

串口可以说是我们最容易见到，也最容易接触到的一种总线，台式机上一般都有二个，而现在很多下位机、仪器等很多都还是使用串口通信的。论坛上很多朋友都经常会使用到串口，并遇到一些问题，这里有必要做一个详细的说明，以方便广大会员朋友方便使用。

首先补充一个比较重要的问题，就是在 LabVIEW 中使用串口的话一定要先安装 VISA 这个驱动，然后生成的 EXE 运行时也需要在目标机上安装 VISA Runtime Engine，可以在打包时一起打包。

1.串口扩展的问题:

先说一下串口的扩展问题，一般的台式机或工控机上都至少有二个串口，一般都是够用的，但是现在市场上已经很难找到带串口的笔记本了，而有时候在外出调试时需要在笔记本中使用到串口的，这时一般是使用 **USB-RS232 的转接线**，价格从十几到一百多都有，很多朋友反应在使用价格低的转接线时会出现乱七八糟的问题，而贵一点的线就很少听说有其它问题的，所以大家在使用便宜的 USB-RS232 转接线时要特别注意线的质量，遇到一些奇怪的问题时先考虑换一根好一点的线。PCI-RS232 扩展卡也同理，便宜的卡也容易出问题，尽量买好一点的，以免因小失大。PCI-RS232 一般至少能扩展 2 个串口，有些 BT 一点的可以扩展到 8-16 个，一堆线和接头。转接线和扩展卡一般是要装驱动的。

2.串口功能的确认:

在使用串口之前，最好先确认一下串口是否正常，特别是使用转换接或扩展卡的。检查的方法很简单，就是将串口的 2、3 脚短接起来，3 脚是发送数据，2 脚接收数据，就是这个串口自发自收。电脑上的串口软件一般是用串口调试助手，很出名的，也好用。如下图所示：



图 1 串口调试助手

打开软件，选择已经短接好的串口号，点击“手动发送”，如果串口是好的，2、3脚又短接起来了，马上就可以在上方的接收框里看到接收的数据就是发送的数据。稍微要注意一下的是有些电脑的 COM1 和 COM2 的位置是反过来的，所以要确定好串口调试助手左上角的串口上择的是已经短接的那一个，如果 COM1 没接收到，可以再先 COM2 再发一次看一下。

3.串口线的检查:

检查好串口后，一般也要注意一下使用的串口线，标准的串口线是 9 根线都是用上，但有一些是只使用了三根线的：2、3、5。**第 3 个脚管是发送，第 2 个管脚接收，另一个 5 是地线，这里叫它简化的串口线**，简化的串口线能用上的地方，标准的串口线也肯定能用上，因为标准线的 9 根线已经包括了简化串口线中的 3 根线，但标准串口线能用上的地方，简化串口线就不一定能用上，所以在使用串口线之间一定要确定好串口线的类型，一般买的串口线都是标准线，但自制的串口线因为应用场合不同就要先确定一下。

G 串口线还有一个地方需要注意一下的，就是 2、3 脚的接法，标准接法中是 2、3 脚交叉的，即这边的 2 接另一边的 3，这边的 3 接另一边的 2，扭了一下，所以叫**交叉线**，因为正常使用时，这边第二脚是发送数据，另一边第三脚是接收数据，所以要将这二个管脚连接起来，这样才能正常使用。但是有些情况下，2、3 是直连的，即这边的第 2 脚连接另一边的第 2 脚，第 3 脚连接第 3 脚，**这种叫直连线**，这种线一般是用于**延长串口的**，比如需要将工控机的串口接头引到机柜表

面上时，就使用这种线，这样机柜表面的串口线的定义还是跟电脑接出来的一样，外面的那一根串口线再使用交叉线。从电脑主板上将串口引到主机后面板上的线就是这样的直连线。购买串口线的时候一般也会问你买直连的还是交叉的，要区别对待。

串口线还有一个要稍微注意一下的就是 DB 头，因为电脑上接出来的一般是公头（针），要跟电脑接的话要母头（孔），一般仪器的串口也是公头，所以二边都是母头的串口线比较常见。串口的接头一般是 DB9 的，也有 DB25 的，但比较少用，有些比较 BT 的仪器厂家居然用 RJ11（水晶头那种）作为串口头，让人不爽！

总之，使用串口前一定要先确定好硬件没问题，不然很浪费时间的。

4.串口参数设置:

在 LabVIEW 中使用串口时，有几个参数比较重要，需要先说明一下的。一个是串口初始化这个节点的“终止符”和“禁用终止符？”这二个输入端，这二个输入端是相互作用的，“终止符”默认值为 10，它的十六进制是“0x0A”，这是一个 ASCII 码，是一个换行符，可以从 LabVIEW 中的字符串的不同显示形式看出来，如下图：

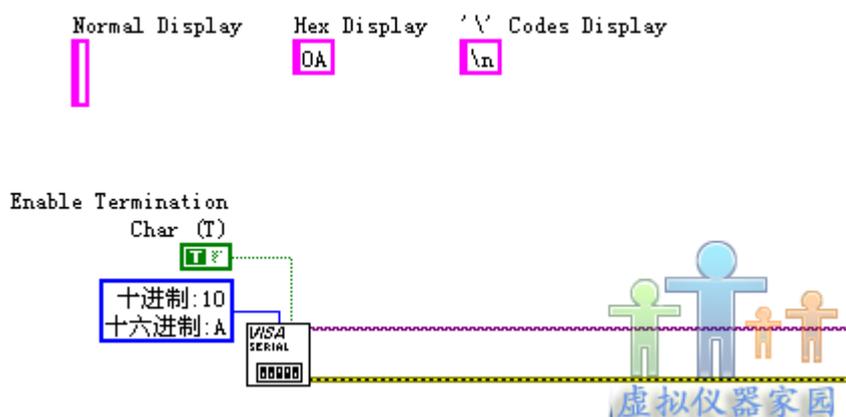


图 2 串口消息终止符

左边是字符串的正常显示，中间是十六进制显示，右边是“\代码显示”，这三个字符串的值都是一样的。终止符是 10，表示在接收数据时，遇到 ASCII 码为 10 的字符（即换行符）时就停止接收数据，后面会有例子进行说明。

而“禁用终止符？”的默认值是 FALSE，即启用终止符，启用终止符会有什么效果呢？终止符的意思就是当程序接收到这个字符时，就认为已经到了所有数据的

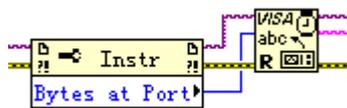


图 4 使用 Bytes at Port

5.在 LabVIEW 中使用串口

目前串口的应用一般有二种类型的（以我接触到的来分类，不严格），一种是仪器控制类型的，一般是上位机发送一个指令，然后下位机作出响应，返回数据给上位机，上位机再读取出来，完成一次通信，即一问一答；另一类是被动接收形的，即下位机会一直发送数据上来。这二种类型的串口通信在处理上会不太一样。

5.1 仪器控制类型

由于在仪器控制时一般都是这种一问一答的方式，所以叫它仪器控制类型。以仪器控制为例来说一下需要注意的事项。

首先是要确认仪器选择的通信模式是串口通信模式。现在的仪器一般都至少有二种通信模式，一种是 RS232，一种是 GPIB，如果仪器是设置为 GPIB 通信的话，RS232 是不可能通信上的，所以要先确认一下，方法一般是在仪器面板上选择设置>>远程控制>>GPIB/RS232，各个仪器稍微不同，可以查仪器手册看一下。

然后就是确认串口的通信参数的配置，包括波特率、数据长度、校验方法等，有些仪器的某些参数是固定的，比如校验方法固定为奇校验，不能修改，只能在电脑上跟仪器设置为一样的。波特率一般是可以修改的。这些参数的配置一定要根据仪器手册上的来设置。如果参数设置不正确，也能收到一些数据，但一般是乱码，如果收到的数据都是乱码的话，就要先检查一下串口参数设置是否正确了。只有电脑和仪器边的串口参数完全一致时才能收到正确的数据。

接着是要注意发送指令和读回数据之间要有一定的延时，即 VISA WRITE 和 VISA READ 之间要有一定的延时，一般 200 毫秒即可，因为串口是底层硬件，数据从软件到串口上要一点点时间，然后仪器对指令作出响应也要一点点时间，这些时间加起来肯定比软件运行二个节点的时间要短，所以延时是一定要加的。在调试时如果发现正常运行时不能收到数据，但高亮运行就能收到数据，就很有可能是没有加延时的原因！

或者是发一个查询指令，但返回的是上一条查询指令的结果，也可能是因为没有延时或延时不够。

5.2 被动接收类型

被动接收形的串口通信稍微麻烦一点，由于上位机是被动接收的，上位机不知道什么时候开始下位机就已经有数据上来了，很有可能下位机发送到一半时，上位机刚好开始接收数据，这时只能接收到后面一半的数据了，所以对于这种通信，一般是采用数据帧的方式进行通信。

这种数据帧的通信方式至少由三部分数据组成：帧头、数据、帧尾（如果数据是固定长度的话，似乎帧尾也可以省掉）。

帧头是为了告诉上位机：从这以后的数据就是有用的数据了，相当于约定好的暗号，一般帧头至少会用二个以上字节，如果只用一个字节的话，万一数据中的数据跟这个帧头一样了就会误以为这个数据是帧头从而导致解析数据出错，帧尾的作用也差不多，告诉上位机从这之前的数据才是有用的数据。但实际上一般的数据帧远不止这几个部分，还会加上一些校验字节、时间信息、帧计数器之类的东东在上面。

其中校验字节是为了检查数据在传输过程中有没有出错的，跟串口的校验位要区分清楚，校验位也是检查数据传输时有没有出错的，但由底层硬件来实现，校验方法由标准规定好，但有几种可以选择，只有一个位（Bit，只能是0或1）。

校验字节是由软件层来实现的，至少有一个字节（Byte,有8个位），而且校验方式由用户定义，非常灵活。

由于被动方式中串口的缓冲区中一直会有数据在，为了保持数据的连续性，在读取数据时跟第一种仪器控制类型不一样。而是采取将读取的所有串口数据都保存在移位寄存器中，在软件上处理完这些数据后再将它们从移位寄存器中删除。

由于VISA READ的输出是字符串，所以一般使用“连接字符串”这个函数将它们连接起来，然后接到循环结构中的移位寄存器中进行保存，当移位寄存器中的数据量达到一定时或满足数据处理的条件时，才停止这个循环输出读取到的数据。一般如下图所示：

图5 被动接收类型中的数据接收

在接收下位机发送的帧数据时，一定要先了解帧格式，这样才能正确解析出帧里面的数据来。

下面以例子来说明数据帧格式的通信。

设定通信数据帧每7个字节为一帧数据，其中以0xAC、0x96二个字节作为数据帧头，第三、四个字节为帧计数器，最大值为0xFFFF，到达最大值后重新从0开始计数，第4、5、6三个字节是数据信息，分别代表数据的高中低位，第7位为状态标志字节，它的第一位为1时表示下位机出错，为0时表示功能正常。

由于LabVIEW中接收到的数据都是以字符串的形式显示出来的，所以需要将字符串转换为ASCII码，一般可以直接使用“转换为U8数组”这个函数，如下图所示：

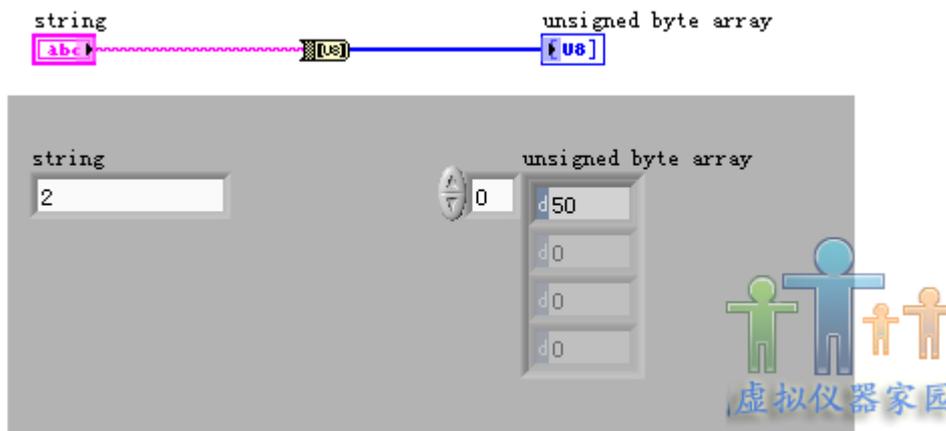


图 5 字符串转换为 U8 字节

转换为 U8 字节后，得到的是所传输字符的 ASCII 码，我们就很容易进行数据帧的判断了，现收到以下的字符串数据：

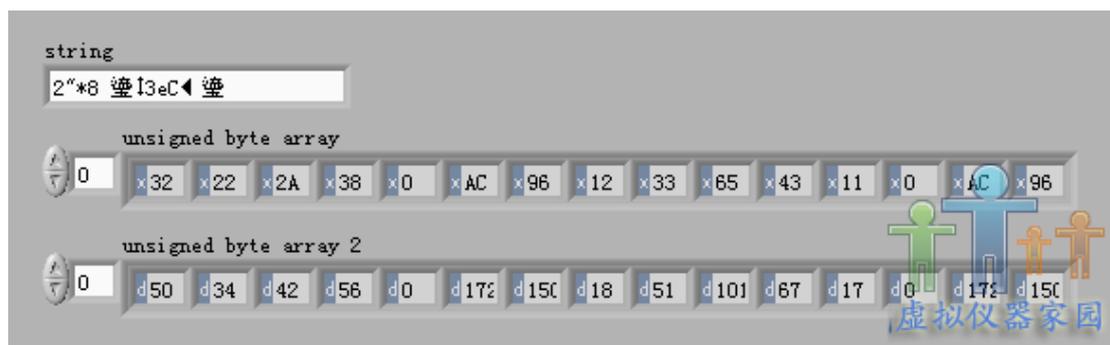


图 6 实际接收到的字符串

上图中下半部分显示的数组是使用“字符串转换为 U8 数组”的函数转换之后得到的数组，一个是十六进制显示，另一个为十进制显示。对照定义的数据帧格式，就很容易得到我们需要的数据了。

首先是要看从哪里开始才是完整的第一帧，从上面十六进制显示的数组中我们可以看到，并不是第一个字节就是我们需要的帧头，因为下位机是一直处于发送数据的状态，很可能在串口发送一帧数据的过程中串口就被初始化或者被清空了一次缓冲区，那么这一帧数据的前面部分数据可能就会丢失，只留下后面一部分数据。

以上图为例，第一二个字节为 0x32、0x22，显示不是我们要的帧头，我们要的帧头是在第 6、7 个字节，以程序来实现的话就是先查找第一个帧头，使用“搜索字符串”，如果找到则判断它下一个字节是否是第二个帧头，如果是，表明已经找到帧头，输出帧头的位置；如果它下一个字节不是第二个帧头，说明这里不是真正的帧头，继续查找下一个帧头，直到找到帧头或搜索整个字符串都找不到帧头。

这是一个程序的算法问题，具体实现的程序如下图所示：

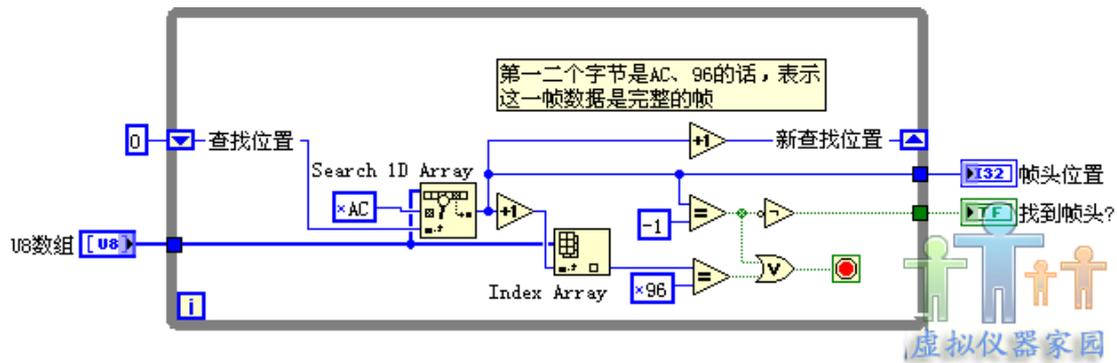


图 7 帧头查找程序

帧头查找到以后，再找数据就容易了，根据之前的定义，第 4、5、6 个字节是才是我们要的数据，所以直接使用索引号进行索引输出即可。

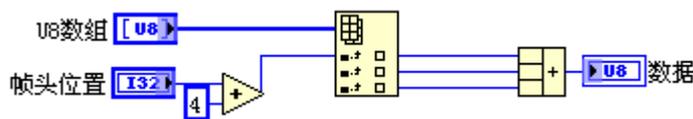


图 8 获取实际数据

一般情况下，如果是用三个字节表示一个数据的话，那么这三个字节分别表示为一个数据的高中低字节，即高字节要乘以 25536 再加上中字节乘以 256 再加上低字节的，这样定义后可表示的数据的范围就会扩大很多，但这里为了说明问题，直接认为三个字节的数据相加就是我们想要的实际数据，在实际使用过程中应该根据帧格式的字义来解析这个数据。

另外帧格式中定义了最后一个字节为状态标志位，所以提取数据前还要检查一下这个标志位是否正常，不正常时要进行相应的处理，这里不再详细描述。

至此完成一次数据帧的提取。

如果是没什么特殊的要求的话，这里应该也算到一段落了，有一些对测试时间有要求的地方，就会要求在最短的时间内得到最多的信息。图 6 中我们可以看到，接收到的数据帧中，除了中间一个完整的帧之外，头尾还有一些无用的数据，其实这些数据中也包含了有用的信息的！

比如我们可以从 0xAC、0x96 这二个帧的位置中推断出它前面的 0x22、0x2A、0x38 这三个字节也是我们想要的的数据字节，但是由于没有接收到它的帧头，所以程序没能提取出来，但我们可以从后一帧的帧头推算出前面那一帧的数据字节是哪些，即使没收到前面那一帧的帧这里只给出一个流程，不再给出具体的程序。

另外有可能接收的数据长度比较长，可能就不止包含了一帧的数据在里面，所以在程序中也要判断一下剩下的数据还够不够一帧的数据长度，如果够则可以根据上一次查找的帧头位置+数据帧长度来确定下一个数据帧的帧头位置了，不需要使用搜索的方法。也可能存在处理完一帧数据后，剩下的数据不够一个帧的，这时可以将这些剩下的数据保留起来，将它添加到下一次接收到的数据前面，组成新的数据再进行处理。去掉已经处理的数据可以使用“删除数组元素”这个函数来实现。这里也不再给出具体的程序。

6. 串口数据类型的转换

由于 LabVIEW 中 VISA Read/Write 这二个函数都是只能读取/写入字符串类型的数据的，而有时候需要接收/写入的数据类型不一定是字符串，导致在刚开始接触的时候会有一点困惑。

在进行数据转换时，只要记住计算机中所有数据都是以二进制保存这个原则就容易解决问题了。串口线上传输的也是高低，串口接收到的也是二进制数据，只是到 LabVIEW 后被转换为字符串格式了。还是以例子进行解释。

6.1 LabVIEW 从串口接收数据

①假设 LabVIEW 从串口接收到的数据为“1234”（正常显示模式下），那么这个数据在串口底层的时候其实是这样的二进制数据：

```
00110001 00110010 00110011 00110100
```

只是在 LabVIEW 中，这些二进制数据是以字符串形式显示出来的，它们的实质还是二进制数据，这几个二进制数据转换为十进制数据分别是“49，50，51，52”，由于字符串都是以 ASCII 码形式保存在计算机中的，那么 49，50，51，52 这几个数在 ASCII 表中就表示是字符串“1，2，3，4”。所以这几个数据在 LabVIEW 中就显示为字符串的 1，2，3，4 了。

如果明白这里面的转换关系，那么要进行数据转换时就很容易了，比如上面的例子中，如果 LabVIEW 中接收到的是字符串“1234”，而原本下位机传送的是数值型数据，只需要将“1234”字符串转换为对应的 ASCII 值就是实际上下位机传上来的数据了，就是“49，50，51，

LabVIEW 中将字符串转换为对应的 ASCII 值的函数是“字符串至字节数组转换”这个函数：

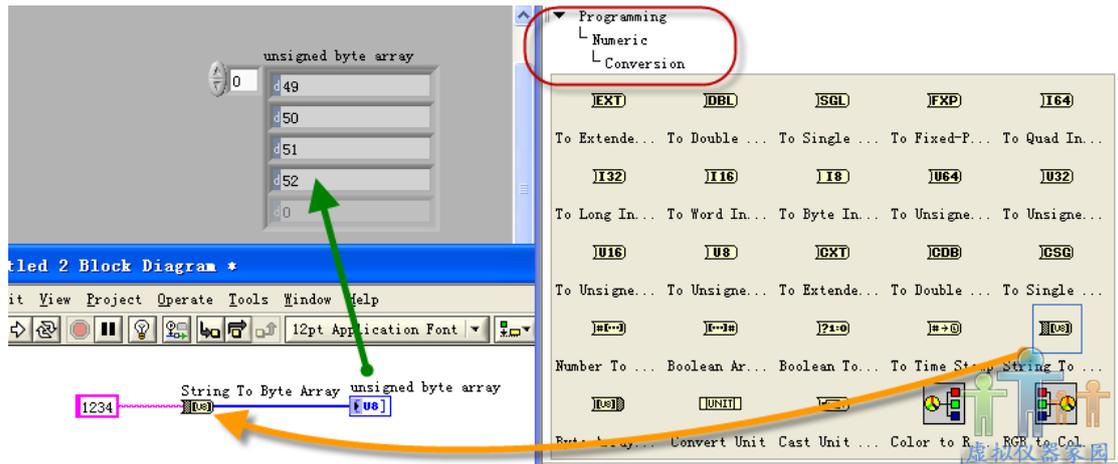


图 9 字符串转换为字节数组

上面说的是下位机发送的是数值类型的数据的，使用“字符串至字节数组转换”这个函数。

②如果是下位面发送的是字符串类型的数据，那么 LabVIEW 已经直接转换好了。

个人总结：无论下位机发送的是何种类型的数据，LABVIEW 从串口接收的时候，总是将每 8 位二进制数，作为字符的 ASCII 码解读。

6.2 使用 LabVIEW 发送数据

还有一个问题是使用 LabVIEW 发送数据的问题

①如果下位机接收的是字符串数据的话，直接用 VISA 写入对应的字符串就行了，现在的仪器一般都是接收字符串的，所以可以直接使用 VISA 发送而不需要转换。

②如果下位机接收的是数值型数据的话，就需要转换一下，其中数值型又是十进制和十六进制二种用得比较多，这二种数据间相互转换一下就行了，其实是一样的。

由于在 LabVIEW 中字符串直接有十六进制的显示方式，所以发送十六进制的数据比较方便，比如要发送十六进制数值类型的“0xAF”，那么在 VISA Write 的写入缓冲区字符串常量上右键>>十六进制显示，如图 1，直接输入“AF”即可，那么下位机接收到的就是正确的数据（十六进制数值类型）了。

但实际使用过程中，一般都是需要将某个子 VI 输出一个动态的字符串通过 VISA Write 发送到下位机的，这时候就需要对数据进行转换一下，这个转换过程描述起来就是：将字符串 A 转换为字符串 B，使得正常显示的字符串 A 跟十六进制显示的字符串 B 是一样的。

由于转换目标（十六进制显示的字符串）的数据类型是十六进制，要想十六进制显示的字符串跟正常显示的字符串一样，这个正常显示的字符串必须都是十六进制的字符，即只能由 0-9,A-F 这十六个字母中的字母组合而成。否则就没办法使二种显示方式的字符串一致了。

这个转换过程首先将字符串转换为十六进制数值型，然后再通过将这个十六进制数值创建一个数组，最后再使用“字节数组至字符串转换”这个函数转换为字符串即可，实际上就是图 5 字符串转换为 U8 字节的反向操作，只不过是这个十六进制的值初始类型是十六进制，要先转换为数值类型。具体程序如下图所示。

图 10 正常显示字符串转换为相同的十六进制显示的字符串

由于十六进制数据由二个字节构成，而字符只有一个字符，所以每二个字符表示一个十六进制数据，如果字符多于二个的话要先进行截取，每二个字符转换为一个十六进制数据。也可以用空格将正常显示的字符串每二个字符用一个空格断开，然后先将这个字符串以空格为分隔符转换为一个字符串数组，再转换为十六进制数值再转换为字符串。

需要注意一下的是如果正常显示的字符串并不是 2 的整数倍，那么上图的转换程序就会少转换一个字符，可以用程序动态判断一下这个字符串的长度，如果是奇数的话在它最左边补一个“0”再使用上面的程序就正常了。

7.串口问题汇总：

7.1 串口资源被占用：

这时候在 LabVIEW 会报错，提示串口号存在，但当前不能对其进行操作，同时打开 MAX 时也可以在对应的串口号下看到同样的错误，这表示这个串口已经被其它程序占用了，比如有时候打开了串口调试助手来调试串口，然后又想在 LabVIEW 里面试一下，这时就会报这个错，因为串口已经被串口调试助手调用了，它不能被二个程序同时使用。

解决的方法是关掉其它程序即可，串口调试助手里也可以关闭这个串口。

还有一种情况是调试 OK 后生成 EXE，运行 EXE 也出现这个问题，这时是因为串口被原来的 LabVIEW 程序打开，再用 EXE 打开时就会报错，解决办法是关掉原来的 LabVIEW 程序。最好是关掉 LabVIEW。

在使用串口的过程中一定要关闭串口（使用 VISA CLOSE），否则程序在退出的时候会报错说数据丢失。