RvConverterDLL 使用說明書

2018/7/6

影量科技 RasVector Technology.
http://www.RasVector.url.tw

RvConvert 程式與使用範例下載

www.rasvector.url.tw/Download Pages/Download_Engineering.htm

教學影片

RvConverter DLL 使用示範 - YouTube
https://goo.gl/f9AWia

內容

修改歷史紀錄

2021 / 12 / 22

1. 新增 快速教學。

2021 / 09 / 24

1. 新增設定 ODB 繪圖時是否顯示 Child Steps。
   DLL_Set_Visible_ODB_ChildSteps()

2021 / 06 / 30

1. 新增 傳回 ODB 某個 Layer，所有經過他的鑽孔層。
   DLL_GetODB_RelatedDrillLayers()
2. 新增 4-Bytes align 轉換函式。
   DLL_Image_Align4Bytes()

2021 / 06/ 19

1. 新增記憶體中的圖形 FlipX, FlipY, Rotate,Scale
   DLL_Image_FlipX()
   DLL_Image_FlipY()
   DLL_Image_Rotate()
   DLL_Image_Scale()

2021 / 06/ 15

1. 新增 描繪記憶體中的圖形(全圖) 到 畫布上
   DLL_Paint_Home()
2. 新增 描繪記憶體中的圖形(設定區域) 到 畫布上
   DLL_Paint_Zoom()

2021 / 06/ 07

3. 新增 取得 ODB 最底層 Child Steps 清單
   DLL_GetOdbBottomStepsLayers()
4. 新增 取得 ODB 不屬於 Parent 或 Child 的獨立 Steps 清單
   DLL_GetOdbIndependentStepsLayers()
5. 新增擬真繪圖到記憶體函式 DLL_RenderImageInMemory_SimulateRealityA();
   DLL_RenderImageInMemory_SimulateRealityA ()。請參考圖三。

2021 / 06/ 03

1. 新增 ODB Parent XY -> Child XY 函式，輸入 Parent Step 座標系統上的 Parent XY，傳回在哪個

Child Step 座標系統內，及 Child Step 座標系統內的 ChildStep XY
DLL_Inside_OdbParentChildStep()

2021 / 06/ 02

1. 新增 排版函式、反排版函式
   DLL_GetStepRepeatXY0 ()
   DLL_ReverseGetStepRepeatXY0 ()

2021 / 06/ 01

1. 新增 將繪製到記憶體的圖形存出圖檔
   DLL_SaveImageFromMemory()
2. 新增 只取得 ODB 有繼承關係的母子 Steps.
   DLL_GetOdbStepLayersA()
3. 新增 取得 ODB 所有非 Child 的最上層 Steps.
   DLL_GetOdbTopStepsLayers()

2021 / 05/ 28

1. 新增 取得 ODB 特定 Step, 特定 Layer 的範圍
   DLL_GetOdbStepLayerMinMax()

2021 / 05/ 27

1. 新增 解壓縮 TGZ 檔案函式
   DLL_UnZipTgzFile()

2021 / 05/ 26

1. 新增 影像<->資料 座標轉換函式
   DLL_View_Pixel2MmXY()
   DLL_View_Mm2PixelXY()
2. 新增 快速 影像<->資料 座標轉換函式，適合轉換大量座標時使用
   DLL_QuickView_UpdateView() 轉換之前必須先執行一次 DLL_QuickView_UpdateView()。
   DLL_QuickView_Pixel2MmXY()
   DLL_QuickView_Mm2PixelXY()

2021 / 05/ 24

1. 新增擬真繪圖輸出圖檔函式 DLL_OutputImageFile_SimulateReality();
   DLL_OutputImageFile_SimulateRealityA()。請參考圖三。

2021 / 05/ 18

1. 新增 CAD(Gerber274X, NC, DXF, DWG…) 繪圖輸出函式
   DLL_OutputImageFile_CAD();
2. 新增 CAD(Gerber274X, NC, DXF, DWG…) 繪圖函式
   DLL_RenderImageInMemory_CAD ();

2021/5/ 17

1. 新增 ODB 轉檔函式
   DLL_FileConvert_Odb2CAD();
2. 新增 CAD(Gerber274X, NC, DXF, DWG…)　檔轉檔函式
   DLL_FileConvert_CAD2CAD();

2021 / 05/ 13

1. 新增取得 ODB 排版資訊
   DLL_GetOdbStepRepeatInfo ( )。

2021 / 03 / 01

1. 新增讀圖檔到記憶體
   DLL_LoadImageInMemory( )。

2018 / 07 / 09

1. 新增繪圖到記憶體
   DLL_RenderImageInMemory( )。

快速教學

讀入 CAD 檔案取得圖形資料指標

ODB++/TGZ 檔案讀取與繪圖

```
function DLL_RenderImageInMemory(
    const sOdbDirOrTgzFileName:PAnsiChar;
    const sStepName, sLyrNames:PAnsiChar;
    const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double; outputDPI:double;
    var pImageStart0:Pointer;
    var imageSizeTotalMB, stride_BytesPerRow;
    var imagePixelWidth, imagePixelHeight:integer;
    outputBitPerPixel:byte=8;
    sSaveToBmpFileName:PAnsiChar=nil
    ):TReturnCode; stdcall;
```

ODB++/TGZ 檔案讀取與擬真繪圖

```
function DLL_RenderImageInMemory_SimulateRealityA(
  const sOdbDirOrTgzFileName:PAnsiChar;
  const sStepName:PAnsiChar;   atSide:TVectSide;
  const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double; outputDPI:double;
  var pImageStart0:Pointer;
  var imageSizeTotalMB, stride_BytesPerRow;
  var imagePixelWidth, imagePixelHeight:integer;
  outputBitPerPixel:byte=8;
  sSaveToBmpFileName:PAnsiChar=nil
  ):TReturnCode; stdcall;
```

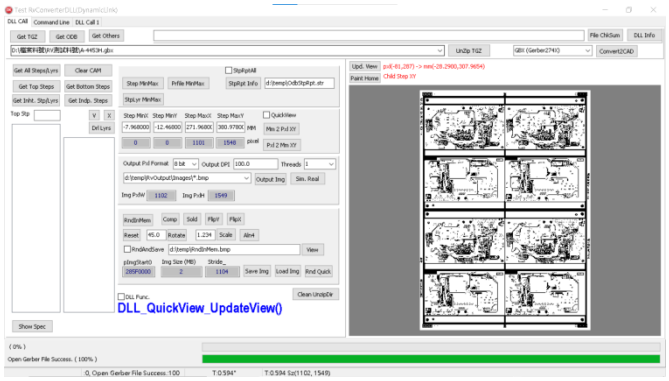
Gerber/DXF/NC CAD 檔案讀取與繪圖

```
function DLL_RenderImageInMemory_CAD(
    const sCadFileName:PAnsiChar;
    const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double; outputDPI:double;
    var pImageStart0:Pointer;
    var imageSizeTotalMB, stride_BytesPerRow;
    var imagePixelWidth, imagePixelHeight:integer;
    outputBitPerPixel:byte=8;
    sSaveToBmpFileName:PAnsiChar=nil
```

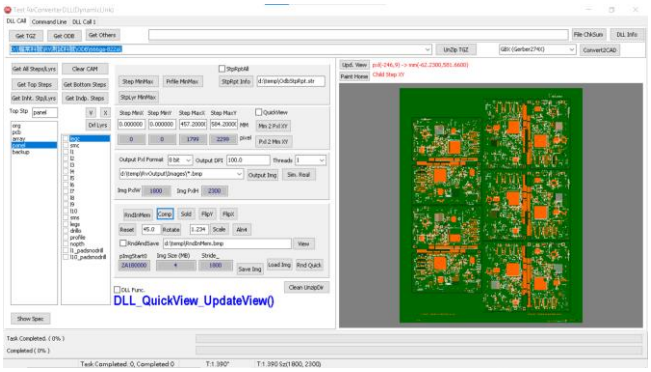
):TReturnCode; stdcall;

顯示圖形資料

一般繪圖



擬真繪圖



全圖顯示

function DLL_Paint_Home(

const CnvScan0: Pointer; CnvRowBytes, CnvWidth, CnvHeight, CnvBitsPerPixel: integer;

const ImgScan0: Pointer; ImgRowBytes, ImgWidth, ImgHeight, ImgBitsPerPixel: integer;

cnvDIBUpWard:LongBool=true; ImgDIBUpWard: LongBool=true;

pReturnRastView:PRastView=nil):TReturnCode; stdcall;

縮放顯示

function DLL_Paint_Zoom(

ImgZoomMinX,ImgZoomMinY,ImgZoomMaxX,ImgZoomMaxY:integer;

const CnvScan0: Pointer; cnvRowBytes, CnvWidth, CnvHeight, CnvBitsPerPixel: integer;

const ImgScan0: Pointer; ImgRowBytes, ImgWidth, ImgHeight, ImgBitsPerPixel: integer;

cnvDIBUpWard, ImgDIBUpWard: LongBool;

pReturnRastView:PRastView=nil):TReturnCode; stdcall;

存出圖形資料

將圖形指標資料存出圖檔

function DLL_SaveImageFromMemory(

const sImageFileName:PAnsiChar;

const pImageStart0:Pointer; const imageSizeTotalMB,

stride_BytesPerRow, imagePixelWidth, imagePixelHeight:integer;

const bitPerPixel:byte):TReturnCode; stdcall;

檔案格式轉換

ODB++ 轉換成圖檔

```
function DLL_OutputImageFile(
    const sOdbDirOrTgzFileName:PAnsiChar;
    const sStepName, sLyrNames:PAnsiChar;
    const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double; outputDPI:double;
    const sOutputFullFileName:PAnsiChar;
    var imagePixelWidth, imagePixelHeight:integer;
    outputBitPerPixel:byte=8;
    threadCount:integer=0):TReturnCode; stdcall;
```

Gerber/DXF/NC CAD 轉換成圖檔

```
function DLL_OutputImageFile_CAD(
    const sCadFileName:PAnsiChar;
    const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double; outputDPI:double;
    const sOutputFullFileName:PAnsiChar;
    var imagePixelWidth, imagePixelHeight:integer;
    outputBitPerPixel:byte=8;
    threadCount:integer=0):TReturnCode; stdcall;
```

ODB++ 轉換成 Gerber/DXF/NC CAD 檔案

```
function DLL_FileConvert_Odb2CAD (
    const sOdbDirOrTgzFileName:PAnsiChar;
    const sStepName:PAnsiChar;
    const sLayerNames:PAnsiChar;
    const sOutputCadFileName:PAnsiChar):TReturnCode; stdcall;
```

Gerber/DXF/NC CAD 檔案之間的互相轉換

```
function DLL_FileConvert_CAD2CAD (
    const sInputCadFileName:PAnsiChar;
    const sOutputCadFileName:PAnsiChar):TReturnCode; stdcall;
```

通用轉換函式

```
function DLL_VectImportExport(
        const sInputParams:PAnsiChar; const sOutputParams:PAnsiChar;
        blShowForm:LongBool=false):TReturnCode; stdcall;
```

DLL 輸出函式說明與宣告

函式功能說明

DLL_AssignCallBackFunc( );
指定 CallBack 函式，以便接收程式執行進度。

DLL_CleanUnZipBuffer( );
清除 ODB TGZ 解壓縮暫存區，避免相同 ODB++卻壓縮成不同名稱 TGZ 造成的問題。

DLL_ClearCamData( );
清除記憶體中的 Cam 資料。

DLL_FileConvert_Odb2CAD( );
檔案轉換，指定輸入的 ODB++ / Step/Layers，輸出的 CAD 檔名(Gerber274X, ExcellonNC,DXF…)。

DLL_FileConvert_CAD2CAD( );
檔案轉換，指定輸入的 CAD 檔名，輸出的 CAD 檔名，不同格式間的轉換。

DLL_GetStepRepeatXY0( );
傳入 oXY，經過旋轉、鏡射、位移之後得到新的排版 XY

DLL_GetReverseStepRepeatXY0( );
傳入排版 XY，反向位移、鏡射、旋轉之後得到的 oXY

DLL_GetODB_RelatedDrillLayers( );
取的 ODB++/TGZ 料號的在某層，經過該層的 所有鑽孔層名稱清單。

DLL_GetOdbStepLayers( );
取的 ODB++/TGZ 料號的 所有 Steps / Layers 名稱清單。

DLL_GetOdbStepLayersA( );
只取得 ODB 有繼承關係的母子 Steps, Layers.

DLL_GetOdbTopStepsLayers( );
取的 ODB++/TGZ 料號的 所非 Child 的最上層 Steps 和 Layers 名稱清單。

DLL_GetOdbBottomStepsLayers( );
取的 ODB++/TGZ 料號的 指定的 TopStep 所屬或所有最底層 Child Steps 和 Layers 名稱清單。

DLL_GetOdbIndependentStepsLayers( );
取的 ODB++/TGZ 料號的 所有不屬於 Parent 或 Child 的獨立 Steps 和 Layers 名稱清單。

DLL_GetOdbStepMinMax( );
取得指定的 ODB++ / Step 的資料範圍大小。

DLL_GetOdbStepLayerMinMax( );
取得指定的 ODB++ / Step / Layer 的資料範圍大小。

DLL_GetCamFileMinMax( );
取得非 ODB++檔案的資料範圍大小。

DLL_GetOdbStepRepeatInfo( );
取得指定的 ODB++ / Step 的排版資訊。

DLL_Image_FlipX( );
記憶體中圖形 FlipX。

DLL_Image_FlipY( );
記憶體中圖形 FlipY。

DLL_Image_Rotate( );
記憶體中圖形旋轉任意角度。

DLL_Image_Scale( );
記憶體中圖形任意比例縮放。

DLL_ImageAlign4Bytes();
將記憶體中的圖形資料的一列的總 Bytes 數，轉換成 4 Bytes 的倍數，以符合某些編譯器只支援 4Bytes-Align 造成的圖形顯示錯誤。

DLL_Inside_OdbParentChildStep ( );
輸入 ODB Parent Step 座標系統上的 Parent XY，傳回位在哪個 Child Step 座標系統內，及 Child Step 座標系統內的 ChildStep XY。

DLL_IsAuthorized( );
檢查程式是否為合法授權。

DLL_LoadImageInMemory ( );
讀入圖檔圖形到記憶體並傳回 指標、長寬像素、格式。

DLL_OutputImageFile( );

根據輸出條件，輸出 Bmp 影像檔案。

DLL_OutputImageFile_CAD( );
讀入 CAD 檔案，根據輸出條件，輸出 Bmp 影像檔案。

DLL_OutputImageFile_SimulateReality( );
根據輸出條件，輸出模擬真實 PCB 版外觀的彩色 Bmp 影像檔案。參考圖三

DLL_OutputImageFile_SimulateRealityA( );
根據輸出條件，輸出模擬真實 PCB 版外觀的彩色 Bmp 影像檔案。參考圖三

DLL_Paint_Home( );
將記憶體中的圖形以全圖繪畫到畫布上

DLL_Paint_Zoom( );
將記憶體中的圖形以輸入的範圍繪畫到畫布上

DLL_QuickView_UpdateView( );
影像/資料 大量高效率座標轉換，只要有變更資料和影像的輸出範圍，執行 DLL_QuickView_XXX() 轉換之前，都必須先執行一次此函式。

DLL_QuickView_Mm2PixelXY( );
影像/資料 大量高效率座標轉換，Mm XY 轉 Pixel XY。

DLL_QuickView_Pixel2MmXY( );
影像/資料 大量高效率座標轉換，Pixel XY 轉 Mm XY。

DLL_RenderImageInMemory( );
根據輸出條件，在記憶體繪圖並傳回指標。

DLL_RenderImageInMemory_SimulateRealityA( );
根據輸出條件，在記憶體繪製擬真圖形圖並傳回指標。

DLL_RenderImageInMemory_CAD( );
讀入 CAD 檔案，根據輸出條件，在記憶體繪圖並傳回指標。

DLL_SaveImageFromMemory ( );
將繪製到記憶體的圖形，存出到圖檔。

DLL_UnZipTgzFile( );
將 ODB TGZ 解壓縮到指定的資料夾，並傳回 ODB++ 料號目錄。

DLL_View_Mm2PixelXY( );
影像/資料 座標轉換，Mm XY 轉 Pixel XY。

DLL_View_Pixel2MmXY( );
影像/資料 座標轉換，Pixel XY 轉 Mm XY。

DLL_VectImportExport( );
輸入並輸出檔案。

宣告方式

Delphi Pascal

資料型態 ：

Call Back 函式
```
TRunningProgress = procedure(var aMainPercent, aSubPercent:byte;
      const aMainTask,aSubTask:PAnsiChar);
```

資料單位
```
TValueUnit = (uInch=0, uMil, uCM, uMM,uUM);
```

函式傳回值
```
TReturnCode=(rcFail=0, rcSuccess, rcUnAuthorized, rcFinal);
```

PCB Side
```
TVectSide=( vsNone=0, vsComp, vsSold, vsBoth);
```

檢視函式的矩陣運算參數結構指標
```
PVectView = Pointer;
```

函式宣告與參數說明 ：

```
procedure DLL_AssignCallBackFunc(
      const pCallBackFunc:Pointer); stdcall;
```
說明 ： 指定 Callback 函式，接收 RvConverterDLL 的執行進度。
參數 ：Callback 函式指標。

Function DLL_GetODB_RelatedDrillLayers(

const sOdbDirTgzFileName:PAnsiChar;

const sAtLayer:PAnsichar; const sRelatedDrillLayers:PAnsiChar

):TReturnCode; stdcall

說明 ： 取得 ODB 某一層，經過該層的所有鑽孔層名稱清單。

參數 ：

sOdbDirTgzFileName: 完整路徑的 ODB++ 資料夾或 TGZ 檔案

sAtLayer: 指定一個板層。

sRelatedDrillLayers: 傳回的所有經過 sAtLayer 的鑽孔層清單(“ivh1-2,ivh3-5,pth”)。

function DLL_GetOdbStepLayers(

const sOdbDirOrTgzFileName:PAnsiChar;

const sSteps:PAnsichar;

const sLayers:PAnsiChar): TReturnCode; stdcall;

說明 ： 取得 ODB 所有 Steps, Layers 名稱清單。

參數 ：

完整路徑的 ODB++ 資料夾或 TGZ 檔案

傳回的 Steps 清單(“panel,array,pcb”)

傳回的 Layers 清單(“cslk,csm,comp,l1,l2,l3,l4,l5,sold,ssm,sslk,pth”)。

function DLL_GetOdbStepLayersA(

const sOdbDirOrTgzFileName:PAnsiChar;

const sSteps:PAnsichar;

const sLayers:PAnsiChar;

blGetTopStepAndInheritedChildStepsOnly:boolean=false;

atTopStepName:PAnsiChar=nil): TReturnCode; stdcall;

說明 ： 參數決定是否只取得 ODB 有繼承關係的母子 Steps, Layers 名稱清單。

參數 ：

完整路徑的 ODB++ 資料夾或 TGZ 檔案

傳回的 Steps 清單(“panel,array,pcb”)

傳回的 Layers 清單(“cslk,csm,comp,l1,l2,l3,l4,l5,sold,ssm,sslk,pth”)。

是否只取有繼承關係的母子 Steps, blGetTopStepAndInheritedChildStepsOnly

指定最上層 Step 名稱, atTopStepName

function DLL_GetOdbTopStepsLayers(

const sOdbDirOrTgzFileName:PAnsiChar;

const sSteps:PAnsichar;

const sLayers:PAnsiChar): TReturnCode; stdcall;

說明 ： 取得 ODB 所有非 Child 的最上層 Steps 和 Layers 名稱清單。

參數 ：

完整路徑的 ODB++ 資料夾或 TGZ 檔案

傳回的 Steps 清單(“panel,array,pcb”)
傳回的 Layers 清單(“cslk,csm,comp,l1,l2,l3,l4,l5,sold,ssm,sslk,pth”)。

function DLL_GetOdbBottomStepsLayers(

const sOdbDirOrTgzFileName:PAnsiChar;
const sSteps:PAnsichar;
const sLayers:PAnsiChar;
atTopStepName:PAnsiChar=nil): TReturnCode; stdcall;

說明 : 取得 ODB 指定 topStep 所屬或所有最底層 Steps 和 Layers 名稱清單。
參數 :

sOdbDirOrTgzFileName: 完整路徑的 ODB++ 資料夾或 TGZ 檔案
sSteps: 傳回的 Steps 清單(“panel,array,pcb”)
sLayers: 傳回的 Layers 清單(“cslk,csm,comp,l1,l2,l3,l4,l5,sold,ssm,sslk,pth”)。
atTopStepName: 指定 ParentStep，只傳回 ParentStep 所屬的最底層 Child Steps 清單。
若 atTopStpName=’’，則傳回所有最底層的 Child Steps 清單。

function DLL_GetOdbIndependentStepsLayers(

const sOdbDirOrTgzFileName:PAnsiChar;
const sSteps:PAnsichar;
const sLayers:PAnsiChar): TReturnCode; stdcall;

說明 : 取得 ODB 不屬於 Parent 或 Child 的獨立 Steps 和 Layers 名稱清單。通常都是那些備份資料，不是實際生產板子的 Steps 資料。
參數 :

完整路徑的 ODB++ 資料夾或 TGZ 檔案
傳回的 Steps 清單(“panel,array,pcb”)
傳回的 Layers 清單(“cslk,csm,comp,l1,l2,l3,l4,l5,sold,ssm,sslk,pth”)。

function DLL_GetOdbStepMinMax(

const sOdbDirOrTgzFileName:PAnsiChar;
const sStepName:PAnsiChar;
var stepMinX,stepMinY,stepMaxX,stepMaxY:double;
valueUnit:TValueUnit=uMM;
inStepProfileMinMax:boolean=True):TReturnCode; stdcall;

說明 : 取得 ODB 某個 step 的資料範圍大小。
參數 :

完整路徑的 ODB++ 資料夾或 TGZ 檔案
指定的 Step 名稱
傳回的 Step MinX,MinY,MaxX,MaxY
指定傳回值的資料單位
是否以 StepProfile 大小為傳回範圍。

function DLL_GetOdbStepLayerMinMax(

const sOdbDirOrTgzFileName:PAnsiChar;

const sStepName:PAnsiChar;

const sLayerName:PAnsiChar;

var stepMinX,stepMinY,stepMaxX,stepMaxY:double;

valueUnit:TValueUnit=uMM):TReturnCode; stdcall;

說明 ： 取得 ODB 某個 step 特定 Layer 的資料範圍大小。

參數 ：

完整路徑的 ODB++ 資料夾或 TGZ 檔案

指定的 Step, Layer 名稱

傳回的 Step MinX,MinY,MaxX,MaxY

指定傳回值的資料單位。

function DLL_GetCamFileMinMax(

const sCamFileName:PAnsiChar;

var aMinX,aMinY,aMaxX,aMaxY:double;

valueUnit:TValueUnit=uMM):TReturnCode; stdcall;

說明 ： 取得非 ODB 檔案(Gerber, DXF, Excellon NC, DPF…CAM 檔案) 的資料範圍大小。

參數 ：

完整路徑的檔案名稱

傳回的 Step MinX,MinY,MaxX,MaxY

指定傳回值的資料單位。

function DLL_OutputImageFile(

const sOdbDirOrTgzFileName:PAnsiChar;

const sStepName, sLyrNames:PAnsiChar;

const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double;

outputDPI:double;

const sOutputFullFileName:PAnsiChar;

var imagePixelWidth, imagePixelHeight:integer;

outputBitPerPixel:byte=8;

threadCount:integer=0):TReturnCode; stdcall;

說明 ： 傳入輸入檔案(ODB 則包含 Step, Layer(s)) 名稱，根據輸出條件，輸出 Bmp。

參數 ：

完整路徑的檔案名稱

指定的 Step 名稱 / 指定的 Layers 名稱 (ODB 才需要，可一次傳入多層，
以”,”隔開。Eg: “comp,l2,l3,l4”)

自定輸出範圍 output MinMax，當全部為 0 時，程式會即時自動取得資料的大小範圍。

指定輸出的解析度，單位 DPI (dot per inch)。

指定輸出的路徑和檔案格式。Eg : “d:\RvOutput\Images*.bmp”

傳回輸出的影像尺寸，單位 Pixel。

指定輸出的像素格式 1/8/24 bpp (bit per pixel)

指定多執行緒輸出數目 (0:程式自動優化選取， 1~8 : 自行指定執行緒數目，輸出所需時間和執行緒數目成反比。 Time (N-thread) = Time (oneThread) / threadCount。

function DLL_OutputImageFile_SimulateReality(

const sOdbDirOrTgzFileName:PAnsiChar;
const sStepName, sLyrNames:PAnsiChar;
const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double;
outputDPI:double;
const sOutputFullFileName:PAnsiChar;
var imagePixelWidth, imagePixelHeight:integer;
outputBitPerPixel:byte=8):TReturnCode; stdcall;

說明 : 傳入 ODB 檔案，根據輸出條件，輸出 模擬真實版子彩色外觀的 Bmp。請參考圖三。

參數 :

完整路徑的檔案名稱
指定的 Step 名稱 / 指定的 Layers 名稱 (ODB 才需要，需傳入 Comp / Sold Side 的層名)
指定輸出範圍 MinX,MinY,MaxX,MaxY，當全部為 0 時，程式會即時自動取得資料的大小範圍。
指定輸出的解析度，單位 DPI (dot per inch)。
指定輸出的路徑和檔案格式。Eg : "d:\RvOutput\Images*.bmp"
傳回輸出的影像尺寸，單位 Pixel。
指定輸出的像素格式 8/24 bpp (bit per pixel)

function DLL_OutputImageFile_SimulateRealityA(

const sOdbDirOrTgzFileName:PAnsiChar;
const sStepName:PAnsiChar;
atSide:TVectSide; outputDPI:double;
const sOutputFullFileName:PAnsiChar;
var imagePixelWidth, imagePixelHeight:integer;
outputBitPerPixel:byte=8):TReturnCode; stdcall;

說明 : 傳入 ODB 檔案，根據輸出條件，輸出 模擬真實版子彩色外觀的 Bmp。請參考圖三。

參數 :

完整路徑的檔案名稱
指定的 Step 名稱名稱
指定輸出的 Side (vsComp or vsSold)。
指定輸出的解析度，單位 DPI (dot per inch)。
指定輸出的路徑和檔案格式。Eg : "d:\RvOutput\Images*.bmp"
傳回輸出的影像尺寸，單位 Pixel。
指定輸出的像素格式 8/24 bpp (bit per pixel)

function DLL_RenderImageInMemory(

const sOdbDirOrTgzFileName:PAnsiChar;
const sStepName, sLyrNames:PAnsiChar;
const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double;

```
    outputDPI:double;
    var pImageStart0:Pointer;
    var imageSizeTotalMB, stride_BytesPerRow;
    var imagePixelWidth, imagePixelHeight:integer;
    outputBitPerPixel:byte=8;
    sSaveToBmpFileName:PAnsiChar=nil
    ):TReturnCode; stdcall;
```

說明 ： 傳入輸入檔案(ODB 則包含 Step, Layer(s)) 名稱，根據輸出條件，輸出 Bmp。

參數 ：

完整路徑的檔案名稱

指定的 Step 名稱 / 指定的 Layers 名稱 (ODB 才需要，可一次傳入多層，以”,”隔開。Eg: “comp,l2,l3,l4”)

指定輸出範圍 MinX,MinY,MaxX,MaxY，可利用 DLL_GetOdbStepLayerMinMax() 取值。當全部為 0 時，程式會即時自動取得資料的大小範圍。

指定輸出的解析度，單位 DPI (dot per inch)。

傳回影像在記憶體內的起始指標。

傳回記憶體內影像的 Stride(BytePerRow), 影像的總共大小(Mega Bytes)

傳回輸出的影像尺寸，單位 Pixel。

指定輸出的像素格式 1/8/24 bpp (bit per pixel)

指定輸出檔案名稱 **sSaveToBmpFileName，不輸出則留白(nil)，預設值為 nil**

```
function DLL_RenderImageInMemory_SimulateRealityA(
    const sOdbDirOrTgzFileName:PAnsiChar;
    const sStepName:PAnsiChar;    atSide:TVectSide;
    const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double;
    outputDPI:double;
    var pImageStart0:Pointer;
    var imageSizeTotalMB, stride_BytesPerRow;
    var imagePixelWidth, imagePixelHeight:integer;
    outputBitPerPixel:byte=8;
    sSaveToBmpFileName:PAnsiChar=nil
    ):TReturnCode; stdcall;
```

說明 ： 傳入輸入檔案(ODB 則包含 Step, Layer(s)) 名稱，根據輸出條件，在記憶體繪製擬真圖形。

參數 ：

完整路徑的檔案名稱

指定的 Step 名稱(ODB 才需要)

指定輸出的 Side (TVectSide vsComp or vsSold)

指定輸出範圍 MinX,MinY,MaxX,MaxY，當全部為 0 時，程式會即時自動取得資料的大小範圍。

指定輸出的解析度，單位 DPI (dot per inch)。

傳回影像在記憶體內的起始指標。

傳回記憶體內影像的 Stride(BytePerRow), 影像的總共大小(Mega Bytes)

傳回輸出的影像尺寸，單位 Pixel。
指定輸出的像素格式 1/8/24 bpp (bit per pixel)
指定輸出檔案名稱 **sSaveToBmpFileName，不輸出則留白**

function DLL_Image_FlipX(

const pImageStart0:Pointer;
stride_BytesPerRow, imagePixelWidth, imagePixelHeight:integer; BitPerPixel:byte
):TReturnCode; stdcall;

說明 ：將記憶體中的圖形資料做 X 翻轉。
參數 ：
pImageStart0 :影像在記憶體內的起始指標。
Stride_BytesPerRow: 影像每列的大小 bytes。
imagePixelWidth,imagePixelHeight : 影像的寬長 Pixel 數。
BitPerPixel :影像每個像素的 Bit 數。

function DLL_Image_FlipY(

const pImageStart0:Pointer;
stride_BytesPerRow, imagePixelWidth, imagePixelHeight:integer; BitPerPixel:byte
):TReturnCode; stdcall;

說明 ：將記憶體中的圖形資料做 Y 翻轉。
參數 ：
pImageStart0 :影像在記憶體內的起始指標。
Stride_BytesPerRow: 影像每列的大小 bytes。
imagePixelWidth,imagePixelHeight : 影像的寬長 Pixel 數。
BitPerPixel :影像每個像素的 Bit 數。

function DLL_Image_Rotate(

const pImageStart0:Pointer;
stride_BytesPerRow, imagePixelWidth, imagePixelHeight:integer;
BitPerPixel:byte; rotateDegree:double;
var pRotImageStart0:Pointer; var rotStride_BytesPerRow, rotImagePixelWidth,
rotImagePixelHeight:integer):TReturnCode; stdcall;

說明 ：將記憶體中的圖形資料旋轉任意角度。
參數 ：
pImageStart0 :影像在記憶體內的起始指標。
Stride_BytesPerRow: 影像每列的大小 bytes。
imagePixelWidth,imagePixelHeight : 影像的寬長 Pixel 數。
BitPerPixel :影像每個像素的 Bit 數。
rotateDegree : 旋轉角度。
pRotImageStart0 : 旋轉好的新影像在記憶體內的起始指標。
rotStride_BytesPerRow: 旋轉好的新影像每列的大小 bytes。
rotImagePixelWidth,rotImagePixelHeight : 旋轉好的新影像的寬長 Pixel 數。

function DLL_Image_Scale(

const pImageStart0:Pointer;

stride_BytesPerRow, imagePixelWidth, imagePixelHeight:integer;

BitPerPixel:byte; scale:double;

var pScaleImageStart0:Pointer; var scaleStride_BytesPerRow, scaleImagePixelWidth,

scaleImagePixelHeight:integer):TReturnCode; stdcall;

說明 ： 將記憶體中的圖形資料做任意比例縮放。

參數 ：

pImageStart0 :影像在記憶體內的起始指標。

Stride_BytesPerRow: 影像每列的大小 bytes。

imagePixelWidth,imagePixelHeight : 影像的寬長 Pixel 數。

BitPerPixel :影像每個像素的 Bit 數。

scale : 縮放比例。

pScaleImageStart0 :新影像在記憶體內的起始指標。

scaleStride_BytesPerRow:新影像每列的大小 bytes。

scaleImagePixelWidth, scaleImagePixelHeight :新影像的寬長 Pixel 數。

function DLL_Image_Align4Bytes(

const pImageStart0:Pointer;

stride_BytesPerRow, imagePixelWidth, imagePixelHeight:integer;

BitPerPixel:byte;

var pAlignedImageStart0:Pointer;

var alignedImagePixelWidth, alignedStride_BytesPerRow:integer):TReturnCode; stdcall;

說明 ： 將記憶體中的圖形資料的一列的總 Bytes 數，轉換成 4 Bytes 的倍數，以符合某些編譯器只支援 4Bytes-Align 造成的圖形顯示錯誤。

注意 ： 若使用此函式，圖形的像素寬度將可能被改變，使用者必須更新 DLL_RenderImageInMemory() 時，所使用的 outputMaxX_mm，以符合解析度運算。

參數 ：

pImageStart0 :影像在記憶體內的起始指標。

Stride_BytesPerRow: 影像每列的大小 bytes。

imagePixelWidth,imagePixelHeight : 影像的寬長 Pixel 數。

BitPerPixel :影像每個像素的 Bit 數。

pAlignedImageStart0 : 新影像在記憶體內的起始指標。

alignedImagePixelWidth: 新影像的寬 Pixel 數

alignedStride_BytesPerRow:新影像每列的大小 bytes。

function DLL_OutputImageFile_CAD(

const sCadFileName:PAnsiChar;

const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double;

outputDPI:double;

const sOutputFullFileName:PAnsiChar;

var imagePixelWidth, imagePixelHeight:integer;

outputBitPerPixel:byte=8;

threadCount:integer=0):TReturnCode; stdcall;

說明 ： 傳入輸入 CAD 檔案(Gerber274X,NC,DXF,DWG…) 名稱，根據輸出條件，輸出 Bmp。

參數 ：

完整路徑的檔案名稱

指定輸出範圍 MinX,MinY,MaxX,MaxY，可利用 DLL_GetCamFileMinMax() 取值。

當全部為 0 時，程式會即時自動取得資料的大小範圍。

指定輸出的解析度，單位 DPI (dot per inch)。

指定輸出的路徑和檔案格式。Eg : “d:\RvOutput\Images*.bmp”

傳回輸出的影像尺寸，單位 Pixel。

指定輸出的像素格式 1/8/24 bpp (bit per pixel)

指定多執行緒輸出數目 (0:程式自動優化選取， 1~8 : 自行指定執行緒數目，輸出所需時間和執行緒數目成反比。 Time (N-thread) = Time (oneThread) / threadCount。

function DLL_RenderImageInMemory_CAD(

const sCadFileName:PAnsiChar;

const outputMinX_mm,outputMinY_mm,outputMaxX_mm,outputMaxY_mm:double;

outputDPI:double;

var pImageStart0:Pointer;

var imageSizeTotalMB, stride_BytesPerRow;

var imagePixelWidth, imagePixelHeight:integer;

outputBitPerPixel:byte=8;

sSaveToBmpFileName:PAnsiChar=nil

):TReturnCode; stdcall;

說明 ： 傳入輸入檔案 CAD 檔案(Gerber274X,NC,DXF,DWG…)名稱，根據輸出條件，輸出 Bmp。

參數 ：

完整路徑的檔案名稱

指定輸出範圍 MinX,MinY,MaxX,MaxY，可利用 DLL_GetCamFileMinMax() 取值。

當全部為 0 時，程式會即時自動取得資料的大小範圍。

指定輸出的解析度，單位 DPI (dot per inch)。

傳回影像在記憶體內的起始指標。

傳回記憶體內影像的 Stride(BytePerRow), 影像的總共大小(Mega Bytes)

傳回輸出的影像尺寸，單位 Pixel。

指定輸出的像素格式 1/8/24 bpp (bit per pixel)

指定輸出檔案名稱 **sSaveToBmpFileName，不輸出則留白**

function DLL_VectImportExport(

const sInputParams:PAnsiChar; const sOutputParams:PAnsiChar;

blShowForm:LongBool=false):TReturnCode; stdcall;

說明 ： 傳入輸入字串(sInputParams)，輸出字串(sOutputParams)，完成多檔轉換。

參數 ：

sInputParams :

["ODB folder / TGZ / Gerber / DXF / Zip / Rar... file"][@ODB Step Name][@Odb Lyr1,lyr2...]
sOutputParams:
["Full Output FileName][@b#][@r#][@omm,minx_mm,miny_mm,max_mm,maxy_mm]
例:
sInputParams: 輸入字串參數
"d:\odb\666ga-882\@panel@comp,l2,l3,l4,l5,l6,sold" // 輸入 odb 資料夾的 panel/l2,l3,l4,l5,l6,sold 多層資料
"d:\odb\ab53a.tgz@panel@comp,l2,l3,l4,l5,l6,sold" // 輸入 tgz 檔案的 panel/l2,l3,l4,l5,l6,sold 多層資料
"d:\gerbers\comp.gbx" // 輸入一個 gerber 檔案
"d:\gerbers" // 輸入 gerbers 資料夾下的所有檔案
"d:\CompressedFiles\allData.zip" // 解壓縮並輸入 allData.zip 內所有的檔案
***支援輸入的檔案格式有 ODB++, TGZ, Gerber274X, DXF, Excellon NC, DPF, BMP, IPC356a, MMF... 依選購模組不同而有差異。**

sOutputParams : 輸出字串參數。
"d:\RvOutput\Images*.bmp" // 輸出所有層的資料為 Windows BMP 檔案
"d:\RvOutput\Gerbers*.gbx" // 輸出所有層的資料為 Gerber274X 檔案
其他支援輸出檔案格式 :
.bmp(影像)，.gbx(Gerber 274X)，*.enc(Excellon F1 NC)，*.dxf(AutoCad R12 DXF)，*.DPF，*.tgz, *.rar, *.zip ...
***支援輸出的檔案格式有 ODB++, TGZ, Gerber274X, DXF, Excellon NC, DPF, BMP, IPC356a, MMF... 依選購模組不同而有差異。**

function DLL_CleanUnZipBuffer(
const pUnzipFolder:PAnsichar):TReturnCode; stdcall;
說明 : 清除 RvConverterDLL 解壓縮檔案的暫存路徑。有些相同檔案內容卻有不同的壓縮檔名，這時可利用此功能清除解壓縮暫存區。
參數 : 傳回 RvConverterDLL 用來解壓縮檔案所在的完整路徑

function DLL_UnZipTgzFile(
const sTgzFileName:PAnsiChar; const pUnzipFolder:PAnsichar;
const pReturnOdbFolder:PAnsiChar):TReturnCode; stdcall;
說明 : 傳入 TGZ 檔案，指定解壓縮路徑，傳回解壓縮後的 ODB++完整目錄名稱。
參數 : sTgzFileName, pUnZipFolder, pReturnOdbFolder

function DLL_IsAuthorized(
):TReturnCode; stdcall;
說明 : 檢查程式是否為合法授權。
參數 :

function DLL_ClearCamData(

):TReturnCode; stdcall;

說明 ： 清除記憶體中的 Cam 資料。

參數 ：

function DLL_LoadImageInMemory(

const sImageFileName:PAnsiChar;

var pImageStart0:Pointer; var imageSizeTotalMB,

stride_BytesPerRow, imagePixelWidth, imagePixelHeight:integer;

var bitPerPixel:byte):TReturnCode; stdcall;

說明 ： 讀入圖檔到記憶體。支援超級大圖檔案(>20GB BMP)。

參數 ：

sImageFileName 完整路徑的檔案名稱

pImageStart0 傳回影像在記憶體內的起始指標。

imageSizeTotalMB 影像圖形大小 MegaBytes

stride_BytesPerRow 傳回記憶體內影像的 Stride(BytePerRow), 影像的總共大小(MegaBytes)

imagePixelWidth, imagePixelHeight 傳回輸出的影像尺寸，單位 Pixel。

bitPerPixel 傳回像素格式 1/8/24 bpp (bit per pixel)

function DLL_SaveImageFromMemory(

const sImageFileName:PAnsiChar;

const pImageStart0:Pointer; const imageSizeTotalMB,

stride_BytesPerRow, imagePixelWidth, imagePixelHeight:integer;

const bitPerPixel:byte):TReturnCode; stdcall;

說明 ： 將繪製到記憶體的圖形，存出到圖檔。

參數 ：

sImageFileName 完整路徑的檔案名稱

pImageStart0 影像在記憶體內的起始指標。

imageSizeTotalMB 影像圖形大小 Mega bytes

stride_BytesPerRow 記憶體內影像的 Stride(BytePerRow), 影像的總共大小(Bytes)

imagePixelWidth, imagePixelHeight 影像尺寸，單位 Pixel。

bitPerPixel 像素格式 1/8/24 bpp (bit per pixel)

function DLL_GetOdbStepRepeatInfo(

const sOdbDirOrTgzFileName:PAnsiChar;

const sStepName:PAnsiChar;

const outputStepRepeatFullFileName:PAnsiChar;

var stepMinX,stepMinY,stepMaxX,stepMaxY:double;

valueUnit:TValueUnit=uMM;

blExportAllStepRepeatedProfiles:Boolean=false):TReturnCode; stdcall;

說明 ： 取得 ODB 某個 parent step 及其所屬 Child Steps 的排版資料。

參數 ：

完整路徑的 ODB++ 資料夾或 TGZ 檔案
指定的 Step 名稱
指定排版資訊輸出的檔案名稱(*.str)
傳回的 Step MinX,MinY,MaxX,MaxY
指定傳回值的資料單位。valueUnit
是否將所有子片的輪廓點資料 排版後輸出 blExportAllStepRepeatedProfile

function DLL_Inside_OdbParentChildStep (

const parentStepName:PAnsichar;

parentMmX,parentMmY:double; const inSideStepName:PAnsiChar;

var insideMmX,insideMmY:double):TReturnCode; stdcall;

說明 ： 輸入 ODB Parent Step 座標系統上的 Parent XY，傳回位在哪個 Child Step 座標系統內，及 Child Step 座標系統內的 ChildStep XY。

參數 ：

指定的 parent Step 名稱 **parentStepName**

輸入 parentStep 上某個 座標 **parentMm**XY

傳回 **parentMm**XY 在哪個 ChildStepName 內。

傳回 **parentMm**XY 位在 ChildStepName 座標系統上的座標 insideMmXY。

function DLL_FileConvert_Odb2CAD (

const sOdbDirOrTgzFileName:PAnsiChar;

const sStepName:PAnsiChar;

const sLayerNames:PAnsiChar;

const sOutputCadFileName:PAnsiChar):TReturnCode; stdcall;

說明 ： 傳入輸入檔名(ODB 資料夾 或 TGZ 完整檔案名稱)，輸出完整 CAD 檔案名稱。完成多檔轉換。

參數 ：

sOdbDirOrTgzFileName :

ODB++資料夾 或 TGZ 完整檔名。

sStepName:

要輸出的 ODB++ Step 名稱。

sLayerNames:

要輸出的所有 Layer 名稱，並以”,” 逗號隔開。Eg. “L1,l2,l3,sold,drl”。

sOutputCadFileName:

要輸出的 CAD 檔案名稱。需指定正確的有效 CAD 檔案副檔名。

有效 CAD 檔案副檔名:

'.gbx' : Gerber274X

'.enc' : Excellon

'.dpf' : DPF

'.dxf' : DXF

'.rvc' : rvc

'.tgz' : tgz

'.ipc' : ipc
'.emm' : emm
'.mnf2' : ipc

例:
輸出 Gerber
DLL_FileConvert_Odb2CAD ('d:\odb\myOdb.tgz', 'Panel', 'comp,l2,l3,sold,drl', 'd:\output*.gbx');
輸出 AutoCad DXF
DLL_FileConvert_Odb2CAD ('d:\odb\myOdb.tgz', 'Panel', 'comp,l2,l3,sold,drl', 'd:\output*.dxf');

function DLL_FileConvert_CAD2CAD (
const sInputCadFileName:PAnsiChar;
const sOutputCadFileName:PAnsiChar):TReturnCode; stdcall;

說明 : 傳入完整 CAD 輸入檔名，輸出完整 CAD 檔案名稱。完成多檔轉換。
參數 :
sInputCadFileName:
輸入的檔案名稱。
sOutputCadFileName:
要輸出的 CAD 檔案名稱。需指定正確的有效 CAD 檔案副檔名。
有效 CAD 檔案副檔名:
'.gbx' : Gerber274X
'.enc' : Excellon
'.dpf' : DPF
'.dxf' : DXF
'.rvc' : rvc
'.tgz' : tgz
'.ipc' : ipc
'.emm' : emm
'.mnf2' : ipc

例:
輸出 Gerber
DLL_FileConvert_CAD2CAD ('d:\odb\myDXF.dxf.', 'd:\output*.gbx');
輸出 AutoCad DXF
DLL_FileConvert_CD2CAD ('d:\odb\myGerber.gbx', 'd:\output*.dxf');

function DLL_View_Mm2PxelXY (
viewMinX_mm,viewMinY_mm,viewMaxX_mm,viewMaxY_mm:double;
viewMinX_pixel,viewMinY_pixel,viewMaxX_pixel,viewMaxY_pixel:integer;
inputMmX,inputMmY:double;

var outputPixelX,outputPixelY:integer):TReturnCode; stdcall;

說明 ： 傳入實際資料範圍(mm) 和 影像大小範圍(Pixel)。將 mm XY 轉換成 Pixel XY。

參數 ：

viewMinX_mm,viewMinY_mm,viewMaxX_mm,viewMaxY_mm： 資料範圍

viewMinX_pixel,viewMinY_pixel,viewMaxX_pixel,viewMaxY_pixel： 影像範圍。例: 0,0,ImgWidth-1,ImgHeight-1。

inputMmX,inputMmY： 輸入 mm XY

var outputPixelX,outputPixelY： 輸出 Pixel XY

function DLL_View_Pixel2MmXY (

viewMinX_mm,viewMinY_mm,viewMaxX_mm,viewMaxY_mm:double;

viewMinX_pixel,viewMinY_pixel,viewMaxX_pixel,viewMaxY_pixel:integer;

inputPixelX, inputPixelY:integer;

var outputMmX,outputMmY:double):TReturnCode; stdcall;

說明 ： 傳入實際資料範圍(mm) 和 影像大小範圍(Pixel)。將 pixel XY 轉換成 Mm XY。

參數 ：

viewMinX_mm,viewMinY_mm,viewMaxX_mm,viewMaxY_mm： 資料範圍

viewMinX_pixel,viewMinY_pixel,viewMaxX_pixel,viewMaxY_pixel： 影像範圍。例: 0,0,ImgWidth-1,ImgHeight-1。

inputPixelX,inputPixelY： 輸入 Pixel XY

var outputMmX,outputMmY： 輸出 Mm XY

function DLL_QuickView_UpdateView (

viewMinX_mm,viewMinY_mm,viewMaxX_mm,viewMaxY_mm:double;

viewMinX_pixel,viewMinY_pixel,viewMaxX_pixel,viewMaxY_pixel:integer;

pReturnView:PVectView=nil):TReturnCode; stdcall;

說明 ： 只要有變更資料和影像的輸出範圍，執行 DLL_QuickView_XXX() 轉換之前，都必須先執行一次此函式。傳入實際資料範圍(mm) 和 影像大小範圍(Pixel)。

參數 ：

viewMinX_mm,viewMinY_mm,viewMaxX_mm,viewMaxY_mm： 資料範圍

viewMinX_pixel,viewMinY_pixel,viewMaxX_pixel,viewMaxY_pixel： 影像範圍。例: 0,0,ImgWidth-1,ImgHeight-1。

pReturnView： 傳回 TVectView 參數，預設值 nil。

function DLL_QuickView_Mm2PxelXY (

inputMmX,inputMmY:double;

var outputPixelX,outputPixelY:integer;

pAtView:PVectView=nil):TReturnCode; stdcall;

說明 ： 將 mm XY 轉換成 Pixel XY。

參數 ：

inputMmX,inputMmY： 輸入 mm XY

var outputPixelX,outputPixelY： 輸出 Pixel XY

pAtView:指定轉換參數。預設值=nil。

function DLL_QuickView_Pixel2MmXY (

inputPixelX, inputPixelY:integer;

var outputMmX,outputMmY:double;

pAtView:PVectView=nil):TReturnCode; stdcall;

說明 ： 將 pixel XY 轉換成 Mm XY。

參數 ：

inputPixelX,inputPixelY： 輸入 Pixel XY

var outputMmX,outputMmY： 輸出 Mm XY

pAtView:指定轉換參數。預設值=nil。

function DLL_GetStepRepeatXY0 (

const oX,oY:double; var aStpRptX,aStpRptY:double;

datumnX,datumnY:double; ccwRotDeg:double; mirrorX:boolean;

shiftX,shiftY:double):TReturnCode; stdcall;

說明 ： 傳入 oXY，透過排版指令後得到新的排版 aStpRptXY。

參數 ：

oX,oY： 輸入未排版 XY

aStpRptX,aStpRptY： 傳回排版 XY

datumnx,datumnY： 旋轉中心

ccwRotDeg： 逆時鐘旋轉角度

mirrorX： 是否鏡射 X

shiftX,ShiftY： 位移量。

function DLL_GetReverseStepRepeatXY0 (

const aStpRptX,aStpRptY:double; var oX,oY:double;

datumnX,datumnY:double; ccwRotDeg:double; mirrorX:boolean;

shiftX,shiftY:double):TReturnCode; stdcall;

說明 ： 傳入 aStpRptXY，透過排版指令後得到逆排版 XY。

參數 ：

aStpRptX,aStpRptY： 輸入排版 XY

oX,oY: 傳回未排版 XY

datumnx,datumnY： 旋轉中心

ccwRotDeg： 逆時鐘旋轉角度

mirrorX： 是否鏡射 X

shiftX,ShiftY： 位移量。

function DLL_Paint_Home(

const CnvScan0: Pointer; CnvRowBytes, CnvWidth, CnvHeight, CnvBitsPerPixel: integer;

const ImgScan0: Pointer; ImgRowBytes, ImgWidth, ImgHeight, ImgBitsPerPixel: integer;

cnvDIBUpWard:LongBool=true; ImgDIBUpWard: LongBool=true;

pReturnRastView:PRastView=nil):TReturnCode; stdcall;

說明 ：將記憶體中的圖形以全圖繪畫到畫布上。

參數 ：

CnvScan0： 畫布記憶體起始位址

CnvRowBytes, CnvWidth,CnvHeight,CnvBitsPerPixel:畫布的每列像素量、寬度、高度、每像素所 byte 數。

ImgRowBytes, ImgWidth,ImgHeight,ImgBitsPerPixel:記憶體中影像的每列像素量、寬度、高度、每像素所佔 byte 數。

cnvDIBUpWard, ImgDIBUpWard: 畫布和影像的 Y 資料順序。預設值 true。

pReturnRastView： 傳回繪圖時計算並使用的繪圖參數。

function DLL_Paint_Zoom(

ImgZoomMinX,ImgZoomMinY,ImgZoomMaxX,ImgZoomMaxY:integer;

const CnvScan0: Pointer; cnvRowBytes, CnvWidth, CnvHeight, CnvBitsPerPixel: integer;

const ImgScan0: Pointer; ImgRowBytes, ImgWidth, ImgHeight, ImgBitsPerPixel: integer;

cnvDIBUpWard, ImgDIBUpWard: LongBool;

pReturnRastView:PRastView=nil):TReturnCode; stdcall;

說明 ：將記憶體中的圖形以設定的範圍繪畫到畫布上。

參數 ：

ImgZoomMinX,ImgZoomMinY,ImgZoomMaxX,ImgZoomMaxY:縮放繪製的影像範圍。

CnvScan0： 畫布記憶體起始位址

CnvRowBytes, CnvWidth,CnvHeight,CnvBitsPerPixel:畫布的每列像素量、寬度、高度、每像素所 byte 數。

ImgRowBytes, ImgWidth,ImgHeight,ImgBitsPerPixel:記憶體中影像的每列像素量、寬度、高度、每像素所佔 byte 數。

cnvDIBUpWard, ImgDIBUpWard: 畫布和影像的 Y 資料順序。預設值 true。

pReturnRastView： 傳回繪圖時計算並使用的繪圖參數。

function DLL_Set_Visible_ODB_ChildSteps(

childStepVisible:boolean):TReturnCode; stdcall;

說明 ：設定繪圖時是否顯示 ChildSteps。

參數 ：

childStepVisible: 是否顯示 ChildSteps。

CSharp

C# 函式功能與參數的詳細說明，請參考前節 [Delphi Pascal](#) 敘述。

資料型態

```
public enum TValueUnit : int
{
```

```
        uInch = 0, uMil = 1, uCM = 2, uPIXEL = 3, uUM = 4
}

public enum TReturnCode : int
{
    rcFail = 0, rcSuccess, rcUnAuthorized, rcFinal
}

public enum TVectSide : int
{
    vsNone=0, vsComp=1, vsSold=2, vsBoth=3
}
```

函式宣告

```
private const CallingConvention TargetCallingConvention = CallingConvention.StdCall;
private const CharSet TargetCharSet = CharSet.Ansi;

// Callback 函式定義
public delegate void OnRunningProgressHandler(
    ref byte mainPercent, ref byte childPercent, IntPtr mainTask, IntPtr childTask);

// Callback 函式宣告
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_AssignCallBackFunc",
CallingConvention = TargetCallingConvention)]
private static extern void DLL_AssignCallBackFunc(OnRunningProgressHandler callback);

// 取得 ODB/TGZ 某一層，經過該層的所有鑽孔層名稱清單
[DllImport(AssemblyName, EntryPoint = "DLL_GetOdb_RelatedDrillLayers", CallingConvention =
TargetCallingConvention)]
private static extern TReturnCode DLL_GetOdb_RelatedDrillLayers(IntPtr sOdbDirOrTgzFileNam, IntPtr
sAtLayer, IntPtr sRelatedDrillLayers);

// 取得 ODB/TGZ 的 Step, Layers 名稱清單
[DllImport(AssemblyName, EntryPoint = "DLL_GetOdbStepLayers", CallingConvention =
TargetCallingConvention)]
private static extern TReturnCode DLL_GetOdbStepLayers(IntPtr sOdbDirOrTgzFileNam, IntPtr steps,
IntPtr layers);

// 取得 ODB/TGZ 的 Step, Layers 名稱清單，參數設定是否只取得有繼承關係的母子 Steps
[DllImport(AssemblyName, EntryPoint = "DLL_GetOdbStepLayersA", CallingConvention =
```

```
TargetCallingConvention)]
private static extern TReturnCode DLL_GetOdbStepLayersA(IntPtr sOdbDirOrTgzFileNam, IntPtr steps,
IntPtr layers; bool blGetTopStepAndInheritedChildStepsOnly=false;
IntPtr atTopStepName= default(IntPtr));

// 取得 ODB/TGZ 的所有非 Child 的最上層 Steps 和 Layers 名稱清單
[DllImport(AssemblyName, EntryPoint = "DLL_GetOdbTopStepsLayers", CallingConvention =
TargetCallingConvention)]
private static extern TReturnCode DLL_GetOdbTopStepsLayers(IntPtr sOdbDirOrTgzFileNam, IntPtr
sSteps, IntPtr layers);

// 取得 ODB/TGZ 的指定 ParentStep 所屬或所有最底層 Steps 和 Layers 名稱清單
[DllImport(AssemblyName, EntryPoint = "DLL_GetOdbBottomStepsLayers", CallingConvention =
TargetCallingConvention)]
private static extern TReturnCode DLL_GetOdbBottomStepsLayers(IntPtr sOdbDirOrTgzFileNam, IntPtr
sSteps, IntPtr layers, IntPtr atTopStepNames(=default(IntPtr));

// 取得 ODB/TGZ 的所有不屬於 Parent 或 Child 的獨立 Steps 和 Layers 名稱清單
[DllImport(AssemblyName, EntryPoint = "DLL_GetOdbIndependentStepsLayers", CallingConvention =
TargetCallingConvention)]
private static extern TReturnCode DLL_GetOdbIndependentStepsLayers(IntPtr sOdbDirOrTgzFileNam,
IntPtr sSteps, IntPtr layers);

// 取得 ODB/TGZ 某個 Step 的資料範圍大小
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_GetOdbStepMinMax",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_GetOdbStepMinMax(IntPtr sOdbDirOrTgzFileName, IntPtr
sStepName, ref double stepMinX, ref double stepMinY, ref double stepMaxX, ref double stepMaxY,
TValueUnit valueUnit = TValueUnit.uMM,    bool inStepProfileMinMax = true);

[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_GetOdbStepLayerMinMax",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_GetOdbStepLayerMinMax(IntPtr sOdbDirOrTgzFileName, IntPtr
sStepName, IntPtr sLayerName, ref double stepMinX, ref double stepMinY, ref double stepMaxX, ref
double stepMaxY,TValueUnit valueUnit = TValueUnit.uMM);

// 傳入輸出檔案名稱 Step, Layer 名稱，根據輸出條件，輸出影像檔案 bmp
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_OutputImageFile",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_OutputImageFile(IntPtr sOdbDirOrTgzFileName, IntPtr
sStepName, IntPtr sLyrName, double stepMinX, double stepMinY, double stepMaxX, double stepMaxY,
double outputDPI, IntPtr sOutputFullFileName, ref int imagePixelWidth, ref int imagePixelHeight, byte
```

```
outputBitPerPixel = 8, int threadCount=1);

// 傳入輸出檔案名稱 Step, Layer 名稱，根據輸出條件，輸出模擬真實 PCB 外觀的彩色影像檔案 bmp。請參考圖三。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint =
"DLL_OutputImageFile_SimulateReality", CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_OutputImageFile_SimulateReality(IntPtr
sOdbDirOrTgzFileName, IntPtr sStepName, IntPtr sLyrName, double stepMinX, double stepMinY,
double stepMaxX, double stepMaxY, double outputDPI, IntPtr sOutputFullFileName, ref int
imagePixelWidth, ref int imagePixelHeight, byte outputBitPerPixel = 24);

// 傳入輸出檔案名稱 Step, Layer 名稱，根據輸出條件，輸出模擬真實 PCB 外觀的彩色影像檔案 bmp。請參考圖三。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint =
"DLL_OutputImageFile_SimulateRealityA", CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_OutputImageFile_SimulateRealityA(IntPtr
sOdbDirOrTgzFileName, IntPtr sStepName, TVectSide atSide, double outputDPI, IntPtr
sOutputFullFileName, ref int imagePixelWidth, ref int imagePixelHeight, byte outputBitPerPixel = 24);

// 傳入輸出檔案名稱 Step, Layer 名稱，根據輸出條件，在記憶體繪圖並傳回指標
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_RenderImageInMemory",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_RenderImageInMemory(IntPtr sOdbDirOrTgzFileName, IntPtr
sStepName, IntPtr sLyrName, double stepMinX, double stepMinY, double stepMaxX, double stepMaxY,
double outputDPI, ref IntPtr pImageStart0, ref int imageSizeTotalMB, ref int stride_BytesPerRow,
ref int imagePixelWidth, ref int imagePixelHeight, byte outputBitPerPixel = 8, IntPtr
sSaveToBmpFileName=Default(IntPtr));

// 傳入輸出檔案名稱 Step 名稱，TVectSide，根據輸出條件，在記憶體繪製擬真圖形並傳回指標
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint =
"DLL_RenderImageInMemory_SimulateRealityA", CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_RenderImageInMemory_SimulateRealityA (IntPtr
sOdbDirOrTgzFileName, IntPtr sStepName, TVectSide atSide, double stepMinX, double stepMinY,
double stepMaxX, double stepMaxY, double outputDPI, ref IntPtr pImageStart0, ref int
imageSizeTotalMB, ref int stride_BytesPerRow,
ref int imagePixelWidth, ref int imagePixelHeight, byte outputBitPerPixel = 8, IntPtr
sSaveToBmpFileName =default(IntPtr));

// 對記憶體內圖像做 X 翻轉
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_Image_FlipX",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_Image_FlipX(IntPtr pImageStart0, int stride_BytesPerRow, int
```

```
imagePixelWidth, int imagePixelHeight, byte BitPerPixel);

// 對記憶體內圖像做 Y 翻轉
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_Image_FlipY",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_Image_FlipY(IntPtr pImageStart0, int stride_BytesPerRow, int
imagePixelWidth, int imagePixelHeight, byte BitPerPixel);

// 對記憶體內圖像旋轉任意角度成新的影像
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_Image_Rotate",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_Image_Rotate(IntPtr pImageStart0, int stride_BytesPerRow, int
imagePixelWidth, int imagePixelHeight, byte BitPerPixel, double rotateDegree,
ref IntPtr pRotImageStart0, ref int rotStride_BytesPerRow, ref int rotImagePixelWidth,
ref int rotImagePixelHeight);

// 對記憶體內圖像縮放任意比例成新的影像
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_Image_Scale",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_Image_Scale(IntPtr pImageStart0, int stride_BytesPerRow, int
imagePixelWidth, int imagePixelHeight, byte BitPerPixel, double scale,
ref IntPtr pScaleImageStart0, ref int scaleStride_BytesPerRow, ref int scaleImagePixelWidth,
ref int scaleImagePixelHeight);

//將記憶體中的圖形資料的一列的總 Bytes 數，轉換成 4 Bytes 的倍數，以符合某些編譯器只支援 4Bytes-Align 造成的圖形顯示錯誤。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_Image_Align4Bytes",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_Image_Align4Bytes(IntPtr pImageStart0, int
stride_BytesPerRow, int imagePixelWidth, int imagePixelHeight, byte BitPerPixel,
ref IntPtr pAlignedImageStart0, ref int alignedImagePixelWidth, ref int alignedStride_BytesPerRowt);

// 傳入輸出 CAD 檔案(Gerber274X,NC,DXF,DWG...)名稱 Step, Layer 名稱，根據輸出條件，輸出影像檔案 bmp
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_OutputImageFile_CAD",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_OutputImageFile_CAD(IntPtr sCadFileName, double stepMinX,
double stepMinY, double stepMaxX, double stepMaxY, double outputDPI, IntPtr sOutputFullFileName,
ref int imagePixelWidth, ref int imagePixelHeight, byte outputBitPerPixel = 8, int threadCount=1);

// 傳入輸出檔案(Gerber274X,NC,DXF,DWG...)名稱，根據輸出條件，在記憶體繪圖並傳回指標
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_RenderImageInMemory_CAD",
```

```
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_RenderImageInMemory_CAD(IntPtr sCadFileName, double
stepMinX, double stepMinY, double stepMaxX, double stepMaxY, double outputDPI, ref IntPtr
pImageStart0, ref int imageSizeTotalMB, ref int stride_BytesPerRow,
ref int imagePixelWidth, ref int imagePixelHeight, byte outputBitPerPixel = 8, IntPtr
sSaveToBmpFileName = default(IntPtr) );

// 取得非 ODB/TGZ CAM 檔案的資料範圍大小
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_GetCamFileMinMax",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_GetCamFileMinMax(IntPtr sCamFileName,
ref double aMinX, ref double aMinY, ref double aMaxX, ref double aMaxY,TValueUnit valueUnit =
TValueUnit.uMM);

// 傳入“輸入參數字串”和“輸出參數字串”，完成批次多檔檔案轉換。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_VectImportExport",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_VectImportExport(IntPtr inputArgs, IntPtr outArgs, bool
showForm);

// 清除 RvConverterDLL 用來解壓縮檔案的暫存資料夾
[DllImport(AssemblyName, EntryPoint = "DLL_CleanUnZipBuffer", CallingConvention =
TargetCallingConvention)]
private static extern TReturnCode DLL_CleanUnZipBuffer(IntPtr pUnzipFolder);

//傳入 TGZ 檔案，指定解壓縮路徑，傳回解壓縮後的 ODB++完整目錄名稱。
[DllImport(AssemblyName, EntryPoint = "DLL_UnZipTgzFile", CallingConvention =
TargetCallingConvention)]
private static extern TReturnCode DLL_UnZipTgzFile(IntPtr sTgzFileName , IntPtr pUnzipFolder, IntPtr
pReturnOdbFolder);

// 清除記憶體中的 Cam 資料
[DllImport(AssemblyName, EntryPoint = "DLL_ClearCamData", CallingConvention =
TargetCallingConvention)]
private static extern TReturnCode DLL_ClearCamData();

// 檢查程式是否為和否授權
[DllImport(AssemblyName, EntryPoint = "DLL_IsAuthorized", CallingConvention =
TargetCallingConvention)]
private static extern TReturnCode DLL_IsAuthorized();

// 傳入圖檔名稱，讀入圖形到記憶體繪圖並傳回 圖形指標、大小、格式
```

```
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_LoadImageInMemory",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_LoadImageInMemory(IntPtr sImageFileName,
ref IntPtr pImageStart0, ref int imageSizeTotalMB, ref int stride_BytesPerRow,
ref int imagePixelWidth, ref int imagePixelHeight, ref byte bitsPerPixel );

[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_SaveImageFromMemory",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_SaveImageFromMemory(IntPtr sImageFileName,
IntPtr pImageStart0, int imageSizeTotalMB, int stride_BytesPerRow,
int imagePixelWidth, int imagePixelHeight, byte bitsPerPixel );

// 取得 ODB/TGZ 某個 Parent Step 及其所屬所有 Child Steps 的排版資料
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_GetOdbStepRepeatInfo",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_GetOdbStepRepeatInfo(IntPtr sOdbDirOrTgzFileName, IntPtr
sStepName, IntPtr outputStepRepeatFullFileName, ref double stepMinX, ref double stepMinY, ref
double stepMaxX, ref double stepMaxY,
TValueUnit valueUnit = TValueUnit.uMM,
bool blExportAllStepRepeatedProfiles = false);

//輸入 ODB Parent Step 座標系統上的 Parent XY，傳回位在哪個 Child Step 座標系統內，及 Child Step
座標系統內的 ChildStep XY。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_Inside_OdbParentChildStep",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_Inside_OdbParentChildStep (IntPtr parentStepName, double
parentMmX,parentMmY, ref IntPtr insideStepName,, ref double insideMmX, ref double insideMmY);

// 傳入 輸入 ODB++資料夾或 TGZ 檔名，輸出 CAD 檔案。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_FileConvert_Odb2CAD",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_FileConvert_Odb2CAD(IntPtr sOdbDirOrTgzFileName,
IntPtr sStepName, IntPtr sLayerNames, IntPtr sOutputCadFileName);

// 傳入 輸入 CAD 檔名，輸出 CAD 檔案。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_FileConvert_CAD2CAD",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_FileConvert_CAD2CAD(IntPtr sInputCadFileName,
  IntPtr sOutputCadFileName);

// 傳入資料 mmXY 輸出影像 pixel XY。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_View_Mm2PixelXY",
```

```
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_View_Mm2PixelXY(
    double viewMinX_mm, double viewMinY_mm, double viewMaxX_mm, double viewMaxY_mm,
    Int viewMinX_pixel, Int viewMinY_pixel, Int viewMaxX_pixel, Int viewMaxY_pixel,
    double inputMmX, double inputMmY,
    ref Int outputPixelX, ref Int outputPixelY);

// 傳入 影像 Pixel XY 輸出資料 Mm XY。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_View_Pixel2MmXY",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_View_Pixel2MmXY(
    double viewMinX_mm, double viewMinY_mm, double viewMaxX_mm, double viewMaxY_mm,
    Int viewMinX_pixel, Int viewMinY_pixel, Int viewMaxX_pixel, Int viewMaxY_pixel,
    Int inputPixelX, Int inputPixelY,
    ref double outputMmX, ref double outputMmY);

//只要有變更資料和影像的輸出範圍，執行 DLL_QuickView_XXX() 轉換之前，都必須先執行一次此函式。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_QuickView_UpdateView",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_ QuickView_UpdateView (
    double viewMinX_mm, double viewMinY_mm, double viewMaxX_mm, double viewMaxY_mm,
    Int viewMinX_pixel, Int viewMinY_pixel, Int viewMaxX_pixel, Int viewMaxY_pixel,
    IntPtr pReturnView:PVectView=NULL);

// 高效率轉換函式。傳入資料 mmXY 輸出影像 pixel XY。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_QuickView_Mm2PixelXY",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_QuickView_Mm2PixelXY(
    double inputMmX, double inputMmY,
    ref Int outputPixelX, ref Int outputPixelY,
    IntPtr pAtView=NULL);

//高效率轉換函式。傳入 影像 Pixel XY 輸出資料 Mm XY。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_QuickView_Pixel2MmXY",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_QuickView_Pixel2MmXY(
    Int inputPixelX, Int inputPixelY,
    ref double outputMmX, ref double outputMmY,
    IntPtr pAtView=NULL);

//傳入 oXY，透過排版指令後得到新的排版 aStpRptXY。
```

```
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_GetStepRepeatXY0",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_GetStepRepeatXY0(
    const double oX, const double oY, ref double aStpRptX, ref double aStpRptY,
    double datumnX, double datumnY, double ccwRotDeg, bool mirrorX,
    double shiftX, double shiftY);

//傳入 aStpRptXY，透過排版指令後得到逆排版 oXY。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_GetReverseStepRepeatXY0",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_GetReverseStepRepeatXY0(
    const double aStpRptX, const double aStpRptY, ref double oX, ref double oY,
    double datumnX, double datumnY, double ccwRotDeg, bool mirrorX,
    double shiftX, double shiftY );

//將記憶體中的圖形以全圖繪畫到桌布上。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_Paint_Home",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_Paint_Home(
    IntPtr CnvScan0, int CnvRowBytes, int CnvWidth, int CnvHeight, int CnvBitsPerPixel,
    IntPtr ImgScan0, int ImgRowBytes, int ImgWidth, int ImgHeight, int ImgBitsPerPixel,
    Bool cnvDIBUpWard =true, bool ImgDIBUpWard =true,
    IntPtr pReturnRastView=default(IntPtr) );

//將記憶體中的圖形以設定的縮放範圍繪畫到桌布上。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_Paint_Zoom",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_Paint_Zoom(
    int ImgZoomMinX, int ImgZoomMinY, int ImgZoomMaxX, int ImgZoomMaxY,
    IntPtr CnvScan0, int CnvRowBytes, int CnvWidth, int CnvHeight, int CnvBitsPerPixel,
    IntPtr ImgScan0, int ImgRowBytes, int ImgWidth, int ImgHeight, int ImgBitsPerPixel,
    Bool cnvDIBUpWard =true, bool ImgDIBUpWard =true,
    IntPtr pReturnRastView=default(IntPtr) );

//設定 ODB++ 繪圖時，是否顯示 Child Steps。
[DllImport(AssemblyName, CharSet = TargetCharSet, EntryPoint = "DLL_Set_Visible_ODB_ChildSteps",
CallingConvention = TargetCallingConvention)]
private static extern TReturnCode DLL_ Set_Visible_ODB_ChildSteps (
    Bool childStepVisible);
```

C++

C++ 函式功能與參數的詳細說明，請參考前節 Delphi Pascal 敘述。

資料型態

```
// 函式傳回數值定義
enum TReturnCode
{
   rcFail,   //0
   rcSuccess,      //1;
   rcUnAuthorized,    //2;
   rcFinal    // 3
}

// 座標單位定義
enum TValueUnt
{
   uInch,   // 0
   uMil,      //1
   uCM,      //2
   uPIXEL,   //3
   uUM,      //4
   uNM       //5
}

//座標轉換參數定義
typedef void* PVectView;
```

函式宣告

```
// Callback 函式定義
typedef void (__stdcall *TRunningProgress) (byte *aMainPercent, byte *aSubPercent, const char
*aMainTask, const char *aSubTask);

// Callback 函式指定
typedef void (__stdcall *DLL_AssignCallBackFunc)(TRunningProgress *pCallBackFunc);

// 傳入“輸入參數字串”和“輸出參數字串”，完成批次多檔檔案轉換
typedef TReturnCode (__stdcall *DLL_VectImportExport)(char *sInputParams, char
```

```
*sOutputParams,    bool blShowForm = false);

// 清出 TGZ 檔案解壓縮路徑下的所有資料
typedef TReturnCode (__stdcall *DLL_CleanUnZipBuffer)( char *pUnzipFolder);

//傳入 TGZ 檔案，指定解壓縮路徑，傳回解壓縮後的 ODB++完整目錄名稱。
typedef TReturnCode (__stdcall *DLL_UnZipTgzFile)(char * sTgzFileName,
  char *pUnzipFolder, char *pReturnOdbFolder);

// 取得 ODB/TGZ 某層，經過該層所有鑽孔層的名稱清單
typedef TReturnCode (__stdcall *DLL_GetOdb_RelatedDrillLayers)(char *sOdbDirOrTgzFileName,
char *sAtLayer, char *sRelatedDrillLayers);

// 取得 ODB/TGZ 所有 Steps, Layers 名稱
typedef TReturnCode (__stdcall *DLL_GetOdbStepLayers)(char *sOdbDirOrTgzFileName, char
*sSteps, char *sLayers);

// 取得 ODB/TGZ 所有 Steps, Layers 名稱，參數決定是否只取有繼承關係的母子 Steps
typedef TReturnCode (__stdcall *DLL_GetOdbStepLayersA)(char *sOdbDirOrTgzFileName, char
*sSteps, char *sLayers, bool blGetTopStepAndInheritedChildStepsOnly=false,
char *atTopStepName = NULL);

// 取得 ODB/TGZ 所有非 Child 的最上層 Steps 和 Layers 名稱
typedef TReturnCode (__stdcall *DLL_GetOdbTopStepsLayers)(char *sOdbDirOrTgzFileName, char
*sSteps, char *sLayers);

// 取得 ODB/TGZ 指定 ParentStep 所屬或所有最底層 Steps 和 Layers 名稱
typedef TReturnCode (__stdcall *DLL_GetOdbBottomStepsLayers)(char *sOdbDirOrTgzFileName,
char *sSteps, char *sLayers, char* atTopStepName/*=NULL*/);

// 取得 ODB/TGZ 所有不屬於 Parent 或 Child 的獨立 Steps 和 Layers 名稱
typedef TReturnCode (__stdcall *DLL_GetOdbIndependentStepsLayers)(char
*sOdbDirOrTgzFileName, char *sSteps, char *sLayers);

// 取得 ODB/TGZ 某個 Step 的資料範圍大小
typedef TReturnCode (__stdcall *DLL_GetOdbStepMinMax)(char *sOdbDirOrTgzFileName,    char
*sStepName,    double *stepMinX, double *stepMinY, double *stepMaxX, double *stepMaxY, int
valueUnit=TValueUnt.uMM, bool inStepProfileMinMax = True);

// 取得 ODB/TGZ 某個 Step / Layer 的資料範圍大小
typedef TReturnCode (__stdcall *DLL_GetOdbStepLayerMinMax)(char *sOdbDirOrTgzFileName,
char *sStepName, char *sLayerName, double *stepMinX, double *stepMinY, double *stepMaxX,
```

double *stepMaxY, int valueUnit=TValueUnt.uMM);

// 取得 CAD 檔案(Gerber,DXF,Excellon NC...)資料範圍大小
typedef TReturnCode (__stdcall *DLL_GetCamFileMinMax)(char *sCamFileName, double *aMinX, double *aMinY, double *aMaxX, double *aMaxY, int valueUnit = TValueUnt.uMM);

// 取得 ODB++ 檔案的排版資訊
typedef TReturnCode (__stdcall *DLL_GetOdbStepRepeatInfo)(char *sOdbDirOrTgzFileName, char *sParentStepName, char *outputStepRepeatFullFileName, double *stepMinX, double *stepMinY, double *stepMaxX, double *stepMaxY, int valueUnit=TValueUnit.uMM, bool blExportAllStepRepeatedProfiles = false);

//輸入 ODB Parent Step 座標系統上的 Parent XY，傳回位在哪個 Child Step 座標系統內，及 Child Step 座標系統內的 ChildStep XY。
typedef TReturnCode (__stdcall *DLL_Inside_OdbParentChildStep)(char *parentStepName, double parentMmX, double parentMmY, char *insideStepName, double *insideMmX, double *insideMmY);

// 輸出 BMP 影像檔案
typedef TReturnCode (__stdcall *DLL_OutputImageFile)(char *sOdbDirOrTgzFileName, char *sStepName, char *sLyrName, double stepMinX, double stepMinY, double stepMaxX, double stepMaxY, double outputDPI, char *sOutputFullFileName, int *imagePixelWidth, int *imagePixelHeight, byte outputBitPerPixel = 8, int threadCount=1);

// 傳入輸出檔案名稱 Step, Layer 名稱，根據輸出條件，輸出模擬真實 PCB 外觀的彩色影像檔案 bmp。請參考圖三。
typedef TReturnCode (__stdcall *DLL_OutputImageFile_SimulateReality)(char *sOdbDirOrTgzFileName, char *sStepName, char *sLyrName, double stepMinX, double stepMinY, double stepMaxX, double stepMaxY, double outputDPI, char *sOutputFullFileName, int *imagePixelWidth, int *imagePixelHeight, byte outputBitPerPixel = 8);

// 傳入輸出檔案名稱 Step, Layer 名稱，根據輸出條件，輸出模擬真實 PCB 外觀的彩色影像檔案 bmp。請參考圖三。
typedef TReturnCode (__stdcall *DLL_OutputImageFile_SimulateRealityA)(char *sOdbDirOrTgzFileName, char *sStepName, TVectSide atSide, double outputDPI, char *sOutputFullFileName, int *imagePixelWidth, int *imagePixelHeight, byte outputBitPerPixel = 24);

// 傳入輸出檔案名稱 Step, Layer 名稱，根據輸出條件，在記憶體繪圖並傳回指標
typedef TReturnCode (__stdcall *DLL_RenderImageInMemory)(char *sOdbDirOrTgzFileName, char *sStepName, char *sLyrName, double stepMinX, double stepMinY, double stepMaxX, double stepMaxY, double outputDPI, void *pImageStart0, int *imageSizeTotalMB, int

*stride_BytesPerRow, int *imagePixelWidth, int *imagePixelHeight, byte outputBitPerPixel = 8, char* sSaveToBmpFileName=NULL);

// 傳入輸出檔案名稱 Step 名稱，Side，根據輸出條件，在記憶體繪製擬真圖並傳回指標
typedef TReturnCode (__stdcall ***DLL_RenderImageInMemory_SimulateRealityA**)(char *sOdbDirOrTgzFileName, char *sStepName, TVectSide atSide, double stepMinX, double stepMinY, double stepMaxX, double stepMaxY, double outputDPI, void *pImageStart0, int *imageSizeTotalMB, int *stride_BytesPerRow, int *imagePixelWidth, int *imagePixelHeight, byte outputBitPerPixel = 8, char* sSaveToBmpFileName=NULL);

// 傳入輸出 CAD 檔案(Gerber274X,NC,DXF,DWG…)名稱 Step, Layer 名稱，根據輸出條件，輸出影像檔案 bmp
typedef TReturnCode (__stdcall ***DLL_OutputImageFile_CAD**)(char *sCadFileName, double stepMinX, double stepMinY, double stepMaxX, double stepMaxY, double outputDPI, char *sOutputFullFileName, int *imagePixelWidth, int *imagePixelHeight, byte outputBitPerPixel = 8, int threadCount=1);

// 傳入輸出檔案(Gerber274X,NC,DXF,DWG…)名稱，根據輸出條件，在記憶體繪圖並傳回指標
typedef TReturnCode (__stdcall ***DLL_RenderImageInMemory_CAD**)(char *sCadFileName, double stepMinX, double stepMinY, double stepMaxX, double stepMaxY, double outputDPI, void *pImageStart0, int *imageSizeTotalMB, int *stride_BytesPerRow, int *imagePixelWidth, int *imagePixelHeight, byte outputBitPerPixel = 8; char* sSaveToBmpFileName=NUL);

// 對記憶體中的圖形做 X 翻轉
typedef TReturnCode (__stdcall ***DLL_Image_FlipX**)(void *pImageStart0, int stride_BytesPerRow, int imagePixelWidth, int imagePixelHeight, byte BitPerPixel);

// 對記憶體中的圖形做 Y 翻轉
typedef TReturnCode (__stdcall ***DLL_Image_FlipY**)(void *pImageStart0, int stride_BytesPerRow, int imagePixelWidth, int imagePixelHeight, byte BitPerPixel);

// 對記憶體中的圖形旋轉任意角度成新圖像
typedef TReturnCode (__stdcall ***DLL_Image_Rotate**)(void *pImageStart0, int stride_BytesPerRow, int imagePixelWidth, int imagePixelHeight, byte BitPerPixel, double rotateDegree, void *pRotImageStart0, int* rotStride_BytesPerRow, int* rotImagePixelWidth, int* rotImagePixelHeight);

// 對記憶體中的圖形縮放任意比例成新圖像
typedef TReturnCode (__stdcall ***DLL_Image_Scale**)(void *pImageStart0, int stride_BytesPerRow, int imagePixelWidth, int imagePixelHeight, byte BitPerPixel, double scale, void *pScaleImageStart0, int* scaleStride_BytesPerRow, int* scaleImagePixelWidth, int* scaleImagePixelHeight);

```
//將記憶體中的圖形資料的一列的總 Bytes 數，轉換成 4 Bytes 的倍數，以符合某些編譯器只支援 4Bytes-Align 造成的圖形顯示錯誤。
typedef TReturnCode (__stdcall *DLL_Image_Align4Bytes)(void *pImageStart0, int stride_BytesPerRow, int imagePixelWidth, int imagePixelHeight, byte BitPerPixel, void *pAlignedImageStart0, int* alignedImagePixelWidth, int* alignedStride_BytesPerRow);

// 檢查程式是否為和否授權
typedef TReturnCode (__stdcall *DLL_IsAuthorized)();

// 清除記憶體中的 Cam 資料
typedef TReturnCode (__stdcall *DLL_ClearCamData)();

// 傳入圖檔名稱，讀入圖形到記憶體繪圖並傳回 圖形指標、大小、格式
typedef TReturnCode (__stdcall *DLL_LoadImageInMemory)(char *sImageName, void*pImageStart0, int * imageSizeTotalMB, int *stride_BytesPerRow, int *imagePixelWidth, int *imagePixelHeight, byte *outputBitPerPixel);

// 將記憶體內的圖形，存出到圖檔
typedef TReturnCode (__stdcall *DLL_SaveImageFromMemory)(char *sImageName, const void *pImageStart0, const int imageSizeTotalMB, const int stride_BytesPerRow, const int imagePixelWidth, const int imagePixelHeight, const byte outputBitPerPixel);

// 傳入 輸入 ODB++資料夾或 TGZ 檔名，輸出 CAD 檔案。
typedef TReturnCode (__stdcall *DLL_FileConvert_Odb2CAD)(char *sOdbDirOrTgzFileName, char *sStepName, char *sLayerNames, char *sOutputCadFileName);

// 傳入 輸入 CAD 檔名，輸出 CAD 檔案。
typedef TReturnCode (__stdcall *DLL_FileConvert_CAD2CAD)(char *sInputCadFileName, char *sOutputCadFileName);

// 傳入資料和影像範圍，做 mm XY -> pixel XY 轉換。
typedef TReturnCode (__stdcall *DLL_View_Mm2PixelXY)(
    double viewMinX_mm, double viewMinY_mm, double viewMaxX_mm, double viewMaxY_mm,
    int viewMinX_pixel, int viewMinY_pixel, int viewMaxX_pixel, int viewMaxY_pixel,
    double inputMmX, double inputMmY,
    int *outputPixelX, int *outputPixelY );

// 傳入資料和影像範圍，做 Pixel XY -> Mm XY 轉換。
typedef TReturnCode (__stdcall *DLL_View_Pixel2MmXY)(
    double viewMinX_mm, double viewMinY_mm, double viewMaxX_mm, double viewMaxY_mm,
    int viewMinX_pixel, int viewMinY_pixel, int viewMaxX_pixel, int viewMaxY_pixel,
```

```
    int inputPixelX, int inputPixelY,
    double *outputMmX, double *outputMmY );

  //只要有變更資料和影像的輸出範圍，執行 DLL_QuickView_XXX() 轉換之前，都必須先執行一次此函式。
  typedef TReturnCode (__stdcall *DLL_QuickView_UpdateView)(
    double viewMinX_mm, double viewMinY_mm, double viewMaxX_mm, double viewMaxY_mm,
    int viewMinX_pixel, int viewMinY_pixel, int viewMaxX_pixel, int viewMaxY_pixel,
    void* pAtView=NULL );

  // 大量使用轉換函式，可使用 DLL_QuickView_XXX，做 mm XY -> pixel XY 轉換。
  typedef TReturnCode (__stdcall *DLL_QuickView_Mm2PixelXY)(
    double inputMmX, double inputMmY,
    int *outputPixelX, int *outputPixelYl,
    void* pAtView=NULL );

  //大量使用轉換函式，可使用 DLL_QuickView_XXX，做 Pixel XY -> Mm XY 轉換。
  typedef TReturnCode (__stdcall *DLL_ QuickView_Pixel2MmXY)(
    int inputPixelX, int inputPixelY,
    double *outputMmX, double *outputMmYl,
    void* pAtView=NULL );

  //傳入 oXY，透過排版指令後得到新的排版 aStpRptXY。
  typedef TReturnCode (__stdcall *DLL_ GetStepRepeatXY0)(
    const double oX, const double oY, double* aStpRptX, double* aStpRptY,
    double datumnX, double datumnY, double ccwRotDeg, bool mirrorX,
    double shiftX, double shiftY );

  //傳入 aStpRptXY，透過排版指令後得到逆排版 oXY。
  typedef TReturnCode (__stdcall *DLL_ GetReverseStepRepeatXY0)(
    const double aStpRptX, const double aStpRptY, double* oX, double* oY,
    double datumnX, double datumnY, double ccwRotDeg, bool mirrorX,
    double shiftX, double shiftY );

//將記憶體中的圖形以全圖繪畫到桌布上。
typedef TReturnCode (__stdcall *DLL_Paint_Home)(
    void*CnvScan0, int CnvRowBytes, int CnvWidth, int CnvHeight, int CnvBitsPerPixel,
    void* ImgScan0, int ImgRowBytes, int ImgWidth, int ImgHeight, int ImgBitsPerPixel,
    bool cnvDIBUpWard =true, bool ImgDIBUpWard =true,
    void* pReturnRastView=NULL);

//將記憶體中的圖形以設定的縮放範圍繪畫到桌布上。
```

```
typedef TReturnCode (__stdcall *DLL_Paint_Zoom)(
        int ImgZoomMinX, int ImgZoomMinY, int ImgZoomMaxX, int ImgZoomMaxY,
        void*CnvScan0, int CnvRowBytes, int CnvWidth, int CnvHeight, int CnvBitsPerPixel,
        void*ImgScan0, int ImgRowBytes, int ImgWidth, int ImgHeight, int ImgBitsPerPixel,
        bool cnvDIBUpWard =true, bool ImgDIBUpWard =true,
        void*pReturnRastView=NULL );

//設定 ODB 繪圖時是否顯示  Child Stepss。
typedef TReturnCode (__stdcall *DLL_Set_Visible_ODB_ChildSteps)(
        bool childStepVisible );
```

CPP

```
    void OnRunningProgress(byte *aMainPercent, byte *aSubPercent, const char *aMainTask, const
char *aSubTask);
    {
    }

    const char *cDLL_FileName =
    "D:\\TestRvConverterDLL\\ReleaseTestRvConverterDLL\\RvConverterDLL.dll";

    DLL_AssignCallBackFunc _DLL_AssignCallBackFunc;
    DLL_VectImportExport _DLL_VectImportExport;
    DLL_GetOdbStepLayers _DLL_GetOdbStepLayers;
    DLL_GetOdbStepMinMax _DLL_GetOdbStepMinMax;
    DLL_RenderImageInMemory_DLL_RenderImageInMemory;


    HINSTANCE hInstLibrary = LoadLibrary(TEXT(cDLL_FileName));
    // LoadLibrary("L""RvConverterDLL.dll"); //

    if (hInstLibrary)
    {
        _DLL_AssignCallBackFunc = (DLL_AssignCallBackFunc)GetProcAddress(hInstLibrary,
    "DLL_AssignCallBackFunc");
        _DLL_VectImportExport = (DLL_VectImportExport)GetProcAddress(hInstLibrary,
    "DLL_VectImportExport");
        _DLL_GetOdbStepLayers = (DLL_GetOdbStepLayers)GetProcAddress(hInstLibrary,
    "DLL_GetOdbStepLayers");
        _DLL_GetOdbStepMinMax = (DLL_GetOdbStepMinMax)GetProcAddress(hInstLibrary,
    "DLL_GetOdbStepMinMax");
        _DLL_RenderImageInMemory = (DLL_RenderImageInMemory)GetProcAddress(hInstLibrary,
```

```
"DLL_RenderImageInMemory");


   if (_DLL_AssignCallBackFunc)
   {
   _ DLL_AssignCallBackFunc( &OnRunningProgress );
   }

   if (_DLL_VectImportExport)
   {
         If (TReturnCode.rcSuccess    ==
_DLL_VectImportExport("D:\\ODB\\3239DV18.TGZ@edit@ep,to,ts",
                     "C:\\RvOut\\*.bmp@b1@rdpi600", false))
         //@b1 -> 1bit,      @r0.015 -> 15 um/pixel
         {
               MessageBoxA(NULL,"Sucess","Information", MB_OK);
         } // success
         else
         {
              MessageBoxA(NULL, "Fail", "Information", MB_OK);
         }
   }

   double fMinX = 0;
   double fMinY = 0;
   double fMaxX = 0;
   double fMaxY = 0;

   if (TReturnCode.rcSccess==_DLL_GetOdbStepMinMax(sTgzFile.GetBuffer(0), “panel”,
&fMinX,&fMinY,&fMaxX,&fMaxY, TValueUnit.uMM))
   else return;

   void* pBuffer = NULL;
   int iBufferSize = 0;
   int iRowBytes = 0;
   int iWidth = 0;
   int iHeight = 0;
   double outputDPI = 100.0f;
   byte iBitPerPixel = 8;

   if (TReturnCode.rcSuccess==_DLL_RenderImageInMemory(a_sGerberFile.GetBuffer(0), "pcb",
"drill", fMinX, fMinY, fMaxX, fMaxY, outputDPI,
```

```
        &pBuffer, &iBufferSize, &iRowBytes, &iWidth, &iHeight, iBitPerPixel) ;
}
```

下載與教學

RvConvert 程式與使用範例下載

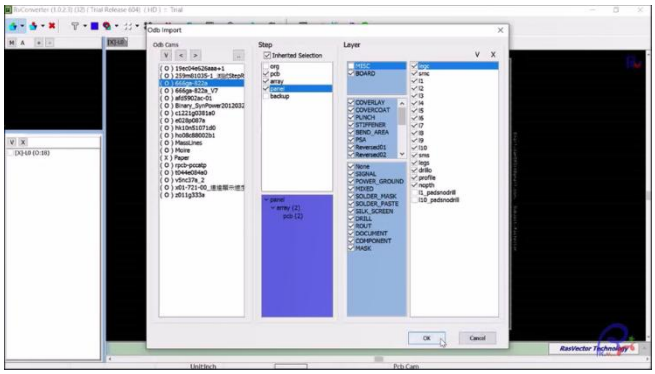
www.rasvector.url.tw/Download Pages/Download_Engineering.htm

教學影片

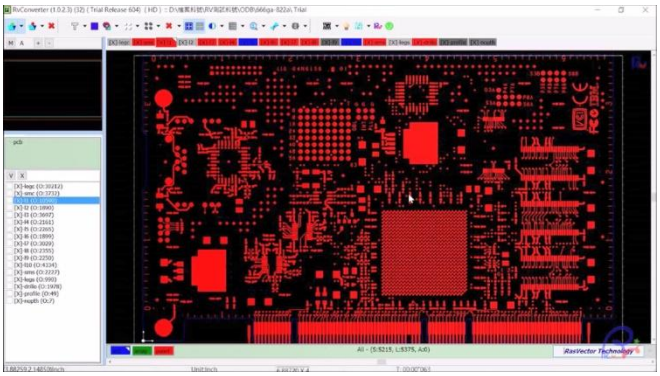
RvConverter DLL 使用示範 - YouTube
https://goo.gl/f9AWia

RvConverter 圖形顯示和編輯

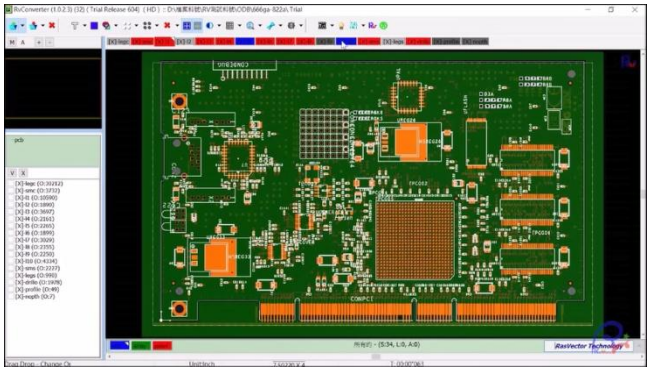
圖一：讀入 ODB++檔案



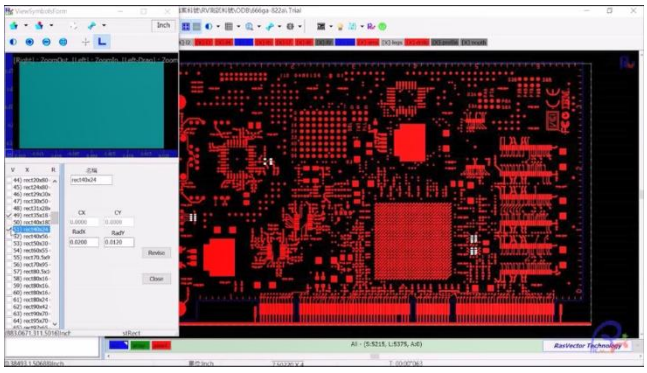
圖二：圖形顯示



圖三：擬真繪圖



圖四：物件屬性修改

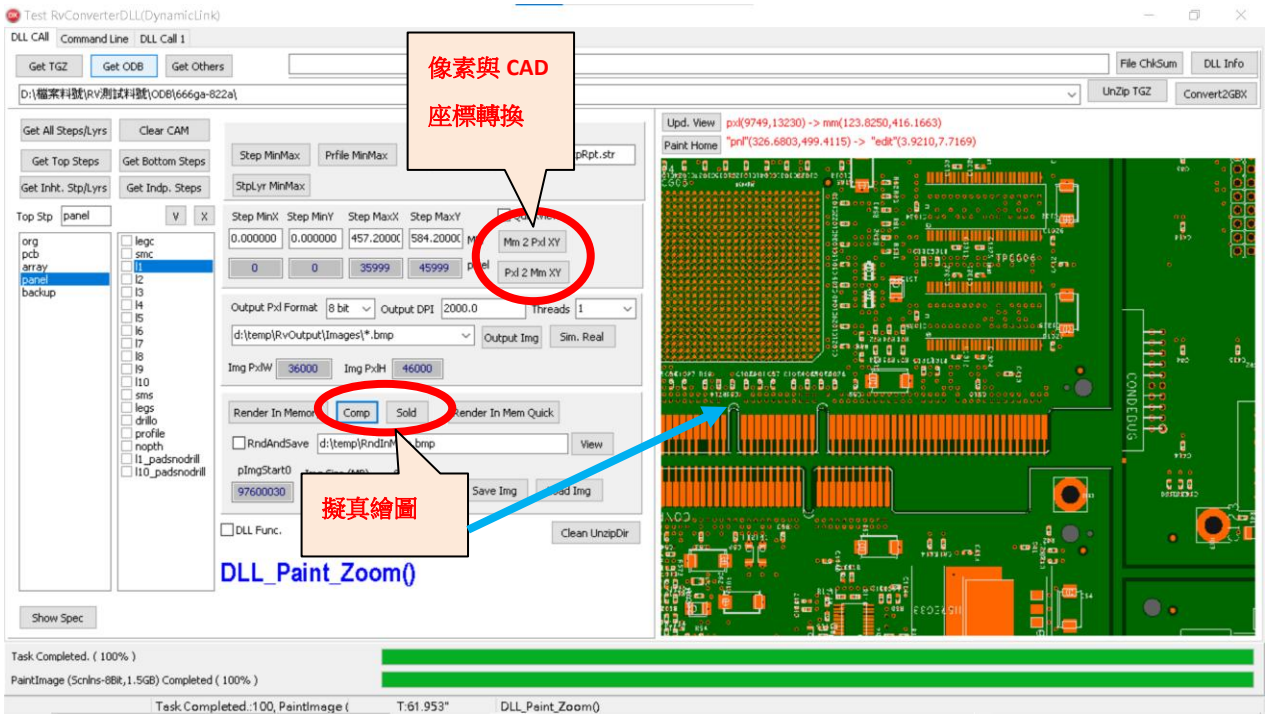


DLL 測試程式介面說明

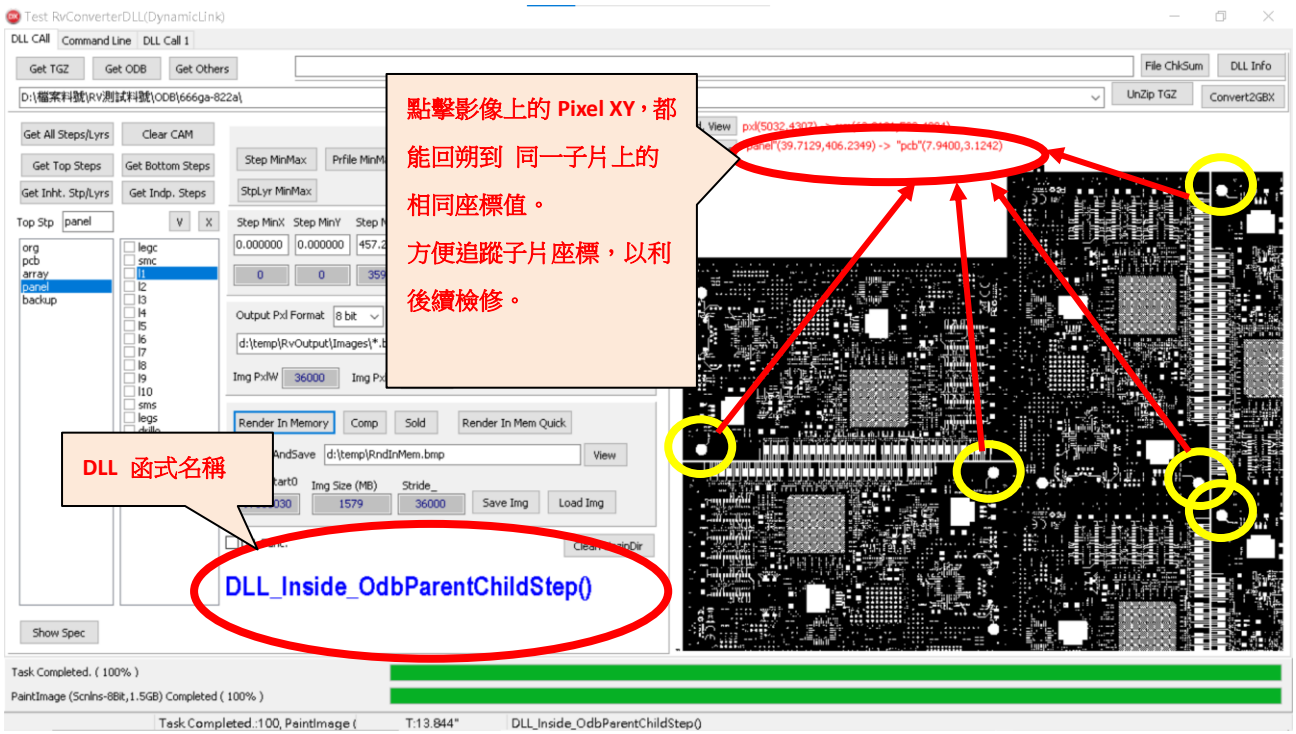
繪製到記憶體 或 圖檔(BMP)

擬真繪圖

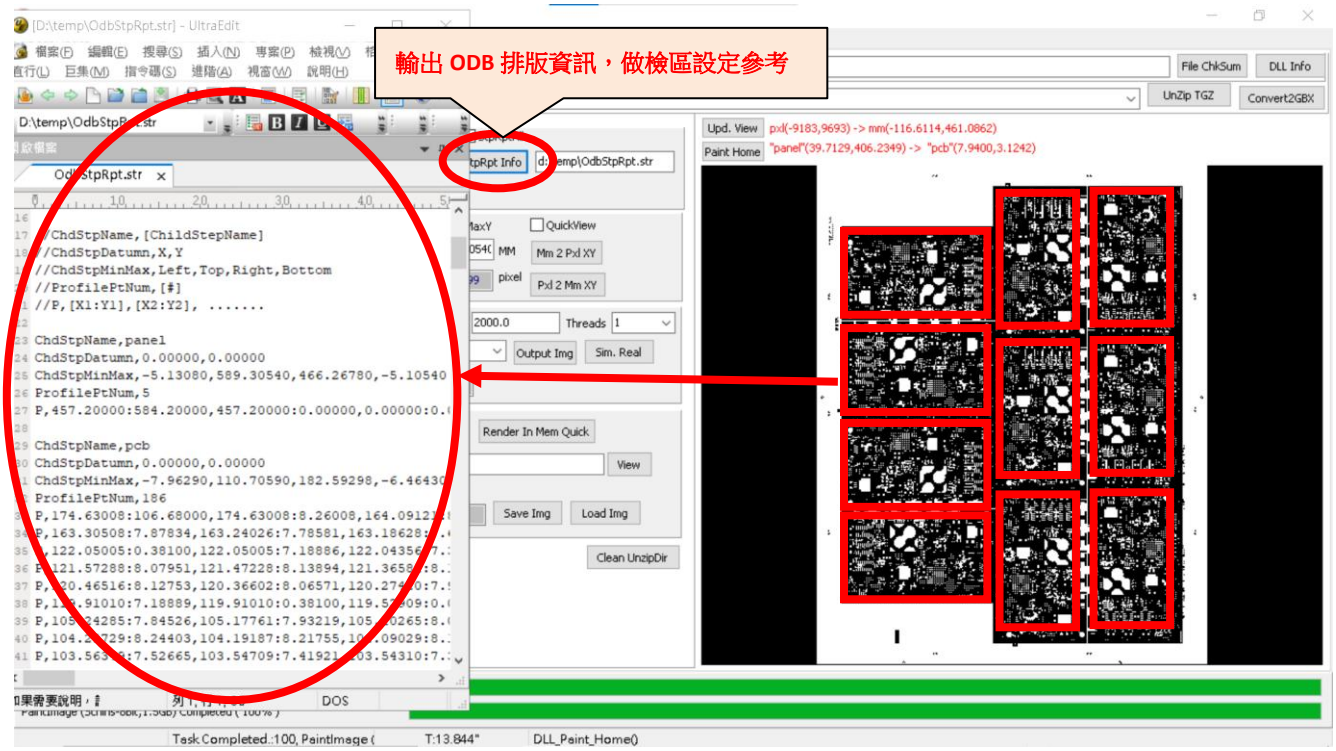
像素(pixel) 與 CAD(mm)座標轉換



ODB++ CAD 座標轉換與回朔

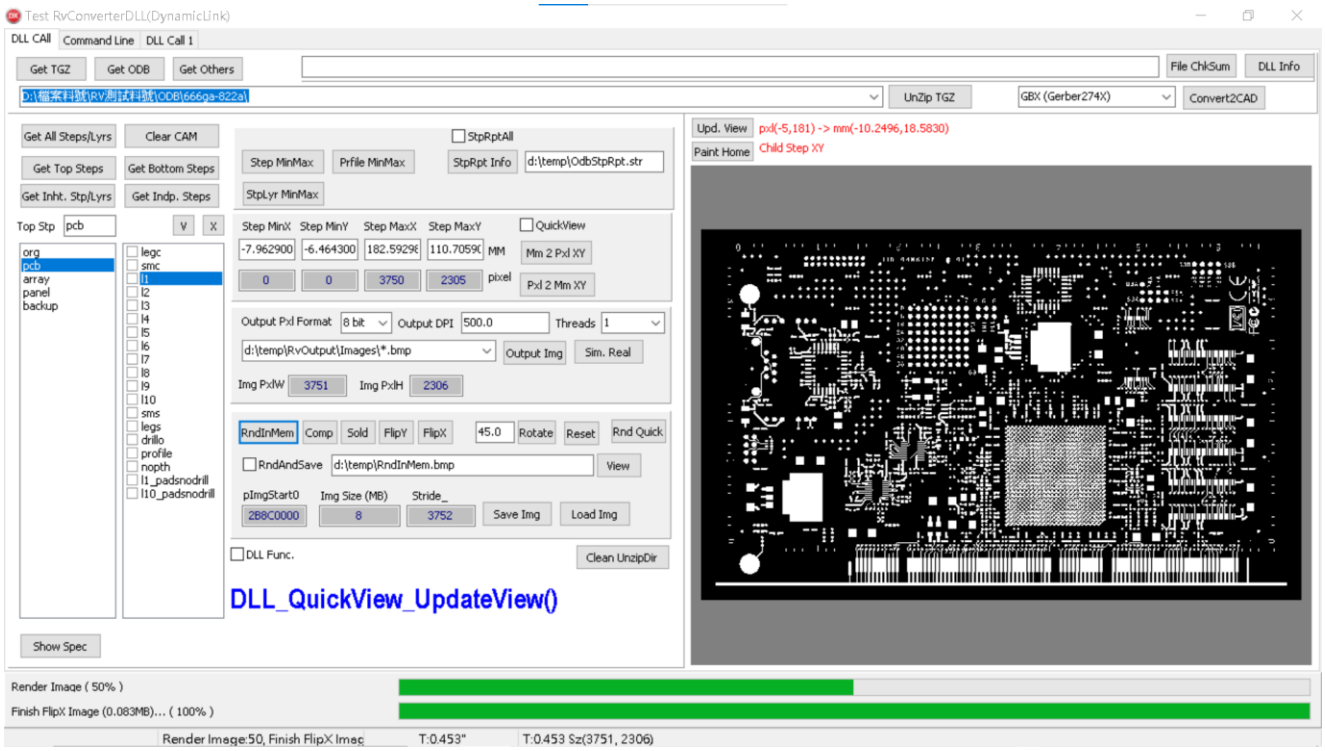


輸出排版資訊

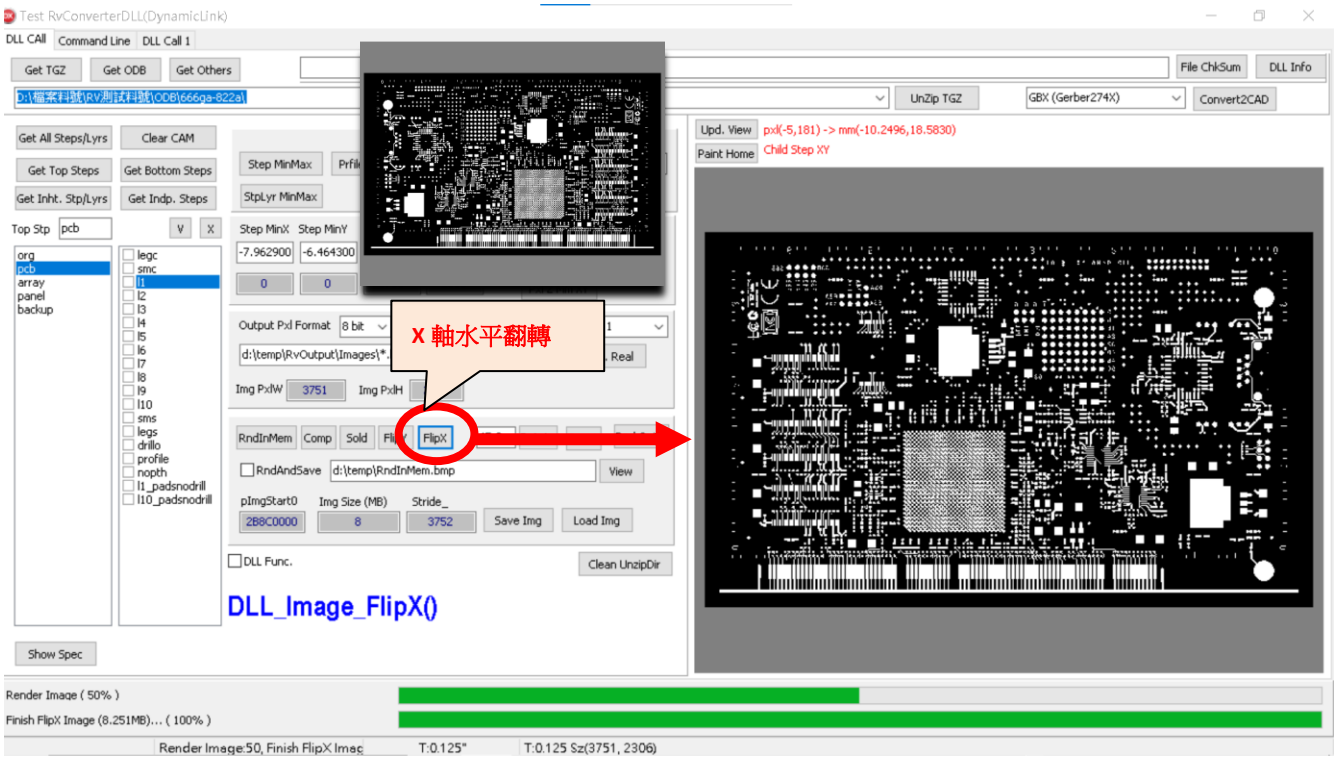


影像 FlipX, FlipY 和旋轉

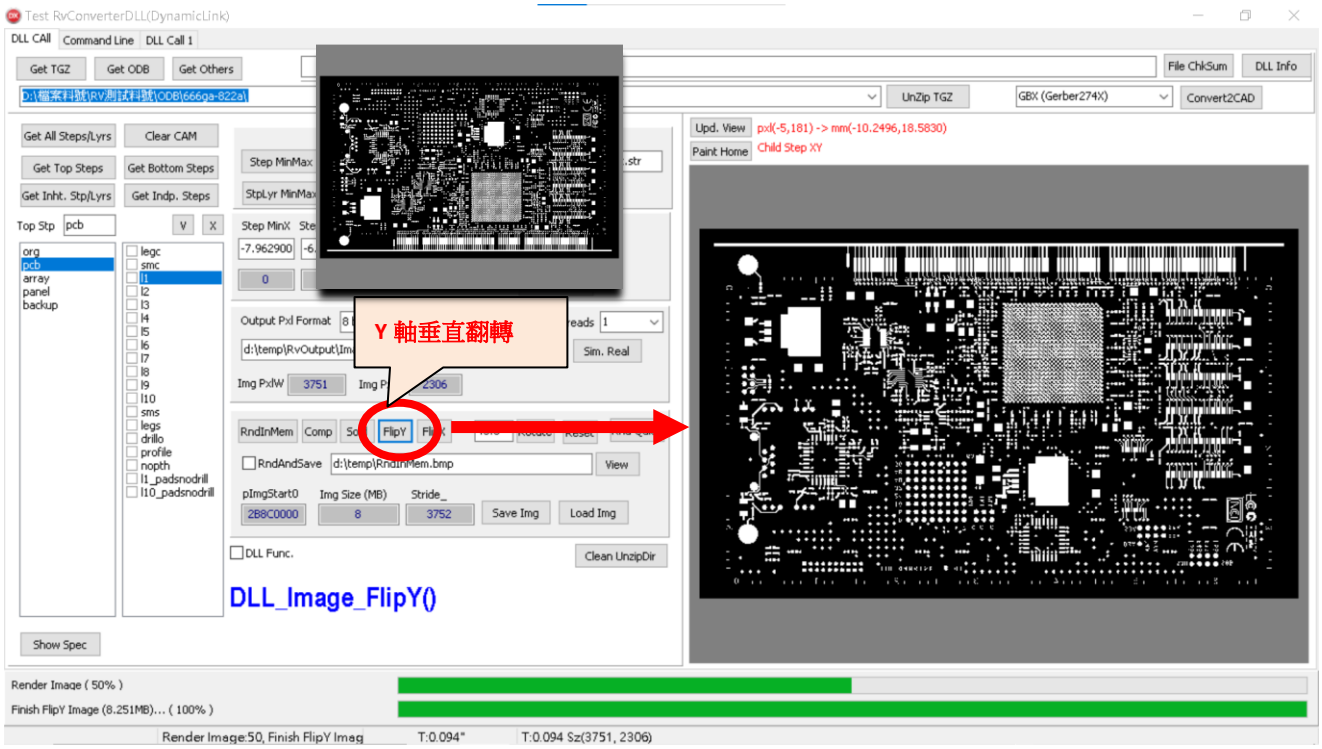
原始影像



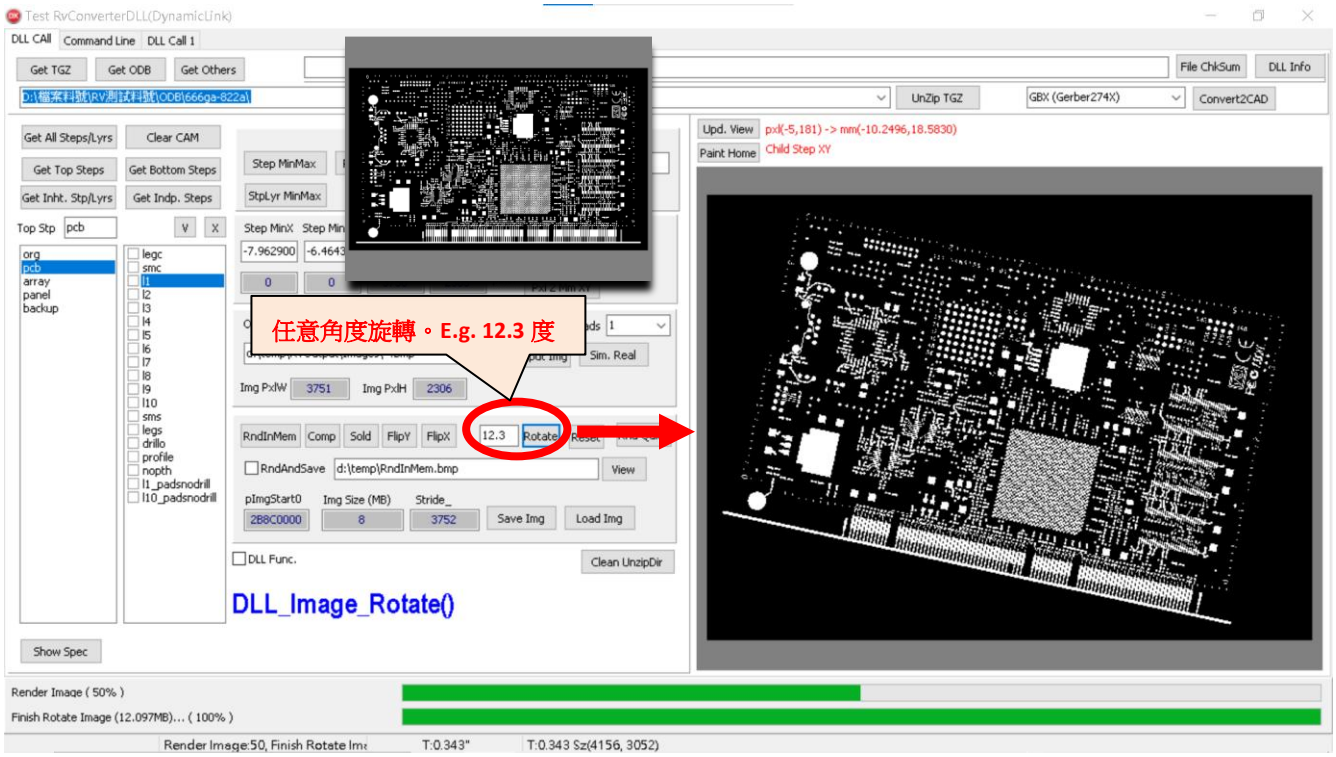
影像 FlipX



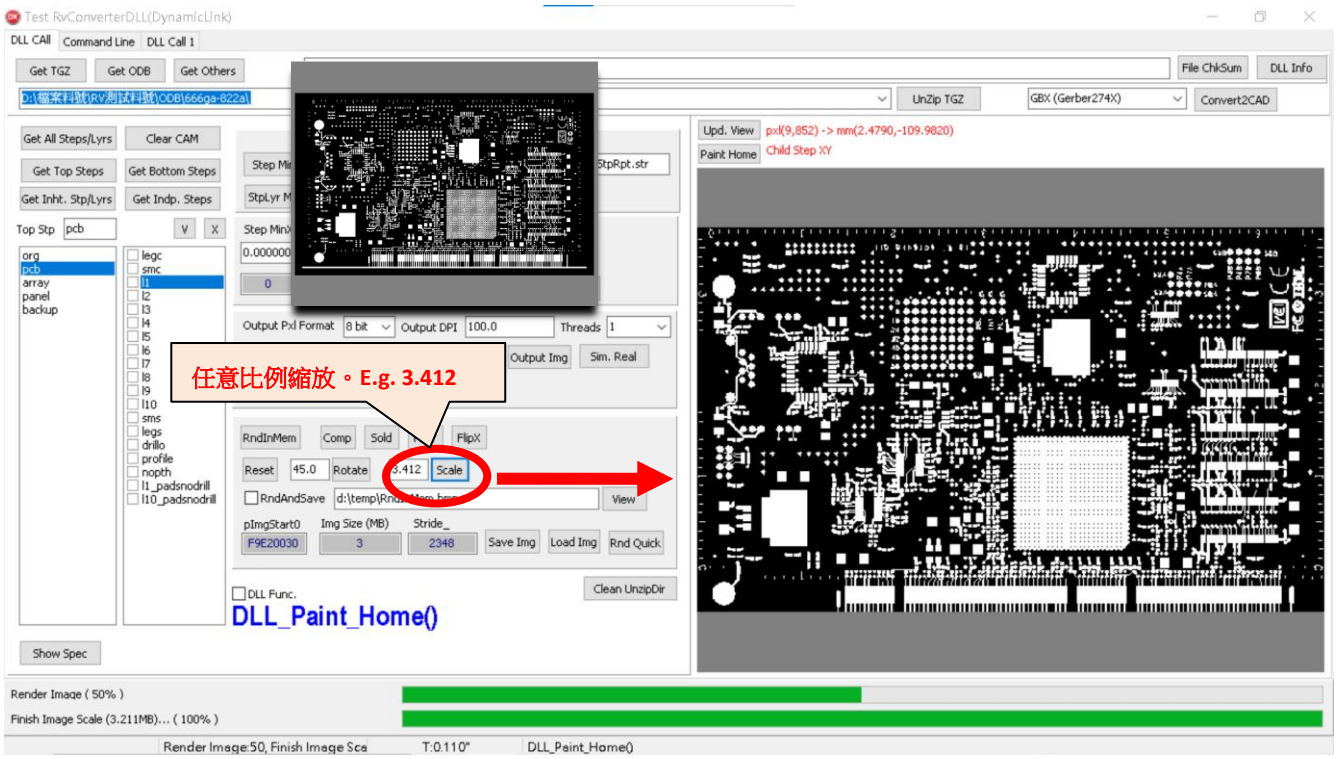
影像 FlipY



影像旋轉



影像縮放


Test RvConverterDLL(DynamicLink)
DLL Call
Command Line
DLL Call 1
Get TGZ
Get ODB
Get Others
File ChkSum
DLL Info
UnZip TGZ
GBX (Gerber274X)
Convert2CAD
Get All Steps/Lyrs
Clear CAM
Get Top Steps
Get Bottom Steps
Get Inht. Stp/Lyrs
Get Indp. Steps
Top Stp
pcb
org
pcb
array
panel
backup
Upd. View
pxl(9,852) -> mm(2.4790,-109.9820)
Paint Home
Child Step XY
Output Pxl Format
8 bit
Output DPI
100.0
Threads
Output Img
Sim. Real
任意比例縮放。E.g. 3.412
RndInMem
Comp
Sold
FlipX
Reset
45.0
Rotate
3.412
Scale
RndAndSave
View
pImgStart0
Img Size (MB)
Stride_
F9E20030
3
2348
Save Img
Load Img
Rnd Quick
Clean UnzipDir
DLL Func.
DLL_Paint_Home()
Show Spec
Render Image (50%)
Finish Image Scale (3.211MB)... (100%)
Render Image:50, Finish Image Sca
T:0.110"
DLL_Paint_Home()

Q&A

DLL_RenderImageInMemory()

Q :

為甚麼 DLL_RenderImageInMemory() 最後一個參數設定儲存檔案時，和未設定儲存檔案時，回傳的 strideBytes_PerRow 會不一樣?

A:

儲存檔案時，為了相容某些不支援 4-Bytes 對齊的讀圖軟體，會以 4-Bytes 對齊的格式輸出。但是記憶體內的影像，為了和實際大小 Mm 符合，不能擅自增加 strideBytes_PerRow 否則會造成解析度不匹配。解析度=RealSize(Mm)/ImageWidth(pixel)

Q :

用 DLL_RenderImageInMemory() 回傳的影像，使用編譯器顯示圖形時，為甚麼發生顯示扭曲，圖形錯誤的情況。

A:

DLL_RenderImageInMemory() 回傳的影像，為了和實際大小 Mm 符合不一定是 4-Bytes 對齊的格式。但是有些編譯器只支援 4-Bytes 對齊，因此，使用者可以使用 **DLL_Image_Align4Bytes()** 轉換資料，或者自行整理資料，將 DLL_RenderImageInMemory() 回傳的影像資料變成 4-Bytes 對齊的資料。範例如下:

C# 範例:

```
public static Sys_Draw.Bitmap IntPtr_To_Bitmap(IntPtr image, int width, int height, int
stride_BytesPerRow, System.Drawing.Imaging.PixelFormat pixelFormat)
 {
     byte[] Ori_Image_Array = new byte[stride_BytesPerRow * height];
     Marshal.Copy(image, Ori_Image_Array, 0, stride_BytesPerRow * height);

     // 4 byte Align
     int Align_stride_BytesPerRow = (stride_BytesPerRow % 4 == 0) ? stride_BytesPerRow :
((stride_BytesPerRow / 4 + 1) * 4);
     byte[] Align_Image_Array = new byte[height * Align_stride_BytesPerRow];

     for (int i = 0; i < height; i++)
     {
         for (int k = 0; k < Align_stride_BytesPerRow; k++)
         {
            if (k < stride_BytesPerRow)
            {
                Align_Image_Array[Align_stride_BytesPerRow * i + k] =
```

```
    Ori_Image_Array[stride_BytesPerRow * i + k];
                    }
                    else
                {
                    Align_Image_Array[Align_stride_BytesPerRow * i + k] = 0;
                }
            }
        }

        Sys_Draw.Bitmap bmp;
        // Get bmp
        bmp = new Sys_Draw.Bitmap(width, height, pixelFormat);
        Sys_Draw_Img.BitmapData bmpData = bmp.LockBits(new Sys_Draw.Rectangle(0, 0, width, height),
Sys_Draw_Img.ImageLockMode.ReadWrite, bmp.PixelFormat);
        bmpData.Stride = Align_stride_BytesPerRow;
        IntPtr ptr = bmpData.Scan0;
        Marshal.Copy(Align_Image_Array, 0, ptr, Align_Image_Array.Length);
        bmp.UnlockBits(bmpData);
        bmp.RotateFlip(Sys_Draw.RotateFlipType.RotateNoneFlipY);
        return bmp;
        }
```

C++ 範例:

```
    stride_BytesPerRow = (stride_BytesPerRow%4==0)?stride_BytesPerRow:((stride_BytesPerRow/4+1)*4);
          byte* tempbuffer = new byte[imagePixelHeight*stride_BytesPerRow];

          for(int i =0;i <imagePixelHeight;i++)
          {
                memcpy(tempbuffer+i*stride_BytesPerRow, pBuffer
    +(imagePixelHeight-i-1)*imagePixelWidth,(size_t)imagePixelWidth);
          }

          QImage image(tempbuffer,    stride_BytesPerRow, imagePixelHeight, stride_BytesPerRow,
    QImage::Format_Grayscale8);
```

DLL_GetOdbStepLayers

Q :

讀取 TGZ 檔案時，有時候成功，有時候失敗，不太穩定。

A :

TGZ 檔案 是 ODB++資料的壓縮檔案，RvConverter 在讀取 ODB++資料之前會先將 TGZ 檔案解壓縮到 unzip 資料夾下。而預設的 unzip 資料夾可能因為權限問題或編譯階段編譯器會清除緩存的緣故將之刪除。如果有此情形發生，請利用 DLL_UnZipTgzFile() 先將 TGZ 解壓縮到自己指定的目錄下，然後再讀取解壓後的 ODB++ 資料。