GRX Migration Guide

GstarCAD 2022

GstarCAD

Table of Contents

1. Brief Introduction

Application Developers from various industries have developed a large number of ARX applications based on AutoCAD®, and one of the highlights of GstarCAD GRX is GstarCAD can migrate the ARX program which runs on the AutoCAD platform to GstarCAD with almost no code changing. Realized source-level compatibility with AutoCAD®ARX programs, which means one set of code can be suitable for two platforms. Developers can refer this guide to easily migrate the code to GstarCAD.

2. Development Environment

- Microsoft Visual Studio Enterprise 2017（Version 15.9.17）
- CPU：

Basic Requirement：1.6 GHz CUP，Recommend：3.0 GHz CPU and above

- RAM：
- Basic Requirement：2 GB，Recommend：8 GB and above
- Operation System
- Windows 10 version 1507 and advanced version：Including home version, professional version, education version and enterprise version（not support LTSC and Windows 10 S）
- Windows 8.1（with updated 2919355）： Core version, professional version and enterprise version
- Windows 7 SP1（with latest Windows updated）： Including home version, professional version, enterprise version and Ultimate version
- Monitor Resolution:
- Traditional monitor：1028 x 800 and above CRT monitor, including 4K (3840 x 2160) monitor
- GRX SDK 2022
- GstarCAD 2022
- .NET Framework 4.8 and above

3. Original Code

Any ObjectARX application for ObjectARX 2010 or a higher version, the application source code needs to be transplanted to VS2017 build environment first, and then it needs to be compiled in the form of multi-byte character sets and multi-thread.

4. Project Configuration

NOTE: <sdkpath> indicates the installation path of GstarCAD SDK, for example: *C:\grxsdk*.

1) General/Target Extension:.grx

2) Select the C/C++ Node under Configuration Properties and set the configuration as below.

Select General and Set Additional Include Directories to <sdkpath>\inc\Arx2010

Add TOOLKIT_IN_DLL macro definitions to Preprocessor/Preprocessor Definitions.

Set the configurations under linker as shown below.

Select General and set Accessory library directory(32-Bit)：<sdkpath>\lib-x86

Select General and set Accessory library directory(64-Bit)：<sdkpath>\lib-x64

NOTE: If '_DEBUG' macro definition needs to be removed from 'DEBUG project configuration'. Please also change 'Runtime Library of Code Generation Node' to 'Multi-threaded Debug DLL (/MDd)' at the same time.

3) DLL Entry Function:

The Entry Point Function of a GRX program is gcrxEntryPoint. To be compatible with ARX, acrxEntryPoint still could be used as the Entry Point of a module but with grxport.lib been imported. This module is a static link library, providing an auto mapping from GRX Entry Point to ARX Entry Point. Correspondingly, RxExport.def will be set as the module definition file under path <sdkpath>\inc\Arx2010.

Note: grxport.lib must be placed in the first position of additional dependencies.

4) Accessory Items：

Lib Name	Be imported	Relative ARX	description
grxport.lib	always	No	acrxEntryPoint entry mapping module
TD_Alloc.lib	Usually	No	Memory Management Module

TD_Root.lib	Usually	AcRx Module	Basic running module
TD_DbRoot.lib	Usually	AcDb Module	Database base module
TD_Db.lib	Usually	AcDb Module	Data management module
TD_Ge.lib	Usually	AcGe Module	Universal Geometry Library
TD_Gi.lib	Usually	AcGi Module	Graphics drawing interface
TD_Gs.lib	Usually	AcGs Module	Graphics System Management Module
gcap.lib	Usually	AcAp Module	Document Management Module
gcui.lib	Usually	AcUi Module	MFC extension module
gcut.lib	Usually	AcUt Module	Auxiliary Interface
gced.lib	Usually	AcEd Module	User interaction module
gcdb.lib	Usually	AcDb Module	ADS database access interface
gcad.lib	Usually	ACAD Module	Application interface management
gcax.lib	occasionally	AcAx Module	COM base class interface
gcgs.lib	occasionally	AcGs Module	Graphics System
Others	Barely	No	Most are internal us

Select Input/ Additional Dependencies under Linker of Configuration Properties.

Please remove lib file of ARX module and add lib file of GRX module:

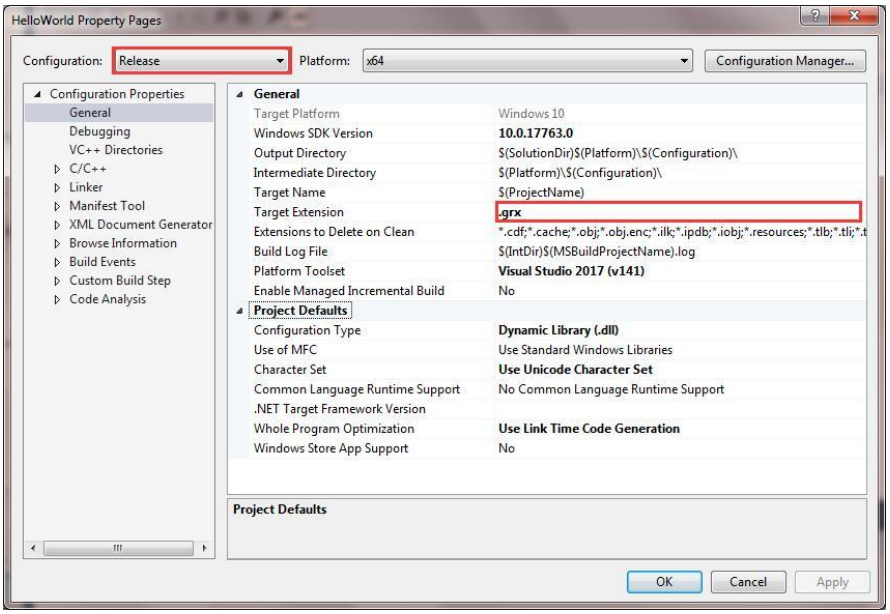
grxport.lib;Td_Root.lib;Td_DbRoot.lib;Td_Db.lib;Td_Ge.lib;Td_Gi.lib;Td_Gs.lib;gcad.lib;gcap.lib;gcdb.lib;gced.lib;gcgs.lib;gcut.lib;gcui.lib

Meanwhile, please set Module Definition file (GRX Enter point function file) path to <sdkpath>\inc\Arx2010\RxExport.def

5. Migrate ARX project with VS2017

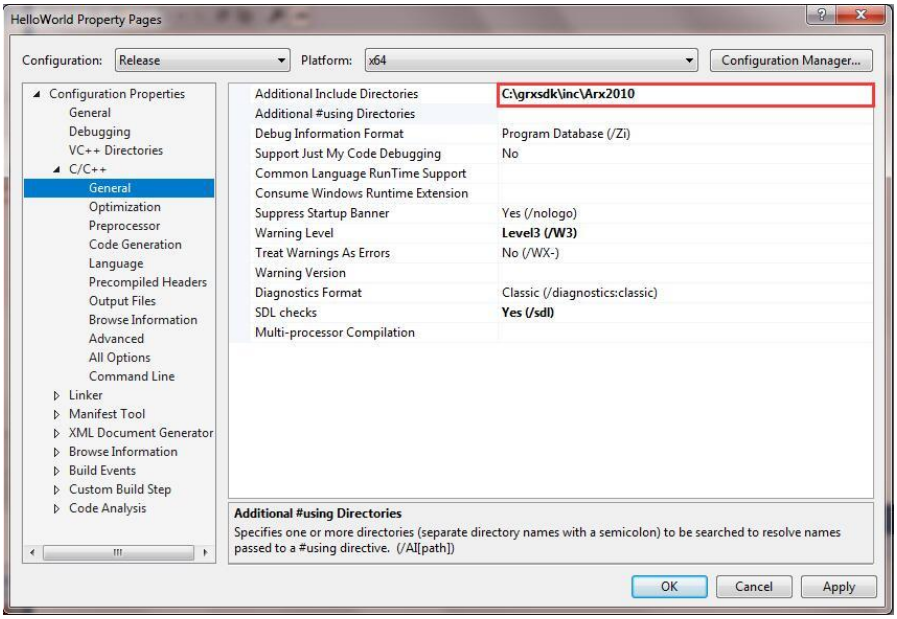
5.1. The Configuration of Target Extension

Select the General Node under Configuration Properties, and set Target Extension as .GRX. Meanwhile set Configuration as Release.



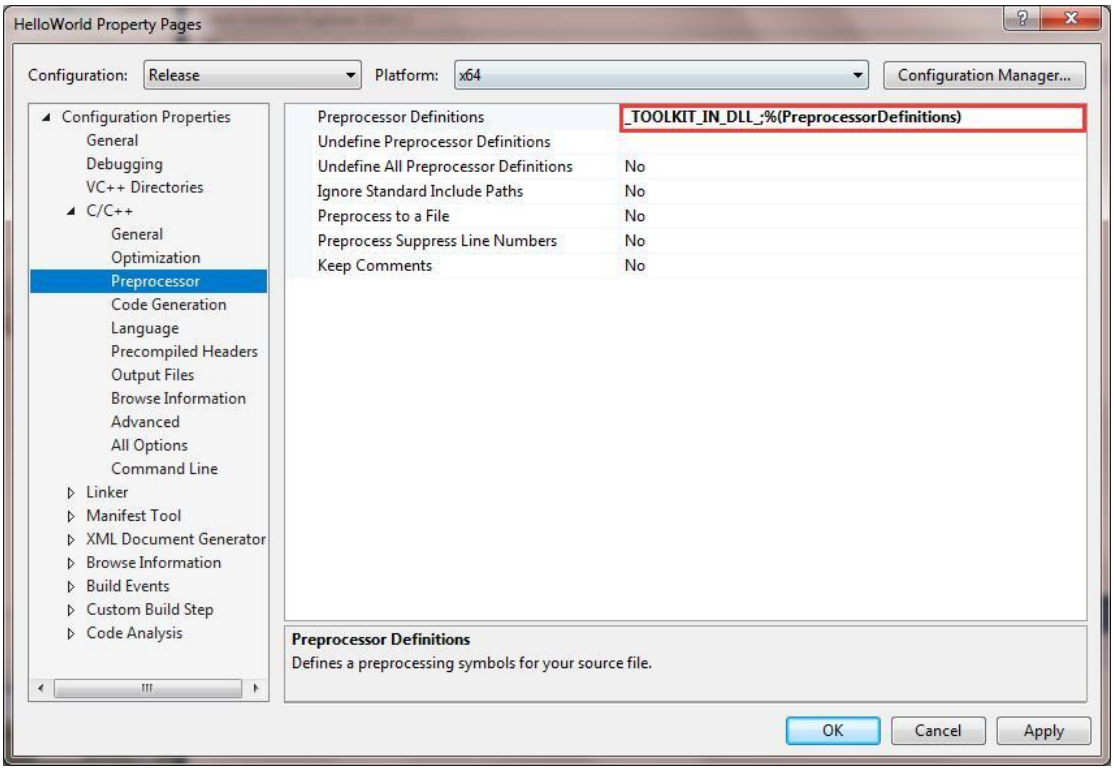
5.2. The Configuration of Include Directories

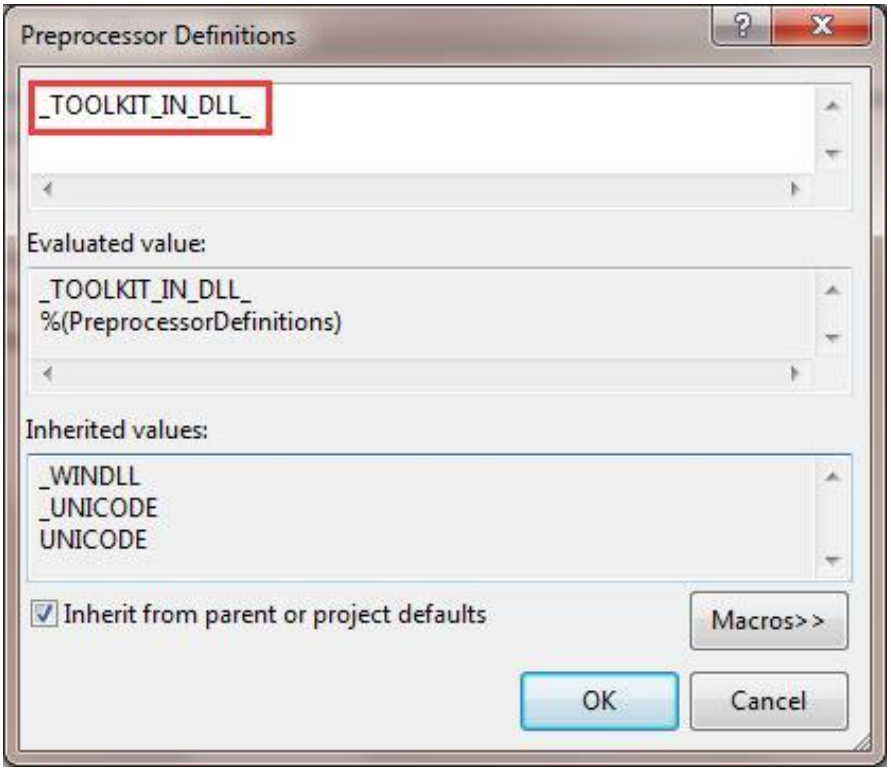
Select the C/C++ Node under Configuration Properties and set Additional Include Directories of General as <sdkpath>\inc\Arx2010 as shown below.



5.3. The Configuration of Preprocessor

Select the Preprocessor under C/C++ from Configuration Properties. Add Marco _TOOLKIT_IN_DLL_ into Preprocessor Definitions as shown below.





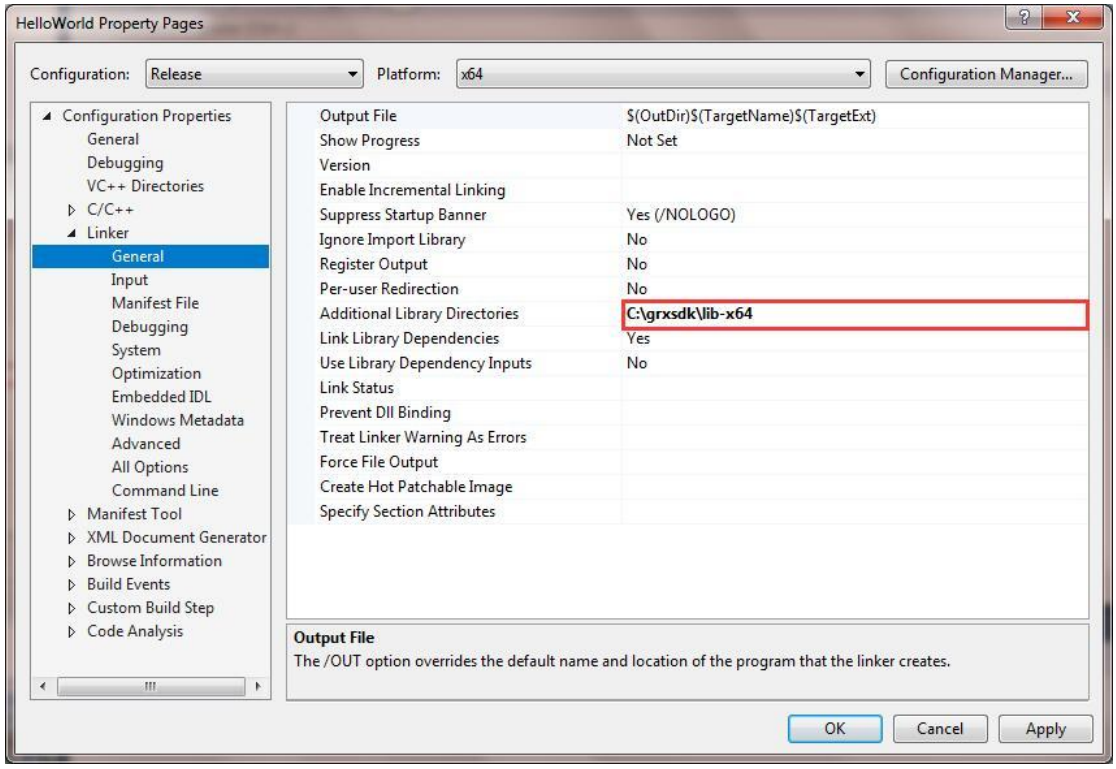
NOTE: If '_DEBUG' needs to be removed from 'DEBUG project configuration'. Please also change 'Runtime Library of Code Generation Node' to 'Multi-threaded Debug DLL (/MDd)' at the same time.

"Language" node：Please set compliance mode as "No".

5.4. The Configuration of Linker

1) General Configuration

Select **General** under **linker** of **Configuration Properties** and modify Additional Library Directories to <sdkpath>\lib-x86（32-Bit）or <sdkpath>\lib-x64（64-Bit）, **as** shown below.



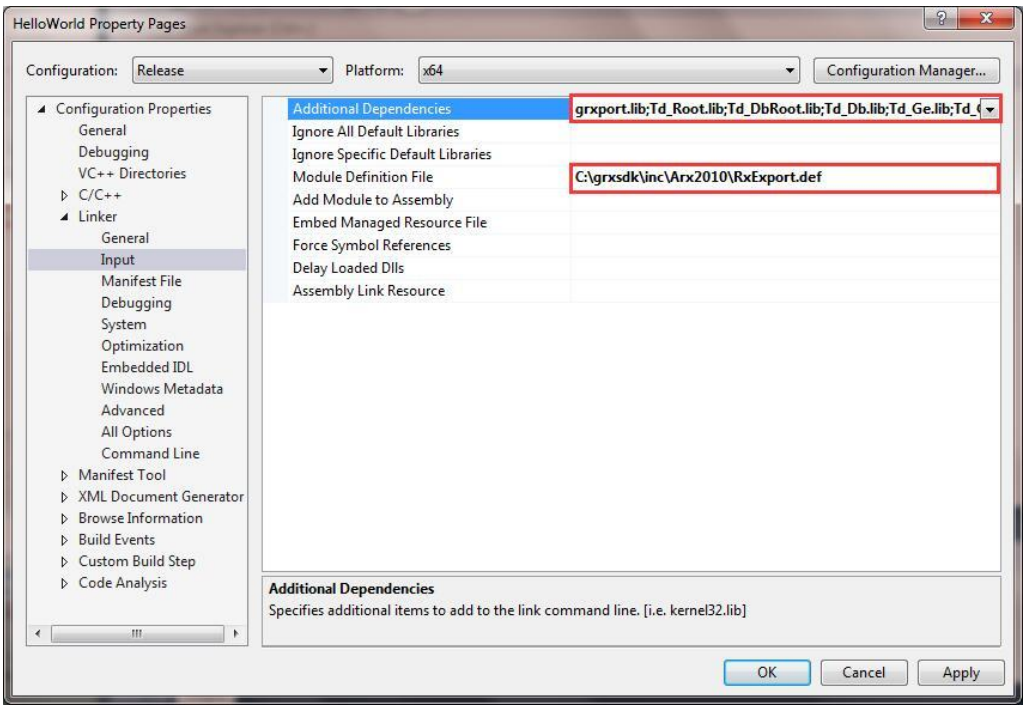
2) Input Configuration

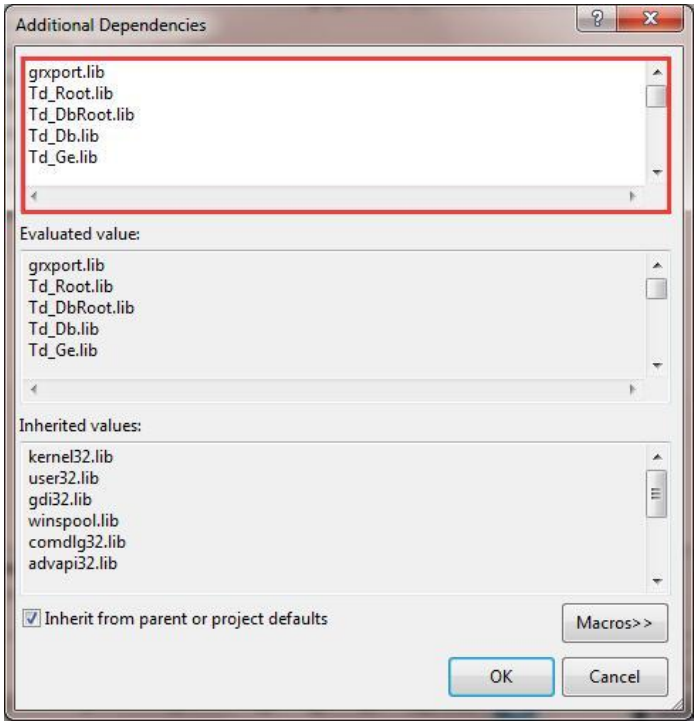
Select Input/ Additional Dependencies under Linker of Configuration Properties.

Please remove the lib file related to ARX and add lib file of GRX:

"grxport.lib,Td_Root.lib,Td_DbRoot.lib,Td_Db.lib,Td_Ge.lib,Td_Gi.lib,Td_Gs.lib,gcad.lib,gcap.lib,gcdb.lib,gced.lib,gcgs.lib,gcut.lib,gcui.lib" (the main GRX library files)

Meanwhile, please set Module Definition file (GRX Enter point function file) path to <sdkpath>\inc\Arx2010\RxExport.def





Note: **grxport.lib** must be placed in the first position of additional dependencies.

5.5. Compile Program

You can compile the project after completing above steps to ensure it works. Otherwise reconfigure the settings.

6. The Differences between GRX and ARX Interface

6.1. Interface for “Plot”

When load Plot interface, the begin Document function in GRX will call beginPage function automatically. Meanwhile, the beginPage will call beginGenerateGraphics function automatically. Which means when you need to load Plot interface, you don’t need to call beginPage, beginGenerateGraphics, endGenerateGraphics and endPage functions any more. Use the GCAD_CORE macro to distinguish the different uses of GRX and ARX as shown below.

Example：

```
    AcPlPlotEngine* mPlotEngine;
    es = AcPlPlotFactory::createPublishEngine(mPlotEngine);
    es = mPlotEngine->beginPlot(NULL);
    AcPlPlotInfo plotInfo;
    AcPlPlotInfoValidator plotInfoValidator;
    plotInfo.setLayout(mPlotSettings->id());
    plotInfo.setOverrideSettings(mPlotSettings);
    plotInfoValidator.setMediaMatchingPolicy(AcPlPlotInfoValidator::kMatchEnabled);
    es = plotInfoValidator.validate(plotInfo);
    es = mPlotEngine->beginDocument(plotInfo, curDoc()->fileName(), NULL,1, true,
_T("c:\\test.eps"));
#ifndef GCAD_CORE
    AcPlPlotPageInfo pageInfo;
    es = mPlotEngine->beginPage(pageInfo, plotInfo, true);
    es = mPlotEngine->beginGenerateGraphics();
    es = mPlotEngine->endGenerateGraphics();
    es = mPlotEngine->endPage();
#endif
```

```
    es = mPlotEngine->endDocument();
    es = mPlotEngine->endPlot();
    mPlotEngine->destroy();
```

6.2. AcGsView::add() Realized by Different Method

AcGsView::add()realized by different method，Use GCAD_CORE macro to distinguish GRX and ARX usage.

Example：

```
Acad::ErrorStatus initDrawingCtrl(AcDbDatabase *pDb,const TCHAR *chSpace)
{
// have we got a valid drawing
if (NULL == pDb)
      {
  return Acad::eNullBlockName;
}
// initialize the preview control
mPreviewCtrl.init();
mPreviewCtrl.SetFocus();
AcDbBlockTableRecordPointer bptr(chSpace, pDb, AcDb::kForRead);
AcDbObjectId idCurrenVs;
// set the view to the Active AutoCAD view
idCurrenVs = SetViewTo(mPreviewCtrl.mpView, pDb, m_viewMatrix);
// tell the view to display this space
mPreviewCtrl.view()->add(bptr, mPreviewCtrl.model());
mPreviewCtrl.view()->setVisualStyle(idCurrenVs);
return Acad::eOk;
}
#ifdef GCAD_CORE
```

```
AcGePoint3d ptMin ( 1e20, 1e20, 1e20);
AcGePoint3d ptMax ( -1e20, -1e20, -1e20);
m_tempExt.set ( ptMin , ptMax ) ;
OdDbDatabasePtr pDb = acdbHostApplicationServices()->createDatabase();
initDrawingCtrl(pDb,L"*Model_Space");
#else
m_tempExt.minPoint().set(1e20, 1e20, 1e20);
m_tempExt.maxPoint().set(-1e20, -1e20, -1e20);
AcDbDatabase* pDb = new AcDbDatabase;
initDrawingCtrl(pDb,L"*Model_Space");
#endif
```

6.3. CAdUiPaletteSet Class Implementation is Different from ARX

Migrating ARX code to GRX, an error is reported when compiling the code and the error prompts that the CFrameWnd::ShowControlBar parameter list does not match.

It is recommended to use the GCAD_CORE macro to distinguish the different usages of GRX and ARX.

Example：

```
extern CPaletteSetWnd *glo_pMainWnd;
void SampleCmd()
{
    if (glo_pMainWnd == NULL)
    {
        glo_pMainWnd=new CPaletteSetWnd;
        if (glo_pMainWnd != 0)
        {
    CSize sizeDefault(200, 450);
            CRect rcDefault(CPoint(150, 200), sizeDefault);
                DWORD dwStyle = WS_CHILD | WS_VISIBLE | CBRS_SIZE_DYNAMIC |
```

```
        CBRS_GRIPPER;
                dwStyle |= CBRS_FLOATING;
                glo_pMainWnd->Create(_T("Main ..."),dwStyle,rcDefault,acedGetAcadFrame());
                glo_pMainWnd->EnableDocking(CBRS_ALIGN_LEFT|CBRS_ALIGN_RIGHT);
                glo_pMainWnd->RestoreControlBar();
#ifdef _GRX_APP // GstarCAD
                CMDIFrameWndEx *mdifwCurrent=acedGetAcadFrame();
                if (mdifwCurrent != 0)
                    mdifwCurrent->DockPane(glo_pMainWnd);
#else // AutoCAD
                acedGetAcadFrame()->ShowControlBar(glo_pMainWnd, TRUE, FALSE);
#endif
            }
            else
            {
#if defined(_GRX_APP) // GstarCAD

                glo_pMainWnd->ShowPane(TRUE, FALSE, TRUE);
#else // AutoCAD
                acedGetAcadFrame()->ShowControlBar(glo_pMainWnd, TRUE, FALSE);
#endif
            }
      }
}
```

The CPaletteSetWnd is a panel derived class

```
class CPaletteSetWnd : public CAdUiPaletteSet
{
......
}
```

You can also refer to the example in grxsdk\sample\ SimplePalette under the SDK development kit

7. GRX Class Library Description

The following libraries are frequently used in GRX. These libraries have the same effect with the corresponding libraries under ARX, mainly including:：

- GcRx- is same with the AcRX library, which is used to bind an application and register and identify for the running class.。
- GcEd- is same with the AcEd library, used to register custom commands and event notifications.
- GcDb-is same with AcDb library, it's GstarCAD database class.
- GcGi- is same with the AcGi library and is used for the graphics class of GstarCAD.
- GcGe- is same with the AcGe library, and is used for general linear algebra calculation and application classes of geometric objects.

7.1. GcRx

The GcRx class library is used for DLL initialization and the system level for running class registration and identification. It provides the functions below:

- Class identification and inheritance analysis of object runtime.
- Add a new protocol to an existing class at runtime.
- Object equality and composition judgment.
- Object copy.

7.2. GcEd

The GcEd class library is used to define and register new GstarCAD commands, which are the same as GstarCAD internal/original commands.

7.3. GcDb

The GcDb class library represents the classes that build up the GstarCAD database. The database stores the information of all graphic objects. These graphic objects are called

“Entities”. These graphic objects and non-graphic entities (such as layers, line types, and text styles) constitute GstarCAD graphics.

7.4. GcGi

GcGe library provides the graphic interface which is used to draw CAD entities.

7.5. GcGe

GcGe library provides some geometric tools (e.g., vector and matrix) to perform the 2D and 2D geometric operations, it also provides the basic geometric objects, such as point, curve and surface.

8. Copyright

Copyright reserved: Gstarsoft Co.,Ltd

Usage License: Allows copying, quote any part of this document. Changes of any part of this document are forbidden without authorization. Please keep this statement when copying and quoting it, otherwise it will be held liable.

* AutoCAD® is a product of Autodesk® Company which is one of the CAD software solution providers. ARX is an abbreviated name of ObjectARX, which is AutoCAD® Runtime eXtension, and C++ SDK provided by Autodesk®.

GstarCAD 2022

https://www.gstarcad.net/

Gstarsoft