選択した文字列の情報をエクセルシートに表示し、編集内容をCADに反映させる

2つのコマンド「OUT-TEXT」、「MOD-TEXT」を作成し、CAD図上で選択したテキストの内容を エクセルシート上で編集後、結果をCADに反映させる例です。

■ やりたいこと

図上にて選択したテキストの情報をエクセルシートに書きだす。
書き出した内容をエクセルシート上で編集する
編集内容を図に反映する。

■ 書き出したい情報

座標
画層
文字の高さ
幅係数
回転角度
位置合せ情報
文字内容

■ エクセル表示イメージ

No.	ハンドル	X座標	Y座標	画層	文字高さ	幅係数	回転角度	位置合せ	文字内容
1	393	1510	3847.5	0	5	1	0	20	OD-PS100
2	394	1510	3837.5	0	5	1	0	20	OD-DS120
3	395	1510	3827.5	0	5	1	0	20	WT-MT180
4	396	1510	3817.5	0	5	1	0	20	SW-3D045
5	397	1510	3807.5	0	5	1	0	20	DC-MS350
6	398	1510	3797.5	0	5	1	0	20	DC-LZ400
7	399	1510	3787.5	0	5	1	0	20	CC-ST200
8	39A	1510	3777.5	0	5	1	0	20	LK-4P180
9	39B	1510	3767.5	0	5	1	0	20	LK-12P360
10	39C	1510	3757.5	0	5	1	0	20	CB-ST2D90
11	39D	1510	3747.5	0	5	1	0	20	CB-ST4D180
12	39E	1510	3737.5	0	5	1	0	20	BK-GL120

■ 操作手順イメージ

① 図上にて選択範囲を指定
② カスタムコマンド「OUT-TEXT」実行（選択範囲が指定されていない場合は、選択が促される。）
③ 選択範囲に含まれるTEXT(文字オブジェクト) の情報がエクセルシートに書きだされる
④ エクセルシート上で、「画層、文字高さ、文字内容」などを編集する
⑤ カスタムコマンド「MODIFY-TEXT」を実行
⑥ エクセルシート上の編集内容が、図上のTEXTに反映される。

■ 使用するAutoCADカスタムコマンドLISPファイル

1．OUT-TEXT.lsp

コマンド **Out-Text** を記述したファイル
選択範囲からTEXT(文字オブジェクト) の情報を取得し、**TextProperties.csv** に書き出す。
1行分の書き出しフォーマットは以下のイメージ

通番, ハンドル, X座標,Y座標,画層,文字高さ,幅係数,回転角度,位置合せ情報,文字内容

書出し後は、VBScriptファイル **csv-to-xlsm.vbs** を起動して終了する。
（VBScript **csv-to-xlsm.vbs** は、Excel **TextPropertiesList.xlsm** を開くと同時に、マクロ**DispTextProperties** を起動する。）
（マクロ**DispTextProperties** は、 **TextProperties.csv** を読んで、シート上にTEXT情報リストを表示する。）

2．MODIFY-TEXT.lsp

コマンド **Modify-Text** 、および、**MODTXT** を記述したファイル

■ Modify-Text

VBScript **xlsm-to-csv.vbs** を起動して終了する。
（VBScript **xlsm-to-csv.vbs**は、Excel **TextPropertiesList.xlsm** マクロ **OUT-CSV** を起動する。）
（マクロ **OUT-CSV** は、 **TextProperties.csv** に編集内容を上書きし、AutoCAD コマンド **MODTXT** を起動する。）

■ MODTXT

TextProperties.csv を読み、文字情報に変更があれば、その内容に従って TEXTを再描画する。

■ AutoLISP ファイルの他に必要なファイル

1．TextProperties.csv

AutoCAD と Excel との間で、TEXT情報を受け渡すためのファイル。
1行分の書き出しフォーマットは以下のイメージ

通番, ハンドル, X座標,Y座標,画層,文字高さ,幅係数,回転角度,位置合せ情報,文字内容

2．TextPropertiesList.xlsm

TEXT 情報を表示／編集するためのExcelワークブック。
下記２つのマクロを有する

■ DispTextProperties

TextProperties.csv のTEXT情報を読み取り、シート上にリスト表示するマクロ。

■ OUT-CSV

シート上のTEXT情報を、 **TextProperties.csv** に上書きし、AutoCAD コマンド **MODTXT** を起動するマクロ。

3．csv-to-xlsm.vbs、xlsm-to-csv.vbs

この２つの VBScript は、AutoLISP からの Excel 起動（ウィドウ設定、マクロ起動など）が、思うようにできなかったため、AutoLISP → VBScript → Excel というように、AutocAD と Excel 間の制御の受け渡しに用いています。
※ VBScript は、近々サポート終了とのことで、PwerShell への移行が推奨されています。

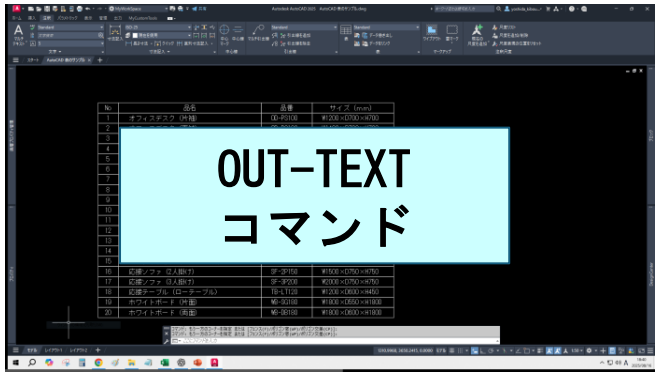
■ csv-to-xlsm.vbs

AutoCAD コマンド **OUT-TEXT** によって起動される。
Excelブック **TextPropertiesList.xlsm** を開いて、マクロ**DispTextProperties** を起動します。
（起動された、**DispTextProperties** は、**TextProperties.csv** を読んで、シート上に TEXT情報リストを表示します。）

■ xlsm-to-csv.vbs

AutoCAD コマンド **MODIFY-TEXT** によって起動される。
Excelブック **TextPropertiesList.xlsm** の マクロ**OUT-CSV** を起動します。
（**OUT-CSV** は、シート上のTEXT情報を **TextProperties.csv** に上書きし、AutoCAD コマンド **MODTXT**を起動します。）
（**MODTXT** は、**TextProperties.csv** を読み、文字情報に変更があれば、その内容に従って TEXTを再描画します。

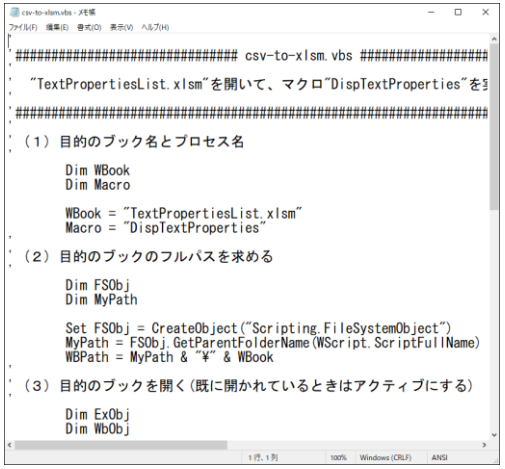
■ TEXT 書換えイメージ



OO.dwg

① TEXT情報 書出し

② xlsm-to-csv 起動



```
' ############################## csv-to-xlsm.vbs ##################
'   "TextPropertiesList.xlsm"を開いて、マクロ"DispTextProperties"を実
' #################################################################
'
' （1）目的のブック名とプロセス名
'
        Dim WBook
        Dim Macro

        WBook = "TextPropertiesList.xlsm"
        Macro = "DispTextProperties"
'
' （2）目的のブックのフルパスを求める
'
        Dim FSObj
        Dim MyPath

        Set FSObj = CreateObject("Scripting.FileSystemObject")
        MyPath = FSObj.GetParentFolderName(WScript.ScriptFullName)
        WBPath = MyPath & "¥" & WBook
'
' （3）目的のブックを開く（既に開かれているときはアクティブにする）
'
        Dim ExObj
        Dim WbObj
```

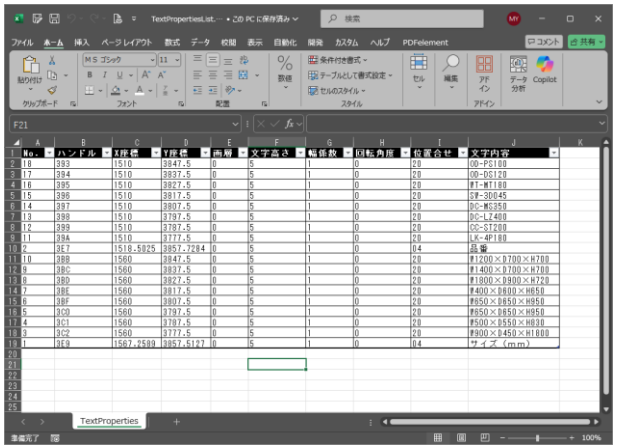
csv-to-xlsm.vbs

③ オープン
④ DispTextProperties 起動

```
No.,ハンドル,X座標,Y座標,画層,文字高さ,幅係数,回転角度,位置合せ,文
1,39E,1510,3737.5,0,5,1,0,20,BK-GL120
2,39D,1510,3747.5,0,5,1,0,20,CB-ST4D180
3,39C,1510,3757.5,0,5,1,0,20,CB-ST2D90
4,39B,1510,3767.5,0,5,1,0,20,LK-12P360
5,39A,1510,3777.5,0,5,1,0,20,LK-4P180
6,399,1510,3787.5,0,5,1,0,20,CC-ST200
7,398,1510,3797.5,0,5,1,0,20,DC-LZ400
8,397,1510,3807.5,0,5,1,0,20,DC-MS350
9,396,1510,3817.5,0,5,1,0,20,SW-3D045
10,395,1510,3827.5,0,5,1,0,20,WT-MT180
11,394,1510,3837.5,0,5,1,0,20,OD-DS120
12,393,1510,3847.5,0,5,1,0,20,OD-PS100
```

TextProperties.csv

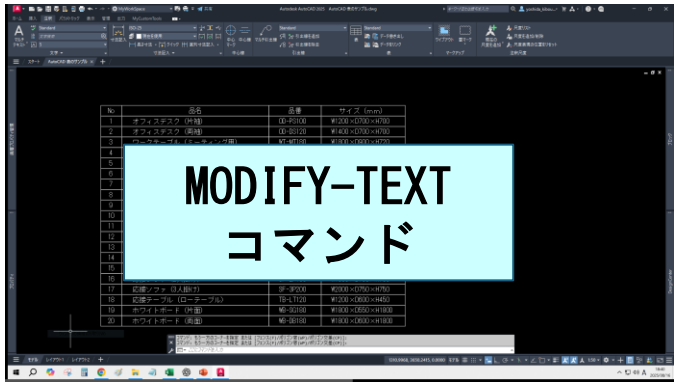
⑤ TEXT情報 読み込み
⑥ TEXT情報 リスト表示



No.	ハンドル	X座標	Y座標	画層	文字高さ	幅係数	回転角度	位置合せ	文字内容
12	393	1510	3847.5	0	5	1	0	20	OD-PS100
11	394	1510	3837.5	0	5	1	0	20	OD-DS120
10	395	1510	3827.5	0	5	1	0	20	WT-MT180
9	396	1510	3817.5	0	5	1	0	20	SW-3D045
8	397	1510	3807.5	0	5	1	0	20	DC-MS350
7	398	1510	3797.5	0	5	1	0	20	DC-LZ400
6	399	1510	3787.5	0	5	1	0	20	CC-ST200
5	39A	1510	3777.5	0	5	1	0	20	LK-4P180
4	39B	1510	3767.5	0	5	1	0	20	LK-12P360
3	39C	1510	3757.5	0	5	1	0	20	CB-ST2D90
2	39D	1510	3747.5	0	5	1	0	20	CB-ST4D180
1	39E	1510	3737.5	0	5	1	0	20	BK-GL120
0	3E7	1518.5825	3857.7284	0	5	1	0	04	品番
10	3B8	1560	3847.5	0	5	1	0	20	W1200×D700×H700
9	3BC	1560	3837.5	0	5	1	0	20	W1400×D700×H700
8	3BD	1560	3827.5	0	5	1	0	20	W1800×D900×H720
7	3BE	1560	3817.5	0	5	1	0	20	W400×D600×H650
6	3BF	1560	3807.5	0	5	1	0	20	W650×D850×H950
5	3C0	1560	3797.5	0	5	1	0	20	W650×D850×H950
4	3C1	1560	3787.5	0	5	1	0	20	W500×D550×H830
3	3C2	1560	3777.5	0	5	1	0	20	W900×D450×H1800
1	3E8	1567.2589	3857.5127	0	5	1	0	04	サイズ（mm）

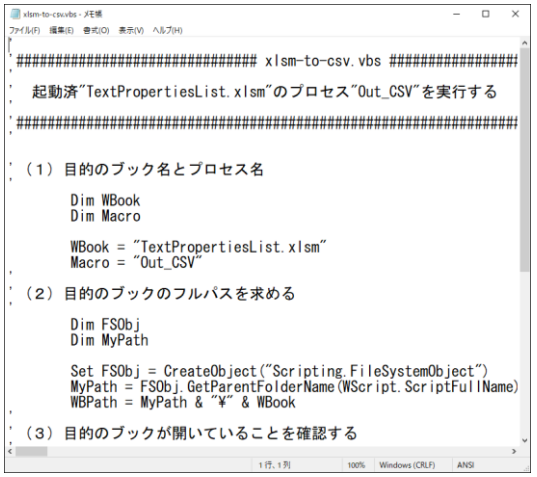
TextPropertiesList.xlsm

■ TEXT情報 書出しイメージ



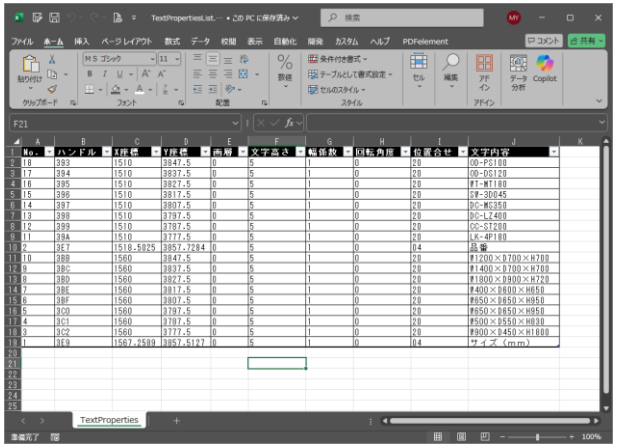
〇〇.dwg

① xlsm-to-csv 起動



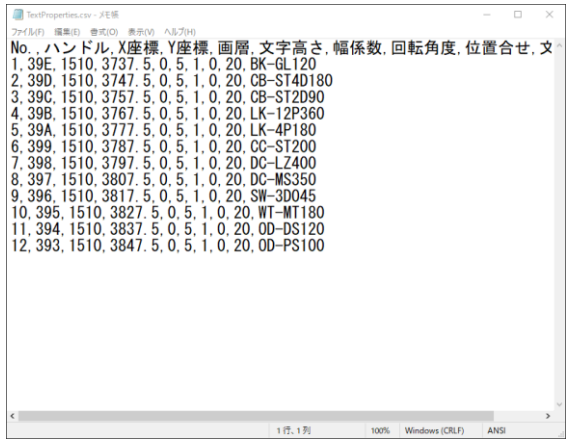
xlsm-to-xlsm.vbs

② マクロ OUT-CSV 起動



TextPropertiesList.xlsm

③ TEXT情報 上書き



TextProperties.csv

④ MODTXT 起動

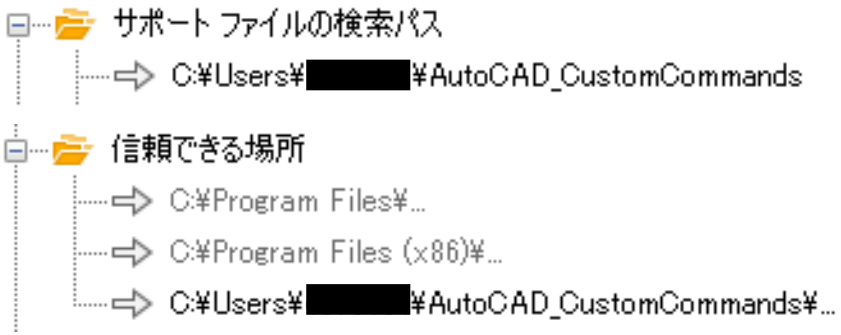
⑤ TEXT情報 取得
⑥ TEXT 書換

■ フォルダ構造（例）

```
[C:/Users]
 ├──── [ユーザ名]
 │     ├──── [AutoCAD_CustomCommands]
 │     │     ├──── [Function]
 │     │     │     ├──── [Change-CSV-to-List.lsp]          Change-CSV-to-List
 │     │     │     └──── [Modify-EntProperties.lsp]        Modify-EntProperties
 │     │     ├──── [TextEditor]
 │     │     │     ├──── [Out-Text.lsp]                    OUT-TEXT
 │     │     │     ├──── [Modify-Text.lsp]                 Modify-Text,MODTXT
 │     │     │     ├──── [TextProperties.csv]
 │     │     │     ├──── [TextPropertiesList.xlsm]
 │     │     │     ├──── [csv-to-xlsm.vbs]
 │     │     │     └──── [xlsm-to-csv.vbs]
 │     │     └──── [acaddoc.lsp]
```

① 個人ユーザーフォルダ直下にフォルダ [AutoCAD_CustomCommands]を置きます
② その下に、[Function] フォルダ、[TextEditor] フォルダ、acaddoc.lsp を置きます。
③ [Function] には、他のコマンドと共有使用するカスタム関数を置きます。
④ [TextEditor] には、今回のTEXT編集に必要なファイル一式を置きます。

■ AutoCAD フォルダ設定



※「¥...」は、下層の全てのフォルダを含むことを表します。

■ acaddoc.lsp の記述

acaddoc.lsp は、図面を開くタイミングで実行されるLISP ファイルです。
以下に、そのサンプルを示します。

```
;■初期設定
(vl-load-com)                                   ; Visual LISPを有効にします。
(command-s “._COMMANDLINE”)                     ; コマンドラインを表示します。
(setvar “PALETTEOPAQUE” 1)                      ; パレット、コマンドウィンドウが半透明になる機能を無効化
(setvar “CMDECHO” 1)                            ; コマンドエコー有効化
(setvar “DYNMODE” 3)                            ; ダイナミック入力有効化
;
;■カスタム関数をロード
;
;【共通関数】                                   ; ファンクションをロード
      (load "Function/Change-CSV-to-List.lsp")
      (load "Function/Change-List-to-CSV.lsp")
      (load "Function/Modify-EntProperties.lsp")
;
;【テキスト編集コマンド】                       ;コマンドをロード
      (load "TextEditor/Out-Text.lsp")
      (load "TextEditor/Modify-Text.lsp")
```

上記は、テキスト編集コマンドを行うときに最低限必要なLISPファイルのみをロードしています。
初期設定部分は、不要であれば、省略しても問題ありません。

■ ExcelVBA マクロ詳細

1．**DispTextProperties**：TextProperties.csv を読み シートにリスト表示

① Excelウィンドウ操作
・最前面に移動し、アクテブに
・以前のサイズを復元
② シート初期化
・TextProperties シートをアクティブにし、セル「A1」を画面左上に
・オートフィルタを解除
・全データ削除し、セルの書式設定（全セル：文字列）
③ データ読み込み（**Load_TextProperties**）
・TextProperties.csv のパスを取得しリードオープン
・1行目を読み込み
・区切り文字「,」にてデータを分解、配列化
・配列の要素をセルに書き込む
・データが無くなるまで繰り返し
・TextProperties.csv クローズ
④ テーブル化
・セルのひとまとまりをテーブル化
⑤ ソート（**Sort_TBL_Key1to5**）
・X座標昇順、Y座標降順 にてソート

2．**OUT-CSV**：編集結果を TextProperties.csv に上書き後、MODTXT を起動

セルデータの書き出しには、TextStream オブジェクトと WriteLine メソッドを使用して、1行単位でデータを書きだします。

① TextStream オブジェクトの作成
・TextProperties.csv の TextStream オブジェクト取得（上書きモードで TextProperties.csv をオープン）
② データ行数の取得
③ 書き込み
・1行データを配列に代入
・同データを「,」区切り文字列に変換して書き込み
・行数分繰り返し
・TextProperties.csv クローズ
④ TextStream オブジェクト開放
⑤ AutoCAD コマンド MODTXT を起動する

■ AutoCAD カスタムコマンド(AutoLISP)詳細（１／２）

１．Out-Text：選択TEXT情報を TextProperties.csv に書き込み後、csv-to-xlsm.vbs 経由でExcelマクロDispTextPropertiesを起動

- TextProperties.csv と csv-to-xlsm.vbs のパスを取得
- 選択したオブジェクトからTEXTを抽出し 選択セット**Texts**に代入
- TextProperties.csv ライトオープン
- 見出し行データを作成し、１行書き込み
- TEXTデータ行を作成し、１行書き込み。これを、TEXTの数分繰り返す
    - No　　：通番
    - Handle　　：ハンドル（グループコード 5）
    - X　　：X座標（グループコード 10）
    - Y　　：Y座標（グループコード 10）
    - Layer　　：画層（グループコード 8）
    - Hight　　：文字の高さ（グループコード 40）
    - Width　　：幅係数（グループコード 41）
    - Angle　　：回転角度（グループコード 50）
    - PosiSetType　　：垂直方向の文字位置合わせタイプ + 水平方向の文字位置合わせタイプ
    - TEXT　　：文字内容
    - WriteData　　：上記のカンマ区切りデータ
- TextProperties.csv クローズ
- csv-to-xlsm.vbs を起動

２．Modify-Text：xlsm-to-csv.vbs 経由でExcelマクロOUT-CSVを起動

- xlsm-to-csv.vbs のパスを取得
- xlsm-to-csv.vbs を起動

■ AutoCAD カスタムコマンド(AutoLISP)詳細（２／２）

３．**MODTXT**：TextProperties.csv を読んで、TEXT情報に変更があれば それに従ってTEXTを再描画する

・TextPropertiesのパスを取得し、リードオープン
・１行（見出し部分）読む　※ 使用しない
・１行（TEXT情報）読み、関数Change-Csv-to-List を用いてリスト化し、**DataList** に格納
・DataList から各要素を抽出
・TEXT 書換用の グループコードと値のドットペアリストを作成
・関数Modify-EntProperties を用いて、TEXTを再描画（書換）
・これを、TextProperties.csv の最終データまで繰り返す
・TextProperties.csv をクローズ

【 Modify-EntProperties 】関数

引数1　：Handle（ハンドル）
引数2　：DotPearList（（１. TEXT）（８. LAYER）・・・・）の様な、グループコードとその値のドットペアリスト

・Handle から、図形情報を取得
・その図形情報と引数2のDotPearList の 同一グループコードにおける値を比較
・値の異なるグループコードがあれば、その値に従って該当の図形を再描画（書換）を行う

■ VBScript 詳細

１．**csv-to-xlsm**：エクセルブックTextPropertiesList.xlsm を開いて、マクロDispTextProperties を起動する

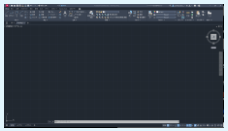
・TextPropertiesList.xlsm のパスを取得
・TextPropertiesList.xlsm を開く（既に開かれているときは、アクティブにする）
・マクロDispTextProperties を起動

２．**Xlsm-to-csv**：エクセルブックTextPropertiesList.xlsm のマクロOUT-CSV を起動する

・TextPropertiesList.xlsm のパスを取得
・マクロOUT-CSV を起動

■ サンプル動画

TEXT を編集する様子を動画で見てみましょう



クリック
動画が再生されます

本動画は、エクセルワークシート上にて作成した表をそのまま書き写すというものです。

※　AutoCADには、ワークシートの内容を用いて AutoCADに表を作成する、または、ワークシートとAutoCADの表とを リンクさせて同期させるというような機能が既に用意されています。使用用途によっては、そちらのほうが便利な場合もありますので、そちらも勉強してみてください。

■ ExcelVBA マクロソース（１／８）

DispTextProperties

① **TextProperties.csv** を読み、シート上にリスト表示します。

```
'FindWindow：クラス名とウィンドウ名にて指定した文字列と一致するウィンドウへのハンドルを取得
   Declare PtrSafe Function FindWindow Lib "user32" Alias "FindWindowA" _
                              (ByVal lpClassName As String, ByVal lpWindowName As String) As Long

'SetForegroundWindow：指定したウィンドウを最前面に移動し、ウィンドウをアクティブにします
   Declare PtrSafe Sub SetForegroundWindow Lib "user32" (ByVal hwnd As Long)


Sub DispTextProperties()

'(１) ブック"ThisWorkbook"、シート"TextProperties"を最前面に表示 ----------------------------------------------------------------------------

    Dim WindowHandle As Long
    WindowHandle = FindWindow(vbNullString, ThisWorkbook.Name)      '自身のウィンドウへのハンドルを取得
    SetForegroundWindow WindowHandle                                '取得したハンドルを使用して自身のウィンドウを最前面に移動
    ThisWorkbook.Activate                                           '自身のワークブック"TextPropertiesList.xlsm"をアクティブに
    If ActiveWindow.WindowState = xlMinimized Then                  'ウィンドウが最小化されているときは
       Application.WindowState = xlNormal                           '以前のサイズに拡大
    End If
    Worksheets("TextProperties").Activate                           'シート"TextProperties"をアクティブに
    Application.Goto Reference:=Range("A1"), Scroll:=True           'ウィンドウの左上を"A1"にスクロール

'(２) シート"TextProperties"を初期化 ----------------------------------------------------------------------------------------------------

    With ActiveSheet
       If .FilterMode = True Then .ShowAllData                      'オートフィルタを解除
       .Cells.Delete                                                '全データ削除
       .Cells.NumberFormat = "@"                                    '全セル → 文字列
    End With
```

```
'(3) "TextProperties.csv"を読み込んで、シート"TextProperties"に表示 ----------------------------------------------------------------------

    Call Load_TextProperties

'(4) 表示データをテーブル化 ----------------------------------------------------------------------------------------------------------------

    Dim TBL As ListObject
    Set TBL = ActiveSheet.ListObjects. _
            Add(xlSrcRange, Cells(1, 1).CurrentRegion, , xlYes)          'Cells(1, 1)からのひとまとまりをテーブル化(先頭行：見出し)
    With TBL
       .Name = "TextProperties"                                          'テーブル名称 --> "TextProperties"
       .TableStyle = "TableStyleMedium15"                                'テーブルスタイル --> "TableStyleMedium15"
       .ShowTableStyleRowStripes = False                                 '縞模様 --> なし
    End With

'(5) 表示データをソート ----------------------------------------------------------------------------------------------------------------

    Call Sort_TBL_Key1to5(TBL, "X座標", 1, "Y座標", 2)                   'Y列を降順ソート後、X列を昇順ソート
    TBL.Range.Columns.AutoFit

  End Sub
```

■ ExcelVBA マクロソース（3／8）

Load_TextProperties

① **TextProperties.csv** を1行読み、得られたカンマ区切り文字列を配列に変換します。
② 配列データをセル1行目に書き込みます。
③ これを、行をインクリメントしながら、読み出しデータが無くなるまで繰り返します。

```
Public Sub Load_TextProperties()

'(1) "¥TextProperties.csv"をリードオープン ----------------------------------------------------------------------------------------------------

    Dim Filename As Variant
    Dim CsvNo As Integer
    Filename = ActiveWorkbook.Path & "¥TextProperties.csv"
    CsvNo = FreeFile
    Open Filename For Input As #CsvNo

'(2) データロード ------------------------------------------------------------------------------------------------------------------------------

    Dim LineData As String
    Dim Properties() As String
    Dim i As Long
     i = 1                                                          'Loopインデックス i = 1
    Do Until EOF(CsvNo)                                             'データがある限りループ
       Line Input #CsvNo, LineData                                  'i番目のデータを"LineData"に読み込み
       Properties = Split(LineData, ",")                            '配列"Properties"にカンマ区切りLineDataを分解して格納
       Range(Cells(i, 1), Cells(i, UBound(Properties) + 1)) = Properties   '配列"Properties"の要素を「i行1列」から順に書き込み
       i = i + 1                                                    'i = i + 1
    Loop                                                            '(Loop)

'(3) "¥TextProperties.csv"をクローズ ----------------------------------------------------------------------------------------------------

    Close #CsvNo

End Sub
```

■ ExcelVBA マクロソース（4 ／ 8）

Sort_TBL_Key1to5

① 引数にて指定したテーブルのソートを行います。
② 指定可能なキーと昇順／降順指定は5種類
③ 1番目のキー以外は省略可能です。

```
Public Sub Sort_TBL_Key1to5 _
      (ByVal TBL As ListObject, _
       ByVal Key1 As String, Optional Odr1 As Integer = 1, _
       Optional Key2 As String = "", Optional Odr2 As Integer = 1, _
       Optional Key3 As String = "", Optional Odr3 As Integer = 1, _
       Optional Key4 As String = "", Optional Odr4 As Integer = 1, _
       Optional Key5 As String = "", Optional Odr5 As Integer = 1)

 '(1) 例外処理 ----------------------------------------------------------------------------------------------------------------------------

      If TBL Is Nothing Then Exit Sub                                  '指定のテーブルが無ければ終了
      If TBL.DataBodyRange Is Nothing Then Exit Sub                    '指定のテーブルにデータが無ければ終了

 '(2) ソートフィールド設定 ----------------------------------------------------------------------------------------------------------------

      Dim Clm1 As Long, Clm2, Clm3, Clm4, Clm5

       With ActiveSheet.Sort.SortFields
         .Clear
         If Key1 <> "" Then
            Clm1 = TBL.ListColumns(Key1).Range.Column
            .Add Key:=Cells(1, Clm1), SortOn:=0, Order:=Odr1, DataOption:=1
         End If
         If Key2 <> "" Then
            Clm2 = TBL.ListColumns(Key2).Range.Column
            .Add Key:=Cells(1, Clm2), SortOn:=0, Order:=Odr2, DataOption:=1
         End If
```

```
        If Key3 <> "" Then
            Clm3 = TBL.ListColumns(Key3).Range.Column
            .Add Key:=Cells(1, Clm3), SortOn:=0, Order:=Odr3, DataOption:=1
        End If          If Key4 <> "" Then
            Clm4 = TBL.ListColumns(Key4).Range.Column
            .Add Key:=Cells(1, Clm4), SortOn:=0, Order:=Odr4, DataOption:=1
        End If
        If Key5 <> "" Then
            Clm5 = TBL.ListColumns(Key5).Range.Column
            .Add Key:=Cells(1, Clm5), SortOn:=0, Order:=Odr5, DataOption:=1
        End If
    End With

'(3) ソート実行 ----------------------------------------------------------------------------------------------------------------------------------

    With ActiveSheet.Sort
        .SetRange TBL.Range
        .Header = xlYes
        .MatchCase = True
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With

End Sub
```

■ ExcelVBA マクロソース（6／8）

Out_CSV

① シート上のデータを1行ずつ、カンマ区切り文字列に変換しながら、**TextProperties.csv** に書き込みます。
※ FUNCTION LineData(i , minClm , maxClm) を用いてカンマ区切り文字列に変換しています。
② AutoCAD コマンド、MODTXT を起動します。
※ AutoCAD 2025 用になっています。バージョンに合わせて 25 の文字を変更してください。

```
Sub Out_CSV()

    Dim CsvFileName As Variant
    Dim FileSysObj As New FileSystemObject
    Dim RowsTextStream As TextStream
    Dim minRow As Long, maxRow As Long, minClm As Long, maxClm As Long
    Dim i As Long

    CsvFileName = ActiveWorkbook.Path & "¥TextProperties.csv"

    minRow = 1
    minClm = 1
    With Cells(1, 1).CurrentRegion
        maxRow = .Rows.Count
        maxClm = .Columns.Count
    End With

    Set RowsTextStream = FileSysObj.CreateTextFile(Filename:=CsvFileName, Overwrite:=True)
    For i = minRow To maxRow
        RowsTextStream.WriteLine LineData(i, minClm, maxClm)
    Next
    RowsTextStream.Close

    Set RowsTextStream = Nothing
    Set FileSysObj = Nothing
```

```
    Dim acadApp As Object, acadDoc As Object

    Set acadApp = GetObject(, "AutoCAD.Application.25")
    Set acadDoc = acadApp.ActiveDocument

    Run_CMD = "MODTXT" & vbCrLf
    acadDoc.SendCommand Run_CMD

    Set acadDoc = Nothing
    Set acadApp = Nothing

End Sub
```

LineData （Function）

① 引数にて指定された、「行、列の範囲」内のセルデータを カンマ区切り文字列にて返します。

```
Public Function LineData(ByVal xRow As Long, minClm As Long, maxClm As Long) As String

  Dim buf As Variant
  Dim xClm As Long


'(1) LineData 初期化 ----------------------------------------------------------------------------------------------------------------------------

  LineData = ""

'(2) 1行分のデータをカンマ区切りデータに変換 ----------------------------------------------------------------------------------------------------

  For xClm = minClm To maxClm                    '列方向 minClm ～ maxClm 繰り返し

    buf = Cells(xRow, xClm)                      'セル値を読む
    Select Case True
      Case IsNumeric(buf)                        '数値は文字列に変換
        buf = CStr(buf)
      Case InStr(buf, ","), InStr(buf, """")     '「,」と「""」が含まれる文字列は「""""」で挟む
        buf = """" & buf & """"
    End Select

    If LineData = "" Then                        '最初
      LineData = buf                             '
    Else                                         '2回目以降
      LineData = LineData & "," & buf        '
    End If

  Next

End Function
```

■ AutoLISP ソース（１／１１）

C:Out-Text

① 選択した図形の中から、TEXTの情報を取得し、**TextProperties.csv** に書き込む
② VBScript **csv-to-xlsm.vbs** を起動し、Excelファイル **TextPropertiesList.xlsm** のTEXT情報リスト表示用マクロを実行させる。

```
(Defun C:Out-Text (
/
startmsec                ;テキスト選択完了時のシステム時刻
endmsec                  ;処理終了時のシステム時刻
CsvFileName              ;「CSV-TextProperty.csv」のパス(テキスト情報格納用カンマ区切りデータファイル)
VbsFileName              ;「Run-ExcelBook-AfterOpen.vbs」のパス(EXCELブック「E_TextPropertyList.xlsm」起動用VBSファイル)
CSV                      ;上記「CSV-TextProperty.csv」のファイルオブジェクト
Header                   ;CSVに書き出す見出しデータ(カンマ区切り"No.,Handle,X,Y,LAYER,HEIGHT,WIDTH,ANGLE,PosiSetType,TEXT")
WriteData                ;CSVに書き出すデータ(1行分のカンマ区切りテキストデータ)
Texts                    ;テキストの選択セット
TextLen                  ;テキスト選択セットの要素数
TextIdx                  ;テキスト数ループ時のインデックス(0～)
TextEntName              ;テキストの図形名
TextProperties           ;entgetにて取得したテキストの情報リスト
No                       ;EXCEL「No.」カラムに表示するデータ(連番)
Handle                   ;[ハンドル](図形識別コード)
Coord                    ;[挿入座標]
X                        ;X座標
Y                        ;Y座標
Layer                    ;[画層名]
Hight                    ;[文字の高さ]
Width                    ;[幅係数]
Angle                    ;[回転角度]（°）
H_SetType                ;[水平位置合わせ]
V_SetType                ;[垂直位置合わせ]
PosiSetType              ;位置合わせ
TEXT                     ;[内容]
DataList                 ;[ハンドル]/X座標/Y座標/[画層名]/[文字の高さ]/[幅係数]/[回転角度]/位置合わせ/[内容]
)
```

■ AutoLISP ソース（２／１１）

```
;前処理
(princ "¥n[Out-Text]")
(setvar "CMDECHO" 0)
(setvar "DYNMODE" 3)
;
;必要ファイルのパスを取得
(setq TextEditor_Path (findtrustedfile "TextEditor"))
(setq CsvFileName (strcat TextEditor_Path "¥¥TextProperties.csv"))
(setq VbsFileName (strcat TextEditor_Path "¥¥csv-to-xlsm.vbs"))
;
;テキストの選択
(setq Texts (ssget '((0 . "TEXT"))))
;
;現在時刻取得
(setq startmsec (getvar "MILLISECS"))
;
;テキストデータ書き出し
(if
	Texts
;
	(progn
		;TextProperties.csv オープン
		(setq CSV (open CsvFileName "w"))
;
		;見出し作成
		(setq Header "No.,ハンドル,X座標,Y座標,画層,文字高さ,幅係数,回転角度,位置合せ,文字内容")
		(write-line Header CSV)
;
```

```
;取得したテキスト数繰り返す
(setq TextLen (sslength Texts))
(setq TextIdx 0)
(repeat TextLen
        (setq No (rtos (1+ TextIdx) 2))
        ;(princ "[")(princ (- TextLen TextIdx))(princ "]")
        ;
        ;テキストの情報を取得
        (setq TextEntName (ssname Texts TextIdx))
        (setq TextProperties (entget TextEntName))
        ;
        (setq Handle (cdr (assoc 5 TextProperties)))
        (setq Coord (cdr (assoc 10 TextProperties))) (setq X (car Coord) Y (cadr Coord))
        (setq Layer (cdr (assoc 8 TextProperties)))
        (setq Hight (cdr (assoc 40 TextProperties)))
        (setq Width (cdr (assoc 41 TextProperties)))
        (setq Angle (cdr (assoc 50 TextProperties))) (setq Angle (/ (* 180 Angle) PI))
        (setq H_SetType (cdr (assoc 72 TextProperties)) V_SetType (cdr (assoc 73 TextProperties)))
                (setq PosiSetType (strcat (itoa V_SetType) (itoa H_SetType)))
        (setq TEXT (cdr (assoc 1 TextProperties)))
        (setq DataList (list No Handle X Y Layer Hight Width Angle PosiSetType TEXT))
        ;
        ;テキスト情報をカンマ区切りデータに変換（数値は十進表記文字列に変換）
        (setq WriteData (Change-List-to-Csv DataList))
        ;
        ;1行分のデータ(テキスト1文字分）書き出し
        (write-line WriteData CSV)
        ;
        ;インデックス +1
        (setq TextIdx (1+ TextIdx))
        ;
);repeat
```

```
;
                ;CSV クローズ
                (close CSV)
;
                ;EXCEL起動用のVBSを起動
                (command-s "._START" VbsFileName)
;
        );progn
;
        (progn
                (princ "¥n選択した図形の中にテキストが含まれていませんでした")
        );progn
;
);if
;終了処理
(setvar "CMDECHO" 1)
(setq endmsec (getvar "MILLISECS"))
(princ "¥n *** end ")(princ (- endmsec startmsec))(princ " msec")
(acet-sys-beep 0)
;
(princ)
)
```

■ AutoLISP ソース（５／１１）

Change-List-to-Csv、num-to-str

① 引数として受け取ったリストを カンマ区切りの文字列として返す。
※ リストの要素に、カンマなどの特殊文字がある場合の処理はしていない。

```
(Defun Change-List-to-Csv (Lst / Itm STR)

   (setq STR (num-to-str (car Lst)))
   (foreach Itm (cdr Lst)
      (setq STR (strcat STR "," (num-to-str Itm)))
   );foreach
   STR

);defun
;
;
(Defun num-to-str (val / )
   (if
      (numberp val)
      (setq val (rtos val 2))
   );if
   val
);defun
```

■ AutoLISP ソース（６／１１）

C:Modify-Text

① VBScript **xlsm-to-csv.vbs** を起動し、Excelファイル **TextPropertiesList.xlsm** のTEXT情報の書き出し用マクロを実行させる。

```
(Defun C:Modify-Text ( / TextEditor_Path VbsFileName)
;
;前処理
(princ "¥n[Modify-Text]")
(setq startmsec (getvar "MILLISECS"))
(setvar "CMDECHO" 0)
(setvar "DYNMODE" 3)
(command-s "._COMMANDLINE")
;
;パスを取得
(setq TextEditor_Path (findtrustedfile "TextEditor"))
(setq VbsFileName (strcat TextEditor_Path "¥¥xlsm-to-csv.vbs"))
;
;[xlsm-to-csv.vbs]を起動
(command-s "._START" VbsFileName)
;
(setq endmsec (getvar "MILLISECS"))
(princ " end ")(princ (- endmsec startmsec))(princ " msec")
(setvar "CMDECHO" 1)
(princ)
);defun
```

■ AutoLISP ソース（７／１１）

C:MODTXT

① **TextProperties.csv** を読み、変更されたTEXT情報があれば、それに従ってTEXTを再描画する。

```
(Defun C:MODTXT ( / TextEditor_Path CsvFileName CSV
                    Header HeaderList
                    Data DataList n Item
                    No Handle X Y LAYER HEIGHT ANGLE PosiSetType TEXT Coord V_SetType H_SetType
                    TextName DotPearList)

(princ "¥n[MODTXT]")
(setq startmsec (getvar "MILLISECS"))
(setvar "CMDECHO" 0)
;
(setq TextEditor_Path (findtrustedfile "TextEditor"))
(setq CsvFileName (strcat TextEditor_Path "¥¥TextProperties.csv"))
;
;[TextProperties.csv]を開く
(setq CSV (open CsvFileName "r"))
;
;見出し部分を読む
(setq Header (read-line CSV))
;
```

■ AutoLISP ソース（８／１１）

```
;テキスト情報を1つずつ読みながら再描画 これをデータ数分繰り返す
(while
        ;1行読む（データが無ければ終了）
        (setq Data (read-line CSV))
        ;
        ;カンマ区切りデータをリスト化
        (setq DataList (Change-Csv-to-List Data))
        ;
        ;テキスト情報の抽出
        (setq No (nth 0 DataList))
        (setq Handle (nth 1 DataList)) (if (numberp Handle) (setq Handle (rtos Handle)))
        (setq X (atof (nth 2 DataList))) (setq Y (atof (nth 3 DataList))) (setq Coord (list X Y))
        (setq LAYER (nth 4 DataList))
        (setq HEIGHT (atof (nth 5 DataList)))
        (setq WIDTH (atof (nth 6 DataList)))
        (setq ANGLE (atof (nth 7 DataList))) (setq ANGLE (* (/ PI 180) ANGLE))
        (setq PosiSetType (nth 8 DataList)) (setq V_SetType (atoi(substr PosiSetType 1 1))) (setq H_SetType (atoi(substr PosiSetType 2 1)))
        (setq TEXT (nth 9 DataList))
        ;
        ;テキスト書換用のリスト(グループコードと値の連想リスト)を作成
        (setq DotPearList (list (cons 1 TEXT)
                                (cons 8 LAYER)
                                (cons 40 HEIGHT) (cons 41 WIDTH)
                                (cons 50 ANGLE)
                                (cons 72 H_SetType) (cons 73 V_SetType)
                          )
        )
        ;
        ;再描画（書換）
        (Modify-EntProperties Handle DotPearList)
        ;
);while
```

```
;
;[TextProperties.csv]を閉じる
(close CSV)
;
;終了処理
(setq endmsec (getvar "MILLISECS"))
(princ " end ")(princ (- endmsec startmsec))(princ " msec")
(setvar "CMDECHO" 1)
(acet-sys-beep 0)
;
(princ)
);defun
```

■ AutoLISP ソース（１０／１１）

Change-Csv-to-List

①　引数であるカンマ区切りの文字列を リストに変換する

```
(Defun Change-Csv-to-List (Str / Lst Pos Dat)
    (setq Lst nil)
    (while
        (setq Pos (vl-string-search "," Str))    ;Pos = カンマの位置(先頭は0)
        ;
        (setq Dat (substr Str 1 Pos))            ;Dat = 先頭からPosまでの文字列
        (setq Str (substr Str (+ Pos 2)))        ;Str = StrからDatと"," を除いた文字列)
        (setq Lst (append Lst (list Dat)))       ;LitにDatを追加していく
    );while
    (setq Lst (append Lst (list Str)))           ;Strの残り部分(","を含まない末尾)をリストに追加
    Lst
);defun
```

■ AutoLISP ソース（１１／１１）

Modify-EntProperties

① 引数としてTEXTのハンドル、及び、グループコードとその値のドットペアのリストを受け取る。
② ハンドルに該当するTEXTの情報を取得する。
③ 取得したTEXTの情報と引数のリストとを比較し、異なっているものがあればそれに従ってTEXTを再描画する。

```
(Defun Modify-EntProperties ( Handle DotPearList / EntName EntProperties ModEn Len Idx G-Code NewDotPear OrgDotPear)

;
	(setq EntName (handent Handle))
	(setq EntProperties (entget EntName))
	(setq ModEn nil)
	(setq Len (length DotPearList))
	(setq Idx 0)
	(repeat Len
		(and
			(setq NewDotPear (nth Idx DotPearList))
			(setq G-Code (car NewDotPear))
			(setq OrgDotPear (assoc G-Code EntProperties))
			(/= (cdr NewDotPear) (cdr OrgDotPear))
			(setq EntProperties (subst NewDotPear OrgDotPear EntProperties))
			(setq ModEn T)
		);and
		(setq Idx (1+ Idx))
	);repeat
	(if
		ModEn
		(entmod EntProperties)
	);if
;
(princ)
);defun
```

■ VBScript ソース（1／3）

csv-to-xlsm.vbs

① **TextPropertiesList.xlsm** が開かれていなければ開く
② 開かれていればアクティブにする
③ マクロ **DispPropertiesList** を実行する

```
‘(1)目的のエクセルファイル名とマクロ名を設定
        Dim WBook
        Dim Macro
        WBook = "TextPropertiesList.xlsm"
        Macro = "DispTextProperties"
'
'(2)目的のエクセルファイルのフルパスを求める
        Dim FSObj
        Dim MyPath
        Set FSObj = CreateObject("Scripting.FileSystemObject")          'ファイルシステムオブジェクト
        MyPath = FSObj.GetParentFolderName(WScript.ScriptFullName)      '自身のフォルダパスを取得
        WBPath = MyPath & "¥" & WBook                                   '起動するエクセルブックのフルパス
'
'(3)目的のブックを開く(既に開かれているときはアクティブにする)
        Dim ExObj
        Dim WbObj
        On Error Resume Next                                            '発生するエラーを無視(On Error GoTo 0 まで)
          Set ExObj = GetObject(, "Excel.application")                  'EXCELアプリケーションオブジェクト取得
          If Err.Number = 0 Then                                        'EXCELアプリケーションが起動しているとき
                Set WbObj = ExObj.Workbooks(WBPath)                     '"TextPropertiesList.xlsm"ワークブックオブジェクトを取得
                If Not Err.Number = 0 Then                              '存在していないとき
                   ExObj.Workbooks.Open WBPath                          '"TextPropertiesList.xlsm"を開く
                End If                                                  '
          Else                                                          'EXCELアプリケーションが起動していないとき
                Set ExObj = WScript.CreateObject("Excel.Application")   'Excel起動
                ExObj.Visible = True                                    '可視化
                ExObj.Workbooks.Open WBPath                             '"TextPropertiesList.xlsm"を開く
          End If                                                        '
        On Error GoTo 0                                                 '発生したエラーを無効にして次に進む
```

■ VBScript ソース（２／３）

```
'
'（４）マクロを実行する
        ExObj.Run WBook & "!" & Macro                    'フルパスを使用するとNG
'
'（５）実行結果のウィンドウをアクティブにする
        Dim WS_ShlObj
        Set WS_ShlObj = WScript.CreateObject("WScript.Shell")    'WshShellオブジェクトを取得
        WS_ShlObj.AppActivate WBook                      '"TextPropertiesList.xlsm"ウィンドウをアクティブ化
'
'（６）各オブジェクトの開放
        Set FSObj = Nothing                              'ファイルシステムオブジェクトの開放
        Set ExObj = Nothing                              'EXCELアプリケーションオブジェクト解放
        Set WbObj = Nothing                              'EXCELブックオブジェクト解放
        Set WS_ShlObj = Nothing                          'Windowsシェルオブジェクトの解放
```

■ VBScript ソース（3／3）

csv-to-xlsm.vbs

① **TextPropertiesList.xlsm** のマクロ **DispPropertiesList** を実行する
※ TextProperiesList.xlsm は開かれていることを前提にしています。

```
'（1）目的のブック名とプロセス名
        Dim WBook
        Dim Macro
        WBook = "TextPropertiesList.xlsm"
        Macro = "Out_CSV"
'
'（2）EXCELアプリケーションオブジェクト取得
        Dim ExObj
        On Error Resume Next                                    '発生するエラーを無視(On Error GoTo 0 まで)
        Set ExObj = GetObject(, "Excel.application")            'EXCELアプリケーションオブジェクト取得
        On Error GoTo 0                                         '発生したエラーを無効にして次に進む
'
'（3）マクロを実行する
        ExObj.Run WBook & "!" & Macro                           'フルパスを使用するとNG
'
'（4）各オブジェクトの開放
        Set ExObj = Nothing                                     'EXCELアプリケーションオブジェクトの開放
```