SYNCOPATION SOFTWARE

TECH NOTE

Problems that arise when converting an Excel spreadsheet to a DPL program can often be traced to the use of spreadsheet functions that are either unsupported by DPL or are supported only in a certain functional form. When using a large, complicated spreadsheet model, the existence and location of these functions in the spreadsheet is sometimes difficult to determine. This memo documents the supported, unsupported and "conditionally" supported spreadsheet functions. The proper form of the conditionally supported functions is also described.

1.1 Excel Functions Supported In DPL (Exhaustive List)

Excel	DPL	Excel	DPL	Excel	DPL
ABS	@abs	LN	@ln	ROUND	@round
AND	Xland	LOG10	@log	ROUNDUP	@roundup
AVERAGE	@avg	MATCH	xlmatch	ROUNDDOWN	@rounddown
CHOOSE	@choose	MAX	@max	ROWS**	@rows
COLUMNS**	@cols	MEDIAN	xlmedian	SLN	@sln
COUNT	@count	MIN	@min	SMALL	xlsmall
DDB**	@ddb	MOD**	xlmod	SQRT	@sqrt
EXP	@exp	NA	@na	STDEV	@std
FALSE	@false	NOT	!	SUM	@sum
FV**	@fv	NPER**	@cterm,@term	SUMPRODUCT*	@sumproduct
HLOOKUP**	@hlookup	NPV	@npv	SYD	@syd
IF**	@if	OR	xlor	TRANSPOSE	xltranspose
INDEX**	xlindex	PI	@pi	TRUE	@true
INT	xlint	PMT**	@pmt	TRUNC**	@int
IRR**	@irr	PV**	@pv	VAR	@var
LARGE	xllarge	RATE**	@rate	VLOOKUP**	@vlookup

*May not convert when the function arguments are array formulas.
**See Table 1.3 and accompanying notes.

1.2 Excel Functions That Are Not Supported In DPL (Not Exhaustive)

All Date Functions	RANK
COS	SIN
COUNTIF	SUMIF
INDIRECT	TREND
ISBLANK	
ISERR	
ISNUMBER	
OFFSET	

1.3 Excel Functions That Are Supported But Can Fail To Convert

Excel	DPL	Excel	DPL
COLUMN***	N/A	MOD	@mod
DDB	@ddb	NPER	@cterm, @term
FV	@fv	PMT	@pmt
HLOOKUP	@hlookup	PV	@pv
IF	@if	RATE	@rate
INDEX	xlindex	ROW***	N/A
IRR	@irr	TRUNC	@int
LOG	@log	VLOOKUP	@vlookup
MATCH	xlmatch		

***May require option "suppress series generation". These functions are translated into index expressions; there are no corresponding functions in DPL.

1.4 Description Of The Conversion Requirements For Functions

Italics indicate an optional functional argument.

1.4.1 Mathematical Spreadsheet Functions

1.4.1.1 LOG10(number), LOG(number, *base*)

LOG10(number) and LOG(number) convert to @log(number).

The LOG function will not convert to DPL code when the optional argument *base* is specified. The *base* of LOG is assumed to be 10 when converting to DPL code.

LOG and LOG10 are not defined when the argument is 0 or negative.

1.4.1.2 MOD(number, divisor)

MOD(number, divisor) converts to @mod(number, divisor).

The divisor cannot be 0.

When number, divisor > 0 or number, divisor < 0, @mod and MOD have the same value. When the number and divisor have opposite signs, however, @mod and MOD give different outputs because they use different rounding conventions.

In Excel, the MOD function is calculated such that 0 < |MOD(number, divisor)| < divisor. The formula for MOD is given by

MOD(number, divisor) = number - divisor*INT(number/divisor)

where INT is an Excel function that rounds its argument to the largest integer smaller than the argument. For example, INT(-9.5) = -10. Note that the Excel INT function uses a different convention for negative numbers than does the DPL @int function. Using this INT function, MOD produces the output

MOD(-7, 3) = 2 and MOD(7, -3) = - 2

In DPL, the @mod function is given by

@mod(number, divisor) = number – divisor*@int(number/divisor)

where @int is a DPL function that takes the integer part of its argument. In contrast to the Excel INT function, @int(-9.5) = -9. Therefore, the @mod functions produces the following output:

@mod(-7, 3) = -1 and @mod(7, -3) = 1

When converting the MOD function to DPL code be careful when you have negative arguments. To be on the safe side, you can replace MOD(number, divisor) with number – divisor*TRUNC(number/divisor), which is equivalent to the @mod function (see TRUNC description).

1.4.1.3 TRUNC(number, num_digits)

TRUNC(number) converts to @int(number).

DPL will not convert TRUNC when the optional argument num_digits is specified. DPL assumes num_digits is 0 when converting.

1.4.2 Financial Spreadsheet Functions

Naming conventions used in this section:

- rate = interest rate
- nper = number of payment periods
- *pmt* = payment
- pv = present value
- *fv* = future value

1.4.2.1 DDB(cost, salvage, life, *period*, *factor*)

DDB(cost, salvage, life, *period*) converts to @ddb(cost, salvage, life, period).

DDB will not convert to @ddb when the optional argument *factor* is specified. The arguments life and *period* should not be 0.

DPL assumes *factor* to be 2 (double-declining balance method).

1.4.2.2 FV(rate, nper, *pmt*, *pv*, *type*)

FV(rate, nper, *pmt*) converts to @fv(-pmt, rate, nper).

FV will not convert when the optional arguments *pv* and *type* are specified.

The argument rate should not be 0.

DPL assumes *type* equals 0 (payment at end of the month).

1.4.2.3 IRR(values, *guess*)

IRR(values, *guess*) converts to @irr(values, guess).

IRR will not convert when the optional argument *guess* is omitted.

1.4.2.4 NPER(rate, *pmt*, pv, *fv*, *type*)

NPER converts to two different DPL functions, depending on which of its arguments are specified:

NPER(rate,, pv, *fv*) converts to @cterm(rate, fv, -pv);

NPER(rate, *pmt*,, fv) converts to @term(-pmt, rate, fv).

NPER will not convert to DPL code when more than three arguments are specified.

No argument in NPER should be 0.

For conversion to @term, DPL assumes *type* equals 0 (payment at end of period). Payment *type* for @cterm doesn't matter.

1.4.2.5 PMT(rate, nper, pv, *fv*, *type*)

PMT(rate, nper, pv) converts to @pmt(-pv, rate, nper).

PMT will not convert when the optional arguments *fv* and *type* are specified.

The arguments rate and nper should not equal 0.

DPL assumes *type* equals 0 (payment at end of period).

1.4.2.6 PV(rate, nper, *pmt*, *fv*, *type*)

PV(rate, nper, *pmt*) converts to @pv(-pmt, rate, nper).

PV will not convert when the optional arguments *fv* and *type* are specified. The argument rate should not equal 0.

DPL assumes *type* equals 0 (payment at end of period).

1.4.2.7 RATE(nper, *pmt*, pv, *fv*, *type*, *guess*)

RATE(nper,, pv, *fv*) converts to @rate(fv, -pv, nper).

RATE will not convert when the optional arguments *pmt*, *type*, or *guess* are specified.

The arguments nper and pv should not equal 0. Payment *type* for @rate doesn't matter.