属性エクセル編集

AutoCAD のカスタムコマンドとExcelVBA マクロを連携させたツールの作成を試みます。

■ やりたいこと

図上のブロックの属性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。

■ ブロック内属性定義の内容

設備NO　　：ブロックの表す設備の識別番号
品名　　：ブロックの表す設備の一般的な名称
寸法　　：ブロックの表す設備の寸法を W1000xD500xH800 形式で記入
品番　　：ブロックの表す設備のメーカー品番

■ エクセル表示イメージ

番号	Handle	ブロック名	X座標	Y座標	設備NO	品名	寸法	品番
9	B65	椅子	1265.88	4121.23	5001	チェア	W420xD520xH790-860	CHR-S800
8	B69	椅子	1265.88	3371.23	5002	チェア	W420xD520xH790-860	CHR-S800
7	B72	ホワイトボード	2223.59	4956.23	8001	ホワイトボード	W1860xD600xH1850	BD1800-LT
4	3793	椅子	3175.88	4121.23	5007	チェア	W420xD520xH790-860	CHR-S800
3	37A0	椅子	3175.88	3371.23	5008	チェア	W420xD520xH790-860	CHR-S800
16	9A9	複合機	4555.5	5535.25	9001	複合機	W1200xD700xH850	RC4000
5	E90	キャビネット	4799.75	3984.5	3005	ローキャビネット	W680xD380xH700	KB-S700
6	E43	キャビネット	4799.75	3984.5	3001	ローキャビネット	W680xD380xH700	KB-S700
11	B06	椅子	7188.04	5565.75	5016	チェア	W420xD520xH790-860	CHR-S800
13	B01	椅子	7188.04	4365.75	5017	チェア	W420xD520xH790-860	CHR-S800
10	B0A	机 ﾜｺﾞﾝx1	7895.75	5340.75	1005	デスク1200	W1200xD700xH700	OD1200-KB1
15	AFB	机 ﾜｺﾞﾝx1	7895.75	5340.75	1011	デスク1200	W1200xD700xH700	OD1200-KB1
2	3EAF	椅子	8594.13	5146.7	5022	チェア	W420xD520xH790-860	CHR-S800
1	3EBC	椅子	8594.13	3946.7	5023	チェア	W420xD520xH790-860	CHR-S800

■ 操作手順イメージ

① カスタムコマンド「EXPPARTSATT」を起動
② 選択オブジェクトの中からブロックの属性「設備NO、品名、寸法、品番」を読み取られ、エクセルシートにリスト表示される。
③ エクセルシート上で「設備NO、品名、寸法、品番」を任意の値に編集する。
④ カスタムコマンド「MODPARTSATT」を起動
⑤ 編集後のエクセルシートの内容に従って、該当するブロックの属性「設備NO、品名、寸法、品番」が更新･再描画される。

■ 使用するAutoCAD カスタムコマンドLISPファイル

EXPPARTSATT.lsp

コマンド **EXPPARTSATT** を記述したファイル。
選択したオブジェクトからブロック参照の情報を取得し **PartsList.csv** に書き出す。
書き出しフォーマットは以下の通り。

通番,ブロック参照のハンドル,ブロック定義のブロック名,X座標,Y座標,設備NO属性値,品名属性値,寸法属性値,品番属性値

書き出した後、**Run_GetPartsList.ps1** を起動して、Excel **PartsList.xlsm** を開いてマクロ**GetPartsList** を起動する。

MODPARTSATT.lsp

コマンド **MODPARTSATT**、および、**EXECMOD** を記述したファイル

MODPARTSATT

Run_PutPartsList.ps1 を起動して、Excel **PartsList.xlsm** のマクロ **PutPartsList** を実行させる。
（同マクロにて **PartsList.csv** は、編集後の属性値に書き換わる。）

EXECMOD

Excel **PartsList.xlsm** マクロ**PutPartsList**の **PartsList.csv** の書換処理後、同マクロから起動される。**PartsData.csv** を参照し、変更されたブロック参照の属性値を書き換える。

■ AutoLISP ファイルの他に必要なファイル

1．PartsList.csv

AutoCAD の**EXPPARTSATT** コマンド、および、EXCEL **PartsList.xlsm** のマクロ**PutPartsList** によってブロック参照の属性データが書き込まれるデータファイル。一行のデータ構造は以下の通り。

通番,ブロック参照のハンドル,ブロック定義のブロック名,X座標,Y座標,設備NO属性値,品名属性値,寸法属性値,品番属性値

2．PartsList.xlsm

属性リストを表示するためのExcel ワークブックで、下記2つのマクロを記録しておきます。

GetPartsList

AutoCAD側 の**EXPPARTSATT** コマンドにて書き出されたブロック参照の属性データ **PartsList.csv** を読み込み、ワークシートにリスト表示するマクロ。後述の、 **Run_GetPartsList.vbs** にて起動されます。

PutPartsList

ワークシート上のブロック参照の属性データを **PartsList.csv** に書き出す（上書き）マクロ。後述の、 **Run_PutPartsList.vbs** にて起動されます。

また、PartsList.csv の書換終了後、AutoCADカスタムコマンド **EXECMOD** を起動します。

3．Run_GetPartsList.ps1 、Run_PutPartsList.ps1

この2つのPowerShellスクリプトについては、AutoLISP から、直接、Excelブックのマクロを起動する方法が見つけられなかったため AutoLISP → PowerShell 起動 → Excelマクロ起動 という方法をとっています。VBScript を用いても同様のことが行え、PowerShell のように処理中余計なウィンドウが表示されず、処理速度も速そうなのですが、VBSscript は2027年以降OSから削除予定とのことで、PowerShell を使用しました。

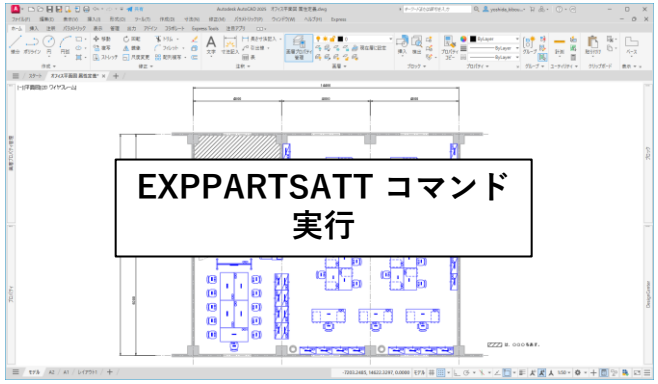
Run_GetPartsList.ps1

AutoCADの **EXPPARTSATT**コマンドにて起動され、Excelブック**PartsList.xlsm**を開き、マクロ**GetPartsList** を起動します。

Run_PutPartsList.ps1

AutoCADの **MODPARTSATT**コマンドにて起動され、Excelブック**PartsList.xlsm**の マクロ**PutPartsList** を起動します。

■ 属性値 書き出しイメージ



〇〇.dwg

① 属性データを書きだし

```
番号,Handle,ブロック名,X座標,Y座標,設備NO,品名,寸法,品番
1,3F0A,椅子,12107.13,3946.7,5035,チェア,W420xD520xH790-860,CHR-S800
2,3EFD,椅子,12107.13,5146.7,5034,チェア,W420xD520xH790-860,CHR-S800
3,3EF0,椅子,12107.13,6346.7,5033,チェア,W420xD520xH790-860,CHR-S800
4,3EE3,椅子,12107.13,7546.7,5032,チェア,W420xD520xH790-860,CHR-S800
5,3ED6,椅子,12107.13,8746.7,5031,チェア,W420xD520xH790-860,CHR-S800
6,3EC9,椅子,12107.13,9946.7,5030,チェア,W420xD520xH790-860,CHR-S800
7,3EBC,椅子,8594.13,3946.7,5023,チェア,W420xD520xH790-860,CHR-S800
8,3EAF,椅子,8594.13,5146.7,5022,チェア,W420xD520xH790-860,CHR-S800
9,3EA2,椅子,8594.13,6346.7,5021,チェア,W420xD520xH790-860,CHR-S800
10,3E95,椅子,8594.13,7546.7,5020,チェア,W420xD520xH790-860,CHR-S800
11,3E88,椅子,8594.13,8746.7,5019,チェア,W420xD520xH790-860,CHR-S800
12,3E7B,椅子,8594.13,9946.7,5018,チェア,W420xD520xH790-860,CHR-S800
13,37A0,椅子,3175.88,3371.23,5008,チェア,W420xD520xH790-860,CHR-S800
14,3793,椅子,3175.88,4121.23,5007,チェア,W420xD520xH790-860,CHR-S800
15,3786,椅子,3175.88,1871.23,5010,チェア,W420xD520xH790-860,CHR-S800
16,3779,椅子,3175.88,1121.23,5011,チェア,W420xD520xH790-860,CHR-S800
17,376C,椅子,3175.88,2621.23,5009,チェア,W420xD520xH790-860,CHR-S800
19,1E88,机 ﾜｺﾞﾝx1,11408.75,5340.75,1023,デスク1200,W1200xD700xH700,0D1200-KB1
21,179D,キャビネット,4874.75,11034.5,4001,ハイキャビネット,W680xD380xH1650,KB-S1
22,1784,キャビネット,4874.75,10154.5,4002,ハイキャビネット,W680xD380xH1650,KB-S1
23,176B,キャビネット,4874.75,9274.5,4003,ハイキャビネット,W680xD380xH1650,KB-S16
```

PartsList.csv

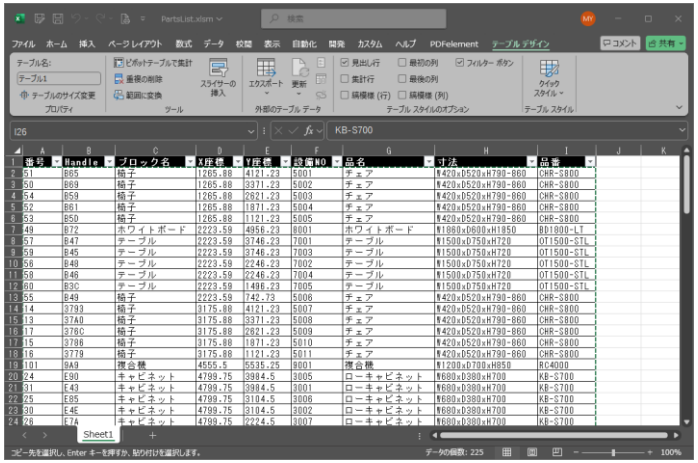
② Run_GetPartsList.ps1
起動

```
# PowerShell 変数定義と初期化
$scriptPath = $MyInvocation.MyCommand.Path
$folderPath = Split-Path -Parent $scriptPath
$excelFileName = "PartsList.xlsm"
$macroName = "GetPartsList"
$fullExcelPath = Join-Path $folderPath $excelFileName

# Excel アプリケーション取得または起動
$excelApp = $null
try {
    $excelApp = [Runtime.InteropServices.Marshal]::GetActiveObject("Excel.Appl
} catch {
    $excelApp = New-Object -ComObject Excel.Application
    $excelApp.Visible = $true
}

# ブックを開くまたは既存のブックを参照
$workbook = $null
foreach ($wb in $excelApp.Workbooks) {
    if ($wb.FullName -eq $fullExcelPath) {
        $workbook = $wb
        break
    }
}
```

Run_GetPartsList.ps1

③ PartsList.xlsm を開いて
マクロGetPartsListを起動

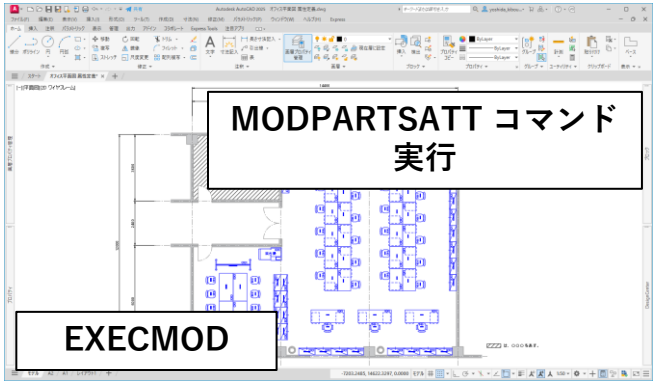
④ 属性データを読み込み
リスト表示



番号	Handle	ブロック名	X座標	Y座標	設備NO	品名	寸法	品番
1	3F0A	椅子	12107.13	3946.7	5035	チェア	W420xD520xH790-860	CHR-S800
2	3EFD	椅子	12107.13	5146.7	5034	チェア	W420xD520xH790-860	CHR-S800
3	3EF0	椅子	12107.13	6346.7	5033	チェア	W420xD520xH790-860	CHR-S800
4	3EE3	椅子	12107.13	7546.7	5032	チェア	W420xD520xH790-860	CHR-S800
5	3ED6	椅子	12107.13	8746.7	5031	チェア	W420xD520xH790-860	CHR-S800
6	3EC9	椅子	12107.13	9946.7	5030	チェア	W420xD520xH790-860	CHR-S800
7	3EBC	椅子	8594.13	3946.7	5023	チェア	W420xD520xH790-860	CHR-S800
8	3EAF	椅子	8594.13	5146.7	5022	チェア	W420xD520xH790-860	CHR-S800
9	3EA2	椅子	8594.13	6346.7	5021	チェア	W420xD520xH790-860	CHR-S800
10	3E95	椅子	8594.13	7546.7	5020	チェア	W420xD520xH790-860	CHR-S800
11	3E88	椅子	8594.13	8746.7	5019	チェア	W420xD520xH790-860	CHR-S800
12	3E7B	椅子	8594.13	9946.7	5018	チェア	W420xD520xH790-860	CHR-S800
13	37A0	椅子	3175.88	3371.23	5008	チェア	W420xD520xH790-860	CHR-S800
14	3793	椅子	3175.88	4121.23	5007	チェア	W420xD520xH790-860	CHR-S800
15	3786	椅子	3175.88	1871.23	5010	チェア	W420xD520xH790-860	CHR-S800
16	3779	椅子	3175.88	1121.23	5011	チェア	W420xD520xH790-860	CHR-S800
17	376C	椅子	3175.88	2621.23	5009	チェア	W420xD520xH790-860	CHR-S800
19	1E88	机 ﾜｺﾞﾝx1	11408.75	5340.75	1023	デスク1200	W1200xD700xH700	0D1200-KB1
21	179D	キャビネット	4874.75	11034.5	4001	ハイキャビネット	W680xD380xH1650	KB-S1
22	1784	キャビネット	4874.75	10154.5	4002	ハイキャビネット	W680xD380xH1650	KB-S1
23	176B	キャビネット	4874.75	9274.5	4003	ハイキャビネット	W680xD380xH1650	KB-S16

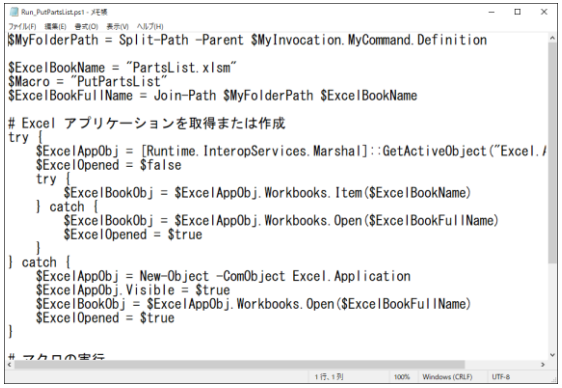
PartsList.xlsm

■ 属性値 書換イメージ



〇〇.dwg

① Run_PutPartsList.ps1
起動

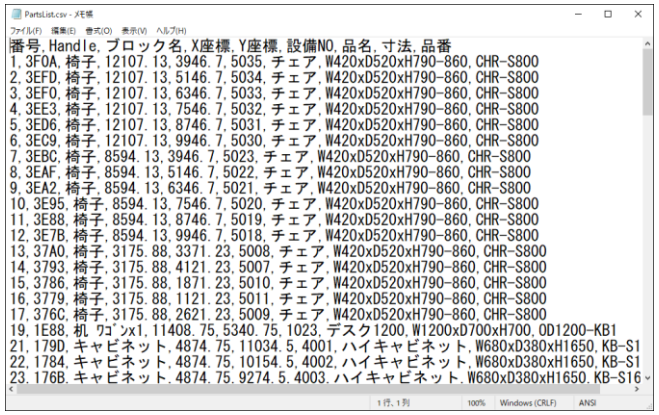


Run_PutPartsList.ps1

② マクロPutPartsListを起動

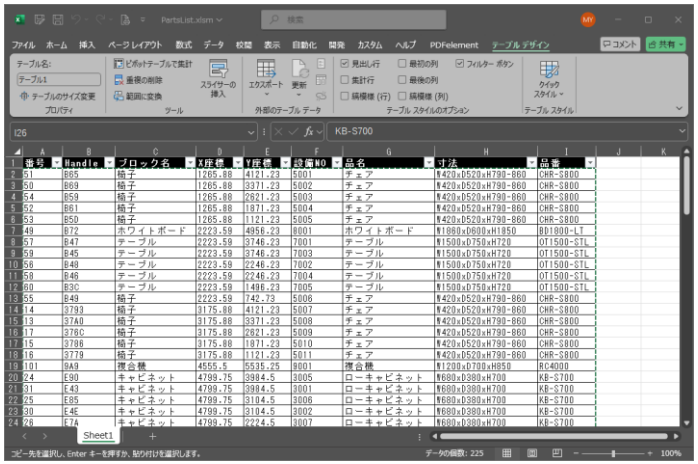
④ EXECMOD
起動

⑤ 属性データを読み出して
ブロック参照を書き換え



PartsList.csv

③ 属性データを上書き



PartsList.xlsm

■ フォルダの構造（例）

```
[C:/Users]
 ├──── [ユーザ名]
 │     ├──── [AutoCAD_CustomCommands]
 │     │      ├──── [Function]
 │     │      │      ├──── [Get-Insert-Attributes.lsp]        Get-Insert-Attributes
 │     │      │      ├──── [Change-CSV-to-List.lsp]          Change-CSV-to-List
 │     │      │      └──── [Modify-Attributes.lsp]           Modify-Attributes
 │     │      ├──── [AttributeEditor]
 │     │      │      ├──── [EXPPARTSATT.lsp]                 EXPPARTSATT
 │     │      │      ├──── [MODPARTSATT.lsp]                 MODPARTSATT,EXECMOD
 │     │      │      ├──── [PartsList.csv]
 │     │      │      ├──── [PartsList.xlsm]
 │     │      │      ├──── [Run_GetPartsList.ps1]
 │     │      │      └──── [Run_PutPartsList.ps1]
 │     │      └──── [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、[AttributeEditor] フォルダ、acaddoc.lsp を置きます。
③ [Function] には、カスタム関数を置きます。
④ [AttributeEditor] には、属性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/Get-Insert-Attributes.lsp")
      (load "Function/Modify-Attributes.lsp")
;
;【属性編集コマンド】                           ;コマンドをロード
      (load "AttributeEditor/EXPPARTSATT.lsp")
      (load "AttributeEditor/MODPARTSATT.lsp")
```

上記は、属性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．GetPartsList

① シート初期化
・オートフィルタを解除
・全データ削除
・セルの書式設定（全セル：文字列）
・見出し行を作成
　通番，ハンドル，ブロック名，X座標，Y座標，識別番号，品名，寸法，品番

② データ読み込み
・PartsList.csv のファイルパスを取得し(Excelブックと同じフォルダに作成とする) 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
・PartsList.csv クローズ

③ テーブル化
・セルのひとまとまりをテーブル化

④ ソート
・X座標昇順、Y座標降順 にてソート

2．PutPartsList

ワークシートの書き出しには、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
・PartsList.csv のファイルパス(Excelブックと同じフォルダに作成とする)を取得
・TextStream オブジェクト取得（上書きモード）

② データ行数の取得
・テーブルの行数を取得

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
・1行書き込み
・行数分繰り返し

④ TextStream オブジェクト開放
・TextStream オブジェクトを開放する

⑤ AutoCAD コマンド EXECMOD を起動する

■ AutoCAD カスタムコマンド(AutoLISP)詳細（１／２）

１．EXPPARTSLIST

・選択中のオブジェクトからブロック参照を抽出し 選択セット**Inserts**に代入
・**Insets**が空の時は、操作者にオブジェクトの選択を促し、ブロック参照を抽出して 選択セット**Inserts**に代入
・**PartsList.csv** のファイルパスを取得し、ライトオープン
・**Inserts** 内 n番目のブロック参照の図形名を取得
・図形名の図形情報を取得
・図形情報から ハンドル・X座標・Y座標・ブロック名 を取得
・関数 **Get-Insert-Attributes** にて「ATT01 ～ ATT04」を取得
・1行分のデータ文字列**「通番|ハンドル|ブロック名 |X座標|Y座標|ATT01| ATT02|ATT03|ATT04」**を作成
・**PartsList.csv** に1行分データを書き込み
・選択セットInserts 内のブロック参照の数 これを繰り返す。
・**PartsList.csv** クローズ
・**PartsList.xlsm** のマクロ **GetPartsList** を起動する

【Get-Insert-Attributes】関数

引数　　：InsName（図形名）
戻り値　　：((属性名 . 属性値) (属性名 . 属性値) (属性名 . 属性値) (属性名 . 属性値) ・・・・)

引数 **InsName(図形名)** からプロパティを取得
グループコード66を確認。(=1 であれば、属性あり)

関数 **entnext**にて **InsName**にネストされた図形名を取得し、その図形名にリンクするプロパティを取得
図形タイプ(グループコード0) を確認。**”SEQEND”**であれば終わり。**”ATTRIB”**であれば、属性情報を取得する。

InsName のプロパティからブロック名(グループコード2) を取得
ブロック名にリンクするブロックテーブルオブジェクト名を関数 **tblobjname** にて取得
このテーブルオブジェクトにネストするオブジェクトのプロパティを取得
図形タイプが **“ATTDEF”** で、固定値、かつ、可視であれば、その固定値を取得
ネストするオブジェクトがなくなるまで繰り返す。

■ AutoCAD カスタムコマンド(AutoLISP)詳細（２／２）

２．MODPARTSATT

・**PartsList.xlsm** のマクロ **GetPartsList** を起動する

３．EXECMOD

・**PartsList.csv** のファイルパスを取得し、リードオープン
・1行目のデータを読む
・データから属性名と属性値のリストを作成する
　（(属性名 . 属性値) (属性名 . 属性値) (属性名 . 属性値) (属性名 . 属性値) ）
・関数 **ModAttributes** を用いて 該当するハンドルを持つ図形の属性を更新する
・これをデータの行数分繰り返す。
・**PartsList.csv** をクローズ

【Modify-Attributes】関数

引数1　　：InsName（図形名）
引数2　　：AttList ((属性名 . 属性値) (属性名 . 属性値) (属性名 . 属性値) (属性名 . 属性値) ・・・・)

・関数 **entnext**にて **InsName**にネストされたオブジェクト(図形名)を取得し、その図形名にリンクするプロパティを取得
・図形タイプ(グループコード0) を確認。**”SEQEND”**であれば終わり。**”ATTRIB”**であれば、属性情報を取得する。
・属性名を取得
・同名の属性名が、引数の**AttList** に存在するかを確認
・存在すれば、属性値を確認
・属性値が異なる場合は、同オブジェクト情報の属性値を置換して、関数 **entmod** を用いて書き換える
・これを **“SEQEND”** まで繰り返す。

■ PowerShell 詳細

1．Run_GetPartsList

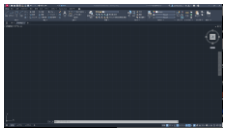
・**PartsList.xlsm** が開かれていなければ開く
・マクロ**GetPartsList** を起動する

2．Run_PutPartsList

・**PartsList.xlsm** が開かれていなければ開く→ 不要？
・マクロ**PutPartsList** を起動する

■ サンプル動画

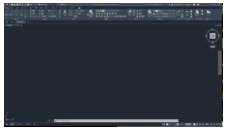
属性を編集する1



クリック
動画が再生されます

・それぞれの備品をブロックとして定義、「設備NO」、「品名」、「寸法」、「品番」の属性を持たせる。
・同種の設備毎に設備NOを付与、併せて、品名、寸法、品番を記録していく

属性を編集する2



クリック
動画が再生されます

・すべての備品に、属性を登録完了
・全ブロック定義を選択して、リスト表示
・オートフィルタを用いて、表示変更することで、品名とその員数などを確認できる。

※ 本動画は、Excelマクロの起動に、VBScript を用いたときのもので、PowerShell の場合とは少し処理中画面が異なりますが機能は同じです。

■ 本ツール作成の効果(メリット／デメリット)

属性をエクセルにてリスト表示することで、以下に示すメリットが生じると考えます。
・属性値の登録･編集作業が、エクセルの持つ多彩な機能をて行えるため、作業効率が向上します。
・エクセルのオートフィルタなどの機能を用いて、設備の種類・員数などの確認が容易にできるようになります。
・さらに、ExcelVBAを用いて「設備の項目と員数などをまとめた構成表を作成する」、「設備の仕入れ価格･仕入先などを記憶したデータベースとリンクさせて手配書･見積書を作成する」など、アイデア次第で有用なツールの作成が可能です。

しかしながら、以下のデメリットも考えられます。
・使用するブロック定義を一元的に管理･徹底する運用を前提としないと、かえって誤り･混乱を生じさせる恐れがある。

■ ExcelVBA マクロソース（１／５）

１．GetPartsList

```
Declare PtrSafe Function FindWindow Lib "user32" Alias "FindWindowA" ( _
    ByVal lpClassName As String, ByVal lpWindowName As String) As LongPtr
Declare PtrSafe Function SetWindowPos Lib "user32" ( _
    ByVal hWnd As LongPtr, _
    ByVal hWndInsertAfter As LongPtr, _
    ByVal X As Long, _
    ByVal Y As Long, _
    ByVal cx As Long, _
    ByVal cy As Long, _
    ByVal wFlags As Long) As Long
Public Const HWND_TOPMOST = -1
Public Const SWP_NOMOVE = &H2
Public Const SWP_NOSIZE = &H1
Public Const SWP_SHOWWINDOW = &H40

Public Sub GetPartsList()

'■ シートの初期化
    'ウィンドウを最前面に
    Dim hWnd As LongPtr
    hWnd = FindWindow("XLMAIN", Application.Caption)
    SetWindowPos hWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE Or SWP_NOSIZE Or SWP_SHOWWINDOW
    'ブックをアクティブに
    ThisWorkbook.Activate
    '最小化されていたら元に戻す
    If ActiveWindow.WindowState = xlMinimized Then
        Application.WindowState = xlNormal
    End If
```

（つづく）

■ ExcelVBA マクロソース（２／５）

１．GetPartsList

（つづき）

```
'■ Sheet(1)を表示
    Worksheets(1).Activate
    'シートの表示を初期化
    Application.Goto Reference:=Range("A1"), Scroll:=True      'セル(1 , 1)が表示されるようスクロール
    With ActiveSheet
        If .FilterMode = True Then .ShowAllData                'オートフィルタ解除
        .Cells.Delete                                          '全セル クリア
        .Cells.NumberFormat = "@"                              '全セル 書式→文字列
    End With
'
'■ PartsList.csv オープン
    Dim Filename As Variant
    Dim CsvNo As Integer
    Filename = ActiveWorkbook.Path & "¥PartsList.csv"
    CsvNo = FreeFile
    Open Filename For Input As #CsvNo
'
'■ ブロック参照属性データロード
    Dim LineDataStr As String
    Dim LineDataAry() As String
    Dim i As Long
    i = 1
    Do Until EOF(CsvNo)
        Line Input #CsvNo, LineDataStr
        LineDataAry = Split(LineDataStr, ",")
        Range(Cells(i, 1), Cells(i, UBound(LineDataAry) + 1)) = LineDataAry
        i = i + 1
    Loop
'
'■ PartsList.csv クローズ
    Close #CsvNo
 ‘
```

（つづく）

■ ExcelVBA マクロソース（３／５）

１．GetPartsList

（つづき）

```
'■ テーブル化
    Dim TBL As ListObject
    Set TBL = ActiveSheet.ListObjects.Add(xlSrcRange, Cells(1, 1).CurrentRegion, , xlYes)
    With TBL
        .TableStyle = "TableStyleMedium15"
        .ShowTableStyleRowStripes = False
    End With
'
'■ 列幅調整
    TBL.Range.Columns.AutoFit
'
'■ ソート
    If TBL.DataBodyRange Is Nothing Then Exit Sub

    Dim Clm1 As Long, Clm2
    Clm1 = TBL.ListColumns("X座標").Range.Column
    Clm2 = TBL.ListColumns("Y座標").Range.Column
    With ActiveSheet.Sort.SortFields
        .Clear
        .Add Key:=Cells(1, Clm1), SortOn:=0, Order:=1, DataOption:=1
        .Add Key:=Cells(1, Clm2), SortOn:=0, Order:=2, DataOption:=1
    End With
    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（4／5）

2．PutPartsList

```
Public Sub PutPartsList()

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

    CsvFileName = ActiveWorkbook.Path & "¥PartsList.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
```

（つづく）

２．PutPartsList

（つづき）

```
    Dim acadApp As Object
    Dim acadDoc As Object
    Set acadApp = GetObject(, "AutoCAD.Application.25")
    Set acadDoc = acadApp.ActiveDocument

    Run_CMD = "EXECMOD" & vbCrLf
    acadDoc.SendCommand Run_CMD

    Set acadDoc = Nothing
    Set acadApp = Nothing

End Sub


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

    Dim buf As Variant
    Dim xClm As Long

    LineData = ""
    For xClm = minClm To maxClm
        buf = Cells(xRow, xClm)
        If IsNumeric(buf) Then buf = CStr(buf)
        If LineData = "" Then
            LineData = buf
        Else
            LineData = LineData & "," & buf          '
        End If
    Next

End Function
```

■ AutoLISP ソース（１／１０）

１．EXPPARTSATT.lsp

```
(Defun C:ExpPartsAtt ( / )
;
(vl-load-com)
(princ "¥nStart[ExpPartsAtt]¥n")
(setq startmsec (getvar "MILLISECS"))
(setvar "CMDECHO" 0)
(setvar "DYNMODE" 3)
;
(setq CommandPath (findfile "AutoCAD_CustomCommands"))
(setq CsvFilePath (strcat CommandPath "¥¥AttributeEditor¥¥PartsList.csv"))
(setq PSFilePath (strcat CommandPath "¥¥AttributeEditor¥¥Run_GetPartsList.ps1"))
;
;オブジェクト選択
(setq Inserts (ssget "_I" '((0 . "INSERT"))))
(if
       (= Inserts nil)
       (setq Inserts (ssget '((0 . "INSERT"))))
);if
;
;PartsList.csv オープン
(setq CSV (open CsvFilePath "w"))
;見出し行書き出し
(write-line "番号,Handle,ブロック名,X座標,Y座標,設備NO,品名,寸法,品番" CSV)
;
;                    （つづく）
```

■ AutoLISP ソース（２／１０）

１．EXPPARTSATT.lsp

```
;                              （つづき）
;
;属性取得＆書き出し
(if
       Inserts
       (progn
              (setq InsLen (sslength Inserts))
              (setq InsIdx 0)
              (repeat InsLen
                     (setq No (itoa (1+ InsIdx)))
                     ;ブロック参照のプロパティを取得
                     (setq EntName (ssname Inserts InsIdx))
                     (setq Properties (entget EntName))
                     (setq Handle (cdr (assoc 5 Properties)))
                     (setq Coord (cdr (assoc 10 Properties)))
                     (setq X (car Coord))
                     (setq Y (cadr Coord))
                     (setq BlkName (vla-get-Effectivename (vlax-ename->vla-object EntName)))
                     (if
                            ;関数Get-Insert-Attributesにて 属性を取得
                            (setq Attributes (Get-Insert-Attributes EntName))
                            (progn
                                   ;1行分のデータリストを作成
                                   (setq DataList (list Handle BlkName X Y
                                          (cdr(assoc "設備NO" Attributes))
                                          (cdr(assoc "品名" Attributes))
                                          (cdr(assoc "寸法" Attributes))
                                          (cdr(assoc "品番" Attributes))
                                   ))
                                   ;リスト内の nil を "ーーー" に差し替え
                                   (setq DataList (subst "ーーー" nil DataList))
                                   (setq DataList (subst "ーーー" "" DataList))
;
;                              （つづく）
```

■ AutoLISP ソース（３／１０）

１．EXPPARTSATT.lsp

（つづき）

```
                                        ;1行分のカンマ区切り文字列を作成(同時にリスト内の数値は文字列に変換)
                                        (setq WriteData No)
                                        (setq LstLen (length DataList))
                                        (setq LstIdx 0)
                                        (repeat LstLen
                                                (setq Data (nth LstIdx DataList))
                                                (if
                                                        (numberp Data)
                                                        (setq Data (rtos Data 2 2))
                                                );if
                                                (setq WriteData (strcat WriteData "," Data))
                                                (setq LstIdx (1+ LstIdx))
                                        );repeat
                                        ;1行分のデータを書き込み
                                        (write-line WriteData CSV)
                                );progn
                        );if
                        (setq InsIdx (1+ InsIdx))
                );repeat
        );progn
);if
;エクセルマクロ[GetPartsList]起動
(startapp "powershell" (strcat "-ExecutionPolicy Bypass -File ¥"" PSFilePath "¥""))
;PartsList.csv クローズ
(close CSV)
;
(setvar "CMDECHO" 1)
(setq endmsec (getvar "MILLISECS"))
(princ "¥n*** end")(princ (- endmsec startmsec))
(princ)
)
```

■ AutoLISP ソース（４／１０）

２．MODPARTSATT.lsp

```
(Defun C:ModPartsAtt ( / CommandPath VbsFilePath)
;
(princ "¥n[Run_PutPartsList]¥n")
(setq startmsec (getvar "MILLISECS"))
(setvar "CMDECHO" 0)
(setvar "DYNMODE" 3)
(command-s "._COMMANDLINE")
;
(setq CommandPath (findfile "AutoCAD_CustomCommands"))
(setq PSFilePath (strcat CommandPath "¥¥AttributeEditor¥¥Run_PutPartsList.ps1"))
;
;エクセルマクロ[PutPartsList]起動
(startapp "powershell" (strcat "-ExecutionPolicy Bypass -File ¥"" PSFilePath "¥""))
;
(setvar "CMDECHO" 1)
(princ)
);defun
;-------------------------------------------------------------------------------------------------------------------
(Defun C:EXECMOD ( / CommandPath CsvFilePath CSV Hder HdLst Dat DatLst HdlIdx Hdl AttLst AttLen Idx AttName AttDat
AttDotPear InsName)
;
(princ "¥nStart[Mod]¥n")
(setq startmsec (getvar "MILLISECS"))
(setvar "CMDECHO" 0)
(setvar "DYNMODE" 3)
(command-s "._COMMANDLINE")
;
(setq CommandPath (findfile "AutoCAD_CustomCommands"))
(setq CsvFilePath (strcat CommandPath "¥¥AttributeEditor¥¥PartsList.csv"))
;
```

■ AutoLISP ソース（５／１０）

２．MODPARTSATT.lsp

（つづき）

```
;PartsList.csv オープン
(setq CSV (open CsvFilePath "r"))
;
(setq Hder (read-line CSV))
(setq HdLst (Change-Csv-to-List Hder))
(while
       (and
              (setq Dat (read-line CSV))
              (setq DatLst (Change-Csv-to-List Dat))
       );and
       ;
       (setq HdlIdx (vl-position "Handle" HdLst))
       (setq Hdl (nth HdlIdx DatLst))
       (setq AttLst nil)
       (setq AttLen 4)
       (setq Idx 5)
       (repeat AttLen
              (setq AttName (nth Idx HdLst))
              (setq AttDat (nth Idx DatLst))
              (if
                     (/= AttDat "ーーー")
                     (progn
                            (setq AttDotPear (cons AttName AttDat))
                            (setq AttLst (append AttLst (list AttDotPear)))
                     );progn
              );if
              (setq Idx (1+ Idx))
       );repeat
       (setq InsName (handent Hdl))
       (Modify-Attributes InsName AttLst)
);while
```

（つづく）

■ AutoLISP ソース（６／１０）

２．MODPARTSATT.lsp

（つづき）

```
;
(close CSV)
;
(setvar "CMDECHO" 1)
(setq endmsec (getvar "MILLISECS"))
(princ "¥n*** end ")(princ (- endmsec startmsec))(princ " msec")
(princ)
);defun
;-------------------------------------------------------------------------------------------------------------------
```

■ AutoLISP ソース（７／１０）

３．Get-Insert-Attributes.lsp

```
;INSERTの属性(ATTRIB)のドットペアリストを返す関数
;
(Defun Get-Insert-Attributes (InsName / InsProperties AttFlg NextName NextProperties Type AttName AttValue AttList)
	(setq AttList nil)
	(setq InsProperties (entget InsName))
	(if
		(= (cdr (assoc 66 InsProperties)) 1)	;属性あり
		(progn
			;■[INSERT]内を検索
			(setq NextName InsName)
			(while
				(and
					(setq NextName (entnext NextName))
					(setq NextProperties (entget NextName))
					(setq Type (cdr (assoc 0 NextProperties)))
					(/= Type "SEQEND")
				);and
				(if
					(and
						(= Type "ATTRIB")
						(or
							(not (assoc 60 NextProperties))
							(= (cdr (assoc 60 NextProperties)) 0)
						)or
					);and
					(progn
						(setq AttName (cdr (assoc 2 NextProperties)))
						(setq AttValue (cdr (assoc 1 NextProperties)))
						(setq AttList (append AttList (list (cons AttName AttValue))))
					):progn
				);if
			);while
```

（つづく）

■ AutoLISP ソース（８／１０）

３．Get-Insert-Attributes.lsp

（つづき）

```
                        ;■[BLOCK]内を検索
                        (setq InsProperties (entget InsName))
                        (setq BlkName (cdr (assoc 2 InsProperties)))
                        (setq NextBlk (tblobjname "Block" BlkName))
                        (while
                               (and
                                      (setq NextBlk (entnext NextBlk))          ;ネストされたオブジェクトあり
                                      (setq BlkProperties (entget NextBlk))
                                      (setq Type (cdr (assoc 0 BlkProperties)))
                               );and
                               (if
                                      (and
                                             (= Type "ATTDEF")
                                             (= (boole 1 (cdr (assoc 70 BlkProperties)) 2) 2)
                                             (or
                                             (not (assoc 60 BlkProperties))
                                             (= (cdr(assoc 60 BlkProperties)) 0)
                                      )
                                      );and
                                      (progn
                                             (setq AttName (cdr(assoc 2 BlkProperties)))
                                             (setq AttValue (cdr(assoc 1 BlkProperties)))
                                             (setq AttList (append AttList (list (cons AttName AttValue))))
                                      );progn
                               );if
                        );while
                 );progn
          );if
    ;
    AttList
    )
```

（つづく）

■ AutoLISP ソース（９／１０）

３．Get-Insert-Attributes.lsp

（つづき）

```
;引数[Str]    : カンマ区切り文字列
;戻り値[Lst]  : 引数をカンマにて分割したリスト
;
(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（１０／１０）

４．Modify-Attributes.lsp

```
;引数1：InsName              INSERTの図形名
;引数2：AttValueList ドットペアリスト（(属性名 . 属性値)（属性名 . 属性値）･･･）
;戻り値：なし
;
(Defun Modify-Attributes (InsName AttValueList / BlkOrgName NextName NextProperties NextType AttName NewAttValue)
;
	(setq BlkOrgName (vla-get-Effectivename (vlax-ename->vla-object InsName)))
	(setq NextName InsName)
	(while
		;ネストされた図形名が存在し、そのタイプが"SEQEND"ではない
		(and
			(setq NextName (entnext NextName))
			(setq NextProperties (entget NextName))
			(setq NextType (cdr (assoc 0 NextProperties)))
			(/= NextType "SEQEND")
		);and
		(if
			;属性情報の属性名と同じ属性名が引数2:AttValueList に存在する
			(and
				(= NextType "ATTRIB")
				(setq AttName (cdr (assoc 2 NextProperties)))
				(setq OldValue (cdr (assoc 1 NextProperties)))
				(assoc AttName AttValueList)
				(setq NewValue (cdr (assoc AttName AttValueList)))
				(/= NewValue OldValue)
				(setq NextProperties (subst (cons 1 NewValue) (cons 1 OldValue) NextProperties))
			);and
			(entmod NextProperties)
		);if
	);while
	(princ)
)
```

■ PowerShellソース（１／２）

１．Run_GetPartsList

```
$scriptPath = $MyInvocation.MyCommand.Path
$folderPath = Split-Path -Parent $scriptPath
$excelFileName = "PartsList.xlsm"
$macroName = "GetPartsList"
$fullExcelPath = Join-Path $folderPath $excelFileName
# Excel アプリケーション取得または起動
$excelApp = $null
try {
    $excelApp = [Runtime.InteropServices.Marshal]::GetActiveObject("Excel.Application")
} catch {
    $excelApp = New-Object -ComObject Excel.Application
    $excelApp.Visible = $true
}
# ブックを開くまたは既存のブックを参照
$workbook = $null
foreach ($wb in $excelApp.Workbooks) {
    if ($wb.FullName -eq $fullExcelPath) {
        $workbook = $wb
        break
    }
}
if (-not $workbook) {
    $workbook = $excelApp.Workbooks.Open($fullExcelPath)
}
# マクロ実行
$excelApp.Run("$excelFileName!$macroName")
# COM オブジェクトの解放
[System.Runtime.InteropServices.Marshal]::ReleaseComObject($workbook) | Out-Null
[System.Runtime.InteropServices.Marshal]::ReleaseComObject($excelApp) | Out-Null
[GC]::Collect()
[GC]::WaitForPendingFinalizers()
```

■ PowerShellソース（２／２）

２．Run_PutPartsList

```
$MyFolderPath = Split-Path -Parent $MyInvocation.MyCommand.Definition
$ExcelBookName = "PartsList.xlsm"
$Macro = "PutPartsList"
$ExcelBookFullName = Join-Path $MyFolderPath $ExcelBookName

# Excel アプリケーションを取得または作成
try {
    $ExcelAppObj = [Runtime.InteropServices.Marshal]::GetActiveObject("Excel.Application")
    $ExcelOpened = $false
    try {
        $ExcelBookObj = $ExcelAppObj.Workbooks.Item($ExcelBookName)
    } catch {
        $ExcelBookObj = $ExcelAppObj.Workbooks.Open($ExcelBookFullName)
        $ExcelOpened = $true
    }
} catch {
    $ExcelAppObj = New-Object -ComObject Excel.Application
    $ExcelAppObj.Visible = $true
    $ExcelBookObj = $ExcelAppObj.Workbooks.Open($ExcelBookFullName)
    $ExcelOpened = $true
}

# マクロの実行
$macroFullName = "$ExcelBookName!$Macro"
$ExcelAppObj.Run($macroFullName)

# 解放処理（COM オブジェクトのクリーンアップ）
[System.Runtime.InteropServices.Marshal]::ReleaseComObject($ExcelBookObj) | Out-Null
[System.Runtime.InteropServices.Marshal]::ReleaseComObject($ExcelAppObj) | Out-Null
[GC]::Collect()
[GC]::WaitForPendingFinalizers()
```