GRX Migration Guide

GstarCAD 2027

Table of Contents

1. Introduction

GRX is the Runtime eXtension programming environment, which is the latest application development SDK. It provides Object-oriented development environment and APIs based on C++, which can be used to develop application programs, extension CAD classes and protocols, and create new commands.

One of GRX's advantages is that it supports seamless migration of ARX application programs running on AutoCAD® with little change to the original source code, keeping the source-code-level compatibility.

2. Programming Environment

- Microsoft® Visual Studio 2022 (version 17.8.0)
- Windows SDK 10.0 (latest installed version)
- CPU:

  Basic: 1.6 GHz CPU

  Recommended: 3.0 GHz CPU and above
- RAM:

  Basic: 2 GB

  Recommended: 8 GB and above
- Operation System (OS)

  Windows 11

  Windows 10 (version 1507 and above):

  Home, Professional, Education and Enterprise (not support LTSC and Windows 10 S)
- Monitor Resolution:

  1028x800 and above true color display, including 4K (3840x2160) display
- GRXSDK
- Microsoft .NET 8.0

3. Requirement on Original Source Code

For the application source code with version lower than ObjectARX 2025, it needs to upgrade the source code to the ObjectARX 2025 first. Source code shall be compiled with Visual Studio 2022, under 'Use Unicode Character Set' and 'Multi-threaded DLL(/MD)' environment.

4. Install SDK

Download SDK.

Unzip GRXSDK.ZIP file to the local disk (e.g. '*C:\grxsdk*') and there will be 5 directories generated (in '*C:\grxsdk*') which are: **arx**, **inc**, **inc-x64**, **lib-x64** and **utils**.

arx contains the header files, library files and sample programs used for porting ARX programs to GRX programs. It contains the following directories:

- **inc:** Header files used for porting from ARX to GRX
- **inc-x64**: Files used by COM and .NET (for 64-bit)
- **lib-x64**: GRX libraries (for 64-bit)
- **Samples**: Sample projects, including Dotnet, fact_dg, HelloADS, HelloARX and SimplePalette.
  - **Dotnet**: .NET programming samples
    1) **Addline**: .NET programming sample of adding solid lines
    2) **Hello**: .NET programming sample of outputting prompt information
    3) **Vbhello**: Sample of .NET programming with VB .NET
  - **fact_dg**: Sample of LISP function definition
  - **HelloADS**: Sample of ADS programming
  - **HelloARX**: Sample of GRX programming
  - **SimplePalette**: Programming sample of how to create a set Palette windows
- **Utils:** Directory contains sub-directories of GRX extended applications, including APIs for extended function development, e.g. BREP for boundary representation.

Inc: Header files used for programming the GRX

inc-x64: Files used by COM and .NET (for 64-bit)

lib-x64: GRX libraries (for 64-bit)

Utils: Directory contains subdirectories of GRX extended applications, including APIs for extended function development, e.g. BREP for boundary representation.

5. Project Configuration

NOTE: '*<sdkpath>*' indicates the installation path of SDK, e.g. '*C:\grxsdk*'.

1) **General/Target Extension**: *.grx*

2) Select **C/C++** in **Configuration Properties** and set as below:

   General/Additional Include Directories: *<sdkpath>\arx\inc*

3) Select **Linker** in **Configuration Properties** and set as below:

   General/Accessory library directory: *<sdkpath>\arx\lib-x64*

NOTE: Under debug configuration, please delete **_DEBUG** macro definition from **C/C++** drop down list **Preprocessor -Preprocessor Definitions** and set **Code Generation/Runtime Library** as '*Multi-threaded DLL (/MD)*'.

4) Additional Dependencies:

Lib Name	Need to import or not	Description of related module
AecModeler.lib	As needed	Faceted solid modeler
GcDbConstraints.lib	As needed	Constraint
gcdyn.lib	As needed	Dynamic block
gcad.lib	Usually	GUI/Controls
gcax.lib	As needed	COM service
gcbase.lib	Usually	Basic APIs, including memory, graphics, strings, etc
gcbr.lib	As needed	3D solid boundary
gccore.lib	Usually	Editing/Document
gcdb.lib	Usually	Database/Entity
GcDbPointCloudObj.lib	As needed	Point cloud
GcGeoIocationObj.lib	As needed	Geographic map
gcgs.lib	As needed	Display
GcImaging.lib	As needed	Image
GcModelDocObj.lib	As needed	Symbolic entity
gplot.lib	As needed	Printing

Select **Input/Additional Dependencies** under **Linker** in **Configuration Properties**, remove lib files of ARX and add lib files of GRX:

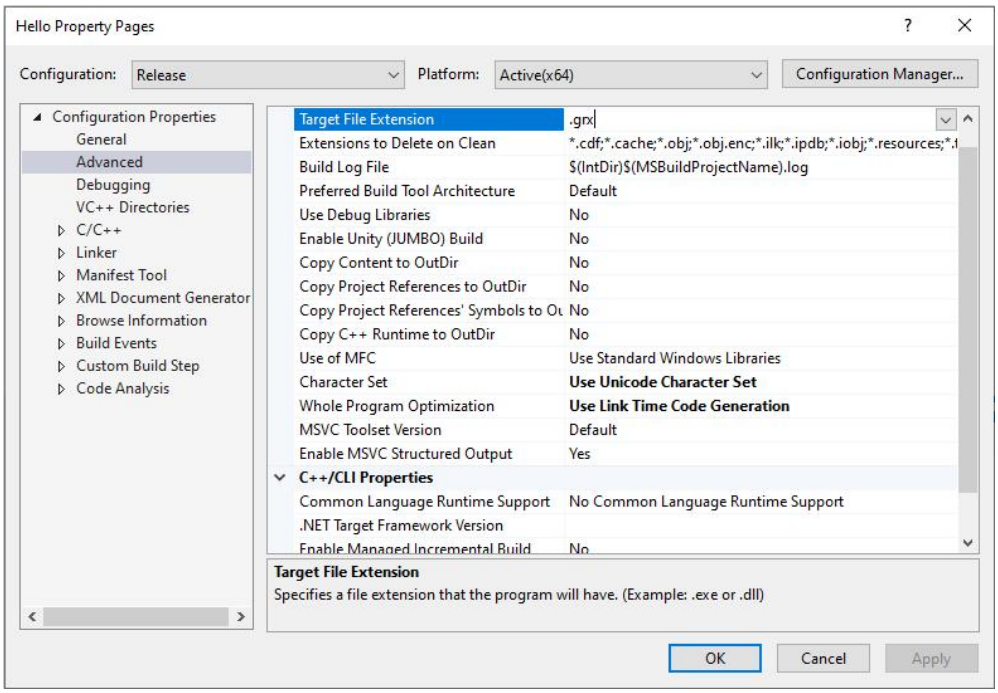
'*AecModeler.lib;gcad.lib;gcax.lib;gcbase.lib;gcbr.lib;gccore.lib;gcdb.lib;GcDbConstraints.lib;GcDbPointCloudObj.lib;gcdyn.lib;GcGeolocationObj.lib;gcgs.lib;GcImaging.lib; GcModelDocObj.lib;gplot.lib;*'

Meanwhile, set **Module Definition File** path as '*<sdkpath>\arx\inc\AcRxDefault.def*'

6. Migrate ARX Project with Visual Studio 2022

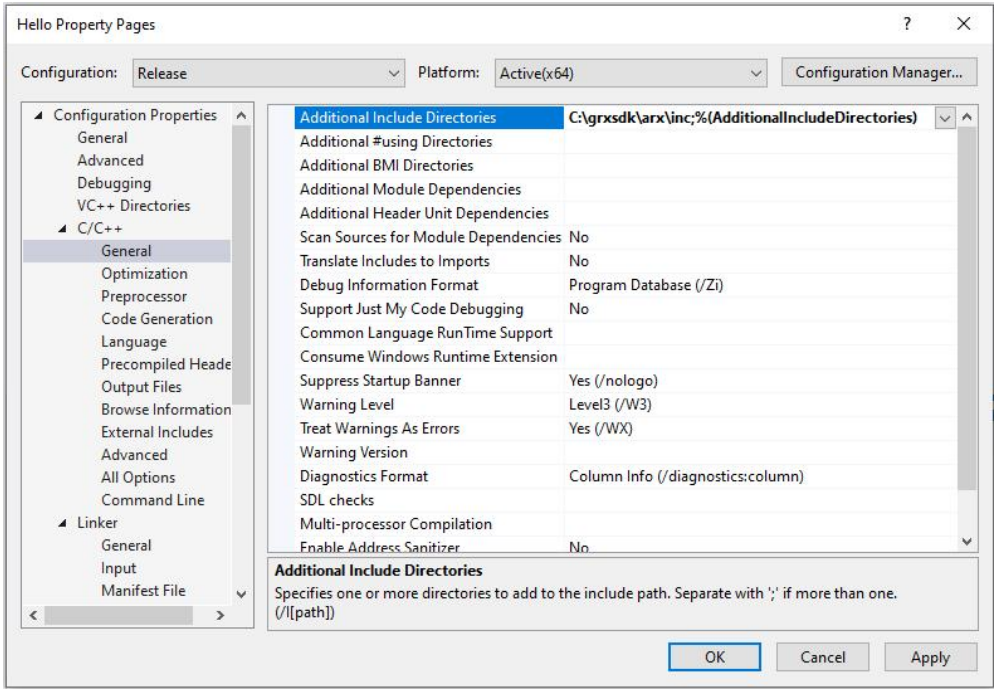
6.1. Configuration of Target Extension

Select **Advanced** in **Configuration Properties** and set **Target Extension** as '*.grx*', and set **Configuration** as '*Release*'.



6.2. Configuration of Include Directory

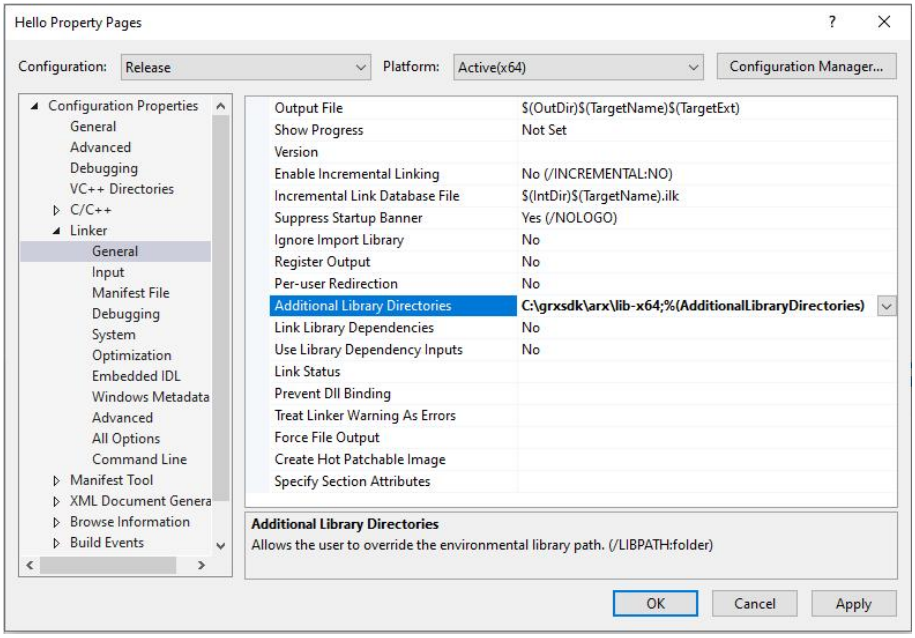
Select **C/C++** in **Configuration Properties** and set **Additional Include Directories** of **General** as '*<sdkpath>\arx\inc*' as shown below.



6.3. Configuration of Link

1) General Configuration

Select **General** under **Linker** of **Configuration Properties** and set **Additional Library Directories** to '*<sdkpath>\arx\lib-x64*' , as shown below.

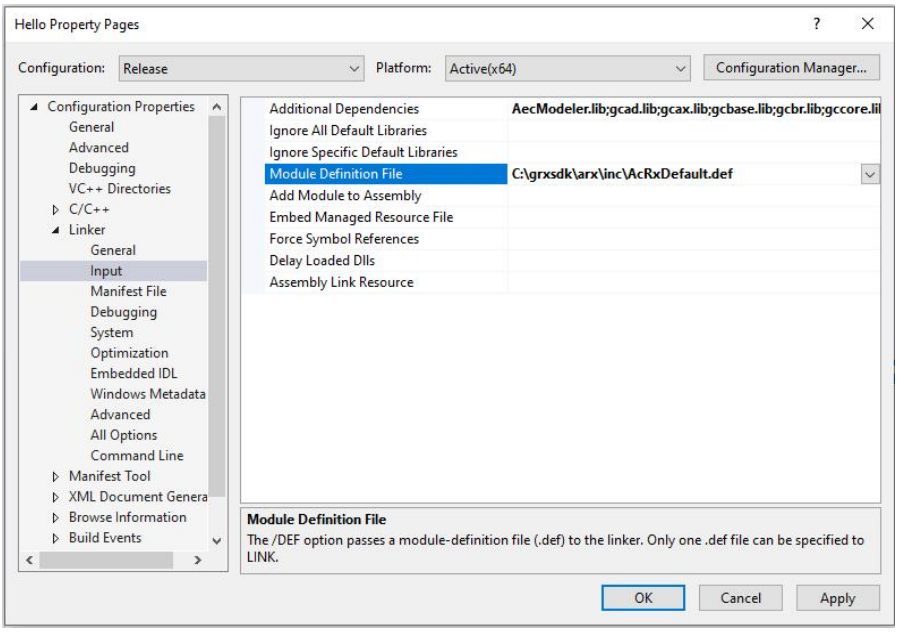


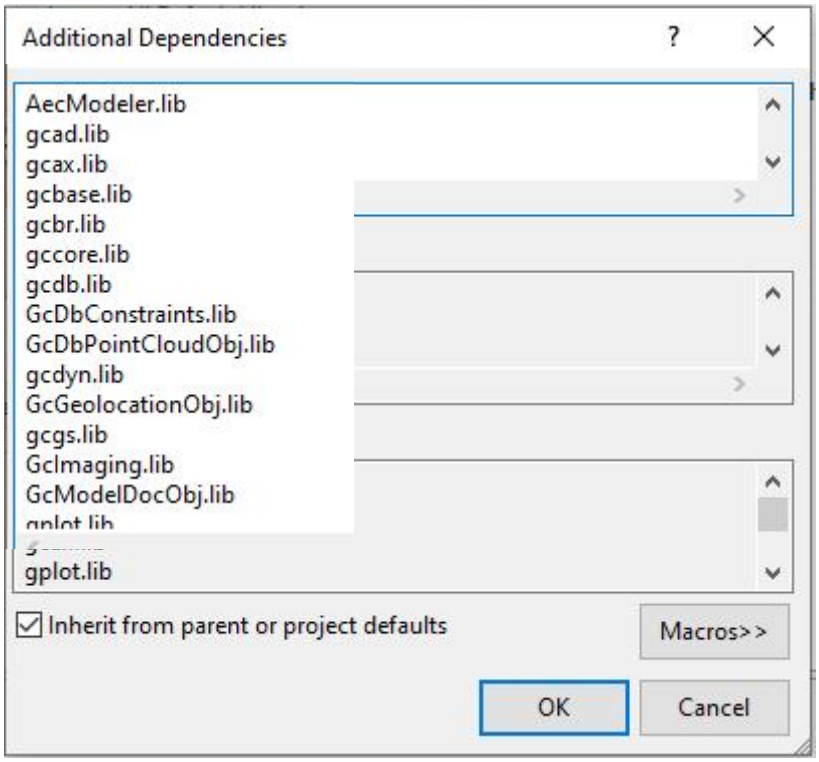
2) Input Configuration

Select **Input/Additional Dependencies** under **Linker** of **Configuration Properties**, remove the lib files of ARX and add lib files of GRX:

'*AecModeler.lib;gcad.lib;gcax.lib;gcbase.lib;gcbr.lib;gccore.lib;gcdb.lib;GcDbConstraints.lib;GcDbPointCloudObj.lib;gcdyn.lib;GcGeolocationObj.lib;gcgs.lib;GcImaging.lib; GcModelDocObj.lib;gplot.lib;*'

Meanwhile, set **Module Definition File** path as '*<sdkpath>\arx\inc\AcRxDefault.def*'





6.4. Compile Program

After completing the above steps, compile the project and make sure that the compilation is successful. Otherwise repeat the above steps to reconfigure the project settings.

7. Description of GRX Class Library

The following libraries are frequently used in programming with GRX. These libraries have same functions with the corresponding ARX libraries.

- **GcRx**: same as **AcRX** of **ARX**, classes for binding an application and for runtime class registration and identification
- **GcEd**: same as **AcEd** of **ARX**, classes for registering commands and for event notification
- **GcDb**: same as **AcDb** of **ARX**, database class
- **GcGi**: same as **AcGi** of **ARX**, graphics classes for rendering entities
- **GcGe**: same as **AcGe** of **ARX**, utility classes for common linear algebra and geometric objects

7.1. GcRx

The **GcRx** class library provides system-level functionality such as DLL initialization and linking, and runtime class registration and identification as below:

- Object runtime class identification and inheritance analysis
- Runtime addition of new protocol to an existing class
- Object equality and comparison testing
- Object copy

7.2. GcEd

The **GcEd** class library is used to define and register new commands which operate in the same manner as the original ones.

7.3. GcDb

The **GcDb** classes are components of the database. This database stores all the information for the graphical objects that compose a drawing, and the non-graphical objects (such as layers, linetypes, and text styles) that are also part of a drawing.

7.4. GcGi

The **GcGi** class library provides the graphic interface used for drawing CAD entities.

7.5. GcGe

The **GcGe** class library provides classes used for performing common 2D and 3D geometric operations, including utility classes such as vectors and matrices and basic geometric objects such as points, curves, and surfaces.

8. About the Use of Macro OBJECTPARAM_ALLOWED

In order to be compatible with ObjectARX2025, in GRX, if GcString is passed into a variable parameter function such as gcutPrintf, it is necessary to call GcString :: constPtr (); as follows:

Previous version:

GcString sz ;

gcutPrintf (_T(" \ n%s "), sz);

Current Edition

GcString sz ;

gcutPrintf (_T(" \ n%s "), sz.constPtr ());

If this modification is not made (i.e. constPtr () is not called), a compilation error will be reported when GRX is compiled:

error C4840: non-portable use of class " GcString " as argument to variadic function

Developers can use this compilation error to modify the code where GcString is called in previous versions.

This compatibility change will result in a compilation error when CStirng is passed into a variable parameter function such as gcutPrintf :

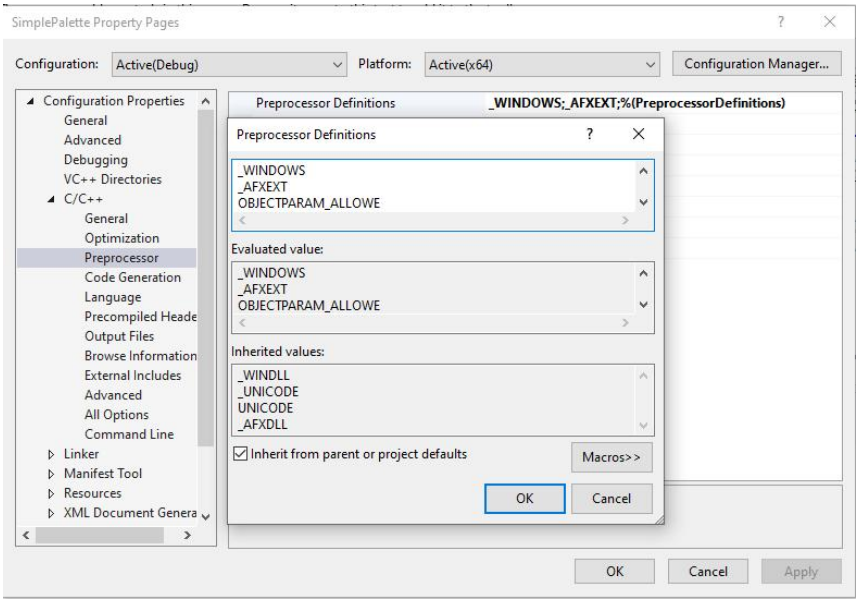
error C4840: non-portable use of class "ATL::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>" as argument to a variadic function

At this point, you can choose one of the following two methods to modify it:

Method 1: Change to

gcutPrintf (_T("%s"), (LPCTSTR (sz)));

Method 2: After modifying all GcString codes , add OBJECTPARAM_ALLOWED definition in the project configuration . This macro definition can ignore all such compilation errors. As shown below



9. Copyright

GstarCAD 2027

https://www.gstarcad.net

Gstarsoft