CvtRPGIV

By

2809 S. 160th St. Ste. #401
Omaha, NE 68130

Telephone: (402) 697 - 7575
Toll free: (800) 228 - 6318

Web: **http://www.prodatacomputer.com**

Email Sales: sales@prodatacomputer.com
Email Tech Support: help@prodatacomputer.com

CVTRPGIV is a software utility created to enhance your conversion to RPG IV from earlier RPG versions. It provides easier to read and more native RPG IV source code than the standard conversion utility provided by IBM.

CVTRPGIV has been used to successfully convert thousands of RPG source members with 99.9% accuracy. It can be used on any IBM AS/400 or AS/400 RISC model running OS/400 Release 3, Version 1 or later with ILE RPG/400 installed.

CVTRPGIV is a conversion tool only. No attempt is made to correct or change program logic or deficiencies. Only operations that can be safely converted will be.

Operation codes and new values will be converted to mixed case to highlight changed source statements and expression operations (EQ, NE, GT, LT, GE and LE) will be converted to new RPG IV equivalent expression operators (=, <>, >, <, >=, <=).

CVTRPGIV has the same basic restrictions as IBM's CVTRPGSRC command. The FROM library, file and member (if specified) must exist. The TO library and file must exist and the user must have authority to add members to them. The TO member must not exist. The TO file (default=QRPGLESRC) should be created with a record length of at least 112 instead of the default of 92 or some comments will be lost.

CVTRPGIV converts single, generic* or *ALL source members for a given file source.

CVTRPGIV provides two conversion options: *BASIC and *ADVANCED. *BASIC performs the conversions as detailed on the following pages. *ADVANCED can be utilized to improve code readability with additional conversions. With *ADVANCED, all internal program operation codes (MOVE, ELSE, ENDIF, etc.) are converted to mixed case (Move, Else, Endif, etc.), while external operation codes (CALL, READ, CHAIN, etc.) remain in upper case. Additionally, certain special values (*ON, *OFF, *BLANKS, etc.) are converted to mixed case (*On, *Off, *Blanks, etc.).

It is the user's responsibility to fully test all converted source code before using it in a production environment.

NOTE: When using **CVTRPGIV** in the demo mode, you will see a message that you are in a 5 minute delay period when converting. When the product is purchased, this function will be disabled.

USING CVTRPGIV

CVTPRGIV may be used interactively or in batch. When performing mass conversions, better performance is usually seen by submitting jobs to batch.

1. To use interactively, enter to following commands.

- ADDLIBLE LIB(CVTRPG)
- Key CVTRPGIV and press F4 and the following screen will be displayed. Pressing F10 will bring up additional parameters on the bottom half of the screen.

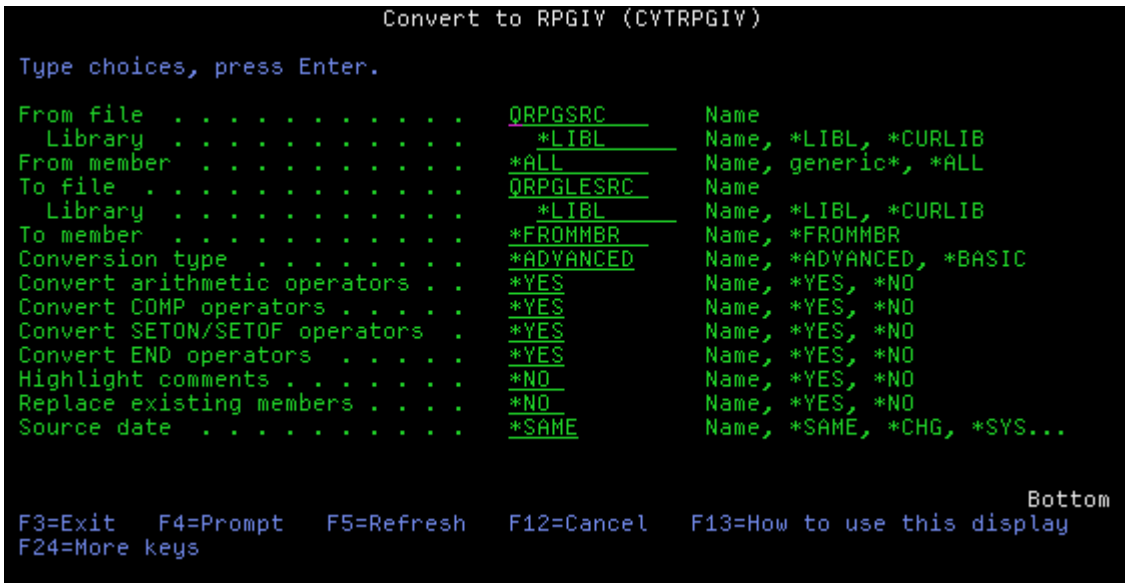

Figure 1

- Enter appropriate values for the following parameters:

From file - Specifies the name of the source file that contains the RPG source code to be converted. This is a required parameter; QRPGSRC is the default name.

QPRGSRC - The default source file and contains the source member(s) to be converted.

Source file name - Enter the name of the source file that contains the source to be converted.

Library - Specifies the name of the library where the source file is contained.

***LIBL** - The system searches the library list to find the library where the source file is stored.

***CURLIB** - The current library is used to find the source file. If you have not specified a current library, then the library QGPL is used.

Library name - Enter the name of the library where the source file is stored.

From member - Specifies the name(s) of the source member(s) to be converted. This is a required parameter and there is no default name. The valid source member types of source members to be converted are RPG, RPT, RPG38, RPT38 and SQLRPG. The convert to RPGLE command does not support RPG36, RPT36 and other non-RPG source member types.

Source file name - Enter the name of the source member to be converted.

***ALL** - The command converts all members in the source file specified.

Generic* - Enter the generic name of members having the same prefix in their names followed by a '*' (asterisk). The command converts all the members having the generic

name in the source file specified. For example, specifying FROMMBR(PR*) will result in the conversion of all members whose names begin with 'PR'.

To file - Specifies the name of the source file that contains the converted source members. The source file must exist and should have a record length of 112 characters: 12 for the sequence number and date, 80 for the code and 20 for the comments.

QRPGLESRC - The default source file QRPGLESRC contains the converted source members.

Source file name - Enter the name of the converted source file that contains the converted source member(s).

Library - Specifies the name of the library that contains the converted source members.

***LIBL** - The system searches the library list to find the library where the converted source file is stored.

***CURLIB** - The current library is used to find the converted source file. If you have not specified a current library, then the library QGPL is used.

Library name - Enter the name of the library where the converted source file is stored.

To member - Specifies the name(s) of the converted source member(s) in the converted source file. If the value specified on the FROMMBR parameter is *ALL or generic*, then TOMBR must be equal to *FROMMBR.

***FROMMBR** - The member name specified in the FROMMBR parameter is used as the converted source member name. If FROMMBR(*ALL) is specified, then all source members in the FROMFILE are converted. The converted source members will have the same names as those of the original source member. If a generic name is specified in the FROMMBR parameter, then all source members having the same prefix in their names are converted. The converted source members will have the same names as those of the original generic source members.

Source file member name - Enter the name of the converted source member. If the member does not exist, it will be created.

Conversion Type - Specifies the type of conversion to perform. Valid options are *BASIC or *ADVANCED.

***BASIC** - Performs basic conversions as outlined in the following documentation.

***ADVANCED** - In addition to *BASIC conversion, *ADVANCED improves code readability by converting all internal program operation codes (MOVE, ELSE, ENDIF, etc.) to mixed case (Move, Else, Endif, etc.). External operation codes (CALL, READ, CHAIN, etc.) remain unchanged. Additionally, certain special values (*ON, *OFF, *BLANKS, etc.) are converted to mixed case (*On, *Off, *Blanks, etc.).

Convert Arithmetic Operators (ARITH) - Specifies whether or not arithmetic operators are converted to EVAL statements. Arithmetic operators that will be converted include: ADD, Z-ADD, SUB, Z-SUB, MULT and DIV.

***YES** - Convert arithmetic operators.

***NO** - Do not convert arithmetic operators.

Convert COMP Operators (COMP) - Specifies whether or not COMP operators are converted to EVAL statements.

***YES** - Convert compare operators.

***NO** - Do not convert compare operators.

Convert SETON/SETOFF Operators(SET) - Specifies whether or not SETON/SETOFF operators are converted to EVAL statements.

***YES** - Convert SETON/SETOFF operators.

***NO** - Do not convert SETON/SETOFF operators.

Convert END Operators (END) - Specifies whether or not END operators are converted to Endif, Enddo, Endcs and Endsl statements. Selected END operators will be converted only if no errors or mismatches are encountered. If any errors are encountered, this option will NOT be performed.

***YES** - Convert END operators.

***NO** - Do not convert END operators.

Highlight Comments - **CVTRPGIV** will highlight comment lines if specified to do so.

***YES** - Will highlight comments.

***NO** - Will not highlight comments.

Replace Existing Members - Specifies whether or not existing TO members are to be replaced.

***YES** - Replace existing members.

***NO** - Do not replace existing members.

Source Date - **CVTRPGIV** will either retain source line dates, set changed, new dates to the current system date, set all dates to the current system date or zero all source dates.

***SAME** - Will retain source date.

***CHG** - Set new dates to the current system date.

***SYS** - Set all dates to the current system date.

***ZERO** - Set source dates to zero.

2. To submit to batch, enter the following command:

- SBMJOB
- The following screen will display.

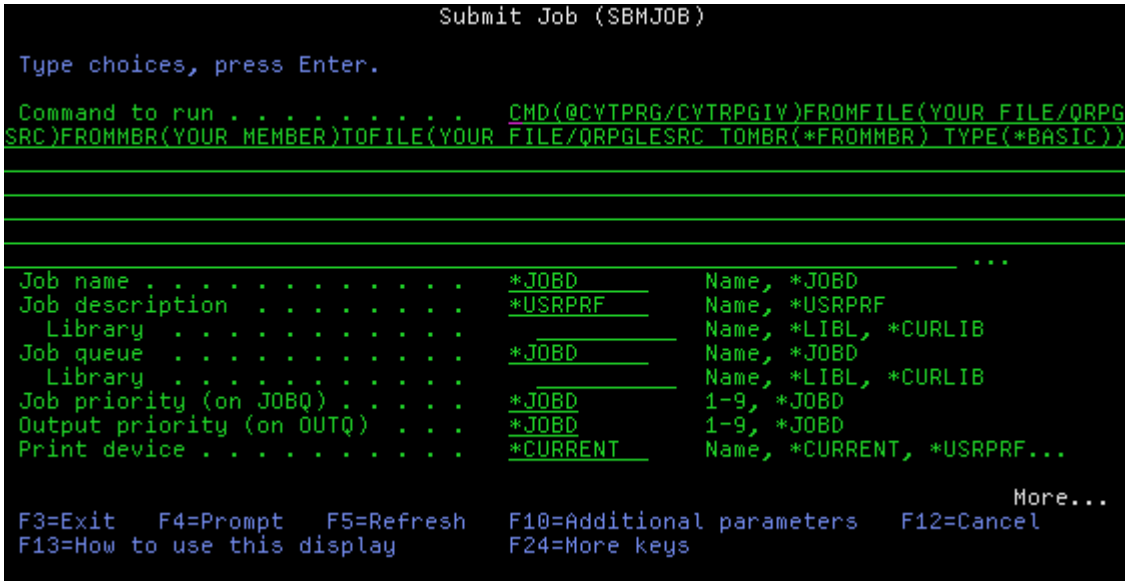

Figure 2

- Be sure that library CVTRPG is included in the submitting job's description or qualify the command library as shown above.
- Enter appropriate values for from and to libraries / files / members.

The following pages will show examples of the different conversions that **CVTRPGIV** will perform to your RPG code.

DEFINED CALCULATION FIELDS

CVTRPGIVconverts defined calculations fields to new D specifications.

The following statements would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                   MOVE      *ZEROS                    FLDN6     9 2
C                   MOVE      *ZEROS                    FLDN7     2 0
C                   MOVE      *BLANKS                   FLDA1     1
C                   MOVE      *ALL'9'                   FLDN6     9 2
C     *LIKE         DEFN      FLDA1                     FLDA8
```

To:

```
DName++++++++++ETDsFrom+++T0/L+++IDc.Keywords++++++++++++++++++++++
D   flda1                 S                   1
D   fldn6                 S                   9  2
D   fldn7                 S                   2  0
D   flda8                 S                       Like(FLDA1)
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                         MOVE    *ZEROS          FLDN6
C                         MOVE    *ZEROS          FLDN7
C                         MOVE    *BLANKS         FLDA1
C                         MOVE    *ALL'9'         FLDN6
*   ***   **********      ***     ***         * *  ***         ****   * **  ***   ***
```

All new D specifications will be sorted alphabetically.

Numeric fields will default to Packed.

Fields defined multiple times (such as FLDN6 above) will only have one D specification statement created.

Calculation specifications with result field lengths will not be deleted, but the length and decimal position values will be removed.

*LIKE/DEFINE calculation statements will be deleted after conversion to D specifications.

IFXX, WHXX AND DOXXX OPERATIONS

The following IFXX statements would convert from:

```
 CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 C       FLDA1             IFEQ            FLDA4
 C       FLDA2             OREQ            FLDA4
 C       FLDA3             OREQ            FLDA4
 C       FLDA1             IFGT            FLDA4
```

To:

```
 CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++
 C           If      FLDA1=FLDA4  or
 C                   FLDA2=FLDA4  or
 C                   FLDA3=FLDA4
 C           If      FLDA1>FLDA4
```

The following WHXX statements would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C       FLDA1                WHEQ              FLDA2
C       FLDA1                WHEN              FLDA2
```

To:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
C         When               FLDA1=FLDA2
C         When               FLDA1<>FLDA2
```

The following DOXX statements would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C       FLDN1                DOWGT               FLDN2
C       *IN99                ANDEQ*              OFF
C       FLDN1                DOUEQ               FLDN2
```

To:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
C                            Dow      FLDN1>FLDA2 and
C                                     *IN99=OFF
C                            Dou      FLDN1=FLDN2
*   ***    **********        ***      ***     *  *  ***    ****     *  **   ***    ***
```

ARITHMETIC OPERATIONS

The following ADD statements would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C       FLDN1                ADD               FLDN3
C                            ADD               FLDN3
```

To:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
C         Eval               FLDN3=FLDN1 + FLDN2
C         Eval               FLDN3=FLDN3 + FLDN4
```

The following SUB statements would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C      FLDN1          SUB        FLDN2          FLDN3
C                     SUB        FLDN2          FLDN3
```

To:

```
CL0N01Factor1_++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++
C           Eval       FLDN3=FLDN1 - FLDN2
C           Eval       FLDN3=FLDN3 - FLDN2
```

The following Z-ADD statements would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C           Z-ADD          *ZEROS                FLDN3
C           Z-ADD          999                   FLDN3
```

To:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++
C           Eval       FLDN3=*ZEROS
C           Eval       FLDN3=999
```

The following Z-SUB statements would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C           Z-SUB          FLDN2                 FLDN3
C           Z-SUB          1000                  FLDN3
```

To:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++
C           Eval       FLDN3=0 - FLDN2
C           Eval       FLDN3=0 - 1000
```

The following MULT statements would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C      FLDN1          MULT       FLDN2          FLDN3
C      FLDN1          MULT       .5             FLDN3
```

To:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
C           Eval            FLDN3=FLDN1 * FLDN2
C           Eval            FLDN3=FLDN1 * .5
```

The following DIV statements would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C     FLDN1               DIV         FLDN2                FLDN3
C     FLDN1               DIV         15                   FLDN3
```

To:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
C           Eval            FLDN3=FLDN1 / FLDN2
C           Eval            FLDN3=FLDN1 / 15
```

The EVAL operation has no equivalent to the DIV / MVR operation combination, so any DIV operation followed by a MVR will not be converted.

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C     FLDN8               DIV         FLDN4                FLDN6
C                         MVR                              FLDN3
```

Arithmetic operations will not be converted if resulting indicators are present.

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                       ADD     FLDN2          FLDN3              3231
C         FLDN4         SUB     FLDN5          FLDN3                   30
*   ***   **********    ***     ***        * * ***        **  * * * *      ***
```

MOVE/MOVEL OPERATIONS

CVTRPGIV converts MOVE and MOVEL operation codes to Eval operations when *BLANKS or *ZEROS are specified in factor 2.

The following MOVE/MOVEL operations would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C           MOVE            *BLANKS                  FLDA1
C           MOVE            *ZEROS                   FLDN1
C           MOVE            *BLANKS                  FLDA1
C           MOVE            *ZEROS                   FLDN1
```

To:

```
 CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
 C           Eval            FLDA1=*Blanks
 C           Eval            FLDN1=*Zeros
 C           Eval            FLDA1=*Blanks
 C           Eval            FLDN1=*Zeros
```

CVTRPGIV converts MOVE and MOVEL operation codes to Eval operations when '1', '0', *ON or *OFF are specified in factor 2 and the result field is a single indicator.

The following MOVE/MOVEL operations would convert from:

```
 CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 C           MOVE              '1'                *IN10
 C           MOVE              '0'                *IN11
 C           MOVE              *ON                *IN10
 C           MOVE              *OFF               *IN11
```

To:

```
 CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
 C             Eval              *IN10=*On
 C             Eval              *IN11=*Off
 C             Eval              *IN10=*On
 C             Eval              *IN11=*Off
```

MOVE and MOVEL operations will not be converted if resulting indicators are present.

```
 CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 C                         MOVE  FLDN1               FLDN2                  12   11
 * *** **************   ***     *************   * * ***      **** *** **** *** ***
```

SETON, SETOFF AND COMP OPERATIONS

CVTRPGIV converts SETON, SETOFF and COMP operation codes to Eval operations.

The following SETON statement would convert from:

```
 CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
 C             SETON                                          1210
```

To:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
C           Eval                *IN12=*On
C           Eval                *IN10=*On
```

The following SETOFF statement would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C             SETOFF                                        22          21
```

To:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
C           Eval                *IN22=*Off
C           Eval                *IN20=*Off
```

The following COMP statement would convert from:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C     FLDN1           COMP        FLDN2                   3230        31
C     FLDN1           COMP        FLDN2                   3232
```

To:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
C           Eval            *IN32=FLDN1 > FLDN2
C           Eval            *IN30=FLDN1 < FLDN2
C           Eval            *IN31=FLDN1 = FLDN2
C           Eval            *IN32=FLDN1 >= FLDN2
```

SETON, SETOFF and COMP operations with continuation statements (AN / OR in positions 7 & 8) will not be converted if resulting indicators are present.

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C N50
C AN60                  SETON                                   12    11
C 99
C OR97  FLDN1           COMP    FLDN2                           3230  31
*  ***    **************  ***     ***     * * ***********  *** *****  ** ***   ***
```

MISCELLANEOUS

CVTRPGIV will not correct any pre-existing problems or ambiguous code. The old adage, "Garbage In, Garbage Out" still applies.

CVTRPGIV can be used on existing RPG IV source members to make use of the above conversion features, but you must be very careful when doing this. RPG IV is still a column oriented language and **CVTRPGIV** is looking for specific values in specific positions. Caution is advised when using **CVTRPGIV** on previously converted RPG IV source members if modifications have been made after conversion.

The Eval operation requires the type of expression be the same as the type of the result. Therefore, if *BLANKS were moved into a numeric field, the converted Eval statement will produce a COMPILER error and must be corrected.

The Eval operation is very particular regarding field lengths. When a numeric result is not large enough to contain the expression, a RUNTIME error is produced. If this occurs, either increase the result field size or decrease the field sizes of the Eval expression. If this is not possible, change the Eval statement back to its pre-conversion value.

Since numeric field definitions default to packed, pay particular attention to any numeric parameters. A RUNTIME error will occur if parameters do not match the calling or called program.

CVTRPGIV performs a statement by statement conversion. You may wish to review your converted source to take full advantage of new features.

For example:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++++++++++++++
C           Eval        FLDN5 = FLDN1 + FLDN2
C           Eval        FLDN5 = FLDN5 + FLDN3
C           Eval        FLDN5 = FLDN5 + FLDN4
```

Could be better expressed by the following:

```
C                     Eval FLDN5 = FLDN1 + FLDN24 + FLDN3 + FLDN4
* *** ************** ***  *** * * ***********  *** ***** ** *** ***
```

Note: All samples included in this documentation are shown in RPG IV format for easier comparison.