TurboDXF

TurboDXF is a Windows Dynamic Link Library for creating industry standard DXF files from any Windows development tool. The DXF files created by your application and TurboDXF can then be imported into many popular Windows applications such as Corel Draw, Micrografx Designer and Word for Windows. In addition the DXF files are fully compatible with all CAD applications such as AutoCAD.

Features

1) Dynamic link library callable from many compilers and development tools

2) Includes routines for drawing lines, arcs, circles, filled blocks, polylines, text, points

3) Includes routines for creating multiple layers

Supported Applications

TurboDXF is the ideal addition to Visual Basic, Microsoft Excel, ObjectVision, Turbo Pascal for Windows and any Windows C compiler:

1) Create DXF graphics in Visual Basic and export them to other applications, like Corel Draw, for touch up and printing.

2) Create an ObjectVision application to view a record in a data base, click a button and have that record added to the DXF file. When all the records needed in the graphic have been added, lose the DXF file. Open Corel Draw (or another supported application) to view the graphic. TurboDXF includes simplified charting commands.

3) From C and C++ create applications that generate maps from data bases. Viewing results of computer simulations graphically is much more productive than pouring over pages of printed output.

All DXF files created by TurboDXF can be viewed in AutoCAD, MicroStation and most otherCAD applications capable of importing DXF files.

Demonstration Version

The DLL included with this demonstration package is limited to creating 25 entities per drawing. The full version of the DLL will create an unlimited number of elements and layers. The sample DXF files were created with the

full version of the DLL. When you run the demonstration applications you will see only part of the sample DXF files.

TurboDXF
The TurboDXF DLL is written completely in Turbo Pascal for Windows. A TurboDXF TPU (for DOS) is also available for Turbo Pascal v.6 and earlier . In addition the DOS version includes full AutoCAD block and attribute support.

Pricing
Both TurboDXF libraries (DOS and Windows) are priced at $50US ($60CDN) each. Both libraries can be ordered together for $80US ($90CDN). The libraries each include a printed manual and a disk with the library, sample programs and sample output. Include $5 for shipping and handling.

Ordering
Turbo DXF is available from Ideal Engineering Software.
Ideal Engineering Software
#105 1280 Fir Street
White Rock, British Columbia
Canada
V4B 4B1

As mentioned the cost is US$50 (CDN$65) for the Windows version of TurboDXF. You will receive a full version of the TurboDXF DLL with which you can create an unlimited number of entities. In addition you will receive a printed manual and runtime versions of the Visual Basic and ObjectVision applications.

Sample Applications
I have included five sample applications with this demo. The sample applications included with this demonstration version of the TurboDXF DLL are intended to highlight probable Windows applications as opposed to CAD applications. Most Windows applications do not support layers, polylines, three dimensional drawings, blocks and attributes. This DLL includes full support of these DXF commands so you can easily create DXF files for AutoCAD use.

To view the files that are produced by these applications you will need an application that supports DXF imports. The software has been tested with the following applications:

1) AutoCAD (DOS)

2) Drawing Librarian (DOS)
3) Drawing Librarian (Windows version)
4) Corel Draw (Windows)
5) Micrografx Designer (Windows)
6) Microsoft Word for Windows 2.0 (Windows)

The Sample Applications

1. A **Microsoft Visual Basic** program that creates the demo DXF file on disk. This application includes the VBDEMO.BAS file with the SUB declarations. When running the applications simply click the "Create DXF file" button to make the sample DXF file. As noted above only part of the DXF file will be created due to the 25 entity limit in the demonstration DLL. Look at the file VBDEMO.DXF for the full output from this program. In addition to the source files, a compiled EXE file is and the Visual Basci runtime included.

2. A **Turbo Pascal for Windows** program that creates the same sample DXF file. Compile the TPDEMO1.PAS file with Turbo Pascal for Windows. Run the application and then double click to close the window. Again the full sample file will not be created due to the 25 entity limit in the demonstration DLL. Check the file TPDEMO1.DXF for the full results of this run (it is similar to the drawing created by the Visual Basic example). This application is included in source (.PAS) and compiled form. No extra software (other than Windows) is required to run this application.

3. A **Turbo Pascal for Windows** program that creates a bar chart. Enter 10 "Y" values as prompted. The sample program TPDEMO2.PAS will create a scaled bar chart with your values.

4. A **Borland ObjectVision** (OV) application (OVDEMO1.OVD) that creates a sample DXF file similar to the previous two applications. Simply click on the button to create the file. You will require either the runtime or full version of ObjectVision for this demo.
5. A **Borland OV** application (OVDEMO2.OVD) connected to a Paradox data base to create a drawing of lines and text. There are about 20 records in the data base each consisting of two X,Y pairs (the end points of the line) and a text string. You can change or add to the little data base using the OV application. There are also "Open DXF", "Add to DXF" and "Close DXF" buttons. Open the DXF file first (if you don't the application will crash). Next scroll through the data base with the "Previous" and "Next" buttons. When you see a record you like press either (or both) "Add" buttons to add the text string and/or the line. When finished press the "Close DXF" button and exit OV. Load the DXF file into a supported application to see the graphic you created from the data base. You will require either the runtime or full version of ObjectVision for this demo. A DXF file created with this application is in OVDEMO2.DXF.

6. I have started work on adding the DXF commands to Microsoft Excel. Excel can register DLLs for its use. A future sample application could allow the user to highlight a block of X,Y pairs and use TurboDXF to create a polyline from the data.

To run the sample applications, copy the TURBODXF.DLL from the distribution disk to your main Windows directory. Copy the applications from the two diskettesto sub-directories on your hard disk. I have included the Visual Basic runtime on Disk 1. To run the ObjectVision demos you will have to use Disk 2 and PKUNZIP to decompress the runtime files. You will require about 1Mb of disk space for the ObjectVision runtime. To view the source code for either of these demos you have to install the respective distribution software for those systems.

To run the Turbo Pascal applications you don't need any extra software. To view the Pascal source code use Turbo Pascal for Windows or any text editor.
To view the DXF results created by these programs you will need a DXF aware program. See the above list for tested software.

Mini Manual

This section briefly describes the routines in this sample DLL. When you purchase the full version you will receive a complete detailed manual. The data types for all the calls are PChar,double and integer. Most applications seem to support these data types. I had to use "double" because that is the only floating point type that ObjectVision supports. If your development tool has trouble with these data types let me know and I will do my best to change the DLL.

The general format of a DXF file is:

1. Create and open the DXF file (ASCII)
2. Add some header information (drawing limits, layer names and colours etc.)
3. Add all the entities (lines, arcs, polygons, text etc.)
4. Close the DXF file

The following procedures are included in the DLL. As these are Turbo Pascal procedures (as opposed to functions) they don't return a value. See the respective applications for the required code to register the DLL in each environment.

DXFOpen(FileName:PChar)

This must be the first command issued. This opens the DXF file and adds some house keeping information. The **filename** should include the .DXF extension so other applications will recognize the file.

DXFHeader(X1,Y1,X2,Y2:double)

The **X's** and **Y's** are the corners of the drawing. The **X1**,**Y1** pair defines the lower left corner while the other pair defines the upper right corner. Most Windows applications ignore these numbers and scale the drawing to fit their working area but all CAD applications use it. You must include this call in your application because while the application may not use the values they have places to fill in the DXF file.

DXFStartTables(NumLayers : integer)

Again most Windows applications don't support layers the same way CAD packages do. (Designer has layers but doesn't make the translation from the DXF file - don't ask me why). The **NumLayers** parameter is the number of DXFAddLayer calls you will be making. Most Windows applications aren't sensitive to this number (ie you set **NumLayers**=5 and call DXFAddLayer 10 times). CAD applications

are very sensitive to this parameter. If you are careless here you could create a DXF file that will load into Corel Draw but not AutoCAD. This may or may not be important. Just so you know.

DXFAddLayer(LayerName : PChar; LayerColor : integer);

This procedure adds layers to the drawing. If you are using AutoCAD (or another application that truly supports layers) you will be able to selectively turn the layers on and off. This is fundamental to CAD drawings. Most Windows applications (Corel Draw and Micrografx Designer for example) do not properly support DXF layers. Each entity that is added to a layer is given that layers color. The **LayerColor** is an integer as follows:

1. red
2. blue
3. yellow
4. cyan
5. black
6. white
7. magenta

DXFStartViewTable(NumViews:integer)

AutoCAD supports named views. In AutoCAD you can call up a named view. This will zoom to a pre-defined view. Most Windows applications don't support these named views. **A call to this procedure must be included for the DXF file to be read by any application.** Even if your intended application doesn't support named views you must include this procedure. Simply set **NumViews** to 0.

DXFAddView(ViewName: PChar;Height,Width,CentreX,CentreY : double);

If **NumViews** in the above (DXFStartTable) is "n" (non-zero) then include "n" calls to this procedure. Note that this is really only useful if you will be using AutoCAD to view your DXF drawings. The **ViewName** is a string. The other parameters are self explanatory.

DXFEndTables

No parameters. A call to this procedure is always required. This closes the "housekeeping" section of the DXF file and prepares the file to accept entity data.

DXFAddText10(X1,Y1,Z1,Height,Rotate : double;Txt,LayerName : PChar)

This is the standard command to add text to the DXF file. The "10" suffix indicates it is AutoCAD rel. 10 compatible. There is another more sophisticated command in the full version of TurboDXF that supports the richer AutoCAD rel. 11 text command. The **Z1** value is only required in AutoCAD since most Windows applications force all the elements to zero. Experiment with the **Z1** to see if your application supports it. The **height** is the text height. In AutoCAD it is in the units of the drawing (feet or meters for example). Windows applications tend to scale the drawing to fit their drawing space so again experimentation is required. **Rotate** is the angular rotation in degrees.**Txt** is the text string you want to add. Line breaks are not supported. To do a paragraph of text you will need multiple DXFAddText10 calls.

DXFAddPoint(X1,Y1,Z1 : double; Layer : PChar);

Adds a point to a drawing at the X,Y,Z coordinates. Points are very small (their size is not a parameter) and I have never really found an application for them. I suppose if you had enough of them you could create some sort of scatter diagram. I think it would be better to add small circles instead. Included for completeness.

DXFAddArc(X1,Y1,Z1,Radius,StartAngle,EndAngle : double;Layer : PChar)

X,Y,Z is the centre point for the arc. Radius is self explanatory. The two angle parameters are in degrees (positive counter-clockwise from parallel). Note that a StartAngle of -10 is the same as 350. See any of the demo applications.

DXFAddSolid(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4 : double;Layer : PChar)

A four point filled polygon. The fill colour is the layer colour. Set the "4" series parameters equal to the "3" series to get a triangle. There is a DXFAddBlock command that makes creating bar charts easier. It calls this procedure.

DXFAddLine(X1,Y1,Z1,X2,Y2,Z2: double;LayerName : PChar);

Very simple. Adds a line from the first point to the second. The colour of the line is set by the layer the line is on. These lines are always hairlines (very thin).

DXFAddPoly(Layer : PChar)

The DXF format supports polylines. Polylines are multi-segment lines that can have varying thicknesses. The lines are filled with the colour

of the layer. To create a polyline call this procedure once. For each segment of the polyline call the DXFAddVertex procedure with the point (vertex) data. Add as many vertices as required then finish with a DXFEndPoly.

DXFAddVertex

Call this procedure after a single call to DXFAddPoly. Call it as many times as required for each vertex. True DXF aware applications support different start and end widths for a line segment. On the other hand Windows applications support of polylines is very variable. Corel Draw ignores the line thickness. Micrografx Designer makes each segment the same thickness as the StartWidth. Load one of the demo drawings into AutoCAD to see the variable width polyline. For multi segment lines this is easier than multiple calls to the DXFAddLine procedure.

DXFEndPoly

Required once after all the calls to DXFAddVertex are made. This finishes the polyline. No parameters are required.

DXFAdd3DFace(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4:double;Layer: PChar)

Adds a 3D polygon to the DXF file. Parameters as before. This is a 3D command so Windows support is variable (typically non-existent). Use DXFAddSolid. AutoCAD supports this command. Very useful for creating surfaces etc.

DXFAddCircle(X1,Y1,Z1,Radius,Extrusion:double;LayerName:PChar)

Adds a circle centred at X,Y,Z. Radius is self-explanatory. The extrusion makes the circle into a tube in 3D. Typically not supported in Windows applications but you should experiment with your application.

DXFClose

When all the entities have been added a single call to this procedure will close the DXF file. **One and only one call is required**

DXFAddBar(X1,Y1,Width,Height:double; Layer:PChar)

Adds a bar to the DXF file. This is a very simple and useful way of creating bar charts. Simplifies the use of the DXFAddSolid procedure. The X,Y pair is the lower left corner of the bar. The width and height are self-explanatory. The colour of the bar will be controlled by the layer. See the sample program GRAFDEMO.PAS.

DXFAddXAxis(X1, Y1, X2 : double; NumTicks : integer; Layer : PChar)

This procedure adds an X axis to the DXF file. This is intended to be used with DXFAddBar above to create DXF bar charts. The X1,Y1 pair is the start point of the axis (usually 0,0). X2 is the ending X value (ie Max X). The number of ticks is controlled with NumTicks -- text labels will be printed just below the axis at all ticks. See the sample program GRAFDEMO.PAS.

DXFAddYAxis(X1, Y1, Y2 : double; NumTicks : integer; Layer : PChar)

This procedure adds an Y axis to the DXF file. This is intended to be used with DXFAddBar above to create DXF bar charts. The X1,Y1 pair is the start point of the axis (usually 0,0). Y2 is the ending Y value (ie Max Y). The number of ticks is controlled with NumTicks -- text labels will be printed just below the axis at all ticks. See the sample program GRAFDEMO.PAS.

Turbo Pascal Sample Program TPDEMO1.PAS

```
Program TPDemo1;
uses WinCrt;

Procedure DXFOpen(FileName:PChar); far; external 'TURBODXF' index 1;
Procedure DXFHeader(X1,Y1,X2,Y2:double); far; external 'TURBODXF' index 2;
Procedure DXFStartTables(NumLayers : integer); far; external 'TURBODXF' index 3;
Procedure DXFAddLayer(LayerName : PChar; LayerColor : integer); far; external 'TURBODXF' index 4;
Procedure DXFStartViewTable(NumViews:integer); far; external 'TURBODXF' index 5;
Procedure DXFAddView(ViewName: PChar;
                      Height,Width,CentreX,CentreY : double);far; external 'TURBODXF' index 6;
Procedure DXFEndTables; far; external 'TURBODXF' index 7;
Procedure DXFAddText10(X1,Y1,Z1,Height,Rotate : double; Txt,LayerName : PChar); far; external
                            'TURBODXF'index 9;
Procedure DXFAddPoint(X1,Y1,Z1 : double; Layer : PChar); far; external 'TURBODXF' index 10;
Procedure DXFAddArc( X1,Y1,Z1,Radius,StartAngle,EndAngle : double;Layer : PChar); far; external
                           'TURBODXF' index 11;
Procedure DXFAddSolid( X1,Y1,Z1,
                     X2,Y2,Z2,
                     X3,Y3,Z3,
                     X4,Y4,Z4 : double;
                     Layer : PChar);far; external 'TURBODXF' index 12;
Procedure DXFAddLine(X1,Y1,Z1,X2,Y2,Z2: double;LayerName : PChar); far;
                      external 'TURBODXF' index 13;
Procedure DXFAddPoly(Layer : PChar);far; external 'TURBODXF' index 14;
Procedure DXFAddVertex( X1, Y1, Z1, StartWidth, EndWidth : double;
                       Layer : PChar);far; external 'TURBODXF' index 15;
Procedure DXFEndPoly; far; external 'TURBODXF' index 16;
Procedure DXFAdd3DFace(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4:double;
                     Layer:PChar);far; external 'TURBODXF' index 17;
Procedure DXFAddCircle(X1,Y1,Z1,Radius,Extrusion:double; LayerName:PChar); far; external
                           'TURBODXF' index 18;
Procedure DXFClose; far; external 'TURBODXF' index 19;
Procedure DXFAddBar(X1,Y1,Width,Height:double; Layer:PChar);far; external 'TURBODXF' index 20;
Procedure DXFAddXAxis(X1,Y1,X2:double;NumTicks:integer;Layer:PChar);far; external 'TURBODXF'
                            index 21;
Procedure DXFAddYAxis(X1,Y1,Y2:double;NumTicks:integer;Layer:PChar);far; external 'TURBODXF'
                            index 22;
{=============================================================================================}

var
 i : integer;
 DXFFileName : PChar;

begin
 DXFFileName := 'TPDEMO1.DXF';
 Writeln('Testing TURBODXF DLL. Creating DXF file: ',DXFFileName);
 DXFOpen(DXFFileName);                                    Create and open the ASCII file

 DXFHeader(1,1,10,10);                                    Define the drawing limits. Lower left at
                                                          1,1. Upper right at 10,10

 DXFStartTables(5);                                       5 layers

 DXFADDLayer('LINES',1);                                  Add the layers, The number is
the                                                       colour code.
 DXFADDLayer('TEXT',7);
 DXFADDLayer('ARC',1);
 DXFADDLayer('SOLID',5);
 DXFADDLayer('POLY',7);

 DXFStartViewTable(0);                                    No defined views. Call needed for DXF
                                                          compatibility.

 DXFEndTables;                                            Required. No parameters.
```

```
 DXFAddBar(-2,-2,15,16,'POLY');                          The first entity. On layer
'POLY'. This                                                          makes the
background.

 for I := 1 to 5 do
       DXFAddBar(I*2,1,1,I*2,'SOLID');                   Add 5 bars on layer SOLID.

 for I := 1 to 5 do                                      Add 5 bars on layer LINES
       DXFAddBar(I*2-1,1,1,I*2-1,'LINES');

 DXFAddPoly('POLY');                                     Add a polyline. Applications
vary in                                                          their support of this
entity.
  DXFAddVertex(3,1,0,0.5,0.25,'POLY');
  DXFAddVertex(4,3,0,0.35,0.5,'POLY');
  DXFAddVertex(5,3,0,0.75,0.15,'POLY');
  DXFAddVertex(7,5,0,0.25,0.25,'POLY');
 DXFEndPoly;                                             End the polyline.

 DXFAddLine(0,0.5,0,10,0.5,0,'LINES');                   Add a line.

 DXFAddLine(0,0.5,0,0,8,0,'LINES');

 DXFAddText10(2,12,0,0.5,0,'TurboDXF! from Ideal','TEXT');   Add some text. This is the
                                                                  title

 DXFAddCircle(3,3,0,2,0,'ARC');                          Add a circle.
X,Y,Z,Thickness,Radius.

 for I := 1 to 8 do
   DXFAddText10(3,7,0,0.25,I*45,'TurboDXF!','TEXT');     Add the text "spokes"

 DXFAddArc(6,5,0,5,-10,110,'ARC');                       Add an arc. X,Y,Z,Radius,Start
angle,                                                           end angle

 DXFAddSolid(7.5,1,0,9.5,1,0,9.5,6,0,7.5,3,0,'TEXT');    Add a filled solid

 DXFAddXAxis(0,0,10,10,'TEXT');                          Add an x axis...

 DXFAddYAxis(0,0,10,10,'TEXT');                          and a y axis.

 DXFClose;                                               Close the DXF file.

 Writeln('Test Complete. DXF file closed.');
 writeln('Use Corel Draw, Micrografx Designer etc to view.');

end.
```

When compiled and run this Pascal program creates the TPDEMO1.DXF file.. The source code for this program can be found in TPDEMO1.PAS.