文章编号:1671-4598(2014)08-2644-03　　中图分类号:TP312JA　　文献标识码:A

基于 Java 异步串行通信技术的研究

马　赛，王　忠，陈　典
(四川大学 电气信息学院，成都　610065)

摘要：Java Communication API 是 Java 平台下用于串口通信程序开发的唯一官方解决方案，Java 串口通信程序均可以由官方提供的开发包中的示例程序括展；但以示例程序为基础开发的 Java 串口通信程序一般都存在着在异步通信模式下无法完整接收长串数据的缺陷；为解决这一问题，根据通信前收发信双方可以约定通信的数据长度这一特点，提出一种对接收到的字节进行计数的方法，以接收的数据长度为指标判断通信是否结束；通过实际编程验证，该方法克服了上述缺陷，具有很好的可行性与实用性，可以广泛应用于各种使用 Java Communication API 进行开发的场合。

关键词：Java Communications API；comm. jar；异步串行通信

Study of Asynchronous Serial Communication Technology Based on Java Communication API

Ma Sai, Wang Zhong, Chen Dian
(School of Electrical and Information Engineering, Sichuan University, Chengdu　610065, China)

Abstract: Java Communication API is the only official solution for serial communication. Java serial communication program can be developed based on the sample in the Java Communication API developer toolkit, but such program has the defect that it cannot receive long data string continuously in asynchronous mode. In order to solve this problem, according to the characteristic that transceiver both sides know the length of data to be transferred in asynchronous mode, proposed a method which is to count the number of bytes received, using the counted number as the criteria to judge whether the communication is ended. Verified by programming, the method overcomes the shortcoming above. It is proved that the method has good feasibility and practicality, and can be widely used in various programming using the Java Communication API.

Keywords: Java communications API; comm. jar; asynchronous serial communication

0　引言

异步串行通信是一种常用的通信模式，许多数据采集终端均使用这种模式与上位机进行通信。异步串行通信的特点之一是发送端与接收端之间没有同步时钟，并以彼此约定的波特率作为同步。这种同步方式会导致通信数据中每个字节的时间间隔不一致。基于 Java Communications API 开发的通信程序在异步串行通信模式下由于上述原因会出现长串数据通信不连续的问题。即程序在一次性接收长串数据时由于数据中字节间隔不同，会将长串数据截成几段进行接收，无法保证接收数据的连续性。

为解决上述问题，本文提出一种对接收数据进行计数的解决方法。此方法使用一个类成员整型变量对接收到的数据进行计数，同时将存储接收数据的数组作为一个类成员变量，而不是将其作为示例程序中的局部变量。这样即使接收的数据在未传送完毕时被截断，程序已接收到的数据也可以被保留，并且可以在未传送完的数据到来后被继续接收，保证程序可以接收到完整连续的数据，保持了通信数据的连续性。

收稿日期:2014-05-13；　**修回日期**:2014-06-10。

基金项目:航空科学基金项目(20100119004)；国家级大学生创新创业训练计划项目(201310610109)。

作者简介:马　赛(1992-)，男，北京人，本科，主要从事信号与信息处理方向的研究。

1　异步串行通信程序

串口通信程序是一类重要的计算机程序，开发人员通常会选择 C++或者 VB 进行此类程序的开发。但在 Java 平台上，这些语言无法直接使用。而且 Java 程序运行在虚拟机之中，访问串口比较困难。为解决这一问题，SUN 于 1997 年发布了应用于 Java 串并口编程的 Java Communications API[1]。该标准是 Java 平台上用于串并口通信程序编程的唯一官方标准。SUN 同时给出了一个实现此标准的通信程序开发包，文件名为 Javacomm20－win32. zip。开发包中包含了实用的示例程序，其中 Simple 目录下的 SimpleRead. java 和 SimpleWrite. Java 是最为基本的串口数据接收程序与发送程序的源代码文件，一般功能的 Java Communications API 通信程序均由它们扩展而来。文献[2-6]给出了 Java Communications API 在各种场合的应用。

Java Communications API 的核心思想就是通过中间媒介让用户对通信端口实现符合 Java 标准的 I/O 操作，而 API 实现的中间媒介是对用户不可见的。程序需通过一些与计算机底层交互的配置类获得对应通信端口的 I/O 流。文献[7]对 Java Communications API 的组成及使用方法做了详细说明。

通过 SerialPort 实例便可以获得对接在通信端口上的 I/O 流。如果对串口进行写操作，直接向输出流写入信息即可。接

收信息的实现则需要采用监听器模式，需要实现 Java Communications API 中的接口 SerialPortEventListener。监听器模式的实现需要启动一个线程，让程序使用一个独立的过程专门来监听和处理串口事件，因此串口通信类还要实现 Runnable 接口。

I/O 流的基本操作单位为字节，因此接收数据的方法中需创建一个字节数组，用于存储接收数据。接收数据程序源文件 SimpleRead. java 中的部分代码[8]如下所示（摘自 serialEvent（）方法体）：

```
byte[] readBuffer = new byte[20];
while (inputStream.available() > 0) {
    intnumBytes = inputStream.read(readBuffer);}
```

方法 serialEvent（）中有一个长度为 20 字节的 byte 数组局部变量作为接收数据缓存，采用一个 while 循环对信息进行接收。如果 inputStream 中存在有效字节，就调用 inputStream 的 read（byte []）方法读取，接收完毕退出循环。文献[9]指出的 read（byte []）方法原型如下所示：

```
public int read(byte[] b)throws IOException
```

此方法用于读入一定数目的字节。其传入参数为读入数据的缓冲区，返回数据为读取到的字节数目。由于异步通信模式中没有表示传输结束的控制信号，并且程序使用事件触发机制对数据进行接收，一次事件传送多少字节并不明确，因此示例程序中给出的 read（）方法不适合用在异步通信中。

InputStream 类中有一个逐个读取字节的 read（）方法。此方法在有输入数据时从输入流中读取下一个字节。返回 int 型数据，其低 8 位包含的是读取到的字节型数据。由此改进的接收方法源代码如下所示：

```
int pointer = 0;
while (inputStream.available() > 0) {
    readBuffer[pointer] = (byte)inputStream.read();
    pointer ++;}
```

该改进方法引入了一个 int 型变量 pointer，用于指示当前接收数据在缓存数组中的存储位置。这个方法是将接收的数据逐个接收，在数据流末尾退出循环，适合用于异步通信。

将接收程序做上述改进后使其适应了异步通信数据的接收，但是示例程序的思路本身存在问题，导致其不能连续接收长串数据。下面就对此问题进行分析，并提出一个解决方案。

2　通信接收信息连续性问题

2.1　信息接收的监测

对程序开发包中的 SimpleRead. java 示例程序进行修改，使用它来进行问题的说明。程序开发环境具体信息如下所示：

操作系统：Windows 7 32－bit 旗舰版 SP1

JDK：JDK6u23

JRE：JRE7u55

Eclipse：Eclipse EE Helios Service Release 1

下位机：IC 读卡器 控制器：STC89C52 串行通信工作方式 1

　　波特率 9600 8 位数据 1 位停止

说明：上位机向读卡器发送一个命令，读卡器接收到命令后向上位机发送一串

　　数据：0xFC 0x03 0x01 0x01 0x41 0x02 0x00 0xBC

由于 Java Communication API 底层使用了 dll 文件访问硬件，因此需要将开发包中包含的动态连接库 win32comm. dll 部署至系统目录/System32 下，同时需要在程序中调用 System. loadLibrary（）方法加载 dll 文件。使用 Eclipse IDE 开发时需要将开发包中包含的 comm. jar 文件导入工程。测试程序按照文章的第 1 部分中提到的改进方案进行修改，程序在 Eclipse 中控制台的运行结果如图 3 和图 4 所示。

```
命令已发送
触发了一次串口事件，收到数据：
0xFC|0x03|0x01|0x01|0x41|0x02|0x00|0xBC|
共8个字节
```

图 1　正常情况

```
命令已发送
触发了一次串口事件，收到数据：
0xFC|0x03|0x01|0x01|0x41|0x02|
共6个字节
触发了一次串口事件，收到数据：
0x00|0xBC|
共2个字节
```

图 2　非正常情况

图 1 为正常的运行结果，下位机收到命令后会发送信息。程序检测到串口信息后会产生一个事件，通知监听器进行事件处理，监听器随后会通过输入流接收信息。正常情况下收到的信息应为 8 字节，触发一次串口信息事件就将数据全部接收完毕。可以简单的理解为一次数据接收产生一个串口事件，程序接收一次数据。

图 2 为非正常的运行结果。可见上位机发送了一次命令，但产生了两个串口事件，并且每一次事件中接收到的信息都不是 8 字节，将两次接收到的字节串前后拼接起来，恰好是一次正常的数据，可见下位机发送的数据串被程序截断了。在这里只列举了一种情况，通信数据可能在任何时刻被截断。

在后续的实验中，将下位机向上位机传送的数据长度变长，程序变得很难将数据一次性地全部接收完毕，而会产生多个串口事件，将本应该一次接收完成的数据截成几段分别接收。

Java Communications API 采用监听器模式接收串口数据，这是一种观察者设计模式，需要有事件源[10-11]。在这里，事件源是串口对象。程序在检测到串口数据后会将事件类型包装在 SerialPortEvent 对象中，交给 SerialPortEventListener 实现类中的 serialEvent（）方法处理。串口有信息传送的事件在 SerialPortEvent 对象中被封装成 DATA _ AVAILABLE 事件。程序是否开始接收数据是通过判断事件类型来决定。

异步通信模式的一个特点是，有信息就会接收，在没有标志的情况下，接收方无法判断信息何时传送完毕。接收方会默认一个时间阈值，每接收到一个字节就开始计时，等待下一个字节到来，如果等待时间超过阈值，就判定通信结束，停止接收。此接收流程如图 3 所示。

但异步通信模式的发送方与接收方之间没有同步时钟，这可能会导致每次发送的字节之间的时间间隔不相等。有可能会出现两个字节的间隔超过阈值，但信息并没有传送完毕的情况。这种情形下程序依然会判定串口事件已处理完毕，结束接收信息。余下的字节到达后会触发一个新的串口事件，之前已收到的存储在缓存中的信息会被清空，这就是图 2 所反映的情

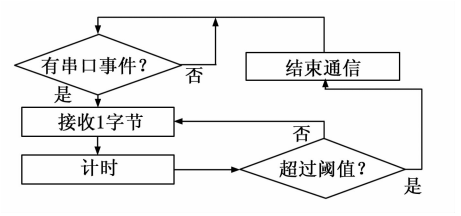

图 3　示例程序的处理流程图

况。问题的示意图如图 4 所示。

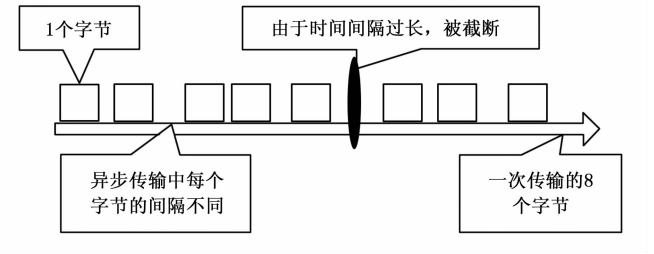

图 4　问题示意图

2.2　解决方案

在 SimpleRead.java 中，存放接收数据的字节数组是 serialEvent（）方法中的局部变量，变量名为 readBuffer。该变量会在程序每次处理串口事件时被重新分配内存单元，前一次存储的数据会被清空。显然，如果程序在接收长串数据时将数据截为几段，数组则只会存放被截断的最后一段数据。这里应将缓存数组作为串口接收程序的类变量。即使数据串被截断，则原有的数据信息也会被保留下来，同时存储空间不会因为退出方法体而被清空。同时使用一个整型类成员变量 pointer 作为接收数据缓存数组的下标，指示最后一个接收到的字节在缓存数组中的位置，也可以表示当前已接收到的字节数。pointer 必须作为类成员（或是全局变量），以保留上一次的数据接收情况。

原有的程序判断一次通信是否结束的标准是字节的间隔，而异步通信字节间隔不相同，那么使用它判定标准是不合适的。异步通信模式中双方可在传送连续数据前先约定通信数据的长度。这里的情形符合上述情况，即程序每次接收下位机数据前数据的长度是确定的，因此程序可以使用接收的字节数作为判断标准。

每次通信时根据要接收的字节数目分配缓存区容量。信息到来时 InputStream 依次将收到的字节存入缓冲数组。pointer 作为数组的下标指示当前执行存储操作的数组单元。一次事件处理完成后，程序判断 pointer 是否等于缓存数组的长度，如果相等，表示数据已经传送完毕，pointer 清 0，程序分配新缓存；如果不等，代表通信还未完成，是两个字节间隔过长，字节流被切断，这时 pointer 不清零，程序不重新分配缓存。下一个字节到来时，程序可以将接收到的数据从 pointer 指示的数组位置继续存储。这样字节串即使被截断，也可以被完整接收。改进程序的接收数据流程如图 5 所示。

按照上文中提到的思路，在原有的程序中添加了如表 1 所示的类成员变量。在程序源代码的接收部分添加如表 6 所示的判断语句。

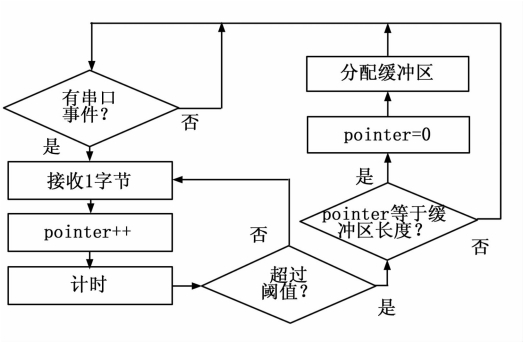

图 5　改进的接收流程

```
if(pointer == readBuffer.length) {
    //提示已完整接收数据,省略
    pointer = 0;
} else {//提示数据尚未接收完毕,省略
}
```

添加的判断语句是判断接收的数据长度是否为双方约定的长度。如果是，则数据已接收完毕；如果否，还没有接收完成，继续等待数据。

表 1　添加的类成员变量

变量名	类型	作用
readBuffer	byte[]	缓存接收到的信息，可以在较长的一段时间内保留接收到的数据。
pointer	int	指示最后接收到的数据在缓存数组中的位置，类似下标。

2.3　改进程序的运行结果与分析

根据以上思路实现的改进程序运行结果如图 6 所示，字节串在第 2 个字节的位置被截断，程序未接收满 8 字节，继续等待后 6 个字节；在接收 8 字节后，确认接收结束，返回完整信息。

```
命令已发送
触发一次串口事件，未传递完毕，已收到：
0xFC|0x03|
共2个字节，继续接收
触发一次串口事件，收到完整数据：
0xFC|0x03|0x01|0x01|0x41|0x02|0x00|0xBC|
共8个字节
```

图 6　改进后的程序运行结果

通过以上实验结果可以看出，由于读写串口的底层实现过程是无法在程序中改变的，因此无法改善“数据串在接收未完成时被截断”这一状况。但是可以根据“接收到的数据串是否等于双方约定的长度”这一标准判断接收是否完成，也就可以根据这个判断条件接收到完整数据。即使数据串被截断，通信尚未完成，程序依然可以在从截断的地方继续接收数据，最后获得的数据是完整的。这样，通信数据不连续的问题得到了解决。

3　结束语

造成信息不连续问题的不是 API，而是异步通信模式本身。异步通信模式十分简单，通信双方直接传送信息，其中不包含同步时钟；通信同步是依赖双方约定的波特率。这种同步方式本身可靠性不高，

（下转第 2649 页）

3.3　拍发质量的评估

训练器具有实时显示拍发速度和拍发质量功能，其中拍发质量与以下参数有关：

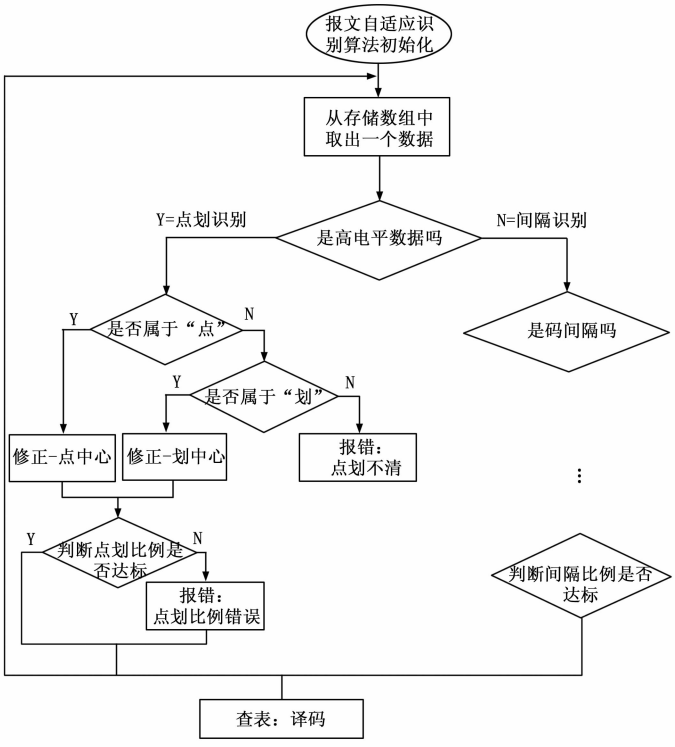

图 3　报文识别算法简化流程图

1）拍发错码数量。在滑动窗内计，点划无法识别、间隔错误的数量。

2）拍发的一致性。点、划、间隔的一致性，用各自方差的均值来衡量。

3）拍发标准性。通过计算点、划码，码间隔、字间隔、词间隔时长的相互比例关系，与标准比例 1：3，1：3：5 的偏差。

4　结语

设计的莫尔斯报训练器可单机、连网使用，识别结果、成绩评定直接在 LCD 上显示，也可上传到服务器上，使用方便灵活。算法识别速度快，错码率低，在算法跟踪速度和识别准确度间取得了良好的平衡。经报务人员大规模使用，证明在 100 码/分条件下误识别率小于 10^{-4}，效果良好。

参考文献：

[1] 杨路刚，陈　斌，王永斌，等. 基于 MSP430 的莫尔斯报训练装置设计 [J]. 现代电子技术，2009，32 (9)：8－10.

[2] 莫　莉，董万福，喻洪平. 基于 TMS320F2812 的液晶显示模块接口设计 [J]. 计算机测量与控制，2009，17 (2)：407－409.

[3] 岳喜才，郑崇勋. 基于离散 Gabor 谱的短波电报信号检测 [J]. 数据采集及信号处理，1995，11 (4)：317－320.

[4] 张汝波，何立刚，李雪耀. 强噪声背景下莫尔斯信号的自动检测与识别 [J]. 哈尔滨工程大学学报，2006，27 (1)：113－117.

[5] Yang C H，Jin L C，Chuang L Y. Fuzzy support vector machines for adaptive Morse code recognition [J]. Medical Engineering & Physics，2006，28：925－931.

(上接第 2646 页)

可能出现字节间隔不等的情况，这种情形中信息流被截断，被程序分几次接收。本文提出的方法有效地解决了这一问题。异步通信模式由于其十分简单，应用范围广泛，因此要根据实际使用情况使用符合应用实情的方案。文中提出的解决方法可以应用于通信双方约定了通信长度的情况中；对于双方不约定长度，通过特定符号结束通信的情况，接收程序则可以采取实时接收的策略，即每次只读取一个字节，不会存在信息不连续的问题。在许多场合中，收发信双方通信满足“双方在通信前已约定通信长度”这一前提，例如通过计算机控制读卡器，计算机通过串口向读卡器发送命令完成读卡动作，读卡器读卡完成后向计算机返回卡号，双方已约定好的通信数据格式。本方法可以广泛地应用于具有串口通信功能的 Java 应用程序的开发中。

参考文献：

[1] Oracle Corporation. Java Communications API [EB/OL]. [2014－2－2]. http：//www. oracle. com/technetwork/java/index－jsp－141752. html，2012.

[2] 丁振凡，王小明，吴小元，等. 客车电气绝缘智能检测系统工控机端设计 [J]. 计算机测量与控制，2012，20 (11)：2940－2942.

[3] 李　良，朱善安. 基于 Java 的串口通信 [J]. 电子器件，2007，(2)：714－720.

[4] 陆颖瑜，张永林. 利用 Java 实现对云台和镜头的远程控制 [J]. 工程设计学报，2007，(8)：324－328.

[5] 吴兴军，胡汉春. Java 实现计算机与 OMRON PLC 串口通信 [J]. 工业仪表与自动化装置，2010，(1)：84－91.

[6] 李新源，赵树法，魏宗寿. 基于 Java 语言的 GPS 接收机的串口通信程序设计 [J]. 铁路计算机应用，2007，16 (5)：4－6.

[7] Oracle Corporation. javax. comm package [EB/OL]. [2014－2－2]. http：//docs. oracle. com/cd/E17802 _ 01/products/products/javacomm/reference/api/javax/comm/package － summary. html，2004.

[8] Sun Microsystems. SimpleRead. java [CP/OL]. [2014－2－2].

[9] Oracle Corporation. InputStream (Java Platform SE 6) [EB/OL]. [2014－2－3]. http：//docs. oracle. com/javase/6/docs/api/java/io/InputStream. html.

[10] Eckel B，陈昊鹏译. Java 编程思想 [M]. 第四版. 北京：机械工业出版社，2007.

[11] Horstmann C S，Cornell G. JAVA 核心技术卷 I：基础知识 [M]. 第 8 版. 叶乃文，等译. 北京：机械工业出版社，2008.