Dummy data to be written into the Excel sheet:

$$A := \begin{bmatrix} \text{“}A\text{”} \\ 0 \\ 1 \end{bmatrix} \qquad B := \begin{bmatrix} \text{“}B\text{”} \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} \qquad C := \begin{bmatrix} \text{“}C\text{”} & 6 & 7 & 8 \end{bmatrix} \qquad D := \begin{bmatrix} \text{“}D1\text{”} & \text{“}D2\text{”} & \text{“}D3\text{”} \\ 9 & 12 & 15 \\ 10 & 13 & 16 \\ 11 & 14 & 17 \end{bmatrix} \qquad E := \text{“}single\text{”}$$

1) Function *add()*, which is similar to *augment()*, but you can use scalars / matrices of different dimensions. Not used cells will be filled by an selectable value (default: NaN)

```
add(M, v) :=
    empty ← NaN                                   Replace NaN for whatever you want an empty cell to be,
    M ← M                                         e.g. "" will give the same result (in Excel) as NaN (Not a Number)
    cM ← cols(M)
    rM ← rows(M)
    if rows(v) = 0
        v ← [v]
    for r ∈ ORIGIN .. max(ORIGIN + rows(v) − 1, ORIGIN + rM − 1)
        if r > (ORIGIN + rM − 1) ∧ (cM > 0)
            for i ∈ ORIGIN .. ORIGIN + cM − 1
                M_{r,i} ← empty
        for c ∈ ORIGIN .. ORIGIN + cols(v) − 1
            if r < ORIGIN + rows(v)
                M_{r, cM + c} ← v_{r,c}
            else
                M_{r, cM + c} ← empty
    return M
```

1a) One way to create the "augmented" matrix to be written

$$M := 0 \quad M := add(M, A) \quad M := add(M, B) \quad M := add(M, C) \quad M := add(M, D) \quad M := add(M, E)$$

1b) but maybe you find it easier to do it that way using an auxiliary function:

```
new_augm(V) :=
    M ← 0
    for v ∈ V
        M ← add(M, v)
    return M
```

$$M := new\_augm\left([A \;\; B \;\; C \;\; D \;\; E]^{T}\right)$$

$$\begin{bmatrix} \text{“}A\text{”} & \text{“}B\text{”} & \text{“}C\text{”} & 6 & 7 & 8 & \text{“}D1\text{”} & \text{“}D2\text{”} & \text{“}D3\text{”} & \text{“}single\text{”} \end{bmatrix}$$

Mos
full
If y
valu
Ena

add

$$M = \begin{bmatrix} 0 & 2 & NaN & NaN & NaN & NaN & 9 & 12 & 15 & NaN \\ 1 & 3 & NaN & NaN & NaN & NaN & 10 & 13 & 16 & NaN \\ NaN & 4 & NaN & NaN & NaN & NaN & 11 & 14 & 17 & NaN \\ NaN & 5 & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN \end{bmatrix}$$

$file := \text{“}Test1.xlsx\text{”}$

$dummy := \text{WRITEFILE}(file, [NaN])$

$dummy := \text{WRITEEXCEL}(file, M)$

Create a new, empty file. WRITEEXCEL will always append to an existing file.
You have to save the Prime worksheet first.
The Excel file will be created in the same directory your Prime file was stored

$$\text{READEXCEL}(file) = \begin{bmatrix} \text{“}A\text{”} & \text{“}B\text{”} & \text{“}C\text{”} & 6 & 7 & 8 & \text{“}D1\text{”} & \text{“}D2\text{”} & \text{“}D3\text{”} & \text{“}single\text{”} \\ 0 & 2 & NaN & NaN & NaN & NaN & 9 & 12 & 15 & NaN \\ 1 & 3 & NaN & NaN & NaN & NaN & 10 & 13 & 16 & NaN \\ NaN & 4 & NaN & NaN & NaN & NaN & 11 & 14 & 17 & NaN \\ NaN & 5 & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN \end{bmatrix}$$

2) Using the Excel component (empty cells are filled with zeros!)

Eingaben

$excel_{\text{“}A1\text{”}} := A$ $excel_{\text{“}C1\text{”}} := C$ $excel_{\text{“}J1\text{”}} := E$

$excel_{\text{“}B1\text{”}} := B$ $excel_{\text{“}G1\text{”}} := D$

A	B	C	6	7	8	D1	D2	D3	single
0	2					9	12	15	
1	3					10	13	16	
	4					11	14	17	
	5								

Ausgaben

$M2 := excel_{\text{“}A1:J5\text{”}}$

You could doubleclick the table and then save the Excel sheet, or do it as follows

$$M2 = \begin{bmatrix} \text{“}A\text{”} & \text{“}B\text{”} & \text{“}C\text{”} & 6 & 7 & 8 & \text{“}D1\text{”} & \text{“}D2\text{”} & \text{“}D3\text{”} & \text{“}single\text{”} \\ 0 & 2 & 0 & 0 & 0 & 0 & 9 & 12 & 15 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 & 10 & 13 & 16 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 11 & 14 & 17 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Empty cells are filled with zeros that way!
Would make more sense if MC would fill with NaN's.

$file := \text{“}Test2.xlsx\text{”}$ $dummy := \text{WRITEFILE}(file, [NaN])$

$dummy := \text{WRITEEXCEL}(file, M2)$

$$\text{READEXCEL}(file) = \begin{bmatrix} \text{“}A\text{”} & \text{“}B\text{”} & \text{“}C\text{”} & 6 & 7 & 8 & \text{“}D1\text{”} & \text{“}D2\text{”} & \text{“}D3\text{”} & \text{“}single\text{”} \\ 0 & 2 & 0 & 0 & 0 & 0 & 9 & 12 & 15 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 & 10 & 13 & 16 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 11 & 14 & 17 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3) User function *WriteExcel()* in collapsed area

Usage: *WriteExcel(filename, data, startcell)*

filename — a string with the path to the EXCEL file the data should be written to
suffix *".xlsx"* is mandatory
if the file does not exist it is created

data — the data to be written, usually a matrix, but may be a scalar or string, too

startcell — the cell where the upper left corner of the data will be written to
can be either
a string in the usual Excel notation like "F7"
notation in upper- and lowercase is possible,
cells with columns bigger than "Z" like "AD123" are supported, too.
or
a 2-element vector with start column and start row in Mathcad notation
So "F7" is equivalent to stack(5,6) if ORIGIN is set to 0 or to stack(6,7) if ORIGIN is set to 1

The routine is limited to the first sheet of the Excel file.
There is no special error handling implemented.

$$
\begin{aligned}
&WriteExcel(fil, dat, cell) := \\
&\quad \text{if } \operatorname{rows}(dat) = 0 \\
&\quad\quad dat \leftarrow [dat] \\
&\quad \text{if } \operatorname{rows}(cell) = 2 \\
&\quad\quad C \leftarrow cell_{ORIGIN} - \text{ORIGIN} \\
&\quad\quad R \leftarrow cell_{ORIGIN+1} - \text{ORIGIN} \\
&\quad \text{else} \\
&\quad\quad cellv \leftarrow \operatorname{str2vec}(cell) \\
&\quad\quad i \leftarrow 0 \\
&\quad\quad C \leftarrow -1 \\
&\quad\quad \text{while } cellv_i > 64 \\
&\quad\quad\quad \text{if } cellv_i > 90 \\
&\quad\quad\quad\quad cellv_i \leftarrow cellv_i - 32 \\
&\quad\quad\quad C \leftarrow (C+1)\cdot 26 + cellv_i - 65 \\
&\quad\quad\quad i \leftarrow i+1 \\
&\quad\quad R \leftarrow \operatorname{str2num}(\operatorname{substr}(cell, i, 100)) - 1 \\
&\quad r \leftarrow \operatorname{rows}(dat) \\
&\quad c \leftarrow \operatorname{cols}(dat) \\
&\quad range \leftarrow \text{“}A1\text{:”} \\
&\quad \text{if } C + c > 26 \\
&\quad\quad range \leftarrow \operatorname{concat}\left(range, \operatorname{vec2str}\left(\begin{bmatrix} 64 + \operatorname{trunc}\left(\frac{C+c}{26}\right) \\ 64 + \operatorname{mod}(C+c, 26) \end{bmatrix}\right)\right)
\end{aligned}
$$

```
range ← concat(range, vec2str([64 + C + c]))
range ← concat(range, num2str(R + r))
try
    M ← READEXCEL(fil, range)
on error
    M ← WRITEEXCEL(fil, [NaN])
    M ← READEXCEL(fil, range)
for rw ∈ 0 .. r − 1
    for cl ∈ 0 .. c − 1
        M[ORIGIN + R + rw, ORIGIN + C + cl] ← dat[ORIGIN + rw, ORIGIN + cl]
M ← WRITEEXCEL(fil, M)
```

$file := \text{“}Test3.xlsx\text{”}$ $dummy := \text{WRITEFILE}(file, [NaN])$

To stay with the examples above, we can use the new function in that way:

$dummy := WriteExcel(file, A, \text{“}A1\text{”})$

$dummy := WriteExcel(file, B, \text{“}B1\text{”})$

$dummy := WriteExcel(file, C, \text{“}C1\text{”})$

$dummy := WriteExcel(file, D, \text{“}G1\text{”})$

$dummy := WriteExcel(file, E, \text{“}K3\text{”})$ displaced on purpose, change to "J1", but delete the file before worksheet recalc

$$\text{READEXCEL}(file) = \begin{bmatrix} \text{“}A\text{”} & \text{“}B\text{”} & \text{“}C\text{”} & 6 & 7 & 8 & \text{“}D1\text{”} & \text{“}D2\text{”} & \text{“}D3\text{”} & NaN & NaN \\ 0 & 2 & NaN & NaN & NaN & NaN & 9 & 12 & 15 & NaN & NaN \\ 1 & 3 & NaN & NaN & NaN & NaN & 10 & 13 & 16 & NaN & \text{“}single\text{”} \\ NaN & 4 & NaN & NaN & NaN & NaN & 11 & 14 & 17 & NaN & NaN \\ NaN & 5 & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN \end{bmatrix}$$

$A =$

$B =$

$C =$

$D =$

$E =$