RvCamDLL 使用手冊

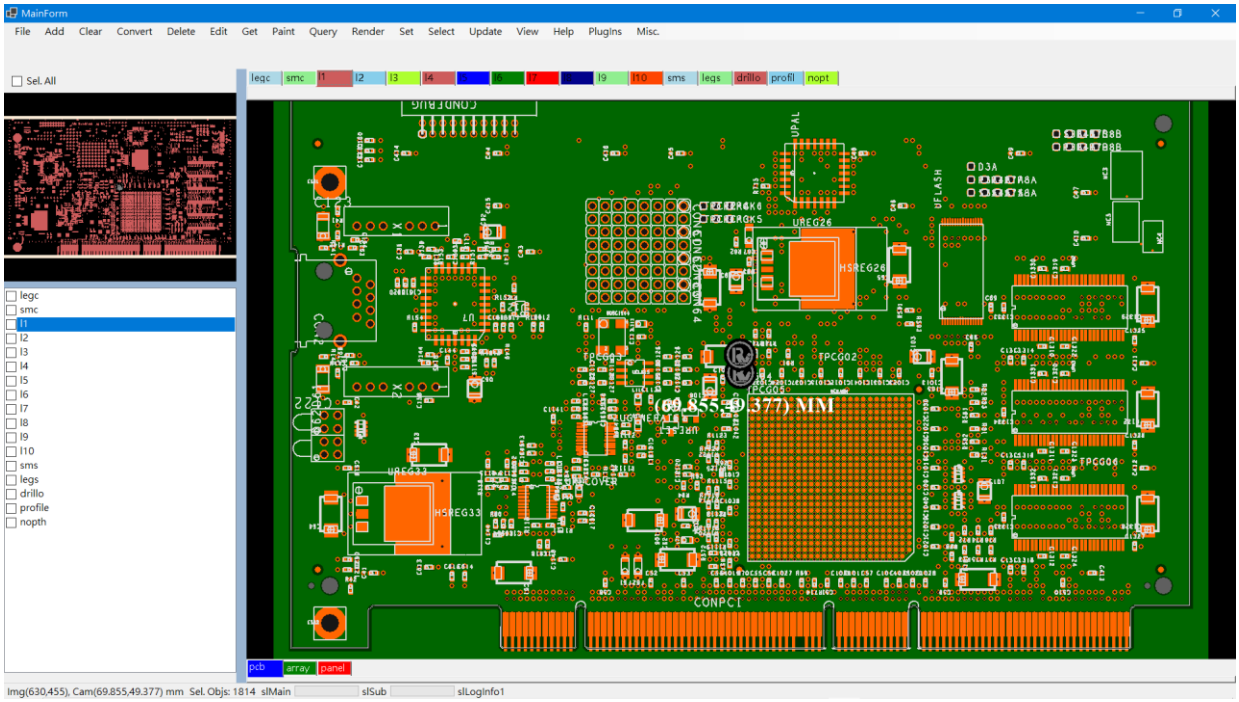
影量科技 RasVector Technology

Daniel Lu dan59314@gmail.com

教學影片； 點擊連結

使用手冊： 點擊連結

源碼下載： 點擊連結



目錄

增修紀錄

2024/7/22 ----------

新增 RvCam_Edit_Layer_Align( )

RvCam_Edit_Step( )

RvCam_Edit_Step_StepRepeat( )

2024/7/19 ----------

新增 RvCam_Get_BlobCXY_Scan0( )

RvCam_Get_BlobCXY_Layer( )

2024/7/17 ----------

新增 RvCam_Edit_Layer( ) 函式

2024/7/12 ----------

初版

RvCamDLL 特色

專為半導體/PCB 的 AOI / AVI / AOM 應用而設計

多種 CAD 檔案格式輸入

ODB++/TGZ、Gerber274X、NC 鑽孔檔、AutoCad DXF、IPC/356/mnf2、RVC、SSF….。

多種 CAD 檔案格式輸出

Gerber274X、NC 鑽孔檔、AutoCad DXF、RVC、SSF…。

支援開發者自編 CAD 讀存檔和資料處理外掛開發

可透過 DLL 函式將外部資料傳入整合核心，程式開發者可以自行開發和販售其他 CAD 格式檔案讀取、儲存和資料處理的外掛 DLL。其他只需利用 TestRvCamDLL 程式的所有圖形操作、編輯介面和所有完整功能。不需自行開發完整 CAM 程式。

自訂高解析圖形 RawData 和圖檔輸出

各種 CAD 檔案，自訂解析度(mm/Pixel)，轉換成高解析影像圖檔、傳回記憶體 Raw 資料…。

完整 CAM 圖形操作界面和豐富編輯功能

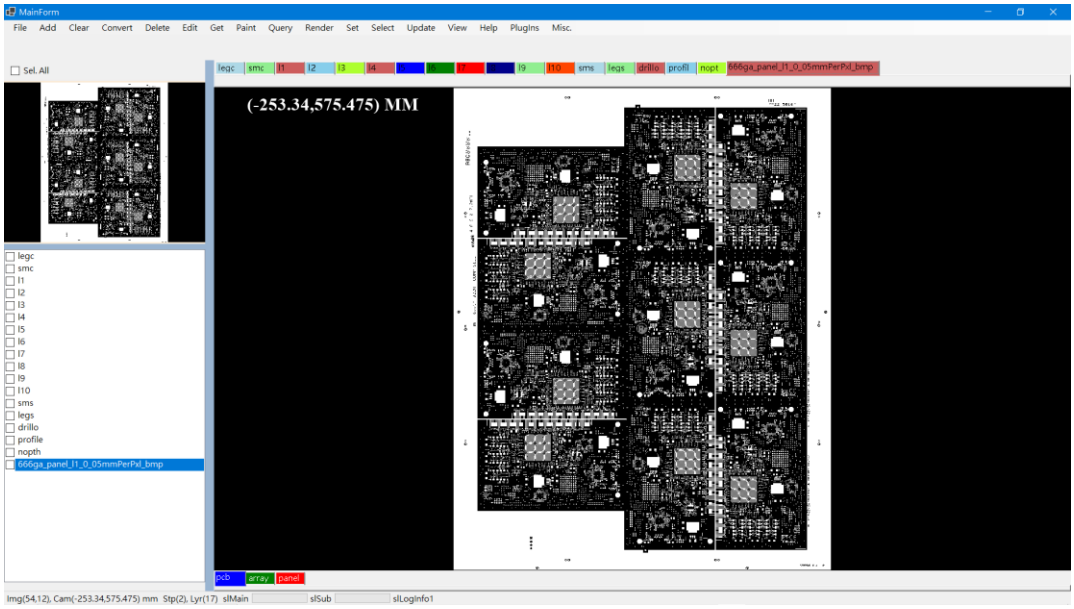
含 C#完整程式碼，除了無介面的處理函式外，包含豐富的介面操作、圖形顯示功能。

支援同時讀入 CAD/BMP 重疊顯示比對

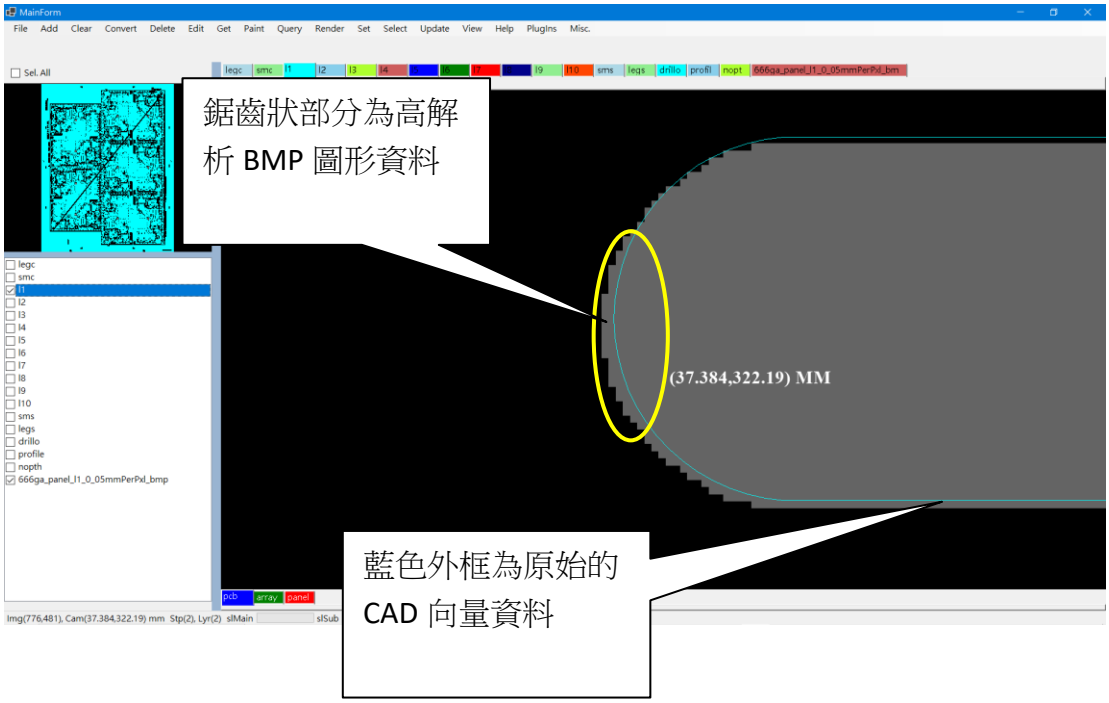
讀入 CAD 檔案 和 BMP 圖檔後，可選取兩層重疊顯示，檢查差異。
BMP 圖檔來源，例如 CAM 輸出的高解析圖形，或是檢查設備掃描得到的圖形資料。
重疊顯示後可以檢視差異的部分。

原始 CAD 圖形

讀入高解析 BMP 圖形 (此例為 RvCamDLL 以 0.01Mm/Pixel 輸出 2.6 GB bmp 圖檔)



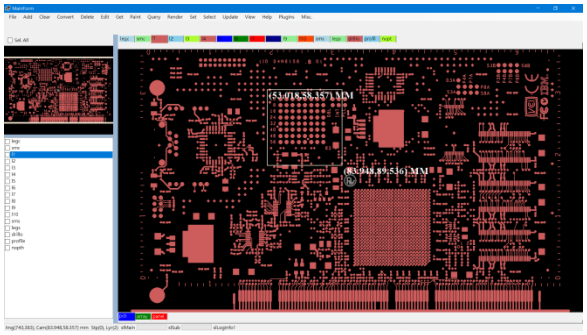
可同時重疊顯示 CAD 和 BMP 圖，比較差異



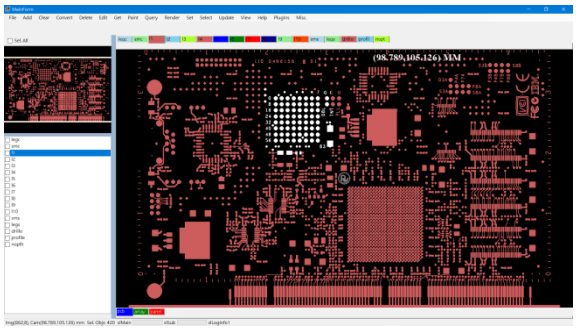
圖形資料編輯功能

詢問、選取、刪除、複製、Mark、旋轉、位移、鏡射 XY、複製排版…。
新增一層、複製多層、選取物件複製到新層…多種 CAD 層資料編輯功能。

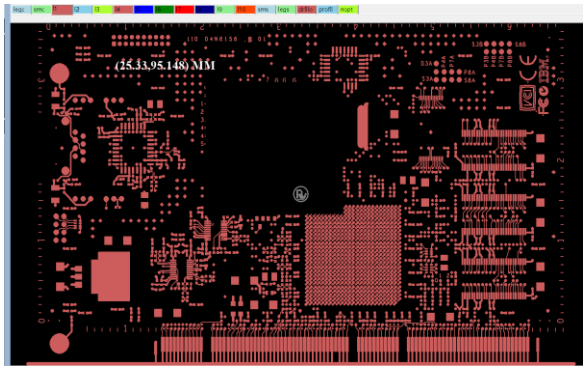
滑鼠框選



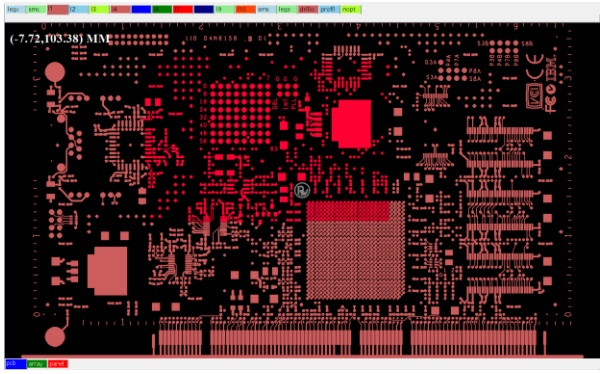
選取



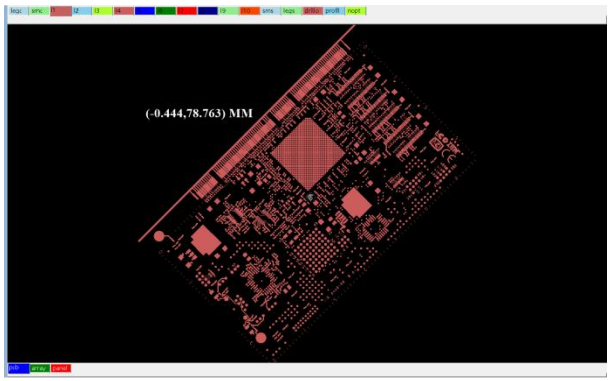
刪除



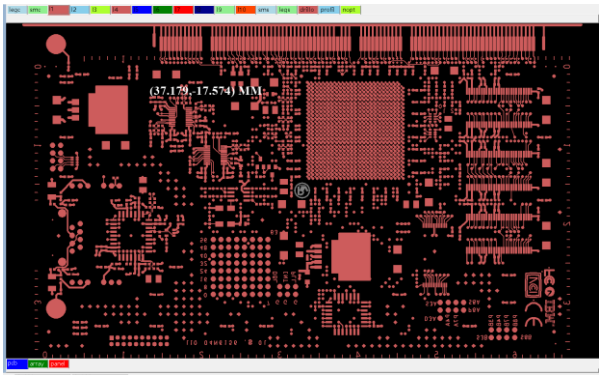
Mark



旋轉



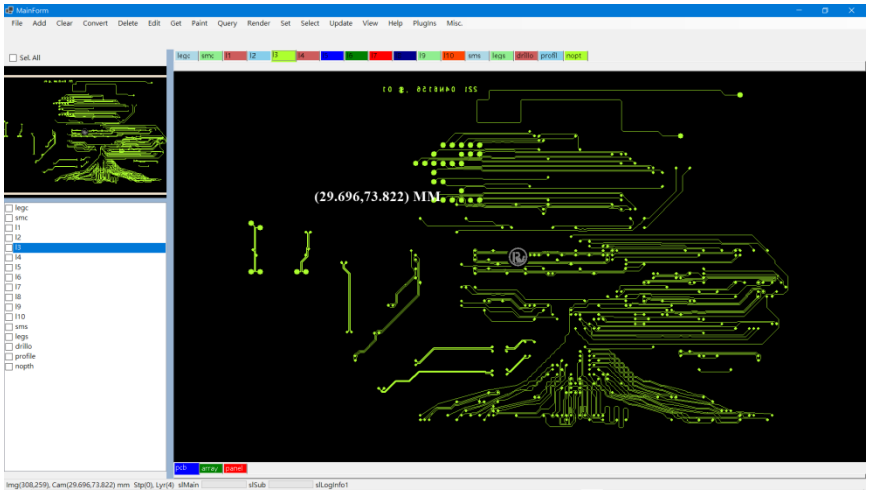
鏡射



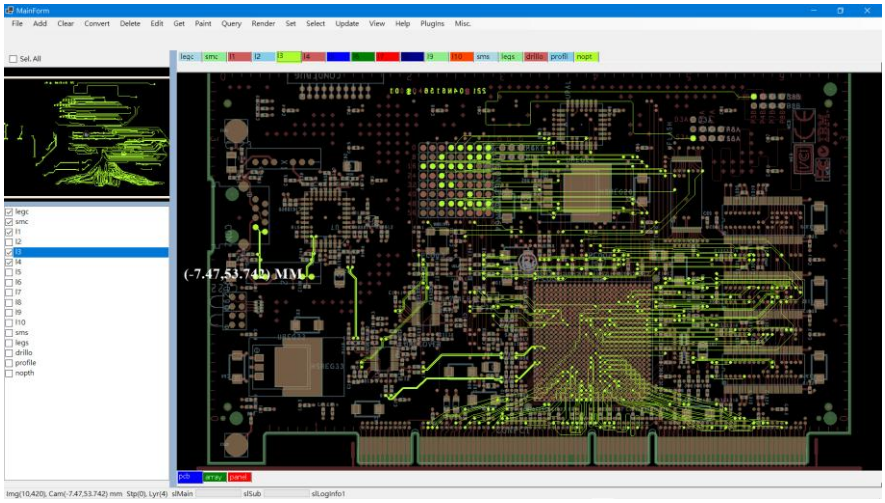
繪圖顯示模式

螢幕上繪圖、旋轉、位移、縮放顯示、各種滑鼠顯示操作。

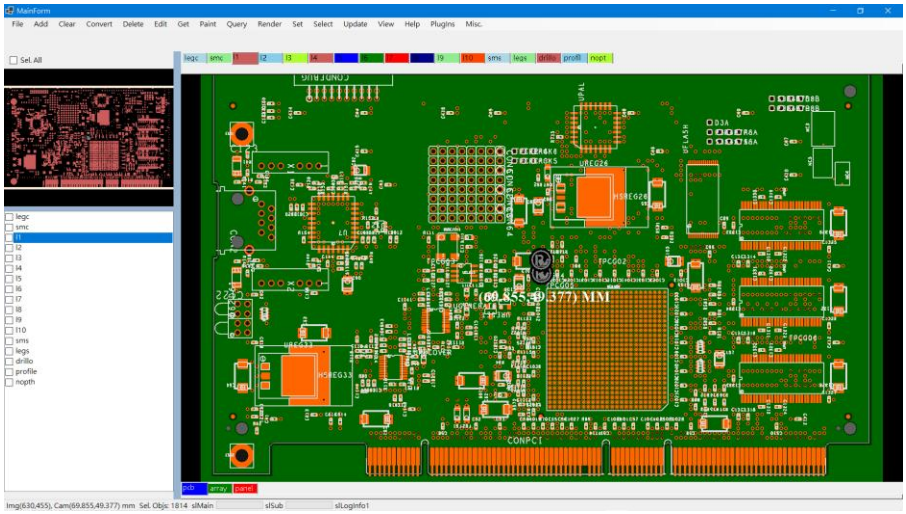
單層模式



多層透明模式



擬真顯示模式



快速教學

下載 TestRvCamDLL_CS C#完整程式碼

包含完整測試程式碼，外掛程式撰寫範例程式碼。下載測試專案完整程式碼案。

常用開檔、轉檔、算圖與存圖功能

背景開啟 ODB++/TGZ 檔，算圖傳回 RawData 圖形指標或儲存圖檔

1. RvCam_Render_Image_ODB( )
2. RvCam_Load_ODB( ) + RvCam_Render_Image_StepLayer( )

背景開啟 CAD 檔案，算圖傳回 RawData 圖形指標或儲存圖檔

1. RvCam_Render_Image_CAD( )
2. RvCam_Load_CAD( ) + RvCam_Render_Image_StepLayer( )

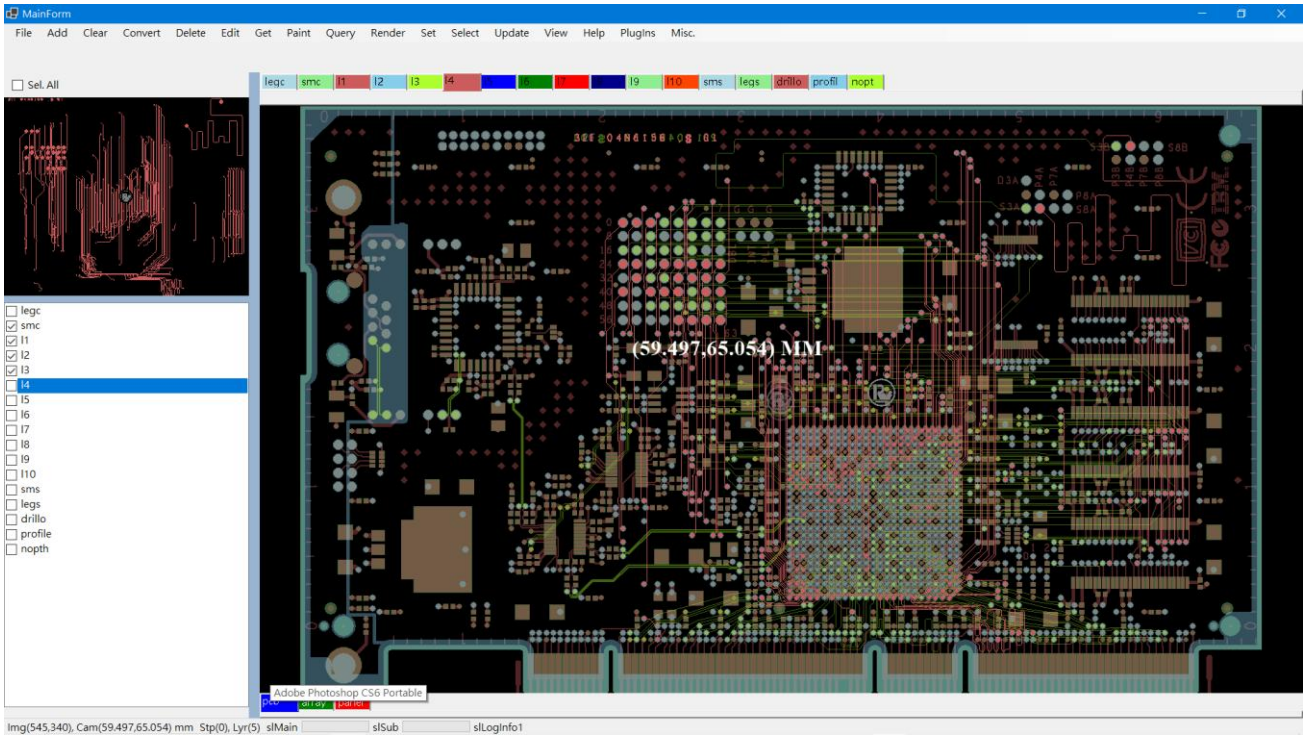
背景開啟 ODB++/TGZ 檔案，轉存 CAD 檔案

1. Convert_File_OdbTGZ_To_CAD( )
2. RvCam_Load_ODB( ) + RvCam_Save_CAD( )

背景開啟 CAD 檔案，轉存 CAD 檔案

1. Convert_File_CAD_To_CAD( )
2. RvCam_Load_CAD( ) + RvCam_Save_CAD( )

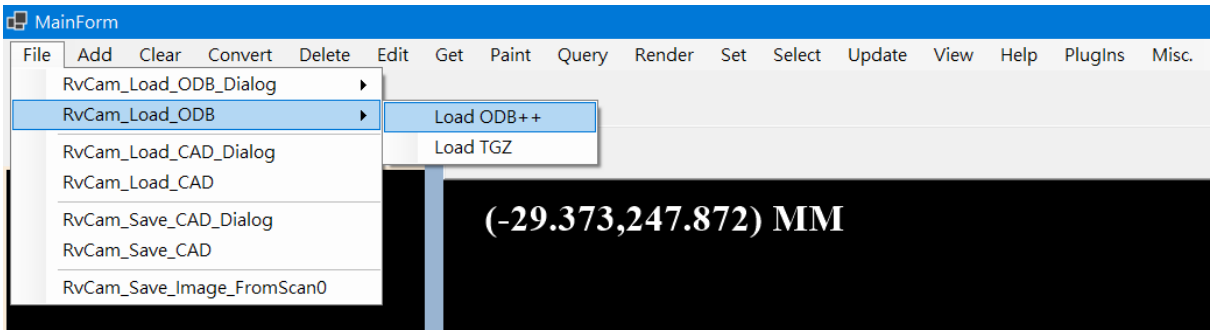
執行 TestRvCamDD_CS.exe



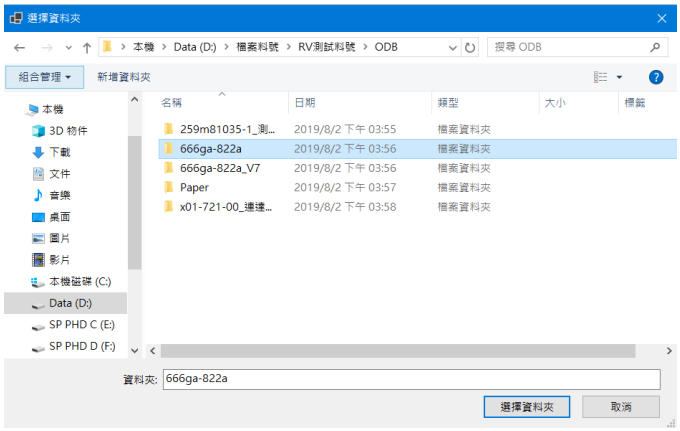
以下執行的所有功能，都可以由程式碼的執行入口檢視 DLL 函數的呼叫使用方式。

開啟 ODB++料號

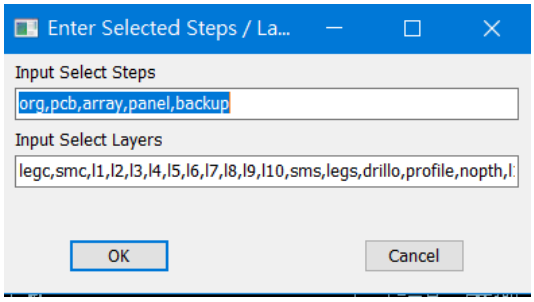
選單選取開啟 ODB 功能



瀏覽到 ODB 目錄，按下 [選擇資料夾]。

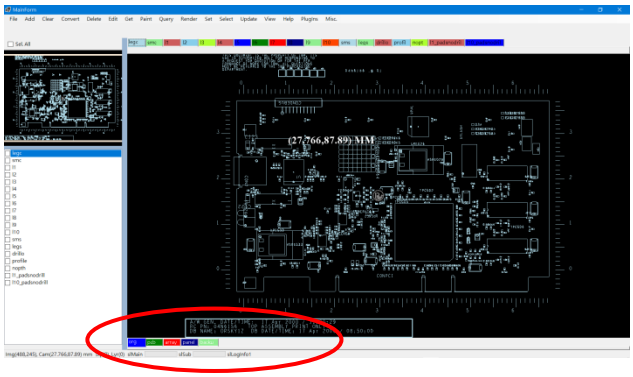


輸入要讀取的 Step/Layer 名稱。

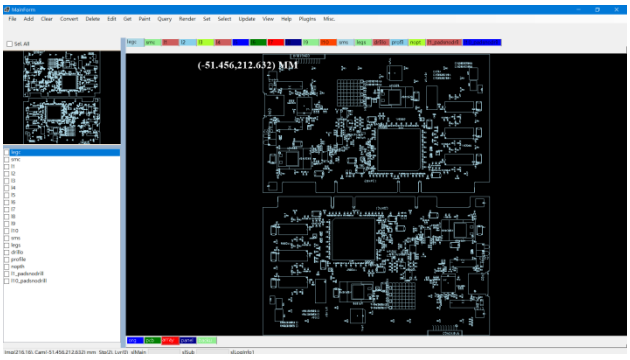


Step 切換

在下方標籤，切換要顯示的 Step (PCB)

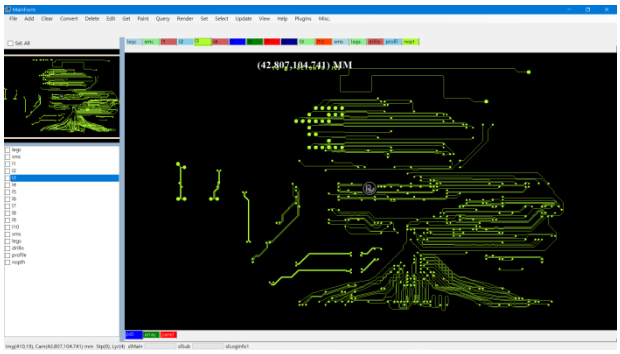


切換 Step (Array)



Layer 切換

在上方標籤切換要顯示的 Layer 名稱。

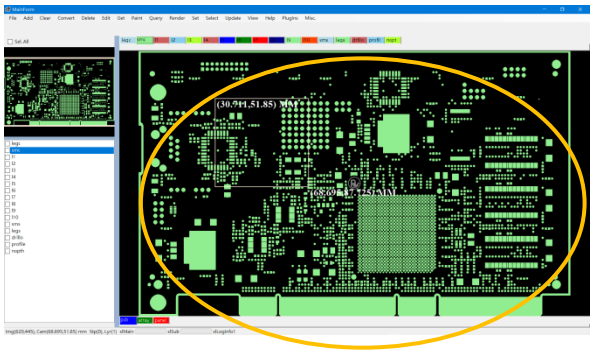


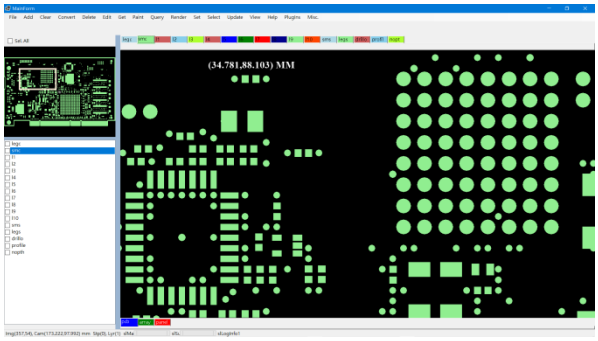
多層顯示

在左邊層名清單核選要重疊顯示的層名。

主繪圖區滑鼠操作

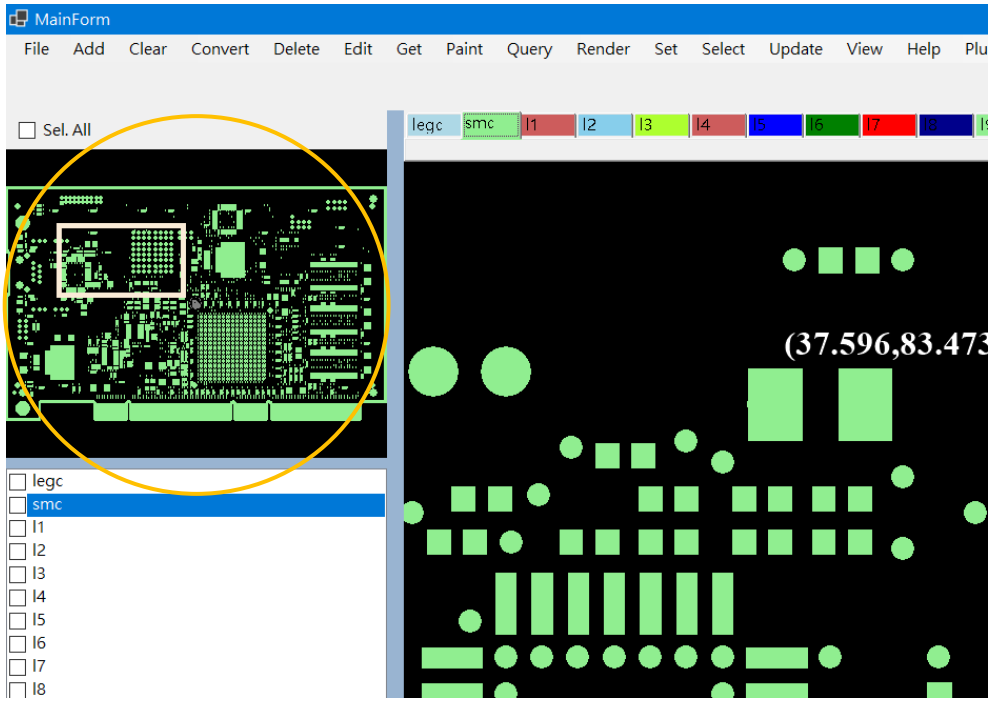
中間主繪圖區視角操作功能 ： 在主繪圖區上按下
滑鼠左鍵拉框放大 / 中鍵全圖顯示 / 右鍵拖移視角 / 滾輪縮放視角 View。





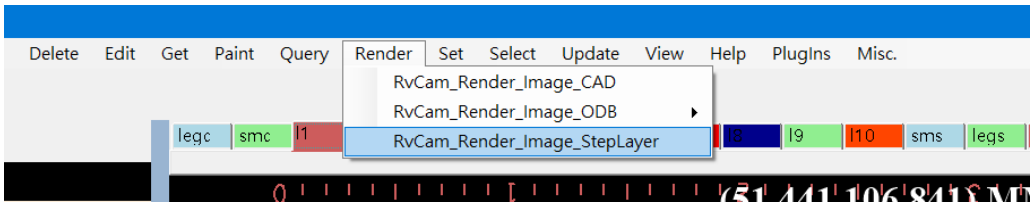
子繪圖區滑鼠操作

左上方子繪圖區滑鼠視角操作功能 ： 在子繪圖區上岸下
左鍵拉框，主繪圖區同步放大顯示 / 右鍵拖移視角，主繪圖區同步更新圖形。

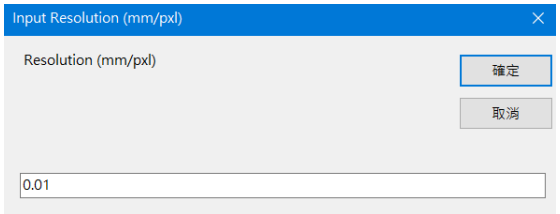


輸出高解析圖形

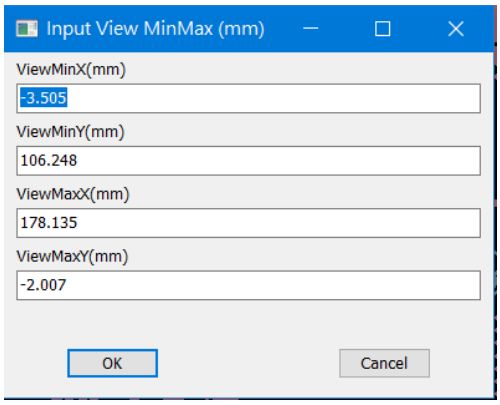
選取高解析出圖功能



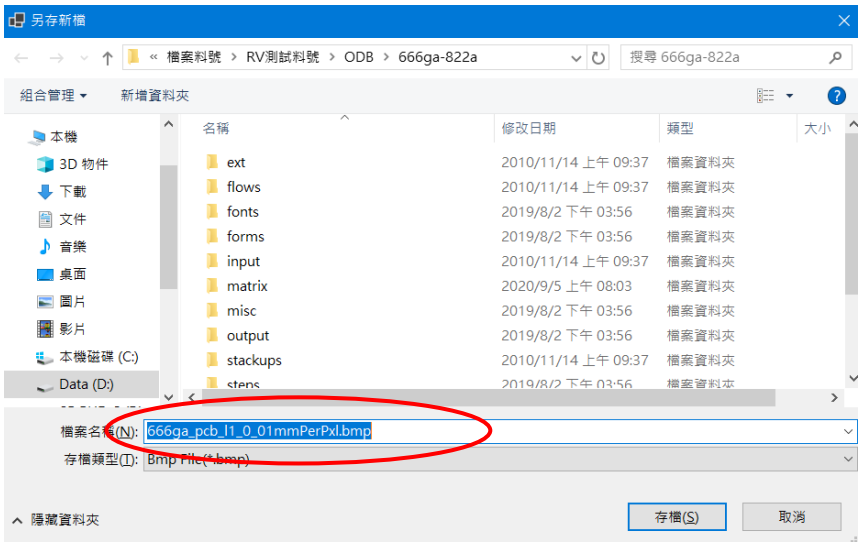
輸入解析度 (mm/pixel)



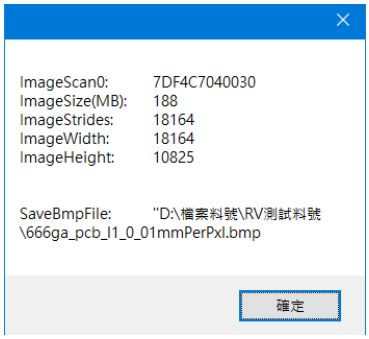
輸入出圖範圍後按下[OK]，或者按[Cancel] 自動輸出整張圖。



選取輸出路徑，輸入 bmp 存檔名稱，按下 [存檔]，程式將在記憶體內算圖後另存圖形檔。
或者按下[取消]不存圖檔，只在記憶體內算圖。

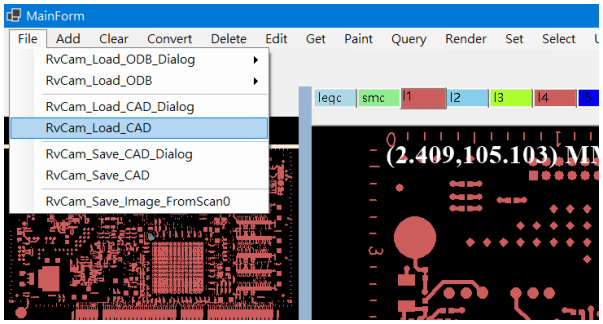


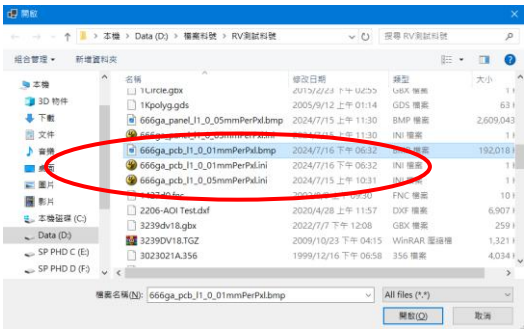
最後程式將在記憶體內算圖，並傳回記憶體中圖形資料的位址。



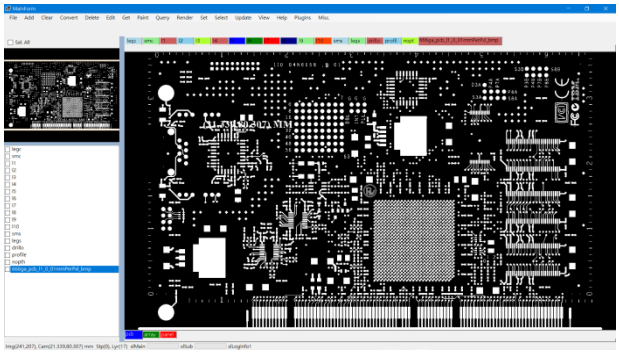
將剛輸出的圖檔讀入，重疊顯示看看 CAD 與 BMP 圖形重疊顯示的效果。

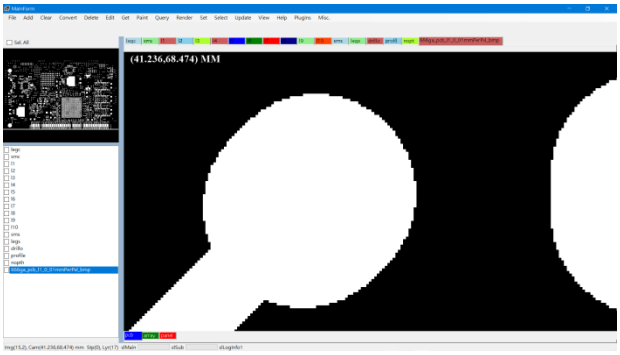
Bmp 圖檔，選取開啟 CAD 檔案功能，程式會自動判別。





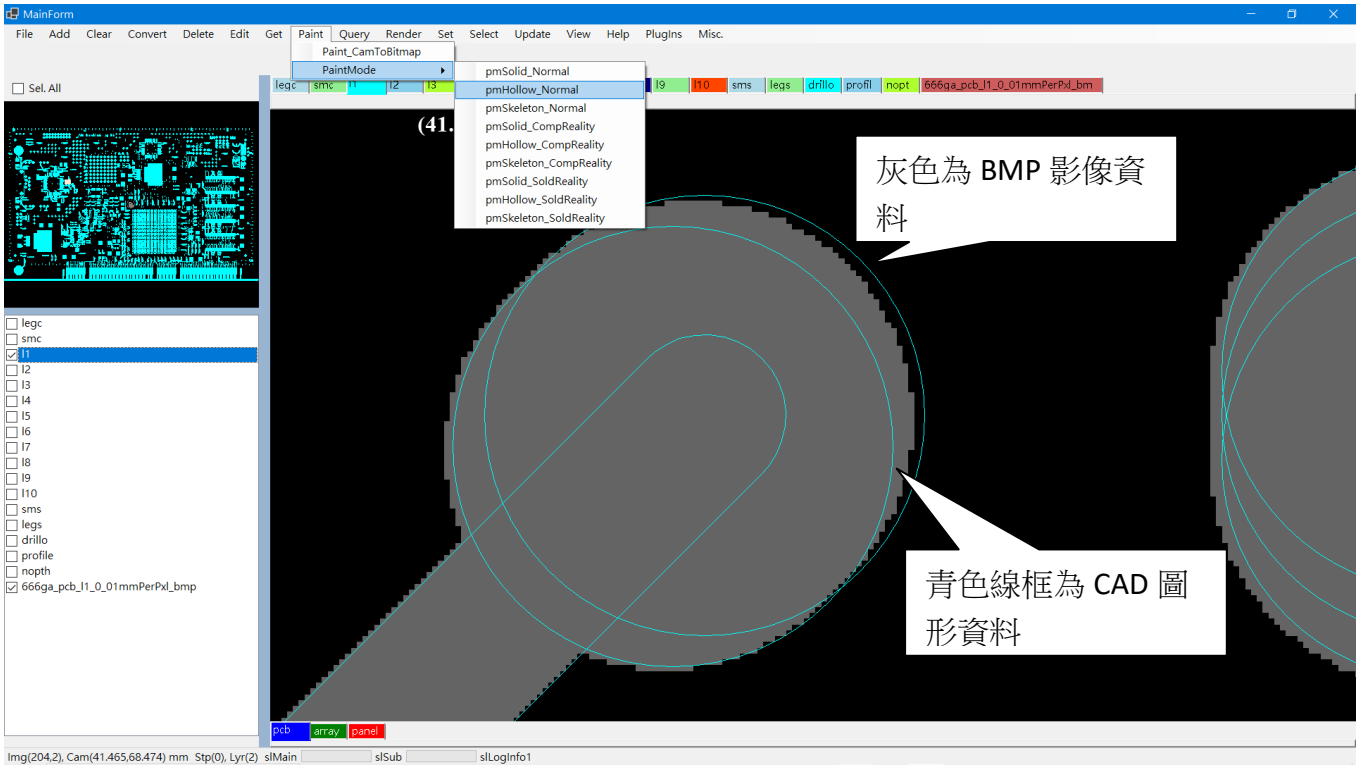
讀入的 BMP 圖形顯示如下，放大後會看到鋸齒狀，表示是像素圖檔。





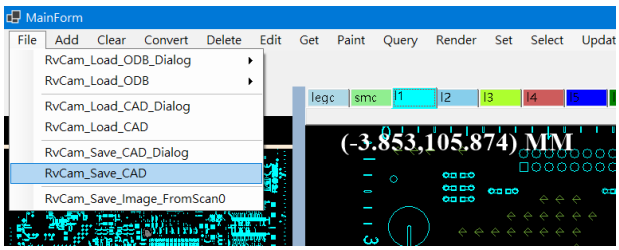
在左方層名清單核選原始 CAD 層和剛讀入的 Bmp 圖檔，以”pmHollow_Normal”模式重疊顯示。

即可看到 CAD 和 BMP 圖形重疊顯示在主繪圖區，方便檢視差異。

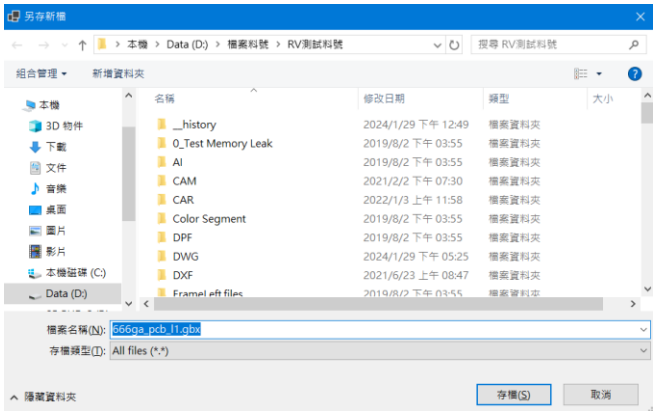


儲存 CAD 檔案

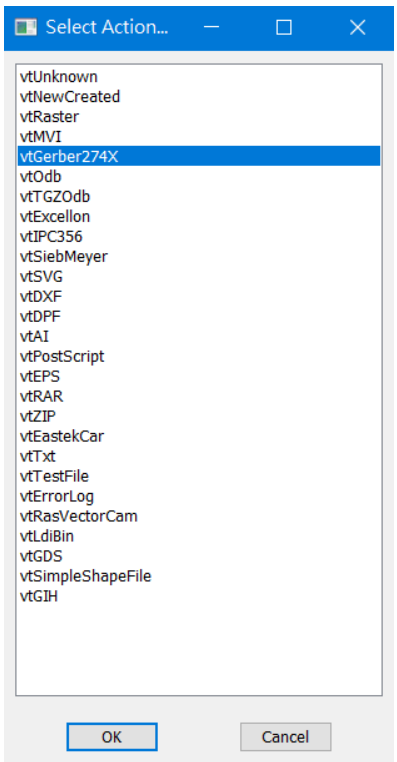
選取存檔功能。



輸入存檔名稱。



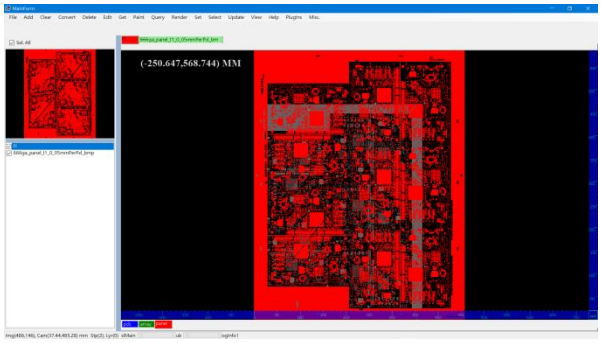
選取存檔檔案格式，按下[OK]後即可儲存檔案。



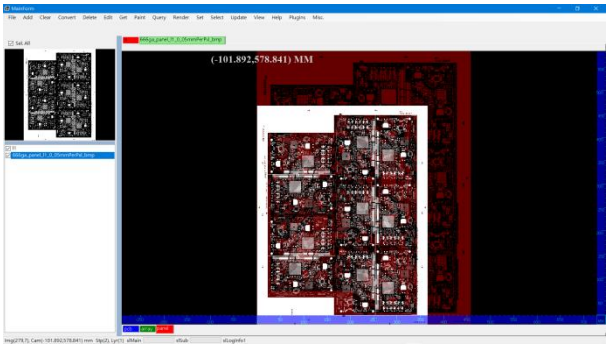
CAD 圖形與 BMP 影像的對位和解析度校正

讀入 CAD 檔和 BMP 影像檔

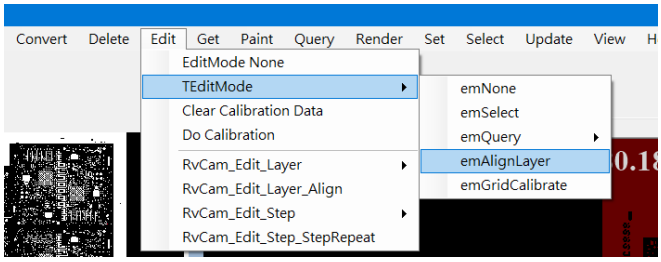
CAD 層



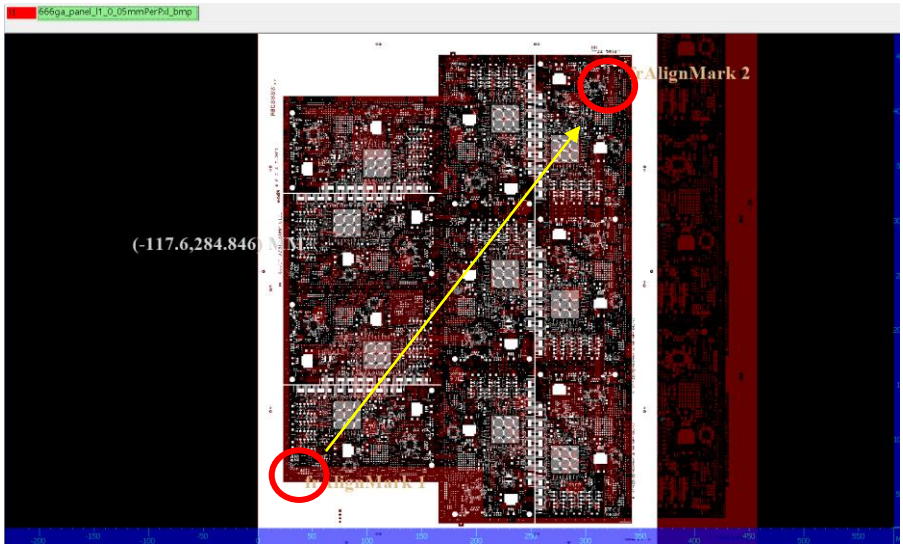
BMP 影像層



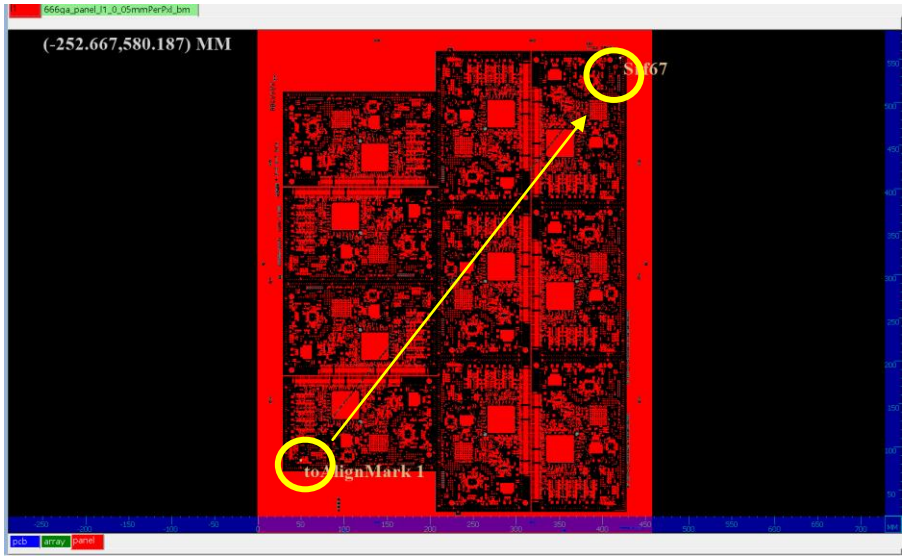
選取 Layer Align 功能



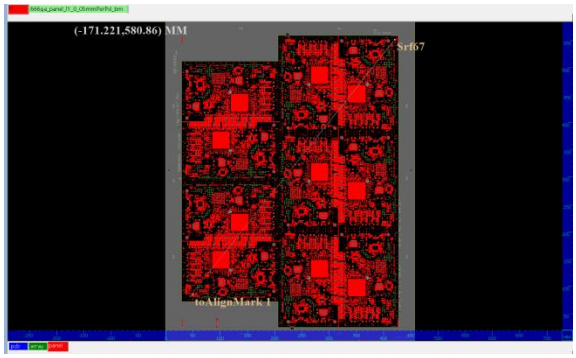
切換到 BMP 層點擊對角線的兩個定位點

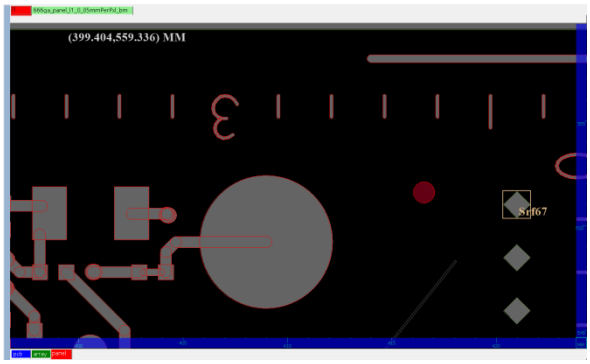


切換到 CAD 層點擊對角線相同的兩個對位點



完成對位和解析度校正





影像補償表建立

讀入校正資料的 CAD 檔案和設備掃描 Image 圖檔

CAD 層

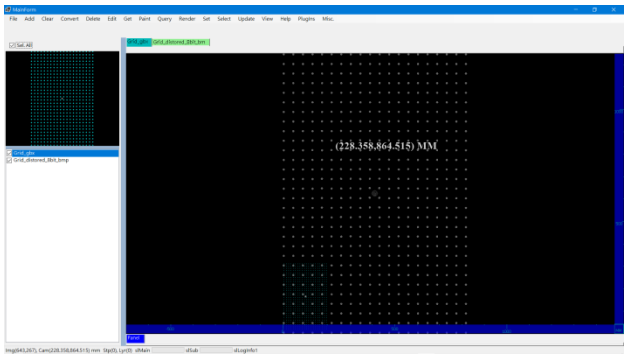
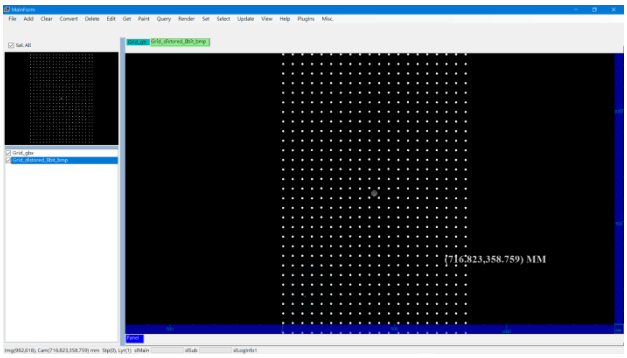
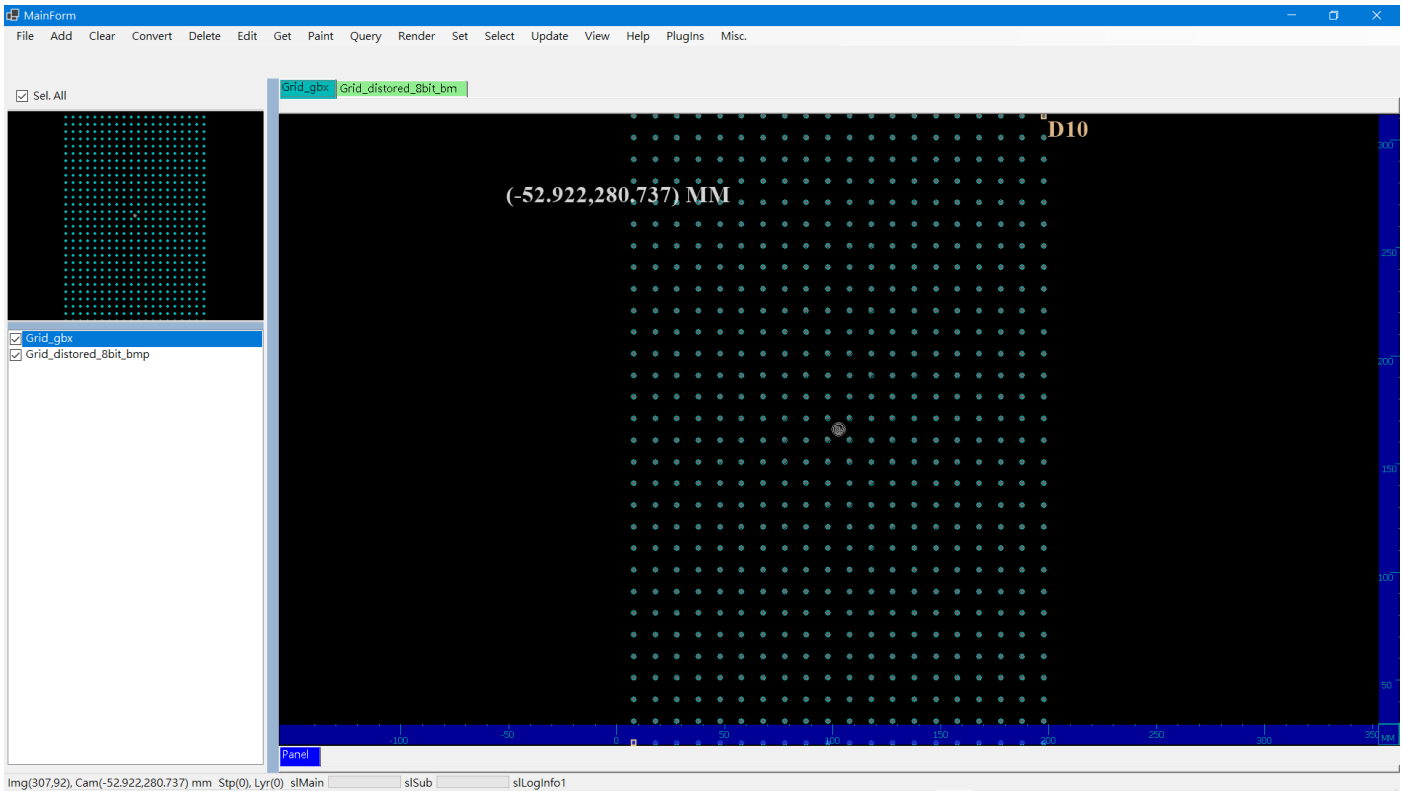


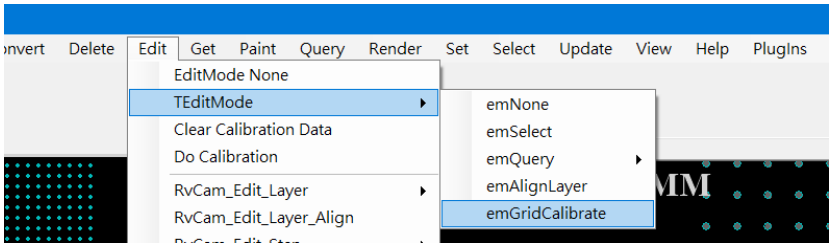
Image 影像層



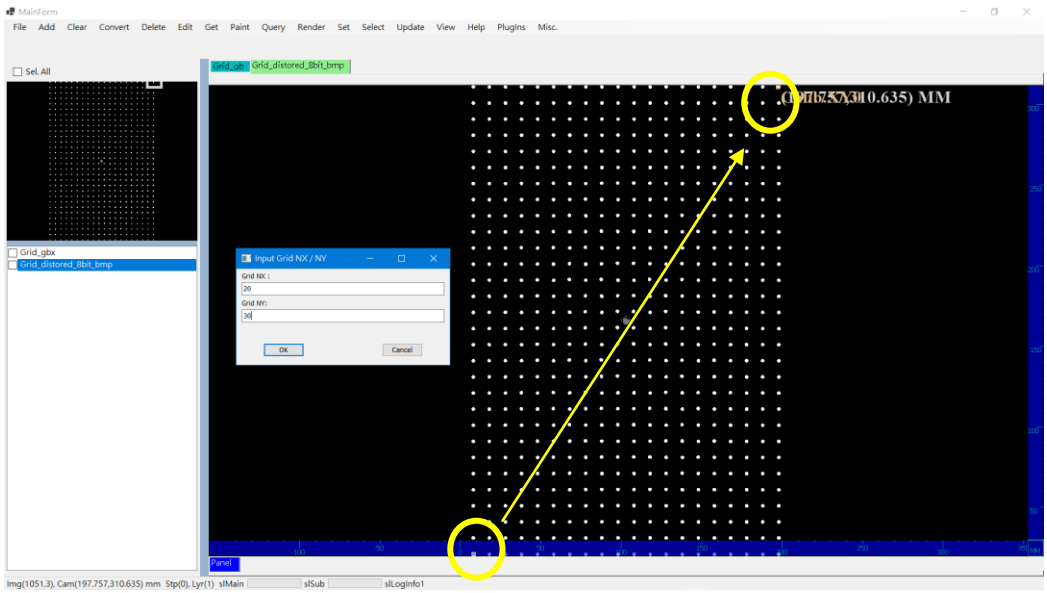
參考上節，做好對位和解析度校正



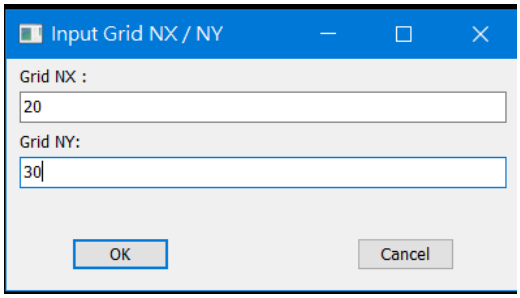
執行格點校正功能



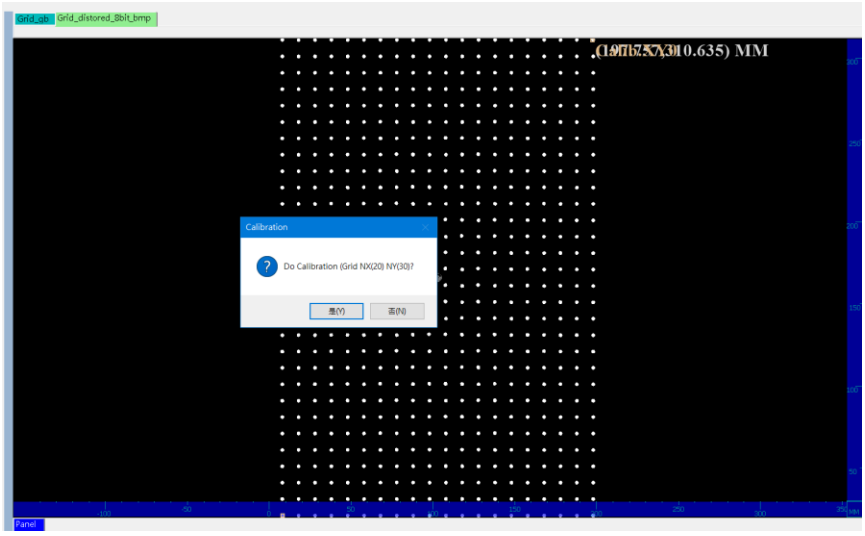
在 Image 層點擊對角線兩點



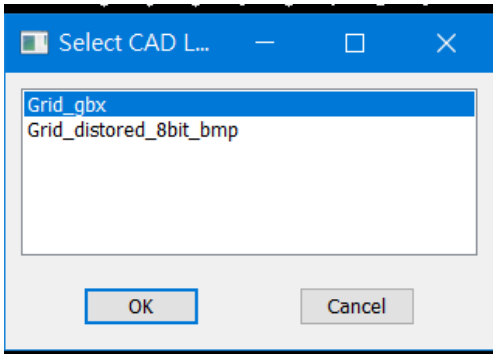
輸入格點的 X 數量 和 Y 數量

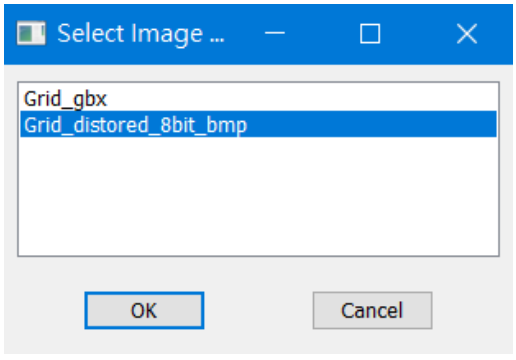


執行補償校正

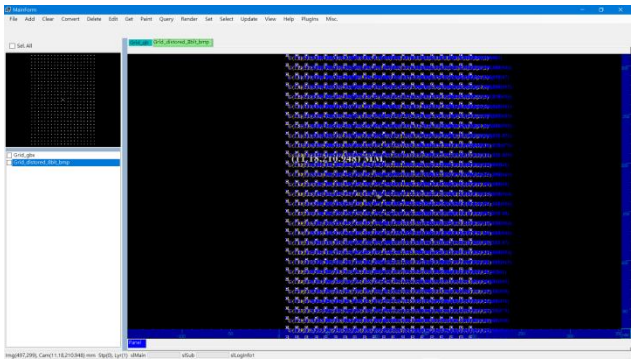


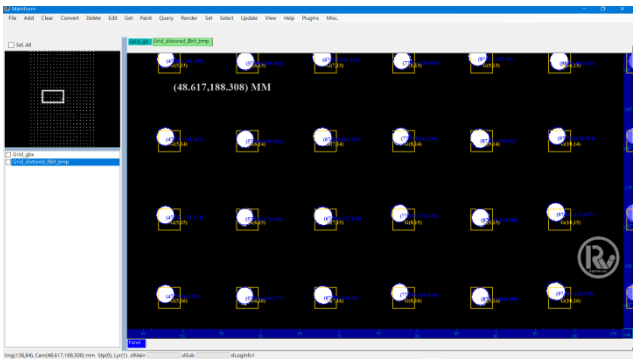
先選取 CAD 層，在選取 Image 層





校正完成



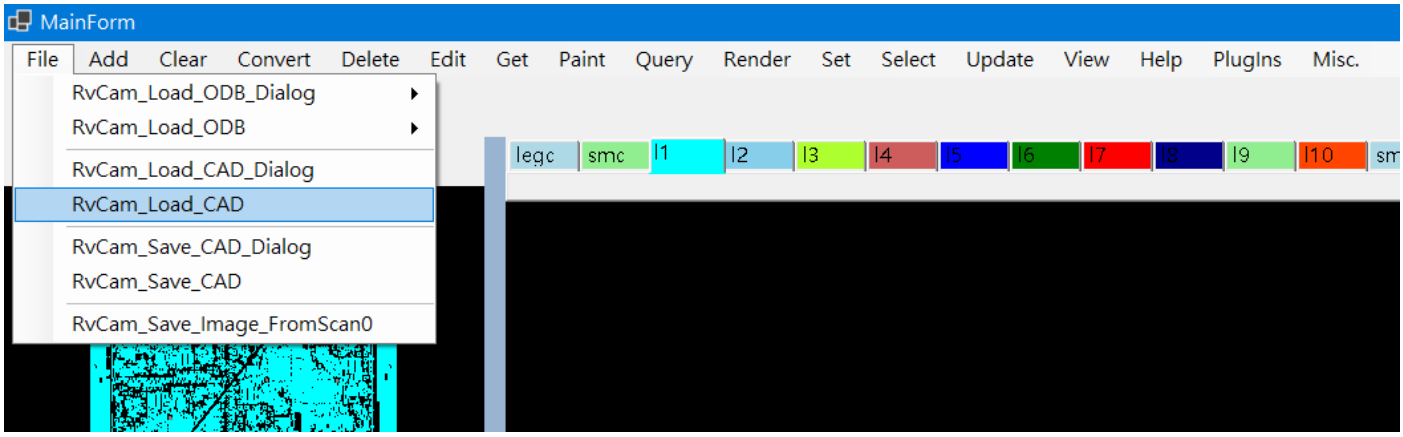


輸出補償校正後影像

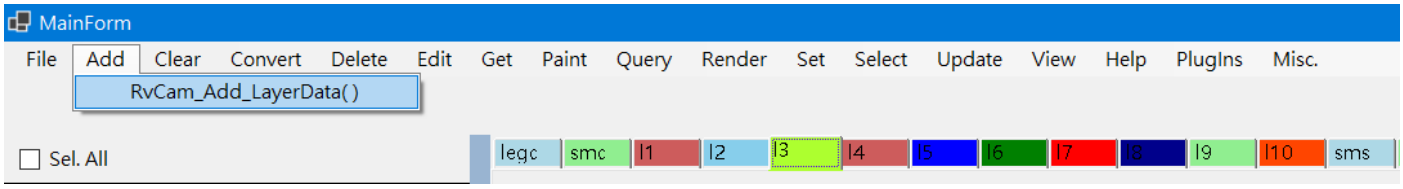
TestRvCamDLL_CS 程式介面與功能說明

以下將介紹如何在程式介面上操作，並呼叫 RvCamDLL 內的函式。上方選單每個項目及其子選單項目，對應到個別的 RvCamDLL 函式，方便學習函式使用方法。

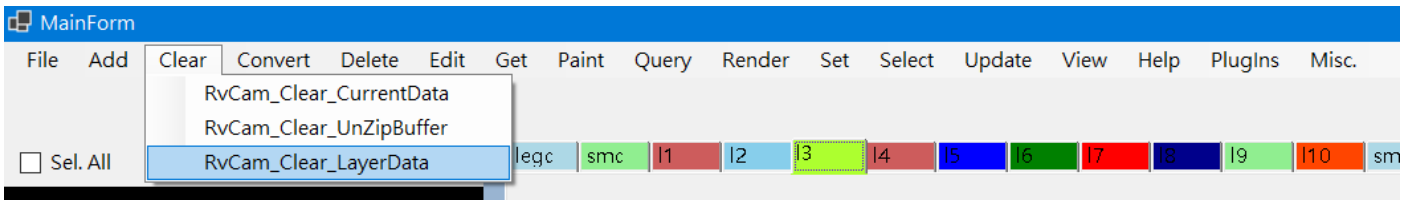
讀檔 與 存檔



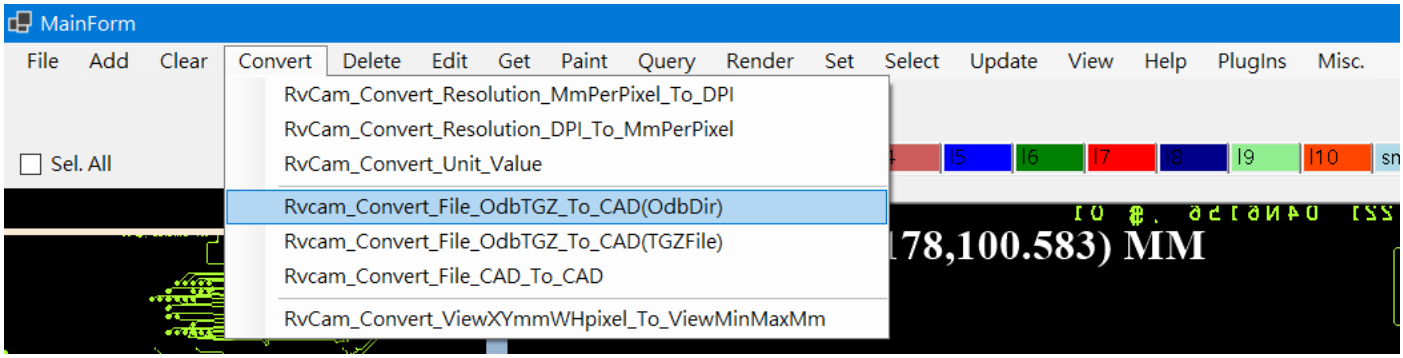
Add 新增



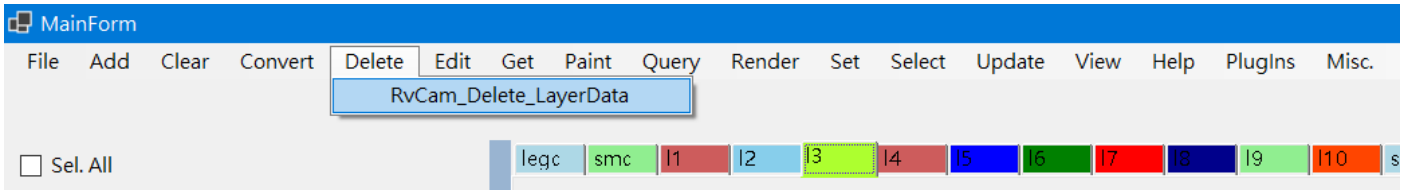
Clear 清除



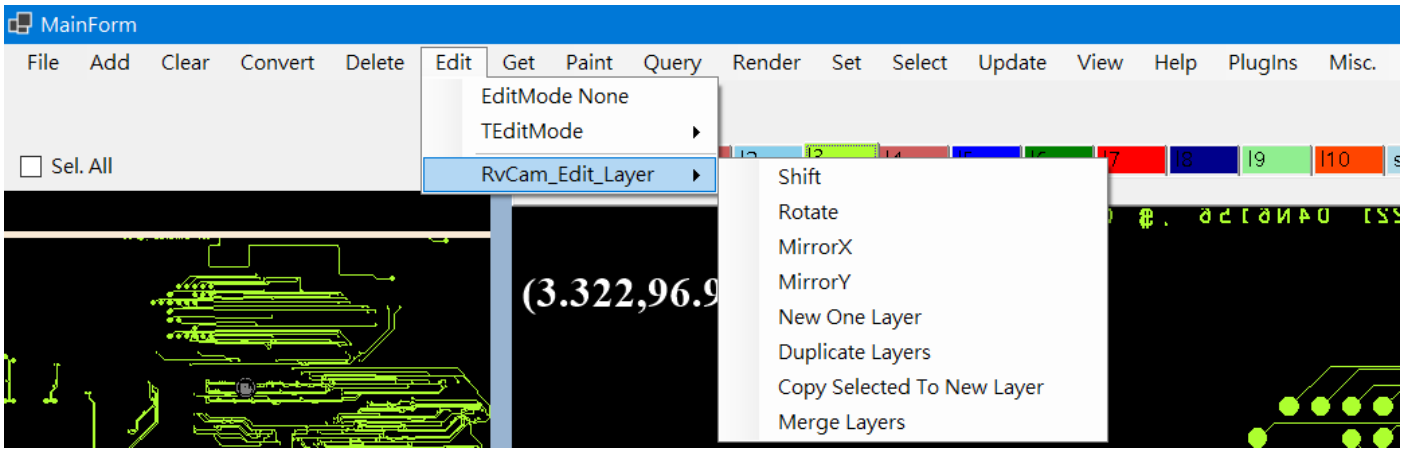
Convert 轉換



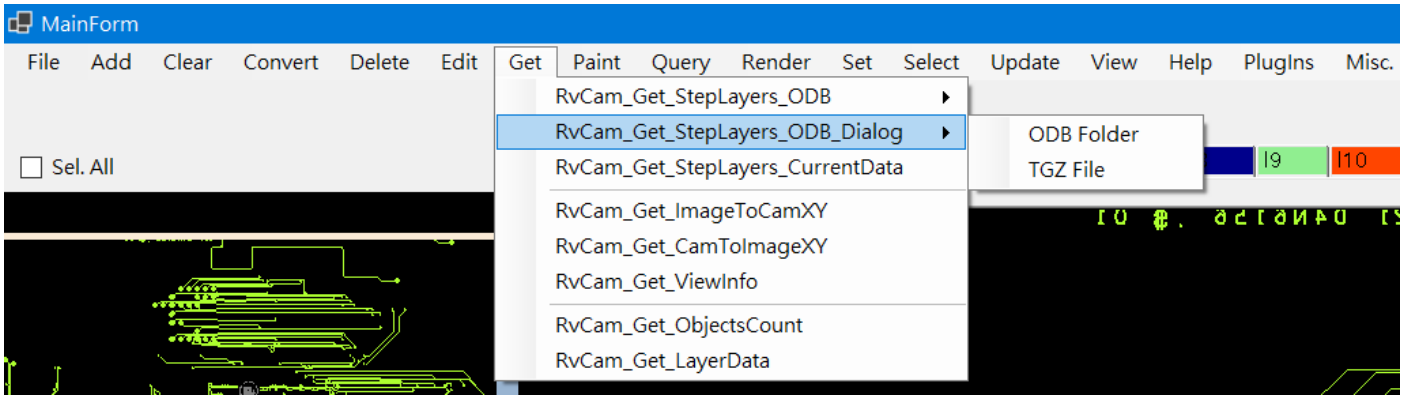
Delete 刪除



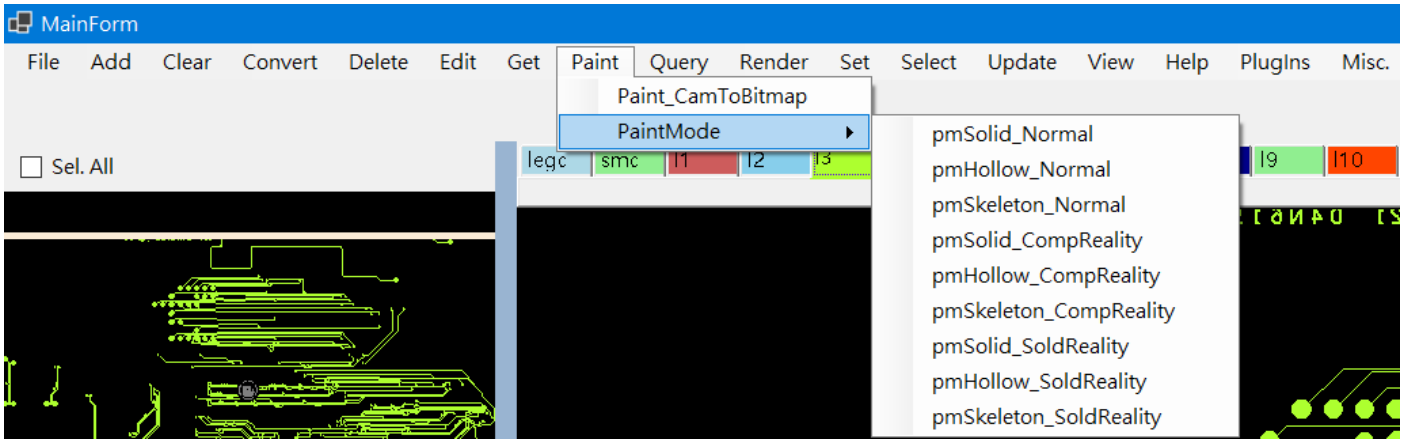
Edit 編輯



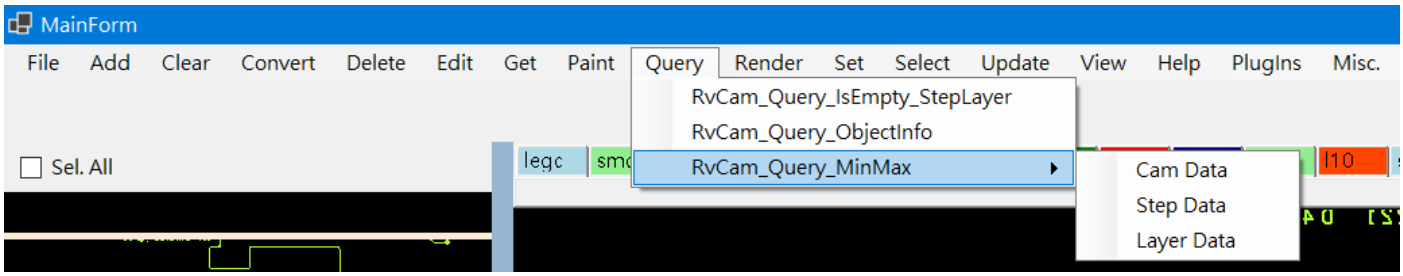
Get 取得



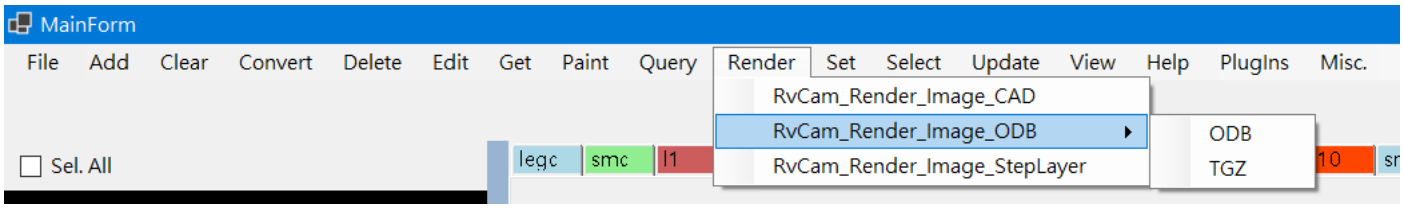
Paint 繪圖



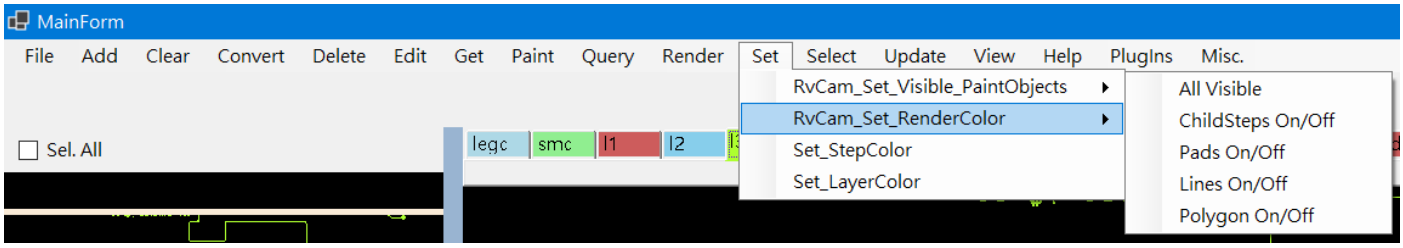
Query 詢問



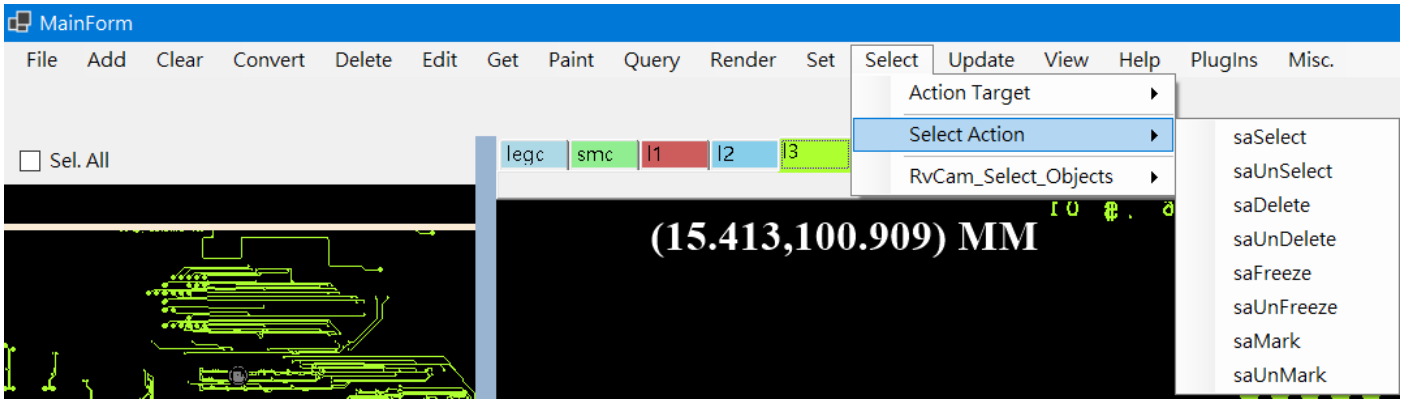
Render 出圖



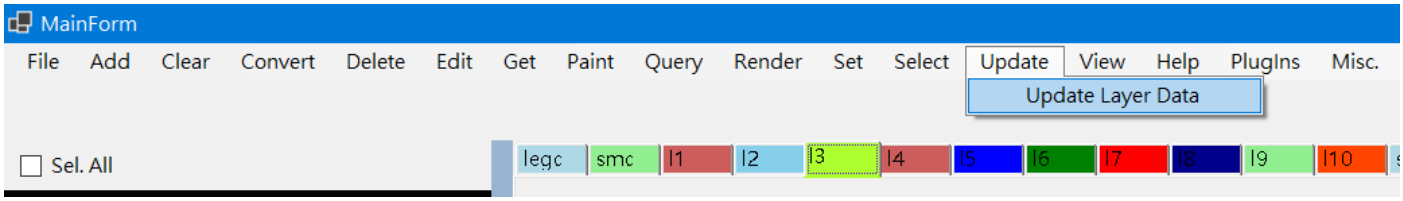
Set 設定



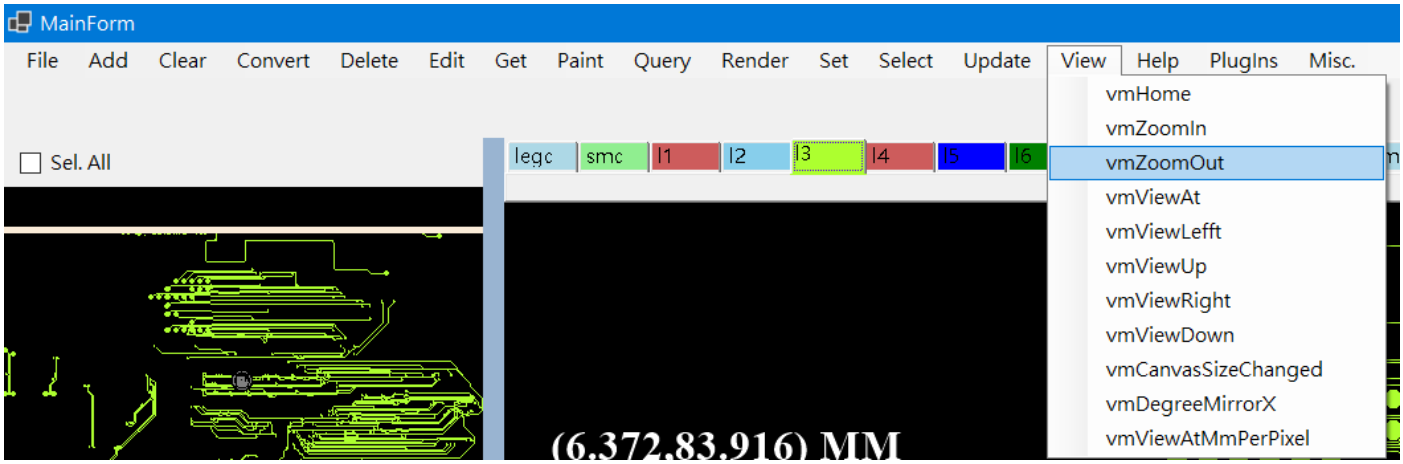
Select 選取



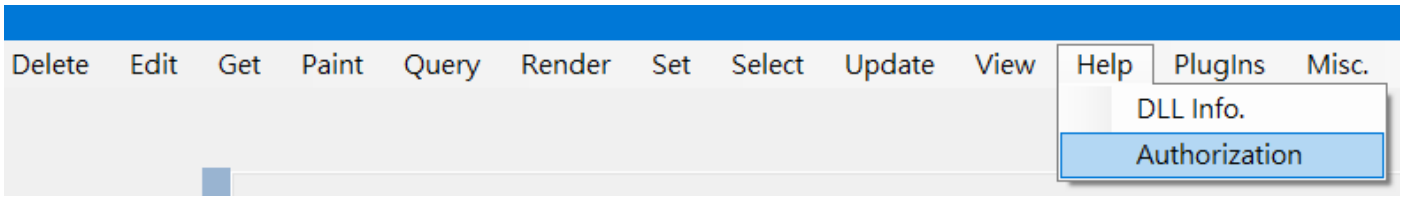
Update 更新



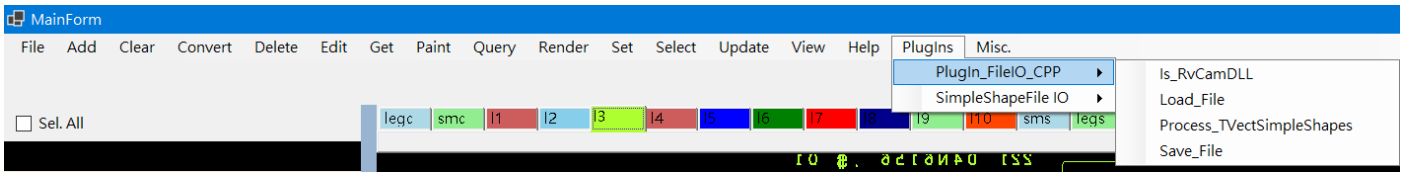
View 視角



Help 說明

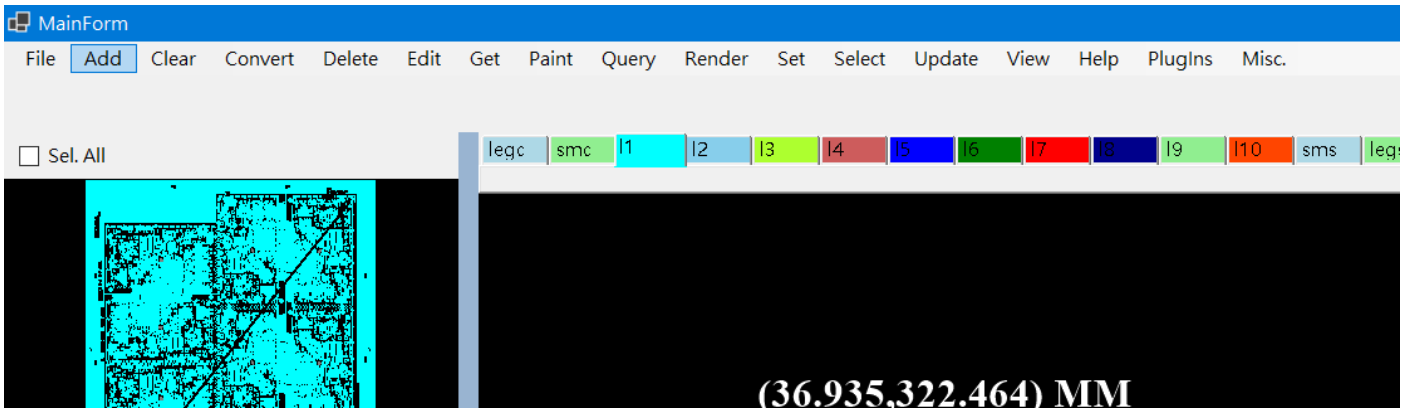


PlugIns 外掛

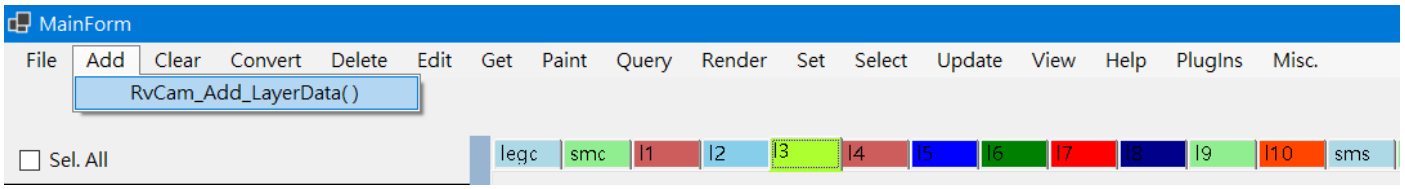


RvCamDLL 函式說明

以下的所有 DLL 函式，對應到 TestRvCamDLL_CS 主程式介面上的選單。可以在選單下拉選取功能後，執行測試，了解使用方式。



Add 函式



將圖形資料 新增到 新 Layer

程式開發者自行撰寫讀檔程序將檔案資料塞入 TVectSimpleShape[]後，

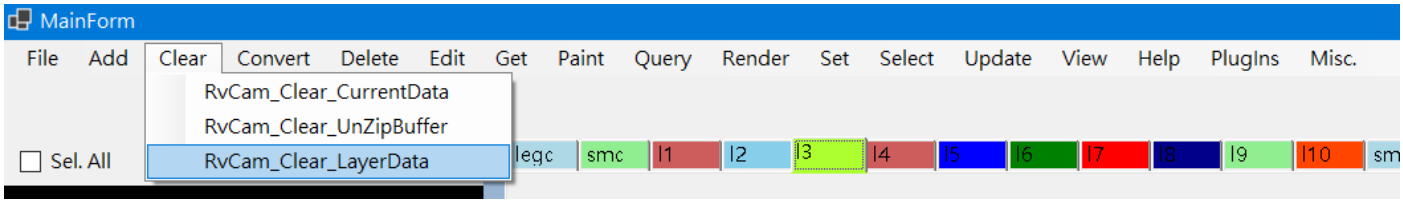
利用此函式傳入 RvCamDLL，即可使用所有的顯示、編輯、存讀檔、轉換、出圖…等等功能。

```
function RvCam_Add_LayerData(
    toStep:integer;                  //指定目的 StepID
    var getAddLayer:integer;         //傳回加到新層的 LayerID
    pShapeDataArray0:Pointer;        //要加入的資料 TVectSimpleShape[ ]   Array0 Pointer
    shapeDataLength:integer;         //要加入的資料數量  ArrayLength
    pAddToNewLayer_SetName:PWideChar=nil   //如果要加到新層，則指定層名稱
    ):TReturnCode; stdcall;
```

將影像 RawData 資料 新增到 新 Layer

```
function RvCam_Add_LayerData_Image(
    toStep:integer;                          //指定目的 StepID
    var getAddLayer:integer;                 //傳回加到新層的 LayerID
    pImageStart0: Pointer;                   //傳入圖形的 pScan0
    imageStride_BytesPerRow,                 //傳入圖形的大小，RawBytes
    imagePixelWidth,
    imagePixelHeight: integer;                    //傳入圖形 寬高 Pixel
    imgageResolution_MmPerPixel:TFloat;           //繪圖解析度 Mm/Pixel
    imageBitPerPixel: byte = 8;                   //指定像素格式
    imageDibUpWard:boolean = true;                //圖形 RawData 順序
    pImageRealMinMaxMm:PFRect = nil;              //傳入圖形資料實際範圍 MinMax (mm)
    pAddToNewLayer_SetName:PWideChar=nil          //指定新層名稱
    ):TReturnCode; stdcall;
```

Clear 函式



清除解壓縮路徑檔案

```
function RvCam_Clear_UnZipBuffer():TReturnCode; stdcall;
```

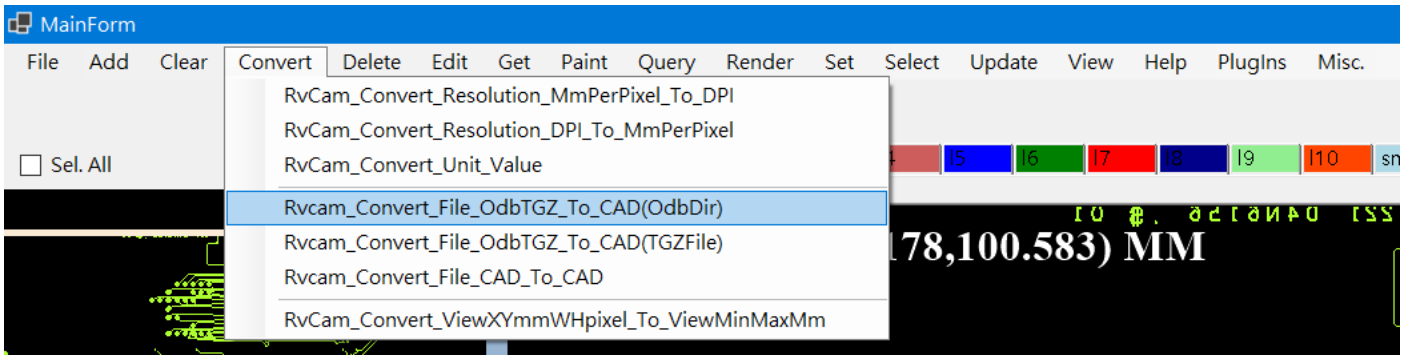
清除目前的 CAM 資料

```
function RvCam_Clear_CurrentData(): TReturnCode; stdcall;
```

清除某一層資料

```
function RvCam_Clear_LayerData(
    atStep, clearLayer:integer          //指定清除的 StepID/LayerID
    ):TReturnCode; stdcall;
```

Convert 函式



轉換檔案格式 ODB/TGZ => CAD 檔案

```
function Rvcam_Convert_File_OdbTGZ_To_CAD(
    loadOdbDir_TGZFile:PWideChar;          //指定讀入的 OdbDir 或 TgzFile
    loadStep:PWideChar;                    //指定讀入的 Step eg.'panel'
    loadLayers:PWideChar;                  //指定讀入的 Layers. eg. 'comp,l2,sold'
    setSaveCadFileDirectory:PWideChar;     //指定輸出路徑 eg. 'd:/outputCadFiles/'
    setSaveCadFileType:TVectFileType;      //指定輸出檔案類型
    var getSavedFileNames:PWideChar        //傳回輸出的所有短檔名 eg. 'comp.gbx,l2.gb
    ):TReturnCode;
```

轉換檔案格式 CAD => CAD 檔案

```
function Rvcam_Convert_File_CAD_To_CAD(
    loadCadFiles:PWideChar;                //指定讀入的 CAD files, 以 ',' 隔開，eg.'c:\a.dxf,
    setSaveCadFileDirectory:PWideChar;     //指定輸出路徑 eg. 'd:/outputCadFiles/'
    setSaveCadFileType:TVectFileType;      //指定輸出檔案類型
    var getSavedFileNames:PWideChar        //傳回輸出的所有短檔名 eg. 'comp.gbx,l2.gb
    ):TReturnCode;
```

轉換解析度單位 Mm/Pixel -> DPI

```
function RvCam_Convert_Resolution_MmPerPixel_To_DPI(
    mmPerPxl:double;   var DPI:double
    ):TReturnCode; stdcall;
```

轉換解析度單位 DPI -> Mm/Pixel

```
function RvCam_Convert_Resolution_DPI_To_MmPerPixel (
    DPI:double ;    var mmPerPxl:double
    ):TReturnCode; stdcall;
```

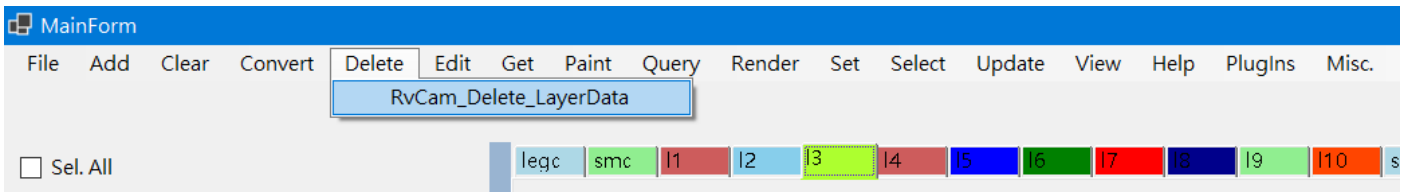
轉換數值單位 Inch, mil, mm, um, cm

```
function RvCam_Convert_Unit_Value(
    frUnit:TValueUnit; frValue:TFloat;          //來源單位，來源數值
    toUnit:TValueUnit; var toValue:TFloat       //目標單位，目標數值
    ):TReturnCode; stdcall;
```

轉換檢視中心(viewXYmm)+畫布範圍(pxlWH)=> 檢視範圍(minMaxMm)

```
function RvCam_Convert_ViewXYmmWHpixel_To_ViewMinMaxMm(
    viewCXmm,viewCYmm:TFloat;                   //檢視中心 XY MM
    canvasWidthPixel, canvasHeightPixel:integer;  //檢視寬高 Pixel
    viewResolution_MmPerPixel:TFloat;           //檢視解析度 MM/Pixel
    var viewMinXmm,viewMinYmm,                  //傳回檢視範圍左下角 MinXY  MM
    viewMaxXmm, viewMaxYmm:TFloat               //傳回檢視範圍右上角 MaxXY  MM
    ):TReturnCode; stdcall;
```

Delete 函式



刪除某一層資料

```
function RvCam_Delete_LayerData(
    delLayer:integer
    ):TReturnCode; stdcall;
```

Dialog 函式

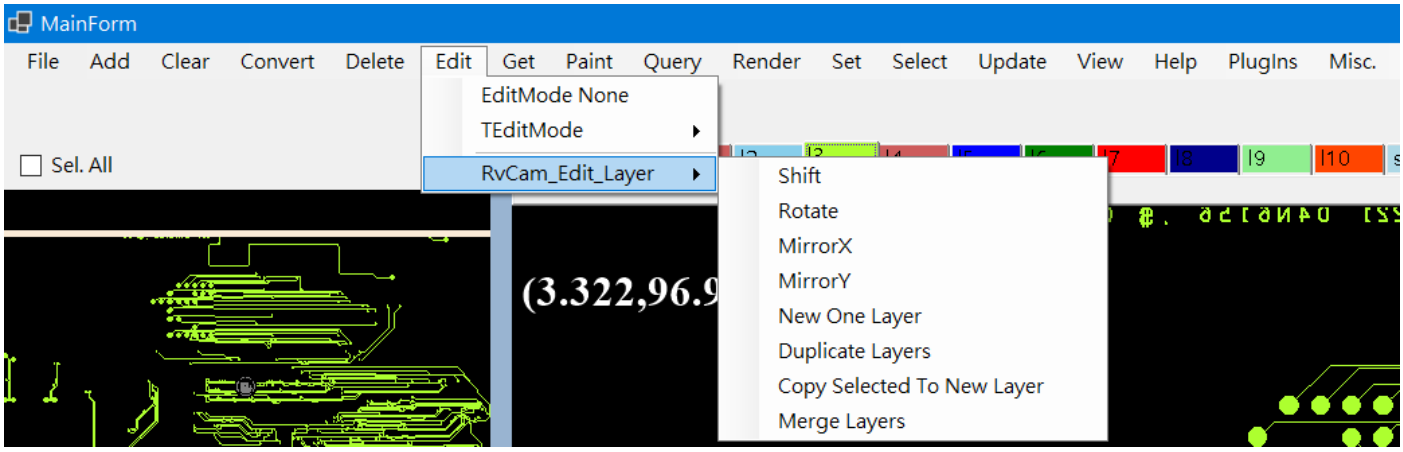
多筆欄位輸入對話框

```
function RvCam_Dialog_MultiInputBox(sTitle:PWideChar;
    sPrompts:PWideChar;          //項目名稱，可輸入多個項目，以','隔開
    var getValues:PWideChar      //回傳輸入的項目值，以','隔開
    ):TReturnCode; stdcall;
```

項目選取對話框

```
function RvCam_Dialog_ItemSelect(sTitle:PWideChar;
    sItems:PWideChar;                    //輸入多個選取項目，以','隔開
    var getItemsSelections:PWideChar;    //傳回選取結果 '0,1,1,0...'代表各項目是否選取
    var selectedIndex:integer;           //選取的項目 Index
    multiSelection:boolean = false       //是否多選
    ):TReturnCode;
```

Edit 函式



Step 資料編輯(複製、位移、旋轉、鏡射 XY...)

對 Step 作旋轉、位移、鏡射、複製...功能。不會受來源 Step 資料變化而影響。

```
function RvCam_Edit_Step(
    editStepName:PWideChar;                 //編輯的參考 Step 名稱
    editMode:TStepEditMode;                 // TStepEditMode
    shiftXmm_scaleX_rotateDegree:TFloat;    // ShiftXmm 或 RotateDegree
```

```
    shiftYmm_scaleY:TFloat;                 // ShiftYmm
    var getNewStepName:PWideChar            //指派(傳入字串)或新增(傳入 null)的 Step 名
    ):TReturnCode; stdcall;
```

Step 資料排版(位移、旋轉、鏡射 X)

對 Step 作排版複製，不複製來源 Step 資料，會受來源 Step 資料變化而影響。

```
function RvCam_Edit_Step_StepRepeat(
    editStepName:PWideChar;                 //編輯的參考 Step 名稱
    shiftXmm, shiftYmm:TFloat;              // ShiftXmm, ShiftYmm
    rotateDegree:TFloat;                    //旋轉角度
    mirrorX : boolean;                      //是否鏡射 X
    stpRptNumX, stpRptNumY:integer;         //X 和 Y 方向的排版片數
    stpRptDXmm, stpRptDYmm:TFloat;          //X 和 Y 方向的排版間隔 (mm)
    var setGetNewStepName:PWideChar         //指派(傳入字串)或新增(傳入 null)的 Step 名
    ):TReturnCode; stdcall;
```

層資料編輯(新增、複製、位移、旋轉、鏡射 XY...)

對 Layer 作旋轉、位移、鏡射、複製...等動作。

```
function RvCam_Edit_Layer(
    editStepName,                   //指定要編輯的 Step 名稱
    editLayerNames:PWideChar;       //指定要編輯的 Layer 名稱，可多筆 eg. "l1,l2,l3"
    editMode:TLayerEditMode;        //層編輯動作 (旋轉、位移、鏡射、複製、合併...)
    shiftXmm_scaleX_rotateDegree:TFloat;    //ShiftX, ScaleX or RotateDegree
    shiftYmm_scaleY:TFloat;         //ShiftY or ScaleY
    var getNewLayerNames:PWideChar  //傳回新增的層名
    ):TReturnCode; stdcall;
```

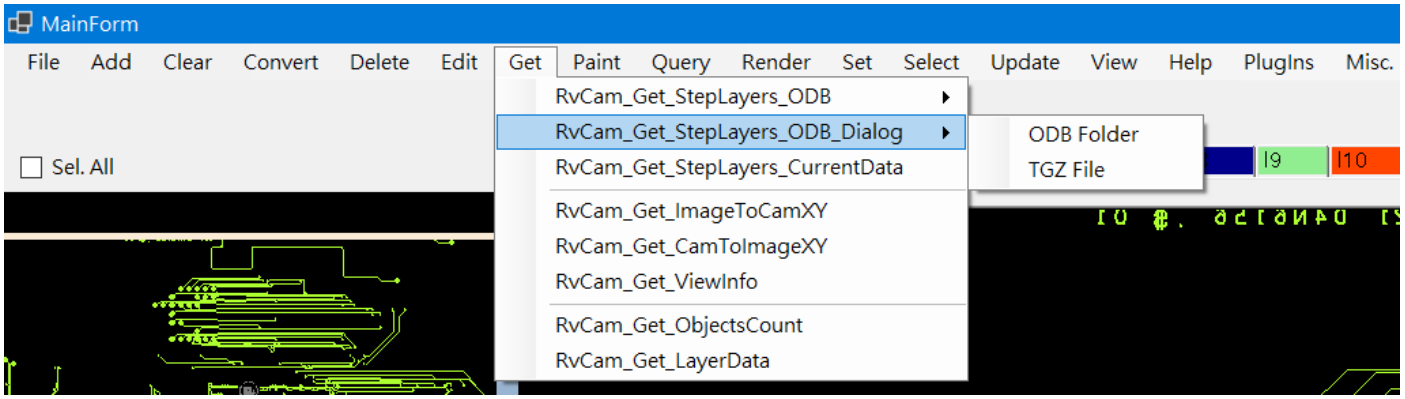
層資料旋轉位移對位

對 Layer 作旋轉、位移對位校正。

```
function RvCam_Edit_Layer_Align(
    editStepID:integer;             //指定要編輯的 Step 名稱
    fromLayerID:integer;            //對位來源層
    fromLayerMark1mm,               //來源層的第一個對位點(eg.左下角)
    fromLayerMark2mm:TFPoint;       //來源層的第二個對位點(eg.右上角)
    toLayerID:integer;              //對位參考層
    toLayerMark1mm,                 //參考層的第一個對位點(eg.左下角)
    toLayerMark2mm:TFPoint;         //參考層的第二個對位點(eg.右上角)
    var getRotateDegree:TFloat;     //傳回來源層旋轉角度
```

```
var getShiftXYmm:TFPoint;          //傳回來源層位移 XYmm
var getScale:TFloat                //傳回來源層縮放比例
):TReturnCode; stdcall;
```

Get 函式



取得 ODB++/TGZ 的 Steps, Layers 名稱

```
function RvCam_Get_StepsLayers_ODB(
    const sOdbDir_TGZFile: PWideChar;                    //Odb 目錄 或 TGZ 檔案
    var getSteps: PWideChar; var getLayers: PWideChar;   //傳回 steps, layer 名稱
    stepsListTp: TOdbStepListType = osAllSteps;          //設定 Steps 自動取得方式
    atTopStepName:PWideChar=nil                          //是否指定讀取的 Step 名稱
    ): TReturnCode; stdcall;
```

取得 ODB++/TGZ 的 Steps, Layers 名稱 (DLL 內讀檔介面)

```
function RvCam_Get_StepsLayers_ODB_Dialog(
    var getOdbDir_TGZFile: PWideChar;                    //傳回開檔視窗選取的 ODB++目錄或 TGZ 檔案
    var getSteps: PWideChar; var getLayers: PWideChar;   //傳回 steps, layer 名稱
    loadOdbTgzTp:TOdbTgzType = otOdbFolder;              //設定選取 ODB++目錄 或 TGZ 檔案
    stepsListTp: TOdbStepListType = osAllSteps           //設定 Steps 自動取得方式
  ): TReturnCode; stdcall;
```

取得目前 CAM 資料的 Steps,Layers 名稱

```
function RvCam_Get_StepsLayers_CurrentData(
    var getSteps: PWideChar; var getLayers: PWideChar    //傳回目前記憶體中所有 steps, layers 名稱
    ): TReturnCode; stdcall;
```

取得 ImageXY (Pixel)-> CamXY (mm)

```
function RvCam_Get_ImageToCamXY(
    canvasID:integer;                    //指定繪圖區的 ID
    imageX, imageY:integer;              //傳入 ImageXY pixel
    var camXmm, camYmm : TFloat;         //傳回對應的 camXY mm
    byStoredView:boolean=false           //是否將目前的 View 結構儲存備用
    ):TReturnCode; stdCall;
```

取得 CamXY (mm) -> ImageXY (Pixel)

```
function RvCam_Get_CamToImageXY(
    canvasID:integer;                    //指定繪圖區的 ID
    camXmm, camYmm:TFloat;               //傳入 camXY mm
    var imageX,imageY:integer            //傳回對應的 ImageXY pixel
    ):TReturnCode; stdcall;
```

取得繪圖的 View 的資訊

```
function RvCam_Get_ViewInfo(
    canvasID:integer;                                //指定繪圖區的 ID
    var viewCXMm,viewCYmm:TFloat;                    //傳回視點中心 CXY mm
    var viewWidthMm, viewHeightMm : TFloat;          //傳回視區長寬 mm
    var viewResolutionMmPerPixel:TFloat;             //傳回檢視的解析度 mm/pixel
    var viewDegree:TFloat;                           //傳回檢視角度
    var viewMirrorX:boolean;                         //傳回檢視是否鏡射 X
    byStoredView:boolean=false                       //是否從備存的 View 資料取得上述資訊
    ):TReturnCode; stdcall;
```

取得 Step/Layer 的物件數量

```
function RvCam_Get_ObjectsCount(
    qryStep,qryLayer:integer;                        //設定要詢問的 StepID/LayerID
    var padCount, lineCount, arcCount:integer        //傳回 PAD, Line, Arc 數量
    ):TReturnCode; stdcall;
```

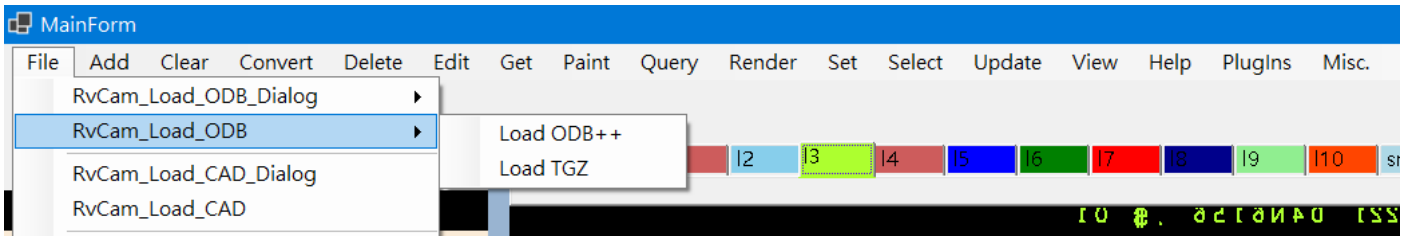
選取顏色

```
function RvCam_Get_Color(
    var getColor:integer                    //傳回選取的顏色 $AARRGGBB
    ):TReturnCode; stdCall;
```

取的 Layer 的向量圖形資料

```
function RvCam_Get_LayerData(
    getStep:integer; getLayer:integer;      //要取得的 Step/Layer
    var pShapeDataArray0:Pointer;           //傳回 Layer 所有物件資料陣列的第一筆指標 PArray0
    var shapeDataLength:integer             //傳回物件陣列總長度(物件數量)
    ):TReturnCode; stdcall;
```

Load 函式



讀取 CAD 檔案(*.GBX, *.DXF, *.NC...)

```
function RvCam_Load_CAD(
    const sCadFileNames: PWideChar;         //CAD 檔名，可讀入多檔以 ',' 分開， eg. 'c:\a.gbx, d:\b.enc'
    var getLoadToStepName:PWideChar;        //傳回讀入到記憶體中的 Step 名稱
    var getLoadToLayerName:PWideChar;       //傳回讀入到記憶體中的 Layer 名稱
    var getFileType:TVectFileType;          //傳回讀取的檔案類型
    const setLoadToStepNameOrNull: PWideChar = nil;   //是否指定讀入到哪個 Step
    blClearCurrentData: boolean = false     //是否先清除目前所有資烙
    ): TReturnCode; stdcall;
```

讀取 CAD 檔案 (DLL 內讀檔介面)

```
function RvCam_Load_CAD_Dialog(
    var getCadFileNames: PWideChar;     //傳回開檔視窗所選的檔案，以 ',' 分開， eg. 'c:\a.g
```

```
    var getLoadToStepName:PWideChar;          //傳回讀入到記憶體中的 Step 名稱
    var getLoadToLayerName:PWideChar;         //傳回讀入到記憶體中的 Layer 名稱
    var getFileType:TVectFileType;            //傳回讀取的檔案類型
    const setLoadToStepNameOrNull: PWideChar = nil;   //是否指定讀入到哪個 Step
    blClearCurrentData: boolean = false       //是否先清除目前所有資烙
  ): TReturnCode; stdcall;
```

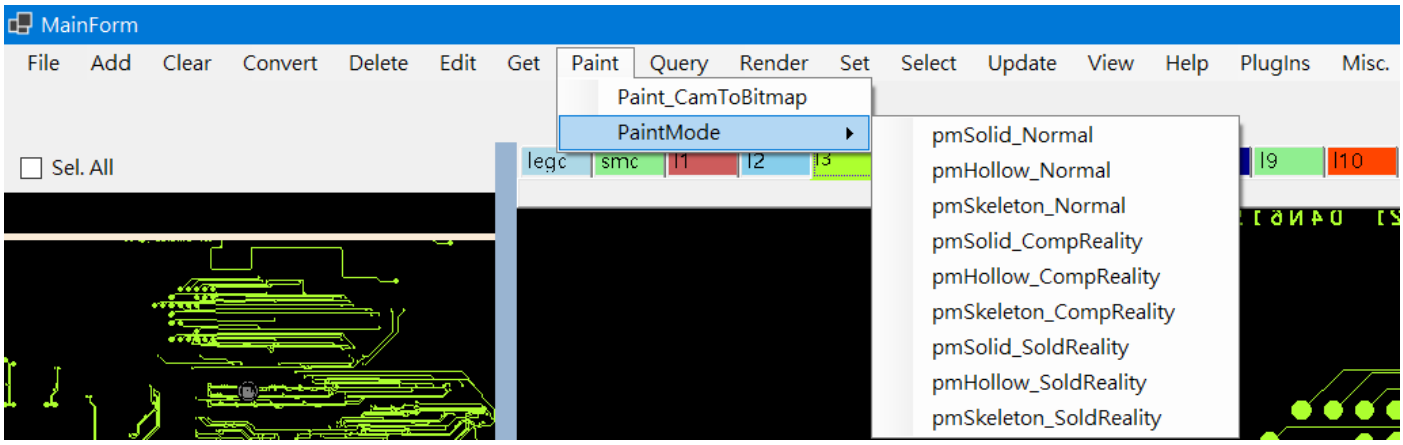
讀取 ODB++目錄 / TGZ 檔案

```
function RvCam_Load_ODB(
    const sOdbDir_TGZFile: PWideChar;     //設定讀取的 ODB++目錄或 TGZ 檔名
    var getSteps: PWideChar;              //傳回所有的 Steps 名稱，以"," 隔開
    var getLayers: PWideChar;             //傳回所有 Layers 名稱, 以"," 隔開
    showImportOdbDialog:boolean = false;      //是否顯示 ODB++讀檔視窗
    stepsListTp: TOdbStepListType = osAllSteps;       //設定 Step 自動讀取方式
    loadOnlySteps:PWideChar=nil;          //指定只讀入自訂 Steps, 以"," 隔開。Eg. 'pcb,array,panel',
    loadOnlyLayers:PWideChar=nil     //指定只讀入自訂 Steps, 以"," 隔開。Eg. 'comp, l2, l3'
    ): TReturnCode; stdcall;
```

讀取 ODB++目錄 / TGZ 檔案 (DLL 內讀檔介面)

```
function RvCam_Load_ODB_Dialog(
  var getOdbDir_TGZFile: PWideChar;       //傳回開檔視窗選取的 ODB++目錄或 TGZ 檔案
  var getSteps: PWideChar;                //傳回所有的 Steps 名稱，以"," 隔開
  var getLayers: PWideChar;               //傳回所有 Layers 名稱, 以"," 隔開
  loadOdbTgzTp:TOdbTgzType = otOdbFolder     //設定讀取 ODB++目錄或 TGZ 檔案
  ): TReturnCode; stdcall;
```

Paint 函式



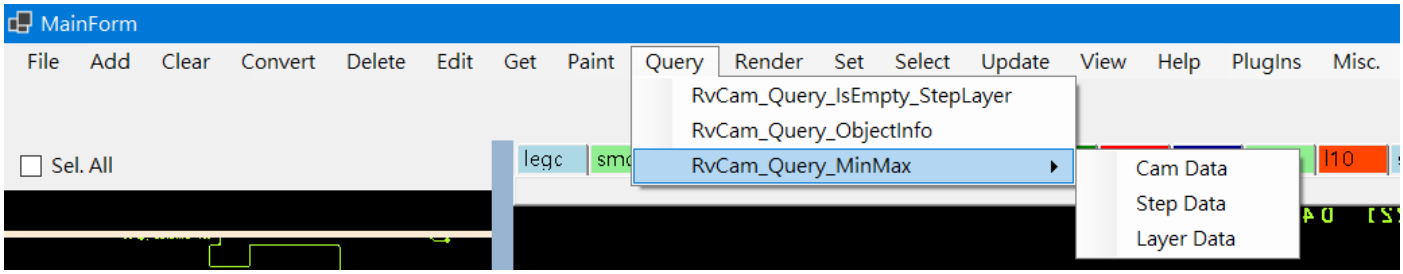
繪圖函式

```
function RvCam_Paint_Canvas(
    paintStep, paintLayer:integer;          //繪圖 StepID/LayerID
    paintColor_AARRGGBB : integer;          //繪圖顏色
    canvasID: integer;                      //繪圖區 ID
    var CnvScan0: Pointer;                  //繪圖區 RawData 的 Scan0
    CnvRowBytes, CnvWidth, CnvHeight,       //繪圖區 RawData 的 StideBytes, 畫布寬高 pixel
    CnvBitsPerPixel: integer;               //繪圖區 RawData 的像素格式
    cnvDIBUpWard: LongBool = true;          //繪圖區的 RawData 順序
    paintMode:TVectPaintMode =
        TVectPaintMode.pmSolid_Normal       //繪圖模式，詳參程式碼定義
    ): TReturnCode; stdcall;
```

畫出尺規

```
function RvCam_Paint_Canvas_Ruler(
    canvasID: integer;                      //繪圖區 ID
    paintColor_AARRGGBB : integer;          //尺規顏色
    setDisplayUnit:TValueUnit;              //尺規顯示單位 (cm,mm,um,nm,inch,mil)
    var CnvScan0: Pointer;                  //繪圖區 RawData Scan0
    CnvRowBytes, CnvWidth, CnvHeight,       //繪圖區 RawData StrideBytes, 寬、長 Pixel
    CnvBitsPerPixel: integer;               //繪圖區 RawData 像素格式
    cnvDIBUpWard: LongBool = true;          //繪圖區 RawData 順序
    rulerPixelWidth:integer=50              //尺規繪圖高度 Pixel
    ): TReturnCode; stdcall;
```

Query 函式



檢查 Step/Layer 是否有資料

```
function RvCam_Query_IsEmpty_StepLayer(
    qryStep, qryLayer:integer;          //詢問的 StepID / LayerID
    var getIsEmpty:Boolean              //傳回是否為空資料層
  ):TReturnCode; stdcall;
```

詢問物件資訊

```
function RvCam_Query_ObjectInfo(
    qryStep,qryLayer:integer;                   //詢問的 StepID / LayerID
    qryXmm, qryYmm, qryTolmm:TFloat;            //詢問的位置和搜尋範圍
    var getObjCXmm,getObjCYmm,                  //傳回 Object 的中心和長寬(mm)
    getObjWmm,getObjHmm,
    var getObjectInfo:PWideChar                 //傳回找到的物件資訊
    ):TReturnCode; stdcall;
```

詢問大小範圍

```
function RvCam_Query_MinMax(
    qryStep,qryLayer:integer;                   //詢問的 StepID / LayerID
    qryTarget:TCamTarget;                       //設定詢問的是 Cam Data, Step 還是 Layer
    var getMinXmm, getMinYmm,                   //傳回取得範圍左下角 MinXY mm
    getMaxXmm, getMaxYmm:TFloat                 //傳回取得範圍右上角 MaxXY mm
    ):TReturnCode; stdcall;
```

詢問影像上 Blob 中心

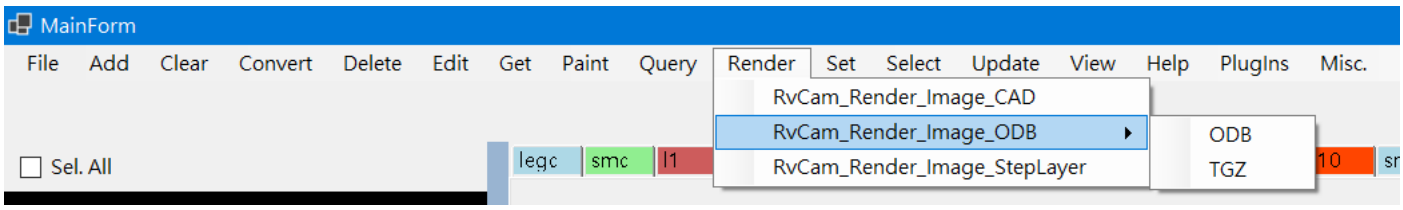
```
function RvCam_Query_BlobCXY_Scan0(
    inImgX, inImgY:TFloat;                      //搜尋的影像座標 pixel
```

```
var getBlobCX, getBlobCY:TFloat;        //傳回找到的 BlobCXY pixel
var getBlobW, getBlobH:TFloat;          //傳回找到的 Blob Width Height pixel
const imgScan0: Pointer;                //影像 RawData Scan0
imgRowBytes, imgWidth, imgHeight,       //StrideBytes, ImageWidth, Image Height
imgBitsPerPixel: integer;               //影像像素格式 Bits/Pixel
imgDIBUpWard: LongBool = true           //影像順序
):TReturnCode; stdcall;
```

詢問 Layer 影像的 Blob 中心

```
function RvCam_Query_BlobCXY_Layer(
    atStep,atLayer:integer;                 //StepID, LayerID
    inCXmm,inCYmm : TFloat;                 //傳入起始搜尋座標 (mm)
    var getBlobCXmm,getBlobCYmm:TFloat;     //傳回找到的 BlobXY (mm)
    var getBlobWmm,getBlobHmm:TFloat;       //傳回找到的 BlobWH (mm)
    var getObjectInfo:PWideChar             //傳回找到的物件資訊
    ):TReturnCode; stdcall;
```

Render 函式



背景讀入 CAD 檔案、算圖、輸出圖檔

```
function RvCam_Render_Image_CAD(
    const cadFn:PWideChar;                  //出圖的 CAD 檔名
    renderResolution_MmPerPixel:TFloat;     //出圖解析度 Mm/Pixel
    var pGetImageStart0: Pointer;           //傳回記憶體內圖形 RawData 的 pScan0
    var getImageSizeTotalMB,                //傳回圖形資料大小 MegaByte
        getStride_BytesPerRow,                      //傳回圖形每列 Byte 數，RawByte
    getImagePixelWidth, getImagePixelHeight: integer;   //傳回圖形寬高 Pixel
    atBitPerPixel: byte = 8;                    //指定輸出像素格式
    pAssignRenderMinMaxMm:Pointer = nil;        //指定輸出範圍 MinMax Pointer
    const pDoSaveToBmpFile: PWideChar = nil;    //是否指定輸出檔名或路徑
    paintMode:TVectPaintMode =
```

```
        TVectPaintMode.pmSolid_Normal                    //指定繪圖模式
    ):TReturnCode; stdcall;
```

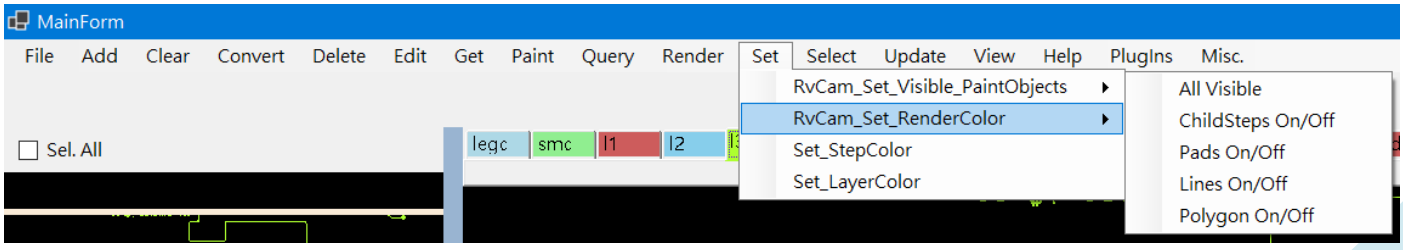
背景讀入 ODB/TGZ 資料、算圖、輸出圖檔

```
function RvCam_Render_Image_ODB(
    const odbTgzFn:PWideChar;                            //出圖的 OdbDir 或 TgzFile
    renderStepName, renderLayerNames:PWideChar;          //指定出圖 StepName, LayerName
    renderResolution_MmPerPixel:TFloat;                  //出圖解析度 Mm/Pixel
    var pGetImageStart0: Pointer;                        //傳回記憶體中圖形 RawData 的 pScan0
    var getImageSizeTotalMB, getStride_BytesPerRow,      //傳回圖形的大小 MegaByte、StrideByte
    getImagePixelWidth, getImagePixelHeight: integer;    //傳回圖形 寬高 Pixel
    atBitPerPixel: byte = 8;                             //指定輸出像素格式
    pAssignRenderMinMaxMm:Pointer = nil;                 //指定輸出範圍 MinMax Pointer
    const pDoSaveToDirectory: PWideChar = nil;           //是否指定輸出檔名或路徑
    paintMode:TVectPaintMode =
        TVectPaintMode.pmSolid_Normal                    //指定繪圖模式
    ):TReturnCode; stdcall;
```

從記憶體 CAM 資料算圖、輸出圖檔

```
function RvCam_Render_Image_StepLayer(
    renderStepID, renderLayerID:integer;                 //指定出圖 StepID, LayerID
    renderResolution_MmPerPixel:TFloat;                  //出圖解析度 Mm/Pixel
    var pGetImageStart0: Pointer;                        //傳回圖形的 pScan0
    var getImageSizeTotalMB, getStride_BytesPerRow,      //傳回圖形的大小 MB，StrideByte
    getImagePixelWidth, getImagePixelHeight: integer;    //傳回圖形 寬高 Pixel
    atBitPerPixel: byte = 8;                             //指定輸出像素格式
    pAssignRenderMinMaxMm:Pointer = nil;                 //指定輸出範圍 MinMax Pointer
    const pDoSaveToBmpFile: PWideChar = nil;             //是否指定輸出檔名或路徑
    paintMode:TVectPaintMode =
        TVectPaintMode.pmSolid_Normal                    //指定繪圖模式
    ):TReturnCode; stdcall;
```

Set 函式



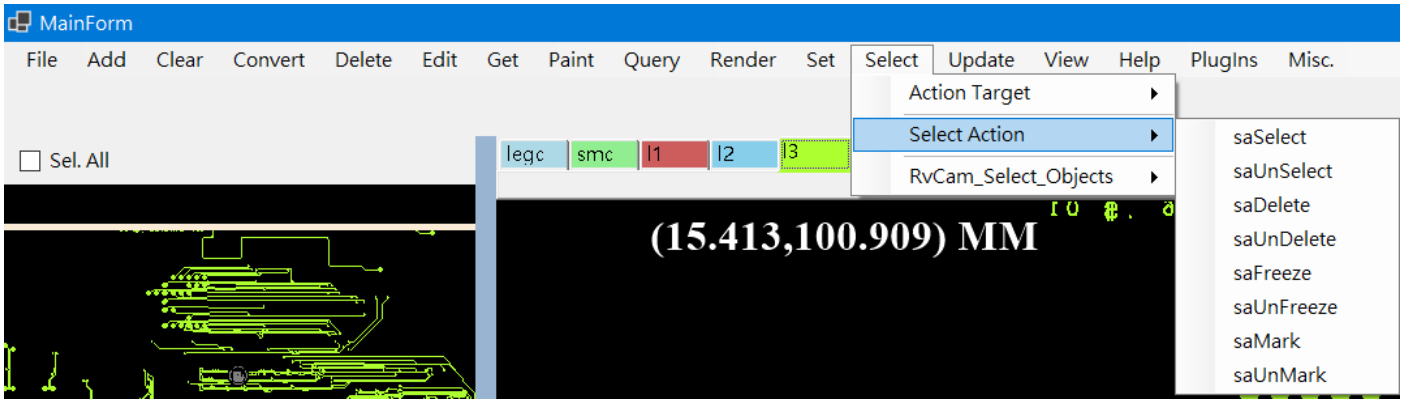
設定顯示/隱藏 物件

```
function RvCam_Set_Visible_PaintObjects(
    childSteps:boolean=true;          //是否顯示排版內的 Child Steps
    pads:boolean=true;                //是否顯示 PAD
    lines:boolean=true;               //是否顯示 Line
    polygons:boolean=true             //是否顯示 Polygon
    ): TReturnCode;
```

設定顏色顯示模式

```
function RvCam_Set_RenderColor(
    renderClr:TVPRenderColor          //物件顯示模式
    ):TReturnCode; stdcall;
```

Select 函式



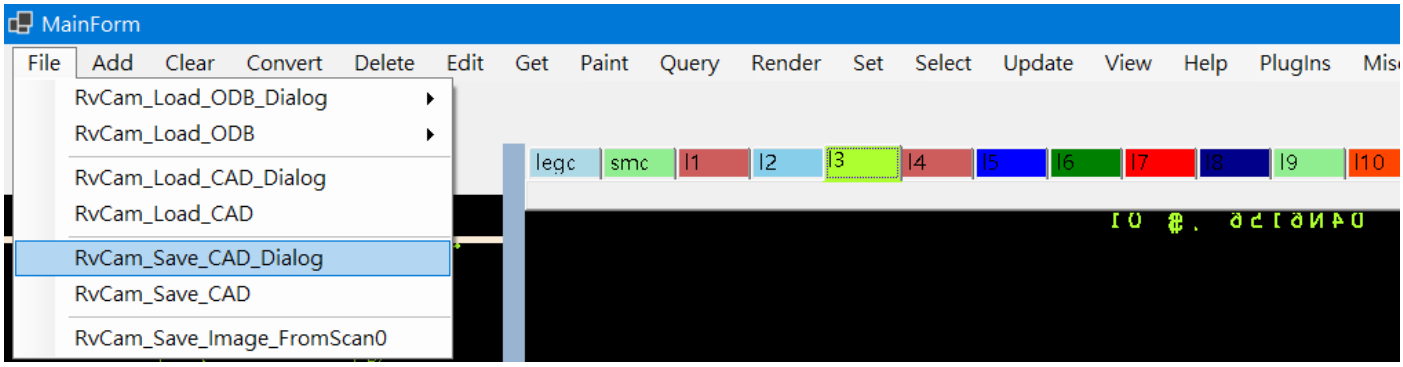
框選範圍選取、刪除、標記、冷凍物件

```
function RvCam_Select_Objects(
    selectStep, selectLayer:integer;          //選取的 StepID, LayerID
    selectCXmm, selectCYmm,                   //atCXYmm(選取中心)
    selectWidthmm,selectHeightmm:TFloat;      //選取範圍寬高
    var getSelectedObjectsCount:integer;      //傳回選取的物件數量
    selectAction:TSelectAction = saSelect;    //選取後執行的動作
    selectTarget :TActionTarget =
        TActionTarget.smTVectObject           //選取目標 object / symbol
    ):TReturnCode; stdcall;
```

指定 SymbolName 選取

```
function RvCam_Select_Objects_BySymbolName(
    selectStep, selectLayer:integer;          //選取的 Step, Layer
    selectSymbolName:PWideChar;               //選取的 SymbolName
    var getSelectedObjectsCount:integer;      //傳回選取的物件數量
    selectAction:TSelectAction = saSelect     //選取後執行的動作
    ):TReturnCode; stdcall;
```

Save 函式



儲存 CAD 檔案 (*.GBX, *.DXF, *.NC...)

```
function RvCam_Save_CAD(
    var sSaveCadFileName: PWideChar;          //儲存的 CAD 檔名。NC/DXF/Gerber....
    saveStepName, saveLayerName:PWideChar;    //指定儲存的 Step/Layer
    saveCadFileType:TVectFileType =
        TVectFileType.vtGerber274X            //指定儲存的檔案格式
    ): TReturnCode; stdcall;
```

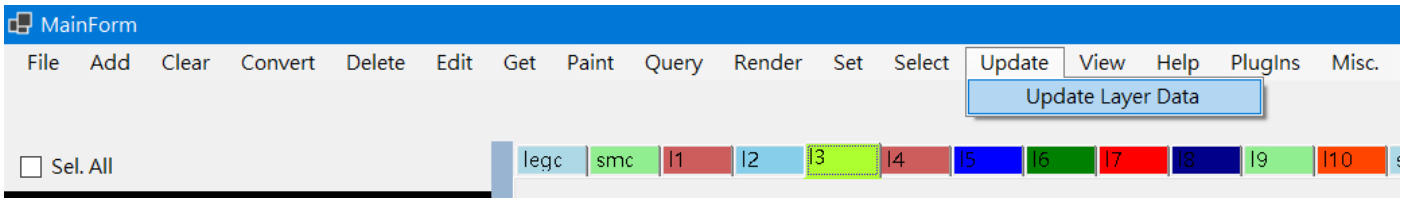
儲存 CAD 檔案 (*.GBX, *.DXF, *.NC...) (DLL 內存檔介面)

```
function RvCam_Save_CAD_Dialog(
    var getSavedCadFileName: PWideChar;       //從存檔視窗取得檔名
    saveStepName, saveLayerName:PWideChar;    //指定儲存的 Step/Layer
    saveFileType:TVectFileType =
        TVectFileType.vtGerber274X            //指定儲存的檔案格式
    ): TReturnCode; stdcall;
```

從記憶體內圖形資料存圖檔

```
function RvCam_Save_Image_FromScan0(
    saveImageFileName:PWideChar;                    //設定儲存圖檔檔名
    const pSaveImageScan0:Pointer;                  //儲存的資料來源 RawData Scan0
    imageSizeTotalMB:integer;                       //儲存的圖形大小 MB
    stride_BytesPerRow,                             //圖形資料的每列 Byte 數，StrideByte
    imagePixelWidth, imagePixelHeight:integer;      //儲存的圖形長寬 Pixel
    atBitPerPixel:byte                              //儲存圖檔的像素格式
    ):TReturnCode; stdcall;
```

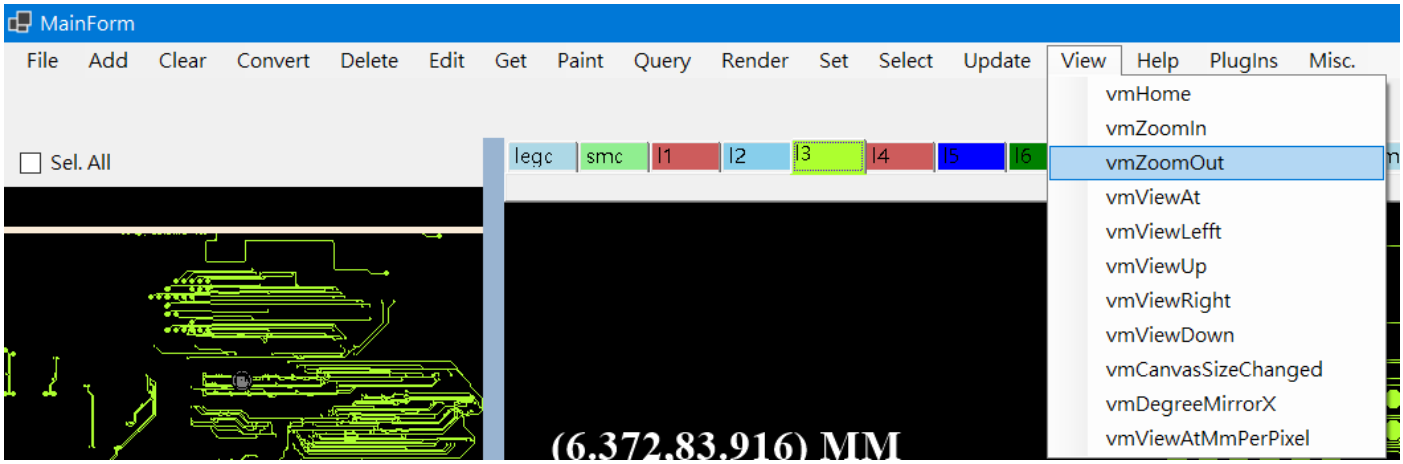
Update 函式



將圖形資料更新到 Layer

```
function RvCam_Update_LayerData(
    atStep:integer; updateLayer:integer;    //設定更新的目標 Step / Layer ID
    pShapeDataArray0:Pointer;               //來源的圖形資料陣列第一筆位址。PArray0
    shapeDataLength:integer                 //來源圖形資料的陣列長度(物件數量)。
    ):TReturnCode; stdcall;
```

View 函式



儲存目前的繪圖檢視 View 資料

```
function RvCam_View_Store(
    canvasID: integer              //儲存目前檢視資烙  View Data
  ): TReturnCode; stdcall;
```

更新目前的 View 資料

```
function RvCam_View_Update(
    canvasID: integer;                    //繪圖區 ID
    cnvW_viewX_viewDeg_atMmPerPxl,        //設定  畫布寬度 Pixlel  或  檢視中心 Xmm 或檢視角度
    cnvH_viewY_viewMrX:TFloat;            //設定  畫布高度 pixel  或  檢視中心 Ymm 或  鏡射 X
    viewStep,viewLayer:integer;           //檢視的  Step /LayerID
    atViewMode: TViewMode =
        vmHome                            //View  更新模式
    ): TReturnCode; stdcall;
```

以檢視範圍更新目前的 View

```
function RvCam_View_Update_ViewMinMax(
    canvasID: integer;                    //繪圖區 ID
    viewMinXmm,ViewMinYmm,                //設定檢視範圍左下角  MinXY mm
    ViewMaxXmm,viewMaxYmm:TFloat;         //設定檢視範圍右上角 MaxXY mm
    viewStep,viewLayer:integer            //檢視的  Step /LayerID
    ): TReturnCode; stdcall;
```

其他函式

檢查軟體是否有授權

```
function RvCam_IsAuthorized(
    ): TReturnCode; stdcall;
```

取得 DLL 訊息

```
function RvCam_GetDLLInfo(
    var sDLLInfo: PAnsiChar        //傳回 DLL 資訊
    ): TReturnCode; stdcall;
```

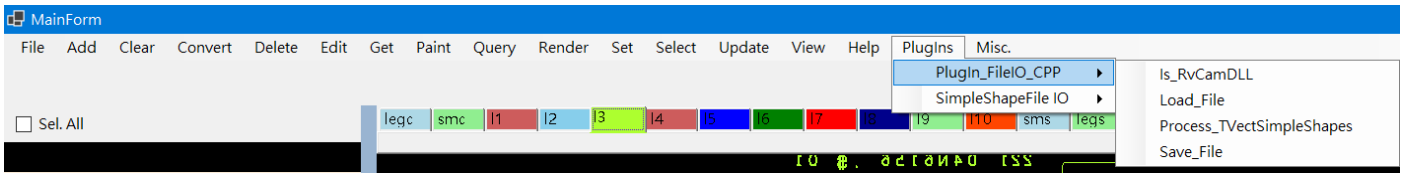
設定 CallBack 函式 (執行進度)

```
procedure RvCam_AssignRunningProgressCallBackFunc(
    const pCallBackFunc: Pointer        //設定執行進度通知的 CallBack 函式
    ); stdcall;
```

設定 CallBack 函式 (Log 訊息)

```
procedure RvCam_AssignLogInfoFunc(
    const pLogInfoFunc: Pointer        //設定 Log Info 通知的 CallBack 函式
    ); stdcall;
```

製作外掛函式 DLL



檔案讀取與儲存外掛

程式開發者，可參考以下範例，製作自訂的 CAD 檔案讀取、編輯和儲存功能，即可完全使用 RvCam 程式上的介面顯示、圖形檢視操作、檔案轉換、算圖、編輯...等等所有功能。

將編譯好的 DLL 檔案放在程式根目錄的 [PlugIns] 資料夾內，程式開啟後即會讀取外掛 DLL，並在程式界面的上方選單的 PlugIns 選單下產生相對應的 DLL 名稱和所屬功能項目。

此外掛包含四個函式:

```
//判斷是否為 RvCamDLL 可接受的外掛------------------------------
extern "C" DLLEXPORT_API TReturnCode    CALLConversion Is_RvCamDLL(
    wchar_t** sDLLDescription );

//讀檔函式，在此函式內撰寫自己的讀檔功能--------------
extern "C" DLLEXPORT_API TReturnCode    CALLConversion Load_File(
    const wchar_t* setLoadFileName,
    void** pShapeDataArray0,
    int* shapeDataLength);

//在此函式內撰寫資料處理編輯功能-----------------------------------
extern "C" DLLEXPORT_API TReturnCode    CALLConversion Process_TVectSimpleShapes(
    void** pShapeDataArray0,
    int* shapeDataLength);

//在此函式內撰寫存檔功能-------------------------------------
extern "C" DLLEXPORT_API TReturnCode    CALLConversion Save_File(
    wchar_t* setgetSaveFileName,
    void* pShapeDataArray0,
    int shapeDataLength);
```

外掛程式碼 VC++範例

下載程式專案。

解壓縮後，開啟目錄下專案 [RvCamPlugIn Examples]\[CPP]\[PlugIn_FileIO]

外掛程式碼 Delphi 範例

解壓縮後，開啟目錄下專案 [RvCamPlugIn Examples]\[Delphi]\

CSharp 程式碼說明

主函式

RvCamDLL.cs

DLL 函式宣告及程式介面使用的函式。

資料型態

M2dTypeDefine.cs

基本資料型態宣告

VectTypeDefine.cs

圖形資料型態宣告

外掛函式庫

RvCamDLL_PlugIn_FileIO.cs

外掛 DLL 自動讀取及引用處理函式

資料型態定義

M2dTypeDefine.CS

```
public enum TValueUnit : int
{
    uInch = 0, uMil = 1, uCM = 2, uMM = 3, uUM = 4
}
```

VectTypedefine.CS

```
public enum TViewMode : int
{
    vmHome = 0, vmZoomIn, vmZoomOut, vmViewAtXY, vmViewLeft, vmViewUp,
    vmViewRight, vmViewDown, vmCanvasSizeChanged, vmDegreeMirrorX,
    vmViewAtMmPerPixel
}

public enum TVectPaintMode : int
{
    pmSolid_Normal = 0, pmHollow_Normal, pmSkeleton_Normal,
    pmSolid_CompReality, pmHollow_CompReality, pmSkeleton_CompReality,
    pmSolid_SoldRelaity, pmHollow_SoldReality, pmSkeleton_SoldReality
}

public enum TVPRenderColor : int
{
    vcByTVectSymbol = 0, vcByTVectObject, vcByTVectLayer, vcByTVectStep,
    vcByTFillRec, vcByTCode, vcByGCode
}

public enum TVectFileType : int
{
    vtUnknown = 0, vtNewCreated, vtRaster, vtMVI, vtGerber274X,
    vtOdb, vtTGZOdb, vtExcellon, vtIPC356, vtSiebMeyer, vtSVG, vtDXF, vtDPF,
    vtAI, vtPostScript, vtEPS, vtRAR, vtZIP, vtEastekCar, vtTxt,
    vtTestFile, vtErrorLog, vtRasVectorCam, vtLdiBin, vtGDS, vtSimpleShapeFile,
    vtGIH
```

```
}

public enum TOdbStepListType : int
{
    osAllSteps = 0, osInheritedStepsOnly, osTopStepsOnly,
    osBottomStepsOnly, osIndependentStepsOnly
}

public enum TOdbTgzType : int
{
    otOdbFolder = 0, otTgzFile = 1
}

public enum TCamTarget : int
{
    ctCam = 0, ctStep, ctLayer
}

public enum TActionTarget : int
{
    smTVectSymbol = 0, smTVectObject, smContinueLines,
    smTFPoints, smTestPoint, sm4wire
}

public enum TSelectAction : int
{
    saSelect = 0, saUnSelect, saDelete, saUnDelete,
    saFreeze, saUnFreeze, saMark, saUnMark, saReverseSelect
}

public enum TLayerEditMode : int
{
    leNone =0, leShift, leRotate, leMirrorX, leMirrorY,
    leNewOneLayer, leDuplicateLayers,
    leCopySelectedToNewLayer,
    leMergeLayers
}

public enum TStepEditMode : int
{
    seNone = 0,
```

```
        seShift, seRotate,
        seMirrorX, seMirrorY,
        seScale,
        seDuplicate
}

public enum TVectSimpleShapeType : int
{
        vstNone = 0, vstArc, vstCircle, vstLine,
        vstRect, vstPolygon, vstPolyLine,
        vstSegments, vstIslandHoleShape
}

public enum TIslandHole : int
{
        ihIsland=0,
        ihHole
}

public enum TIslandHoleShapeMode : int
{
        ihsTransparentIslandHole=0,
        ihsOpaqueIslandHole,
        ihsShapesGroup
}

[StructLayout(LayoutKind.Explicit, Pack = 1, CharSet = CharSet.Ansi)]
  public unsafe struct TVectSimpleShape
  {
        [FieldOffset(0)]    public TFRect vstMinMax;
        [FieldOffset(32)] public TIslandHole vstIslandHole;
        [FieldOffset(36)] public TVectSimpleShapeType vstType;
        [FieldOffset(40)] public IntPtr vstPObj;
        [FieldOffset(48)] public TFloat vstRad;
        [FieldOffset(56)] public TVectSymbolType vstRefSymbolTp;
        [FieldOffset(60)] public IntPtr vstPFill;
        [FieldOffset(68)] public int vstDummyInt;
        [FieldOffset(72)] public int vstDummyInt1;
        [FieldOffset(76)] public int vstDummyInt2;
        [FieldOffset(80)] public TFloat vstDummyFloat;
        [FieldOffset(88)] public IntPtr vstDummyPointer;
```

```
        [FieldOffset(96)] public IntPtr vstDummyPointer1;

        [FieldOffset(104)] public TFPoint arStart;
        [FieldOffset(120)] public TFPoint arEnd;
        [FieldOffset(136)] public TFPoint arCenter;
        [FieldOffset(152)] public TRotateOrient arOrient;
        [FieldOffset(156)] public TLineEndType arEndType;

        [FieldOffset(104)] public TFPoint cirCXY;

        [FieldOffset(104)] public TFPoint lneSXY;
        [FieldOffset(120)] public TFPoint lneEXY;

        [FieldOffset(104)] public TFPoint rcCXY;
        [FieldOffset(120)] public TFloat rcRadY;

        [FieldOffset(104)] public IntPtr PPolygon;    //不能用  TFPoint[] PPolygon  會當掉
        [FieldOffset(112)] public int plgAryLength;

        [FieldOffset(104)] public IntPtr PPolyLine;//IntPtr
        [FieldOffset(112)] public int plnAryLength;

        [FieldOffset(104)] public IntPtr PSegments;//IntPtr
        [FieldOffset(112)] public int segAryLength;

        [FieldOffset(104)] public TIslandHoleShapeMode vstIhsMode;
        [FieldOffset(108)] public IntPtr ihShapeList;      //IntPtr
        [FieldOffset(116)] public int shpCount;
}
```

下載與影片連結

教學影片 ：

https://youtube.com/playlist?list=PLZG_AEGYW1gI1W2-EWJtz_zPMyVE5gcWM&si=hQLKVLmbPTXlYBOp

下載完整 CSharp 程式碼

https://www.rasvector.url.tw/RvCamDLL/TestRvCamDLL_CSharp.rar

使用手冊

https://www.rasvector.url.tw/RvCamDLL/RvCamDLL%20%E4%BD%BF%E7%94%A8%E6%89%8B%E5%86%8A.pdf

Q&A

Q：我要怎麼輸出 CAD 參考影像，和設備的掃描圖形作檢查?

A：參考 常用開檔、轉檔、算圖與存圖功能 這一節說明。

Q：設備掃描的影像有扭曲，和 CAD 影像不完全疊合，我要怎麼作檢查?

A: 將 CAD 和 掃描影像 作補償校正，建立補償表。然後將 CAD 圖形以補償表輸出成匹配設備扭曲的影像，接著和設備掃描的影像作比對檢查。