If you are new to free-format RPG and deal with any kind of character or numeric date data, you have no doubt been frustrated trying to figure out how to convert data from one format to another. Or perhaps you are using date data types in newer databases and are still dealing with legacy date data types in some other format. Or even yet, maybe you would like to be able to use some of the cool date bifs such as *%years or %months %DAYS, %DIFF and so on.. date data types*. Whatever the reason, I was tired of trying to remember how to convert dates.

Below is a program with 72 examples of date data conversions from many different formats and data types. Feel free to print and keep as a 'crib-sheet' or load and compile on your system and you can use to debug or add you own that may not be listed.

Two notes:

Your region my use different separators...in the example listed, I use the American standard slash ('/') for all date separators, not the '-' Hyphen.

Some of the functions listed will *only work properly* ***on OS/400 V5R2 and higher. Enjoy!***

Main D-SPECS EN-These fields are used in the Coded examples below:

```
H option(*nodebugio)

D @charA          S                8    inz('04/12/01')
D @charB          S               10    inz('12/02/2004')
D @charC          S                8    inz('12/03/04')

D @dateA          S                 d   inz(D'2004-12-04')

D @numA           S                6  0 inz(041205)
D @numB           S                7  0 inz(1041206)   DPS PRIMARY DATE FORMAT.
D @numC           S                8  0 inz(20041207)
D @numD           S                6  0 inz(120804)
D @numE           S                8  0 inz(12092004)
```

```
/free
```

// character to character... conversion

```
@charB = %char(%date(@charA:*ymd/):*usa/);          // 'yy/mm/dd'     to  'mm/dd/ccyy'
@charC = %char(%date(@charA:*ymd/):*mdy/);          // 'yy/mm/dd'     to  'mm/dd/yy'
@charA = %char(%date(@charB:*usa/):*ymd/);          // 'mm/dd/ccyy'   to  'yy/mm/dd'
@charC = %char(%date(@charB:*usa/):*mdy/);          // 'mm/dd/ccyy'   to  'mm/dd/yy'
@charA = %char(%date(@charC:*mdy/):*ymd/);          // 'mm/dd/yy'     to  'yy/mm/dd'
@charB = %char(%date(@charC:*mdy/):*usa/);          // 'mm/dd/yy'     to  'mm/dd/ccyy'
```

// character to date... conversion

```
@dateA = %date(@charA:*ymd/);                       // 'yy/mm/dd'     to  D'ccyy-mm-dd'
@dateA = %date(@charB:*usa/);                       // 'mm/dd/ccyy'   to  D'ccyy-mm-dd'
@dateA = %date(@charC:*mdy/);                       // 'mm/dd/yy'     to  D'ccyy-mm-dd'
```

// character to numeric... conversion

```
@numA = %dec(%char(%date(@charA:*ymd/):*ymd0):6:0);       // 'yy/mm/dd'     to  yymmdd
@numB = %dec(%char(%date(@charA:*ymd/):*cymd0):7:0);      // 'yy/mm/dd'     to  cyymmdd
@numC = %dec(%char(%date(@charA:*ymd/):*iso0):7:0);       // 'yy/mm/dd'     to  ccyymmdd
@numD = %dec(%char(%date(@charA:*ymd/):*mdy0):7:0);       // 'yy/mm/dd'     to  mmddyy
@numE = %dec(%char(%date(@charA:*ymd/):*usa0):7:0);       // 'yy/mm/dd'     to  mmddyyyy
@numA = %dec(%char(%date(@charB:*usa/):*ymd0):6:0);       // 'mm/dd/ccyy'   to  yymmdd
@numB = %dec(%char(%date(@charB:*usa/):*cymd0):7:0);      // 'mm/dd/ccyy'   to  cyymmdd
@numC = %dec(%char(%date(@charB:*usa/):*iso0):8:0);       // 'mm/dd/ccyy'   to  ccyymmdd
@numD = %dec(%char(%date(@charB:*usa/):*mdy0):7:0);       // 'mm/dd/ccyy'   to  mmddyy
@numE = %dec(%char(%date(@charB:*usa/):*usa0):7:0);       // 'mm/dd/ccyy'   to  mmddyyyy
@numA = %dec(%char(%date(@charC:*mdy/):*ymd0):6:0);       // 'mm/dd/yy'     to  yymmdd
@numB = %dec(%char(%date(@charC:*mdy/):*cymd0):7:0);      // 'mm/dd/yy'     to  cyymmdd
@numC = %dec(%char(%date(@charC:*mdy/):*iso0):7:0);       // 'mm/dd/yy'     to  ccyymmdd
@numD = %dec(%char(%date(@charC:*mdy/):*mdy0):7:0);       // 'mm/dd/yy'     to  mmddyy
@numE = %dec(%char(%date(@charC:*mdy/):*usa0):7:0);       // 'mm/dd/yy'     to  mmddyyyy
```

// date to character... conversion

```
@charA = %char(@dateA:*ymd/);                    // D'ccyy-mm-dd'  to  'yy/mm/dd'
@charB = %char(@dateA:*usa/);                    // D'ccyy-mm-dd'  to  'mm/dd/ccyy' -> *USA
@charC = %char(@dateA:*mdy/);                    // D'ccyy-mm-dd'  to  'mm/dd/yy'   -> *MDY
```

// date to numeric... conversion

```
@numA = %dec(%char(@dateA:*ymd/) :6:0);          //iso→ D'ccyy-mm-dd'  to  yymmdd
@numB = %dec(%char(@dateA:*cymd/):7:0);  // = iso→ D'ccyy-mm-dd'  to  cyymmdd
@numC = %dec(%char(@dateA:*iso-) :8:0);          //iso→ D'ccyy-mm-dd'  to  ccyymmdd
@numD = %dec(%char(@dateA:*mdy/) :6:0);          //iso→ D'ccyy-mm-dd'  to  mmddyy
@numE = %dec(%char(@dateA:*usa/) :8:0);          //iso→ D'ccyy-mm-dd'  to  mmddccyy
```

// numeric to character... conversion

```
@charA = %char(%date(@numA:*ymd):*ymd/);         // yymmdd          to  'yy/mm/dd'
@charB = %char(%date(@numA:*ymd):*usa/);         // yymmdd          to  'mm/dd/ccyy'


@charC = %char(%date(@numA:*ymd) :*mdy/);        // yymmdd              to  'mm/dd/yy'
@charA = %char(%date(@numB:*cymd):*ymd/);        // cyymmdd             to  'yy/mm/dd'
@charB = %char(%date(@numB:*cymd):*usa/);        // cyymmdd             to  'mm/dd/ccyy'
@charC = %char(%date(@numB:*cymd):*mdy/);        // cyymmdd             to  'mm/dd/yy'

@charA = %char(%date(@numC:*iso) :*ymd/);        // D'ccyy-mm-dd'       to  'yy/mm/dd'
@charB = %char(%date(@numC:*iso) :*usa/);        // D'ccyy-mm-dd'       to  'mm/dd/ccyy'
@charC = %char(%date(@numC:*iso) :*mdy/);        // D'ccyy-mm-dd'       to  'mm/dd/yy'

@charA = %char(%date(@numD:*mdy) :*ymd/);        // mmddyy -6,0         to  'yy/mm/dd'
@charB = %char(%date(@numD:*mdy) :*usa/);        // mmddyy -6,0         to  'mm/dd/ccyy'
@charC = %char(%date(@numD:*mdy) :*mdy/);        // mmddyy -6,0         to  'mm/dd/yy'

@charA = %char(%date(@numE:*usa) :*ymd/);        // mmddccyy  -8,0      to  'yy/mm/dd'
@charB = %char(%date(@numE:*usa) :*usa/);        // mmddccyy  -8,0      to  'mm/dd/ccyy'
@charC = %char(%date(@numE:*usa) :*mdy/);        // mmddccyy  -8,0      to  'mm/dd/yy'
```

// numeric to date... conversion

```
@dateA = %date(@numA:*ymd);                    // yymmdd          to  D'ccyy-mm-dd'
@dateA = %date(@numB:*cymd);                   // cyymmdd         to  D'ccyy-mm-dd'
@dateA = %date(@numC:*iso);                    // ccyymmdd        to  D'ccyy-mm-dd'
@dateA = %date(@numD:*mdy);                    // mmddyy          to  D'ccyy-mm-dd'
@dateA = %date(@numE:*usa);                    // mmddccyy        to  D'ccyy-mm-dd'
@dateA = %date(@numD:*mdy);                    // mmddyy          to  D'ccyy-mm-dd'
@dateA = %date(@numE:*usa);                    // mmddccyy        to  D'ccyy-mm-dd'
```

_// numeric to numeric... conversion

```
@numB = %dec(%char(%date(@numA:*ymd):*cymd0):7:0);      // yymmdd          to  cyymmdd
@numC = %dec(%char(%date(@numA:*ymd):*iso0):8:0);       // yymmdd          to  ccyymmdd


@numD = %dec(%char(%date(@numA:*ymd):*mdy0):6:0);       // yymmdd          to  mmddyy
@numE = %dec(%char(%date(@numA:*ymd):*usa0):8:0);       // yymmdd          to  mmddccyy

@numA = %dec(%char(%date(@numB:*cymd):*ymd0):6:0);      // cyymmdd         to  yymmdd
@numC = %dec(%char(%date(@numB:*cymd):*iso0):8:0);      // cyymmdd         to  ccyymmdd
@numD = %dec(%char(%date(@numB:*cymd):*mdy0):6:0);      // cyymmdd         to  mmddyy
@numE = %dec(%char(%date(@numB:*cymd):*usa0):8:0);      // cyymmdd         to  mmddccyy

@numA = %dec(%char(%date(@numC:*iso):*ymd0):6:0);       // ccyymmdd        to  yymmdd
@numB = %dec(%char(%date(@numC:*iso):*cymd0):7:0);      // ccyymmdd        to  cyymmdd
@numD = %dec(%char(%date(@numC:*iso):*mdy0):6:0);       // ccyymmdd        to  mmddyy
@numE = %dec(%char(%date(@numC:*iso):*usa0):8:0);       // ccyymmdd        to  mmddccyy

@numA = %dec(%char(%date(@numD:*mdy):*ymd0):6:0);       // mmddyy          to  yymmdd
@numB = %dec(%char(%date(@numD:*mdy):*cymd0):7:0);      // mmddyy          to  cyymmdd
@numC = %dec(%char(%date(@numD:*mdy):*iso0):8:0);       // mmddyy          to  ccyymmdd
@numE = %dec(%char(%date(@numD:*mdy):*usa0):8:0);       // mmddyy          to  mmddccyy

@numA = %dec(%char(%date(@numE:*usa):*ymd0):6:0);       // mmddccyy        to  yymmdd
@numB = %dec(%char(%date(@numE:*usa):*cymd0):7:0);      // mmddccyy        to  cyymmdd
@numC = %dec(%char(%date(@numE:*usa):*iso0):8:0);       // mmddccyy        to  ccyymmdd
@numD = %dec(%char(%date(@numE:*usa):*mdy0):6:0);       // mmddccyy        to  mmddyy
```

STANDARD - WORK DATE FLDS: SIMPLY DEFINED:

```
      * ////////////////////
ls004 * Std. ISO Dates and Times Defined:
ls004 *////////////////////
ls004d CurIsoDate      S               D   datfmt(*ISO) inz(*SYS)
ls004d CurUsaDate      S               D   datfmt(*USA) inz(*SYS)
ls004d CurMdyDate      S               D   datfmt(*MDY) inz(*SYS)

ls004d CurDate80       S              8S 0 inz(*zeros)

ls004d Wk_IsoDte       s               D   datfmt(*ISO) inz(*loval)
ls004d Wk_UsaDte       s               D   datfmt(*USA) inz(*loval)

ls004d Wk_IsoDtCh      s             10a
ls004d Wk_UsaDtCh      s             10a

ls004d CurTime         s               T   timfmt(*ISO) inz(*SYS)
ls004d CurTime60       S              6  0
ls004d CurTime8a       S              8a
```

==========

SAMPLE CODE:

==========

1.

2. The following code in Setheader() and CQCODEUPD() needs to be changed...

```
ls004c                    Eval      AddDRV=drcode
ls004c                    Eval      AddCode=*blanks
ls004c
ls004c*,For 'S'afety related codes, 'T' for training related.
ls004c* Date conversion related code:
ls004c                    Eval      AddCodeTyp = ' '
ls004c                    Eval      AddEffDte=%char(CurUsaDate) ... (Use MDY DATES instead...)  '11/19/09'
ls004c                    Eval      AddExpDt=%char(Wk UsaDte) ...
```

```
============================

ALSO,

finish this code THAT NEEDS TO BE FINISHED - ADD THE  TEST(DE)...

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


ls004p ValDQCodes      b
ls004d ValDQCodes      pi             2a
ls004d Q_DriverCD                     6    value
ls004d
ls004d @Errflag        s              2a   inz(*blanks)
ls004d
ls004d**
ls004d** Begin Date conversion fld's in ValDQCodes()...
     d**
ls004d MDYDATE         ds
ls004d  mdy                     1      8
ls004d  mm                      1      2  0
ls004d  slash1                  3      3    inz('/')
ls004d  dd                      4      5  0
ls004d  slash2                  6      6    inz('/')
ls004d  yy                      7      8  0
ls004d
ls004d MDYDATE#        ds
```

```
ls004d  mdy#                    1       6  0
ls004d  mm#                     1       2  0
ls004d  dd#                     3       4  0
ls004d  yy#                     5       6  0
ls004d
ls004 /Free

           Clear REDMSG;
           Clear @Errflag;

ls004      // ////////////////////////////////////////////////
ls004      //,Test Effective and Exp. Dates to ensure are valid:
ls004      //,Just convert dates, and any errors indicate bad
ls004      //,Date formats...
ls004      // ////////////////////////////////////////////////

              EVAL MDYDATE = AddEffdte ;

             // Move to a 6,0 mdy date fld, then to an *iso Date fld:
              EVAL mm# = mm ;
              EVAL dd# = dd ;
              EVAL yy# = yy ;

              Wk_IsoDte = %date(mdy#:*mdy);             // mmddyy to  D'ccyy-mm-dd'
```

```
              If %Error ;
               Eval @Errflag = 'ER';
               Eval REDMSG   = 'Invalid Effective Date';
               Return @Errflag;
              Endif;

ls004            //Eval @Rtrn_flag=ValDQCodes(drcode);
```

---> Also , use this BIF TO TEST DATES: *TEST(DE) the dates - test code...

he input is taken in *MDY format.

Scenario 1 – When the screen field is of 6 length numeric type. Keep this field as factor 2 and specify its format as *MDY.

Scenario 2 – When screen field is of 8 character type. You want to give user the flexibility to enter '/' also. (Though it's possible by applying some suitable edit code in DDS itself with 6 length numeric field, However, for the sake of completeness... you know.)

Test the date first for *MDY and if error occurs, test it for *MDY0. This is because, if we do not specify explicitly that the date does not have any separator, system will always search for it, if length of the character field is greater than possible (6 in this case).

So, in this case something like this should happen

```
*MDY                            Test(DE)                                W@ScrnFld
                                If              %Error
*MDY0                           Test(DE)                                W@ScrnFld
                                EndIf
                                If              %Error
                                Confirmed invalid date
                                Endif
```

```
                    When        W@Format = '*USA0'
C       *USA0       Test(DE)                W@Date
 *
C                   When        W@Format = '*CYMD'
C       *CYMD       Test(DE)                W@Date
 *
C                   When        W@Format = '*CYMD0'
C       *CYMD0      Test(DE)                W@Date
 *
C                   When        W@Format = '*YMD'
C       *YMD        Test(DE)                W@Date
 *
C                   When        W@Format = '*YMD0'
C       *YMD0       Test(DE)                W@Date
 *
C                   When        W@Format = '*DMY'
```

```
C      *DMY          Test(DE)                  W@Date
 *
C                    When      W@Format = '*DMY0'
C      *DMY0         Test(DE)                  W@Date
C                    EndSl
 *
C                      If        %Error
C      'Invalid'       Dsply
```

One last thing, we should keep in our mind which declaring variables to store acting dates..:

- **The Portion of a Numeric variable from left which exceeds the required length for calculation, is ignored by program.**
- **The Portion of a Character variable from right which exceeds the required length for calculation, is ignored by program.**

And Here is a free table of %bifs below for further study:

http://www.rpgiv.com/kb/built-ins.html

http://search400.techtarget.com/tip/1,289483,sid3_gci958687_mem1,00.html?ShortReg=1&mboxConv=search400_RegActivate_Submit&

NOTE: IBM Seems to skip-ship the RPG IV compiler. So RPG IV in V4R1, V4R3 and V4R5 have no new functionality. The next scheduled upgrade is OS/400 V5R1 in Spring 2001.

Release	*Built-in Function*	*Parameters*	*Return Value Description*
	%ADDR	{variable name }	
	Address of variable		
V5R1	%ALLOC	{memory size }	
	Pointer to the allocated storage.		
V4R4	%CHAR	{graphic, date, time, timestamp, or numeric expression}	

The value in character data type

V5R1 %CHECK {compare-value : data-to-search { : start-position } }

Locates the first position in the searched-data that contains a character not in the list of the characters in the compare value.

V5R1 %CHECKR {compare-value : data-to-search { : start-position } }

Finds last position in the searched-data that contains a character not in the list of the characters in the compare value. (Search begins with the right-most character and proceeds to the left.

V5R1 %DATE { value { : date-format-code }

A date data-type value after converting the "value" to the specified date format. If no value is specified, the current system date is returned.

V5R1 %DAYS {days }

A duration value that can be used in an expression to add a number of days to a date value.

V3R7 %DEC {numeric expression :digits : decpos}

Value in packed numeric format. If digits and decpos are specified the result value is formatted to fit in a variable of the number of digits specified.

V3R7 %DECH {numeric expression : digits : decpos}

Half-adjusted value in packed numeric format. The length and decimal positions.

V3R7 %DECPOS {numeric expression }

Number of decimal digits.

V5R1 %DIFF {start-date : end-date : duration-code}

Calculates the difference between two date fields. The type of difference returned is specified by the duration-code.

V4R4 %DIV {Numerator : Denominator}

Performs integer division and returns the quotient (result) of that division operation.

V3R7 %EDITC {non-float numeric expression : edit code {:*CURSYM | *ASTFILL | currency symbol} }

String representing edited value.

V3R7 %EDITFLT {numeric expression }

Character external display representation of float.

V3R7 %EDITW {non-float numeric expression : edit word }

String representing edited value

%ELEM {array, table, or multiple occurrence data structure name }

Number of elements or occurrences.

V4R2 %EOF {file name}

'1' if the most recent file input operation or write to a subfile (for a particular file, if specified) | ended in an end-of-file or | beginning-of-file condition '0' otherwise.

V4R2 %EQUAL {file name}

'1' if the most recent SETLL (for a particular file, if specified) or LOOKUP operation found an exact match '0' otherwise.

V4R2 %ERROR

'1' if the most recent operation code with extender 'E' specified resulted in an error '0' otherwise.

V3R7 %FLOAT

numeric expression Value in float format.

V4R2 %FOUND {file name}

'1' if the most recent relevant operation (for a particular file, if specified) found a record (CHAIN, DELETE, SETGT, SETLL), an element (LOOKUP), or a match (CHECK, CHECKR, SCAN) '0' otherwise.

V4R4 %GRAPHIC Any character value

Converts character data to double-byte character set value.

V5R1 %HOURS Hours

hours A duration value that can be used in an expression to add a number of hours to a time value.

V3R7 %INT numeric expression

Value in integer format

V3R7 %INTH numeric expression

Half-adjusted value in integer format

V3R7 %LEN any expression

1. Returns the length of a variable or literal value, or the current length of a varying length field.
2. When used on the left side of the equal sign, sets the length of a varying length field.

V5R1 %LOOKUPxx search-data : array { : start-index { : elements to search }}

An array index of the element in the array where the search-data is located.

V5R1 %TLOOKUPxx search-data : searched-table { : alternate-table }

*ON if the search is successful, otherwise *OFF. (NOTE: The indexes of the searched-table and alternate-table are set to the index of the search-data if *ON is returned.)

V5R1 %MINUTES minutes

A duration value that can be used in an expression to add a number of minutes to a time value.

V5R1 %MONTHS months

A duration value that can be used in an expression to add a number of months to a date value.

V5R1 %MSECONDS milliseconds

A duration value that can be used in an expression to add a number of milliseconds to a time value.

V3R7 %NULLIND null-capable field name

Value in indicator format representing the null indicator setting for the null-capable field.

V5R1 %OCCUR data-structure

The current occurrence of the data structure, or sets the current occurrence of the data structure.

V4R2 %OPEN file name

'1' if the specified file is open '0' if the specified file is closed. Consider this built-in to be an 'Is this file open?" operation.

%PADDR procedure name

Address of procedure

V3R6 %PARMS Number of parameters passed to procedure

V5R1 %REALLOC {pointer : new-size }

Pointer to the allocated storage.

V4R4 %REM Numerator : Denominator

Performs integer division and returns the remainder from the division operation.

V4R2 %REPLACE

replacement string: source string {:start position {:source length to replace}}

String produced by inserting replacement string into source string, starting at start position and replacing the specified number of characters.

V3R7 %SCAN search argument : string to be searched {:start position}

First position of search argument in string or zero, if not found.

V5R1 %SECONDS seconds

A duration value that can be used in an expression to add a number of seconds to a time value.

V5R1 %SHTDN

*ON if the job is being shut down (e.g., when the PWRDWNSYS command is issued) otherwise *OFF is returned.

%SIZE variable, data structure, array, or literal {: *ALL}

Number of bytes used by variable or literal. *ALL returns the number of bytes used by all the elements of the array, or all the occurrences of the data structure.

V5R1 %SQRT expression or value

The square root of the expression or value.

V4R2 %STATUS {file name}

0 if no program or file error occurred since the most recent operation code with extender 'E' specified most recent value set for any program or file status, if an error occurred if a file is specified, the value returned is the most recent status for that file.

V3R7 %STR {pointer{:maximum length}

Characters addressed by pointer argument up to but not including the first x'00'.

V5R1 %SUBDT {date : duration-code }

The extracted component of the date value. (The functional equivalent of the EXTRCT operation code.)

%**SUBST** { string : start {:length}

Substring value. If length is not specified, the substring begins with start and continues through the end of the string.

V5R1 %**THIS** Used for Java integration.

Returns an Object reference.

V5R1 %**TIME** { value { : time-format-code }

A time data-type value after converting the "value" to the specified time format. If no value is specified, the current system time is returned.

V5R1 %**TIMESTAMP** {value { : *ISO | *ISO0 }

A timestamp data-type value with or without separators.

%**TRIM** string

String with left and right blanks trimmed (removed)

%**TRIML** string

String with left blanks trimmed

%**TRIMR** string

String with right blanks trimmed

V4R4 %**UCS2** Any character value

Returns a varying length value.

V4R2 %**UNS** numeric expression

Value in unsigned format

V4R2 %**UNSH** numeric expression

Half-adjusted value in unsigned format

V5R1 %**XLATE** from-table : to-table : string-to-convert { : starting-position }

The converted string is returned.

V4R4 %**XFOOT** Array name

Cross foots (totals) all the elements in an arr
