MIGRATION RPG USER'S GUIDE

June 2011

Revision/Update Information:	This revised manual supersedes the Migration RPG User's Guide, Version 7.3.
Operating System and Version:	OpenVMS VAX Version 7.1 or higher
Operating System and Version:	OpenVMS Alpha Version 7.3 or higher
Operating System and Version:	OpenVMS Integrity Version 8.2 or higher
Software Version:	Migration RPG Version 8.3 or higher

Migration Specialties International   **Florence, Colorado**

First Printing, August 1996
Revised, March 1998
Revised, October 1999
Revised, January 2001
Revised, March 2002
Revised, January 2005
Revised, December 2005
Revised, June 2011

Contents

Preface

Intended Audience

The Migration RPG User's Guide is intended for programmers who are familiar with computer concepts and the RPG II programming language. Migration RPG was originally developed for users moving from IBM® System/36™ platforms to OpenVMS systems and was modeled after IBM System/36 RPG II. It has since been enhanced beyond the scope of IBM System/36 RPG II to provide a more generic and complete RPG environment under the OpenVMS operating system. Migration RPG still maintains complete IBM System/36 compatibility.

Objectives

This user's guide provides the information needed to compile and run programs using the Migration RPG Compiler. This compiler and its associated utilities have been developed by MSI to facilitate the migration and maintenance of IBM System/36 RPG II applications on OpenVMS systems. Migration RPG can also be used to run other vendor RPG applications under OpenVMS. The amount of effort necessary to migrate a third party vendor RPG application to OpenVMS will depend upon the level of Migration RPG compatibility inherent in the application.

For information on how to install the Migration RPG Compiler, see the *Migration RPG Installation Guide*.

Conventions Used In This Manual

The following conventions are used in this manual to describe commands and keystrokes:

Convention	Meaning
CTRL/X	This sequence indicates that the user must hold down the key labeled CTRL while pressing another key.
$\boxed{\text{PF1}}$ + $\boxed{\text{X}}$	This sequence indicates that the user must first press and release the key labeled PF1, then press and release another key.
$\boxed{\text{RETURN}}$ or <RETURN>	A key name is shown enclosed or within angle brackets to indicate a key on the keyboard to be pressed by the user.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format. The items are omitted because they are not important to the topic being discussed.
( )	In format descriptions, parentheses indicate that, if more than one option is chosen, the options must be enclosed in parentheses.
[ ]	In format descriptions, optional parameters in a command are denoted by square brackets. If a command delimiter, such as a comma or slash, is included within the square brackets, it is also optional. If the delimiter is outside the square brackets, it is required in the command line. Never include the square brackets in a command when typing it at a terminal.
BOLDFACED TEXT	Commands entered by the user at the terminal are printed in boldfaced type.

Associated Documents

Additional information concerning Migration RPG can be found in the following MSI manuals:

- *Migration RPG Language Reference Manual*
- *Migration RPG Screen Format Reference Manual*

Additional information concerning the conversion of IBM System/3™, System/34™, or System/36 RPG applications to an OpenVMS system can be found in the following MSI manuals:

- *OpenVMS S/3X Conversion Assistance Manual*
- *OpenVMS S/3X Conversion Tools User's Guide*

The following IBM manuals maybe helpful when converting IBM RPG II applications to run under OpenVMS using Migration RPG:

- *IBM System/3 RPG II Reference Manual* (Order Number SC21-7504)
- *IBM System/3 System Control Programming Reference Manual* (Order Number GC21-5077)
- *IBM System/34 Data File Utility Reference Manual* (Order Number SC21-7656)
- *IBM System/34 RPG II Reference Manual* (Order Number SC21-7667)
- *IBM System/34 Screen Design Aid Programmer's Guide and Reference Manual* (Order Number SC21-7716)
- *IBM System/34 Sort Reference Manual* (Order Number SC21-7658)
- *IBM System/34 System Support Reference Manual* (Order Number SC21-5155)
- *IBM System/36 Creating Displays: Screen Design Aid and System Support Program* (Order Number SC21-7902)
- *IBM System/36 Programming with RPG II* (Order Number SC21-9006)
- *IBM System/36 Sort Guide* (Order Number SC21-7903-2)
- *IBM System/36 System Reference* (Order Number SC21-9020)

Additional information concerning the use of OpenVMS can be found in the following manuals:

- *OpenVMS User's Manual*
- *Guide to Creating OpenVMS Modular Procedures*
- *OpenVMS DCL Dictionary*
- *Guide to OpenVMS File Applications*
- *OpenVMS Linker Utility Manual*
- *OpenVMS System Messages and Recovery Procedures Reference Manual*
- *OpenVMS EVE Reference Manual*

Additional information concerning RPG programming can be found in the following books:

- *Computer Programming - RPG II*, by Gary B. Shelly & Thomas J. Cashman
- *RPG and RPG II Programming; Applied Fundamentals, A Job Approach to Learning*, by Willian E. Bux & Edward C. Cunningham
- *RPG II Programming*, by Edward L. Essick

1 Introduction

1.1 Migration RPG Overview

MSI's Migration RPG has been developed to facilitate the migration of other vendor RPG applications to OpenVMS systems and to support existing OpenVMS RPG applications on all available OpenVMS platforms. It is modeled after IBM's RPG II programming language, and is fully compatible with Release 6.0 of that product. Migration RPG also contains some additional opcodes and functionality not found in IBM RPG II. The MSI Migration RPG Compiler Kit includes the Migration RPG compiler, a screen format generator, and several support utilities.

The screen format generator (SFG) compiles S, H, and D specifications modeled after those used on the System/36 to generate interactive screens for display on a CRT. The specifications can be generated and maintained using the full screen RPG editor supplied with the Migration RPG Compiler Kit.

The Migration RPG compiler parses RPG source specifications and produces OpenVMS native-mode object modules. These modules can then be linked to produce a native-mode executable image. The modules are linked to the Migration RPG Runtime System, which must be installed at system boot time. If RPG applications are moved to another OpenVMS system, a Migration RPG Runtime License must be purchased for the new system and installed on that system in order to run the RPG applications.

Interactive RPG program terminal support is available on all VT series terminals and terminal emulators. Support is also available for terminals and devices using OpenVMS windowing environments via DECterm windows.

Migration RPG should not be confused with VAX RPG II, which is a retired former Digital Equipment Corporation product designed to run on VAX processors. Migration RPG can be used to migrate VAX RPG II applications to run on any OpenVMS platform.

2 Getting Started

2.1 Overview

Before the MSI Migration RPG compiler and utilities can be used, they must be installed correctly. The Migration RPG Compiler Kit should be installed by the system manager using the **OpenVMS Install Utility**. See the *Migration RPG Installation Guide* for installation instructions.

Each user needs to set up the commands and symbols associated with Migration RPG when they log in. This is accomplished by placing the following command in the user's LOGIN.COM file:

```
$ @S3X$RPG:RPGINSTAL
```

If a large number of users are going to use Migration RPG, the system manager may elect to place this procedure call in a group login procedure or in the system SYLOGIN.COM procedure.

Once the user set up procedure has been run, the user is ready to start using the Migration RPG software.

If a batch procedure is created to run RPG applications, it too must run the RPGINSTAL procedure to create the symbols, logicals, and data files associated with Migration RPG. Be sure that the RPGINSTAL procedure is placed in the user or system login in such a way that it is run for both interactive and batch logins.

2.2 Using A Logout Procedure To Do Account Cleanup (LOGOUT.COM)

The Migration RPG RPGINSTAL procedure creates two single-record sequential data files. One is used for external indicators (U1 - U8) and the system date; the other is used to represent the Local Data Area. New versions of these files are created each time the user logs in. The files are uniquely identified by the user's process ID and are associated to the logical names S3X$EXT and S3X$LDA.

There is no automated mechanism for deleting these files when the user logs off the system. As time goes on, the user will discover that many versions of these two files are being retained since they are recreated each time the user logs in. Purging the files is not effective because the files are always created with unique names. It is recommended that the user develop or be given a logout procedure to complement the user's login procedure. This procedure can be associated to a logout command via a symbol definition or incorporated into the process termination procedures for users running from captive accounts. An example of a logout symbol and procedure follows.

Example 2–1 Logout Symbol and Procedure

```
$!  Place this symbol definition in the user's LOGIN.COM procedure.
$!
$      LO :== @S3X$RPG:LOGOUT.COM
   .
   .
   .
$!                        LOGOUT.COM
$!  Logout command procedure used to delete Migration RPG
$!  external indicator file and Local Data Area.  Place this
$!  procedure in the user's default login area (SYS$LOGIN).
$!
$ DELETE 'F$TRNLNM("S3X$EXT")';0, 'F$TRNLNM("S3X$LDA")';0
$!
$ LOGOUT
```

A sample LOGOUT.COM procedure is included with the Migration RPG Compiler Kit and is located in the S3X$RPG directory.

3 Compiling An RPG Program

3.1 Overview

This chapter explains how to compile an RPG program using MSI's Migration RPG compiler. It describes the qualifiers and the features available with the compiler.

NOTE: **See Chapter 5, "Linking And Running Migration RPG Programs", Section 5.4, for information on the BUILD command, which combines the steps of compiling and linking RPG programs and screens.**

It is important to remember that Migration RPG is *source code compatiblel* across all platforms that support OpenVMS. Object and image files created under one OpenVMS architecture (VAX, Alpha, or Integrity) will not link or run under other OpenVMS architectures. To port RPG programs to a new OpenVMS architecture, the source code must be transferred, recompiled and relinked. A Migration RPG Compiler Kit is required for each architecture on which the RPG programs are to be run.

3.2 RPG Program Compilation

Before it is possible to run an RPG program, an RPG source file must first be compiled, then linked. The Migration RPG compiler will syntax check an RPG program and produce an object module if the program compiles without errors. If warning or error messages are generated, they are displayed along with the line of code that generated them. If a listing is requested with the compile, warning and error messages are included in the listing following the line of code that generated them.

The compiler is invoked with the command RPG.

RPG

FORMAT

RPG *filename*

PARAMETERS

filename

Name of the program source file to be compiled. This can be a fully qualified OpenVMS file description. The first character in the file name cannot be a number (0 - 9). The compiler assumes a file type of .RPG unless the file type is explicitly stated in the input line.

DESCRIPTION

The Migration RPG compiler is used to compile an RPG source file and produce an OpenVMS object file. Qualifiers are used to control the output and functions of the RPG compiler. The qualifiers are position-independent, meaning they can be specified in any order following the RPG command or the file name.

QUALIFIERS

/END (Default)
/NOEND

By default, the Migration RPG compiler produces object files with a defined transfer address. When linking two or more RPG programs together, each with an established transfer address, the following warning message will be issued by the linker:

Example 3–1 Transfer Address Link Warning Message

```
$ LINK BOSS, PEON1, PEON2
%LINK-W-MULTFR, multiply defined transfer address
        in module PEON1 file PEON1.OBJ;1
%LINK-W-MULTFR, multiply defined transfer address
        in module PEON2 file PEON2.OBJ;1
   .
   .
   .
```

This warning message has no impact on the user’s ability to run the linked image and can be ignored. However, should the programmer wish to eliminate the link-time message, this can be accomplished by compiling the subprograms being linked into the image using the /NOEND qualifier. To eliminate the link warning messages in the above example, the program BOSS would be compiled normally, while the programs PEON1 and PEON2 would be compiled using the /NOEND qualifier.

RPG programs compiled using the /NOEND qualifier can only be linked into an image as subprograms. They cannot be linked as stand-alone images.

/LIST[=filename]
/NOLIST (Default)

Controls whether a list file is produced when the RPG program is compiled. By default, the Migration RPG compiler does not produce a listing. The list file can optionally be given a different name by specifying the file name following the /LIST qualifier. The list file name can be a fully qualified OpenVMS file description. If no list file name is specified, the list file is given the program name and a .LIS file type.

If /NOLIST is specified, the compiler will not produce a program listing.

Example 3–2 Compile Command With /LIST Qualifier

```
$ RPG /LIST JAZZ
```

This command compiles the program JAZZ.RPG to produce an object file labeled JAZZ.OBJ and a listing file labeled JAZZ.LIS.

/OBJECT[=filename] (Default)
/NOOBJECT

By default, the Migration RPG compiler will produce an object module with the program name and an .OBJ file type if no errors are found during compilation. An object file with a different name can be generated by specifying a file name following the /OBJECT qualifier. The object file name can be a fully qualified OpenVMS file description. If /NOOBJECT is specified, the compiler will simply parse the source file and report any warnings or errors. This qualifier is useful for quick syntax checking of RPG source code.

Example 3–3 Compile Command With /OBJECT Qualifier

```
$ RPG /OBJECT=[PARTY]FOXY.OBJ REDDRESS
```

This command compiles the program REDDRESS.RPG to produce the object file FOXY.OBJ in the directory [PARTY].

/PAGE=nnn

Specifies the number of lines per page to print on the output listing generated by the Migration RPG compiler. By default, the compiler assumes 60 lines per page. The /PAGE qualifier can be specified in a range from 10 to 200. Invalid entries are ignored and the default value of 60 is used. If the /LIST qualifier is not specified or the /NOLIST qualifier is used, the /PAGE qualifier is ignored.

/VERSION

This qualifier displays the version number of the Migration RPG compiler currently in use.

/WARNING (Default)
/NOWARNING

Migration RPG programs are capable of displaying warning messages at runtime when a condition occurs which has the potential to cause a problem. An example of this would be the passing of a parameter from a calling program to a called program, where the size of the parameter in the calling program does not match the size defined for the parameter in the called program. In a case like this, a warning message will be displayed to the terminal informing the user of the potential problem.

Should the programmer wish to disable Migration RPG's ability to display warning messages at runtime, this can be accomplished by using the /NOWARNING qualifier at compile time.

NOTE

Qualifier names can be abbreviated to a minimum number of unique characters. If a qualifier is entered more than once on a command line, the last entry takes precedence over all previous entries. For example, in the command:

```
$ RPG /LIST WHATLST /NOLIST
```

the program WHATLST.RPG would be compiled, but no compile time listing would be produced since the /NOLIST qualifier would supersede the /LIST qualifier.

3.3 Compiler Directives

Compiler directives can be included in an RPG source file to include additional source modules into the main program at compile time or provide output format directives concerning the optional program listing the compiler generates.

All compiler directives begin with a slash (/) in column 7. Compiler directives can appear anywhere in an RPG program. Column 6 can be blank when a compiler directive is specified. The following compiler directives are recognized by Migration RPG:

- ***/COPY Directive***

  The /COPY directive is used to copy additional modules of RPG source code into a program at compile time. The RPG name of the file containing the RPG source that is to be included should start in column 13. The source file name is assumed to end at the first blank encountered.

  The source file name can be a fully qualified OpenVMS file description. The /COPY directive assumes a file type of .RPG for any source module referenced by a /COPY directive if no file type is provided.

  The contents of the copybook referenced by the /COPY directive will be read and inserted into the RPG program being compiled at the point of the /COPY directive. This allows the programmer to code large programs in modules and copy the modules into programs at the appropriate places at compile time.

Example 3–4 /COPY Directive Usage

```
         1         2         3         4         5
12345678901234567890123456789012345678901234567890123456789012345678
      /COPY USER1:GOODSTUFF.COPYBOOK
```

  In this example, the logical USER1 represents a directory on the system. The RPG source module GOODSTUFF.COPYBOOK, which is located in the directory identified by USER1, will be read into the RPG program at the point of the /COPY statement at compile time. The contents of the file GOODSTUFF.COPYBOOK will be compiled as part of the program.

  Migration RPG also supports the /COPY statement using the Auto Report Utility. See Chapter 10, "Auto Report Utility - AUTOC", in this manual for more information.

- ***/EJECT Directive***

  The /EJECT directive can be used to force form-feeds in a compiler listing. When the /EJECT directive is encountered by the Migration RPG compiler, it will place a form-feed in the program listing file and start a new page. If no listing has been specified for the compile, the /EJECT directive is ignored.

Example 3–5 /EJECT Directive Usage

```
         1         2         3         4         5
12345678901234567890123456789012345678901234567890123456 78
      *
0086 I                                    236 2430NXTRRN
0087 I                                    244 2460GRADE
0088 I                                    247 247 RCODE
0089 I                                    248 248 OP
     C/EJECT
0090 C                SETOF                        46
0091 C   99           MOVE *BLANK    NAMET  40
0092 C   99           EXSR LDAIN
```

In this example, the /EJECT directive is being used to force a new page in the program listing at the beginning of the Calculation specifications.

- ***/SPACE Directive***

  The /SPACE directive can be used to force blank lines in a compiler listing. The programmer specifies the number of blank lines to be inserted immediately following the /SPACE directive. When the /SPACE directive is encountered by the compiler, it will insert the specified number of lines in the listing file. By default, the /SPACE directive will insert one blank line in a listing. If no listing has been specified for the compile, the /SPACE directive is ignored.

Example 3–6 /SPACE Directive Usage

```
         1         2         3         4         5
12345678901234567890123456789012345678901234567890123456789012345678
      *
0050 FDIFFHISTIC  F  96  96R           DISK
0051 FREPORT  O   F 132 132     OF     PRINTER
      /SPACE 4
0052 E                    TABA    8   8  2  A
0053 E                    #ERR    1   2 41
0054 E                    TST    50  50  1
0055 E                    ANS        50  1
      /SPACE 4
0056 IWORKSTN NS  99   1 C
0057 I       OR   01   1 C1
0058 I                                     2  41 NAME
     I/SPACE 2
0059 IDIFFHISTNS
0060 I                                     1  79 PROMPT
```

In this example, the /SPACE directive is being used to insert blank lines between specification sections in a program listing. Four (4) blank lines will be inserted preceding the first E and I specifications. Two (2) blank lines will be inserted preceding the DIFFHIST record definition.

- ***/TITLE Directive***

  The /TITLE directive can be used to force a programmer-specified heading line to appear at the beginning of each page in a compiler listing. Columns 14 - 74 can be used to insert the text which the programmer wishes to display. When the /TITLE directive is encountered by the compiler, it will insert the specified text in the listing heading area, place a form-feed in the program listing, and start a new page, displaying the programmer-specified text on the page heading. The text specified in the /TITLE directive will appear on each subsequent page unless modified by the programmer with another /TITLE directive. If no listing has been specified for the compile, the /TITLE directive is ignored.

Example 3–7 /TITLE Directive Usage

```
         1         2         3         4         5
12345678901234567890123456789012345678901234567890123456
      /TITLE Test compile of ENTPRS subroutine code
0001 H
0002 H****************************************************
0003 H*                                                  *
0004 H* PROGRAM NAME-START                               *
0005 H* FUNCTION-INTRO TO DATA PROCESSING TEST # 1       *
```

In this example, the /TITLE directive is used to insert the text "Test compile of ENTPRS subroutine code" in the page headings of the program listing.

3.4 Compiler File Designations and Defaults

All file names used by the compiler are standard OpenVMS file designations. A file designation can include a device name, directory specification, file name, type, and version number. If portions of the file designations are omitted, certain appropriate values (default values) are assumed for the missing portions. The assumed device is the user's system device (SYS$DISK:). The assumed directory is the current default directory. The assumed file types are .RPG for the source, .LIS for the listing, and .OBJ for the object file. On output files, the assumed version number is the latest version number, plus one.

3.5 RPG Error Handling

During program compilation, the Migration RPG compiler can generate two types of messages: warnings and errors. Warning messages are generated to bring something to the programmer's attention that may cause a problem, but is not severe enough to abort the generation of an object file. Error messages indicate a problem severe enough to abort the generation of an object file. A program must compile error free before an object module will be generated.

During compilation, the compiler lists all error and warning messages to the terminal. If a listing has been selected, the messages will be displayed in the listing immediately following the line of RPG source code that generated them. The message and the line which generated it are always displayed on the terminal from which the compilation was invoked, regardless of whether or not a listing was selected. At the end of each listing, a message will be displayed stating the number of errors that were found. If errors were found, the count is also logged to the invoking terminal.

Most of the generated error messages are self-explanatory. If more information on an error message is needed, summaries of the compile time warning and error messages are given in Appendix B, "Migration RPG Compiler Warning Messages", and Appendix C, "Migration RPG Compiler Error Messages", in this manual.

During compilation, standard OpenVMS error messages will be displayed if file errors or other problems are encountered. For more information on these error messages, refer to the *OpenVMS System Messages and Recovery Procedures Reference Manual*.

3.6 Programming Considerations

Some RPG programming considerations are discussed in the following sections.

3.6.1 Linking RPG Programs Together

Using the CALL, FREE, PLIST, PARM, and EXTRN opcodes, it is possible to link RPG programs together and pass data back and forth between them. Migration RPG code is fully re-entrant and there are no restrictions on a subprogram calling a main level program or calling another subprogram that is at a higher level. For example, program A calls program B which then calls program C. Program C can call program A, program B, itself, or a fourth program. If RPG applications are developed using re-entrant code, extreme caution is advised, since locating errors in a re-entrant application may be very difficult.

For more information on linking RPG programs together, see Chapter 5, "Linking And Running Migration RPG Programs", in this manual and the *Migration RPG Language Reference Manual*.

3.6.2 Debugging an RPG Program

Migration RPG does not support the use of the OpenVMS Symbolic Debugger. Debugging of RPG programs must be done using the DEBUG opcode within the RPG program. See the *Migration RPG Language Reference Manual* for more information on the DEBUG opcode.

3.6.3 Logical and Physical Files

Files specified in the RPG program File (F) specifications can be referenced by physical or logical file name. For more information on naming conventions and file access methods, see Chapter 11, "File Names and Conventions", in this manual.

3.6.4 READP Opcode

When using the READP opcode to process an indexed file, the key fields in the indexed file must be defined correctly or the function will fail. See Chapter 11, "File Names and Conventions", and the *Migration RPG Language Reference Manual* for more information on the indexed file definitions used with READP.

4 Compiling Workstation Screens (SFG Utility)

4.1 Overview

Workstation screens are developed through the coding of Screen (S), Help (H), and Field Definition (D) specifications. MSI's Migration RPG uses specifications modeled on those used on the System/36. These coding specifications are described in detail in the *Migration RPG Screen Format Reference Manual*.

This chapter explains how to compile a workstation screen using the Migration RPG Screen Format Generator. It describes the qualifiers and the features available with the screen compiler.

NOTE: See Chapter 5, "Linking And Running Migration RPG Programs", Section 5.4, for information on the BUILD command which combines the steps of compiling and linking RPG programs and screens.

It is important to remember that object and image files are different under each OpenVMS architecture (VAX, Alpha, and Integrity). Programs which run under one OpenVMS architecture will not run under any of the others without being recompiled and relinked. If you would like to run RPG programs under multiple OpenVMS architecture, you will need a Migration RPG Compiler Kit specific to each architecture.

4.2 Workstation Screen Compilation

Before a workstation screen can be displayed, it must first be compiled and then linked to a program. Screens which link to Migration RPG programs are compiled using the Migration RPG Screen Format Generator, SFG. This utility will compile the screen format specifications and produce a linkable object module. The object module is then linked to an interactive RPG program to provide the program with its screen specifications and screen handling routines.

All application and help screens used by an RPG program must be located in one source file. Multiple screen files cannot be linked together after they have been compiled. In the case of multiple interactive RPG programs being linked together, all of the screen source members must be combined into one file and compiled to produce one object file.

The Screen Format Generator is invoked with the command SFG.

SFG

FORMAT

SFG *filename*

PARAMETERS

filename
Name of the screen specification file to be compiled. The file name can be a fully qualified OpenVMS file description. The first character in the screen file name cannot be a number (0 - 9). The compiler assumes a file type of .FRM unless explicitly stated in the input line. The standard format for screen specification file names is:

(INDENT\ 1)program-nameFM.FRM

For example:

Using standard naming conventions, the interactive program LASER.RPG would be associated with a screen specification file labeled LASERFM.FRM.

DESCRIPTION

The Migration RPG SFG compiler is used to compile a workstation screen source file and produce an OpenVMS object file. Qualifiers are used to control the output and functions of the SFG compiler. The qualifiers are position-independent, meaning they can be specified in any order following the SFG command or the file name.

QUALIFIERS

/LIST[=filename]
/NOLIST (Default)
Controls whether a list file is produced when the screen format file is compiled. By default, the SFG compiler does not produce a listing. The list file can optionally be given a different name by specifying the file name following the /LIST qualifier. The list file name can be a fully qualified OpenVMS file description. If no list file name is specified, the list file is given the screen format file name and a .LIS file type.

If /NOLIST is specified, the compiler will not produce a program listing.

Example 4–1 Screen Compile Command With /LIST Qualifier

```
$ SFG JAZZFM /LIST
```

This command compiles the screen format file JAZZFM.FRM to produce an object file labeled JAZZFM.OBJ and a listing file labeled JAZZFM.LIS.

/OBJECT[=filename] (Default)
/NOOBJECT

By default, the screen compiler will produce an object module with the screen format file name and an .OBJ file type if no errors are found during compilation. An object file with a different name can be generated by specifying a file name following the /OBJECT qualifier. The object file name can be a fully qualified OpenVMS file description. If /NOOBJECT is specified, the screen compiler will simply parse the source file and report any warnings or errors. This qualifier is useful for quick syntax checking of screen specification source code.

Example 4–2 Screen Compile Command With /OBJECT Qualifier

```
$ SFG /OBJECT=FOXYFM.OBJ REDRESSFM
```

This command compiles the screen specification file REDRESSFM.FRM to produce the object file FOXYFM.OBJ.

/PAGE=nnn

Specifies the number of lines per page to print on the output listing generated by the SFG compiler. By default, the compiler assumes 60 lines per page. The /PAGE qualifier can be specified in a range from 10 to 200. Invalid entries are ignored and the default value of 60 is used. If the /LIST qualifier is not specified or the /NOLIST qualifier is used, the /PAGE qualifier is ignored.

NOTE

Qualifier names can be abbreviated to a minimum number of unique characters. If a qualifier is entered more than once on a command line, the last entry takes precedence over all previous entries. For example, in the command:

```
$ SFG /LIST WHATLSTFM /NOLIST
```

the screen format file WHATLSTFM.FRM would be compiled, but no compile time listing would be produced since the /NOLIST qualifier would supersede the /LIST qualifier.

4.3 Screen Compiler File Designations and Defaults

All file names used by the compiler are standard OpenVMS file designations. A file designation can include a device name, directory specification, file name, type, and version number. If portions of the file designations are omitted, certain appropriate values (default values) are assumed for the missing portions. The assumed device is the user's system device (SYS$DISK:). The assumed directory is the current default directory. The assumed file types are .FRM for the source, .LIS for the listing, and .OBJ for the object file. On output files, the assumed version number is the latest version number, plus one.

4.4 Screen Format File Error Handling

During program compilation, the Migration RPG Screen Format Generator can generate two types of messages: warnings and errors. Warning messages are generated to bring something to the programmer's attention that may cause a problem, but is not severe enough to abort the generation of an object file. Error messages indicate a problem severe enough to abort the generation of an object file. A screen format file must compile error free before an object module will be generated.

During compilation, the screen compiler lists all error and warning messages to the terminal. If a listing has been selected, the messages will be displayed in the listing immediately following the screen specification line that generated them. The message and the line which generated it are always displayed on the terminal from which the compilation was invoked, regardless of whether or not a listing was selected. At the end of each listing, a message will be displayed stating the number of errors that were found. If errors were found, the count is also logged to the invoking terminal.

Most of the generated error messages are self-explanatory. If more information on an error message is needed, summaries of the screen compile time warning and error messages are given in Appendix D, "Screen Format Generator (SFG) Warning Messages ", and Appendix E, "Screen Format Generator (SFG) Error Messages", in this manual.

During compilation, standard OpenVMS error messages will be displayed if file errors or other problems are encountered. For more information on these messages, refer to the *OpenVMS System Messages and Recovery Procedures Reference Manual*.

5 Linking And Running Migration RPG Programs

5.1 Overview

This chapter addresses the linking and running of batch and interactive RPG programs. Before a program can be run, it must first be compiled and linked. The compiler processes a source file to produce an object file. The linker processes the object file to produce an executable image. The executable image can then be run on any compatible OpenVMS system that has the same or a higher version of the Migration RPG Runtime System and OpenVMS installed.

It is important to remember that Migration RPG is *source code compatiblel* across all platforms that support OpenVMS. Object and image files created under one OpenVMS architecture (VAX, Alpha, or Integrity) will not link or run under other OpenVMS architectures. To port RPG programs to a new OpenVMS architecture, the source code must be transferred, recompiled and relinked. A Migration RPG Compiler Kit is required for each architecture on which the RPG programs are to be run.

The OpenVMS Linker Utility links the program object file and any associated object files (screen specifications, subroutines, other programs, etc.) to the Migration RPG and OpenVMS runtime systems. This process produces an executable image (.EXE) file which can then be run by the user. If the user wishes to move the executable image to another OpenVMS system and run it, it will be necessary to install a Migration RPG Runtime System on the secondary OpenVMS system to run the Migration RPG images.

All executable images created by compiling and linking RPG programs are fully shareable and can be installed on the system as shareable images.

Migration RPG allows multiple RPG programs to be linked together to form one image. When linking multiple RPG programs together, only one screen object module can be linked into the image. This means that all of the screen source members must be combined in one file and compiled by the Screen Format Generator to produce a single object module.

See the *OpenVMS Linker Utility Manual* for more information on the OpenVMS Linker Utility.

5.1.1 Alpha and Integrity Image Link Limitation

Migration RPG object modules created on an Alpha and Integrity systems cannot be linked with translated or vested images. This limitation is imposed by HP software and cannot be directly addressed by MSI; therefore, MSI cannot predict when this limitation might be lifted. If this limitation impacts your applications, please let us know.

5.2 Linking RPG Batch Programs

Once an RPG batch program has been successfully compiled, it can be linked using the OpenVMS LINK command. Any called programs or user-written subroutines which are referenced by the program must be included in the link command.

Example 5–1 LINK Command

```
$ LINK DATACRUNCH
```

In this example, the program object file DATACRUNCH.OBJ will be linked, producing the executable image DATACRUNCH.EXE. DATACRUNCH.EXE can then be run by the user.

Example 5–2 LINK Command Including Subroutine Object File

```
$ LINK HIVE,WRKRB
```

In this example, the program object file HIVE.OBJ and the user-supplied subroutine object file WRKRB.OBJ will be linked to produce the executable image HIVE.EXE

Example 5–3 LINK Command Including Called Programs

```
$ LINK SPACE_SHOT,ROCKET,ASTRONAUT
```

In this example, the programs SPACE_SHOT, ROCKET, and ASTRONAUT have all been linked together. The executable image will be labeled SPACE_SHOT.EXE. When linking multiple RPG programs together, the main or first level program should always be placed at the beginning of the list of program names. All three programs must have been compiled without errors prior to executing the link command.

5.3 Linking an Interactive RPG Program

When linking an interactive RPG program, the screen specification object file must be included in the command line. As the following example illustrates, if this file is not included, the linker will generate a series of undefined symbol warnings and the program will abort when it is run.

Example 5–4 Invalid Link of an Interactive RPG Program, Screen Module Missing

```
$ LINK KERBLAST
%LINK-W-NUDFSYMS, 1 undefined symbol:
%LINK-I-UDFSYM,         N_SCREEN_SPEC_TBL
%LINK-W-USEUNDEF, undefined symbol N_SCREEN_SPEC_TBL referenced
        in psect $CODE offset %X00008E77
        in module KERBLAST file $255$DUA8:[USERX]KERBLAST.OBJ;3
%LINK-W-USEUNDEF, undefined symbol N_SCREEN_SPEC_TBL referenced
        in psect $CODE offset %X00008EB7
        in module KERBLAST file $255$DUA8:[USERX]KERBLAST.OBJ;3
   .
   .
   .
```

Multiple screen object files cannot be linked together. All screen source members associated to an image must be combined into one file and compiled to produce a single object module. If two or more screen object modules are linked into an image, the linker will issue a series of warning messages and the program will abort when it is run. The following example illustrates the results of linking two screen modules with a program module:

Example 5–5 Invalid Link of an Interactive RPG Program, Too Many Screen Modules

```
$ LINK PHASER1, PHASER1FM, PHASER2FM
%LINK-W-MULDEF, symbol F$SBUF multiply defined
 in module SCREEN_MODULE file $255$DUA1:[FRED]PHASER2FM.OBJ;1
%LINK-W-MULDEF, symbol F$SSIZ multiply defined
 in module SCREEN_MODULE file $255$DUA1:[FRED]PHASER2FM.OBJ;1
%LINK-W-MULDEF, symbol N_SCREEN_SPEC_TBL multiply defined
 in module SCREEN_MODULE file $255$DUA1:[FRED]PHASER2FM.OBJ;1
   .
   .
   .
```

The screen object file must be linked with the RPG program doing I/O to the workstation device. The screen object file is linked with the RPG program object file in the same manner as an external subroutine or called program.

The following is an example of the compile and link commands necessary to generate an RPG program that uses a workstation screen.

Example 5–6 Compile and Link Commands for RPG Program and Screen

```
$ RPG TIGER
$ SFG TIGERFM
$ LINK TIGER, TIGERFM
```

In this example, the RPG program TIGER is linked with the screen file TIGERFM to produce the interactive executable image TIGER.EXE.

Called programs and external subroutines can also be referenced by interactive RPG programs. The object files of the subroutines must be included in the link command in the same manner in which they are included in a batch RPG program link.

5.4 BUILD Command

The BUILD command initiates a command procedure which will carry out any or all of the following functions:

- Compile an RPG program
- Compile a workstation screen
- Link a program
- Link a program and workstation screen

BUILD

FORMAT **BUILD** *filename [list clean_up compile_select]*

DESCRIPTION

The BUILD command can be used to compile and link an RPG program with one command. Parameters are used to control the output and functions of the BUILD command. The parameters are position-dependent and must be specified in the order listed above. If an intermediate parameter is not used, a null parameter ("") must be specified to take its place. The file name parameter is the only required parameter; all other parameters are optional.

PARAMETERS

filename

Name of the program source file to be compiled. This can be a fully qualified OpenVMS file description, excluding the file type and version number. The BUILD procedure will assume a file type of .RPG and a version number of ;0 (current version). This is a required parameter and will be prompted for if it is not specified.

list options: Y
N (Default)

Specify a Y as the list parameter if compile-time listings are to be generated by the Migration RPG and SFG compilers within the BUILD procedure. The default is to produce no compile-time listings.

clean_up options: PURGE
DELETE

If the PURGE option is selected, the BUILD procedure will purge all versions of the program and screen object and executable files using the following PURGE command after completing the compile and/or link operations:

```
$ PURGE filename.obj, filename.exe
```

If the DELETE option is selected, the BUILD procedure will delete all object program and screen object files using the following DELETE command after completing the compile and/or link operations:

```
$ DELETE filename.OBJ;*
```

By default, the BUILD procedure does not carry out any purge or delete operations.

compile_select options: RPG (Default)
SCR
LINK

By default, the BUILD procedure will compile the specified program, compile the associated workstation screen file if the program is a workstation program, and link the resulting program and screen object modules to create an executable image. If only one of these actions is desired, use this parameter to indicate the desired action:

- **RPG -** Only compile the specified RPG program.
- **SCR -** Only compile the workstation screen associated to the specified program.
- **LINK -** Only link the specified RPG program.

The BUILD procedure executes the following steps when run without special parameters:

1 The RPG program is located and searched. If /COPY compiler directives are found, the Migration RPG Auto Report Utility is selected to generate and compile the program. The Auto Report Utility is invoked with the AUTOC command. If no /COPY compiler directives are found, the Migration RPG compiler is used to compile the program.

   If warnings or errors are encountered during the compile, they are logged to the user's screen. If the list parameter has been specified, the messages are also logged to the compiler listing file.

2 If the program being compiled is a workstation program, the BUILD procedure will attempt to locate and compile the workstation screen file after the program source module has been compiled. The BUILD procedure locates the workstation screen file using the following process:

   - The RPG program will be searched for an FMTS continuation specification. If found, the file name defined by the FMTS specification will be used as the workstation screen file name.
   - If an FMTS continuation specification is not present, the procedure will search for a workstation screen file using the program file name. It will search for the workstation screen file using the following formats:

     — filenameFM.FRM

     — filenameSC.FRM

3 If a workstation screen file is located, it is compiled. If warnings or errors are encountered during the compile, they are logged to the user's screen. If the list parameter has been specified, the messages are also logged to the compiler listing file.

4 The BUILD procedure links the compiled RPG program and the workstation screen file, provided one was present, producing an executable image. If compile time errors were encountered during the BUILD process, the link command is bypassed.

The following examples depict how the BUILD command can be used:

Example 5–7 Using BUILD to Compile and Link an RPG Batch Program

```
$ BUILD TSTDBG
   9-APR-1996 16:18:44
MRPG /OBJECT=$DISK1:[AR]TSTDBG $DISK1:[AR]TSTDBG.RPG
   9-APR-1996 16:19:11
   9-APR-1996 16:19:19
LINK /EXECUTABLE=$DISK1:[AR]TSTDBG -
  $DISK1:[AR]TSTDBG
   9-APR-1996 16:19:26

Directory $DISK1:[AR]
```

Example 5–7 Cont'd on next page

Example 5–7 (Cont.) Using BUILD to Compile and Link an RPG Batch Program

```
TSTDBG.EXE;1          9-APR-1996 16:19:24.15
TSTDBG.OBJ;1          9-APR-1996 16:18:51.02
TSTDBG.RPG;9         25-MAR-1996 16:35:17.58

Total of 3 files.

$
```

In this example, the RPG batch program TSTDBG is compiled and linked using the BUILD procedure.

Example 5–8 Using BUILD to Compile and Link an RPG Workstation Program

```
$ BUILD DIAMON
   9-APR-1996 16:20:38
MRPG /OBJECT=$DISK1:[AR]DIAMON $DISK1:[AR]DIAMON.RPG
   9-APR-1996 16:20:52
   9-APR-1996 16:20:57
SFG /OBJECT=$DISK1:[AR]DIAMONFM $DISK1:[AR]DIAMONFM.FRM
   9-APR-1996 16:21:02
   9-APR-1996 16:21:03
LINK /EXECUTABLE=$DISK1:[AR]DIAMON -
  $DISK1:[AR]DIAMON, $DISK1:[AR]DIAMONFM
   9-APR-1996 16:21:09

Directory $DISK1:[AR]

DIAMON.EXE;1          9-APR-1996 16:21:07.73
DIAMON.OBJ;1          9-APR-1996 16:20:43.36
DIAMON.RPG;2         26-MAR-1996 08:16:36.37
DIAMONFM.FRM;1       25-JUL-1996 15:25:32.28
DIAMONFM.OBJ;1        9-APR-1996 16:21:00.65

Total of 5 files.

$
```

In this example, the workstation program DIAMON is compiled and linked by the BUILD procedure.

Example 5–9 Using BUILD With the List Parameter

```
$ BUILD POKER Y
   9-APR-1996 16:26:08
MRPG /OBJECT=$DISK1:[CARDS]POKER /LIST $DISK1:[CARDS]POKER.RPG
   9-APR-1996 16:26:41
   9-APR-1996 16:26:47
SFG /OBJECT=$DISK1:[CARDS]POKERFM /LIST $DISK1:[CARDS]POKERFM.FRM
   9-APR-1996 16:26:55
   9-APR-1996 16:26:56
LINK /EXECUTABLE=$DISK1:[CARDS]POKER -
  $DISK1:[CARDS]POKER, $DISK1:[CARDS]POKERFM
   9-APR-1996 16:27:02

Directory $255$DUA8:[CLAREMONT.TEST]

POKER.EXE;1           9-APR-1996 16:27:02.48
POKER.LIS;1           9-APR-1996 16:26:18.97
POKER.OBJ;1           9-APR-1996 16:26:18.95
POKER.RPG;11         19-JUN-1996 11:23:25.97
POKERFM.FRM;4         5-JUN-1996 13:33:22.03
POKERFM.OBJ;1         9-APR-1996 16:26:50.56
POKERFM.LIS;1         9-APR-1996 16:26:50.58

Total of 7 files.
```

In this example, the workstation program POKER is compiled and linked by the BUILD procedure. The list parameter has been selected, so compile-time listing files are produced for both the source and screen modules.

5.5 Running An RPG Program

Once an RPG program has been compiled and linked successfully, it can be run using the OpenVMS RUN command. The following command will execute the program STOMPER.

```
$ RUN STOMPER
```

There is no difference in the run command used for a batch or interactive RPG program.

In most cases, the files referenced by an RPG program will have logical assignments made to indicate the files' names and locations. Logical file assignments are made using the DCL DEFINE or ASSIGN commands. See the *OpenVMS DCL Dictionary* for more information on these commands. The following example illustrates a typical RPG program call in a DCL command procedure:

Example 5–10 RPG Program Call in a DCL Command Procedure

```
$ DEFINE /USER POWDER   ARMORY:GUN_POWDER.DAT
$ DEFINE /USER BALL     ARMORY:IRON_BALL.DAT
$ DEFINE /USER REPORT   SYS$SCRATCH:CANNON.LIS
$ RUN ARMY:CANNON
$ PRINT /DELETE SYS$SCRATCH:CANNON.LIS
```

In this example, the logical names ARMORY, ARMY and SYS$SCRATCH have been used to denote device and directory specifications. The program, CANNON.EXE, processes the data files GUN_POWDER and IRON_ BALL, using the logical names POWDER and BALL, to produce the report CANNON.LIS. This report is then submitted to the user's default print queue.

5.6 Interacting with RPG Workstation Programs

The following section describes the workstation keyboard mapping, field editing, and characteristics of Migration RPG interactive programs.

Interactive RPG terminal support is available on all VT series terminals and terminal emulators. Support is also available for terminals and devices using DECwindows or Motif via a DECterm window.

5.6.1 Workstation Key Assignments

The following keys have been defined for WORKSTN programs. These key definitions cannot be modified or changed when using a VT series terminal. However, VT series terminal emulation software generally allows keyboard remapping.

If Migration RPG keymapping is altered using terminal emulation software, be sure that the command sequences sent to the program match those used for a standard VT series terminal.

Table 5–1 Command and Function Key Definition Chart

Command	Keystroke(s)
Clear	PF1 + C
Command	PF1 followed by 1-9, 0, -, = for command keys 1 - 12 and by PF1 followed by !, @, #, $, %, ^, &, *, (,), _, + for command keys 13 - 24.
DUP-Duplicate	PF3
ENTER/REC ADV	PF4
Help	Help or PF2 or PF1 + H
Home	PF1 + T
Print	PF1 + P
Roll Up	Prev Screen or PF1 + U
Roll Down	Next Screen or PF1 + D

5.6.1.1 Function and Command Keys

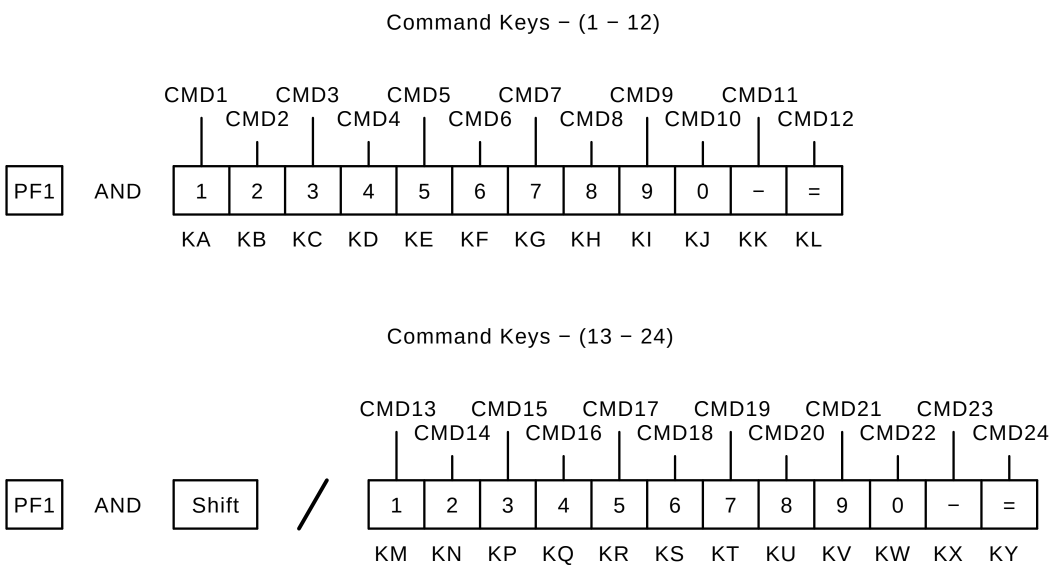
This diagram displays the command keys and their associated indicators as they would be used on a standard LK201 or LK401 keyboard. To enter a Command 7, turning on the KG indicator, the user would use the following key strokes:

PF1 + 7

To enter a Command 15, turning on the KP indicator:

PF1 + Shift / 3

Figure 5–1 Command Key Definition Diagram



5.6.1.2 Field Editing Keys

Table 5–2 Chart of Field Editing Keys

Keystroke	Function
Up Arrow	Field exit and position cursor at first field on previous line.
Down Arrow	Field exit and position cursor at first field on next line.
Left Arrow	Within a field: Move cursor one position to the left. At beginning of field: Position cursor at the beginning of previous field.
Right Arrow	Within a field: Move cursor one position to the right. At end of field: Position cursor at the beginning of the next field.
LF or F13	Clear field.
RETURN or ENTER	Field Exit key. Any data to the right of the cursor is deleted.
PF1 + RETURN or PF1 + ENTER	Field Minus Key. Can only be used on fields defined as signed numeric. Enters the numeric field as a negative number and places a minus sign at the end of the field.
TAB	Field Advance. Advances cursor to next field on the screen.
BACKSPACE or F12	Field Backup. Current field is exited and the cursor is positioned at the beginning of the previous field.
CTRL/A	Toggles the terminal between overstrike and insert mode for character entry. If the last character in a field is not blank, the terminal will not switch to insert mode. When the last character in a field is filled, the terminal will default to overstrike mode. If the <CTRL/A> is entered and the terminal is already in insert mode, it is switched back to overstrike mode. The terminal always defaults to overstrike mode when entering a new field.
CTRL/W	Screen Refresh Key. This command will cause the CRT screen to be cleared and repainted with the contents of the current screen.

VT series terminal keypads can also be used to enter Command Keys 0 - 11, as well as several of the Function keys. The keypads are defined as numeric within the context of the RPG program and can be used to enter numeric data. The following diagram displays the commands available on the keypad.

Figure 5–2 VT Series Terminal Keypad Diagram

PF1 Command	PF2 Help	PF3 DUP	PF4 Enter/ Record Advance
7 (Cmd 7)	8 (Cmd 8)	9 (Cmd 9)	– (Cmd 11)
4 (Cmd 4)	5 (Cmd 5)	6 (Cmd 6)	,
1 (Cmd 1)	2 (Cmd 2)	3 (Cmd 3)	ENTER Field +
0 (Cmd 10)		.	(Field −)

The functions enclosed in parentheses are activated by first entering a Command (<PF1>) Key, followed by the desired function key. The command keys 12 - 24 are located on the standard keyboard as the characters =, !, @, #, $, %, ^, &, *, (,), _, and +, respectively.

5.6.2 Field Editing Within An RPG Workstation Screen

Editing of characters entered in a data field on a workstation screen takes place as the characters are keyed in by the user. The system will not accept invalid characters into a data field. For example, if a user attempts to key the character 'A' in a field defined as numeric (N in column 27 of the D specification), the terminal bell will sound, the cursor will remain at the same position in the field, and the character will not be accepted by the data entry screen. The invalid character will remain displayed on the screen and the user will be able to key over it.

Invalid command and function keys are treated as field exit keys. A bell will sound and the cursor will be repositioned at the beginning of the field.

5.7 Help Support Within An RPG Workstation Program

RPG Help specifications (H) are supported by Migration RPG. The help screens referenced must be included within the S and D specifications source file. The Screen Format Generator, as it compiles the S and D specifications, will be compiling the help screens along with the data screens. Thus, the screen object module linked to the RPG workstation program will include the referenced help screens.

To access help screens from a workstation program, press either the <HELP> key, the <PF2> key on the numeric keypad, or the <PF1> key followed by the <H> key. The help screen defined for the screen location at which the cursor currently resides will be displayed. Depending upon the completion options defined in the help screen, the initial screen will be redisplayed and a prompt for input will occur, or control will be returned to the RPG workstation program.

See the *Migration RPG Screen Format Reference Manual* for more information on Help screen specifications.

5.8 Column Separator Simulation

D specifications can specify that column separators be used in a field (i.e., column 49 = Y). On a System/34 or System/36 using a 5251 display station, this would display a vertical bar on either side of the column. The column separator bars do not require additional character positions on the 5251 display. This type of column separator display is not possible on a VT series terminal. Migration RPG simulates it by initializing each column to an underscore character ("_").

5.9 Terminal Characteristics Not Reset After Program Abort

When an RPG program executes, it modifies the terminal setup characteristics. When the program exits, the terminal characteristics are restored to the values that existed before the program was executed. If the program aborts or is aborted for some reason at run time, the terminal characteristics may not be restored. Default terminal characteristics can be restored by using the DCL command:

```
$ SET TERMINAL /INQUIRE /LINE_EDIT
```

The following terminal qualifiers are set at run time by an RPG program:

- ESCAPE
- FORM
- NOWRAP
- NUMERIC_KEYPAD
- NOLINE_EDITING

5.10 Handling The F-Spec Continuation Option ID - RPGCONFIG.DAT

In an RPG program, it is possible to obtain the ID of the terminal from the system. On the System/36, the format for this is a one-position alpha character (usually a "W") followed by single number (1 - 9). RPG programs may be coded to check this terminal ID in order to allow the program to validate the user's authority to run the program. This operation is emulated within Migration RPG by means of a translation file that is processed in the Migration RPG install command file (i.e., RPGINSTAL.COM).

When the RPG install command file is invoked, a file labeled RPGCONFIG.DAT, which resides in the S3X$RPG directory, is input and a comparison is made on the last four positions of each record of this file with the first four positions of the user's OpenVMS terminal ID. If a match is found, the data from the configuration file in record positions 6 - 9 are moved into the RPG external date and indicator file, S3X$EXT. This mechanism provides a translation facility to translate the actual terminal ID in the OpenVMS format to any value desired by the user. The last two positions of the replacement ID constitute the value which will be supplied to the user by the RPG program when the terminal ID field is referenced. Thus, using this translation file, it is possible to allow coding to remain intact as it would appear on the System/36 and to integrate terminal ID's in the System/36 format while running under OpenVMS.

The data file for RPGCONFIG.DAT is a text file and can be maintained using the standard OpenVMS EDT or TPU editors. Records in the file are configured as follows:

Example 5–11 RPGCONFIG.DAT Format

```
         1         2
1234567890123456789012345

xxxx,yyyy
xxxx,yyyy
```

- **xxxx -** Last four characters of the OpenVMS terminal identifier.
- **, -** Terminal identifier separator.
- **yyyy -** IBM terminal identifier.

Example 5–12 Sample RPGCONFIG.DAT

```
         1         2
1234567890123456789012345

TXA1,TXW5
TXA7,TXW9
```

In this example, the OpenVMS terminal TXA1 has been assigned the IBM terminal ID TXW5. The OpenVMS terminal TXA7 has been assigned the terminal ID TXW9.

It is important to note that when using terminal servers, it is not possible to assign a permanent terminal ID to a specific terminal. Each time a user logs in using a terminal server, the user's terminal is assigned a different ID number. In this case, the best way to establish a permanent terminal ID for the user is to have the user's LOGIN.COM file access the S3X$EXT file after the RPGINSTAL.COM procedure has been run and have it update the S3X$EXT file with a user-assigned terminal ID. The following DCL commands can be used to access and update the S3X$EXT file:

Example 5–13 Sample DCL to Access and Update the S3X$EXT File

```
$         OPEN /READ /WRITE N_EXT S3X$EXT
$         READ N_EXT N_EXTREC
$         N_EXTREC[14,4] := "TXA0"
$         WRITE /SYMBOL /UPDATE N_EXT N_EXTREC
$         CLOSE N_EXT
```

In this example, the S3X$EXT file is updated with the terminal ID 'TXA0' to replace the terminal ID specified by the terminal server.

5.11 RPG HALT Message

Migration RPG supports the use of halt indicators in the interactive processing environment. When the execution of an RPG program run in interactive mode is halted due to the setting of a HALT indicator (H1 - H9), a halt message will be displayed indicating which halt indicator has been set on and the user will be prompted for input, as the following example illustrates.

Example 5–14 Sample Halt Message

```
$ RUN JMPSTRT
RPG HALT NUMBER 6
RTS-HLT020 - Enter C, E or K (Continue, Exit or Kill):
```

In this example, the halt message indicates that the H6 indicator has been set on and, as a result, the program has halted execution. The user may select one of three options at the halt prompt: CONTINUE, EXIT, or KILL. The CONTINUE option turns off the halt indicator and continues the program. The EXIT option terminates the program image in an orderly fashion, shutting down I/O channels, and closing files. The KILL option aborts the program immediately, without running through the normal image rundown procedure.

When RPG programs are run in batch mode, halt indicators are treated as severe errors and the program is aborted.

RPG programs that use the halt indicators H1 - H9 may not operate properly when run in a batch queue. If a program activates a halt indicator while executing in the batch queue, it will immediately abort and continue with the rest of the job stream. The user does not get an option to continue the program.

A suggested method for preventing halt indicators from getting set on in programs that run in the batch queue is to use external indicators. The following example shows how to test for the process operation mode (BATCH or INTERACTIVE) and how to set an external indicator based on the results of the test.

Example 5–15 Sample DCL to Test for INTERACTIVE or BATCH Modes and Set an External Indicator

```
   .
   .
   .
$   IF F$MODE().EQS."INTERACTIVE"
$     THEN
$        REX /IND=0XXXXXXX
$     ELSE
$        REX /IND=1XXXXXXX
$    ENDIF
   .
   .
   .
$   RUN LOD_LIB:PROG.EXE
$!
$! If in batch mode, then test here for
$! your external switch.
$!
```

NOTE: See Chapter 8, "Setting Dates and External Indicators (REX Utility)", for more information on the REX Utility.

Now the RPG source code can be modified to test for the switch U1 before all statements that may set on one of the halt indicators.

In the following example, if U1 is off and the comparison is true for the H1 indicator, then H1 will be set on. U1 will be off if the program is running in interactive mode. However, if the program is running in the batch queue, U1 will be on and halt indicators should not used by the program. Instead, set on another indicator and program around the halt via an external switch, special file, or LDA mechanism.

Example 5–16 Sample RPG Code to Work Around HALT Indicator in a Batch Queue

```
         1         2         3         4         5
1234567890123456789012345678901234567890123456789012345678

     C* Original code.
     C*         FLDA      COMP FLDB                H1H1
     C*
     C* Code employing a workaround.
     C*
     C   U1     FLDA      COMP FLDB                U2U2
     C  NU1     FLDA      COMP FLDB                H1H1
     C*
     C* U1 indicates the job is being run in a batch queue.
     C* U2 serves as a substitute for the H1 indicator when
     C* U1 is on.
```

6 Console Files (RPGCON Utility)

6.1 Overview

MSI’s Migration RPG supports the CONSOLE device. Using a CONSOLE device in an RPG program enables the operator to input data to an executing RPG program from a display terminal. Display screen formats are automatically generated from the RPG input specifications to prompt the operator for data. The data entered by the operator is returned to the RPG program as a contiguous record, as if the data had been read from a record-oriented device. CONSOLE files can only be used to input data; they cannot be used to display or update data.

6.2 Restrictions for Using a Console File

The following restrictions apply to programs with CONSOLE files:

- Only one CONSOLE file is allowed per program.
- No other devices relating to terminal I/O may be used.
- The maximum alphanumeric field length is 66 characters.
- The maximum numeric field length is 15 characters.
- The maximum number of fields in a record type is 80.
- The maximum number of record types for a CONSOLE file is 20.
- The CONSOLE device screen cannot be used to update or display existing data.

6.3 Building an RPG Program Using the CONSOLE Device

Compiling and linking an RPG program using the CONSOLE device is a four-step process.

1. The RPG program is compiled.
2. The CONSOLE screen is generated using the RPGCON Utility.
3. The CONSOLE screen is compiled using the Screen Format Generator.
4. The compiled program and screen are linked together to produce an executable image.

6.4 Running the RPGCON Utility

The RPGCON Utility uses an RPG source file for input. The RPG source file must specify the CONSOLE device in the File Specifications. The Input specifications associated to the CONSOLE device are parsed by the RPGCON Utility and used to generate S & D specifications. The generated S & D specifications can be customized by the programmer or simply compiled with the Screen Format Generator (SFG) and linked to the RPG program. It is recommended that the RPG program be compiled before running the RPGCON Utility against it to ensure that the Input specifications being parsed are syntactically correct.

The RPGCON Utility is invoked with the command RPGCON.

RPGCON

FORMAT

RPGCON *filename*

PARAMETERS

filename

The name of the RPG source file containing the CONSOLE device. The file name can be a fully qualified OpenVMS file designation. A file type of .RPG is assumed unless explicitly stated otherwise in the command line. The file name is a required parameter and will be prompted for if it is not supplied.

Example 6–1 RPGCON Command

```
$ RPGCON ALCATRAZ
```

DESCRIPTION

The output file generated by the RPGCON Utility will have the same name as the input file, with an FM appended to the file name and an .FRM file type in place of an .RPG file type. In the previous example, the RPG source file ALCATRAZ.RPG would be processed by the RPGCON Utility, generating the screen file ALCATRAZFM.FRM.

6.5 Display Screen Format

The workstation display screen generated by the RPGCON Utility consists of a top line containing control information for the operator and data fields arranged on lines 2 through 23.

- The first field displayed on line 1 is the record identification code for the record being prompted. This field is a 1 or 2 character field, as defined in the Input specifications by the record identification characters (columns 21 - 34). Different record formats may be selected by moving the cursor into this field, entering a valid record identification code, and pressing the <ENTER/RECORD ADVANCE> key (<PF4>). The new format will be displayed and the operator prompted for input without any data being returned to the RPG program.

  The first record format defined in the CONSOLE device input specifications is assumed to be the default format and will be displayed when the program initializes the screen. The default format is also displayed whenever an invalid record identification code is entered.

- The second field displayed on the first line is the record identification indicator for the format being displayed. This field is an output only field.

- The third field displayed on the first line is the record identification indicator for record types other than the one displayed. These may be selected by the operator by entering the appropriate value in the record identification code field. This field is an output only field.

- Data fields are generated in either a one, two, three, or four column format, depending on the number of fields and their sizes. When a data format is displayed, the cursor is always positioned in the first data field of the format. Cursor control keys may be used to move from one field to another. No data is returned to the RPG program until the <ENTER RECORD/ADVANCE> key (<PF4>) is pressed. End-of-file on the CONSOLE device is indicated by pressing the <CMD12> key (<PF1> + <=>). This returns end-of-file to the RPG program and sets on the KL indicator.

- The prompt for each data field consists of 14 characters laid out as follows:

Table 6–1 CONSOLE Data Field Prompt Format

Position	Description
1 - 1	Blank.
2 - 7	Field name taken from program Input specification.
8 - 8	Blank.
9 - 9	Field type: A - Alphanumeric, N - Numeric.
10 - 13	Field length: For alphanumeric fields, positions 10 and 11 are blank, and positions 12 and 13 contain the field length. For numeric fields, positions 10 and 11 contain the field length, position 12 is a decimal point, and position 13 contains the number of decimal places.
14 - 14	Blank.

Data entry and field editing is handled in the same manner as on a standard workstation screen. The Arrow, Backspace, Return, Enter and Tab keys can be used to move between the fields. The <ENTER RECORD/ADVANCE> key (<PF4>) is used to return the input record to the program.

7 Local Data Area

7.1 Overview

The Local Data Area is a single record 512-byte sequential file created each time the user logs onto the system. It can be used to pass information between RPG programs, DCL procedures, and programs written in other languages. A separate Local Data Area exists on disk for each user that logs onto the system. The entire Local Data Area is available to the user to enter and extract data.

7.2 Local Data Area Creation and Initialization

When a user or batch job logs onto the system, the Local Data Area (LDA) file is created on disk and initialized to blanks. This is done by the program RPGINILDA, which is executed in the RPGINSTAL.COM command file. Should the user wish to reinitialize the LDA, they may do so by running the RPGINILDA program using the following command:

```
$ RUN S3X$RPG:RPGINILDA
```

7.3 Accessing the LDA from an RPG Program

To access the Local Data Area from an RPG program, the programmer must first define a data structure with a maximum of 512 characters in the Input specifications. A "U" must be coded in column 18 of the DS Input specification to indicate that this data structure is for use by the Local Data Area. When the program is executed, the contents of the LDA are loaded into the internal program data structure and the data is made available to the program. At program termination, the contents of the internal program data structure are written back to the LDA.

For users converting multiple requester programs from the System/36, an external subroutine labeled SUBR21 is provided for compatibility purposes. This subroutine is for compatibility only and has no effect on the loading or writing of the Local Data Area.

For more information on coding data structures and the LDA data structure within an RPG program, see the *Migration RPG Language Reference Manual*.

7.4 Accessing the LDA from DCL

The Local Data Area may be accessed or altered within command procedures through the use of DCL statements. Access is accomplished using the DCL commands OPEN, CLOSE, READ, and WRITE. The Local Data Area exists on disk as a sequential file which contains a single 512-byte, fixed-length record. The file is referenced via the logical name **S3X$LDA**.

The following example shows the Local Data Area being opened, read, updated, and closed using DCL commands.

Example 7–1 DCL Used to Access the Local Data Area

```
$ OPEN /READ /WRITE N_LDA S3X$LDA      !Open LDA file.
$ READ N_LDA N_LDAREC                  !Read LDA record.
$!
$ DATA = F$EXTRACT((12-1),3,N_LDAREC)  !Extract data from LDA record.
$ N_LDAREC[(25-1),4] := COUNT          !Place data in LDA record.
$!
$ WRITE /UPDATE /SYMBOL N_LDA N_LDAREC !Update LDA file.
$ CLOSE N_LDA                          !Close LDA file.
```

7.5 Accessing the LDA from a Non-RPG Program

The LDA can also be accessed from a non-RPG program. In this case, the user simply opens, reads, updates, and closes the single record 512-byte LDA file referenced via the logical name S3X$LDA. Caution should be taken not to inadvertently add records to the LDA, rather than updating the existing record, as Migration RPG programs only read and write to the first record in the LDA file.

8 Setting Dates and External Indicators (REX Utility)

8.1 Overview

Migration RPG programs can access eight external indicators, U1 - U8. These indicators can be set within or outside of an RPG program. RPG programs can also reference the system date via the reserved words UDATE, UMONTH, UDAY, UYEAR, $UDATE, $UMNTH, $UDAY, $UYEAR. The REX Utility allows the user to set and clear the external indicators and change the system date referenced by Migration RPG programs.

For more information on indicators and date keywords, see the *Migration RPG Language Reference Manual*.

8.2 Running the REX Utility

The REX Utility can be invoked from the DCL command prompt or used within a DCL procedure. It allows the user to set and clear external indicators and change the date referenced by Migration RPG programs.

The REX Utility is invoked using the command REX.

REX

FORMAT

REX *qualifiers*

PARAMETERS

qualifiers

The REX Utility has two qualifiers associated to it. At least one of the qualifiers must be specified with the command. If no qualifier is specified, the utility will prompt for one.

QUALIFIERS

/DATE=mmddyyyy

The /DATE qualifier allows the user to modify the system date referenced by the RPG keywords UDATE, UMONTH, UDAY, and UYEAR. The new user-defined date must be entered in the format *mmddyyyy*. By default, Migration RPG programs use the CPU-defined system date. If the date is set using the REX Utility, Migration RPG programs run form that account will use the user-defined date until it is cleared or the user logs out. Using the REX Utility to set a date does not in anyway affect the actual system date on your OpenVMS system. The date set via the REX Utility is process specific.

Example 8–1 Using the REX Utility to Set a User-Defined Date

```
$ REX /DATE=11061996
```

In this example, the REX Utility has been used to set the date referenced by Migration RPG programs to 11/06/1996.

Example 8–2 Using the REX Utility to Clear a User-Defined Date

```
$ REX /DATE=
```

In this example, the REX Utility has been used to clear the user-defined date for Migration RPG programs. The programs will now default to the system date when referencing date information via the date keywords.

/IND=n[nnnnnnn]

The /IND qualifier allows the user to set or clear one or more of the external indicators U1 through U8. Each position following the /IND qualifier corresponds to an external indicator. Thus, the first position corresponds to U1, the second to U2, and so forth. Valid settings for each indicator are:

Table 8–1 External Indicator Values

Setting	Description
0	Set corresponding indicator off.
1	Set corresponding indicator on.
X	Do not modify corresponding indicator.

Example 8–3 Using the REX Utility to Set and Clear External Indicators

```
$ REX /IND=1XX001
```

In this example, the REX Utility has been used to set external indicators U1 and U6 on; U4 and U5 off; and leave the settings of U2, U3, U7, and U8 alone. Note that it is not necessary to specify indicators U7 and U8 in this example as they are not being modified and are not needed as place holders, as U2 and U3 were.

QUALIFIER NOTE

The /DATE and /IND qualifiers can be specified independently or together. The following example illustrates the use of both qualifiers on one command line:

Example 8–4 Using the REX Utility to Set a Date and External Indicators

```
$ REX /DATE=11061996 /IND=11XX0010
```

In this example, the REX Utility has been used to set the date referenced by Migration RPG programs to 11/06/1996 and to set external indicators U1, U2 and U7 on; U5, U6 and U8 off; and ignore U2 and U3.

8.3 Referencing External Indicators and the User-Defined Date From DCL or Programs Written in Other Languages

External indicator and date settings are stored in a 170-byte, fixed-length, single-record, sequential data file referenced by the logical name **S3X$EXT**. This file is created along with the Local Data Area file by the RPGINSTAL.COM procedure each time a user logs in. When the file is created, the external indicators and date fields are initialized to ASCII zeros (0).

The S3X$EXT file can be referenced by DCL procedures or programs written in other languages by treating it as a single-record sequential file. Date and external indicator settings can then be examined and modified. The S3X$EXT date file is laid out as follows:

Table 8–2 External Indicator and Date File Layout

Position(s)	Description
1	U1
2	U2
3	U3
4	U4
5	U5
6	U6
7	U7
8	U8
9 - 14	6-digit date (format: mmddyy); non-Year 2000 compliant
15 - 18	Terminal ID. Either the last four characters of the OpenVMS terminal ID or the terminal ID extracted from the RPGCONFIG.DAT file.
19 - 50	Reserved for future use.
51 - 58	8-digit date (format: mmddyyyy); Year 2000 compliant
59 - 170	Reserved for future use.

Should the S3X$EXT file be deleted, the REX Utility can be used to recreate it. When invoked, the REX Utility will create an S3X$EXT file if it cannot find one already in existence.

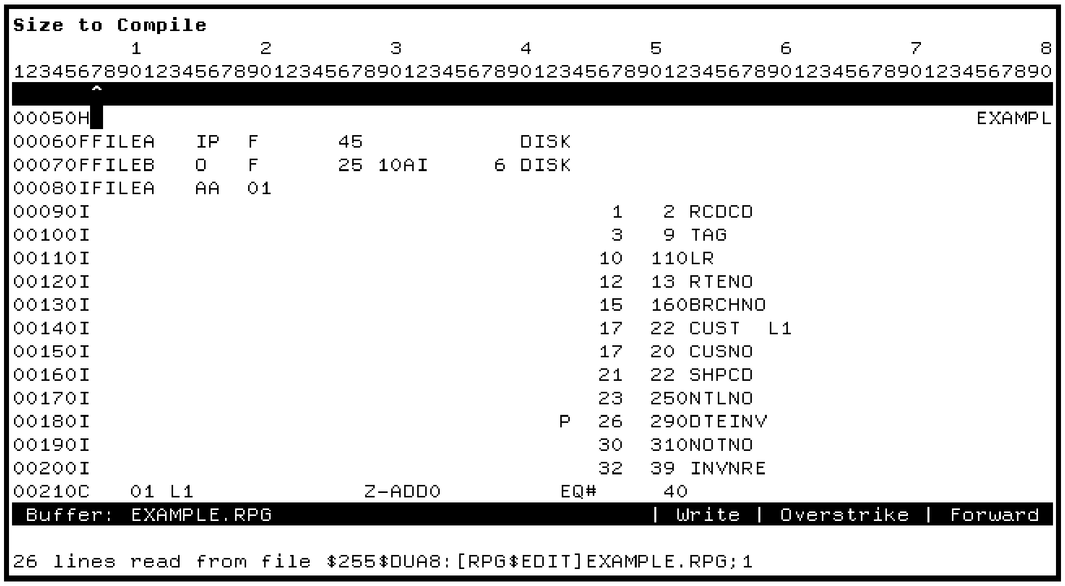
9 Creating and Editing RPG Programs (RED Editor)

RESTRICTION: The RED Editor is compatible with version 5.5 and higher of OpenVMS. This editor is *NOT* compatible with earlier versions of OpenVMS.

9.1 Overview

MSI's Migration RPG RED Editor is a TPU-based, full screen text editor designed specifically for the entry and manipulation of RPG source code. Through its features, the user may move, copy, scan, delete, replace, enter, or update RPG source specifications. The editor also has the ability to display, scan, and copy text from another program into the program being edited. The entry and updating of RPG source code is simplified by the editor's ability to track the cursor position within the specification type, regardless of which line in the window is being edited.

Figure 9–1 The RED Editor



```
Size to Compile
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
      ^
00050H                                                                    EXAMPL
00060FFILEA   IP  F      45            DISK
00070FFILEB   O   F      25 10AI     6 DISK
00080IFILEA   AA  01
00090I                                        1   2 RCDCD
00100I                                        3   9 TAG
00110I                                       10  110LR
00120I                                       12  13 RTENO
00130I                                       15  160BRCHNO
00140I                                       17  22 CUST  L1
00150I                                       17  20 CUSNO
00160I                                       21  22 SHPCD
00170I                                       23  250NTLNO
00180I                                    P  26  290DTEINV
00190I                                       30  310NOTNO
00200I                                       32  39 INVNRE
00210C   01 L1             Z-ADD0         EQ#     40
 Buffer:  EXAMPLE.RPG                            | Write | Overstrike | Forward

26 lines read from file $255$DUA8:[RPG$EDIT]EXAMPLE.RPG;1
```

The RED Editor screen is designed in the following manner:

- Line 1 is used to display the description of the field for the specification where the cursor currently resides. As the cursor is moved throughout the editing window, the description of the field is continuously updated. If column 6 contains a blank or an invalid specification type, line 1 will be blank.
- Lines 2 and 3 comprise a column number reference ruler.
- Line 4 indicates the current cursor position. The "^" corresponds with the column where the cursor is positioned for any line in the editing window. The field for the specification in which the cursor is positioned is highlighted.
- Lines 5 through 21 contain the editing buffer. If the editing window is split and more than one editing buffer is shown, lines 1 through 4 track the cursor for the buffer being edited.
- Line 22 is the status line. The status line contains the name of the buffer, whether the buffer is write capable or read only, whether it is in insert or overstrike mode, and the direction of cursor movement.
- Lines 23 and 24 are used to display messages and enter TPU commands.

9.2 Invoking the RED Editor

To invoke the editor, enter the command RED followed by the file specification of the program to be edited at the DCL prompt. If the file specification is omitted, the editor will contain an empty buffer. If the file is found, the source code is read into a buffer for editing. If the file is not found, a message is displayed stating that a new file will be created.

RED

FORMAT **RED** *filename*

PARAMETERS

filename
Name of the source file to be edited. The file name can be a fully qualified OpenVMS file description. Note that the .RPG file type is not assumed.

QUALIFIERS

The qualifiers are standard TPU qualifiers. Please refer to the OpenVMS *EVE Reference Manual* for more information about TPU qualifiers.

9.3 Special Keys

The RED editor is an extension of the OpenVMS EVE editor. All commands and key definitions included with EVE also apply to the RED editor. Please refer to the OpenVMS *EVE Reference Manual* for more information on the use of EVE. To simplify the use of the editor, a number of additional keys have been defined to supplement those provided with EVE. These keys are discussed in this section.

9.3.1 Cursor Placement Keys

Cursor Placement Keys are used to move the cursor to the next occurrence of a particular specification type within an existing source file. If no more occurrences of the specification are found, the cursor is positioned to the first occurrence of the requested specification. A Cursor Placement Key consists of two keys that are pressed in conjunction with one another: the Command Key followed by a key representing the specification to which the user wishes to position the cursor. The Command Key refers to the <PF1> key and is represented by <CMD>.

Table 9–1 RED Cursor Placement Keys Within RPG Program Source

Command Key	Cursor Placement
CMD / H	Position to the Control Specification
CMD / F	Position to next File Description Specification
CMD / E	Position to next Extension Specification
CMD / L	Position to next Line Counter Specification
CMD / I	Position to next Input Specification
CMD / C	Position to next Calculation Specification
CMD / O	Position to next Output Specification
CMD / U	Position to the Option Specification

Table 9–2 RED Cursor Placement Keys Within RPG Screen Source

Command Key	Cursor Placement
CMD / S	Position to next Display Control Specification
CMD / H	Position to next Help Specification
CMD / D	Position to next Field Definition Specification

9.3.2 TAB and BACKTAB

Tab stops are set for the fields in each specification type. When the cursor is moved to a line that contains a different specification, the tab stops correspond to the fields in the new specification type. If the specification type of the current line is changed, the tab stops correspond to the new specification type. Tab stops are set at five spaces for comment lines and lines that do not contain a valid specification type in column 6 (i.e., compile time tables and arrays). The <TAB> and <BACKTAB> keys are defined as movement only keys, allowing them to be used without moving the text. <BACKTAB> is defined as the <BACKSPACE> key for VT100 terminals and <F12> for other VT style terminals.

9.3.3 RETURN

When the <RETURN> key is pressed, a new line is started from the current cursor location. If text follows the cursor, it will be brought to the new line, just as in a standard EVE editing session. The specification type of the current line is retained as the specification type of the new line. The cursor is placed in column 6 and the tab stops for the specification type remain in effect. If a different specification type is to be entered, changing the entry in column 6 will cause the tab stops to be modified to correspond to the new specification type.

9.3.4 ENABLE/DISABLE Cursor Tracking

The cursor tracking capabilities of the RED Editor can be disabled by pressing the <F17> or <CMD/T> keys. Cursor tracking can be toggled back on by pressing the <F17> or <CMD/T> keys again. When cursor tracking is disabled, the line that contains the field description (Line 1) and the line that displays the cursor position (Line 4) are blank.

9.4 Customizing the RED Editor

The RED Editor has been layered on the EVE editor and will support EVE initialization files. Initialization files allow the editor to be customized for each user’s preferences. The following example shows an initialization file for a user that prefers the EDT-like keypad and has defined a key for the QUIT command:

```
SET KEYPAD EDT
DEFINE KEY=gold/q QUIT
```

A symbol can be defined in the user’s individual LOGIN.COM file to call the RED Editor using an initialization file:

Example 9–1 Sample RED Symbol Definition

```
$   RED :== EDIT/TPU/SECTION=S3X$RPG:RPG$EDIT.TPU$SECTION-
                             /INIT=USER_INIT.EVE
```

Because the RED Editor is a specialized editor, there are two areas that should not be customized in the user’s initialization file. They are:

- SET CURSOR BOUND

  This command will not allow the cursor to be moved into an unused portion of the buffer.

  For example, with a bound cursor, if the cursor is at the last character of a line and the TAB key is pressed, the cursor would move to the start of the next line. By contrast, with a free cursor (RED Editor default), the cursor can be positioned anywhere in the buffer whether characters are already there or not.

- SET TAB

  The RED Editor defines the TAB key as movement only and defines all tab stops. Any modifications to the TAB definition using the SET TAB command will be ignored.

For more information on EVE initialization files, see the *Guide to OpenVMS Text Processing*.

The TPU source file for the RED editor is located in the S3X$RPG directory under the name RPG$EDIT.TPU. If you would like to modify this file, it is recommended that you copy it to a different location and modify the copy. This will avoid corrupting the original source file.

10 Auto Report Utility - AUTOC

10.1 Overview

MSI's Migration RPG Auto Report Utility can be used to generate RPG programs from source and copybook program modules. It is modeled on the System/36 Auto Report Utility. This chapter describes how to run the Migration RPG Auto Report Utility. Creating Auto Report programs is described in detail in the *Migration RPG Language Reference Manual*.

AUTOC

FORMAT

AUTOC *filename*

PARAMETERS

filename

Name of the program to be generated and compiled. This can be a fully qualified OpenVMS file specification. The compiler assumes a file type of .RPG unless explicitly stated in the input line.

DESCRIPTION

The Migration RPG Auto Report Utility builds and compiles an RPG program. Qualifiers are used to control the output and functions of the RPG compiler. The qualifiers are position-independent, meaning they can be specified in any order following the AUTOC command or the file name. The following qualifiers can be used to condition Auto Report program construction and compilation.

Please note that some of these qualifiers can be overridden by the U specification within the RPG source file. U specification settings will always take precedence over command line qualifiers.

QUALIFIERS

/LIST[=filename]
/NOLIST (Default)

Controls whether a list file is produced when the Auto Report program is compiled. By default, Auto Report does not produce a listing. The list file can optionally be given a different name by specifying the file name following the /LIST qualifier. If no list file name is specified, the list file is given the program name and a .LIS file type.

If /NOLIST is specified, the compiler will not produce a program listing.

Example 10–1 AUTOC Compile Command With /LIST Qualifier

```
$ AUTOC /LIST BIG1
```

This command generates and compiles the program BIG1.RPG to produce an object file labeled BIG1.OBJ, a source file labeled BIG1.AUT, and a listing file labeled BIG1.LIS.

/OBJECT[=filename] (Default)
/NOOBJECT

By default, Auto Report will produce an object module with the program name and an .OBJ file type if no errors are found during program construction and compilation. An object file with a different name can be generated by specifying a file name following the /OBJECT qualifier. If /NOOBJECT is specified, Auto Report will build an RPG source file and parse it for errors. This qualifier is useful for quick syntax checking of the RPG source code.

Example 10–2 AUTOC Compile Command With /OBJECT Qualifier

```
$ AUTOC /OBJECT=BLASTER.OBJ CAT
```

This command generates and compiles the program CAT.RPG, producing the object file BLASTER.OBJ.

/PAGE=nnn

Specifies the number of lines per page to print on the output listing generated by the Auto Report. By default, the compiler assumes 60 lines per page. The /PAGE qualifier can be specified in a range from 10 to 200. Invalid entries are ignored and the default value of 60 is used. If the /LIST qualifier is not specified or the /NOLIST qualifier is in use, the /PAGE qualifier is ignored.

/SAVE[=filename] (Default)
/NOSAVE

Controls whether the RPG source code generated by the Auto Report Utility is saved. By default, the program is saved using the original RPG source program name and an .AUT file type. A different save name can optionally be specified following the /SAVE qualifier. If the /NOSAVE qualifier is specified, the source program is deleted as soon as it has been compiled.

/SEQUENCE_NUMBERS (Default)
/NOSEQUENCE_NUMBERS

By default, the Auto Report Utility will place sequence numbers, starting with 0010 and incrementing by tens, in the first four positions of each line in the generated source file. This feature can be turned off by specifying the /NOSEQUENCE_NUMBERS qualifier. If the /NOSEQUENCE_NUMBERS qualifier is used, blanks are output in the first four columns of each line. Compile time table and array data at the end of the program are not affected by sequence numbering.

NOTE

Qualifier names can be abbreviated to four characters. If a qualifier is entered more than once on a command line, the last entry takes precedence over all previous entries.

10.1.1 Auto Report File Designations and Defaults

All file names used by Auto Report are standard OpenVMS file designations. A file designation can include a device name, directory specification, file name, type, and version number. If portions of the file designations are omitted, certain appropriate values (default values) are assumed for the missing portions. The assumed device is the user's system device (SYS$DISK:). The assumed directory is the current default directory. The assumed file types are .RPG for the input source file, .AUT for the generated RPG source file, .LIS for the listing, and .OBJ for the object file. On output files, the assumed version number is the latest version number, plus one.

11 File Names and Conventions

11.1 Overview

This chapter describes the file naming conventions and accessing methods used by Migration RPG. More detailed information on file access methods is available in the *Migration RPG Language Reference Manual*.

11.2 File Names

The file names specified in columns 7 - 14 of the RPG File (F) specifications are processed as follows at runtime:

1. The logical tables available to the process running the RPG program are scanned for a match with the file name specified in columns 7 - 14. If a match is found, the logical name is used as the file designation.
2. If no logical name is associated with the file name in columns 7 - 14 of the File (F) specification, the name is treated as a physical file name and the user's current default directory is used to search for or create the file. A .DAT file type is assumed for DISK files and a .LIS file type is assumed for PRINT or PRINTER files.
3. If the file cannot be found or created using one of these two criteria, the program will abort with an RMS file not found error message.

11.2.1 Physical File Names

If the user wishes to use physical file naming when running an RPG program, the data files must be located in the directory from which the user is running the program and the file names must exactly match the file names specified in columns 7 - 14 of the program's File (F) specifications. A default file type of .DAT is assumed for all data files referenced by physical file name. Print files created using physical file names are given a .LIS file type.

11.2.2 Logical File Names

Logical file assignments can be made using the DCL ASSIGN or DEFINE command. Using logical file names permits the user to use file names greater than 8 characters in length and to include device and directory designations in the file assignment. Any valid OpenVMS file name can be used as a logical file assignment to an RPG program.

For example, if the program SAWDUST.RPG has an RPG file description specification with the name TREETOP in columns 7 - 14, at execution time the program would look for a logical assignment associating TREETOP with a file specification. The following example illustrates the creation of this logical assignment prior to executing the program:

Example 11–1 Logical File Assignment for an RPG Program

```
$ DEFINE /USER TREETOP DRA0:[LUMBER]CHAINSAW.DAT
$ RUN SAWDUST
```

In this example, the data file CHAINSAW.DAT is assigned to the logical name TREETOP for the duration of the next image. When the program SAWDUST is executed, the file CHAINSAW.DAT will be accessed on device DRA0 in the directory [LUMBER].

When using logical file assignments, file types are never assumed. Therefore, when a logical file assignment is created, it is necessary to use the complete file designation. Only the version number can be omitted.

Logical assignments use paged dynamic pool space. It is recommended that the /USER qualifier be used when assigning logical file names. The /USER qualifier allows the logical file assignment to be deassigned automatically after an image has executed.

If the user chooses not to use the /USER qualifier, it is important to realize that a logical assignment made in a procedure remains in effect until it is deassigned, reassigned, or the user logs off. To conserve pool space, it is recommended that assignments are deassigned as rapidly as possible after execution of the program or job stream needing them.

See the *OpenVMS DCL Dictionary* for more information on the ASSIGN and DEFINE commands and the /USER qualifier.

The following example shows how logical file assignments can be made for an RPG program. The program contains three File specifications which reference the file names BANK, SAFE, and REPORT. The ASSIGN statements in this example create logical file assignments for these files.

Example 11–2 Sample DCL for Logical File Assignments

```
$ ASSIGN /USER USER1:FORTKNOX.DAT        BANK
$ ASSIGN /USER USER2:BIGVAULT.DAT        SAFE
$ ASSIGN /USER SYS$SCRATCH:CONTENTS.LIS  REPORT
$ RUN LOD_LIB:CUSTUPD
$ PRINT/DELETE SYS$SCRATCH:CONTENTS
```

11.3 File Sharing

File sharing is defined as two or more file streams accessing the same data file at the same time. By default, RPG programs under OpenVMS allow file sharing according to the rules of RMS. File sharing can be qualified by specifying a share code in column 73 of the RPG File (F) specification.

11.3.1 File Sharing using Column 73

Migration RPG uses column 73 of the RPG File specification to define the type of access that will be allowed to a file that is opened by the program. The access codes are defined in the following table.

Table 11–1 File Sharing Access Codes

Access Code	Definition
N	No shared access. The file will be open exclusively by this file stream and will not be accessible to any other open/read/write request from any other source.
R	Read only access. The file stream controlling the file will allow other file streams read access to the file.
S or Blank	Read/Write/Delete access (Default). The current RMS default file access rules will be applied to the file stream.

A file with an 'N' in column 73 of the File specification will not allow any other file streams access to the data file. A file with an 'R' in column 73 will allow other file streams Read access to the data file, but will not allow records to be read for update, write, or deletion from the file. A file with an 'S' or blank in column 73 allows other file streams full access to the data file, as long as this access does not conflict with the default RMS restrictions on the file.

When using the file sharing code in column 73, it is important to realize that the share code specified for one file stream affects all other file streams that attempt to access the file, even if the file streams come from within the same program. Therefore, if file share codes are used incorrectly, it is possible for a program to deny itself access to a file by having two conflicting file share qualifiers set up on two file streams accessing the same file.

Consider the following example:

Example 11–3 File Sharing

```
$ DEFINE /USER HOG     USER1:CLUMSY.DAT
$ DEFINE /USER BEGGAR  USER1:CLUMSY.DAT
$ RUN LOD_LIB:FEEDTIME
```

In this example, the program FEEDTIME assigns two file streams to the data file CLUMSY.DAT. Under default conditions, this operation would present no problems. However, if in the program the file HOG has a share code of 'N', the program will be unable to connect a second stream to the data file and will abort when it tries to connect the stream for BEGGAR.

11.4 Record Locking

When a program attempts to access a record that is currently locked by another program, the program attempting to access the record will display a record lock message to the terminal and go into a wait state, waiting for the record it is attempting to access to be freed. If the record is not freed within a reasonable amount of time (about 5 minutes), the program will abort with a record access error.

11.5 Using Alternate Keys

The RMS file system under OpenVMS allows the creation and access of indexed files with multiple keys. An indexed file must have a primary key and may have up to 254 alternate keys. The Migration RPG Compiler supports up to 99 (01 - 99) alternate keys.

11.5.1 Specifying an Alternate Key

To access a file through an alternate key from an RPG program, make the following entries in the RPG File specification.

- Columns 29 - 30 are coded with the length of the alternate key to be used.
- Columns 35 - 38 are coded with the starting position of the alternate key to be used.
- Columns 68 - 69 are coded with the reference number of the alternate key (01 - 99) that was specified for the key when the file was defined. These columns must be blank or zero if the primary key is being used for access.
- The remainder of the RPG file description specification is coded in the same manner an indexed file accessed by the primary key would be.

Example 11–4 RPG File Specification Defining an Alternate Index

```
0        1         2         3         4         5         6         7
1234567890123456789012345678901234567890123456789012345678901234567890

     FHOTDOG  IC  F 136 136R03AI    22 DISK                        01
```

In this example, the file HOTDOG is referenced via its first alternate key.

Files with multiple keys are defined using the OpenVMS FDL Editor and the CREATE or CONVERT commands. See the *OpenVMS DCL Dictionary*, *Guide to OpenVMS File Applications*, and *OpenVMS User's Manual* for more information on creating indexed file descriptions.

11.6 File Setup for READP Opcode

No special modifications on fixed-length files with sequential and relative organizations are necessary to perform read prior (READP) operations. However, indexed files that are to be read sequentially backwards must be defined correctly or the operation will fail.

11.6.1 Indexed File Definitions for READP

Only indexed files that are to be processed using the READP opcode need to be defined according to the following rules. Indexed files that are not processed using the READP opcode can be set up normally. If, at a later date, a program is developed that accesses a normal indexed file using the READP opcode, it will be necessary to modify the file's FDL description to conform with the following rules.

When an RPG program opens an indexed file for any type of sequential processing, it uses a primary or alternate key definition to specify the sort order in which the file is to be read. For normal processing, this key can be ascending or descending and can be defined anywhere within the data record.

When processing an indexed file using the READP opcode, the key specified to open the file must be defined twice: once as ascending and once as descending. These key definitions can be accomplished using the OpenVMS FDL Editor. It is not necessary to doubly define all the keys in an indexed file that is processed using READP; only those key fields that are used to define the sequential file processing order needed for READP operations need to be defined twice.

11.6.1.1 Rules for Key Definition when using READP

When defining a key field that will be processed by READP, the same key must be defined twice. The first definition is the normal indexed key field definition that would be used for any other type of indexed or sequential file operation (CHAIN, READ, READE, SETLL). The second key definition is a duplicate of the first or primary definition in every way except for the sort order. If the primary key is defined as an ascending key, the secondary or READP key must be defined as descending. If the primary key is defined as descending, then the READP key must be defined as ascending. The READP key definition must **immediately** follow the primary key definition or the READP operation will not function correctly. If the primary key is Key 1, the READP key **must** be defined as Key 2. If the primary key (for a READP setup) is the sixth alternate key definition for the file (Key 6), the READP key **must** be defined as Key 7.

The rules for creating a READP key may be stated as follows:

- The primary and READP key fields must define the same physical location in the data record.
- The READP key must have the opposite sort order of the primary key.
- The READP key definition must immediately follow the primary key definition in the FDL file description block.

The following examples are FDL description files for indexed files which contain key field descriptions set up for use with the READP opcode.

Example 11–5 FDL Example 1 for READP Opcode

```
IDENT   "29-JUN-1996 17:17:50   OpenVMS FDL Editor"

SYSTEM
        SOURCE                  "OpenVMS"

FILE
        ORGANIZATION            indexed

RECORD
        CARRIAGE_CONTROL        carriage_return
        FORMAT                  fixed
        SIZE                    80

AREA 0
        ALLOCATION              3
        BEST_TRY_CONTIGUOUS     yes
        BUCKET_SIZE             3
        EXTENSION               3

AREA 1
        ALLOCATION              0
        BEST_TRY_CONTIGUOUS     yes
        BUCKET_SIZE             3
        EXTENSION               3

AREA 2
        ALLOCATION              0
        BEST_TRY_CONTIGUOUS     yes
        BUCKET_SIZE             3
        EXTENSION               12

KEY 0
        CHANGES                         no
        DATA_AREA               0
        DATA_FILL               100
        DATA_KEY_COMPRESSION    yes
        DATA_RECORD_COMPRESSION         yes
        DUPLICATES              no
        INDEX_AREA              1
        INDEX_COMPRESSION       yes
        INDEX_FILL              100
        LEVEL1_INDEX_AREA       1
        NAME                    "PRIMARY KEY"
        PROLOG                  3
        SEG0_LENGTH             10
        SEG0_POSITION           0
        TYPE                    string
```

Example 11–5 Cont'd on next page

Example 11–5 (Cont.) FDL Example 1 for READP Opcode

```
KEY 1
        CHANGES                         no
        DATA_AREA               2
        DATA_FILL               100
        DATA_KEY_COMPRESSION    yes
        DUPLICATES              no
        INDEX_AREA              2
        INDEX_COMPRESSION       yes
        INDEX_FILL              100
        LEVEL1_INDEX_AREA       2
        NAME                    "READP KEY"
        SEG0_LENGTH             10
        SEG0_POSITION           0
        TYPE                    dstring

KEY 2
        CHANGES                         no
        DATA_AREA               2
        DATA_FILL               100
        DATA_KEY_COMPRESSION    yes
        DUPLICATES              yes
        INDEX_AREA              2
        INDEX_COMPRESSION       yes
        INDEX_FILL              100
        LEVEL1_INDEX_AREA       2
        NAME                    "ALTERNATE KEY"
        SEG0_LENGTH             5
        SEG0_POSITION           44
        TYPE                    string
```

In this example, the primary file key (Key 0) also serves as the primary key field for a READP definition (Key 1). Note that Key 0 and Key 1 are identical in all respects except for the sort order, which is indicated by the ‘TYPE’ field. Key 0 is an ascending key field; Key 1 is a descending key field. When this file is open with Key 0 as the key of reference, READP operations will execute correctly.

The Key 2 field defined in this example is the first alternate key in the file. READP operations could not be performed if the file were open with this key as the key of reference. To allow READP operations using this key, it would be necessary to create a Key 3, identical to Key 2 in every respect except for the sort order (TYPE).

Example 11–6 FDL Example 2 for READP Opcode

```
IDENT   "29-JUN-1996 17:12:56   OpenVMS FDL Editor"

SYSTEM
        SOURCE                  "OpenVMS"

FILE
        ORGANIZATION            indexed

RECORD
        CARRIAGE_CONTROL        carriage_return
        FORMAT                  fixed
        SIZE                    256

AREA 0
        ALLOCATION              3
        BEST_TRY_CONTIGUOUS     yes
        BUCKET_SIZE             3
        EXTENSION               3

AREA 1
        ALLOCATION              0
        BEST_TRY_CONTIGUOUS     yes
        BUCKET_SIZE             3
        EXTENSION               3

AREA 2
        ALLOCATION              0
        BEST_TRY_CONTIGUOUS     yes
        BUCKET_SIZE             3
        EXTENSION               12

KEY 0
        CHANGES                         no
        DATA_AREA               0
        DATA_FILL               100
        DATA_KEY_COMPRESSION    yes
        DATA_RECORD_COMPRESSION         yes
        DUPLICATES              no
        INDEX_AREA              1
        INDEX_COMPRESSION       yes
        INDEX_FILL              100
        LEVEL1_INDEX_AREA       1
        PROLOG                  3
        SEG0_LENGTH             12
        SEG0_POSITION           0
        TYPE                    string

KEY 1
        CHANGES                         no
        DATA_AREA               2
        DATA_FILL               100
        DATA_KEY_COMPRESSION    yes
        DUPLICATES              yes
        INDEX_AREA              2
        INDEX_COMPRESSION       yes
        INDEX_FILL              100
        LEVEL1_INDEX_AREA       2
        NAME                    "PRIMARY KEY"
        SEG0_LENGTH             5
        SEG0_POSITION           12
        TYPE                    ddecimal
```

Example 11–6 Cont’d on next page

Example 11–6 (Cont.) FDL Example 2 for READP Opcode

```
KEY 2
        CHANGES                                no
        DATA_AREA                 2
        DATA_FILL                 100
        DATA_KEY_COMPRESSION      yes
        DUPLICATES                yes
        INDEX_AREA                2
        INDEX_COMPRESSION         yes
        INDEX_FILL                100
        LEVEL1_INDEX_AREA         2
        NAME                      "READP KEY"
        SEG0_LENGTH               5
        SEG0_POSITION             12
        TYPE                      decimal
```

In this example, Key 1, the first alternate key in the program, has been set up for READP processing. Key 1 is the primary key and Key 2 is the READP key. Key 1 is defined as a descending decimal key and the Key 2 definition is identical except that it is an ascending decimal key. In this case, Key 0, the primary file key, cannot be used for READP processing. If the user wanted to use Key 0 for READP processing, it would be necessary to redefine Key 1 as a descending match for Key 0, redefine Key 2 as the current Key 1, and create Key 3 to replace the current Key 2.

No modifications to the RPG source code are necessary to use the READP opcode on any file type.

12 Maintaining Message Files (RPGMSG Utility)

12.1 Overview

MSI’s Migration RPG Message File Maintenance Utility, RPGMSG, is used to create and maintain files containing text that may be accessed by an RPG program through message numbers. There are two types of message files: system and user. Each type of message file has two levels. Level 1 of each type of message file allows messages of up to 75 characters. Level 2 of each type of message file allows messages of up to 225 characters. Each message has a 4-digit message number associated with it, which is used to access the message.

12.2 Invoking RPGMSG

The Message File Maintenance Utility is invoked with the command RPGMSG.

RPGMSG

DESCRIPTION

The RPGMSG Utility will prompt for a file code as follows:

Enter MIC file code: S1, S2, U1, U2, or eXit:

CODES

Table 12–1 Message Member File Codes

File Code	Definition
S1	Corresponds to System File, Level 1.
S2	Corresponds to System File, Level 2.
U1	Corresponds to User File, Level 1.
U2	Corresponds to User File, Level 2.
X	Causes the program to exit.

Enter the desired function code (in upper case) and press the <RETURN> key.

12.3 Adding or Changing Messages

After the file code has been entered, the utility will prompt for a message number value. Enter an X to terminate the utility or enter the number that corresponds to the message that is to be added or changed.

If the message number has been previously defined, it will be displayed and the utility will wait for the new data to be entered. If the <RETURN> key is pressed without entering data, the message is left as is and another message number prompt appears. If data is entered prior to pressing the <RETURN> key, the text of the message is replaced with the data that was entered.

If a message was not defined for the message number that was entered, a message will be displayed stating this and instructing that data be entered. If the <RETURN> key is pressed without entering data, nothing is added and another message number prompt appears. If data is keyed prior to pressing the <RETURN> key, a new message number is defined containing the data that was keyed.

12.4 Message File Assignments

Four logical assignments are used to access the four message files:

- RPGSYSLV1 - System Level 1
- RPGSYSLV2 - System Level 2
- RPGUSRLV1 - User Level 1
- RPGUSRLV2 - User Level 2

By default, the RPGINSTAL.COM command procedure assigns the message file logicals to the following data files in the S3X$RPG directory:

- RPGSYSLV1 - S3X$RPG:RPGSYSLV1.MSG
- RPGSYSLV2 - S3X$RPG:RPGSYSLV2.MSG
- RPGUSRLV1 - S3X$RPG:RPGUSRLV1.MSG
- RPGUSRLV2 - S3X$RPG:RPGUSRLV2.MSG

Thus, by default, all programs access the same message files. If an application requires unique message files, the default assignments can be overridden using the DCL ASSIGN or DEFINE commands.

An empty System Level 1 message file is provided with the Migration RPG Compiler Kit. It is left to the user to create or convert additional message files. The CVTMIC Utility in MSI's OpenVMS S/3X Conversion Tools can be used to convert MIC message members from the IBM System/3, System/34, and System/36.

12.5 Accessing Message Files From DCL

Message members may be accessed from DCL by using the OPEN, CLOSE, and READ commands.

A Level 1 file is an indexed file which contains 84-byte, fixed-length records, with the message number residing in the last 4 bytes. The message number is the file key.

A Level 2 file is an indexed file which contains 300-byte, fixed-length records, with the message number residing in the last 4 bytes. The message number is the file key.

13 Dumping Files (RDP Utility)

13.1 Overview

MSI’s RDP File Dump Utility is used to create a hexadecimal dump of an ASCII or EBCDIC file. The dump can be made to the terminal or a listing file. The RDP Utility allows the user to dump all or part of a file.

13.2 Running the RDP Utility

The RDP Utility is invoked with the command RDP.

RDP

FORMAT

RDP *input-filespec [output-filespec]*

PARAMETERS

input-filespec

The name of the input ASCII or EBCDIC file to be dumped. This is a required parameter and will be prompted for if it is not supplied. Any type of file can be submitted to the RDP Utility, but it is most useful when dumping data files containing packed or binary data fields. The file name can be a fully qualified OpenVMS file description.

output-filespec

The name of the output file to which the hex dump is to be written. This can be a fully qualified OpenVMS file description. If an output file is not specified, the RDP Utility will generate a disk file in the current default directory with the same name as the input file and a .DMP extension. This file can be displayed on a terminal or printed.

If the user would like to see the dump displayed directly to a terminal, rather than sent to a disk file, specify the logical name TT: as the output file. This will direct the RDP Utility's output directly to the screen.

DESCRIPTION

The RDP Utility is used to create a hexadecimal dump of an OpenVMS file. The following qualifiers are used to control the output and functions of the RDP Utility. These qualifiers are position-independent, meaning they can be specified in any order following the RDP command or the file names.

If no qualifiers are specified, the default is to dump the first 10 records of the input file.

QUALIFIERS

/EBCDIC

The /EBCDIC qualifier is used to specify that the input file contains EBCDIC data. By default, input is assumed to be ASCII data.

Example 13–1 RDP Command With /EBCDIC Qualifier

```
$ RDP /EBCDIC CLEANUP.DAT TT:
```

In this example, the first 10 records of the EBCDIC file CLEANUP.DAT will be dumped to the user's terminal.

/LAST

The /LAST qualifier is used to dump approximately the last two pages of data from a file. By default, the RDP Utility will dump the first 10 records in a file.

Example 13–2 RDP Command With /LAST Qualifier

```
$ RDP /LAST DUSTRAG.DAT
```

In this example, the last two pages of data will be dumped from the file DUSTRAG.DAT to the listing file DUSTRAG.DMP.

/RANGE=x[:y]

The /RANGE qualifier allows the user to specify a range of records that is to be dumped from the input file. By default, the RDP Utility dumps the first 10 records in a file. The /RANGE qualifier allows the user to dump fewer or more records.

The first number (x) entered following the /RANGE qualifier is the first record to be dumped. If this is the only number specified, only that record will be dumped. The second number (y) specified, following a colon (:), is the last record in the range to be dumped. If this number exceeds the total number of records in the file, the file will be dumped from the starting value to the end of the file.

Example 13–3 RDP Command With /RANGE Qualifier

```
$ RDP /RANGE=21:42 DUSTBAL.DAT SWEEPER.LIS
```

In this example, the file DUSTBAL.DAT will have records 21 - 42 dumped to the listing file, SWEEPER.LIS.

13.3 Dump Listing Format

A record dump consists of three lines of information. The first line is the character image of the record. In the case of packed, binary, and EBCDIC data, many of these characters may not be printable. Immediately under each character position of the record is the hexadecimal representation of the character, with the high four bits being displayed on the second line and the low four bits being displayed on the third line.

Example 13–4 RDP Command Output - Sample of an ASCII File Dump

```
** RECORD NUMBER 000001 **

CUSTOMER  100  123     999  A
45554445223332233322222333224
3534FD52001000012300000999001

** RECORD NUMBER 000002 **

CUSTOMER  200  456     888  B
45554445223332233322222333224
3534FD52002000045600000888002
```

In reading this dump, the character C in the word CUSTOMER in the first record of the dump is represented by a Hex 43. The value of the high four bits (4) is displayed on the second line and the value of the low four bits (3) is displayed on the third line. The Hex value of the character U is 55; the Hex value of the character S is 53; and so forth.

The RDP Utility will display a maximum of 100 characters on a line. Records exceeding 100 characters in length are continued on the following lines.

13.4 RDP Error Messages

- **ERROR DMP-004 - CANNOT OPEN INPUT FILE**

  The input file cannot be found or a physical channel to the file cannot be established. Verify the command format, the location of the input file, the amount of free disk space, and the file and directory protection set in the current work area.

- **ERROR DMP-005 - CANNOT OPEN OUTPUT FILE**

  The output file cannot be opened or a physical channel to the file cannot be established. Verify the command format, the file description of the output file, the amount of free disk space, and the file and directory protection set in the current work area.

- **ERROR DMP-006 - INPUT FILE MUST BE SPECIFIED**

  The name of the input ASCII or EBCDIC file to be dumped must be specified.

- **ERROR DMP-010 - START OF RANGE CANNOT BE GREATER THAN RANGE**

  If the /RANGE=x:y qualifier is included, the start range number (x) must be a smaller value than the ending range number (y).

- **ERROR DMP-011 - /LAST AND /RANGE CANNOT BE SPECIFIED TOGETHER**

  The /LAST and /RANGE qualifiers are mutually exclusive. Re-enter the command using either the /LAST qualifier or the /RANGE qualifier, but do not specify both.

- **ERROR DMP-012 - INVALID COMMAND LINE QUALIFIER**

  A qualifier which is not valid for the RDP Utility was entered on the command line. Review the format of the RDP command and re-enter the command.

- **ERROR DMP-013 - RANGE MUST HAVE STARTING NUMBER**

  The /RANGE=x:y qualifier must specify a starting number (x). This error message is displayed when neither a starting or ending range number has been specified.

  If a starting number (x) is specified without an ending number, only that record having the value of the starting record number will be dumped.

  If only the ending range number is specified, the utility will default to dumping the first ten (10) records of the file.

14 MENU Utility (MENU)

14.1 Overview

The MENU Utility supports menus that use an S & D specification file to generate the menu screen.

14.2 MENU Utility Features

The MENU Utility supports the following features:

- Procedure calls.
- Menu calls.
- The OFF command (LOGOUT).
- Entering a zero (0) to exit the menu (EXIT).
- Entering a valid DCL command on the menu command line and executing it.
- Placement of valid DCL commands in the menu command file.
- Help specifications and help processing.
- Indicator-based output and display characteristic control via the indicators 91 - 98, which are controlled via the external indicators U1 - U8.

14.3 Running the Utility

The MENU Utility is invoked with the command MENU.

MENU

FORMAT

MENU *menu-screen-format-file*

PARAMETERS

menu-screen-format-file

File name of the S & D specification file containing the menu screen format. The menu screen format file name may be fully qualified with a directory and file type. If no file type is provided, the default file type of .FRM is assumed.

NOTE

In addition to the menu screen format file, there must also be a menu command file. The menu command file contains the menu selection numbers and the procedure and menu calls that correspond to the menu generated by the menu screen format file. The menu command file must reside in the same directory as the menu screen format file. It must have the same name as the menu screen format file, followed by an __.MN1. Thus, if the format for the menu MENU01 exists in the menu screen format file MENU01.FRM, the menu command file must reside in the same directory and have the name MENU01__.MN1.

Example 14–1 MENU Command

```
$ MENU MENU01
```

This command would run the menu MENU01. It would cause the MENU Utility to search the user's current default directory for the menu screen format file MENU01.FRM. The MENU Utility would use that file to generate the screen display for MENU01. If the user were to make a numeric selection off of MENU01, the MENU Utility would then search for the menu command file in the same directory under the file name MENU01__.MN1. The MENU Utility would scan the file for a selection number corresponding to the one entered by the user. If found, the MENU Utility would execute that command.

14.4 Operation

The MENU Utility uses the menu screen format file to generate a screen display and prompts the user for input. The MENU Utility will accept numbers corresponding to the selections in the menu procedure file. It will also accept and execute DCL commands, provided this capability has not been disabled within the menu screen format file. Alpha character entry can be disabled by defining the menu selection input line to accept numeric entry only.

The MENU Utility will treat any numeric entry of 4 digits or less as valid input and search the menu command file for a matching selection number. If a matching selection is not found, the MENU Utility will prompt for a new selection. Any numeric entry with more than 4 digits will be treated as an error and ignored. Any alpha or alphanumeric entry will be treated as a DCL command.

The MENU Utility will treat any library names encountered in the command text source member as logical names on the OpenVMS system. It is up to the user to create corresponding logical names on the OpenVMS system. The user can also place logical names directly into the menu command file, allowing a single menu to access and execute procedures from anywhere on the system.

The MENU Utility recognizes the OFF command and treats it as a DCL LOGOUT command.

The MENU Utility attempts to execute any procedure selected from the menu in the user's current default directory unless a different directory has been specified in the menu command file. Users can avoid problems with menu commands not finding procedures by having the menu and procedures reside in the same directory, always running the menu from SYS_PROC, or placing the appropriate logical names in the menu command file.

Valid DCL commands can be placed in the menu command file and executed as selections from the menu. System/36 OCL commands other than procedure calls, menu calls, and the OFF command are not valid on an OpenVMS system.

The following table shows the layout of a menu command file.

Table 14–1 Menu Command File Layout

Columns	Description
01 - 04	Selection Number - A 4-digit, right-adjusted number corresponding to a menu option number contained within the menu screen format file.
05	Blank
06 - 261	Command to be executed:
	Procedure to be called
	Menu to be called
	DCL command to be executed - The DCL command must be preceded with a dollar sign and a space.
	OCL OFF command - Executed as the DCL LOGOUT command.

NOTE: Command text source members converted from a System/36 will contain an initial line which identifies the menu name and type of source member. This line is unnecessary to the MENU Utility on a OpenVMS system and will be treated as a comment line.

14.5 Using DCL Commands with the MENU Utility

The MENU Utility accepts DCL commands from within the menu command file (the file with the __.MN1 appended to the name). DCL commands can be placed within the menu command file by prefixing them with a dollar sign and a single space ($). For example, following is a menu command file with a call to the DCL procedure PAYROLL, which is located in the FINANCE area:

Example 14–2 Sample Menu Command File: MASTER__.MN1

```
MASTER##,2
0001 TRANS
0002 CHGCORR
0003 EDITLST
0004 CUSINQ
0005 $ @FINANCE:PAYROLL
0006 UPDMASTR
0007 MENU EOM
0008 MENU EOQ
0009 MENU EOY
0099 OFF
```

The MENU Utility supports the processing of the following System/36 commands when they are located within a menu command file:

- Procedure calls.
- MENU command.
- The OFF command (LOGOUT).

Any command prefixed with a dollar sign and a space ($) will be treated as a DCL command and processed as is. Any unidentified command will be treated as a procedure call.

Following is the menu screen format file which would be associated with the previous sample menu command file.

Example 14–3 Sample Menu Screen Format File: MASTER.FRM

```
0        1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
      * MASTER.FRM
      * Sample of Migration RPG Menu screen format file.
      *
     SMASTER              YY        Y                          56C
     DWSID       2 178Y
     DINPUT    12022 3   YO      Y
     DCOMMAND    7 1 2Y                                CCOMMAND
     DMSTRMNU   40 120Y               Y                C              MENU: MASX
     DTER
     DPRMPT1    3821 2Y                                CReady for option numberX
     D or command
     DFM0000    28 3 2Y                                CSelect one of the folloX
     Dwing:
     DFM0001     2 516Y                                C1.
     DFM0002    18 520Y                                CEnter transactions
     DFM0003     2 616Y                                C2.
     DFM0004    25 620Y                                CEnter Changes/CorrectioX
     Dns
     DFM0005     2 716Y                                C3.
     DFM0006    15 720Y                                CPrint Edit List
     DFM0007     2 816Y                                C4.
     DFM0008    23 820Y                                CCustomer Inquiry Screen
     DFM0009     2 916Y                                C5.
     DFM0010    11 920Y                                CRun Payroll
     DFM0009     21016Y                                C6.
     DFM0010    181020Y                                CUpdate Master File
     DFM0011     21116Y                                C7.
     DFM0012    171120Y                                CEnd-of-Month Menu
     DFM0013     21216Y                                C8.
     DFM0014    191220Y                                CEnd-of-Quarter Menu
     DFM0015     21316Y                                C9.
     DFM0016    161320Y                                CEnd-of-Year Menu
     DFM0027     31815Y                                C99.
     DFM0028     81820Y                                CSign Off
```

14.6 MENU Error Messages

- **010: Invalid command in '.MN1' file**

  The menu command file contains an OCL command which is not supported by the Migration RPG MENU Utility. Delete the command from the file or replace it with a procedure call containing a comparable DCL command.

- **100: Screen Format file not found**

  The menu screen format file (having a default .FRM file type) cannot be found. Ensure that the file exists, that it is accessible, and that the file name has been specified correctly.

- **110: Menu Command not found**

  The menu command file (having a default .MN1 file type) cannot be found. Ensure that the file exists, that it is accessible, and that the file name has been specified correctly. Remember that the menu command file uses the naming convention *menufile__.MN1*

- **129: Invalid field length (Columns 15 - 18)**

  The length specified in columns 15 - 18 of the D specification is not valid.

- **130: Invalid line number (Columns 19 - 20)**

  The line number specified in columns 19 - 20 of the D specification is not valid.

- **132: Invalid column number (Columns 21 - 22)**

  The column number specified in columns 21 - 22 of the D specification is not valid.

- **400: Invalid continuation**

  The continuation character in column 80 of the D specification is either invalid, it should be 'X', or has been specified for a constant defined with a length of 23 or less.

15 PROMPT Utility (PROMPT)

15.1 Overview

The PROMPT Utility allows a user to maintain and update the symbols P1 through P64 in their process symbol table. A file containing Screen (S), Help (H), and Field Definition (D) specifications is used to define an interactive screen on which the process symbols P1 through P64 can be displayed and modified.

15.2 Running the Utility

This utility is invoked with the command PROMPT.

PROMPT

FORMAT

PROMPT *screen-format-file -*
/FORMAT=display-format-name

PARAMETERS

screen-format-file

The file name of the format file containing the Screen (S), Help (H), and Field Definition (D) specifications used to define the display screen. This can be a fully qualified OpenVMS file name. A file type of .FRM is assumed if one is not provided. This parameter is required and will be prompted for if it is not provided.

/FORMAT=display-format-name

The FORMAT parameter specifies the name of the display format that is to be displayed from within the screen format file. This is a required parameter. The display format name can be up to eight (8) characters in length. The display format name is compared to the format name specified in columns 7 - 14 of each Screen (S) specification in the screen format file. If no match is found, the utility will abort with an error message.

QUALIFIERS

/LENGTH=n or /LENGTH="n,n,n..."

By default, the PROMPT Utility assigns each symbol a length of eight (8). The length value specified in columns 15 - 18 of the Field Definition (D) specification for input fields is ignored by the PROMPT Utility. Symbol values read in by the PROMPT Utility that exceed this length are truncated; symbol values read in that are less than eight characters in length are padded with blanks. The /LENGTH qualifier allows the user to explicitly specify the length of each symbol that the PROMPT Utility processes. The length value range is 0 to 256. If multiple symbol lengths are to be specified with the LENGTH qualifier, they must be enclosed in double quotes (") and separated by commas. A single value does not need to be enclosed in quotes. Length values specified following the /LENGTH qualifier will affect the symbols prompted for in the order the symbols are processed by the PROMPT Utility. Thus, the first symbol processed by the PROMPT Utility will be assigned the first length value specified; the second symbol processed will be assigned the second length value; and so forth. If a null length value is specified (",,"), the default length of eight (8) is assigned to that symbol. If a value of zero (0) is assigned as a length, the associated symbol is neither displayed nor prompted for, and its value is unaffected by the PROMPT Utility.

Example 15–1 PROMPT Command With /LENGTH Qualifier

```
$ PROMPT /FORMAT=FLOWERS GARDEN:SEEDS /START=4 /LENGTH="3,3,0,,10"
```

In this example, the PROMPT Utility will process the screen format file SEEDS.FRM pointed to by the logical name GARDEN. The display format labeled FLOWERS will be displayed to the screen. The START qualifier specifies that the first symbol processed by the PROMPT Utility will be

P4. The symbols P4 - P8 will be affected by the LENGTH qualifier string as follows:

```
Symbol  Length
------  ------
  P4      3
  P5      3
  P6      0 (Not processed or prompted for)
  P7      8 (Default)
  P8     10
```

Any additional symbols prompted for (P9 - P64) would be assigned the default length of eight.

/START=n
/START=1 (Default)

The START qualifier is used to specify the number of the first symbol to be displayed and/or prompted for in the display format. The START qualifier is a numeric value in the range of 1 to 64. If START is not specified, a value of one (1) is used as a default. Symbols less than the START value are ignored by the PROMPT Utility.

Example 15–2 PROMPT Command With /START Qualifier

```
$ PROMPT LITTLE /FORMAT=PIGGIES /START=7
```

In this example, the PROMPT Utility will start symbol processing with the symbol P7. Symbols P1 - P6 will be ignored by the utility and their values will be unaffected.

/UPSI=xxx
/UPSI=NO (Default)

The UPSI qualifier is used to indicate whether the PROMPT Utility checks the external indicator (U1 - U8) settings when displaying a PROMPT screen. The PROMPT Utility equates external indicators U1 through U8 to the internal indicators 91 through 98 if the UPSI qualifier is set to YES. Valid values for the UPSI qualifier are YES and NO.

15.3 Using the PROMPT Utility

The PROMPT Utility uses Screen (S), Help (H), and Field Definition (D) specifications to describe the interactive screens it displays when prompting for or displaying symbol data. The use of S, H, and D specifications to define an interactive screen is detailed in the *Migration RPG Screen Format Reference Manual*.

The PROMPT Utility is used to prompt for and/or display the data contained in the process symbols labeled P1 through P64. Symbols P1 through P8 are created when a process is created, such as when a user logs in or a batch job is started, and always exist at the process level. Symbols P9 through P64 do not exist by default. The PROMPT Utility will create them as they are needed if they have not already been defined by the user.

The PROMPT Utility processes the symbols P1 through P64 in the order they are specified in the Field Definition (D) specifications in the screen format file. When processing the PROMPT display format in the screen format file, the PROMPT Utility reads the D specifications sequentially and associates each D specification listed as an input field (Y in column 26) with the next symbol in sequence, starting with the symbol value specified by the /START qualifier (default /START value is 1). If there are more input fields than symbols, the extra input fields are ignored by the PROMPT Utility.

The order of the input and output fields in the screen format file and the order in which the fields are displayed and prompted for on the screen do not have to be the same. D specifications allow the user to specify the field's position on the screen by row and column.

The PROMPT Utility only manipulates the symbols for which it prompts. If a PROMPT screen prompts for symbols P1 - P10, then P11 - P64 remain unaffected. If the user does not create symbols P9 - P64, the PROMPT Utility will create them if it is required to access them. The PROMPT Utility will not create a non-existent symbol unless it is asked to access that symbol.

By default, all symbols prompted for and/or displayed have a length of eight (8) characters. The length field in the D specifications (columns 15 - 18) for input fields is ignored by the PROMPT Utility. The user can use the /LENGTH qualifier to override the default length of eight and specify any length up to 256 characters. Length values greater than 256 will generate an error and abort the utility. If the user specifies a length of zero (0), the associated symbol is ignored by the PROMPT Utility. It is neither displayed or updated. If the symbol does not currently exist, it will not be created.

When reading in symbol data from the symbols P1 - P64, symbols with a length greater than the value specified by the PROMPT Utility will be truncated on the right. Symbols with a length less than that specified by the PROMPT Utility will be padded to the right with blanks. Use the LENGTH qualifier to explicitly specify symbol lengths.

15.4 Using Indicators Within PROMPT Display Screens

The PROMPT Utility allows the use of indicators 01 through 64 and 91 through 98 in the PROMPT display format. Indicators 65 - 90 and 99 will always be off in a PROMPT display. Indicators 91 - 98 can be associated to the external indicators U1 - U8 by specifying YES to the UPSI qualifier on the PROMPT command line. If UPSI is specified as YES, the current settings of U1 through U8 are transferred to indicators 91 through 98.

Indicators 01 through 64 are controlled by the values stored in symbols P1 through P64. If a symbol contains only zeros, blanks, or is not defined, the associated indicator remains off (Example: P5 = "00" - indicator 05 remains off). If a symbol contains a non-blank character other than zero, the associated indicator is turned on (Example: P12 = "ALIVE" - indicator 12 is turned on).

All indicators can be used to control display characteristics and constant output. See the *Migration RPG Screen Format Reference Manual* for more information on the use of indicators in screen format files.

15.5 Return Code Processing (CD)

In addition to creating and updating the symbols P1 - P64, the PROMPT Utility also returns a status code in the symbol "CD" to indicate which key was pressed to terminate the display screen and update the symbols. Status values are defined as follows:

Table 15–1 PROMPT Utility - Return Code Status Values

Status Value	Definition
0000	Enter/Record Advance $\boxed{\text{PF4}}$
2001 through 2024	Command Keys 1 through 24 $\boxed{\text{PF1}}$ + $\boxed{\text{1}}$ ...
2090	Roll Up $\boxed{\text{Next Screen}}$ or $\boxed{\text{PF1}}$ + $\boxed{\text{U}}$
2091	Roll Down $\boxed{\text{Prev Screen}}$ or $\boxed{\text{PF1}}$ + $\boxed{\text{D}}$
2092	Help $\boxed{\text{Help}}$ or $\boxed{\text{PF1}}$ + $\boxed{\text{H}}$
2093	Home $\boxed{\text{PF1}}$ + $\boxed{\text{T}}$
9999	Any other Function key

The use of command and function keys can be controlled using the Screen (S) specification in the screen format file. The return status value is placed in the symbol CD, which can be interrogated by the DCL procedure and used to make process decisions.

15.6 PROMPT Warning Messages

- **010: Unable to access external indicator file (S3X$EXT)**

  The external indicator file, S3X$EXT, cannot be found or a physical channel to the file cannot be open. Verify that the external indicator file exists and is accessible.

15.7 PROMPT Error Messages

- **010: Screen format name not specified; /FORMAT entry required.**

  The name of the screen format that is to be displayed from within the screen format file is a required parameter. Include the /FORMAT qualifier with a display format name in the command line.

- **050: Display format name too long. Max display name length = 8**

  The display format name can be up to eight (8) characters in length. Modify the display format name to be eight (8) characters or less.

- **060: Non-numeric entry encountered in /LENGTH qualifier string**

  An alpha or alphanumeric value has been specified as a symbol length in the /LENGTH qualifier. The symbol length value range is from 0 to 256. If no value is specified, a default length of eight (8) will be assigned.

- **062: Invalid length parameter, value must be between 0 and 256**

  The value of a symbol length must be in the range of 0 to 256. If no value is specified, a default length of eight (8) will be assigned.

- **064: Too many parameters in length qualifier, only 64 allowed**

  Only sixty-four (64) lengths may be specified in the /LENGTH qualifier. Each length corresponds to a symbol, P1 - P64.

- **070: invalid start parameter, value must be between 1 and 64**

  To specify the first symbol to be displayed and/or prompted for, enter a value between 1 and 64 in the /START qualifier. If /START is not specified, a value of one (1) is used as a default.

- **100: Could not open screen format file - file name *filename***

  The screen format file (having a default file type of .FRM) cannot be found or a physical channel to the file cannot be open. Verify that the screen format file exists and is accessible.

- **112: Format display member *display-format-name* not found in screen file**

  The display format name specified in the /FORMAT qualifier cannot be found in the screen format file. Verify the display format name.

- **129: Invalid field length (Columns 15 - 18)**

  The length specified in columns 15 - 18 of the D specification is not valid.

- **130: Invalid line number (Columns 19 - 20)**

  The line number specified in columns 19 - 20 of the D specification is not valid.

- **132: Invalid column number (Columns 21 - 22)**

  The column number specified in columns 21 - 22 of the D specification is not valid.

- **300: Could not set symbol to entered value**

  The LIB$SET_SYMBOL System Service failed to create a local symbol. Verify the field definition and field length.

- **400: Invalid continuation**

  The continuation character in column 80 of the D specification is either invalid, it should be 'X', or has been specified for a constant defined with a length of 23 or less.

A Migration RPG Compiler Informational Messages

A.1 Overview

The following section lists the informational messages that can be generated by the Migration RPG compiler. Informational messages are intended to being a feature or function to the attention of the programmer. Informational messages do not have any impact on program compilation.

Typically, in the program listing the informational message will immediately follow the line which generated it.

A.2 Informational Messages

- **2-001 - Debug mode enabled**

  A 1 has been specified in column 15 of the Header specification, indicating that DEBUG commands in the program will be processed at runtime.

B Migration RPG Compiler Warning Messages

B.1 Overview

The following list represents the warning messages that can be generated by the Migration RPG compiler. Warning messages indicate a possible problem that was not severe enough to warrant aborting the compile. Warning messages can also be found on the program listing if one was selected.

Typically, in the program listing the warning message will immediately follow the line which generated it. However, this is not always the case. For example, fields may be defined anywhere in the Calculation specifications. Therefore, warnings pertaining to invalid field formats are not given until the end of the Calculation specifications.

B.2 Warning Messages

- **1-001 - There are no File or Input specifications in this program**

  The RPG program does not contain any File or Input specifications.

- **1-002 - Extra Header (H) spec ignored**

  Additional Header specifications in a program are ignored. Only the first Header specification is processed.

- **1-010 - Invalid PAGE size entered, page size defaulted to 60**

  The page size entered following the /PAGE qualifier is not valid. It must be a value between 10 and 200. The value entered has been ignored and the default value of 60 will be used.

- **1-011 - Invalid /SPACE directive. Line ignored**

  The compiler listing directive /SPACE was encountered with an invalid number of lines to space following the directive. Valid entries following the /SPACE directive are blank or 1 - 9. The directive is ignored by the compiler.

- **1-066 - *filename* filename is defined, but never used**

  The file is defined, but not referenced.

- **2-051 - Symbolic device name ignored (Columns 47 - 52)**

  Symbolic device names entered in columns 47 - 52 have no meaning under Migration RPG and are ignored by the compiler.

- **3-100 - Length specified on unnamed data structure. Length ignored**

  An entry was made in columns 48 - 51 for an unnamed data structure definition. An unnamed data structure cannot be given a length. The entry in columns 48 - 51 is ignored by the compiler.

- **3-101 - Length specified on LDA data structure. Length ignored**

  An entry was made in columns 48 - 51 of the Local Data Area (LDA) data structure definition. The Local Data Area data structure always defaults to a length of 512. The entry in columns 48 - 51 is ignored by the compiler.

- **3-266 - Field name already used in this record**

  This field has been defined more than once within this record type.

- **3-267 - Field name already used with different definition**

  This field has been defined previously with a different definition. The first definition will take precedence.

- **4-900 - Resulting indicators ignored**

  The resulting indicators should be blank for this Calculation specification. The resulting indicators will be ignored by the compiler.

- **4-901 - Factor 2 not present, Factor 1 ignored**

  If factor 1 is specified, then factor 2 is required. If factor 2 is not present, factor 1 will be ignored.

- **4-902 - Factor 1 not present, Factor 2 ignored**

  If factor 2 is specified, then factor 1 is required. If factor 1 is not present, factor 2 will be ignored.

- **4-903 - CAS stmt follows unconditional CAS stmt. Stmt ignored**

  A Case statement has been specified following an unconditioned Case statement. Any Case statement following an unconditioned Case statement is unreachable and will be ignored by the compiler.

- **4-904 - Entry in columns 49 - 51 ignored**

  The result field length should be blank for this Calculation specification. The contents of the result field length will be ignored by the compiler.

- **4-905 - Entry in column 52 ignored**

  The decimal positions field should be blank for this Calculation specification. The contents of the decimal positions field will be ignored by the compiler.

- **4-906 - Field already defined, DEFN statement ignored**

  This field has been defined previously in the program. The DEFN statement redefining the field will be ignored by the compiler.

- **4-908 - Length & Decimal flds must be blank (Col 49 - 52) - Entry ignored**

  The result field length and decimal positions fields should be blank for this Calculation specification. The result field length and decimal positions fields will be ignored by the compiler.

- **4-909 - Half adjust field must be blank (Col 53) - Entry ignored**

  The half adjust field should be blank for this Calculation specification. The half adjust field length will be ignored by the compiler.

- **4-910 - Invalid indicator in columns 54 - 55. Indicator ignored**

  The resulting indicator in columns 54 - 55 should be blank for this Calculation specification. The resulting indicator will be ignored by the compiler.

- **4-911 - Invalid entry in col 49-51, result fld defined as a table or array**

  The result field name has been previously defined as a table or array, thus entries in colums 49-51 are invalid and will be ignored by the compiler.

- **4-912 - Result field does not match previous definition**

  The result field has been defined previously with a different definition. The new definition will be ignored by the compiler.

- **4-914 - Invalid control level specified, replaced with SR control level**

  The control level specified in columns 7 - 8 of the Calculation specification is not valid in the subroutine section. The invalid control level will be replaced with an SR by the compiler.

- **4-916 - Invalid indicator in columns 58 - 59. Indicator ignored**

  The indicator in columns 58 - 59 is not valid and will be ignored by the compiler.

- **4-918 - PLIST not followed by PARM statement(s). Statement ignored**

  A PLIST statement has been processed that is not followed by any PARM statements. The PLIST statement will be ignored by the compiler. However, if another PLIST is defined with the same name, the compiler will generate an error message.

- **4-920 - Invalid PARM stmt. Must follow a CALL or PLIST stmt. Statement ignored**

  A PARM statement has been encountered that does not immediately follow a CALL or PLIST statement. The PARM statement will be ignored by the compiler.

- **4A-930: Line *nnnn*: Factor 3 fld length should be 1 for bit opcode**

  A BITOFF, BITON, or TESTB statement has been encountered where the field referenced in factor 3 has a length greater than one. Bit operations are designed to work on one character fields only.

- **5-414 - EXCPT name not defined in Calc specs**

  The EXCPT name used in the Output specification has not been defined in the Calculation specifications. The EXCPT name will be ignored by the compiler.

- **5-420 - Invalid use of *PLACE. No end position specified.**

  The *PLACE special word was used in an Output specification, but no end position was specified. The Output specification will be ignored.

- **5-425 - Fetch overflow is invalid with overflow indicator. No FETCH is assumed**

  An Output specification contains a fetch overflow with an overflow indicator. This is an invalid use of fetch overflow and no FETCH will be assumed.

- **6-415 - EXCPT name *XXXXXX* not defined in Output specs**

  An EXCPT statement that references the named Output specification *XXXXXX* has been processed in the Calculation specifications. However, there is no corresponding Output specification named *XXXXXX*. Remove the EXCPT statement or add an Output specification named *XXXXXX*.

- **6-600 - No compile time array or table defined for these entries**

  Compile time data was encountered at the end of the program which does not have a corresponding Extension specification defining it. The data is ignored by the compiler.

- **6-697 - Sequenced Table/Array contains equal elements: *array-name***

  The table/array contains elements which are not unique.

- **6-698 - Too many entries supplied for array/table *array-name***

  The compile time array or table has more entries defined at the end of the program than the Extension specification allows. The extra entries are ignored by the compiler.

- **6-699 - Too few entries supplied for array/table *array-name***

  The number of data elements supplied for the compile time table or array is less than the table or array capacity specified in the Extension specification. The remaining elements in the table or array will be initialized to blanks or zeros.

- **8-110 - RMS will not create *filename* using a secondary key**

  Migration RPG can only create a single key indexed file. It is unable to create a multiple key indexed file because it has no mechanism to specify multiple keys. If the file to be created uses multiple keys, it should be created beforehand so that the key definitions are already established. An alternate method for creating a multiple-key indexed file would be to have the RPG program build a single key indexed file and then convert it to a multiple key format using the appropriate FDL definition file and the OpenVMS CONVERT Utility.

C Migration RPG Compiler Error Messages

C.1 Overview

The following list represents the error messages that can be generated by the Migration RPG compiler. All error messages are fatal errors. Error messages can also be found on the program listing if one was selected.

Typically, in the program listing the error message will immediately follow the line which generated it. However, this is not always the case. For example, fields may be defined anywhere in the Calculation specifications. Therefore, errors pertaining to undefined fields, invalid field formats, and so on are not given until the end of the Calculation specifications.

C.2 Error Messages

- **1-002 - No source code specifications found**

  A file containing no RPG source code specifications has been submitted to the Migration RPG compiler.

- **1-003 - No records in source file**

  The RPG source file submitted to the Migration RPG compiler contains no records.

- **1-004 - Could not open input file**

  The compiler was unable to locate the specified program. Verify the command format and the location of the program to be compiled.

- **1-005 - Fatal error, compile aborted**

  A fatal error has been encountered which has aborted compilation of the program. Associated error messages should indicate the root cause of the problem.

- **1-006 - Error processing input source file name**

  There was an error processing the input source file specification. Check the file name and try again.

- **1-007 - Could not open print file**

  The compiler could not open a file to be used for the program listing. Verify the print file specification in the command line (if specified), the amount of free disk space, and the file and directory protection set in the current work area.

- **1-008 - Error reading from source file**

  An error has occured while reading from the RPG source file. This error results in a compiler abort. Associated error messages should indicate the root cause of the problem.

- **1-009 - Copybook file name missing**

  A /COPY compiler directive was encountered that was not followed by a copybook file name. Enter a valid copybook file name following the /COPY command.

- **1-010 - Could not open copybook: *copybook-name***

  A problem was encountered opening the specified copybook. Review the associated RMS error message to identify and correct the problem. This message was updated to display the affected copybook name.

- **1-011 - Could not read from copybook: *copybook-name***

  A problem was encountered reading from the specified copybook. Review the associated RMS error message to identify and correct the problem.

- **1-012 - Invalid compiler directive specified starting in column 7**

  A slash (/) was encountered in column 7 of a specification. The slash indicates that a compiler directive follows. No valid compiler directive was found.

- **1-013 - Record code in column 6 is not valid**

  The value contained in column 6 of the current record is not a defined RPG specification code and the record is not a part of a compile time table or array. Valid RPG specification codes are H, F, E, L, I, C, and O.

- **1-014 - Could not open object work file**

  The compiler could not open the work file necessary for the building of the object module. Verify the amount of the disk space available and the file and directory protection set in the current work area.

- **1-015 - Total Calc spec appears inside subroutine section**

  A total time Calculation specification appears within the subroutine section. Total time Calc specs must appear before the subroutine section. If this is an Auto Report program or a program using copybooks, use the AUTOC command to compile it. AUTOC ensures the program specifications are loaded in the correct order.

- **1-020 - Initial memory allocation error**

  The compiler was unable to allocate enough memory to compile the program. Check the memory available to the user account and the memory available on the system.

- **1-021 - Memory expansion allocation error**

  The compiler was unable to allocate enough memory to continue compiling the program. Check the memory available to the user account and the memory available on the system.

- **1-022 - Header (H) specification out of sequence**

  A Header specification appears out of sequence in the program (e.g., H after F). The specification sequence will need to be corrected before the program will compile successfully. If this is an Auto Report program or a program using copybooks, use the AUTOC command to compile it. AUTOC ensures the program specifications are loaded in the correct order.

- **1-023 - File (F) specification out of sequence**

  A File specification appears out of sequence in the program (e.g., F after I). The specification sequence will need to be corrected before the program will compile successfully. If this is an Auto Report program or a program using copybooks, use the AUTOC command to compile it. AUTOC ensures the program specifications are loaded in the correct order.

- **1-024 - Extension (E) specification out of sequence**

  An Extension specification appears out of sequence in the program (e.g., E after I). The specification sequence will need to be corrected before the program will compile successfully. If this is an Auto Report program or a program using copybooks, use the AUTOC command to compile it. AUTOC ensures the program specifications are loaded in the correct order.

- **1-025 - Line (L) specification out of sequence**

  A Line specification appears out of sequence in the program (e.g., L after I). The specification sequence will need to be corrected before the program will compile successfully. If this is an Auto Report program or a program using copybooks, use the AUTOC command to compile it. AUTOC ensures the program specifications are loaded in the correct order.

- **1-026 - Input (I) specification out of sequence**

  A Input specification appears out of sequence in the program (e.g., I after C). The specification sequence will need to be corrected before the program will compile successfully. If this is an Auto Report program or a program using copybooks, use the AUTOC command to compile it. AUTOC ensures the program specifications are loaded in the correct order.

- **1-027 - Calculation (C) specification out of sequence**

  A Calculation specification appears out of sequence in the program (e.g., C after O). The specification sequence will need to be corrected before the program will compile successfully. If this is an Auto Report program or a program using copybooks, use the AUTOC command to compile it. AUTOC ensures the program specifications are loaded in the correct order.

- **1-028 - Output (O) specification out of sequence**

  An Output specification appears out of sequence in the program (e.g., O before C). The specification sequence will need to be corrected before the program will compile successfully. If this is an Auto Report program or a program using copybooks, use the AUTOC command to compile it. AUTOC ensures the program specifications are loaded in the correct order.

- **1-065 - Input spec required for file *filename***

  This file has a File specification defining an input file, but no Input specification for the file was provided.

- **1-066 - Output specs required for Update, Combine or Add file *filename***

  This file has a File specification defining an output file, but no Output specification was provided.

- **1-067 - Lengths of matching record control fields disagree**

  The lengths of the fields defined for matching record control are different. The lengths of these fields must be the same.

- **1-068 - Indicators used but not defined**

  The indicators listed have been used to control program logic, but were not defined in the program.

- **1-069 - File *filename* is missing the associated E specification**

  The file definition requires an associated Extension specification.

- **1-080 - Fld def. could not be matched for *LIKE DEFN *fld1 fld2* on line *nnnn***

  Field 1 is not defined in the program. Therefore, the definition of field 2 using the *LIKE DEFN opcode failed.

- **1-082 - Field length invalid for *LIKE DEFN *fld1 fld2* on line *nnnn***

  The field length requested in the *LIKE DEFN statement on the specified line is not valid for the data type specified.

- **1-100 - Field or data structure name matches a reserved word**

  A field or data structure name has been encountered that matches an RPG reserved word. The following words are reserved for use by RPG:

*AUTO	*BLANK	*BLANKS	*CANCL
*DETC	*GETIN	*INP	*MODE
*OPCODE	*OUT	*PLACE	*RECORD
*SIZE	*STATUS	*ZERO	*ZEROS

- **1-101 - Compiler limit of 99,999 fields exceeded**

  A compiler limit of 99,999 fields definitions has been reached.

- **1-102 - Compiler limit of 99,999 data structures** exceeded

  A compiler limit of 99,999 data structure definitions has been reached.

- **1-103 - Nested copybook exceeds nesting limit** exceeded

  This error is generated by a copybook that is nested too deeply. A nested copybook is a copybook that is called from within another copybook. Copybooks can be nested up to a maximum of three levels.

- **1-109 - Error reading copybook. Contact tech support** exceeded

  An unexpected error has been encountered while processing a copybook. Please contact MSI about the problem and provide a copy of the source code that demonstrates the error.

- **1-270 - Data structure named *DS-name* larger than associated data field**

  The named data structure must be the same size or smaller than the associated data field.

- **2-001 - Maximum number (1036) of array/table definitions exceeded**

  Migration RPG supports a maximum of 1,036 tables and arrays in a single program.

- **2-022 - Column 15 has invalid debug option**

  Column 15 must contain a blank or a 1.

- **2-023 - Column 19 has invalid date format spec.**

  Column 19 must contain a blank, M, D, or Y.

- **2-024 - Column 21 has invalid inverted print spec.**

  Column 21 must contain a blank, I, J, or D.

- **2-026 - Column 26 has invalid collating sequence spec.**

  Column 26 must contain a blank, E, or S.

- **2-027 - Col 60 has invalid zoned-decimal entry, must be blank or Z**

  Column 60 must contain a blank (trailing numeric overpunch) or Z (zoned-decimal).

- **2-028 - Header control record is not first record processed**

  The header control record must be the first record to be processed, excluding comment lines at the beginning of the program. If the first specification processed is a U specification, the program is an Auto Report program and should be compiled using the AUTOC command.

- **2-029 - Invalid processing mode specified in column 28.**

  Column 28 should contain a blank for this type of file specification.

- **2-030 - File spec not first or preceded by Control spec.**

  File specifications must follow a Control (H) specification and precede any other specification types in the program. File specifications can also be the first specifications in a program if no Control specification is present. This error indicates that a File specification has been processed out of order.

- **2-031 - Invalid char in file name (Columns 7 - 14)**

  The file name does not meet OpenVMS naming standards. Valid characters in the file name are $, _, A - Z, and 0 - 9.

- **2-032 - File name not unique (Columns 7 - 14)**

  All file names must be unique.

- **2-033 - Invalid file type (Column 15)**

  Column 15 must contain an I, O, U, D, or C.

- **2-033A - Invalid file type (Column 15), Must be I, O, U, or C**

  Column 15 must contain an I, O, U, or C.

- **2-034 - Invalid file designation (Column 16)**

  Column 16 must contain a blank, P, S, R, C, D, or T.

- **2-034A - Invalid file designation (Column 16), Must be P, S, or D**

  Column 16 must contain a P, S, or D for a SPECIAL device file.

- **2-035 - Invalid End-Of-File (Column 17)**

  Column 17 must contain a blank or an E.

- **2-036 - Invalid sequence (Column 18)**

  Column 18 must contain a blank, A, or D.

- **2-037 - Invalid file format (Column 19)**

  Column 19 must contain a blank, F, or V. Blanks will default to fixed-length records (F).

- **2-038 - Invalid record size (Columns 24 - 27)**

  Record size must be between 1 and 9999. A blank record size for terminals or printers will default to 140.

- **2-039 - Invalid block size (Columns 20 - 23)**

  Block size must be blank or an even multiple of the record size. If blank, block size is assumed to be the same as the record size.

- **2-040 - Invalid mode of processing (Column 28)**

  Column 28 must be blank, L, or R.

- **2-041 - Invalid key size (Columns 29 -30)**

  Key size must be blank or from 1 to 99. Key size cannot be blank for indexed or record address files.

- **2-042 - Key size not valid for file (Columns 29 - 30)**

  Key size should only be specified for indexed or record address files.

- **2-043 - Invalid record address type (Column 31)**

  Column 31 should be blank, A, I, P, or K.

- **2-043A - Invalid record address type (Column 31), Must be blank**

  Column 31 must be blank for a SPECIAL device file.

- **2-044 - Invalid file organization (Column 32)**

  Column 32 must contain a blank, I, T, or 1 through 9.

- **2-044A - Invalid file organization (Column 32), Must be blank**

  Column 32 must be blank for a SPECIAL device file.

- **2-045 - Invalid overflow indicator (Columns 33 - 34)**

  Columns 33 - 34 must be blank or must contain a valid overflow indicator (OA-OG or OV).

- **2-045A - Invalid overflow indicator (Columns 33 - 34), Must be blank**

  Columns 33 - 34 must be blank for a SPECIAL device file.

- **2-046 - Key starting location not numeric (Columns 35 - 38)**

  Key starting column must be specified for indexed files and must be a numeric value.

- **2-047 - Invalid key starting location (Columns 35 - 38)**

  Key starting column plus key length must define a key contained within the given record size.

- **2-047A - Invalid key starting location (Columns 35 - 38), Must be blank**

  Columns 35 - 38 must be blank for a SPECIAL device file.

- **2-048 - Invalid extension code (Column 39)**

  Column 39 must be blank, E, or L.

- **2-048A - Invalid extension code (Column 39), Must be blank**

  Column 39 must be blank for a SPECIAL device file.

- **2-049 - Invalid device (Columns 40 - 46)**

  Valid devices are:

CONSOLE	CRT	DBDISK	DECTAP
DISC	DISK	DISK40	DKDISK
DMDISK	DPDISK	KEYBORD	PRINT
PRINTER	PRINTR	READ40	READER
SPECIAL	TAPE	TTY	WORKSTN

  These variations are necessary for compatibility with Digital Equipment Corporation, IBM, and UNIVAC® versions of RPG.

- **2-049A - A SPECIAL device file requires a subroutine name in columns 54-65.**

  Columns 54 - 65 must contain a subroutine name for a SPECIAL device file.

- **2-050 - Invalid device unit number (Column 46)**

  Used for Digital Equipment Corporation compatibility, unit numbers may be blank or from 1 to 7.

- **2-052 - Invalid tape label processing (Column 53)**

  Column 53 must be a blank or an S.

- **2-052A - Invalid tape label processing (Column 53), Must be blank**

  Column 53 must be blank for a SPECIAL device file.

- **2-053 - Invalid file addition code (Column 66)**

  Column 66 must be a blank, A, or U.

- **2-053A - Invalid file addition code (Column 66), Must be blank**

  Column 66 must be blank for a SPECIAL device file.

- **2-055 - Invalid external indicator (Columns 71 - 72)**

  Columns 71 - 72 must be blank or specify a valid external indicator (U1 - U8).

- **2-056 - Invalid tape rewind code (Column 70)**

  Column 70 must be a blank, U, R, N, K, L, or M.

- **2-056A - Invalid tape rewind code (Column 70), Must be blank**

  Column 70 must be blank for a SPECIAL device file.

- **2-057 - Invalid file sharing code (Column 73)**

  Column 73 must contain a blank, N, R, or S.

- **2-057A - Invalid file sharing code (Column 73), Must be blank**

  Column 73 must be blank for a SPECIAL device file.

- **2-058 - Invalid continuation keyword (Columns 54 - 59)**

  Keyword given in continuation of a File specification is invalid. Valid keywords are:

FMTS	ID	IND	INFDS
INFSR	NUM	SLN	

- **2-059 - Invalid name (Columns 60 - 65)**

  Invalid characters in field name following keyword in continuation line.

- **2-060 - Continuation line cannot be first**

  Continuation line not preceded by a File specification.

- **2-061 - More than 98 files specified**

  The maximum limit of 98 files in one program has been exceeded.

- **2-062 - Invalid file type or designation (Columns 15 - 16)**

  The pairing of entries in column 15 (file type) and column 16 (file designation) is invalid.

- **2-063 - Invalid combination in columns 16, 28, 31, 32**

  The type of file access specified by the entries in columns 16, 28, 31, and 32 is not allowed. Check and correct the file access specifications.

- **2-064 - Invalid file type for additions**

  The file type specified does not allow for the addition of new records.

- **2-067 - Invalid key size**

  The key size specified is not valid. Make sure the key length does not exceed the record size.

- **2-068 - EXTK valid only for Indexed files using segmented keys**

  The EXTK can only be specified with indexed file formats.

- **2-069 - Invalid key of reference**

  Columns 68 - 69 of the file specification must be blank or zero for accessing an indexed file by the primary key, or 01 - 99 to specify the number of the alternate key to be used.

- **2-069A - Invalid key of reference, Must be blank**

  Columns 68 - 69 of the file specification must be blank for a SPECIAL device file.

- **2-071 - No files defined**

  A file name has been processed in an Extension specification when there were no files defined in the File specifications.

- **2-072 - File not defined (Columns 11 - 18)**

  The file specified in the Extension specification has not been defined in a File specification.

- **2-073 - File referenced invalid (Columns 11 - 18)**

  The file referenced in the Extension specification was not defined as a table file in the File specifications.

- **2-073A - In file (Col 11-18) invalid, E missing from Col 39 of F-spec**

  The file referenced in the Extension specification was not defined as an extension file in the File specifications.

- **2-073B - In file (Col 11-18) is referenced by more than one E-specification**

  The file referenced in the Extension specification is also referenced in a previous Extension specification.

- **2-073C - To file (Col 19-26) is not defined as a Output file in the F-specs**

  The file referenced in the Extension specification as an output file is not defined in the File specifications as an output file.

- **2-073D - Out file (Col 19-26) is referenced by more than one E-spec**

  The file referenced in the Extension specification as an output file is also referenced in a previous Extension specification as an output file.

- **2-074 - Record address cannot link to this file**

  Two record address files have been specified in the Extension specification. A record address file cannot link to another record address file.

- **2-074A - Invalid combination of entries in Columns 16, 28, 31**

  The processing mode specified in columns 16, 28, and 31 is not valid when referencing an record address file. If column 16 is a P or S, column 28 must be an R and column 31 must be an I.

- **2-075 - Invalid file name (Columns 19 - 26)**

  The To File name specified in the Extension specification does not meet OpenVMS naming standards. Valid characters in the file name are $, _, A - Z, and 0 - 9.

- **2-076 - File not defined (Columns 19 - 26)**

  The To File name specified in the Extension specification has not been defined in the File specifications.

- **2-078 - Table or array name missing (Columns 27 - 32)**

  A table or array name must be supplied in the Extension specification. Table names must begin with ‘TAB’.

- **2-079 - Table or array name not unique (Columns 27 - 32)**

  Another table or array has been defined with the same name. Table and array names must be unique.

- **2-079A - Alternate table name not unique (Columns 46 - 51)**

  The alternate table or array name specified is not unique. Table and array names must be unique.

- **2-080 - Invalid number entries per record (Columns 33 - 35)**

  Entries per record must be a numeric value.

- **2-081 - Invalid entries per table or array (Columns 36 - 39)**

  Entries per table or array must be a numeric value.

- **2-082 - Invalid number decimals (Column 44)**

  Column 44 must be blank or 0 - 9.

- **2-082A - Invalid alternating number decimals (Column 56)**

  Column 56 must be blank or 0 - 9.

- **2-083 - Invalid format code (Column 43)**

  Column 43 must be blank, N, P, or B.

- **2-083A - Invalid alternating format code (Column 55)**

  Column 55 must be blank, N, P, or B.

- **2-084 - Invalid entry length (Columns 40 - 42)**

  Length of entry must be a numeric value.

- **2-084A - Invalid alternating entry length (Columns 52 - 54)**

  Length of entry must be a numeric value.

- **2-085 - Invalid sequence code (Column 45)**

  Column 45 must be blank, A, or D.

- **2-085A - Invalid alternating sequence code (Column 57)**

  Column 57 must be blank, A, or D.

- **2-087 - More than 97 tables defined**

  The maximum limit of 97 tables in one program has been exceeded.

- **2-088 - Table or array name is invalid**

  Invalid characters are present in the table or array name.

- **2-088A - Alternate table name is invalid**

  Invalid characters are present in the alternate table or array name.

- **2-091 - No files were defined by file spec.**

  No File specifications have been processed in this program. Therefore, there is no file to associate to the Line Counter specification.

- **2-092 - Invalid file name (Columns 7 - 14)**

  The file name specified in the Line Counter specification has not been defined in a File specification.

- **2-093 - Invalid entry in columns 18 - 19 (Form length)**

  Columns 18 and 19 must contain FL.

- **2-094 - Invalid overflow line (Columns 20 - 22)**

  The overflow line must be a numeric value.

- **2-095 - Invalid lines per page during overflow edit**

  The lines per page must be a numeric value.

- **2-096 - Overflow line greater than lines per page**

  The value assigned to the overflow line cannot exceed the total lines per page.

- **2-097 - Line spec valid only for PRINTER files**

  The device type of the file referenced in the Line Counter specification is not a printer. Only print files can be associated to Line Counter specifications.

- **2-098 - Invalid lines per page (Columns 15 - 17)**

  The total lines per page cannot exceed 112.

- **2-099 - Invalid entry in columns 23 - 24 (Overflow line)**

  Columns 23 - 24 must contain OL.

- **2-100 - Invalid continuation line. Line must follow a WORKSTN spec**

  Continuation lines are only permitted following a File specification defining a workstation device.

- **2-102 - Primary entry is table, secondary entry must also be a table**

  The primary entry in the Extension specification defined a table. Therefore, the secondary entry must also define a table. Table definitions must begin with TAB.

- **2-105 - More than 99 SPECIAL device files have been defined**

  Migration RPG only supports 99 SPECIAL device files.

- **2-106 - More than 1 continuation line for a SPECIAL device file has been defined**

  SPECIAL device files are only allowed to have one continuation line. The continuation line specifies the associated array.

- **2-107 - Invalid associated array name (Columns 54 - 59)**

  If a continuation line is coded for a SPECIAL device file, an associated array must be coded in columns 54 - 59.

- **2-200 - Invalid deferred write entry in column 74, must be N or blank**

  The entry made in the deferred write field of a File specification is not valid. This column must contain a blank or N. An entry of N indicates that the program is not to use the deferred write feature, which will increase process I/O and reduce program performance.

Disabling deferred write will cause each record that is output to this file to be immediately written to disk.

- **2A-901 - Invalid file description for file *filename***

  The combination of codes in columns 15, 16, 28, 31, 32, and 66 is not valid for this file.

- **2A-902 - Internal error while processing SPECIAL device file *filename***

  An internal compiler error has been encountered while processing the named SPECIAL device file. Please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **3-101 - Invalid data structure size (Columns 48 - 51)**

  An invalid size entry has been made for a data structure definition in columns 48 - 51.

- **3-102 - File must be referenced before defining fields**

  The first Input specification for a file must include the file name in columns 7 - 14.

- **3-103 - Previous data structure does not have a length or subfields**

  The previously defined data structure does not have a length defined in columns 48 - 51, nor does it contain any subfield definitions. One or the other is required for a valid data structure definition.

- **3-104 - Named DS len in col 48-51 does not match input fld/array len**

  The data structure length specified in columns 48 - 51 of the data structure definition does not match the length of the input field or array that the data structure redefines. Correct the data structure length or the redefined element length.

- **3-202 - No files defined - Input not permitted**

  No files were defined in this program. Therefore, Input specifications are not permitted.

- **3-203 - File name not defined (Columns 7 - 14)**

  The file name referenced in the Input specification is not defined in a File specification.

- **3-204 - Input spec valid only for input or update file**

  The file name referenced in the Input specification is not defined as an input or update file in the File specifications.

- **3-205 - Input spec not valid for table file**

  The file referenced in the Input specification has been defined in the File specifications as a table file. Input specifications are not valid for a file defined as a table file.

- **3-206 - Alpha sequence code, column 17, should be blank**

  The alpha sequence code in column 17 should be blank for this Input specification.

- **3-207 - Alpha sequence code, column 18, should be blank**

  The alpha sequence code in column 18 should be blank for this Input specification.

- **3-208 - Invalid sequence code (Columns 15 - 16)**

  Columns 15 - 16 must contain a numeric entry.

- **3-209 - Column 17 must be ‘1’ or ‘N’**

  If a numeric sequence is given in columns 15 - 16, column 17 must contain a 1 or N.

- **3-210 - Column 18 must be blank or ‘O’**

  If column 17 is a 1 or an N, column 18 must be blank or O.

- **3-211 - Invalid record indicator (Columns 19 - 20)**

  Columns 19 - 20 must be blank, "**" for look-ahead, or must contain a valid indicator (01-99, L1-L9, LR, H1-H9, RT) for record definitions.

- **3-212 - Invalid record position**

  Record position value must be blank or numeric. If it is a numeric value, it should define a position less than or equal to the record size that is defined in the File specification.

- **3-213 - Missing record position**

  The record position has been omitted from a record identification code (columns 21 - 24, 28 - 31, 35 - 38).

- **3-214 - Invalid entry following record position**

  Record position entry in a record identification code must be followed by a blank or N.

- **3-215 - Invalid C/Z/D entry**

  The character, zoned, or decimal portion of a record identification code must contain a C, Z, or D (columns 26, 33, 40).

- **3-225 - Columns 14 - 16 not ‘AND’ or ‘OR’**

  Columns 14 - 16 contain an invalid entry. On an Input specification that does not identify a file, only AND or OR can be specified in columns 14 - 16.

- **3-226 - Columns 17 - 20 should be blank for continuation line**

  Columns 17 - 20 should be blank on an Input specification continuation line. Continuation lines are defined by AND and OR specifications.

- **3-230 - Column 43 not blank, ‘P’, or ‘B’**

  The field format code in column 43 is invalid. It should be a blank (alpha or numeric), P (packed), or B (binary).

- **3-231 - Invalid field start (Columns 44 - 47)**

  The field starting column is not numeric, is zero, or is greater than the record size that has been defined for the file.

- **3-232 - Invalid field end (Columns 48 - 51)**

  Field ending column is not numeric, is zero, or is a value greater than the record size that has been defined for the file.

- **3-233 - Field too long**

  Field length exceeds 256 characters for an alphanumeric field, 15 characters for a numeric field, 8 for a packed field, or 4 for a binary field.

- **3-234 - Invalid decimal positions**

  Decimal positions must be blank or 0 - 9 and less than or equal to the field length.

- **3-235 - Duplicate field name in record**

  The field name in this Input specification has already been defined in the current record. Field names within a record must be unique.

- **3-236 - Definition of field does not match previous definition**

  The field has been previously defined in another record, but with a different length and/or format. Duplicate field names in the Input specifications must have matching sizes and formats.

- **3-237 - Field name illegal**

  Invalid characters used in field name. Valid characters are $, _, A - Z, and 0 - 9.

- **3-238 - Invalid control level (Columns 59 - 60)**

  Control level indicator specified must be a blank or L1 - L9.

- **3-239 - Invalid file type for control level**

  Control level indicators (L1 - L9) can only be assigned to Input specifications associated to primary and secondary input and update files or record address files.

- **3-240 - Control level cannot be binary field**

  Control levels (L1 - L9) cannot be assigned to binary data fields.

- **3-241 - Invalid matching field (Columns 61 - 62)**

  Matching field indicator must be a blank or M1 - M9.

- **3-242 - Invalid file type for matching field**

  Matching record indicators (M1 - M9) can only be assigned to Input specifications associated to primary and secondary input and update files or record address files.

- **3-243 - Invalid field or record entry (Columns 63 - 64)**

  Columns 63 - 64 not blank or a valid record indicator. Valid indicators are:

01 - 99	H1 - H9	L1 - L9
MR	RT	U1 - U8

- **3-244 - Invalid plus indicator (Columns 65 - 66)**

  The indicator specified for the field plus indicator is not valid. Valid indicators are:

01 - 99	H1 - H9	RT	U1 - U8

- **3-245 - Invalid minus indicator (Columns 67 - 68)**

  The indicator specified for the field minus indicator is not valid. Valid indicators are:

01 - 99	H1 - H9	RT	U1 - U8

- **3-246 - Invalid zero or blank indicator (Columns 69 - 70)**

  The indicator specified for the field zero or blank indicator is not valid. Valid indicators are:

01 - 99	H1 - H9	RT	U1 - U8

- **3-250 - Look-ahead sequence must be alpha (Columns 15 - 16)**

  The look-ahead sequence specified in columns 15 - 16 must contain alpha characters (A - Z).

- **3-251 - Columns 21-43 must be blank for look-ahead**

  Columns 21 - 43 must be blank on a look-ahead specification.

- **3-252 - Columns 59-70 must be blank for look-ahead**

  Columns 59 - 70 must be blank on a look-ahead specification.

- **3-260 - Data structure name cannot match file name**

  A file name has already been defined with the same name as this data structure. File and data structure names must be unique.

- **3-261 - Invalid character in data structure name**

  An invalid character has been encountered in the data structure name. Valid characters are $, _, A - Z, and 0 - 9.

- **3-262 - Column 18 invalid for data structure**

  Column 18 should be blank when defining a data structure.

- **3-263 - Invalid data structure subfield name**

  Illegal character contained in the data structure field name in columns 53 - 58. Valid characters are $, _, A - Z, and 0 - 9.

- **3-264 - Duplicate data structure subfield name**

  This field name has already been defined in this data structure. Duplicate field names are not allowed within a data structure.

- **3-265 - Field size must not exceed array definition size**

  The total array size defined in the Extension specifications is less than the field length specified for the array in this Input specification.

- **3-266 - Data structure name cannot exceed six characters**

  A data structure name can contain a maximum of six characters. Columns 13 and 14 of the Input specification naming a data structure must be blank.

- **3-267 - Columns 21 - 43 must be blank when defining a subfield**

  Columns 21 - 43 must be blank when defining a subfield in a data structure.

- **3-268 - Invalid duplicate data structure name**

  The name specified for the data structure has already been used by something other than an input field. Use a different name for one of the elements.

- **3-269 - Duplicate data structure name, name already used**

  The data structure name specified has already been used in another data structure or subfield definition. Change one of the names in question.

- **3-270 - Input field name conflicts with LDA data structure name**

  The name selected for the Local Data Area data structure conflicts with an input field name. Change one of the names.

- **3-271 - Invalid character in column 18, U or blank expected**

  Column 18 of an Input specification describing a data structure must contain a U to indicate a data structure for the Local Data Area or a blank to indicate a normal data structure.

- **3-272 - LDA data structure specified more than once**

  A data structure define prior to the one generating this error has already been designated to represent the Local Data Area (U in column 18 of data structure Input specification). Redefine or remove one of the conflicting data structures.

- **3-273 - Subfield end position exceeds size defined for data struct**

  The subfield in question is a part of a data structure which is related to an input field and has had its size predefined by the input field, or is part of a named data structure which had its size defined in columns 48 - 51 of the DS specification. The end position of the subfield exceeds the predefined data structure size. Correct the end position of the subfield, redefine the input field, or correct the length specified for the named data structure.
  This error may also appear for a subfield defined within the Local Data Area (LDA) data structure if the subfield end position exceeds 512.

- **3-274 - Data structure subfield length not equal to input field len**

  An input field has been redefined by a data structure subfield and the lengths of the two field definitions do not match. Data may be lost due to field truncation brought on by the differing field lengths.

- **3-275 - Subfield end position exceeds size of Local Data Area**

  The end position of a subfield in the Local Data Area data structure exceeds the length of the Local Data Area. The Local Data Area is defined to be 512 bytes long. Correct the subfield definition.

- **3-276 - Data structure subfield length not equal to array length**

  A subfield has been parsed which redefines an array entry. The length of the subfield must be equal to the entire length of the array. The length of the array is equivalent to the number of entries in the array multiplied by the entry length. Remember that packed and binary arrays are stored in memory in zoned decimal format, so calculate the subfield length accordingly.

- **3-277 - Field size exceeds array element size**

  A field has been parsed which defines an array element. The length of the field must be equal to the array element length defined in the array's E specification.

- **3A-297 - Internal error while processing SPECIAL device file *filename***

  An internal compiler error has been encountered while processing the named SPECIAL device file. Please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **3A-298 - Too many files. Over 98 files defined for input.**

  More than 98 files have been specified in the Input specifications. The maximum number of files which can be defined within an RPG program is 98.

- **3A-299 - Multiple definitions of file in Input specifications**

  The same file has been defined in a previous section of Input specifications. Multiple record and field definitions for the same file must be grouped together within the Input specifications.

- **4-300 - Invalid control level (Columns 7 - 8)**

  Columns 7 - 8 must be blank, AN, OR, SR, L1 - L9, or LR.

- **4-301 - Invalid value in column 9**

  Column 9 should contain a blank or N.

- **4-302 - Invalid value in column 12**

  Column 12 should contain a blank or N.

- **4-303 - Invalid value in column 15**

  Column 15 should contain a blank or N.

- **4-304 - Invalid controlling indicator in columns 10 - 11**

  Columns 10 - 11 should be blank or contain a valid indicator. Valid indicators are:

01 - 99	H1 - H9	KA - KN	KP - KY
L1 - L9	LR	MR	OA - OG
OV	RT	U1 - U8	*

- **4-305 - Invalid controlling indicator in columns 13 - 14**

  Columns 13 - 14 should be blank or contain a valid indicator. Valid indicators are:

01 - 99	H1 - H9	KA - KN	KP - KY
L1 - L9	LR	MR	OA - OG
OV	RT	U1 - U8	

- **4-306 - Invalid controlling indicator in columns 16 - 17**

  Columns 16 - 17 should be blank or contain a valid indicator. Valid indicators are:

01 - 99	H1 - H9	KA - KN	KP - KY
L1 - L9	LR	MR	OA - OG
OV	RT	U1 - U8	

- **4-307 - Invalid operation code in columns 28 - 32**

  The opcode specified in this Calculation specification is not a defined Migration RPG opcode.

- **4-308 - Invalid use of * indicator in columns 9 - 11**

  Columns 7 - 8 cannot contain AN or OR and columns 9 - 10 must be blank when using the * indicator.

- **4-309 - Too many literals**

  The number of literals defined in this program exceeds the compiler's capacity to deal with them. Reduce the number of literals in the program.

- **4-310 - BEGSR and ENDSR statements out of sequence**

  BEGSR and ENDSR statements are not paired correctly.

- **4-311 - Missing operation code in columns 28 - 32**

  An opcode must be specified in a Calculation specification conditioned by the * indicator.

- **4-312 - * indicator in columns 9 - 11 not allowed on first detail calculation**

  The first detail Calculation specification cannot be conditioned by the * indicator.

- **4-313 - * indicator in columns 9 - 11 not allowed on first total calculation**

  The first total time Calculation specification cannot be conditioned by the * indicator.

- **4-314 - Result field name invalid (Columns 43 - 48)**

  Result field name contains invalid characters. Valid characters are $, _, A - Z, and 0 - 9.

- **4-315 - Invalid resulting indicator (Columns 54 - 55)**

  Columns 54 - 55 should be blank or contain a valid indicator. Valid indicators are:

01 - 99	H1 - H9	KA - KN	KP - KY
L1 - L9	LR	RT	U1 - U8

- **4-316 - Invalid resulting indicator (Columns 56 - 57)**

  Columns 56 - 57 should be blank or contain a valid indicator. Valid indicators are:

01 - 99	H1 - H9	KA - KN	KP - KY
L1 - L9	LR	RT	U1 - U8

- **4-317 - Invalid resulting indicator (Columns 58 - 59)**

  Columns 58 - 59 should be blank or contain a valid indicator. Valid indicators are:

01 - 99	H1 - H9	KA - KN	KP - KY
L1 - L9	LR	RT	U1 - U8

- **4-318 - Invalid half-adjust option (Column 53)**

  Column 53 should contain a blank or H.

- **4-319 - Factor 1 should be blank (Columns 18 - 27)**

  This opcode requires Factor 1 to be blank.

- **4-320 - Factor 2 should be blank (Columns 33 - 42)**

  This opcode requires Factor 2 to be blank.

- **4-321 - Result field should be blank (Columns 43 - 48)**

  This opcode requires the result field to be blank.

- **4-323 - Resulting indicators must be present (Columns 54 - 59)**

  This opcode requires resulting indicators.

- **4-324 - Name previously defined**

  This label was previously used with a TAG or BEGSR opcode.

- **4-325 - Invalid label format**

  A label must begin with an alphabetic character and not exceed 6 characters in length.

- **4-326 - ENDSR stmt must be preceded by BEGSR stmt**

  An ENDSR opcode has been encountered that is not associated with a BEGSR opcode.

- **4-327 - Controlling indicators must be blank (Columns 9 - 17)**

  This opcode requires columns 9 - 17 to be blank.

- **4-328 - Factor 2 must be blank unless INFSR subroutine**

  Factor 2 must be blank or an ENDSR statement unless the subroutine is defined as an INFSR subroutine.

- **4-329 - Invalid file definition for use with this opcode**

  The file name specified in the Calculation specification was not defined in a File specification.

- **4-330 - Factor 1 or Result field must have entry**

  Factor 1 or the result field is required with this opcode.

- **4-331 - Illegal value in mask (Columns 33 - 42)**

  A bit mask may contain up to 8 characters, 0 - 7.

- **4-332 - When using XFOOT, Factor 2 must be a numeric array**

  When using the XFOOT operation code, Factor 2 must be a numeric array.

- **4-333 - Resulting indicator required (Columns 54 - 59)**

  The opcode used in this Calculation specification requires a resulting indicator.

- **4-334 - Resulting indicator required (Columns 58 - 59)**

  The opcode used in this Calculation specification requires a resulting indicator.

- **4-335 - Invalid file type**

  The file type specified in column 15 of File specification is not valid for this operation.

- **4-338 - HI or LO indicators invalid with unsequenced arrays. (Columns 54 - 57)**

  Only the EQUAL indicator may be used for an unsequenced array.

- **4-339 - Cannot specify both HI and LO indicators with LOKUP. (Columns 54 - 57)**

  Valid resulting indicator combinations with the LOKUP opcode are a sequenced table or array are: HIGH; LOW; HIGH and EQUAL; or LOW and EQUAL.

- **4-340 - Factor 2 not array or table**

  Factor 2 in a LOKUP operation must be an array or table.

- **4-341 - Factor 2 and Result field must be tables**

  Factor 2 and the result field must be an array or table in a LOKUP operation.

- **4-342 - Tables cannot have subscripts**

  Table elements cannot be referenced by subscript. Only arrays can be referenced by subscript.

- **4-343 - Must have at least one resulting ind. (Col. 54 - 59)**

  This opcode requires at least one resulting indicator to be specified.

- **4-344 - Factor 1 invalid (Columns 18 - 27)**

  Factor 1 must be blank, a field name, or a literal.

- **4-345 - Factor 2 invalid (Columns 33 - 42)**

  Factor 2 must be an array name with no index or subscript.

- **4-346 - MVR operation must immediately follow a DIV operation**

  The MVR (Move Remainder) opcode is only valid immediately following a divide operation (DIV).

- **4-347 - Columns 9 through 27 must be blank**

  Columns 9 - 27 must be blank when using the RLABL opcode.

- **4-348 - Columns 53 through 59 must be blank**

  Columns 53 - 59 must be blank when using the RLABL opcode.

- **4-349 - Resulting indicators must be present (Columns 54 - 59)**

  Resulting indicators must be present with this opcode.

- **4-350 - Resulting indicators not allowed (Columns 54 - 59)**

  Resulting indicators are not allowed with this opcode.

- **4-351 - Invalid number decimals (Column 52)**

  The decimal field is not blank, is not 0 - 9, or is greater than the total field length.

- **4-360 - Factor 1 required (Columns 18 - 27)**

  Factor 1 is required with this opcode.

- **4-361 - Factor 2 required (Columns 33 - 42)**

  Factor 2 is required with this opcode.

- **4-362 - Result field required (Columns 43 - 48)**

  The result field is required with this opcode.

- **4-364 - Invalid control level specified in subroutine section (Col 7 - 8)**

  An invalid control level has been specified in columns 7 - 8 within the subroutine section of the program. Columns 7 - 8 should be blank or contain an SR within the subroutine section of an RPG program.

- **4-366 - Invalid ANd or OR control level specified (Col 7 - 8)**

  The And (AN) or OR specified in columns 7 - 8 is not valid in this context.

- **4-370 - Invalid subroutine control level entry (Columns 7 - 8)**

  A subroutine control level (SR) entry has been made in columns 7 - 8 in code that is not part of a subroutine.

- **4-371 - Compiler limit of 9,999 PARM statements reached**

  The compiler is capable of processing up to 9,999 PARM statements in a single program. This error indicates that this limit has been reached. It will be necessary to reduce the number of PARM statements in the program to compile it successfully.

- **4-372 - Literal field size of 9,999 bytes exceeded**

  A literal field definition exceeds 9,999 bytes in length. How you succeeded in doing this is a mystery. Please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **4-373 - Invalid literal**

  The literal expression in factor 1 or factor 2 is not valid. Correct the expression and recompile the program.

- **4-374 - Invalid numeric literal**

  The numeric literal expression in factor 1 or factor 2 is not valid. Correct the expression and recompile the program.

- **4-375 - Too many decimal points in numeric literal**

  The numeric literal expression in factor 1 or factor 2 contains too many decimal points. Correct the expression and recompile the program.

- **4-380 - Invalid subscript on result field (Columns 43 - 48)**

  The subscript used on the result field in an RLABL statement is not numeric.

- **4-381 - Invalid result field length**

  The length specified for the result field is invalid. Alpha fields can have a maximum length of 256 characters. Numeric fields can have a maximum length of 15 digits.

- **4-500 - AND stmt does not follow a DOU, DOW, IF, OR, or AND**

  An ANDxx opcode is only valid immediately following a statement containing a DOUxx, DOWxx, IFxx, ORxx, or ANDxx opcode.

- **4-501 - OR stmt does not follow a DOU, DOW, IF, OR, or AND**

  An ORxx opcode is only valid immediately following a statement containing a DOUxx, DOWxx, IFxx, ORxx, or ANDxx opcode.

- **4-801 - *Opcode* statement at line *nnnn* missing END statement**

  The structured opcode on the specified line is missing its closing END statement.

- **4-802 - END should follow last CAS in series beginning on line *nnnn***

  The last CASxx statement beginning on the specified line should be followed by a closing END statement.

- **4-900 - END stmt encountered with no matching IF, DO, or CAS stmt**

  An END opcode has been encountered that does not correspond to any structured opcode previously processed by the compiler. END statements must always be paired with a structured opcode (IFxx, DOxx, CASxx).

- **4-902 - Structured opcode nesting level exceeds 100 levels deep**

  The compiler does not support the nesting of structured opcodes beyond 100 levels. This is for your own protection. Just say 'No!' to uncontrolled nesting.

- **4-903 - END statement must immediately follow last CAS statement**

  An END statement must immediately follow the last CASxx statement in a Case construct.

- **4-904 - Number of DO stmts exceeds compiler limitation** *of 9999*\ bold)

  The number of DO statements processed in the program exceeds a compiler limit of 9,999. Hey, somebody has to set a limit on what you are DOing.

- **4-905 - Invalid subroutine name (Columns 43 - 48)**

  An invalid character has been specified in a subroutine name. Valid characters are $, _, A - Z, and 0 - 9.

- **4-906 - Cannot add DO index to table compiler limit of 256 exceeded**

  The compiler limit of 256 nested DOxx constructs has been exceeded. This limit has been established for your own protection. Watch what you are DOing!

- **4-907 - Cannot extract DO index from table**

  The compiler is unable to extract the DOxx index from its internal table. Please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **4-909 - ELSE statement must be part of an IFxx/END pair**

  An ELSE opcode has been encountered outside the bounds of an IFxx/END construct. ELSE statements must be associated to IF statements.

- **4-911 - Factor 1 must be *LIKE for DEFN statement**

  The only keyword permitted in factor one of a DEFN statement is *LIKE.

- **4-912 - Result field already defined as a table or array**

  A result field which has previously been defined in an Extension specification as a table or array cannot be redefined in the Calculation specifications.

- **4-913 - Factor 2 cannot be a literal**

  Literals are not permitted in factor 2 in this context.

- **4-914 - Cannot add DOU fields to table compiler limit of 256 exceeded**

  The compiler limit of 256 nested DOUxx constructs has been exceeded. This limit has been set for your own protection. No DOing Until you go blind.

- **4-915 - Cannot extract DOU fields from table**

  The compiler is unable to extract DOUxx construct information from an internal table. Please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **4-916 - Factor 2 must be a literal**

  Factor 2 must be a literal in an EXTRN statement.

- **4-917 - Factor 2 exceeds maximum length of 31 characters**

  The program or routine name specified in an EXTRN statement cannot exceed 31 characters.

- **4-918 - Factor 1 must be a field name, not an array or literal**

  Factor 1 must be a field name, not a table, array, or literal.

- **4-919 - Duplicate field name, EXTRN field names must be unique**

  Fields used in CALL statements and defined by an EXTRN statement must have unique names.

- **4-921 - Invalid literal specified in factor 2**

  The literal specified in factor 2 is not valid; an ending quote (') was not found.

- **4-922 - Invalid field name, field not defined by an EXTRN statement**

  The field name specified in factor 2 was not defined by an EXTRN statement. It is not valid for this opcode.

- **4-924 - Duplicate field name, PLIST name already in use**

  The name used in this PLIST statement has already been used as a field, table, or array name.

- **4-925 - Duplicate field name, PLIST name already defined by another PLIST**

  The name used in this PLIST statement has already been used in another PLIST statement.

- **4-926 - Over 9,999 CALL statements processed, CALL opcode limit exceeded**

  A compiler limit of 9,999 CALL statements has been exceeded. Please do not CALL again.

- **4-927 - Literal or named constant cannot be specified in F1 with PARM opcode**

  Literal and named constants are not allowed in factor 1 of a PARM statement.

- **4-928 - Array element not allowed in factor 1 or result fld of a *ENTRY PLIST**

  Array elements using a variable as the index field cannot be specified in factor 1 or the result field of a PARM statement used in a *ENTRY PLIST.

- **4-929 - Result field cannot be a table name**

  The result field in a PARM statement cannot be a table name.

- **4-930 - Multiple definition of *ENTRY PLIST**

  A *ENTRY PLIST has been defined more than once by a PLIST statement. Only one *ENTRY PLIST is allowed in a program.

- **4-931 - Undefined array**

  An undefined array element is present in factor 1, factor 2, or the result field.

- **4-932 - Invalid array subscript. Subscript cannot be zero**

  An array element is subscripted with a zero. Zero is not a valid subscript.

- **4A-350 - Opcode *opcode* is undefined**

  The opcode referenced in this Calculation specification is not a defined opcode.

- **4A-352 - Field *fld-name* is undefined at line number *nnnn***

  The specified field has not been defined in the Extension, Input, or Calculation specifications.

- **4A-353 - Array *ary-name* exceeded by explicit index in line *nnnn***

  The explicit subscript specified on this array exceeds the total number of elements defined for the array.

- **4A-354 - Index *idx* is not numeric at line *nnnn***

  The subscript field specified is not defined as a numeric field. All subscript fields must be defined as numeric.

- **4A-355 - Field *fld-name* is not numeric at line *nnnn***

  This field is used in a numeric opcode and is not defined as a numeric field. If this error is generated in association with the PARM opcode, then one of the following conditions is true:

  - Factor 2 is defined as an alpha field and the result field is defined as a numeric field.
  - The result field is defined as an alpha field and factor 1 is defined as a numeric field.

  In either case, the system uses a Z-ADD operation to move the results of a PARM operation to the receiver field. The Z-ADD operation does not allow any of the fields it acts upon to be defined as alpha. Correct the PARM statement so that all of the fields referenced have the same data type.

- **4A-356 - Both fields must be numeric or alpha on line *nnnn***

  Both fields must be alpha or numeric in a Compare (COMP) operation.

- **4A-357 - Tag *tag-name* is not defined - Line *nnnn***

  The specified tag has been used, but has not been defined anywhere in the program.

- **4A-358 - Invalid destination for tag *tag-name* on line *nnnn***

  The program has attempted to use a tag to branch to an invalid destination. For example, branching from the detail Calculation specification section to the subroutine section or a section of level break Calculation specifications using a GOTO is not allowed.

- **4A-359 - Factor 1 length does not match key size - Line *nnnn***

  The length of the field specified in factor 1 does not match the key size of the file being processed.

- **4A-360 - Field *fld-name* is undefined at line number *nnnn***

  A field has been referenced on the specified line that is not defined in the Extension, Input, or Calculation specifications.

- **4A-361 - Result field *field-name* must be alpha for TESTN opcode at line number**

  The result field must be defined as an alpha field when used with the TESTN opcode.

- **4A-366 - Subroutine *tag-name* referenced, but not defined**

  A subroutine name has been referenced in an EXSR statement, but the subroutine is not defined by a BEGSR statement.

- **4A-370 - Array specified in factor 1 of COMP on line *nnnn* must have a subscript**

  An array element specified in a COMP operation must be subscripted.

- **4A-371 - Array specified in factor 2 of COMP on line *nnnn* must have a subscript**

  An array element specified in a COMP operation must be subscripted.

- **4A-375 - File not defined on line *nnnn***

  A file has been referenced on the specified line, but was not defined in the File specifications.

- **4A-380 - PLIST name *field-name* used on line *nnnn* not defined**

  The PLIST specified in the CALL statement on the indicated line has not been defined anywhere in the program Calculation specifications.

- **4A-381 - Error building transfer address for *field-name* on line *nnnn***

  The compiler is unable to construct a transfer address for the field specified in the result field of the PARM statement on the indicated line. Check the PARM statement for validity and, if correct, please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **4A-382 - The field *field-name*, used on line *nnnn*, is not defined**

  The field specified has not been defined in the Extension, Input, or Calculation specifications.

- **4A-383 - Field type for PARM field *field-name* on line *nnnn* could not be determined (Build)**

  The compiler has been unable to determine the field type (alphanumeric, numeric, packed, binary, array, table, data structure) of the named field used in the result field of a PARM statement while building a descriptor. Please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **4A-384 - Field type for PARM field *field-name* on line *nnnn* could not be determined (Send)**

  The compiler has been unable to determine the field type (alphanumeric, numeric, packed, binary, array, table, data structure) of the named field used in the result field of a PARM statement while preparing the field as a sending field. Please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **4A-385 - Field type for PARM field *field-name* on line *nnnn* could not be determined (Receive)**

  The compiler has been unable to determine the field type (alphanumeric, numeric, packed, binary, array, table, data structure) of the named field used in the result field of a PARM statement while preparing the field as a receiving field. Please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **4A-900 - END stmt encountered with no matching IF, DO or CAS stmt**

  An END statement has been encountered that does not have a matching structured opcode associated to it. END statements must be paired with IFxx, DOxx, or CAS opcodes.

- **4A-901 - Number of structured opcodes exceeds compiler limit of 1,000,000**

  A compiler limit of 1,000,000 structured opcodes in one program has been reached. Does anyone understand what this program does?

- **4A-902 - Structured opcode nesting level exceeds 100 levels deep**

  A compiler limit of 100 levels of nested structured opcodes has been reached. This limit has been established for your own protection. Please help prevent uncontrolled nesting.

- **4A-903 - Number of logical ANDs and ORs exceeds compiler limit of 50,000,000**

  A compiler limit of 50,000,000 boolean structures for one statement has been reached. Do *not* send this code to the Migration RPG support center.

- **4A-909 - ELSE statement must be part of an IFxx/END pair**

  An ELSE statement has been encountered that is not part of an IFxx construct. ELSE statements must always be included in an IFxx/END construct.

- **4A-910 - CALL parameter limit of 5,000 exceeded**

  A compiler limit of 5,000 PARM statements associated to a single CALL has been exceeded. Get a grip!

- **4A-930: Line *nnnn*: Factor 2 field length must be 1 for bit opcode**

  A BITOFF, BITON, or TESTB statement has been encountered where the field referenced in factor 2 has a length greater than one. Bit operations are designed to work on one character fields only.

- **5-010 - Calculated output field size of *fld-siz* exceeds end position**

  The size of the output field in this Output specification exceeds the specified ending position of the file.

- **5-401 - Invalid data in columns 14 - 16**

  Column 15 may be blank, H, D, T, or E. If it does not contain one of these codes, columns 14 - 16 must be blank or contain an AND or OR instruction.

- **5-402 - Invalid data in column 23**

  Column 23 must contain a blank or N.

- **5-403 - Invalid data in column 26**

  Column 26 must contain a blank or N.

- **5-404 - Invalid data in column 29**

  Column 29 must contain a blank or N.

- **5-405 - Invalid indicator in columns 24 - 25**

  An invalid indicator has been specified in columns 24 - 25 of the Output specification. Valid indicators are:

01 - 99	1P	H1 - H9	KA - KN
KP - KY	L1 - L9	LR	MR
OA - OG	OV	RT	U1 - U8
*			

- **5-406 - Invalid indicator in columns 27 - 28**

  An invalid indicator has been specified in columns 27 - 28 of the Output specification. Valid indicators are:

01 - 99	1P	H1 - H9	KA - KN
KP - KY	L1 - L9	LR	MR
OA - OG	OV	RT	U1 - U8

- **5-407 - Invalid indicator in columns 30 - 31**

  An invalid indicator has been specified in columns 30 - 31 of the Output specification. Valid indicators are:

01 - 99	1P	H1 - H9	KA - KN
KP - KY	L1 - L9	LR	MR
OA - OG	OV	RT	U1 - U8

- **5-408 - Edit code (Col 38) must be blank when defining WORKSTN scrn**

  The edit code in column 38 must be blank when the output field defines a WORKSTN screen.

- **5-409 - No files were defined**

  Output specifications supplied when no output files have been defined in the File specifications.

- **5-410 - File name not defined**

  The file name specified in the Output specification was not defined in the File specifications.

- **5-411 - File not defined as output or update**

  The file defined in the Output specification was not defined in the File specifications as an output capable file.

- **5-412 - Invalid line type (Column 15)**

  Column 15 must be H, D, T, or E for file and record output specifications.

- **5-413 - Invalid Skip Before (Columns 19 - 20)**

  Columns 19 - 20 must be blank, 01 - 99, A0 - A9, or B0 - B2.

- **5-414 - Field not defined (Columns 32 - 37)**

  The field referenced in this Output specification has not been defined by an Extension, Input, or Calculation specification.

- **5-415 - Invalid edit code (Column 38)**

  The edit code specified in column 38 is not valid. Valid edit codes are blank, 1, 2, 3, 4, A, B, C, D, J, K, L, M, X, Y, and Z.

- **5-416 - Invalid data type in column 44: blank, B, P, or Z expected**

  Column 44 must contain a data type entry of blank (alphanumeric), B (binary), P (packed-decimal), or Z (zoned-decimal).

- **5-417 - Invalid Blank After (Column 39)**

  Column 39 must contain a blank or B.

- **5-418 - Invalid edit word (Columns 45 - 70)**

  The edit word specified in columns 45 - 70 of the Output specification is not a valid edit word.

- **5-419 - Invalid field ending position (Columns 40 - 43)**

  Field ending position is not within the record size as defined in the File specification.

- **5-420 - Invalid literal definition (Columns 45 - 70)**

  The literal specified in columns 45 - 70 of the Output specification is not valid. Check the use of quotes (') and be sure a field name has not been specified in columns 32 - 37.

- **5-421 - Invalid Skip After (Columns 21 - 22)**

  Columns 21 - 22 must contain a blank, 01 - 99, A0 - A9, or B0 - B9.

- **5-422 - Invalid data in columns 16 - 18**

  Columns 16 - 18 must contain an ADD, DEL, fetch overflow, or line spacing.

- **5-423 - AND/OR must be preceded by file entry**

  An AND or OR entry has been encountered that is not preceded by a file entry or record entry.

- **5-424 - Field entry must be preceded by file entry**

  A file entry must precede a field entry.

- **5-425 - Cannot subscript a field**

  A subscript cannot be specified on a data field.

- **5-426 - Subscript field not defined or illegal**

  The subscript field used on this Output specification line is not defined or is invalid as a subscript.

- **5-427 - Cannot edit alpha field**

  An edit code or edit mask cannot be specified for an alpha field.

- **5-428 - Edit word length does not match field length**

  The length of the edit word does not match the length of the data field.

- **5-429 - 'ADD' required in columns 16 - 18**

  The ADD instruction is required in columns 16 - 18 of this output file or record specification.

- **5-430 - Field length exceeds output field end position**

  The ending position specified in this Output specification is less than the length of the field being output. The ending position of a field Output specification must be at least equal to the length of the field.

- **5-431 - Invalid number of replaceable characters in edit mask**

  The number of replaceable characters in the edit mask is less than the size of the output field. Enlarge the edit mask.

- **5-432 - Use of * indicator is invalid with AND/OR line**

  The * indicator cannot be used with AND and OR conditions.

- **5-433 - Invalid use of * indicator, line type must match last line type specified**

  If the * indicator is used, the line type, H, D, E, or T, must match the line type of the previous output record.

- **5-434 - Invalid use of * indicator, can not be specified on first field of record**

  The first field in an output record cannot specify the * indicator.

- **5-435 - Invalid use of * indicator, columns 23 - 24 must be blank**

  Columns 23 and 24 must be blank when using the * indicator.

- **5-436 - Use of AND/OR line invalid with * indicator**

  AND and OR operators cannot be used with the * indicator.

- **5-437 - Invalid field name entry, contains embedded blanks**

  Field names cannot contain embedded blanks.

- **5-450 - Invalid entry in Space Before field (Column 17)**

  The entry in the Space Before column is not valid. Valid entries are blank and 0 - 3.

- **5-451 - Invalid entry in Space After field (Column 18)**

  The entry in the Space After column is not valid. Valid entries are blank and 0 - 3.

- **5-460 - External reference field not valid for output**

  Fields defined by the EXTRN opcode cannot be used in the Output specifications.

- **5-461 - PLIST name invalid as an output field**

  Fields defined by the PLIST opcode cannot be used in the Output specifications.

- **5-462 - Cannot apply an edit mask to a packed or binary field**

  An edit mask cannot be applied to a packed-decimal or binary output field. Remove the edit mask or clear column 44 (data type).

- **5-463 - Data Type (Col 44) must be blank when defining WORKSTN scrn**

  The data type in column 44 must be blank in an output field that defines a WORKSTN screen.

- **5-464 - Data Type (Col 44) must be blank when outputting alpha data**

  The data type in column 44 must be blank when outputting alpha data.

- **5A-470 - Internal error while processing SPECIAL device file**

  An internal compiler error has been encountered while processing a SPECIAL device file. Please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **6-502 - *Field* used in RLABL but not defined in line *nnnn***

  The specified field is used is an RLABL statement, but not defined in the program.

- **6-601 - Invalid format for compile time table**

  The format of the compile time data entered at the end of the program is not consistent with the data type specified in column 43 of the Extension specification defining the table.

- **6-602 - No space available in table/array**

  More compile time data was entered at the end of the program than the Extension specification defining the table or array allowed for. Remove the extra data elements or enlarge the array or table definition.

- **6-603 - Invalid character combination in entry**

  The delimiter used in the table/array data is not valid. Valid delimiters are a single quote ('), double quote ("), back-slash (\ ), and tilde (~).

- **6-605 - Table/array is out of sequence**

  Column 45 of the Extension specification indicates ascending or descending order. The table/array is not in the specified sequence.

- **6-612 - Field *fld-name* referenced by a CALL but not defined by an EXTRN**

  A CALL statement used the specified field name to reference a program or routine, but the field name was not defined by an EXTRN statement.

- **6-620 - Invalid file translate entry**

  The file translation entry specified does not contain a file name.

- **6-621 - Non-numeric data in numeric table**

  The table was defined with a numeric format code in the Extension specification, but is being loaded with non-numeric data from the compile time table.

- **6-630 - Invalid associated array *array-name* used with SPECIAL file *filename***

  The associated array name supplied is not a valid Migration RPG array defined in the Extension specifications. NOTE: Migration RPG does not support the use of associated tables with SPECIAL device files.

- **8-700 - Internal field allocation error on *fld-name***

  The compiler has lost track of a field definition internally. Please send the source code generating this error and a copy of the compile command used to compile it to the Migration RPG support center.

- **8-701 - Subscript *fld-name* not defined**

  The specified subscript was not defined in the Input or Calculation specifications.

D Screen Format Generator (SFG) Warning Messages

D.1 Overview

The following list represents the warning messages which can be generated by the Screen Format Generator. Warning messages indicate a condition which may cause a problem at runtime, but is not serious enough to warrant aborting the compile. Warning messages are displayed to the terminal and can also be found on the screen listing if one was selected.

D.2 Warning Messages

- **001 - TAB character encountered and converted to a blank**

  The screen compiler does not expand tabs; it simply converts them to blanks. This generally causes problems with the format of the line being processed. Do not use tabs when coding S, H, and D specifications.

- **005 - Indicator based output not allowed with constant fld Y assumed in col. 23**

  Indicator-based output is not allowed when a field type of Constant has been assumed. The indicator will be ignored by the compiler.

- **007 - Constant defined but not output; Y assumed in column 23**

  A constant value was defined in columns 57 - 79, but a Y was not specified in columns 23 - 24. The compiler will place a Y in column 23 and assume the data is intended for output.

- **010 - Help screen table size of 1000 entries exceeded**

  A compiler limit of 1,000 help screens has been reached. Further help screens will be ignored.

- **011 - Screen table size of 1000 entries exceeded**

  A compiler limit of 1,000 screens has been reached. Further screens will be ignored. Try screening your screens.

- **012 - Invalid PAGE size entered, page size defaulted to 60**

  The page size entered on the command line using the /PAGE qualifier must be between 10 and 200. Invalid page sizes are ignored and the compiler uses the default page size of 60.

- **013 - Position cursor entry on output only field. Entry ignored**

  The position cursor field (columns 32 - 33) contains an entry in a D specification that defines an output only field. Since the cursor cannot be positioned on an output only field, a warning is issued and the entry is ignored.

- **100 - Duplicate screen name, *XXXXXXXX* in S specification**

  The specified screen name has already been used by a previous Screen specification. Change one of the screen names to make it unique.

- **102 - No S specification found for help screen *XXXXXXXX***

  The page size entered on the command line using the /PAGE qualifier must be between 10 and 200. Invalid page sizes are ignored and the compiler uses the default page size of 60.

E Screen Format Generator (SFG) Error Messages

E.1 Overview

The following list contains the error messages which can be generated by the Screen Format Generator. All error messages are fatal errors. The source code generating the message must be corrected before the screen will compile successfully. Error messages are displayed to the terminal and can also be found on the screen listing if one was selected.

E.2 Error Messages

- **003 - No records in source file**

  The screen source file submitted to the compiler contains no records.

- **004 - Error opening or connecting to the input file**

  The compiler was unable to open or connect to the input file. Review the system error message associated with this error and correct the problem.

- **006 - Error processing input source file name**

  The screen format file in the compile command line could not be found. Verify the command format and the location of the file to be compiled.

- **014 - Could not open work file**

  The compiler was unable to create or connect to the work file it needs to generate when compiling the source file. Review the system error message associated with this error and correct the problem.

- **021 - Could not open listing file**

  The compiler was unable to create or connect to the listing file the user has requested. Review the system error message associated with this error and correct the problem.

- **103 - Invalid record code (Column 6)**

  The record code in column 6 is invalid. Valid entries are a blank (assuming column 7 is an asterisk (*) to indicate a comment), S, H, and D.

- **104 - Invalid screen name (Columns 7 - 14)**

  An invalid screen name has been processed in a Screen specification. Valid characters are $, _, A - Z, and 0 - 9.

- **105 - Invalid starting line (Columns 17 - 18)**

  The starting line field must be blank, V, or 01 through 48. If a numeric value is specified, it must be less than or equal to the number of lines on a display.

- **106 - Invalid alarm value (Columns 25 - 26)**

  Valid sound alarm values are blank, N, Y, or 01 through 99.

- **107 - Invalid function key value (Column 27)**

  Valid enable function key values are blank, N, Y, or R.

- **108 - Invalid command key value (Column 28)**

  Valid enable command key values are blank, N, Y, or R.

- **109 - Invalid return input code (Column 22)**

  Valid return input code values are N, Y, or blank.

- **110 - Help area defined too large for this screen**

  The number of lines specified in the help area is greater than the number of lines in the screen.

- **111 - Start line exceeds maximum allowed value (Col 17 - 18)**

  The entry in the Screen specification Start Line field is not valid. Valid entries are blanks, "V", and 01 - 48.

- **120 - Invalid code in col. 51 (Lowercase).**
  Must be Y, N, or blank

  Valid lowercase values are Y, N, or blank.

- **123 - Y, N, or blank required in 132 column support field (Column 39)**

  An invalid code has been entered in column 39 of the Screen specification. Valid 132-column support format values are Y, N, or blank.

- **125 - Y, N, or blank required in Right-to-Left field (Column 40)**

  Valid Right-to-Left Display values are Y, N, or blank.

- **127 - Invalid field length. Field size equals 0**

  Valid field length (Columns 15 - 18) values are from 1 through 1919.

- **128 - Invalid fld length. Minimum fld len for signed numeric fld is 2**

  Fields assigned the signed numeric data type require a minimum field length of 2 in order to display the field’s sign. Signed numeric fields are always defined one character larger in the Field Definition (D) specifications than they are in a program so that the screen format utility will allow a space to display the field’s sign.

- **129 - Invalid field length**

  The entry in columns 15 - 18 of the Field Definition specification must be numeric and be a valid field length of 1 - 1919.

- **130 - Invalid line number (Columns 19 - 20)**

  The line number specified is not valid for the screen being defined. The line number cannot exceed 48.

- **131 - Invalid row number (Columns 34 - 35 or 39 - 40)**

  The Upper Left or Lower Right row specified on the Help specification is not valid for the screen being defined.

- **132 - Invalid column number (Columns 21 - 22)**

  The column specified is not valid for the screen being defined. The maximum column values permitted are 80 for an 80 column screen and D2 for a 132 column screen.

- **133 - Invalid column number (Columns 36 - 37 or 41 - 42)**

  The Upper Left or Lower Right column specified on the Help specification is not valid for the screen being defined.

- **134 - Invalid input allowed (Column 26)**

  Valid entries for the Input Allowed field are blank, N, Y.

- **135 - Invalid DUP key allowed (Column 34)**

  Valid entries for the DUP Key Allowed field are blank, N, or Y.

- **136 - Invalid field exit allowed (Column 35)**

  Valid entries for the Field Exit Allowed field are blank, N, or Y.

- **137 - Invalid auto record advance allowed (Column 36)**

  Valid entries for the Auto Record Advance Allowed field are blank, N, or Y.

- **138 - Invalid check digit code (Column 30)**

  Valid entries for the Check Digit Code are blank, E, or T.

- **139 - Current field overlays a previous field in this screen**

  Fields defined within a screen cannot overlay each other.

- **140 - Field extends beyond end of screen**

  The specified field extends beyond the end of the screen. Fields must be defined to fit within the boundaries of a screen.

- **200 - Illegal character in field name**

  The field name in columns 7 - 14 contains an invalid character. Valid characters are $, _, A - Z, and 0 - 9.

- **201 - EOF on input while looking for continuation**

  The compile came to the end of the source file while processing a continuation from a Field Definition specification (X in column 80).

- **202 - Illegal combination of characters in constant**

  The compiler was unable to find the terminator in a constant field. Valid terminators are \ , #, @, ', and a blank.

- **203 - Illegal code (Column 56)**

  Valid constant type values are blank, C, or M.

- **204 - Illegal value in columns 57 - 79 for code 'M'**

  If 'M' is specified for constant type (Column 56), columns 57 - 60 must contain the 4-digit message number and columns 61 - 62 must contain the 2-character message member identifier (U1, U2, S1, S2). If a message member identifier is not specified, a U1 will be assumed.

- **250 - Invalid character in col. 80 (Continuation) must be X or blank**

  Valid continuation character (Column 80) values are X or blank.

- **252 - Invalid character(s) in output field (Columns 23 - 24)**

  Valid output data values are blank, N, Y, or 01 through 99.

- **300 - More than 300 data fields exist for one screen**

  The compiler limit of 300 data fields per screen has been exceeded.

- **302 - Invalid field protect (Columns 37 - 38)**

  Valid protect field values are blank, N, Y, or 01 through 99.

- **303 - Invalid erase input fields (Columns 31 - 32)**

  Valid erase input fields values are blank, N, Y, or 01 through 99.

- **304 - Invalid override (Columns 33 - 34)**

  Valid override fields values are blank, N, Y, or 01 through 99.

- **305 - Invalid suppress input (Columns 35 - 36)**

  Valid suppress input values are blank, N, Y, or 01 through 99.

- **306 - Invalid blink field (Columns 41 - 42)**

  Valid blink field values are blank, N, Y, or 01 through 99.

- **307 - Invalid position cursor field (Columns 32 - 33)**

  Valid position cursor values are blank, N, Y, or 01 through 99.

- **308 - Invalid no display (Columns 43 - 44)**

  Valid nondisplay values are blank, N, Y, or 01 through 99.

- **309 - Invalid underline field (Columns 47 - 48)**

  Valid underline values are blank, N, Y, or 01 through 99.

- **310 - Invalid reverse image field (Columns 45 - 46)**

  Valid reverse image values are blank, N, Y, or 01 through 99.

- **311 - Invalid high intensity field (Columns 39 - 40)**

  Valid high intensity values are blank, N, Y, or 01 through 99.

- **312 - Position cursor field already set in this screen (Col 32 - 33)**

  The position cursor field in colums 32 - 33 has already been defined with a "Y" entry earlier in this screen specification. Remove one of the position cursor entries to eliminate the error.

- **500 - Illegal value for suppress selection indicator (Columns 44 - 45)**

  Valid suppress selection indicator values are blank or 01 through 99.

- **501 - Illegal value for restore application format (Columns 47 - 48)**

  Valid restore application format values are blank, Y, N, or 01 through 99.

- **502 - Illegal value for boundary indicator (Columns 50 - 51)**

  Valid boundary indicator values are blank, N, Y, or 01 through 99.

F Migration RPG Runtime System Halt, Warning, and Error Messages

F.1 Overview

The following sections present the halt, warning, and error messages that can be generated by the Migration RPG Runtime System.

F.2 Halt Messages

Migration RPG halt messages pause a program if it is being run interactively and solicit user input. However, if a halt message is displayed while a program is being run from a batch queue, the program will abort. Halt messages are triggered by the halt indicators, H1 - H9, and by special circumstances detected by the Migration RPG Runtime System.

- **RTS-HLT020 - Enter C, E or K (Continue, Exit or Kill):**

  A HALT indicator has been turned on and detected in interactive mode. Respond to the prompt to continue (C - ignore the halt indicator and continue processing), exit (E - exit the program normally), or kill (K - abort the program immediately) the execution of the program. If the program is running in a batch queue, it will abort.

- **RTS-HLT021 - Invalid response, enter C, E or K (Continue, Exit or Kill):**

  A HALT indicator has been turned on and detected in interactive mode. The response to the HALT message was invalid. Enter a C, E, or K to continue (C - ignore the halt indicator and continue processing), exit (E - exit the program normally), or kill (K - abort the program immediately) the execution of the program.

- **RTS-HLT100 - Insufficient disk space to write rec. Enter A to Abort, R to Retry:**

  An attempt to write a record to a file has failed due to insufficient disk space. Check the amount of free space on the disk. If the disk is full, free up space and enter R to retry. If the disk is not full, analyze the disk to determine the source of the problem.

- **RTS-HLT1001 - Unidentified record type**

  A record type has been encountered which does not match any of the record types defined in Input (I) specifications. If the program is being run interactively, the user is given the choice to Continue, Exit, or Kill. If the program is running in a batch queue, it will abort.

- **RTS-HLT1002 - Error occurred during an I/O operation to a SPECIAL device file**

  The subroutine which supports the SPECIAL device file has returned an error status to the RPG program. If the program is being run interactively, the user is given the choice to Continue, Exit, or Kill. If the program is running in a batch queue, it will abort.

F.3 Runtime System Warning Messages

Migration RPG runtime warning messages do not impact the execution of the RPG program which generates them. When an RPG program which has generated a warning message terminates, it will return a status of 0, indicating that one or more a warnings were generated by the program. The return status can be trapped in the local symbol $SEVERITY and used to condition the actions of the calling DCL procedure.

- **WARNING RTS-001 - Unable to access System Level 1 MIC Message file (RPGSYSLV1)**

  The program was unable to access the System Level 1 MIC Message file, which is referenced by the logical name RPGSYSLV1. Ensure that the MIC message file actually exists, that it is accessible to the user, and that the RPGSYSLV1 logical name has been properly defined.

- **WARNING RTS-002 - Unable to access System Level 2 MIC Message file (RPGSYSLV2)**

  The program was unable to access the System Level 2 MIC Message file, which is referenced by the logical name RPGSYSLV2. Ensure that the MIC message file actually exists, that it is accessible to the user, and that the RPGSYSLV2 logical name has been properly defined.

- **WARNING RTS-004 - Unable to access User Level 1 MIC Message file (RPGUSRLV1)**

  The program was unable to access the User Level 1 MIC Message file, which is referenced by the logical name RPGUSRLV1. Ensure that the MIC message file actually exists, that it is accessible to the user, and that the RPGUSRLV1 logical name has been properly defined.

- **WARNING RTS-003 - Unable to access User Level 2 MIC Message file (RPGUSRLV2)**

  The program was unable to access the User Level 2 MIC Message file, which is referenced by the logical name RPGUSRLV2. Ensure that the MIC message file actually exists, that it is accessible to the user, and that the RPGUSRLV2 logical name has been properly defined.

- **RTS-100 - Invalid numeric data - Enter 0 - 9 only**

  An attempt was made to enter non-numeric data into a numeric field via the KEYBORD device.

- **RTS-910 - Record currently locked by another user in file *filename***

  The record in the indicated file is locked by another The program will wait about 10 seconds and then try to access the record again. The program will continue to try to access the record every 10 seconds until it is successful or five minutes elapse. If the program cannot access the record after five minutes, it will abort with a fatal error.

- **RTS-911 - Will try Read again in ten seconds**

  This message is display after every unsuccessful read to a locked record. See the RTS-910 warning message for an explanation of locked record processing.

F.4 Runtime System Error Messages

Migration RPG runtime error messages are fatal, aborting the RPG program which generates them. The aborted RPG program will return a status of 2, indicating a termination caused by a fatal error. The return status can be trapped in the local symbol $SEVERITY and used to condition the actions of the calling DCL procedure.

- **RTS-010 - Error on Open - Indexed files cannot be opened for relative access under RMS**

  The OpenVMS Record Management Services (RMS) do not support relative access to indexed files.

- **RTS-020 - Error on Open - Key specified in File spec does not match key returned by RMS**

  The key described in the File (F) specification does not match the Record Management Services (RMS) key description. Review the File Definition Language (FDL) file setup and the File (F) specification entry in the program (key position, length, and key number) to locate the discrepancy in the file definition.

- **RTS-030 - MIC Message file not found**

  A SET or KEY opcode specifying a MIC Message member to be displayed has been encountered. However, the message file cannot be opened or a physical channel to the message file cannot be established. Ensure that the file exists and is accessible to the users.

- **RTS-050 - More than one record of a type in group**

  Column 17 of the Input (I) specification contains a 1 to designate that the file will contain only one record of that type in the group specified by the sequence checking in columns 15 - 16. More than one record of that type has been encountered in the file.

- **RTS-060 - Missing non-optional record type**

  Sequence checking has been specified and column 18 of the Input (I) specification contains a blank to designate a mandatory record type. A record of that type was not found in the proper sequence.

- **RTS-070 - Unable to establish input channel to CRT using process logical SYS$COMMAND**

  The program was unable to establish an input channel to the CRT device using the process logical SYS$COMMAND. Check the definition of SYS$COMMAND and ensure that it defines a valid device.

- **RTS-072 - Unable to establish output channel to CRT using process logical SYS$OUTPUT**

  The program was unable to establish an output channel to the CRT device using the process logical SYS$OUTPUT. Check the definition of SYS$OUTPUT and ensure that it defines a valid device.

- **RTS-110 - *message-number* - Invalid MIC Message Indicator**

  A SET or KEY opcode specifying a message member to be displayed has been encountered. However, the message member cannot be found within the message file.

- **RTS-120 - Unable to Open *filename***

  The specified file cannot be opened. Review the associated the system error and correct the problem.

- **RTS-122 - Unable to Create *filename***

  The specified output file cannot be created. Review the associated the system error and correct the problem.

- **RTS-124 - Unable to Create if not found *filename***

  The specified file cannot be found or an attempt to create the file has failed. Review the associated system error and correct the problem.

- **RTS-126 - Unable to Connect to *filename***

  The physical channel to the specified file cannot be established. Review the associated the system error and correct the problem.

- **RTS-130 - Error on Read of *filename***

  No error indicator was specified in columns 56 - 57 of the Calculation (C) specification on a READ or CHAIN operation. An unexpected error occurred while reading the file. Review the associated the system error and correct the problem.

- **RTS-132 - Error on READP of *filename***

  No error indicator was specified in columns 56 - 57 of the Calculation (C) specification on a READP operation. An unexpected error occurred while reading the file. Review the associated the system error and correct the problem.

- **RTS-140 - Error on Write to *filename***

  An error has occurred while writing to the specified file. Review the associated the system error and correct the problem.

- **RTS-142 - Unable to Delete record from *filename***

  An error has occurred while attempting to delete a record from the specified file. Review the associated the system error and correct the problem.

- **RTS-150 - Error on Update of *filename***

  An error has occurred while attempting to update a record in the specified file. Review the associated the system error and correct the problem.

- **RTS-160 - File sequence error in *filename***

  Matching record has been specified for the file. The file does not contain records in the proper ascending or descending sequence.

- **RTS-200 - Array/Table length exceeded by table file *filename***

  The number of elements in the pre-execution time array/table is greater than the array/table size defined in the program.

- **RTS-220 - Array/Table "*array-name*" has sequence error in data**

  The pre-execution time array/table has been defined with an ascending or descending sequence specified. The data contained within the array/table does not follow the specified sequence.

- **RTS-230 - Index less than or equal zero for *array-name***

  The index specified is out of the bounds of the defined size of the array.

- **RTS-231 - Index exceeds size of array for *array-name***

  The index specified is out of the bounds of the defined size of the array.

- **RTS-240 - Missing non-optional record type**

  Sequence checking has been specified and column 18 of the Input (I) specification contains a blank to designate a mandatory record type. A record of that type was not found in the proper sequence.

- **RTS-500 - WORKSTN Screen not found> *screen-name***

  The screen format to be output cannot be found in the screen (.FRM) file.

- **RTS-510 - Attempted to read or write beyond end of screen *****

  An attempt has been made to read or write beyond the last defined row of a screen. The maximum number of rows on a screen is 48.

- **RTS-520 - Help screen format *format-name* does not exist *****

  The help screen format to be output cannot be found in the screen format file.

- **RTS-530 - Starting row exceeds screen size**

  The starting row of a screen cannot be greater than 48.

G FDL File Creation Utilities

G.1 Overview

This appendix describes the FDL File Creation Utilities provided with MSI's Migration RPG Compiler and OpenVMS S/3X Conversion Tool Kits. These utilities are made available on an unsupported basis to help automate the file creation process.

The FDL File Creation Utilities can be modified by the user to meet specific user needs. It is recommended that the procedures be copied or renamed to a user-defined name if they require user modification. This will prevent them from being overwritten by future updates to the utilities.

The purpose of the FDL File Creation Utilities is to make the creation of file definitions for indexed, relative, and sequential files a simple process. By entering a limited amount of data, an FDL file is created for use by the OpenVMS Record Management Services (RMS). See the *Guide to OpenVMS File Applications* for more information on designing files and defining specifications for data files.

G.2 Operation

The FDL File Creation Utilities include:

- **IDXFDL** - Creates an FDL description file for a single-keyed, fixed-length, indexed file.
- **RELFDL** - Creates an FDL description file for a fixed-length, relative file.
- **SEQFDL** - Creates an FDL description file for a fixed-length, sequential file.

The utilities are located in the **S3X$RPG** directory in the Migration RPG Compiler Kit and the **S3X$TOOLS** directory in the OpenVMS S/3X Conversion Tools Kit. The FDL File Creation Utilities share the following features:

— If the utility is invoked and no parameters are included on the command line, prompts will be displayed for the missing data.

— Parameters are positional and must be entered in their proper order. If a null parameter ("") is entered, a prompt will be displayed for the missing data.

— No editing of the data entered occurs. It is important to review the parameters entered when the utility displays them. If they are not correct, they may be re-entered by entering "N" at the verification prompt.

— To exit the utility, select one of the following options:

Y	Normal end-of-job exit; .FDL file created.
N	To re-enter data at display prompts.
CTRL/Z	Normal end-of-job exit; .FDL file not created.

— When the appropriate data has been entered and the utility has been exited by entering "Y" at the verification prompt, an output file will be created. The output file will have the name of the data file which was entered by the user and will have an .FDL file type.

— A final review of the newly created FDL file is recommended. The file may be viewed by entering the DCL "TYPE" command. Because the .FDL file is a text file, it may also be viewed or modified by invoking the FDL editor (EDIT/FDL filename.FDL) or one of the OpenVMS text editors.

IDXFDL

FORMAT

IDXFDL *filename record_size key_start_position - key_length*

NOTE

The IDXFDL Utility parameters are position-dependent. They must be specified in the order indicated above if they are entered on the command line. If they are not entered on the command line when the utility is invoked, a prompt will be displayed for each missing parameter.

PARAMETERS

filename

Name of the data file to be defined. No file type should be specified. The IDXFDL Utility will automatically append an .FDL file type to the specified data file name.

record_size

Defines fixed-length record size in bytes.

key_start_position

Defines the primary key starting position within the record. The key starting position should be entered relative to 1. The IDXFDL Utility will automatically adjust this entry in the FDL file definition to make it relative to 0, which follows the RMS standards. The primary key will be defined as a character string type by default. To change the default key type, modify the .FDL file by using any of the OpenVMS text editors or by invoking the FDL editor (EDIT/FDL filename.FDL).

key_length

Defines the length of the key field in bytes.

Example G–1 IDXFDL Command

```
$ IDXFDL IDXMASTR 256 1 10
```

This example invokes the IDXFDL File Creation Utility. The file IDXMASTR.FDL will be created and will contain the file definition for the indexed data file IDXMASTR.DAT. The fixed-length record size will be defined as 256 bytes. A primary key starting in position 1 with a length of 10 bytes will be defined within the IDXMASTR.FDL file definition. The primary key will be defined as a character string type by default.

RELFDL

FORMAT

RELFDL *filename record_size*

NOTE

The RELFDL Utility parameters are position-dependent. They must be specified in the order indicated above if they are entered on the command line. If they are not entered on the command line when the utility is invoked, a prompt will be displayed for each missing parameter.

PARAMETERS

filename

Name of the data file to be defined. No file type should be specified. The RELFDL Utility will automatically append an .FDL file type to the specified data file name.

record_size

Defines fixed-length record size in bytes.

Example G–2 RELFDL Command

```
$ RELFDL RELCUST 80
```

This example invokes the RELFDL File Creation Utility. The file RELCUST.FDL will be created and will contain the file definition for the relative data file RELCUST.DAT. The fixed-length record size will be defined as 80 bytes.

NOTE: By default, relative files are created with no records under OpenVMS.

SEQFDL

FORMAT **SEQFDL** *filename record_size*

NOTE

The SEQFDL Utility parameters are position-dependent. They must be specified in the order indicated above if they are entered on the command line. If they are not entered on the command line when the utility is invoked, a prompt will be displayed for each missing parameter.

PARAMETERS

filename
Name of the data file to be defined. No file type should be specified. The SEQFDL Utility will automatically append an .FDL file type to the specified data file name.

record_size
Defines fixed-length record size in bytes.

Example G–3 SEQFDL Command

```
$ SEQFDL SEQMASTR 256
```

This example invokes the SEQFDL File Creation Utility. The file SEQMASTR.FDL will be created and will contain the file definition for the sequential data file SEQMASTR.DAT. The fixed-length record size will be defined as 256 bytes.

H Migration RPG Line Mode Editor (RPGEDT)

H.1 Overview

RPGEDT is a line mode editor which has been replaced by the full screen RED editor. It has been retained in MSI's Migration RPG Compiler Kit for those users that like using it. The RPGEDT editor is no longer a supported product and will not receive any further enhancements or bug fixes.

RPGEDT is an interactive text editor designed specifically for the entry and manipulation of RPG source code. Through its features, the user may move, copy, scan, delete, replace, enter, or update RPG programs. The editor also has the ability to display, scan, and copy text from another program into the program being edited. The entry and updating of RPG source code is simplified by the editor's use of formatted screens for each type of RPG specification, complete with cursor movement from field to field (forward and backward) and the right-justification of numeric fields.

H.2 Special Keys

To simplify the use of the editor, a number of keys have been defined to have special meaning. These keys are discussed in the remainder of this chapter.

H.2.1 Command Keys

Command Keys are used to change modes of operation. A Command Key consists of two keys that are pressed in conjunction with one another: the <PF1> key to alert the program that a command is to follow, and a key from the upper row of the keyboard or the keypad to identify the command.

Table H–1 RPGEDT Command Key Definitions

Command Key	Keystroke(s)	Definition
CMD / 1	PF1 + 1	MOVE Mode
CMD / 2	PF1 + 2	COPY Mode
CMD / 3	PF1 + 3	SCAN Mode
CMD / 4	PF1 + 4	DELETE Mode
CMD / 5	PF1 + 5	ENTER/UPDATE Mode
CMD / 6	PF1 + 6	REPLACE Mode
CMD / 7	PF1 + 7	EOJ Mode
CMD / 8	PF1 + 8	SELECT FORMAT Mode
CMD / 9	PF1 + 9	DISPLAY Mode
CMD / 10	PF1 + 0	INCLUDE Mode
CMD / 12	PF1 + =	Exits INCLUDE Mode

H.2.2 FORWARD ENTER

The FORWARD ENTER key is only valid for the ENTER/UPDATE Mode. It is defined as the <PF4> key. Pressing the <PF4> key causes the record currently being manipulated to be placed into the work file. The updated record is displayed at the top of the screen as the previous record. If records are being updated, the next sequential record from the work file is displayed. If records are being entered, a clear format of the same type as the last record entered is displayed and the internal line number is incremented by one.

H.2.3 REVERSE ENTER

The REVERSE ENTER key is only valid for the ENTER/UPDATE Mode. It is defined as the <PF3> key. If records are being entered, pressing the <PF3> key will cause the identical action that pressing the FORWARD ENTER key causes when records are being entered. If records are being updated, pressing the <PF3> key will be identical to pressing the FORWARD ENTER key, except that the previous sequential record will be displayed rather than the next sequential record.

H.2.4 RETURN

The <RETURN> key should be used for entry of fields in all modes. It causes the data keyed in a field to be stored and processed by the program. In the ENTER/UPDATE Mode, it has the additional function of causing the cursor to be moved to the next defined tab stop.

H.2.5 TAB

The <TAB> key is only valid in the ENTER/UPDATE Mode. In this mode, it works in the same manner as the <RETURN> key, except that the cursor is moved to the preceding tab stop rather than the next tab stop.

H.2.6 " –>" (RIGHT ARROW) and "<–" (LEFT ARROW)

These keys are only valid for the ENTER/UPDATE Mode. Because of the tab stops defined in these formats and the function of the <RETURN> and <TAB> keys, it is possible that record positions may be skipped. By pressing the RIGHT ARROW key or the LEFT ARROW key, the cursor will be moved one position to the right or left, respectively.

H.3 Line Numbers

The editor functions by storing data in a temporary work file. This data will be the code that is loaded at the beginning of the edit session if an existing program is being updated, plus all data that is entered by the user. An internal line number is assigned to each item stored. This number is not related to the RPG source code sequence number that may appear in columns 1 - 5 of the source code.

This line number is the number by which data is referenced for all functions in all modes. When an existing program is being edited, the line numbers that are assigned will begin at 0001.00 and be incremented by 1.00 until the end of the program (i.e., 0002.00, 0003.00, ...). This means that the maximum program size that can be edited is 9999 statements. In addition, no more than 99 statements can be entered between two existing statements. The user can work around this latter restriction. If more than 99 statements are to be inserted between two existing statements, move the upper line number to a higher line number before starting to enter new lines. See Section H.5.1 for an explanation of the MOVE Mode.

For example:

> If statements are numbered 5.00, 6.00, 7.00, and the user wants to insert 150 statements between 5.00 and 6.00, the user can use the MOVE mode to move 6.00 to 6.99, causing the sequence to become 5.00, 6.99, 7.00. There is now enough space to insert the lines.

H.4 Invoking the Editor

To invoke the editor, log into an account. When the DCL prompt is displayed, enter the command RPGEDT followed by the file specification of the program to be edited. If the file specification is omitted, the editor will prompt the user for the file spec with the prompt RED>. If the file name entered is found, the data is loaded into a temporary work file for editing and a unique internal line number is assigned to each statement. If the file is not found, a message is displayed stating that a new file will be created. (Note that the .RPG file type is not assumed.) For example:

Example H–1 Invoking the RPGEDT Line Editor

```
$ RPGEDT GL107.RPG

     - or -

$ RPGEDT
RED> GL107.RPG
```

H.5 Modes of Operation

Each function that is supported by the editor constitutes a different mode of operation. Each mode is accessed by a command key. Modes can be changed at any time by pressing the appropriate command key. A menu of command keys is always displayed at the bottom of the screen. These command keys and their modes are discussed below.

H.5.1 MOVE Mode - Command Key 1

The MOVE Mode allows text to be moved from one place to another. The user is prompted for the FROM and THROUGH line numbers to identify what is to be moved, and for the TO line number to identify where the data is to be placed. The data is moved to the new location, with each line being displayed and then deleted from the original location. If data is moved to a line number that already exists, the existing line will be replaced by the data being moved.

H.5.2 COPY Mode - Command Key 2

The COPY Mode allows text to be copied from one place to another without deleting it from the original location. The user is prompted for the FROM and THROUGH line numbers to identify what is to be copied, and for the TO line number to identify where the data is to be placed. The data is copied to the new location, with each line that is copied being displayed. If data is copied to a line number that already exists, the existing line will be replaced by the data being copied.

H.5.3 SCAN Mode - Command Key 3

The SCAN Mode allows the user to search for the occurrence of a string of up to 25 characters. The user is prompted for the starting record position in which the scan is to begin in each record and for the string of characters to be located. The file is searched from the beginning. Each match that is located is displayed, one at a time, and the user is prompted as to whether or not the search should continue.

H.5.4 DELETE Mode - Command Key 4

The DELETE Mode deletes records from the work file. The user is prompted for the FROM and TO line numbers to identify the records to be deleted. (Note that this DELETE is inclusive; that is, the TO line and the FROM line are deleted along with all lines in between.) The first and last records of the group are displayed, and the prompt ARE YOU SURE? is displayed. If the user responds N to the prompt, the records are not deleted. If the user responds Y to the prompt, the records are deleted and a count of the number of records deleted is displayed.

H.5.5 ENTER/UPDATE Mode - Command Key 5

This mode is used to enter new lines of code or to update existing lines of code. The user is prompted for the line number to be updated. If that line number is not being used in the work file, an ENTER Mode is assumed and a clear format is displayed, ready for entry. This format may be changed by exiting to the SELECT FORMAT Mode and returning to the ENTER/UPDATE Mode. If the line number is found in the work file, a format is selected based on the record code in column 6, and the record is displayed in that format with an UPDATE Mode being assumed. In the ENTER/UPDATE Mode, the user has access to all the keys defined. The record that is being manipulated is not placed into the work file until either the FORWARD ENTER or REVERSE ENTER key is pressed. In the formats, all numeric fields are highlighted with reverse video. Data keyed in these fields must be numeric and will be right-justified when the RETURN key or the TAB key is pressed.

H.5.6 REPLACE Mode - Command Key 6

The REPLACE Mode allows the user to search for the occurrence of a string. The user is prompted for the starting record position in which the scan is to begin in each record, for the string of characters to be located, and for a replacement string of characters. The file is searched from the beginning to locate the search string. In each record where the search string is found, the replacement character string is inserted in place of the search string.

H.5.7 EOJ Mode - Command Key 7

This mode terminates execution of the editor. The user is prompted to specify whether or not the source file is to be updated. If the user responds N, the program terminates, the work file is deleted, and nothing is output back to disk. If the user responds Y, the work file will be copied back to disk with the same output file specification as the one that was entered at invocation of the editor, but with an incremented version number.

Prior to outputting the work file, the user is asked if serialization and/or a listing is desired. If serialization is selected, the output source file will have a sequence number starting with 10 and incrementing by 10 placed in record positions 1 through 5. Serialization stops at the end of file or when an "**" is found in positions 1 through 3 of the work file (denoting the beginning of a table). If a listing is selected, a DCL PRINT command is spawned to queue a listing of the file to SYS$PRINT.

NOTE: **The responses to the prompts mentioned above must be entered in upper-case only.**

H.5.8 SELECT FORMAT Mode - Command Key 8

This mode is used to select a format in which items are to be displayed in the ENTER/UPDATE Mode. A list of options will be displayed, and the user must enter the number corresponding to the format desired. The user is then returned to the ENTER/UPDATE Mode. Note that the field descriptions are provided for help only.

H.5.9 DISPLAY Mode - Command Key 9

This mode is used to display items from the work file. The user is prompted for the FROM and TO line numbers of the records to be displayed. If no entry is made at the FROM prompt, the editor defaults to the first record of the file. If no entry is made at the TO prompt or the total number of lines to display is greater than 16, the editor will display the first 16 lines of code starting with the FROM line number and then prompt the user to continue.

H.5.10 INCLUDE Mode - Command Key 10

This mode is used to locate data in a program other than the one that is being edited and to copy selected portions from it into the program that is being edited. The user is prompted for the name of the file to be included. If the specified file is not found, a FILE NOT FOUND error message is displayed. This file is loaded into a special temporary work file and has line numbers assigned.

An abbreviated sub-menu is displayed showing these functions and the command keys valid for the INCLUDE Mode:

- Command Key 2 allows data to be copied from the included file into the edited file.
- Command Key 3 allows the included file to be scanned for a character string.
- Command Key 9 allows items to be displayed from the included file.

These modes are identical to those already described, except that in the COPY from the included file, the FROM and THROUGH prompts refer to line numbers in the included file, and the TO prompt refers to the line number in the edited file where the records are to be placed. No changes are ever made to the included file. Note that Command Key 12 causes the editor to exit the INCLUDE mode and return to the standard menu and the ENTER/UPDATE Mode.

I Migration RPG Report Viewing Utility

I.1 Overview

This appendix describes the report viewing utility supplied with Migration RPG. The View Utility reformats Migration RPG reports for presentation on a terminal screen.

I.2 Creating the View Utility

Use the following steps to create the View Utility:

1 Log into a privileged account.

2 Set default to the S3X$RPG directory.

```
$ SET DEFAULT S3X$RPG:
```

3 Compile and link the RPG program PRINT1.RPG.

```
$ BUILD PRINT1
```

4 Set the PRINT1.EXE image file so it is world executable.

```
$ SET FILE /PROTECTION=W:E PRINT1.EXE
```

5 Create a command symbol for the View Utility. Place the symbol definition in the SYS$MANAGER:SYLOGIN.COM procedure or a similar procedure that users using the View Utility will execute when they log in.

```
$ VIEW :== @S3X$RPG:VIEW.COM
```

I.3 Operation

The View Utility reformats a Migration RPG report file for display on a terminal screen. The utility works as follows:

1 The user calls the View Utility and passes it a report name.

2 The View Utility reads the report file and reformats it for display on a terminal screen. Based on the report width, the View Utility will use one of the following record lengths when displaying the report: 80, 132, 256. Reports wider than 256 characters are truncated.

3 The reformated report is displayed as a read only file using the OpenVMS TPU editor. The report can be reviewed using standard TPU edit commands.

4 A $\boxed{\text{CTRL/Z}}$ or Exit command can be used to exit the View Utility.

VIEW

FORMAT **VIEW** *filename*

PARAMETERS

filename
Name of the report file to be viewed. The report file name can be fully qualified. If only the report name is specified, the utility will search the current default directory, the SYS$LOGIN directory, and the SYS$SCRATCH directory for the file with and without a .LIS extension.

Example I–1 VIEW Command

```
$ VIEW ACCT_BAL
```

This example invokes the VIEW Utility to display the report file ACCT_ BAL. Since the report file name is not fully qualified, the VIEW Utility will search following file specifications in the order presented:

— ACCT_BAL.

— ACCT_BAL.LIS

— SYS$LOGIN:ACCT_BAL.

— SYS$LOGIN:ACCT_BAL.LIS

— SYS$SCRATCH:ACCT_BAL.

— SYS$SCRATCH:ACCT_BAL.LIS

The utility will display the first occurance of the file it finds in the search list.

Glossary of Terms

The following list presents definitions for terms used in this manual. Additional terms may be found in the *OpenVMS Glossary*.

access violation: An exception that takes place after an attempt to reference an address that is either not mapped into virtual memory or not accessible by the current access mode.

account: A character-string name or number that identifies an individual user when the user logs in. Account information tells the system where the user's files are located and the type of access that the user is authorized for when using other files.

Under OpenVMS, the account is a key to the system and a unit of accounting. Each system user, including parts of the system itself, has an account. The system manager creates these accounts and assigns account privileges, quotas, and priorities.

address: A number used by the operating system and application software to identify a storage location in memory.

ADDROUT file: A record address disk file produced by the sort program. It contains addresses of records in a disk file and can be used to process input or update files that are designated as primary or secondary files by the Migration RPG program.

alphanumeric: Alphanumeric data can contain letters, numbers and special characters. Special characters are those characters other than alphabetic or numeric characters.

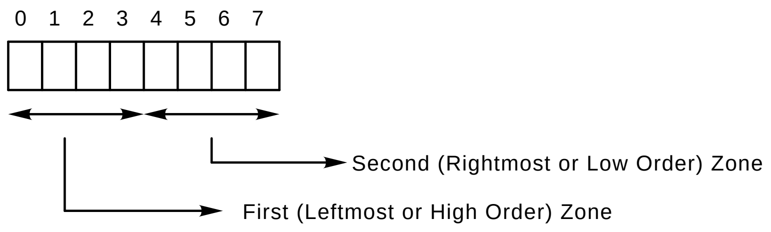
ASCII: American Standard Code for Information Interchange. A set of 8-bit binary numbers representing the alphabet, punctuation, numerals, and other special symbols used in text representation and communications protocol. ASCII is the standard format used in Digital computer systems. See also: *EBCDIC*.

attribute: A particular quality or characteristic of a file.

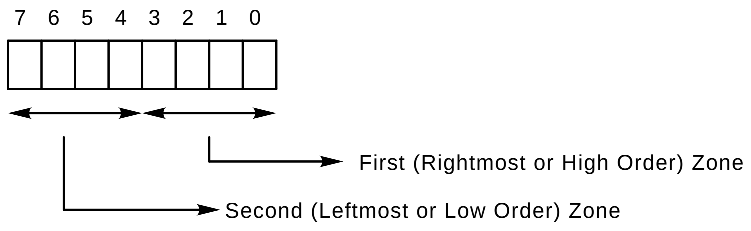
batch processing: The technique of executing a set of computer programs without user interaction or direction during their execution.

binary: Data in binary format is represented in storage in binary digits, that is, as a number to the base 2. (Either 0 or 1.) A binary field usually occupies less storage than a zoned decimal field and sometimes occupies less storage than a packed decimal field.

On IBM systems, the bits are numbered from **left to right** 0 through 7.


0 1 2 3 4 5 6 7
Second (Rightmost or Low Order) Zone
First (Leftmost or High Order) Zone

Under OpenVMS, the bits are numbered from **right to left** 0 through 7.


7 6 5 4 3 2 1 0
First (Rightmost or High Order) Zone
Second (Leftmost or Low Order) Zone

The CVTFILE Utility swaps the high and low zones of binary fields when converting EBCDIC data to ASCII and vice versa.

block: (1) The smallest logically addressable unit of data that a specified device can transfer in an I/O operation (under OpenVMS, usually 512 contiguous bytes for most disk devices). (2) An arbitrary number of contiguous bytes used to store logical records.

bucket: A storage structure of 1 to 32 blocks used for building and processing files of relatives and indexed organization. A bucket contains one or more records or record cells. Buckets are the units of contiguous transfer between OpenVMS Record Management Services (RMS) buffers and the disk.

byte: Eight contiguous bits starting on any addressable boundary.

captive account: A type of OpenVMS account that limits the activities of the user. Typically, the user is restricted to using certain command procedures and commands. For example, the user may not be allowed to use the <CTRL/Y> key.

CATALOG: IBM system command used to list names of files on disk. This is similar to the OpenVMS DCL DIRECTORY command.

chained files: Input, output, or update disk files that use the CHAIN operation code to read records randomly from an indexed or direct file.

character string: A contiguous set of bytes. A character string is identified by two attributes: an address and a length. Its address is the address of the byte containing the first character of the string. Subsequent characters are stored in bytes of increasing addresses. The length is the number of characters in the string. See *alphanumeric*.

command: In Digital Command Language (DCL), an instruction, generally an English word, entered by the user at a terminal or included in a command procedure. A command requests the software monitoring a terminal or reading a command procedure to perform an activity. For example, entering the COPY command requests the system to copy the contents of one file into another file. See also: *DCL*.

command interpreter: OpenVMS system program that interprets the DCL commands entered by the user.

command procedure: A file containing commands and data that the command interpreter can accept in lieu of the user entering the commands individually on a terminal. Thus, command procedures provide a means of automatically passing commands to the operating system. In addition, they permit users to employ such programming techniques as loops, counters, labels, and symbol substitution to set up elaborate command sequences that can be altered through user interaction. Command procedures can also be submitted to the system for processing as batch jobs.

CVTCCP: OpenVMS S/3X Conversion Tools utility used to translate a CCP program and screen to an interactive RPG program and screen.

CVTDFU: OpenVMS S/3X Conversion Tools utility used to convert IBM DFU definition files to a format executable under OpenVMS.

CVTFILE: OpenVMS S/3X Conversion Tools utility used to convert files between EBCDIC and ASCII data format and to rebuild logical record structures.

CVTMENU: OpenVMS S/3X Conversion Tools utility used to convert IBM System/34 menu definitions to DCL command procedures.

CVTOCL: OpenVMS S/3X Conversion Tools utility used to convert IBM OCL procedures to OpenVMS DCL command procedures.

CVTS34LIB: OpenVMS S/3X Conversion Tools utility used to convert IBM System/34 Libraries to ASCII format and expand the members into individual source files.

CVTS36LIB: OpenVMS S/3X Conversion Tools utility used to convert IBM System/36 Libraries to ASCII format and expand the members into individual source files.

CVTSCAN: OpenVMS S/3X Conversion Tools utility used to identify expanded library source members.

CVTTAPE: OpenVMS S/3X Conversion Tools utility used to read 9-track magnetic tapes created on an IBM System/3, System/34, or System/36.

DCL: OpenVMS Digital Command Language. It provides a means of communication between the user and the operating system. DCL is designed for ease of use. Commands are English words.

DECforms: DECforms is an implementation of a Form Interface Management System (FIMS) ANSI/ISO standard prepared by the CODASYL Form Interface Management System Committee.

DECforms integrates text and simple graphics into forms (screens) and menus. Application programs use these forms and menus as user interfaces. DECforms also provides extensive facilities for specifying full control of the user interface within the form rather than in the application program.

default: Value supplied by the system when a user does not specify a required command parameter or qualifier.

device: The general name given for any peripheral connected to the OpenVMS system that is capable of receiving, storing, or transmitting data.

DFU: Data File Utility used to create, maintain and display data files.

DFUEDT: OpenVMS S/3X Conversion Tools utility to create or modify Data file format (.DFN) files.

direct file: See *relative file*.

directory: Under OpenVMS, a file that catalogs a set of files stored on disk or tape. The directory includes the name, type, and version number of each file in the set.

directory name: The field in a file specification that identifies the directory in which a file is listed.

echo: The display of a character, either on the screen or on hard copy, that is typed on a terminal keyboard. Typing on the terminal sends input to the computer. Echoing is the process of receiving output from the computer. In the no-echo mode, characters typed on the terminal are not displayed. No-echo mode can be used to preserve confidentiality, such as when entering passwords.

editor: Program that creates or modifies files. In the OpenVMS system, the default system editor is EVE, which is an interactive editor. See also: *RED* and *DFUEDT*.

EBCDIC: Extended Binary Coded Decimal Interchange Code. EBCDIC is the binary number (code) assigned to each printing character and several non-printing characters used to control printers and communication devices. EBCDIC is the standard format used by most IBM computer systems.

EXG: OpenVMS S/3X Conversion Tools utility used to read 8.5", single-sided, single-density, 128-byte sector diskettes created on an IBM system using the Digital Equipment Corporation RX02 diskette reader.

File Definition Language (FDL): A special-purpose language used to describe the organization of data files under OpenVMS. These specifications are written in text files called FDL files; they are then used by OpenVMS Record Management Services (RMS) utilities and library routines to create and modify data file definitions.

file name: The field containing a name for a file that precedes a file type in a file specification.

file specification: A unique name for a file on mass storage media. It identifies the node, the device, the directory name, the file name, the file type and the version number under which a file is stored.

file type: Or file extension, generally describes the nature of a file, or how it is used. For example:

- name.DAT - indicates a data file.
- name.RPG - indicates an RPG source program.

In the OpenVMS environment, file types are very meaningful and may be required parameters for file manipulation commands.

index: The structure that allows retrieval of records by key value in an indexed file.

indexed file: The records of an indexed file are arranged randomly and accessed through one or more indexes. An index contains a portion of each record called a key; the keys are arranged in sequence from lowest to highest (by binary, numeric, or ASCII value depending upon data type); one key is called the primary key.

An index allows a program to process required records by referring to the key of the record. For example, if you have an indexed file containing customer number, amount ordered, and balance due, your program can use the customer number as the key to find a record for a particular customer without having to read any other records.

You can also read indexed files sequentially by specifying an index. Then, records are read in ascending sequence according to the key values for that index, starting with the current record.

Indexed files require more space since, in addition to the data, the index structures must be stored.

indicator: An internal switch that communicates a condition between parts of a program or procedure.

initialize: To prepare for use. For example, to initialize a diskette or magnetic tape.

key: The user-defined length and location (field) within an indexed file that determines the retrieval order of records.

lexical functions: A set of OpenVMS functions that return information about character strings, attributes of the current process, system information, and file information.

library: On IBM System/3X systems, a defined area on disk containing programs, procedures and related files, with the exception of DATA files. A library is similar to an OpenVMS directory.

library member: See *member*.

LDA: Local Data Area. On the IBM system, this is a 512-byte area in memory that can be used to pass information between programs and/or OCL procedures. A separate LDA exists for each command display station. Submitted batch jobs reference the same LDA as the submitting procedure.

Under OpenVMS, the LDA exists as a one-record 512-byte data file on disk. It is also used to passed information between programs and/or DCL procedures. The LDA is automatically created by the RPGINSTAL command procedure, which is normally invoked during the login process. A separate LDA is created for each user and submitted batch job.

logical name: A user-specified name that replaces any portion or all of a file specification or any character string. For example: a command or program can refer to a file by a logical name, which, when executed, translates to its defined equivalence. Another example: a directory name can be replaced by a shorter logical name. All future references to that directory name can be made by typing in the logical name equivalent. Logical names are a powerful feature of OpenVMS and can be used to make applications device independent.

logging in: The identification of a user to the operating system. When users log in, they type an account name and password in response to the appropriate prompts from the system. If the name and password match an account on the system, a user process is created and the user is granted access to the system.

logging out: Entering the DCL command LOGOUT, which informs the operating system that the user has finished a session and that the user's process should be terminated.

mass storage device: An input/output device on which data and other types of files are stored. Typical mass-storage devices include disks, magnetic tapes, floppy disks, and optical disks.

member: On IBM systems, member refers to one of the programs or files in a library.

On OpenVMS systems, member generally refers to the second number in the User Identification Code, which uniquely identifies the user.

module: Software: A discrete program unit, such as a source module, object module, or image module.

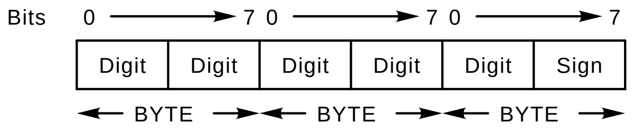
network: A collection of interconnected individual computer systems.

node: An individual system on a network.

OCL: IBM Operation Control Language. A procedure language used on System/3X systems to convey the user's requirements to the Operating System. This is similar to DCL on the OpenVMS system.

overpunched negative numbers: See *zoned decimal*.

packed decimal: Packed decimal format means that each byte (except for the low-order byte) can contain two digits. Each byte, except the low-order byte, is divided into two 4-bit digit portions. The rightmost portion of the low-order byte contains the sign (plus or minus) for that field. The packed decimal format looks like this:



On the IBM system, Positive Sign: hex F
Negative Sign: hex D

When a packed decimal field is converted to a zoned decimal format, the zoned decimal field always contains an odd number of bytes. The following table shows the packed decimal equivalents for zoned decimal fields up to 15 bytes long.

Table 1 Packed decimal equivalents for zoned decimal fields

Zoned Decimal length in bytes	Packed Decimal length in bytes
15	8
14	8
13	7
12	7
11	6
10	6
9	5
8	5
7	4
6	4
5	3
4	3
3	2
2	2
1	1

password: A character string that users provide at login time to validate their identity and as a form of proof of their authorization to access the account.

primary key: The mandatory key within the data records of an indexed file; used by OpenVMS Record Management Services (RMS) to determine the placement of records within the file and to build the primary index.

record: A set of related data that a program treats as a unit.

RED: A TPU-based, full screen text editor designed specifically for the entry and manipulation of RPG source code. RED is part of MSI's Migration RPG Compiler Kit.

relative file: A fixed-length disk file in which records are referenced by the relative record number. The relative record number specifies the location of a record (or its offset) in relation to the beginning of the file.

RMS: Record Management Services are a set of OpenVMS operating system services that provide access to files and records within files. RMS is the data management subsystem of the OpenVMS operating system. Most standard input/output in OpenVMS goes through the RMS layer. RMS recognizes sequential, direct, and indexed files.

RMS services include:

- creating, opening and closing files,
- reading, writing, updating and deleting records,
- extending and deleting files.

sequential file: The records of a sequential file are arranged in the order in which they were created. Records must be read from the file in this order. The file must be rewritten (another version of the file must be created) in order to update it.

SSP: System Support Program. The system software that manages the programs and devices on IBM System/3X computers.

string: A sequence of characters. When using an editor to search for a word or phrase, the user enters a string as the search argument.

UIC: User Identification Code. The account number by which the user is known to an OpenVMS system.

variable: A logical address in memory used to hold data.

VAX: Virtual Address Extension.

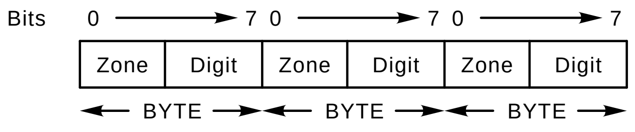
version number: The field following the file type in an OpenVMS file specification. It is used to identify the revision level of the file.

VFC: Variable file with fixed control. A file format in which records of variable length contain an additional fixed-length control area. The control area may be used to contain file line numbers and print format controls.

OpenVMS: Open Virtual Memory System. The operating system used by computers running the OpenVMS operating system.

wildcard character: A nonalphanumeric character, such as an asterisk or a percent sign, that is used within, or in place of, a file name, file type, directory name, or version number in a file specification to indicate "all" for the given field or position(s) within the field.

zoned decimal: For data in zoned decimal format, each byte of storage represents a single character. Each byte of storage is divided into two parts: a 4-bit zone portion and a 4-bit digit portion. The zone portion of the rightmost byte contains the representation of the sign (+ or -). Zoned decimal format looks like:



On IBM System/3X systems, a positive sign is represented by hexadecimal F (1111) or
C (1100) and a negative sign is represented by hexadecimal D (1101).

When a user enters data via an RPG program, overpunched numeric encoding is used to retain the value of the field's sign. That is, the last character of the zoned field is converted and stored as follows:

Table 2 Overpunched Numbers

Digit	Character	EBCDIC hex	EBCDIC binary	ASCII hex	ASCII binary
-0	}	D0	1101 0000	7D	0111 1101
-1	J	D1	1101 0001	4A	0100 1010
-2	K	D2	1101 0010	4B	0100 1011
-3	L	D3	1101 0011	4C	0100 1100
-4	M	D4	1101 0100	4D	0100 1101
-5	N	D5	1101 0101	4E	0100 1110
-6	O	D6	1101 0110	4F	0100 1111
-7	P	D7	1101 0111	50	0101 0000
-8	Q	D8	1101 1000	51	0101 0001
-9	R	D9	1101 1001	52	0101 0010

Index