3

THE CONVERSION PROCESS

The first major step in updating the legacy application involves converting the code to RPG IV. You've just discovered you have a minimal amount to learn before you can start using RPG IV. With just the basics, it is as easy—if not easier—to program in RPG IV as it is in RPG III.

To convert an RPG III program to an RPG IV program, you merely run the source through a conversion tool and recompile the program! Are there any potential difficulties? You have to give some thought to copy members. Apart from that, the process is embarrassingly simple.

The thought of code conversion can send a shiver down one's spine. Conversion would seem to necessitate changes to all of the programs in a legacy application and subsequent retesting, but that process is more the exception than the norm. Converting source and recompiling a program does not necessarily mean that the program must be retested. Although it sounds like heresy, it is true. Whether or not a program needs retesting depends upon whether you've changed the code. Conversion *reformats* the code but does not change it.

Perhaps you also shudder at the thought of maintaining programs using two versions of RPG. Why not just standardize on RPG IV? You don't have to convert all programs in one fell swoop; do it only when the code requires changes. RPG III programs will quite happily co-exist with RPG IV programs.

You should only consider converting all of the programs in an application if you intend to perform a lot of re-engineering. For example, perhaps you decide to change the database to incorporate date and time fields and you wish to rid yourself of all your date and time subroutines and to make use of the new date and time operations instead.

Whether you choose the straightforward conversion or the re-engineering approach, you will convert your code in some manner. Let's look at some conversion options.

CONVERSION OPTIONS

The process entails a simple conversion of the source and recompiling of the program. Three conversion options are available to you:

- Use the IBM option. It comes with the RPG compiler and is ready to go.
- Use a third-party option. The available third-party conversion options will, for the most part, perform more re-engineering than the IBM option.
- Write your own.

Preparing for Conversion

Regardless of which conversion option you choose, you must give some consideration to the following points concerning source files and copy members.

The default name for RPG IV source physical files is QRPGLESRC, and sources will have a member type of RPGLE.

Since RPG IV now has a 100-character source statement, you would assume that the source file would need to have a length of 112 instead of the usual 92. The assumption is correct if you have comments in comment columns on a specification. Otherwise, you can get away with the default length of 92.

Conversion tools give you the option of converting the copy members separately (the default) or including them in the converted source. Including them in the source means you have lost the benefit of the copy member and have instead duplicated it in every source.

You also need to pay attention to the actual /COPY statements themselves and how they identify the source file for the copy member. This topic warrants further discussion when you look at the conversion reports later in this chapter.

THE IBM OPTION

The AS/400 RPG compiler has a built-in conversion tool: the Convert RPG Source (CVTRPGSRC) command. This tool performs only *basic* conversion of RPG III to RPG IV, with no conversion to lowercase, no inclusion of EVAL statements, and so forth. Code simply gets translated to its RPG IV equivalent.

The nearest that this conversion comes to re-engineering the source is in moving the E-specs, data structures, and named constants to the new D-spec. If you want the easiest way to begin using RPG IV, turn to CVTRPGSRC. The converted code is as close as you can get to RPG III.

The Log File

Before you run any conversions, you must create a log file that will contain details of every conversion run and every program converted. To create this log file, named QRNCVTLG by default unless you specify otherwise, you copy the file QARNCVTLG in library QRPGLE.

Converting

You are now ready to convert the application source code. To do so with IBM's CVTRPGSRC, you issue the commands shown in Figure 3.1.

```
CVTRPGSRC FROMFILE(ALLTHAT101/QCPYSRC) FROMMBR(*ALL)
          TOFILE(ALLTHAT102/QCPYSRC)

CVTRPGSRC FROMFILE(ALLTHAT101/QRPGSRC) FROMMBR(*ALL)
          TOFILE(ALLTHAT102/QRPGLESRC)
```

Figure 3.1: Converting source with IBM's CVTRPGSRC conversion tool.

These commands convert all members from ALLTHAT101/QCPYSRC to ALLTHAT102/QCPYSRC and all members from ALLTHAT101/QRPGSRC to ALLTHAT102/QRPGLESRC. You have elected, by default, to convert copy members separately.

As well as converting the source, the CVTRPGSRC command also generates a conversion report. You can create a conversion report without actually converting the source if you issue the command in Figure 3.2.

```
CVTRPGSRC FROMFILE(ALLTHAT101/QRPGSRC) FROMMBR(*ALL) TOFILE(*NONE)
```

Figure 3.2: Creating the conversion report without converting the source.

Figure 3.3 duplicates part of the conversion report. Basically, for each program, the report shows each CALL, DEBUG, and FREE operation, as well as each /COPY directive. The report calls attention to DEBUG and FREE operations because they are no longer supported in RPG IV (what a loss!), and the CALL operations are highlighted for ILE consideration.

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+
 5769RG1 V4R2M0  980228 RN        IBM ILE RPG
  Command  . . . . . . . . . . . :   CVTRPGSRC
    Issued by  . . . . . . . . . :     TUOHYP
  From file  . . . . . . . . . . :   QRPGSRC
    Library  . . . . . . . . . . :     ALLTHAT101
  From member  . . . . . . . . . :   *ALL
  To file. . . . . . . . . . . . :   QRPGLESRC
    Library  . . . . . . . . . . :     ALLTHAT102
  To member  . . . . . . . . . . :   *FROMMBR
  Log file . . . . . . . . . . . :   QRNCVTLG
    Library  . . . . . . . . . . :     *LIBL
  Log member . . . . . . . . . . :   *FIRST
  Expand copy members. . . . . . :   *NO
  Print conversion report  . . . :   *YES
  Include second level text. . . :   *NO
  Insert specification template. . :   *NO
  From file  . . . . . . . . . . :   ALLTHAT101/QRPGSRC(ALL001A)
  To file. . . . . . . . . . . . :   ALLTHAT102/QRPGLESRC(ALL001A)
```

Figure 3.3: CVTRPGSRC conversion report. (Part 1 of 2)

```
  Log file . . . . . . . . . . . . :   *LIBL/QRNCVTLG(QARNCVTLG)
                     C o n v e r s i o n    R e p o r t
Sequence <------------ Source Specifications ------
Number   ....1....+....2....+....3....+....4....+....5....+....6....+.
  015700 C                   CALL      'GENCLRM'
*RNM0511 00 CALL operation code found.
  023700 C                   CALL      'GENCLRM'
*RNM0511 00 CALL operation code found.
  032400 C                   CALL      'GENSNDM'     WMSGLS
*RNM0511 00 CALL operation code found.
  033500 C                   CALL      'GENSNDM'     WMSGLS
*RNM0511 00 CALL operation code found.
  034500 C                   CALL      'GENSNDM'     WMSGLS
*RNM0511 00 CALL operation code found.
  035400 C                   CALL      'GENSNDM'     WMSGLS
*RNM0511 00 CALL operation code found.
  042800 C                   CALL      'GENSNDM'     WMSGLS
*RNM0511 00 CALL operation code found.
  043600 C                   CALL      'GENSNDM'     WMSGLS
*RNM0511 00 CALL operation code found.
  046200 C                   CALL      'GENSNDM'     WMSGLS
*RNM0511 00 CALL operation code found.
  048300 C/COPY QCPYSRC,SPSSR
*RNM0508 00 /COPY compiler directive found.
  048400 C/COPY QCPYSRC,PGENSNDM
*RNM0508 00 /COPY compiler directive found.
       * * * * *  E N D  O F  S O U R C E  * * * * *
M e s s a g e  S u m m a r y
 Msg id  Sv Number Message text
*RNM0508 00      2 /COPY compiler directive found.
*RNM0511 00      9 CALL operation code found.
       * * * * *  E N D  O F  M E S S A G E  S U M M A R Y  * * *
                      F i n a l  S u m m a r y
  Message Totals:
    Information  (00) . . . . . . . :       11
    Warning      (10) . . . . . . . :        0
    Severe Error (30+)  . . . . . . :        0
   --------------------------------   -------
    Total . . . . . . . . . . . . . :       11
  Source Totals:
    Original Records Read . . . . . :      484
    Converted Records Written . . . :      484
    Highest Severity Message Issued . :      0
         * * * * *  E N D  O F  F I N A L  S U M M A R Y  * * * *
```

Figure 3.3: CVTRPGSRC conversion report. (Part 2 of 2)

The relevance of showing the /COPY directives is to help you in determining how to handle copy members. Let's consider a couple of examples.

A directive of /COPY SPSSR would indicate that the copy member is in the same source physical file as the program. Therefore, the conversion process is a simple one: since both the program and copy source members are converted from QRPGSRC to QRPGLESRC, the QRPGLESRC source physical file can be created in the same library and be treated just as another source file.

However, a directive of /COPY QCPYSRC,SPSSR causes one to rethink. To avoid having to change the program source, the converted copy members must also be in a source physical file named QCPYSRC. Therefore, you would have to have your sources in a new library.

The Converted Source

```
0001.00  H        1

0064.00   *
0065.00  FALL001D CF  E                    WORKSTN      KINFSR *PSSR
0066.00  F                                        #RRN  KSFILE SUBREC
0067.00  F                                        #RRN  KSFILE OLDREC
0068.00   *
0069.00  FCATEGOR1UF  E           K        DISK         KINFSR *PSSR A
0070.00  F                                             KINFDS RECFDS
0071.00  F            CATEGORR                         KRENAMERECFMT
0072.00   *

=============================================================================

0001.00 H DEBUG

0064.00  *
0065.00 FALL001D   CF   E             WORKSTN INFSR(*PSSR)
0066.00 F                                     SFILE(SUBREC:#RRN)
0067.00 F                                     SFILE(OLDREC:#RRN)
0068.00  *
0069.00 FCATEGOR1  UF A E           K DISK    INFSR(*PSSR)
0070.00 F                                     INFDS(RECFDS)
0071.00 F                                     RENAME(CATEGORR:RECFMT)
0072.00  *
```

Figure 3.4: ALL001A—Converted H- and F-specs.

Let us examine some of the conversion results. Figure 3.4 shows how the H- and F-specs have been restructured. The H-spec is completely free-format and the F-specs have adopted a DDS appearance. Yet, something seems strange within the restructured F-spec. Whereas a lot of column orientation is replaced with keywords and values, the exception is the A for *addition* in column 66: it moves to column 20!

Figure 3.5 shows how the E- and I-specs have been replaced by the new D-spec. This is even closer to DDS and definitely more legible.

```
0072.00  *
0073.00  *------------------------------------------------------------
0074.00  *
0075.00  * Array to check for duplicate Adds
0076.00  *
0077.00 E                    CODE       50  2
0078.00  *
0079.00  *------------------------------------------------------------
0080.00  *
0081.00  * INFDS for the update file.
0082.00  *
0083.00 IRECFDS      DS
0084.00 I                                    *STATUS  RECSTS
0085.00  *
0086.00  *------------------------------------------------------------
0087.00  *
0088.00  * MODS to allow for same field names on DSPF and DB files
0089.00  *
0090.00 IDSOCUR    E DSCATEGOR                  2
0091.00  *
0092.00  *------------------------------------------------------------
0093.00  *
0094.00  * DS used to check image of original record against the
0095.00  * image of the rechained record, to ensure that it has not
0096.00  * been updated in the meantime
0097.00  *
0098.00 ICHKOLD      DS                           500
0099.00  *
0100.00  *------------------------------------------------------------
0101.00  *
0102.00  * Work
0103.00  *
```

Figure 3.5: ALL001A's E- and I-specs converted to D-specs. (Part 1 of 3)

```
0104.00 I              '0000000000000000000'C         #ZR20
0105.00  *
0106.00  *-------------------------------------------------------------
0107.00  *
0108.00  * Program status
0109.00  *
0110.00 I           SDS
0111.00 I                                       *PROGRAM WPGMNM

============================================================================

0072.00  *
0073.00  *-------------------------------------------------------------
0074.00  *
0075.00  * Array to check for duplicate Adds
0076.00  *
0077.00 D CODE            S              2    DIM(50)
0078.00  *
0079.00  *-------------------------------------------------------------
0080.00  *
0081.00  * INFDS for the update file.
0082.00  *
0083.00 D RECFDS          DS
0084.00 D  RECSTS           *STATUS
0085.00  *
0086.00  *-------------------------------------------------------------
0087.00  *
0088.00  * MODS to allow for same field names on DSPF and DB files
0089.00  *
0090.00 D DSOCUR        E DS                  OCCURS(2) EXTNAME(CATEGOR)
0091.00  *
0092.00  *-------------------------------------------------------------
0093.00  *
0094.00  * DS used to check image of original record against the
0095.00  * image of the rechained record, to ensure that it has not
0096.00  * been updated in the meantime
0097.00  *
0098.00 D CHKOLD          DS           500
0099.00  *
0100.00  *-------------------------------------------------------------
0101.00  *
0102.00  * Work
0103.00  *
0104.00 D #ZR20           C                    CONST('00000000000000000')
0105.00  *
```

Figure 3.5: ALL001A's E- and I-specs converted to D-specs. (Part 2 of 3)

```
0106.00  *-------------------------------------------------------
0107.00  *
0108.00  * Program status
0109.00  *
0110.00 D                SDS
0111.00 D  WPGMNM           *PROC
```

Figure 3.5: ALL001A's E- and I-specs converted to D-specs. (Part 3 of 3)

The C-specs have also been restructured. Figure 3.6 shows the new indexing format for arrays (with the index now enclosed in brackets rather than delimited by a comma) and the elongated format of the operation codes. For all intents and purposes, it appears like a stretched C-spec.

```
        CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
0178.00 C                    MOVEA'001'     *IN,51

0342.00 C           CATCOD   LOKUPCODE                    90

0361.00 C                    ADD  1         X
0362.00 C                    MOVE CATCOD    CODE,X

======================================================================

        CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
0178.00 C                   MOVEA     '001'         *IN(51)

0342.00 C     CATCOD        LOOKUP    CODE                                 90

0361.00 C                   ADD       1             X
0362.00 C                   MOVE      CATCOD        CODE(X)
```

Figure 3.6: ALL001A's restructured C-specs.

As you can see from these examples, the programs, although converted, are still obviously RPG code. All that remains is to recompile them, using option 14 in PDM or using the Create Bound RPG Program (CRTBNDRPG) command. This is the direct replacement for the Create RPG Program (CRTRPGPGM) command.

Listings B.1, B.2, B.3, B.4, B.5, and B.6 in Appendix B show the converted program sources in full.

THE THIRD-PARTY OPTION

If you want a little more than the IBM option offers, you can invest in a third-party conversion tool. Linoma Software offers such a tool at

```
www.linomasoftware.com
```

as does ProData at

```
http://as400.prodatacomputer.com
```

or you can even make use of a shareware tool from programmer Brad V. Stone at

```
www.bvstools.com
```

Trial versions are readily available for download from the respective Web sites.

This book uses Linoma Software's Convert to ILE RPG (CVTILERPG) tool to walk you through conversion using a third-party tool. This choice does not suggest that Linoma's product is any better or worse than the others—just that I am familiar with it. Evaluate all the tools and choose the one that best suits your needs and budget.

The Extras

As well as converting source to RPG IV, Linoma Software's CVTILERPG conversion tool also provides options to:

- Redefine data structures' fields by reordering them naturally, converting from/to positions to actual lengths, and indenting subfields.
- Redefine *LIKE DEFN-defined fields in the D-specs.

- Redefine C-spec–defined fields in the D-specs.
- Redefine any of the ADD, SUB, MULT, DIV, Z-ADD, Z-SUB, MOVE, MOVEL, SETON, SETOF, and COMP operations as free-form EVAL operations.
- Perform case conversion to either lowercase or mixed case.

CVTILERPG automatically converts structured operation codes to their free-form equivalents. It also offers a host of documentation options such as converting the constants 1 and 0 to *ON and *OFF, converting end operations to their corresponding Endxx operations, highlighting comment lines, and more.

Converting

To convert the legacy application with Linoma Software's CVTILERPG, you would issue the commands in Figure 3.7.

```
CVTILERPG FROMFILE(ALLTHAT101/QCPYSRC) FROMMBR(*ALL)
          TOFILE(ALLTHAT103/QCPYSRC)

CVTILERPG FROMFILE(ALLTHAT101/QRPGSRC) FROMMBR(*ALL)
          TOFILE(ALLTHAT103/QRPGLESRC)
```

Figure 3.7: Converting source with Linoma Software.

CVTILERPG produces the same conversion report as CVTRPGSRC (because it uses the CVTRPGSRC command to perform the initial conversion), showing each CALL, DEBUG, and FREE operation and each /COPY directive. Refer to Figure 3.3 again to see part of the conversion report.

The Converted Source

The most obvious impact of this conversion is the change to mixed case, which is optional, by the way. This gives the programs a new and unfamiliar look. Most traditional RPG programmers find the mixed case and blank lines one of the hardest things to get used to, but with a little perseverance you will start to appreciate their benefits. You can use the mixture of uppercase and lowercase to make

names more legible (CustFile as opposed to CUSTFILE), and you no longer have to key an asterisk to emulate a blank line.

Figure 3.8 shows the inclusion of field definitions on the D-specification. While the definition of the fields has been moved to the D-spec, the original C-specs, although restructured, are still intact. You will read about changing this in Chapter 5.

```
FMT C  CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments+++++++..
0477.00 C           *INZSR    BEGSR
0478.00  *
0479.00 C                     Z-ADD0         X       30
0480.00 C                     MOVE '*END'    #CTL    4
0481.00  *
0482.00 C                     ENDSR

============================================================================
    FMT D  DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++
0112.00  *---------------------------------------------------------------
0113.00  * BEGIN of work fields added by the CVTILERPG utility
0114.00  *---------------------------------------------------------------
0115.00 D #Ctl            S              4
0116.00 D X               S              3  0
0117.00  *---------------------------------------------------------------
0118.00  * END of work fields added by the CVTILERPG utility
0119.00  *---------------------------------------------------------------


    FMT C
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
0485.00 C     *INZSR        BEGSR
0486.00  *
0487.00 C                   EVAL      X = 0
0488.00 C                   MOVE      '*END'        #Ctl
0489.00  *
0490.00 C                   ENDSR
```

Figure 3.8: ALL001A—Definition of fields in the D-specs.

Figure 3.9 shows the change to EVAL. Figure 3.10 shows the change to free-format structured operation codes.

```
 FMT C
CLON01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments+++++++..
0357.00 C                   Z-ADD#RRN      #ERREC

0361.00 C                   ADD  1         X

0412.00 C                   Z-ADD1         #RRN

******************************************************************************

 FMT CX
CLON01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++
0365.00 C                  EVAL      #Errec = #Rrn

0369.00 C                  EVAL      X = X + 1

0420.00 C                  EVAL      #Rrn = 1
```

Figure 3.9: ALL001A—Redefinition to EVAL.

Listings C.1, C.2, C.3, C.4, C.5, and C.6 in Appendix C show the converted program sources in full. If you compare these to the corresponding sources in Appendix B, you will see how a little bit of re-engineering can give the programs what seems to be a whole new appearance.

No matter which conversion option you use, you will always find extra candidates for conversion, which leads to a discussion of the third conversion option.

```
 FMT C
CLON01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments+++++++..
0343.00 C           *IN90     IFEQ *ON

0347.00 C           #ERREC    IFEQ 0

0381.00 C           #ERREC    IFNE 0

0415.00 C           *IN90     DOWEQ*OFF
0416.00  *
0417.00 C           #RRN      IFLE #NOREC
******************************************************************************
```

Figure 3.10: ALL001A—Redefinition to free-format structured operation code. (Part 1 of 2)

```
 FMT CX
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++
0351.00 C                   IF        *IN90 = *ON

0355.00 C                   IF        #Errec = 0

0389.00 C                   IF        #Errec  0

0423.00 C                   DOW       *IN90 = *OFF
0424.00  *
0425.00 C                   IF        #Rrn <= #Norec
```

Figure 3.10: ALL001A—Redefinition to free-format structured operation code. (Part 2 of 2)

WRITE YOUR OWN

Generally speaking, the state of your source code (as in adherence to standards) determines the benefit of writing your own conversion programs. To demonstrate the principle of writing your own conversion program, let's explore how you might handle two simple examples of further conversion requirements.

- Chapter 2's coverage of the H-spec mentioned a growing practice of defining a standard H-spec in a copy member and then including it in programs using a /COPY directive. Therefore, let's make the conversion program replace the H-spec with a /COPY directive. Figure 3.11 shows the copy member for the H-spec.

- Operation codes and keywords are still in uppercase and, since I've grown quite fond of mixed case, I would prefer all keywords on F- and D-specs and all operation codes to be in mixed case.

```
H Debug DatEdit(*DMY/) Option(*SrcStmt)
```

Figure 3.11: Copy member STDHSPEC—standard H-spec for all programs.

The conversion now involves a two-step process. First, you convert the source using either the CVTRPGSRC or CVTILERPG command; then you apply the

additionally required changes by issuing your own Convert RPG Local Changes (CVTRPGLCL) command, as shown in Figure 3.12.

```
CVTRPGLCL FROMFILE(ALLTHAT103/QRPGLESRC) FROMMBR(MemberName)
          TOFILE(ALLTHAT104/QRPGLESRC)
```

Figure 3.12: Additional conversion with your own CVTRPGLCL.

The additional conversion routine consists of a command (CVTRPGLCL), a CL command processing program (CVTRPGLCLC), and an RPG IV program (CVTRPGLCLR), all of which are documented in Appendix D. But, since the RPG program uses some of the new string-handling and built-in functions of RPG IV, you might fare better if you examine the program in detail only after covering more features of the language in Chapters 4 and 5.

For now, suffice it to say that the conversion program replaces the H-spec with a /COPY directive and uses scan operations and arrays of from and to values to replace keywords and operation codes.

Figure 3.13 shows the inclusion of the /COPY directive for the H-specification and the keywords in mixed case on the F- and D-specs.

```
0001 H/COPY QCPYSRC,STDHSPEC

 FMT FX
FFilename++IPEASF.....L.....A.Device+.Keywords++++++++++++++++++++++++++
0065.00 FAll001D   CF   E             WorkStn InfSR(*PSSR)
0066.00 F                                     SFile(Subrec:#Rrn)
0067.00 F                                     SFile(Oldrec:#Rrn)
0068.00  *
0069.00 FCategor1  UF A E           K DISK    InfSR(*PSSR)
0070.00 F                                     InfDS(Recfds)
0071.00 F                                     Rename(Categorr:Recfmt)
```

Figure 3.13: ALL001A—Keywords converted to mixed case.

Figure 3.14 show the operation codes in mixed case.

```
 FMT C
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
0140.00 C                   Exsr      Initsr
0141.00  *
0142.00 C                   DoU       #Ctl = '*END'
0143.00  *
0144.00 C     #Ctl          CasEQ     'LOAD'        Loadsf
0145.00 C     #Ctl          CasEQ     'DISP'        Dispsc
0146.00 C     #Ctl          CasEQ     'VALD'        Valdsf
0147.00 C     #Ctl          CasEQ     'PROC'        Procsf
0148.00 C                   EndCS
0149.00  *
0150.00 C                   EndDO
0151.00  *
0152.00 C                   Move      *ON           *INLR
```

Figure 3.14: ALL001A—Operation codes converted to mixed case.

THE COST

The major cost of conversion is space. If you choose to increase the size of the source physical file to 112, RPG IV source code automatically consumes 21 percent more space.

Program objects, when recompiled, will be larger because you are creating ILE programs. Although you might not use any ILE features yet, they are still ILE programs. You will recoup some, if not all, of this space as your shop re-engineers programs and implements more ILE features.

Table 3.1: Comparison of Space Increases

	Base Size	CVTRPGSRC Size	% Inc.	CVTILERPG Size	% Inc.	CVTRPGLCL Size	% Inc.
Library	1073152	1282048	19.47	1310720	22.14	1310720	22.14
ALL001A	159744	204800	28.21	208896	30.77	208896	30.77
ALL002A	143360	188416	31.43	188416	31.43	188416	31.43
ALL002B	172032	217088	26.19	221184	28.57	221184	28.57
ALL003A	131072	155648	18.75	159744	21.88	159744	21.88

Table 3.1 shows a comparison of size increases using the different conversion options. Still, the size of the increase really depends on what the program does. As you can see from the table, the increases vary from 18 percent to 31 percent.

IN CONCLUSION

The conversion process is a simple one. The system-supplied conversion tool—the best place to start—offers more than enough to get you on the road to RPG IV. Take a couple of your programs and give it a try. Fifteen minutes later, you'll wonder what all the fuss was about. If your more adventurous side begs for an enhanced version that incorporates more RPG IV features, you can invest in a third-party tool, write your own conversion program, or do both.

But, when all is said and done, there is only a certain amount that conversion can do. To start making use of the new features, you have to consider re-engineering. You can begin by looking more closely at some of these new features in the coming chapters.