



Programmer Reference

10 August 2011

The information contained in this document is the latest available at the time of preparation; therefore, it may be changed without notice, and it does not represent a commitment on the part of KMSYS Worldwide, Inc. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy this software on magnetic tape, disk, or any other medium for any purpose other than stated in the terms of the agreement, or without the express written permission of KMSYS Worldwide, Inc.

©Copyright 1994-2008 by KMSYS Worldwide, Inc. All rights reserved.

This material constitutes proprietary and confidential property of KMSYS Worldwide, Inc., having substantial monetary value and is solely the property of KMSYS Worldwide, Inc. This property is disclosed to the recipient thereof in confidence only and pursuant to the terms and conditions and for the purpose set forth in written agreements by and between KMSYS Worldwide, Inc., and the recipient of this material.

If you have any comments about the software or documentation, notify KMSYS Worldwide, Inc., in writing at the following address:

KMSYS Worldwide, Inc.
P.O. Box 669695
Marietta, Georgia 30066
U.S.A.

Technical Support (770) 635-6363 - Main Number (770) 635-6350 - Fax (770) 635-6351
Q-LINK, Release 6R6, November 1999

eQuate, Host Gateway Server, I-QU PLUS-1, I-QU ReorgComposer, InfoQuest, InfoQuest Client, Q-LINK, QPlex, QPlexView, T27 eXpress IT, T27 eXpress Net, T27 eXpress Plus, T27 eXpress Pro, UTS eXpress IT, UTS eXpress Net, UTS eXpress Plus, UTS eXpress Pro and WinQ are trademarks or registered trademarks of KMSYS Worldwide, Inc. Microsoft, Windows, Visual Basic and Visual C++ are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Delphi is a trademark of Borland International. Sperry, Unisys, UTS, UNISCOPE and BIS are trademarks of Unisys Corporation. Enable is a trademark of Cypress Software, Inc. All other trademarks and registered trademarks are the property of their respective owners.

RESTRICTED RIGHTS LEGEND

If this Product is acquired by or for the U.S. Government, then it is provided with Restricted Rights. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, or subparagraphs (c)(1) and (2) of the Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, or clause 18-52.227-86(d) of the NASA Supplement to the FAR, as applicable.

Table of Contents

Chapter 1: Introduction.....	1-1
1.1 Description of Chapters.....	1-1
1.2 Additional Q-LINK Documentation	1-2
1.3 Syntax Notation.....	1-2
Chapter 2: Q-LINK Operating Environment	2-1
2.1 BIS to Q-LINK Interface.....	2-1
2.2 Invoking the Q-LINK BIS Function QLK	2-2
2.3 QLK Error Status Codes	2-3
2.4 The Q-LINK Request Stream.....	2-3
2.4.1 The #INSERT Directive	2-4
2.4.2 The #EOF Directive	2-4
2.4.3 Object Requests.....	2-4
2.4.4 Multiple Job Steps	2-4
2.5 Using the QUTIL BIS Run	2-5
2.6 QLSIM Processor: Accessing Q-LINK without BIS	2-8
2.7 Remote Q-LINK Execution	2-9
Chapter 3: General Information	3-1
3.1 Q-LINK Basic Structure.....	3-1
3.2 Commands vs. Directives.....	3-1
3.3 Command Line Format.....	3-2
3.4 File Systems Accessed by Q-LINK	3-2
Chapter 4: Q-LINK Internal Structure	4-1
4.1 Record Delivery Area (RDA)	4-1
4.1.1 Direct RDA Reference.....	4-1
4.1.2 Item Name RDA Reference	4-4
4.1.3 RDA Indexing and Subscripting	4-4
4.1.4 Variable RDA Referencing	4-5
4.2 Variable Data Storage Area (DSA).....	4-5
4.3 Object Program Area	4-6
Chapter 5: Defining Variables.....	5-1
5.1 Alphanumeric String Variables (DEFINE A)	5-1
5.2 Decimal Numeric Variables (DEFINE N)	5-1
5.3 Floating Point Numeric Variables (DEFINE FP).....	5-2
5.4 KanjiVariables (DEFINEK).....	5-2
Chapter 6: Reserved Words.....	6-1
6.1 Reserved Variables.....	6-1
6.2 Special Names.....	6-3
Chapter 7: Record Delivery Area (RDA).....	7-1
7.1 Alternate Record Area Definition (DEFINE RA)	7-1
7.2 Record Delivery Area Field Definition (DEFINE RDA)	7-3
7.3 Subscript Variable Definition (DEFINE SUB).....	7-4
Chapter 8: Control Directives	8-1
8.1 ADD	8-1
8.2 COMPILE.....	8-1

8.3 DIAG	8-2
8.4 INDEX	8-2
8.5 INIT	8-2
8.6 LISTOFF.....	8-2
8.7 LISTON.....	8-2
8.8 LOAD.....	8-2
8.9 OBJECT.....	8-2
8.10 PRINTER	8-2
8.11 RETLOG	8-3
8.12 RUN.....	8-3
8.13 SAVE	8-4
8.14 SHUTDOWN.....	8-4
8.15 Object Program Considerations	8-4
Chapter 9: General Procedural Commands	9-1
9.1 ACCEPT	9-2
9.2 BITMERGE.....	9-3
9.3 BITSPLIT.....	9-4
9.4 BREAK	9-5
9.5 CASE.....	9-6
9.6 CONNECT.....	9-7
9.7 CSF	9-8
9.8 DATE.....	9-9
9.9 DATESET	9-11
9.10 DECIMAL.....	9-16
9.11 DISCONNECT.....	9-17
9.12 DISPLAY.....	9-18
9.13 DO	9-20
9.14 DUMP	9-22
9.15 EDIT	9-23
9.16 FACERR.....	9-26
9.17 GO	9-27
9.18 IF.....	9-28
9.19 PCONTROL	9-30
9.20 ROUND	9-32
9.21 SCAN.....	9-33
9.22 SET	9-35
9.23 SHIFT	9-40
9.24 STOP	9-41
9.25 TABS	9-42
9.26 TIME.....	9-43
9.27 TRACE	9-44
9.28 TRANSFER.....	9-45
9.29 TRIMDISP.....	9-46
9.30 TRIMEDIT.....	9-47
9.31 WAIT	9-48
9.32 WILDCARD	9-49

Chapter 10: PCIOS and SFS 2200 File Interface	10-1
10.1 PCIOS/SFS File Usage and Access Modes	10-2
10.2 PCIOS/SFS File Definition (DEFINE F)	10-3
10.3 CDELETE	10-6
10.4 CLOSE	10-7
10.5 OPEN	10-8
10.6 READ	10-10
10.7 READNEXT	10-11
10.8 REWRITE	10-12
10.9 START	10-13
10.10 WRITE	10-14
10.11 Special PCIOS Status Returned	10-15
Chapter 11: SORT Interface	11-1
11.1 RELEASE	11-2
11.2 RETURN	11-3
11.3 SORT	11-4
Chapter 12: DMS 2200 Interface	12-1
12.1 Subschema Invocation (INVOKE)	12-1
12.2 INVOKE Considerations	12-3
12.3 DML Commands	12-4
12.4 ACQUIRE	12-5
12.5 CLOSE	12-7
12.6 DELETE	12-8
12.7 DEPART	12-9
12.8 DISPLAY Database Error	12-10
12.9 FETCH/FIND	12-11
12.9.1 FETCH/FIND Format-1	12-11
12.9.2 FETCH/FIND Format-2	12-12
12.9.3 FETCH/FIND Format-3	12-13
12.9.4 FETCH/FIND Format-4	12-14
12.9.5 FETCH/FIND Format-5	12-15
12.9.6 FETCH/FIND Format-6	12-16
12.9.7 FETCH/FIND Format-7	12-17
12.10 FREE	12-18
12.11 IF (DML)	12-19
12.12 IMPART	12-20
12.13 INSERT	12-21
12.14 KEEP	12-22
12.15 MODIFY	12-23
12.16 OPEN	12-24
12.17 REMOVE	12-25
12.18 SET CURRENT	12-26
12.19 SET DBDN (Database Data Name)	12-28
12.20 SET NON-FATAL (DML Errors)	12-29
12.21 STORE	12-30
12.22 SUPPRESS Clause	12-31

Chapter 13: Direct I/O Access.....	13-1
13.1 DIO (DEFINE F)	13-1
13.2 DIO	13-2
Chapter 14: RDMS 2200 Interface.....	14-1
14.1 The RDMS Command	14-2
14.2 The RDMS+ Command.....	14-4
14.3 Automated RDA Definitions.....	14-5
Chapter 15: BIS DTM Interface	15-1
15.1 Queue-alias Definition (DEFINE F)	15-1
15.2 Alternate Queue-alias Areas (DEFINE RA).....	15-3
15.3 CLOSE	15-4
15.4 OPEN	15-5
15.5 READ	15-6
15.6 WRITE	15-7
Chapter 16: Command and Keyword Abbreviations.....	16-1
Chapter 17: QINDEX Reference	17-1
17.1 Introduction	17-1
17.2 QINDEX, General Information	17-2
17.2.1 Input to QINDEX	17-2
17.2.2 Output of QINDEX	17-2
17.3 Running QINDEX	17-3
17.3.1 QINDEX Processor Options.....	17-3
17.3.2 QINDEX Directives	17-3
17.3.2.1 INVOKE	17-4
17.3.2.2 S\$PROC	17-4
17.3.2.3 RECORD.....	17-4
17.3.2.4 FILE	17-5
17.4 Building an Application Definition	17-6
17.5 A QINDEX Execution	17-7
Chapter 18: QLK External Function Errors	18-1
Chapter 19: Q-LINK Common Bank Error Conditions	19-1
Chapter 20: Q-LINK DDP Errors.....	20-1

Figures

Figure 2-1: QUTIL Main Menu	2-5
Figure 2-2: QUTIL "C" Function Input	2-6
Figure 2-3: QUTIL "R" Function Input	2-6
Figure 2-4: QUTIL "R" Function Result RID Update	2-7
Figure 2-5: QUTIL "P" Function Input	2-7
Figure 2-6: Q-LINK Remote Execution	2-10

Tables

Table 4-1: Data Types	4-2
Table 4-2: Computational Field Byte Lengths	4-3
Table 10-1: File Usage/Access Modes	10-2
Table 16-1: Keyword Abbreviation	16-1
Table 17-1: QINDEX Data Allocations	17-6

Preface

This manual contains technical information regarding the use of the Q-LINK programming language. This manual is directed to programming personnel. The manual contains information necessary to write Q-LINK programs.

For those sites who are migrating from earlier release levels of Q-LINK, please read the "Q-LINK README.TXT" file on the Documentation CD from KMSYS Worldwide prior to executing existing Q-LINK programs. Some of the differences described in that document may effect the Q-LINK operating environment at your site.

Any Q-LINK programs in object form, SAVEd with levels of Q-LINK prior to this version, must be recompiled to execute under this version. KMSYS Worldwide, Inc., provides an SSG routine called "COMPILE-RUNS" which will automatically recompile all Q-LINK programs. This routine can be found in the second Q-LINK product file, SYS\$LIB\$*QLINK-1 (default mode install).

With DMS-2200, level 9R1 or higher, the term, "DMR", is no longer used. Instead, Unisys documentation now refers to a logical data manager (LDM) for DMS running under Universal Data System (UDS) Control. Q-LINK can interface with any number of LDMs through UDS Control. However, this documentation will continue to use the term, "DMR", in a generic fashion since Q-LINK can interface with non-UDS DMS applications (e.g., 8R3) as well as DMS applications running under UDS Control.

Chapter 1: Introduction

Q-LINK is a high-level database manipulation processor. Q-LINK will process requests issued from BIS runs and return fully formatted replies to the requesting BIS user. Q-LINK gives the programmer or analyst the ability to create BIS reports with data from non-BIS sources, and to apply updates to DMS 2200 databases, standard PCIOS files, RDMS 2200 database tables, BIS reports via DTM, and/or TIP/FCSS files using simple procedural commands. All Q-LINK requests operate in a real-time environment. Q-LINK allows BIS runs to invoke DMS 2200 subschemas, begin RDMS threads, open PCIOS files, connect to TIP files, access BIS reports via DTM and process in any or all of these environments in a single Q-LINK request, using all of the security and recoverability features inherent in the native mode of each environment. Q-LINK requests imparting to DMS 2200 are treated as standard COBOL/DML run units. Thus, run unit rollback protection is operational, audit trail activities take place as specified in the schema, and any local site security, such as database procedures, are invoked normally by the DMR.

1.1 Description of Chapters

Chapter 1 contains an introduction to Q-LINK, a summary of each chapter of the Programmer Reference and the rules for its use.

Chapter 2 contains general information regarding the operating environment of Q-LINK: the BIS to Q-LINK interface, the QLK BIS run function, the QLK error status and the Q-LINK request stream.

Chapter 3 contains general information regarding the format of Q-LINK commands and directives. Also included is a description of the basic structure of Q-LINK and the file interfaces that can be accessed by Q-LINK.

Chapter 4 will describe how Q-LINK is organized internally. It also contains the rules by which data items may be referenced in a Q-LINK program.

Chapter 5 describes the types of user-defined variables that can be used within a Q-LINK program or session.

Chapter 6 lists the system defined (reserved) variables that can be referenced within a Q-LINK program.

Chapter 7 describes how to utilize the Record Delivery Area (RDA).

Chapter 8 contains all control directives other than DEFINE and INVOKE which are covered in chapters that are more appropriate to their use.

Chapter 9 contains all of the Q-LINK procedural commands except those used for file I/O that are discussed in succeeding chapters.

Chapter 10 includes the directives and commands used for defining and accessing flat files, PCIOS and SFS 2200 files.

Chapter 11 describes the internal sort interface that may be used within a Q-LINK program.

Chapter 12 contains all the directives and commands necessary to interface with DMS 2200 either through the standard multi-thread interface provided by UDS Control.

Chapter 13 describes the EXEC/FCSS/TIPDMS direct I/O (if configured) that may be used with any 2200 file structure.

Chapter 14 contains the syntax requirements to use the RDMS 2200 interface.

Chapter 15 describes the directives and commands required to use the BIS (MAPPER) DTM interface.

Chapter 16 lists all the command and keyword abbreviations that may be used through the Q-LINK environments.

Chapter 17 describes the Q-LINK utility, QINDEX, used to build and maintain a data item index file, its processor call format, options and parameters.

Chapter 18 lists the error codes, explanations of errors and possible error resolutions for all errors reported to BIS runs by the @QLK external function and the QLSIM (Q-LINK/InfoQuest only) program.

Chapter 19 lists the error codes, explanations of errors and possible error resolutions for all errors reported by QMON, Q-LINK servers and other Q-LINK utility programs.

Chapter 20 lists the error codes, explanations of errors and possible error resolutions for all errors reported by QLDDP.

1.2 Additional Q-LINK Documentation

In addition to the Programmer Reference Manual, Q-LINK manuals include:

- Q-LINK Release Differences Document
- Q-LINK Quick Reference
- Q-LINK Applications Development User Guide
- Q-LINK Installation and Operation Guide

The User Guide contains many examples illustrating Q-LINK being used for applications development, system prototyping and program checkout.

1.3 Syntax Notation

The following conventions are used throughout this manual in the description of Q-LINK commands:

- Changes to this document since its last publication are marked with a change bar (an elongated vertical bar) as shown to the right of this paragraph.
- Important notes and warnings are encased in a box as shown around this bullet.
- All words in UPPERCASE letters (not italicized) are reserved keywords and must be entered exactly as shown or in their abbreviated forms.
- Many keywords may be abbreviated. Please refer to Chapter 16, "Command and Keyword Abbreviations," for a full list of permissible abbreviations. In addition, abbreviations will be highlighted in **BLUE** throughout the syntax shown in this manual. For example, **DEFINE** may be abbreviated as "DEF".
- All italicized words (mostly in lowercase letters) are to be substituted by a user supplied name or value.

- Ellipsis (...) implies allowable, but omitted, repetitions in the published syntax. Please note that the ellipsis is **not** allowed in the command or directive when parsed. In the following example, multiple set names may be specified:

```
set-name-1 [... set-name-n]
```

- A vertical bar (|) represents an "or" or "and/or" operator.
- Selections appearing within brackets, "[]", are lists of optional items of which one may be selected. In the following example, neither A nor B is required, but either one or the other may be selected:

```
[A | B]
```

- An underlined word in optional brackets represents the default value when not entered. For example, in the syntax, "[RECORDS | CHARACTERS]," RECORDS is the default and would be assumed if omitted.
- Selections appearing within braces, "{ }", are lists of items of which one and only one must be selected. In the following example, one of either C, D or E must be selected:

```
{C | D | E}
```

- Selections appearing within double vertical bars, "||", are lists of items of which one or more must be selected. The items between vertical bars are referred to as permutations and may be selected in any order. In the following example, one or more of F, G, and/or H must be selected:

```
||F | G | H||
```

To use the SUPPRESS clause as an example:

```
SUPPRESS {ALL | ;
          || AREA | ;
          RECORD | ;
          {SET | ;
          set-name-1 [... set-name-n] } || }
```

Either of the following would be correct:

```
. . . SUPPRESS AREA RECORD
. . . SUPPRESS RECORD AREA
```

The following example uses several of the above command notation rules:

```
DO procedure [ {UNTIL | WHILE} conditional-expression ]
```

In this example, the word DO is the command keyword and must be entered exactly as shown. The procedure is user-defined and must always be present. The UNTIL/WHILE clause is optional as indicated by the block formed by square brackets. If the UNTIL/WHILE clause is used, either the keyword WHILE or UNTIL (not both) must be entered as indicated by the block formed by braces, and a valid *conditional-expression* must follow. The following examples are possible forms of this command syntax:

```
DO TOTAL-ROUTINE . (This is the minimal form)
DO TOTAL-ROUTINE UNTIL CNT NOT = 1
DO TOTAL-ROUTINE WHILE CNT < LIST-RESULT
```


Chapter 2: Q-LINK Operating Environment

2.1 BIS to Q-LINK Interface

Q-LINK operates as a stand-alone run unit serving as a request processor to BIS. The process can be thought of as a "requester" to "server" relationship: BIS is the requester, and Q-LINK is the server. A Q-LINK request is created in a BIS report. The request may consist of a Q-LINK program and input data, or the name of a compiled Q-LINK program along with the required input data. In either case, the request is passed to a waiting Q-LINK server run unit for processing. Output from the Q-LINK program is returned to BIS as a result RID. A Q-LINK program is a user-developed procedure written using the Q-LINK procedural language. The Q-LINK operating environment consists of three major components: an external BIS function, QLK; the server module (Q-LINK Processor); and QMON. QLK is a KMSYS Worldwide written BIS run function, which communicates with the Q-LINK server modules. When called from a BIS run, QLK will read the RID containing the Q-LINK request and pass input (the request) to the server module. The server module is the heart of the Q-LINK system. This component will load and process the specified Q-LINK program. The user-coded Q-LINK program can pass data back to BIS, read and/or update DMS 2200 database files, PCIOS files, TIP/FCSS files and other file systems. Data returned to BIS is written to a result RID by QLK.

QMON is an optional monitor program providing an operator interface to the Q-LINK system. QMON can be run from the system console or a demand terminal.

The communication between BIS and Q-LINK is accomplished using standard features of the 2200 operating system. The line of communication established between QLK and the server module is a two-way path. Once the session between these two Q-LINK components is established, input data (the request) and output data (the reply) can pass between the BIS and non-BIS environments as required by the user-written Q-LINK program. There is no physical I/O involved in the interface between BIS and Q-LINK, and the user-coded Q-LINK program is not aware of the communication protocol.

Multiple Q-LINK server modules may be active simultaneously. Each server belongs to a server class, and multiple server modules may be active in any class. The QLK function call in the BIS run will specify a server class name to be used in processing a particular request. Q-LINK will locate the first idle server belonging to the specified class and assign the BIS request to this server. For the duration of the request, the assigned server module is "owned" by the requesting BIS run. At the completion of the request, the server is marked as available, and is placed in an idle state. If all servers in a class are busy when a request is received, the incoming request is queued for the specified class. The first server in the requested class to become idle will immediately have the queued request routed to it for processing. An idle server module does not execute any code whatsoever. The activity is completely idle, waiting to be activated by a requesting BIS run.

2.2 Invoking the Q-LINK BIS Function QLK

Interaction with a Q-LINK server from a BIS run is very simple. The run invokes the QLK function as follows:

```
@QLK,m1,t1,r1,m2,t2,server-class[,error-label],[maximum-lines]\
      ,[estimated-lines],[password],[headers-switch],[server-run-id | .system-name] ]
```

Where:

<i>m1,t1,r1</i>	Input mode, type, RID of the request stream (the input to the Q-LINK server).
<i>m2,t2</i>	Mode and type of the created result (the output of the Q-LINK server).
<i>server-class</i>	Server class designated to process the Q-LINK request stream.
<i>error-label</i>	Run label where control is returned if an error condition is encountered.
<i>maximum-lines</i>	Maximum number of reply lines to accept back into BIS. If the <i>maximum-lines</i> parameter is not given, the default value is taken from the RID type. The number of output lines is the lesser of the maximum lines (from the RID type) and the maximum lines parameter from Q-LINK class configuration.
<i>estimated-lines</i>	Estimated number of lines to be returned to BIS.
<i>password</i>	Password required to bypass the system specified maximum result lines.
<i>headers-switch</i>	This optional parameter may contain either a "Y" or an "N". Headers associated with the reply's form type will automatically be inserted into the result by the QLK function by default. This action may be suppressed by using the "Y" option here, thus allowing the request to produce its own header lines.
<i>server-run-id</i>	This optional parameter is used to direct the QLK function to access one particular server within a server class. The server's actual EXEC run-id or TIP unique run-id must be used.
<i>system-name</i>	If a <i>system-name</i> is specified it must be prefaced by a period (.). <i>System-names</i> are limited to five characters. Q-LINK will connect to the specified system using the DDP-PPC interface and the QLDDP processor, transfer the Q-LINK request for execution, and receive the result. Valid system-names are defined in the TELCON parameter file and have the label REMOTE.

Parameters *m1*, *t1*, *r1*, *m2*, *t2* and *server-class* are required parameters when calling QLK. The remaining parameters on the function call are optional.

The *err-label* parameter is supplied only if control is to be returned to a run line other than the line following the QLK function call if an error is encountered.

The *maximum-lines* parameter is used only if the requesting run must limit the number of lines in the result RID. If this parameter is non-zero, and if the number of result lines reaches this number, the Q-LINK program is terminated and control is returned to the requesting run. If an error label was specified, control will return at the label. This parameter does not override the Q-LINK maximum number of result lines specified for the server class. If a request is terminated due to a maximum result lines error, the result will contain those lines written before the maximum line error was encountered.

The *est-lines* parameter should be used only if the expected size of the result is greater than 500 lines. However, this parameter is not required for results greater than 500 lines. For large results, specifying this parameter will improve the efficiency of QLK in creating the result.

The *password* parameter will cause QLK to ignore the Q-LINK system-wide maximum result lines. When Q-LINK is installed, the system maximum lines is set. All requests are monitored by Q-LINK to ensure the result RID does not exceed the maximum number of lines. If a request creates a result with this number of lines, the request is terminated and control is returned to the error label. The result RID will contain only the system-maximum number of lines. The correct password will allow a request to create a result with no Q-LINK applied restrictions.

2.3 QLK Error Status Codes

The reserved BIS words STAT1, STAT2 and STAT3 are used by Q-LINK. The QLK function will always set STAT2 to the number of lines in the created result RID. If an error condition is encountered, STAT1 will contain the QLK error number (these errors may be found in Chapter 18, "QLK External Function Errors", and Chapter 19, "Q-LINK Common Bank Error Conditions"). STAT3 will contain any user status code returned via a STOP command (see page 9-46 for details).

Errors detected by the Q-LINK server will be sent back via the reply report. If an error was encountered, identifying information will be placed in an error line at the end of the reply. The error line will have the format:

```
.$ERRORS,file.element/version,SERVER:class/index/run-id  
.$PC = program-counter
```

If the Q-LINK request issued a "RUN *program-name* FROM *filename*", the *program-name* will be returned in the *element/version* portion of the error line and *filename* will be in the file portion. The server that processed the request will be identified by *class*, *index* and *run-id*.

In the case of a run time error (general error type 4), one more line will be added to the reply. It will start with ".\$PC = ", followed by the program-counter for the command at which the error occurred.

To find and diagnose errors returned from the Q-LINK server, the server's diagnostic log file should be retrieved and examined. The log file can be retrieved by specifying the "L" function of the QUTIL BIS run (see Section 2.5, "Using the QUTIL BIS Run").

2.4 The Q-LINK Request Stream

The Q-LINK request may reside in any BIS form type having lines of at least 80 characters in length. This is the input stream to Q-LINK and will be referred to as the request stream. The QLK run function will insert the requesting user-id, department number and station number at the beginning of the request stream when the request is passed to the server module. The request RID will contain the Q-LINK commands and directives required to process the request. The request stream will generally specify the Q-LINK program name to be run, followed by the input data required by the program.

Alternately, the request can supply the actual Q-LINK program source code. In the latter case, any required input data would follow the Q-LINK program. Input data is limited to 256 characters per line.

2.4.1 The #INSERT Directive

Input data to be processed by the Q-LINK server may be provided within the request stream RID; however, large amounts of data may be dynamically inserted into the stream via the #INSERT directive. The #INSERT directive has the following format:

```
#INSERT mode,type,RID[[start-line],[line-quantity] ]
```

The #INSERT directive is seen and processed only by the QLK run function. When encountered in the request stream, QLK will automatically insert the RID specified. The specified RID may contain a #INSERT directive. If one is found within an inserted RID, QLK will stop reading the current RID and begin reading the new RID (the first RID will be considered complete). There may be up to 10 levels of inserted RIDs. Note: This is not true nesting. The requests are simply "chained" together. The base request stream may contain any number of #INSERT directives.

The type must be a B-type through I-type cabinet.

For Q-LINK programs using the DDP-PPC interface and the QLDDP processor, any #INSERT directives will be satisfied from data on the requester system; not from the executor system. Control directives (RUN, SAVE, etc.), however, will be processed on the executor system.

2.4.2 The #EOF Directive

The #EOF directive provides a means of detecting the end of portions of the request stream in the Q-LINK Server. The #EOF is sensed by the ACCEPT command when it is used with an "AT END" clause. This is called a soft EOF because the request stream may still have more lines to be read. Many #EOF images may be used in the request stream to segment various portions of the input data. A hard end of file will be generated at the end of the request stream.

2.4.3 Object Requests

Before a Q-LINK program may be run by name, it must first be passed to the server module to be compiled and saved. Once saved, the source program need not be passed to Q-LINK in order to be executed. The request stream simply names the program to be loaded and executed. If a Q-LINK program is to be compiled and saved, it is generally not executed by the server during the compile request. A request to compile and save a Q-LINK program consisting of 50 Q-LINK commands would require only seconds from the time the QLK function was called until control was returned to the requesting BIS run. Actual compile time for a given program will be dependent on such factors as the size of the DMS 2200 subschema being invoked, and the number of Q-LINK commands. On subsequent executions of this program, the compile phase can be skipped and procedure execution will begin immediately. Any number of Q-LINK programs may be compiled and saved. See the Control Directives COMPILE, SAVE, RUN and LOAD for details on object program processing.

2.4.4 Multiple Job Steps

A single Q-LINK request may contain multiple job steps. A job step can be defined as one execution of a Q-LINK program. The job step is initiated when the RUN directive is issued. For a new request, the contents of the data storage area (see "Record Delivery Area" in Chapter 4) will be initialized; however, for a request containing more than one job step (multiple RUN directives), the contents of the RDA will be preserved between job steps. If it is desirable to initialize the RDA between steps, the INIT directive may be used.

2.5 Using the QUTIL BIS Run

Since Q-LINK requests must be sent to Q-LINK servers via a BIS run function, a mechanism is needed by which a request can be created, sent to Q-LINK, and the reply displayed without having to write a new BIS run each time. The QUTIL BIS run has been furnished for this purpose. QUTIL is a general-purpose run that automatically creates certain Q-LINK requests based on menu selections, allows the creation of a short request on the screen, or sends a request created earlier in a permanent RID. QUTIL is particularly useful as a testing tool when developing or altering Q-LINK requests.

To run QUTIL, it is necessary to know the name of a Q-LINK server that will process the request.

To invoke the QUTIL run, enter "QUTIL" while signed on to BIS and transmit. The main menu (see page 2-7) requires a server class and function be entered. The Q-LINK requests automatically created from this screen and requiring no additional input are TEST, OBJECT DUMP, STATUS INFORMATION, ERROR EXPLANATIONS and RETRIEVE LOG. The TEST function is used to see if a server is available. It simply echoes the date and time (and gives some idea of response time). The RETRIEVE LOG function causes the Q-LINK server's diagnostic log to be retrieved into the result RID. If the server is in diagnostic mode, all source program commands and error diagnostics will be displayed on this log, making it very useful for debugging. A server set up for production may have logging suppressed for all but run time errors, while a server set up for test and development may automatically append the diagnostic log to the end of all replies. Runtime status and configuration information for all Q-LINK servers configured may be retrieved with the "?" function. This function will show which server classes are configured, currently active, pending, queued, etc. All information available using the demand QMON program will be returned in the result.

If the "C" function is selected (shown below), QUTIL will display a screen containing 16 blank lines which may be used to enter a short Q-LINK request. This is a convenient way of creating short Q-LINK requests without having to modify any permanent RIDs. When you XMIT, the contents of the screen will be sent to the specified Q-LINK server class.

```

=====
                        Q U T I L
          The Q-LINK Developer's Utility Run
          (For use with Q-LINK Release 6)
          (c) Copyright 1985-1992 by KMSystems, Inc. All rights reserved.
=====
Send Request to Server Class: test1
      Function Code: C (from choices below)
      Password: [ ] (for restricted functions)
      Server's EXEC run id: [ ] (for routing to a specific server)
Change to default Mode: 0 Type: A (for Q-LINK reply result)
Suppress RID 0 headers: [ ] (Y/N - default is Y)
=====
Functions: L = Retrieve Server's diagnostic log file.
          E = Display QLK function error status code explanations.
          O = Take an object dump of server to server log file.
          T = Test server (echo data/time).
          C = Send request (build or update up to 16 lines on screen).
          R = Send request using a temporary result of any size.
          P = Send request (from an existing report).
          ? = Retrieve Q-LINK configuration and status information.
=====

```

Figure 2-1: QUTIL Main Menu

If syntax errors occur on the Q-LINK commands, an error screen will be displayed (status 6). By pressing F1, the QUTIL run's main menu will be displayed. Then, by choosing the "C" function a second time, the original request will be displayed where the commands in error can be corrected.

```

Line#| Enter Q-LINK request below:
  1 |
  2 |
  3 |
  4 |
  5 | display ">>>> This is an example of the 16-line c-function <<<<"
  6 |
  7 |
  8 | run█
  9 |
 10 |
 11 |
 12 |
 13 |
 14 |
 15 |
 16 |

```

Figure 2-2: QUTIL "C" Function Input

The "R" function (shown below) works in a similar fashion to the "C" function; however, the R function produces an initial result RID where, unlike the "C" function, lines for Q-LINK commands may be inserted, duplicated or deleted with conventional BIS manual functions. If syntax errors occur or if the request does not return the desired result, the F1 key may be used to return to the QUTIL run's main menu.

```

Update this report to create your request. RSM (F1) to send.
.DATE 20 JUN 07 09:41:34      REPORT GENERATION      USER
. Your request goes here...

. The QUTIL R-function allows Q-LINK requests to be processed from
. a result RID.

. There are 21 lines available on the initial screen to contain Q-LINK
. commands and directives including the "RUN" directive.

. If more lines are required, a result RID will appear after you transmit.
. Use BIS functions to get additional lines as required.

D " .....
D ">>>> This is an example of the QUTIL result R-function <<<<\"
D " .....
RUN█

```

Figure 2-3: QUTIL "R" Function Input

Transmitting will produce the result RID. Note: Be sure to transmit with the text cursor located after the last characters on the screen that is to be included in the result. In this case, "RUN".

```

line▶ 1      fmt▶  r1▶  shft▶  hld chrs▶  hld ln▶  ▶ RESULT▶
.DATE 20 JUN 07 09:57:55 REPORT GENERATION JEFF
. Your request goes here...

. The QUTIL R-function allows Q-LINK requests to be processed from
. a result RID.

. There are 21 lines available on the initial screen to contain Q-LINK
. commands and directives including the "RUN" directive.

. If more lines are required, a result RID will appear after you transmit.
. Use BIS functions to get additional lines as required.

D "....."
D ">>>> This is an example of the QUTIL result R-function <<<<"
D "....."
RUN

```

Figure 2-4: QUTIL "R" Function Result RID Update

As with the "C" function, the original request will be displayed by choosing the "R" function a second time.

The "P" function (shown below) is used to send a request built in a permanent RID earlier. This will be useful when you create and test longer Q-LINK requests. When the "P" function is entered, QUTIL will display a screen where you may enter the mode, type and RID of the request stream you wish to send to the Q-LINK server.

```

You have requested server class: test
sending request from mode, type, RID entered below:
Enter mode:|0|
Type:|A|
RID:|3| (Q-LINK Input RID number)

```

Figure 2-5: QUTIL "P" Function Input

The “E” function may be used to display descriptions of the error codes that may be returned from the QLK run function. This will be a handy quick reference.

In all cases, QUTIL only affects result RIDs. No permanent RID will be updated unless a manual REPLACE or DUP function is entered. All replies will be results, using the current work mode and form type.

2.6 QLSIM Processor: Accessing Q-LINK without BIS

The QLSIM processor allows Q-LINK programs to be executed outside of the BIS environment. It simulates the functions performed by BIS in the Q-LINK interface. QLSIM contains all the functionality of the QLK BIS run function, except that instead of reading and writing BIS RIDs, it reads and writes program file elements.

QLSIM can be useful, especially for testing Q-LINK programs when BIS is not available, or for testing a Q-LINK installation, before installing a version of BIS with the QLK function generated into it.

QLSIM is automatically generated with Q-LINK, and will be installed with other components of the system. Its name will vary depending on the COMUS install mode specified when the product is installed. If MODE=QLINKA is used, its name will be QLASIM, but if the default mode is used, its name will be QLSIM. QLSIM is called as follows:

```
@QLSIM, options [in-RID-element],[out-RID-element],server-class ;
    [,error-label] [, max-lines] [,est-lines] [, password] [,hdr-switch] ;
    [, {server-run-id | .system-name}] [,user] [, dept] [, station]
```

All but the *server-class* parameter are optional. These parameters are the same as would be used with the QLK run function explained earlier in this chapter. The *error-label* has no effect under QLSIM — it is present only to maintain format compatibility with QLK. There are three possible options: “B,” “I” and “S.” The maximum number of characters per input or output line is 256 characters. The B-option specifies batch mode operation, which allows output display of up to 132 characters. Without the B-option, demand mode is assumed allowing only 79 character of display. The I-option indicates that input lines will follow the @QLSIM; otherwise, input images will be read from the element specified in the *in-RID-element* parameter. The S-option specifies that the output be displayed as well as written to the element specified in the *out-RID-element* parameter. If *out-RID-element* is not specified, an element named “ORID” in TPF\$ will be used. The optional user, dept and station set the reserved variables USER\$, DEPN\$ and STNUM\$ respectively and are only necessary when the Q-LINK request program references these variables.

If the *max-lines* parameter is not specified on the @QLSIM call line the default number of lines is 2000. The actual number of output lines is the lesser of the *max-lines* input on the @QLSIM processor call line and the maximum lines value configured in Q-LINK for the class. If the user wishes to use the maximum lines configured for the class, the *max-lines* parameter on the call line must be set to a value larger than that configured for the class, or must be 0.

When QLSIM is called, the first thing it does is attempt to connect to the specified server class. Once connected to a Q-LINK server, QLSIM will remain connected until it is exited. This fact is important to remember if you are using QLSIM interactively, as it will tie up a server for the entire session. It is also important to note that while the Q-LINK server is processing the request, the batch/demand session will appear to be idle while it awaits the reply from the server. Any files assigned during the batch/demand session are unknown to the Q-LINK server; therefore, if the request requires files that are not dynamically assigned by a particular file interface (i.e., DMS 2200, UDS Control, etc.), they must be assigned from within the request via the Q-LINK CSF command.

The following is a simple example of using QLSIM to test a Q-LINK program without BIS:

```
@QLSIM,IS,,TEST
INVOKE DEMOSUB IN DEMOSHEMA FILE DMS*SCHABS
DEF A PROD-ID ' '
  IMPART
  OPEN DEMO-ORD
  ACCEPT PROD-ID
  FETCH4 FIRST ORDER-LINE-REC DEMO-ORD AREA
  DO WHILE ERROR-NUM = 0
    IF RDA OL-PRODUCT = PROD-ID
      X = X + 1
    ENDIF
    FETCH4 NEXT ORDER-LINE-REC DEMO-ORD AREA
  ENDDO
  DISPLAY 'Found ' +
  TRIMDISP X +
  DISPLAY ' order line items for product ' +
  DISPLAY PROD-ID
  STOP EXIT
RUN
AJ1234
@EOF
```

In this example, server class TEST is being used to process the Q-LINK request. All input (program and data) follows the processor call. The results will be displayed and written in element TPF\$.ORID. Note that the @EOF causes Q-LINK to sense end-of-input for the request. The @EOF does not terminate the session, and QLSIM will remain linked to the Q-LINK server to process additional requests. To end the session, an @EOF X, or any other control image must be entered to force a hard end-of-file.

2.7 Remote Q-LINK Execution

If a DDP-PPC connection exists between systems, Q-LINK allows one system (the requester) to submit a Q-LINK request for execution on a different host (the executor). The executor will process the request using data files that exist on the executor system. The output of the Q-LINK program will be passed back to the requester system.

Q-LINK programs do not need any changes to execute on a different host. The executing host system name is specified by using the server-run-id field available on the @QLK function or the @QLSIM processor. A valid system name consists of a period (.) followed by up to five characters. Valid system names are defined in the TELCON parameter file with the label REMOTE.

A sample Q-LINK remote execution session proceeds in the following manner:

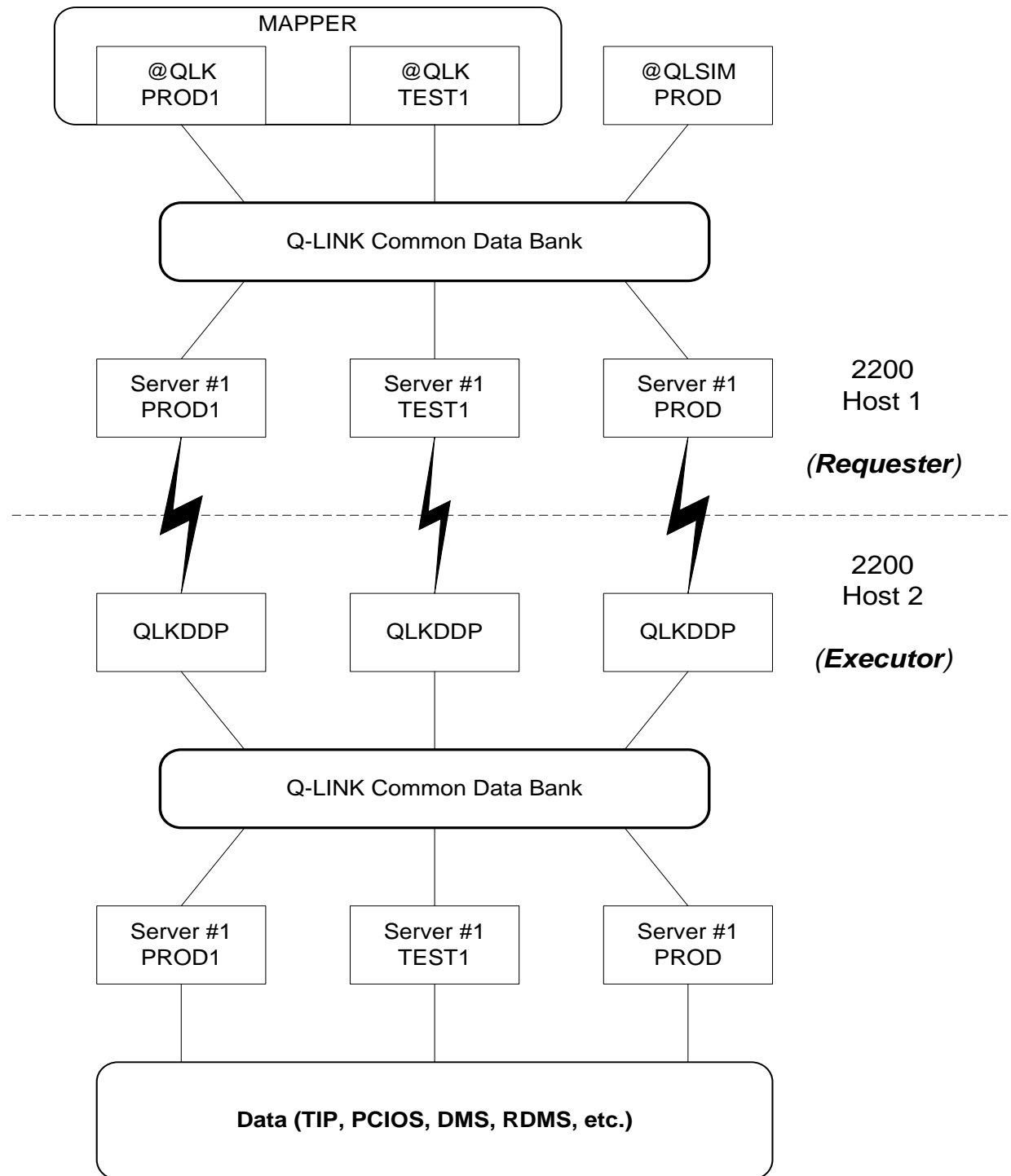


Figure 2-6: Q-LINK Remote Execution

The PROD1 server on Host 1 (requester) examines the @QLK parameters and recognizes that the submitted request is to be passed to a second host for processing. The server PROD1 connects to the DDP-PPC facility that causes the program, QLDDP, to execute on Host 2 (executor). In the example on the following page, the Q-LINK software was installed

using a "MODE=K" install in COMUS. As a result of the "MODE=K" install, the QLDDP absolute name was changed to QLKDDP. The QLKDDP program starts and connects to DDP PPC on Host 2. The Q-LINK program from Host 1 is sent (via QLKDDP) to the equivalent PROD1 server on Host 2 for execution. The result of the Q-LINK request is returned to the PROD1 server on Host 1 via QLKDDP.

For Q-LINK programs using the DDP-PPC interface, any #INSERT directives will be satisfied from data on the requester system; not from the executor system. Control directives (RUN, SAVE, etc.), however, will be processed on the executor system.

Chapter 3: General Information

3.1 Q-LINK Basic Structure

The Q-LINK processor is logically divided into two major components: the COMMAND EDITOR and the COMMAND EXECUTOR. The COMMAND EDITOR checks each command entered for correct syntax and attempts to translate it into an internally encoded object command. If an error is detected during this process, an error diagnostic is displayed and the command is rejected. If the command passes all edits, it is stored in the object program area for later execution. The COMMAND EDITOR also controls the allocation of data variables and the processing of control directives, which will be discussed later.

The COMMAND EXECUTOR interprets the encoded object command passed from the COMMAND EDITOR and actually performs the operation. The COMMAND EXECUTOR has been designed to do the least amount of interpretation possible, therefore achieving maximum throughput during program execution.

3.2 Commands vs. Directives

There are two types of control statements in Q-LINK: commands and directives.

Directives are used to establish and control the program environment of the Q-LINK program. These are always executed during the "Command Editor" phase of the Q-LINK program; i.e., immediately upon being received by Q-LINK. They are used to:

- Define files, records, buffers and fields;
- Invoke subschemas and data item index files;
- Initialize data storage areas;
- Suppress or allow the listing of commands;
- Dump Q-LINK internal areas;
- Run a program stored in the object program area or from an object library;
- Save a program in the object program area to an object library;
- Compile a program to resolve program label references.

Commands are used to perform data manipulation, program flow control, input/output, etc. They are executed during the "Command Executor" phase of the Q-LINK program.

Directives and program labels must always be entered beginning in column position one. Commands must always begin after column 1.

3.3 Command Line Format

Command formats must follow certain rules. In general, commands may be entered in free form. Restrictions are as follows:

- All directives must be entered starting in column 1.
- All procedural commands must be entered starting after column 1.
- Any word starting in column 1 will be regarded as either a control directive (if it matches one of the valid control directives) or a program or procedure label.
- A program or procedure label may be up to 30 characters in length. However, only the first 16 characters will be used by Q-LINK, and therefore must be unique.
- Only one command may be entered per line.
- A decimal numeric literal may not exceed 18 digits, and if signed, the sign must immediately precede the first digit. Example: 123, -5 or +1000.00.
- Floating-point numeric literals must be entered in standard floating point notation with a decimal point. Example: 1.E-10.
- Alphanumeric literals may not exceed 150 characters. In addition, they must be enclosed within single or double quotes. If an alphanumeric literal definition begins with a double quote, the single quote (apostrophe) character may be used in the literal, and vice versa. Additionally a string of two single (') or double (") quotes within a literal will generate one single or double quote.

Alphanumeric literal examples:

```
'THIS IS AN ALPHANUMERIC LITERAL'
"This is also 'ONE' literal."
'AND "THIS IS A LITERAL" TOO!'
'This is a single quote (') within parentheses'
```

- All command components must be entered between columns 1 through 80. A semicolon following any word within a command line will cause Q-LINK to skip to the next line to search for the next word of the command; therefore, line continuation is possible as shown in the following example:

```
IF RDA CUSTOMER-NAME ;
OF CUSTOMER = 'APEX'
```

- To continue an alphanumeric literal, end the first part with a single (or double) quote followed by a semicolon. Note: A space must not precede the semicolon. Continue the literal anywhere on the following line enclosed in quotes. For example:

```
'...PQRSTUV';
'WXYZ'
```

- A period, followed by a space, is used in a command line to separate the command from commentary text. Q-LINK will ignore any text following a period that is not embedded within a literal. Examples of comments are:

```
DO HEADER-RTN . This a Comment on a Command Line
. *** This entire line is a Comment ***
```

3.4 File Systems Accessed by Q-LINK

All DMS 2200 database record placement strategies are supported by Q-LINK. There is a Q-LINK Data Manipulation Language (DML) command corresponding to each COBOL/DML command. However, Q-LINK syntax may differ from COBOL/DML, and Q-LINK commands can be entered in an abbreviated format.

The Q-LINK Processor can be used to read, write, and update standard Processor Common Input/Output System (PCIOS) files. PCIOS files may be accessed in combination with DMS 2200 database areas and other file systems, or separately.

Q-LINK also supports access to other file types through its Direct I/O (DIO) feature. This feature provides the ability to access TIP/FCSS, TIP/DMS, or any sector-formatted EXEC mass-storage file or any tape file.

RDMS 2200 relational databases can be accessed through two Q-LINK commands that provide RDMS 2200 interface information. This information is comprised of an RDML formatted command (SQL format is not supported), RDML error status variables and application program variables (table column names, table column values, etc.). All RDML formatted commands are supported and may be used in conjunction with Q-LINK commands that interface with other file systems.

The Q-LINK processor can be used to read and write BIS reports via an interface with BIS's Data Transfer Module (DTM). Q-LINK allows "queue-aliases" to be defined in order to access more than one BIS report simultaneously through the same BIS queue. The DTM interface may be used in conjunction with any other Q-LINK file system interface.

Chapter 4: Q-LINK Internal Structure

Q-LINK internal storage is organized into three areas of interest to the user:

- The Record Delivery Area (RDA);
- The Variable Data Storage Area;
- The Object Program Area.

This chapter will describe how these areas are used.

4.1 Record Delivery Area (RDA)

The RDA is used for all input and output for DMS 2200 database records, PCIOS file records and all other file interfaces. The size of this area will vary depending upon how Q-LINK was generated. It is important to remember that all read, write, fetch, modify, etc. operations assume that the data record begin in the first position of this area. If multiple records are to be resident simultaneously, then data from the RDA can be moved to variable data storage prior to a subsequent I/O. Another way to access multiple records simultaneously is to define alternate record areas using the "DEFINE RA" directive. This directive allows the user to define alternate areas of the RDA to be used for I/O for specific records or files. See the DEFINE RA directive for more information.

Since the RDA will be generated larger than any record that may be read, the upper ranges may be used for additional workspace, and the allocation of tables, etc. When issuing the WRITE command, information may be built in an unused area of the RDA and appended to the record when it is written.

The RDA contents are initialized whenever a new Q-LINK request begins execution or if an INIT directive is processed. The RDA contents are preserved between job steps (multiple RUN directives) in the same user session (request).

The RDA may be referenced in two ways: directly by character position, or by the data item name. The following sections describe RDA referencing in detail.

4.1.1 Direct RDA Reference

When using the direct RDA reference, the starting byte (or character) position, field length, data type, and index offset or subscript are specified.

The direct RDA reference format is:

RDA (*starting-byte-position*,*length*) [*data-type*] [:*index* | :*subscript*]

The *starting-byte-position* is the offset (relative to 1) of the referenced data from the beginning of the RDA or of a record within the RDA.

The *length* is in bytes and is the actual length of the referenced data. For computational data items, this is the actual byte length of the data, not the COBOL picture length. For example, if the COBOL picture is PIC 9(10) COMP, the actual byte length is 4 (assuming the

data is ASCII). Figure 3.2 may be used to determine the byte length of various computational field definitions.

The *index* specifies a numeric literal, or a decimal integer numeric variable, immediately following a colon (:). Refer to page 3-6 for detailed information regarding RDA indexing.

The subscript specifies the name of a defined subscript variable to be used in addressing the data item. Refer to page 4-5 for detailed information on RDA subscripting.

Data within the RDA may be referenced in ASCII (quarter word, 9-bit bytes) or FIELDATA (sixth word, 6-bit bytes) form. Data representation is indicated by the *data-type* field as shown in the following table:

Data Type	Description
DISP A9 SN9 UN9 COMP SB9 UB9	ASCII alphanumeric display (default). ASCII alphanumeric display. Signed ASCII numeric display. Unsigned ASCII numeric display. Signed ASCII aligned binary. Signed ASCII aligned binary. Unsigned ASCII aligned binary.
DISP-1 A6 SN6 UN6 COMP-4 SB6 UB6	Fieldata alphanumeric display. Fieldata alphanumeric display. Signed fieldata numeric display. Unsigned fieldata numeric display. Signed fieldata aligned binary. Signed fieldata aligned binary. Unsigned fieldata aligned binary.
COMP-1 FP1 COMP-2 FP2	Single-precision floating point. Single-precision floating point. Double-precision floating point. Double-precision floating point.
DISP-2 or A18	Kanji. Note: For alignment purposes within the RDA, you must specify two bytes for each double-byte character required; therefore, if you have three Kanji double-byte characters, you would specify RDA (n,6) DISP-2.
MAPNUM	BIS numeric data format.

Table 4-1: Data Types

Please note that exact binary (i.e., PIC 1(3), etc.) is not currently supported in Q-LINK; however, exact binary is supported in QINDEX for alignment purposes only (see Chapter 17).

If using the ASCII data code, the *starting-byte-position* and *length* must be given as ASCII. If using the FIELDATA data code, locations must be given in terms of FIELDATA lengths.

The number of bytes required for signed aligned binary fields can be determined by utilizing the following table:

ASCII (COMP)		FIELDATA (COMP-4)	
Positions in Picture Clause	Record Area Bytes	Positions in Picture Clause	Record Area Bytes
1-2	1	1	1
3-5	2	2-3	2
6-7	3	4-5	3
8-10	4	6	4
11-13	5	7-8	5
14-15	6	9-10	6
16-18	7	11-12	7
		13-14	8
		15	9
		16-17	10
		18	11

Table 4-2: Computational Field Byte Lengths

COMP-1 and COMP-2 are always 4 bytes and 8 bytes respectively for ASCII alignment, and 6 bytes and 12 bytes for FIELDATA alignment (one and two words).

MAPNUM items must be set as if they are alpha items; however, they can be used to set numeric items and must be tested as numeric items (see the IF and SET commands).

Here are some examples of RDA direct references with the following data in the RDA:

```
1...5...10...15...20...25...30...35...
ABCDEFGHIJKLMN OPQRSTUVWXYZ123456789
```

1. RDA (2,3)
2. RDA (27,2) COMP
3. RDA (5,4) UB9
4. RDA (1,5) :X
5. RDA (7,6) COMP-4
6. RDA (2,9) MAPNUM

Explanations:

Example 1 would refer to the 3-character field starting in position 2 of the RDA, the value of which is "BCD" assuming the above data is ASCII.

Example 2 would refer to an ASCII computational field of two bytes starting in position 27.

Example 3 would refer to the second full word in the RDA as unsigned binary.

Example 4 would refer to positions 1 through 5 offset by the value of X. If X=5, the value would be "FGHIJ" assuming the above data is ASCII (see the subsection "RDA Indexing and Subscripting" below).

Example 5 would refer to a full word FIELDATA computational field starting in byte position 7 (word 2 of the RDA).

Example 6 refers to an RDA field in BIS numeric data format. BIS numeric data is always in edited form, which may include a leading sign, spaces and decimal point. An example of a MAPNUM data item would be '-123.58'.

4.1.2 Item Name RDA Reference

The second method of referencing the RDA uses the Q-LINK data item index file. A primary data item index is created automatically during INVOKE processing for DMS database items (see "Subschema Invocation (INVOKES)" in the "DMS 2200 Interface" chapter). A secondary data item index may also be specified via the INDEX directive for non-DMS data items and redefined DMS database items. The QINDEX Processor is used to create data item index files from COBOL definitions. Refer to Chapter 17, "QINDEX Reference," for an explanation on how to create alternate Q-LINK data item index files. To use this method of referencing the RDA, it is necessary to specify the data item name and its index offset or subscript, if any.

The format for a named RDA reference is:

```
RDA item-name [ {IN | OF} {record-name | file-name | queue-alias | SORT} ] ;
               [:index | :subscript]
```

The *item-name* must be defined in one of the data item index files currently in use (built by the INVOKE and INDEX directives) or by a previously issued DEFINE RDA directive.

The IN/OF option allows the *item-name* to be qualified to the proper database *record-name*, PCIOS *file-name*, BIS DTM *queue-alias* or SORT area.

4.1.3 RDA Indexing and Subscripting

An RDA index specifies a byte offset to be used when addressing the data. The index may be entered as a numeric literal, or a numeric variable, immediately following a colon (:).

Subscripting differs from indexing in that a subscript refers to an occurrence of an item within a table of items, while an index refers to a number of bytes from the start of the item. Subscripting in Q-LINK is similar to that used in COBOL, except that a defined subscript variable must only be used to reference the OCCURS items for which it was defined. RDA subscripting may be used on multidimensional RDA references, whereas RDA indexes may not.

The index or subscript options are used in the same way as with a direct reference in specifying a byte offset from the beginning of the field (indexing), or a specific occurrence (subscripting).

Given the following portion of a record definition:

```
05  RATE-TABLE.
    10  RATE OCCURS 10 TIMES PIC 9(5) .
05  HIGHEST-RATE                PIC 9(5) .
```

An example of indexed references follows:

To refer to the first RATE, simply enter "RATE".

To refer to the second RATE, enter "RATE :5". (RATE offset by five bytes.)

To refer to any RATE, use "RATE :RATE-INDEX", where RATE-INDEX is a numeric variable used to index to the desired occurrence of RATE.

To use subscripted reference, a subscript variable must be defined using the DEFINE SUB directive. The defined subscript variable may then be used in much the same manner as a COBOL subscript.

The following program example illustrates the necessary Q-LINK code to examine all 10 occurrences of RATE in the above record definition using RDA indexing:

```
DEF N RATE-CTR
DEF N RATE-INDEX
.....
SET RATE-CTR = 0
SET RATE-INDEX = 0
```



```

DO RATE-LOOP UNTIL RATE-CTR = 10
.....

RATE-LOOP PROCEDURE
DISPLAY RDA RATE :RATE-INDEX
SET RATE-CTR = RATE-CTR + 1
SET RATE-INDEX = RATE-INDEX + 5
ENDPROC

```

Note that the numeric variable RATE-INDEX is incremented by the byte length of the field being examined, and not by one (1) as in standard COBOL subscripting or indexing.

The following is the same example using RDA subscripting:

```

DEF SUB RATE-SUA RATE
.....
SET RATE-SUB = 1
DO RATE-LOOP WHILE RATE-SUB NOT = 10
.....
RATE-LOOP PROCEDURE
DISPLAY RDA RATE :RATE-SUB
SET RATE-SUB = RATE-SUB + 1
ENDPROC

```

In this case, a subscript variable, RATE-SUB, was defined for use in referencing the table. RATE-SUB is incremented the same way as a COBOL subscript. RATE-SUB may only be used to reference the table for which it was defined.

4.1.4 Variable RDA Referencing

A direct RDA reference allows the use of a variable *start-byte-position* and/or *length*. To use this feature, two reserved numeric variables, S\$ and L\$, have been provided. The contents of these variables will be used in a direct RDA reference whenever a zero *start-byte-position* and/or *length* are specified. The value of S\$ will be substituted for a *start-byte-position* of zero; the value of L\$ will be substituted for a length of zero.

Caution: If the values in either S\$ or L\$ cause an invalid RDA reference, runtime errors will result.

Examples:

1. RDA (0,0)
2. RDA (20,0) DISP-1

In the case of Example 1, both the *start-byte-position* and *length* will be obtained from the reserved variable S\$ and L\$.

Example 2 would refer to a FIELDATA field starting in position 20 with the length of the value in the reserved variable L\$.

4.2 Variable Data Storage Area (DSA)

The variable data storage area is where all reserved variables, user variables and literals are allocated by Q-LINK. Reserved variables are automatically defined and allocated by Q-LINK upon initialization. Literals are automatically allocated in this area as they are encountered by the command editor. The size of this area is a Q-LINK generation parameter. Definition of user variables, and the names and contents of reserved variables, will be covered later.

4.3 Object Program Area

The Object Program Area is where the internally encoded object program is stored for execution by the command executor. The size of this area is specified at generation time.

Chapter 5: Defining Variables

Q-LINK does not allow implicit variable definition. All variables must be defined using the `DEFINE` directive prior to being referenced. The `DEFINE` directive may be entered at any point, but must appear before the variable is referenced. There are only three types of data variables: signed decimal numeric, double precision floating point numeric and alphanumeric strings.

5.1 Alphanumeric String Variables (`DEFINE A`)

Alphanumeric variables may be defined as any length; however, they may not exceed the generated size of the variable data storage area. They may be specified with an initial literal value from 1 to 150 characters in length. When alphanumeric variables are defined with a specified length and no initial value, their initial contents will be undetermined. If an initial literal is specified with no length, the size of the variable will be the size of the literal.

Format:

`DEFINE A variable-name { || length-of-variable | 'string-literal' || }`

`DEFINE` may be abbreviated `DEF`.

If both *string-literal* and *length-of-variable* are specified, and *length-of-variable* is greater than the length of *string-literal*, the *string-literal* will be left justified and the variable will be padded with spaces to the right. If *length-of-variable* is less than *string-literal*, truncation will occur.

Examples:

1. `DEFINE A STARS '*****'`
2. `DEFINE A HOLD-REC 80`
3. `DEFINE A COMPANY-VARIABLE 80 'KMSYS Worldwide, Inc.'`

Explanations:

`STARS` in example 1 will contain 11 asterisks and is equivalent to:

```
DEFINE A STARS '*****' 11
```

In example 2, `HOLD-REC` will have a length of 80, but its initial contents will be unknown.

In example 3, `COMPANY-variable` will initially be set to "KMSYS Worldwide, Inc." but will have a length of 80.

5.2 Decimal Numeric Variables (`DEFINE N`)

All decimal numeric variables are allocated as 18-digit signed numbers. They may be defined with an initial value. If a value is not specified, the variable is initialized to zero.

Format:

DEFINE N *variable-name* [*decimal-value-and-precision*]

DEFINE may be abbreviated **DEF**.

The *decimal-value-and-precision* determines both the initial value of the variable and the number of implied decimal positions (scale factor) it will have.

Examples:

1. **DEFINE** N COUNT 123
2. **DEFINE** N MINUS-FIVE -5
3. **DEFINE** N HOLD
4. **DEFINE** N TOT-AMT .00

Explanations:

In example 1, COUNT will have an initial value of 123.

In example 2, MINUS-FIVE will be assigned an initial value of -5.

In example 3, HOLD will have an initial value of zero.

In example 4, TOT-AMT will have an initial value of zero, and two implied decimal positions.

5.3 Floating Point Numeric Variables (**DEFINE FP**)

All floating-point variables are stored in double precision format for maximum accuracy. When floating-point variables are moved to or from RDA single precision fields or decimal numeric items, translation will be made according to ASCII COBOL rules.

Format:

DEFINE FP *variable-name* [*value*]

DEFINE may be abbreviated **DEF**.

The *value* of a floating-point variable may be specified as in either decimal form, or standard floating-point notation.

Examples:

1. **DEFINE** FP VARIANCE 1.2400E+2
2. **DEFINE** FP RESULT

Additional information on floating-point usage may be found on the EDIT command.

5.4 KanjiVariables (**DEFINEK**)

Kanji variables may be defined as any length, but may not exceed the generated size of the variable data storage area. They may be specified with an initial literal value from 1 to 75 double-byte characters in length. When kanji variables are defined with a specified length and no initial value, their initial contents will be undetermined. If an initial literal is specified with no length, the size of the variable will be the size of the literal.

Format:

DEFINE K *variable-name* { || *length-of-variable-in-double-bytes* | '*string-literal*' || }

DEFINE may be abbreviated **DEF**.

If both *string-literal* and *length-of-variable-in-double-bytes* are specified, and *length-of-variable-in-double-bytes* is greater than the length of *string-literal*, the *string-literal* will be left justified and the variable will be padded with spaces to the right. If *length-of-variable-in-double-bytes* is less than *string-literal*, truncation will occur.

Chapter 6: Reserved Words

Q-LINK maintains a set of reserved words that can be used on many Q-LINK commands. These reserved words fall into two categories: reserved variables and special names.

6.1 Reserved Variables

Q-LINK automatically reserves a set of variables upon initialization. These variables occupy the same storage area as user variables, and may be accessed and modified in the same manner. The following is a list of the reserved variables and an explanation of how each are used:

Reserved Name	Description
\$TAB	\$TAB is a one-character alpha variable initially set to octal 011 (decimal 9), which represents a horizontal tab character. This may be used for formatting a BIS output line instead of the automatic tab insertion feature.
A8 and A30	A8 and A30 are general purpose alphabetic variables with a length of 8 and 30 characters respectively and an initial value of spaces. They may used by the user as desired.
C-AKEY	C-AKEY is a numeric variable with an initial value of 0. It will automatically be set to the AREA-KEY of the DMCA after the execution of a DML command.
C-AREA-NAME	C-AREA-NAME is a twelve-character alpha variable with an initial value of spaces. It is automatically set to the current AREA-NAME from the DMCA after all DML commands.
C-DBK	C-DBK is a numeric variable with an initial value of 0. It is set to the current of run unit database key when the SET CURRENT DBK command is executed.
C-O-R	C-O-R is a thirty-character alpha variable with an initial value of spaces. It will be set automatically after the execution of DML commands and will contain the current RECORD-NAME from the DMCA.
C-PAGE	C-PAGE is a numeric variable with an initial value of 0. It will automatically be set to the current PAGE-NUM of the AREA-KEY after each DML command.
C-REC	C-REC is a numeric variable with an initial value of 0. It will automatically be set to the current RECORD-NUM of the AREA-KEY after each DML command.

Reserved Name	Description
DATE	DATE is an alpha variable with a length of 10 characters. It is initially set to the current Gregorian date in edited form (example: July 25, 1994 = "1994/07/25"). This variable is altered by the DATE command.
DATE-NUM	DATE-NUM is a numeric variable with an initial value of the current Gregorian date in the form YYYYMMDD. This variable is altered by the DATE command.
DAY	DAY is a numeric variable with an initial value of the current day of the month. This variable is altered by the DATE command.
DEPN\$	DEPN\$ is a numeric variable initialized with the requesting BIS user's department number.
ERROR-NUM	ERROR-NUM is a numeric variable with an initial value of 21. It will automatically be set to the value of ERROR-NUM from the DMCA after the execution of all DML commands.
G-AKEY	G-AKEY is a numeric variable with an initial value of 0. It is not set by Q-LINK, but may be used by the Q-LINK user as desired.
G-AREA-NAME	G-AREA-NAME is a twelve-character alpha variable with an initial value of spaces. It is not used by Q-LINK, but may be used by the Q-LINK user as desired.
G-PAGE	G-PAGE is a numeric variable with an initial value of 0. It is not set by Q-LINK, but may be used by the Q-LINK user as desired.
G-REC	G-REC is a numeric variable with an initial value of 0. It is not set by Q-LINK, but may be used by the Q-LINK user.
G-RECORD-NAME	G-RECORD-NAME is a thirty-character alpha variable with an initial value of spaces. It is not used by Q-LINK, but may be used by the Q-LINK user.
IMPART-DEPART	IMPART-DEPART is a numeric variable with an initial value of 0. It is set to 1 after the execution of the DML IMPART, and to 2 after the execution of the DEPART command.
J-DAY	J-DAY is a numeric variable with an initial value of the current Julian date in the form YYYYDDD. This variable is altered by the DATE command.
L\$	L\$ is a numeric variable with an initial value of 0. It is used to replace the length-in-bytes value in a direct RDA reference if a length-in-bytes of zero is specified.
MONTH	This is a numeric variable with an initial value of the current month. This variable is altered by the DATE command.
RB-CODE	This is a numeric variable with an initial value of zero. It will be set to the rollback error code when a DMS rollback error occurs.
REC\$LEN	REC\$LEN is a numeric variable with an initial value of 0. It is automatically set to the record length (in words) after each sequential file read.

Reserved Name	Description
S\$	S\$ is a numeric variable with an initial value of 0. It is used to replace the start-byte value in a direct RDA reference if a start-byte of zero is specified.
SERVER\$	SERVER\$ is a six-character alphabetic variable initialized with the server's class name.
STNUM\$	STNUM\$ is a numeric variable containing the requesting user's BIS station number.
TIME	TIME is a twelve-character alpha variable with an initial value of spaces. It is set to the time of day in display format by execution of the TIME command. The format is HH:MM:SS.DDD.
TIME-MSPM	TIME-MSPM is a numeric variable with an initial value of 0. It is set to the time of day in milliseconds past midnight (MSPM) by execution of the TIME command.
USER\$	USER\$ is a 12-character alphabetic variable initially set to the requesting BIS user's user-id.
YEAR	YEAR is a numeric variable with an initial value of the current year (example: 1985). This variable is altered by the DATE command.
X, Y, and Z	X, Y, and Z are numeric integer variables with an initial value of zero with no implied decimal positions. They are set up for the convenience of the user and may be used for any purpose.

6.2 Special Names

Q-LINK automatically reserves a set of special names that can be used on several Q-LINK commands. The following is a list of special names and a description of how they are used:

Reserved Name	Description
\$ALPHA	\$ALPHA name can only be used in a conditional-expression on the IF or DO commands. The characters that meet the condition must be within the range of 'A' through 'Z', 'a' through 'z' (ASCII only), or space. A Q-LINK configuration parameter can change the interpretation of this special name.
\$HIVALS	\$HIVALS correspond to ASCII COBOL HIGH-VALUES and may be used in setting alpha variables or RDA items (see the SET command). \$HIVALS may also be used in a conditional-expression on the IF or DO commands. They are interpreted as all octal 177s (or 377s) for ASCII or all octal 77s for FIELDATA. A Q-LINK configuration parameter can change the interpretation of this special name.
\$LOVALS	\$LOVALS correspond to ASCII COBOL LOW-VALUES and may be used in setting alpha variables or RDA items (see the SET command). \$LOVALS may also be used in a conditional-expression on the IF or DO commands. They are interpreted as all binary zeroes.
\$NUM	\$NUM can only be used in a conditional-expression on the IF or DO commands. The characters that meet the condition must be within the range of 0 through 9.

Reserved Name	Description
\$PBUFF	\$PBUFF refers to the current contents of the print buffer which is filled by using the Q-LINK concatenation symbol on the DISPLAY, EDIT, TRIMDISP and TRIMEDIT commands. \$PBUFF may be referenced on the SET and RDMS commands. It will be cleared by a DISPLAY, EDIT, TRIMDISP or TRIMEDIT without a concatenation symbol or when referenced by a SET or RDMS command.
\$SPACES	\$SPACES may be used to set an alpha variable or RDA item to all spaces (see the SET command). \$SPACES may also be used in a conditional-expression on the IF command.

Chapter 7: Record Delivery Area (RDA)

This chapter covers the defining or partitioning of the input/output area of Q-LINK. This area is called the Record Delivery Area (RDA). The starting position of files, records, buffers, etc. can be defined using the DEFINE RA directive. Data items can be defined by using the DEFINE RDA directive and subscripts established (DEFINE SUB directive) for repeating data item definitions (OCCURS clause in COBOL).

The DEFINE F (Define File) directive is not covered in this chapter since its use and format is dependent upon a particular file system; PCIOS, DTM, etc. Refer to each file system's chapter in this manual for the DEFINE F directive required.

7.1 Alternate Record Area Definition (DEFINE RA)

All input and output of either DMS 2200 records, PCIOS file records or BIS DTM queue-alias areas default to position one of the RDA. This is not always desirable. When working with a database record whose location mode is via set and set occurrence selection is location mode of owner, or when data must be extracted from several different record types that must be current at the same time, having several records resident in the RDA is a requirement. The DEFINE RA directive allows alternate record areas to be used in I/O operations for DMS 2200 records, PCIOS files, DTM queue-alias areas and SORT records.

Format:

```
DEFINE RA {filename | record-name queue-alias | SORT} ;  
         {absolute-word-number | {OVERLAY | AFTER} } ;  
         {file-name | record-name | queue-alias | SORT } }
```

DEFINE may be abbreviated **DEF**.

For proper displacement of items within a named file/record, the DEFINE RA directive should be coded prior to referencing those items in the program.

Note: If alternate records delivery areas are to be defined for DMS records, they should be specified after INVOKE but prior to IMPART.

Filename applies to PCIOS files. The file must be currently defined with a DEFINE F directive (see Chapter 10, "PCIOS and SFS 2200 File Interface" in this manual). The *filename* may be an entry defined in a permanent data item index file by the QINDEX processor. By defining the file with the QINDEX processor, Q-LINK will be able to address file data items by name, and automatically determine the RDA location of the record.

For information on creating a permanent data item index file, see Chapter 18, "QINDEX Reference".

Record-name applies to DMS 2200 records. The record must be currently invoked (see "DMS 2200 Interface" in this manual). In addition, a DEFINE RA which references a DMS 2200 record must be coded prior to the IMPART for proper record offsetting to occur.

Queue-alias applies to BIS DTM reports. The *queue-alias* must be currently defined with a DEFINE F directive (see Chapter 15, "BIS DTM Interface," in this manual).

SORT specifies the area to be used for all sort RELEASEs and RETURNS.

Absolute-word-number is the beginning word address within the RDA that will be used for I/O. Word addresses may range from 1 to the generated maximum RDA length. Word number is used here instead of character address to ensure that the record area is established on a full word boundary.

OVERLAY allows one record area to overlay another. The OVERLAY clause is used instead of an absolute word number. If the record to be overlaid is located in an alternate record area, its DEFINE RA must precede the definition in which the OVERLAY is specified.

AFTER is used when one record area is to follow another. The AFTER clause is used instead of an absolute word number. If the record to be followed is located in an alternate record area, its DEFINE RA must precede the definition in which the AFTER is specified.

When using absolute or direct RDA references, e.g., RDA (21,4), care should be taken when used in relationship to the DEFINE RA ... AFTER directive. When the AFTER clause references a record/file named on a previous DEFINE RA ... AFTER directive, the location of the record/file that is being defined will start one word (four ASCII bytes) beyond the length of the referenced record. Take, for example, five records that are ten words each. The DEFINE RA directives might be coded as follows:

```
DEF RA REC2 AFTER REC1
DEF RA REC3 AFTER REC2
DEF RA REC4 AFTER REC3
DEF RA REC5 AFTER REC1
```

REC1 Words 1-10	REC2 & 5 Words 11-20	21	REC3 Words 22-31	32	REC4 Words 33-42
--------------------	-------------------------	----	---------------------	----	---------------------

Words 21 and 32 are not used. REC2 and REC5 do not reference a record named on a previous DEF RA directive; therefore, they begin immediately after REC1. REC 3 and REC 5 reference a record named on a previous DEF RA; therefore, they begin one word to the right of the referenced record. This situation was detected in an earlier release of I Q-LINK and cannot be changed for reasons of upward compatibility; i.e., changing Q-LINK to eliminate this extra word, could adversely affect existing Q-LINK programs that contain direct references.

An alternate record area for any single file, record or SORT may be defined only once in an Q-LINK program.

Examples:

1. DEFINE RA R0001-PART-MASTER 100
2. DEFINE RA SORT AFTER R0001-PART-MASTER
3. DEFINE RA OUTFILE OVERLAY SORT

Explanations:

In example 1, DMS 2200 will use the area beginning at word 100 of the record delivery area for all input and output for the R0001-PART-MASTER record.

Example 2 causes all SORT Release/Return commands to use the area beginning at the next word following the R0001-PART-MASTER record in the RDA.

In example 3, all I/O for the PCIOS file OUTFILE will be done from the same area used by the SORT.

7.2 Record Delivery Area Field Definition (DEFINE RDA)

Areas within the RDA may be defined for reference by name without an external data item index, or items already defined in a data item index may be redefined by the DEFINE RDA directive. This is useful when there is a need to redefine fields, or when referencing files for which no data item index file has been generated. DEFINE RDA allows the user to assign a name to a direct, or relative, RDA reference, thus simplifying future references to the data.

Format:

```
DEFINE RDA item-name (start-byte-position, length) ;
      [ data-type ] [ decimal-precision ]
```

Where:

start-byte-position = {*RDA-absolute-start-position* | * | *item-name* | **item-name*}

DEFINE may be abbreviated **DEF**.

The *start-byte-position* may be specified in several ways. The *start-byte-position* may be given as an integer offset from the start of the RDA. An asterisk (*) in the *start-byte-position* indicates that this item begins immediately following the last RDA field defined. If an asterisk is used on the first DEFINE RDA in the program, position one (1) will be assumed. If the *start-byte-position* is given as a defined RDA *item-name*, either defined with a previous DEFINE RDA or in the current primary or secondary data item index, the item being defined will start at the same position of the named item. If an RDA *item-name* is used, preceded by an asterisk (*), the item being defined will start immediately following the named item. An RDA *item-name* used in a DEFINE RDA may be qualified by record or file name.

The *length* is the actual length of the referenced data. For computational data items, this is the actual byte length of the data, not the COBOL picture length. For example, if the COBOL picture is PIC 9(10) COMP, the actual byte length is 4 (assuming the data is ASCII). Figure 3.2 may be used to determine the byte length of various computational field definitions.

If the definition specifies zero in either position of the RDA reference, the value of S\$ and/or L\$ will be used when *item-name* is referenced. In addition, *item-name* can be referenced with an index or subscript when used in a command.

The decimal-precision is a literal showing how many decimal positions are implied (the scale factor). The decimal-precision specification does not assign a value to the RDA field.

The data-type must be one of the allowed Q-LINK RDA data types.

Examples:

1. DEFINE RDA PART-KEY (1,10)
2. DEFINE RDA PART-KEY-FIRST-4 (PART-KEY,4)
3. DEFINE RDA PART-KEY-REST (*,6)
4. DEFINE RDA PRICE-CODE (11,4) COMP
5. DEFINE RDA PRICE (*,4) COMP .00
6. DEFINE RDA PART-TABLE (0,3)
7. DEFINE RDA TITLE-FIRST-CHAR (*EMP-NAME OF PAYMAST,1)

Explanations:

In example 1, PART-KEY defines a display field in the first 10 characters of the RDA.

In example 2, PART-KEY-FIRST-4 redefines the first 4 positions of PART-KEY defined in example 1.

PART-KEY-REST, in example 3, redefines the last 6 positions of PART-KEY. The asterisk indicates that PART-KEY-REST immediately follows PART-KEY-FIRST-4 in the RDA since it was the last RDA field defined.

In example 4, PRICE-CODE is a computational data item immediately following the PART-KEY in the RDA.

In example 5, PRICE is a computational field with 2 implied decimal positions. The equivalent COBOL picture would be PIC S9(8)V99 COMP. PRICE will immediately follow PRICE-CODE in the RDA.

In example 6, PART-TABLE defines a 3-character field with variable start position to be determined at run time based on the content of the reserved variable S\$.

Example 7 demonstrates the redefinition of a portion of a data item defined in a DMS 2200 record. TITLE-FIRST-CHAR starts immediately following the item named EMP-NAME in record PAYMAST (see qualified RDA item referencing).

7.3 Subscript Variable Definition (DEFINE SUB)

Subscript variables may be defined for use in referencing RDA data item arrays. A subscript variable may only be used to reference items within a single array; therefore, each array to be referenced would require a different subscript variable. A subscript variable contains the length of the subscripted item as well as the desired occurrence number of the item to be referenced. The item length portion may not be accessed by the user. Only the occurrence number can be referenced.

Format:

DEFINE SUB *subscript-name* *RDA-reference* [*:subscript-name*]

DEFINE may be abbreviated **DEF**.

The *RDA-reference* either may be a direct RDA reference or may refer to a data item name from the currently invoked subschema, secondary data item index file or a user defined RDA reference.

The optional *:subscript-name* on the RDA-reference is used in setting up subscripts for multi-dimensional arrays. It names the subscript variable used to reference the RDA-reference name within an occurring group. When the optional *:subscript-name* is used, the RDA-reference may not be direct.

Examples:

Given the following COBOL two-dimensional table definition:

```
05 BUDGET-TABLE.
   10 YEAR-ENTS OCCURS 5.
       15 MONTH-ENTS OCCURS 12.
           20 BUDGET-AMT PIC 9(7)V99.
```

Two subscript variables would be defined as follows:

```
1. DEFINE SUB YSUB YEAR-ENTS
2. DEFINE SUB MSUB MONTH-ENTS :YSUB
```

Explanations:

In example 1, YSUB is set up to reference each of the 5 YEAR-ENTS entries.

In the second DEFINE SUB, MSUB is set up to reference MONTH-ENTS within YEAR-ENTS.

To reference the BUDGET-AMT for the 2nd month of the 3rd year, the following may be used:

YSUB = 3

MSUB = 2

TOT-BUDGET = TOT-BUDGET + RDA BUDGET-AMT :YSUB :MSUB

Chapter 8: Control Directives

The following directives are used to perform non-procedural control functions.

8.1 ADD

The ADD directive is used to obtain program input from a symbolic element in an external EXEC file. ADD will cause Q-LINK to begin reading the specified element which may contain directives and/or procedural commands. When end-of-file is reached on the ADDED element, Q-LINK will then resume reading input from the request stream. Any number of ADD directives may be placed anywhere in the Q-LINK request. ADD directives may be nested up to a depth of ten (10) levels. The formats for the ADD directive are:

Format 1:

`ADD element-name [FROM qualifier*filename]`

Format 2:

`ADD * FROM qualifier*filename`

ADD may be abbreviated `AD`.

If the `FROM qualifier*filename` is not specified, the default Q-LINK program source code library file, `Q$LNK*Q$LNKLIB.`, will be used (the qualifier, `Q$LNK`, is a runtime configuration parameter and may differ from site to site).

If Format 2 is used, the `qualifier*filename` is required and must be an SDF data file.

Format 2 is used when adding a file instead of an element.

Examples:

```
ADD CUSTMAST/DEFS          .   Use default lib.
ADD MISC-DEFS FROM IQU*PROCS
ADD * FROM DATA*FILE.
```

8.2 COMPILE

The COMPILE directive causes Q-LINK to resolve all program and procedure labels, all IF/ENDIF pairs, and all PROCEDURE/ENDPROC pairs to complete the internal program and prepare it for execution or saving. If any errors are detected during this process, further processing of the program will not be attempted. The COMPILE is implied automatically if it has not been called before a RUN or SAVE directive. There are no parameters associated with the COMPILE.

8.3 DIAG

The DIAG directive is used to put a Q-LINK server into diagnostic mode for the duration of the request. For most complete results, the DIAG directive should be the first image in the request stream. Any prior information contained on the server's diagnostic log will be lost.

8.4 INDEX

The INDEX directive is used to assign a secondary data item index file. The INDEX directive must specify the name of a Q-LINK data item index file created by the QINDEX processor, or by the Q-LINK Processor. Only one secondary data item file may be assigned at any given time.

For information on creating a permanent data item index file, see Chapter 17, "QINDEX Reference".

Format:

INDEX *qualifier*filename*

8.5 INIT

The INIT directive is used to reinitialize all data storage areas (the RDA). All reserved variables will be reset to initial values and all user variables will be deallocated.

8.6 LISTOFF

The LISTOFF directive will suppress the listing of commands as they are edited by Q-LINK.

8.7 LISTON

The LISTON directive will cause resumption of the listing of commands as they are edited.

8.8 LOAD

The LOAD directive is used to load a previously compiled program into Q-LINK's internal areas. Once loaded, the program may be executed using the RUN directive, or saved to another object program file. See the RUN and SAVE directives for more information.

Format:

LOAD *program-name* [**FROM** *qualifier*filename*]

LOAD may be abbreviated **L**.

If the *qualifier*filename* is omitted, Q-LINK will attempt to load the object program from the default file Q\$LNK*Q\$LNKOBJ (the qualifier, Q\$LNK, is a runtime configuration parameter and may differ from site to site).

8.9 OBJECT

The OBJECT directive will produce a formatted dump of all internal areas. The dump will not destroy the current contents of the data storage areas.

8.10 PRINTER

The PRINTER directive is used to change the default device name to which alternate print output will be @SYMed when the request is complete. The device is selected by the user by

entering a number between 0 and 4. Each number represents a device name specified during the Q-LINK dynamic configuration process (see the Q-LINK Installation and Operation Guide).

Format:

PRINTER *site-printer-number*

PRINTER may be abbreviated **PR**.

This directive applies only to alternate default print files.

8.11 RETLOG

This directive is used to close the current Q-LINK server diagnostic log file and copy it to the BIS reply file. The log file will then be re-opened for output, thus starting a new log. RETLOG allows the Q-LINK server log file to be retrieved into a BIS report for examination. For multiple servers running in the same server class, you must direct the request to a specific server by using its EXEC run-id on the @QLK function call (see QLK function) or by using the KMSYS Worldwide supplied QUTIL BIS run. The server run-id which processed the request can be found as the last field on the .\$ERRORS line returned at the end of the request. The .\$ERRORS line only appears if the request encountered an error.

8.12 RUN

The RUN directive is used to execute a Q-LINK program that has been entered and compiled, or to load and execute an object version of a program that has been compiled previously and saved in object form (see SAVE directive).

Format:

RUN [*program-name* [FROM *qualifier*filename*]]

RUN may be abbreviated **R**.

If both *program-name* and *qualifier*filename* are omitted, Q-LINK will begin executing the currently compiled program. If the program has not been previously compiled, the RUN directive will cause an implicit compilation prior to execution.

If the FROM *qualifier*filename* is omitted, Q-LINK will attempt to load the object program from the default object library, Q\$LNK*Q\$LNKOBJ (the qualifier, Q\$LNK, is a runtime configuration parameter and may differ from site to site).

The RDA contents are initialized whenever a new Q-LINK request begins execution. The RDA contents are preserved between job steps (multiple RUN directives) in the same user session (request).

Examples:

RUN

The above would require that a program had just been entered, and would force a compile if one had not already been done.

RUN CUST-QUERY

The above will load and run the object form of the program CUST-QUERY from the default object file Q\$LNK*Q\$LNKOBJ.

RUN SALES FROM SALES*QLINK-OBJ

The above will load and run the object program SALES from the user-cataloged file SALES*QLINK-OBJ.

8.13 SAVE

The SAVE directive is used to save the currently compiled Q-LINK program, in object form, as an OMNIBUS element. The object form includes all internal tables (including D\$WORK and S\$WORK) required to execute the program. The SAVE directive may immediately follow a COMPILE directive. An implicit COMPILE of the current program will be performed if required.

Format:

SAVE *program-name* [INTO *qualifier*filename*]

SAVE may be abbreviated **SA**.

If the *qualifier*filename* is omitted, Q-LINK will attempt to save the object program into the default object library, Q\$LNK*Q\$LNKOBJ (the qualifier, Q\$LNK, is a runtime configuration parameter and may differ from site to site).

The SAVE directive may not be preceded by a RUN directive.

Examples:

```
SAVE CUST-QUERY
```

The above will save the currently compiled object program as CUST-QUERY in the default file Q\$LNK*Q\$LNKOBJ.

```
SAVE SALES INTO SALES*QLINK-OBJ
```

The above will save the object of the currently compiled program as SALES in the user-cataloged file SALES*QLINK-OBJ.

8.14 SHUTDOWN

The SHUTDOWN directive causes a complete shutdown of the Q-LINK server processing the request. SHUTDOWN is a restricted directive to be used only at the direction of the system administrator (a password is required on the QLK function call). In a multiple server class, the RUN ID of the desired server should be specified on the @QLK request.

8.15 Object Program Considerations

Using object forms of programs will provide substantial performance improvements because the invoke processing, command interpretation, and compile are eliminated. An object Q-LINK program is sensitive to the same types of schema and subschema changes that cause COBOL/DML programs to be recompiled; therefore, it may be necessary to recompile and save Q-LINK programs after certain schema and/or subschema changes are implemented. In addition, certain changes in the configuration of the Q-LINK processor may make an object program incompatible, requiring recompilation and saving.

The default object file Q\$LNK*Q\$LNKOBJ must be catalogued by the user site prior to use of Q-LINK. The qualifier of the default object file may be changed when configuring the runtime portion of Q-LINK.

Typically, while a program is in the development phase, full source would be sent in the request from BIS, including an INVOKE, COMPILE and RUN. Once a program is tested and ready for production use, it would be compiled and saved in object form (see SAVE directive). The request sent from the BIS run would consist of the following:

```
RUN program-name . <---Load and execute object.
data-images . <---Optional input data.
```


Chapter 9: General Procedural Commands

Procedural commands can be divided into seven categories:

- General Procedural Commands;
- COBOL File (PCIOS/SFS 2200) Handling Commands;
- SORT Interface Commands;
- Direct I/O Commands;
- DMS 2200 Data Manipulation Language Commands;
- RDMS 2200 Relational Database Commands;
- BIS DTM Interface Commands.

This chapter only describes the general procedural commands. The other categories will be covered in chapters that follow.

The general procedural commands give the Q-LINK user the ability to display and change data in both the RDA and the variable data storage area.

9.1 ACCEPT

The ACCEPT command is used to set variables to values entered by the Q-LINK user. When executed, ACCEPT will display a prompt string defined by the programmer (if specified), then read from the input stream (from BIS) or the system console. Q-LINK will set the specified variable to the value entered. Input is limited to 256 characters (132 characters for BIS level 34 and earlier). Numeric input may contain a leading sign with no intervening spaces. The optional *prompt string* may not exceed 50 characters. It should only be used when the CONSOLE option is specified.

Format:

```
ACCEPT variable ['prompt-string'] [ [FROM] CONSOLE] ;  
[ AT END {BREAK | program-label} ]
```

ACCEPT may be abbreviated A; while CONSOLE, CONS or CONSOL.

The AT END clause is optional. BREAK may only be used instead of a label when the ACCEPT is contained within a defined procedure, or an in-line DO block.

Examples:

1. ACCEPT BIS-DATA AT END END-RUN
2. ACCEPT X 'Enter Product Id:' CONSOLE

Example 1 shows how a variable may be accepted from the terminal or runstream. If an AT END condition occurs, control will be returned at the program label END-RUN.

In example 2, the ACCEPT will prompt and obtain input from the system console to put into variable X.

9.2 BITMERGE

This command is used to combine a specified number of bits from three numeric variables to form a result in a single numeric variable. For this operation, numeric variables are viewed as unsigned single-word entities. The same variable may be repeated in any position within the command syntax. There must be three counts specified. The first bit count may be zero, but the sum of the three must add up to 36. Bits will be extracted from the right-most positions of the source variables and deposited into the result variable starting from the left.

Format:

```
BITMERGE result-variable ;
      FROM variable-name-1,variable-name-2,variable-name-3 ;
      (bit-count-1,bit-count-2,bit-count-3)
```

BITMERGE may be abbreviated **BM** or **BITM**.

The *result-variable*, *variable-name-1*, *variable-name-2* and *variable-name-3* must be defined numeric variables. The *bit-count-1*, *bit-count-2* and *bit-count-3* must be given as unsigned integer literals.

Examples:

1. The following code builds a database key with 10 bits for the area, 17 for the page and 9 for the record number:


```
SET X = 121                . AREA CODE
SET Y=50                   . PAGE NUMBER
SET Z=1                    . RECORD NUMBER
BITMERGE X FROM X,Y,Z (10,17,9) . MAKE DBK IN
```
2. The example below builds an area key in the pre-defined numeric variable G-AKEY. The page number is placed in the first half of the word, and the record number in the second half. Y is assumed to contain the page number and Z the record number.

```
BITMERGE G-AKEY FROM Y,Y,Z (0,18,18) . AREA KEY
```

The first reference to Y in the area key example above is a dummy reference needed because all variables are required in the syntax.

9.3 BITSPLIT

The BITSPLIT command is used to disperse bits from one numeric variable into three numeric variables. Each variable used is viewed as an unsigned single-word entity. The same variable may be repeated in any position. There must be three-bit counts specified. The first specified bit-count may be zero, but the sum of the three must equal 36. Bits will be extracted from the source variable beginning from the right and deposited into the low-order positions of the result variables.

Format:

```
BITSPLIT source-variable ;
      INTO variable-name-1,variable-name-2,variable-name-3 ;
      (bit-count-1,bit-count-2,bit-count-3)
```

BITSPLIT may be abbreviated **BS** or **BITS**.

The *source-variable*, *variable-name-1*, *variable-name-2* and *variable-name-3* must be defined numeric variables. The *bit-count-1*, *bit-count-2* and *bit-count-3* must be given as unsigned integer literals.

Examples:

1. Split a database key into its three components (area code, page number and record number):

```
BITSPLIT C-DBK INTO X,Y,Z (10,17,9)
. SPLIT DBK
```

2. Isolate the high-order bit of a word:

```
SET X = RDA (21,4) UB9
. GET WORD FROM THE RDA
BITSPLIT X INTO X,Y,Z (0,1,35)
. ISOLATE BIT IN Y
IFY=1
DISPLAY 'BIT SET'
ELSE
DISPLAY 'BIT NOT SET'
ENDIF
```

9.4 BREAK

The BREAK command is used only in conjunction with a DO command. It is used to force an immediate termination of an in-line DO or defined procedure, regardless of condition. For an in-line DO, control passes to the statement following the DO/ENDDO pair. If the BREAK is in a defined procedure, control passes to the command following the DO from which the procedure was entered.

Format:

BREAK

Examples:

1. To BREAK out of a defined procedure:

```
DO COUNT WHILE Z < 900
...
COUNT PROCEDURE
  ACCEPT X 'Enter a value:'
  IF X = 0
    DISPLAY 'Zero entered, end of run'
    BREAK
. Exit DO loop immediately
ENDIF
Z = Z + X
ENDPROC
```

3. To BREAK out of a DO/ENDDO in-line loop:

```
...
. *** Search for first use type "A" or "G"
FETCH3 FIRST PART-USAGE SET
DO WHILE ERROR-NUM <> 7
  IF RDA PART-USE-TYPE = 'A'
  OR RDA PART-USE-TYPE = 'G'
    DISPLAY '*** FOUND IT ***'
    BREAK
  ENDIF
  FETCH3 NEXT PART-USAGE SET
ENDDO
...
```

9.5 CASE

The CASE command may be used to control case sensitivity for the DO, IF and SCAN commands. The default is CASE ON (for compatibility with older levels of Q-LINK) which implies that any test will be case sensitive. The CASE command is only significant when checking ASCII alpha display data.

Format:

CASE {OFF | ON}

Examples:

1. ACCEPT REPLY 'Summary? Enter (Y)es/(N)o:' CONS
CASE OFF
IF REPLY = 'y' . "Y" or "y" will test true.
DO SUMMARY
ENDIF
CASE ON

2. DEF F CMD-FILE SEQ 80,0
DEF RDA COMMAND (1,7)
:
CASE OFF
DO WHILE COMMAND <> 'process'
READ CMD-FILE
:
ENDDO

3. FETCH3 FIRST CUSTOMER-REC CUSTMST AREA
CASE OFF
DO WHILE ERROR-NUM = 0
SET X = 0 . Initialize length/result variable.
SCAN RDA CUSTOMER-NAME 'JONES' X . SCAN for string
IF X > -1 .Found?
DISPLAY RDA CUSTOMER-NAME . Display name
SET FIND-CNT = FIND-CNT + 1 . Count it
ENDIF
FETCH3 NEXT CUSTOMER-REC CUSTMST AREA
ENDDO

9.6 CONNECT

The CONNECT command allows a Q-LINK program to connect to the TIP on-line system. The command is only required when handling recoverable TIP/FCSS or TIPDMS files with the DIO command. The command may not be used while IMPARTed to DMS. This command may not be used to change the application association (application number).

Format:

```
CONNECT [ || {TO [APPLICATION] | APPLICATION} application-number | ;  
        AT level-number | ;  
        OPTION recovery-options || ]
```

CONNECT may be abbreviated **CONN**, with **APPL** for APPLICATION and **OPT** for OPTION.

The *application-number* must be from 1 to 9. If not specified, application 0 is implied.

The *level-number* is the connection level which defines which TIP facilities are available to this program. Valid values are 1 through 3.

The *recovery-options* are specified as an integer number and the specified bits must be as discussed for the CONNECT TIP primitive. If not specified, "no recovery" is implied and the *application-number* is ignored by the EXEC.

No attempt is made to detect or prevent redundant CONNECT commands. Q-LINK will issue requests to MCB/EXEC as submitted.

For more information, refer to the OS 2200 Transaction Processing Programming Reference Manual and the OS 2200 Transaction Processing Administration and Operations Reference Manual.

9.7 CSF

The CSF command allows the Q-LINK user to submit any valid CSF image to the operating system. The status returned from the operating system will be placed in the numeric variable specified. Q-LINK will not attempt to edit the image being submitted, so care must be taken to ensure that the format is acceptable to the operating system.

Format:

CSF *status-variable* {'*CSF-image-literal*' | *CSF-image-variable*}

CSF may be abbreviated **CS**.

Examples:

```
CSF X '@ASG,A MY*FILE.,T,REEL01'      .  Assign a file
IF X < 0                               .  Test status
    FACERR X                           .  Display reasons
    STOP EXIT
ENDIF

...
CSF X '@ASG,A INV*MASTER.  '
    IF X NOT = 0
        FACERR X
        STOP EXIT
    ENDIF
CSF X '@USE QLINK$USE$0.,INV*MASTER.  '
.  Leave file assigned to server
IF X NOT = 0
    FACERR X
    STOP EXIT
ENDIF

...
CSF X '@START PAY*RUNFILE.LIST-DB ' .  START A RUN
IF X NOT = 0
    DISPLAY 'Start failed, status follows:'
    DUMP X
ENDIF
```

Only the following control statements have significance in the Q-LINK environment:

@ADD **@CAT** **@FREE** **@START** **@SYM** **@USE**

9.8 DATE

The DATE command is used to reset the related DATE reserved variables to the current date. The DATE command may also be used to translate a user-furnished date from Julian to Gregorian or from Gregorian to Julian. The DWTIME option will convert a value in DWTIME\$ format to Gregorian and Julian formats and set a time variable.

Format:

```
DATE [ {date-variable | RDA RDA-reference} TO ;
      {JULIAN | GREGORIAN | DWTIME} ]
```

DATE may be abbreviated DA.

This command will set the values of the following predefined variables:

Variable	Data Type	Format
DATE	Alphanumeric	YYYY/MM/DD
DATE-NUM	Numeric	YYYYMMDD
YEAR	Numeric	YYYY
MONTH	Numeric	MM
DAY	Numeric	DD
J-DAY	Numeric	YYYYDDD
TIME (DWTIME option only)	Alphanumeric	HH:MM:SS.DDD

If the DATE command is used with no parameters, all date variables will be set to the current date.

The *date-variable* or *RDA-reference* must be defined as numeric and contain a positive numeric integer in the form YYMMDD or YYYYMMDD if converting to Julian, or YYDDD or YYYYDDD if converting to Gregorian. If the year is only two digits (YY), the century is derived from today's date. For example, if today were 2000-01-01 and the date variable contained 940729, DATE would reveal 2094/07/29.

When using the DWTIME option, the *date-variable* or *RDA-reference* must be defined as ASCII alphanumeric display for 8 bytes (characters); however, they must contain an ASCII aligned binary value, which represents nanoseconds since midnight, December 31, 1899. This value is said to be in DWTIME\$ format and is used internally in such places as the ASCII System Log, the Universal Data System, LINC 2200 applications, etc. In addition to setting the six reserved variables associated with a date (see above), the DWTIME option also sets the reserved variable TIME.

If the date cannot be converted, the result variables will be set to zero.

Examples:

1. The DATE command used alone to set the current date into all date variables:

```
DATE
```

2. Use the DATE command to determine if a given year is a leap year. Assume the variable ENTRY-YEAR is 2000.

```
DEF A ENTRY-YEAR 4
DEF RDA LEAP-A (4001,6)
DEF RDA LEAP-N (4001,6) SN9
...
```

```

ACCEPT ENTRY-YEAR 'Enter year: '
TD ENTRY-YEAR '060' +
SET RDA LEAP-A = $PBUFF
DATE RDA LEAP-N TO GREGORIAN
IF DAY = 29

```

```

TD YEAR ' is a leap year!'
...

```

The answer would be:

```

2000 is a leap year!

```

3. Convert the date in the variable ST-DATE to Julian. Assume ST-DATE contains 940719. J-DAY contains the Julian result. The value of ST-DATE will be set up in all other date variables:

```

DEF N ST-DATE
...
DATE ST-DATE TO JULIAN
D 'START DATE ' DATE +
ED J-DAY ' IS JULIAN 99/999.'

```

The result would be as follows:

```

START DATE 1994/07/19 IS JULIAN 1994/200.

```

4. Convert the DWTIME\$ formatted RDA reference, DWTIME-OF-START:

```

DEF RDA DWTIME-OF-START (*,8)
...
DATE DWTIME-OF-START TO DWTIME
D 'The program started on ' DATE +
D ' at ' TIME

```

The above example will yield results similar to the following:

```

The program started on 1994/09/25 at 16:17:41.041

```

9.9 DATESET

The DATESET command is used to convert a variable or RDA reference to a format suitable for displayed output or arithmetic operations.

Format:

```
DATESET {variable | RDA RDA-reference} ;
    [FORMAT {$DATEn | 'format-literal' | format-variable} ] = ;
    {variable | RDA RDA-reference} ;
    [FORMAT {$DATEn | 'format-literal' | format-variable} ]
```

The format field on either the sending or receiving side of the equal sign can be any of three choices, or omitted entirely if the default format is desired:

1. \$DATE*n* provides a set of predefined date formats that should handle the majority of date formatting tasks. The "*n*" can be 0-8 for standard BIS date formats, or 20-23 for KMSYS Worldwide defined formats. These formats are defined in the table below.
2. The *format-literal* can be used to replicate the \$DATE*n* formats or create other user-defined formats. The descriptors necessary to create these strings are defined in the date descriptor table on the following page.
3. The *format-variable* must be defined as an alphanumeric variable. The use of the *format-variable* allows the programmer to define format strings that may be set at execution time, which implies that the sending or receiving date string format is not necessarily known at compile time.
4. The default format (the FORMAT clause is omitted) is an internal Julian date based on 1/1/0001. Where 1 = 1/1/0001 , 2=1/2/0001 , 59 = 2/28/0001, etc. This format should be used when performing date calculations.

When converting dates from a 2-digit year to a 4-digit year, Q-LINK assumes the missing century portion of the year to be the current century. Likewise, if a 4-digit year is input and the century portion is "00", the current century will be assumed. It must be noted that because of these assumptions, 4-digit years in the range of "0001" through "0099" will not convert properly to an internal Julian date.

The following \$DATE*n* formats are predefined:

\$DATE<i>n</i>	Format	Type	Example
\$DATE0	YMMDD (Y1M2D2)	N	20327
\$DATE1	YYMMDD (Y2M2D2)	N	920327
\$DATE2	DD MMM YY (D2BM3BY2)	AN	27 MAR 92
\$DATE3	YDDD (Y1D3)	N	2087
\$DATE4	YYDDD (Y2D3)	N	92087
\$DATE5	DDMMYY (D2M2Y2)	N	270392
\$DATE6	MM/DD/YY (M2/D2/Y2)	AN	03/27/92
\$DATE7	MMMMMMMM DD,YYYY (M9BD2,Y4)	AN	MARCH 27,1992
\$DATE8	MMDDYY (M2D2Y2)	N	032792
\$DATE20	YYYYMMDD (Y4M2D2)	N	19920327
\$DATE21	YYYYDDD (Y4D3)	N	1992087

\$DATE_n	Format	Type	Example
\$DATE22	DDMMYYYY (D2M2Y4)	N	27031992
\$DATE23	MMDDYYYY (M2D2Y4)	N	03271992

The predefined formats have a data type of either numeric (N) or alphanumeric (AN). A numeric format may be used on a numeric or alphanumeric variable or RDA reference; however, an alphanumeric format may only be used on an alphanumeric variable or RDA reference. Consider the following alpha variable, numeric RDA reference and DATESET commands:

```
DEFINE A HDRDATE 17
DEFINE RDA PODATE (10,6) UN9
    DATESET HDRDATE FORMAT $DATE8 = RDA PODATE FORMAT $DATE1 . Correct
    DATESET HDRDATE FORMAT $DATE8 = RDA PODATE FORMAT $DATE6 . Incorrect
```

The first DATESET command is correct. Both \$DATE8 and \$DATE1 are numeric and can reference either a numeric field (PODATE) or an alpha field (HDRDATE). The second DATESET command will cause an error stating that the alphanumeric format (\$DATE6) is not compatible with the numerically defined field (PODATE).

The following date descriptors are allowed in the *format-literal* or *format-variable*:

Descriptor	Type	Explanation
-	A	Dash: Allows a dash in the input field, or inserts a dash in the output field.
,	A	Comma: Allows a comma in the input field, or inserts a comma in the output field.
.	A	Period: Allows a period in the input field, or inserts a period in the output field. The period is used in European date formats which are normally formatted as D2.M2.Y2 (e.g., "31.01.94").
/	A	Slash: Allows a slash in the input field, or inserts a slash in the output field.
B	A	Blank: Allows a blank in the input field, or inserts a blank in the output field. An actual blank or space character can also be used as the descriptor (e.g., "M9 D2, Y4" will produce the same result as "M9BD2,BY4").
D2	N	Day of the Month: Valid values are in the range of 01-31; however, the value must not exceed the maximum number of days allowed for the specified month.
D3	N	Day of the Year: Valid values are in the range of 001-366; however, 366 is only valid for a leap year.
I	N	Internal Julian Date: This format is the default when the FORMAT clause is omitted. This descriptor may not be used in combination with any other date descriptor. In this format, all dates are relative to the first day of year one.
M2	N	Month of the Year: Valid values are in the range of 01-12.
M3	A	Month of the Year Abbreviation: This format will always be 3 alphabetic characters (e.g., "JAN", "FEB", etc.).

Descriptor	Type	Explanation
M9	A	Alphabetic Month of the Year: The longest month name is 9 characters. Shorter month names will be space filled to the right.
W1	N	Day of the Week: Valid values are in the range of 1-7 where 1 is Sunday, 2 is Monday, etc.
W3	A	Day of the Week Abbreviation: This format will always be 3 alphabetic characters (e.g., "SUN", "MON", etc.).
W9	A	Alphabetic Day of the Week: The longest weekday name is 9 characters. Shorter names will be space filled to the right.
Yn	N	Year: n is the number of positions (1-4) required to represent the year. Valid values are in the range of 0001-2100. If n is less than 4, the high order positions of the year are truncated (e.g., if Y2 is specified, 1994 would be formatted as "94").

Each date descriptor is classified as either a numeric (N) or alphabetic (A) data type. If all the descriptors used in a format string are "A", the format is strictly alphabetic and can not be used with numeric variables or RDA references. If the string contains both "A" and "N" types, the format is alphanumeric and must be used with alphanumeric variables or RDA references. If all descriptors are "N", the format is numeric and may be used with both numeric or alphanumeric variables and RDA references.

The maximum number of date descriptors allowed in one *format-literal* or *format-variable* is six (6).

During execution, the DATESET instruction takes the input variable or RDA reference on the sending side of the equal sign and converts it to an internal Julian date based on the supplied format string or variable. If there is NOT enough information in the input data to exactly determine a specific date the missing values are used from today's date.

Assume the DATESET command looked something like the following:

```
DATESET AA1 FORMAT $DATE7 = AA2 FORMAT 'D2'
```

In this case, DATESET will only have the day of the month to work with. The month and year will be taken from the system clock and used to complete the date. This date will then be used to format the AA1 variable on the receiving side of the expression according to the \$DATE7 format string.

Given the variety of data types and different format strings that can be used together with this instruction, it makes sense that some error may occur that the DATESET instruction cannot handle. In this case, DATESET will set the global variable ERROR-NUM to an error number (following table). Upon successful completion of the DATESET command, the ERROR-NUM variable will be 0. ERROR-NUM should be checked after each execution of this instruction.

ERROR-NUM	Description
0	No error status, the instruction completed successfully.
1	The format string does not contain any characters. This should only happen when the format string is empty.
2	Invalid descriptor in format string.
3	Invalid length for descriptor. This will occur if any descriptor is given with an invalid number of characters. For example if a long Month is required but the specification is M8 this error will be returned.
4	Invalid number of descriptors in the format string. The number of date descriptors is limited to 6.
5	Not enough characters in format string. This will happen if for example a "W" is used without a field width specified.
6	Format string contains a descriptor that can only occur alone. This will only happen when the "I" descriptor occurs with any other descriptor.
11	Invalid descriptor type for a numeric string. The type of the variable does not match the type indicated in the format string.
12	The sending field format is numeric but contains an alphabetic month descriptor.
13	The sending field format is numeric but contains an alphabetic day of the week descriptor.
21	RDA variable of type MAPNUM on the receiving side of the equal sign is not large enough to hold the result of date formatting.
31	Year is out of range, less than 0001.
32	Month is out of range, must be 1-12.
33	Day of month is invalid for month specified; this error considers leap year for the number of days in February.

Example 1:

```

DEF A AA1 20
DEF N NN1
AA1 = 'MARCH 23,1992'
DATESET NN1 FORMAT $DATE1 = AA1 FORMAT $DATE7
      . INPUT IN M9BD2,Y4
TD AA1 ' CONVERTS TO ' NN1

```

This example will produce the following result:

```

MARCH 23,1992 CONVERTS TO 920323

```

Example 2:

```
DEF A CURR-DATE 18
DATE
DATESET CURR-DATE FORMAT 'M9BD2,BY4' = ;
    DATE-NUM FORMAT $DATE20
TD "Today's date is " CURR-DATE '.'
```

This example will produce the following result:

```
Today's date is AUGUST 12, 1994.
```

Example 3:

```
DEF RDA MAPDATA (1,132)
DEF RDA MAPDATE (14,8) MAPNUM . MM/DD/YY format
DEF A MAPVAR 132
ACCEPT MAPVAR AT END BREAK
RDA MAPDATA = MAPVAR
DATESET X = RDA MAPDATE FORMAT $DATE6
    . Convert to internal Julian day for increment
X = X + 120 . Add 120 days to the date
DATESET RDA MAPDATE FORMAT $DATE6 = X
    . Convert to MM/DD/YY

DISPLAY RDA MAPDATE
```

If the input was "02/15/92", this example will produce the following result:

```
06/14/92
```

9.10 DECIMAL

The DECIMAL command may be used to control decimal point sensitivity dynamically. Decimal sensitivity is initially ON by default, but may be turned off at any point in the program using this command. This command exists primarily for compatibility with older versions of the processor in which decimal point sensitivity did not exist.

Format:

DECIMAL {ON | OFF}

DECIMAL may be abbreviated DEC.

9.11 DISCONNECT

The DISCONNECT command allows a Q-LINK program that has issued a previous CONNECT command to detach itself from the TIP on-line system.

Format:

DISCONNECT

DISCONNECT may be abbreviated **DIS**.

No attempt is made to detect or prevent redundant DISCONNECT commands. Q-LINK will issue requests to MCB/EXEC as submitted.

9.12 DISPLAY

The DISPLAY command is used to display the contents of user or reserved variables or the contents of the RDA. DISPLAY may also be used to display alphanumeric literals.

Multiple display commands using the "+" symbol may be entered to build, but temporarily suppress, output. When "+" is encountered, data is moved into the print buffer, but no output is produced. When a DISPLAY without the "+" is executed, the data is moved to the print buffer and the contents of the print buffer are printed. (See the SET command to determine how to capture the contents of the print buffer, \$PBUFF).

An output column may be specified by entering an integer or numeric variable following the command keyword DISPLAY. If the numeric variable is specified, it must be preceded by the "at" symbol (@). If no output column is specified, data is moved into the next available position of the print buffer. If a column is specified, data is moved to the specified position of the print buffer and the next available position will become the position following the data.

If automatic TAB character insertion is on (see TABS command), a tab character will be inserted in the first output position, followed by the data to be displayed.

Format:

```
DISPLAY [ [column-literal | @numeric-variable] {RDA RDA-reference | ;
          {variable-1 | 'alpha-literal-1'} ... [variable-4 | 'alpha-literal-4']} [+]
```

DISPLAY may be abbreviated **D**.

A DISPLAY with no other options will generate a blank line.

Only one *RDA-reference* may be used within a single DISPLAY. However, from one to four alpha strings (*alpha-literal-n*) and variables (*variable-n*), or a combination of each, may be displayed using a single DISPLAY command.

Decimal numeric variables will be edited as 18-digit integers, zero suppressed with leading minus sign (COBOL edit picture '-----9'). Floating-point numeric variables will be displayed in general floating-point format. No other formatting will take place.

When using an *RDA-reference* in a DISPLAY, specified fields within the RDA may be printed using either direct RDA reference or data item name (if the data item index is available).

Examples:

Example 1: Display two literals on the same line:

```
DISPLAY '**** LIST OF PRODUCTS IN THE CURRENT ' +
DISPLAY 'DATABASE *** '
```

- . Use the + to concatenate the output
- . of two DISPLAY.

or

```
DISPLAY '**** LIST OF PRODUCTS IN THE CURRENT ';
'DATABASE *** '
```

- . Use the ; immediately after the second string
- . delimiter (') to continue the literal.

or

```
DISPLAY '**** LIST OF PRODUCTS IN THE CURRENT ' ;
'DATABASE *** '
```

- . Use the ; preceded by at least one space
- . to continue the command.

Regardless of the method used, the output will appear as follows:

```
**** LIST OF PRODUCTS IN THE CURRENT DATABASE ****
```

Example 2: Tab to column 5 and display three totals.

```
DISPLAY 5 '*** Totals for X, Y and Z are:'XYZ
```

Example 3: Display two lines containing one total on each.

```
DISPLAY 'VALUE OF X = ' X
DISPLAY 'VALUE OF Y = ' Y
```

Example 4: Display the same information as in Example 3, but on one line.

```
DISPLAY 'VALUE OF X = ' X ' AND VALUE OF Y = ' Y
```

Example 5: Display characters 1 through 8 of the RDA.

```
DISPLAY RDA (1,8)
```

Example 6: Display a field in the RDA beginning in output position 5.

```
DISPLAY 5 RDA PRODUCT-ID
```

Example 7: Display an RDA field with the output beginning in the position designated by the numeric variable, X.

```
DISPLAY @X RDA PRODUCT-ID
```

9.13 DO

The DO command can be used in two ways. First, the DO command may be used like the COBOL PERFORM statement. Used in this manner the DO must reference a defined procedure. A procedure must begin with a PROCEDURE label command and end with an ENDPROC command. If the optional WHILE/UNTIL clause is omitted, the procedure will be executed until the ENDPROC command is encountered, or a BREAK command is executed. If the WHILE/UNTIL clause is used, the procedure will be repeated per the condition specified (see below).

The second way in which the DO may be used is to control the execution of a group of commands in-line. In this manner, commands following the DO through a matching ENDDO command are executed repeatedly. This is called an in-line DO block.

Format-1, DO a defined procedure:

DO *procedure* [{UNTIL | WHILE} *conditional-expression*]

Format-2, DO in-line code:

DO [{UNTIL | WHILE} *conditional-expression*]
 commands(s)
 ENDDO

Where *conditional-expression* may take the following form:

{*numeric-literal* | *variable* | RDA *RDA-reference*} ;
 { [NOT] {= | < | >} | <> | <= | >= } ;
 {*numeric-literal* | *variable* | *special-name* '*alpha-literal*' | RDA *RDA-reference*}

Conditional expressions may be in any form acceptable to the IF command including the AND/OR extensions. Please see the IF command for a complete discussion of conditional expressions.

ENDDO may be abbreviated ENDD.

Format-1 Rules:

Doing a procedure differs from an in-line DO in that it may be entered from several different places throughout the user's program. The procedure is formally defined by the PROCEDURE and ENDPROC commands. The format for defining a procedure is:

Procedure-name PROCEDURE
 Command(s)
 [label] ENDPROC

PROCEDURE may be abbreviated PRO, with ENDP for ENDPROC.

If the procedure is entered with a DO UNTIL command, the condition will be tested each time the ENDPROC is executed. If the condition is met (TRUE), control will go to the command following the DO. Otherwise, the PROCEDURE will be re-entered from the beginning. Using a DO UNTIL, the procedure will always be executed at least once.

If the procedure is entered with a DO WHILE command, the condition will be tested at the beginning (at the PROCEDURE command). If the condition is not true, control returns to the command following the DO, otherwise the procedure is re-entered.

A procedure may be exited immediately by executing a BREAK command. The BREAK simply causes control to be passed back to the next command following the DO regardless of conditions.

A PROCEDURE / ENDPROC pair may not occur within a defined procedure (may not be nested). A PROCEDURE may call (DO) other procedures, including itself.

Format-2 Rules:

The DO must be terminated by a matching ENDDO. If no WHILE/UNTIL clause is used, the DO block will be repeated until a BREAK command is executed; otherwise, the same WHILE/UNTIL rules apply as in format-1. In-line DO blocks may be nested up to 50 levels.

Examples:**DOing defined PROCEDURES (Format-1):**

```

MAIN-CONTROL
  X = 0
  DO PROC-1                      . See NOTE-1
  X = 0
  DO PROC-1 UNTIL X = 100        . See NOTE-2
  . . .
  DO PROC-1 WHILE X < 900        . See NOTE-3
  . . .
PROC-1 PROCEDURE                 . Procedure label
  X = X + 1
  IF X > 500
    BREAK                        . Unconditional exit
  ENDIF
  DISPLAY X
ENDPROC

```

NOTE-1: PROC-1 will be performed 1 time.

NOTE-2: PROC-1 will be performed 100 times and the values of X displayed will be in the range from 1 through 100.

NOTE-3: PROC-1 will be performed 400 times and the values of X displayed will range from 101 through 500. The WHILE condition will never be met because of the BREAK command within the IF.

DOing in-line commands (Format-2):

```

. . .
FETCH4 NEXT PART-MAST PARTS AREA
FETCH3 FIRST PART-USAGE SET
DO WHILE ERROR-NUM = 0          . See Note-4
  DISPLAY "Usage codes for step number:" +
  DISPLAY PU-STEP-NO
  SUB1 = 1
  DO WHILE SUB1 <= PU-NO-ENTS    . Nested DO
    AND PU-USE-CODE :SUB1 <> $ALPHA
    DISPLAY PU-USE-CODE :SUB1
    SUB1 = SUB1 + 1
  ENDDO
  FETCH3 NEXT PART-USAGE SET
ENDDO
. . .

```

NOTE-4: If the FETCH3 FIRST results in an ERROR-NUM = 13 (no members in the set), the entire DO block will be skipped.

9.14 DUMP

The DUMP command is used to produce an octal, and ASCII, dump of the contents of the RDA, a specified variable or database data name (DBDN), or to take a full object dump programmatically (same dump as produced by the OBJECT directive). When DUMPing the RDA, the user may specify any range within the RDA either by specific word boundaries, or by DMS 2200 record, BIS DTM record (queue-alias) or PCIOS file name. If DUMP is given with no parameters, the entire RDA will be dumped. An octal DUMP of a variable will include the internal control portions as well as the data. A DBDN DUMP will include only the current contents of the specified database data name. If the DUMP ALL is used, a full object dump will be taken and program execution will resume.

Format:

```
DUMP { [start-word] number-of-words | filename | record-name | queue-alias | ;
      variable | DBDN database-data-name | ALL }
```

DUMP may be abbreviated DU.

Examples:

DUMP 20	. Dump words 1 through 20.
DUMP 21,4	. Dump words 21 through 60.
DUMP PART-MASTER	. Dump PART-MASTER record area.
DUMP ALL	. Take a full object dump and continue processing
DUMP DBDN MS-ANAME	. Dump a database data name
DUMP BIS-ROW	. Dump queue-alias retrieved/sent via DTM

9.15 EDIT

The EDIT command is used to display either numeric variables or numeric data in the RDA using an edit mask.

Using the EDIT command, one numeric variable or RDA data item may be edited and moved to the print buffer. If the item contains implied decimal positions, it will be scaled to match the decimal position in the edit mask.

Format:

```
EDIT ['column-literal' | @numeric-variable] ;
    {numeric-variable | RDA RDA-reference} ;
    [*] {'edit-mask-literal' | edit-mask-variable} [+]
```

EDIT may be abbreviated **ED**.

Edit masks are furnished as a literal or alpha variable containing mask characters. The edit mask is similar to that used in COBOL. The following edit mask characters may be used for editing decimal numbers:

1. Insertion characters:

Any character may be used as an insertion character; however, there are two special insertion characters, which are:

"B" blank;

"H" hyphen (-).

2. Zero suppression control characters are:

"Z" suppress leading zeros;

"9" display leading zeros.

3. Sign control characters are:

"+" print plus or minus sign depending on the value of the number;

"-" print minus sign only if the number is negative.

4. Floating characters are:

"\$" print dollar sign to the left of the most significant digit when the dollar sign is the first character of the mask and followed by the character, "Z" ;

"-" for negative numbers, print minus sign to the left of the most significant digit when the minus sign is the first character of the mask and is followed by the character, "Z".

Decimal number edit mask usage examples:

The number -1234, EDITed with a mask of '9999+' would print as:

```
`1234-'
```

The same number EDITed with a mask of 'ZZ,ZZZ' would print as:

```
` 1,234'
```

The number, 5678.909, EDITed with a mask of 'ZZZ.99+' would print as:

```
`678.91+'`
```

Note: Number was scaled to mask and rounded.

The number, -1234, EDITed with a mask of '\$ZZZ,ZZZ' would print as:

```
`$1,234'
```

The same number, EDITed with a mask of '\$999,999' would print as:

```
`$001,234'
```

The same number , EDITed with a mask of '-ZZZ,ZZZ' would print as:

'-1,234'

The same number, EDITed with a mask of '-999,999' would print as:

'-001,234'

Output may be temporarily suppressed by using the "+" symbol, as with the DISPLAY command. Column number specifications and automatic TAB character insertions also function as with the DISPLAY command.

An option may be used to indicate when a value overflows an edit mask. If an asterisk (*) is placed in front of the edit mask literal or variable, and an overflow condition occurs, the entire output (allocated by the edit mask) will be filled with asterisks. Without the asterisk, if the numeric value being edited is greater than the significant output positions allowed by the edit mask, the remaining high-order digits are truncated.

Examples:

X = 1234567

.

EDIT X 'ZZ,ZZ9.99' . No overflow indication

EDIT X * 'ZZ,ZZ9.99' . Overflow indication

The above would produce the output "34,567.00" (high-order truncated) from the first EDIT, and "*****" (indicating overflow has occurred) from the second EDIT.

The use of the overflow indication option does not affect the rounding or truncation of low-order digits. The numeric value is always scaled to match the decimal precision of the edit mask, and either rounded or truncated as necessary depending upon whether or not rounding is turned on.

There are three possible methods to edit floating-point numbers: fixed decimal format, generalized format and scientific format. In each case, the total output width and maximum precision will be specified. The floating-point edit mask formats are as follows:

Fixed decimal format:

Fw.p

Generalized format:

Gw.p

Scientific format:

Sw.p

Where "w" is an integer in the range of 1 to 32 that specifies the output field width, and "p" is an integer in the range of 0 to "w" that specifies the maximum precision to be displayed. In generalized format, "p" indicates the total number of significant digits.

In scientific format, the rules used to display images are as follows:

1. If $w - p \leq 5$ and the value being displayed is positive or if $w - p \leq 6$ and the value is negative, "p" significant digits are displayed. They will always be left justified. The exponent is displayed left justified and space filled up to $w - p - 2$ digits if positive or $w - p - 3$ digits if negative.
2. If $w - p > 5$ and the value is positive or $w - p > 6$ and the value is negative, "p" significant digits are displayed right justified and space filled with the sign preceding the value if negative. The exponent is displayed right justified and zero filled.

Examples:

Example 1: Assume X contains the value -12345.


```

EDIT X 'ZZ,ZZZ.99+' +
DISPLAY'/' +
EDIT X '-999999'

```

The output of the above commands would look like this:

```

Columns:1...5....10...15...20
      12,345.00- / -012345

```

Example 2:

```

DEF A PROD-MASK '999B9999'      . Product-id
                                . EDIT MASK ...
EDIT 3 RDA PRODUCT-ID PROD-MASK . Display EDITed
                                . product-id.

```

If product-id is 1012046, output from the above EDIT statement will print starting in column 3 as follows:

```

Columns:1...5....10...15...20
      101 2046

```

Example 3:

```

EDIT RDA (6,4) '99/99'

```

If positions 6 through 9 of the RDA contain 0922, output from the above EDIT statement will print starting in column 1 as follows:

```

Columns:1...5....10...15...20
      09/22

```

Example 4: Edit the value of Y into the column designated by X:

```

EDIT @X Y '-ZZZ,ZZZ'

```

If X = 5 and Y = -2424, output from the command would appear as follows:

```

Columns:1...5....10...15...20
      -2,424

```

Example 5: Floating-point numeric editing. Given the following:

```

DEFINE FP FLOAT-NUM 1.2400E+2
      EDIT FLOAT-NUM 'F9.5' . Edit fixed decimal format
      EDIT FLOAT-NUM 'S9.5' . Edit scientific format
      EDIT FLOAT-NUM 'G9.5' . Edit generalized format

```

The edited output would look like this:

```

Columns:1...5...10...15...20
      124.00000
      1.2400E+2
      124.00

```

9.16 FACERR

This command is used to decode and display the facility status word after execution of the Q-LINK CSF command. The specified variable must be numeric. FACERR will print the diagnostic message associated with each bit set in the facility status word. The display is directed to the current output (see the PCONTROL command).

Format:

FACERR *CSF-status-variable*

FACERR may be abbreviated **FA** or **FE**.

Example:

```
CSF X '@ASG,A MY*FILE.'      .  Assign file
IF X < 0                      .  Fac reject?
    FACERR X                  .  Display reason
    STOP 99                   .  Quit
ENDIF
```

9.17 GO

The GO command is used to transfer control of command execution to another point within the Q-LINK program.

Format:

GO [TO] *program-label*

Example:

```
.  A SIMPLE LOOP...
START
    SET X = X + 1
    IF X = 10
        DISPLAY 'DONE'
    ELSE
        GO START
    ENDIF
.  START IS A PROGRAM LABEL
.  GO BACK TO START
```

9.18 IF

The IF command is used to control conditional execution of Q-LINK commands. The IF must be used in conjunction with an ENDIF command. If a false condition results, control is passed to the command following the corresponding ELSE or ENDIF. IF/ENDIF pairs may be nested to 10 levels.

Format:

```
IF conditional-expression
  [ {AND | OR} ] conditional-expression
    command(s)
[ELSE]
    command(s)
ENDIF
```

Where *conditional-expression* may be in the following forms:

```
{numeric-literal | variable | RDA RDA-reference} ;
{ [NOT] {= | < | >} | <> | >= | <=} ;
{numeric-literal | variable | special-name | 'alpha-literal' | RDA RDA-reference}
```

ELSE may be abbreviated **EL**, with **ENDI** for ENDIF.

When comparing two variables, both must be the same data type. When comparing two alphanumeric strings (variables or RDA), the length of the shortest argument will stop the comparison. For example: "ABCDEF" = "ABC" is true; "ABCDEF" = "ABCx" is false.

Alphanumeric string comparisons that use an "is less than" (<) or "is greater than" (>) operator will use one of two collating sequences depending upon the character set of the two operands. If both operands are ASCII, the ASCII collating sequence will be used. If either or both operands are FIELDATA, the FIELDATA collating sequence will be used.

When the IF command is used to test an RDA reference that has a data type of MAPNUM, the reference must be tested as a numeric item (e.g., IF RDA QTY-ON-HAND = 0).

The IF command may also be used to test for HIGH-VALUES, LOW-VALUES, ALPHABETIC, or NUMERIC values by using one of the following special names instead of an alphanumeric literal:

\$HIVALS	All ASCII 177 (or 377) or Fielddata 77.
\$LOVALS	All binary zeroes.
\$ALPHA	All characters within ranges "A" through "Z", "a" through "z" (ASCII only) and space.
\$NUM	All characters within range 0 through 9.
\$SPACES	All spaces.

These special names may be used only with an equal or not equal test.

The IF command may be extended by using AND and OR commands immediately following the IF command. AND/OR logic operates as follows: all conditions connected by ORs are evaluated separately; all conditions connected by ANDs are evaluated together.

Consider the following compound condition:

```
IF A = 1
AND B = 2
OR C = 3
```

If A is 1, B is 3, and C is 3, the condition will be considered TRUE. This is because "C = 3" was true and ORed with "A = 1 and B = 2" that was false. This Q-LINK condition is interpreted the same as "IF (A = 1 AND B = 2) OR C = 3" would be by COBOL.

The ELSE may also be used in conjunction with the IF. The only restrictions on the use of AND, OR and ELSE is that they must be entered on separate command lines.

Examples:

```

FETCH3 NEXT PART-USAGE SET .  Fetch next of set
IF ERROR-NUM = 7 .  End of set ...
OR ERROR-NUM = 13 .  OR set empty
  IF ERROR-NUM = 7 .Isit7?
    GO END-SET .  ERROR-NUM is 7
  ELSE
    GO ERROR-ROUTINE .  Must be 13
  ENDIF
ENDIF
GO PROCESS-REC .  Continue processing
...

```

In the following, C-O-R is a 30-character alpha-string variable. If it contains any string that begins with "PRODU", the IF will be evaluated as true.

```

DEF RDA NEW-RATE (*,10) MAPNUM .00
DEF N MAX-RATE 25.00
...
  IF C-O-R = 'PRODU'
    GO ....
  ENDIF

  IF RDA NEW-RATE > MAX-RATE
  AND RDA PRODUCT-TYPE = 'A'
    DO RATE-TOO-HIGH
  ENDIF

```

9.19 PCONTROL

The PCONTROL command is used to direct the disposition of Q-LINK print output. Q-LINK output is normally directed to the BIS reply; however, the user may redirect output to the system console, or an alternate print file assigned by Q-LINK. Also, print may be directed to both an alternate print file and the BIS reply. PCONTROL functions are defined as follows:

BRKPT	Redirect all print to alternate print file.
CONSOLE	Redirect all print to the system console.
CLOSE	Close (@BRKPT and @FREE) the current print file.
ECHO	Redirect the output to both the reply and the alternate file.
EJECT	Page advance immediately (redirected output).
LOCAL	Direct output to PRINT\$ (initial mode).
OPTION	Send an optional print control image to the current print file.

Output is limited to 4,000 characters.

Format:

```
PCONTROL {BRKPT ['qualifier*filename' | variable] | ;
          CONSOLE | ;
          CLOSE | ;
          ECHO ['qualifier*filename' | variable] | ;
          EJECT | ;
          LOCAL | ;
          OPTION {print-control-image-variable | 'print-control-image-literal' } }
```

PCONSOLE may be abbreviated **PC**, with **CONS** or **CONSOL** for CONSOLE, **EC** for ECHO, **E** for EJECT, **L** for LOCAL and **OPT** for OPTION.

If the CONSOLE option is used, the print line should be limited to 50 characters to prevent truncation at the system console.

If the optional *qualifier*filename* is used with BRKPT or ECHO, Q-LINK will use the specified file. If the file does not exist, it will be created. Each time a new file is specified, the previous file will be @BRKPTed (closed) and @FREEed (released).

If no *qualifier*filename* is specified with BRKPT or ECHO, Q-LINK will attempt to create a file with a name in the range Q\$LKPR0 through Q\$LKPR9 (Q-LINK's default alternate print files). If all ten alternate print files already exist, an error will occur.

Upon exiting Q-LINK, any Q\$LKPRn files created will be @BRKPTed and @SYMed. User specified files will be @BRKPTed and @FREEed.

The *qualifier*filename* may be given as either an alpha literal or an alpha variable.

The PCONTROL command with the OPTION clause is used to call for special forms, change margins, or print headings. PCONTROL will output the print control function image furnished by the user in the *print-control-image-variable*, or the *print-control-image-literal*, to the current print file. Valid print control function formats will be found in the EXEC Programmers Reference Manual. In LOCAL mode, all OPTION images are ignored.

Below, some of the commonly used print control function formats are shown:

Special Forms Request:

S,forms-name-text

Headings:

H,options,page-number-to-start,heading-text

Where *options* are: N = Do not print heading; X = Suppress printing page number and date.

Line Skip:

L,line-number

Where *line-number* is an integer value one less than the desired line number. For example, to skip to line 15, use "L,14".

Margins:

M,lines-per-page,at-top,at-bottom,lines-per-inch

More than one print control function may be entered in a single image if separated by periods. For example, the following image in the print output will call for form 1-PART 9X11 to be mounted, and will cause each page to be headed with CUSTOMER LIST.

```
'S,1-PART 9X11.H,,1,CUSTOMER LIST'
```

Examples:

```
PCONTROL BRKPT . Breakpoint to default file
PCONTROL BRKPT 'MY*PRTF' . Breakpoint to a user file
```

The following will set 88 lines per page, with four lines at top and bottom margin, and print eight lines per inch.

```
PCONTROL OPTION 'M,88,4,4,8'
```

The following will call for special INVOICE forms to be mounted when the print file is printed.

```
PCONTROL OPTION 'S,INVOICE'
```

The following will cause all report pages to have the title "A/R SUMM" with the current date and page numbers starting at page 1.

```
PCONTROL OPTION 'H,,1,A/R SUMM'
```

9.20 ROUND

The ROUND command is used to toggle automatic rounding of arithmetic results on and off. Automatic rounding is initially ON. Rounding may occur on any command that produces a numeric result.

Format:

ROUND {ON | OFF}

ROUND may be abbreviated **ROU**.

Examples:

```
DEF N INT-NUM 0
DEF N DEC-NUM .000
...

DEC-NUM = 12.567
INT-NUM = DEC-NUM
DISPLAY INT-NUM
ROUND OFF
INT-NUM = DEC-NUM
DISPLAY INT-NUM
```

The preceding would produce the following:

```
13 <--- (Result ROUNDED)
12 <--- (Result not ROUNDED)
```


9.21 SCAN

This command is used to search for a specified string of characters within a field in the RDA. If the string is found, its offset from the start of the field being searched is returned, else a -1 is returned.

Format:

SCAN RDA *RDA-reference* {'*alpha-literal*' | *variable*} *length-result-variable*

SCAN may be abbreviated **SC**.

The *length-result-variable* is used to specify the length of the string to be compared in the scan. If it is set to zero, the length of the search literal ('*alpha-literal*') or *variable* will be assumed. The same variable (*length-result-variable*) will be used to return the result of the scan. If the string is found, the variable is set to the offset from the first character of the field at which the find was made. If the string was not found, a -1 is returned.

The *length-result-variable* may not exceed the length of the search literal or variable.

The search literal (*alpha-literal*) or *variable* may not exceed 256 characters.

The *length-result-variable* should be initialized each time a SCAN is to be executed.

Examples:

Search for all customer names containing the name "JONES".

```
...
FETCH3 NEXT CUSTOMER-REC CUSTMST AREA
IF ERROR-NUM = 7
    GO END-AREA
ENDIF
SET X = 0                                . Initialize length/result variable.
SCAN RDA CUSTOMER-NAME 'JONES' X        . SCAN for string
IF X > -1                                . Found?
    DISPLAY RDA CUSTOMER-NAME            . Display name
    SET FIND-CNT = FIND-CNT + 1          . Count it
ENDIF
...
```

The following procedure will perform a table lookup to convert an alpha code to its numeric value. The following table is stored in the RDA:

1001	1010	1020	1030	1040	1050	1060
...
A 01B	02C	03D	04E	05F	06G	07H 08I 09J 10K 11L 12M 13N 14O 15...

```
DEFINE A ALPHA-CODE 2
DEFINE N NUM-CODE
...
ALPHA-CODE = RDA REGION-CODE
DO CONVERT-CODE
IF NUM-CODE = 0
    DISPLAY 'ERROR-INVALID ALPHA REGION CODE'
ELSE
    DISPLAY NUM-CODE
ENDIF
...

CONVERT-CODE PROCEDURE
```

```
X = 0
NUM-CODE = 0
SCAN RDA (1001,104) ALPHA-CODE X      .   Search for code
IF X NOT = -1
    SET NUM-CODE = RDA (1003,2) UN9:X  .   Get value
ENDIF ...
```

9.22 SET

This form of the SET command will transfer data to the specified data storage area variable or RDA reference. The SET command can also determine the:

- Current or ending print position of the print buffer (\$PBUFF);
- Start position, length and data type of RDA references;
- Length of an item if leading and trailing spaces were suppressed;
- Number of decimal positions on numeric items;
- Switch settings (ON/OFF) of the I Q-LINK ROUND, DECIMAL and CASE commands.

Literals specified will first be stored in the data storage area by the command editor. In all cases of this form of SET, the data type of the object variable or RDA reference must be the same as the literal, variable or RDA reference from which the value will be obtained.

When the SET command is used to set an RDA reference that has a data type of MAPNUM, the reference must be set as an alphabetic item (e.g., SET RDA QTY-ON-HAND = '0'). If a numeric variable or numeric RDA reference is set to the contents of an RDA reference with a data type of MAPNUM, Q-LINK will convert the MAPNUM item to an internal format and then decimal align it (if required) for the receiving field. Any excess fractional digits will be rounded/truncated.

Format:

```
[SET] {variable | RDA RDA-reference} = ;
      {variable | ;
       special-name | ;
       RDA RDA-reference | ;
       'alpha-literal ' | ;
       (integer-literal, integer-literal) | ;
       expression | ;
       $VAL value-operation}
```

The command keyword "SET" is implied and may be omitted. SET may be abbreviated **S**.

Where *expression* may have the following format:

```
{numeric-literal | numeric-variable | RDA RDA-reference} ;
  {+ | - | * | /} ;
  {numeric-literal | numeric-variable | RDA RDA-reference}
```

Special-name may be one of four possible values: \$HIVAL, \$LOVAL, \$SPACES and \$PBUFF. \$HIVAL and \$LOVAL correspond to ASCII COBOL HIGH-VALUES and LOW-VALUES and may be used in setting alpha variable or RDA items. \$SPACES may be used to set an alpha variable or RDA item to all spaces. \$PBUFF refers to the current contents of the print buffer. It may be used to set an alpha variable or RDA item. Using this option will set the specified variable to the contents of the print buffer, and then space fill the print buffer as if it had been printed.

Where *value-operation* may have the following format:

```
{CURRPCOL | ;
  LASTPCOL | ;
  ROUND | ;
  DECIMAL | ;
  CASE | ;
  {STARTPOS {variable | RDA RDA-reference} ;
    TRIMLEN {variable | RDA RDA-reference} | ;
    LENGTH {variable | RDA RDA-reference} | ;
    DECPOS {variable | RDA RDA-reference} | ;
    TYPE {variable | RDA RDA-reference} } }
```

The \$VAL option is used to determine a value based on a specific function. The value returned will always be numeric; therefore, the receiving *RDA-reference* or *variable* (to the left of the =) must always be numeric. The other *RDA-reference* or *variable* (to the right of the =) may be either numeric or alphabetic within the limits described in the table below. The functions of the \$VAL option are:

Function	Purpose
CURRPCOL	This function returns the current print position maintained by Q-LINK in the print buffer (\$PBUFF). The current print position is the next available print position for a subsequent DISPLAY, TRIMDISP, EDIT or TRIMEDIT command unless overridden by a column specification. This function is only meaningful when preceded by a DISPLAY, TRIMDISP, EDIT or TRIMEDIT command utilizing the plus (+) option.
LASTPCOL	This function returns the highest print position currently in the print buffer (\$PBUFF). This function is useful to ensure that printed output will not exceed the physical limitations of the output device. This function is only meaningful when preceded by a DISPLAY, EDIT, TRIMDISP or TRIMEDIT command utilizing the plus (+) option.
ROUND	This function returns the value of the ROUND switch. A value of zero is returned if the ROUND switch is off, and a value of 1 is returned if the ROUND switch is on (see the ROUND command, page 9-37).
DECIMAL	This function returns the value of the DECIMAL switch. A value of zero is returned if the DECIMAL switch is off, and a value of 1 is returned if the DECIMAL switch is on (see the DECIMAL command, page 9-17).
CASE	This function returns the value of the CASE switch. A value of zero is returned if the CASE switch is off, and a value of 1 is returned if the CASE switch is on (see the CASE command, page 9-6).
STARTPOS	This function returns the start position of the named RDA-reference within the RDA. If a variable is specified, a start position of zero (0) is returned.
TRIMLEN	This function returns the size of the significant portion of the RDA-reference or variable. The returned value represents the size as if displayed by the TRIMDISP command with leading and trailing spaces deleted. This function is useful when determining the size of a single data item or an assembled print line for centering or right justification (see the examples below).

Function	Purpose									
LENGTH	This function returns the actual length of an RDA-reference. If a variable is specified, a length of zero (0) is returned.									
DECPOS	This function returns the number of decimal positions defined on a numeric RDA-reference or numeric variable. If an alphanumeric RDA-reference or alphanumeric variable is specified, a zero (0) result is returned.									
TYPE	This function returns the type code of the RDA-reference or variable. The following values are defined:									
	<i>ASCII</i>		<i>FIELDATA</i>		<i>Floating-Point</i>		<i>BIS</i>		<i>Variables</i>	
	DISP	16	DISP	48	FP1	5	MAPNUM	22	DEF A	201
	A9	16	A6	48	FP2	6			DEF FP	206
	SN9	19	SN6	51					DEF N	200
	UN9	18	UN6	50						
	COMP	1	COMP4	33						
	SB9	1	SB6	33						
	UB9	0	UB6	32						

Examples:

Given the following definitions:

```

DEFINE N OLD-DBP
DEFINE RDA DBP-RDA (1000,4) UB9
. Database Pointers (DBP)      and Database Keys
. (DBK) are always UB9
DEFINE N N1 .000
DEFINE N N2
DEFINE A HOLD-1 6
DEFINE A HOLD-2 6

.
.   RDA DBP-RDA = OLD-DBP      .   Named RDA ref.
.   RDA (1000,4) UB9 = OLD-DBP .   Direct RDA ref.
.
.   X = 0
.   N1 = 0
.   N1 = X + 1
.   N2 = -5
.   N1 = N2 + N1                .   N1 should now contain -4.000
.
.   HOLD-1 = 'ABCDEF'
.   HOLD-2 = HOLD-1              .   Move 'ABCDEF' to HOLD-2 also
.   RDA (1,6) = HOLD-1           .   Move HOLD-1 to RDA pos. 1-6
.   HOLD-1 = RDA (3,8)          .   Move RDA pos. 3-8 to HOLD-1
.                               .   Pos. 9-10 will be truncated
.   HOLD-2 = $LOVALS            .   Move binary zeros to HOLD-2
.
.   RDA PRODUCT-ID = 1011234
.   RDA PRODUCT-DESC = 'ABCD'
.   RDA (5,4) COMP = -505
.   RDA PRODUCT-ID = RDA LINE-PRD

```

```
RDA (1,100) = $LOVALS . Clear 100 chrs of RDA
```

The following form of the SET packs two integer numbers into a numeric variable.

```
G-AKEY = (1,5) . Set PAGE-NUM to 1
. and RECORD-NUM to 5
```

The actual decimal value of G-AKEY after the above SET would be 262149 (octal 0000001000005).

The following illustrates use of the \$PBUFF special value. First, values are formatted into the print buffer using EDIT, DISPLAY, TRIMEDIT and TRIMDISP commands. The '+' is necessary to prevent Q-LINK from printing the string.

```
DISPLAY 'The total is ' +
TRIMEDIT N1 '-ZZ,ZZZ.99' +
. + prevents $PBUFF from being cleared
HOLD-THING = $PBUFF
```

HOLD-THING will then contain "The total is -4.00" and the print buffer will be cleared.

Since an RDA item that is defined as having a data type of MAPNUM cannot be set as a numeric item, \$PBUFF can be used as a staging area for numeric data. For example:

```
DEF RDA NEW-RATE (*,10) MAPNUM .00
DEF N RATE-INCREASE .00
. . .

RATE-INCREASE = RDA NEW-RATE
. Numeric variable set to MAPNUM
RATE-INCREASE = RATE-INCREASE * 1.10 . Add 10%
TRIMEDIT RATE-INCREASE 'ZZZ,ZZZ.99' +
RDA NEW-RATE = $PBUFF
```

To avoid exceeding the length of an output device (e.g., 100 characters) the following \$VAL LASTCOL and TRIMLEN function may be used. This example creates a record containing trimmed amounts delimited by a single space (123.56 98.76):

```
DEF F AMOUNTS-OUT SEQ 100,10
DEF RDA OUT (1,100)
DEF RDA AMOUNT (101,10) .00
. . .

X = 0
DO WHILE X < 500 . Table contains 50 AMOUNTs
  DO UNTIL Y > 100 . Next TD would be > 100
    TD RDA AMOUNT :X +
    DISPLAY ' ' +
    Y = $VAL LASTCOL . Get last pos. of $PBUFF
    X = X + 10 . Bump by 10 for next AMOUNT
    Z = $VAL
    TRIMLEN RDA AMOUNT :X
    . Get trim length of it
    Y = Y + Z
  ENDDO
  RDA OUT = $PBUFF . Clears $PBUFF
  WRITE AMOUNT-OUT
ENDDO
```

The following example will center a staged header using variable substitution:

```
DEF A HEADER 132
. . .
  DISPLAY 'The following information is for ' +
  TRIMDISP RDA C01-CUSTOMER-NAME +
  DISPLAY ', ' +
  TRIMDISP RDA C01-CITY +
  DISPLAY ', ' +
  TRIMDISP RDA C01-STATE +
  X = $VAL LASTPCOL           . Get last pos. in $PBUFF
  X = 132 - X
  X = X / 2                   . Get start pos. of center
  HEADER = $PBUFF
  TRIMDISP @X HEADER          . HEADER begins at (@) X
. . .
```

9.23 SHIFT

The SHIFT command shifts lower-case characters to uppercase characters, or vice versa. FIELDATA RDA data items will not be affected by the SHIFT command. The shift only changes characters "A" through "Z" (to LOWER), and "a" through "z" (to UPPER).

Format:

SHIFT {*variable* | RDA *RDA-reference*} TO {UPPER | LOWER}

SHIFT may be abbreviated **SHI**.

Example:

```
DEF A ANSWER 1
.
. *** Accept until a valid answer is received ***
  DO WHILE ANSWER <> 'Y'
  AND ANSWER <> 'Y'
    ACCEPT ANSWER "Want to continue (Y or N)?" CONSOLE
    SHIFT ANSWER TO UPPER
  ENDDO
...
```


9.24 STOP

All Q-LINK programs (requests) must be terminated by the STOP command. A STOP command is automatically generated following the last command executed as a result of the RUN directive.

Format:

STOP {*exit-code-literal* | *exit-code-variable* | EXIT}

If the optional fields are omitted, the Q-LINK request will not terminate. Instead, Q-LINK will continue reading the request stream for additional directives. If no more input is found in the request stream, the Q-LINK request will then terminate and return control to BIS. If EXIT is used, the Q-LINK request will be terminated. If an exit code is used, the Q-LINK request will terminate, and the exit code value will be set in the BIS reserved name STAT3.

9.25 TABS

The TABS command is used to turn automatic tab insertion on and off. If on, a tab character will be inserted in the first output position of every DISPLAY, EDIT, TRIMDISP or TRIMEDIT followed by the data. The initial setting of TABS is off.

Format:

TABS {ON | OFF}

TABS may be abbreviated **TA**.

Example:

```
IF ERROR-NUM NOT = 13
  TABS ON
  . Output TABS in front of each field
  DISPLAY RDA KEY          +
  DISPLAY RDA DESC         +
  DISPLAY 27 RDA STATUS +
  ...
  TABS OFF
ELSE
  DISPLAY '. NO FIND ON ' +      . No TABS
  DISPLAY KEY
ENDIF
```

The tab line produced in the above example would look like this:

```
columns      1...5....10...15...20...25...
              ^2009^DOUBLE WIDGET          ^S
```

The caret symbol (^) is used in the above example to represent the transparent tab character.

9.26 TIME

The TIME command is used to set the current time of day into the reserved variables TIME-MSPM and TIME. The variable TIME-MSPM will be set to the current time of day in milliseconds past midnight (MSPM). TIME will be set to an edited display consisting of hours, minutes, seconds and thousandths of seconds (example: 13:23:45:028). If the user furnishes the time in milliseconds as a numeric input variable, this command will convert the variable to the edited time format and place the result in the variable TIME. This command may be used to time various Q-LINK program operations, or to limit run execution.

Format:

TIME [*time-variable* | *time-literal*]

TIME may be abbreviated **TI**.

Example:

```

DEFINE N ST-TIME                . Variable to save start time.
DEFINE N ELAPSED-TIME           . Variable to calc elapsed time.
...
TIME                            . Set time -beginning of run.
ST-TIME = TIME-MSPM             . Save for calculation later.
...
...
TIME                            . Set current time -end of run.
ELAPSED-TIME = TIME-MSPM -ST-TIME
                                . Calc elapsed time
IF ELAPSED-TIME > 60000         . Is time > 60 seconds?
    TIME ELAPSED-TIME          . Convert to edited form DISPLAY TIME
ENDIF

```

9.27 TRACE

The TRACE command is used to turn the Q-LINK TRACE routines on or off. When TRACE is turned on, Q-LINK will display the program counter (\$PC) value for each command executed. This command is often used in conjunction with the DIAG directive or a development server when debugging Q-LINK program logic.

Format:

TRACE {ON | OFF}

TRACE may be abbreviated **TRA**.

Example:

```
DIAG . On 1st line of program - causes
      . compilation listing showing program
      . counter beside each Q-LINK command when
      . using a production server (not needed
      . for development servers).
INVOKE ACCTSUB IN ACCT FILE ACCT*SCHEMA
DEF N TOT-ACCUM
...

TRACE ON . Will list the program counter of each
          . Q-LINK command executed.
F4 F ACCTREC ACCTAREA A
DO WHILE ERROR-NUM = 0
. . .
```

9.28 TRANSFER

The TRANSFER command is used to perform record level movement of data within the RDA. All record TRANSFERs are specified by actual word addresses, defined file names, invoked DMS 2200 record names or defined BIS DTM queue-alias names. The TRANSFER command is much more efficient than the SET command when moving records within the RDA.

Format:

```
TRANSFER {filename | record-name | queue-alias | SORT | from-word} TO ;
        {filename | record-name | queue-alias | ;
        SORT [number-of-words] | to-word [number-of-words] }
```

TRANSFER may be abbreviated TRANS.

Restrictions:

Specifying the same name or word address in both the "from" and "to" fields is illegal.

Specific word addresses cannot be used in both the "from" and "to" fields unless the *number-of-words* is specified.

Number-of-words only applies when both *from-word* and *to-word* addresses are specific word addresses, or when SORT is specified in conjunction with a *from-word* or *to-word* address.

The *number-of-words* to move when *filename*, *record-name* and/or *queue-alias* names are used will be determined by the length of the shortest or only defined area; *number-of-words* does not apply in this case.

Examples:

```
DEFINE RA R0009-MST 100      .  Alternate record area.
DEFINE RA SORT 200
...

IF RDA 0009-KEY OF R0009-MST = 'AB0001'
    TRANSFER R0009-MST TO SORT
                                .  Move to SORT
    RELEASE R0009-MST          .  Release len.  of R0009-MST
ENDIF
...
RETURN AT END END-OF-JOB
TRANSFER SORT TO 1,50         .  To words 1-50 of the RDA
WRITE COBFIL
...
```

9.29 TRIMDISP

The TRIMDISP command performs the same functions as the DISPLAY for variables and RDA items, except that any leading and trailing spaces are not moved to the print buffer (except for alpha literals). Also, the next available output position will only be advanced by the number of significant characters actually moved to the print buffer. This command allows strings to be concatenated with no intervening spaces.

Format:

```
TRIMDISP [column-literal | @numeric-variable] {RDA RDA-reference | ;
          {variable-1 | 'alpha-literal-1'} ... {variable-4 | 'alpha-literal-4'} } [+]
```

TRIMDISP may be abbreviated **TD** or **TRIMD**.

A TRIMDISP with no other options will generate a blank line.

Only one *RDA-reference* may be used within a single TRIMDISP. However, from one to four alpha string literals (*alpha-literal-n*) and variables (*variable-n*), or a combination of each, may be displayed using a single TRIMDISP command.

Trailing and leading spaces will not be suppressed on alpha string literals.

Output may be temporarily suppressed by using the "+" symbol, as with the DISPLAY command. Column number specifications and automatic TAB character insertions also function as with the DISPLAY command.

Examples:

```
TRIMDISP MONTH '/' DAY '/' +
TRIMDISP YEAR
```

The above would display a date in the following manner:

```
3/5/1994
```

Right justify variable output on a 132-character print line:

```
DEF A SUBTOTALHOLD 132
DEF N SUBTOTAL
  DISPLAY 'Subtotal is ' +
  TRIMEDIT SUBTOTAL '$ZZ,ZZZ,ZZZ.99' +
  X = $VAL LASTPCOL           . Get the length of display
  X = 132 - X                 . Compute starting column
  SUBTOTHOLD = $PBUFF         . Set hold variable
  TRIMDISP @X SUBTOTHOLD     . Print at (@) column X
```

9.30 TRIMEDIT

The TRIMEDIT command performs the same functions as the EDIT for variables and RDA items, except that any leading and trailing spaces are not moved to the print buffer. Also, the next available output position will only be advanced by the number of significant characters actually output to the print buffer. This command allows strings to be concatenated into the print buffer with no intervening spaces.

Format:

```
TRIMEDIT [column-literal | @numeric-variable] ;
        {numeric-variable | RDA RDA-reference} ;
        [*] {'edit-mask-literal' | edit-mask-variable} [+]
```

TRIMEDIT may be abbreviated **TE**, **TED** or **TRIME**.

Output may be temporarily suppressed by using the "+" symbol, as with the DISPLAY command. Column number specifications and automatic TAB character insertions also function as with the DISPLAY command.

Example:

```
DEFINE N AVERAGE .000
DEFINE N COUNT 0
DEFINE A NAME 14
...
TRIMDISP '*** ' NAME ' had ' +
TRIMEDIT COUNT 'ZZZ,ZZ9' +
DISPLAY ' sales, which averaged ' +
TRIMEDIT AVERAGE '-ZZZ,ZZZ.999' +
DISPLAY ' each.'
```

If NAME contains "SAM", COUNT contains 21 and AVERAGE contains 14.230, the preceding sequence of commands would produce the following line:

```
*** SAM had 21 sales, which averaged 14.230 each.
```

If the TRIMEDIT and TRIMDISP commands were replaced by EDIT and DISPLAY commands, respectively, the line would look like this:

```
*** SAM          had          21 sales, which averaged          14.230 each.
```

9.31 WAIT

The WAIT command is used to cause a voluntary delay in program execution. The delay is specified as an integer number of seconds.

Format:

WAIT {*numeric-literal* | *variable* | RDA *RDA-reference*}

Example:

```
ASG-FILE
CSF Z '@ASG,A MASTER.'
IF Z < 0                                . Can't get file

    X = X + 1
    IF X < 10                            . Less than 10 tries
        WAIT 5                          . Wait 5 more seconds
        GO ASG-FILE                     . Go try again
    ELSE
        DISPLAY "CAN'T ASSIGN MASTER"
        STOP EXIT
    ENDIF
ENDIF
```


9.32 WILDCARD

The WILDCARD command is used to set a wild card character to be used in alpha string compare operations of the IF and SCAN commands. The WILDCARD command may also be used to inhibit the use of any wild card characters in comparisons (the default).

Format:

WILDCARD IS {'*alpha-literal*' | *variable* | RDA *RDA-reference* | NONE}

WILDCARD may be abbreviated **WI**.

If a *variable* or *RDA-reference* is used, it must define an alphabetic item. If the supplied item is greater than one character in length, the leftmost character will be used. The wild card character will stay in effect until it is changed either to another character, or deactivated by specifying the keyword NONE in the command.

Examples:

```
WILDCARD IS '%' . Designate wild card char.
...
IF RDA PART-NUM = 'A%%1%%'
  DO SELECT
ENDIF
WILDCARD IS NONE . Clear wild card char.
```

Any time the PART-NUM contains an "A" in the first position and a "1" in the fifth position, the test will be true regardless of the contents of the rest of the field.

Chapter 10: PCIOS and SFS 2200 File Interface

The number of the Processor Common Input/Output System (PCIOS) and Shared File System (SFS) 2200 files that can be processed during one session is dependent upon how many files were configured when Q-LINK was installed. The default is 10. Each file must be defined using the DEFINE F directive before being referenced by the PCIOS/SFS file handling commands. All PCIOS/SFS file formats are supported.

For all file input/output commands, data will be read into, and written from, the RDA beginning at position one, or at the beginning of the area specified by a DEFINE RA directive for the file.

Special PCIOS status codes returned from Q-LINK can be found at the end of this chapter.

PCIOS error codes returned by the Processor Interface Module (PIM) can be found in the Unisys publication, PCIOS Administration and Programming Reference Manual (7831 0588). SFS 2200 error codes returned by the Logical Data Manager (LDM) can be found in the Unisys publication, UDS SFS 2200 Administration and Support Reference Manual (7831 0786).

10.1 PCIOS/SFS File Usage and Access Modes

The following table shows the allowed access and usage modes for each of the PCIOS file types:

File Type	Usage Modes				Access Modes
	INPUT	OUTPUT	UPDATE	EXTEND	
SEQ* (Sequential)	YES	YES	YES*	YES	SEQ
DIRECT (Also called Relative or DSDF)	YES	YES	YES	YES	DIRECT
INDEXED** (Also called MSAM)	YES	YES	YES	NO	SEQ, RANDOM, DYNAMIC
ISAM	YES	YES	YES	NO	SEQ, RANDOM, DYNAMIC
TAPE	YES	YES	NO	YES	SEQ

Table 10-1: File Usage/Access Modes

* SEQ files may be either disk or tape; however, tape files cannot be opened with a usage mode of UPDATE.

** INDEXED files may have a primary key, or a primary key with up to 19 alternate or secondary keys.

The file's type is specified in the DEFINE F directive, while the USAGE and ACCESS modes are specified on the PCIOS file OPEN command.

10.2 PCIOS/SFS File Definition (DEFINE F)

Before a PCIOS/SFS file may be referenced, it must be defined. Q-LINK supports standard PCIOS/SFS file formats. Files may be opened, referenced and closed as necessary. All file input and output is done through the RDA.

If multiple records are to be read and compared, the user may set up alternate record areas using the DEFINE RA directive, or move necessary data items to a defined variable storage location.

If a file is to remain assigned to the server once the request has finished execution, an @USE name of QLINK\$USE\$0 through QLINK\$USE\$9 may be applied via the CSF command (see the CSF command, page 9-8).

There are several formats used for file definition depending on the file type. The following file types may be defined:

SEQ	PCIOS sequential SDF files.
DIRECT	PCIOS direct (or relative) files (DSDF).
INDEXED	PCIOS indexed-sequential files (MSAM)
TAPE	PCIOS ANSI tape files.
ISAM	COBOL indexed files assigned to "MASS-STORAGE"

Format-1 SEQ:

```
DEFINE F[F] filename SEQ record-length,block-size [RECORDS | CHARACTERS]
```

Format-2 DIRECT:

```
DEFINE F[F] filename DIRECT [SHARED record-length ;
maximum-records-in-file,relative-key-variable
```

Format-3 INDEXED:

```
DEFINE F[F] filename INDEXED [SHARED] ;
record-length,block-size [RECORDS | CHARACTERS] record-key ;
[secondary-key-1 [DUPS] ] [secondary-key-19 [DUPS] ]
```

Format-4 TAPE:

```
DEFINE F[F] filename TAPE record-length,block-size [RECORDS | CHARACTERS] ;
[ANSI-format [UNIVAC | IBM] ]
```

Format-5 ISAM:

```
DEFINE F[F] filename ISAM [RECORDS | CHARACTERS] actual-key-variable
```

DEFINE may be abbreviated **DEF**; **RECS** or **RECORD** for RECORDS; and **CHARS** for CHARACTERS.

If DEFINE F is used, the *record-length* is based on ASCII characters. If DEFINE FF is used, the *record-length* is in FIELDATA characters.

SEQ files may be externally assigned to either tape or disk. If assigned to tape, they will be in SDF format.

DIRECT, INDEXED (MSAM) and ISAM files must be assigned to disk.

TAPE files must be assigned to tape.

For DIRECT and INDEXED files, the optional SHARE parameter allows these file types to be accessed by SFS 2200 under the control of the Universal Data System (UDS). SFS 2200 files are "shared" files that provide a site the same recovery, locking and queuing features

used for other file systems under UDS Control. SFS 2200 files must have File Description Tables (FDTs) created by the UDS Unisys Repository Manager (UREP). See "Defining and Maintaining Storage Areas" in the Repository for ClearPath OS 2200 Administration Guide (7830 8087), or contact the appropriate data processing support personnel at your site for assistance. To access SFS 2200 files, the Q-LINK program must first perform an explicit (single) BEGIN THREAD prior to the file OPEN. Use the RDMS command to perform the BEGIN THREAD (see Section 14.1, "RDMS Command"). Also, use the RDMS command to perform the END THREAD at the end of the program.

The *record-length* parameter specifies the number of characters per record. The *block-size* can be specified as the number of records or characters per block. If the optional RECORDS/CHARACTERS keyword is not specified, Q-LINK will default to records per block. If block-size is zero (0), Q-LINK will determine the block size by reading the physical record header. The zero specification is only valid for files that are opened as INPUT, UPDATE or EXTEND.

The *relative-key-variable* and *maximum-records-in-file* parameters apply only to DIRECT files. The variable named must be numeric. The *maximum-records-in-file* parameter is a numeric integer literal. It is used to initialize all records in a DIRECT file when it is opened for OUTPUT or EXTEND. It is ignored when the file is opened as INPUT or UPDATE.

For INDEXED files, the *record-key* and *secondary-key-n* are given as RDA references, either direct or by data item name. When defining an INDEXED file to be opened for INPUT, UPDATE or EXTEND, all keys must be specified in the same order and for the same positions as when the file was originally created. The DUPS clause indicates a secondary key that has duplicates allowed. A maximum of 19 secondary keys may be specified.

For files defined as tape, the following ANSI-formats are supported in Q-LINK:

ANSII Format	Meaning
U	Undefined
F	Fixed
V	Variable
FB	Fixed Blocked
VB	Variable Blocked (Default)

If UNIVAC is specified for an ANSI TAPE file, the record length will be internally adjusted to the next multiple of four (4) characters for even word counts. If IBM is specified, the character length will be as specified. The default is UNIVAC.

The *actual-key-variable* applies only to ISAM files (older version single-key indexed sequential) and must be defined as alphanumeric. The length of the *actual-key-variable* cannot exceed 1024 characters.

Most programming shops use MSAM as opposed to the old ISAM format. The programming techniques required to handle ISAM are the same as that required for MSAM; however, multiple key access is not supported for ISAM. If there is a question as to whether a file created in a COBOL program is MSAM or ISAM, look for the ASSIGN clause on the SELECT statement in the COBOL ENVIRONMENT DIVISION. If it is assigned to DISC, it is MSAM; if assigned to MASS-STORAGE, it is ISAM. The ISAM key cannot be specified as an RDA reference in the record. A user-defined variable must be specified.

Examples:

1. DEFINE F SAVE-FILE SEQ 100 50
2. DEFINE F DIR-FILE DIRECT 80 2500 REL-KEY
3. DEFINE F MSAM-FILE INDEXED 122 24 (1,4) (9,8) DUPS
4. DEFINE F ISAM-FILE INDEXED 50 20 PROD-NUM
5. DEFINE F OUTFIL TAPE 334 3584 CHARACTERS
6. DEFINE F IBMTAPE TAPE 90 22 FB IBM
7. DEFINE F DIR-SFS DIRECT SHARED 875 5000 REC-NO
8. DEFINE F MSAM-SFS INDEXED SHARED 122,0 (1,4) KEY2 DUPS

Explanation:

The file in Example 1 is a PCIOS sequential file with 100-character records, blocked 50 records.

Example 2 defines a PCIOS DIRECT (or relative) file. Records contain 80 characters. The file will be initialized to 2500 records if opened for OUTPUT. The variable, REL-KEY, must be set to the relative record number for read, write and delete operations.

Example 3 is an INDEXED file with a primary and one secondary key. The primary key is 4 characters starting in position 1, and the secondary key is 8 characters starting in position 9. Each record contains 122 characters, blocked 14.

Example 4 is an INDEXED file with only a primary key. The key is specified by its name, requiring a data item index file. The records are 50 characters, blocked 20.

Example 5 defines a PCIOS ANSI tape file with 334 character records, blocked 3,584 characters.

Example 6 is an ANSI tape file in IBM fixed blocked format.

Example 7 defines a shared relative I/O file under SFS 2200. The file will contain a maximum of 5000 records, which are each 875 characters in length. The relative key variable used to access the records is called REC-NO.

Example 8 sets up an MSAM shared file under SFS 2200. Each record contains 122 characters. The zero block size indicates that the file's blocking factor will be determined from the file header when the file is OPENed for INPUT or UPDATE. The primary key begins in the first position of the record and extends for four characters. The secondary key is named KEY2, and duplicates are allowed.

10.3 CDELETE

The CDELETE command may be used on any file type. It is used to delete the record just read. The file must be opened for update.

Format:

CDELETE *filename* [INVALID KEY *program-label* | STATUS *status-variable*]

CDELETE may be abbreviated **CD**.

The INVALID KEY clause only applies to DIRECT, INDEXED (MSAM) and ISAM files. The STATUS clause may be used in lieu of the INVALID KEY clause to receive the PCIOS/SFS error status from the PCIOS Processor Interface Module (PIM) or from SFS 2200. The *status-variable* must be defined as a numeric variable. The INVALID KEY and STATUS clauses are optional in conversational mode.

Example:

```
SET REL-KEY = 22 . Set RELATIVE key
READ REL-FILE INVALID KEY NO-FIND . Read DIRECT rec.
CDELETE REL-FILE INVALID KEY ERROR . Delete record.
```


10.4 CLOSE

The CLOSE command is used to close a PCIOS/SFS file. If the file is opened for OUTPUT, end of file processing will take place. Disk files will not automatically be @FREEed; tape files will automatically rewind unless the NO REWIND option is used. Files may be closed or re-opened with different usage and access modes.

Format:

CLOSE *filename* [[WITH] NO REWIND]

CLOSE may be abbreviated **CL**.

10.5 OPEN

The OPEN command is used to ready a file for use in a program. It establishes the usage and access modes for the file. The file must be assigned to the run before the OPEN is executed.

Format:

```
OPEN filename {INPUT | OUTPUT | UPDATE | EXTEND} ;
    {SEQ | DIRECT | RANDOM | DYNAMIC} [ [WITH] NO REWIND] ;
    [number-of-buffers-literal [BUFFERS] ]
```

OPEN may be abbreviated **O**, with **I** for INPUT, **O** for OUTPUT and **U** for UPDATE.

INPUT, OUTPUT, UPDATE and EXTEND are usage modes and have the following meanings:

INPUT	The file will only be read.
OUTPUT	The file will only be written. Any existing data will be destroyed.
UPDATE	The file may be read and written.
EXTEND	The file will only be written to. All data will be written after the last data block found when the file is opened.

All file types, except TAPE, may be opened for INPUT, OUTPUT or UPDATE. TAPE files may not be opened for UPDATE. SEQ, TAPE and DIRECT files may be opened for EXTEND.

SEQ, DIRECT, RANDOM and DYNAMIC access modes have the following meanings:

SEQ	The file will only be accessed sequentially.
DIRECT	The file will only be accessed directly by relative record position. File type must be DIRECT.
RANDOM	The file will only be accessed randomly by primary or secondary record key. File type must be INDEXED or ISAM.
DYNAMIC	The file may be accessed both sequentially and randomly. File type must be INDEXED or ISAM.

SEQ files may only be accessed sequentially. Files defined as DIRECT may only be opened for DIRECT access. If a DIRECT file is to be accessed sequentially, it may be defined as a SEQ file with a block factor of one. INDEXED files may be accessed either SEQ, RANDOM or DYNAMIC.

The WITH NO REWIND clause can be used with files DEFINED as TAPE or SEQuential files assigned to tape for the purpose of stacking files (i.e., multiple file volumes).

The *number-of-buffers-literal* can only be used for PCIOS INDEXED (MSAM) files. It provides a means of increasing the number of I/O buffers used by Q-LINK. If the number of buffers is not specified, Q-LINK will calculate the number of buffers based on the following formula:

Number of buffers = 4 + the total number of keys

MSAM performance may be improved by allocating additional buffers at the expense of additional memory requirements. If buffer allocation exceeds memory available for allocation, a runtime error occurs. This runtime error may occur while executing a Q-LINK program with many open files, large file blocks, or an excessive number of buffers allocated. If this error occurs, buffer allocation should be reduced, or unused files closed. If less than five buffers are specified, Q-LINK will allocate five buffers; if more than 24 buffers are specified, Q-LINK will allocate 24 buffers.

Examples:

1. Open to read a file sequentially:
`OPEN TX-FILE INPUT SEQ`
2. Open to create a new file sequentially:
`OPEN NEW-MAST OUTPUT SEQ`
3. Open to create a new MSAM file in random key order:
`OPEN BOND-MAST OUTPUT RANDOM`
4. Open to read an MSAM file randomly or sequentially:
`OPEN DEPARTMENT INPUT DYNAMIC`
5. Open MSAM file to update existing records accessed by key:
`OPEN EMP-MAST UPDATE RANDOM`
6. Open to create new file by relative record number:
`OPEN TAG-FILE OUTPUT DIRECT`
7. Open to append records to a file:
`OPEN LOG-DUMP EXTEND SEQ`

10.6 READ

The READ command is used to read a standard file. The file must have been previously opened. The AT END label must be specified when not in conversational mode.

Format:

```
READ filename [AT END {program-label | BREAK} | ;
                INVALID KEY program-label | ;
                STATUS status-variable] ;
[USING KEY key-number-literal]
```

The AT END clause only applies to sequential reads. The INVALID KEY clause only applies to DIRECT, INDEXED and ISAM files. The STATUS clause may be used in lieu of the AT END or INVALID KEY clauses to receive the PCIOS/SFS error status from the PCIOS Processor Interface Module (PIM) or from SFS 2200. The *status-variable* must be defined as a numeric variable. The AT END, INVALID KEY and STATUS clauses are optional in conversational mode.

BREAK may only be used instead of a label when the READ is contained within a defined procedure or an in-line DO block.

The USING KEY clause applies to MSAM files when accessing by an alternate record key. The *key-number-literal* specifies which alternate key to use from the file's definition (example: 1 = the first secondary key defined after the primary key; 2 = the second, etc.).

Following the READ of a SEQ file, the reserved numeric variable REC\$LEN will contain the actual length of the record read in words. This variable may be useful in processing variable length data records.

If the file is TAPE, REC\$LEN will contain the record length in characters.

Examples:

SEQuential File:

```
READ SEQ-IN-FILE AT END END-OF-RUN
```

DIRECT File:

```
SET DIR-KEY = 20 . Relative key = 20.
READ DIR-FILE INVALID KEY NOFIND . Read DIRECT
GO FOUND
```

INDEXED File (MSAM) - Primary key:

```
SET RDA PART-NUMBER = '12-994 AB' . Set Key in RDA
READ PARTMSTR INVALID KEY NO-FIND . Read Record
```

INDEXED File (MSAM) - Secondary key:

File defined (with primary key plus two secondary keys) as follows:

```
DEFINE F XMAST INDEXED 120,1 (1,5) (6,8) DUPS (14,3) DUPS
...
RDA (14,3) = SEARCH-VALUE
READ XMAST INVALID KEY ERROR-1 ;
    USING KEY 2 . Key (14,3)
```

10.7 READNEXT

The READNEXT command applies only to INDEXED (MSAM) or ISAM files opened as INPUT or UPDATE with an access mode of SEQUENTIAL or DYNAMIC. READNEXT is used to read the next logical record in the file. This command must follow a READ, START or a READNEXT.

Format:

READNEXT *filename* [AT END {*program-label* | **BREAK**} | STATUS *status-variable*]

READNEXT may be abbreviated **READN**.

The STATUS clause may be used in lieu of the AT END clauses to receive the PCIOS/SFS error status from the PCIOS Processor Interface Module (PIM) or SFS 2200. The *status-variable* must be defined as a numeric variable. The AT END and STATUS clauses are optional in conversational mode.

BREAK may only be used when the READNEXT is contained within a defined procedure or an in-line DO block.

See the START command for usage examples.

10.8 REWRITE

The REWRITE command may be used with any file type. The command is used to write the last record read, back to the same file location from which it was read. The file must be opened for update, and the number of words written must be the same as the original record length.

Format:

```
REWRITE filename [length-literal | length-variable] ;  
[INVALID KEY program-label | STATUS status-variable]
```

REWRITE may be abbreviated **REW**.

The *length-literal* or *length-variable* entry is optional. If omitted, the defined record length for the file will be written.

The INVALID KEY clause only applies to DIRECT, INDEXED (MSAM) and ISAM files. The STATUS clause may be used in lieu of the INVALID KEY clause to receive the PCIOS/SFS error status from the PCIOS Processor Interface Module (PIM) or from SFS 2200. The *length-variable* and *status-variable* must be defined as numeric variables. The INVALID KEY and STATUS clauses are optional in conversational mode.

Example:

```
READ WORK-FILE AT END END-RUN  
RDA (1,5) = 'ABCDE'           . Change the record.  
REWRITE WORK-FILE             . Rewrite it.
```

10.9 START

The START command applies only to INDEXED (MSAM) and ISAM files opened for INPUT or UPDATE with an access mode of DYNAMIC. START is used to position to a point in the file. The user may position to a specific key or a point just following a specified key. The START command does not read a record into the RDA. To read the record into the RDA, use the READNEXT command.

Format:

```
START filename [INVALID KEY {program-label | BREAK} | ;
               STATUS status-variable] ;
[USING KEY key-number-literal] [EQ | GT | EQGT]
```

START may be abbreviated ST.

The INVALID KEY clause only applies to INDEXED (MSAM) and ISAM files. BREAK may only be used instead of a label when the START is contained within a defined procedure or an in-line DO block. The STATUS clause may be used in lieu of the INVALID KEY clause to receive the PCIOS/SFS error status from the PCIOS Processor Interface Module (PIM) or from SFS 2200. The *status-variable* must be defined as a numeric variable. The INVALID KEY and STATUS clauses are optional in conversational mode.

BREAK may only be used instead of a label when the START is contained within a defined procedure or an in-line DO block.

The USING KEY clause is used to specify positioning on a secondary key, and therefore, only applies to INDEXED (MSAM) files. The *key-number-literal* corresponds to the position of the secondary key in the DEFINE for the file.

EQ, GT or EQGT are used to determine where to position in the file in relation to the key supplied in the RDA. If not present, EQ is assumed. EQ indicates that a record with a matching key must be found; otherwise, an invalid key condition will result. GT indicates that positioning to the next key following the given record key is to be used. If no higher key exists, an invalid key condition results. EQGT indicates to position to an equal key or to the next greater key in the file. If no equal condition or higher key exists, an invalid key error results.

Example:

The following directive defines an MSAM file with one secondary key:

```
DEFINE F MSTR-FILE INDEXED 100 1 MSTR-NUM MSTR-SCNDRY-KEY
...
.   *** Position based on a secondary key.
.   *** Position to record greater than the letter 'A'.
RDA MSTR-SCNDRY-KEY = 'A'
START MSTR-FILE INVALID KEY ERR2 USING KEY 1 GT
.   *** Read the record.
READNEXT MSTR-FILE AT END END-RUN
...
.   Position based on primary key.
RDA MSTR-NUM = 100000
START MSTR-FILE INVALID KEY ERR1 EQ
READNEXT MSTR-FILE AT END CLOSE-OUT .   Read the record.
```

10.10 WRITE

The WRITE command is used to write a specified number of words or characters (depending on file type) from the RDA to a file. The file may be any standard format.

For standard PCIOS files, the length of the record is specified in characters. If the number of characters to write is omitted, I Q-LINK will assume the maximum record length specified in the file definition.

Format:

```
WRITE filename [length-literal | length-variable] ;  
[INVALID KEY program-label | STATUS status-variable]
```

WRITE may be abbreviated **W**.

The *length-literal* or *length-variable* entry is optional. If omitted, the defined record length for the file (as specified on the DEFINE F directive) will be written.

The INVALID KEY clause only applies to DIRECT, INDEXED (MSAM) and ISAM files. The STATUS clause may be used in lieu of the INVALID KEY clause to receive the PCIOS/SFS error status from the PCIOS Processor Interface Module (PIM) or SFS 2200. The *length-variable* and *status-variable* must be defined as numeric variables. The INVALID KEY and STATUS clauses are optional in conversational mode.

Examples:

```
DIR-KEY = DIR-KEY + 1           . Set to write next record.  
WRITE DIR-FILE INVALID KEY ERR-1 . Write the record.  
  
. *** Write a variable length INDEXED record...  
R-LEN = SUB1 * 45              . Calc length of occurs items.  
R-LEN = R-LEN + 32             . Add on fixed part of rec.  
WRITE PAYMAST R-LEN INVALID KEY KEY-ERROR
```


10.11 Special PCIOS Status Returned

The following status codes can be returned from I Q-LINK:

Status	Meaning
---------------	----------------

000000	Normal status returned.
900000	User has reached end-of-file.
900001	File is not assigned. Assign the file with a CSF command.
900002	Not enough memory to allocate file buffers. Use a smaller block size or open fewer files at one time.
900003	Invalid key returned on READ, START, REWRITE or WRITE command.
900004	User specified a zero (0) block size and block size information not found in file header. The file is probably not a PCIOS file.

Chapter 11: SORT Interface

This group of commands gives the user the ability to SORT records within a Q-LINK program. Records are passed to the SORT and returned from the SORT using the RDA. The SORT interface routine will require SORT work files to be assigned (XA, XB, through XZ are possible). The size of these files, and the number of SORT work files, must be determined by the user depending on the number and size of records to be SORTed and the core SORT buffer size. (Refer to the SORT manual). Q-LINK will dynamically assign the XA, XB and XC files as follows:

```
@ASG,T XA.,///5000
@ASG,T XB.,///5000
@ASG,T XC.,///5000
```

11.1 RELEASE

This command RELEASEs a specified number of characters from the RDA to the SORT. The number of characters RELEASEd may be specified as an integer value or as a database record name. If a record name and an integer value are specified, the length of the named DMS 2200 record plus the integer value will be RELEASEd. The RELEASE may be given as RELEASEF, in which case the *integer-characters* parameter will be assumed to be FIELDATA.

Format:

RELEASE[F] {*record-name* [*integer-characters*] | *integer-characters*}

RELEASE may be abbreviated **REL**.

Examples:

```
.   *** Release using the length of the PRODUCT-RECORD
.   *** plus 20 additional characters.
RELEASE PRODUCT-RECORD 20
...

.   *** Release a record of 35 FIELDATA characters in
.   *** length.
RELEASEF 35
```

11.2 RETURN

This command is used to RETURN records from the SORT subroutine to the RDA. The first execution of the RETURN will cause the termination of the SORT release phase. The record will be placed in the RDA exactly as released. The AT END clause is required.

Format:

RETURN AT END {*program-label* | **BREAK**}

RETURN may be abbreviated **RET**, with **B** for BREAK.

BREAK may only be used instead of a label when the RETURN is contained within a defined procedure or an in-line DO block.

Example:

```
SORT-OUT PROCEDURE
  RETURN AT END BREAK          .  Return a record
  DISPLAY RDA PART-NAME +      .  list data from
  DISPLAY '-' +                 .  the record area
  DISPLAY RDA PART-NUM
  ...
ENDPROC
```

11.3 SORT

The SORT command is used to initialize the SORT interface. It is used to specify the length of records to be SORTed, and to define the SORT KEYS. This command must be used each time a new SORT is to be done. A maximum of ten SORT KEYS may be specified. The SORT command may be given as either SORT or SORTF. If SORT is used, the maximum/minimum record lengths are assumed ASCII characters, while SORTF indicates FIELDATA characters. The representation code specified affects the calculation used by Q-LINK in determining the record size in words.

Format:

```
SORT[F] maximum-record-length minimum-record-length ;
      key-definition-1 [ ... key-definition-10 ] [( core-buffer-size )]
```

SORT may be abbreviated **SOR**.

The *minimum-record-length* field indicates the shortest record being SORTed, when SORTing variable length records. If records are fixed length, *maximum-record-length* and *minimum-record-length* must be equal. All SORT KEYS must fall within the shortest record length.

The format of *key-definition-n* is:

```
start-character,key-byte-length,data-type,sequence-specifier
```

Allowed *data-types* are the same as those used in direct RDA reference. *Start-char* is relative to the first character of the SORT area. The SORT area may be defined by the DEFINE RA SORT directive. If the DEFINE RA SORT directive is not used, the SORT area begins in position one (1) of the RDA. See Section 3.1.1 for all possible data types.

Allowed *sequence-specifier* are:

```
A for ascending,
D for descending.
```

The *core-buffer-size* parameter is optional. It is used to change the default SORT buffer from 22.5K to the size required, depending on the SORT volume. Currently the maximum sort buffer size allowed is 40,000. See the Unisys SORT manual for SORT file and buffer sizing.

Examples:

The following initializes the SORT interface to SORT 80-character records on a 9(10) COMP field starting in position 1, and an X(20) field starting in position 5.

```
SORT 80 80 1,4,UB9,A 5,20,DISP,D
```

This command sets up a SORT of variable length FIELDATA records on positions 2 through 7 alphanumeric ascending. A 40,000-word sort buffer will be allocated.

```
SORTF 100 20 2,6,DISP-1,A (40000)
```

Chapter 12: DMS 2200 Interface

The following commands are the Q-LINK equivalent of standard ASCII COBOL/DML statements. Q-LINK rules for record selection and database currency are the same as in standard DML. For a complete explanation of the function of DML commands, refer to the DMS 2200 ASCII COBOL/DML Manual.

12.1 Subschema Invocation (INVOKE)

As with any COBOL/DML program, before database access can begin, a DMS 2200 subschema must be invoked. Unlike COBOL/DML, Q-LINK is able to invoke a subschema dynamically at runtime by using the INVOKE directive. The INVOKE directive must be issued prior to the DML IMPART command. INVOKE will access the object schema and subschema to initialize Q-LINK's S\$WORK and D\$WORK. Upon INVOKE, the Q-LINK Primary Data Item Index file will be assigned, initialized, and built automatically to be used in DML command editing and decoding, and in RDA item name reference decoding. The INVOKE directive may be used to switch from one subschema to another so long as the current subschema is not IMPARTed.

Format:

```
INVOKE subschema [ {OF | IN} subschema] ;
  [ {FILE qualifier*filename | TIP [FILE] file-code} [FOR DMR-name] ;
  [ || APPLICATION application-number | ;
    KEY IS INVOKE-key | ;
    POINTER AREA LOAD | ;
    {NX | INDEX} || ]
```

INVOKE may be abbreviated **INV**, with **APPL** for APPLICATION.

The *subschema* specification may reference an ACOB-generated or UCOB-generated subschema.

The OF/IN schema parameter is optional only when a default schema is configured for a particular DMR or LDM (see the note, below).

The FILE/TIP clause is optional only when a default schema file is configured for a particular DMR.

Please note: With DMS 2200, level 9R1 or higher, the term, "DMR", is no longer used. Instead, Unisys documentation now refers to a logical data manager (LDM) for DMS running under Universal Data System (UDS) Control. Q-LINK can interface with any number of LDMs through UDS Control. This documentation, however, will continue to use the term, "DMR", in a generic fashion.

The FOR *DMR-name* parameter is also optional. The *DMR-name* refers to the multi-thread invoke name specified for a particular "DMR" during dynamic configuration of Q-LINK through COMUS. If it is not specified on the INVOKE directive, the first multi-thread DMR

configured will be used. The security feature allows the configuration of additional DMR default values.

The optional APPLICATION *application-number* parameter allows Q-LINK to explicitly connect to an application group upon INVOKE. This parameter may be used when invoking from a TIP schema file.

The EXEC will not allow a program to connect to two application groups within the same program execution. You must exit Q-LINK after utilizing an application group before attempting to access a different application group. If you fail to exit before attempting access to a different application group, DMS will return a rollback error 94 and display a console message indicating that a STEP CONTROL error 067 was encountered.

The INVOKE-key literal, on the KEY IS parameter, is the LOCK FOR INVOKE literal specified in the subschema. It is only required when coded in the subschema.

The POINTER AREA LOAD parameter is only required when initially loading pointer array areas.

The optional NX/INDEX parameter controls the building of the primary data item index file on INVOKE. If the NX option is used, the Q-LINK primary data index file will be built without data items. The NX option may be used if a permanent data item index file (previously built by the QINDEX processor) is to be used instead of the dynamically built primary index. The INDEX option will cause the primary data item index file to be built with data items. If this parameter is not specified, the default, as dynamically configured through COMUS, will be used.

For information on creating a permanent data item index file, see Chapter 17, "QINDEX Reference".

Note: If alternate records delivery areas are to be defined (DEFINE RA) for DMS records, they should be specified after INVOKE but prior to IMPART.

Examples:

1. INVOKE ACCTS OF FINANCE TIP FILE 77 NX
INDEX DMS*DB-INDEX
2. INVOKE ACCTS OF FINANCE FILE TST*SCHFILE FOR STHREAD
3. INVOKE TRIV OF FINANCE FILE DMS*SCHFILE KEY IS 'FTRV'

Explanations:

In the first example, a TIP object schema file will be used. In addition, no data item entries will be created in the primary data item index (NX option). A secondary data item index will be required to reference data items by name in this case (INDEX directive). In addition, the DMR name is not given, implying that the INVOKE will default to the first multi-thread DMR configured.

In the second example, the object subschema and schema will be obtained from the EXEC schema file, "TST*SCHFILE", and is being used with the "STHREAD" DMR.

In the last example, the "TRIV" subschema requires an INVOKE key, "FTRV", for processing.

12.2 INVOKE Considerations

The INVOKE accesses the object schema and subschema specified by the user. Using the object schema and subschema tables, the INVOKE performs two major tasks: initialization of D\$WORK and S\$WORK, and creation of the Primary Data Item Index File. In a COBOL program, D\$WORK is created by the ASCII Data Manipulation Language Pre-Processor (ADMLP), while S\$WORK is created when the Subschema Data Definition Language (SDDL) is compiled. Both of these are collected into the object program by the MAP processor. Q-LINK creates both of these dynamically when the INVOKE directive is processed.

The Primary Data Item Index File is used by Q-LINK only during the editing of commands. It contains definitions of areas, records, sets, database data names and data items from the invoked subschema. This allows the processor to translate names entered by the user to codes or RDA references to be used during actual execution of commands.

The D\$WORK and S\$WORK areas are in all DMS 2200 programs, including Q-LINK. They are used in communications between the program and the data management routine (DMR). D\$WORK contains lists of WORKING and/or COMMON STORAGE addresses of each record and database data name in the user program. This is how the

DMR knows where to find records and database data names in your program. S\$WORK is where the DMR maintains information on run unit, area, record and set currency. S\$WORK is also used as workspace for many DMR operations.

S\$WORK also contains the name of the schema and subschema, and the file from which the DMR must obtain its copy of the schema and subschema absolutes. This is a very important and often misunderstood point. When Q-LINK processes the INVOKE, the subschema file name is copied to S\$WORK directly from the subschema absolute, not from the file named on the INVOKE directive. Upon IMPARTing, the DMR uses the file name found in S\$WORK to find the subschema. The DMR then uses the schema file named in the subschema tables to find the object schema, which is why Q-LINK cannot simply move the file name on the INVOKE directive to S\$WORK. The DMR would still look for the schema in the schema file named in the subschema tables. It is very important to insure that the subschema and schema INVOKEd are the same as what the DMR will use. In a production application environment, it is best to INVOKE from the production on-line schema file.

12.3 DML Commands

Most commands available in standard COBOL/DML are also available in Q-LINK. The ON ERROR and AT END clauses are not included in Q-LINK DML syntax. The Q-LINK procedural IF command is used to determine these conditions.

Unless otherwise stated, area-name, record-name and set-name may be given as a specific area, record or set name, or as alphanumeric variables that contain a valid area, record or set name (known as generalized names in DMS 2200 DML terminology). When an alphabetic variable is used in place of an actual name, the variable length must be specified as follows:

area-name	12 characters
record-name	30 characters
set-name	30 characters

Q-LINK may be dynamically configured in COMUS to allow abbreviation of AREA, RECORD and SET. If this option is taken, the user may enter either the first "N" characters of the name, or the entire name, in all cases except where the name is used as a literal enclosed in quotes. "N" characters will vary depending on the number of characters required to identify the AREA, RECORD, or SET. For example, if all record names begin with the letter "R" and the three-digit record code (example: R0002-PARTS-MASTER-CONTROL), Q-LINK may be configured to recognize the record name when only the first five characters have been entered (example: R0002).

12.4 ACQUIRE

The ACQUIRE command is used to retrieve a list of database keys for a specified set whose mode is pointer array or indexed pointer array. The ACQUIRE command functions in the same manner as the standard COBOL/DML ACQUIRE command. The only difference is the Q-LINK acquire list is specified as a location within the RDA rather than a COBOL WORKING-STORAGE location. The acquire list in the RDA is in the same format as in a COBOL/DML program. The first word of the specified RDA location contains the number of DBKs retrieved followed by the list of DBKs.

Format:

```
ACQUIRE number-of-keys [BEGINNING database-key] ;
      TO RDA RDA-reference FROM set-name [USING DEFINED KEYS]
```

ACQUIRE may be abbreviated **ACQ**.

The *RDA reference* must be a multiple of four (4) because each database key will occupy one word. Each DBK may be referenced using the RDA reference "(n,4) UB9" (n = start position).

Example:

The following is an example of the ACQUIRE command used to obtain two lists of database keys which will then be matched for retrieval of records that participate in both pointer array sets.

```
DEF N DBKVAR . Number of DBKs
DEF RDA DBKLIST1 (1001,400) . A 100-word ACQUIRE list
DEF RDA DBK1 (1001,4) UB9 . A DBK in ACQUIRE list 1
DEF RDA DBKLIST2 (1501,400) . Another ACQUIRE list
DEF RDA DBK2 (1501,4) UB9 . A DBK in ACQUIRE list 2
DEF SUB X1 DBK1 . Subscript for list 1
DEF SUB X2 DBK2 . Subscript for list 2
...
. *** Owner of sets S0002-PART-USE and S0003-PART-MOD
. have been made made current of set. Now
. ACQUIRE lists of DBKs for both sets.
RDA DBKLIST1 = $LOVALS . Clear list of binary zeros
RDA DBKLIST2 = $LOVALS
DBKVAR = 50 . Set to retrieve 50 keys
ACQUIRE DBKVAR TO RDA DBKLIST1 FROM S0002-PART-USE
ACQUIRE DBKVAR TO RDA DBKLIST2 FROM S0003-PART-MOD
DO DBK-MATCH
...
. *** This proc matches DBKs from both ACQUIRE lists.
. When a DBK is found in both lists, the
. FETCH-IT procedure is executed.
DBK-MATCH PROCEDURE
IF DBK1 = 0 . If either list is empty
OR DBK2 = 0 . or no rec meets criteria
GO DM-EXIT . EXIT this routine
ENDIF
X1 = 1 . 1stDBKinlist1
DM005
X2 = 1 . 1st DBK in list 2
Z = RDA DBK1 :X1 . Isolate 1 DBK from list 1
```

```

DM010
  IF RDA DBK2 :X2 = Z           .  Is it in list 2?
    DO FETCH-IT                 .  In both lists, FETCH it.
  ENDIF
  X2 = X2 + 1                   .  Increment list 2 subscript
  IF RDA DBK2 :X2 = $LOVALS     .  End-of-list-2?
    X1 = X1 + 1                 .  Increment list 1 subscript
    IF RDA DBK1 :X1 = $LOVALS   .  End-of-list-1?
      GO DMEXIT                 .  Yes, done.....
    ENDIF
    GO DM005                     .  Repeat match
  ENDIF
  GO DM010                       .  Match next entry
DMEXIT
ENDPROC
...
FETCH-IT PROCEDURE
  FETCH1 Z                       .  FETCH member using DBK
  DISPLAY RDA .....
  ...
ENDPROC

```

12.5 CLOSE

The CLOSE command releases database areas and associated quick-look files. The command may be used to close a single area or all areas currently open.

Format:

`CLOSE {ALL | area-name}`

CLOSE may be abbreviated `CL`.

12.6 DELETE

The DELETE command is used to delete a record from the database. If ONLY is specified, the record will not be deleted if it is the owner of any VIA-SET records. If ALL is specified, the record and all members of any sets of which it is the owner will be deleted. The ONLY and ALL options have the same meaning as in standard COBOL/DML.

Format:

DELETE *record-name* [ALL | ONLY]

DELETE may be abbreviated **DEL**.

Examples:

```
FETCH4 FIRST PRODUCT-REC PRODUCTS AREA
DELETE PRODUCT-REC ONLY          .  Delete, if no members
...
FETCH3 NEXT PRODUCT-REC PRODUCTS AREA
DELETE PRODUCT-REC ALL           .  Delete it & all members.
```

12.7 DEPART

The DEPART is used to disconnect Q-LINK from the DMR. The ROLLBACK option may be used to negate any updates that were done during the session.

Format:

DEPART [ROLLBACK]

DEPART may be abbreviated DEP.

12.8 DISPLAY Database Error

This form of DISPLAY will print a fixed format DML error status. When executed, it will not disturb the current contents of the print buffer.

Format:

DISPLAY DBERROR

DISPLAY may be abbreviated D, with ERROR for DBERROR.

12.9 FETCH/FIND

Q-LINK supports all formats of the FETCH/FIND command. In the following descriptions, the word FETCH will be used in all formats. The same formats will apply to the FIND command.

12.9.1 FETCH/FIND Format-1

The FETCH Format 1 is used to obtain any record directly instead of by its specific location mode. A Format 1 FETCH can be executed in one of two ways:

1. A numeric variable containing a database key can be set;
2. An alpha variable containing an area-name and a numeric variable containing an area-key can be set.

The record-name is optional on the Format 1 FETCH. If used, it must be given as an actual record name, not a variable.

Format:

```
{FETCH1 | FIND1} [record-name] ;
    {database-key-variable | area-name-variable,area-key-variable}
    [SUPPRESS-clause]
```

FETCH1 and FIND1 may be abbreviated **F1** and **FN1**, respectively.

The format of the suppress clause can be found in Section 12.22 at the end of this chapter.

Examples:

```
DEFINE N HOLDDBK . To hold database key.
    HOLDDBK = RDA DBK-OF-LINE-ITEM . Get DBK
    FETCH1 HOLDDBK . Fetch the line item.
    ...
    G-AREA-NAME = 'PRODUCTS' . Set area-name.
    G-AKEY = (1,1) . Set area-key to page 1
    . and record 1.

FIND1 PRODUCT G-AREA-NAME G-AKEY
```

12.9.2 FETCH/FIND Format-2

The FETCH Format 2 is used to make a record current of run unit. The record must be current of that record type (i.e., previously accessed).

Format:

```
{FETCH2 | FIND2} [record-name] ;  
    {database-key-area | area-name-variable,area-key-variable} ;  
    [suppress-clause]
```

FETCH2 and FIND2 may be abbreviated **F2** and **FN2**, respectively.

The format of the suppress clause can be found in Section 12.22 at the end of this chapter.

12.9.3 FETCH/FIND Format-3

The FETCH Format 3 is used to retrieve unspecified record types within sets or areas using set or area currency.

Format:

```
{FETCH3 | FIND3} {CURRENT | NEXT | PRIOR | FIRST | LAST | OWNER} ;  
  {set-name SET | area-name AREA} ;  
  [SUPPRESS-clause]
```

FETCH3 and FIND3 may be abbreviated F3 and FN3, respectively.

The format of the suppress clause can be found in Section 12.22 at the end of this chapter.

Examples:

```
FETCH3 LAST PRODUCTS AREA
```

```
FIND3 CURRENT PROD-USAGE SET
```

12.9.4 FETCH/FIND Format-4

The FETCH Format 4 is similar to the FETCH Format 3, except that a specific record type is to be selected.

Format:

```
{FETCH4 | FIND4} {NEXT | PRIOR | FIRST | LAST} record-name ;  
  {set-name SET | area-name AREA} ;  
  [SUPPRESS-clause]
```

FETCH4 and FIND4 may be abbreviated F4 and FN4, respectively.

The format of the suppress clause can be found in Section 12.22 at the end of this chapter.

Examples:

```
FETCH4 FIRST PRODUCT-REC PRODUCTS AREA
```

```
FIND4 LAST ASSEMBLY-REC PARTS-USAGE SET
```

12.9.5 FETCH/FIND Format-5

The FETCH Format 5 is used to retrieve a record using its primary location mode. Before a Format 5 FETCH is executed, it may be necessary to set database data names specified in the schema using the SET DBDN command. If a record key is to be initialized, it may be set into the RDA using the SET RDA command.

Format:

```
{FETCH5 | FIND5} [NEXT [DUPLICATE] | PRIOR | FIRST | LAST] record-name ;
    [SUPPRESS-clause]
```

FETCH5 and FIND5 may be abbreviated F5 and FN5, respectively.

The format of the suppress clause can be found in Section 12.22 at the end of this chapter.

Examples:

```
DBDN PROD-AREA = 'PRODUCTS'
RDA PRODUCT-ID = 101100          . Set record's key.
FETCH5 PRODUCT-REC              . Fetch the record.
FIND5 NEXT DUPLICATE PRODUCT-REC . Find dup., if any.
...
DBDN DIR-AREA = 'DIRECT-AREA'   . Set area name
DBDN DIR-KEY = 5,1              . Set area key
FETCH5 DIRECT-REC              . Fetch direct rec.
```

12.9.6 FETCH/FIND Format-6

The FETCH Format 6 is used to retrieve a record via a specific set. If the optional USING clause is included, the record will be selected based on the values of the data items specified.

Format:

```
{FETCH6 | FIND6} record-name set-name ;  
  [USING data-item-1 [... ,data-item-n] ] ;  
  [SUPPRESS-clause]
```

FETCH6 and FIND6 may be abbreviated **F6** and **FN6**, respectively.

Where *data-items* may be one or more data field names defined within the object record. A *data-item* must be defined in the schema and may not be qualified.

The format of the suppress clause can be found in Section 12.22 at the end of this chapter.

Example:

```
RDA PROD-TYPE = 'T'  
RDA PROD-CAT = '2A'  
FETCH6 PROD-DET PROD-DET-SET USING PROD-TYPE PROD-CAT
```

12.9.7 FETCH/FIND Format-7

The FETCH Format 7 is generally used following positioning within a set to find a record with matching values on the specified data items. The *record-name* is optional. If used, it must be current of run unit and cannot be given as a variable.

Format:

```
{FETCH7 | FIND7} [record-name] set-name USING data-item-1 [... ,data-item-n] ;  
[SUPPRESS-clause]
```

FETCH7 and FIND7 may be abbreviated **F7** and **FN7**, respectively.

Where *data-items* may be one or more data field names defined within the object record.

A *data-item* must be defined in the schema and may not be qualified.

The format of the suppress clause can be found Section 12.22 at the end of this chapter.

Example:

```
FETCH7 PROD-DET PROD-DET-SET USING PROD-TYPE PROD-CAT
```

12.10 FREE

The FREE command is used to release all locks caused by the KEEP command, or any updates that have taken place. The FREE also releases all QUICK-LOOKS, making all updates permanent.

Format:

FREE

12.11 IF (DML)

The DML IF command is used to determine whether a record participates in a set, or whether a set is empty. When in input mode, the IF command must be accompanied by an ENDIF command. For a complete description of IF/ENDIF pairs, refer to the IF description in the "General Procedure Commands" chapter of this manual.

Format:

```
IF { [NOT] {OWNER | MEMBER} set-name [SET] | ;  
    set-name [NOT] EMPTY}
```

SET may be abbreviated [S](#).

Restrictions: The DML IF cannot be extended by AND and OR operators as with the procedural IF; however, the DML IF may be used in conjunction with the ELSE and may be nested within other DML IF or procedural IF commands.

Examples:

```
FETCH4 NEXT PRODUCT-REC PRODUCTS AREA  
IF NOT OWNER PRODUCT-ON-ORDER      . Any members?  
...  
ENDIF
```

The following is an example of a DML IF nested within a procedural IF:

```
IF ERROR-NUM = 0  
    IF PRODUCT-ON-ORDER EMPTY      . Is the set empty?  
        GO END-RUN  
    ELSE  
        DO PRINT-RECS              . Set not empty.  
    ENDIF  
ENDIF
```

12.12 IMPART

The IMPART command is used to connect Q-LINK to the DMR. An IMPART cannot be done unless a successful INVOKE has been completed. If a ROLLBACK label is specified, control will be returned at the label upon receipt of a rollback error. The rollback error code will be placed in the reserved variable RBCODE. If no ROLLBACK label is specified, Q-LINK will perform a controlled abort upon receipt of a rollback error.

Format:

IMPART [ROLLBACK *program-label*]

IMPART may be abbreviated **IMP**.

Note: If alternate records delivery areas are to be defined (DEFINE RA) for DMS records, they should be specified after INVOKE but prior to IMPART.

Example:

```
      IMPART ROLLBACK RB-ERR-PROCESS
      ...
RB-ERR-PROCESS
      DISPLAY DBERROR
      DISPLAY "Processing abandoned"
      STOP EXIT
```

12.13 INSERT

The INSERT command is used to include the current record as a manual member of a set. The record must be defined as a participant in the set, and the owner record must be current of set type. The user may specify ALL sets, in which case all owner records must be current.

Format:

INSERT *record-name* {*set-name* | ALL}

INSERT may be abbreviated **INS**.

Example:

```
FIND5  ELT-MSTR           .  Find manual owner
FIND2  STEP-DTL           .  Make current
INSERT STEP-DTL ELT-USAGE .  INSERT into set.
```

12.14 KEEP

The KEEP command is used to extend a record currency lock. Currency locking is done to keep other runs from updating a particular record occurrence while the user is retrieving other records. The object of the KEEP must be current of run unit.

Format:

KEEP *record-name*

INSERT may be abbreviated **INS**.

12.15 MODIFY

The MODIFY command is used to replace the contents of a record in the database with the contents of the RDA. The object record must be current of run unit. The optional data-item-name(s) may be used to change the record's set participation (qualified modify). The owner identified by the data-item-name must be current of the set where participation is to change.

Format:

MODIFY *record-name* [*data-item-name-1* [... *data-item-name-n*]]

MODIFY may be abbreviated **M**.

The optional *data-item-name(s)* must be defined in the schema and may not be qualified. A *data-item-name* must be a field name in a defined owner record of the record being modified. The *data-item-name* identifies the set in which set participation is to change.

Examples:

FETCH5 PRODUCT-REC	. Fetch the product record.
RDA PRODUCT-DESC = 'XYZ'	. Change a field
MODIFY PRODUCT-REC	. Modify the database.

12.16 OPEN

The OPEN command is used to specify which database areas are to be accessed and in which usage mode. The default usage mode is retrieval.

Format:

`OPEN {area-name | ALL} [{RETRIEVAL | UPDATE} [EXCLUSIVE] | LOAD]`

OPEN may be abbreviated O, with RET for RETRIEVAL, U for UPDATE, EX for EXCLUSIVE and L for LOAD

The ALL option may not be specified with a usage mode of LOAD.

Examples:

```
OPEN CUSTOMERS UPDATE EXCLUSIVE
```

```
OPEN PRODUCTS LOAD
```

```
OPEN ALL . Opens all areas for retrieval
```

12.17 REMOVE

The REMOVE command is used to remove a record from a manual set relationship. The record must be current of run unit. The ALL option may be used, if the record is to be removed from all manual sets in which it currently participates.

Format:

REMOVE *record-name* {*set-name* | ALL}

REMOVE may be abbreviated **REM**.

Examples:

```
FETCH3 NEXT SERIAL SET      .  Get manual member.
REMOVE SERIAL-REC SERIAL    .  Remove record.
```

12.18 SET CURRENT

Format 1 and 2 of the SET CURRENT command is equivalent to a MOVE CURRENCY STATUS command in ASCII COBOL/DML. The third format allows a user change file to be specified when executing in the DMS test mode. To execute a Q-LINK program in DMS test mode, a separate server class must be established utilizing the "O" option. For batch servers, the option is placed on the Q-LINK processor call (@QLINK,O) in the server's runstream. For TIP servers, the option must be placed on the VALTAB entry (OPT=O).

Format-1:

```
[SET] CURRENT {DBK | AKEY | ANAME} ;
      [ = {RU | RECORD record-name | AREA area-name | SET set-name} ]
```

Format-2:

```
[SET] CURRENT {AKEY | ANAME} = database-key-variable
```

Format-3:

```
[SET] CURRENT CHANGEFILE = {RDA RDA-reference | alpha-variable}
```

The command keyword "SET" is implied and may be omitted. SET may be abbreviated S. CURRENT may be abbreviated CURR, with A for AREA and S for SET.

When the DBK option is used, the resulting database key is set in the reserved variable, C-DBK. When using the AKEY option, the resulting area key is set in the reserved variable C-AKEY. When the ANAME option is used, the resulting area name is set into the reserved variable C-AREA-NAME.

Format 3 only applies when interfacing with DMS 10R1 or a higher release level. The SET CURRENT CHANGEFILE command is not applicable to the single-thread interface. The *RDA-reference* or *alpha-variable* must contain a valid UREP 2200 storage area name (see UDS DMS 2200 Administration and Support Guide, 7830 7568, for more on change file considerations). The storage area name must be comprised of 12 or fewer ASCII characters. The SET CURRENT CHANGEFILE command must be issued prior to IMPART. If the change file is an EXEC file (not TIP), it must be assigned prior to IMPART. If test mode is specified and no SET CURRENT CHANGEFILE command is issued prior to IMPART, the system change file will be used.

Examples:

```
1.  FETCH4 PRODUCT-REC PRODUCTS AREA
    CURRENT DBK                               .   Save currency
    ...
    FIND1 PRODUCT-REC C-DBK
    CURRENT ANAME = RECORD PROD-REC          .   C-AREA-NAME = area
                                           .   name of the current
                                           .   PROD-REC.

2.  DEF A CFILE 12 'DBUFILE'                  .   DBUFILE is storage-area
                                           .   in UREP 2200
    CSF X '@ASG,A MY*DBUFILE.'                .   EXEC area must be
    IF X < 0                                    .   pre-assigned.
        FACERR X
        STOP EXIT
    ENDIF
    CSF X '@USE DBUFILE.,MY*DBUFILE.'
    IF X < 0
```



```
FACERR X
STOP EXIT
ENDIF
SET CURRENT CHANGEFILE = CFILE
IMPART
```

```
3. DEF RDA CHGFILE (*,12)
```

```
. . .
```

```
SET RDA CHGFILE = 'MY-CHANGE'
CURR CHANGEFILE = RDA CHGFILE
IMPART
```

```
. MY-CHANGE is TIP
. storage area.
```

12.19 SET DBDN (Database Data Name)

This form of the SET command takes a value from the data storage area and places it in a specified database data name for use by a DML command. The database data name may be an area key, database key, area name, etc. Data types must be compatible. The referenced database data name must be defined in the currently invoked subschema.

Format:

```
[SET] DBDN database-data-name = {integer-literal [,integer-literal] | ;
                                variable-name [,variable-name] | ;
                                'string-literal' } ;
      [(byte-length) {DISP | DISP-1 | COMP | COMP-4} ]
```

The command keyword "SET" is implied and may be omitted. SET may be abbreviated S.

The *database-data-name* must be defined in the currently invoked subschema.

The *byte-length* is the length of the database data name in bytes. The data type (DISP, DISP-1, COMP and COMP-4) are used to describe the form of the database data name. The *byte-length* and data type parameters must only be used when the database data name item being set does not have a usage mode of AREA-KEY, DATABASE-KEY, AREA-NAME, RECORD-NAME, or SET-NAME. These database data names have a form specifically defined within the object schema — no others, such as alias keys, do.

The *byte-length* for COMP and COMP-4 fields should be obtained from the conversion chart shown in Section 3.1.1.

Examples:

DBDN PRODAREA = 'PRODUCTS'	. Set an area-name for calc.
DBDN LOCAKEY = 501,1	. Set an area-key to page
	. 501, record 1.
DBDN LOCAKEY = HOLD-AREA-KEY	. Set an area-key from a
	. variable.
DBDN CONTROL-KEY = HOLD-PG, HOLD-REC	. Set an area-key from two
	. variables.
DBDN ACCESS-CTL = '\$XXX123' (12) DISP	. Set an access control
	. key.
DBDN PART-ALIAS = '123-0931-12' (30) DISP-1	. Set a FIELDATA alias key.

12.20 SET NON-FATAL (DML Errors)

This form of the SET command is used to eliminate the need to check for fatal error conditions resulting from the execution of DML commands. When a DML general error condition is encountered during execution, the non-fatal list will be checked. If the error number is not included in the list, Q-LINK will execute its DML error termination routine and abort. Initially, the non-fatal list includes general error numbers 6, 7 and 13. Error numbers may be added and removed from the list at any time using this command. Up to 50 general errors may be specified. Rollback errors may not be specified as non-fatal. Check your DML Reference Manual for the meanings of these error codes.

Format:

[[SET](#)] NON-FATAL *error-num* {ON | OFF}

The command keyword "SET" is implied and may be omitted. SET may be abbreviated [S](#).

Examples:

NON-FATAL 5 ON	. Make error-number 0005 non-fatal.
NON-FATAL 13 OFF	. Make error-number 0013 fatal.

12.21 STORE

The STORE command is used to store the contents of the RDA to a specific DMS 2200 record type. The data in the RDA may be created by using the SET RDA command, or it may be the result of a previous FETCH.

Format:

STORE *record-name* [*SUPPRESS-clause*]

STORE may be abbreviated **STR**.

The format of the suppress clause can be found in Section 12.25 at the end of this chapter.

Examples:

```
RDA PRODUCT-ID = 1011234
RDA PRODUCT-DESC = 'ABCDE'
DBDN PROD-AREA = 'PRODUCTS'      .   Set the area name DBDN.
STORE PRODUCT-REC                 .   Store the record.
```

12.22 SUPPRESS Clause

The SUPPRESS CLAUSE may be used as the last parameter of any FETCH/FIND or STORE command. It is used to suppress updating specified levels of currency when executing DML commands.

Format:

```
SUPPRESS {ALL | ;  
        || AREA | ;  
        RECORD | ;  
        {SET | ;  
         set-name-1 [... set-name-n] } || }
```

SUPPRESS may be abbreviated SUP, with A for AREA and S for SET.

The AREA, RECORD and SET options may be used in combination. If the ALL option is used, no other option may be used. If the SET option is used, then *set-names* may not be listed.

Examples:

Suppress update of area and set currency.

```
FETCH3 NEXT PROD-DETAIL-SET SET SUPPRESS AREA SET
```

Suppress update of set currency for the two named sets. All other currency indicators will be updated.

```
FETCH5 PROD-MASTER SUPPRESS PROD-DET-SET PRSPL-SET
```


Chapter 13: Direct I/O Access

The Q-LINK Processor can access sector formatted mass-storage and tape files using direct I/O.

13.1 DIO (DEFINE F)

For DIO, there is no I/O buffering used; data is transferred directly to and from the RDA. Mass-storage files may be EXEC, TIP/FCSS or TIP/DMS files. Tape files may be any format. Format:

`DEFINE F filename DIO maximum-I/O-length-in-words`

DEFINE may be abbreviated `DEF`.

To define a TIP/FCSS file, specify the filename as follows:

`TIP#nnn`

To define a TIP/DMS file, specify the filename as follows:

`TIPDMS#nnn`

Where *nnn* is the FCSS or DMS file code, i.e., `TIPDMS#21` for DMS area code 21, or `TIP#200` for FCSS file 200. If an FCSS or DMS file code of 0 (zero) is specified, the value in the reserved variable `S$` will be used as the file code when the file is accessed in a DIO command.

The *maximum-I/O-length-in-words* must not exceed the length of the RDA. This value will be used if the record is offset using a `DEFINE RA` directive.

Examples:

1. `DEFINE F WORK-FILE DIO 1000`
2. `DEFINE F TIPDMS#151 DIO 448`
3. `DEFINE F TIP#66 DIO 896`

Explanations:

Example 1 defines an EXEC file for DIRECT I/O. A maximum of 1000 words may be read from, or written to, the file in a single operation.

Example 2 defines a TIPDMS file (possibly a DMS 2200 database area) for DIRECT I/O.

Example 3 is a definition for a TIP/FCSS file.

13.2 DIO

The DIO command is used to perform direct I/O functions on a file defined for direct I/O. Data will be transferred directly between the RDA and the file using the IOW\$ (DM\$IOW in the case of TIP/DMS files) executive request, or the TIP FCSS file control primitives. Files accessed via the DIO command are not opened or closed as with PCIOS files. In the case of FCSS I/O, the three status words associated with the FCSS request will precede the data portion of the record in the RDA, automatically created for the user by the file DEFine.

Format:

DIO filename function status-variable count {sector-address | record-number}

To access a TIP/FCSS file, specify the filename as follows:

TIP#nnn

To access a TIP/DMS file, specify the filename as follows:

TIPDMS#nnn

Where *nnn* is the FCSS or DMS file code, i.e., TIPDMS#21 for DMS area code 21, or TIP#200 for FCSS file 200. If an FCSS or DMS file code of 0 (zero) is specified, the value in the reserved variable S\$ will be used as the file code when the file is accessed in a DIO command.

For EXEC and TIP/DMS files, the available functions are:

WRITE	Mass-storage or tape.
WRITEEOF	Tape only
READ	Mass-storage or tape.
READBACK	Tape only (read backwards).
RDLOCK	Mass-storage read with lock.
UNLOCK	Unlock mass-storage locked by RDLOCK.
REWIND	Tape only.
MOVE+	Tape only (move forward nn files).
MOVE-	Tape only (move backward nn files).
SPACE+	Tape only (space forward nn blocks).
SPACE-	Tape only (backspace nn blocks).

EXEC files must be @ASGed to the run before being accessed (see CSF command).

For TIP/FCSS files, the available functions are:

RD	Read.
RL	Read and Lock.
LK	Lock.
WR	Write.
WW	Write without lock.
WL	Write and keep lock.
UN	Unlock.
FL	File lock.
FR	File write lock.
AS	Assign file.
RV	Reserve file.
RE	Release file.
CG	Change file.
LF	List file.

In all cases, FCSS will execute synchronous I/O (FCDONE option).

The *status-variable* must be a defined numeric variable. It will contain the status at the completion of the command.

Count is used to specify the number of words to read or write; or the number of files or blocks in a MOVE or SPACE function. *Count* must be 0 for REWIND and WRITEEOF functions. The *count* may be entered as either a positive numeric integer literal or a numeric variable.

Sector-address applies to EXEC and TIP/DMS files, and is required on mass-storage READ and WRITE functions. It may be entered as either a positive numeric integer literal or a numeric variable.

Record-number applies only to FCSS files, and is required on READ and WRITE functions. It may be entered as either a positive numeric integer literal or a numeric variable.

Upon return from the DIO command, the I/O status will be put into the *status-variable*. For EXEC and TIP/DMS Read and Write functions, the final word count will be put into the reserved variable REC\$LEN. These values will be displayed to the user when executing in conversational mode.

Examples:

EXEC Direct I/O:

```
. Define tape and mass-storage files for Direct I/O.
DEFINE F TAPEIN DIO 2000
DEFINE F DISKFILE DIO 2000
...
    DIO TAPEIN MOVE+ X 1          . Move forward past
                                . one file.
    IF X NOT = 0                  . Check I/O status.
        DISPLAY 'I/O ERROR'
    ENDIF
    DIO TAPEIN READ X 2000        . Read 2000 words.
...
    DIO DISKFILE READ X 1000,64   . Read 1000 words at
                                . sector 64.
```

TIP/FCSS Direct I/O:

```
.  Define a TIP/FCSS file.
DEFINE F TIP#121 DIO 31
...
DIO TIP#121 RL TIPSTAT 28,1      .  Read record number 1
IF TIPSTAT < 0                  .  FCSS error?
    GO TIP-ERROR
ENDIF
DISPLAY RDA (13,4) UB9          .  Display 1st word of
                                .  record (positions
                                .  1 thru 12 are the
                                .  3 status words).
SET RDA (13,4) UB9 = 0          .  Set 1st word of
                                .  record to zero.
DIO TIP#121 WW TIPSTAT 28,1      .  Write record back.
IF TIPSTAT < 0                  .  Check for FCSS error
                                .  again ...
```

Chapter 14: RDMS 2200 Interface

The Q-LINK commands, RDMS and RDMS+, provide the interface to RDMS 2200 databases. The commands are used in a similar fashion to the following COBOL commands:

```
ENTER MASM 'ACOB$DMR' ...
```

```
ENTER MASM 'RSA$PARAM' ...
```

The RDMS+ command is conceptually similar to 'RSA\$PARAM' but must immediately follow the preceding RDMS (or RDMS+) command (not even a label is allowed in between). Also, the RDMS call must always pass the first three parameters. As many additional parameters as desired can be passed on the RDMS call subject to the following limitations: a maximum of 110 fields can appear in any Q-LINK command including a maximum of 70 alphabetic words or names ("RDA RDA-reference" counts as two), 30 numeric literals and 10 quoted string literals. If more fields are required, the RDMS+ command can be used.

Refer to the Enterprise Relational Database Server for ClearPath OS 2200 SQL Programming Reference Manual, 7830 8160, for a complete discussion of available RDML commands.

14.1 The RDMS Command

The RDMS command provides the necessary RDMS 2200 interface information to access RDMS 2200 databases. This information is comprised of an RDML or SQL formatted command, RDML error status variables and application program variables. These variables may be RDA references or reserved/user variables. In certain cases, references to \$PBUFF may be used as well as numeric and non-numeric literals.

Format:

```
RDMS RDML/SQL-command error-status auxiliary-information ;
      [program-variable-1 [, , program-variable-n] ]
```

Where *RDML/SQL-command* can be any, valid RDML command stated in one of the following ways:

```
{'command-literal' | $PBUFF | RDA RDA-reference | variable}
```

Note: Refer to the Relational Database Server for ClearPath OS 2200 SQL Programming Reference Manual (7830 8160), for a complete discussion of available RDML commands.

Where *error-status* may be one of the following ASCII alphabetic references, 4 characters in length:

```
{RDA RDA-reference | alpha-variable}
```

Where *auxiliary-information* may be one of the following binary numeric references:

```
{RDA RDA-reference | numeric-variable}
```

If the "*RDA RDA-reference*" is used, the item must be defined as UB9, SB9 or COMP.

Where *program-variable-1* through *program-variable-n* can be stated in one of the following ways:

```
{RDA RDA-reference | variable | numeric-literal | 'alpha-literal' | $PBUFF}
```

Any non-result parameter on the RDMS call can be a quoted string literal. Q-LINK allows a maximum of 10 string literals on a single command. For purposes of this check, RDMS and RDMS+ are separate commands; i.e., there may be up to 10 string literals on each occurrence of the RDMS and RDMS+ commands (see The RDMS+ in the next section).

Examples:

```
RDMS 'BEGIN THREAD FOR APPLICATION UDSSRC UPDATE ;' ;
RDMS-STAT RDMS-AUX
RDMS 'END THREAD ;' RDMS-STAT RDMS-AUX
```

Notice the use of the semicolons in the above example. The semicolon within the quoted string is a part of the RDML syntax and denotes the end of the command. The second semicolon (outside the literal) is standard Q-LINK notation. In this example, it is preceded by a space and it tells Q-LINK that the command is continued on the next line.

The semicolon can also be used to continue lengthy alphabetic literals as shown in the following example. Here, the semicolon is placed immediately after (no space) the second single quote on the first line. The semicolon used in this manner tells Q-LINK that the literal is continued on the next line. The second and third lines use command continuation as explained above.

```
RDMS 'GET DESCRIPTION INTO ' ;
      '$P1, $P2, $P3, $P4, $P5, $P6, $P7, $P8, $P9;' ;
RDMS-STAT RDMS-AUX ;
QUAL TAB VERS COL TYPE LEN DEC NULL KEY
```

The RDMS command allows the use of the \$PBUFF special name for any non-result parameter. Thus, you can build long commands (or data items) for RDMS by:

```

DISPLAY ' (long command or data, part 1) ' +
DISPLAY ' (long command or data, part 2) ' +
DISPLAY ' (long command or data, part 3) ' +
...
RDMS $PBUFF status-var aux-info-var ...

```

As with other uses of \$PBUFF, the print buffer is cleared once the call to RDMS is performed. \$PBUFF could appear multiple times in the command but that only makes sense if the SAME data needs to be supplied multiple times since the data passed would be the same for each occurrence.

Further examples:

```

D 'DECLARE LST CURSOR SELECT * FROM ' +
TD TABLE +
D ' WITH DESCRIPTION ;' +
RDMS $PBUFF RDMS-STAT RDMS-AUX

D 'GET DESCRIPTION INTO ' +
D '$P1, $P2, $P3, $P4, $P5, $P6, $P7, $P8, $P9;' +

RDMS $PBUFF RDMS-STAT RDMS-AUX ;
      QUAL TAB VERS COL TYPE LEN DEC NULL KEY

```

14.2 The RDMS+ Command

A maximum of 110 fields can appear in any Q-LINK command including a maximum of 70 alphabetic words or names ("RDA RDA-reference" counts as two), 30 numeric literals and 10 quoted string literals. If an RDMS command requires more parameters than can be parsed on a single command (or if you just don't like long, continued commands `;') the RDMS+ command can be used to extend the number of parameters supplied for the RDMS call. The RDMS+ command must immediately follow the preceding RDMS (or RDMS+) command (not even a label is allowed in between).

The first three parameters for the RDMS call must be supplied on the RDMS call itself. The fourth and subsequent parameters can be supplied via the RDMS+ command. The RDMS+ command may not be used in conversational mode (the previous RDMS command would have already been completely processed and executed before you could enter the RDMS+ command).

Format:

RDMS+ *program-variable-1* [... *program-variable-n*]

Where *program-variable-1* through *program-variable-n* can be stated in one of the following ways:

{RDA RDA-reference | variable | numeric-literal | 'alpha-literal' | \$PBUFF}

Examples:

```
D 'FETCH NEXT CUST_CURSOR INTO $P1,' +
D '$P2,$P3,$P4,$P5,$P6,$P7,$P8,$P9,$P10,$P11' +
RDMS $PBUFF STAT-VAR AUX-INFO
RDMS+ RDA CUST_KEY RDA CUST_LAST_NAME
RDMS+ RDA CUST_FIRST_NAME RDA CUST_MI ADDR1 ADDR2
RDMS+ RDA ADDR3 RDA ADDR4 RDA CITY RDA STATE
RDMS+ RDA ZIP
```

14.3 Automated RDA Definitions

Several RDML commands afford the ability to reference many program variables (column names). As shown in the previous example, these program variables may be RDA references. With this release of Q-LINK, a Q-LINK program is provided to facilitate and automate the definition of these RDA references. The seventh file (Q6) of the release tape contains a source element called RDMS-PICS. After Q-LINK installation, this element can normally be found in SYS\$LIB\$*QLINK-1.

RDMS-PICS retrieves the column names from an RDMS table and creates COBOL 02-level definitions suitable for input to the QINDEX processor. The QINDEX utility can then be used to create a data item index file that can be referenced by the Q-LINK INDEX directive. In addition to the COBOL 02-level definitions generated, RDMS-PICS generates a QINDEX FILE directive and a COBOL 01-level definition for each RDMS table chosen. The RDMS table name is used for the file name on the FILE directive and the 01-level data name. RDMS column names become the data names on the 02-level statements.

The QINDEX FILE directive generated allows a Q-LINK program to use a pseudo DEFINE F directive for each table and thus, take advantage of alternate record area processing (DEFINE RA) as with PCIOS files. For example, assume a Q-LINK program included the following definition of a PCIOS file, PAYMAST, and the pseudo definition for an RDMS 2200 table, PAY-RATE-TAB:

```
DEFINE F PAYMAST SEQ 120,0      . PCIOS file
DEFINE F PAY-RATE-TAB SEQ 42,0  . Pseudo define for
                                . RDMS Table
DEFINE RA PAY-RATE-TAB AFTER PAYMAST
```

The column values of PAY-RATE-TAB will occupy a different part of the RDA than the data item values of the file PAYMAST. The QINDEX FILE directive generated also makes it possible to reference column names without having to qualify a reference with the pseudo DEFINE F file name. However, since file names are restricted to 12 characters (table names can have as many as 30 characters), shorten the FILE directive file name to 12 or less characters prior to executing QINDEX. This is only necessary if you plan to use the pseudo DEFINE F.

The file cataloged and created by RDMS-PICS is QKMS*RDMSFILES.

The following is the source listing of RDMS-PICS:

```
. Create data item QINDEX input from RDMS 2200 column names
INPUT
INIT
LISTOFF
. RDMS VARS
DEF A APPLICATION 30      . Solicited from user to BEGIN
DEF A QUALIFIER 30        . THREAD and USE appropriate
DEF A VERSION 30          . QUALIFIER.TABLE:VERSION
DEF A TABLE 30           . (QUALIFIER = schema).
DEF A RDMS-STAT4''        . Error number from RDMS
DEF N RDMS-AUX 0          . Column if syntax error
DEF A ERROR-MSG 132       . Long message
DEF A PREV TAB 30         . Hold table for FILE and 01 gen
. RDMS $P1 -$P9           . Values returned from GET DESC
DEF A QUAL 30             . Not used
DEF A TAB 30              . Used to gen FILE and 01 name
DEF A VERS 30             . Not used
```

```

DEF A COL 30                .  Used to gen data name
DEF A TYPE 14               .  Not used
DEF N LEN                   .  Used to gen PIC clause
DEF N DEC                   .  Used to gen V dec positions
DEF N NULL                  .  Not used
DEF N KEY                   .  Not used
.
.  Solicit user input -QUIT on any query will terminate program
ACCEPT APPLICATION 'ENTER APPLICATION (<UDSSRC>/application_name): '
IF APPLICATION = $SPACES
    APPLICATION = 'UDSSRC'    .  UDS default application name
ELSE
    SHIFT APPLICATION TO UPPER
    IF APPLICATION = 'QUIT'
        GO QUITIT
    ENDIF
ENDIF
DO UNTIL QUALIFIER <> $SPACES
    ACCEPT QUALIFIER 'ENTER TABLE QUALIFIER (qualifier_name/';
                                'schema_name): '
    IF QUALIFIER = $SPACES
        D '*** QUALIFER MUST BE NON-BLANK ***'
    ELSE
        SHIFT QUALIFIER TO UPPER
        IF QUALIFIER = 'QUIT'    .  No default schema/qualifier
            GO QUITIT
        ENDIF
    ENDIF
ENDIF
ENDDO
ACCEPT VERSION 'ENTER TABLE VERSION (<PRODUCTION>/version_name): '
IF VERSION = $SPACES
    VERSION = 'PRODUCTION'    .  DDS default version name
ELSE
    SHIFT VERSION TO UPPER
    IF VERSION = 'QUIT'
        GO QUITIT
    ENDIF
ENDIF
.
.  BEGIN THREAD in RETRIEVE mode and set DEFAULTs
.
TD 'BEGIN THREAD FOR APPLICATION ' APPLICATION ' RETRIEVE ;' +
RDMS $PBUFF RDMS-STAT RDMS-AUX
DO ERRCHK
TD 'USE DEFAULT QUALIFIER ' QUALIFIER ' ;' +
RDMS $PBUFF RDMS-STAT RDMS-AUX
DO ERRCHK
TD 'USE DEFAULT VERSION ' VERSION ' ;' +
RDMS $PBUFF RDMS-STAT RDMS-AUX
DO ERRCHK
DO                                .  Do until AT END or BREAK

    ACCEPT TABLE 'ENTER TABLE (Table_name/QUIT): ' AT END BREAK
    SHIFT TABLE TO UPPER

```



```

IF TABLE = $SPACES
OR TABLE = 'QUIT'
    BREAK
ENDIF
PC BRKPT 'QKMS*RDMSFILES' . Will be input to QINDEX
DO TABLE
PC LOCAL . Reset temporarily for next ACC
ENDDO . - will append to BRKPT file
GO FINALE

```

TABLE PROCEDURE

```

TD 'DECLARE LST CURSOR SELECT * FROM ' TABLE ' WITH DESCRIPTION ;' +
RDMS $PBUFF RDMS-STAT RDMS-AUX
DO ERRCHK
DO
    RDMS 'GET DESCRIPTION INTO ' ;
        '$P1, $P2, $P3, $P4, $P5, $P6, $P7, $P8, $P9 ;' ;
        RDMS-STAT RDMS-AUX ;
        QUAL TAB VERS COL TYPE LEN DEC NULL KEY
    IF RDMS-STAT = '6001' . End of column names in table
        BREAK
    ENDIF
    DO WHILE TAB <> PREVTAB . Gen FILE and 01 name 1 time
        TD 'FILE ' TAB . per table
        TD 8 '01 ' TAB '.'
        PREVTAB = TAB . Hold current table name
    ENDDO
    SHIFT TYPE TO UPPER
    IF TYPE = 'NCHAR' . 16-bit char set not supported
        GO NEXT-COLUMN
    ENDIF
    D 12 '02 '+'
    D 16 COL + . Data name = column name
    IF TYPE = 'REAL'
    OR TYPE = 'FLOAT'
    OR TYPE = 'DOUBLE'
        D 47 ' COMP-2.'
    ELSE
        D 47 'PIC '+'
        IF TYPE = 'DECIMAL'
        OR TYPE = 'NUMERIC'
        OR TYPE = 'DATE'
        OR TYPE = 'TIME'
        OR TYPE = 'TIMESTAMP'
            D 52 'S9(' + . All decimal fields are signed
            IF DEC = 0
                TD LEN ')' COMP.'
            ELSE
                LEN = LEN - DEC
                TD LEN ')V9(' DEC ')' COMP.'
            ENDIF
        else

```

```

        if type = 'INTEGER'
        or type = 'SMALLINT'
            d 52 'S9(10) COMP.'
        else
            TD 52 'X(' LEN ')'. ' . CHARACTER/NCHARACTER
        endif
    ENDIF
ENDIF
NEXT-COLUMN
ENDDO
RDMS 'DROP CURSOR LST;' RDMS-STAT RDMS-AUX
DO ERRCHK
ENDPROC

FINALE
RDMS 'COMMIT WORK ;' RDMS-STAT RDMS-AUX
DO ERRCHK
RDMS 'END THREAD ;' RDMS-STAT RDMS-AUX
DO ERRCHK

QUITIT
STOP EXIT

ERRCHK PROCEDURE
    IF RDMS-STAT <> '0000'
        PC LOCAL
        D 'RDMS ERROR STATUS:' +
        D RDMS-STAT +
        D ' RDMS AUX INFO:' +
        D RDMS-AUX
        DO UNTIL ERROR-MSG = $SPACES
            RDMS 'GETERROR INTO $P1 ;' ;
            RDMS-STAT RDMS-AUX ERROR-MSG
            D '<ERRMSG>' +
            D ERROR-MSG
        ENDDO
        STOP EXIT
    ENDIF
ENDPROC

```

The program can be run as a batch or demand job utilizing the QLSIM processor as the following example illustrates:

```

@QLSIM, IS , , PROD1
ADD RDMS-PICS FROM SYS$LIB$*QLINK-1

DEMOSHEMA                                <- Application name
                                          <- Qualifier/Schema name
                                          <- Version name
CUST_KEY_TAB                            <- Table name #1
CUST_ADDR_TAB                           <- Table name #2
QUIT                                     <- Terminates RDMS-PICS
@EOF X
@CAT,P CUSTINDEX.
@QINDEX,IL CUSTINDEX.
@ADD QKMS*RDMSFILES.

```

The above session uses the RDMS 2200 default application name (UDSSRC) and the DDS 2200 default version (PRODUCTION) by simply transmitting a blank line. The information on the right side of each line (<- Application name, Qualifier, etc.) is not part of the actual input.

The output from RDMS-PICS for the above example would appear as follows:

```

FILE  CUST_KEY_TAB
      01  CUST_KEY_TAB.
          02  CUST_KEY          PIC X(9) .
          02  NAME              PIC X(33) .
          02  CHG_TIME          PIC S9(9) .
FILE  CUST_ADDR_TAB
      01  CUST_ADDR_TAB.
          02  STATE             PIC X(2) .
          02  CITY              PIC X(18) .
          02  ACCOUNT           PIC X(9) .
          02  CUSTNAME          PIC X(33) .
          02  ADDR1             PIC X(30) .
          02  ADDR2             PIC X(30) .
          02  ADDR3             PIC X(30) .
          02  ZIP               PIC X(9) .
          02  AREACODE          PIC X(3) .
          02  EXCHANGE          PIC X(3) .
          02  TELNUM            PIC X(4) .

```

The RDA references available by using the INDEX directive in a Q-LINK program would be:

```

CUST_KEY_TAB          (00001,00051)
CUST_KEY              (00001,00009)
NAME                  (00010,00033)
CHG_TIME              (00043,00009)  SN9
CUST_ADDR_TAB         (00001,00171)
STATE                 (00001,00002)
CITY                  (00003,00018)
ACCOUNT               (00021,00009)
CUSTNAME              (00030,00033)
ADDR1                 (00063,00030)
ADDR2                 (00093,00030)
ADDR3                 (00123,00030)
ZIP                   (00153,00009)
AREACODE              (00162,00003)
EXCHANGE              (00165,00003)
TELNUM                (00168,00004)

```


Chapter 15: BIS DTM Interface

Q-LINK programs can interface with BIS's Data Transfer Module (DTM). Two Q-LINK directives and four commands are provided to handle requests through BIS queues: the DEFINE F and DEFINE RA directives; and the OPEN, CLOSE, READ and WRITE commands.

It should be noted that #INSERT/ACCEPT is a faster method for accessing BIS data than using the BIS DTM interface.

For a complete example using the BIS DTM interface, see the Q-LINK Application Development User Guide.

15.1 Queue-alias Definition (DEFINE F)

Before BIS data can be referenced, a queue-alias must be defined with the DEFINE F directive. A queue-alias can be looked at as an internal file name to be associated with a particular BIS RID. Lines of data can be read from or written to this RID via a BIS queue name (configured with a BIS system through the DTM configuration report). Since it is possible to be accessing more than one RID simultaneously through the same BIS queue, Q-LINK requires that a separate queue-alias be defined for each of these opened reports. Queue-alias files may be opened, referenced and closed as necessary. All Queue-alias file and database input and output is done through the RDA. If multiple records are to be read and compared, the user may set up alternate record areas using the DEFINE RA directive, or move necessary data items to a defined variable storage location.

Format:

```
DEFINE F queue-alias MAPPER [maximum-transfer-size ;
    parameter-block [ {IN | OF} filename]
```

DEFINE may be abbreviated DEF.

The *maximum-transfer-size* may be between 8 and 256 characters. The default is 132 characters.

Note that the DEFINE FF format (available for PCIOS files) is NOT supported since the DTM interface only supports ASCII (FCS) transfers.

The *queue-alias* is used on the OPEN, READ, WRITE and CLOSE commands to refer to the specific BIS OPEN-*queue-alias*/*parameter-block* combination. The *queue-alias* is a logical name and does not convey the actual destination queue name. The use of the *queue-alias* allows the Q-LINK program to have several *queue-aliases* open concurrently to the same BIS queue name. The actual destination queue name is passed as a field in the *parameter-block*.

The *parameter-block* contains fields necessary to affect data transfer with the DTM. It must be defined prior to any queue-alias DEFINE F directive. The format of the *parameter-block* is as follows:

```

.   DTM interface parameter block for Q-LINK
.
def rda param-block          (*,168)
def rda pb-dest-queue        (param-block,12)  A9
def rda pb-userid            (*,12)            A9
def rda pb-dept              (*,4)              UN9
def rda pb-password          (*,6)              A9
def rda pb-filler1           (*,2)
def rda pb-mode              (*,12)            A9
def rda pb-type              (*,1)              A9
def rda pb-filler2           (*,3)
def rda pb-rid               (*,4)              A9
def rda pb-start-line        (*,4)              UN9
def rda pb-xfer-lines        (*,4)              UN9
def rda pb-run-name          (*,12)            A9
def rda pb-status            (*,1)              UB9
def rda pb-filler3           (*,3)
def rda pb-err-code          (*,8)              A9
def rda p--b-err-message     (*,80)            A9
.                                     168 character positions total
.

```

This data structure is defined in the element DTM-PARM-BLK in the file, SYS\$LIB\$*QLINKx-1, and can be copied directly into a Q-LINK program using the ADD directive. Consult the person responsible for installing Q-LINK in order to determine the actual file name of this file on your system.

The *parameter-block* may be qualified by a filename previously defined on a DEFINE F directive.

The *parameter-block* specifications are established by KMSYS Worldwide software development and as such are subject to change from release level to release level. For this reason, KMSYS Worldwide highly recommends that the definition for the parameter-block be copied from the file, SYS\$LIB\$*QLINKx-1, rather than hard coding it in a Q-LINK program.

The fields PB-DEST-QUEUE through PB-RUN-NAME must be filled in by the Q-LINK program before opening the queue-alias file. These parameter values, under similar but different COBOL names, are described in the Unisys Business Information Server for 2200 SCHDLR Interface Programming Reference Manual, 7832 1122, with the exception of the PB-START-LINE and PB-XFER-LINES fields, which are described below.

The PB-FILLERx fields are for structure alignment and must be present. The PB-STATUS, PB-ERR-CODE and PB-ERR-MESSAGE fields are returned following any DTM related Q-LINK command and must be tested by the Q-LINK program to determine the success of the operation. This is different from the Q-LINK PCIOS implementation where any error other than end-of-file or invalid-key is fatal to the program.

KMSYS Worldwide supplies a BIS run under the default name of QLINK\$DTM in the file, SYS\$LIB\$*QLINKx. It is in non-BIS format without headers and should be retrieved into BIS and registered by the BIS coordinator. This run can be restricted by BIS mode and/or user-id at the discretion of the BIS coordinator.

15.2 Alternate Queue-alias Areas (DEFINE RA)

The DEFINE RA directive for DTM performs similarly to its PCIOS implementation. Note that it defines an alternate record source/delivery area, which MUST be at least 256 characters (132 for BIS level 34 and earlier) in length for input operations regardless of the actual width of the BIS report being retrieved. This directive does not affect the location of the param-block referenced on the DEFINE F directive.

Format:

```
DEFINE RA {filename | record-name | queue-alias | SORT} ;  
        {absolute-word-number | ;  
        {OVERLAY | AFTER} {filename | record-name | queue-alias | SORT}
```

DEFINE may be abbreviated DEF.

15.3 CLOSE

The CLOSE command is similar to its PCIOS equivalents in that it releases any pending buffers, requests SCHDLR/BIS to complete the request and returns the completion status to the Q-LINK program. This command will be performed implicitly if an INIT directive is entered or if Q-LINK is about to terminate normally.

Format:

CLOSE *queue-alias*

CLOSE may be abbreviated **CL**.

15.4 OPEN

Currently BIS reports can be retrieved (INPUT) or created/replaced (OUTPUT). Other access modes are not currently supported. A partial report can be retrieved by setting the PB-START-LINE and PB-XFER-LINES fields in the parameter block BEFORE opening the queue-alias file. These parameters are not used when opening for OUTPUT. Other access modes (particularly APPEND) may be supported in future Q-LINK releases. Note that the DTM/MTQ interface in BIS limits the size of messages, which may be passed to 113,000 words (approximately 5100 80-character or 3000 132-character ASCII [FCS] lines). Thus, reports written to BIS that exceed this limitation must be written into multiple RIDs and combined within BIS. Larger reports can be conveniently retrieved from BIS into a Q-LINK program by opening the queue-alias multiple times using different line range specifications. The default BIS run supports referencing reports either by name or by mode number. If the RID number is supplied as zero on an OPEN for OUTPUT, the next available RID number will be used and the report actually created (by the AR [Add Report] function will be returned in the parameter block following the CLOSE command.

Format:

OPEN *queue-alias* {INPUT | OUTPUT} SEQ

INPUT may also be abbreviated INP.

15.5 READ

The READ command performs similarly to the PCIOS equivalent for sequential input files. The data is transferred to the record delivery area (its location is either position one of the RDA or as was modified by the DEFINE RA directive). The REC\$LEN reserved Q-LINK variable is set with the actual number of significant characters in the record (from zero to 256). Note that 256 characters (132 for BIS level 34 and earlier) is always delivered to the record delivery area, even if the significant data is shorter. Short records are padded with ASCII spaces.

Format:

```
READ queue-alias [AT END {program-label | BREAK} ]
```

15.6 WRITE

This command performs similarly to the PCIOS equivalent for sequential output files. The data is transferred from the record delivery area (its location is either position one of the RDA or as was modified by the DEFINE RA directive). The length transferred is either as specified on the WRITE command or is 256 characters (132 for BIS level 34 and earlier) if omitted.

Format:

WRITE *queue-alias* [*length*]

WRITE may be abbreviated **W**.

Chapter 16: Command and Keyword Abbreviations

The following is a list showing the minimum number of characters required for keywords and commands. Also shown are accepted abbreviations and mnemonics:

ACCEPT = A	ENDIF = ENDI	OPEN = O
ACQUIRE = ACQ	ENDPROC = ENDP	OPTION = OPT
ADD=AD	EXCLUSIVE = EX	OUTPUT = O
APPLICATION = APPL	EXIT = EXI	OWNER = O
AREA = A	FACERR = FA or FE	PCONTROL = PC
BITMERGE = BITM or BM	FETCH1 = F1	PRINTER = PR
BITSPLIT = BITS or BS	FETCH2 = F2	PRIOR = P
BREAK = B	FETCH3 = F3	PROCEDURE = PRO
BRKPT = B	FETCH4 = F4	READNEXT = READN
CDELETE = CD	FETCH5 = F5	RECORDS = RECS or
CHARACTERS = CHARS	FETCH6 = F6	RECORD
CLEAR = CLE	FETCH7 = F7	RELEASE = REL
CLEARSCREEN = CLS or	FIND1 = FN1	REMOVE = REM
CLEAR	FIND2 = FN2	RETRIEVAL = RET
CLOSE = CL	FIND3 = FN3	RETURN = RETU
COMPILE = C	FIND4 = FN4	REWRITE = REW
CONNECT = CONN	FIND5 = FN5	ROUND = ROU
CONSOLE = CONS or	FIND6 = FN6	RUN=R
CONSOL	FIND7 = FN7	SAVE = SA
CONV = CON	FIRST = F	SCAN = SC
CSF=CS	FREE = FR	SET=S
CURRENT = CURR	GREGORIAN = G	SHIFT = SHI
DATE = DA	IMPART = IMP	SORT = SOR
DBERROR = ERROR	INDEX = IND	START = ST
DECIMAL = DEC	INIT = INI	STORE = STOR or STR
DEFINE = DEF	INPUT = INP or I	SUPPRESS = SUP
DELETE = DEL	INSERT = INS	TABS = TA
DEPART = DEP	INVOKE = INV	TIME = TI
DISCONNECT = DIS	JULIAN = J	TRACE = TRA
DISPLAY = D	KEEP = K	TRANSFER = TRANS
DUMP = DU	LAST = L	TRIMDISP = TD or TRIMD
DUPLICATE = DUP	LISTOFF = LISTOF	TRIMEDIT = TE, TED or
ECHO = EC	LOAD = L	TRIME
EDIT = ED	LOCAL = L	UPDATE = U
EJECT = E	MODIFY = M	WILDCARD = WI
ELSE = EL	NEXT = N	WRITE = W
ENDDO = ENDD	OBJECT = OB	

Table 16-1: Keyword Abbreviation

Chapter 17: QINDEX Reference

QINDEX is a processor used to create a data item index file of data item definitions. QINDEX is executed in demand mode and is installed on your system when Q-LINK is installed. This chapter describes the purpose and rules for using the QINDEX processor.

17.1 Introduction

The data item index file is used by the Q-LINK processor to obtain the location and format of data items in the record delivery area (RDA) automatically. Q-LINK automatically creates a primary data item index file on start-up. When an INVOKE directive is processed, information related to DMS 2200 areas, records, sets and database datanames, and information related to data items (fields within records) is added to the primary index. This information is used in editing and encoding DMS 2200 DML commands, and in the resolution of RDA names used within various commands and directives. Data item definitions in the primary data item index are obtained directly from the object schema and subschema.

Data redefinitions coded in the subschema are not available via the INVOKE process, because they are not included in the object subschema. They are only present in the S\$PROC element created by the SDDL and copied into user programs by the ADMLP processor. To make redefinitions available to Q-LINK, a secondary data item index may be created using QINDEX.

When a data item name is used in a Q-LINK command or directive, the processor will first search its internal data definition table for the item name. Then, if the user has not defined the item (DEFINE RDA), the search will continue to the primary data item index for the correct definition. If the name is not found in the primary index, the secondary index, if present, is searched. The search of either index file will also match the record qualifier, if one was used. If the item name appears in more than one record and no record qualification is used, the first matching definition will be used.

The search order used in resolving a data item name is as follows:

1. Current DEFINE RDA names (always overrides data item index);
2. The primary data item index;
3. The secondary data item index.

A major use of the QINDEX processor is to create a single definition for all data within an entire application. The definition may be composed from a DMS 2200 subschema and schema plus any number of non-DMS 2200 file definitions. Non-DMS 2200 files may include any file types supported by the Q-LINK processor.

17.2 QINDEX, General Information

The QINDEX processor is designed to create compatible data item index files from object schemas and subschemas, and standard COBOL definitions for use in the Q-LINK processor. The data item index file created by QINDEX may be invoked by the Q-LINK processor by using the INDEX directive (see the Q-LINK Programmer Reference).

17.2.1 Input to QINDEX

The input to QINDEX consists of several directives, COBOL data definitions and optionally, an object schema and subschema. COBOL data definitions from many elements may be processed at one time (the processor call may be followed by many @ADD statements). No limit exists to the number of items that can be included in the data item index file. COBOL data definitions are input in standard COBOL source image format. The following restrictions apply:

- QINDEX will assume that the COBOL source input has been processed through the COBOL compiler at some point, and therefore will perform no COBOL syntax checking.
- Each 01 level definition from the source input will begin a new record definition. The generated RDA position pointer will be reset to character position one. The first data item image input to the QINDEX Processor must be an 01 level.
- Input images may contain comments, COBOL PROC headings and END statements. These will be ignored by the QINDEX processor.
- QINDEX will ignore all 66, 77 and 88 level items. FILLER items will be used to determine data item positions, and will not be output to the data item index file. Any VALUE clauses encountered will also be ignored.
- Input must not contain mixed FIELDATA and ASCII definitions. Allocation is based on the processor option settings. QINDEX will not make a distinction between DISPLAY and DISPLAY-1 or COMP and COMP-4. QINDEX will not support or recognize FIELDATA binary alignment (for example, PIC H99999).
- QINDEX supports UCOB data types of BINARY and BINARY-1. Also, both ACOB and UCOB S\$PROC elements may be input to QINDEX.
- Double-byte characters as specified by the DISPLAY-2 data type are supported for the Asian character sets.
- On items within an OCCURS clause, QINDEX will generate a definition that refers to the first item occurrence. References to subsequent occurrences may be made by using either the RDA Indexing or Subscripting features of the Q-LINK Processor.
- QINDEX supports exact binary notation for both DMS 2200 records (INVOKE directive) and PCIOS files (FILE directive). QINDEX can properly align data items that are included in record/file definitions (COBOL 01 level) containing exact binary data items. For example:

```
PIC 1(36)
PIC 1(5)
PIC 1(1)
```

This feature is currently supported for field alignment purposes only. A future release of Q-LINK will support access-to-bit definitions.

17.2.2 Output of QINDEX

The final output of QINDEX is a Q-LINK compatible secondary data item index file. In addition to the data item index file, QINDEX may optionally produce a listing of the data

item index file's contents. The listing will show each entry in alphabetical sequence, grouped by areas, records, sets, database data names and data items.

17.3 Running QINDEX

QINDEX must be run as a processor (not an @XQT) using the following format:

@ QINDEX,*options index-file-name*

Followed by DIRECTIVES and user-supplied COBOL data definition source images

The processor call name given to QINDEX (the default name) is determined when QINDEX is installed. For the correct processor name at your site, consult with the person responsible for installing QINDEX at your site.

The *index-file-name* must be the name of an existing mass-storage file that will be used as the Data Item Index File.

Following the processor call line are the optional QINDEX directives and user-supplied COBOL source images that will be used in processing the Data Item Index File.

17.3.1 QINDEX Processor Options

Available execution options are:

- B Causes the execution of QINDEX to be treated as a batch mode execution.
- D Causes the execution of QINDEX to be treated as a demand mode execution. This option is the converse of the B-Option.
- F Option "F" indicates that the definitions being processed are FIELDATA. The default is ASCII.
- I The "I" option will cause the index file to be initialized. Any existing definitions will be lost. If this option is not present, the index file will be opened for update. The INVOKE directive may only be used when the I option is used.
- L The "L" option will result in the listing of all source input images as they are read, and will cause the entire data item index file contents to be listed upon completion of processing of the source input.
- N Option "N" is used to produce a list of the contents of the index file with no update and no directives or source input. If this option is used, all other options are ignored.
- S Option "S" will cause the listing of all source input images as they are read.
- T For KMSYS Worldwide debugging only. Use only if directed to by KMSYS Worldwide' personnel.
- O The "O" option will cause the processor to overwrite any duplicate definitions encountered. The definition of the last duplicated item name will be used. If this option is not used, a warning message will be displayed for each duplicated item name encountered, and the first definition will be used.
- W Option "W" causes QINDEX to produce a data item list, replacing the "RDA REF" portion of the listing with starting word, starting bit and bit length of each item. The starting word is relative to zero, that is, for the first word in a record/file, starting word = 00000. The starting bit begins at bit one of the word and proceeds from left to right. If the "W" option is not specified, the default RDA reference (starting character, number of characters) will be displayed.

17.3.2 QINDEX Directives

There are several QINDEX directives that control the creation or update of the Data Item Index File. The following describes the function of each.

The QINDEX directives are:

INVOKE
S\$PROC
RECORD
FILE

17.3.2.1 INVOKE

The INVOKE directive for QINDEX works in a similar manner to the INVOKE in the Q-LINK processor in that it accesses the object schema and subschema to build the Data Item Index File. The difference is that no initialization of D\$WORK and S\$WORK is involved.

The INVOKE will create a data item index file record for each area, record, set, database dataname and data item included in the specified subschema. The data item index file will not include redefinitions of data items in the subschema, because redefinitions are not included in the subschema or schema object. Redefinitions are only present as source images in the S\$PROC element for the subschema which is copied into COBOL programs by the ADMPL Processor prior to program compilation (see the S\$PROC directive on the following page).

The INVOKE may only be used when initializing ("I" option) a new Data Item Index file, and must be the first directive read.

Format:

```
INVOKE subschema-name {IN | OF} sschema-name {FILE filename | TIP file-code} ;  
    [ [KEY] invoke-key ]
```

The *filename* or *file-code* must contain both the subschema and schema object elements.

Example:

```
INVOKE SUB-REQMTS IN MFG-SCHEMA FILE DMS*SCHEMAFILE
```

17.3.2.2 S\$PROC

The purpose of the S\$PROC directive is to apply data item redefinitions to an initial Data Item Index file created by the INVOKE directive. The S\$PROC element contains the COBOL source for the subschema as generated by the SDDL process, including any item redefines statements. INDEX will match the S\$PROC source to the records built by the INVOKE to apply redefinitions to the appropriate records. The S\$PROC directive is optional, and must immediately follow the INVOKE directive.

Format:

```
S$PROC [filename]
```

The *filename* is optional, and only needs to be specified if the S\$PROC element for the subschema named on the INVOKE is in a different file, or if the INVOKE specified a TIP schema file. If omitted, QINDEX will use the same file name used in the INVOKE. S\$PROC cannot be processed from a TIP schema file.

17.3.2.3 RECORD

The RECORD directive is used to add additional item definitions to an existing DMS 2200 record. In this manner, a record can be redefined in any way desired beyond any definitions that exist in either the schema or subschema. This redefinition may be required if an application uses a generic record definition in the schema with the record defined in many different formats within various application programs. Definitions added via the RECORD directive will be tied directly to the specified record. All RECORD directives must follow the INVOKE and S\$PROC directives, if present.

Format:

RECORD *subschema-record-name*

Followed by COBOL description source

The *subschema-record-name* must currently exist in the data item index file. It is placed in the data item index file by the INVOKE when the file is initialized.

If the subschema specifies only selected items of the record via an ITEMS ARE clause, care must be taken to ensure that added definitions match the mapped form of the record. The mapped form is the form containing only those items included in the ITEMS ARE clause.

Examples:

```
RECORD PART-MSTR
    01  PART-MSTR.
    *** THE FOLLOWING DEFINES THE FIRST 80 POSITIONS
    *** OF THE PART MASTER RECORD FOR USE AS A
    *** HEADER (THIS IS NOT DEFINED IN THE SCHEMA) .
        05  PART-MSTR-HDR          PIC X(80) .
        05  FILLER                 PIC X(102) .
```

The above definitions will be applied to the PART-MSTR record defined in the currently invoked subschema.

17.3.2.4 FILE

The file directive allows the addition of non-DMS 2200 file records. These definitions are not created by the INVOKE, S\$PROC or RECORD directives. When a file definition is included in a data item index file, the Q-LINK processor will be able to address file data items by name, and automatically determine the RDA location of the record.

When a file is defined in QINDEX, it is assigned an internal file code beginning with 4097. Each subsequent file is assigned the next higher number. This range is used to distinguish non-DMS 2200 files from DMS 2200 records (the highest possible record code is 4095). These codes are used in the Q-LINK processor to determine the record's RDA position automatically when items within records are referenced. In order for this feature to function correctly in Q-LINK, the INDEX directive must be processed before all file definitions (DEFINE Fs).

Format:

FILE *filename* [IBM]

Followed by COBOL description source

The *filename* must be the same name used when the file is defined in I Q-LINK with the DEFINE F directive.

Example:

```
FILE PART-HIST
    01  PART-HIST.
        05  PH-PART-NUMBER          PIC 9(5) .
        05  PH-PART-NAME            PIC X(50) .
        05  PH-PRICE                PIC 9(5)V999 .
        05  PH-PRICE-DATE           PIC 9(6) .
```

The file definition may contain REDEFINES and additional 01 levels as needed. Once a file has been defined in this manner, RDA item referencing and automatic record area offset operate the same as for DMS 2200 records.

QINDEX can be instructed to treat the COBOL PIC clause as if it had been defined for an IBM 360/370 environment: i.e., allocate/align each data item according to the rules required for the EBCDIC character set by adding the "IBM" option to the QINDEX FILE directive.

The allocation rules are defined below:

Data Type	UNISYS 2200 Allocation	IBM 360/370 Allocation
DISPLAY	9 bits per character (quarter word aligned)	9 bits per character (quarter word aligned)
COMP	1-2 digits, 9 bits 3-5 digits, 18 bits 6-7 digits, 27 bits 8-10 digits, 36 bits 11-13 digits, 45 bits 14-15 digits, 54 bits 16-18 digits, 63 bits (quarter word aligned)	1-4 digits, 18 bits 5-9 digits, 36 bits 10-18 digits, 72 bits (quarter word aligned)
COMP-1	36 bits (word aligned)	36 bits (quarter word aligned)
COMP-2	72 bits (word aligned)	72 bits (quarter word aligned)
COMP-3	9 bits per digit (quarter word aligned)	1 digit 9 bits 2-3 digits 18 bits 4-5 digits 27 bits 6-7 digits 36 bits 8-9 digits 45 bits 10-11 digits 54 bits 12-13 digits 63 bits 14-15 digits 72 bits 16-17 digits 81 bits 18 digits 90 bits (quarter word aligned)

Table 17-1: QINDEX Data Allocations

17.4 Building an Application Definition

The following example builds a new application Data Item Index file from the base subschema and schema, applies redefinitions from the S\$PROC element, adds more redefinitions to database records and finally adds PCIOS file definitions. The Data Item Index File built may then be considered an entire application data definition to be used in Q-LINK applications.

```
@ASG,UP MFG*MFGQINDEX.
@QINDEX,IL MFG*MFGQINDEX.
INVOKE REQMTS-SUB IN MFG-SCHEMA FILE DMS*SCHEMAFILE
S$PROC
RECORD PART-MSTR
    01 PART-MSTR.
    *** THE FOLLOWING DEFINES THE FIRST 80
    *** POSITIONS OF THE PART MASTER RECORD FOR
    *** USE AS A HEADER (NOT DEFINED IN SCHEMA).
        05 PART-MSTR-HDR      PIC X(80).
        05 FILLER              PIC X(102).
FILE DLY-TRANS
    01 DLY-TRANS-REC.
```

```

05 DT-ACCOUNT-NUM      PIC 9(10) .
05 DT-ACCOUNT-ALP REDEFINES DT-ACCOUNT-NUM.
    10 DT-DIVISION      PIC 9(3) .
    10 DT-REGION        PIC 9(3) .
    10 DT-SERIAL        PIC 9(5) .
05 DT-CTGY-CODE        PIC X.
05 DT-TRANS-CODE       PIC X.
05 DT-AMOUNT           PIC S9(5)V99 COMP.

@EOF

```

The proper sequence for using the sample data item index in a Q-LINK application program is as follows:

```

INVOKE REQMTS-SUB IN MFG-SCHEMA FOR MT NX
INDEX MFG*MFGQINDEX.
DEFINE F DLY-TRANS SEQ 16,200
DEFINE RA DLY-TRANS AFTER PART-MSTR
...

```

The file name used in the DEFINE F directive must match the file name used when the file was defined in QINDEX in order for Q-LINK to detect record area offsets automatically.

Once the application has been defined as shown here, any data item in any DMS 2200 record or non-DMS file may be referenced without qualification. The referencing of DMS 2200 records or non-DMS files without qualification is possible because each item definition has been linked to its DMS 2200 record or file name internally by QINDEX. When an item is referenced in Q-LINK, the command editor automatically looks up the item's record or file entry to determine if the record or file has been relocated within the RDA via a DEFINE RA directive. If an item name appears within more than one record or file within the application, it will be necessary to qualify its reference.

17.5 A QINDEX Execution

The following example shows the executions of QINDEX to build the data item index file used in the examples shown throughout this guide. The file contains both DMS and PCIOS data item definitions. In the execution, a FILE directive is supplied so that the Q-LINK DEFINE RA (record area) directive can be used in the generated DBM code for InfoQuest.

```

@DELETE,C MKTG*CO-INDE.
FURPUR 31R5 (990611 1158:59) 1999 Sep 23 Thu 1545:20
END DELETE.
@ASG,UPV MKTG*CO-INDEX.
I:002333 ASG complete.
@QINDEXA,LI MKTG*CO-INDEX.
QINDEX 6R6-0826 (Release 6R6) (991109 1425:41) 1999 Nov 18 Thu 1545:21
(C) Copyright 1985-1999 by KMSYS Worldwide, Inc. All Rights reserved.
This program licensed for use by KMSYS Worldwide, INC.
File:MKTG*???????CO-INDEX.???? already assigned
***Index will be INITIALIZED.
(00001)INVOKE DEMOSUB IN DEMOSCH FILE uds$$src*schabs
(00002)RECORD CUST-ADDR-REC
(00003)    *** THE FOLLOWING REDEFINES TO CA-LOCKEY FIELD ***
(00004)    01 CUST-ADDR-REC-RE-DEF.
(00005)        05 CA-LOCKEY-REDEF.
(00006)            10 CA-STATE          PIC XX.
(00007)            10 CA-CITY           PIC X(18) .

```

```

(00008)          10  CA-ACCOUNT          PIC 9(9) .
(00009) FILE ORDERFILE
(00010)          01  ORDER-RECORD.
(00011)          ** Secondary key, duplicates allowed **
(00012)          05  OR-CUSTKEY.
(00013)          10  OR-CUSTDIV          PIC 9.
(00014)          10  OR-CUSTNUM          PIC X(5) .
(00015)          10  OR-CUSTSHIPTO      PIC 999.
(00016)          ** Primary key **
(00017)          05  OR-ORDER-IDENT.
(00018)          10  OR-ORDER-LOC        PIC 99.
(00019)          10  OR-ORDER-KEY        PIC X(7) .
(00020)          10  OR-SHIP-LOC         PIC 99.
(00021)          05  OR-ORDER-TYPE-CODE  PIC X.
(00022)          05  OR-PRODLINE-CODE    PIC X.
(00023)          05  OR-ENTRY-DATE.
(00024)          10  OR-ENTRY-MO         PIC XX.
(00025)          10  OR-ENTRY-DA        PIC XX.
(00026)          10  OR-ENTRY-YR        PIC XX.
(00027)          05  OR-BUYER.
(00028)          10  OR-BYPASS           PIC X.
(00029)          10  FILLER              PIC X(10) .
(00030)          05  OR-PURCHASE-ORD     PIC X(8) .
(00031)          05  OR-REQ-SHIPDATE     PIC X(6) .
(00032)          05  OR-SHIP-VIA         PIC X(11) .
(00033)          05  OR-CREDIT-HOLD      PIC X.
(00034)          05  OR-HOLD-CODE        PIC X.
(00035)          05  OR-INVOICE-CODE     PIC X.
(00036)          05  OR-BOL-PRT-CODE     PIC X.
(00037)          05  OR-PAY-METHOD-CODE PIC XXX.
(00038)          05  OR-WORKORD-CODE     PIC X.
(00039)          05  OR-SPECIAL-TERMS    PIC X(20) .
(00040)          05  OR-TERMS.
(00041)          10  OR-TERM-CODE        PIC X.
(00042)          10  OR-TERM-PER        PIC V9(4) COMP.
(00043)          10  OR-TERM-DATE-DAYS   PIC 9(6) .
(00044)          05  OR-DELETE-FLAG      PIC X.
(00045)          05  OR-INPROCESS-HOLD   PIC X.
(00046)          05  FILLER              PIC XX.
(00047)          05  OR-ACTUAL-SHIP-DATE.
(00048)          10  OR-SHIP-YR          PIC XX.
(00049)          10  OR-SHIP-MO          PIC XX.
(00050)          10  OR-SHIP-DA          PIC XX.
(00051)          05  OR-PIECES           PIC 9(5) COMP.
(00052)          05  FILLER              PIC XX.
(00053)          05  OR-WEIGHT           PIC 9(7) COMP.
(00054)          05  OR-SHIP-FEE         PIC 9(5)V99 COMP.
(00055)          05  OR-TOT-CHARGES     PIC 9(6)V99 COMP.
(00056)          05  OR-TOT-CLC         PIC 9(6)V99 COMP.
(00057)          05  OR-DISCOUNT       PIC 9(5)V99 COMP.
(00058)          05  OR-CREDIT-REL-DATE.
(00059)          10  OR-CREL-YR          PIC XX.
(00060)          10  OR-CREL-MO          PIC XX.
(00061)          10  OR-CREL-DA          PIC XX.
(00062)          05  OR-WORKORD-PRT-DATE.
(00063)          10  OR-WKORD-PRT-YR     PIC XX.
(00064)          10  OR-WKORD-PRT-MO     PIC XX.

```

```

(00065)          10  OR-WKORD-PRT-DA      PIC XX.
(00066)          05  OR-STATE-TAX         PIC S9(4)V99 COMP.
(00067)          05  OR-CITY-TAX          PIC S9(4)V99 COMP.
(00068)          05  OR-COUNTY-TAX        PIC S9(4)V99 COMP.
(00069)          05  OR-CREDIT-USERID     PIC X(8).
(00070)          05  OR-NBR-PALLETS       PIC S9(2) COMP.
(00071)          05  OR-PALLET-CHG        PIC S9(3)V99 COMP.
(00072)          05  OR-TOT-PALLET-COST   PIC S9(5)V99 COMP.
(00073)          05  OR-AUTHDLR-CODE      PIC X.
(00074)          05  OR-LINE-COUNT        PIC 9(10) COMP.
(00075)          05  OR-ORDER-LINE-DATA OCCURS 1 TO 50
(CONT.)          DEPENDING ON OR-LINE-COUNT.
(00076)  ***  Order line item data ***
(00077)          10  OR-PRODUCT           PIC X(6).
(00078)          10  OR-TYPE-ORD-CODE     PIC X.
(00079)          10  OR-QUANTITY          PIC S9(5) COMP.
(00080)          10  OR-UNIT-PRICE        PIC 9(5)V99 COMP.
(00081)          10  OR-DESC              PIC X(25).
(00082)          10  OR-LINE-WEIGHT       PIC 9(5)V99 COMP.
(00083)          10  OR-PACKAGE           PIC X(8).
(00084)          10  OR-PRICE-CODE        PIC XX.
(00085)          10  OR-BOL-KEY           PIC 999 COMP.
(00086)          10  OR-TAX-CODE          PIC X.
(00087)          10  OR-REG-CODE          PIC X.
(00088)          10  OR-SHIP-QTY          PIC 9(5) COMP.
(00089)          10  OR-BILL-ONLY-CODE    PIC X.
(00090)          10  OR-PRICE-CHANGE      PIC X.
(00091)          10  OR-LAST-DATE         PIC X(6).
(00092)          10  OR-PRIORITY          PIC X.
(00093)          10  OR-EXCEPTION-SW      PIC X.
(00094)          10  OR-SUB-ITEM          PIC X(6).
(00095) FILE CUSTFILE
(00096)          01  CUSTORMER-MASTER-REC.
(00097)  ** Primary key **
(00098)          05  CM-ACCOUNT           PIC X(9).
(00099)  ** Secondary key, duplicates allowed **
(00100)          05  CM-LOCKEY.
(00101)          10  CM-STATE             PIC XX.
(00102)          10  CM-CITY              PIC X(18).
(00103)          05  CM-CUSTNAME          PIC X(33).
(00104)          05  CM-ADDR1            PIC X(30).
(00105)          05  CM-ADDR2            PIC X(30).
(00106)          05  CM-ADDR3            PIC X(30).
(00107)          05  CM-ZIP              PIC X(9).
(00108)          05  CM-TELEPHONE.
(00109)          10  CM-AREACODE          PIC XXX.
(00110)          10  CM-EXCHANGE          PIC XXX.
(00111)          10  CM-TELNUM           PIC XXXX.
.....End of QINDEX processing.
*****
*** Date Item Index File List ***
*****
( Subschema names and codes listed )

<<< Schema/Subschema >>>
DEMOSCH      DEMOSUB      SCHEMA/SUBSCHEMA

***** COUNT 0000000001

```

<<< Areas >>>

DEMO-ADDR	AREA CODE: 00001
DEMO-CKEY	AREA CODE: 00003
DEMO-INDX	AREA CODE: 00002
DEMO-ORD	AREA CODE: 00004

***** COUNT 0000000004 remove

<<< Records >>>

CUST-ADDR-REC	RECORD CODE:00002
CUST-KEY-REC	RECORD CODE:00001
ORDER-COMMENT-REC	RECORD CODE:00004
ORDER-HEADER-REC	RECORD CODE:00003
ORDER-LINE-REC	RECORD CODE:00005

***** COUNT 0000000005

<<< Sets >>>

CUST-KEY-ADDR-SET	SET CODE: 00001
CUST-ORD	SET CODE: 00004
ORDH-CMT	SET CODE: 00003
ORDH-LINE	SET CODE: 00002

***** COUNT 0000000004

<<< Database Datanames >>>

AREA-DEMO-CKEY	DBDN CODE: 00002
AREA-DEMO-ORD	DBDN CODE: 00003
SH-MISC-ANAME	DBDN CODE: 00001

***** COUNT 0000000003

<<< Files >>>

CUSTFILE	FILE CODE: 04098
ORDERFILE	FILE CODE: 04097

***** COUNT 0000000002

<<< Data Items >>>

IX ITEM NAME	RDA REF	DATA TYPE	DEC POS	REC/FILE COD
-- -----	-----	-----	-----	-----
18:CA-ACCOUNT	(00021,00009)	UN9		00002
17:CA-ADDR1	(00063,00030)	A9		00002
17:CA-ADDR2	(00093,00030)	A9		00002
17:CA-ADDR3	(00123,00030)	A9		00002
17:CA-AREACODE	(00162,00003)	A9		00002
16:CA-CITY	(00003,00018)			00002
17:CA-CUSTNAME	(00030,00033)	A9		00002
17:CA-EXCHANGE	(00165,00003)	A9		00002
17:CA-LOCKEY	(00001,00029)	A9		00002
16:CA-LOCKEY-REDEF	(00001,00029)			00002
16:CA-STATE	(00001,00002)			00002
17:CA-TELEPHONE	(00162,00010)	A9		00002
17:CA-TELNUM	(00168,00004)	A9		00002
17:CA-ZIP	(00153,00009)	A9		00002
00:CK-CHG-TIME	(00043,00004)	UB9		00001
17:CK-CUST-KEY	(00001,00009)	A9		00001
17:CK-NAME	(00010,00033)	A9		00001
16:CM-ACCOUNT	(00001,00009)			04098
16:CM-ADDR1	(00063,00030)			04098

Q-LINK Programmer Reference

QINDEX Reference

16:CM-ADDR2	(00093,00030)		04098
16:CM-ADDR3	(00123,00030)		04098
16:CM-AREACODE	(00162,00003)		04098
16:CM-CITY	(00012,00018)		04098
16:CM-CUSTNAME	(00030,00033)		04098
16:CM-EXCHANGE	(00165,00003)		04098
16:CM-LOCKEY	(00010,00020)		04098
16:CM-STATE	(00010,00002)		04098
16:CM-TELEPHONE	(00162,00010)		04098
16:CM-TELNUM	(00168,00004)		04098
16:CM-ZIP	(00153,00009)		04098
17:CUST-ADDR-REC	(00001,00180) A9		00002
16:CUST-ADDR-REC-RE-DEF	(00001,00029)		00002
17:CUST-KEY-REC	(00001,00048) A9		00001
16:CUSTOMER-MASTER-REC	(00001,00171)		04098
17:OC-BOL-TYPE	(00082,00001) A9		00004
17:OC-COMMENT	(00001,00080) A9		00004
17:OC-TYPE	(00081,00001) A9		00004
17:OH-ACTUAL-SHIP-DATE	(00106,00006) A9		00003
17:OH-AUTHDLR-CODE	(00168,00001) A9		00003
17:OH-BOL-PRT-CODE	(00068,00001) A9		00003
17:OH-BUYER	(00029,00011) A9		00003
17:OH-BYPASS	(00029,00001) A9		00003
04:OH-CITY-TAX	(00148,00003) SB9	02	00003
04:OH-COUNTY-TAX	(00151,00003) SB9	02	00003
17:OH-CREDIT-HOLD	(00065,00001) A9		00003
17:OH-CREDIT-REL-DATE	(00133,00006) A9		00003
17:OH-CREDIT-USERID	(00154,00008) A9		00003
17:OH-CREL-DA	(00137,00002) A9		00003
17:OH-CREL-MO	(00135,00002) A9		00003
17:OH-CREL-YR	(00133,00002) A9		00003
18:OH-CUSTDIV	(00001,00001) UN9		00003
17:OH-CUSTKEY	(00001,00009) A9		00003
17:OH-CUSTNUM	(00002,00005) A9		00003
18:OH-CUSTSHIPTO	(00007,00003) UN9		00003
17:OH-DELETE-FLAG	(00102,00001) A9		00003
03:OH-DISCOUNT	(00130,00003) UB9	02	00003
17:OH-ENTRY-DA	(00025,00002) A9		00003
17:OH-ENTRY-DATE	(00023,00006) A9		00003
17:OH-ENTRY-MO	(00023,00002) A9		00003
17:OH-ENTRY-YR	(00027,00002) A9		00003
17:OH-HOLD-CODE	(00066,00001) A9		00003
17:OH-INPROCESS-HOLD	(00103,00001) A9		00003
17:OH-INVOICE-CODE	(00067,00001) A9		00003
01:OH-NBR-PALLETS	(00162,00001) COMP		00003
17:OH-ORDER-IDENT	(00010,00011) A9		00003
17:OH-ORDER-KEY	(00012,00007) A9		00003
18:OH-ORDER-LOC	(00010,00002) UN9		00003
17:OH-ORDER-TYPE-CODE	(00021,00001) A9		00003
04:OH-PALLET-CHG	(00163,00002) SB9	02	00003
17:OH-PAY-METHOD-CODE	(00069,00003) A9		00003
00:OH-PIECES	(00112,00002) UB9		00003
17:OH-PRODLIN-CODE	(00022,00001) A9		00003
17:OH-PURCHASE-ORD	(00040,00008) A9		00003
17:OH-REQ-SHIPDATE	(00048,00006) A9		00003
17:OH-SHIP-DA	(00110,00002) A9		00003
03:OH-SHIP-FEE	(00119,00003) UB9	02	00003

QINDEX Reference

Q-LINK Programmer Reference

18:OH-SHIP-LOC	(00019,00002)	UN9		00003
17:OH-SHIP-MO	(00108,00002)	A9		00003
17:OH-SHIP-VIA	(00054,00011)	A9		00003
17:OH-SHIP-YR	(00106,00002)	A9		00003
17:OH-SPECIAL-TERMS	(00073,00020)	A9		00003
04:OH-STATE-TAX	(00145,00003)	SB9	02	00003
17:OH-TERM-CODE	(00093,00001)	A9		00003
18:OH-TERM-DATE-DAYS	(00096,00006)	UN9		00003
03:OH-TERM-PER	(00094,00002)	UB9	04	00003
17:OH-TERMS	(00093,00009)	A9		00003
03:OH-TOT-CHARGES	(00122,00004)	UB9	02	00003
03:OH-TOT-CLC	(00126,00004)	UB9	02	00003
04:OH-TOT-PALLET-COST	(00165,00003)	SB9	02	00003
00:OH-WEIGHT	(00116,00003)	UB9		00003
17:OH-WKORD-PRT-DA	(00143,00002)	A9		00003
17:OH-WKORD-PRT-MO	(00141,00002)	A9		00003
17:OH-WKORD-PRT-YR	(00139,00002)	A9		00003
17:OH-WORKORD-CODE	(00072,00001)	A9		00003
17:OH-WORKORD-PRT-DATE	(00139,00006)	A9		00003
17:OL-BILL-ONLY-CODE	(00057,00001)	A9		00005
00:OL-BOL-KEY	(00051,00002)	UB9		00005
17:OL-DESC	(00013,00025)	A9		00005
17:OL-EXCEPTION-SW	(00066,00001)	A9		00005
17:OL-LAST-DATE	(00059,00006)	A9		00005
17:OL-PACKAGE	(00041,00008)	A9		00005
17:OL-PRICE-CHANGE	(00058,00001)	A9		00005
17:OL-PRICE-CODE	(00049,00002)	A9		00005
17:OL-PRIORITY	(00065,00001)	A9		00005
17:OL-PRODUCT	(00001,00006)	A9		00005
01:OL-QUANTITY	(00008,00002)	COMP		00005
17:OL-REG-CODE	(00054,00001)	A9		00005
00:OL-SHIP-QTY	(00055,00002)	UB9		00005
17:OL-SUB-ITEM	(00067,00006)	A9		00005
17:OL-TAX-CODE	(00053,00001)	A9		00005
17:OL-TYPE-ORD-CODE	(00007,00001)	A9		00005
03:OL-UNIT-PRICE	(00010,00003)	UB9	02	00005
03:OL-WEIGHT	(00038,00003)	UB9	02	00005
16:OR-ACTUAL-SHIP-DATE	(00106,00006)			04097
16:OR-AUTHDLR-CODE	(00168,00001)			04097
16:OR-BILL-ONLY-CODE	(00229,00001)			04097
00:OR-BOL-KEY	(00223,00002)	UB9		04097
16:OR-BOL-PRT-CODE	(00068,00001)			04097
16:OR-BUYER	(00029,00011)			04097
16:OR-BYPASS	(00029,00001)			04097
01:OR-CITY-TAX	(00148,00003)	COMP	02	04097
01:OR-COUNTY-TAX	(00151,00003)	COMP	02	04097
16:OR-CREDIT-HOLD	(00065,00001)			04097
16:OR-CREDIT-REL-DATE	(00133,00006)			04097
16:OR-CREDIT-USERID	(00154,00008)			04097
16:OR-CREL-DA	(00137,00002)			04097
16:OR-CREL-MO	(00135,00002)			04097
16:OR-CREL-YR	(00133,00002)			04097
18:OR-CUSTDIV	(00001,00001)	UN9		04097
16:OR-CUSTKEY	(00001,00009)			04097
16:OR-CUSTNUM	(00002,00005)			04097
18:OR-CUSTSHIPTO	(00007,00003)	UN9		04097
16:OR-DELETE-FLAG	(00102,00001)			04097

Q-LINK Programmer Reference

QINDEX Reference

16:OR-DESC	(00185,00025)			04097
00:OR-DISCOUNT	(00130,00003)	UB9	02	04097
16:OR-ENTRY-DA	(00025,00002)			04097
16:OR-ENTRY-DATE	(00023,00006)			04097
16:OR-ENTRY-MO	(00023,00002)			04097
16:OR-ENTRY-YR	(00027,00002)			04097
16:OR-EXCEPTION-SW	(00238,00001)			04097
16:OR-HOLD-CODE	(00066,00001)			04097
16:OR-INPROCESS-HOLD	(00103,00001)			04097
16:OR-INVOICE-CODE	(00067,00001)			04097
16:OR-LAST-DATE	(00231,00006)			04097
00:OR-LINE-COUNT	(00169,00004)	UB9		04097
00:OR-LINE-WEIGHT	(00210,00003)	UB9	02	04097
01:OR-NBR-PALLETS	(00162,00001)	COMP		04097
16:OR-ORDER-IDENT	(00010,00011)			04097
16:OR-ORDER-KEY	(00012,00007)			04097
16:OR-ORDER-LINE-DATA	(00173,00072)			04097
18:OR-ORDER-LOC	(00010,00002)	UN9		04097
16:OR-ORDER-TYPE-CODE	(00021,00001)			04097
16:OR-PACKAGE	(00213,00008)			04097
01:OR-PALLET-CHG	(00163,00002)	COMP	02	04097
16:OR-PAY-METHOD-CODE	(00069,00003)			04097
00:OR-PIECES	(00112,00002)	UB9		04097
16:OR-PRICE-CHANGE	(00230,00001)			04097
16:OR-PRICE-CODE	(00221,00002)			04097
16:OR-PRIORITY	(00237,00001)			04097
16:OR-PRODLINE-CODE	(00022,00001)			04097
16:OR-PRODUCT	(00173,00006)			04097
16:OR-PURCHASE-ORD	(00040,00008)			04097
01:OR-QUANTITY	(00180,00002)	COMP		04097
16:OR-REG-CODE	(00226,00001)			04097
16:OR-REQ-SHIPDATE	(00048,00006)			04097
16:OR-SHIP-DA	(00110,00002)			04097
00:OR-SHIP-FEE	(00119,00003)	UB9	02	04097
18:OR-SHIP-LOC	(00019,00002)	UN9		04097
16:OR-SHIP-MO	(00108,00002)			04097
00:OR-SHIP-QTY	(00227,00002)	UB9		04097
16:OR-SHIP-VIA	(00054,00011)			04097
16:OR-SHIP-YR	(00106,00002)			04097
16:OR-SPECIAL-TERMS	(00073,00020)			04097
01:OR-STATE-TAX	(00145,00003)	COMP	02	04097
16:OR-SUB-ITEM	(00239,00006)			04097
16:OR-TAX-CODE	(00225,00001)			04097
16:OR-TERM-CODE	(00093,00001)			04097
18:OR-TERM-DATE-DAYS	(00096,00006)	UN9		04097
00:OR-TERM-PER	(00094,00002)	UB9	04	04097
16:OR-TERMS	(00093,00009)			04097
00:OR-TOT-CHARGES	(00122,00004)	UB9	02	04097
00:OR-TOT-CLC	(00126,00004)	UB9	02	04097
01:OR-TOT-PALLET-COST	(00165,00003)	COMP	02	04097
16:OR-TYPE-ORD-CODE	(00179,00001)			04097
00:OR-UNIT-PRICE	(00182,00003)	UB9	02	04097
00:OR-WEIGHT	(00116,00003)	UB9		04097
16:OR-WKORD-PRT-DA	(00143,00002)			04097
16:OR-WKORD-PRT-MO	(00141,00002)			04097
16:OR-WKORD-PRT-YR	(00139,00002)			04097
16:OR-WORKORD-CODE	(00072,00001)			04097

QINDEX Reference

Q-LINK Programmer Reference

16:OR-WORKORD-PRT-DATE	(00139,00006)	04097
17:ORDER-COMMENT-REC	(00001,00084) A9	00004
17:ORDER-HEADER-REC	(00001,00172) A9	00003
17:ORDER-LINE-REC	(00001,00072) A9	00005
16:ORDER-RECORD	(00001,03772)	04097

***** COUNT 0000000195

**** End of Data Item Index List ****

Chapter 18: QLK External Function Errors

The following is a list of error codes, explanations of errors and possible error resolutions for all errors reported to BIS runs by the @QLK external function and the QLSIM (Q-LINK/InfoQuest only) program:

ERROR**CODE DESCRIPTION**

- | | |
|---|---|
| 1 | <p>SPECIFIED SERVER NOT ACTIVE</p> <p>Explanation: The server class passed via the QLK function or the third specification field on the QLSIM processor does not have any servers active and no servers were successfully activated by the server auto-start process (if configured).</p> <p>Resolution: If not using the server auto-start, you might configure it. If batch servers are taking too long to start, consider using TIP servers. If you do not use the server auto-start, consider having an initial number of servers started when the system is initialized by QMON (R-option in conjunction with the L-option or the RESET command), and carefully consider the effects of using the terminate idle server timer configuration. Investigate why auto-started servers are not able to start. Check the QMON STATUS key-in to verify that the requested class is in the CDB tables. If the PEND or FAIL or QMON columns of the class status are non-zero, investigate why auto-start requests are being delayed or rejected. Remember if using the QMON auto-start configuration (versus BATCH or TIP) that auto-start requests will be held by the CDB until a QMON or a QMON,S run is activated.</p> |
| 2 | <p>CDB ID CONSTANT VALIDATION FAILED</p> <p>Explanation: A BIS caller could not access all of the CDB due to an addressing error or the Q-LINK CDB has been corrupted.</p> <p>Resolution: Check your BIS main buffer pool address limits versus your CDB configuration to determine if an address conflict should occur. If the problem persists see the action for error 1.</p> |
| 3 | <p>*** Error code not currently in use. ***</p> <p>Explanation:</p> <p>Resolution:</p> |

**ERROR
CODE DESCRIPTION**

- | | |
|---|---|
| 4 | MAPPER RELEASED BY QMON

Explanation: A BIS run which was waiting for a server to act was released because the server aborted processing the request.

Resolution: Check for server aborts due to either E-keyins or other operational events. Report unexpected server aborts via the procedure outlined for common bank error 1. |
| 5 | MAX REQUESTS QUEUED FOR SERVER CLASS

Explanation: More than the configured maximum number of requests for this server class would have been outstanding had this request been queued.

Resolution: Investigate why requests are not completing faster. Consider increasing the number of servers that can be active. The auto-start and auto-terminate configuration can make this painless, and minimize the impact on operations at other times of the day. Ensure that the number of servers that can be auto-started is actually starting promptly and successfully. Consider moving long running requests from this class into another class to allow short requests to complete. Consider the priority and execution type of the server. TIP requests generally (based on VALTAB) have higher priority than BATCH requests and TIP transaction type servers would usually be configured for higher priority than TIP batch-transaction type servers. Check other system bottlenecks for resource problems including CPU/memory/IO saturation and DMS run-unit queuing problems. |
| 6 | SERVER REQUESTED ABORT OF REQUEST

Explanation: A runtime error in the Q-LINK program was detected by the server (other than max reply lines, max time or a security error).

Resolution: Retrieve the server log (if not already returned for DEVELOPMENT mode servers) and inspect it for error messages defining the type and cause of the error. |
| 7 | SERVER CLASS QUEUE NOT IN MCT

Explanation: The server class passed via the QLK function or the third specification field on the QLKSIM processor is not currently configured in the CDB.

Resolution: Check the spelling and configuration of the specified class name. Use the QMON CONFIG command to verify that the proper configuration is resident in the CDB. |
| 8 | USER ABORTED FUNCTION WITH MESSAGE WAIT KEY

Explanation: The Q-LINK server has detected that the user has depressed the message wait key (or the F4 key when running through MCB) and has terminated processing of this request. When running through the QLSIM program, the @@X C and the operator II-key-in will also terminate a request in this fashion.

Resolution: None. |
| 9 | ALL SERVER CLASSES LOCKED

Explanation: The QMON program has been used to lock all request classes and prevent the queuing or acceptance of new requests.

Resolution: Contact your system administrator to determine why Q-LINK access has been terminated. |

ERROR CODE	DESCRIPTION
10	<p>THIS SERVER CLASS IS LOCKED</p> <p>Explanation: The QMON program has been used to lock this specific class and prevent the queuing or acceptance of new requests.</p> <p>Resolution: Contact your system administrator to determine why Q-LINK access has been terminated for this specific class.</p>
11	<p>ASSIGNED SERVER VANISHED!</p> <p>Explanation: The assigned server was no longer in the CDB tables when BIS returned for/with the next block of lines.</p> <p>Resolution: See error code 1.</p>
12	<p>BAD RID - OPEN I/P TO READ</p> <p>Explanation: A nonexistent input RID was specified on the @QLK function or QLKSIM utility call line.</p> <p>Resolution: Verify the desired MODE/TYPE/RID combination for the request.</p>
13	<p>RID # TOO BIG FOR TYPE ON OPEN</p> <p>Explanation: The input RID number specified is larger than the largest configured in the BIS startup parameters.</p> <p>Resolution: Verify the desired MODE/TYPE/RID combination for the request.</p>
14	<p>LINE # TOO BIG ON OPEN I/P RID</p> <p>Explanation: The input RID did not contain even one line of data.</p> <p>Resolution: Verify the desired MODE/TYPE/RID combination for the request. Verify that the desired data is actually in the requested RID.</p>
15	<p>SYSTEM ERROR OPENING I/P RID</p> <p>Explanation: A BIS system error occurred trying to open the input RID.</p> <p>Resolution: Refer this problem to your BIS coordinator.</p>
16	<p>ERROR OPENING O/P RID</p> <p>Explanation: A BIS system error occurred trying to open the output RID.</p> <p>Resolution: Refer this problem to your BIS coordinator.</p>
17	<p>ERROR WRITING OUTPUT RID</p> <p>Explanation: A BIS system error occurred trying to write to the output RID.</p> <p>Resolution: Refer this problem to your BIS coordinator.</p>
18	<p>ERROR CLOSING OUTPUT RID</p> <p>Explanation: A BIS system error occurred trying to finish writing the output RID.</p> <p>Resolution: Refer this problem to your BIS coordinator.</p>
19	<p>ERROR WRITING DATE LINE</p> <p>Explanation: A BIS system error occurred trying to write the date line to the output RID.</p> <p>Resolution: Refer this problem to your BIS coordinator.</p>

ERROR CODE	DESCRIPTION
20	<p>ERROR WRITING HEADER LINE TO OUTPUT RID</p> <p>Explanation: A BIS system error occurred trying to write the header lines to the output RID.</p> <p>Resolution: Refer this problem to your BIS coordinator.</p>
21	<p>SYS MAX LINES REACHED</p> <p>Explanation: The Q-LINK user program was terminated when it tried to generate more lines of output than are permitted for this class.</p> <p>Resolution: Investigate how much data the request should be designed to return. Occasional large output can be allowed via the Q-LINK password on the @QLK function or QLSIM program call.</p>
22	<p>USER-SPECIFIED MAX LINES REACHED</p> <p>Explanation: The Q-LINK user program was terminated when it tried to generate more lines of output than were specified on the @QLK function or QLKSIM program call.</p> <p>Resolution: Investigate why the program attempted to return more data than the designer intended.</p>
23	<p>SYSTEM ERROR READING I/P RID</p> <p>Explanation: A BIS system error occurred trying to read from the input RID.</p> <p>Resolution: Refer this problem to your BIS coordinator.</p>
24	<p>CDB ADDRESSING WINDOW PROBLEMS</p> <p>Explanation: The combination of the BIS main buffer pool sizing and the CDB installation selected base address has produced an overlap of the CDB address space and the BIS buffer pool address space.</p> <p>Resolution: Either BIS must be reconfigured, or the CDB must be reinstalled in such a way as to eliminate the addressing overlap. See also the action for common bank errors 11 and 12.</p>
25	<p>UNABLE TO TRMRG\$ MAPPER ACTIVITY</p> <p>Explanation: The ER TRMRG\$ performed by the CDB when registering a BIS activity has been rejected by the EXEC.</p> <p>Resolution: If the problem persists, submit an EXEC dump immediately following the error occurrence, along with the information requested for an internal Q-LINK error, to KMSYS Worldwide, Inc., for analysis.</p>
26	<p>CDB FAILED INITIAL STATUS CHECKS</p> <p>Explanation: The CDB is improperly installed, or is from an incompatible level of Q-LINK. Or BIS was improperly generated with regard to the CDB bank name, type and/or BDI.</p> <p>Resolution: Investigate the type and status of the bank installed for the Q-LINK CDB and the configuration of the BIS TCF and the status and completion log of the BIS generation used to install the @QLK function. For BIS 34 and higher, there is no QLK CDB parameter card defining the CDB BDI. If the problem persists, submit an EXEC dump immediately following the error occurrence along with the information requested for an internal Q-LINK error to KMSYS Worldwide, Inc., for analysis.</p>

ERROR CODE	DESCRIPTION
27	<p>ERROR OPENING #INSERT RID</p> <p>Explanation: The Q-LINK function in BIS was unable to open the RID specified on the #insert directive.</p> <p>Resolution: Verify the desired MODE/TYPE/RID combination for the request.</p>
28	<p>ERROR READING #INSERT RID</p> <p>Explanation: An error occurred trying to read from the input RID.</p> <p>Resolution: Refer this problem to your BIS coordinator.</p>
29	<p>SYNTAX ERROR ON #INSERT DIRECTIVE</p> <p>Explanation: The format of the #insert directive is incorrect.</p> <p>Resolution: Verify the format of the #insert command from the Q-LINK documentation. Any comment on the directive must be preceded by at least one space, a period and at least one additional space, and the comment must follow all required fields.</p>
30	<p>SERVER CLASS MAXIMUM XQT TIME EXCEEDED</p> <p>Explanation: The Q-LINK user program was terminated when it tried to execute and consume more resources (SUPS) than are permitted for this class.</p> <p>Resolution: Investigate how long the request program should run. Consider processing fewer records during the request. Consider more efficient database location paths for the desired data. As a last resort, the execution time limit configured for the class can be increased, but this allows requests that might loop to consume unwarranted amounts of system resources. If requests requiring differing magnitudes of system resources are currently lumped together in one class, consider creating an additional class(es) each of which has an appropriate time limit for the type of processing required.</p>
31	<p>ACCESS NOT ALLOWED (#INSERT MODE EDITING)</p> <p>Explanation: The BIS run function attempted to access data within a MODE that is not allowed by the run registration RID for that run.</p> <p>Resolution: Verify the desired MODE/TYPE/RID combination for the #insert request. Request that your BIS coordinator update the run registration log for this run if you feel your need to access the data is legitimate. Alternatively, move the data to a MODE, which the run can access.</p>
32	<p>TOO MANY CHAINED #INSERTS ENCOUNTERED</p> <p>Explanation: When a #INSERT is encountered in a RID being read with #INSERT, the first RID is closed and the new RID is opened and read. Q-LINK does not keep a history of which RIDs have been processed in a chain of #INSERT commands, and so cannot explicitly detect a situation where a RID contains a #INSERT which directly or indirectly references itself. To prevent BIS from looping forever in such a case, Q-LINK counts the number of #INSERT directives encountered while reading from a RID selected by another #INSERT. If the count of chained #INSERT commands exceeds ten, this error is issued.</p> <p>Resolution: Check the #INSERT commands in the data stream, and ensure that the maximum chaining of RIDs is not greater than ten. Also, ensure that the chain of #INSERT commands never references a RID present earlier in the chain.</p>

ERROR CODE	DESCRIPTION
33	<p>SECURITY ERROR - NO ACCESS TO CLASS</p> <p>Explanation: Your BIS user group (configured in Q-LINK security) is not allowed access to the server class that you have selected.</p> <p>Resolution: Contact your site security officer to determine which server classes you can access.</p>
34	<p>SECURITY ERROR - COMPILE NOT ALLOWED</p> <p>Explanation: Your BIS user group (configured in Q-LINK security) is not allowed to compile requests with this server class.</p> <p>Resolution: Contact your site security officer to determine which server class you can use to compile requests.</p>
35	<p>SECURITY ERROR - RUN NOT ALLOWED</p> <p>Explanation: Your BIS user group (configured in Q-LINK security) is not allowed to run requests with this server class.</p> <p>Resolution: Contact your site security officer to determine which server classes you can use to run requests.</p>
36	<p>QLKFUN IMAGE LENGTH ERROR</p> <p>Explanation: A bad ACW was transferred to QLKFUN (BIS) from the server.</p> <p>Resolution: Contact KMSYS Worldwide, Inc.</p>
51 through 81	<p>COMMON DATA BANK DIRECTED ERRORS</p> <p>Resolution: For error codes 51 through 81, subtract 50 from the error code and use the result to reference the errors in Chapter 19, "Q-LINK Common Bank Error Conditions"; i.e., @QLK error code 68 can be found in the next chapter as error code 18 ($68 - 50 = 18$), "INSUFFICIENT PRIVILEGE FOR REQUEST".</p>

Chapter 19: Q-LINK Common Bank Error Conditions

The following is a list of error codes, explanations of errors and possible error resolutions for all errors reported by QMON, Q-LINK servers and other Q-LINK utility programs:

ERROR

CODE DESCRIPTION

- | | |
|---|---|
| 1 | <p>SERVER TABLE NOT FOUND</p> <p>Explanation: Internal error. A server run-id was not found in the CDB when it was expected to be there.</p> <p>Resolution: Take a dump of the common bank immediately following the error using the QMON DUMP command. Take a dump of the utility program reporting the error using @PMD. Initialize the common data bank using the QMON INIT command and try again. Submit the dumps and a listing of your configuration (both generation and runtime) to KMSYS Worldwide, Inc., for analysis.</p> |
| 2 | <p>UNKNOWN FUNCTION REQUESTED</p> <p>Explanation: Internal error. An unused or out of range function code was passed to QCOM (part of the Q-LINK server program) on a request call.</p> <p>Resolution: See error code 1.</p> |
| 3 | <p>INIT REQ AND SERVER ALREADY ACTIVE</p> <p>Explanation: Internal error. A server initialization function was received by the CDB from a run-id, which was already marked as active in the CDB tables.</p> <p>Resolution: See error code 1.</p> |
| 4 | <p>MAXIMUM NUMBER OF SERVERS ACTIVE</p> <p>Explanation: Configuration or operational error. The number of servers allowed to be active concurrently for all classes is too small for the number of servers attempting to connect to the CDB. The runtime configuration for maximum servers active needs to be as large as the sum of the maximum active servers for all classes.</p> <p>Resolution: Change your runtime configuration to reduce the number of servers allowed for some classes, or increase the maximum number of servers globally allowed active by the CDB.</p> |

ERROR CODE	DESCRIPTION
------------	-------------

- | | |
|---|--|
| 5 | <p>QMON IS SHUTTING DOWN THIS SERVER -OR- SERVER WAS ABORTED (E-KEY, ...) -OR- Q-LINK TIMER ACTIVITY IS SHUTTING DOWN SERVER</p> <p>Explanation: This condition is expected by Q-LINK and, as such, it should not appear as an error; i.e., the CDB is requesting a server to exit.</p> <p>Resolution: Investigate QMON EX and TERM command usage at your site. Ensure that the runtime configuration for the time a server is to remain idle before exiting (DURATION SGS) is as large as you would like it to be.</p> |
| 6 | <p>MAXIMUM SERVERS ALREADY ACTIVE FOR CLASS</p> <p>Explanation: Configuration or operational error. The number of servers allowed to be active concurrently for this class is too small for the number of servers attempting to connect to the CDB. The only time this condition should occur is when someone is manually starting servers, or if auto-started servers were held for some reason while one or more QMON RESET or INIT commands were done.</p> <p>Resolution: If you really want more servers active, update the runtime configuration through COMUS. If not, investigate why servers are being delayed in starting, or who is manually starting extra servers.</p> |
| 7 | <p>NO CLASS QUEUE SLOT FOUND</p> <p>Explanation: A server attempted to initialize with a class name which was not in the CDB's current configuration. This could be caused by a misspelled server's class name on the @QLK command in a manually started batch server. This could also occur if the CDB was initialized with a configuration not containing this class name. Since class names cannot be dynamically added to the CDB, you can successfully process a configuration update but still not have access to the new server class.</p> <p>Resolution: Verify that the server name requested is really in the current COMUS configuration. Verify what classes are currently available in the CDB by using the QMON CONFIG command. If you need to add the class to the CDB, you must shut down the Q-LINK system using the following commands to QMON: LOCK, ALL, TERM and then initialize the Q-LINK system using the QMON INIT command.</p> |
| 8 | <p>ABORT OF A REQUEST SIGNALLED REQUESTED FROM MAPPER -OR- MAPPER HAS ABORTED (E-KEY,...)</p> <p>Explanation: This error is not fatal! Someone has externally requested the termination of the current request for this server. Possible cause includes a shutdown of the BIS system that submitted this request to the server.</p> <p>Resolution: Investigate the BIS down/purge times as well as any manual shutdowns of BIS or any cases of BIS aborting.</p> |
| 9 | <p>**** Error code not currently in use. ****</p> <p>Explanation:</p> <p>Resolution:</p> |

ERROR CODE	DESCRIPTION
-----------------------	--------------------

- | | |
|----|--|
| 10 | <p>ERROR LOCATING OR READING CONFIG ELT</p> <p>Explanation: All the Q-LINK utility programs and Q-LINK batch servers locate the runtime configuration by finding an absolute element called QLNK\$CFIG in the file from which they were executed. The TIP servers are passed the filename from which the CDB configuration was initialized, and they expect the QLNK\$CFIG absolute to be in that file. BIS has the CDB type as specified by a parameter card and does not need access to the configuration absolute. If the configuration element cannot be found or is badly formatted, or if an I/O error occurs when attempting to read it, then this error is reported. This can be caused by a hardware or software damaged execution file, by copying a Q-LINK program into a different file (i.e., TPF\$) and not copying the configuration element also, or by attempting to process a configuration element produced by a different level of Q-LINK. Note that immediately following a Q-LINK generation and/or install, the installation file (Q5 or SYS\$LIB\$*QLINKx) contains a dummy QLNK\$CFIG element which cannot be loaded by the Q-LINK programs.</p> |
| 11 | <p>HIGHEST ADDRESS REQUIRED FOR CDB IS GREATER THAN 0777000 (1ST PCT BLK ADDR)</p> <p>Explanation: The runtime configuration of maximum servers active and number of classes defined, combined with the base address for the CDB chosen at install time, has produced a CDB whose highest address exceeds 0777000.</p> <p>Resolution: Reduce the upper address limit of the CDB by reducing one (or more) of the following: CDB base address (INSTALL), maximum servers concurrently active (CONFIGURE), or number of defined classes (CONFIGURE). Remember not to reduce the base address of the CDB below the highest address of the main buffer pool of BIS or intermittent addressing status errors may be returned to @QLK users. If you are running a very large configuration and running multiple BIS installations or InfoQuest batch servers, you can split these to use a second (or more) CDB although it seems extremely unlikely that anyone would have a configuration this large.</p> |
| 12 | <p>Q-LINK UNKNOWN ERROR</p> <p>Explanation:</p> <p>Resolution: Contact KMSYS Worldwide, Inc.</p> |
| 13 | <p>UNABLE TO DO AN ER TO TRMRG\$</p> <p>Explanation: The ER TRMRG\$ performed by the CDB when registering a server or QMON run has been rejected by the EXEC.</p> <p>Resolution: If the problem persists, submit an EXEC dump immediately following the error occurrence, along with the information requested for an internal Q-LINK error, to KMSYS Worldwide, Inc., for analysis.</p> |
| 14 | <p>ALL CLASSES LOCKED/UNLOCKED</p> <p>Explanation: A QMON LOCK (UNLOCK), ALL directive was performed when the global request lock was already set (clear).</p> <p>Resolution: Perform a QMON STATUS key-in prior to issuing the LOCK/UNLOCK command. Investigate the usage of the QMON LOCK/UNLOCK command at your site.</p> |

**ERROR
CODE****DESCRIPTION****15** **QUEUE CLASS NOT FOUND**

Explanation: A QMON LOCK (UNLOCK) class directive was performed for a class whose name is not currently in the configuration tables of the CDB. Alternatively, an attempt was made to update the configuration by adding a class name to the CDB dynamically. Yet another alternative, by some strange timing quirk, a class name was removed from the CDB (by a QMON INIT command) while a second QMON was auto-starting a TIP MCB connected server.

Resolution: Verify the currently active configuration using the QMON CONFIG command. If you have added a class by the configure process, but the CDB was already initialized, you must use the QMON TERM and INIT commands to allow the reloading of the configuration data into the CDB.

16 **SERVER INDEX OUT OF RANGE**

Explanation: An invalid server index was used on a QMON EX key-in.

Resolution: See error code 1.

17 **QMON ALREADY ACTIVE**

Explanation: Only one run can be registered for the QMON privileged commands at any one time.

Resolution: Use the "S" option on one or all of the QMON executions. The QMON copy, that received this error, automatically acted as if the "S" option were specified. The QMON STATUS keyin will display the run-id and start time of the currently registered privileged QMON run.

18 **INSUFFICIENT PRIVILEGE FOR REQUEST**

Explanation: Certain CDB requests require that you be signed on under one of a specific list of user-ids before you are allowed the privilege associated with that request.

Resolution: Check the list of user-ids configured in the security SGSs for utility security. Check the user-id of the offending run to make sure that it is configured. If the user is a TIP transaction server, its pseudo user-id will consist of the program name prefixed with a '\$' character; i.e., the QLINK transaction program requires configuration of a \$QLINK user-id.

19 **MUST HAVE QMON UNLIMITED PRIVILEGE FOR REQUEST**

Explanation: Certain CDB requests require that the caller be the privileged QMON user registered with the CDB. If a non-privileged user requests such a function (reload the CDB, etc.) it will be rejected.

Resolution: The "S" option on QMON prevents registration as the privileged user and can lead to this error if privileged commands are then attempted. As pointed out for error 17, if a user is not the first to attempt to register for these privileges, it is as if he used the "S" option when starting QMON, and that can lead to this error. Check the QMON privileges for the user in the security SGSs.

ERROR CODE	DESCRIPTION
------------	-------------

20	BANK NOT YET LOADED
----	---------------------

Explanation: Only the QMON program can initialize the CDB following a COMUS installation, a system reboot or a console reload bank keyin. If any other caller attempts to link to the CDB prior to QMON beginning the bank initialization, the caller will be rejected.

Resolution: The QMON program must be executed in some fashion before starting servers or other utility programs. BIS can be started before the CDB is initialized, but cannot access Q-LINK until the CDB is initialized. An

@SYS\$LIB\$*QLINKx.QMONx ,L in either the system startup runstream or the BIS startup runstream or the server startup runstream will allow the common bank to be loaded in an automatic fashion. Be sure to use the QMON and the QLNK\$CFIG element from the current installation file as whatever configuration is in the execution file will be the one which is loaded.

21	TEMPLATE BANK ILL-FORMED -OR- REQUESTER NOT IN QUARTER-WORD MODE
----	--

Explanation: Either the CDB absolute has been damaged by some software or hardware malfunction, or the CDB does not match the level of the Q-LINK utility/server program being executed.

Resolution: Verify the level of Q-LINK used when installing the CDB and the level of Q-LINK being used for execution. If all else fails, de-install the Q-LINK software and then repeat the installation of the Q-LINK software with COMUS.

22	CDB CONTINGENCY (INCL RELOAD)
----	-------------------------------

Explanation: Some contingency has occurred which the CDB is not prepared to handle. This will occur if someone reloads the CDB using the console key-in or a privileged reload program. Other errors constitute an internal error in the Q-LINK system. The contingency packet information is returned to the caller in A1 and A2.

Resolution: Investigate the usage of reload key-ins/programs affecting the Q-LINK CDB. Check for installation of other software using COMUS that uses the same BDI or bank name as the Q-LINK CDB. If the problem persists, see the action for error 1.

23	ILLEGAL CB FUNCTION
----	---------------------

Explanation: Internal error. An unused or out of range function code was passed to the CDB in A0.

Resolution: See error code 1.

24	SERVER(S) ARE ACTIVE
----	----------------------

Explanation: Attempt to terminate Q-LINK with the QMON TERM command while servers were processing a request.

Resolution: Use the QMON LOCK ALL command to prevent new requests from starting. Once all active requests have finished, retry the TERM command.

**ERROR
CODE****DESCRIPTION**

- | | |
|----|--|
| 25 | <p>A MAPPER RUN WAS RELEASED</p> <p>Explanation: Attempt to initialize the Q-LINK CDB with the QMON INIT command while a BIS run was waiting for a server activity to process. The BIS run has been released.</p> <p>Resolution: The INIT command should be retried after first using the QMON LOCK, ALL and allowing active requests to terminate.</p> |
| 26 | <p>NO BIS RUN TO RELEASE</p> <p>Explanation: A BIS run was to be released by the CDB and no BIS run was waiting for the specified server.</p> <p>Resolution: See error code 1.</p> |
| 27 | <p>SERVER NOT CLOSED; I.E., IS BUSY</p> <p>Explanation: The server specified on an EX/TERM key-in could not be shutdown since it was not idle.</p> <p>Resolution: Lock the class queue using the QMON LOCK command and then wait till the server is idle before terminating it.</p> |
| 28 | <p>SERVER NOT CLOSED; I.E., IS TERMINATED/ABORTED</p> <p>Explanation: The server specified on an EX/TERM key-in could not be shutdown since it is in a termination or abort state.</p> <p>Resolution: If this condition does not clear shortly, then this is an internal error and it should be reported as for error 1.</p> |
| 29 | <p>AN IDLE SERVER WAS RELEASED</p> <p>Explanation: This status is used between the CDB and QMON as part of the TERM key-in processing. A message is issued to the user and the request will be retried. Once all idle servers are terminated, QMON will finish.</p> <p>Resolution: None.</p> |
| 30 | <p>BAD BASE/RETURN ON LBJ</p> <p>Explanation: A caller to the MCB had invalid information captured by the hardware related to the BDR on which the CDB is based and the return address portion of the linkage register.</p> <p>Resolution: See error code 1 if this error persists.</p> |
| 31 | <p>USER BUFFER ACW INVALID</p> <p>Explanation: A caller to the CDB passed an invalid buffer length/address for a function requiring data transfer between the program's DBANK and the CDB data areas.</p> <p>Resolution: See error code 1.</p> |

Chapter 20: Q-LINK DDP Errors

The following is a list of error codes, explanations of errors and possible error resolutions for all errors reported by QLDDP:

ERROR

CODE DESCRIPTION

- | | |
|------|--|
| 1200 | DDP INIT. BIND Error. Class: ### Detail: ###
Explanation: Q-LINK could not connect to the DDP-PPC facility. The Class and Detail status codes are listed in the Unisys manual OS2200 Distributed Data Processing (DDP-PPC/DDP-FJT) Messages Reference Manual (3787-3528). |
| 1201 | DDP INIT. OPEN Error. Class: ### Detail: ###
Explanation: Q-LINK issued an OPEN request that could not be completed by the DDP-PPC facility. The Class and Detail status codes are listed in the Unisys manual OS2200 Distributed Data Processing (DDP-PPC/DDP-FJT) Messages Reference Manual (3787-3528). |
| 1202 | DDP INIT. No REPLY received.
Explanation: This is a protocol error. The QLKDDP program did not reply correctly to the Q-LINK request to initiate a conversation.
Contact KMSYS Worldwide. |
| 1210 | DDP XFER. Expected OUTPUT EOF.
Explanation: This is a protocol error. Q-LINK was notified that no more data exists to be returned. Q-LINK is expecting a final status message to confirm the OUTPUT EOF status; but the received message was not the correct type.
Contact KMSYS Worldwide. |
| 1211 | DDP RECV. RECEIVE Error. Class: ### Detail: ###
Explanation: Q-LINK received an error from the DDP-PPC facility while attempting to receive data. The Class and Detail status codes are listed in the Unisys manual OS2200 Distributed Data Processing (DDP-PPC/DDP-FJT) Messages Reference Manual (3787-3528). |
| 1212 | DDP SEND. SEND Error. Class: ### Detail: ###
Explanation: Q-LINK received an error from the DDP-PPC facility while attempting to send data. The Class and Detail status codes are listed in the Unisys manual OS2200 Distributed Data Processing (DDP-PPC/DDP-FJT) Messages Reference Manual (3787-3528). |

**ERROR
CODE DESCRIPTION**

- 1213 DDP TOKEN. SEND TOKEN Error.
Explanation: An error occurred while Q-LINK was attempting to pass the control token to the other member of the conversation.
Contact KMSYS Worldwide.
- 1220 DDP EXIT. Receive CLOSE Error. Class: ### Detail: ###
Explanation: Q-LINK received an error from the DDP-PPC facility while attempting to close the current conversation. The Class and Detail status codes are listed in the Unisys manual OS2200 Distributed Data Processing (DDP-PPC/DDP-FJT) Messages Reference Manual (3787-3528).
- 1221 DDP EXIT. Expected CLOSE message.
Explanation: This is a protocol error. Q-LINK has received the final status message from remote, and was expecting a close message which was not received.
Contact KMSYS Worldwide.
- 1222 DDP EXIT. Send CLOSE Response failed. Class: ### Detail: ###
Explanation: Q-LINK received an error from the DDP-PPC facility while attempting to close the conversation. The Class and Detail status codes are listed in the Unisys manual OS2200 Distributed Data Processing (DDP-PPC/DDP-FJT) Messages Reference Manual (3787-3528).
- 1230 DDP-PPC Abort.
Explanation: This error can be caused when one party of DDP-PPC conversation is abnormally terminated.
- 1231 DDP Facility not configured.
Explanation: This message is issued when a request is made for remote execution of a Q LINK request and the DDP-PPC facility has not been configured or enabled. In order to enable the DDP-PPC facility, Q-LINK must be build with the required DDP-PPC files and the dynamic DDP switch must be set to ON. These fields are located on the COMUS configuration screen BUILD and BASIC1.
Contact your Q-LINK coordinator.

Index

- #EOF, 2-4
- #INSERT, 2-4
- \$ALPHA, 9-28
- \$DATE, 9-9, 9-11
- \$HIVAL, 9-28, 9-35
- \$LOVAL, 9-28, 9-35
- \$NUM, 9-28
- \$PBUFF, 9-18, 9-35
- \$SPACES, 9-28, 9-35
- \$TAB, 6-1
- @QLK, 2-2
- + symbol, 9-18
- A MAPPER RUN WAS RELEASED, 19-6
- A18, 4-2
- A30, 6-1
- A6, 4-2, 9-37
- A8, 6-1
- A9, 4-2, 9-37
- Abbreviations, 16-1
- ABORT OF A REQUEST SIGNALLED
REQUESTED FROM MAPPER, 19-2
- ACCEPT command, 9-2
- Access modes (PCIOS/SFS), 10-2
- ACCESS NOT ALLOWED (#INSERT MODE
EDITING), 18-5
- ACQUIRE command (DMS), 12-5
- ADD directive, 8-1
- ALL CLASSES LOCKED/UNLOCKED, 19-3
- ALL SERVER CLASSES LOCKED, 18-2
- ALPHA, 9-28
- Alphanumeric literals, 3-2
- Alphanumeric string variables, 5-1
- Alternate record area, 7-1
- AN IDLE SERVER WAS RELEASED, 19-6
- AND, 9-28
- Application Definitions, 17-6
- Applications, 9-7
- area key, 9-3
- AREA-KEY, 6-1
- area-name, 12-4
- AREA-NAME, 6-1
- ASCII packed, 4-3
- ASSIGNED SERVER VANISHED!, 18-3
- BAD BASE/RETURN ON LBJ, 19-6
- BAD RID - OPEN I/P TO READ, 18-3
- BANK NOT YET LOADED, 19-5
- Batch option (B), 17-3
- BIS DTM, 15-1
- BIS/Q-LINK Interface, 2-1
- BITMERG command, 9-3
- BITSPLIT command, 9-4
- braces, 1-3
- BREAK command, 9-5, 9-20
- BRKPT, 9-30
- Building an Application Definition, 17-6
- Bytes required, 4-3
- C-AKEY, 6-1
- C-AREA-NAME, 6-1
- CASE, 9-36
- CASE command, 9-6
- CDB ADDRESSING WINDOW PROBLEMS,
18-4
- CDB CONTINGENCY (INCL RELOAD), 19-5
- CDB FAILED INITIAL STATUS CHECKS,
18-4
- CDB ID CONSTANT VALIDATION FAILED,
18-1
- C-DBK, 6-1
- CDELETE command (PCIOC/SFS), 10-6
- Chapter descriptions, 1-1
- CLOSE, 9-30
- CLOSE command (BIS DTM), 15-4
- CLOSE command (DMS), 12-7
- CLOSE command (PCIOS/SFS), 10-7
- Column limits, 3-2
- Column specification, 9-18
- Command abbreviations, 16-1
- Command Editor/Executor, 3-1
- Command Line Format, 3-2
- Commands, 3-1
- Commands, BIS DTM
CLOSE, 15-4

- OPEN, 15-5
- READ, 15-6
- WRITE, 15-7
- Commands, DIO
 - DIO, 13-2
- Commands, DMS
 - ACQUIRE, 12-5
 - CLOSE, 12-7
 - DELETE, 12-8
 - DEPART, 12-9
 - DISPLAY, 12-10
 - FETCH1, 12-11
 - FETCH2, 12-12
 - FETCH3, 12-13
 - FETCH4, 12-14
 - FETCH5, 12-15
 - FETCH6, 12-16
 - FETCH7, 12-17
 - FIND1, 12-11
 - FIND2, 12-12
 - FIND3, 12-13
 - FIND4, 12-14
 - FIND5, 12-15
 - FIND6, 12-16
 - FIND7, 12-17
 - FREE, 12-18
 - IF, 12-19
 - IMPART, 12-20
 - INSERT, 12-21
 - KEEP, 12-22
 - MODIFY, 12-23
 - OPEN, 12-24
 - REMOVE, 12-25
 - SET CURRENT, 12-26
 - SET DBDN, 12-28
 - SET NON-FATAL, 12-29
 - STORE, 12-30
 - SUPPRESS clause, 12-31
- Commands, general
 - ACCEPT, 9-2
 - BITSPLIT, 9-4
 - BREAK, 9-5
 - CASE, 9-6
 - CONNECT, 9-7
 - CSF, 9-8
 - DATE, 9-9
 - DATESET, 9-11
 - DECIMAL, 9-16
 - DISCONNECT, 9-17
 - DISPLAY, 9-18
 - DO, 9-20
 - DUMP, 9-22
 - EDIT, 9-23
 - FACERR, 9-26
 - GO, 9-27
 - IF, 9-28
 - NITMERGE, 9-3
 - PCONTROL, 9-30
 - ROUND, 9-32
 - SCAN, 9-33
 - SET, 9-35
 - SHIFT, 9-40
 - STOP, 9-41
 - TABS, 9-42
 - TIME, 9-43
 - TRACE, 9-44
 - TRANSFER, 9-45
 - TRIMDISP, 9-46
 - TRIMEDIT, 9-47
 - WAIT, 9-48
 - WILDCARD, 9-49
- Commands, PCIOS/SFS
 - CDELETE, 10-6
 - CLOSE, 10-7
 - OPEN, 10-8
 - READ, 10-10
 - READNEXT, 10-11
 - REWRITE, 10-12
 - START, 10-13
 - WRITE, 10-14
- Commands, RDMS
 - RDMS, 14-2
 - RDMS+, 14-4
- Commands, SORT

- RELEASE, 11-2
- RETURN, 11-3
- SORT, 11-4
- Comments, 3-2
- Common Bank Errors, 19-1
- COMMON DATA BANK DIRECTED ERRORS, 18-6
- communications, 2-1
- COMP, 4-2, 9-37
- COMP-1, 4-2
- COMP-2, 4-2
- COMP4, 9-37
- COMP-4, 4-2
- COMPILE directive, 8-1
- Computational, 4-3
- Conditional expression, 9-28
- Conditional expressions, 9-20
- CONNECT, 9-17
- CONNECT command, 9-7
- CONSOLE, 9-30
- Contents, 1-1
- Continuations, 3-2
- Convert dates, 9-9, 9-11
- Convert time, 9-43
- C-O-R, 6-1
- CSF command, 9-8
- Current time, 9-43
- CURRPCOL, 9-36
- Data name reference, 4-4
- Data Storage Area (DSA), 4-5
- Data types, 4-2
- database key, 9-3, 9-4
- DATE, 9-9
- DATE command, 9-9
- DATE-NUM, 9-9
- DATESET command, 9-11
- DAY, 9-9
- DDP, 2-10
- DDP Errors, 20-1
- DDP EXIT. Expected CLOSE message., 20-2
- DDP EXIT. Receive CLOSE Error., 20-2
- DDP EXIT. Send CLOSE Response failed., 20-2
- DDP Facility not configured., 20-2
- DDP INIT. BIND Error., 20-1
- DDP INIT. No REPLY received., 20-1
- DDP INIT. OPEN Error., 20-1
- DDP RECV. RECEIVE Error., 20-1
- DDP SEND. SEND Error., 20-1
- DDP TOKEN. SEND TOKEN Error., 20-2
- DDP XFER. Expected OUTPUT EOF., 20-1
- DDP-PPC Abort., 20-2
- DECIMAL, 9-36
- DECIMAL command, 9-16
- Decimal numeric variables, 5-1
- Decimal precision, 5-1, 7-3
- DECPOS, 9-37
- DEF A, 9-37
- DEF FP, 9-37
- DEF N, 9-37
- DEFINE A, 5-1
- DEFINE F (PCIOS/SFS), 10-3
- DEFINE F directive (BIS DTM), 15-1
- DEFINE F directive (DIO), 13-1
- DEFINE FP, 5-2
- DEFINE K, 5-2
- DEFINE N, 5-1
- DEFINE RA, 7-1
- DEFINE RA (BIS DTM), 15-3
- DEFINE RDA, 7-3
- Define record field, 7-3
- DEFINE SUB, 7-4
- DELETE command (DMS), 12-8
- Demand option (D), 17-3
- DEPART command (DMS), 12-9
- DIAG directive, 8-2
- DIO, 13-1
- DIO command (DIO), 13-2
- DIRECT, 10-3, 10-8
- Direct I/O, 13-1
- Direct References, 4-1
- Directives, 3-1
 - ADD, 8-1
 - COMPILE, 8-1

- DEFINE A, 5-1
- DEFINE F (BIS DTM), 15-1
- DEFINE F (DIO), 13-1
- DEFINE F (PCIOS/SFS), 10-3
- DEFINE FP, 5-2
- DEFINE K, 5-2
- DEFINE N, 5-1
- DEFINE RA, 7-1
- DEFINE RA (BIS DTM), 15-3
- DEFINE RDA, 7-3
- DEFINE SUB, 7-4
- DIAG, 8-2
- INDEX, 8-2
- INIT, 8-2
- INVOKE (DMS), 12-1
- INVOKE (QINDEX), 17-4
- LISTOFF, 8-2
- LISTON, 8-2
- LOAD, 8-2
- OBJECT, 8-2
- PRINTER, 8-2
- RECORD (QINDEX), 17-4
- RETLOG, 8-3
- RUN, 8-3
- S\$PROC (QINDEX), 17-4
- SAVE, 8-4
- SHUTDOWN, 8-4
- DISCONNECT command, 9-17
- DISP, 4-2, 9-37
- DISP-1, 4-2
- DISP-2, 4-2
- DISPLAY command, 9-18
- DISPLAY command (DMS), 12-10
- Displaying numbers, 9-23, 9-47
- Displays, 9-46
- DMS, 12-1
- DO command, 9-20
- Documents, 1-2
- Double quotes, 3-2
- double vertical bars, 1-3
- DTM, 15-1
- DUMP command, 9-22
- DWTIME\$, 9-9
- DYNAMIC, 10-8
- ECHO, 9-30
- EDIT commands, 9-23
- Edits, 9-47
- EJECT, 9-30
- Ellipsis, 1-3
- ERROR CLOSING OUTPUT RID, 18-3
- ERROR LOCATING OR READING CONFIG
ELT, 19-3
- ERROR OPENING #INSERT RID, 18-5
- ERROR OPENING O/P RID, 18-3
- ERROR READING #INSERT RID, 18-5
- ERROR WRITING DATE LINE, 18-3
- ERROR WRITING HEADER LINE TO
OUTPUT RID, 18-4
- ERROR WRITING OUTPUT RID, 18-3
- ERROR-NUM, 9-14
- Errors, 2-3
 - Date conversions, 9-14
 - PCIOS, 10-15
- Examine a string, 9-33
- EXEC Applications, 9-7, 9-17
- EXEC CSF submissions, 9-8
- EXTEND, 10-8
- FACERR command, 9-26
- FETCH command (DMS), 12-11
- FETCH1 command (DMS), 12-11
- FETCH2 command (DMS), 12-12
- FETCH3 command (DMS), 12-13
- FETCH4 command (DMS), 12-14
- FETCH5 command (DMS), 12-15
- FETCH6 command (DMS), 12-16
- FETCH7 command (DMS), 12-17
- Field definition, 7-3
- FIELDATA option (F), 17-3
- FIELDATA packed, 4-3
- File types, 3-2
- Filename, 7-1, 9-45
- FIND command (DMS), 12-11
- FIND1 command (DMS), 12-11
- FIND2 command (DMS), 12-12
- FIND3 command (DMS), 12-13

- FIND4 command (DMS), 12-14
- FIND5 command (DMS), 12-15
- FIND6 command (DMS), 12-16
- FIND7 command (DMS), 12-17
- Floating-point literals, 3-2
- Floating-point numeric variables, 5-2
- Flow of control, 9-27
- Formats
 - Commands vs. Directives, 3-2
- FP1, 4-2, 9-37
- FP2, 4-2, 9-37
- FREE command (DMS), 12-18
- Get current time, 9-43
- GO commands, 9-27
- Gregorian, 9-9
- Guides, 1-2
- HIGHEST ADDRESS REQUIRED FOR CDB
IS GREATER THAN 0777000 (1ST PCT
BLK ADDR), 19-3
- HIVALS, 9-28, 9-35
- IF command, 9-28
- IF command (DMS), 12-19
- ILLEGAL CB FUNCTION, 19-5
- IMPART command (DMS), 12-20
- INDEX directive, 8-2
- INDEXED, 10-3
- Indexing, 4-1, 4-4
- INIT directive, 8-2
- INIT REQ AND SERVER ALREADY ACTIVE,
19-1
- Initialize option (I), 17-3
- INPUT, 10-8
- Input data, 2-4
- INSERT (DMS), 12-21
- INSUFFICIENT PRIVILEGE FOR REQUEST,
19-4
- INVOKE considerations, 12-3
- INVOKE directive (DMS), 12-1
- INVOKE directive (QINDEX), 17-4
- ISAM, 10-3
- italicized words, 1-2
- J-DAY, 9-9
- Julian, 9-9
- Kanji variables, 5-2
- KEEP command (DMS), 12-22
- Keywords, 16-1
- Labels, 3-2
- LASTPCOL, 9-36
- LENGTH, 9-37
- LINE # TOO BIG ON OPEN I/P RID, 18-3
- List all option (L), 17-3
- List only option (N), 17-3
- List source option (S), 17-3
- LISTOFF directive, 8-2
- LISTON directive, 8-2
- LOAD directive, 8-2
- LOCAL, 9-30
- LOVALS, 9-28, 9-35
- Manuals, 1-2
- MAPNUM, 4-2, 9-37
- MAPPER HAS ABORTED (E-KEY,...), 19-2
- MAPPER RELEASED BY QMON, 18-2
- MAX REQUESTS QUEUED FOR SERVER
CLASS, 18-2
- MAXIMUM NUMBER OF SERVERS ACTIVE,
19-1
- MAXIMUM SERVERS ALREADY ACTIVE FOR
CLASS, 19-2
- MODIFY command (DMS), 12-23
- MONTH, 9-9
- Multiple RUN directives, 2-4
- MUST HAVE QMON UNLIMITED PRIVILEGE
FOR REQUEST, 19-4
- NO BIS RUN TO RELEASE, 19-6
- NO CLASS QUEUE SLOT FOUND, 19-2
- Notation, 1-2
- NUM, 9-28
- Numeric literals, 3-2
- Numeric variables, 5-1
- OBJECT directive, 8-2
- Object program area, 4-6
- Object program considerations, 8-4
- Object programs, Q-LINK, 2-4
- OPEN command (BIS DTM), 15-5
- OPEN command (DMS), 12-24
- OPEN command (PCIOC/SFS), 10-8

OPTION, 9-30
Options
 QINDEX Processor call, 17-3
OR, 9-28
Other References, 1-2
OUTPUT, 10-8
Overwrite option (O), 17-3
Packed decimal, 4-3
PBUFF, 9-18, 9-35
PCIOS, 10-1
PCIOS errors, 10-15
PCONTROL command, 9-30
Print buffer, 9-18
Print control, 9-30
PRINTER directive, 8-2
Procedures, 9-20
Processor call
 QINDEX, 17-3
Program, 8-4
QINDEX, 17-1
QINDEX example, 17-7
QINDEX processor call, 17-3
Q-LINK TIMER ACTIVITY IS SHUTTING
 DOWN SERVER, 19-2
Q-LINK UNKNOWN ERROR, 19-3
Q-LINK/BIS Interface, 2-1
QLK Errors, 2-3, 18-1
QLK Function, 2-2
QLKFUN IMAGE LENGTH ERROR, 18-6
QLSIM Processor, 2-8
QMON, 2-1
QMON ALREADY ACTIVE, 19-4
QMON IS SHUTTING DOWN THIS SERVER,
 19-2
qualification, 4-4
QUEUE CLASS NOT FOUND, 19-4
Queue-alias, 7-1, 9-45
Quotes, 3-2
Q-UTIL, 2-5
RANDOM, 10-8
RDA, 4-1
RDA Definitions (RDMS), 14-5
RDA Direct Reference
RDA Initialization, 2-4
RDA named reference, 4-4
RDA variable reference, 4-5
RDMS, 14-1
RDMS command (RDMS), 14-2
RDMS RDA Definitions, 14-5
RDMS+ command (RDMS), 14-4
READ command (BIS DTM), 15-6
READ command (PCIOC/SFS), 10-10
READNEXT command (PCIOC/SFS), 10-11
Record area, 7-1
Record Delivery Area, 4-1
Record Delivery Area (RDA), 7-3
RECORD directive (QINDEX), 17-4
Record level moves, 9-45
record-name, 12-4
Record-name, 7-1, 9-45
RELEASE command (SORT), 11-2
Remote Q-LINK Execution, 2-9
REMOVE command (DMS), 12-25
Reply lines, 2-2
Request Stream, Q-LINK, 2-3
REQUESTER NOT IN QUARTER-WORD
 MODE, 19-5
Required bytes, 4-3
RETLOG directive, 8-3
RETURN command (SORT), 11-3
REWRITE command (PCIOC/SFS), 10-12
RID # TOO BIG FOR TYPE ON OPEN, 18-3
ROUND, 9-36
ROUND command, 9-32
RUN directive, 8-3
S\$PROC directive (QINDEX), 17-4
SAVE directive, 8-4
SB6, 4-2, 9-37
SB9, 4-2, 9-37
SCAN command, 9-33
Search for a string, 9-33
SECURITY ERROR - COMPILE NOT
 ALLOWED, 18-6
SECURITY ERROR - NO ACCESS TO
 CLASS, 18-6

- SECURITY ERROR - RUN NOT ALLOWED, 18-6
- SEQ, 10-3, 10-8
- Server class, 2-2, 2-8
- SERVER CLASS MAXIMUM XQT TIME EXCEEDED, 18-5
- SERVER CLASS QUEUE NOT IN MCT, 18-2
- SERVER INDEX OUT OF RANGE, 19-4
- SERVER NOT CLOSED; I.E., IS BUSY, 19-6
- SERVER NOT CLOSED; I.E., IS TERMINATED/ABORTED, 19-6
- SERVER REQUESTED ABORT OF REQUEST, 18-2
- SERVER TABLE NOT FOUND, 19-1
- SERVER(S) ARE ACTIVE, 19-5
- servers, 2-1
- Set a field, 9-35
- Set a variable, 9-35
- SET command, 9-35
- SET CURRENT command (DMS), 12-26
- SET DBDN command (DMS), 12-28
- SET NON-FATAL command (DMS), 12-29
- set-name, 12-4
- SFS, 10-1
- SHIFT command, 9-40
- SHUTDOWN directive, 8-4
- Single quotes, 3-2
- SN6, 4-2, 9-37
- SN9, 4-2, 9-37
- SORT, 7-1, 9-45, 11-1
- SORT command, 11-4
- SPACES, 9-28, 9-35
- SPECIFIED SERVER NOT ACTIVE, 18-1
- Spooling, 9-30
- Stage print buffer, 9-18
- START commands (PCIOC/SFS), 10-13
- STARTPOS, 9-36
- STAT1, 2 and 3, 2-3
- STOP command, 9-41
- STORE command (DMS), 12-30
- Subscripting, 4-1, 4-4, 7-4
- SUPPRESS clause (DMS), 12-31
- SYNTAX ERROR ON #INSERT DIRECTIVE, 18-5
- Syntax Notation, 1-2
- SYS MAX LINES REACHED, 18-4
- SYSTEM ERROR OPENING I/P RID, 18-3
- SYSTEM ERROR READING I/P RID, 18-4
- System name, 2-2
- Tab insertion, 9-42
- TAB insertion, 9-18
- TABS command, 9-42
- TAPE, 10-3
- TEMPLATE BANK ILL-FORMED, 19-5
- Testing a Q-LINK request, 2-5, 2-8
- THIS SERVER CLASS IS LOCKED, 18-3
- TIME, 9-9, 9-43
- TIME command, 9-43
- TOO MANY CHAINED #INSERTS ENCOUNTERED, 18-5
- TRACE command, 9-44
- TRANSFER command, 9-45
- TRIMDISP command, 9-46
- TRIMEDIT command, 9-47
- TRIMLEN, 9-36
- TYPE, 9-37
- UB6, 4-2, 9-37
- UB9, 4-2, 9-37
- UN6, 4-2, 9-37
- UN9, 4-2, 9-37
- UNABLE TO DO AN ER TO TRMRG\$, 19-3
- UNABLE TO TRMRG\$ MAPPER ACTIVITY, 18-4
- underlined word, 1-3
- UNKNOWN FUNCTION REQUESTED, 19-1
- UNTIL, 9-20
- UPDATE, 10-8
- UPPERCASE, 1-2
- Usage modes (PCIOC/SFS), 10-2
- USER ABORTED FUNCTION WITH MESSAGE WAIT KEY, 18-2
- USER BUFFER ACW INVALID, 19-6
- USER-SPECIFIED MAX LINES REACHED, 18-4
- Variable DSA, 4-5

Variable substitution, 4-5

Variables

 Alphanumeric, 5-1

 Floating-point, 5-2

 Kanji, 5-2

 Numeric, 5-1

vertical bar, 1-3

vertical bars, 1-3

WAIT command, 9-48

What is included?, 1-1

WHILE, 9-20

WILDCARD command, 9-49

Word reference option (W), 17-3

WRITE command (BIS DTM), 15-7

WRITE command (PCIOC/SFS), 10-14

YEAR, 9-9

If you would like to help us make our documentation better, please take a few moments to complete this form and return it to KMSYS Worldwide. We are always looking for ways to improve our products and your feedback will help us reach our goal.

Name _____

Company _____

Address _____

City _____ State/Province _____

Country _____ Zip/Mail Code _____

Document Name _____ OS Level _____

KMSYS Worldwide Product _____ Level _____

Please rate the documentation on a scale of 1 to 5:

	5	4	3	2	1	
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Incomplete
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Inaccurate
Usable	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Unusable
Readable	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Unreadable
Understandable	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Unintelligible
Attractive	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Unattractive
Excellent	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Poor

What information did you expect to find that was omitted?

Is more information needed? ☐ Yes ☐ No. If yes, on what topic?

Did you find factual errors in the documentation? ☐ Yes ☐ No. If yes, please give page number and description of the error.

If the documentation is difficult to understand, please specify page number and problem.

Is the documentation intimidating? ☐ Yes ☐ No.

Are the manuals: ☐ Too long? ☐ Too short? ☐ About the right length?

Other suggestions or comments? (Use back of form if necessary.)

(Additional Comments)

. Fold along dotted line.



WORLDWIDE, INC

**P.O. Box 669695
Marietta, GA 30066
U.S.A.**

Attn: Technical Documentation Section