中国科学技术大学

本科毕业论文

论文题目：基于LabVIEW的仪器整合控制和实验数据记录与查询系统的设计与实现

英文题目：The design and implementation of instruments control and notes recording & querying system based on LabVIEW

作者姓名：王 晨 坤

作者学号：PB10203270

学科专业：近代物理系应用物理专业

导师姓名：郑 坚 教授

完成时间：二零一四年五月

致　谢

四年的大学生活已经临近尾声，在这即将离校的时刻，我的本科毕业论文也圆满地画上了句号。在此，我要向所有关心和帮助过我的老师、同学、朋友及家人表示最诚挚的感谢！

首先，要感谢郑坚教授和袁鹏师兄。本篇毕业论文能够顺利完成，离不开他们的全力支持和悉心指导。从最开始的选题，到中期的答疑解惑，再到最后的审核定稿，他们为这篇毕业论文倾注了大量的时间和心血。除了在每周例行的组会上，甚至在午休、深夜里，郑老师和袁鹏师兄经常对我的工作进度给出一些很有启发性的点拨。在我遇到各种困难时，他们也都给了我很多意见和指导，这使我受益匪浅。论文中，很多棘手的问题最终能解决，都得益于他们的指点。另外，他们两位严肃的科学态度，严谨的治学精神，精益求精的工作作风，也深深地感染和激励着我。

其次，要感谢实验室的师兄和同学。在这篇毕业论文的完成过程中，实验室的涂少勇师兄、龚涛师兄、杨涛师兄、梁亦寒师兄、张琛师兄等给予了我莫大的支持和鼓励。整个实验室良好的学习和工作氛围让我更快更好的完成了整个毕业论文的工作。另外，实验室高年级的师兄师姐，像龙诚德师兄，于明海师兄，郑伟真师姐等同学，在这最后一年里也都给我树立的良好的榜样，同时提供了很多工作学习甚至生活上的帮助。

最后，还要感谢我的女朋友杨格格和远在故乡的父母，无论是在我遇到困难焦头烂额脾气不好时，还是我忙于工作身体欠佳时，他们的鼓励和关怀给了我无尽的动力！

目　录

摘　要

随着各种实验室仪器的数量不断增加，仪器生产商的不断兴起，每个实验室中往往都会有很多不同牌子不同型号的实验仪器。每台仪器在购买时往往会带有仪器生产商自己设计的仪器控制软件，但不同的控制软件导致在一次实验中需要打开多种软件分别控制。而且，厂商为了满足不同用户的不同需求，所附带的仪器控制软件往往具有很多实际实验中不必要的功能，使用起来也不便捷。另外，实验室数据笔记的记录往往没有一个统一的规范，导致之后实验的查找显得不够方便。

本论文利用Labview虚拟仪器开发平台，通过混合编程技术，开发了两类（VISA接口通信与SDK底层编程）仪器控制软件，将这两种类型的仪器整合到一个软件中。市面上的大多数实验仪器使用的都是这两种控制软件设计方式。同时将实验室数据、笔记规范记录和查找功能一并整合其中。设计并实现了一个基于Labview的实验室仪器控制、笔记记录软件。做到了一个软件控制多台仪器，一个平台记录所有实验数据，大大方便了各种实验过程中的调整、记录和查找。

关键词：Labview，仪器控制，混合编程，数据记录

Abstract

With the increasing number of laboratory instruments and the rising of instrument manufactures, each laboratory always have a lot of experimental apparatus in different brands and models. When buying the instruments, each of them will be attached with a relevant software designed by the producers themselves. But different controlling software leads that you need to open many controlling software during one experiment. In addition, in order to meet the different requirements of users, the software designed by the producer always has many useless and inconvenient functions which are unnecessary during experiments. What's more, the notes of data and parameters don't have a standard to record which results that it is inability to find information in an easy way.

This dissertation introduces two kinds of ways, VISA serial communication and underlying programming from SDK, developing instruments' controlling software using virtual instrument development platform, Labview, and mixed programming techniques. In the meantime, the functions of recording and searching the laboratory notes are added into this software. I designed and implemented an instruments' controlling and notes record software which make the "one software controlling & one platform recording" come true. This software create a significantly easier experimental adjustments, notes recording and data searching.

Keywords: Labview, instruments control, mixed programming, data record

第一章 绪论

1.1LabVIEW虚拟仪器开发平台及图形化编程语言

LabVIEW（Laboratory Virtual Instrumentation Engineering Workbench，实验室虚拟仪器工程平台）是由美国国家仪器公司（National Instruments）所开发的图形化程序编译平台，发明者为杰夫·考度斯基（Jeff Kodosky），程序最初于1986年在苹果计算机上发表。LabVIEW早期是为了仪器自动控制所设计，至今转变成为一种逐渐成熟的高级编程语言，市场普及率仅次于C/C++语言。

LabVIEW的开发环境类似于C和BASIC，但是LabVIEW与其他计算机语言的显著区别是：其他计算机语言都是采用基于文本的语言产生代码，而LabVIEW使用的是图形化编辑语言G编写程序，产生的程序是框图的形式。G是一种数据流编程模式，它有别于基于文本语言的线性结构。在LabVIEW中执行程序的顺序是由块之间的数据流决定的，而不是传统文本语言的按命令次序连续执行的方式。图形化编程与传统编程语言之优势便在于在于程序流程采用”数据流”之概念打破传统之思维模式，使得程序设计者在流程图构思完毕的同时也完成了程序的撰写。

LabVIEW写出的程序被称为虚拟仪器（Virtual Instrument），简称VI。每个VI包含三个部分：前面板、框图程序和图标/连接口。前面板用于输入和显示输出量，类似于我们常说的用户界面（GUI）。程序的大体情况如图1所示。

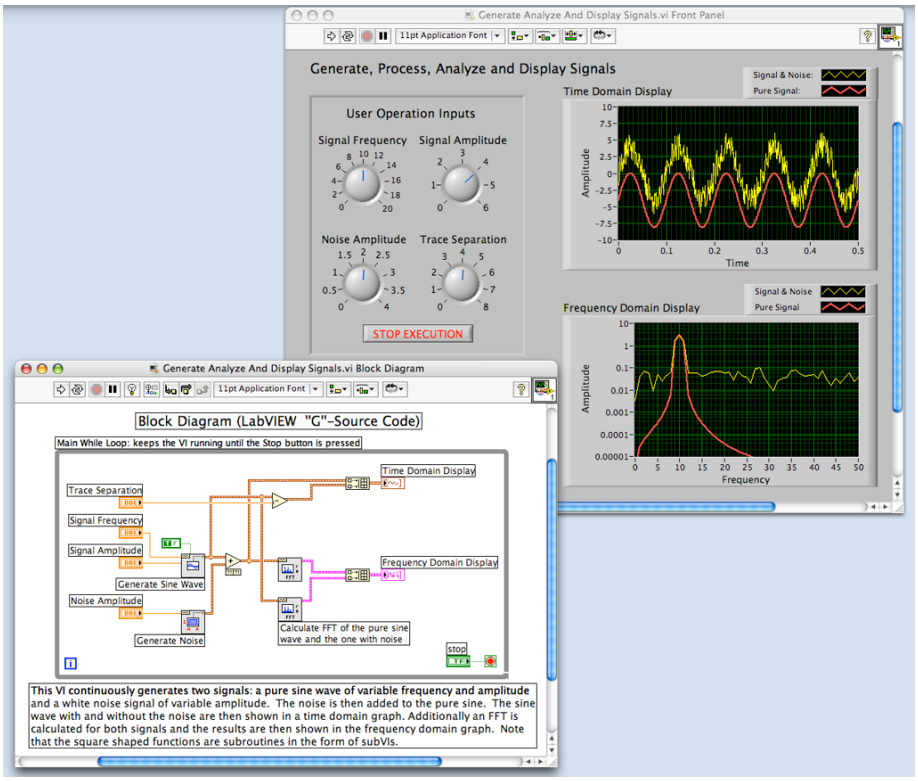

图1 LabVIEW程序大体结构

从图中的框图程序不难看出，这种编程方式与人类的思维过程非常接近。

自LabVIEW 8.x开始也引入了面向对象（OOP）的编程概念，使得LabVIEW在更多的领域有了更广泛的用途。

除了图形化的编程方式简单、方便之外，LabVIEW的优势还体现在以下几个方面：

（1）强大的图形化处理、分析能力

LabVIEW提供了很多处理、分析数据的VI库和工具包，控制系统仿真工具包、PID工具包、模糊控制工具包等。结合LabVIEW独特的数据结构（波形数据、簇、动态数据类型等）使得测量数据的分析、处理非常简单、方便、并且实用性很强。

（2）开放的开发平台

LabVIEW是一个开放的开发环境，它拥有大量的与其他应用程序进行通信的VI库，提供广泛的软件集成工具、运行库和文件格式，可以方便的与第三方设计和仿真仪器连接，如DLL、共享库。

（3）快速的开发

囊括了DAQ、GPIB、PXI、VXI、RS232-245在内的各种仪器通讯总线标准的所有功能函数，通过仪器驱动程序可以与大多数仪器进行通讯。从而使得用户不需要学习仪器的低级编程协议，简化了仪器控制软件的开发，缩短了开发时间。

（4）简单的方案

使用“所见即所得”的可视化编程技术建立友好的人机界面。

（5）灵活的仪器

将LabVIEW与一般的数据采集仪器设备加以组合，可以设计出灵活的系统，并可以随时将仪器移植到最适合用户的平台上使用。

（6）很强的通信能力

LabVIEW的TCP/IP和网络能够与应用系统通信，Internet工具箱还为应用系统增加了Email、FTP和Web能力。利用远程自动化还可以对多台设备进行分散控制。

本文便利用LabVIEW良好的计算机与仪器互联通信能力，以及其方便快捷的前面板（用户界面）生成能力，开发一套实验室多种仪器整合控制软件，总结仪器控制软件常用的开发方法。同时，考虑到实验室笔记记录及查询的效率，利用LabVIEW方便的文件读写控件，将笔记记录查找功能一并整合其中。

1.2 软件开发工具包（SDK）

软件开发工具包（Software Development Kit，SDK）一般是一些被软件工程师用于为特定的软件包、软件框架、硬件平台、操作系统等创建的应用软件的开发工具的集合。它可以简单的为某个程序设计语言提供应用程序接口API（ApplIcation Programming Interface）的一些文件，但也可能包括能与某种嵌入式系统通讯的复杂的硬件。一般的工具包括用于调试和其他用途的实用工具。SDK 还经常包括示例代码、支持性的技术注解或者其他的为基本参考资料澄清疑点的支持文档。

为了鼓励开发者使用其系统或者语言，许多 SDK 是免费提供的。软件工程师通常从目标系统开发者那里获得软件开发包，也可以直接从互联网下载，有时也被作为营销手段。例如，营销公司会免费提供构件SDK 以鼓励人们使用它，从而会吸引更多人由于能免费为其编程而购买其构件。在实验室使用的很多仪器中，很大一部分仪器生产商喜欢将自己仪器的控制软件SDK免费提供给使用者。并鼓励使用者对其进行开发，以创造出更多的相关插件与所有用户共享。但是，仪器生产商又不愿意将所有的底层代码进行公开，往往将其封装成.DLL，.LIB，.H的各种文件，并配以相应的说明文档，在一定的限制下允许用户进行开发。在之后的介绍中，本文将对这类SDK的开发，.DLL/.LIB/.H文件的使用进行详细介绍。

对于不同的操作系统，厂家会提供不同的运行环境下不同编程语言的SDK，如Microsoft的DirectX SDK、iOS的iOS SDK、Sun Microsystems的Java SDK、英特尔AMT的SDK等。同时不同的系统会有相应的文件格式，比如MacOS系统下的.dylib文件都包含了SDK相应的库函数。

1.3 串口通信

串口通信是指外设和计算机间，通过数据信号线 、地线、控制线等，按位进行传输数据的一种通讯方式。这种通信方式使用的数据线少，在远距离通信中可以节约通信成本，但其传输速度比并行传输低。

所谓的串口通信，其实就是串行接口（Serial Port）通信，主要用于串行式逐位数据传输。串行接口是一种可以将接受来自CPU的并行数据字符转换为连续的串行数据流发送出去，同时可将接受的串行数据流转换为并行的数据字符供给CPU的器件。串行接口按电气标准及协议来分，包括RS-232-C、RS-422、RS485、USB等。 RS-232-C、RS-422与RS-485标准只对接口的电气特性做出规定，不涉及接插件、电缆或协议。USB

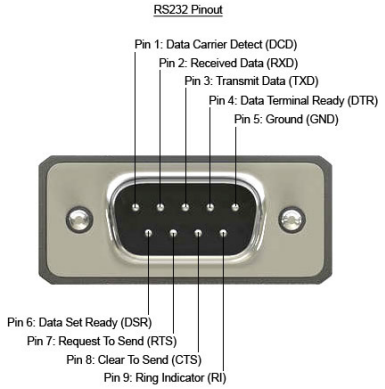

图2 RS-232接口

是近几年发展起来的新型接口标准，主要应用于高速数据传输领域。所以在实验室的仪器控制通信上，最常用的就是RS-232，一般的普通计算机使用的都是这种规格的接口。RS-232（ANSI/EIA-232标准）是IBM-PC及其兼容机上的串行连接标准。可用于许多用途，比如连接鼠标、打印机或者Modem，同时也可以接工业仪器仪表。用于驱动和连线的改进，实际应用中RS-232的传输长度或者速度常常超过标准的值。RS-232只限于PC串口和设备间点对点的通信。RS-232串口通信最远距离是50英尺。

串口通信的原理非常简单，串口按位（bit）发送和接受字节。尽管看起来比按字节（byte）的并行通信慢，串口形容一下就是一条车道，而并口就是有8个车道同一时刻能传送8位（一个字节）数据。但是，并不是并口通信就一定快，由于8位通道之间的互相干扰。传输时速度就受到了限制。而且当传输出错时，要同时重新传8个位的数据。串口没有干扰，传输出错后重发一位就可以了。另外一方面，串口可以在使用一根线发送数据的同时用另一根线接收数据。它很简单并且能够实现远距离通信。比如IEEE488定义并行通行状态时，规定设备线总长不得超过20米，并且任意两个设备间的长度不得超过2米；而对于串口而言，长度可达1200米。

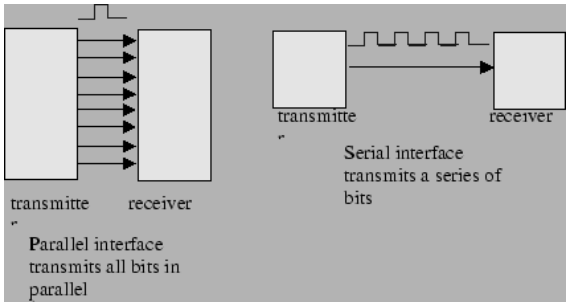

图3 并口通信和串口通信

串口通信最重要的参数是波特率、数据位、停止位和奇偶校验。对于两个进行通信的端口，这些参数必须匹配。具体参数的介绍将在下一章中介绍。

在LabVIEW中，有专门的VISA串口通信组件来之间对以上四个参数进行控制，以实现计算机和串口另一端仪器的通信和交流。

第二章 仪器整合控制模块

2.1 仪器控制简介

仪器控制是指通过PC上的软件远程控制总线上的一台或多台仪器。它不但需要仪器与计算机连接协同工作，同时还需要根据实际实际需求拓展和开发软件的功能。通过计算机强大的运算处理、分析、显示和储存能力，可以极大程度上的扩充仪器的功能。一个完整的仪器控制系统除了包括计算机和仪器外，还必须建立仪器与计算机的通路记忆上层应用程序。

通路包括总线和真多不同仪器的驱动程序，上层应用程序用于发送控制命令、仪器的控制面板显示以及数据的采集、处理、分析、显示和储存等。

要构建一套完整的仪器与计算机通路，需要设计合适的总线和仪器的驱动程序。

（1）总线

开发控制系统时，选择正确的总线与选择一个具有合适采样率和分辨率的设备一样重要。不同的总线可以影响仪器的反应速率、运行性能以及整个系统的搭建时间。一般厂商生产的仪器自身会支持多种总线，可以通过不同的总线控制仪器。如果PC本身不支持某种总线的链接，就可以使用别的总线方式来替代，或者通过外部添加一个转换器实现各种类型总线之间的转换。用于仪器控制的总线一般可分为两类：

①独立总线，用于架式和堆式仪器的通信T&M专用总线（如GPB）和PC标准总线（如RS-232、以太网、USB、无线）；

②模块化总线，将接口总线合并到仪器中（PXI、PCI、OCIExpress、VXI）。

大部分仪器都是用独立总线，因为其更为便捷的链接方式和PC端丰富的仪器接口。

（2）仪器驱动程序

计算机与仪器的进行通信的方式有两种：一种是基于寄存器的通信方式，另一种是基于消息的通信方式。PXI和很多VXI仪器都采用寄存器的通信方式，使用这种方式可以在一个较低层次直接对仪器的控制寄存器读写二进制信息。GPB、串口、USB使用基于消息的通信格式，对这些仪器发送的命令和读回的数据都是高层次的ASCII字符，仪器本身具有处理器负责解析字符串命令和发送字符串数据。

而一个仪器的驱动驱动是一个软件例程集合，创建了一个硬件与硬件或者硬件与软件的沟通接口。该集合对应于一个计划的操作，如配置仪器、从仪器读取、向仪器写入和触发仪器等。它将底层的通信命令或寄存器配置等封装起来，用户只需要调用封装好的函数库就能轻松实现对应于改仪器的任何功能。

虽然几乎实验室的所有仪器都是通过这样的方式进行通信控制，但不同的仪器在开发上还是有不同的方法。一般可分为两类，一类是将底层字符串通信格式说明的串口数据流通信；另一类是将底层通信模式封装，只给相关的SDK利用所给函数进行控制。

2.2 利用串口数据流通信的仪器控制

2.2.1 串口通信简介

串口通信就是串行接口通信，主要用于串行式逐位传输数据。串行接口按电器标准协议来分，包括常见的RS-232，工业计算机应用的半双工RS-485与全双工RS-422，甚至包括

USB等。RS-232是目前最常见的一种串行通讯接口，也叫标准串口，大部分的仪器和设备都通过RS-232与计算机相连。它是由美国电子工业协会（EIA）在1970年联合贝尔系统、调制解调器厂家及计算机终端生产厂家共同制定的用于串行通讯的标准。他的全名是“数据终端设备（DTE）和数据通讯设备（DCE）之间串行二进制数据交换接口技术标准”。现在所使用的RS-232-C的端口一般都使用9芯D型插座。

串口通信是在一根传输线上一位一位的传送信息的，即按位（bit）发送和接受字节（byte）。因此，这样的发送方式便于远距离传输信息，同时不容易出错。对于仪器这类高精度操作的通信来说，是一个很好地选择。相比于并行通信，即一个字节一个字节传送信息的方式，串口通信在一定程度上并不一定比并口慢。串口就好比是一条车道，并口就是同时有八个车道同一时刻传送8位（一个字节）数据。但由于8位通道之间的互相干扰，传输时的速度就受到了限制，为保证更低的出错率，并行通信往往慢于串行。同时，如果传输出错时，并行通信需要同时重新传输8个位的数据，而串口通信只需重发一位即可。

串口通信在使用时需要在软件中设置相应的参数，以保证端口间的参数匹配。最常见也是最重要的参数为波特率、数据位、停止位和奇偶校验位。

a. 波特率（Baud Rate）：是指一个 设备发到另一个设备的波特率，即每秒钟多少比特（bit/s）。典型的波特率是300，1200，2400，9600，19200，115200bit/s等。一般通信两端设备都要设为相同的波特率，但有些设备也可以设置为自动检测波特率。

b. 数据位（Data Bits）：这是衡量通信中实际数据位的参数。当计算机发送一个信息包，实际的数据不会是8位的，标准的值是6、7和8位。如何设置取决于你想传送的信息。比如，标准的ASCII码是0~127（7位）。拓展的ASCII码是0~255（8位）。如果数据使用简单的文本（标准ASCII码），那么每个数据包使用7位数据。每个包是指一个字节，包括开始停止位，数据位和奇偶校验位。

c. 停止位（Stop Bits）：实在每个字节传输之后发送的，用来帮助接受信号方硬件重同步。典型的值为1，1.5和2位。适用于停止位的位数越多，不同时钟同步的容忍程度就越大，但是数据传输率同时也变慢。

d. 奇偶校验位（Parity）：是用来验证数据的正确性。奇偶校验一般不使用，如果使用，那么既可以做奇校验（Odd Parity）也可以做偶校验（Even Parity）。奇偶校验是通过修改每一发送字节（也可以限制发送的字节）来工作的。如果不作奇偶校验，那么数据是不会被改变的。在偶校验中，因为奇偶校验位会被相应的置1或0（一般是最高位或最低位），所以数据会被改变以使得所有传送的数位（含字符的各数位和校验位）中“1”的个数为偶数；在奇校验中，所有传送的数位（含字符的各数位和校验位）中“1”的个数为奇数。奇偶校验可以用于接受方检查传输是否发送生错误——如果某一字节中“1”的个数发生

了错误，那么这个字节在传输中一定有错误发生。如果奇偶校验是正确的，那么要么没有发生错误要么发生了偶数个的错误。如果用户选择数据长度为8位，则因为没有多余的比特可被用来作为同比特，因此就叫做“无位元（Non Parity）”。

2.2.2 串口通信控制软件的开发

在仪器串口和计算机串口已将相互识别的情况下，就可以进行仪器和计算机间的通信，以实现计算机对仪器的控制。

几乎所有的这类仪器生产商都会在其相应的说明书或者手册中，详细的描述仪器串口通信规范。那里会对仪器所能识别的命令格式字符串进行介绍说明，解释其使用方法。要开发这类仪器的控制软件，首先就要对其所有的命令方式有清楚的了解。由于这类控制方式的限制，使用串口通信直接传入字符串的仪器往往控制方式比较简单，仪器行为比较单一。常见的有步进电机，真空计等。这类仪器的命令模式就如同Windows的命令行，一行命令就代表了一段仪器行为。不同的仪器都有自己的命令格式，一般如下：

其中的M和G就代表某种命令模式（会在说明文档中解释），M之后的12345就代表这种命令所对应的参数值，最后的/r（有些情况下是/n，根据仪器识别的结束符为准）代表一段命令的结束。将这一串字符串通过串口给入仪器后，仪器便能识别其中的命令模式、命令参数并采取相应的行动。比如SIGMA KOKI生产的步进电机对于上诉的一段字符串的识别如下：M代表move也就是步进电机前进，12345代表前进的步数，是12345步，/r代表命令结束；G代表go也就是步进电机根据之前的命令要求开始运作，同样的/r代表命令结束。当步进电机读到如上命令时，便会进行前进12345步的操作。

同样的，我们也需要让仪器将仪器的一些参数返回给计算机。原理完全相同，当仪器接收到返回仪器参数的命令时，便会通过串口将仪器的参数以一定的格式输出到计算机的寄存器中，使用者可以通过读取串口的返回字符串来了解仪器当前的状态。

2.2.3 LabVIEW实现串口数据流通信控制

在LabVIEW中，提供了一个很好的仪器I/O的软件库接口，VISA（Virtual Instruments Software Architecture）虚拟仪器软件体系结构接口。VISA是应用于仪器编程的标准I/O应用程序接口，是工业界通用的仪器驱动器标准API（应用程序接口），采用面对对象编程，具有很好的兼容性、扩展性和独立性。用户可用一个API控制包括VXI、GPIB及串口仪器在内的不同种类的仪器。它还支持多平台工作、多接口控制，是一个多类型的函数库。

在LabVIEW中编写VISA接口程序非常简单，因为LabVIEW为用户设计好了非常实用的程序模块。在编程过程中只要调用相应的编程模块便能实现对仪器的控制，其中最为方便的就是利用VISA节点进行串口通信编程。为方便用户使用，LabVIEW将这些VISA节点单独组成一个子模块，共包含8个节点，分别实现初始化串口、串口写、串口读、中断以及关闭串口等功能。

（1）初始化串口

在之前提到的串口通信的介绍中，我们说串口的四个不同的参数决定了两个串口间的通信方式。LabVIEW中初始化串口的组件便能一次性确定这四种常用的参数，以实现串口的选择和初始化。组件的节点介绍如图4：

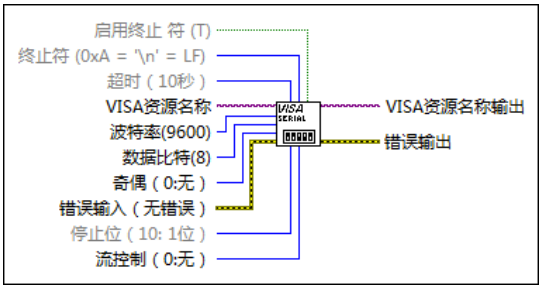

图4 初始化串口框图连线端说明

可以看到，通过对波特率、数据位、奇偶校验位、停止位等的设置，可以讲连于计算机的仪器串口通信方式与计算机相适应。仪器串口的通信方式一般会在仪器的使用手册中有说明。相应的设置之后，由VISA资源名称输出节点输出的端口，即为仪器所能识别的串口。

（2）串口写入

在设定串口的参数和串口号之后，只需要将相应的命令行字符串由串口写入仪器即可。LabVIEW也提供的相应的组件。组件的节点介绍如图5：

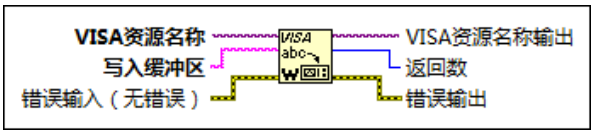

图5 串口写入框图连线端说明

只需将之前确定的VISA资源名称，也就是那个仪器所对应的串口连入，并将想要写入仪器的命令字符串写入缓冲区，便能实现计算机对仪器的串口控制。命令字符的格式和规范都要严格按照仪器手册说明，否则容易导致仪器出现故障。输入仪器的命令可以根据需要在输入之前进行一系列的操作，以满足仪器控制过程中的各种需要。

（3）串口读取

与串口写入相同，LabVIEW为仪器的串口通讯时返回的字符串也提供了相应的组件用于读取。组件将自动读取仪器写入缓冲区的返回串，并以字符串的形式返回给使用者。组件

的节点介绍如图6：

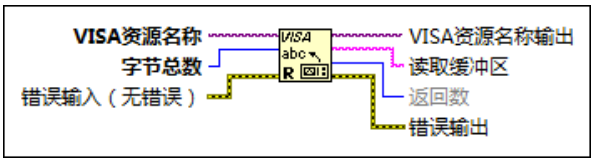

图6 串口读取框图连线端说明

同样的方法，将仪器的串口VISA资源名称连入组件，组件便会自动读取所有返回到缓冲区内的数据。用户只需通过显示控件，将读取缓冲区节点内得到的字符串显示出来即可。仪器所返回的数值的格式和意义也会在仪器手册中有详细说明，用户可以根据自己需要对读出的字符串进行一系列的操作。

（4）VISA关闭

VISA关闭是一个非常简单的函数，LabVIEW将其简单的做成一个模块，用户只需将所要关闭的串口资源连入该组件便能实现VISA串口的关闭。为了避免不必要的数据流入与流出，在对仪器的操作结束之后及时的关闭VISA是一个很好的习惯。

通过以上的这些组件便能实现LabVIEW的VISA串口通信，从而实现计算机对仪器的控制。用户可以根据自己其他方面的需求，利用LabVIEW编程方面的其他组件，优化通信的模式和字符串的给入方式，最终达到仪器控制软件的设计目的。

2.3 利用自带可开发软件的仪器控制

2.3.1 SDK简介

SDK（Software Development Kit，软件开发工具包）一般是一些被软件工程师用于为特定的软件包、软件框架、硬件平台、操作系统等创建应用软件的开发工具集合。软件的SDK一般包含软件设计所需要的相关应用程序接口（API）的一些文件，示例代码，支持性的技术注解或者其他为基本参考资料澄清疑点的支持文档。实验室中所使用的部分仪器会随仪器附赠一套相应的控制软件，同时为方便不同实验室对仪器的不同需求，厂商还会提供改软件的SDK以方便使用者根据自己的需求开发。为鼓励开发者使用该仪器、该软件，厂商会免费提供SDK，一般能够直接从网站上下载。有些厂商甚至向出色的新开发软件支付费用以购买其版权，方便更多有需要的用户使用。

要介绍SDK的使用方法和原理，首先要了解以下几个概念：

（1）API

API（Application Programming Interface，应用程序接口）又称为应用编程接口，就是软件系统不同组成部分衔接的约定。由于近年来软件的规模日益庞大，常常需要把复杂的系统划分成小的组成部分，编程接口的设计十分重要。程序设计的实践中，编程接口的设

计首先要使软件系统的职责得到合理划分。良好的接口设计可以降低系统各部分的相互依赖，提高组成单元的内聚性，降低组成单元间的耦合程度，从而提高系统的维护性和扩展性。在Windows中，应用程序通过调用这些应用程序接口，以调用函数的方式来实现系统操作。

（2）DLL

DLL，即Dynamic Link Library（动态链接库）。在Windows环境下含有大量的.dll格式的文件，这些文件就是动态链接库文件，其实也是一种可执行文件格式。但与.exe文件不同的是，.dll文件是不能直接执行的，他们通常由.exe在执行时装入。DLL中含有一些资源和实际可执行代码。其实Windows的 三大模块就是以DLL的形式提供的（kernel.dll，User32.dll，GDI32），里面就含有了API函数的执行代码。为了使用DLL中的API函数，必须要有API函数的什么（.h)和其导入库（.LIB），导入库可以理解为是为了在DLL中找到API的入口点而使用的。

为了编译所需要的API函数，我们就要有跟API所对应的.h和.LIB文件，为了实际运行API函数，我们就要包含API函数的DLL文件，而SDK正是提供了一整套开发应用程序所需要的相关文件、范例、工具和说明的“工具包”。图7直观的说明了以上几者的关系：

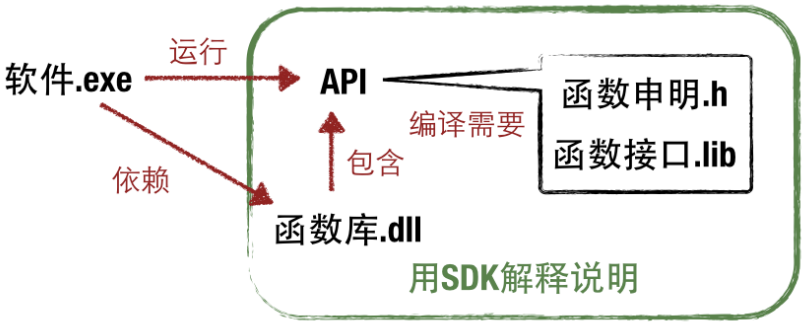

图7 API、动态库、SDK等的关系图

SDK中会详细解释软件所提供的各种函数的参数和返回值，同时介绍函数的使用方法和编程方式。同时通过各种例子，直观的给用户一系列函数的使用方法。但不同的厂商SDK的格式和用法都不太相同，但总归是通过调用函数来实现仪器控制的，因此只要仔细查看说明文档了解函数的使用方式即可。

2.3.2 自带控制软件的SDK开发

不同的控制软件所提供的SDK也会不同，但都大同小异。一般的仪器所给的控制软件本身就是由厂商设计的一系列函数组合并加以GUI设计的，更有甚者，软件本身就是SDK中的一个demo。换句话说，如果能够理解全部函数的使用方式和意义，就可以设计出一个与自带软件完全相同的通用软件。而实际上，在实验室的仪器使用过程中，软件很多的功能是从来没有使用过的。之所以软件设计时提供了这种功能，因为不同的实验室对仪器的需求不同，

为满足所有用户的需要，软件一般设计的比较宽泛。随之而来的就是不实用或者使用起来不方便的问题，而这些都可以通过利用SDK再开发来优化。

以AVT的Manta系列CCD的SDK为例，SDK中给了编译所需要的PvAPI.h，PvAPI.lib文件，同时还将所有的函数都打包放在了PvAPI.dll中。同时，SDK中包含了详细的对其中所包含的函数的说明，虽然不知道函数底层是如何实现仪器控制的，但只需在C中调用这些函数，就能实现函数所命令的操作。由于CCD涉及到很多参数的设置，下面以一个枚举类的参数为例，图8是AVT在SDK的说明文档中介绍的一个设置枚举类的函数：

Function Reference

///ALLIED Vision Technologies

PvAttrEnumSet

Set the value of an enumeration attribute.

Prototype

```
tPvErr PvAttrEnumSet
(
  tPvHandle        Camera,
  const char*      Name,
  const char*      Value
);
```

Parameters

Camera	Handle to open camera.
Name	Attribute name.
Value	The enumeration value (a null terminated string).

Return Value

tPvErr type error code. Typical error codes for this function:

ePvErrSuccess	Function successful.
ePvErrOutOfRange	The value is not a member of the current enumeration set.
ePvErrForbidden	The attribute cannot be set at this time.
ePvErrNotFound	The attribute does not exist.
ePvErrWrongType	The attribute is not an enumeration type.

图8 AVT CCD枚举类参数设置函数

该函数实现了将一个枚举类的参数设置成用户所需的参数。函数的三个参数分别为所控制的CCD、枚举类参数的名称、所需要设置的其中一个值。而函数返回的是参数设置是否成功。

同样来自于软件所带的SDK说明，图9是AVT Manta系列CCD的一个枚举类参数TriggerSource，这个参数设置了CCD的触发源，有自由触发、线路1/2/3/4、固定速率触发、软件触发。

TriggerSource [Enum] R/W

Determines how an image frame is initiated within an acquisition stream.

Note An acquisition stream must be started in order to trigger/receive individual frames. For *Freerun* and *FixedRate* the first frame is synchronized to *AcquisitionStart* trigger.

Freerun	[Default] Camera runs at maximum supported frame rate depending on the exposure time and region of interest size
Line1	External trigger *Line1*
Line2	External trigger *Line2*
Line3	External trigger *Line3*
Line4	External trigger *Line4*
FixedRate	Camera self-triggers at a fixed frame rate defined by AcquisitionFrameRateAbs
Software	Software initiated image capture

图9 AVT CCD“触发源”枚举变量

通过这个枚举类参数和之前那个设置枚举类参数的函数结合使用，便能实现“触发源”这个参数的设置。代码如下：

PvAttrEnumSet (Camera.Handle, “TriggerSource”, “Software”);

以上函数就实现了将触发源设置成软件触发。同样的方法，还有PvAttrEnumGet()函数，能反过来将仪器的参数取得到计算机上。

知道如何使用这些函数之后，就可以通过编写所需要的功能函数，以实现最终控制实验室所需的CCD功能，并且优化仪器本身的软件算法。为了使用方便，还能为底层代码写一个GUI。一般的软件都是使用C或者C++作为底层的控制，关于界面的选择就有很多种，如Qt、LabVIEW等。

2.3.3 LabVIEW实现可开发软件控制

这里将介绍如何用LabVIEW结合软件的SDK来实现仪器的控制，同时生成非常好用的GUI。为实现LabVIEW控制这类仪器，就必须实现LabVIEW与C的混合编程。虽然LabVIEW包含了丰富的函数库和子程序，但G语言的流程控制导致程序有很多密密麻麻的线，不易于修改和改进，而传统的C语言编程方式能更快的实现复杂的计算并且很利于修改。而且，对于不愿意公开软件底层控制语言的仪器生产商来说，C语言所封装的DLL包很好的提供了保护。CIN（Code Interface Node）和DLL是LabVIEW调用外部语言代码的两种方法。由于仪器生产商更愿意使用DLL包的方式，我对LabVIEW调用DLL做一个简单介绍。

LabVIEW为这类调用提供了一个很好的模块，Call Library Function，以实现直接对DLL的调用。组件的连线段介绍如图10：

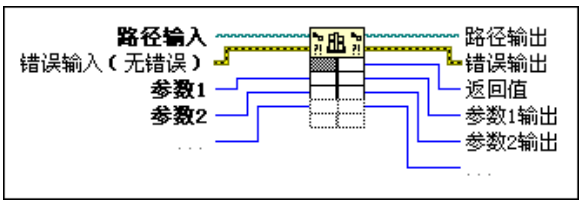

图10 调用库函数框图连线端说明

将所需要使用的DLL引入该组件中，组件会自动识别DLL中的所有函数，可以挑选其中一个作为该图标所代表的函数。函数所需要的参数可以在参数端口设置；函数的返回值将会在返回值中返回；函数参数的最终变化情况会在参数输出中返回。函数的参数、返回值的数据类型都需要自己在LabVIEW中进行设置，LabVIEW无法自动识别参数类型，因为LabVIEW的参数类型比C语言的更单一明确。这也导致在直接调用软件所给函数时，容易出现一些LabVIEW无法识别的参数，比如指针。

通过LabVIEW对外部DLL的调用实现的LabVIEW与C的混合编程，也让LabVIEW控制这类自带软件的仪器成为可能。另外LabVIEW调用C代码的能力也大大丰富了LabVIEW的可编程能力和运算速度。

2.4 利用LabVIEW整合仪器控制

了解了LabVIEW实现各种不同仪器的控制方法，如何将这些仪器一同整合在一个软件中LabVIEW也提供了很好的解决方法。LabVIEW可以选择在前面板编程，也可以利用程序框图编程。因此在之前设计好的不同仪器的控制软件已经有大体程序结构的情况下，可以利用前面板编程，将各种软件进行整合，以满足实验室方便控制的需求。

在前面板中，有一个选项卡控件，通过设计不同的选项卡，将不同的仪器放入不同的选项卡中，用户可以在在软件中通过选择选项卡来选择相应的仪器。另外，为了防止不使用的仪器在后台依旧运行其控制软件，选项卡控件还能实现当用户选择不同选项卡时，只运行该选项卡下的软件功能。这样就能更高效更快速的使用各种仪器并不造成不必要的内存占用。另外，LabVIEW还支持将外部vi直接作为一个框图连线，以实现外部子vi的调用。这样做一来可以将每个仪器控制软件写成一个子vi，方便修改和更新；二来可以使总程序条理更清晰，一个选项卡实际就是一个子vi的调用。子vi的调用方式如图11：

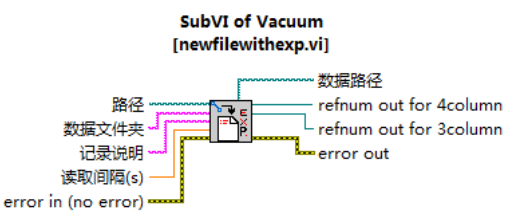

图11 子VI调用框图范例

图中的框图即为一个子vi，在子vi中对该vi的连线点进行设置，以实现输入端和输出端的共用。在调用时将子vi的绝对或相对路径给入，并选择是动态（运行到的时候才调用）或是静态（一运行程序就提前调用）两种引用方式。然后LabVIEW会自动识别子vi所带有的连线端，用户将相应数据连线即可。

这样的调用方式，既减少了总软件的框图复杂度，同时能够将不同的仪器控制实现方式无缝的衔接起来，达到整合控制的目的。

第三章 实验笔记记录、查询模块

不同实验室的笔记记录都有不同的方式，大多数的实验室甚至是大型的研究所依旧使用的是纸质笔记的记录方式。往往会有一张表格来记录实验所需要的一些常用数据，但这种记录方式无论是对记录还是对之后的查询都带来了很多麻烦。现在，国外越来越多的实验室开始使用笔记记录软件来实现实验室的笔记记录和查询。由于不同的实验室所需要记录的数据都不相同，所以软件也五花八门。为方便实验室的笔记记录和查询功能，将仪器统一控制软件中加入笔记记录和查询模块。一方面，可以实现一个软件完成实验室实验过程所需要的所有功能；另一方面，能同时将仪器在使用时的参数自动记录下来，不需要人为读取和记录，实现了更好的自动化操作。

3.1 实验笔记记录模块

作为一个研究激光产生等离子体相关现象的实验室，实验室所记录的很多数据都是每发次激光的相关参数以及改系列实验中光学平台上器件的信息。因此，笔记记录模块通过读取激光发出的信号，将此发激光的相关仪器参数自动记录。同时，实验员也可手动输入一些如透镜距离、滤片个数等常用参数的数据，还可以以一般笔记的方式记录一些实验所需的笔记。由于实验的不同所记录的数据都很不一样，笔记记录模块只能将一些常用的数据直接以相关输入框的形式记录，其余的笔记需要在字符串输入框中直接将信息和参数一个一个字码出来。

对于带有参数名的以及仪器的信息，笔记记录模块将自动以一种规范格式根据发次写入一个txt文件。软件将自动读取当时的日期和实验员的姓名，将此txt文件以实验员的姓名和日期命名，方便之后的查找。对于非标准模式的实验笔记，会以记录者记录的格式写入该文件的末尾。由于一天的实验可能会有比较多的发次，软件将自动计数所打发次，将发次同时记录如规范格式数据中。为减小记录文件的大小，笔记在记录时会记录的比较紧凑，特别是标准格式的参数，以避免在长时间使用之后数据量越来越大造成的系统空间的大量占用。

另外，由于实验可能还有相关的CCD所拍摄图片，示波器所显示波形等图片数据。如仪器已整合入仪器整合控制系统，笔记记录模块也会将仪器所读到的数据直接以日期和发次的格式记录在笔记同目录下。

3.2 实验笔记查询模块

为方便实验室笔记的读取和查找，软件还同时设计了笔记查询模块。通过对目录下不同日期的笔记的识别，将当日的实验笔记以一个规范的格式读取。若是以标准记录格式记录的参数将重新读回更直观的标准格式。比如之前直接在滤片个数框中输入的滤片个数，软件会将个数读出并显示在相应的滤片个数框中。同时用户可以看到不同发次所记录的其他一系列参数，由于记录过程中为减小文件大小将笔记记录的比较紧凑。但在笔记查询中会重新将其恢复成更为直观的表示方式。对于非标准记录的笔记，软件会保证之前记录时的格式，显示给查询者。

对于图片类的笔记数据，为了避免每次查询时需要打开一张一张的图片，在查询模块还设计了图片直接预览的部分。用户可以在不关闭软件的情况下直接在软件中看到之前所拍摄的实验数据。

另外，为了方便实验者能够更快的找到自己所要查看的实验笔记。软件还设计了相关的查询模块，软件不但能通过实验日期查询相关笔记，还能通过实验者甚至是实验过程中笔记所记录的关键词来查找实验相关的笔记。软件会将搜索到的用户所需相关笔记以列表的形式显示出来，使用者可以根据自己的需要查看相关的笔记。

3.3 记录、查询一体化的实现

为保证记录与查询的无缝衔接，一方面将数据的记录格式规范化，以达到查询时能更准确的找到相关数据，这也要求在之后添加参数时要讲记录与查询系统成对修改；另一方面，软件将记录与查询功能做进了一个选项卡，用户可以在记录的同时查询之前的笔记，方便规范实验室的实验人员以更为统一的方式记录非规范格式数据。

实验室笔记记录查询一体化的实现，满足了实验室对于冗杂的实验数据的方面记录、保存、查询的要求。也方便实验员之间的沟通和交流，数据的交换。软件为更好的保存一些比较珍贵的实验数据，之后希望能利用硬盘备份的方式，定期的将实验数据备份入专用硬盘之中。

由于实验室的仪器的实验方向是在不断变化的，实验笔记记录查询模块也需要不断的更新，添入新的仪器和新的标准数据。因此这是一个需要人工和智能不断更新进步的长期过程。

第四章 软件模块详细设计与实现

4.1 整合仪器控制部分

4.1.1 AVT Manta系列CCD的控制

AVT(Allied Vision Technologies)Manta系列CCD是典型的利用自带软件SDK开发的仪器，但有一点与一般的这类仪器不同。那就是AVT的CCD功能较全，导致其软件的函数库比较复杂。为了满足所有使用AVT公司CCD的用户的不同需求，为这款CCD所设计的DLL函数库就显得特别底层。就是函数都是对CCD最最基本的功能或参数进行单个的设置，如果将这个DLL导入LabVIEW，那必然导致控制软件需要很多很多次的调用同一个DLL中的不同函数，这样一来使LabVIEW的框图显得很复杂，二来也会让软件在运行时调用部分消耗过多时间。而且，对于一般LabVIEW的控制思路和参数的传递方式，不适合写这类函数调用类的软件。然而，这些正是C语言所在行的。因此，在这类仪器控制软件的开发过程中，就要用到C与LabVIEW混合编程。

通过C将一系列的函数操作完成，并生成相应的一个大函数，包入一个新的DLL库中，最后通过LabVIEW调用新DLL中的大函数，以实现一系列函数操作的完成。对于这种方式，一定要实现将大函数的参数接口设计好，因为之后生成的DLL是不能修改的，每次修改需要修改之前的C代码并且重新编译运行生成DLL。

对于AVT的CCD，通过将所要控制的参数函数写入一个MFC App Wizard(DLL)的工程中。先讲软件自带的PvAPI.h包含入工程的头文件，将PvAPI.lib链接入编译器，再利用一个大函数将所有参数一次性设置完成，同时将这个大函数的参数设置成无符号整型参数，用来代表不同的参数设置值，作为这个大函数与LabVIEW的接口。由于要生成相应的DLL函数，所以需要在整个cpp的开头加上：

```
extern "C" __declspec(dllexport)OutputType Function_Name(InputType Parameter1, InputType Parameter2, ...);
```

语句，然后编译运行之后就会生成新的DLL（运行时需要将PvAPI.dll文件放在同一子目录中）。在这个新的DLL中便包含了所写的新的一次性设置参数的大函数（新DLL的源代码见附件一）。

在LabVIEW中，让用户选择相应的参数设置之后，以无符号整型的形式引入函数相应的dll控件中。LabVIEW通过调用dll内部的函数，实现仪器的参数设置。LabVIEW部分底层框图如图12：

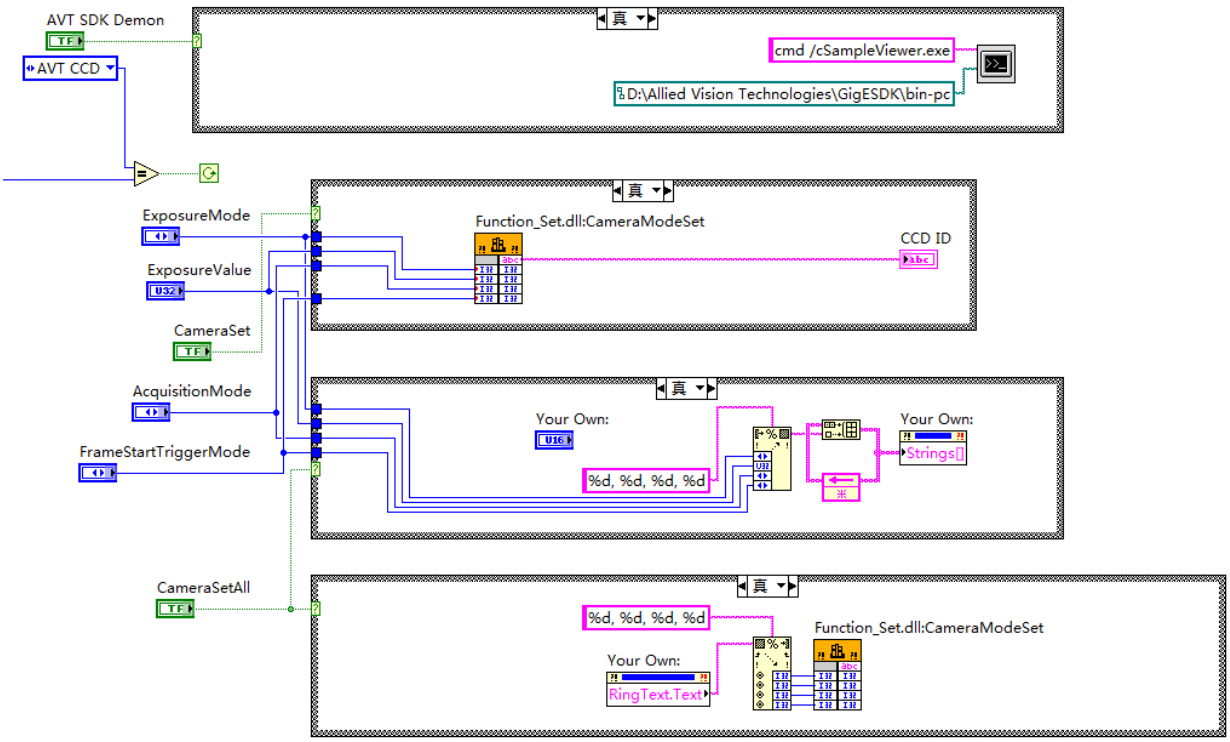

图12 AVT CCD控制程序部分底层LabVIEW框图

框图中的function函数即为外部调用的dll函数，另外框图中还调用了系统命令行参数，以运行软件自带的demo以备不时之需。

控制软件通过设置仪器的触发模式、曝光时间、曝光模式、获得模式等设置仪器的一系列参数，同时监控仪器的温度和内存的使用情况等。另外，由于实验室实验的需要，不同的发次之间需要调整一整套的参数，同时在两套参数之间切换。之前仪器自带的软件需要每次切换重新设置所有参数，而新的软件优化了这一功能。将所有的整套参数可以直接保存下来，在下次使用是可以一次性的设置所有参数。

4.1.2 步进电机的控制

步进电机的品牌比较多，常见的有Zolix（卓立汉光）、SIGMA KOKI等。但这些步进电机平台控制都大同小异，都是使用之前介绍过的标准数据流串口通信实现控制的。唯一不同的无非只是步进电机所接受的串口字符串命令格式不同。而相应的命令模式都会在相应的用户手册中有详细介绍，这里便以SIGMA KOKI的SHOT-202双轴平台控制为例。

从相应的用户手册中，可以很快的找到所需要的命令格式。图13是SIGMA KOKI公司的SHOT-202_204MS的用户手册截图，里面详细介绍了相关的命令模式：

The following is a list of available commands:

Command	String	Details	Response
Control Commands 1 (Drive Commands)			
Return to mechanical origin	H	Detect mechanical origin	When new system (MAIN) is used: Command received normally: OK Problem receiving command: NG When old system (SUB) is used: No response
Set number of pulses for relative movement	M	Axis of movement, direction, number of pulses	
Set number of pulses for absolute movement	A	Absolute coordinates	
Settings for linear movement	E	Circular interpolation (Move at minimum speed (S))	
Settings for rotary movement	K	Linear interpolation (Move at minimum speed (S))	
Jog command	J	Move at minimum speed (S)	
Drive command	G	Start	
Control Commands 2 (Settings)			
Set electronic (logical) origin	R	Set the electronic (logical) origin to the current position	When new system (MAIN) is used: Command received normally: OK Problem receiving command: NG When old system (SUB) is used: No response
Stop	L	Stop or reduce speed	
Speed settings	D	Set S, F, and R	
Alarm reset	U	Alarm reset	
Wait	W	Specify command wait time	
Trigger output	T	Trigger output	
Free motor	C	Excitation ON/OFF	
Switch number of steps	S	Switch number of steps	
Confirmation Commands			
Status1	Q	Return current position etc.	refer to P25
Status2	!	Return B/R (READY)	
Internal information	?	Check internal information	
I/O Commands			
Output	O	Output to I/O connector	When new system (MAIN) is used: Command received normally: OK Problem receiving command: NG When old system (SUB) is used: No response
Input	I	Input from I/O connector	refer to P26
Other Commands			
Mode change	P : R	Enter remote (execute) mode	When new system (MAIN) is used: Command received normally: OK Problem receiving command: NG When old system (SUB) is used: No response
	P : H	Enter host (computer) mode	
Mode change	P : P 1or 2	Specify program number (1 or 2)	
Start command	P : S	Start independent programmed operation	
Stop command	P : E	Stop independent programmed operation	
Pause command	P : U0	Pause program	
Pause cancel	P : U1	Cancel program pause	
Select signal sent on completion of operation	P : C0	Signals can not be sent at completion of programmed operation	
	P : C1	Signal is sent at completion of programmed operation	
Select signal sent on Trigger	P : T0	Trigger signal can not be sent at trigger signal output	
signal output	P : T1	Trigger signal is sent at trigger signal output	

图13 SIGMA KOKI步进电机用户手册截图

通过选取相应的命令，得到相应的命令行参数，从而通过LabVIEW控制。实验室的SIGMA KOKI的二维平台是用于切割实验使用的靶材而设计的。因此需要实现直接的一圈一圈的循环移动，就需要在LabVIEW内部进行编程，以实现这一目的。图14为相应的LabVIEW底层框图的部分设计：

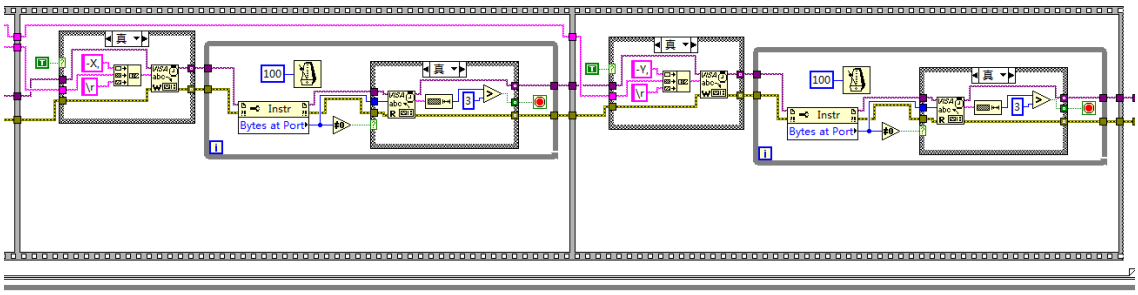

图14 二维切割平台控制程序LabVIEW底层部分框图

因此，外部软件控制串口通信编程能优化仪器控制的功能，使软件具有更多的通过手柄直接控制不能实现的功能。

图15为靶室平台控制软件的底层框图：

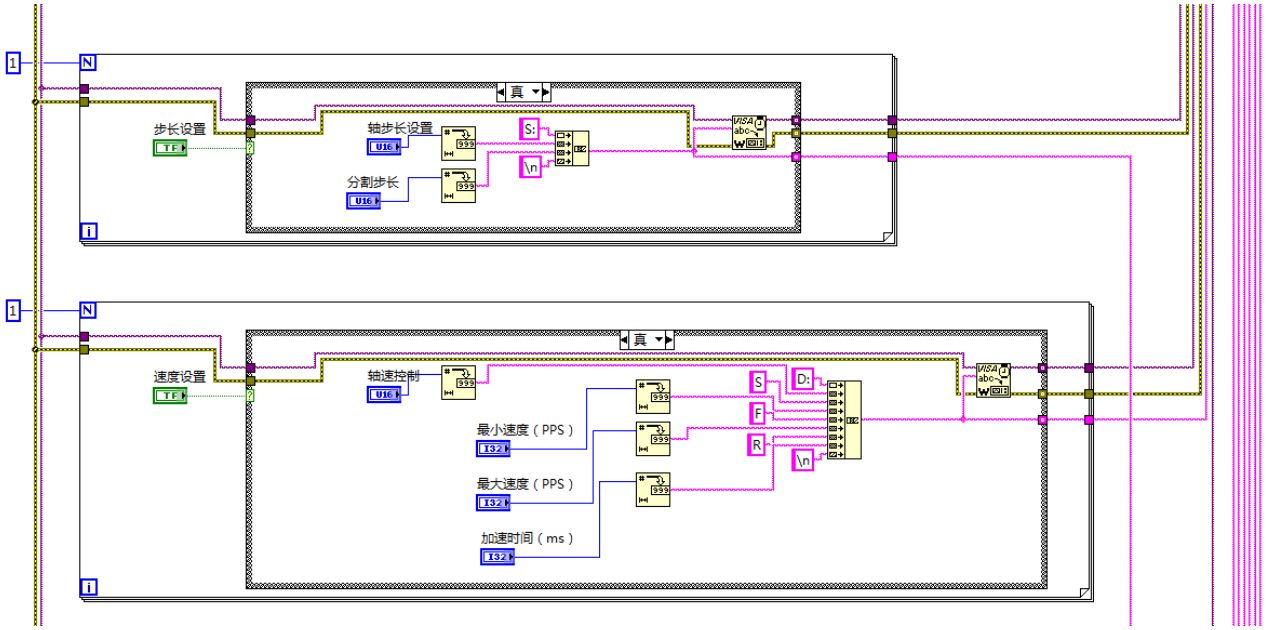

图15 靶室平台控制软件LabVIEW底层部分框图

靶室平台的控制本身也是使用手柄实现，控制起来比较缓慢也不直观。通过软件对XYZ三轴和旋转轴的直观控制，可以更好的实现软件的功能，同时对于当前靶室平台状态也有更好的体现。

4.1.3 整合仪器控制的界面设计

为优化软件的使用方式，将不同的仪器放进不同的选项卡中。另外，为使仪器尽可能的简洁易上手，将仪器控制界面尽可能的简化。最终希望达到的目标是将软件界面尽可能的拟物化，比如示波器的控制界面可以做的和示波器类似，这样只要会使用示波器的实验者就能使用这款软件。

仪器控制软件的界面见图16-18：

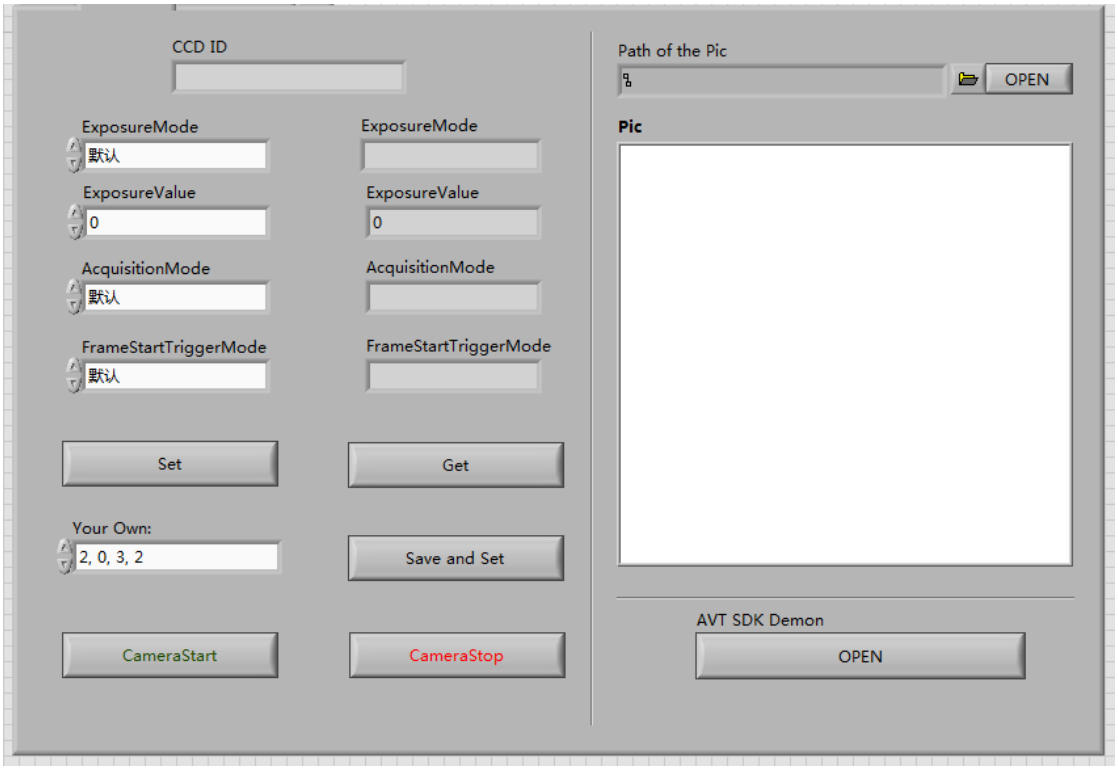

图16 AVT公司 Manta系列CCD的控制软件界面

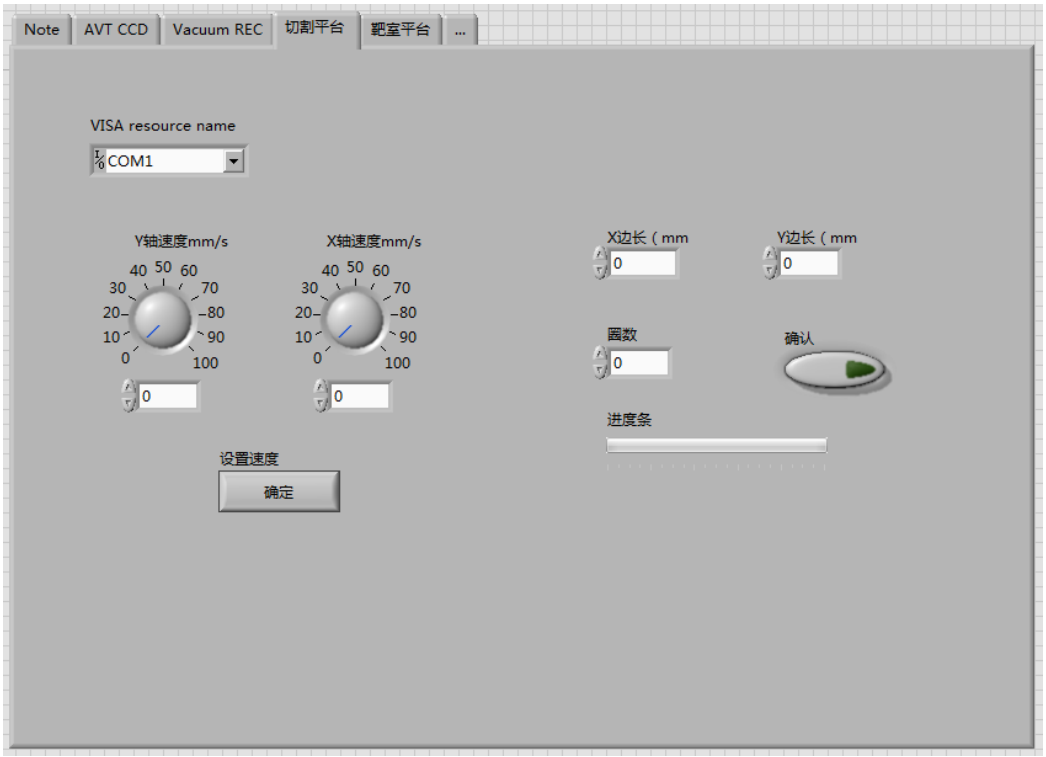

图17 SIGMA KOKI切割平台控制软件界面

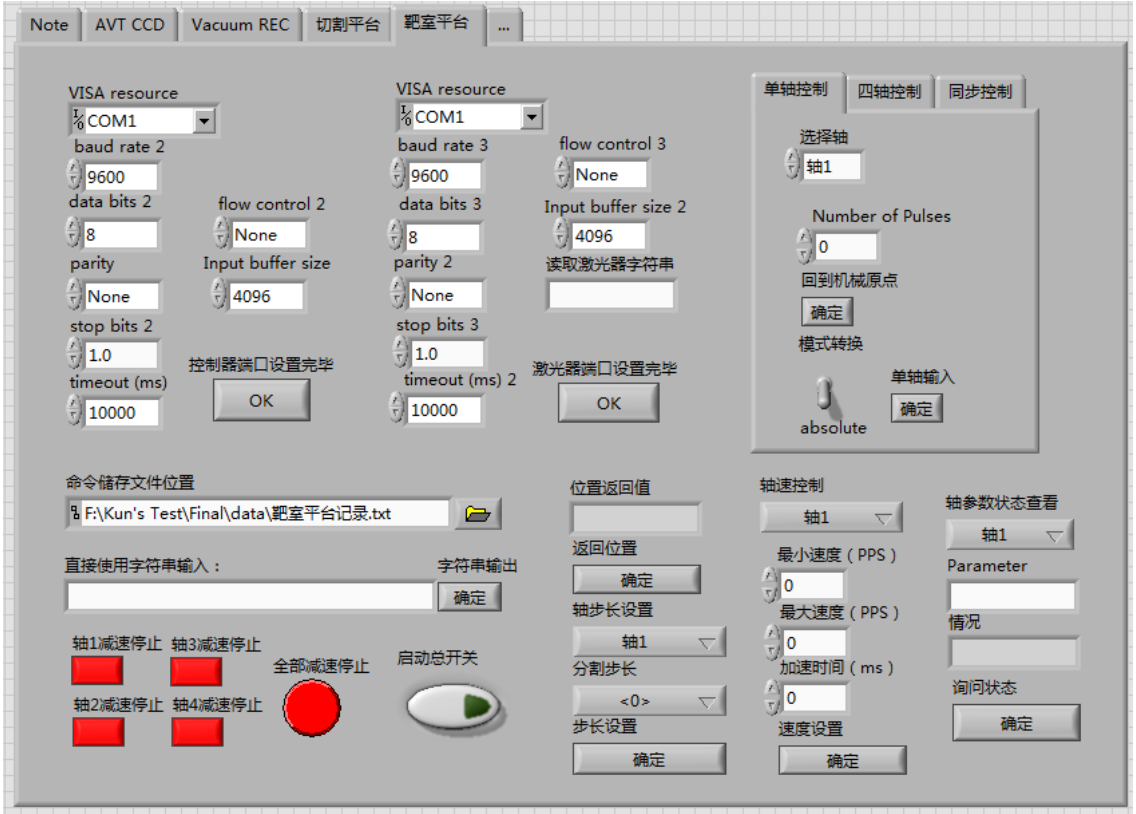

图18 Tolix靶室多轴平台控制软件的界面

4.2 笔记记录与查询部分

为实现笔记的记录与查询发方便快捷，将可以规范化记录的笔记都实现自动命名只用输数据，将仪器的参数在每发次进行时自动记录。

4.2.1 笔记记录与查询的界面设计

通过尽可能的简化笔记记录的流程，将参数设计得尽量简洁，实验者只需输入一些简单的数字就能记录常用的参数。另外，留出较多的空间用于一般笔记的记录，保证在记录时能更直观的看到所记录的信息。同时，通过对文件的扫描和对笔记记录者的搜索，用户可以很快的找到相应的笔记。记录与查询部分的软件界面见图19，从图中可以看到，笔记记录者可以选择自己的名字，相应的笔记会以改实验员的名字命名。通过查询实验员的姓名，可以更快的找到相应的笔记。由于每个人的实验一般不同，更多的时候实验者只关心自己的笔记信息。

4.2.2 笔记记录与查询的底层设计

笔记记录与查询模块LabVIEW都有比较实用的控件对文件的读写查找进行操作。图20为笔记记录查询模块的部分程序框图：

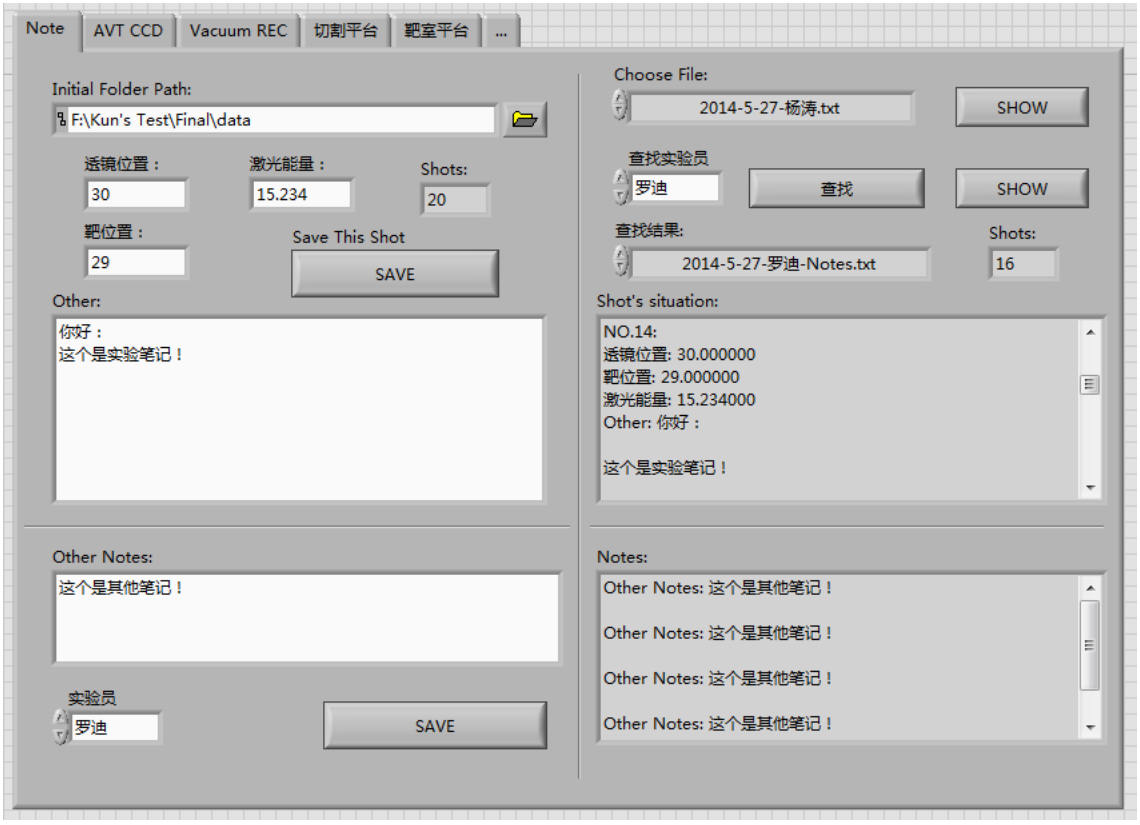

图19 笔记记录与查询模块软件界面

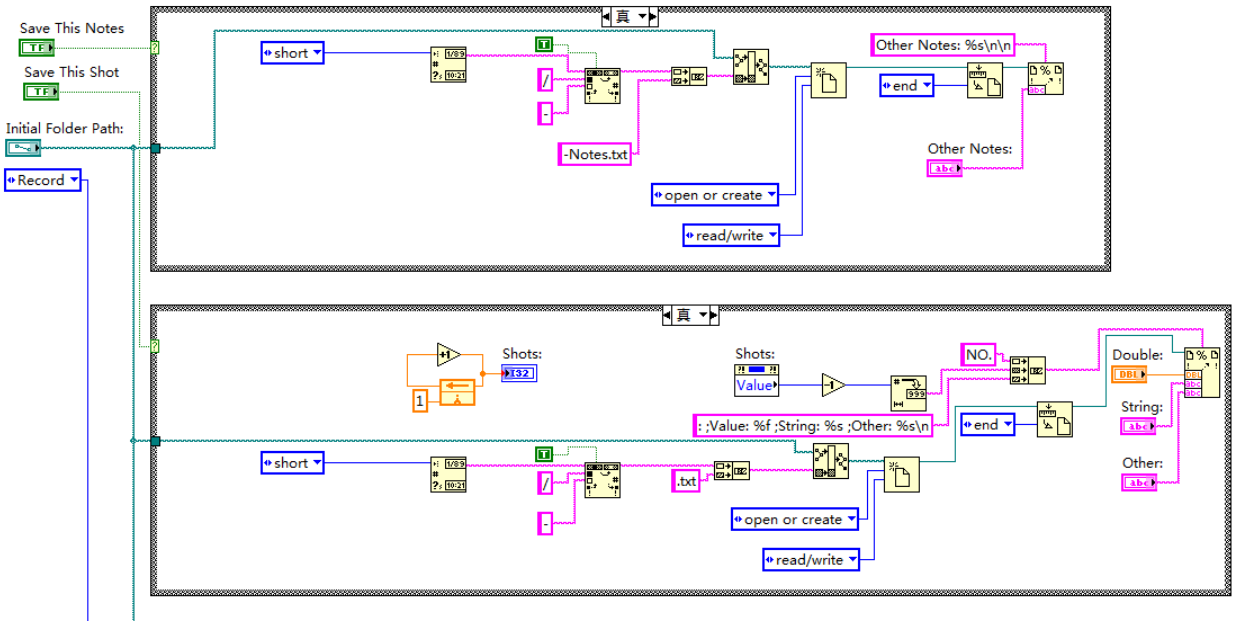

图20 笔记记录与查询模块LabVIEW底层部分框图

上图中利用字符串的连接和读写控件实现了笔记的规范化压缩记录，另外通过文件名扫描和枚举菜单的属性节点实现了文件的查找。

对于之后增加不同格式的输入，软件这边也给出了Double型和String型的记录模式的模板，以便添加更多的记录数据和信息。

4.3 软件整体设计与实现

软件通过将各种仪器的控制软件模块加入一个选项卡中，并将笔记记录和查询模块也涵盖其中。实现了一个软件控制多种仪器，软件也将之前已经写好的一个真空计的控制采集程序整合进去。真空计的采集程序界面如图21：

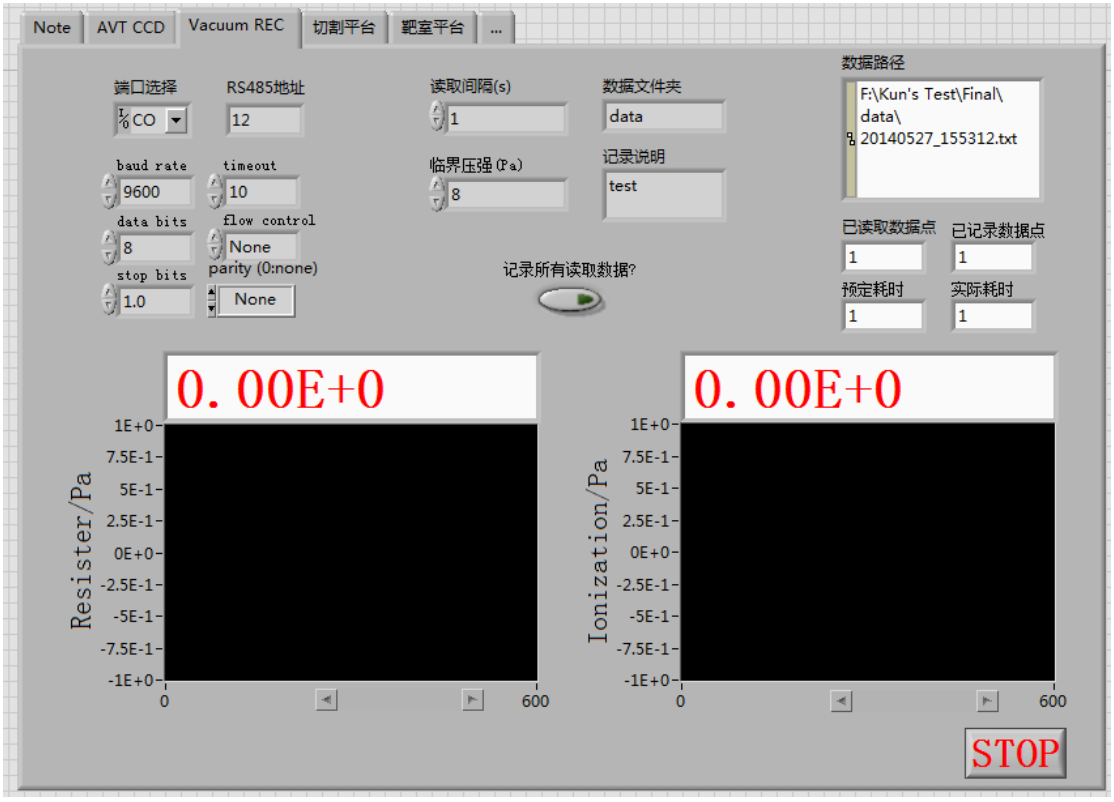

图21 真空计监视及控制软件的界面

通过对不同软件的整合，现整体软件已实现对AVT CCD、真空计、SIGMA KOKI切割平台、Tolix靶室平台的控制，同时还有笔记记录和查询模块。之后，可以简单的通过添加选项卡的方式来实现更多的仪器控制。

整合仪器控制软件的实现，大大减小了实验时由于软件不统一造成的繁琐操作，规范了实验室的笔记记录。使实验室仪器控制更简洁、更快速。这也必然会是之后大量实验室的最终方向。

第五章 总结与展望

该软件通过标准数据流的串口通信和自带控制软件SDK开发两种方法，实现了对不同类型的实验室仪器的控制软件的开发。同时，结合LabVIEW丰富的函数库和直观方便的界面生成，以及LabVIEW与C的混合编程，实现了LabVIEW上的仪器控制整合系统的设计。

为了解决实验室笔记的记录与查询不规范不方便的问题，还将实验室笔记记录及查询模块也整合其中，实现了笔记的规范化记录和更为便捷的查找相关数据参数。

软件不但优化了很多自带软件或者手动操作的不足，也同时将实验室的仪器尽可能的整合到一台电脑一个软件操作。另外，避免了仪器很多数据和结果存放于不同实验电脑需要互相拷贝的问题，减少了很多实验过程中浪费时间和精力的繁琐操作。

但是，与此同时，该软件还存在着不少不足：

（1）由于激光器的控制（或者是光开关的控制）还未整合到软件中，无法及时的读取到激光发射的信号，还很难实现自动记录实验参数的能力，激光信号需要人为给出。

（2）对于不同仪器的兼容性还不够完美，特别是需要LabVIEW与C混合编程的控制程序，自带软件所给的SDK在编译时受到软件生产商的限制太多。

（3）笔记的查询功能还不够完美，检索是通过检索文件名实现的，并未实际检索文件内容，导致无法进行实验记录中的关键词检索。

（4）由于软件处于初级阶段，并未考虑内存消耗等软件性能相关的情况。当笔记容量增加，对于笔记的搜索效率等能力都有待提升。

（5）界面的设计还不够拟物化，尝试将软件界面设计的与仪器相同，但不少仪器本身就不是面板控制的，导致软件的上手不是很快，需要相关的说明文档用以说明。

对于软件未来所需要的工作，一方面需要增加所能控制的仪器个数，努力实现所有实验室仪器一个软件控制；另一方面，要提升软件的性能，随着仪器数目的增加，软件的性能一定会不断降低。如何实现一个更好的平衡，使这类软件能更好的发展也是之后设计改进过程中必须要考虑的。

现在软件只是一个在本实验室的尝试，并未实现一个应用于所有实验室，兼容性很好的普适软件。如果其他实验室需要开发类似的软件需要从零开始，而在之后的开发过程中，是否能实现实验室间的共用同一套软件体系。比如开发相关的仪器接口，如步进电机类、CCD类、激光器类、示波器类等大类通用接口，只需将相应的仪器品牌型号输入，软件便能自动的添加相应的控制单元。

总之，这种仪器整合控制和笔记记录查询软件很大的提高了实验室的工作效率，规范了实验室实验流程，是未来实验室仪器控制软件发展的大方向。

参 考 文 献

1. 吕向锋, 高洪林, 马亮, 等. 基于 LabVIEW 串口通信的研究[J]. 国外电子测量技术, 2009 (12): 27-30.
2. 王春霞. 基于 LabVIEW 的控制实验系统的设计与开发[D]. 东北大学, 2008.
3. 李永红, 程耀瑜, 隋海洋. 基于 SDK 的软件设计方法[J]. 科技情报开发与经济, 2006, 16(6): 228-230.
4. 赵思奇, 王仕秀. 利用 C 语言的 SDK 开发测试管理系统[J]. 电脑编程技巧与维护, 2008 (8): 18-20.
5. 李扬, 郑莹娜. 图形化编程语言 LabVIEW 环境及其开放性[J]. 计算机工程, 1999, 25(4): 63-65.
6. 李俊, 陈湘波. LabVIEW 与 C 语言的混合编程[J]. 自动化与仪器仪表, 2001 (5): 62-64.
7. 施雅婷, 郭前岗, 周西峰. 一种改进的 LabVIEW 串口通信系统的实现[J]. 电子测试, 2010 (8): 64-69.
8. 周红霞, 张恒杰, 张春芳. 基于 LabVIEW 的虚拟仪器及串口通信的实现[J]. 石家庄职业技术学院学报, 2007, 19(4): 17-19.
9. 董智强, 徐世荣, 余雅敏. 基于 VISA 的串口通信程序设计[J]. 电脑知识与技术, 2008, 6: 95-98.
10. 吴亮, 高峰, 李俊杰, 等. 基于 LabVIEW 的通用仪器控制软件设计[J]. 舰船科学技术, 2009, 31(10): 73-75.
11. 郭占山, 张德山. 在 LabVIEW 下仪器控制的实现方法[J]. 计量技术, 2002 (7): 24-26.
12. Allied Vision Technologies. *Camera and Driver Feathers, V1.0.1*. 2013, 9.
13. Allied Vision Technologies. *PvAPI Programer's Reference Manual, V1.26*. 2013, 7.
14. SIGMA-KOKI. *SHOT-202/204MS User's Manual, 3rd Edition*. 2008,9.

附录一：底层DLL源代码

```
// Function_Set.cpp : Defines the entry point for the DLL application.
//
/*
|
============================================================================
===
| Copyright (C) 2006-2011 Allied Vision Technologies.  All Rights Reserved.
|
| This code may be used in part, or in whole for your application development.
|
|
============================================================================
===
|
============================================================================
===
| Copyright (C) 2014-2015 Chenkun Wang.  All Rights Reserved.
|
| This code may be used in part, or in whole for your application development.
|
|
============================================================================
===
*/
#include "stdafx.h"

#include <stdio.h>
#include <stdlib.h>
#include <string>

#define WIN32_LEAN_AND_MEAN
#include <Windows.h>

#include "PvApi.h"

using namespace std;

extern "C" __declspec(dllexport)void CameraStart();

extern "C" __declspec(dllexport)void CameraStop();

extern "C" __declspec(dllexport)string CameraModeSet(int EM, int EV, int AM, int FSTM);

extern "C" __declspec(dllexport)string CameraModeGet(char* EM, unsigned long EV, char* AM,
char* FSTM);


BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                                             )
{
    return TRUE;
}
```

```
#define _STDCALL __stdcall

#define FRAMESCOUNT 15

typedef struct
{
   unsigned long   UID;
   tPvHandle       Handle;
   tPvFrame        Frames[FRAMESCOUNT];
   bool            Abort;

} tCamera;

tCamera GCamera;

bool CameraGet(string SerialNo)
{
   tPvUint32 count,connected;
   tPvCameraInfoEx list;
         count = PvCameraListEx(&list,1,&connected, sizeof(tPvCameraInfoEx));
   if(count == 1)
   {
      GCamera.UID = list.UniqueId;
      SerialNo=list.SerialNumber;
   }
         return true;
}

bool CameraSetup()
{
         bool failed = false;
         unsigned long FrameSize = 0;
         PvCameraOpen(GCamera.UID,ePvAccessMaster,&(GCamera.Handle));
         PvAttrUint32Get(GCamera.Handle,"TotalBytesPerFrame",&FrameSize);
         for(int i=0;i<FRAMESCOUNT && !failed;i++)
         {
                  GCamera.Frames[i].ImageBuffer = new char[FrameSize];
                  GCamera.Frames[i].ImageBufferSize = FrameSize;
         }
         return !failed;
}

void CameraUnsetup()
{
         PvCameraClose(GCamera.Handle);

   for(int i=0;i<FRAMESCOUNT;i++)
      delete [] (char*)GCamera.Frames[i].ImageBuffer;

   GCamera.Handle = NULL;
}

bool CameraSet()
{
         bool failed = false;
         PvCaptureAdjustPacketSize(GCamera.Handle,8228);
         PvCaptureStart(GCamera.Handle);
```

```
	for(int i=0;i<FRAMESCOUNT && !failed;i++)
	{
		PvCaptureQueueFrame(GCamera.Handle,&(GCamera.Frames[i]),NULL);
	}
	if (failed)
		return false;

	return true;
}

void CameraStart()
{
	PvCommandRun(GCamera.Handle,"AcquisitionStart");
}

void CameraStop()
{
	PvCommandRun(GCamera.Handle,"AcquisitionStop");
	Sleep(200);
	PvCaptureQueueClear(GCamera.Handle);
	PvCaptureEnd(GCamera.Handle);
}


string CameraModeSet(int EM, int EV, int AM, int FSTM)
{
	string SerialNo;
	PvInitialize();
	memset(&GCamera,0,sizeof(tCamera));
	CameraGet(SerialNo);
	CameraSetup();
	CameraSet();
	switch(EM)
	{
		case 1: PvAttrEnumSet(GCamera.Handle,"ExposureMode","Manual");
		case 2: PvAttrEnumSet(GCamera.Handle,"ExposureMode","Auto");
		case 3: PvAttrEnumSet(GCamera.Handle,"ExposureMode","AutoOnece");
		case 4: PvAttrEnumSet(GCamera.Handle,"ExposureMode","External");
	}
	PvAttrUint32Set(GCamera.Handle,"ExposureValue",EV);
	switch(AM)
	{
		case 1: PvAttrEnumSet(GCamera.Handle,"AcquisitionMode","Continuous");
		case 2: PvAttrEnumSet(GCamera.Handle,"AcquisitionMode","SingleFrame");
		case 3: PvAttrEnumSet(GCamera.Handle,"AcquisitionMode","MultiFrame");
		case 4: PvAttrEnumSet(GCamera.Handle,"AcquisitionMode","Recorder");
	}
	switch(FSTM)
	{
		case 1: PvAttrEnumSet(GCamera.Handle,"FrameStartTriggerMode","Freerun");
		case 2: PvAttrEnumSet(GCamera.Handle,"FrameStartTriggerMode","SyncIn1");
		case 3: PvAttrEnumSet(GCamera.Handle,"FrameStartTriggerMode","SyncIn2");
		case 4: PvAttrEnumSet(GCamera.Handle,"FrameStartTriggerMode","FixedRate");
		case 5: PvAttrEnumSet(GCamera.Handle,"FrameStartTriggerMode","Software");
	}
	/*This place is used for other functions
```

```
        */
        return SerialNo;
}


string CameraModeGet(char* EM, unsigned long EV, char* AM, char* FSTM)
{
        string SerialNo;
        PvInitialize();
        memset(&GCamera,0,sizeof(tCamera));
        CameraGet(SerialNo);
        CameraSetup();
        CameraSet();
        PvAttrEnumGet(GCamera.Handle,"ExposureMode",EM,NULL,NULL);
        PvAttrUint32Get(GCamera.Handle,"ExposureValue",&EV);
        PvAttrEnumGet(GCamera.Handle,"AcquisitionMode",AM,NULL,NULL);
        PvAttrEnumGet(GCamera.Handle,"FrameStartTriggerMode",FSTM,NULL,NULL);
        /*This place is used for other functions



        */
        return SerialNo;
}
```