Procedures for Building An ARX Application
Using MSVC++ 4.0 IDE on WindowsNT Workstation

The first step in building an ARX application is to make a Project Workspace. The Project contains all the settings and files that are use to build the ARX application. ARX applications are Windows dynamic-linked libraries (DLL), and therefore you must choose the Dynamic-Link Library Project type (NOT using MFC) when creating the Project Workspace. A minimum application would consist of: TEMPLATE.CC (renamed as TEMPLATE.CPP), TEMPLATE.DEF and the libraries (RXAPI.LIB, ACAD.LIB, ACEDAPI.LIB). During the project creation any .CPP or (.C), RC, and DEF are added for your program.

NOTE: *Unless your ARX program contains Windows specific code that requires a .RC file, you will not need any .RC files.*

The basic steps to create a minimum project are:

1. Start the Microsoft Visual C++ 4.0 Developer Studio. From the "File" pulldown, select "New".
2. Select "Project Workspace" in the dialog that appears.

3. In the next dialog, enter your desired project name in the appropriate edit box. Select "Dynamic-Link Library" as the project type. Fill in the directory information as you desire. Then select "Create".

4. Copy TEMPLATE.CC and TEMPLATE.DEF from the \COM\ADSRX\SAMPLES directory to the directory you specified for the project workspace in the previous step. Rename the TEMPLATE.CC file to TEMPLATE.CPP.

5. Next you will need to choose your files for the project. From the “Insert” pulldown, select “Files into Project.” Select the source (.C, .CPP, etc.), definition (.DEF), resource (.RC), and library (.LIB) files to use in your project. For this example choose TEMPLATE.CPP and TEMPLATE.DEF which you already copied to the project workspace directory. Also add the necessary ARX libraries (RXAPI.LIB, ACAD.LIB, ACEDAPI.LIB) usually found in the \COM\ADSRX\LIB directory. By adding TEMPLATE.CPP, TEMPLATE.DEF and the ARX libraries, the minimum ADS files have been included in your project.

Now the Compiler values need to be set:

6. In the "Build" pulldown, select "Settings" to bring up the "Project Settings" dialog. Initially the "Settings for:" list box should have both "Win32 Debug" and "Win32 Release" highlighted so all setting changes will apply to both.

7. Select the "C/C++" tab to bring forward the compiler settings. Here are the categories and appropriate settings. Settings not shown should be left as the default values. Other settings may work, but these are the recommended values:

Code Generation

Use run-time library->	Single-threaded (using other settings will likely cause errors)

Preprocessor

Preprocessor Definitions->	Add "ACRXAPP, RADPACK" to the list.
Additional Include Directories ->	Add in the \com\ads and \com\adsrx\inc directories and others necessary to your project. Using the ARX SDK, you can just add the \arx\inc directory.

NOTE: *You may also add the ADS/ARX directory paths to the general MSVC++ IDE include search paths and leave it out of "Additional Include Directories" edit box.*

Finally the Linker values need to be set:

8. Select the "Link" tab to bring forward the linker settings. Settings not listed should be left as the default values. Other settings may work, but these have been tested:

General

Output File Name ->	Enter the desired name. If you use an extension other than .dll, MSVC++ 2.x will use it. It is recommended that you use the extension .ARX because this is the default extension that AutoCAD will use for ARX applications. For this example use TEMPLATE.ARX.

NOTE: *MSVC++ 4.x may not automatically place an extension on the final DLL file. So, If you do not include an extension then there will be no extension on the final DLL file.*

Output

Base Address ->	0x1c000000
Entry-Point Symbol ->	DllEntryPoint@12

9. Choose the OK button and the project is now complete and is ready to be built into a Windows ARX program. This single executable can be run in any Windows based Release 13 ARX environment. Currently these are: Windows 3.1x (with AutoCAD R13 properly installed including Win32s), Windows NT 3.5x, and Windows 95.

NOTE: *There is currently a problem where applications built using MS Visual C++ 4.0 are NOT compatible with the DOS386 platform. Using Visual C++ 2.1 or 2.2 an ARX application can be loaded in the AutoCAD DOS386 version. This problem is currently being evaluated and we will update this document in the future depending on the out come.*

Now all the settings have been made in order to build the application.

Debugging ARX with the MSVC++ 4.0 IDE

To debug an ARX session you need to start AutoCAD from the debug environment. Here are the steps to set this up:

1. From the "Build" pulldown choose "Settings."

2. Select the "Debug" tab and enter the full path of your AutoCAD executable in the "Executable for debug session" edit box. This would normally be \R13\WIN\ACAD.EXE.

3. In the "Working directory" edit box enter the directory you want AutoCAD to start up in. This might be the AutoCAD directory (\R13\WIN) or your project directory. This setting is not required.

4. In the "Program arguments" edit box enter any AutoCAD command line arguments. This is might be important if you are running AutoCAD with a working directory set to your project directory, but you want AutoCAD to see a configuration file from somewhere else. You might set this as: /c d:\cfg\r13nt.

The basic steps for a simple debug session follows. The source code used as an example is the TEMPLATE.C file using the following adsfunc function definition:

```
int adsfunc()
{
      int i,j;
      for (i = 0; i < 10; i++) {
            j = i*100;
            ads_printf ("\ni = %d, j= %d", i, j);
      };

      ads_printf("\nDone!");
      ads_retvoid();
      return RSRSLT;
}
```

To debug this application:

1. Create a project according to the aforementioned steps using the TEMPLATE.C application.

2. Add the adsfunc function mentioned above.

3. Build the project.

4. Set a break point. Open the TEMPLATE.C program by double clicking on the file in the FileView Tab of the Workspace(Left hand Window within the Developer Studio). Then place the cursor at the line that contains:

   j = I*100;

   Right click and choose "Insert/Remove Breakpoint." This should insert a new break point and you will see it with a red dot display in the margin.

5. From the Build pulldown, choose Debug and then Go. AutoCAD should start running now.

6. Switch back to the AutoCAD Window and load the ADS application. Use APPLOAD or (XLOAD "<yourapp>")

7. Execute the application (ADSFUNC). Control will switch back to the Developer Studio and you can step through the contents for the variable j.

Laboratory #1-a: 3-D Solid Model Generation Using AutoCAD

1. Introduction

Solid modeling offers several advantages. A significant advantages of solid modeling over wireframe and surface modeling is the ability to determine mass properties for a solid or set of solids. In this lab

session you will construct a solid model of a pulley, and find several mass properties useful for engineering calculations. The following section summarizes the primary solid modeling commands.

Within AutoCAD solid models can be constructed by combining a set of solid primitives. The solid primitives available are:

Box	(solbox command)
Sphere	(solsphere command)
Wedge	(solwedge)
Cone	(solcone command)
Cylinder	(solcyl command)
Torus	(soltorus command)

Two other types of solids can be defined, solids of revolution and extrusions. Solids of revolution are defined by sweeping a curve around an axis, (solrev command). Extrusions (soltext command) are defined by extruding a curve along a specified axis.

The solid primitives are combined using Boolean operations, i.e.,

Union	(solunion command)
Subtraction	(solsub command)
Intersection	(solint command)

2. Lab Exercise

a. Set solwdens system variable to 6.
This controls the quality of hidden line representations. Setting it to higher values increases processing time.

b. Create the pulley shown in the figure 1.

- Begin by defining 1/2 of it cross section with lines and arcs. leave out the keyway and 5 holes on face. These will be defined later. Convert these lines and arcs to a polyline with the pedit command. (The polyline is required by the solrev command).

- Sweep the cross section about the centerline.
Use the solrev command to sweep the 2D cross section through 360 degrees. The centerline can be defined with the entity option of solrev.

- Create the 5 holes in the face
Create one hole with the solcyl command. Then use the array command to copy the cylinder to each of the file locations. Use the solsub command to subtract the cylinders from the pulley face.

- Create the keyway
Create a box primitive of appropriate dimensions for the keyway. Use solsub to subtract

the box from the pulley.

c. Calculate the Pulley's Mass Properties

Set the solsubdiv system variable to 3. This variable controls the accuracy of mass property calculations.

- Calculate the mass properties with the solmassp command. Write the properties to a file.

d. Create a fully dimensioned drawing of the pulley as shown in figure 1
Create the drawing in the paper space. Use the solprof command to generate visible and hidden line views as required. Create the section view with the solsect command.

3. Lab requirements:

You are required to submit:

a. Your completed drawing of the pulley. It should appear as shown in figure 1.

b. A print our of the mass property calculation file.