING DIL-STATUS,
                                DIT$_CONNECTEVENT
                          GIVING DIL-MATCH-COND.

    IF NO-MATCH
        DISPLAY "? NFGND: Unexpected or invalid status returned: "
        STOP RUN.

ACCEPT-LINK
* Accept link

    CALL "DIT$NFACC" USING NETLN, DIT$K_LTYPE_ASCII, COUNT-OPT-DATA, OPT-DATA
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFACC Status return: " DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? NFACC: unsuccessful status return "
        STOP RUN.

CHECK-FOR-DATA.
* Wait for a data event on the link

    CALL "DIT$NFGND" USING NETLN, DIT$K_WAIT_YES
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFGND Status return: " DIL-DISPLAY.

    CALL "LIB$MATCH_COND" USING DIL-STATUS,
                                DIT$_DATAEVENT
                          GIVING DIL-MATCH-COND.
    IF NO-MATCH
        DISPLAY "? NFGND: Unexpected or invalid status returned: "
        STOP RUN.
```

```
READ-THE DATA.
* Read the data received over the link

    MOVE 100 TO MESSAGE-SIZE.

    CALL "DIT$NFRCV" USING NETLN, MESSAGE-UNITS-SIZE, MESSAGE-SIZE,
                           RECEIVE-DATA, DIT$K_MSG_MSG, DIT$K_WAIT_YES
                     GIVING DIL-STATUS.

    DISPLAY " NFRCV Status return: " DIL-STATUS.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? NFRCV: unsuccessful status return "
        STOP RUN.

    DISPLAY " Data received: ".
    DISPLAY RECEIVE-DATA.

SEND-SOME-DATA.
* Send some data over the link

    MOVE 100 TO MESSAGE-SIZE.

    DISPLAY " Enter some data to be sent over the link: ".
    ACCEPT SEND-DATA.

    CALL "DIT$NFSND" USING NETLN, MESSAGE-UNITS-SIZE, MESSAGE-SIZE,
                           SEND-DATA, DIT$K_MSG_MSG
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFSND Status return: " DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? NFSND: unsuccessful status return "
        STOP RUN.

CHECK-FOR-CLOSE.

    CALL "DIT$NFGND" USING NETLN, DIT$K_WAIT_YES
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFGND status return: " DIL-DISPLAY.

    CALL "LIB$MATCH_COND" USING DIL-STATUS,
                                DIT$_ABREJEVENT,
                                DIT$_DISCONNECTEVENT
                          GIVING DIL-MATCH-COND.

    IF NO-MATCH
       DISPLAY "? NFGND: Invalid status returned on link close"
       STOP RUN.

    DISPLAY " PASC32 successful".

    STOP RUN.
```

```
INDENTIFICATION DIVISION.

PROGRAM-ID.

        ACTC32.

AUTHOR.

        SOFTWARE ENGINEERING.

        This program opens an active link by connecting to the passive
        task set up by the program PASC32. Once the link is
        established, user specified messages are sent in both
        directions across the link. Then the link is closed.

INSTALLATION.

        DEC MARLBORO.

ENVIRONMENT DIVISION.

.

.

.

DATA DIVISION.

WORKING-STORAGE SECTION.

01  INTERFACE-FILES.
    COPY DIT$COBOL OF "SYS$LIBRARY:DIL.TLB".
    COPY DIL$COBOL OF " SYS$LIBRARY:DIL.TLB".

* DIL status return
01  DIL-STATUS PIC S9(9) COMP.

01  DATA-RECORDS.
    05  SEND-DATA PIC X(100).
    05  RECEIVE-DATA PIC X(100).

01  COUNT-OPT-DATA PIC S9(9) COMP VALUE 0.
01  OPT-DATA PIC X(16) VALUE SPACES.
01  NETLN PIC S9(9) COMP.
01  HOSTNAME PIC X(06).
01  OBJECT-ID PIC X(16).
01  DESCRIPT PIC X(16) VALUE SPACES.
01  TASKNAME PIC X(16).
01  USERID PIC X(39) VALUE SPACES.
01  PASSWD PIC X(39) VALUE SPACES.
01  ACCT PIC X(39) VALUE SPACES.
01  MESSAGE-UNITS-SIZE PIC S9(9) COMP VALUE 8.
01  MESSAGE-SIZE PIC S9(9) COMP VALUE 100.

01  SYNCH-DISCONN PIC S9(9) COMP VALUE 0.
01  DIL-DISPLAY PIC X(12).

01  DIL-MATCH-COND PIC S9(9) COMP.
    88  NO-MATCH VALUE 0.
    88  MATCH-1 VALUE 1.
    88  MATCH-2 VALUE 2.
```

```
PROCEDURE DIVISION.

CONNECT-TO-PASSIVE.
* Ask for a connection to the passive link

   MOVE "0" TO OBJECT-ID.
   MOVE "SERVER" TO DESCRIPT.
   MOVE SPACES TO TASKNAME.

   CALL "DIT$NFOPA" USING NETLN, HOSTNAME, OBJECT-ID, DESCRIPT, TASKNAME,
                    USERID, PASSWD, ACCT, OPT-DATA, DIT$K_WAIT_NO
                   GIVING DIL-STATUS.

   MOVE DIL-STATUS TO DIL-DISPLAY.
   DISPLAY " NFOPA Status return: ", DIL-DISPLAY.
   IF DIL-STATUS IS NOT SUCCESS
       DISPLAY "? NFOPA: unsuccessful status return "
       STOP RUN.

CHECK-FOR-CONNECT.
* wait for confirmation of the connection

    CALL "DIT$NFGND" USING NETLN, DIT$K_WAIT_YES
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFGND Status return: " DIL-DISPLAY.

    CALL "LIB$MATCH_COND" USING DIL-STATUS,
                                DIT$ CONNECTEVENT
                           GIVING DIL-MATCH-COND.

    IF NO-MATCH
        DISPLAY "? NFGND: Unexpected or invalid status returned: "
        STOP RUN.

SEND-SOME-DATA.
* Send some data over the link

    MOVE 100 TO MESSAGE-SIZE.

    DISPLAY " Enter some data to be sent over the link: ".
    ACCEPT SEND-DATA.

    CALL "DIT$NFSND" USING NETLN, MESSAGE-UNITS-SIZE, MESSAGE-SIZE,
                           SEND-DATA, DIT$K_MSG_MSG
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFSND Status return: " DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? NFSND: unsuccessful status return "
        STOP RUN.
```

```
    CHECK-FOR-DATA.
    * Wait for a data event on the link

        CALL "DIT$NFGND" USING NETLN, DIT$K_WAIT_YES
                         GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFGND Status return: " DIL-DISPLAY.

    CALL "LIB$MATCH_COND" USING DIL-STATUS,
                                DIT$_DATAEVENT
                          GIVING DIL-MATCH-COND.

    IF NO-MATCH
        DISPLAY "? NFGND: Unexpected or invalid status returned: "
        STOP RUN.

READ-THE-DATA.
* Read the data received over the link

    MOVE 100 TO MESSAGE-SIZE.

    CALL "DIT$NFRCV" USING NETLN, MESSAGE-UNITS-SIZE, MESSAGE-SIZE,
                           RECEIVE-DATA, DIT$K_MSG_MSG, DIT$K_WAIT_YES
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFRCV Status return: " DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? NFRCV: unsuccessful status return "
        STOP RUN.

    DISPLAY " Data received: ".
    DISPLAY RECEIVE-DATA.

CLOSE-LINK.
* Close the link

    CALL "DIT$NFCLS" USING NETLN, SYNCH-DISCONN, COUNT-OPT-DATA, OPT-DATA
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " NFCLS Status return: " DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? NFCLS: unsuccessful status return "
        STOP RUN.

    DISPLAY " ACTC32 successful".
    STOP RUN.
```

10.4.2 VMS FORTRAN Task-to-Task Examples

```
C PASF32

C       This program opens a passive link and then waits for a
C       connection from an active task (created by the program
C       ACTC36).  Once a link is established, user specified messages
C       are sent in both directions across the link.  The link is
C       closed by the program ACTC36 and this program waits for a
C       confirmation of the close.

C Use the DIL interface files.

        INCLUDE 'SYS$LIBRARY:DIL.TLB (DIT$FORTRAN)'
        INCLUDE 'SYS$LIBRARY:DIL.TLB (DIL$FORTRAN)'

C Data records

        DIMENSION SENDD (25), RECD (25)

C DIL task to task routine parameters

        DIMENSION OPTDAT (4), OBJID (4), DESCR (4), TASKN (4)

        INTEGER NETLN, DILSTS, MSGSIZ, MUNTSZ, CNTOPD

C       Link description fields -- passive end

        DATA OBJID /'0    ', '    ', '    ', '    '/
        DATA DESCR /'     ', '    ', '    ', '    '/
        DATA TASKN /'SERV', 'ER  ', '    ', '    '/

        DATA OPTDAT /'    ', '    ', '    ', '    '/

C Program messages

777     FORMAT (' Invalid status returned... ')
778     FORMAT (' Enter some data to be sent over the link: ')
779     FORMAT (' Data received: ')
200     FORMAT (' NFOPP Status return: ', I12)
202     FORMAT (' NFGND Status return: ', I12)
203     FORMAT (' NFACC Status return: ', I12)
204     FORMAT (' NFSND Status return: ', I12)
205     FORMAT (' NFRCV Status return: ', I12)


C       initialize sending and receiving message data fields

        DO 100 I = 1, 25
        SENDD (I) = 0
100     RECD (I) = 0

C       Open a passive link

        NETLN = DIT$K_PAS_NFIREUP
        DILSTS = DIT$NFOPP (NETLN, OBJID, DESCR, TASKN, DIT$K_WAIT_NO)

        WRITE (6, 200) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 110
        WRITE (5, 777)
        STOP
```

```
C       Wait for a connect request

110     DILSTS = DIT$NFGND (NETLN, DIT$K_WAIT_YES)

        WRITE (6,202) DILSTS
        IF (DILSTS .EQ. DIT$_CONNECTEVENT) GO TO 120
        WRITE (6, 777)
        STOP

C       Accept link

120     CNTOPD = 0
        DILSTS = DIT$NFACC (NETLN, DIT$K_LTYPE_ASCII, CNTOPD, OPTDAT)

        WRITE (6, 203) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 130
        WRITE (6, 777)
        STOP

C       Wait for a data event on the link

130     DILSTS = DIT$NFGND (NETLN, DIT$K_WAIT_YES)

        WRITE (6,202) DILSTS
        IF (DILSTS .EQ. DIT$_DATAEVENT) GO TO 140
        WRITE (6, 777)
        STOP

C       Read the data received over the link

140     MSGSIZ = 100
        MUNTSZ = 8

        DILSTS = DIT$NFRCV (NETLN, MUNTSZ, MSGSIZ, RECD, DIT$K_MSG_MSG,
      1 DIT$K_WAIT_YES)

        WRITE (6, 205) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 150
        WRITE (6, 777)
        STOP

150     WRITE (6, 779)
155     FORMAT (' ' 25A4)
        WRITE (6, 155) RECD

C       Send some data over the link

        WRITE (6, 778)
157     FORMAT (25A4)
        ACCEPT 157, SENDD

        MSGSIZ = 100
        MUNTSZ = 8

        DILSTS = DIT$NFSND (NETLN, MUNTSZ, MSGSIZ, SENDD, DIT$K_MSG_MSG)

        WRITE (6, 204) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 160
        WRITE (6, 777)
        STOP
```

```
C       Check for the link being closed

160     DILSTS = DIT$NFGND (NETLN, DIT$K_WAIT_YES)

        WRITE (6, 202) DILSTS
        IF (DILSTS .EQ. DIT$_ABREJEVENT) GO TO 170
        IF (DILSTS .EQ. DIT$_DISCONNECTEVENT) GO TO 170
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 170
        WRITE (6, 777)
        STOP

170     WRITE (6,175)
175     FORMAT (' PASF32 test successful ')
        STOP
        END


C ACTF32

C       This program opens an active link by connecting to the passive
C       task set up by the program PASC36.  Once the link is
C       established, user specified messages are sent in both
C       directions across the link.  Then the link is closed.

C Use the DIL interface files.

        INCLUDE 'SYS$LIBRARY:DIL.TLB (DIT$FORTRAN)'
        INCLUDE 'SYS$LIBRARY:DIL.TLB (DIL$FORTRAN)'

C Data records

        DIMENSION SENDD (25), RECD (25)

C DIL task to task routine parameters

        DIMENSION HSTNAM (4), OPTDAT (4), OBJID (4), DESCR (4)
        DIMENSION PASSWD (10), ACCT (10), USERID (10), TASKN (4)

        INTEGER NETLN, DILSTS, MSGSIZ, MUNTSZ, CNTOPD, SYNCDS

C       Link description fields

        DATA OBJID /'0', '    ', '    ', '    '/
        DATA DESCR /'SERV', 'ER  ', '    ', '    '/
        DATA TASKN /'    ', '    ', '    ', '    '/
        DATA HSTNAM /'    ', '    ', '    ', '    '/
        DATA PASSWD /'    ', '    ', '    ', '    ', '    ',
        1           '    ', '    ', '    ', '    ', '    '/
        DATA USERID /'    ', '    ', '    ', '    ', '    ',
        1           '    ', '    ', '    ', '    ', '    '/
        DATA ACCT /'    ', '    ', '    ', '    ', '    ', '    ',
        1         '    ', '    ', '    ', '    ', '    ', '    '/

        DATA OPTDAT /'    ', '    ', '    ', '    '/
```

```
C Program messages

777     FORMAT (' Invalid status returned... ')
778     FORMAT (' Enter some data to be sent over the link: ')
779     FORMAT (' Data received: ')
201     FORMAT (' NFOPA Status return: ', I12)
202     FORMAT (' NFGND Status return: ', I12)
204     FORMAT (' NFSND Status return: ', I12)
205     FORMAT (' NFRCV Status return: ', I12)
206     FORMAT (' NFCLS Status return: ', I12)

C       initialize sending and receiving message data fields

        DO 100 I = 1, 25
        SENDD (I) = 0
100     RECD (I) = 0

C       Ask for a connection to the passive link

        DILSTS = DIT$NFOPA (NETLN, HSTNAM, OBJID, DESCR, TASKN,
     1  USERID, PASSWD, ACCT, OPTDAT, DIT$K_WAIT_NO)

        WRITE (6, 201) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 110
        WRITE (6, 777)
        STOP

C       Wait for confirmation of the connection

110     DILSTS = DIT$NFGND (NETLN, DIT$K_WAIT_YES)

        WRITE (6, 202) DILSTS
        IF (DILSTS .EQ. DIT$_CONNECTEVENT) GO TO 120
        WRITE (6, 777)
        STOP

C       Send some data over the link

120     WRITE (6, 778)
125     FORMAT (25A4)
        ACCEPT 125, SENDD

        MSGSIZ = 100
        MUNTSZ = 8

        DILSTS = DIT$NFSND (NETLN, MUNTSZ, MSGSIZ, SENDD, DIT$K_MSG_MSG)

        WRITE (6, 204) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 130
        WRITE (6, 777)
        STOP

C       Wait for a data event on the link

130     DILSTS = DIT$NFGND (NELTN, DIT$K_WAIT_YES)

        WRITE (6,202) DILSTS
        IF (DILSTS .EQ. DIT$_DATAEVENT) GO TO 140
        WRITE (6, 777)
        STOP
```

```
C       Read the data received over the link

140     MSGSIZ = 100
        MUNTSZ = 8

        DILSTS = DIT$NFRCV (NETLN, MUNTSZ, MSGSIZ, RECD, DIT$K_MSG_MSG,
      1 DIT$K_WAIT_YES)

        WRITE (6, 205) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 150
        WRITE (6, 777)
        STOP

150     WRITE (6, 779)
155     FORMAT (' ' 25A4)
        WRITE (6, 155) RECD

C       Close the link to self

        SYNCDS = 0

        DILSTS = DIT$NFCLS (NETLN, SYNCDS, CNTOPD, OPTDAT)

        WRITE (6, 206) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 160
        WRITE (6, 777)
        STOP

160     WRITE (6,165)
165     FORMAT (' ACTF32 test successful ')
        STOP
        END
```

CHAPTER 11
VMS REMOTE FILE ACCESS

CHAPTER 11

VMS REMOTE FILE ACCESS

11.1 REMOTE FILE ACCESS FROM VMS COBOL

The information included in this section assumes

-You are writing a COBOL program
-You plan to use the program on a VMS system

To store a file number, file name and user attributes, to read or write a record you must represent several data items in your program. Users generally, but not necessarily, allocate space for these data items in WORKING-STORAGE.

11.1.1 Including the Interface Support Files

VMS systems have two different classes of Interface Support files for each supported language. The first class of files include native VMS-type names for each of the various codes for VMS COBOL and VMS FORTRAN. The second class includes files with TOPS-10/TOPS-20 COBOL compatible names for VMS COBOL, and files that include ANSI Standard names for the interface to VMS FORTRAN. The Interface Support files for VMS are provided as a text library called DIL.TLB. The library includes the support files for both VMS COBOL and VMS FORTRAN.

Native VMS COBOL

You can use the COBOL COPY verb to retrieve information from DIL.TLB at compilation time. There are three library elements for Native VMS COBOL in DIL.TLB. The elements are DIL$COBOL, DIT$COBOL, and DIX$COBOL.

The library element DIL$COBOL defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$COBOL. Severity codes and system codes are defined in element DIL$COBOL. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration:

```
COPY DIL$COBOL OF "SYS$LIBRARY:DIL.TLB".
```

In the following example, the DIL$COBOL element of the library is retrieved and included in your program:

```
01  interface-files.
    COPY DIL$COBOL OF "SYS$LIBRARY:DIL.TLB".
```

The library element DIT$COBOL defines codes specific to the Task-to-Task and Remote File Access Routines. This includes the DIT status codes which are standard VMS condition values. Also included are Remote File Access file types, Remote File Access open modes, Remote File Access record formats, Remote File Access record attributes and Remote File Access close options. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration as shown above for the DIL$COBOL element:

```
COPY DIT$COBOL OF "SYS$LIBRARY:DIL.TLB".
```

For programs which use the Remote File Access Routines with the native VMS names, you must include both the DIL$COBOL and DIT$COBOL library elements.

TOPS-10/TOPS-20 Compatible COBOL

If you want to write a program that can be easily transported to a DECsystem-10 or a DECSYSTEM-20, you may want to include the TOPS-10/TOPS-20 compatible names in your program rather than the native VMS names.

You can use the COBOL COPY verb to retrieve information from DIL.TLB at compilation time. There are three library elements for TOPS-10/TOPS-20 compatible COBOL in DIL.TLB. The elements are DIL$COBOL_20, DIT$COBOL_20, and DIX$COBOL_20.

The library element DIL$COBOL_20 defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$COBOL_20. Severity codes and system codes are defined in element DIL$COBOL_20. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration:

```
COPY DIL$COBOL_20 OF "SYS$LIBRARY:DIL.TLB".
```

In the following example, the DIL$COBOL_20 element of the library is retrieved and included in your program:

```
01  interface-files.
    COPY DIL$COBOL_20 OF "SYS$LIBRARY:DIL.TLB".
```

The library element DIT$COBOL_20 defines codes specific to the Task-to-Task and Remote File Access Routines. The codes include the DIT status codes in the compatible COBOL format which provides only the condition identification portion of the status code. Also included are Remote File Access file types, Remote File Access open modes, Remote File Access record formats, Remote File Access record attributes and Remote File Access close options. To define these names in your program, include the following statement in your WORKING-STORAGE section after an 01-level declaration as shown above for the DIL$COBOL_20 element.

```
COPY DIT$COBOL_20 OF "SYS$LIBRARY:DIL.TLB".
```

For programs that use the Remote File Access Routines with the compatible names, you must include both the DIL$COBOL_20 and DIT$COBOL_20 library elements.

11.1.2 Storing a File Number

A file number consists of one longword of VMS memory. To store a file number, you must define a data item with the following format:

```
01  your-fnum    PIC S9(9) USAGE COMPUTATIONAL.
```

When you call the DIT$ROPEN routine use your-fnum as the value for the file number to be returned. When the routine successfully finishes processing, it returns a value in your-fnum.

11.1.3 Storing Account Information

To include your user code, password and account number or charge code in a call to the DIT$ROPEN routine, you must describe these attributes as data items in Working-Storage. The format for these data items must be exactly as follows:

```
01  userid          PIC X(39)       USAGE DISPLAY.
01  passwd          PIC X(39)       USAGE DISPLAY.
01  acct            PIC X(39)       USAGE DISPLAY.
```

11.1.4 Reading and Writing Remote Data

To read or write data from a file on a remote system, the program on the local system must define an area to accept the data and a one-word computational item for the length of the data.

11.1.5 Checking the Status of a Remote File Access Routine

Section 9.1.4 explains how to check the status of a DIL routine from VMS COBOL.

A simple call to the RFA Routines with an error check might be:

```
CALL "DIT$ROPEN" USING fnum, fnam, userid, passwd, acct, mode,
   dattyp, recfor, recatt, recsiz, runit GIVING stat.
IF stat IS FAILURE
   THEN DISPLAY "error".
```

11.2 REMOTE FILE ACCESS FROM VMS FORTRAN

The information included in this section assumes

-You are writing a FORTRAN program
-You plan to use the program on a VMS system

This section explains how to store a file number, file name and user attributes. It also explains how to read or write a record or to perform a status check.

11.2.1 Including the Interface Support Files

VMS systems have two different types of Interface Support files for each supported language. The first class of files includes native VMS type names for each of the various codes for VMS COBOL and VMS FORTRAN. The second class includes files with TOPS-10/TOPS-20 COBOL compatible names for VMS COBOL, and files that include ANSI Standard names for the interface to VMS FORTRAN. The Interface Support files for the VMS are provided as a text library called DIL.TLB. The library includes the support files for both VMS COBOL and VMS FORTRAN.

Native VMS FORTRAN

You can use the FORTRAN INCLUDE statement to retrieve information from DIL.TLB at compilation time. There are three library elements for Native VMS FORTRAN in DIL.TLB. The elements are DIL$FORTRAN, DIT$FORTRAN, and DIX$FORTRAN.

The library element DIL$FORTRAN defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$FORTRAN. Severity codes and system codes are defined in element DIL$FORTRAN. To define these names in your program, include the statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIL$FORTRAN)'
```

This retrieves the DIL$FORTRAN element of the library file and includes it in your program.

The library element DIT$FORTRAN defines codes specific to the Task-to-Task and Remote File Access Routines. The codes include the DIT status codes which are standard VMS condition values. Also included are Remote File Access file types, Remote File Access open modes, Remote File Access record formats, Remote File Access record attributes and Remote File Access close options. To define these names in your program, include the statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIT$FORTRAN)'
```

For programs which use the Remote File Access Routines with the native VMS names, you must include both the DIL$FORTRAN and DIT$FORTRAN library elements.

ANSI Standard Compatible FORTRAN

If you want to write a program that can be easily transported to another system, you may want to include the ANSI Standard FORTRAN names in your program rather than the native VMS names.

You can use the FORTRAN INCLUDE statement to retrieve information from DIL.TLB at compilation time. There are three library elements for ANSI Standard FORTRAN in DIL.TLB. The elements are DIL$ANSI_FORTRAN, DIT$ANSI_FORTRAN, and DIX$ANSI_FORTRAN.

The library element DIL$ANSI_FORTRAN defines general codes and names applicable to the Data Conversion Routines, the Task-to-Task Routines and the Remote File Access Routines. The general success status code (SS-NORMAL) is defined in element DIL$ANSI_FORTRAN. Severity codes and system codes are defined in element DIL$ANSI_FORTRAN. To define these names in your program, include the statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIL$ANSI_FORTRAN)'
```

This retrieves the DIL$ANSI_FORTRAN element of the library file and includes it in your program.

The library element DIT$ANSI_FORTRAN defines codes specific to the Task-to-Task and Remote File Access Routines. This includes the DIT status codes which are standard VMS condition values. Also included are Remote File Access file types, Remote File Access open modes, Remote File Access Record formats, Remote File Access record attributes and Remote File Access close options. To define these names in your program, include the following statement:

```
INCLUDE 'SYS$LIBRARY:DIL.TLB (DIT$ANSI_FORTRAN)'
```

For programs that use the remote file access routines with the compatible names, you must include both the DIL$ANSI_FORTRAN and DIT$ANSI_FORTRAN library elements.

11.2.2 Storing a File Number

To store a file number, you must implicitly or explicitly declare an integer with the following format:

```
INTEGER    fnum
```

When you call the DIT$ROPEN routine use fnum as the file number to be returned.

11.2.3 Storing Account Information

To include your user code, password, account number (or charge code) in a call to the DIT$ROPEN routine, you must first declare these values in your program. The format for these data items is as follows:

```
INTEGER userid (10)   or   CHARACTER*39   userid
INTEGER passwd (10)   or   CHARACTER*39   passwd
INTEGER acct (10)     or   CHARACTER*39   acct
```

NOTE

By default, VAX-11 FORTRAN passes arguments of type CHARACTER by descriptor. In order to use CHARACTER type variables as arguments to DIL (on VMS), the %REF built-in function must be used because DIL string arguments must be passed by reference.

11.2.4 Reading and Writing Remote Data

To read or write data from a file on a remote system, you must declare an area on the local system to accept the data, and an integer for the length of the data.

11.2.5 Checking the Status of a Remote File Access Routine

Section 9.2.4 explains how to check the status of a DIL Routine from VMS FORTRAN.

A simple call with an error check might then be:

```
        status = DIT$RREAD (fnum,dunit,maxsiz,bufnam)
        IF (status .NE. SS$_NORMAL) GOTO 100
        .
        .
        .
100     TYPE 101
101     FORMAT (' error occurred')
        .
        .
        .
```

11.3 VMS REMOTE FILE ACCESS REFERENCE

11.3.1 DIT$ROPEN - Open a Remote File

PURPOSE:

This routine opens a remote, sequential ASCII file for processing. It also assigns a file number to the opened file.

CALL FORMAT:

COBOL: CALL "DIT$ROPEN" USING fnum, fnam, userid, passwd, acct, mode, dattyp, recfor, recatt, recsiz, runit GIVING stat.

FORTRAN: status = DIT$ROPEN (fnum, fnam, userid, passwd, acct, 1 mode, dattyp, recfor, recatt, recsiz, runit)

where:

fnum is a long integer that gives the file number of the file that you want to open. This number is assigned by the subroutine.

fnam is an ASCII-8 character string that gives the name of the file to be opened. The file name must also include the node name of the file's system of origin. See Section 4.1 of this manual for further information.

NOTE

The next three arguments supply accounting information. If the remote node is a DECSYSTEM-20 or DECsystem-10 you must supply a userid and password. You must also specify an account unless the remote system sets a default account. If the remote node is a VAX, these parameters are optional. If you do not specify accounting information, the VMS system uses the default DECnet directory.

userid is your user code. A user code contains thirty-nine ASCII-8 characters. If your user code is less than thirty-nine characters, left-justify the field.

passwd is your password. A password contains thirty-nine ASCII-8 characters. If your password is less than thirty-nine characters, left-justify the field.

acct is your account. An account contains thirty-nine ASCII-8 characters. If your account is less than thirty-nine characters, left-justify the field.

mode is a long integer that indicates the mode in which you plan to use the file after it is opened. This version of the DIL allows you to read the file, write the file or append data to the file. Check with your system manager to make sure the FAL on the file's computer allows the type of access you have in mind.

Mode	DIL Name
Read	MODE-READ
Write	MODE-WRITE
Append data to this file	MODE-APPEND

dattyp is a long integer that indicates the data type of the file. The DIL name for this argument is:

TYPE-ASCII

recfor is a long integer that gives the record format. This argument, and the next three arguments, refer to the records which you plan to process after opening the file with the DIT$ROPEN routine.

Record Format	DIL Name
Undefined	RFM-UNDEFINED
Fixed length	RFM-FIXED
Variable length	RFM-VARIABLE
Variable length with fixed length control (VFC)	RFM-VFC
Stream	RFM-STREAM

recatt is a long integer that indicates the record attributes of the file that you plan to process.

Record Attribute	DIL Name
Unspecified	RAT-UNSPECIFIED
Implied <LF><CR> envelope	RAT-ENVELOPE
VMS printer carriage control	RAT-PRINT
FORTRAN carriage control	RAT-FORTRAN

Record attribute
RAT-PRINT is only valid for record format RFM-VFC.

NOTE

If you plan to use the RFA routines to read a record from another system, you can use the RFM-UNDEFINED or RAT-UNSPECIFIED values for the "recfor" and "recatt" arguments. You only have to specify values for these arguments if you want to write a file on a VMS system.

recsiz is a long integer that gives the record size, in bytes. If you plan to write the file, this will be its maximum record size. Use zero if you want to read or append to the file. Use zero if you don't want to give the file a maximum record size.

runit is a long integer that gives the record size unit. The value of this argument is always zero.

STATUS CODES:

DIL Name	Meaning
DIT-TOOMANY	You can't open any more files. Version 2.0 of the DIL allows a maximum of 20 open files.
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-UNSFILETYPE	You attempted to write a file whose file type is unsupported on the remote system.
DIT-FILEINUSE	The file is already being processed by another program.
DIT-NOFILE	The file does not exist or is not available to you.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

11.3.2 DIT$RREAD - Read Data From a Remote File

PURPOSE:

The DIT$RREAD routine reads a record from a file opened with the DIT$ROPEN routine.

CALL FORMAT:

COBOL: CALL "DIT$RREAD" USING fnum, dunit, maxsiz, bufnam GIVING stat.

FORTRAN: status = DIT$RREAD (fnum, dunit, maxsiz, bufnam)

where:

fnum is a long integer that gives the number of the file that you want to read. File number is assigned by the DIT$ROPEN routine.

dunit is a long integer that gives the data unit size. The value for this argument is always zero.

maxsiz is a long integer that specifies the maximum record size in characters. When DIT$RREAD finishes processing, this argument contains the number of characters actually read by the routine.

bufnam is where the routine places the data that it reads. It is an ASCII-8 character string that must contain room for at least the number of characters specified in "maxsiz," above.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-EOF	The routine reached the end of the file it was reading.
DIT-OVERRUN	The record being read is too large to fit into the user buffer area.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

11.3.3 DIT$RWRITE - Write Data to a Remote File

PURPOSE:

The DIT$RWRITE routine writes a record into a file opened by DIT$ROPEN.

CALL FORMAT:

```
COBOL:    CALL "DIT$RWRITE" USING fnum, dunit, length, data
               GIVING stat.

FORTRAN:  status = DIT$RWRITE (fnum, dunit, length, data)
```

where:

fnum — is a long integer that gives the file number of the file that you want to write. File number is assigned by the DIT$ROPEN routine.

dunit — is a long integer that specifies the data unit size. The value of this argument is always zero.

length — is a long integer that gives the data length, in characters.

data — is an ASCII-8 character string that represents the data to write.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network check-sum error.
DIT-NOFILE	The file does not exist or is not available to you.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

11.3.4 DIT$RCLOSE - Close a Remote File

PURPOSE:

The DIT$RCLOSE routine closes the file that you opened with the DIT$ROPEN routine.

CALL FORMAT:

COBOL: CALL "DIT$RCLOSE" USING fnum, clopt GIVING stat.

FORTRAN: status = DIT$RCLOSE (fnum, clopt)

where:

fnum is a long integer that gives the file number of the file that you want to close. File number is assigned by the DIT$ROPEN routine.

clopt is a long integer that gives the close option. This argument tells what you want to do with the file once it's closed.

Close Option	DIL Name
No special action	OPT-NOTHING
Submit file for remote batch processing	OPT-SUBMIT
Submit file for remote printing	OPT-PRINT
Delete the remote file	OPT-DELETE

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

11.3.5 DIT$RDEL - Delete a File

PURPOSE:

The DIT$RDEL routine deletes a file. You can only delete a closed file. If you want to delete an open file, delete it with the DIT$CLOSE routine using the OPT-DELETE close option.

CALL FORMAT:

COBOL: CALL "DIT$RDEL" USING fnam, userid, passwd, acct
GIVING stat.

FORTRAN: status = DIT$RDEL (fnam, userid, passwd, acct)

where:

fnam is an ASCII-8 character string that gives the name of the file to be deleted. The file name must also include the node name of the file's system of origin.

NOTE

The next three arguments supply accounting information. If the remote node is a DECSYSTEM-20 or DECsystem-10 you must supply a userid and password. You must also specify an account unless the remote system sets a default account. If the remote node is a VAX, these parameters are optional. If you do not specify accounting information, the VMS system uses the default DECnet directory.

userid is your user code. This field contains thirty-nine ASCII-8 characters. If your user code is less than thirty-nine characters, left-justify the field.

passwd is your password. This field contains thirty-nine ASCII-8 characters. If your password is less than thirty-nine characters, left-justify the field.

acct is your account. This field contains thirty-nine ASCII-8 characters. If your account is less than thirty-nine characters, left-justify the field.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-NOFILE	The file does not exist or is not available to you.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

11.3.6 DIT$RSUB - Submit a File For Batch Processing

PURPOSE:

The DIT$RSUB routine submits a remote file for batch processing on the remote system. You may only submit closed files for processing. If you want to submit an open file for batch processing, use the DIT$RCLOSE routine with the OPT-SUBMIT close option.

CALL FORMAT:

```
COBOL:    CALL "DIT$RSUB" USING fnam, userid, passwd, acct
               GIVING stat.

FORTRAN:  status = DIT$RSUB (fnam, userid, passwd, acct)
```

where:

fnam — is an ASCII-8 character string that gives the name of the file to be submitted. The file name must also include the node name of the file's system of origin.

NOTE

The next three arguments supply accounting information. If the remote node is a DECSYSTEM-20 or DECsystem-10 you must supply a userid and password. You must also specify an account unless the remote system sets a default account. If the remote node is a VAX, these parameters are optional. If you do not specify accounting information, the VMS system uses the default DECnet directory.

userid — is your user code. This field contains thirty-nine ASCII-8 characters. If your user code is less than thirty-nine characters, left-justify the field.

passwd — is your password. This field contains thirty-nine ASCII-8 characters. If your password is less than thirty-nine characters, left-justify the field.

acct — is your account. This field contains thirty-nine ASCII-8 characters. If your account is less than thirty-nine characters, left-justify the field.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-NOFILE	The file does not exist or is not available to you.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

11.3.7 DIT$RPRINT - Print a File

PURPOSE:

The DIT$RPRINT routine prints a remote file at the remote system. If you want to print an open file, use the DIT$RCLOSE routine with the OPT-PRINT close option.

CALL FORMAT:

```
COBOL:    CALL "DIT$RPRINT" USING fnam, userid, passwd, acct
               GIVING stat.

FORTRAN:  status = DIT$RPRINT (fnam, userid, passwd, acct)
```

where:

fnam — is an ASCII-8 character string that gives the name of the file to be printed. The file name must also include the node name of the file's system of origin.

NOTE

The next three arguments supply accounting information. If the remote node is a DECSYSTEM-20 or DECsystem-10 you must supply a userid and password. You must also specify an account unless the remote system sets a default account. If the remote node is a VAX, these parameters are optional. If you do not specify accounting information, the VMS system uses the default DECnet directory.

userid — is your user code. This field contains thirty-nine ASCII-8 characters. If your user code is less than thirty-nine characters, left-justify the field.

passwd — is your password. This field contains thirty-nine ASCII-8 characters. If your password is less than thirty-nine characters, left-justify the field.

acct — is your account. This field contains thirty-nine ASCII-8 characters. If your account is less than thirty-nine characters, left-justify the field.

STATUS CODES:

DIL Name	Meaning
DIT-INVARG	You passed an incorrect or invalid argument.
SS-NORMAL	The routine successfully completed processing.
DIT-NETOPRFAIL	You attempted an impossible network operation.
DIT-CHECKSUM	The network returned a network checksum error.
DIT-NOFILE	The file does not exist or is not available to you.
DIT-HORRIBLE	This code is returned in the event of a system or unexpected error.

11.4 VMS REMOTE FILE ACCESS EXAMPLES

11.4.1 VMS COBOL Remote File Access Examples

```
IDENTIFICATION DIVISION.

PROGRAM-ID.

        CDAP32.

        This program opens a remote  file named DAP.TST and writes  an
        ASCII record into it,  closes the file,  reopens the file  and
        reads the record back  and then closes  the file again.   This
        program tries to  write and  read the file  DAP.TST using  the
        default DECnet directory.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER.

        VAX-11.

OBJECT-COMPUTER.

        VAX-11.

DATA DIVISION.

WORKING-STORAGE SECTION.

01  INTERFACE-FILES.
    COPY DIT$COBOL OF "SYS$LIBRARY:DIL.TLB".
    COPY DIL$COBOL OF "SYS$LIBRARY:DIL.TLB".

* DIL status return
01  DIL-STATUS PIC S9(9) COMP.

* File and directory description fields

01  FILE-NAME PIC X(39) VALUE "DAP.TST".
01  USERID PIC X(39) VALUE SPACES.
01  PASSWD PIC X(39) VALUE SPACES.
01  ACCT PIC X(39) VALUE SPACES.

* Record and file description fields

01  FILE-NUMBER USAGE COMP PIC S9(9).
01  RECORD-UNIT-SIZE USAGE COMP PIC S9(9) VALUE 0.
01  RECORD-SIZE USAGE COMP PIC S9(9) VALUE 100.

01  DATA-RECORD PIC X(100).

01  DIL-DISPLAY PIC X(12).
```

```
PROCEDURE DIVISION.

BEGIN-CDAP32.

* Open file DAP.TST for output; note:  for remote file access, if  you
* leave the passwd and userid blank on a VAX, the connected  directory
* will be used

    CALL "DIT$ROPEN" USING FILE-NUMBER, FILE-NAME, USERID, PASSWD, ACCT,
                           DIT$K_MODE_WRITE, DIT$K_TYPE_ASCII,
                           DIT$K_RFM_STREAM, DIT$K_RAT_UNSPECIFIED,
                           RECORD-SIZE, RECORD-UNIT-SIZE
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " ROPEN Status return: " DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? ROPEN: unsuccessful status return "
        STOP RUN.

* Accept a record and write it to the file

    DISPLAY " Enter data for the record for the remote file: ".
    ACCEPT DATA-RECORD.

    CALL "DIT$RWRITE" USING FILE-NUMBER, RECORD-UNIT-SIZE,
                            RECORD-SIZE, DATA-RECORD
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " RWRITE Status return: " DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? RWRITE: unsuccessful status return. "
        STOP RUN.

* Close the file

    CALL "DIT$RCLOSE" USING FILE-NUMBER, DIT$K_OPT_NOTHING
                      GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " RCLOSE Status return: ", DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? RCLOSE: unsuccessful status return."
        STOP RUN.

* Open the file to read the record

    CALL "DIT$ROPEN" USING FILE-NUMBER, FILE-NAME, USERID, PASSWD, ACCT,
                           DIT$K_MODE_READ, DIT$K_TYPE_ASCII,
                           DIT$K_RFM_STREAM, DIT$K_RAT_UNSPECIFIED,
                           RECORD-SIZE, RECORD-UNIT-SIZE
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " ROPEN Status return: ", DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? ROPEN: unsuccessful status return."
        STOP RUN.
```

```
* Read the record

    MOVE SPACES TO DATA-RECORD.

    CALL "DIT$RREAD" USING FILE-NUMBER, RECORD-UNIT-SIZE,
                           RECORD-SIZE, DATA-RECORD
                     GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " RREAD returned ", DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? RREAD: unsuccesful status return."
        STOP RUN.

    DISPLAY " The record was: ".
    DISPLAY DATA-RECORD.

* Close the file

    CALL "DIT$RCLOSE" USING FILE-NUMBER, DIT$K_OPT_NOTHING
                      GIVING DIL-STATUS.

    MOVE DIL-STATUS TO DIL-DISPLAY.
    DISPLAY " RCLOSE Status return: ", DIL-DISPLAY.
    IF DIL-STATUS IS NOT SUCCESS
        DISPLAY "? RCLOSE: unsuccessful status return."
        STOP RUN.

    DISPLAY " CDAP32 successful. ".

    STOP RUN.
```

11.4.2 VMS FORTRAN Remote File Access Example

```
C FDAP32
C       This program opens a remote  file named DAP.TST and writes  an
C       ASCII record into it,  closes the file,  reopens the file  and
C       reads the record back and then closes the file again.

C Use the DIL interface files.

        INCLUDE 'SYS$LIBRARY:DIL.TBL (DIT$FORTRAN)'
        INCLUDE 'SYS$LIBRARY:DIL.TBL (DIL$FORTRAN)'

C File and directory description fields
        INTEGER FILNAM (10), USERID (10), PASSWD (10), ACCT (10), FILNUM
C Sending and receiving data records
        INTEGER DATA1 (25), DATA2 (25)
C DIL Status code
        INTEGER DILSTS
C Record size and record unit size
        INTEGER RECSIZ, RUNTSZ


        DATA FILNAM /'DAP.', 'TST ', '    ', '    ', '    ',
        1       '    ', '    ', '    ', '    ', '    '/

C Note: For remote file access on the VAX, if you leave the userid and
C passwd blank, the connected directory will be used.

        DATA USERID /'    ', '    ', '    ', '    ','    ',
        1       '    ', '    ', '    ', '    ', '    '/
        DATA PASSWD /'    ', '    ', '    ', '    ', '    ',
        1       '    ', '    ', '    ', '    ', '    '/
        DATA ACCT /'    ', '    ', '    ', '    ', '    ',
        1       '    ', '    ', '    ', '    ', '    '/

C Program messages

200     FORMAT (' ROPEN status return: ', I10)
201     FORMAT (' RWRITE status return: ', I10)
202     FORMAT (' RCLOSE status return: ', I10)
203     FORMAT (' RREAD status return: ', I10)
700     FORMAT ('? Invalid status returned... ')


C Request the password

100     FORMAT (' Enter the password:')
        WRITE (6,100)
105     FORMAT (10A4)
        ACCEPT 105, PASSWD
```

```
C Open file DAP.TST for output.

        RUNTSZ = 0
        RECSIZ = 100

        DILSTS = DIT$ROPEN (FILNUM, FILNAM, USERID, PASSWD, ACCT,
        1       DIT$K_MODE_WRITE, DIT$K_TYPE_ASCII, DIT$K_RFM_STREAM,
        2       DIT$K_UNSPECIFIED, RECSIZ, RUNTSZ)

        WRITE (6,200) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 115
        WRITE (6,700)
        STOP

C Accept a record and write it to the file

110     FORMAT (' Enter data for the record:')
115     WRITE (6,110)
120     FORMAT (25A4)
        ACCEPT 120, DATA1

        DILSTS = DIT$RWRITE (FILNUM, RUNTSZ, RECSIZ, DATA1)

        WRITE (6,201) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 125
        WRITE (6,700)
        STOP

C Close the file.

125     DILSTS = DIT$RCLOSE (FILNUM, DIT$K_OPT_NOTHING)

        WRITE (6,202) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 130
        WRITE (6,700)
        STOP

C Open the file to read the record


130     DILSTS = DIT$ROPEN (FILNUM, FILNAM, USERID, PASSWD, ACCT,

        1       DIT$K_MODE_READ, DIT$K_TYPE_ASCII, DIT$K_RFM_STREAM,
        2       DIT$K_RAT_UNSPECIFIED, RECSIZ, RUNTSZ)

        WRITE (6,200) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 135
        WRITE (6,700)
        STOP
```

```
C Read the record

135     DILSTS = DIT$RREAD (FILNUM, RUNTSZ, RECSIZ, DATA2)


        WRITE (6,203) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 145
        WRITE (6,700)
        STOP

140     FORMAT (' The record read was: ')
145     WRITE (6,140)
146     FORMAT (' ', 25A4)
        WRITE (6,146) DATA2

C Close the file

        DILSTS = DIT$RCLOSE (FILNUM, DIT$K_OPT_NOTHING)

        WRITE (6,202) DILSTS
        IF (DILSTS .EQ. SS$_NORMAL) GO TO 155

        WRITE (6,700)
        STOP

150     FORMAT (' FDAP32 successful ')
155     WRITE (6,150)
        STOP
        END
```

CHAPTER 12
LINKING A VMS PROGRAM

CHAPTER 12

LINKING A VMS PROGRAM

The linker associates your program with the Data Interchange Library routines that your program calls. VMS supports two forms of the DIL, an object library and a shareable image. When you have successfully compiled your program, link it by typing the following command sequence:

For object library:

```
$ LINK usrprg, SYS$LIBRARY:DIL/LIBRARY, SYS$LIBRARY:XPORT/LIBRARY
```

where usrprg is the name of your program.

For sharable image:

```
$ LINK usrprg, SYS$LIBRARY:DIL/OPTIONS
```

where: usrprg is the name of your program, and OPTIONS identifies SYS$LIBRARY:DIL.OPT as a file containing VMS linker option specifications.

APPENDIX A
LANGUAGE–SPECIFIC VALUES FOR DIL NAMES

APPENDIX A

LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

This appendix shows the DIL Names for status codes, severity codes, data type names and routine parameter values as defined in the DIL Interface Support files. It then gives the equivalent names for COBOL and FORTRAN on the VAX, the DECsystem-10 and the DECSYSTEM-20.

If you plan to write a COBOL program on a TOPS-10 or TOPS-20 system, use the ANSI COBOL names. If you plan to write a FORTRAN program on a TOPS-10 or TOPS-20 system, use the ANSI FORTRAN names. If you want to write a program on a VMS system use the Native Vax names. If you plan to write a program which can be easily transported between TOPS-10/TOPS-20 and VMS, use the ANSI COBOL or ANSI FORTRAN names.

LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

STATUS CODES:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX
DIT-ABORTREJECT	DIT-C-ABORTREJECT	ABRTRJ	DIT$_ABORTREJECT
DIT-ABREJEVENT	DIT-C-ABREJEVENT	ARJEVT	DIT$_ABREJEVENT
DIT-CHECKSUM	DIT-C-CHECKSUM	CHKSUM	DIT$_CHECKSUM
DIT-CONNECTEVENT	DIT-C-CONNECTEVENT	CONEVT	DIT$_CONNECTEVENT
DIT-DATAEVENT	DIT-C-DATAEVENT	DATEVT	DIT$_DATAEVENT
DIT-DISCONNECTEVENT	DIT-C-DISCONNECTEVENT	DSCEVT	DIT$_DISCONNECTEVENT
DIT-EOF	DIT-C-EOF	DITEOF	DIT$_EOF
DIT-FILEINUSE	DIT-C-FILEINUSE	FILIU	DIT$_FILEINUSE
DIT-HORRIBLE	DIT-C-HORRIBLE	SYSERR	DIT$_HORRIBLE
DIT-INFONOTAVAIL	DIT-C-INFONOTAVAIL	NOTAVL	DIT$_INFONOTAVAIL
DIT-INFOOUTOFRANGE	DIT-C-INFOOUTOFRANGE	INFOUR	DIT$_INFOOUTOFRANGE
DIT-INTDATAEVENT	DIT-C-INTDATAEVENT	INTEVT	DIT$_INTDATAEVENT
DIT-INTERRUPT	DIT-C-INTERRUPT	INTRCV	DIT$_INTERRUPT
DIT-INVARG	DIT-C-INVARG	INVARG	DIT$_INVARG
DIT-NETOPRFAIL	DIT-C-NETOPRFAIL	NETFAL	DIT$_NETOPRFAIL
DIT-NODATAAVAILABLE	DIT-C-NODATAAVAILABLE	NODATA	DIT$_NODATAAVAILABLE
DIT-NOFILE	DIT-C-NOFILE	NOFILE	DIT$_NOFILE
DIT-NOMOREFILES	DIT-C-NOMOREFILES	NOMORE	DIT$_NOMOREFILES
DIT-NOTENOUGH	DIT-C-NOTENOUGH	NOTENF	DIT$_NOTENOUGH
DIT-OVERRUN	DIT-C-OVERRUN	OVRRUN	DIT$_OVERRUN
DIT-TOOMANY	DIT-C-TOOMANY	TOOMNY	DIT$_TOOMANY
DIT-UNSFILETYPE	DIT-C-UNSFILETYPE	UNSTYP	DIT$_UNSFILETYPE
DIX-ALIGN	DIX-C-ALIGN	ALIGN	DIX$_ALIGN
DIX-FMTLOST	DIX-C-FMTLOST	FMTLST	DIX$_FMTLOST
DIX-GRAPHIC	DIX-C-GRAPHIC	GRAPHC	DIX$_GRAPHIC
DIX-IMPOSSIBLE	DIX-C-IMPOSSIBLE	IMPOSS	DIX$_IMPOSSIBLE
DIX-INVALCHAR	DIX-C-INVALCHAR	INVCHR	DIX$_INVALCHAR
DIX-INVBYTSIZ	DIX-C-INVBYTSIZ	BYTSIZ	DIX$_INVBYTSIZ
DIX-INVDATTYP	DIX-C-INVDATTYP	DATTYP	DIX$_INVDATTYP
DIX-INVPDDGT	DIX-C-INVPDDGT	PDDGT	DIX$_INVPDDGT
DIX-INVPDSGN	DIX-C-INVPDSGN	PDSGN	DIX$_INVPDSGN
DIX-INVLNG	DIX-C-INVLNG	INVLNG	DIX$_INVLNG
DIX-INVDNCHR	DIX-C-INVDNCHR	DNMCHR	DIX$_INVDNCHR
DIX-INVDNSGN	DIX-C-INVDNSGN	DNMSGN	DIX$_INVDNSGN
DIX-INVSCAL	DIX-C-INVSCAL	INVSCL	DIX$_INVSCAL
DIX-NONPRINT	DIX-C-NONPRINT	NONPRN	DIX$_NONPRINT
DIX-ROUNDED	DIX-C-ROUNDED	RNDED	DIX$_ROUNDED
DIX-TOOBIG	DIX-C-TOOBIG	TOOBIG	DIX$_TOOBIG
DIX-TRUNC	DIX-C-TRUNC	TRUNC	DIX$_TRUNC
DIX-UNIMP	DIX-C-UNIMP	UNIMP	DIX$_UNIMP
DIX-UNKARGTYP	DIX-C-UNKARGTYP	ARGTYP	DIX$_UNKARGTYP
DIX-UNKSYS	DIX-C-UNKSYS	UNKSYS	DIX$_UNKSYS
DIX-UNNORM	DIX-C-UNNORM	UNNORM	DIX$_UNNORM
DIX-UNSIGNED	DIX-C-UNSIGNED	UNSIGN	DIX$_UNSIGNED
SS-NORMAL	SS-C-NORMAL	NORMAL	SS$_NORMAL

LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

SEVERITY CODES:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
STS-WARNING	STS-K-WARNING	STSWRN	STS$K_WARNING	0
STS-SUCCESS	STS-K-SUCCESS	STSSUC	STS$K_SUCCESS	1
STS-ERROR	STS-K-ERROR	STSERR	STS$K_ERROR	2
STS-INFO	STS-K-INFO	STSINF	STS$K_INFO	3
STS-SEVERE	STS-K-SEVERE	STSSEV	STS$K_SEVERE	4

SYSTEM CODES:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
SYS-10-20	DIX-SYS-10-20	SYS36	DIX$K_SYS_10_20	1
SYS-VAX	DIX-SYS-VAX	SYSVAX	DIX$K_SYS_VAX	2

REMOTE FILE ACCESS FILE TYPES:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
TYPE-ASCII	DIT-TYPE-ASCII	TASCII	DIT$K_TYPE_ASCII	1

REMOTE FILE ACCESS OPEN MODES:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
MODE-READ	DIT-MODE-READ	MREAD	DIT$K_MODE_READ	1
MODE-WRITE	DIT-MODE-WRITE	MWRITE	DIT$K_MODE_WRITE	2
MODE-APPEND	DIT-MODE-APPEND	MAPPND	DIT$K_MODE_APPEND	3

REMOTE FILE ACCESS RECORD FORMATS:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
RFM-UNDEFINED	DIT-RFM-UNDEFINED	FUNDEF	DIT$K_RFM_UNDEFINED	0
RFM-FIXED	DIT-RFM-FIXED	FFIXED	DIT$K_RFM_FIXED	1
RFM-VARIABLE	DIT-RFM-VARIABLE	FVAR	DIT$K_RFM_VARIABLE	2
RFM-VFC	DIT-RFM-VFC	FVFC	DIT$K_RFM_VFC	3
RFM-STREAM	DIT-RFM-STREAM	FSTM	DIT$K_RFM_STREAM	4

LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

REMOTE FILE ACCESS RECORD ATTRIBUTES:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
RAT-UNSPECIFIED	DIT-RAT-UNSPECIFIED	AUNSPC	DIT$K_RAT_UNSPECIFIED	0
RAT-ENVELOPE	DIT-RAT-ENVELOPE	AENVLP	DIT$K_RAT_ENVELOPE	1
RAT-PRINT	DIT-RAT-PRINT	APRINT	DIT$K_RAT_PRINT	2
RAT-FORTRAN	DIT-RAT-FORTRAN	AFTN	DIT$K_RAT_FORTRAN	3

REMOTE FILE ACCESS CLOSE OPTIONS:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
OPT-NOTHING	DIT-OPT-NOTHING	ONTHNG	DIT$K_OPT_NOTHING	0
OPT-SUBMIT	DIT-OPT-SUBMIT	OSBMIT	DIT$K_OPT_SUBMIT	1
OPT-PRINT	DIT-OPT-PRINT	OPRINT	DIT$K_OPT_PRINT	2
OPT-DELETE	DIT-OPT-DELETE	ODLETE	DIT$K_OPT_DELETE	4

TASK-TO-TASK WAIT CODES:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
WAIT-NO	DIT-WAIT-NO	WAITLN	DIT$K_WAIT_NO	0
WAIT-YES	DIT-WAIT-YES	WAITLY	DIT$K_WAIT_YES	1

TASK-TO-TASK LINK TYPES for NFACC:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VAX VALUE	TOPS-10/20 VALUE
LTYPE-ASCII	DIT-LTYPE-ASCII	LASCII	DIT$K_LTYPE_ASCII	1	0
LTYPE-BINARY	DIT-LTYPE-BINARY	LBIN	DIT$K_LTYPE_BINARY	2	1
LTYPE-8BIT	DIT-LTYPE-8BIT	L8BIT	DIT$K_LTYPE_8BIT	3	2

TASK-TO-TASK MESSAGE MODES:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
MSG-MSG	DIT-MSG-MSG	MSGMSG	DIT$K_MSG_MSG	1
MSG-STM	DIT-MSG-STM	MSGSTM	DIT$K_MSG_STM	0

LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

VMS TASK FIRE-UP CODES:

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
PAS-FIREUP	DIT-PAS-FIREUP	FIREUP	DIT$K_PAS_FIREUP	0
PAS-NFIREUP	DIT-PAS-NFIREUP	NFREUP	DIT$K_PAS_NFIREUP	1

DATA TYPES

DIL NAME	ANSI COBOL	ANSI FORTRAN	NATIVE VAX	VALUE
ASCII-7	DIX-DT-ASCII-7	ASCII7	DIX$K_DT_ASCII_7	257
ASCII-8	DIX-DT-ASCII-8	ASCII8	DIX$K_DT_ASCII_8	258
ASCIZ	DIX-DT-ASCIZ	ASCIZ	DIX$K_DT_ASCIZ	259
D-FLOAT	DIX-DT-D-FLOAT	DFLOAT	DIX$K_DT_D_FLOAT	769
DN6LO	DIX-DT-DN6LO	DN6LO	DIX$K_DT_DN6LO	1025
DN6LS	DIX-DT-DN6LS	DN6LS	DIX$K_DT_DN6LS	1026
DN6TO	DIX-DT-DN6TO	DN6TO	DIX$K_DT_DN6TO	1027
DN6TS	DIX-DT-DN6TS	DN6TS	DIX$K_DT_DN6TS	1028
DN6U	DIX-DT-DN6U	DN6U	DIX$K_DT_DN6U	1029
DN7LO	DIX-DT-DN7LO	DN7LO	DIX$K_DT_DN7LO	1030
DN7LS	DIX-DT-DN7LS	DN7LS	DIX$K_DT_DN7LS	1031
DN7TO	DIX-DT-DN7TO	DN7TO	DIX$K_DT_DN7TO	1032
DN7TS	DIX-DT-DN7TS	DN7TS	DIX$K_DT_DN7TS	1033
DN7U	DIX-DT-DN7U	DN7U	DIX$K_DT_DN7U	1034
DN8LO	DIX-DT-DN8LO	DN8LO	DIX$K_DT_DN8LO	1035
DN8LS	DIX-DT-DN8LS	DN8LS	DIX$K_DT_DN8LS	1036
DN8TO	DIX-DT-DN8TO	DN8TO	DIX$K_DT_DN8TO	1037
DN8TS	DIX-DT-DN8TS	DN8TS	DIX$K_DT_DN8TS	1038
DN8U	DIX-DT-DN8U	DN8U	DIX$K_DT_DN8U	1039
DN9LO	DIX-DT-DN9LO	DN9LO	DIX$K_DT_DN9LO	1040
DN9LS	DIX-DT-DN9LS	DN9LS	DIX$K_DT_DN9LS	1041
DN9TO	DIX-DT-DN9TO	DN9TO	DIX$K_DT_DN9TO	1042
DN9TS	DIX-DT-DN9TS	DN9TS	DIX$K_DT_DN9TS	1043
DN9U	DIX-DT-DN9U	DN9U	DIX$K_DT_DN9U	1044
EBCDIC-8	DIX-DT-EBCDIC-8	EBCDC8	DIX$K_DT_EBCDIC_8	260
EBCDIC-9	DIX-DT-EBCDIC-9	EBCDC9	DIX$K_DT_EBCDIC_9	261
F-FLOAT	DIX-DT-F-FLOAT	FFLOAT	DIX$K_DT_F_FLOAT	770
FLOAT-36	DIX-DT-FLOAT-36	FLOT36	DIX$K_DT_FLOAT_36	771
FLOAT-72	DIX-DT-FLOAT-72	FLOT72	DIX$K_DT_FLOAT_72	772
G-FLOAT	DIX-DT-G-FLOAT	GFLOAT	DIX$K_DT_G_FLOAT	773
G-FLOAT72	DIX_DT_G_FLOAT72	GFLO72	DIX$K_DT_G_FLOAT72	774
H-FLOAT	DIX-DT-H-FLOAT	HFLOAT	DIX$K_DT_H_FLOAT	775
PD8	DIX-DT-PD8	PD8	DIX$K_DT_PD8	1281
PD9	DIX-DT-PD9	PD9	DIX$K_DT_PD9	1282
SIXBIT	DIX-DT-SIXBIT	SIXBIT	DIX$K_DT_SIXBIT	262
SBF128	DIX-DT-SBF128	SBF128	DIX$K_DT_SBF128	513
SBF16	DIX-DT-SBF16	SBF16	DIX$K_DT_SBF16	514
SBF32	DIX-DT-SBF32	SBF32	DIX$K_DT_SBF32	515
SBF36	DIX-DT-SBF36	SBF36	DIX$K_DT_SBF36	516
SBF18	DIX-DT-SBF48	SBF48	DIX$K_DT_SBF48	517
SBF64	DIX-DT-SBF64	SBF64	DIX$K_DT_SBF64	518
SBF72	DIX-DT-SBF72	SBF72	DIX$K_DT_SBF72	519
SBF8	DIX-DT-SBF8	SBF8	DIX$K_DT_SBF8	520
UBF16	DIX-DT-UBF16	UBF16	DIX$K_DT_UBF16	522
UBF32	DIX-DT-UBF32	UBF32	DIX$K_DT_UBF32	523
UBF8	DIX-DT-UBF8	UBF8	DIX$K_DT_UBF8	524

LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

A.1 SPECIFYING DATA NAMES

TOPS-10/TOPS-20 COBOL:

Character sets as specified in the ALPHABET clause in the SPECIAL-NAMES paragraph of the ENVIRONMENT DIVISION:

APLHABET clause specification	DIL data type
ASCII	ASCII-7
STANDARD-1	ASCII-7
NATIVE	ASCII-7
EBCDIC	EBCDIC-9
user specified literals	unsupported

PICTURE clause	USAGE clause	SIGN clause	DIL DATA TYPE	DIL FIELD LENGTH
9(n) where 1<=n<=18	unspecified or DISPLAY or DISPLAY-6		DN6u (or SIXBIT)	n
S9(n) where 1<=n<=18	unspecified or DISPLAY or DISPLAY-6		DN6TO (or SIXBIT)	n
S9(n) where 1<=n<=18	unspecified or DISPLAY or DISPLAY-6	LEADING or LEADING OVERPUNCHED	DN6LO (or SIXBIT)	n
S9(n) where 1<=n<=18	unspecified or DISPLAY or DISPLAY-6	TRAILING or TRAILING OVERPUNCHED	DN6TO (or SIXBIT)	n
S9(n) where 1<=n<=18	unspecified or DISPLAY or DISPLAY-6	LEADING SEPARATE	DN6LS (or SIXBIT)	n + 1
S9(n) where 1<=n<=18	unspecified or DISPLAY or DISPLAY-6	TRAILING SEPARATE	DN6TS (or SIXBIT)	n + 1
9(n) where 1<=n<=18	DISPLAY-7		DN7U (or ASCII-7)	n
S9(n) where 1<=n<=18	DISPLAY-7		DN7TO (or ASCII-7)	n
S9(n) where 1<=n<=18	DISPLAY-7	LEADING or LEADING OVERPUNCHED	DN7LO (or ASCII-7)	n
S9(n) where 1<=n<=18	DISPLAY-7	TRAILING or TRAILING OVERPUNCHED	DN7TO (or ASCII-7)	n
S9(n) where 1<=n<=18	DISPLAY-7	LEADING SEPARATE	DN7LS (or ASCII-7)	n + 1

LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

PICTURE clause	USAGE clause	SIGN clause	DIL DATA TYPE	DIL FIELD LENGTH
S9(n) where 1<=n<=18	DISPLAY-7	TRAILING SEPARATE	DN7TS (or ASCII-7)	n + 1
9(n) where 1<=n<=18	DISPLAY-9		DN9U (or EBCDIC)	n
S9(n) where 1<=n<=18	DISPLAY-9		DN9TO (or EBCDIC)	n
S9(n) where 1<=n<=18	DISPLAY-9	LEADING or LEADING OVERPUNCHED	DN9LO (or EBCDIC)	n
S9(n) where 1<=n<=18	DISPLAY-9	TRAILING or TRAILING OVERPUNCHED	DN9TO (or EBCDIC)	n
S9(n) where 1<=n<=18	DISPLAY-9	LEADING SEPARATE	DN9LS (or EBCDIC)	n + 1
S9(n) where 1<=n<=18	DISPLAY-9	TRAILING SEPARATE	DN9TS (or EBCDIC)	n + 1
9(n) or S9(n) where 1<=n<=10	COMP or COMPUTATIONAL		SBF36	0
9(n) or s9(n) where 10<=n<=18	COMP or COMPUTATIONAL		SBF72	0
	COMP-1 or COMPUTATIONAL-1		FLOAT-36	0
9(n) or S9(n) where 1<=n<=18	COMP-3 or COMPUTATIONAL-3		PD9	n
X(n)	unspecified or DISPLAY or DISPLAY-6		SIXBIT	n
X(n)	DISPLAY-7		ASCII-7	n
X(n)	DISPLAY-9		EBCDIC	n
	INDEX		SBF36	0
	DATABASE-KEY		SBF36	0
	DBKEY		SBF36	0

LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

TOPS-10/TOPS-20 FORTRAN

DATA TYPES	COMPILER SWITCH	DIL DESCRIPTION	FIELD LENGTH
CHARACTER * M		ASCII-7	M
INTEGER		SBF36	0
INTEGER * 2		SBF36	0
INTEGER * 4		SBF36	0
INTEGER read in with: FORMAT (nA5)		ASCII-7	n * 5
LOGICAL		SBF36	0
LOGICAL * 1		SBF36	0
LOGICAL * 4		SBF36	0
LOGICAL * 2		SBF36	0
REAL		FLOAT-36	0
REAL * 4		FLOAT-36	0
REAL * 8		FLOAT-72	0
REAL * 8	/GFLOATING (Produces extended range DOUBLE PRECISION)	G-FLOAT72	0
REAL read in with: FORMAT (nA5)		ASCII-7	n * 5

LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

VMS COBOL:

Character sets as specified in the ALPHABET clause in the SPECIAL-NAMES paragraph of the ENVIRONMENT DIVISON:

APLHABET clause specification	DIL data type
ASCII	ASCII-8
STANDARD-1	ASCII-8
STANDARD-2	unsupported
NATIVE	ASCII-8
EBCDIC	EBCDIC-8
user specified literals	unsupported

PICTURE clause	USAGE clause	SIGN clause	DIL DATA TYPE	DIL FIELD LENGTH
9(n) where 1<=n<=18	unspecified or DISPLAY		DN8U (or ASCII-8)	n
S9(n) where 1<=n<=18	unspecified or DISPLAY		DN8TO (or ASCII-8)	n
S9(n) where 1<=n<=18	unspecified or DISPLAY	LEADING or LEADING OVERPUNCHED	DN8LO (or ASCII-8)	n
S9(n) where 1<=n<=18	unspecified or DISPLAY	TRAILING or TRAILING OVERPUNCHED	DN8TO (or ASCII-8)	n
S9(n)	unspecified or DISPLAY	LEADING SEPARATE	DN8LS (or ASCII-8)	n + 1
S9(n)	unspecified or DISPLAY	TRAILING SEPARATE	DN8TS (or ASCII-8)	n + 1
9(n) or S9(n) where 1<=n<=4	COMP or COMPUTATIONAL		SBF16	0
9(n) or S9(n) where 5<=n<=9	COMP or COMPUTATIONAL		SBF32	0
9(n) or S9(n) where 10<=n<=18	COMP or COMPUTATIONAL		SBF64	0
	COMP-1 or COMPUTATIONAL-1		F-FLOAT	0
	COMP-2 or COMPUTATIONAL-2		D-FLOAT	0
9(n) or S9(n) where 1<=n<=18	COMP-3 or COMPUTATIONAL-3		PD8	n
	unspecified or DISPLAY		ASCII-8	n x(n)
	INDEX		SBF32	0
	POINTER		SBF32	0

LANGUAGE-SPECIFIC VALUES FOR DIL NAMES

VMS FORTRAN

DATA TYPES	COMPILER SWITCH	DIL DESCRIPTION	FIELD LENGTH
BYTE		SBF8	0
CHARACTER * M		ASCII-8	M

NOTE

By default, VAX-11 FORTRAN passes arguments of type CHARACTER by descriptor. In order to use CHARACTER type variables as arguments to DIL (on VMS), the %REF built-in function must be used because DIL string arguments must be passed by reference.

DATA TYPES	COMPILER SWITCH	DIL DESCRIPTION	FIELD LENGTH
INTEGER		SBF32	0
INTEGER	/NOI4	SBF16	0
INTEGER * 2		SBF16	0
INTEGER * 4		SBF32	0
INTEGER read in with: FORMAT (nA4)		ASCII-8	n * 4
LOGICAL		SBF32	0
LOGICAL	/NOI4	SBF16	0
LOGICAL * 1		SBF8	0
LOGICAL * 4		SBF32	0
LOGICAL * 2		SBF16	0
REAL		F-FLOAT	0
REAL * 4		F-FLOAT	0
REAL * 8		D-FLOAT	0
REAL * 8	/G-FLOATING	G-FLOAT	0
REAL * 16		H-FLOAT	0
REAL read in with: FORMAT (nA4)		ASCII-8	n * 4

APPENDIX B
DIL DATA FORMATS

APPENDIX B

DIL DATA FORMATS

This appendix describes, in bit-by-bit detail, the formats of all of the data types referred to in this document. Bit numbering is specified with the least significant bit having the lowest number.

In the diagrams for each type, the following notation is used:

d or D	A data bit
s or S	A sign bit
e or E	An exponent bit
m or M	A mantissa (fraction) bit
x or X	An unused bit
c	Characters

All diagrams show bits grouped into bytes or words with the more significant bits towards the left. They show bit numbering across the top of each diagram.

For string-based data types, the "normal" order of the characters is indicated at the bottom of the diagram. "C1" indicates the first character, "C2" the second character, etc.

B.1 ALPHANUMERIC STRING DATA TYPES

ASCII-7 -- DEC-10/20 7-bit ASCII

7-bit characters, packed 5 to a 36-bit word. The first character in a word occupies the highest-order 7 bits. The lowest-order bit is not used. ASCII representation is used.

Any byte alignment supported by the hardware byte-pointer instructions is acceptable. A string may start anywhere, but if it crosses a word boundary, the first byte in the new word must occupy the first 7 bits.

The length of the string in characters must be explicitly specified.

```
333333222222222211111111111
543210987654321098765432109876543210
-----------------------------------
ddddddDDDDDDDddddddDDDDDDDddddddX
-----------------------------------
   C1     C2     C3     C4     C5
```

ASCII-8 -- PDP-11/VAX 8-bit ASCII

Each character occupies one 8-bit byte. The first, or leftmost, byte of a string occupies the lowest-addressed byte of the storage allocated to the string. Following bytes in the string occupy sequentially higher addresses.

The length of the string in characters must be explicitly specified.

The standard 7-bit ASCII character set is used.

```
76543210
--------
xDDDDDDD
--------
```

ASCIZ -- ASCII-7 with terminating null

This is the same as ASCII-7, except that instead of specifying the length of the string, the string is terminated with a null byte at the end.

EBCDIC-8 -- 8-bit EBCDIC

Each character occupies one 8-bit byte. The first character (leftmost) in a string occupies the lowest-addressed byte of storage allocated to that string. Succeeding characters occupy sequentially higher-addressed bytes.

8-bit EBCDIC character coding is used.

The length of the string in characters must be explicitly specified.

```
76543210
--------
dddddddd
--------
```

EBCDIC-9 -- DEC-10/20 9-bit EBCDIC

Each character occupies one 9-bit byte. 4 characters are packed into each 36-bit word. The first (leftmost) character in the string occupies the high-order byte (most significant bits) of the word.

Any byte alignment supported by the hardware byte-pointer instructions is acceptable. A string may start anywhere, but if it crosses a word boundary, the first byte in the new word must occupy the first 9 bits.

8-bit EBCDIC character coding is used. The high-order bit of each byte is not used.

The length of the string in characters must be explicitly specified.

```
333333222222222211111111 1
543210987654321098765432109876543210
------------------------------------
XddddddddXddddddddXddddddddXddddddddd
------------------------------------
    C1        C2        C3        C4
```

SIXBIT -- SIXBIT Character String

Characters are packed 6 per 36-bit word. The first character in a word occupies the 6 highest-order bits. The length of the string in characters must be explicitly specified.

```
333333222222222211111111111
543210987654321098765432109876543210
-------------------------------------
ddddddDDDDDDddddddDDDDDDddddddDDDDDD
-------------------------------------

  C1     C2     C3     C4     C5     C6
```

B.2 BINARY FIXED-POINT DATA TYPES

SBF128 -- 128-bit signed binary fixed-point

128-bit signed fixed-point number represented in twos-complement notation. Higher-order bytes come at higher addresses than lower-order bytes. Significance increases from right to left within each byte.

The scale factor (number of decimal places to shift the point left) must be specified.

```
332222222222111111111111
109876543210987654321098765432109876543210
-------------------------------------------
ddddddddddddddddddddddddddddddddddddddddd    : A
-------------------------------------------

666655555555554444444444433333333
321098765432109876543210987654321098765432
-------------------------------------------
ddddddddddddddddddddddddddddddddddddddddd    : A + 4
-------------------------------------------

999999888888888877777777776666666
543210987654321098765432109876543210
-------------------------------------------
ddddddddddddddddddddddddddddddddddddddddd    : A + 8
-------------------------------------------

11111111111111111111111111111111
22222222111111111100000000009999
76543210987654321098765432109876
-------------------------------------------
sddddddddddddddddddddddddddddddddddddddddd   : A + 12
-------------------------------------------
```

SBF16 -- 16-bit signed binary fixed-point

A 16-bit signed fixed-point number represented in twos-complement notation. The range of values possible is -32768. to +32767. before scaling. The high-order byte comes after (at a higher address than) the low-order byte.

The scale factor (number of decimal places to shift the point left) must be specified.

```
111111
5432109876543210
----------------
sddddddddddddddd
----------------
```

SBF32 -- 32-bit signed binary fixed-point

A 32-bit signed fixed-point number represented in twos-complement notation. The range of values possible is -2147483648. to +2147483647. before scaling. The bytes are stored at increasing addresses in order of increasing significance.

The scale factor (number of decimal places to shift the point left) must be specified.

```
33222222222211111111111
10987654321098765432109876543210
--------------------------------
sddddddddddddddddddddddddddddddd
--------------------------------
```

SBF36 -- 36-bit signed binary fixed-point

A 36-bit signed fixed-point number represented in twos-complement notation. The range of values possible is -2^35 (-34359738368 decimal) to 2^35 - 1 (34359738367 decimal) before scaling.

The scale factor (number of decimal places to shift the point left) must be specified.

```
333333222222222211111111111
543210987654321098765432109876543210
------------------------------------
sddddddddddddddddddddddddddddddddddd
------------------------------------
```

SBF64 -- 64-bit signed binary fixed-point

A 64-bit signed fixed-point number represented in twos-complement notation. The range of values possible is -2^63 (-9223372036854775808 decimal) to (2^63)-1 (9223377036854775807 decimal) before scaling. The bytes are stored at increasing addresses in order of increasing significance.

The scale factor (number of decimal places to shift the point left) must be specified.

```
33222222222211111111111
10987654321098765432109876543210
--------------------------------
ddddddddddddddddddddddddddddddd   : A
--------------------------------

66665555555555444444444433333333
32109876543210987654321098765432
--------------------------------
sddddddddddddddddddddddddddddddd  : A + 4
--------------------------------
```

SBF72 -- 72-bit signed binary fixed-point

A 72-bit signed fixed-point number represented in twos-complement notation. The range of values possible is -2^70 (-1180591620717411303424 decimal) to (2^70)-1 (1180591620717411303423 decimal) before scaling.

The scale factor (number of decimal places to shift the point left) must be specified.

```
77666666666655555555554444444444333333
10987654321098765432109876543210987654
---------------------------------------
sddddddddddddddddddddddddddddddddddddd   : A
---------------------------------------

33333322222222221111111111
54321098765432109876543210987654321O
---------------------------------------
xddddddddddddddddddddddddddddddddddddd   : A + 1
---------------------------------------
```

SBF8 -- 8-bit signed binary fixed-point

An 8-bit signed fixed-point number represented in twos-complement notation. The range of values possible is -128. to +127. before scaling.

The scale factor (number of decimal places to shift the point left) must be specified.

```
76543210
--------
sddddddd
--------
```

UBF16 -- 16-bit unsigned binary fixed-point

A 16-bit unsigned fixed-point number represented in binary notation. The range of values possible is 0 to 65535 before scaling. The bytes are stored at increasing addresses in order of increasing significance.

The scale factor (number of decimal places to shift the point left) must be specified.

```
111111
5432109876543210
----------------
dddddddddddddddd
----------------
```

UBF32 -- 32-bit unsigned binary fixed-point

A 32-bit unsigned fixed-point number represented in binary notation. The range of values possible is 0 to 4294967295 before scaling. The bytes are stored at increasing addresses in order of increasing significance.

The scale factor (number of decimal places to shift the point left) must be specified.

```
33222222222211111111111
10987654321098765432109876543210
--------------------------------
dddddddddddddddddddddddddddddddd
--------------------------------
```

UBF8 -- 8-bit unsigned binary fixed-point

An 8-bit unsigned fixed-point number represented in binary notation. The range of values possible is 0 to 255 before scaling.

The scale factor (number of decimal places to shift the point left) must be specified.

```
76543210
--------
dddddddd
--------
```

B.3 FLOATING-POINT DATA TYPES

D-FLOAT -- 64-bit D-floating

A signed floating-point number represented in exponential notation. The magnitude of the item is in the approximate range .29 * (10^-38) to 1.7 * (10^38). The precision is approximately 16 decimal digits.

```
3322222222221111111111
1098765432109876543210987654321 0
-------------------------------------
mmmmmmmmmmmmmmmmSeeeeeeeemmmmmmm   : A
-------------------------------------
4444444433333333         5555544   Significance of mantissa bits
7654321098765432         4321098

6666555555555544444444443333333 3
3210987654321098765432109876543 2
-------------------------------------
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm   : A + 4
-------------------------------------
111111          3322222222221111   Significance of mantissa bits
5432109876543210109876543210987 6
```

The value of the data represented is <.1m> * 2^(e - 128) where <.1m> means add a one bit at the most significant end of the mantissa, and consider the resulting string of bits to fall after a binary point.

A value of 0 is represented by s = e = 0. S = 1 e = 0 is a reserved operand.

F-FLOAT -- 32-bit F-floating

A signed floating-point number represented in exponential notation. The magnitude of the item is in the approximate range .29*(10^-38) to 1.7*(10^38). The precision is approximately 7 decimal digits. The bytes are stored at increasing addresses in order of increasing significance.

```
3322222222221111111111
1098765432109876543210987654321 0
-------------------------------------
mmmmmmmmmmmmmmmmSeeeeeeeemmmmmmm
-------------------------------------
111111                   2221111   Significance of mantissa bits
5432109876543210         2109876
```

The value of the data represented is <.1m>*(2^(e-128)) where <.1m> means add a one bit at the most significant end of the mantissa, and consider the resulting string of bits to fall after a binary point.

A value of 0 is represented by s = e = 0. S = 1 e = 0 is a reserved operand.

DIL DATA FORMATS

FLOAT-36 -- 36-bit DEC-10/20 floating

A signed floating-point number represented in exponential notation. The magnitude of the item is in the approximate range .5*(2^-128) (1.47*10^-39 decimal) to (1-(2^-27)) * (2^127) (1.70*10^38 decimal). The precision is approximately 7 decimal digits.

```
333333222222222211111111111
543210987654321098765432109876543210
-------------------------------------
SeeeeeeeeMMMMMMMMMMMMMMMMMMMMMMMMMMM
-------------------------------------
         22222221111111111            Significance of mantissa bits
         654321098765432109876543210
```

If the value is positive (sign = 0), the value of the data represented is <.m>*(2^(e-128)) where <.m> means take the mantissa, and consider the resulting string of bits to fall after a binary point.

If the value is negative (sign = 1), the value of the data represented is the negative of the value you get by taking the twos-complement of the representation and proceeding as above.

FLOAT-72 -- 72-bit DEC-10/20 hardware floating (D-Float)

A signed floating-point number represented in exponential notation. The magnitude of the item is in the approximate range .5*(2^-128) (1.47*10^-39 decimal) to (1-(2^-27))*(2^127)(1.70*10^38 decimal). The precision is approximately 18 decimal digits.

```
333333222222222211111111111
543210987654321098765432109876543210
-------------------------------------
SeeeeeeeeMMMMMMMMMMMMMMMMMMMMMMMMMMM   : A
-------------------------------------
         665555555555444444444433333   Significance of mantissa bits
         109876543210987654321098765

333333222222222211111111111
543210987654321098765432109876543210
-------------------------------------
Xmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm   : A + 1
-------------------------------------
 333332222222222211111111111            Significance of mantissa bits
 432109876543210987654321098765432 10
```

If the value is positive (sign = 0), the value of the data represented is <.m>*(2^(e-128)) where <.m> means take the mantissa, and consider the resulting string of bits to fall after a binary point.

If the value is negative (sign = 1), the value of the data represented is the negative of the value you get by taking the twos-complement of the representation and proceeding as above.

G-FLOAT -- 64-bit G-floating

A signed floating-point number represented in exponential notation. The magnitude of the item is in the approximate range .56 *(10^-308) to .9 *(10^308). The precision is typically 15 decimal digits.

```
33222222222211111111111
10987654321098765432109876543210
--------------------------------
mmmmmmmmmmmmmmmmSeeeeeeeeeeemmmm    : A
--------------------------------
4444444433333333             5544   Significance of mantissa bits
7654321098765432             1098

66665555555555444444444433333333
32109876543210987654321098765432
--------------------------------
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm    : A + 4
--------------------------------
111111          3322222222221111   Significance of mantissa bits
54321098765432101098765432109876
```

The value of the data represented is <.1m> * (2^(e-1024)) where <.1m> means add a one bit at the most significant end of the mantissa, and consider the resulting string of bits to fall after a binary point.

A value of 0 is represented by s = e = 0. S = 1 e = 0 is a reserved operand.

G-FLOAT72 -- 72-bit DEC-10/20 G-floating

A signed floating-point number represented in exponential notation. The magnitude of the item is in the approximate range .5 * (2^-1024) (2.78*10^-309 decimal) to (1 -(2^-59))*(2^1023) (8.99*10^307 decimal). The precision is approximately 17 decimal digits.

```
333333222222222211111111111
543210987654321098765432109876543210
------------------------------------
SeeeeeeeeeeeMMMMMMMMMMMMMMMMMMMMMMM    : A
------------------------------------
            555555555444444444433333   Significance of mantissa bits
            876543210987654321098765

333333222222222211111111111
543210987654321098765432109876543210
------------------------------------
Xmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm    : A + 1
------------------------------------
 333332222222222111111111111           Significance of mantissa bits
 432109876543210987654321098765432210
```

If the value is positive (sign = 0), the value of the data represented is <.m>*(2^(e-1024)) where <.m> means take the mantissa, and consider the resulting string of bits to fall after a binary point.

If the value is negative (sign = 1), the value of the data represented is the negative of the value you get by taking the twos-complement of the representation and proceeding as above.

H-FLOAT -- 128-bit H-floating

A signed floating-point number represented in exponential notation. The magnitude of the item is in the approximate range .84 * (10^-4932) to .59 * (10^4932). The precision is typically 33 decimal digits.

```
33222222222211111111111
10987654321098765432109876543210
```

```
mmmmmmmmmmmmmmmmmmseeeeeeeeeeeeeee   : A
```

```
111111111111                         Significance of mantissa bits
1100000000009999
1098765432109876

6666555555555544444444433333333
3210987654321098765432109876543 2
```

```
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm   : A + 4
```

```
77777777776666669999998888888888   Significance of mantissa bits
98765432109876545432109876543210

99999988888888887777777777666666
54321098765432109876543210987654
```

```
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm   : A + 8
```

```
44444444333333336666555555555544   Significance of mantissa bits
76543210987654323210987654321098

11111111111111111111111111111111
22222222111111111100000000009999
76543210987654321098765432109876
```

```
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm   : A + 12
```

```
111111          3322222222221111   Significance of mantissa bits
54321098765432101098765432109876
```

The value of the data represented is <.lm> * (2^(e-1024)) where <.lm> means add a one bit at the most significant end of the mantissa, and consider the resulting string of bits to fall after a binary point.

A value of 0 is represented by s = e = 0. S = 1 e = 0 is a reserved operand. A value of 0 is represented by s = e = 0. S = 1 e = 0 is a reserved operand.

B.4 DISPLAY NUMERIC DATA TYPES

DN6LO - Sixbit signed leading overpunched

A string of type SIXBIT (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in SIXBIT coding) decimal digits and spaces (interpreted as zeroes), and at most one sign character. The sign character, if present, must be the first character in the field. The sign characters and their meanings are listed:

]	Field is negative, first digit is 0
J	Field is negative, first digit is 1
K	Field is negative, first digit is 2
L	Field is negative, first digit is 3
M	Field is negative, first digit is 4
N	Field is negative, first digit is 5
O	Field is negative, first digit is 6
P	Field is negative, first digit is 7
Q	Field is negative, first digit is 8
R	Field is negative, first digit is 9

The string is interpreted as a decimal number, right-justified in the field. The string may contain a maximum of 18 digits (19 characters including the separate sign).

NOTE

COBOL-10/20 will accept any character code in a DN6LO field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN6LO field invalid if it contains characters other than those listed above.

DN6LS -- Sixbit signed leading separate

A string of type SIXBIT (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The sign is represented by the character "+" for positive or "-" for negative before the first significant digit in the field.

The string should contain (in SIXBIT coding) decimal digits and spaces (interpreted as zeroes), and the sign characters.

The string is interpreted as a decimal number, right-justified in the field.

NOTE

COBOL-10/20 will accept any character code in a DN6LS field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN6LS field invalid if it contains characters other than those listed above.

DN6TO -- Sixbit signed trailing overpunched

A string of type SIXBIT (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in SIXBIT coding) decimal digits and spaces (interpreted as zeroes), and at most one sign character. The sign character, if present, must be the last character in the field. the sign characters and their meanings are listed:

]	Field is negative, last digit is 0
J	Field is negative, last digit is 1
K	Field is negative, last digit is 2
L	Field is negative, last digit is 3
M	Field is negative, last digit is 4
N	Field is negative, last digit is 5
O	Field is negative, last digit is 6
P	Field is negative, last digit is 7
Q	Field is negative, last digit is 8
R	Field is negative, last digit is 9

The string is interpreted as a decimal number, right-justified in the field. The string may contain a maximim of 18 digits (19 characters including the separate sign).

NOTE

COBOL-10/20 will accept any character code in a DN6TO field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN6TO field invalid if it contains characters other than those listed above.

DN6TS -- Sixbit signed trailing separate

A string of type SIXBIT (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in SIXBIT coding) decimal digits and spaces (interpreted as zeroes), and at most one sign character. The sign character is always present, and must be the last character in the field. A "+" indicates a positive field, a "-" a negative field.

The string is interpreted as a decimal number, right-justified in the field. The string may contain a maximum of 18 digits.

NOTE

COBOL-10/20 will accept any character code in a DN6TS field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN6TS field invalid if it contains characters other than those listed above.

DN6U -- Sixbit unsigned display numeric

A string of type SIXBIT (described above) with some restrictions, interpreted as an unsigned fixed-point number (scaling specified separately).

The string should contain (in SIXBIT coding) decimal digits and spaces (interpreted as zeroes). The string may contain a maximum of 18 characters (digits).

The string is interpreted as a decimal number, right-justified in the field.

NOTE

COBOL-10/20 will accept any character code in a DN6U field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN6U field invalid if it contains characters other than those listed above.

DN7LO -- ASCII 7-bit signed leading overpunched

A string of type ASCII-7 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in ASCII-7 coding) decimal digits and spaces (interpreted as zeroes), and at most one sign character. The sign character, if present, must be the first character in the field. The sign characters and their meanings are listed:

]	Field is negative, first digit is 0
J	Field is negative, first digit is 1
K	Field is negative, first digit is 2
L	Field is negative, first digit is 3
M	Field is negative, first digit is 4
N	Field is negative, first digit is 5
O	Field is negative, first digit is 6
P	Field is negative, first digit is 7
Q	Field is negative, first digit is 8
R	Field is negative, first digit is 9

The string is interpreted as a decimal number, right-justified in the field. The string may contain a maximum of 18 digits.

NOTE

COBOL-10/20 will accept any character code in a DN7LO field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN7LO field invalid if it contains characters other than those listed above.

DN7LS -- ASCII 7-bit signed leading separate

A string of type ASCII-7 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The sign is represented by the character "+" for positive or "-" for negative before the first significant digit in the field.

The string should contain (in ASCII-7 coding) decimal digits and spaces (interpreted as zeroes), and the sign characters. The string may contain a maximum of 18 digits (19 characters including the separate sign).

The string is interpreted as a decimal number, right-justified in the field.

NOTE

> COBOL-10/20 will accept any character code in a DN7LS field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN7LS field invalid if it contains characters other than those listed above.

DN7TO -- ASCII 7-bit signed trailing overpunched

A string of type ASCII-7 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in ASCII-7 coding) decimal digits and spaces (interpreted as zeroes), and at most one sign character. The sign character, if present, must be the last character in the field. The sign characters and their meanings are listed:

]	Field is negative, last digit is 0
J	Field is negative, last digit is 1
K	Field is negative, last digit is 2
L	Field is negative, last digit is 3
M	Field is negative, last digit is 4
N	Field is negative, last digit is 5
O	Field is negative, last digit is 6
P	Field is negative, last digit is 7
Q	Field is negative, last digit is 8
R	Field is negative, last digit is 9

The string is interpreted as a decimal number, right-justified in the field. The string may contain a maximum of 18 digits.

NOTE

> COBOL-10/20 will accept any character code in a DN7TO field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN7TO field invalid if it contains characters other than those listed above.

DN7TS -- ASCII 7-bit signed trailing separate

A string of type ASCII-7 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in ASCII-7 coding) decimal digits and spaces (interpreted as zeroes), and at most one sign character. The sign character is always present, and must be the last character in the field. A "+" indicates a positive field, a "-" a negative field.

The string is interpreted as a decimal number, right-justified in the field. The string may contain a maximum of 18 digits (19 characters including the separate sign).

NOTE

COBOL-10/20 will accept any character code in a DN7TS field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN7TS field invalid if it contains characters other than those listed above.

DN7U -- ASCII 7-bit unsigned display numeric

A string of type ASCII-7 (described above) with some restrictions, interpreted as an unsigned fixed-point number (scaling specified separately).

The string should contain (in ASCII-7 coding) decimal digits and spaces (interpreted as zeroes).

The string is interpreted as a decimal number, right-justified in the field. The string may contain a maximum of 18 digits.

NOTE

COBOL-10/20 will accept any character code in a DN7J field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN7U field invalid if it contains characters other than those listed above.

DN8LO -- ASCII 8-bit signed leading overpunched

A string of type ASCII-8 (described above) with some restrictions, interpreted as a signed (fixed-point number (scaling specified separately).

The string should contain (in ASCII-8 coding) decimal digits and at most one sign character. The sign character, if present, must be the first character in the field. The sign characters and their meanings are listed:

]	Field is negative, first digit is 0
J	Field is negative, first digit is 1
K	Field is negative, first digit is 2
L	Field is negative, first digit is 3
M	Field is negative, first digit is 4
N	Field is negative, first digit is 5
O	Field is negative, first digit is 6
P	Field is negative, first digit is 7
Q	Field is negative, first digit is 8
R	Field is negative, first digit is 9

The string is interpreted as a decimal number, right-justified in the field. The string may not contain more than 31 digits.

DN8LS -- ASCII 8-bit signed leading separate

A string of type ASCII-8 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The sign is represented by the character "+" for positive or "-" for negative before the first significant digit in the field. A sign of " " (space) is also acceptable, and is interpreted as positive.

The string should contain (in ASCII-8 coding) decimal digits only, and the sign character. The sign character must be the first character of the field. The string may not contain more than 31 digits (32 characters including the separate sign).

The string is interpreted as a decimal number, right-justified in the field.

DN8TO -- ASCII 8-bit signed trailing overpunched

A string of type ASCII-8 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in ASCII-8 coding) decimal digits and exactly one sign character. The sign character must be the last character in the field. The sign character and their meanings are:

OVERPUNCHED CHARACTERS

Sign VALUE	Digit VALUE	DIL DEFAULT CHAR	Char	Char	Char	Char
+	0	0	{	[	?	
+	1	1	A			
+	2	2	B			
+	3	3	C			
+	4	4	D			
+	5	5	E			
+	6	6	F			
+	7	7	G			
+	8	8	H			
+	9	9	I			
-	0	]	p	]	:	!
-	1	J	q			
-	2	K	r			
-	3	L	s			
-	4	M	t			
-	5	N	u			
-	6	O	v			
-	7	P	w			
-	8	Q	x			
-	9	R	y			

The string is interpreted as a decimal number, right-justified in the field. The string may not contain more than 31 digits.

DN8TS -- ASCII 8-bit signed trailing separate

A string of type ASCII-8 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in ASCII-8 coding) decimal digits and at most one sign character. The sign character is always present, and must be the last character in the field. A "+" indicates a positive field, a "-" a negative field.

The string is interpreted as a decimal number, right-justified in the field. The string may not contain more than 31 digits (32 characters including the separate sign).

DN8U -- ASCII 8-bit unsigned diplay numeric

A string of type ASCII-8 (described above) with some restrictions, interpreted as an unsigned fixed-point number (scaling specified separately).

The string should contain (in ASCII-8 coding) decimal digits only. The string may not have more than 31 digits.

The string is interpreted as a decimal number, right-justified in the field.

DN9LO -- EBCDIC 9-BIT signed leading overpunched

A string of type EBCDIC-9 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in EBCDIC-9 coding) decimal digits and spaces (interpreted as zeroes), and at most one sign character. The sign character, if present, must be the first character in the field. The sign characters and their meanings are listed:

]	Field is negative, first digit is 0
J	Field is negative, first digit is 1
K	Field is negative, first digit is 2
L	Field is negative, first digit is 3
M	Field is negative, first digit is 4
N	Field is negative, first digit is 5
O	Field is negative, first digit is 6
P	Field is negative, first digit is 7
Q	Field is negative, first digit is 8
R	Field is negative, first digit is 9

The string is interpreted as a decimal number, right-justified in the field. The string may not contain more than 18 digits.

NOTE

COBOL-10/20 will accept any character code in a DN9LO field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN9LO field invalid if it contains characters other than those listed above.

DN9LS -- EBCDIC 9-BIT signed leading separate

A string of type EBCDIC-9 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The sign is represented by the character "+" for positive or "-" for negative before the first significant digit in the field.

The string should contain (in EBCDIC-9 coding) decimal digits and spaces (interpreted as zeroes), and the sign characters. The string may not contain more than 18 digits (19 characters including the separate sign).

The string is interpreted as a decimal number, right-justified in the field.

NOTE

COBOL-10/20 will accept any character code in a DN9LS field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN9LS field invalid if it contains characters other than those listed above.

DN9TO -- EBCDIC 9-BIT signed trailing overpunched

A string of type EBCDIC-9 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in EBCDIC-9 coding) decimal digits and spaces (interpreted as zeroes), and at most one sign character. The sign character, if present, must be the last character in the field. The sign characters and their meanings are listed:

]	Field is negative, last digit is 0
J	Field is negative, last digit is 1
K	Field is negative, last digit is 2
L	Field is negative, last digit is 3
M	Field is negative, last digit is 4
N	Field is negative, last digit is 5
O	Field is negative, last digit is 6
P	Field is negative, last digit is 7
Q	Field is negative, last digit is 8
R	Field is negative, last digit is 9

The string is interpreted as a decimal number, right-justified in the field. The string may not contain more than 18 digits.

NOTE

COBOL-10/20 will accept any character code in a DN9TO field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN9TO field invalid if it contains characters other than those listed above.

DN9TS -- EBCDIC 9-BIT signed trailing separate

A string of type EBCDIC-9 (described above) with some restrictions, interpreted as a signed fixed-point number (scaling specified separately).

The string should contain (in EBCDIC-9 coding) decimal digits and spaces (interpreted as zeroes), and at most one sign character. The sign character is always present, and must be the last character in the field. A "+" indicates a positive field, a "-" a negative field.

The string is interpreted as a decimal number, right-justified in the field. The string may not contain more than 18 digits (19 characters including the separate sign).

NOTE

COBOL-10/20 will accept any character code in a DN9TS field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN9TS field invalid if it contains characters other than those listed above.

DN9U -- EBCDIC 9-BIT unsigned display numeric

A string of type EBCDIC-9 (described above) with some restrictions, interpreted as an unsigned fixed-point number (scaling specified separately).

The string should contain (in EBCDIC-9 coding) decimal digits and spaces (interpreted as zeroes). The string may not contain more than 18 digits.

The string is interpreted as a decimal number, right-justified in the field.

NOTE

COBOL-10/20 will accept any character code in a DN9U field and interpret it as some digit, or as a sign and a digit. The Data Conversion Routines will consider a DN9U field invalid if it contains characters other than those listed above.

B.5 PACKED DECIMAL DATA TYPES

PD8 -- Packed decimal 8-bit

A string of 4-bit bytes, packed two per 8-bit byte, representing a decimal number. Bytes at higher addresses hold less significant digits. In each byte, the higher-numbered bits hold the more significant digit.

```
33222222222211111111111
10987654321098765432109876543210

ddddDDDDddddDDDDddddDDDDddddDDDD

 D7  D8  D5  D6  D3  D4  D1  D2
```

The sign is stored in the low-numbered bits of the highest-addressed byte in the string. The digits are represented by unsigned numeric values of 0 through 9 in the 4-bit bytes. The sign is represented as follows:

DECIMAL VALUE	SIGN
10	Positive
11	Negative
12	Positive (preferred representation)
13	Negative (preferred representation)
14	Positive
15	Positive

The length of the string may not exceed 31 digits (32 4-bit bytes, including sign).

PD9 -- Packed decimal 9-bit

A strangely arranged string of 4-bit bytes, representing a decimal number. The string may not exceed 19 bytes (the number represented may not exceed 18 digits).

```
333333222222222211111111111
543210987654321098765432109876543210

XddddDDDDxDDDDddddXddddDDDDxDDDDdddd

  D1  D2   D3  D4   D5  D6   D7  D8
```

The sign of the field is stored in the low-order byte of the field (least significant). The field is right justified on a 9-bit byte boundary.

The digits are represented as unsigned numeric quantities in the 4-bit bytes.

The legal sign values (given as the decimal value of the 4-bit byte containing them) and their meanings are as follows:

DECIMAL VALUE	SIGN
10	Positive
11	Negative
12	Positve (preferred representation)
13	Negative (preferred representation)
14	Positive
15	"Nonprinting plus sign" -- forces absolute value

APPENDIX C
THE DIL SAMPLE APPLICATION

APPENDIX C

THE DIL SAMPLE APPLICATION

The DIL package includes a sample application. The application may be found on the installation tape.

The application is a "Job Ticket" program. This program eliminates the collection of paper job labor tickets and enables each employee to enter weekly job labor information from a terminal, either on a VMS, TOPS-10 or TOPS-20 system. The information is stored in an ISAM file on a TOPS-10/TOPS-20 system.

When prompted by the program, the user either enters a new badge number (and creates a new entry into the ISAM file) or an old badge number (and updates an existing entry in the ISAM file). A summary report, in the form of a sequential file stored on a VMS system, is written from a separate report program on the DECSYSTEM-20 or DECsystem-10.

A server program keeps a passive link open at all times, waiting for links from any other task. The server program keeps a table of information about the different links and their relative statuses at all times.

All of the programs in this application use the DIL Interface Support files.

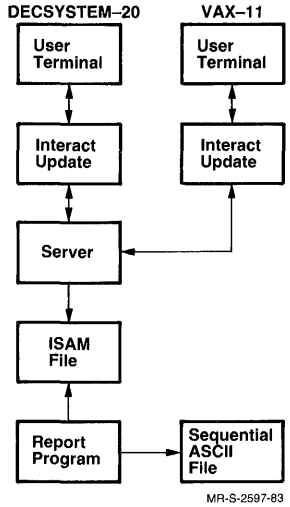


Figure C-1: Sample Application Flowchart

APPENDIX D
TASK IDENTIFICATION

APPENDIX D

TASK IDENTIFICATION

This section shows sets of compatible task identification parameters for use by the Task-to-Task Routines. Before a link can be completed, active and passive tasks must accurately identify the link. The task identification parameters issued by the active and passive tasks must have matching values. The examples, below, illustrate the three types of connections and show proper matching task identification for those connections.

NOTE

When both the passive and active tasks run on the same node, the active task can leave the hostname argument blank.

NOTE

Specifying the taskname of an active task is not meaningful on a VMS system; the VMS operating system assigns a unique taskname. It is also not generally useful on a TOPS20 system, but you may want to supply a taskname for debugging purposes. If you do specify a taskname, that name must be unique. If you do not specify a taskname, the operating system assigns a unique taskname.

D.1 ACCESSING A TASK BY NAME

passive task	active task
objectid: 0	hostname: BOSTON
descriptor: <spaces>	objectid: 0
taskname: SERVER	descriptor: SERVER
	taskname: USER

SERVER is the task name of a general TASK (object type 0) running on node BOSTON. The active task USER can communicate with SERVER by using objectid 0 (or TASK) and the unique taskname SERVER as its descriptor. SERVER must be the only TASK (objectid 0) on node BOSTON whose taskname is SERVER, since taskname must always be unique. If you plan to have several identical passive tasks running on one node, you should use a nonzero objectid (as described in the second example, below.)

D.2 ACCESSING A TASK THAT PROVIDES A CLASS OF SERVICE

passive task	active task
objectid: 128	hostname: MAINE
descriptor: <spaces>	objectid: 128
taskname: <spaces>	descriptor: <spaces>
	taskname: SOURCE

The active task SOURCE requests a connection to a passive task of object type 128 on node MAINE. There can be many passive tasks running on one node, all with object type 128. SOURCE can connect to any of the passive tasks that provide the generic service defined by object type 128.

D.3 ACCESSING SPECIFIC TASK WHICH PROVIDES SERVICE (TOPS-20 ONLY)

passive task	active task
objectid: 17	hostname: LONDON
descriptor: REB	objectid: 17
taskname: AID	descriptor: REB
	taskname: NFT

NFT is the active task; AID is the passive task, running on node LONDON. NFT indicates that it wants to communicate with AID rather than any other passive task on node LONDON which has objectid 17 by specifying REB as the descriptor; this matches the descriptor used by AID. You can only do this if both tasks are running on TOPS20 systems. The objectid is 17. An object type of 17 indicates that the passive task AID behaves as a FAL and that it provides the generic services provided by passive tasks with object type 17. (See the TOPS20 DECnet User's Guide for a list of object types).

D.4 VMS PASSIVE TASKS

VMS systems can initiate a passive task in two ways. When a VMS system receives a connect request from an active task, the system looks for an existing passive task that has the same task name (or object number) as that specified by the active task. If it finds a passive task with matching DECnet taskname, that passive task receives the connect request. This type of Task-to-Task behavior is the same as the behavior of a passive task on a TOPS-20 system; the passive task runs continuously and waits for network connections from active tasks. Section D.4.1 describes this type of VMS passive task. If the operating system does not find a task with a matching taskname, it can, if requested, use the accounting information provided by the active task to log in a new job. The new job then receives the connect request from the active task. Section D.4.2 describes this method of VMS task initiation.

D.4.1 Tasks that Wait for a Connect Request

To create a passive task that waits for connections, you must:

- Set the Network Logical Name to DIT$K_PAS_NFIREUP in your call to the DIT$NFOPP routine
- Use the DIT$NFGND routine to check for network connections
- Have SYSNAM privileges

This type of passive task can accept multiple connections.

D.4.2 Tasks Started as a Result of a Request

To create a passive task as a result of a request from an active task, you **must** specify a valid password, userid and account when the active task calls DIT$NFOPA, DIT$NFOPB or DIT$NFOP8. The operating system logs in a job using this information.

The newly-created job then runs a command file. The command file contains instructions to run the program that calls the passive task open routine, DIT$NFOPP. The command file can also be used to perform other tasks (as shown in the example, below). You must create the command file and put it in the proper directory.

The operating system determines where to look for the command file based on the object type indicated by the active task in its call to DIT$NFOPA, DIT$NFOPB or DIT$NFOP8. If the passive task specifies an object type of zero (object name TASK), put the command file in the default directory that corresponds to the indicated userid. Give the command file the name:

<taskname>.com

where <taskname> is the task name of the passive task that you want to run.

If the passive task has a non-zero object type, put the command file in the Configuration Data Base in SYS$SYSTEM. See the DECnet-VAX System Manager's Guide for information about the Configuration Data Base and SYS$SYSTEM.

To create this type of passive task, you must set the Network Logical Name to DIT$K_PAS_FIREUP in the call to DIT$NFOPP. You do not have to use the DIT$NFGND routine. The passive task starts only as a result of a request from an active task; it does not wait. You do not need SYSNAM privileges.

This type of passive task only accepts one connect request: the request that caused it to start. A VMS passive task started as a result of a connect request may, however, declare a task name or object type. By declaring a taskname, the passive task can serve other requests once it completes the request that caused it to start. You need SYSNAM privileges to create this type of passive task.

D.4.3 Example of a Task Started as a Result of a Request

An active task on another node calls DIT$NFOPA and passes the following parameters:

```
      hostname: VELDT
   object type: 0
    descriptor: IMPALA
        userid: PHONE
      password: SWORDFISH
       account: <spaces>
```

This task wants to communicate with a passive task called IMPALA on the VELDT node, a VMS system. When the operating system receives this request it checks to see whether a task named IMPALA is running on the VELDT node. It cannot find a running IMPALA, so it logs in a task using the userid, password and account provided by the active task. The operating system then searches the PHONE directory for a command file with the name IMPALA.COM. It finds a file containing the following information:

```
$ SET NOON
$ RUN IMPALA.EXE
$ PURGE/KEEP=6  SYS$LOGIN:IMPALA.LOG
$ LOGOUT/BRIEF
```

The new job starts the command file. The IMPALA.EXE program calls DIT$NFOPP and passes the following parameters:

```
        NLN  :  DIT$K_PAS_FIREUP
   objectid  :  0
  task name  :  IMPALA
```

This creates a passive task (IMPALA) to receive the connect request. The operating system sends the connect request to IMPALA. IMPALA then calls the DIT$NFACC routine to accept the connect request and establish a logical link. When it finishes processing the request, IMPALA closes the link with the DIT$NFCLS routine and exits. The job then logs out, deleting all but the latest six copies of its log file (as specified in the command file).

APPENDIX E
DIL STATUS CODES

APPENDIX E

DIL STATUS CODES

ALL POSSIBLE STATUS CODES
FOR DECsystem-10 and DECSYSTEM-20
LISTED BY DECIMAL VALUE

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
1	SS-NORMAL	Success
60948488 60948489 60948490 60948491 60948492	DIX-ROUNDED	Result is rounded
60948496 60948497 60948498 60948499 60948500	DIX-TOOBIG	Converted source field too large for destination field
60948504 60948505 60948506 60948507 60948508	DIX-INVDATTYP	Invalid data type code
60948512 60948513 60948514 60948515 60948516	DIX-UNKARGTYP	Argument passed by descriptor is unknown type
60948520 60948521 60948522 60948523 60948524	DIX-UNKSYS	Unknown system of origin specified
60948528 60948529 60948530 60948531 60948532	DIX-INVLNG	Length invalid or unspecified

DIL STATUS CODES

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
60948536 60948537 60948538 60948539 60948540	DIX-INVSCAL	Scale factor invalid or unspecified
60948544 60948545 60948546 60948547 60948548	DIX-GRAPHIC	Graphic character changed in conversion
60948552 60948553 60948554 60948555 60948556	DIX-FMTLOST	Format effector gained or lost in conversion
60948560 60948561 60948562 60948563 60948564	DIX-NONPRINT	Non-printing character gained or lost in conversion
60948568 60948569 60948570 60948571 60948572	DIX-TRUNC	String too long for destination -- truncated
60948576 60948577 60948578 60948579 60948580	DIX-UNIMP	Unimplemented conversion
60948584 60948585 60948586 60948587 60948588	DIX-INVALCHAR	Invalid character in source field or conversion table
60948592 60948593 60948594 60948595 60948596	DIX-ALIGN	Invalid alignment for data type
60948600 60948601 60948602 60948603 60948604	DIX-UNNORM	Floating point number improperly normalized

DIL STATUS CODES

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
60948608 60948609 60948610 60948611 60948612	DIX-IMPOSSIBLE	Severe internal problems
60948616 60948617 60948618 60948619 60948620	DIX-UNSIGNED	Negative value moved to unsigned field
60948624 60948625 60948626 60948627 60948628	DIX-INVBYTSIZ	Invalid byte size specified
60948632 60948633 60948634 60948635 60948636	DIX-INVDNUMCHR	Invalid display numeric character in source field.
60948640 60948641 60948642 60948643 60948644	DIX-INVDNUMSGN	Invalid display numeric sign character in source field.
60948648 60948649 60948650 60948651 60948652	DIX-INVPDDGT	Invalid packed decimal digit in source field.
60948656 60948657 60948658 60948659 60948660	DIX-INVPDSGN	Invalid packed decimal sign in source field.
61210632 61210633 61210634 61210635 61210636	DIT-HORRIBLE	Internal or system error
61210640 61210641 61210642 61210643 61210644	DIT-TOOMANY	Attempt to open too many files or links

DIL STATUS CODES

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
61210648 61210649 61210650 61210651 61210652	DIT-INVARG	Invalid argument
61210656 61210657 61210658 61210659 61210660	DIT-NETOPRFAIL	Network operation failed
61210664 61210665 61210666 61210667 61210668	DIT-CHECKSUM	Network checksum error
61210672 61210673 61210674 61210675 61210676	DIT-UNSFILETYPE	Unsupported file type
61210680 61210681 61210682 61210683 61210684	DIT-FILEINUSE	File activity precludes operation
61210688 61210689 61210690 61210691 61210692	DIT-NOFILE	File not found
61210696 61210697 61210698 61210699 61210700	DIT-EOF	End of file
61210704 61210705 61210706 61210707 61210708	DIT-OVERRUN	Data overrun
61210712 61210713 61210714 61210715 61210716	DIT-NOMOREFILES	No more files

DIL STATUS CODES

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
61211432 61211433 61211434 61211435 61211436	DIT-CONNECTEVENT	Connect event
61211440 61211441 61211442 61211443 61211444	DIT-ABREJEVENT	Abort/reject event
61211448 61211449 61211450 61211451 61211452	DIT-INTDATAEVENT	Interrupt data event
61211456 61211457 61211458 61211459 61211460	DIT-DATAEVENT	Data event
61211464 61211465 61211466 61211467 61211468	DIT-DISCONNECTEVENT	Disconnect event
61211832 61211833 61211834 61211835 61211836	DIT-ABORTREJECT	Abort/reject
61211840 61211841 61211842 61211843 61211844	DIT-INTERRUPT	Interrupt
61211848 61211849 61211850 61211851 61211852	DIT-NOTENOUGH	Not enough data available
61211856 61211857 61211858 61211859 61211860	DIT-NODATAAVAILABLE	No data available

DIL STATUS CODES

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
61211864 61211865 61211866 61211867 61211868	DIT-INFONOTAVAIL	Information not available
61211872 61211873 61211874 61211875 61211876	DIT-INFOOUTOFRANGE	Information out of range

ALL POSSIBLE STATUS CODES
FOR VMS SYSTEMS
LISTED BY DECIMAL VALUE

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
1	SS-NORMAL	Success
15237128 15237129 15237130 15237131 15237132	DIX-ROUNDED	Result is rounded
15237136 15237137 15237138 15237139 15237140	DIX-TOOBIG	Converted source field too large for destination field
15237144 15237145 15237146 15237147 15237148	DIX-INVDATTYP	Invalid data type code
15237152 15237153 15237154 15237155 15237156	DIX-UNKARGTYP	Argument passed by descriptor is unknown type
15237160 15237161 15237162 15237163 15237164	DIX-UNKSYS	Unknown system of origin specified

DIL STATUS CODES

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
15237168 15237169 15237170 15237171 15237172	DIX-INVLNG	Length invalid or unspecified
15237176 15237177 15237178 15237179 15237180	DIX-INVSCAL	Scale factor invalid or unspecified
15237184 15237185 15237186 15237187 15237188	DIX-GRAPHIC	Graphic character changed in converstion
15237192 15237193 15237194 15237195 15237196	DIX-FMTLOST	Format effector gained or lost in conversion
15237200 15237201 15237202 15237203 15237204	DIX-NONPRINT	Non-printing character gained or lost in conversion
15237208 15237209 15237210 15237211 15237212	DIX-TRUNC	String too long for destination -- truncated
15237216 15237217 15237218 15237219 15237220	DIX-UNIMP	Unimplemented conversion
15237224 15237225 15237226 15237227 15237228	DIX-INVALCHAR	Invalid character in source field or conversion table
15237232 15237233 15237234 15237235 15237236	DIX-ALIGN	Invalid alignment for data type

DIL STATUS CODES

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
15237240 15237241 15237242 15237243 15237244	DIX-UNNORM	Floating point number improperly normalized
15237248 15237249 15237250 15237251 15237252	DIX-IMPOSSIBLE	Severe internal problems
15237256 15237257 15237258 15237259 15237260	DIX-UNSIGNED	Negative value moved to unsigned field
15237264 15237265 15237266 15237267 15237268	DIX-INVBYTSIZ	Invalid byte size specified
15237272 15237273 15237274 15237275 15237276	DIX-INVDNUMCHR	Invalid display numeric character in source field
15237280 15237281 15237282 15237283 15237284	DIX-INVDNUMSGN	Invalid display numeric sign character in source field
15237288 15237289 15237290 15237291 15237292	DIX-INVPDDGT	Invalid packed decimal digit in source field
15237296 15237297 15237298 15237299 15237300	DIX-INVPDSGN	Invalid packed decimal sign in source field
15302664 15302665 15302666 15302667 15302668	DIT-HORRIBLE	Internal or system error

DIL STATUS CODES

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
15302672 15302673 15302674 15302675 15302676	DIT-TOOMANY	Attempt to open too many files or links
15302680 15302681 15302682 15302683 15302684	DIT-INVARG	Invalid argument
15302688 15302689 15302690 15302691 15302692	DIT-NETOPRFAIL	Network operation failed
15302696 15302697 15302698 15302699 15302700	DIT-CHECKSUM	Network checksum error
15302704 15302705 15302706 15302707 15302708	DIT-UNSFILETYPE	Unsupported file type
15302712 15302713 15302714 15302715 15302716	DIT-FILEINUSE	File activity precludes operation
15302720 15302721 15302722 15302723 15302724	DIT-NOFILE	File not found
15302728 15302729 15302730 15302731 15302732	DIT-EOF	End of file
15302736 15302737 15302738 15302739 15302740	DIT-OVERRUN	Data overrun

DIL STATUS CODES

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
15302744 15302745 15302746 15302747 15302748	DIT-NOMOREFILES	No more files
15303464 15303465 15303466 15303467 15303468	DIT-CONNECTEVENT	Connect event
15303472 15303473 15303474 15303475 15303476	DIT-ABREJEVENT	Abort/reject event
15303480 15303481 15303482 15303483 15303484	DIT-INTDATAEVENT	Interrupt data event
15303488 15303489 15303490 15303491 15303492	DIT-DATAEVENT	Data event
15303496 15303497 15303498 15303499 15303500	DIT-DISCONNECTEVENT	Disconnect event
15303864 15303865 15303866 15303867 15303868	DIT-ABORTREJECT	Abort/reject
15303872 15303873 15303874 15303875 15303876	DIT-INTERRUPT	Interrupt
15303880 15303881 15303882 15303883 15303884	DIT-NOTENOUGH	Not enough data available

DIL STATUS CODES

ALL POSSIBLE RETURN CODES	DIL NAME	DESCRIPTION
15303888 15303889 15303890 15303891 15303892	DIT-NOTDATAAVAILABLE	No data available
15303896 15303897 15303898 15303899 15303900	DIT-INFONOTAVAIL	Information not available
15303904 15303905 15303906 15303907 15303908	DIT-INFOOUTOFRANGE	Information out of range

APPENDIX F
BIT TRANSPORT

APPENDIX F

BIT TRANSPORT

This appendix describes how the DIL routines actually move the user data. You do not need this information if you plan to transfer ASCII data or binary fields of common data types. ASCII data appears as ASCII data on any homogeneous or heterogeneous system. Common binary data arrives at the receiving heterogeneous system in a format that can be used by the Data Conversion Routines.

The information supplied by this appendix will be useful if your data consists of bit maps, masks, or other special-purpose or user-defined data types. This type of data is beyond the scope of the Data Conversion Routines.

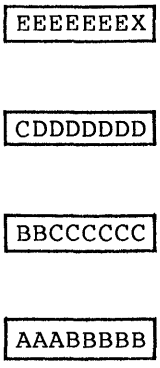
F.1 NFOPB LINKS

Data from a 36-bit system is shipped across the network link as a string of 8-bit bytes. They are sent in order, from right to left. The four high-order bits of the first word are combined to form a byte with the four low-order bits of the next word.

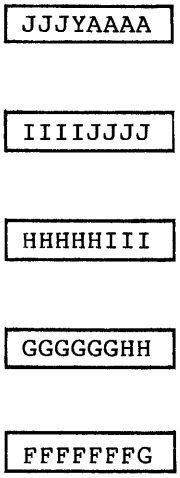
The following example shows two 36-bit words that contain ASCII data.

	A	B	C	D	E	X		F	G	H	I	J	Y
No. of bits	7	7	7	7	7	1		7	7	7	7	7	1

These words actually travel through the network in the following format:

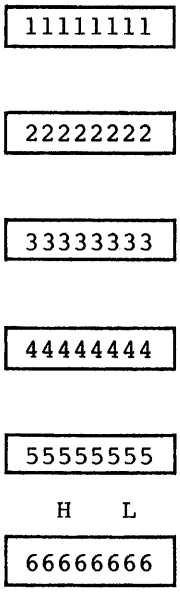


BIT TRANSPORT

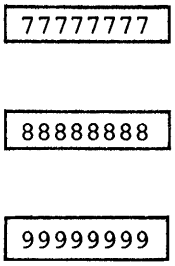


If the receiving system is also a 36-bit system, it stores the bytes in the same order as they were stored on the original system. If the receiving system is a 32-bit system, it stores the bytes in the order they were sent. The data can then be meaningfully accessed with the Data Conversion Routines.

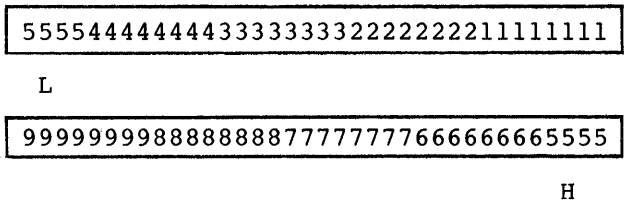
The system sends VMS data through the network as 8-bit bytes. If the following nine bytes exist on a VAX:



BIT TRANSPORT



They are sent through the network in this order. They also arrive at a 32-bit system in the same order. The bytes will arrive at a 36-bit system as the following two words:

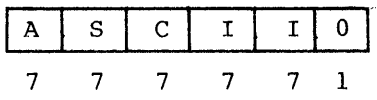


Byte 5 is split between 2 words: low-order bits in the first word and high-order bits in the second word. These 36-bit words are suitable for input to the Data Conversion Routines.

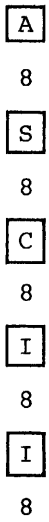
If you send data between homogeneous systems, the data arrives at the receiving system formatted in the same way as it appeared on the sending system. You do not, therefore, need to call the Data Conversion Routines to translate this data.

F.2 REMOTE FILE ACCESS IN ASCII MODE AND NFOPA LINKS

ASCII data, stored as 7-bit bytes on 36-bit systems or as 8-bit bytes on 32-bit systems, becomes a special case when sent in ASCII mode. It is always sent as 8-bit ASCII characters and stored in the format of the receiving system. This means that the following 36-bit word of ASCII data:



Is sent through the network, and arrives at a 32-bit system in the following format:



This data can then be used without calling the Data Conversion Routines, even if it was transported between heterogeneous systems.

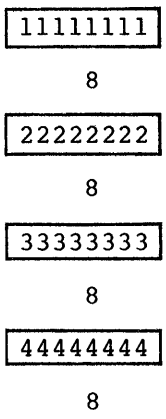
F.3 NFOP8 LINKS

Task-to-Task communication using NFOP8 links should only be used if you plan to move 8-bit bytes of non-textual data. The host system stores and accesses the bytes as it would normally store 8-bit bytes. Unused bits are not considered significant and are not sent over the network. The data will not necessarily be in a format appropriate for input to the Data Conversion Routines.

For example, the 36-bit word:

	11111111	22222222	33333333	44444444	XXXX
No. of bits:	8	8	8	8	4

would be sent through the network and received on a VAX as:



APPENDIX G
TASK–TO–TASK AND REMOTE–FILE–ACCESS ERROR MESSAGES

APPENDIX G

TASK-TO-TASK AND REMOTE-FILE-ACCESS ERROR MESSAGES

This appendix explains error messages and their causes for each Task-to-Task and Remote-File-Access routine.

G.1 REMOTE-FILE-ACCESS ERROR CODES AND THEIR MEANINGS:

G.1.1 ROPEN/DIT$ROPEN Routine:

DIT-INVARG:

1. Filename is a string but is not ASCII-7 (DECsystem-10/20 only). If you are using the routine from a COBOL program, make sure that the filename's USAGE is DISPLAY-7.

2. Userid is a string but is not ASCII-7 (DECsystem-10/20 only). If you are using the routine from a COBOL program, make sure that the userid's USAGE is DISPLAY-7.

3. Password is a string but is not ASCII-7 (DECsystem-10/20 only). If you are using the routine from a COBOL program, make sure that the password's USAGE is DISPLAY-7.

4. Account is a string but is not ASCII-7 (DECsystem-10/20 only). If you are using the routine from a COBOL program, make sure that the account's USAGE is DISPLAY-7.

5. File open mode does not have a valid value. The file must be opened for MODE-READ, MODE-WRITE, or MODE-APPEND.

6. File data type does not have a valid value. The data type must be TYPE-ASCII.

7. File record format does not have a valid value. Valid record formats are RFM-UNDEFINED, RFM-FIXED, RFM-VARIABLE, RFM-VFC, or RFM-STREAM.

8. Record attributes does not have a valid value. Valid record attributes are RAT-UNSPECIFIED, RAT-ENVELOPE, RAT-PRINT, or RAT-FORTRAN.

9. File name has improper syntax.

10. Error in directory name (VMS only).

11. Device full (VMS only).

12. Logical name error (VMS only).

13. Node name error (VMS only). The node name has improper syntax.

14. Error in quoted string (VMS only). There is no need for you to specify access information in a quoted string. If you receive this error without doing so, it is an internal error.

15. Error in file type (VMS only).

16. Error in file version number (VMS only).

SS-NORMAL:

1. Normal successful completion.

DIT-FILEINUSE:

1. File activity precludes this operation (VMS only).

DIT-TOOMANY:

1. Attempt to open more than 20 files. You are limited to 20 files open at once for version 2 of the DIL.

DIT-NOFILE:

1. File to be created already exists.

2. File not found.

3. File not available because it is locked.

4. File protection violation.

5. Error in access control string (probable internal error, VMS only).

6. Directory not found (VMS only).

7. File expiration date not yet reached (VMS only). You cannot overwrite this file because it has not expired yet.

DIT-CHECKSUM:

1. Network checksum error (probable monitor error, VMS only).

DIT-NETOPRFAIL:

1. Cannot connect to the FAL. This may be because the monitor has used up the resources it needs to make the connection, the network node you want is down or does not exist, the node is not currently running a FAL, the FAL on the node you need has aborted, there are already too many connections to the node you need, there are too many connections already to the FAL, you passed invalid accounting information (userid, password, and account) for the remote file, or because there is no network path to the node you need.

2. DAP protocol error (internal error).

3. Network link broken.

4. Operation not supported by remote system.

5. File-transfer mode precludes operation (VMS only).

6. Network operation failed (VMS only).

DIT-HORRIBLE:

1. Internal error.

2. Monitor error.

3. Hardware error.

G.1.2 RREAD/DIT$RREAD:

DIT-INVARG:

1. File number not in valid range or refers to a file which is not open.

SS-NORMAL

1. Normal successful completion.

DIT-EOF:

1. End of file.

DIT-OVERRUN:

1. Data overrun. For files on a DECsystem-10 or DECSYSTEM-20, you must add 2 to the length of a written record to determine the actual record size in the file. This is because files on the DECsystem-10 or DECSYSTEM-20 are always actually written in stream format with <CR><LF>s, and you must allow room in your buffer when reading such a file to accommodate the <CR><LF> as well as your data.

DIT-NETOPRFAIL:

1. Cannot connect to FAL. This may be because the monitor has used up the resources it needs to make the connection, the network node you want is down or does not exist, the node is not currently running a FAL, the FAL on the node you need has aborted, there are already too many connections to the node you need, there are too many connections already to the FAL, you passed invalid accounting information (userid, password, and account) for the remote file, or because there is no network path to the node you need.

2. DAP protocol error (internal error).

3. Network link broken.

4. Operation not supported by remote system.

5. File-transfer mode precludes operation (VMS only).

6. Network operation failed (VMS only).

DIT-CHECKSUM:

1. Network checksum error (probable monitor error, VMS only).

DIT-HORRIBLE:

1. Internal error.

2. Monitor error.

3. Hardware error.

G.1.3 RWRITE/DIT$RWRITE:

DIT-INVARG:

1. File number is out of range or refers to a file which is not open.

SS-NORMAL:

1. Normal successful completion.

DIT-NOFILE:

1. Privilege violation.

DIT-NETOPRFAIL:

1. Cannot connect to FAL. This may be because the monitor has used up the resources it needs to make the connection, the network node you want is down or does not exist, the node is not currently running a FAL, the FAL on the node you need has aborted, there are already too many connections to the node you need, there are too many connections already to the FAL, you passed invalid accounting information (userid, password, and account) for the remote file, or because there is no network path to the node you need.

2. DAP protocol error (internal error).

3. Network link broken.

4. Operation not supported by remote system.

5. File transfer mode precludes operation (VMS only).

6. Network operation failed (VMS only).

DIT-HORRIBLE:

1. In version 2 of the DIL, it is possible to get this error on a DECsystem-10 or DECSYSTEM-20 if you attempt to write a fixed-length file on a VMS system and the length of the record being written does not agree with the record size of the file.

2. Internal error.

3. Monitor error.

4. Hardware error.

G.1.4 RCLOSE/DIT$RCLOSE:

DIT-INVARG:

1. Invalid close option. Valid close options are OPT-NOTHING, OPT-PRINT, OPT-SUBMIT, or OPT-DELETE.

2. File number out of range or refers to a file which is not open.

DIT-NETOPRFAIL:

1. Cannot connect to FAL. This may be because the monitor has used up the resources it needs to make the connection, the network node you want is down or does not exist, the node is not currently running a FAL, the FAL on the node you need has aborted, there are already too many connections to the node you need, there are too many connections already to the FAL, you passed invalid accounting information (userid, password, and account) for the remote file, or because there is no network path to the node you need.

2. DAP protocol error (internal error).

3. Network link broken.

4. Operation not supported by remote system.

5. File transfer mode precludes operation (VMS only).

6. Network operation failed (VMS only).

DIT-HORRIBLE:

1. In version 2 of the DIL, it is possible to get this error on a DECsystem-10 or DECSYSTEM-20 if you attempt to write a fixed-length file on a VMS system and the length of the record being written does not agree with the record size of the file.

2. Internal error.

3. Monitor error.

4. Hardware error.

G.1.5 RDEL/DIT$RDEL:

DIT-INVRG:

1. File name is a string but is not ASCII-7 (DECSYSTEM-10/20 only). If you are using this routine from a COBOL program, the filename's USAGE must be DISPLAY-7.

2. Userid is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the userid's USAGE must be DISPLAY-7.

3. Password is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the password's USAGE must be DISPLAY-7.

4. Account is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the account's USAGE must be DISPLAY-7.

5. Filename has invalid syntax.

SS-NORMAL:

1. Normal successful completion.

DIT-NOFILE:

1. Invalid simultaneous access. Another user is already using this file.

2. File not found.

3. File protection violation.

4. Error in access control string (VMS only).

5. Directory not found (VMS only).

DIT-NETOPRFAIL:

1. Cannot connect to FAL. This may be because the monitor has used up the resources it needs to make the connection, the network node you want is down or does not exist, the node is not currently running a FAL, the FAL on the node you need has aborted, there are already too many connections to the node you need, there are too many connections already to the FAL, you passed invalid accounting information (userid, password, and account) for the remote file, or because there is no network path to the node you need.

2. DAP protocol error (internal error).

3. Network link broken.

4. Operation not supported on remote system.

5. File transfer mode precludes operation (VMS only).

6. Network operation failed (VMS only).

DIT-HORRIBLE:

1. Internal error.

2. Monitor error.

3. Hardware error.

G.1.6 RSUB/DIT$RUB:

DIT-INVARG:

1. File name is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the filename's USAGE must be DISPLAY-7.

2. Userid is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the userid's USAGE must be DISPLAY-7.

3. Password is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the password's USAGE must be DISPLAY-7.

4. Account is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the account's USAGE must be DISPLAY-7.

5. Filename has invalid syntax.

6. Error in directory name (VMS only).

7. Logical name error (VMS only).

8. Node name error (VMS only).

9. Error in quoted string (VMS only).

10. Error in file type (VMS only).

11. Error in version number (VMS only).

DIT-NOFILE:

1. File not found.

2. File protection violation.

3. Error in access control string (VMS only).

4. Directory not found (VMS only).

5. File locked (VMS only).

SS-NORMAL:

1. Normal successful completion.

DIT-NETOPRFAIL:

1. Cannot connect to FAL. This may be because the monitor has used up the resources it needs to make the connection, the network node you want is down or does not exist, the node is not currently running a FAL, the FAL on the node you need has aborted, there are already too many connections to the node you need, there are too many connections already to the FAL, you passed invalid accounting information (userid, password, and account) for the remote file, or because there is no network path to the node you need.

2. DAP protocol error (internal error).

3. Network link broken.

4. Operation not supported on remote system.

5. File transfer mode precludes operation (VMS only).

6. Network operation failed (VMS only).

DIT-HORRIBLE:

1. Internal error.

2. Monitor error.

3. Hardware error.

G.1.7 RPRINT/DIT$RPRINT:

DIT-INVARG:

1. Filename is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, filename's USAGE must be DISPLAY-7.

2. Userid is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, userid's USAGE must be DISPLAY-7.

3. Password is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, password's USAGE must be DISPLAY-7.

4. Account is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, account's USAGE must be DISPLAY-7.

5. Filename has invalid syntax.

6. Error in directory name (VMS only).

7. Logical name error (VMS only).

8. Node name error (VMS only).

9. Error in quoted string (VMS only).

10. Error in file type (VMS only).

11. Error in version number (VMS only).

SS-NORMAL:

1. Normal successful completion.

DIT-NOFILE:

1. File locked.

2. File not found.

3. File protection violation.

4. Error in access control string (VMS only).

5. Directory not found (VMS only).

6. File locked (VMS only).

DIT-NETOPRFAIL:

1. Cannot connect to FAL. This may be because the monitor has used up the resources it needs to make the connection, the network node you want is down or does not exist, the node is not currently running a FAL, the FAL on the node you need has aborted, there are already too many connections to the node you need, there are too many connections already to the FAL, you passed invalid accounting information (userid, password, and account) for the remote file, or because there is no network path to the node you need.

2. DAP protocol error (internal error).

3. Network link broken.

4. Operation not supported by remote system.

5. File transfer mode precludes operation (VMS only).

6. Network operation failed (VMS only).

DIT-HORRIBLE:

1. Internal error.

2. Monitor error.

3. Hardware error.

G.2 TASK-TO-TASK ERRORS AND THEIR MEANINGS:

G.2.1 NFOPA, NFOPB, NFOP8, NFOPP/DIT$NFOPA, DIT$NFOPB, DIT$NFOP8, DIT$NFOPP:

DIT-INVRG:

1. Network Logical Name is not an integer.

2. Network Logical Name has an invalid value (for DIT$NFOPP, VMS only). It must have the initial value of either PAS-FIREUP or PAS-NFIREUP on entry to this routine on a VMS system.

3. Hostname is a string but is not ASCII-7 (DECSYSTEM-10/20 only). If this routine is called from a COBOL program, hostname's USAGE must be DISPLAY-7.

4. Objectid is a string but is not ASCII-7 (DECSYSTEM-10/20 only). If this routine is called from a COBOL program, objectid's USAGE must be DISPLAY-7.

5. Descriptor is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If this routine is called from a COBOL program, descriptor's USAGE must be DISPLAY-7.

6. Taskname is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If this routine is called from a COBOL program, taskname's USAGE must be DISPLAY-7.

7. Password is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If this routine is called from a COBOL program, password's USAGE must be DISPLAY-7.

8. Account is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If this routine is called from a COBOL program, account's USAGE must be DISPLAY-7.

9. User data is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If this routine is called from a COBOL program, the user data must have USAGE DISPLAY-7.

10. Wait code is not an integer. It must have the value WAIT-YES or WAIT-NO.

11. Unknown or unreachable network node name.

12. Invalid object. You may have specified a nonexistent object name (one that is not defined for your system), or used an object number which is out of range or which you do not have the privilege to use for a passive task. See the DECnet manual for your system.

13. Invalid taskname. The taskname you gave may not be unique.

14. Object is already defined.

15. Optional data exceeds 16 bytes.

SS-NORMAL

1. Normal successful completion.

DIT-TOOMANY:

1. Attempt to open more than 20 links at once. You are limited to 20 links at any one time in DIL version 2.

2. The monitor will not allow you to open any more links. On a DECsystem-10 or DECSYSTEM-20, a nonprivileged job has a limit of four open links. The monitor itself may have run out of links.

3. No privileges to open links (VMS only).

4. Insufficient memory to open links (VMS only).

DIT-ABORTREJECT:

1. Waiting link open got an abort or reject.

DIT-HORRIBLE:

1. Internal error.

2. Monitor error.

3. Hardware error.

G.2.2 NFGND/DIT$NFGND:

DIT-INVARG:

1. Network Logical Name is not an integer, is out of range, or refers to an unused link.

2. Wait code is not an integer or has an invalid value. It must have the value WAIT-YES or WAIT-NO.

SS-NORMAL:

1. Wait code was WAIT-NO and nothing new has happened.

DIT-CONNECTEVENT:

1. Connect event occurred (connect request if link is passive; connect accept if link is active).

DIT-ABREJEVENT:

1. Abort or reject event occurred.

DIT-DISCONNECTEVENT:

1. Disconnect event occurred.

DIT-INTDATAEVENT:

1. Interrupt data available.

DIT-DATAEVENT:

1. Data available.

DIT-HORRIBLE:

1. Internal error.
2. Monitor error.
3. Hardware error.

G.2.3 NFACC/DIT$NFACC:

DIT-INVARG:

1. Network Logical Name is not an integer, is out of range, or refers to an unused link.
2. Link type is not an integer or has an invalid value. Valid link types are LTYPE-ASCII, LTYPE-BINARY, or LTYPE-8BIT.
3. Count of optional data is not an integer or has an invalid value. You may have 0 to 16 optional data characters.
4. Optional data is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the optional data must have USAGE DISPLAY-7.

SS-NORMAL:

1. Normal successful completion.

DIT-ABORTREJECT:

1. Link aborted.

DIT-HORRIBLE:

1. Internal error.
2. Monitor error.
3. Hardware error.

G.2.4 NFRCV/DIT$NFRCV:

DIT-INVARG:

1. Network Logical Name is not an integer, is out of range, or refers to an unused link.
2. Data unit size is not an integer.
3. Count of data is not an integer.
4. Message-mode flag is not an integer or has an invalid value. It must have the value MSG-STM or MSG-MSG.
5. Wait code is not an integer or has an invalid value. It must have the value WAIT-YES or WAIT-NO.

SS-NORMAL:

1. Normal successful completion.

DIT-INTERRUPT:

1. Interrupt data message must be read first.

DIT-NOTENOUGH:

1. Not enough data available to satisfy nonwaiting stream mode request (DECSYSTEM-20 only).
2. No data available now for non-waiting read.

DIT-ABORTREJECT:

1. Link aborted or disconnected.

DIT-OVERRUN:

1. Data will not fit into user buffer. For a DECSYSTEM-20 only, no data is lost. Another NFRCV call with a larger buffer may retrieve the data. On other systems, the data message has been truncated to fit in the user's buffer.

DIT-HORRIBLE:

1. Internal error.
2. Monitor error.
3. Hardware error.

G.2.5 NFSND/DIT$NFSND:

DIT-INVARG:

1. Network Logical Name not an integer, is out of range, or has an invalid value.
2. Record unit size is not an integer.
3. Data count is not an integer.
4. Message-mode flag not an integer or has an invalid value. It must have the value MSG-MSG or MSG-STM.

SS-NORMAL:

1. Normal successful completion.

DIT-ABORTREJECT:

1. Link aborted.

DIT-HORRIBLE:

1. Internal error.
2. Monitor error.
3. Hardware error.

G.2.6 NFREJ/DIT$NFREJ:

DIT-INVARG:

1. Network Logical Name not an integer, is out of range, or refers to an unused link.
2. Reject code is not an integer.
3. Count of optional data is not an integer or has an invalid value. It must be from 0 to 16.
4. Optional data is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the optional data must have USAGE DISPLAY-7.

DIT-ABORTREJECT:

1. Link aborted.

SS-NORMAL:

1. Normal successful completion.

DIT-HORRIBLE:

1. Internal error.
2. Monitor error.
3. Hardware error.

G.2.7 NFINT/DIT$NFINT:

DIT-INVARG:

1. Network Logical Name not an integer, is out of range, or refers to an unused link.
2. Count of interrupt data not an integer or has an invalid value. It must have a value between 0 and 16.
3. Interrupt data is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the interrupt data must have USAGE DISPLAY-7.

SS-NORMAL:

1. Normal successful completion.

DIT-ABORTREJECT:

1. Link aborted.

DIT-HORRIBLE:

1. Internal error.
2. Monitor error.
3. Hardware error.

DIT-INTERRUPT:

1. There is an outstanding interrupt message.

G.2.8 NFRCI/DIT$NFRCI

DIT-INVARG:

1. Network Logical Name not an integer, is out of range, or refers to an unused link.
2. Count of interrupt data characters is not an integer.
3. Interrupt data is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine in a COBOL program, the data must have USAGE DISPLAY-7.

SS-NORMAL:

1. Normal successful completion.

DIT-ABORTREJECT:

1. Link aborted.

DIT-NODATAAVAILABLE:

1. No interrupt data available now.

DIT-HORRIBLE:

1. Internal error.
2. Monitor error.
3. Hardware error.

G.2.9 NFCLS/DIT$NFCLS:

DIT-INVARG:

1. Network Logical Name not an integer, is out of range, or refers to an unused link.
2. Type of link close is not an integer.
3. Count of optional close data is not an integer or is out of range. The count of optional data must be from 0 to 16.
4. Optional data is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine from a COBOL program, the optional data must have USAGE DISPLAY-7.

SS-NORMAL:

1. Normal successful completion.

DIT-ABORTREJECT:

1. Link aborted.

DIT-HORRIBLE:

1. Internal error.
2. Monitor error.
3. Hardware error.

G.2.10 NFINF/DIT$NFINF:

DIT-INVARG:

1. Network Logical Name is not an integer, is out of range, or refers to an unused link.
2. Type of information wanted is not an integer.
3. Count of information returned is not an integer.
4. Area for returned information is a string but is not ASCII-7 (DECsystem-10 or DECSYSTEM-20 only). If you are using this routine in a COBOL program, the data area must have USAGE DISPLAY-7.

SS-NORMAL

1. Normal successful completion.

DIT-ABORTREJECT:

1. The link aborted or was rejected.

DIT-INFOOUTOFRANGE:

1. Information requested is not in the range of valid values. Valid information to request is INF-NODE, INF-OBJECT, INF-DESCF, INF-DESC, INF-USERID, INF-PASSWD, INF-ACCT, INF-OPT, INF-SEG, or INF-ABTCODE.

DIT-INFONOTAVAILABLE:

1. Information requested is not available. You cannot get it because this operating system does not supply it, you do not have the necessary privileges, or the information is not available for this type of link.

DIT-HORRIBLE:

1. Internal error.
2. Monitor error.
3. Hardware error.

INDEX

NOTE

To minimize the length of entries, several abbreviations have been used throughout this index. The following list explains these abbreviations:

Abbreviation	Meaning
DIL	Data Interchange Library
DCR	Data Conversion Routines
TTT	Task-to-Task Routines
RFA	Remote File Access Routines
FFD	Foreign Field Descriptor
NLN	Network Logical Name
(T)	TOPS-10/TOPS-20
(TC)	TOPS-10/TOPS-20 COBOL
(TF)	TOPS-10/TOPS-20 FORTRAN
(V)	VMS
(VC)	VMS COBOL
(VF)	VMS FORTRAN

The example shown below tells where to find information about storing a foreign field descriptor from TOPS-10/TOPS-20 COBOL:

```
FFD,
  storing (TC)
```

INDEX

-A-

Access, remote File
  see RFA

-B-

Bit Transport, F-1
Byte Offset
  Introduction, 2-4
  TOPS-10/20, 2-4
  VMS, 2-4

-C-

CHARACTER datatype for TOPS-10/20 FORTRAN, 6-5, 7-4
CHARACTER datatype for VMS FORTRAN, 10-6, 11-6
COBOL
  DCR (T), 5-1
  DCR (V), 9-1
  RFA (TC), 7-1
  RFA (V), 11-1
  TTT (T), 6-1
  TTT (V), 10-1
Code
  condition, 1-5
  facility, 1-5
  message, 1-5
  severity, 1-5
Condition value, 1-4
Condition value, VMS, 1-4
Control Bits, 1-5
CVGEN Routine, 5-30

-D-

Data Conversion Routines
  see DCR
Data Formats, B-1
  fixed-point, B-3
  floating-point, B-7
  string, B-1
Data Interchange Library
  see DIL
Data Names
  for COBOL (T), A-6
  for COBOL (V), A-9
  for FORTRAN (T), A-8
  for FORTRAN (V), A-11
DCR, 1-1
  Byte Offset for, 2-4
  Concepts, 2-1
  Detailed Description, 2-4
  for TOPS-10/20 COBOL, 5-1
  for TOPS-10/20 FORTRAN, 5-4
  for VMS COBOL, 9-1
  for VMS FORTRAN, 9-4
DCR (Cont.)
  Foreign Field Descriptors, 2-2
  Record Layout for, 2-4
  Single-function Routines, 2-2
  Supported Conversions, 2-1
DIL
  Data Formats, B-1
  DIL names, A-1
  Introduction, 1-1
  sample application, C-1
DIL Names
  values for, A-1
DILINI Routine, 5-3, 5-7
DIT$NFACC Routine, 10-23
DIT$NFCLS Routine, 10-32
DIT$NFGND Routine, 10-7
DIT$NFOP8 Routine, 10-18
DIT$NFOPA Routine, 10-12
DIT$NFOPB Routine, 10-15
DIT$NFOPP Routine, 10-21
DIT$NFRCV Routine, 10-26
DIT$NFSND Routine, 10-28
DIT$RCLOSE Routine, 11-12
DIT$RDEL Routine, 11-13
DIT$ROPEN Routine, 11-7
DIT$RPRINT Routine, 11-17
DIT$RREAD Routine, 11-10
DIT$SUB Routine, 11-15
DIT$WRITE Routine, 11-11
DIX$BY_DET Routine, 9-11
DIX$BY_DIX_DES Routine, 9-9
DIX$MAK_DES_DET
  passing to (VC), 9-3
  Routine, 9-7

-F-

FAL, 4-2
FFD, 2-2
  storing (TC), 5-2
  storing (TF), 5-4
  storing (VC), 9-3
  storing (VF), 9-5
File Access Listener, 4-2
File name
  format, 4-2
Foreign Field Descriptor
  see FFD
FORTRAN
  DCR (T), 5-4
  DCR (V), 9-4
  RFA (TF), 7-4
  RFA (V), 11-4
  TTT (T), 6-4
  TTT (V), 10-4

-H-

Heterogeneous Network, 1-1

Homogeneous Network, 1-1

-I-

Instructions, overlay
  TOPS-10/20, 8-1
Interface Files
  compatible, 1-6
  for DCR (TC), 5-1
  for DCR (TF), 5-4
  for DCR (VC), 9-1
  for DCR (VF), 9-4
  for RFA (TC), 7-1
  for RFA (TF), 7-4
  for RFA (VC), 11-1
  for RFA (VF), 11-4
  for TTT (TC), 6-1
  for TTT (TF), 6-4
  for TTT (VC), 10-1
  General, 1-6
  Introduction, 1-6
  native, 1-6
  specific, 1-6
  TTT (VF), 10-4

-L-

Layout, record
  Introduction, 2-4
  TOPS-10/20, 2-4
  VMS, 2-4
Like-to-like translation, 2-1
Linkage Instructions
  TOPS-10, 8-1
  TOPS-20, 8-1
  VMS, 12-1
Logical Link, 3-1

-N-

Network Connection, 3-1
Network Logical Name
  see NLN
Network, heterogeneous, 1-1
Network, homogeneous, 1-1
NFACC Routine, 6-22
NFCLS Routine, 6-32
NFGND Routine, 6-6
NFOP8 Routine, 6-17
NFOPA Routine, 6-11
NFOPB Routine, 6-14
NFOPP Routine, 6-20
NFRCV Routine, 6-25
NFSND Routine, 6-28
NLN, 3-2
  for TOPS-10 FORTRAN, 6-4
  for TOPS-20 COBOL, 6-2
  for TOPS-20 FORTRAN, 6-4
  for VMS COBOL, 10-3
  for VMS FORTRAN, 10-5

-O-

Open Link
  ASCII (T), 6-11
  ASCII (V), 10-12
  Binary (T), 6-14
  binary (V), 10-15
  8-bit (T), 6-17
  8-bit (V), 10-18
  Passive task (T), 6-20
  Passive Task (V), 10-21
Overlay Instructions
  TOPS-10/20, 8-1

-P-

PSI, status
  for TOPS-10, 6-3, 6-5, 7-3, 7-5

-R-

RCLOSE Routine, 7-11
RDEL Routine, 7-13
Record Layout
  Introduction, 2-4
  TOPS-10/20, 2-4
  VMS, 2-4
Remote File Access
  see RFA
RFA, 1-3, 7-1
  Concepts, 4-1
  file name, 4-2
  for TOPS-10 COBOL, 7-1
  for TOPS-10 FORTRAN, 7-4
  for TOPS-20 COBOL, 7-1
  for TOPS-20 FORTRAN, 7-4
  for VMS COBOL, 11-1
  for VMS FORTRAN, 11-4
  Storing fields (TC), 7-2
  Storing fields (TF), 7-4
  storing fields (VC), 11-3
  storing fields (VF), 11-6
ROPEN Routine, 7-6
Routines
  CVGEN, 5-30
  DILINI, 5-3, 5-7
  DIT$NFACC, 10-23
  DIT$NFCLS, 10-32
  DIT$NFGND, 10-7
  DIT$NFOP8, 10-18
  DIT$NFOPA, 10-12
  DIT$NFOPB, 10-15
  DIT$NFOPP, 10-21
  DIT$NFRCV, 10-26
  DIT$NFSND, 10-28
  DIT$RCLOSE, 11-12
  DIT$RDEL, 11-13
  DIT$ROPEN, 11-7
  DIT$RPRINT, 11-17
  DIT$RREAD, 11-10
  DIT$SUB, 11-15
  DIT$WRITE, 11-11
  DIX$BY_DET, 9-11

Routines (Cont.)
  DIX$BY_DIX_DES, 9-9
  DIX$MAK_DES_DET, 9-7
  NFACC, 6-22
  NFCLS, 6-32
  NFGND, 6-6
  NFOP8, 6-17
  NFOPA, 6-11
  NFOPB, 6-14
  NFOPP, 6-20
  NFRCV, 6-25
  NFSND, 6-28
  RCLOSE, 7-11
  RDEL, 7-13
  ROPEN, 7-6
  RPRINT, 7-17
  RREAD, 7-9
  RSUB, 7-15
  RWRITE, 7-10
  XCGEN, 5-10
  XCVFB, 5-13
  XCVFP, 5-14
  XCVST, 5-12
  XDESCR, 5-2, 5-8
RPRINT Routine, 7-17
RREAD Routine, 7-9
RSUB Routine, 7-15
RWRITE Routine, 7-10

-S-

Sample application, C-1
Status Code
  by decimal (T), E-1
  by decimal (V), E-1
  Condition code, 1-5
  Control Bits, 1-5
  Facility Code, 1-5
  for DCR (TC), 5-3
  for DCR (TF), 5-6
  for DCR (VC), 9-3
  for DCR (VF), 9-6
  for RFA (TC), 7-3
  for RFA (TF), 7-5
  for RFA (VC), 11-3
  for RFA (VF), 11-6
  for TTT (TC), 6-3
  for TTT (TF), 6-5
  for TTT (VC), 10-3
  for TTT (VF), 10-6
  Introduction, 1-4
  LIB$MATCH_COND, 1-6
  Message Code, 1-5
  Severity Code, 1-5
  TOPS-10/20, 1-5
  VMS, 1-4
Status PSI
  for TOPS-10, 6-3, 6-5, 7-3, 7-5

-T-

Task
  active, 3-1
  fired-up (V), D-3
  identification, D-1
  not fired-up (V), D-3
  not fired-up example, D-3, D-4
  Passive, D-2
  passive, 3-1
Task-to-Task Routines
  see TTT
TOPS-10/20
  Record Layout, 2-4
TTT, 1-2
  ASCII data (T), 6-11
  ASCII data (V), 10-12
  Binary data (T), 6-14
  binary data (V), 10-15
  8-bit data (T), 6-17
  8-bit data (V), 10-18
  Concepts, 3-1
  for TOPS-10 COBOL, 6-1
  for TOPS-10 FORTRAN, 6-4
  for TOPS-20 COBOL, 6-1
  for TOPS-20 FORTRAN, 6-4
  for VMS COBOL, 10-1
  for VMS FORTRAN, 10-4
  Logical Link, 3-1
  Network Logical Name, 3-2
  Open Passive Task (T), 6-20
  Opening a link, 3-2
  Passive Task (V), 10-21
  Passive Tasks
    (V), D-2
  storing fields (TC), 6-2
  storing fields (TF), 6-4
  storing fields (VC), 10-3
  storing fields (VF), 10-6

-V-

Value, condition, 1-4
Value, VMS condition, 1-4
VMS condition value, 1-4

-X-

XCGEN Routine, 5-10
XCVFB Routine, 5-13
XCVFP Routine, 5-14
XCVST Routine, 5-12
XDESCR Routine, 5-8
  Passing to (TC), 5-2
  Passing to (TF), 5-5

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____ Telephone _____

Street _____

City _____ State _____ Zip Code _____
or Country

-----Do Not Tear - Fold Here and Tape----------

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MRO1–2/L12
MARLBOROUGH, MA 01752

----- Do Not Tear - Fold Here and Tape----------

Cut Along Dotted Line