图标

描述已自动生成

**PQ Kit部署说明文档**

文档版本：V3.5

修订时间：2024.03.22

# 产权说明

注意：此处包含的所有信息均为华航唯实机器人科技股份有限公司的财产。未经华航唯实机器人科技股份有限公司的事先书面许可，本出版物的任何部分（无论是硬拷贝还是电子形式）均不得复制、存储在检索系统中，或以任何形式或通过任何方式如电子、机械、影印、录音、或以其他方式传输。请注意，即使未与包含最终用户许可协议的软件一起分发，本指南中的内容仍受版权法保护。PQ和PQArt的标志是华航唯实机器人科技股份有限公司的注册商标。所有其他商标均为其各自所有者的财产。

本出版物和此处的信息按原样提供，仅供参考，如有更改，恕不另行通知，且不应被视为华航唯实公司的承诺。华航唯实公司明确声明对可能出现在本指南所载信息中的任何错误或不准确之处不承担任何责任或义务，对本文件不作任何（明示、暗示或法定）保证，并明确表示不承担任何关于适销性、适用于特定目的和不侵犯第三方权利的保证。

# 更新记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本编号 | 版本日期 | PQ Kit支持版本 | 更新说明 |
| V3.3 | 2023-07-26 | 9.1.0.6105以及上 | 原2.13拆成2.13、2.14，新增2.2 |
| V3.4 | 2023-10-25 | 9.2.0.6267以及上 | 解决kit目录依赖问题***<重大更新>***，新增2.1.5、2.1.6 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 版本说明

PQKit分为开发版、部署版两种版本。开发版为开发者所用版本，部署版为用户打包部署自己开发完的上层应用所用版本。

开发版与部署版的区别如下：

* 开发版SDK安装包中除了PQKit组件文件以外还包含了各种编程语言下集成PQKit的Sample源码及可执行文件、PQKit部署文档、PQApi手册等。部署版仅仅提供客户安装部署可用的PQKit组件文件安装包。
* 开发版PQKit仅供开发过程使用，可以使用第三方开发工具进行开发、调试等开发工作。部署版仅供程序运行使用。
* 开发版除了提供离线编程应用技术方面技术支持外，还提供开发技术咨询服务，部署版不提供技术支持。

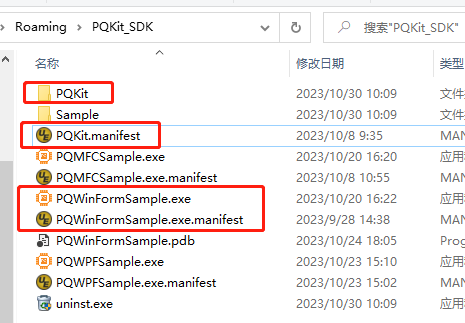
# PQ Kit安装部署

## 开发版安装

双击开发版安装包即可安装PQKit开发版，首次安装可选目录，二次安装会默认覆盖到上一次的安装目录下。



PQKit\_SDK开发版安装结果如下图所示

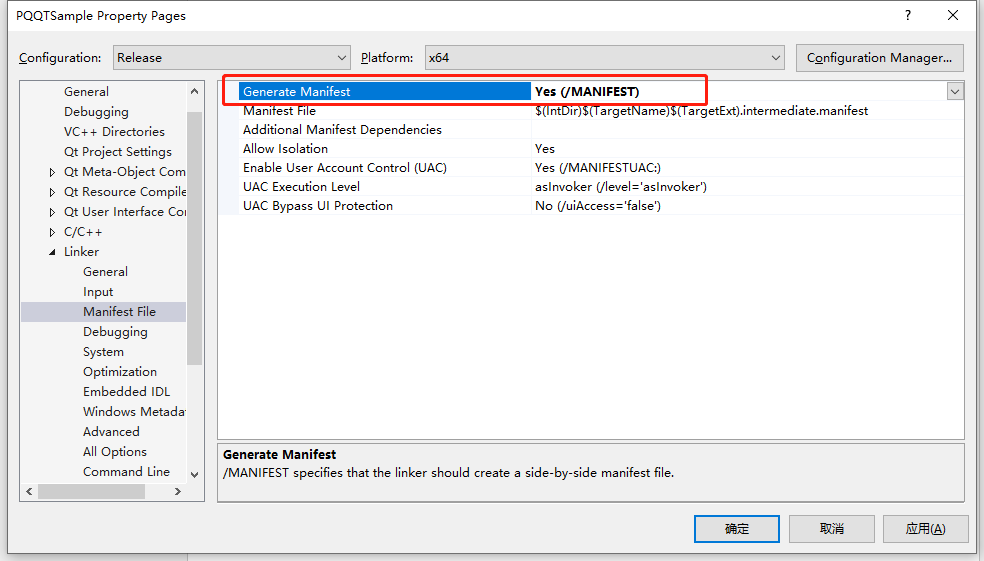


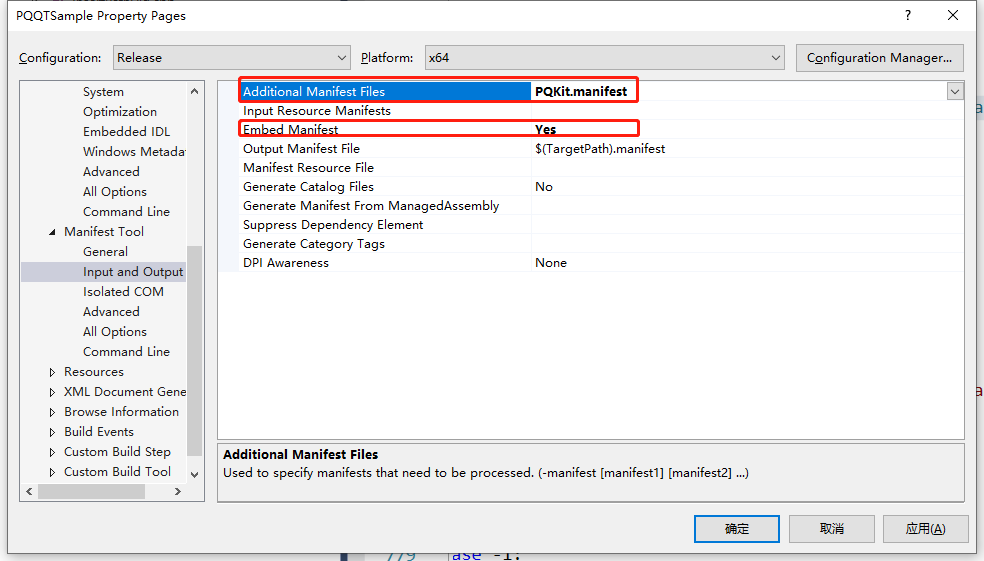
其中PQKit、PQKit.manifest不可更改名称、内容。Sample为提供的集成PQKit的各种开发语言的源码工程，PQMFCSample.exe、PQWinFormSample.exe、PQWPFSample.exe、PQQTSample.exe为相应源码工程编译结果。

### 清单文件

每一个集成PQKit的应用程序都需要集成PQKit.manifest清单文件。PQKit.manifest清单文件有两种集成方式：内嵌式、外置式。

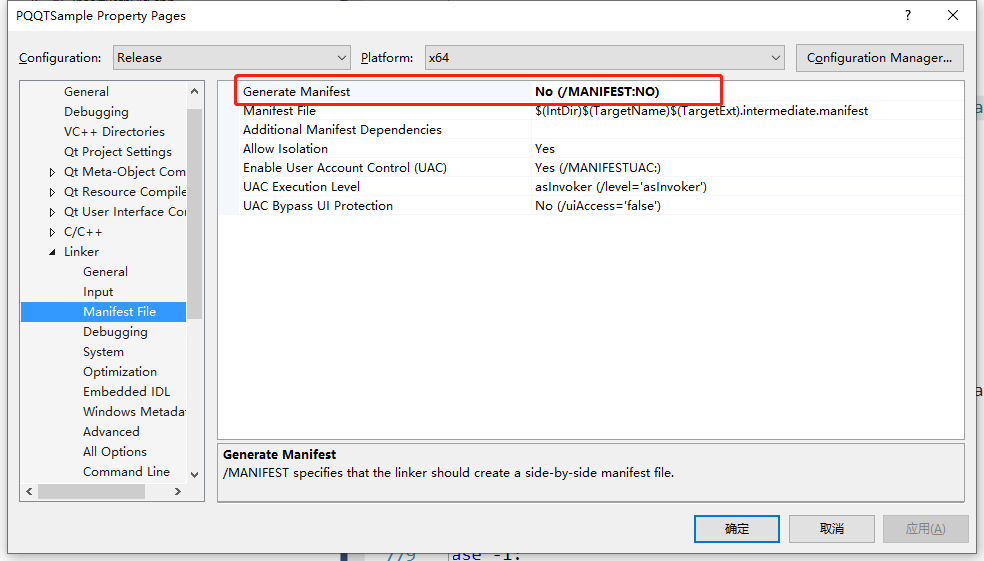
<1>内嵌式，将PQKit.manifest编译到exe中。需要在工程属性中做如下设置。

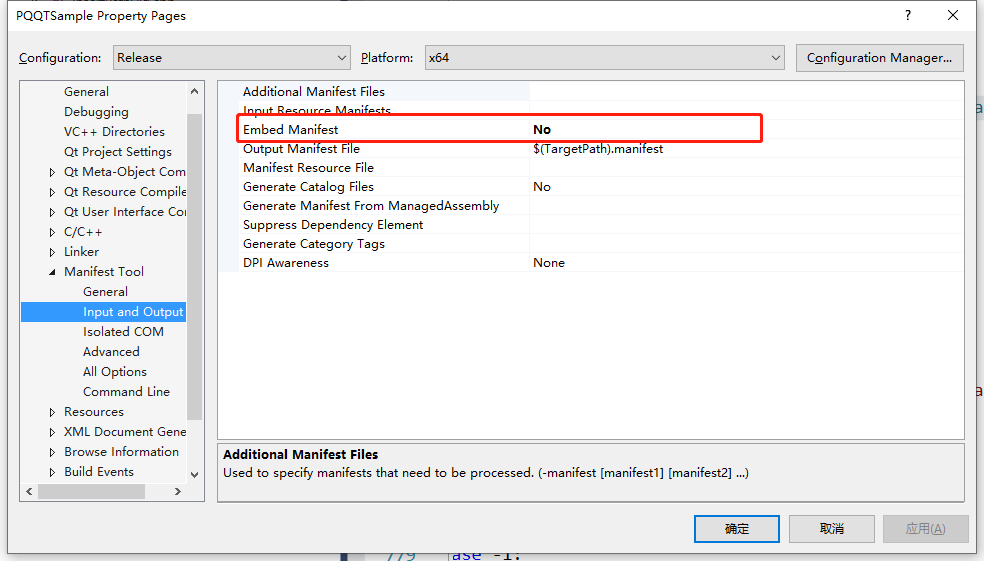




使用内嵌manifest形式，开发的exe只需和PQKit目录在同一文件下即可。

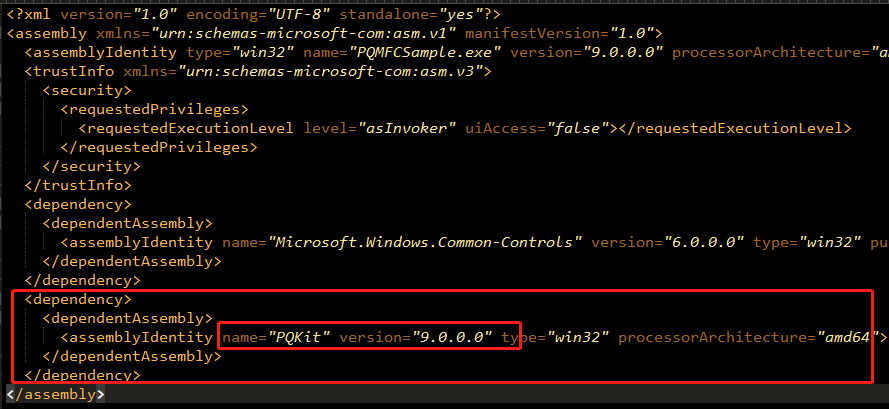
<2>外置式，工程属性设置如下。





外置式需要给开发的exe编写一个manifest文件，并在其中包含PQKit.manifest文件，参考开发版安装目录下的PQMFCSample.exe.manifest等。外置式需要确保开发的集成PQKit的exe、这个exe的manifest以及PQKit目录、PQKit.manifest在同一目录。

|  |
| --- |
| <dependency>  <dependentAssembly>  <assemblyIdentity name="PQKit" version="9.0.0.0" type="win32" processorArchitecture="amd64"></assemblyIdentity>  </dependentAssembly>  </dependency> |



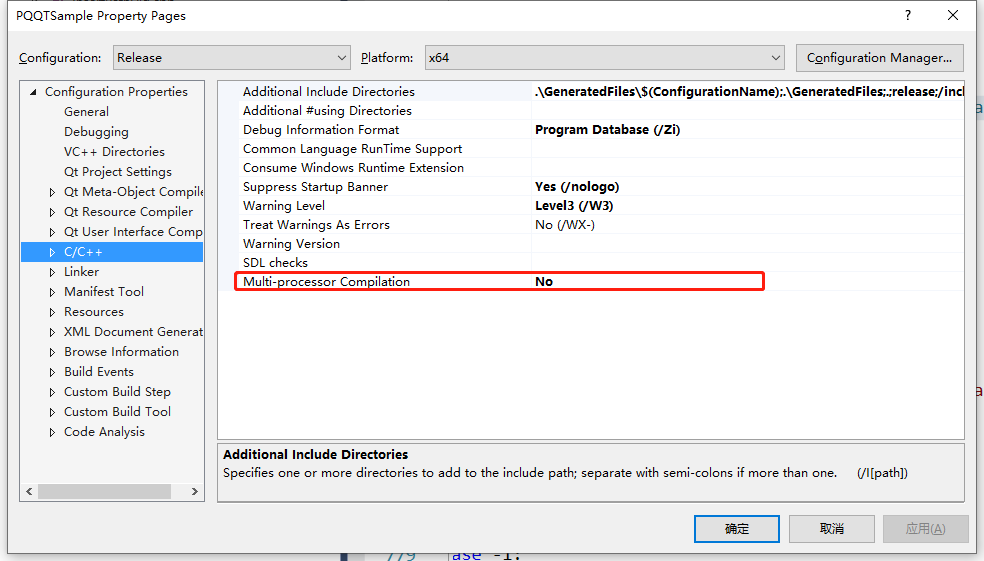
PQKit开发版安装的sample均使用外置式配置。

### RPC.tlb

对于C++语言工程，需要import RPC.tlb。

|  |
| --- |
| #import "RPC.tlb" no\_namespace, named\_guids, raw\_interfaces\_only, raw\_native\_types |

工程属性需做如下设置。

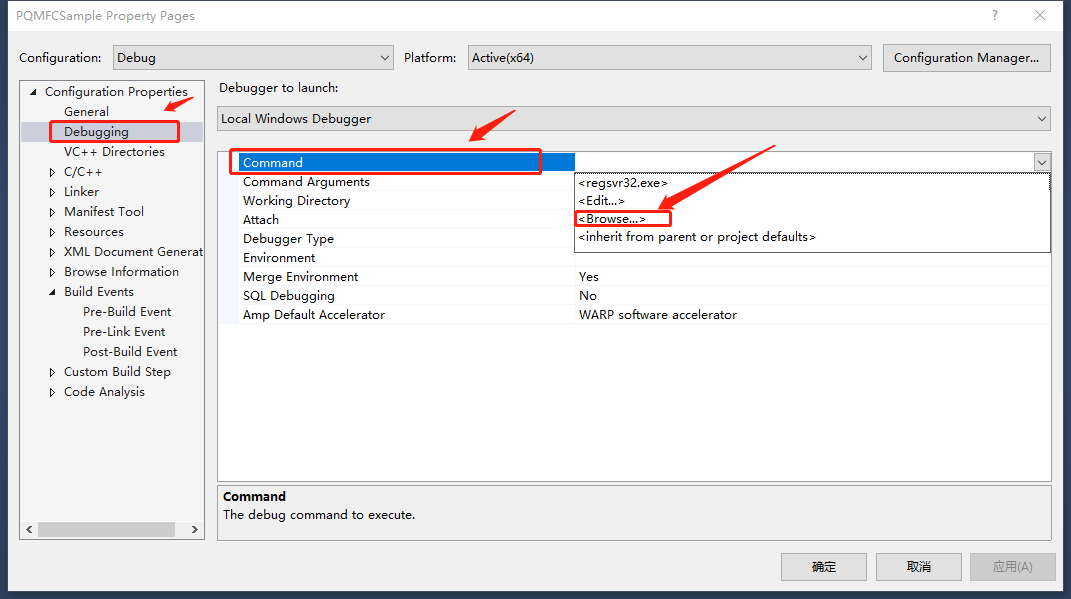


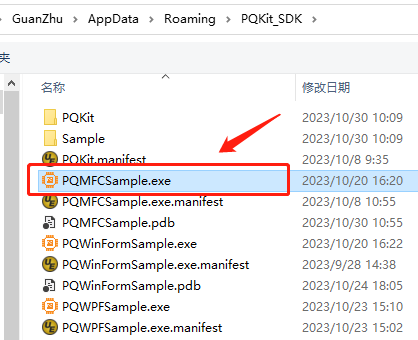
import tlb后工程编译即可生成rpc.tlh文件，在需要用到PQkit组件的地方include rpc.tlh。

## 开发版调试

开发版调试，调试开发者开发的exe即可。分情况举例：

如果要调试PQKit\_SDK安装自带的Sample，Sample工程属性已经设置将其编译结果拷贝至PQKit\_SDK安装跟目录下，即PQKit目录同目录下，调试选择相应的exe即可。





如果开发者要调试一个已有工程或者新建工程，需要确保被调试的exe同目录下有PQKit环境，即PQKit目录以及PQKit.manifest（视使用内嵌还是外置形式而定）。例如，如果新建一个工程，直接启动调试的话，默认是启动编译结果目录下的exe，通常这个目录下是没有PQKit环境的，需要将上述文件拷贝至该目录，如使用外置manifest还需要为该exe编写清单文件。

## 部署版部署

在开发者发布的安装程序中调用PQKit部署版安装包，通过命令行参数/P传入目标安装位置即可。如PQArt\_Kit\_DEP\_zh-cn\_Setup\_x64\_6436.exe /P="D:\test" /S。

# PQ Kit使用

使用PQKit的大致步骤如下：

* 将相关文件加入目标工程，如C#工程需要使用PQKitPlugin.cs，C++工程需要使用RPC.tlb。提及的这2个文件在开发版安装目录Sample下。
* 在工程中创建IPQPlatformComponent对象。
* 从IPQPlatformComponentCallBack派生类，实现接口上的方法。
* 调用PQKit初始化函数启动内核，获取三维显示界面显示在目标区域，PQKit的其他停靠窗口可选。
* 通过IPQPlatformComponent对象调用接口实现功能。

## 初始化

创建IPQPlatformComponent对象后，即可通过该对象调用pq\_InitPlatformComponent启用PQKit内核。

|  |  |
| --- | --- |
| **HRESULT** **pq\_InitPlatformComponent(IPQPlatformComponentCallBack\* pCallBack,int hParentHwnd,BSTR bsName, BSTR bsPWD)** | |
| Parameters： | |
| pCallBack | 回调接口指针，用于接收初始化状态 |
| hParentHwnd | 用于显示组件的窗口句柄 |
| bsName | 登录kit的用户名 |
| bsPWD | 登录kit的用户密码 |

初始化的结果会通过Fire\_Initialize\_Result回调函数告知调用者。

|  |  |
| --- | --- |
| **void Fire\_Initialize\_Result(int lResult)** | |
| Parameters： | |
| 1 | 初始化成功 |
| -1 | 一般性错误 |
| -2 | 初始化超时 |
| -3、-4、-6 | 加载组件动态库错误 |
| -5 | 用户校验失败 |
| -7 | 组件重复初始化 |
| -9、-10、-11、-12 | 内部数据错误 |

如果是账号登录错误，会通过Fire\_Login\_Result回调函数告知调用者。

|  |  |
| --- | --- |
| **void Fire\_Login\_Result (int lResult)** | |
| Parameters： | |
| lResult | Kit登录状态返回值 |
| 返回值状态如下： | |
| 1 | 登录成功 |
| -1 | 密码错误 |
| -2 | 用户不存在 |
| -4 | 用户已过期 |
| -6 | 网络连接失败 |
| -7 | 用户未激活 |
| -14 | 账号不匹配 |

## 三维区域

获取PQ Kit三维显示主区域，将其显示在集成PQKit的目标区域中。

|  |  |
| --- | --- |
| **HRESULT pq\_GetPlatformView(int\* hView)** | |
| Parameters： | |
| hView | 返回Kit三维显示区域句柄 |

## 停靠窗口

PQ Kit中机器人加工管理树、调试窗口、输出窗口、机器人控制窗口可以由集成者控制显示位置及形式。这4个窗体有两种显示方式，其一是内嵌模式，即这4个窗口和PQKit三维区域一体化显示在目标区域；其二是浮动模式，即这4个窗口可以和PQKit三维区域分割显示在不同的显示区域。

a. 机器人加工管理树

|  |  |
| --- | --- |
| **HRESULT pq\_GetModelTreeView (int\* hView)** | |
| Parameters： | |
| hView | 返回机器人加工管理树句柄 |

b. 调试窗口

|  |  |
| --- | --- |
| **HRESULT pq\_GetDebugTreeView (int\* hView)** | |
| Parameters： | |
| hView | 返回调试窗口句柄 |

c. 输出窗口

|  |  |
| --- | --- |
| **HRESULT pq\_GetOutPutView (int\* hView)** | |
| Parameters： | |
| hView | 返回输出窗口句柄 |

d. 机器人控制窗口

|  |  |
| --- | --- |
| **HRESULT pq\_GetRobotControlView (int\* hView)** | |
| Parameters： | |
| hView | 返回机器人控制窗口句柄 |

### 显示形式设置

集成者可通过IPQPlatformComponent对象调用App\_get\_option方法获取IPQComponentOption对象，调用IPQComponentOption对象的 Set\_defaultdock\_embedded设置内嵌还是浮动。传TRUE即设置内嵌，传FALSE即设置浮动。

### 显示设置

停靠窗口内嵌模式下，可通过Doc\_set\_dockwindow\_visible接口设置其可见性。

|  |  |
| --- | --- |
| **HRESULT Doc\_set\_dockwindow\_visible(INT i\_nDockType, BOOL i\_bVisible**) | |
| Parameters： | |
| i\_nDockType | 停靠窗口类型  1 机器人加工管理树  4 机器人控制窗口  7 调试窗口  8 输出窗口 |
| i\_bVisible | TRUE可见 FALSE不可见 |

## 关闭组件

通过调用pq\_CloseComponent接口可以关闭PQKit组件。

## PQKit Option

用户可通过IPQComponentOption来设置一些选项参数。需要通过IPQPlatformComponent对象调用App\_get\_option方法获取IPQComponentOption对象。目前可供设置选项的有停靠窗口的显示形式、是否使用PQKit cef环镜。

|  |  |
| --- | --- |
| **HRESULT Set\_defaultdock\_embedded (BOOL i\_bDefaultDockEmbedded)** | |
| Parameters： | |
| i\_bDefaultDockEmbedded | TRUE内嵌 FALSE浮动 |

|  |  |
| --- | --- |
| **HRESULT Set\_enable\_cefcontext (BOOL i\_bEnableCefContext)** | |
| Parameters： | |
| i\_bEnableCefContext | TRUE启用 FALSE禁用 |

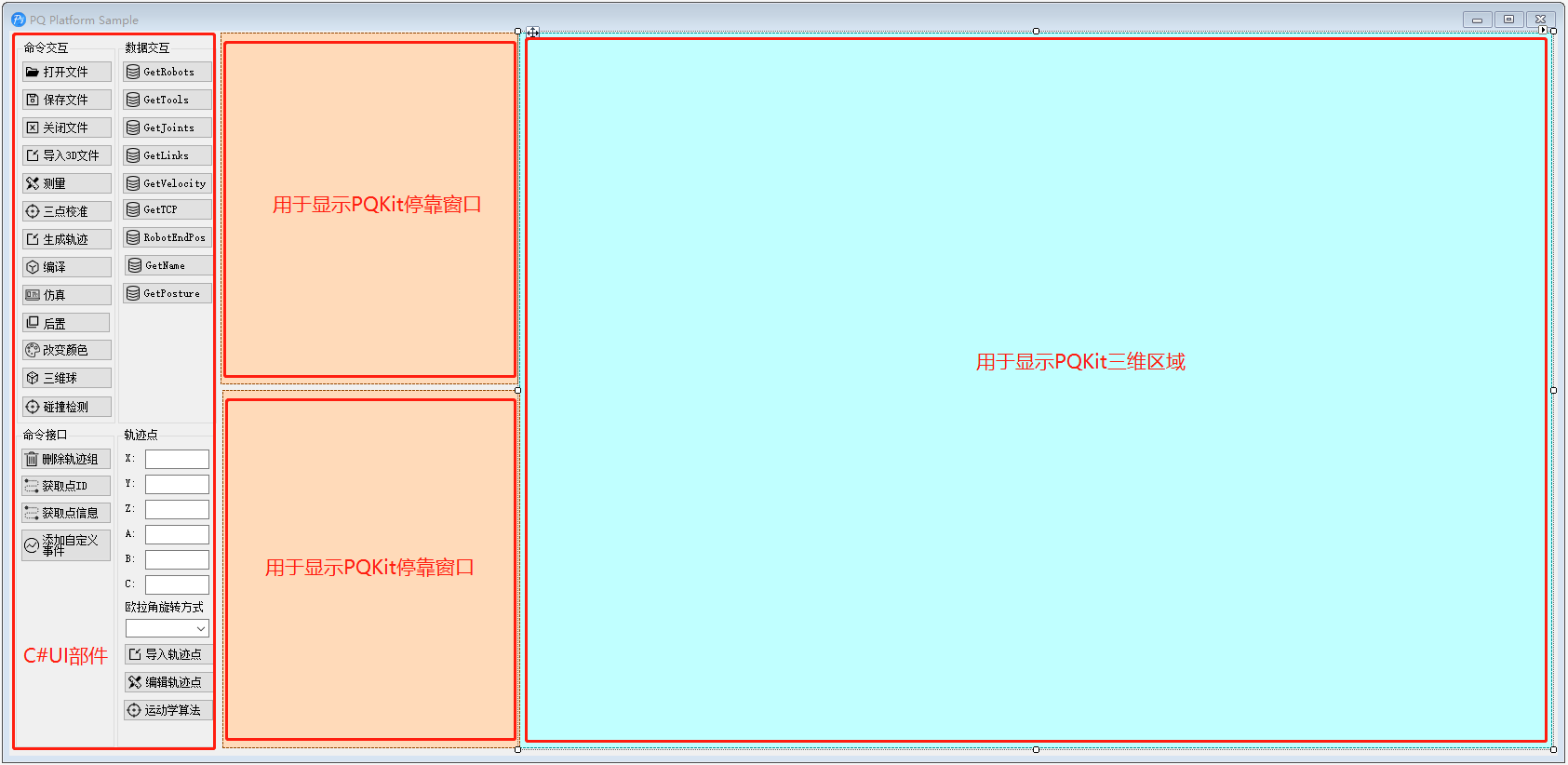
# C#范例

3.1 新建C# WinForm工程。

3.2 将PQKit\_SDK\Sample\include\C#下PQKitPlugin.cs加入工程，并添加引用。有关IPQPlatformComponent、IPQPlatformComponentCallBack的定义均在其中。

3.3 在窗体设计中，拖入一个Panel或者其他Container作为组件的显示容器。

3.4 新增PQPlatformComponentCallBack类实现IPQPlatformComponentCallBack接口方法。（详见Sample）



3.5 在窗体Load函数中新建IPQPlatformComponent接口对象、回调接口对象PQPlatformComponentCallBack，启动线程InitializeKit。

|  |
| --- |
| //PQKit初始化委托  public delegate void OnStartKitInitialize();  //PQKit初始化完成委托  public delegate void OnKitInitialized();  //  IPQPlatformComponent m\_ptrKit = null;  //  PQPlatformComponentCallBack m\_CallBack = null;  private void Form1\_Load(object sender, EventArgs e)  {  //创建内核对象  CPQKitPlugin pqKitPlugin = new CPQKitPlugin();  m\_ptrKit = pqKitPlugin.LoadPQKit();  if (null == m\_ptrKit)  {  MessageBox.Show("PQKit load error! \n请排查当前启动或调试exe同目录是否有PQKit环境。\nPQkit环境包含PQkit文件夹、PQKit.manifest文件、启动或调试exe的manifest文件。");  }  //创建callback对象  m\_CallBack = new PQPlatformComponentCallBack();  m\_CallBack.SetMainWindow(this);  //在线程中初始化kit  Thread workThread = new Thread(new ThreadStart(InitializeKit));  workThread.SetApartmentState(ApartmentState.MTA);  workThread.IsBackground = true;  workThread.Start();  //WinForm程序自己的后续业务逻辑  InitPostureTypeUI();  //  lbuCaller = new OnKitLBtnUP(KitLBtnUpEvent);  //kit 绘图区右键菜单响应  pmCaller = new OnKitPopMenu(KitPopMenuEvent);  }  private void InitializeKit()  {  this.Invoke(new OnStartKitInitialize(KitInitializedFuc));  }  public void KitInitializedFuc()  {  struct\_PQKitOption strOption;  strOption.nEmbedded = 0;  m\_ptrKit.PQAPISetOption(ref strOption);  //此处为内核组件的初始化，需依次传入内核组件回调接口，用于显示内核组件的句柄，用户名、密码  int nHWND = (int)panel1.Handle;  string strName = "";  string strPWD = "";  try  {  m\_ptrKit.pq\_InitPlatformComponent(m\_CallBack, nHWND, strName, strPWD);  }  catch (Exception ex)  {  string strError = "PQKit init info:\n";  strError += ex.Message;  strError += "请参考Fire\_Initialize\_Result、Fire\_Login\_Result函数中的具体lResult值。";  MessageBox.Show(strError);  }  } |

在KitInitializedFuc函数中调用pq\_InitPlatformComponent初始化组件，需要输入用户名、密码(用户名、密码需要申请开通)。

在PQPlatformComponentCallBack类Fire\_Initialize\_Result方法中获取初始化状态。如果初始化成功则将WinForm中的显示容器设为组件显示区域的Parent并移动位置，如果初始化失败，查看失败错误码具体分析即可。

具体细节详见Sample中Winform工程、WPF工程。

# C++范例（MFC）

4.1、新建MFC工程，设计界面

4.2、将PQKit\_SDK\Sample\include\C++下RPC.tlb拷贝至新建工程目录

4.3、import tlb，编译工程（生成rpc.tlh）

4.4、新增CPQComponentCall类实现IPQPlatformComponentCallBack接口方法。（详见Sample）

4.5、在对话框OnInitDialog()中创建PQKit对象。

|  |
| --- |
| *CoInitialize*(*NULL*);  HRESULT hr = m\_ptrComponent.CoCreateInstance(\_\_uuidof(PQPlatformComponent));  if (S\_OK != hr)  {  MessageBox(L"PQKit创建失败！\n请排查当前启动或调试exe同目录是否有PQKit环境。\nPQkit环境包含PQkit文件夹、PQKit.manifest文件、启动或调试exe的manifest文件。", L"PQKit", MB\_OK);  return FALSE;  }  hr = m\_ptrComponent->pq\_InitPlatformComponent(&m\_ptrComponentCall, (int)GetDlgItem(IDC\_STATIC\_CANVAS)->GetSafeHwnd(),  L"", L"");  if (S\_OK != hr)  {  MessageBox(L"PQKit初始化失败！\n请参考Fire\_Initialize\_Result或Fire\_Login\_Result函数中的具体lResult值。", L"PQKit", MB\_OK);  return FALSE;  } |

4.6、在Fire\_Initialize\_Result的处理逻辑中判断初始化结果，初始化成功后获取三维区域句柄、停靠窗口句柄自行显示在目标区域。

# C++范例（QT）

要求：Visual Studio 2015及以上版本、QT 5.12.8。

5.1、新建QT MainWindow工程，设计应用程序界面布局。

5.2、将PQKit\_SDK\Sample\include\C++下RPC.tlb拷贝至新建工程目录。

5.3、import tlb，编译工程（生成rpc.tlh），在使用到接口的地方需要include rpc.tlh。

5.4、新增CPQKitCallback类实现IPQPlatformComponentCallBack接口方法。（详见Sample）

5.5、在MainWindow类构造函数中创建PQKit对象，并创建QThread线程，在线程中调用pq\_InitPlatformComponent方法初始化PQKit组件。其中账户名、密码需要申请方可获取。

|  |
| --- |
| void MainWindow::InitPQKit()  {  //  CoInitializeEx(0, COINIT\_APARTMENTTHREADED);  HRESULT hr = m\_ptrKit.CoCreateInstance(\_\_uuidof(PQPlatformComponent));  if (S\_OK != hr)  {  QString strError = QString::fromLocal8Bit("PQKit创建失败！\n请排查当前启动或调试exe同目录是否有PQKit环境。\nPQkit环境包含PQkit文件夹、PQKit.manifest文件、启动或调试exe的manifest文件。");  QMessageBox::information(NULL, "PQKit", strError, QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);  }  //  m\_ptrKitCallback = new CPQKitCallback(this);  connect(m\_ptrKitCallback, &CPQKitCallback::signalInitializeResult, this, &MainWindow::OnInitializeResult);  connect(m\_ptrKitCallback, &CPQKitCallback::signalRunCMDResult, this, &MainWindow::OnRunCMDResult);  connect(m\_ptrKitCallback, &CPQKitCallback::signalGetDataResult, this, &MainWindow::OnGetDataResult);  connect(m\_ptrKitCallback, &CPQKitCallback::signalRaiseDockwindow, this, &MainWindow::OnRaiseDockwindow);  connect(m\_ptrKitCallback, &CPQKitCallback::signalLoginResult, this, &MainWindow::OnLoginResult);  connect(m\_ptrKitCallback, &CPQKitCallback::signalPathGenerateResult, this, &MainWindow::OnPathGenerateResult);  connect(m\_ptrKitCallback, &CPQKitCallback::signalElementPickup, this, &MainWindow::OnElementPickup);  connect(m\_ptrKitCallback, &CPQKitCallback::signalLButtonUp, this, &MainWindow::OnLButtonUp);  connect(m\_ptrKitCallback, &CPQKitCallback::signalRButtonUp, this, &MainWindow::OnRButtonUp);  connect(m\_ptrKitCallback, &CPQKitCallback::signalMenuPop, this, &MainWindow::OnMenuPop);  //  m\_ptrKitInitThread = new PQKitInitThread;  connect(m\_ptrKitInitThread, &PQKitInitThread::signalInitializeKit, this, &MainWindow::OnInitializeKitThread);  m\_ptrKitInitThread->start();  }  void MainWindow::OnInitializeKitThread()  {  //initialize pqkit  CComBSTR bsName = L"ra\_develop\_wac";  CComBSTR bsPWD = L"develop\_wac";  HRESULT hr = m\_ptrKit->pq\_InitPlatformComponent(m\_ptrKitCallback, (int)(this->winId()), bsName, bsPWD);  if (S\_OK != hr)  {  QString strError = QString::fromLocal8Bit("PQKit初始化失败！\n请参考Fire\_Initialize\_Result或Fire\_Login\_Result函数中的具体lResult值。");  QMessageBox::information(NULL, "PQKit", strError, QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);  }  } |

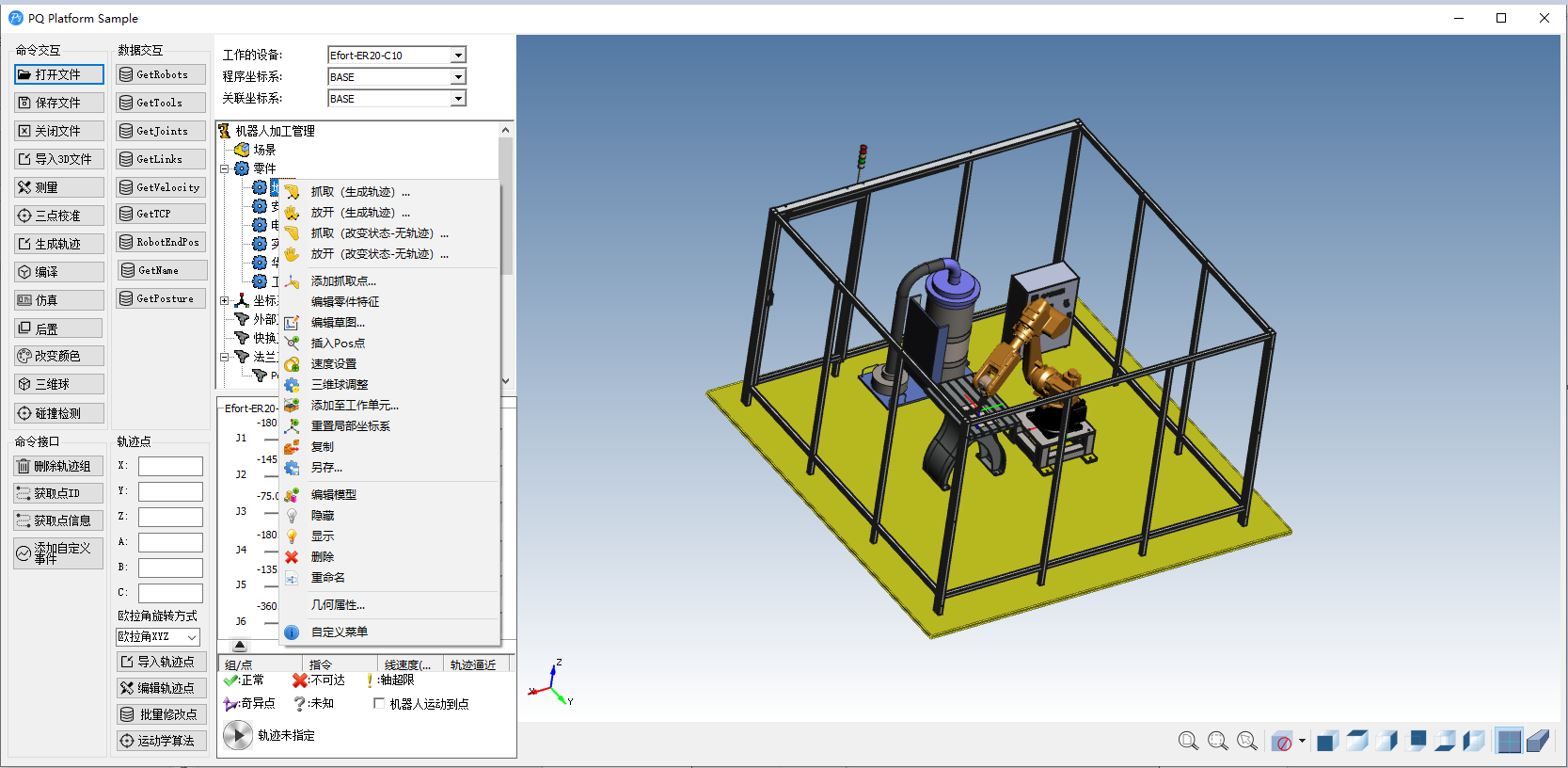
5.6、根据Fire\_Initialize\_Result的返回结果判断PQKit初始化结果，如初始化成功则进行显示PQKit主窗口以及停靠窗口相关逻辑。

5.7、QT作为集成应用的图标设置，除了QT正常的应用程序图标设置外，还需要同一图标的icon版本命名为app.ico至于exe同目录。

# PQ Kit自定义菜单

PQ Ki支持自定义菜单，即用户可以通过自定义菜单功能对PQ Kit的右键菜单进行自行裁剪，以满足不同用户的定制化需求。

自定义菜单的用户行为可以发生在PQ Kit SDK开发者层面，即开发者定制菜单后，发布自己的产品，PQ Kit的最终用户能看到的就是裁剪后的菜单。也可以发生在PQ Kit的最终用户层面，即开发者不裁剪菜单，最终用户去裁剪菜单。



在自定义菜单功能中，可以添加某个命令（某一类型对象支持的命令）至菜单中，也可以将某个命令移出菜单。可以新建分组并添加命令、也可以将分组内的某个命令移出。可以上下移动命令改变其在菜单中的顺序。

6.1 自定义菜单界面介绍

自定义菜单中，顶部下拉列表显示所有支持更改的菜单，并会自动匹配用户选中的菜单。中间左侧列表控件分别显示选中菜单下所有的菜单命令。中间右侧列表控件显示选中菜单当前的组织结构。例如，用户在机器人右键菜单中单击自定义菜单，那么顶部下拉列表会自动选中机器人菜单，中间左侧列表控件显示所有机器人菜单命令，右侧列表控件显示当前机器人右键菜单组织结构。



6.2 添加、删除

6.2.1删除

右侧列表选中，单击删除按钮，即可将该命令从现有菜单中删除。

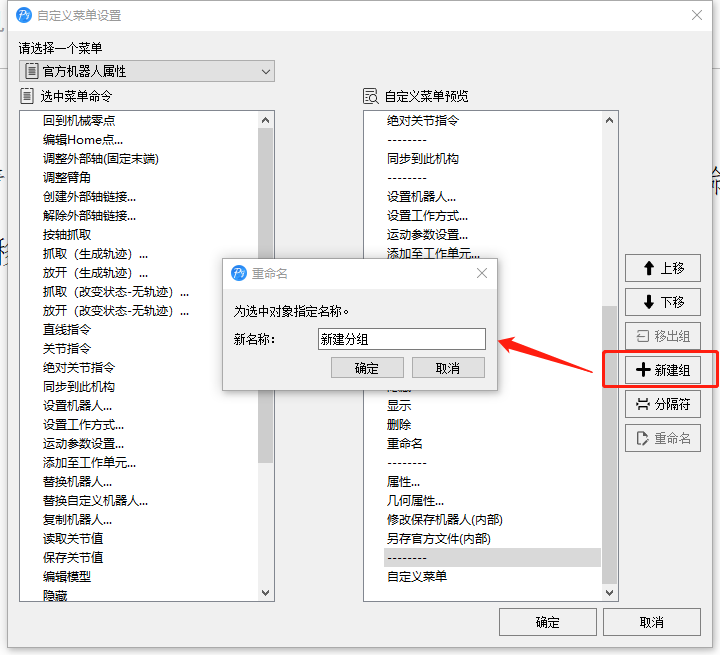
6.2.2添加

左侧选中某个命令，单击添加按钮，即可将该命令加入现有菜单。如添加时右侧选中了某个命令，则将左侧选中命令加到右侧选中命令之后，如添加时右侧选中了某个分组，则将左侧选中命令加到右侧选中分组中。

6.3 上移下移

右侧选中了某个命令或者分组，单击上移、下移按钮，可更改选中命令或者分组在现有菜单中的位置。

6.4分组



单击新建组按钮可创建新的命令分组，可以在创建时为分组指定名称，也可以随时选中分组进行重命名操作。

分组新建后，右侧列表选中该分组，从左侧列表中选择命令，单击添加按钮，即可将该命令加入新建分组中。

选中右侧列表分组下的命令，单击移出组按钮，即可将选中命令移出该分组。移出的命令将放置于顶层菜单末尾。

6.5分隔符

即为分隔符，单击分隔符按钮，可向现有菜单中添加一条分隔符。如添加时右侧选中了某个命令，则将分隔符加到右侧选中命令之后，如添加时右侧选中了某个分组，则将分隔符加到右侧选中分组中。

## 部署自定义菜单

开发版裁剪的自定义菜单文件保存在以下文件：C:\Users\\*\*\*\*\AppData\Local\CHL\PQKit\zh-CN\Config\CustomizeUI.xml，\*\*\*\*部分换成开发机PC名。部署时只需要将CustomizeUI.xml文件和PQKit部署版的安装包放在同一目录即可。