RM/COBOL Development System README

Version 10.01 for 32-Bit Windows

Contents

This document contains the following sections:

Changes and Enhancements

The changes and enhancements in RM/COBOL Version 10 are covered in this readme file. This readme file, when accompanied by the first edition base documents, fully describes version 10 of RM/COBOL.

Technical Notes

Beginning with version 7.5, the runtime system now creates new indexed files with a minimum block size of 1024 bytes. If you find it necessary to recover an indexed file using a template file (-k option), you should ensure that the indexed file to be recovered and the template file have the same block size. A template file with the wrong block size can cause the loss of a large percentage of the recoverable records in your file. See Appendix G of the RM/COBOL User's Guide, First Edition, for more information.

Documentation Changes

CodeBridge User's Guide, First Edition

The CodeBridge User's Guide, First Edition, is the base document for the CodeBridge component in RM/COBOL version 10 and covers the CodeBridge cross-language calling features at the time of the version 9 release. The CodeBridge User's Guide supplement document for version 10 is this readme file; the changes since version 9 are described below, followed by specific corrections to the CodeBridge User's Guide, First Edition.

The CodeBridge Library in version 10 was enhanced to eliminate some 16-bit limits inherent in its coding. The version 10 CodeBridge Library is built into the version 10 RM/COBOL runtime system and these limits are relaxed in the version 10 runtime. The limits were not discussed in the CodeBridge User's Guide, First Edition. The limits were, for the most part, closely correlated with existing limits for

the OCCURS clause in versions 9 and earlier of RM/COBOL. The CodeBridge Library needed to conform to the removal of such limits in the version 10 runtime system.

The rtcallbk.h file, used with CodeBridge during support module compilation, was enhanced in version 10 to properly export the entry points of a module built with CodeBridge Builder for Windows. This eliminates a need for using definitions (.def) files on Windows. Definitions files are not discussed in the CodeBridge User's Guide, First Edition, because it was believed they were not needed. However, there was an error in the rtcallback.h file that prevented the automatic export of a CodeBridge module's entry points (RM_EntryPoints, RM_EnumEntryPoints, RM_AddOn...), which in some cases necessitated an understanding of definitions files on Windows.

A GetTerminationInfo library function has been added so that non-COBOL programs can obtain information about why a program is terminating.

CodeBridge User's Guide, First Edition, Page 165, Appendix F, CodeBridge Library Functions, GetCallerInfo

A version 2 of the CALLER_INFO structure has been defined. RM/COBOL version 10 produces version 2 of the CALLER_INFO structure and places a 2 in the Version field of the structure. The version 2 CALLER_INFO structure is the same as the version 1 structure except for the addition of a new field at the end named ProgramID. (For a complete layout of the CALLER_INFO structure, see the rtcallbk.h header file provided with RM/COBOL version 10.) The ProgramID field is a pointer to a constant character string that contains the program-name specified in the PROGRAM-ID paragraph of the calling program (truncated to 30 characters in length, if necessary). The ProgramName field from version 1 of the structure points to the program-name of the calling program, but that program-name is the name by which the calling program was called and might not match the program-name specified in the PROGRAM-ID paragraph when RM/COBOL's "call by filename" calling method is used.

CodeBridge User's Guide, First Edition, Page 165, Appendix F, CodeBridge Library Functions, GetTerminationInfo (new function)

GetTerminationInfo

GetTerminationInfo obtains information about runtime termination. It is intended to be called from the COBOL-callable entry point, RM_AddOnTerminate. This function returns a pointer to a structure that contains the information about runtime termination.

Calling Sequence

```
CALLER_INFO* _rmdll_RtCall->pGetTerminationInfo();
```

The function has no arguments.

The structure pointed to by the return value is described by a type definition in the supplied header file rtcallbk.h, which is included by the supplied header file cbridge.h. For reference, the structure is as follows:

```
typedef struct tagTerminationInfo
{ /* version 1 and later */
  BIT16 Version;         /* structure version;
                            1 is first version,
```

```
                              2 is second version, ...  */
  BIT16 State;              /* runtime state when error
                              occurred; see #define TIS_...
                              below */
  BIT16 ReturnCode;         /* runtime return code */
  const char *ErrorCode;    /* error code string pointer */
  const char *TermMsgs;     /* termination messages string
                              pointer */
} TERMINATION_INFO;
```

The values in the State field of the TERMINATION_INFO structure have the following meanings (as defined in rtcallbk.h):

```
#define TIS_UNKNOWN      0
#define TIS_INITIALIZING 1
#define TIS_RUNNING      2
#define TIS_TERMINATING  3
```

The ErrorCode and TermMsgs string pointers should never be NULL, but they may point to empty strings (first character is zero). If the runtime is terminating normally due to executing a STOP RUN statement in the COBOL program, the ReturnCode will be zero, the ErrorCode string will be empty, and the normal "COBOL STOP RUN … " message will be in TermMsgs. If the runtime is terminating due to an error, the ReturnCode will be non-zero, the ErrorCode string will contain an error code, and the TermMsgs string will contain as much of the runtime error and traceback messages as will fit in about 1000 bytes. Multiple messages in the TermMsgs string are separated with a newline character.

If the GetTerminationInfo callback function is called from other than RM_AddOnTerminate (not recommended because it is not useful), the State and ReturnCode values will be zero and the ErrorCode and TermMsgs strings will be empty.

CodeWatch User's Guide, First Edition

The CodeWatch User's Guide, First Edition, is the base document for the CodeWatch component in RM/COBOL version 10 and covers the CodeWatch development environment at the time of the version 9 release. The CodeWatch User's Guide supplement document for version 10 is this readme file; the changes since version 9 are described below.

The CodeWatch source code editor in version 10 has the ability to show and hide COPY files. By default, COPY files are now hidden, making source code editing and navigation faster. A hidden COPY file is denoted by a [+] symbol in the right margin of the source edit window. The hidden COPY file text can be revealed by clicking on the [+] symbol, and a revealed COPY file block can be hidden by clicking on the [-] symbol in the margin. In addition, the F11 key toggles the block that contains the cursor, and the View menu and the Source Code Editor popup menu contain menu picks to perform these operations, and reveal/hide all COPY file blocks in the current file. Note that Edit | Find will search only visible COPY file blocks. To search all COPY blocks, use View | Copy Files | Expand All Blocks.

The CodeWatch animation window in version 10 allows double-clicking in the rightmost column to set or clear any breakpoints on a line.

The CodeWatch File menu in version 10 contains options to print the current source. The entire source or just a selected portion of the source can be printed.

CodeWatch version 10 displays a tab containing the name of each open file or window at the bottom of the document area. Clicking on the tab for a particular window brings that window to the top. This is called "workbook" mode, because the document area has the appearance of a binder. The previous mode can be selected with the "Settings" tab in the View | Preferences dialog. Sometimes this feature is also referred to as "tabbed edit windows."

CodeWatch version 10 can load large projects and save large source files much faster than previous versions.

CodeWatch version 10 has fixes for numerous minor problems (such as the problem of the system crash caused by double-clicking the Build button and the failure to restore watchpoints with subscripts when loading a saved workspace).

RM/COBOL Language Reference Manual, First Edition

The RM/COBOL Language Reference Manual, First Edition, is the base document for the COBOL language component of RM/COBOL version 10 and covers the RM/COBOL language at the time of the version 9 release. The RM/COBOL Language Reference Manual supplement document for version 10 is this readme file; the changes since version 9 are described below, followed by specific corrections to the RM/COBOL Language Reference Manual, First Edition.

The RM/COBOL language in version 10 has enhancements to the ACCEPT and DISPLAY statements for syntax features common in other COBOL dialects. In particular, the AT line-column phrase is now supported for positioning, where the line-column refers to a four- or six-digit numeric display integer data item or is a four- or six-digit numeric integer literal. The first half of the line-column data item or literal specifies the line and the second half specifies the column where the ACCEPT or DISPLAY operation should occur on the display device. These enhancements are not necessary for writing new RM/COBOL programs, but aid in migration from another COBOL dialect to RM/COBOL. Note that the definition of a line-column data item is the same as for the CURSOR IS data item, so use of the AT line-column syntax allows direct use of the data item specified in a CURSOR IS clause as a positioning operand in an ACCEPT or DISPLAY statement.

The RM/COBOL language in version 10 has relaxed some of the limits from prior versions of RM/COBOL:

A data element subject to an OCCURS clause is not limited to 65280 characters in length.

The fixed-size header of a variable-size group, that is, a group that contains a data item described with the DEPENDING ON phrase of the OCCURS clause, is not limited to 65280 characters in length.

The total size of distinct literals specified in a single program segment is not limited to 65535 characters.

In each of these cases, the limit is now subject only to the 4GB limit addressable in a 32-bit computer when the entire memory needs of the program are considered collectively.

The RM/COBOL compiler in version 10 has been enhanced to warn when an external name is longer than 30 characters in length. RM/COBOL generally supports names up to 240 characters in length during compilation, but external names are truncated to 30 characters in the object program. This truncation

previously occurred silently and might have caused surprises at runtime. External names that match in the first 30 characters would refer to the same external object. (Note that if the warning is undesired, it can be suppressed using compiler configuration with the NO-DIAGNOSTIC keyword of the COMPILER-OPTIONS configuration record.)

The RM/COBOL compiler in version 10 has been fixed to correct a problem of ignoring the first COPY statement in a file copied using a COPY statement that specifies the REPLACING phrase. (This error was introduced in version 9 as part of a fix to correctly diagnose a COPY statement nested within another COPY statement. Nesting a COPY statement within another COPY statement, that is, lexically between the word COPY and the required period separator terminating the COPY statement, is not allowed. However, nesting a COPY statement within a copied file is allowed by RM/COBOL up to five levels of nesting.)

RM/COBOL Language Reference Manual, First Edition, Page 247, Chapter 6, Procedure Division Statements, ACCEPT Statement (Terminal I-O)

The format of the ACCEPT statement includes the AT line-column syntax: AT identifier-11/literal-9.

A new syntax rule states:

identifier-11 (AT) must refer to an unsigned numeric integer display data item of four or six characters in length. literal-9 (AT) must be an unsigned numeric integer literal of four or six characters in length.

A new general rule states:

If the AT identifier-11/literal-9 phrase is specified, then the first half, that is, the first two or three digits, of the operand specifies the line value and the second half specifies the column value. When this format of the phrase is used, it is as if both the LINE and POSITION phrases have been specified.

RM/COBOL Language Reference Manual, First Edition, Page 263, Chapter 6, Procedure Division Statements, ACCEPT Screen-Name Statement

The format of the ACCEPT statement includes the AT line-column syntax: AT identifier-3/integer-3.

A new syntax rule states:

identifier-3 (AT) must refer to an unsigned numeric integer display data item of four or six characters in length. integer-3 (AT) must be an unsigned numeric integer literal of four or six characters in length.

A new general rule states:

If the AT identifier-3/integer-3 phrase is specified, then the first half, that is, the first two or three digits, of the operand specifies the line value and the second half specifies the column value. When this format of the phrase is used, it is as if both the LINE and POSITION phrases have been specified.

RM/COBOL Language Reference Manual, First Edition, Page 293, Chapter 6, Procedure Division Statements, DISPLAY Statement

(Terminal I-O)

The format of the DISPLAY statement includes the AT line-column syntax: AT identifier-7/literal-7.

A new syntax rule states:

identifier-7 (AT) must refer to an unsigned numeric integer display data item of four or six characters in length. literal-7 (AT) must be an unsigned numeric integer literal of four or six characters in length.

A new general rule states:

If the AT identifier-7/literal-7 phrase is specified, then the first half, that is, the first two or three digits, of the operand specifies the line value and the second half specifies the column value. When this format of the phrase is used, it is as if both the LINE and POSITION phrases have been specified.

RM/COBOL Language Reference Manual, First Edition, Page 301, Chapter 6, Procedure Division Statements, DISPLAY Screen-Name Statement

The format of the DISPLAY statement includes the AT line-column syntax: AT identifier-3/integer-3.

A new syntax rule states:

identifier-3 (AT) must refer to an unsigned numeric integer display data item of four or six characters in length. integer-3 (AT) must be an unsigned numeric integer literal of four or six characters in length.

A new general rule states:

If the AT identifier-3/integer-3 phrase is specified, then the first half, that is, the first two or three digits, of the operand specifies the line value and the second half specifies the column value. When this format of the phrase is used, it is as if both the LINE and POSITION phrases have been specified.

RM/COBOL Language Reference Manual, First Edition, Page 492, Appendix B, Compiler Messages, Message 442 (replace)

0442: E Table element length exceeds 65535 characters and object version restricted to less than 13.

The maximum table element size has been exceeded for the specified object version (Z Compiler Command Option). Up to 65535 characters may be defined in a table element, that is, the data subordinate to an OCCURS clause, in object versions less than 13. Either adjust the object version limit or reduce the size of the occurring data.

RM/COBOL Language Reference Manual, First Edition, Page 495, Appendix B, Compiler Messages, Message 464 (remove)

Message 464 regards the fixed-size portion of a variable-size group exceeding 65280 characters. This limit has been changed to four gigabytes and the program would overflow available memory before the new limit could be reached. Thus, in RM/COBOL version 10, message 464 is no longer used.

RM/COBOL Language Reference Manual, First Edition, Page 520, Appendix B, Compiler Messages, Message 759 (new message)

0759: W External name length exceeds 30 characters.

The indicated EXTERNAL clause is specified in the file description entry for a file-name or the data description entry for a data-name, and the name is longer than 30 characters in length. The indicated index-name is specified in an external record, and is longer than 30 characters in length.

External names are truncated to 30 characters in the object. External names that are the same in the first 30 characters for similar objects will be considered to reference the same external object at runtime.

RM/COBOL Language Reference Manual, First Edition, Page 520, Appendix B, Compiler Messages, Message 760 (new message)

0760: E Line-column item in AT phrase must be four or six characters in length.

The literal or data item specified in the AT phrase of an ACCEPT or DISPLAY statement must be four or six characters in length. The indicated literal or data item in the source program does not meet this requirement.

RM/COBOL Syntax Summary, First Edition

The RM/COBOL Syntax Summary, First Edition, is the base document describing the RM/COBOL language syntax for RM/COBOL version 10 and covers the RM/COBOL syntax at the time of the version 9 release. The only additional syntax added for version 10 was for compatibility with other COBOL dialects in the ACCEPT and DISPLAY statements, which are described below. (The RM/COBOL Syntax Help file for version 10 on Windows has been updated with the new syntax.)

RM/COBOL Syntax Summary, First Edition, Page 2, Compile Command, Z=version option

In RM/COBOL version 10, the value of version (object version) that may be specified in the Z option now ranges from 9 to 13.

RM/COBOL Syntax Summary, First Edition, Page 23, ACCEPT Statement, Format 3: Accept Terminal I-O

The format of the ACCEPT statement includes the AT line-column syntax: AT identifier-11/literal-9.

RM/COBOL Syntax Summary, First Edition, Page 24, ACCEPT Statement, Format 5: Accept Screen-Name

The format of the ACCEPT statement includes the AT line-column syntax: AT identifier-3/integer-3.

RM/COBOL Syntax Summary, First Edition, Page 28, DISPLAY Statement, Format 2: Display Terminal I-O

The format of the DISPLAY statement includes the AT line-column syntax: AT identifier-7/literal-7.

RM/COBOL Syntax Summary, First Edition, Page 28, DISPLAY Statement, Format 3: Display Screen-Name

The format of the DISPLAY statement includes the AT line-column syntax: AT identifier-3/integer-3.

RM/COBOL User's Guide, First Edition

The RM/COBOL User's Guide, First Edition, is the base document for RM/COBOL version 10 users and covers the RM/COBOL user features at the time of the version 9 release. The RM/COBOL User's Guide supplement document is this readme file; the changes since version 9 are described below, followed by specific corrections to the RM/COBOL User's Guide, First Edition.

The main program argument in version 10 has been increased in size from a maximum of 100 characters to a maximum of 2048 characters.

The C$CARG and C$DARG library routines have been enhanced in version 10 to optionally return a pointer to the requested actual argument data item and a pointer to the encoded PICTURE editing string for a requested actual argument data item that is an edited data item. The C$DARG library routine was also corrected for a problem in accessing an actual argument in excess of the number of arguments specified in the Procedure Division header of the program that calls C$DARG.

The C$PARG library routine has been added in version 10 to obtain a pointer to the nth actual argument data item. The value of n may be larger than the number of formal arguments specified in the program that calls C$PARG in order to access actual arguments not specified in the formal argument list. That is, C$PARG can be used to access actual arguments specified in excess of the number of arguments specified in the Procedure Division header of the program that calls C$PARG.

The SYSTEM library routine in version 10 has been enhanced on Windows NT-class operating systems to allow a command-line string of up to 4096 characters in length.

The runtime system in version 10 has been enhanced with support for object version 13. This object version is required for programs that take advantage of the relaxed limit on the size of data items subject to the OCCURS clause. RM/COBOL versions prior to version 10 only allowed data items up to 65280 characters in length when subject to an OCCURS clause, as enforced by pre-version 10 RM/COBOL compilers.

Runtime error messages in version 10 have been enhanced in three ways to better handle long pathnames, which occur more commonly now than they did in the past. (1) The error message maximum length has been increased from 240 to 512 characters. (2) Pathnames that are too long are now truncated on the left instead of on the right. (3) When formatting messages for the screen, the complete message is displayed even when there are sequences of more than 79 characters with no space included.

In version 10, the RM/COBOL-to-Btrieve Adapter Program is provided to support Btrieve access on Linux operating systems.

For RM/COBOL version 10 on Windows, the COM servers for the compiler and runtime automatically and silently register

themselves when necessary if the user has the required permissions to accomplish the registration. This eliminates "class not registered" errors that may occur when the Windows registry is damaged. Automatic registration is also helpful when client machines access the compiler or runtime from a network share and thus have not registered the COM servers on the client machine as part of an install on the client.

RM/COBOL User's Guide, First Edition, Page 48, Chapter 2, Installation and System Considerations for UNIX, Table 9: Environment Variables for UNIX

Add a note to the entry for the environment variable LD_LIBRARY_PATH: Note that this environment variable name is system-specific. Other UNIX operating systems may use the environment variable name LIBPATH or SH_PATH for this purpose.

RM/COBOL User's Guide, First Edition, Page 54, Chapter 3, Installation and System Considerations for Microsoft Windows, Registering the RM/COBOL Compiler and Runtime Executables

Add a note after the introductory section on Compiler Registration, that is, just before the topic, Registering the Compiler:

Note -- In RM/COBOL version 10 and later, the registration of the compiler COM server, when necessary and if the user has the required permissions, occurs automatically and silently when the compiler is invoked. Thus, in version 10 and later, "class not registered" errors indicate that either the COM server could not be found by the normal search sequence or the current user does not have sufficient permissions to register the compiler COM server.

RM/COBOL User's Guide, First Edition, Page 57, Chapter 3, Installation and System Considerations for Microsoft Windows, Registering the RM/COBOL Compiler and Runtime Executables

Add a note after the introductory section on Runtime Registration, that is, just before the topic, Registering the Runtime:

Note -- In RM/COBOL version 10 and later, the registration of the runtime COM server, when necessary and if the user has the required permissions, occurs automatically and silently when the compiler is invoked. Thus, in version 10 and later, "class not registered" errors indicate that either the runtime COM server could not be found by the normal search sequence or the current user does not have sufficient permissions to register the runtime COM server.

RM/COBOL User's Guide, First Edition, Page 111, Chapter 4, System Considerations for Btrieve

Add the following note after the first paragraph under the chapter title:

Note -- The RM/COBOL-to-Btrieve Adapter Program for Linux (librmbtrv.so) is now available. While this chapter primarily describes the Windows systems considerations for Btrieve, most of the content also applies to the implementation of the Btrieve support module on the Linux operating system. For specific considerations on Linux, see Starting the RM/COBOL-to-Btrieve Adapter Program for Linux in this chapter and the EXTERNAL-ACCESS-METHOD configuration

record. (The new value RMBTRV has been added to the NAME keyword of the EXTERNAL-ACCESS-METHOD configuration record to identify Btrieve access on Linux.)

RM/COBOL User's Guide, First Edition, Page 113, Chapter 4, System Considerations for Btrieve, Required Software Components

Add the following:

For Linux
- Pervasive PSQL v8 (or higher)
- RM/COBOL compiler (development system) for Linux
- RM/COBOL runtime system for Linux
- RM/COBOL-to-Btrieve Adapter support module for Linux (librmbtrv.so)

Pervasive PSQL v8 (or higher) for Linux

The Pervasive PSQL components are a set of programs and libraries that communicate with either the server-based or the client-based Btrieve MKDE.

RM/COBOL User's Guide, First Edition, Page 126, Chapter 4, System Considerations for Btrieve, Starting the RM/COBOL-to-Btrieve Adapter Program for Linux (new topic following the topic, Starting the RM/COBOL-to-Btrieve Adapter Program for Windows)

Add the following:

Starting the RM/COBOL-to-Btrieve Adapter Program for Linux

The RM/COBOL-to-Btrieve Adapter program for Linux, librmbtrv.so, is initiated by placing the following configuration record in the RM/COBOL configuration file (see the EXTERNAL-ACCESS-METHOD configuration record for more information on specifying keywords) and starting the RM/COBOL runtime system:

```
EXTERNAL-ACCESS-METHOD NAME=RMBTRV
```

The only installation requirement is that Linux must be able to locate the various executable files that are required. Place librmbtrv.so in the same directory as the RM/COBOL runtime system (runcobol) for Linux. Typically, the support module is copied into the execution directory (/usr/rmcobol, /usr/local/bin, or /usr/bin).

Furthermore, in order for this support module to be loaded properly, you must make sure that you have set the LD_LIBRARY_PATH environment variable. Add the directory that contains the Pervasive libraries, DSOs (dynamic shared objects), to LD_LIBRARY_PATH. For example:

```
export LD_LIBRARY_PATH=/usr/local/psql/lib:/usr/lib
```

Note that if you logged into the system as "psql", these paths will have already been set.

To verify that the shared object, librmbtrv.so, is being loaded properly by the RM/COBOL runtime, type the following from the shell command line (for more information about the V Option, see Configuration Runtime Command Options):

```
runcobol xxx -v
```

If the following line is displayed in the RM/COBOL runtime

banner, the RM/COBOL-to-Btrieve Adapter for Linux has been loaded correctly:

```
$EXEDIR/librmbtrv.so - RM/COBOL Btrieve Adapter -
       Version n.nn.nn.
```

Note The server-based Btrieve MKDE must be started separately. Refer to the appropriate Btrieve installation and operation manual for information on starting server-based Btrieve.

RM/COBOL User's Guide, First Edition, Page 149, Chapter 6, Compiling, Compile Command, Compile Command Options, "Z" Option

In "where, version must be an integer in the range 7 through 12" change "7 through 12" to "9 through 13". (Since 9 is now the minimum value, the following sentence about certain licenses restricting the minimum version to 9 is unnecessary and will be removed from the next edition of the base document.)

In "The default is to use the current object version number (12) as the limit", change "(12)" to "(13)".

RM/COBOL User's Guide, First Edition, Page 167, Chapter 6, Compiling, Compile Command Messages

In the message "Version error: value must be greater than 6 and less than or equal to 12.", change "greater than 6 and less than or equal to 12" to "greater than 8 and less than or equal to 13".

RM/COBOL User's Guide, First Edition, Page 182, Chapter 7, Running, Runtime Command, Runtime Command Options, "A" Option

Add a new paragraph under the description of the A Runtime Command Option:

Note -- Starting in version 10 of the RM/COBOL runtime system, the maximum A Runtime Command Option value length is 2048 characters; previous versions supported a maximum length of 100 characters. A COBOL program may specify a lower value in the OCCURS clause. In particular, existing programs that specify the previous limit of 100 for the maximum are still valid and do not need to be modified. COBOL programs that specify a lower value can even access up to the current maximum 2048 characters if reference modification is used, as in MAIN-PARAMETER(3:).

RM/COBOL User's Guide, First Edition, Page 183, Chapter 7, Running, Runtime Command, Runtime Command Options, "A" Option

In the MAIN-PARAMETER and APARAM definitions, change "OCCURS 0 TO 100 TIMES" to "OCCURS 0 TO 2048 TIMES".

In the sentence "The number of characters between the delimiter characters cannot exceed 100.", change 100 to 2048.

RM/COBOL User's Guide, First Edition, Page 298, Chapter 10, Configuration, COMPILER-OPTIONS, item 22, OBJECT-VERSION

In the sentence "The value must be an integer in the range 7 through 12.", change "7 through 12" to "9 through 13".

RM/COBOL User's Guide, First Edition, Page 308, Chapter 10, Configuration, EXTERNAL-ACCESS-METHOD, item 2, NAME

The values for the NAME keyword now include RMBTRV for Linux.

RM/COBOL User's Guide, First Edition, Page 314, Chapter 10, Configuration, RUN-ATTR, after item 11, ERROR-MESSAGE-DESTINATION, and before item 12, REVERSE

Add the following new item 12 (and renumber the subsequent items):

12. EXCEPTION-HANDLING. This keyword determines which kind of Windows exception handling is performed if a called non-COBOL subprogram causes an exception. There are three values that can be set: TRY-FINALLY, TRY-EXCEPT, and NONE. The default value is TRY-FINALLY.

    When TRY-FINALLY is set and an exception occurs in a non-COBOL subprogram, Windows displays its normal exception dialog, which allows the operator to view details about the exception. When that dialog is closed, the runtime terminates gracefully with the normal runtime informative traceback messages.

    When TRY-EXCEPT is set and an exception occurs in a non-COBOL subprogram, no Windows dialog is displayed and the runtime immediately terminates with the normal runtime informative traceback messages.

    When NONE is set and an exception occurs in a non-COBOL subprogram, Windows displays its normal exception dialog and the runtime is aborted without the chance to terminate gracefully.

    The previously described behavior assumes that the Debugger item in the AeDebug registry entry (HKLM\Software\Microsoft\Windows NT\CurrentVersion\AeDebug\Debugger) on Windows XP or later is not set or is set to an empty value. If the AeDebug Debugger value is set for Dr. Watson ("drwtsn32"), Visual Studio ("vs?jit.exe"), or some other value, the specified debugger is invoked to handle the exception in the manner it deems fitting and the runtime may or may not be allowed to terminate gracefully. (The Dr. Watson program is installed by default in Windows XP. A Visual Studio debugger is installed during a Microsoft Visual Studio product installation.)

RM/COBOL User's Guide, First Edition, Page 318, Chapter 10, Configuration, RUN-FILES-ATTR, Item 10, FILE-PROCESS-COUNT

Change "The maximum value for this keyword is 4096;" to "The maximum value for this keyword is 16384;".

Add the following warning:

WARNING -- For record and file locks to perform correctly, all run units opening a file must use the same file process count. Thus, it is imperative that all file managers (RM/COBOL, RM/InfoExpress, Relativity, Open File Manager, and so forth), use the same value for the file process count configuration. For further information about changing the file process count, contact Liant technical support services.

RM/COBOL User's Guide, First Edition, Page 318, Chapter 10, Configuration, RUN-FILES-ATTR, Item 13, LARGE-FILE-LOCK-LIMIT

The last sentence on the page is changed from "The default value is 64." to "The default value is 64 for sequential and relative files and 512 for indexed files.".

RM/COBOL User's Guide, First Edition, Page 322, Chapter 10, Configuration, RUN-OPTION, Item 3, ENABLE-LOGGING

Add SUB-CALLS to the list of values and add the following bulleted paragraph:

ENABLE-LOGGING=SUB-CALLS turns on logging of COBOL calls to external non-COBOL subprograms. An entry in this log shows the COBOL program name and line number, the name of the called non-COBOL program, the wall clock time at the end of the call, and the elapsed time of the call. The log file generated is named RMCALLS.LOG.

RM/COBOL User's Guide, First Edition, Page 332, Chapter 10, Configuration, TERM-INPUT, Item 1, ACTION

Add FIELD-END to the list of actions.

RM/COBOL User's Guide, First Edition, Page 340, Chapter 10, Configuration, Field Editing Actions, FIELD-END (new item)

Add a new item 9 before FIELD-HOME and increment the existing item numbers 9 - 21 to 10 - 22.

9. FIELD-END. The FIELD-END value moves the cursor to the leftmost trailing blank or prompt character position of the screen field. If there are no trailing blanks nor prompt characters, the cursor moves to the rightmost position of the screen field. In other words, the cursor moves one position past the end of the text in the field, but not beyond the rightmost position of the field. Thus, if the field contains all blanks or all prompt characters, the cursor moves to the leftmost position of the field.

RM/COBOL User's Guide, First Edition, Page 385, Appendix A, Runtime Error Codes

After error code 94,69, add the following two new errors:

94,70 A DEFINE-DEVICE record with NONBLOCKING-FIFO=YES was specified and the OPEN would have to block because there is no process waiting on the "other end" of the FIFO (that is, if OPEN OUTPUT, no other process has the FIFO open for reading; if OPEN INPUT, no other process has the FIFO open for writing).

94,71 A DEFINE-DEVICE record with NONBLOCKING-FIFO=YES was specified but the file name specified by the PATH keyword is not an existing FIFO.

RM/COBOL User's Guide, First Edition, Page 400, Appendix B, Limits and Ranges

Add an introductory paragraph as follows:

For the most part, RM/COBOL is designed for 32-bit computers. Thus, the limit on the total memory footprint of a program is four gigabytes. In the following limit descriptions, some items are said to have a limit of four gigabytes. This is a theoretical limit since if any one such item was four gigabytes there would be no memory left for any other item in the program.

In the paragraph on the limit for the length of elementary data items, add the additional sentence, "A group data item has a length limit of four gigabytes."

Replace the paragraph on the limit for the length of an element of a table with the following:

The maximum length of an element of a table is 65280 characters for object versions less than 13 (that is, data items subordinate to an OCCURS clause cannot exceed 65280 characters in length). For object versions 13 (product version 10) and later, the limit has been expanded to four gigabytes. Also, prior to version 10, the size of the fixed-size header portion of a variable size group was limited to 65280 characters; for version 10 and later, the fixed-size header portion of a variable-size group is limited to four gigabytes. (The fixed-size header portion of a variable-size group is that portion of the group defined prior to the data item described with the OCCURS clause that specifies the DEPENDING ON phrase.)

Replace the paragraph on the limit of the length of literals generated for any one segment type with the following:

The maximum length of literals generated for any one segment type within a program may not exceed four gigabytes. Segment type refers to fixed permanent, fixed overlayable, and independent.

RM/COBOL User's Guide, First Edition, Page 532, Appendix F, Subprogram Library, C$CARG

Two optional fields have been added to the end of the ARGUMENT-DESCRIPTION structure returned by this subprogram. The fields are ARGUMENT-POINTER and ARGUMENT-PICTURE, both of which must be described as POINTER data items. If the ARGUMENT-PICTURE field is present in the structure, the ARGUMENT-POINTER field must also be present. When the ARGUMENT-DESCRIPTION structure, provided as the argument-description argument to C$CARG, contains one or both of these fields, the address of the selected actual argument data item and the address of the encoded PICTURE character-string are returned as COBOL POINTER values in these fields of the structure. For omitted actual arguments, the ARGUMENT-POINTER and ARGUMENT-PICTURE values will be NULL. For non-edited data items, the ARGUMENT-PICTURE value will be NULL. The encoded PICTURE editing string is not the same as the original PICTURE character-string specified in the source and is rarely useful in a COBOL program. No tools are provided for decoding or using the encoded PICTURE editing string, and its internal format is undocumented and subject to change at any time.

RM/COBOL User's Guide, First Edition, Page 537, Appendix F, Subprogram Library, C$DARG

In the description of the argument-number argument, add the following sentence at the end:

The actual number of arguments may exceed the number of formal arguments declared in the Procedure Division header of the program that calls C$DARG. All of the actual arguments can be accessed using C$DARG even though there is no formal argument name available for accessing the actual arguments beyond the number of formal arguments.

Modify the description of the argument-description argument to include the following information:

Two optional fields have been added to the end of the ARGUMENT-DESCRIPTION structure returned by this subprogram. The fields are ARGUMENT-POINTER and ARGUMENT-PICTURE, both of which must be described as POINTER data items. If the ARGUMENT-PICTURE field is present in the structure, the ARGUMENT-POINTER field must also be present. When the ARGUMENT-DESCRIPTION structure, provided as the argument-description argument to C$CARG, contains one or both of these fields, the address of the selected actual argument data item and the address of the encoded PICTURE character-string are returned as COBOL POINTER values in these fields of the structure. For omitted actual arguments, the ARGUMENT-POINTER and ARGUMENT-PICTURE values will be NULL. For non-edited data items, the ARGUMENT-PICTURE value will be NULL. The encoded PICTURE editing string is not the same as the original PICTURE character-string specified in the source and is rarely useful in a COBOL program. No tools are provided for decoding or using the encoded PICTURE editing string, and its internal format is undocumented and subject to change at any time.

RM/COBOL User's Guide, First Edition, Page 538, Appendix F, Subprogram Library, C$DELAY

Change "PICTURE 9(n) BINARY, where n can be a digit from 1 to 9" to "PICTURE 9(n)V999 BINARY, where n can be a digit from 1 to 7".

Add the following to the end of the same paragraph.

Delays can specify a fractional number of seconds. Thus, calling C$DELAY with a value of 1.5 would attempt to delay for one and a half seconds. The minimum allowed value is .001 seconds, which is one millisecond (ms), but trying to delay for only 1 ms may not actually be meaningful on most systems. Many systems have a clock resolution of 20 ms or more for application programs. Also, the system may be busy. Consequently, calling C$DELAY with .001 might delay 1 ms, 20 ms, 50 ms, or any number in between or longer, depending on your system and its current CPU load.

RM/COBOL User's Guide, First Edition, Page 543, Appendix F, Subprogram Library, C$GetRMInfo

Two new fields, RM-BISValue and RM-ThinClientValue, have been added to the RMInfoGroup structure. These fields have associated condition-names, RM-BISIsPresent and RM-ThinClientIsPresent, respectively. The rminfo.cpy file provided in the version 10 and later releases includes these two fields; the C$GetRMInfo in those releases sets these fields when the subprogram is called.

RM/COBOL User's Guide, First Edition, Page 555, Appendix F, Subprogram Library, C$PARG (new library subprogram)

C$PARG

C$PARG returns a pointer to an actual parameter passed in the USING or GIVING phrases in the CALL statement that called a subprogram.

Calling Sequence

```
CALL "C$PARG" USING argument-number, argument-pointer
```

argument-number is the one-relative ordinal position of the actual argument in the USING phrase of the CALL statement used to call the subprogram that calls C$PARG. The value zero obtains a pointer to the actual argument in the GIVING phrase of that CALL statement. If the value specified is less than zero or greater than the number of actual arguments passed, a null pointer will be returned. The actual number of arguments passed can be obtained with the C$NARG subprogram. The actual number of arguments may exceed the number of formal arguments declared in the Procedure Division header of the program that calls C$PARG. All of the actual arguments can be accessed using C$PARG even though there is no formal argument name available for accessing the actual arguments beyond the number of formal arguments.

argument-pointer must be a pointer data item (USAGE POINTER) that will contain the address of the actual argument upon successful completion of the call. A null pointer value is returned if the call is not successful or if the specified actual parameter was specified as OMITTED. If there is a corresponding formal argument and that argument's base address has been changed with format 5 or 6 of the SET statement, the modified address is returned.

RM/COBOL User's Guide, First Edition, Page 578, Appendix F, Subprogram Library, SYSTEM

The first sentence of the Note that states "Under Windows, the command line is restricted to 130 characters." should be changed to the following: "Under Windows, the command line is restricted to 130 characters for Windows 9x-class operating systems and is restricted to 4096 characters for Windows NT-class operating systems."

RM/COBOL User's Guide, First Edition, Page 583, Appendix G, Utilities, Combine Program

Add the following new note:

Note -- The argument list may use commas or semicolons to separate the list elements. Any number of spaces following a comma or semicolon are ignored. (Prior to version 10, spaces in the argument list caused incorrect parsing of the list.)

RM/COBOL User's Guide, First Edition, Page 584, Appendix G, Utilities, Combine Program, rmpgmcom command line

In the paragraph, "If the filenames are specified through the command line, the command line argument is limited to no more than 100 characters. Combining more than 100 characters of filenames requires direct operator input, use of input redirection, or multiple executions of the program.", change 100 to 2048 in both instances.

RM/COBOL User's Guide, First Edition, Page 586, Appendix G, Utilities, Map Program File

Add the following new note:

Note -- The argument list may use commas or semicolons to separate the list elements. Any number of spaces following a comma or semicolon are ignored. (Prior to version 10, spaces in the argument list caused incorrect parsing of the list.)

RM/COBOL User's Guide, First Edition, Page 589, Appendix G, Utilities, Map Indexed File

Add the following new note:

Note -- The argument list may use commas or semicolons to separate the list elements. Any number of spaces following a comma or semicolon are ignored. (Prior to version 10, spaces in the argument list caused incorrect parsing of the list.)

RM/COBOL User's Guide, First Edition, Page 594, Appendix G, Utilities, Define Indexed File

Add the following new note:

Note -- The argument list may use commas or semicolons to separate the list elements. Any number of spaces following a comma or semicolon are ignored. (Prior to version 10, spaces in the argument list caused incorrect parsing of the list.)

RM/COBOL User's Guide, First Edition, Page 627, Appendix H, Object Versions, Object Version 13 (new object version)

Object Version 13

The RM/COBOL compiler and runtime system versions 10.0n support object version 13. The RM/COBOL version 10.0n runtime systems support the language features of object versions 1 through 13.

New language features were added in object version 13 that are not supported by earlier versions. These are as follows:

The total size of a data element subordinate to an OCCURS clause has been expanded from 65280 to four gigabytes. When the total size of a data element subordinate to an OCCURS clause exceeds 65535, object version 13 or later is required.

Note -- Some language features added in the version 10 release of RM/COBOL do not generate object version 13 code for their implementation. These include the expansion of the limit on total literal size, the expansion of the limit on the fixed-size portion of a variable-length group, and the syntax enhancements to the AT phrase in the ACCEPT and DISPLAY statements.

RM/COBOL User's Guide, First Edition, Page 660, Appendix K, Troubleshooting RM/COBOL

Add the following new section and subsections.

RM/COBOL for UNIX

MESSAGE "Error invoking unauthorized copy of ..."

SUBMESSAGE "Semaphore function error (Loc 808)"
or "Semaphore function error (Loc 809)"

On UNIX, the RM/COBOL runtime and compiler use one SEMUNDO

structure per invocation. On some systems, such as SCO OpenServer 5, a small number, such as 30, of SEMUNDO structures are available via the default configuration of the operating system. In order to run more than about 27 or 28 runtimes simultaneously, a larger number of SEMUNDO structures must be made available. This is done by the UNIX System Administrator configuring a larger number. In order for that larger number to become effective, a relink and reboot of the operating system may be required. Consult your specific operating system documentation for information on increasing the number of SEMUNDO structures.

RM/COBOL User's Guide, First Edition, Pages 697 and 709, Index

Add FIELD-END references in the Index just before the entries for FIELD-HOME.

Compatibility with Other Products

CodeBench for Windows and Enterprise CodeBench

Using CodeBench with Versions 7.5 and Later

To use CodeBench for Windows or Enterprise CodeBench with RM/COBOL version 9 or later, the following lines must be added to the CODEBNCH.INI file in your Windows directory:

```
[Defaults]
IgnoreSize=Y
```

CodeBench performs a check on the size of the RUNCOBOL program to ensure that the correct version is being run. The version 9 and later runtimes do not pass this check because the size of the program has been reduced. The above two lines are automatically inserted into CODEBNCH.INI when the version 9 and later runtimes are installed. However, if CodeBench is installed after the later version runtime is installed, the lines above must be manually inserted into this file.

By default, CodeBench looks for RMCOBOL.EXE in the CodeBench installation directory or in the DOS search path.

Starting with version 7.5, RMCOBOL.EXE can be installed as a console-mode compiler. While the console mode compiler works with CodeBench, it causes a black empty window to appear on the screen for the duration of the compilation.

To avoid this, if you are using CodeBench with the version 7.5 or later RM/COBOL compiler, set this option in your CODEBNCH.INI file:

```
[RMCOBOL]
Path=C:\Program Files\RMCOBOL\rmcobolg.exe
```

Replace the path with the directory where the compiler is installed.

'rmcobolg.exe' is the GUI version of the compiler and does not appear on the screen.

Note that the above step is not necessary if the GUI

compiler was selected during installation as the default RMCOBOL.EXE.

However, for consistency, the above .INI file entry can be made even in that case.

The RM/COBOL version 9 and later compilers are incompatible with CodeBench and Enterprise CodeBench if animation of the program during debugging is desired. CodeBench assumes 5-digit listing line numbers and the version 9 and later compilers produce 6-digit listing line numbers to allow for larger programs. Thus, the line numbers necessary for animation are not correctly entered in the debug shadow files produced when compiling with a version 9 compiler under the control of CodeBench. This incompatibility may be addressed in a future release of the compiler or CodeBench. Until then, a version 8 or less compiler must be used with the existing CodeBench and Enterprise CodeBench products.

Using CodeBench with CodeWatch

CodeWatch can be used to debug programs that are compiled under CodeBench for Windows. There are two requirements:

* Version 7.00.01 or later of the RM/COBOL compiler must be used to perform the compilation.

* An option must be manually added to the codebnch.ini file to force CodeBench to set the Y=2 or Y=3 compilation option. These option settings were added after the last CodeBench release and cannot, therefore, be selected in the CodeBench user interface.

To add the option, edit the codebnch.ini file in your Windows directory. This file may be edited from within CodeBench by selecting Edit from the File menu and entering the above name. Alternatively, Notepad or any other DOS or Windows editor may be used.

Add the following lines to this file:

```
[RMCOBOL]
CommandLineTail="Y=3"
```

If the file already contains the [RMCOBOL] tag, then add the line underneath the tag. Use either Y=2 or Y=3 (see the RM/COBOL User's Guide for a discussion of this option). Finally, recompile your projects. The program files can then be loaded into a CodeWatch workspace but maintained from within CodeBench.

Be sure to comment out the above line when building release versions of the programs from within CodeBench. Otherwise, your release executables will contain debugging information regardless of the options selected from within CodeBench. You must also comment out the above line when using a UNIX RM/COBOL compiler earlier than version 7.1 from within Enterprise CodeBench. To comment out the line, insert a semicolon (;) before the CommandLineTail keyword.

CodeWatch reads the CommandLineTail configuration parameter each time a group of compilations is started, so the option may be changed without restarting CodeWatch.

Note that CodeWatch can access only files that are visible to Windows. UNIX debugging with CodeWatch is not supported in this release.

Problems in This Release

There is an incompatibility between the existing CodeBench, including Enterprise CodeBench, and the RM/COBOL compiler starting with the version 9 release. See the section "Compatibility with Other Products" in this document for an explanation of this incompatibility.

Media Contents

The following is a list of files included on your RM/COBOL development system media:

File	Description
READ.ME	This README text file
RMCOBOL.EXE	Compiler (either console or GUI-mode, as selected during the installation, OEM default native character set)
RMCOBOLC.EXE	Compiler (console-mode, OEM default)
RMCOBOLC_ANSI.EXE	Compiler (console-mode, ANSI default)
RMCOBOLC_OEM.EXE	Compiler (console-mode, OEM default)
RMCOBOLG.EXE	Compiler (GUI-mode, OEM default)
RMCOBOLG_ANSI.EXE	Compiler (GUI-mode, ANSI default)
RMCOBOLG_OEM.EXE	Compiler (GUI-mode, OEM default)
RMCBL10C.DLL	Compiler system support module Installed in the Liant Shared directory under Program Files\Common Files.
RMCBL10P.DLL	Compiler system support module Installed in the Liant Shared directory under Program Files\Common Files.
RMNET10C.DLL	Compiler system support module
RMNET10P.DLL	Compiler system support module
RMLIBCMP.DLL	Compiler support module
COMPILER.CFG	Compiler sample configuration file
RMCOBSYN.HLP	Compiler syntax summary help file
RMCOBSYN.CNT	Compiler syntax summary help contents file
RMCW.EXE	CodeWatch debugger
RMCW_RXR.DLL	CodeWatch debugger extension DLL
RMCW.HLP	CodeWatch help file
RMCW.CNT	CodeWatch help contents file
CBRIDGE.EXE	CodeBridge Builder application
PARSER.EXE	CodeBridge Builder support program
RECONSTR.EXE	CodeBridge Builder support program
SCANNER.EXE	CodeBridge Builder support program
DLLGEN.IN	CodeBridge Builder data file
DLLGEN.OUT	CodeBridge Builder data file
DLLGEN.P01	CodeBridge Builder data file
DLLGEN.SYM	CodeBridge Builder data file
RUNCOBOL.EXE	Runtime system (OEM default native character set)
RUNCOBOL_ANSI.EXE	Runtime system (ANSI default)
RUNCOBOL_OEM.EXE	Runtime system (OEM default)
RMSETNCS.EXE	Utility for switching default native character set between ANSI and OEM
RMCBL10R.DLL	Runtime system support module Installed in the Liant Shared directory under Program Files\Common Files.

RMNET10R.DLL	Runtime system support module
RMLIBRUN.DLL	Runtime system support module
WINDOWS.CFG	Runtime sample configuration file
RMGUIFE.DLL	Graphical user interface module
ALLEGRIS.DLL	Graphical user interface support module
RMPROP.DLL	COBOL program property sheet handler
RMUC0.DLL	Use-count support DLL
RMUC1.DLL	Use-count support DLL
RMUC2.DLL	Use-count support DLL
RMICONS.DLL	Runtime system support module
LIBRMCFG.DLL	RM/COBOL Automatic Configuration File module
_UNINSTL.DLL	Un-install support DLL
_UNINSTL.DAT	Un-install support data
UNINST.ISU	Un-install support module
RMUCUTIL.EXE	Use-count utility
RMFM32.DLL	File Manager Dynamic Link Library
OFM32.DLL	Open File Manager Dynamic Link Library
RMTBAR.VRF	Toolbar icon button resource file
WINERROR.TXT	List of Windows error codes
LICENSE.VLT	License vault
LICENSE.VLT.LCK	License vault lock file
RMATTACH.EXE	Attach config file to .EXE file
INI2REG.EXE	Initialization (.INI) file to registry conversion utility
RMCONFIG.EXE	RM/COBOL registry editor
RMCOBOL.REG	RM/COBOL default registry entries
RUNPAN2.COB	RM/Panels runtime library (object)
RMPAN2W.CFG	RM/Panels runtime configuration file
WOWPANRT.DLL	WOW enabled RM/Panels runtime support DLL
WOWMFCRT.DLL	WOW runtime support DLL
WOWRT.DLL	WOW runtime support DLL
RPCPLUS.DLL	WOW Thin Client support DLL
RPCPLUS.INI	WOW Thin Client initialization file
RPCPLUSWOW.DLL	WOW Thin Client support DLL
RPCPlusServer.EXE	WOW Thin Client Server program
RPCPlusService.EXE	WOW Thin Client Service program (Windows service version of RPCPlusServer)
HELOWRLD.COB	WOW Thin Client example COBOL program
DOVERIFY.CBL	Compile and run verify suite (source)
FILETEST.CBL	File system test (source)
NUCTEST.CBL	Nucleus test (source)
PACETEST.CBL	Multi-user file locking test (source)
PRNTEST.CBL	Printer configuration test (source)
SORTTEST.CBL	Sort/merge test (source)
VDTTEST.CBL	Terminal interface test (source)
VERIFY.CBL	Main menu (source)
WINATTRB.CBL	Windows sub-test (source)
WINBORDR.CBL	Windows sub-test (source)
WINCOLOR.CBL	Windows sub-test (source)
WINRELTV.CBL	Windows sub-test (source)
WINSTAT.CBL	Windows sub-test (source)
WINTEST.CBL	Windows test (source)
WINTITLE.CBL	Windows sub-test (source)
WIN.CPY	Windows test copy deck
DEFDEV.CPY	Windows define device copy deck
DEVCAPS.CPY	Windows device capabilities copy deck
LOGFONT.CPY	Windows logical font copy deck
PRINTDLG.CPY	Windows print dialog copy deck
PRINTINF.CPY	Windows printer information copy deck
RMINFO.CPY	Windows RM/COBOL information copy deck
SYSINFO.CPY	Windows system information copy deck
TXTMTRIC.CPY	Windows text metric copy deck
WINDEFS.CPY	Windows miscellaneous definitions copy deck
VERSION.CPY	Version number copy deck

IXVERIFY.CBL	RM/InfoExpress verify program (source)
DOVERIFY.COB	Compile and run verify suite (object)
FILETEST.COB	File system test (object)
NUCTEST.COB	Nucleus test (object)
PACETEST.COB	Multi-user file locking test (object)
PRNTEST.COB	Printer configuration test (object)
SORTTEST.COB	Sort/merge test (object)
VDTTEST.COB	Terminal interface test (object)
VERIFY.COB	Main menu (object)
WINATTRB.COB	Windows sub-test (object)
WINBORDR.COB	Windows sub-test (object)
WINCOLOR.COB	Windows sub-test (object)
WINRELTV.COB	Windows sub-test (object)
WINSTAT.COB	Windows sub-test (object)
WINTEST.COB	Windows test (object)
WINTITLE.COB	Windows sub-test (object)
IXVERIFY.COB	RM/InfoExpress verify program (object)
RMLOGO.BMP	Liant logo bitmap
RMTCP32.CFG	RM/InfoExpress TCP client configuration file
RMBARS.DLL	Menubar, Statusbar, Toolbar utility
MSGBOX.DLL	Example DLL to display a message box
C$TITLE.DLL	Change main window title
C$BITMAP.DLL	Display bitmap image in middle of window
C$GUICFG.DLL	Dynamically configure GUI properties
C$BTRV.DLL	COBOL callable Btrieve interface
RMARG.DLL	RMArg Linkage Arguments support DLL
XMLIF.DLL	XMLIF library support DLL
RMC85CAL.H	Include file for subprograms
STANDDEF.H	Header file used to build COBOL-callable DLLs
RMPORT.H	Header file used to build COBOL-callable DLLs
RTARG.H	Header file used to build COBOL-callable DLLs
RTCALLBK.H	Header file used to build COBOL-callable DLLs
RTCCD.H	Header file used to build user-defined MCS DLL
CBRIDGE.H	CodeBridge include file
RTCBDEFS.H	CodeBridge definitions include file
USRMCS.C	Stub "C" source to build user-defined MCS DLL
ANALYSIS.CBL	Instrumentation data analysis (source)
ANALYSIS.COB	Instrumentation data analysis (object)
RECOVERY.COB	Execute recovery options (object)
RECOVER1.EXE	Recover indexed file in place
RECOVER2.COB	Unload to sequential file (object)
RMDEFINX.COB	Precreate indexed file
RMMAPPGM.COB	List contents of object file or library
RMPGMCOM.COB	Create object library
RMMAPINX.COB	List structure of indexed file
DSPDIALG.COB	Display print dialog
PDEVICE.COB	Get printer device capabilities
PDIALOG.COB	Get printer dialog information
PGETFONT.COB	Get printer font information
PRINFO.COB	Get printer information
PTEXTMET.COB	Get printer text metrics
RMINFO.COB	Get RM information
SYSINFO.COB	Get system information
RMCT16.DLL	Support file for calling 16-bit non-COBOL subprograms in DLLs.
RMCT32.DLL	Support file for calling 16-bit non-COBOL subprograms in DLLs.

```
RMBTRV32.DLL       Btrieve adapter

RMTCP32.DLL        RM/InfoExpress TCP client

RMTBEDIT.EXE       Toolbar button editor
RMTBEDIT.VRF       Toolbar button editor resource file

SAMPLES\*.*        Various sample COBOL programs

CBRIDGE\*.*        CodeBridge samples

SupportTools\*.*   Diagnostic utilities

Subdirectory SupportTools contains miscellaneous
diagnostic utilities.

RMUG_E01.PDF       RM/COBOL User's Guide
RMLR_E01.PDF       RM/COBOL Language Reference Manual
RMSS_E01.PDF       RM/COBOL Syntax Summary
RMCW_E01.PDF       CodeWatch User's Guide
RMCB_E01.PDF       CodeBridge User's Guide

ASYCFILT.DLL    Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
ATL.DLL         Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
COMCAT.DLL      Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
COMCTL32.DLL    Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
CTL3D32.DLL     Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
CTL3DV2.DLL     Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
MFC42.DLL       Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
MFC42U.DLL      Windows NT system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
MFC71.DLL       Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
MSVCP60.DLL     Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
MSVCP71.DLL     Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
MSVCR71.DLL     Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
MSVCRT.DLL      Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
OLEAUT32.DLL    Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
OLEPRO32.DLL    Windows system file required by RM/COBOL.
                  Installed in the system directory if
                  it does not already exist.
STDOLE2.TLB     Windows type library required by RM/COBOL.
                  Installed in the system directory if
```

	it does not already exist.
MTEXTRA.TTF	TrueType Font file required by RM/COBOL. Installed in the system directory if it does not already exist.

Subdirectory SAMPLES contains various sample COBOL and C programs that the user may find useful and instructional. Some of the COBOL programs include comprehensive examples of the use of the P$ subprograms described in Appendix E of the RM/COBOL User's Guide.

Subdirectory CBRIDGE contains various CodeBridge samples. See file CBRIDGE\README.TXT for an explanation of the samples provided in the CBRIDGE subdirectory.