测量仪器软件中的 Qt 插件信号槽通信设计

马风军 [1]
MA Fengjun

摘 要 Qt 作为跨平台开发平台，凭借丰富的类库、开发工具及通用软件框架，能够支持插件与应用程序的快速开发，在多个领域应用广泛。其插件包括 Creator 插件、设计师控件、应用程序插件等类型，其中应用程序插件具备最高灵活性，但现有 Qt 资料仅涵盖插件基本功能的设计方法，未涉及基于信号槽的插件间通信能力实现方案。针对电子测量仪器软件功能复杂，插件开发模式下需满足插件间灵活通信的实际需求，文章以接口为切入点，先分析插件所需的具体通信能力，进而提出插件信号槽通信的设计方案，包括信号槽通信机制的增设与通信数据格式的设计，为解决电子测量仪器插件间高效通信问题提供技术支撑。

关键词 接口；插件；信号槽；Qt

doi：10.3969/j.issn.1672-9528.2025.12.023

0 引言

插件是一种遵循一定规范的应用程序接口编写出来的程序，其只能运行在程序规定的系统平台下，而不能脱离指定的平台单独运行。因为插件遵循接口规范，并且为动态库形式，因此其易于升级维护，更有利于集成在应用程序中扩展新功能，具有广泛的应用。现在的电子测量仪器功能复杂，并且新的功能还在不断增加中，因此也逐渐采用插件架构开发，以保证仪器软件架构的稳定性和各功能模块的独立性。

在跨平台的开发平台 Qt 中，也支持开发插件，其支持的插件种类有 Qt IDE 自身的 Creator 插件，有支持特定应用场景的设计师控件、QML 等插件，也有灵活性最高的应用程序插件。应用程序插件在 Qt 的资料中只介绍了如何设计接口、如何设计插件，这些设计仅满足实现插件这一形式，可将插件的源代码编译为动态库，但并未介绍如何设计插件的通信能力。当设计插件架构时，就对插件通信提出了要求，因为在插件架构中，应用程序的主程序仅仅是启动器和插件的容器，功能主要由各种插件实现，因此将插件组合应用于复杂的业务框架时，就需要插件具备通信能力，因此设计插件的通信功能是开发插件架构的关键技术之一。

1 信号槽通信的技术特点分析

在 Qt 中，通信分为事件与信号槽，二者分析如表 1 所示。

表 1 事件与信号槽通信特点对比

比较项	事件	信号槽
适用范围	用于抽象的类（Class），而不是个体	用于类的具体对象变量，而不是抽象的类
通信需求方	发送者	接收者
通信发送方	主动发起，必须知道接收方才能使用	无需知道接收方就能使用
通信接收方	不知道通信的发送方，除非自定义通信信息中增加此信息	无需知道发送方，但可以知道
耦合度	较高	较低
通信发送方法	基于系统的方法，QCoreApplication::postEvent()	自定义
通信接收方法	系统提供	自定义
通信参数	基于系统提供的 QEvent	自定义
通信处理方法	基于系统提供的方法 QObject::event()	自定义，实为信号接收方法

通过比较两种方法可知，信号槽具备较强的灵活性，这种灵活性是基于对象的个体而不是抽象的类，而架构是一种抽象的模型设计，需要基于类进行通信，因此信号槽在架构设计中作为事件通信的一种补充，主要应用于不依赖接收方的广播通信，将业务从已知的业务领域模型扩展至未知的领域中。

插件架构内需遵循多项设计规范，其中包括通信规范。信号槽过多的灵活性与规范性之间需要协调平衡，才能最大程度地发挥插件架构的优势，因此需要针对信号槽的技术特

1. 中电科思仪科技股份有限公司 山东青岛 266555

点与插件架构的需求进行有针对性的设计。

2 建立信号槽通信机制

分析了信号槽通信的技术特点之后，设计中还需要满足如下技术约束：

（1）发射信号与接收信号的接口类都必须是 QObject 的派生类；

（2）通信功能需要在接口类中设计，而不是在插件类中；

（3）通信要能支持跨线程；

（4）Qt 自身要求 QObject 不支持双重继承；

（5）要规范插件的信号槽通信，在灵活性的基础上避免失控。

基于此分析，接口类必须从 QObject 派生 [1]，而不是留给插件类基于 QObject 与接口类进行双重继承，但代价是接口类的类关系只能为单根继承，不能是多重继承。单根继承可通过充分的分析与设计予以实现。同时，为了支持跨线程通信，接口类中需包含一个专用于通信的 QObject 类对象（本文简称为通信对象），该通信对象不能与插件对象在同一个线程中，应该在子线程中创建，所以接口类中还需要有一个线程类 QThread 的对象（本文称为通信子线程对象），专用于在子线程中创建通信对象，其之间的关系如图 1 所示。

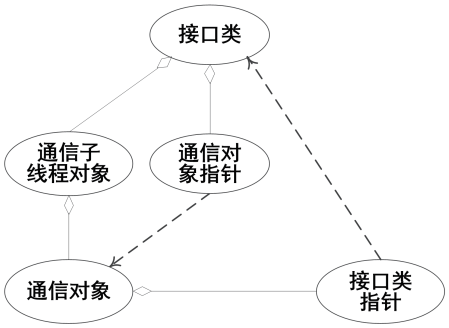

图 1 信号槽机制中各对象关系图

接口类拥有一个负责创建子线程的通信子线程对象与一个指向通信对象的指针，而不是直接拥有通信对象，否则就无法实现跨线程的通信；同时接口类具有专门发射信号的方法与接收信号的方法，即信号收发方法不能由用户自定义，必须使用插件架构提供的方法，用户自定义的通信含义在信号传递的参数中体现，而不是通过信号的函数名。

通信子线程对象在新的子线程中动态创建出通信对象，通信对象有自己专用的中转信号，既负责与发送接口类的信号绑定，也负责与接收接口类的槽绑定，从而实现跨线程信号槽通信。值得一提的是，通信对象只负责转接信号，并未对信号做任何处理。

由上述分析可知，信号槽通信的关键点有 3 处：实现跨线程、通信规范性、数据格式兼容性。

2.1 实现跨线程

保证跨线程就必须先保证有子线程。通信子线程创建了跨线程通信的资源，其要点在于覆盖 QThread::run() 中的设计 [2]，因为在 Qt 中，只有创建并执行了子线程，才能运行 run() 方法；只有在 run() 中才能保证创建的对象属于新的子线程，所以在 run() 中进行资源的创建与赋值，其流程设计如图 2 所示。

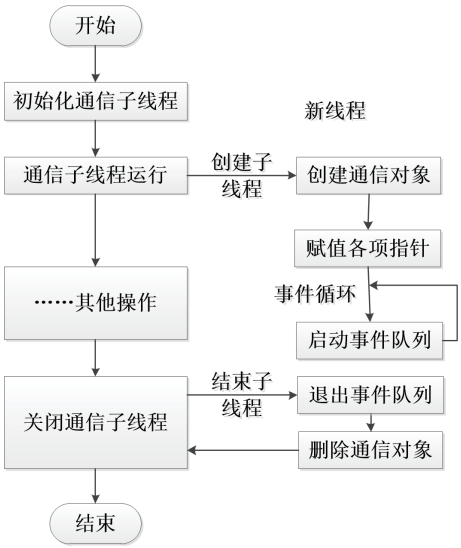

图 2 信号槽机制中各对象使用流程图

接口类在其初始化中，需要初始化通信子线程，例如设置优先级等，然后启动子线程的运行。

当子线程运行后，由系统自动调用通信子线程的 run() 方法，用户可通过覆盖该方法定制自己的线程运行控制过程。在该类中，创建通信对象，并将指针赋值给接口类，然后进入事件队列循环中。至此，事件机制启动完毕，可以收发信号进行通信，因为信号槽的异步、同步通信需要借助事件机制。

当需要卸载插件时，通过接口类的关闭函数，退出通信子线程的事件队列循环，不再处理事件队列，然后给各指针赋空值，取消资源之间的关联，最后删除创建的各对象资源，释放内存。

2.2 实现规范性

规范性的要点将在接口类中实现两个插件的信号槽绑定操作的封装，并且绑定的信号槽由架构给出，无需也不允许用户自定义。在 Qt 的事件机制中，接收的方法由系统指定且系统自动调用，因此用户只需注重发送方。但在信号槽机制中，信号与槽并不是自动连接（除非在 Creator 或 Designer 界面上添加的信号槽），需手动编码连接。连接信号槽时最终需使用 Qt 提供的静态方法 QObject::connect()，该方法的参

数需要指定发送信号的对象的指针与方法（即信号），以及接收信号的对象的指针与方法（即槽）。但为避免信号槽通信失控，此处信号与槽都由架构确定，因此绑定时只需指定收发信号的插件的指针，接口类的绑定方法如下：

qint32 connectMsgToPluginNotifier(const CBaseInterface * kpobjNotifier, Qt::ConnectionType enuType);

其中 kpobjNotifier 即接收信号的插件的指针，enuType 表示信号槽的连接类型，包括异步、同步或直连。

接口类的发射信号的方法为：

void sigMsgNotify(quint32 u32Message, qint64 s64Para, QVariantList objListPara, QVariantMap objMapPara);

接口类的接收信号的槽为：

qint32 onHandleMsgFromPluginNotifier(quint32 u32Message, qint64 s64Para, QVariantList objListPara, QVariantMap objMapPara);

通信对象的中转信号为：

void sigMsgFromPluginNotifier(quint32 u32Message, qint64 s64Para , QVariantList objListPara, QVariantMap objMapPara);

在绑定信号槽时，插件不能直接将自己的信号与接收插件的槽相连，而是将自己的信号与接收插件的通信对象的中转信号相连，且采用用户指定的方式；再将接收插件的通信对象的中转信号与接收插件的槽相连，且采用直连的方式[3]。即信号链路因为中转变延长，但通过中转实现了跨线程。

信号传递的泳道图设计如图 3 所示。

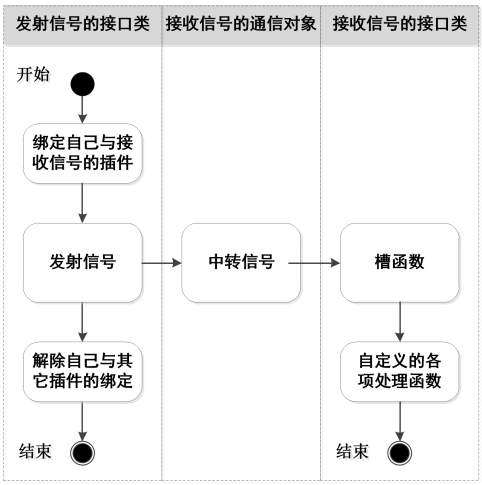

图 3 信号槽传递泳道图

在发射信号之前应先绑定信号槽，若不再使用或卸载插件前应解除绑定。当插件发射信号时，信号先传递给接收插件的中转信号，传递方式在绑定方法指定；信号再从中转信号以直连的方式传递给接收插件的槽做处理，因为第一次传递时已经实现了信号发射连接方式的要求，所以第二次传递采用直连以提高效率。

通过该设计，用户无需考虑收发信号的函数名称与绑定，只需将精力专注于信号携带的参数中，通过这些参数携带自己的业务信息，从而作出识别处理。

2.3 数据格式兼容性

在上文的设计中，两个信号与一个槽的形状完全一致，分别采用了整数与容器，这是考虑既要满足通信的效率，又要兼顾灵活性。效率最高的是用整型数据，灵活性则是采用 QVariant 的相关容器，因为常用的字符串与数组等数据可转为 QVariant，多个 QVariant 可以放入容器中，且自定义的结构体也可通过语法设计转为 QVariant，因此格式主要由这两种组成，具体设计如表 2 所示。

表 2 信号槽通信的参数表

数据名	数据类型	描述
消息	32 位整型	表示具体的消息，类似 Windows 中的消息。
整数参数	64 位整型	8 个字节，可包含各种类型、各种长度的整数、指针。本参数的目的是效率。
列表参数	QVariantList	用于传递数据格式已知的复杂数据，其每一个元素都是 QVariant。
映射参数	QVariantMap	用于传递数据格式未知的复杂数据，其每一个元素都是 QString 与 QVariant 的映射，QString 负责描述 QVariant。

3 总结

经过上述设计的插件信号槽通信功能，满足插件架构中业务扩展的要求，在某嵌入式设备中使用该设计，在不影响原软件运行的前提下实现了功能扩展，并通过了验收测试，验证了设计的有效性。

参考文献：

[1] 维波，栗宝鹃，侯春望 . Qt 5.9 C++ 开发指南 [M]. 北京：人民邮电出版社 ,2020.

[2]LIPPMAN S B, LAJOIE J, MOO B E. C++ Primer(第 5 版) [M].London:Addison-Wesley Professional,2012.

[3]FREEMAN E, FREEMAN E, BATES B. Head First 设计模式 [M]. 北京：中国电力出版社 ,2007.

【作者简介】

马风军（1978—），男，山东济南人，本科，高级工程师，研究方向：电子测量仪器软件架构方向的设计，email: mafengjun@ceyear.com。

（收稿日期：2025-06-27 修回日期：2025-12-05）