**仪表Lab VIEW软件说明文档**

**武汉普赛斯仪表技术有限公司**

本文档为Lab VIEW软件架构详细说明文档，主要说明Lab VIEW软件设计方案及逻辑结构，也为方便熟悉软件结构，指导新产品的软件研发思路制定该文档。

**声明：**本文件所有权和解释权归武汉普赛斯仪表技术有限公司所有，未经武汉普赛斯仪表技术有限公司书面许可，不得复制或向第三方公开。

修订历史记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版次** | **发布日期** | **AMD** | **修订者** | **说明** |
| v1.0 | 2020.04.23 | 首次发行 | 石良胜 |  |
|  |  |  |  |  |
|  |  |  |  |  |

（A-添加，M-修改，D-删除）

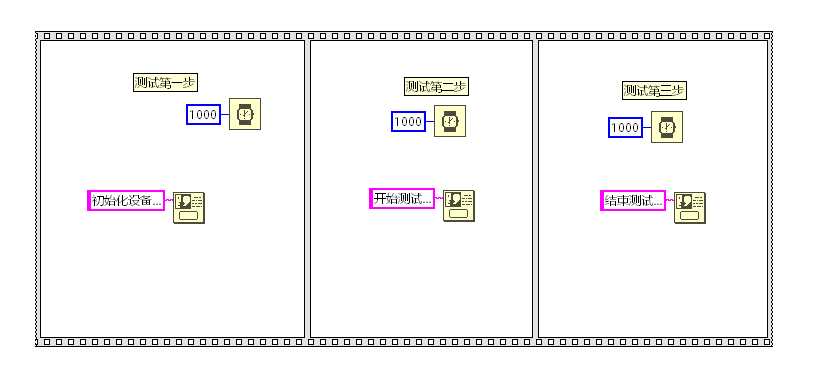
# 需要解决的问题

目前产品对应的LabVIEW上位机软件结构混乱，修改维护不方便，需要梳理重构LabVIEW软件，为了方便后续更新维护以及新产品上位机的开发，因此设计这一套软件框架。传统编程资料多采用顺序结构，修改bug时牵一发而动全身，不利于软件维护，而且代码重复量大，消耗内存多，影响软件开发工作。

# 主流的设计模式

对于较为复杂的项目，在具体编码之前，首先要搭出系统架构，系统架构是流程图代码的体现，好的系统架构可以大大节约系统开发和调试时间，使得逻辑更加清晰。LabVIEW里常见的几种框架如下：

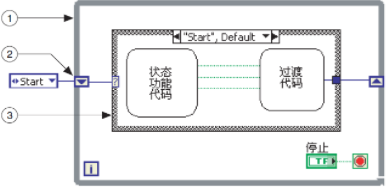
2.1顺序结构。



结构特点：结构简单，程序按照固定的顺序依次执行，但是在很多情况下

顺序模式是静态的，无法满足我们编程的要求；则需要更有效的动态结构来实时改变程序的执行顺序，而且LabVIEW里顺序结构执行效率低，不推荐使用。此时，就引入状态机。

2.2状态机结构。



结构特点：状态机是对系统的一种描述，该结构包含了有限的状态，并

且在各个状态之间可通过一定的条件进行转换。一般可用状态图来对一个状态机进行精确地描述。

在Labview中，任何一个状态机都是由三个基本部分构成：

1）首先外层是一个while循环，用于维持状态机的运行；

2）同时在while循环中包含一个条件结构，条件结构用于对各个不同的状态进行判断；

3）第三个基本部分是移位寄存器，用以将下一个状态机传递到下一次循

环状态判断中。

另外，在一个完整的状态机中，我们一般还会提供初始状态，每一个状态的执行步骤以及下一个状态切换代码等。

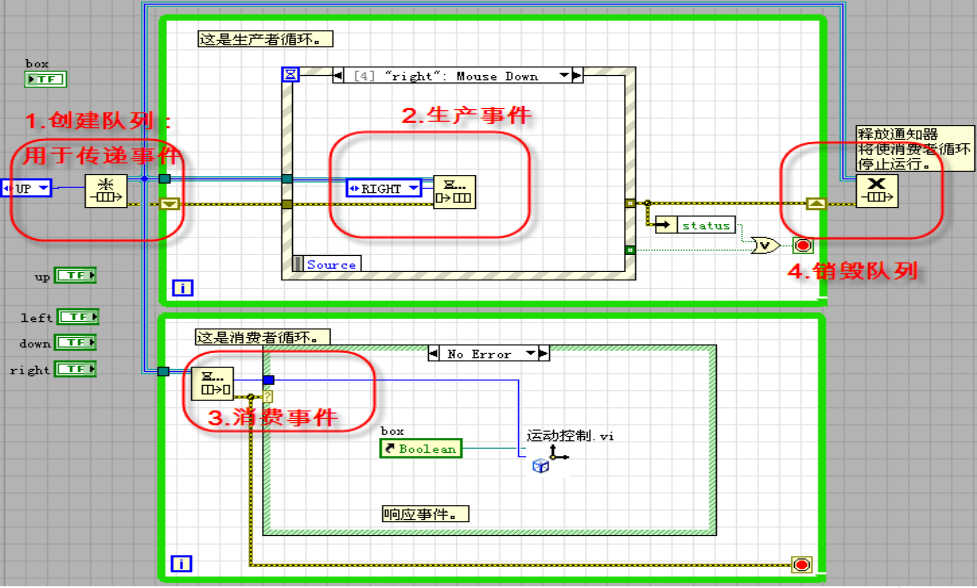
除了前面提到的三个基本部分外，状态机的构建还有一个技巧，就是使用枚举型常量来作为状态机变量，相对于其它数据类型来说，枚举型常量表示了，两组成对的数据，一组是字符串，一组是数值。

由于引入了枚举常量这一有效的状态变量，在后期对状态机的维护中我们会非常轻松，只需要对已经存在的自定义枚举常量进行相应的修改，添加，删除等操作，就能方便地完成状态机中所有状态变量的更新，再对程序框图做必要的修改，就能高效并准确地完成状态机的维护。

2.3生产者消费者结构

许多使用LabVIEW的编程者都知道，LabVIEW是可以多线程编程的，

那么状态机结构只是一种简单的单线程任务，无法满足实际的项目需求。

因此引出生产者消费者结构。

生产者消费者关键在队列，因此学习此结构必须掌握队列的使用。

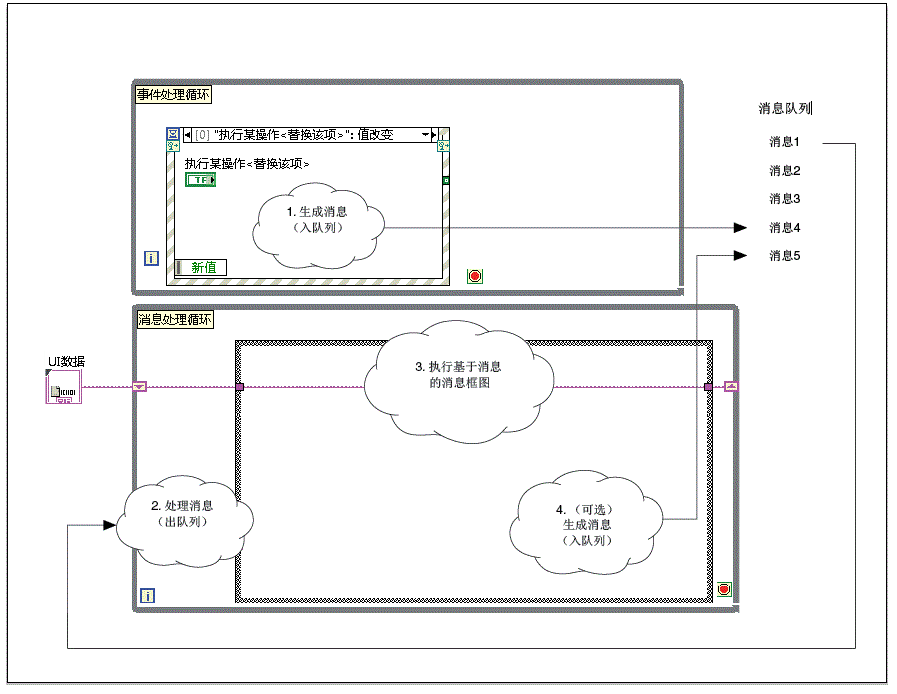
原理分析：

1. 生产者捕获事件，并将数据存储在队列中。
2. 消费者状态机处理队列中的数据。
3. 并行的子VI通过队列互相通讯。

# 仪表上位机使用的框架

在了解了前面几种主流的几种结构后，我们实际项目中如何灵活应用呢？

LabVIEW中的队列数据类型是单一的，但是我们可以灵活的使用簇，将只能传递数据的队列变为消息队列，这样配合我们的生产者消费者结构就可以很方便的开发多线程任务了。

因此我们采用LabVIEW 中稳定的队列消息处理器。队列消息处理器(QMH)模板便于并行执行多段代码，以及在段与段之间发送数据。每段代码表示一个任务，如采集数据。其设计模式和状态机相似由结合了生产者消费者结构多线程的优点，适用于多个任务并行执行的应用程序。

该模板重复执行下列步骤：

1. 用户与前面板交互，导致事件处理循环(EHL)中的事件结构生成一个消息。LabVIEW将消息存储在队列中。
2. 消息处理循环(MHL)从消息队列中读取消息，将消息清除。
3. 消息本身是一个字符串，匹配MHL条件结构的一个分支。因此，读取消息将执行条件结构中对应的分支。
4. （可选）消息框图生成另一个消息，存储在消息队列中。

# 具体实现思路

软件设计初始为PLx00设备，因此软件架构采用多线程方案，此方案具有方便单独调试、独立扩展等优点，建议后续产品均采用此软件结构，根据产品实际需求选择线程数量。

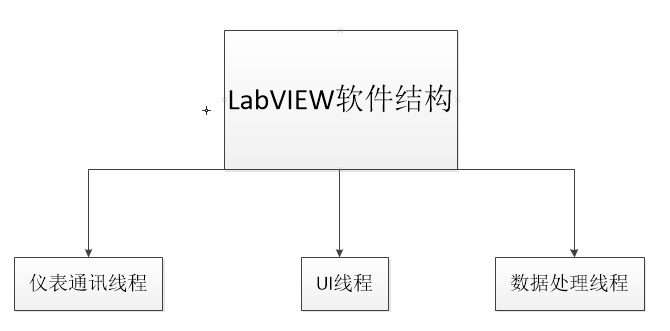
软件分为3个线程：UI消息循环、仪器通讯循环、Log数据记录处理循环；三个线程间通信均采用消息队列方式；每个线程负责处理自己的任务。

图1.1LabVIEW软件框架

4.1 UI交互循环

UI循环也是程序的主循环，控制整个程序的运行情况，响应用户的操作更新界面的状态。

具体实现按照以下步骤：

1）首先软件运行时，创建消息队列获取引用，然后进入UI线程初始化系统参数，直接从本地读取用户上次登录已经设置好的参数并初始化全局变量。同时获取界面控件引用，后面控件的操作直接通过引用来实现，减少控件的复制。

2）在整个过程初始化完成后，进行等待用户操作。如果用户选择更新测试参数，那么该线程进入参数设置部分，参数设置完毕后回到UI主线程，同时将最新的测试参数放入到UI线程移位寄存器中，方便后面直接调取使用。

3）用户选择开始测试后，UI线程取出移位寄存器中的数据，通过消息队列传给仪器通讯线程，然后更新界面状态。

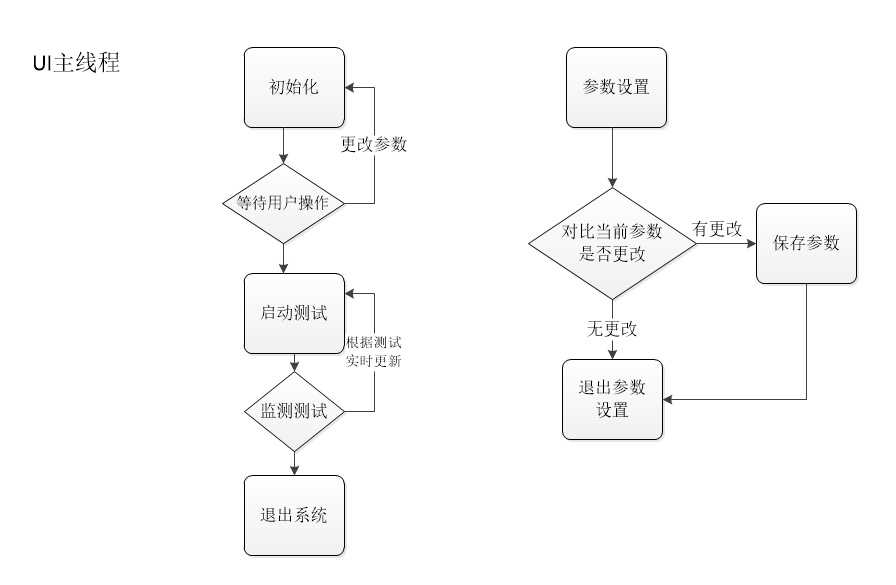


图 1.2 UI交互循环

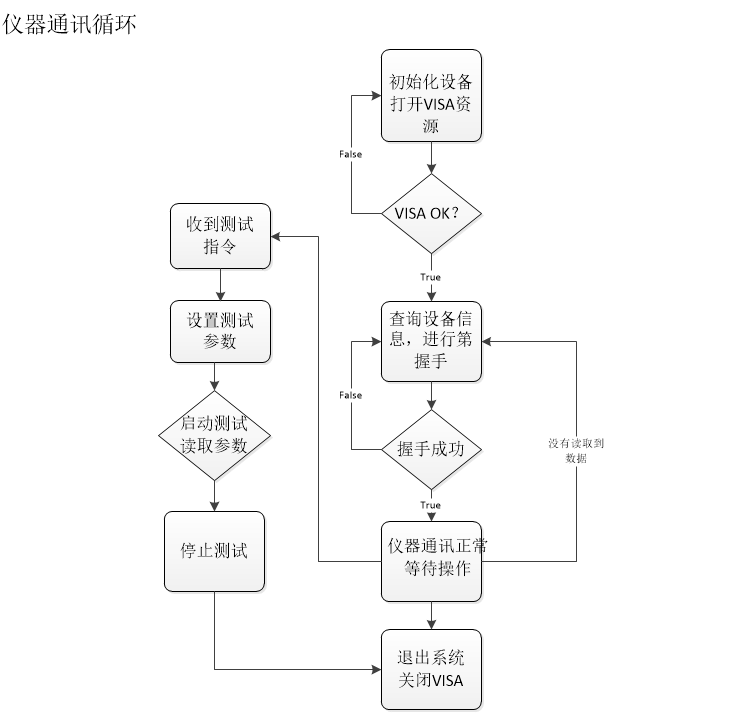
4.2仪器通讯循环

仪器通讯线程主要负责整机对外通信数据的传输、控制相关功能。具体实现按照以下步骤：

1）先进入设备初始化，打开VISA资源，如果VISA资源打开成功，进入握手环节，读取仪器信息，判断仪器是否正常通讯。

2）设备连接成功后等待用户操作，如果收到测试指令，则进行参数下发，启动测试，读取数据，并将读到的数据通过消息队列发送到UI线程进行显示处理。

3）测试结束，退出循环，关闭VISA资源。



4.3数据处理记录

这一块目前上位机软件只有数据导出报表功能，还没有数据自动保存以及历史数据查看功能，根据具体客户需求开发。

# 软件拓展方案

1.仪器测试模式不同，如果需要更改，可以直接将同类仪器功能模块化。

2.增加测试方式，可以直接在主框架上增加相应的线程。