



南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

毕业设计说明书

9161100X01

作 者： 刘海茂 学 号： 22

学 院： 自动化学院

专业(方向): 自动化

班 级： 9161102003

题 目： 基于 CAN 总线的多路数据实时采

集软件设计与实现

指导者： 李银伢 研究员

评阅者：

2020 年 6 月

声 明

我声明，本毕业设计说明书及其研究工作和所取得的成果是本人在导师的指导下独立完成的。研究过程中利用的所有资料均已在参考文献中列出，其他人员或机构对本毕业设计工作做出的贡献也已在致谢部分说明。

本毕业设计说明书不涉及任何秘密，南京理工大学有权保存其电子和纸质文档，可以借阅或网上公布其部分或全部内容，可以向有关部门或机构送交并授权保存、借阅或网上公布其部分或全部内容。

学生签名：

Handwritten signature in black ink, reading '刘海风' (Liu Haifeng).

2020 年 6 月 9 日

指导教师签名：

年 月 日

毕业设计说明书中文摘要

在日常生活和工业现场中，数据采集是获取对象信息的主要方法，它是自动化控制系统的第一步。为解决 PC 端多路 CAN 报文的采集问题，本课题基于 Visual Studio 2019（VS2019）平台，设计了该数据采集系统的软件。其具体步骤为：首先对市场上的数据采集软件进行了调查，列出了需求分析表。然后根据需求表进行软件总体设计并按模块进行详细设计，从采集模块、解析模块到保存模块，设计顺序逻辑符合常理。之后对软件进行一定的测试，检测软件的安全性和稳定性，最终成功设计出了符合要求的采集软件，并且进行了仿真实验，成功地采集、解析了小车运动时四轮速度的 CAN 报文信息，经对比采集的信息和现场的信息完全一致。该软件对实验室用实时、多通道的采集系统有着一定的参考和借鉴价值。

关键词 C# Visual Studio 2019 数据采集系统 控制器局域网

毕业设计说明书外文摘要

Title Design and Implementation of Real-time Multi-channel
 Data Collection Software Based on CAN Bus

Abstract

In daily life and industrial scenes, data collection is the main method for obtaining object information, and it is the first step in an automated control system. In order to solve the problem of collecting multiple CAN messages on the PC side, this paper designs the software of the data collection system based on the Visual Studio 2019 (VS2019) platform. The specific steps are as follows: First, the data acquisition software on the market is investigated, and a demand analysis table is listed. Then according to the requirement table, the overall software design and detailed design according to the module, from the acquisition module, the analysis module to the preservation module, the design sequence logic is in accordance with common sense. After that, the software is tested to check the security and stability of the software. Finally, the acquisition software that meets the requirements was successfully designed. And carried out simulation experiments, successfully collected and analyzed the CAN message information of the four-wheel speed when the car is moving, and the information collected by comparison is completely consistent with the on-site information. The software has certain reference and reference value for the laboratory real-time, multi-channel acquisition system.

Keywords C# Visual Studio 2019 Data Collection System CAN

目 次

1 绪论.....	1
1.1 研究背景与意义.....	1
1.2 国内外相关研究现状.....	1
1.3 主要研究内容及章节安排.....	3
2 软件需求分析.....	5
2.1 前台功能分析.....	5
2.2 后台功能分析.....	8
2.3 可行性分析.....	9
2.4 本章小结.....	10
3 软件概要设计.....	11
3.1 软件架构设计.....	11
3.2 软件模块设计.....	12
3.3 软件数据流设计.....	15
3.4 本章小结.....	17
4 软件详细设计.....	18
4.1 用户界面设计.....	18
4.2 采集模块设计.....	20
4.3 解析模块设计.....	22
4.4 保存模块设计.....	25
4.5 其他模块设计.....	26
4.6 软件容错性设计.....	27
4.7 本章小结.....	28
5 软件功能测试与验证分析.....	29
5.1 单元测试.....	29
5.2 集成测试.....	32
5.3 数据采集案例.....	34
5.4 遇到问题及解决思路.....	37
结论.....	39
致谢.....	40
参考文献.....	41
附录 A 软件实际使用图.....	42

1 绪论

1.1 研究背景与意义

数据采集是对一个或多个信号获取对象信息的过程。数据采集器是一种具有实验室或现场进行实时数据采集、自动存储记录、信号预处理、即时显示、即时状态分析、自动传输等功能的自动化设备^[1]。正因如此,几乎各行各业、不同的领域都离不开数据采集技术。由于数据采集系统的性能优良,超过了传统的自动检测仪表和专用数据采集系统^[1],因此该技术获得了惊人的发展,且现在依旧保持一个热门的势头。数据采集系统起始于 20 世纪 50 年代的美国军方,大约在 60 年代后期,就有成套的数据采集设备产品进入国外市场,70 年代中后期,随着微型机的发展,诞生了采集器、仪表同计算机溶为一体的数据采集系统。80 年代后期,出现了工业计算机、单片机和大规模集成电路的组合,这使得数据采集系统发生了极大的变化,用软件管理,系统的成本能大大的降低,体积能变得很小,功能也可以成倍增加,数据处理能力大大加强。90 年代至今,在国际上技术先进的国家,数据采集技术已经在军事、航空电子设备及宇航技术、工业等领域被广泛应用^[1]。

CAN 总线是一种优秀的现场总线,是由德国 Bosch 公司在上世纪 80 年代为解决汽车内部众多的控制器和电控执行单元之间的数据通信而开发的一种多主机局域网络^[2],它具有可连接设备数目多、传输距离远、抗干扰能力强等许多优点,CAN 总线国际标准的制定,更推动了它的发展与应用,CAN 总线已经被广泛用于工业现场控制、智能大厦、环境监控等众多领域^[3]。

本课题基于 CAN 总线设计了数据采集系统的软件,主要解决在实验室或者工业上需要同时采集多个通道的信息和实时采集并解析这些信号的问题。本软件能够降低实验室采集系统的成本、提高其解析数据的效率,可以作为实验室用采集系统软件的参考。

1.2 国内外相关研究现状

1.2.1 数据采集系统研究概况

现代工业生产和科学研究对数据采集的要求越来越高,设计数据采集软件是一件十分必要且有意义的事。对于电力生产与传输这类领域而言,嵌入式高速电网络数据采集系统就是

一个相当不错的选择。根据电能质量监测系统对检测装置高速数据采集和传输的可靠性要求,提出了一种基于嵌入式 TCP/IP 协议的电网络数据采集与传输系统。创造者是在分析了该电网络数据传输系统的原理和结构的情况下,设计了该系统的硬件,完成了网络通信芯片驱动的编写、嵌入式 TCP/IP 协议栈的实现,以及数据传输系统的优化等,并由此研制出一台基于 DSP(TMS320F2812)的高速数据采集与分析系统。将该系统用于实际,结果确实验证了该系统高速实时数据采集时的可靠性,其综合特性优于现有的电网络数据采集系统^[4]。另外,电力调度方面,文献[5]为我们也提供了一种思路——通过讨论广域数据采集框架及网络构成,深入研究广域环境下的数据订阅机制、数据发布技术和数据采集代理方法,可以得到一种更高效、便捷的广域实时数据传输方式。

虽然各行各业所涉及的领域不同,数据采集系统的建模可能不一样,它们所使用的采集技术也可能不同。但是对于相当一部分的领域来说,通过嵌入式设备进行数据采集是一个通用的、有效的方法。例如,对于车辆监控数据采集系统和气象数据采集系统来说,系统建模有相似之处,都是运用了嵌入式芯片,通过设备与上位机的通讯,借由软件控制芯片驱动,从而间接地采集 CAN 节点上各个传感器的数据包,而且两者是基于 USB-CAN 总线的嵌入式数据采集系统。另外,对于电力调度领域而言,因为采集系统有着地理分布范围广、高实时性和可靠性的特点,又采用了广域数据订阅机制、广域数据发布技术、广域数据采集代理等关键技术,系统的研究方向不在于嵌入式设备,而更多的是面向信息交互方面,所以该系统基于的是一种算法,该方法并不和嵌入式采集的模式冲突。正因这种嵌入式设备硬件采集——上位机软件分析的数据采集模式应用十分广泛、十分有效,人们也基于它,创造出了许多优秀的技术方法。

1.2.2 主要技术方法

下面简要列举几种数据采集器所使用的各种技术方法。

- (1) 基于 CAN 总线。首先将系统的各个功能尽可能地分散到各个节点,各节点以微处理器为核心,从现场采集数据。这些采集到的信息可能是模拟量或数字量,模拟量通过模数转换器转换成数字信息后,发送到 CAN 总线上,总线上的其它节点单元和操作站根据自己的需要和事先设计好的验收码和验收屏蔽码,来判断是否接收该信息^[7]。
- (2) 基于 PCIe 总线。通过 PCIe 接口将数据上传至上位机,使系统能够稳定、长时间采集数据,能够实现最大传输速率为 2GB/s、最大数据存储量为 60GB 的数据传输与存储功能。
- (3) GPRS 技术。GPRS 是通用分组无线业务的简称。GPRS 是 GSMPhase2.1 规范实现的内容

之一,能提供比现有 GSM 网 9.6kbit/s 更高的数据率。GPRS 采用与 GSM 相同的频段、频带宽度、突发结构、无线调制标准、跳频规则以及相同的 TDMA 帧结构。因此,在 GSM 系统的基础上构建 GPRS 系统时,GSM 系统中的绝大部分部件都不需要作硬件改动,只需作软件升级。GPRS 移动数据传输系统的应用范围广泛,几乎所有中低速率的数据传输业务都能应用。

1.3 主要研究内容及章节安排

本文主要研究的内容是设计一个基于 CAN 总线的多路、实时数据采集分析软件,它可以实现多通道的数据采集和 CAN 报文分析等功能。而且该软件需配套市面上的广成科技或立功科技的 CAN 录入采集硬件设备,且在 Windows 环境下进行使用。

为设计该软件,本人先在市场上进行了对软件功能需求分析的调查,然后参考学习了行业标准和当前市场上的商业软件,从而设计了软件的用户界面。然后复习了 C#编程语言,学习面向对象编程的思路,完善了软件的逻辑方法和各部分的功能函数。然后在和老师交流讨论,对 UI 进行再次的反复修改,函数逻辑部分进行反复的修正。最后使用一个例子进行仿真实验,测试软件的可行性,从而实现了实时、多路的数据采集分析软件。

首先,本文第二章便详细地展示了该软件的需求分析表,从用户界面设计到内在函数逻辑,从菜单栏到状态栏,从一级菜单到二级菜单……层层细化,将软件设计这一大目标分解为了多个有关联且容易实现的小目标。其主要内容包括功能分析、可行性分析这两个部分。

然后,本文的第三章介绍了该软件的整体设计,其主要内容包括概要设计、详细设计、数据流设计和本章总结。概要设计的系统结构图和软件使用流程图表明了该软件的属性和用法;而详细设计部分则展示了软件各个模块的流程图,各个模块的原理一目了然;数据流设计部分则表明了软件中数据的属性与内在逻辑,为数据采集打好基础。

接下来,本文的第四章展示了该软件的详细设计思路,即各个模块的设计思路。它的内容主要包括用户界面设计、采集模块设计、解析模块设计、保存模块设计、其他模块设计以及本章总结。用户界面严格按照第二章需求分析来设计,做到了尽可能的人性化;采集模块则搭配硬件设备,充分发挥了采集的实时性和多路性;解析模块的设计则考虑了实时性和非实时性,有这两种模式可选,其解析结果既可以用报文显示又可以用图形直观表达;保存模块则能按用户需求详尽的保存采集到的数据且不丢失;而其他模块则通过调用外部动态链接库和创建新线程,完美地让采集线程与用户操作的线程错开,不会发生冲突。

此时软件已设计完毕,所以第五章则说明了本次软件的测试过程以及测试结果。按照软

件测试标准，本章内容包括：单元测试、集成测试、验收测试，这三个部分。单元测试可以单独检测软件的某个部分，设计该软件的思路让不同部分间的代码耦合程度较低，报错时不会相互影响，这使得出现 bug 时可以找到具体的这一部分代码；集成测试在单元测试的基础上，测试整个软件系统的协调性和流畅度，可以检测出到数据传递是否有问题，第三章数据流设计是否出错；验收测试则做了仿真实验，模拟小车运动时四轮各自的速度变化情况。本次试验成功地接收了小车的一系列速度报文，并且在进行解析、绘制后，其解析结果和结果图与小车车速的误差在可接受范围内，验收成功。

2 软件需求分析

2.1 前台功能分析

2.1.1 菜单栏

菜单栏位于软件的最上方，是用户可操控的区域，如图 2.1。它有着展示软件所有功能的作用，可以开启和关闭软件的各个功能。为了符合软件使用的正常逻辑，菜单栏从左到右的顺序设计为三个下拉按钮：录入端口设置、保存方式和保存类型，然后是五个按钮：保存、另存为、数据分析、帧统计、帮助。

其中录入端口设置下拉框中一共有三个下拉框选项，分别是：设备类型、设备索引、通道。设备类型下拉框可以选择即将打开的与电脑连接的录入硬件设备类型：如 USB-CAN2 C、USB-CAN pro 等；设备索引可以选择与电脑连接的硬件的序号，仅连接一个设备是序号为 0，连接两个时第二个序号为 1；通道则是即将打开的硬件设备的通道序号，一个硬件设备可以有两个及以上的通道，该软件可以选择性的打开某一个或者几个通道，也可以默认选择全部的通道。

保存方式和保存类型这两个按钮可以选择存储录入数据的方式。保存方式可以是自动保存或者手动保存。当该选项为自动保存时，录入设备每接收一次报文，便保存一次录入区全部的数据；而用户选择手动保存时，录入设备仅在用户点击保存或者另存为按钮时才会保存录入区的全部数据。

数据分析按钮点击后可以打开数据解析窗口。该窗口可以对软件接收到的数据进行数据分析、画图等处理；帧统计按钮点击后可以在信息显示区显示各个通道当前已经接收到的报文帧数；帮助按钮点击后会打开帮助文本文档，该文档的内容是软件的说明书和注意事项。

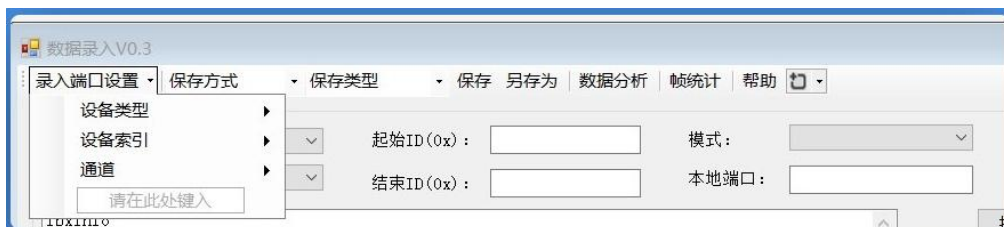


图 2.1 菜单栏

2.1.2 参数配置区和信息显示区

参数配置区位于菜单栏的下方，如图 2.2，在用户选择完录入端口设置后，用户需操作该区域继续进行软件使用配置。

在参数配置区的左半部分，用户在该区域可以进行选择硬件的工作模式、波特率、滤波范围等操作。工作模式选项可以是正常模式（软件可以收发报文）和只听模式（软件只能接收报文）；波特率下拉框可以选择常见的波特率，如 1000kbps、800kbps、666kbps……80kbps 和 50kbps；起始 ID（0x）和结束 ID（0x）是软件的滤波范围，即软件可以选择性地接收帧 ID 在滤波范围内的报文，它们可用十六进制表示。例如，传感器是小车的四个轮胎时，采集到报文的帧 ID 通常在 150800000-150892286 之间，选择该滤波范围可以让软件不接收其他的报文信息。由于该软件默认接收所有报文，所以起始 ID（0x）和结束 ID（0x）的值分别是 00000000 和 FFFFFFFF。

参数配置区的右半部分通常不进行操作，仅当录入设备是 USB-TCP 或 USB-UDP 时才进行配置。该区域一共四个配置选项，分别是：模式、本地端口、远程地址、远程端口。由于本次录入设备硬件并非这两者，故软件设计仅保留其区域供下次开发。

参数配置区的左下角紧接着信息显示区。该区域主要用于显示软件和硬件的状态以及显示统计到的帧数量。

图 2.2 参数配置区和信息显示区

2.1.3 控制按钮区

参数配置区的右下角和报文显示区的右上角是控制按钮区。用户可使用该区域将配置好的参数输入硬件设备中，开启软件的正常功能。该区域一共 5 个软件正常使用按钮和 4 个软件测试用按钮。打开设备按钮可以将用户在上个区域设置好的参数输入到硬件设备中，打开对应设备的各个通道；初始化 CAN 按钮可以在设备打开后初始化软件和硬件的参数，或者在设备异常时让它正常化；启动录入按钮可以开启采集功能且能让显示区的显示列表控件显示采集到的报文；复位按钮可以让设备长时间未采集（挂机）时恢复正常工作；清空录入区可以删除显示列表控件显示的报文信息。

软件测试时，录入测试按钮点击后可以用设备的一条通道向另一条通道发送数据，模拟数据采集的过程；解析测试按钮则是直接向某一条通道写入一条符合协议的报文，以供数据解析模块测试；仿真实验按钮点击后则是用一条通道向另一条通道发送一系列符合某一协议的报文。



图 2.3 控制按钮区

2.1.4 报文显示区

软件接收到报文后，该区域按照一定顺序显示接收到的信息，如图 2.4。该区域从左至右依次为：序号、时间、方向、帧格式、帧类型、DLC（数据长度）、帧 ID、接收的通道、数据。

[illegible]

图 2.4 报文显示区

- (1) 序号从设备接收到的第一条报文开始算起，第一条报文序号为 0。
- (2) 时间有两种显示方式。当接收到的报文的时间标志为 0 时，时间显示为电脑系统自带的时间：“年.月.日.时.分.秒：毫秒”。否则时间显示为硬件设备自带的时间：第一条报文时间为 0，之后的时间为接收的报文与第一条报文的时间差：“时.分.秒：毫秒”。
- (3) 方向有两种选项，接收和发送。由于本机是采集设备，故一般为接收。仅当控制中

心（另一台电脑）向本机（该软件）发送远程帧时，本机需要向控制中心发送对应的数据帧，该选项仅保留供下次开发。

(4) 帧格式分为标准帧和扩展帧。标准帧信息为 11 个字节，前 3 个字节为信息部分。扩展帧信息为 13 个字节，前 5 个字节为信息部分。

(5) CAN 总线的报文类型一共 4 种，本设备采集数据仅接收其中两种，数据帧和远程帧。

(6) DLC（数据长度）显示的是接收报文数据的字节数。

(7) 帧 ID 显示接收到报文的 ID。

(8) 通道显示的是接收到该报文的通道，如第一条通道接收到该条报文，则显示 CAN1。

(9) 数据以 16 进制显示接收到的数据，每一个字节都用空格分隔开。

2.1.5 状态栏

软件右下角或者最下方为状态栏。该栏从上至下依次显示了硬件设备连接状态、数据采集状态、系统缓存区状态、硬件错误码、采集错误码。设备连接未连接时字体背景色为红色，连接后文本显示为设备已连接且背景变为绿色；数据采集状态栏在软件未采集时为红色的数据未接收，采集时为绿色的数据接受中；系统缓存区在软件读取硬件设备状态时更新数据，有绿色的缓存区未满和红色的缓存区已满两种显示文本；硬件错误码和采集错误码在开启设备或采集数据时更新，出现错误则会显示对应的错误码。



图 2.5 状态栏

2.2 后台功能分析

2.2.1 调用函数（接口）库

CAN 采集设备自带了操控该硬件的函数（接口）库，利用这些库函数可以简单高效地打开、关闭硬件设备或者读取设备的状况；同理，对于协议分析模块，立功电子的官网有相应

的库函数和 demo，利用这些库函数可以大幅度降低代码的复杂度，提升了代码可读性。

2.2.2 多线程操作

软件正常使用时，既要实时采集数据、显示数据又要解析数据、绘制解析图，所以软件需要多线程操作，其中采集数据由于其实时性需要单独的线程来操作，用户的操作则放置于另一线程。

2.2.1 各类定时器

软件正常使用时有两个定时器起作用。接收报文定时器用于接收报文、读错误类型码和显示报文；而状态定时器用于更新软件采集界面的状态栏。

而进行软件测试时，新增了两个定时器，仿真定时器和录入定时器。录入定时器按照一定的周期用一个通道向另一个通道发送有效报文；仿真定时器则按照一定的周期发送报文。

2.3 可行性分析

2.3.1 技术可行性分析

对于本次的软件设计，笔者打算使用 C#语言来进行用户界面和软件逻辑的编写，因为在 VS2019 的编译环境下，C#语言可以快速高效地生成窗体界面。在经历了几个月 C#语言的学习后，下一步需要继续学习有关 CAN 总线的知识，来读懂库函数中的报文定义，从而正确地使用硬件函数库接收报文。最后学习有关 DBC 文件的知识，在接收到的报文后，正确地解析它们。

2.3.2 经济可行性分析

购买硬件时，USB-CAN2 便自带了函数（接口）库，使用官方的底层代码收发报文不用额外的花钱；至于解析报文、多线程、调用 DLL 方面，我参考了 CSDN 论坛的开源代码，能够节约一大部分成本。

2.3.3 操作可行性分析

界面设计要符合使用者习惯，要尽可能的提高用户使用舒适性，软件说明书要尽可能的

简单，代码需要开源方便检查和分析，而软件也要易使用、不能太死板（指的是一旦不严格按照说明书使用，软件就会出 bug）。数据录入必须快速、准确并且真实可靠；分析功能的统计要准确，显示区的表格要足够的灵活，容易扩充（例如，用户往下拖滚动条即可看到为显示的内容）。

2.4 本章小结

本章节介绍了软件的需求分析。首先从前台功能分析按逻辑顺序叙述，即从上至下、从左至右的顺序分析了菜单栏、参数配置区和信息显示区、控制按钮区、报文显示区、状态栏等设计所需要的按钮等控件；然后该章节介绍了软件的后台功能分析，指明了支撑前台功能的各种后台程序；最后进行可行性分析，从技术可行性、经济可行性和操作可行性三个方面解释了本次软件设计的可实现性。

3 软件概要设计

3.1 软件架构设计

3.1.1 系统结构图

按照软件设计标准，本次软件设计的系统结构图一共分为五层。其中底层硬件层对于硬件开关问题，调用了官方的函数库来实现；数据层对于窗体间数据的传递，则采用了全局静态变量和公用类；支撑层对于多路、同时操作，是使用了后台多线程和多种定时器；应用层使用了大量 C# 的窗体控件，以此简化代码数量；展现层则直观的让用户看到软件的设计界面。

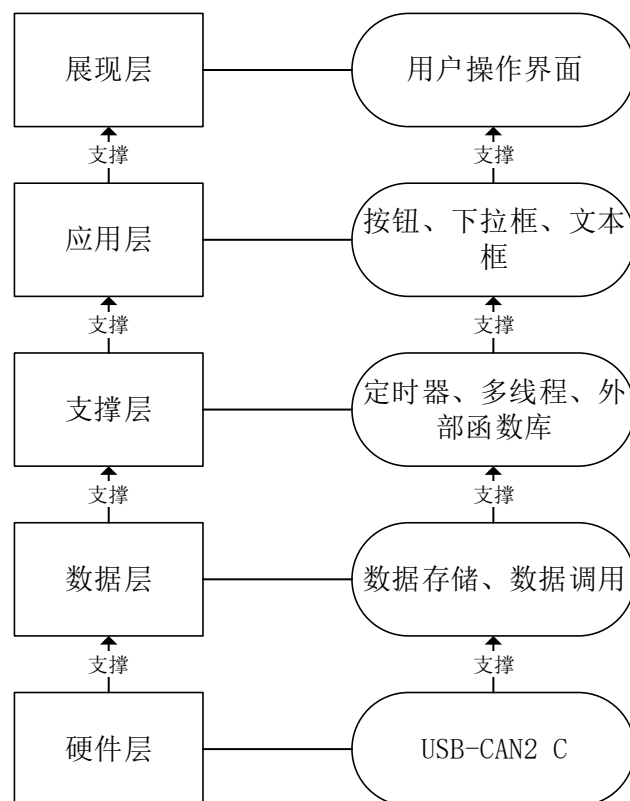


图 3.1 系统结构图

3.1.2 软件流程图

使用该软件时，首先配置好参数并打开设备，然后信息区会显示设备打开是否正常。若不正常则会显示打开错误，需要重新打开；正常则继续初始化设备的各个通道。通道初始化不正常则重新打开设备；正常则点击启动录入按钮，启动采集线程。采集到数据后，根据协

议判断数据是否有效，若有效则显示或保存数据；此时也可以打开解析模块，解析数据并绘制数据解析图。

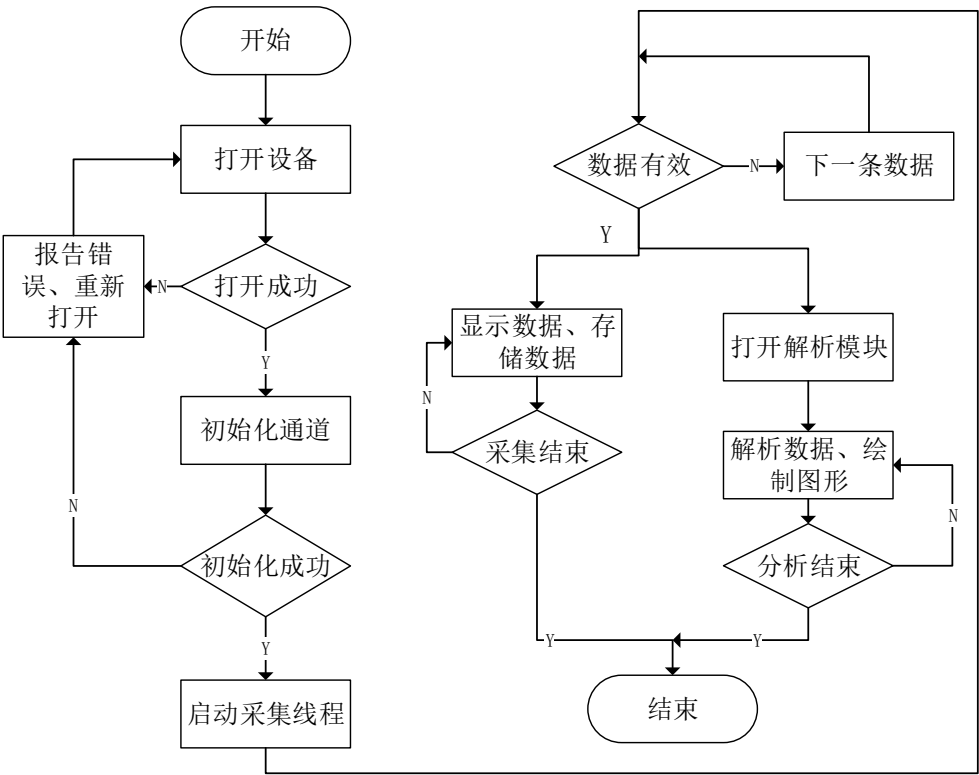


图 3.2 软件流程图

3.2 软件模块设计

3.2.1 采集模块流程图

点击启动录入按钮后，系统前台启动接收定时器，按照固定周期刷新显示区域，实时显示报文；与此同时状态定时器也启动，实时刷新软件的状态栏，显示设备状态、缓存区、错误码等信息。而系统后台启动录入线程，用另一个线程按通道接收数据，程序与用户界面不冲突。

该模块具体工作原理是修改 C#控件的属性，让按钮的名称和背景色改变；打开、关闭定时器控件，让系统周期性地刷新界面；创建或者退出线程，控制后台系统采集数据；修改系统标志位的值决定软件其他功能的开启与关闭；为文本框控件赋值，使得数据能够正确显示。

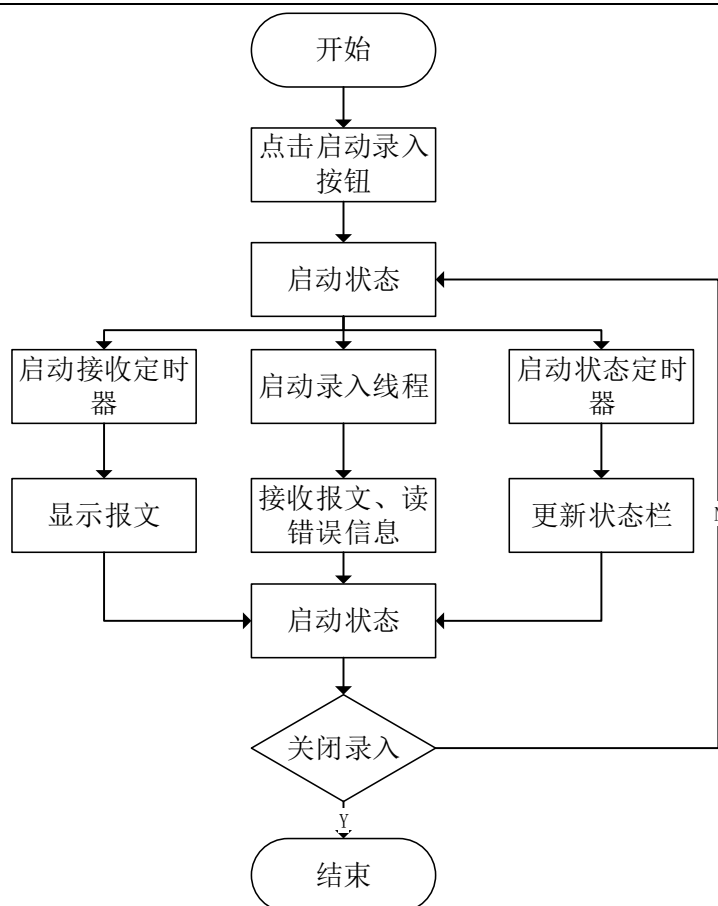


图 3.3 采集模块流程图

3.2.2 解析模块流程图

对于解析界面，软件首先加载 DBC 文件并将协议内容显示在协议区，然后将主界面的报文与协议内容进行比对解析，将解析的结果显示到结果显示区。解析完报文后，可以由用户选择保存解析结果或者绘制解析图。

该模块原理如下：

- (1) 加载 DBC 文件的意思就是用数据流的方式读取 DBC 文件数据，然后将数据信息按协议规定分类存储显示。
- (2) 解析报文就是将报文的帧 ID 和协议中的 ID 比对，找到相同的那一个。然后将数据的内容按照这一个协议的内容进行计算处理。
- (3) 绘制解析图就是将解析后的真实值作为因变量，序号或者接收时间作为自变量画出二维函数图像。

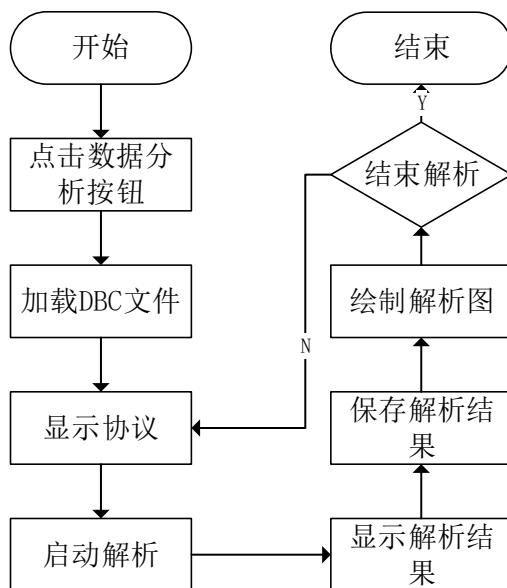


图 3.4 解析模块流程图

3.2.3 保存模块流程图

用户调用保存模块时，软件会根据文件类型来选择将数据保存为 Excel 文档还是文本文档；而保存模式则判断是自动保存还是手动保存，手动保存可以有保存和另存为两种获得保存地址的方式，自动保存获取的地址默认和手动点击保存相同。

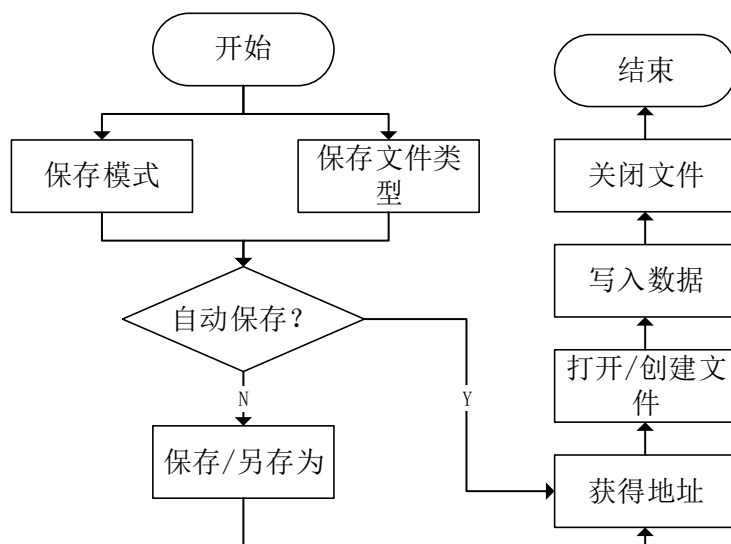


图 3.5 保存模块流程图

保存模块的原理如下：根据获得的地址，软件会打开或者创建相应的目标文件，然后将数据区的数据以数据流的方式写入文件，最后关闭该文件。

3.3 软件数据流设计

3.3.1 数据存储设计

对于采集到的报文信息，该软件用公用（public）结构体保存。对应于报文信息，该结构体成员包括帧 ID、时间标志、接收时间、发送类型（仅在发送时使用）、帧类型、帧格式、数据长度、数据，硬件保留位，如图 3.6 所示。

```
public struct CAN_OBJ
{
    public uint ID;
    public uint TimeStamp;
    public byte TimeFlag;
    public byte SendType;
    public byte RemoteFlag; //远程
    public byte ExternFlag; //扩展
    public byte DataLen;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
    public byte[] data;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
    public byte[] Reserved;
}
```

图 3.6 采集数据结构体

对于硬件信息和系统信息，软件定义了一系列的变量来存储，如表 3.1 所示。这些变量存储了硬件的设备类型号、软件当前的工作状态等信息。

表 3.1 系统信息表

属性	名称	意义
public int	xuhao	判断当前是否接受数据的标志
public static int	JieShouXuHao	当前接收报文的序号
public static uint	m_devind	设备索引号
public static uint	m_canind	设备通道号
public static uint	channelMAX	设备的最大通道数，可自定义，最好和硬件一致
public static uint	m_devtype	本次实验的设备类型号
private int	CANIn	通道 1 接收帧的数量

3.3.2 数据传递逻辑设计

启动软件后，采集到的报文会从采集界面传递到解析界面再传递到绘图界面。为解决数据何时传递的问题，定义了传递标志和传递载体。如表 3.2 所示，在用户操作界面（主界面）和解析操作界面（子界面）定义了四个标志，其中四个标志的属性都是公用静态布尔型。

表 3.2 传递标志

属性	成员	意义
public static bool	SSflag	选择实时解析模式的标志
	JXingflag	软件正在解析的标志
	DATA SendOKflag	采集报文全部传递完毕的标志
	JXOKflag	软件解析完毕的标志

当表 3.2 中某些成员的值变为真（true）时，软件会直接控制两个界面间的数据传递。例如，表 3.3 中的某些成员会存储当前界面的对应数据然后可供另一个界面读取。通常来说，SSflag 标志控制软件传递数据的模式，若 SSflag 的值为假，则主界面只能将传递用户选中的某条报文数据传递给解析界面。反之，软件为实时解析模式——主界面会从第一条报文开始，将每一条报文的数据逐一传递给解析界面。而 JXingflag 的值为真时，主界面会停留在当前的报文，等待解析界面解析完本条报文的数据后，再选择下一条数据。之后当 DATA SendOKflag 的值为真时，这标志着主界面所有数据都传递给解析界面了，软件将会停止传递数据。最后 JXOKflag 的值变为真，所有传递的数据都已经解析完毕。

3.3.3 数据传递实体设计

本次软件设计用属性为静态、公共属性（public static）的字符串存储当前选中的报文数据，以此实现在三个窗体间自由传递。每一条 CAN 报文的各信息，如帧 ID、数据、帧长度、接收到的时间等都分别存储在一个字符串中，如表 3.3 所示。

表 3.3 数据传递

属性	成员	意义
public static string	ZID	将选中行的帧 ID 传递给解析窗体
	Zdata	将选中行的帧数据 传递给解析窗体
	ZDLC	帧长度
	ZTime	接收时间

而另一部分数据，如 3.3.1 中的采集数据结构体等，存放在一个新建类（GCKJ）中，三个窗体分别使用该类，便可调用其中的数据，实现了数据传递。

3.4 本章小结

本章节主要介绍了软件的概要设计。首先介绍了软件的系统结构图和流程图，从整体角度概写了软件总体的使用流程和其属性定义。然后简单介绍了软件各个模块的流程图和工作原理，概述了不同功能的工作过程，为下一章详细介绍各个模块做了铺垫。最后介绍了系统的数据部分，从数据存储和数据传递这两个方面解释了报文采集后的保存和调用。

4 软件详细设计

4.1 用户界面设计

4.1.1 采集界面设计

该界面由七个部分组成，它们分别是：1、菜单区，2、参数配置区，3、信息显示区，4、控制按钮区，5、报文显示区，6、状态栏和 7、后台控件区。用户在实际使用时只能看见前 6 个前台区域。而每个区域所含的控件在第二章的需求分析处已详细描述，本节不赘述。

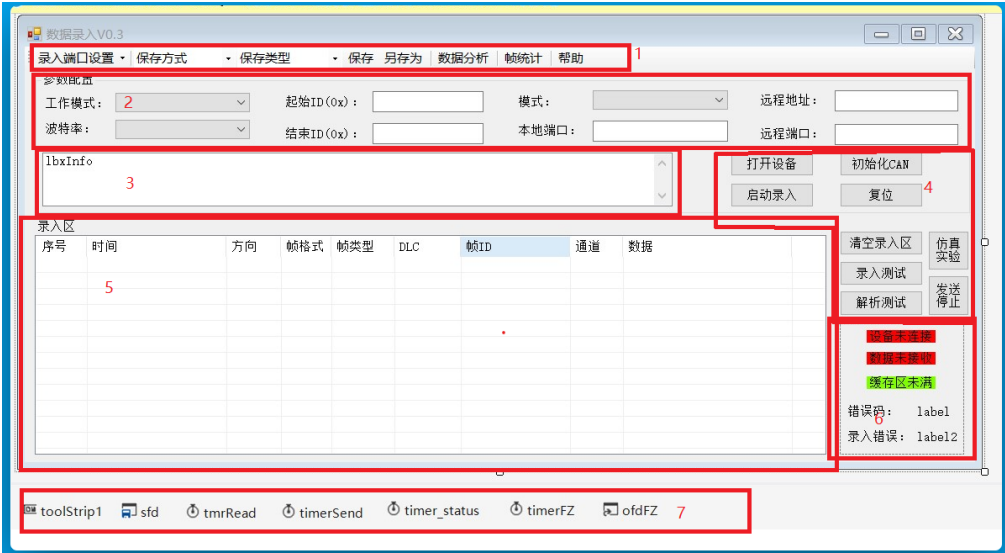


图 4.1 采集界面

该界面的具体使用操作流程为首先鼠标点击菜单栏的录入端口设置，等待软件跳出二级菜单栏；然后根据硬件设备选择相应的设备类型、设备索引（电脑仅连接了一个设备，那么默认为 0）、通道数（不选择默认为 all），此时还需要在工作配置区确定必须的波特率、工作模式、滤波范围等参数；下一步点击按钮控制区的打开设备按钮，信息显示区会显示“打开成功”；然后点击初始化 CAN 按钮，信息显示区会显示“打开 CAN1 成功”（通道数更多，则还会显示打开 CAN2、CAN3 成功）；之后点击启动录入按钮，后台控件区的接收定时器会启动，录入区的显示列表会显示采集到的 CAN 报文，与此同时状态栏的各个文本会实时更新。另外若硬件长时间未采集数据，处于挂机状态，点击复位按钮即可让采集功能恢复正常。

还有用户可以通过点击菜单栏的保存系列按钮来实现保存功能。当用户选择保存方式为“自动保存”时，点击启动录入按钮后，软件每接收到一组报文就会在根目录保存一次数据；而用户选择保存类型为“Excel”时，点击保存按钮可以将报文保存为 Excel 文件；用户点击

另存为按钮时，软件会弹出对话框，用户可自行选择路径、文件类型并进行数据保存。

至于数据分析、帧统计和帮助这三个按钮，用户点击时分别会发生弹出解析界面、信息显示区的显示列表会显示各个通道采集的报文数量、打开软件的使用说明和注意事项文档。

4.1.2 解析界面设计

该界面由五个部分组成，它们分别是：1、菜单区，2、协议显示区，3、解析显示区，4、状态栏和 5、后台控件区。用户实际使用时无法看到或操作后台控件区。该界面的菜单区一共 7 个按钮。其中解析模式下拉框可以选择“实时解析”和“非实时解析”，该界面默认选择后者；解析图按钮可以打开图形界面；复位按钮可以清除协议显示区、解析显示区、状态栏和软件内部存储的数据；返回主界面可以关闭该窗体。

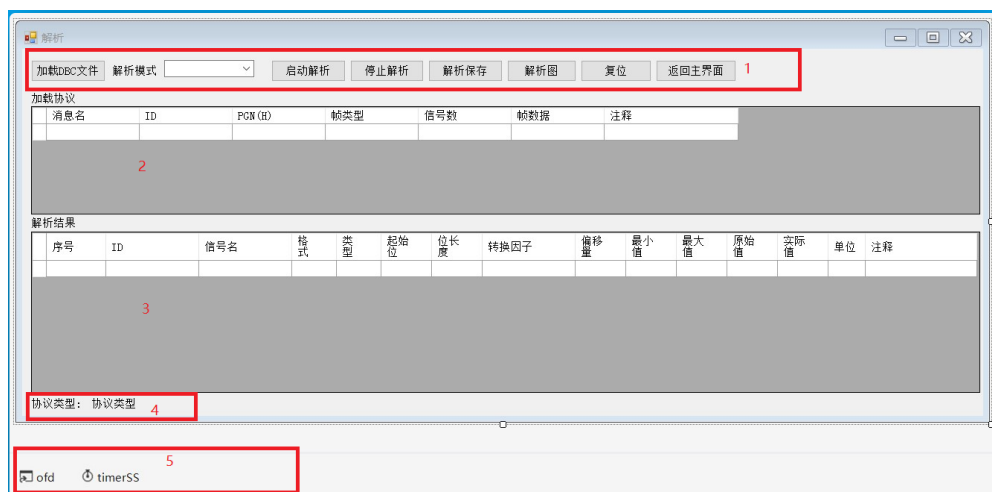


图 4.2 解析界面

该界面具体的设计思路如下：

首先，点击加载 DBC 文件按钮，选中相应的协议文件。然后当协议显示区已经成功显示协议内容时，点击区域内的任意消息，可以在结果区域查看信号的详细内容。下一步选择解析的模式，“非实时解析”模式下，需要在采集界面选中一条想要解析的报文，而“实时解析”模式，采集界面会依次选中每一条报文。然后点击启动解析按钮，解析结果区域会显示该报文的真实信息。

解析保存按钮可以将解析结果区域显示的内容保存为文本文件，而解析图按钮可以打开图形界面并调用解析结果的真实值和序号选项框中的内容。

4.1.3 图形界面设计

图形界面分为图形显示区域和含三个按钮的菜单区。绘制图像按钮可以绘制解析图像；保存按钮可以在软件根目录下保存 png 格式的图片；返回主界面按钮可以关闭该界面。

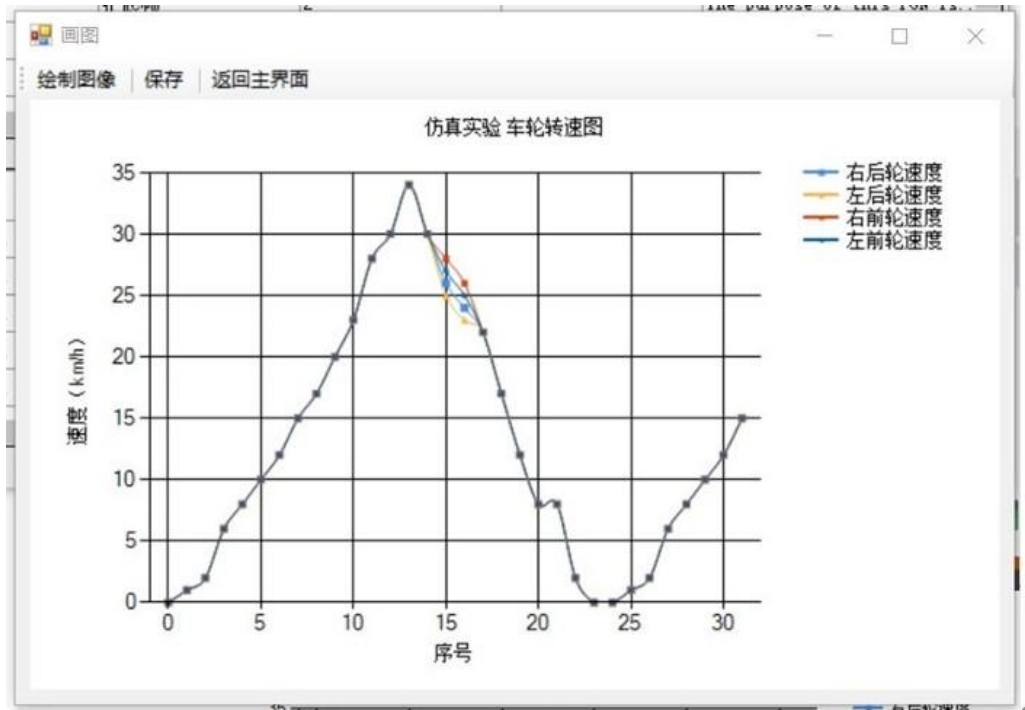


图 4.3 图形界面

4.2 采集模块设计

4.2.1 参数配置

工作模式：两种选择分别是正常或只读。只读状态该通道无法发送数据。

波特率：用户选择对应的波特率，代码需要根据厂商的配置图（图 4.4）确定两个定时器的值。

滤波范围：接收到报文时，先判断该报文的帧 ID 是否大于起始 ID，再判断是否小于结束 ID，都符合则采集该条报文数据，否则不采集该报文。

备注

Timing0和Timing1用来设置CAN波特率，以下是几种常见的波特率。如果要使用表中没有列出的波特率，请联系广成科技技术支持。

CAN波特率	定时器0	定时器1
5Kbps	0xBF	0xFF
10Kbps	0x31	0x1C
20Kbps	0x18	0x1C
40Kbps	0x87	0xFF
50Kbps	0x09	0x1C
80Kbps	0x83	0xFF
100Kbps	0x04	0x1C
125Kbps	0x03	0x1C
200Kbps	0x81	0xFA
250Kbps	0x01	0x1C
400Kbps	0x80	0xFA
500Kbps	0x00	0x1C
666Kbps	0x80	0xB6
800Kbps	0x00	0x16
1000Kbps	0x00	0x14

图 4.4 波特率设置

4.2.2 采集数据

软件先定义一个如图 3.6 的 CAN 报文结构体，然后调用广成科技的接收库函数。传感器采集到的数据便能存储到该结构体之中。其具体步骤为：

1、在公类（GCKJ）中，调用外部动态链接库（DLL），如图 4.5 所示。其中该库函数需要输入的参数依次分别是设备类型号、设备索引号、采集通道序号、接收用 CAN 报文结构体首指针、接收数据帧数组长度和等待超时时间（单位是毫秒）。

2、在主程序中用 if（ECANDLL.Receive(对应参数) == ECANStatus.STATUS_OK）语句判断是否采集成功，若成功则依次存取并显示数据即可，不成功则继续读错误码，显示错误原因。

```
[DllImport("ECANVCI.dll", EntryPoint = "Receive")]
2 个引用
public static extern ECANStatus Receive(
    UInt32 DeviceType,
    UInt32 DeviceInd,
    UInt32 CANInd,
    out CAN_OBJ Receive,
    UInt32 length,
    UInt32 WaitTime);
```

图 4.5 接收函数

4.2.3 显示报文

报文显示区域采用的是 C#语言的显示列表（listview）控件，如图 2.4 所示，该控件可用标准的 ListViewItem 类来为显示列表的每一项添加内容。

显示报文过程如下：首先创建一个新的 ListViewItem 类，然后对其赋值，其中 ListViewItem 类的文本（Text）成员相当于第一项，添加子项（SubItems.Add）相当于添加后一系列的数据。将该类的每一项都赋值之后，即完成了对显示列表一行的内容的赋值，此时将该类添加到显示列表的下一行（Items.Add）即可完成显示。

```
{
    listViewItem.Text = s.Next(100).ToString();
    listViewItem.SubItems.Add("test");
    listViewItem.SubItems.Add("接收");
    listViewItem.SubItems.Add("数据帧");
    listViewItem.SubItems.Add("扩展帧");
    listViewItem.SubItems.Add("6");
    listViewItem.SubItems.Add("217058046");
    listViewItem.SubItems.Add("CAN99");
    listViewItem.SubItems.Add("07 D0 0F A0 0B B8");
}
listView1.Items.Add(listViewItem);
```

图 4.6 显示报文

4.2.4 显示错误

错误的显示由系统状态栏（图 2.5）中的标签（Label）控件和信息显示区（图 2.2）中的列表框（ListBox）控件负责。标签的文本（Text）和背景色（BackColor）属性可以在软件接收到错误信号时直接修改，而列表框则可以通过添加项（Items.Add）的方式在下一行显示错误内容。

4.3 解析模块设计

4.3.1 加载 DBC 协议

加载 DBC 协议用到了打开文件对话框（OpenFileDialog）控件，如图 4.7 所示。该控件可由用户自行选择文件，以获得目标文件的地址。然后通过如图 4.8 所示的库函数按指定格式提取文件中的内容。这两个库函数的两个参数分别是解析模块启动标志和 DBC 文件的指针，

其具体使用方法是：首先定义用来存储 DBC 文件信息的结构体，然后用 DBC_GetFirstMessage 函数获得文件的第一条信息，之后使用 DBC_GetNextMessage 函数获得下一条信息直至最后一条信息，最后将这些信息添加进该结构体。在文件信息已全部载入之后，将内容按指定格式用数据网络视图（DataGridView）控件可显示其结果。

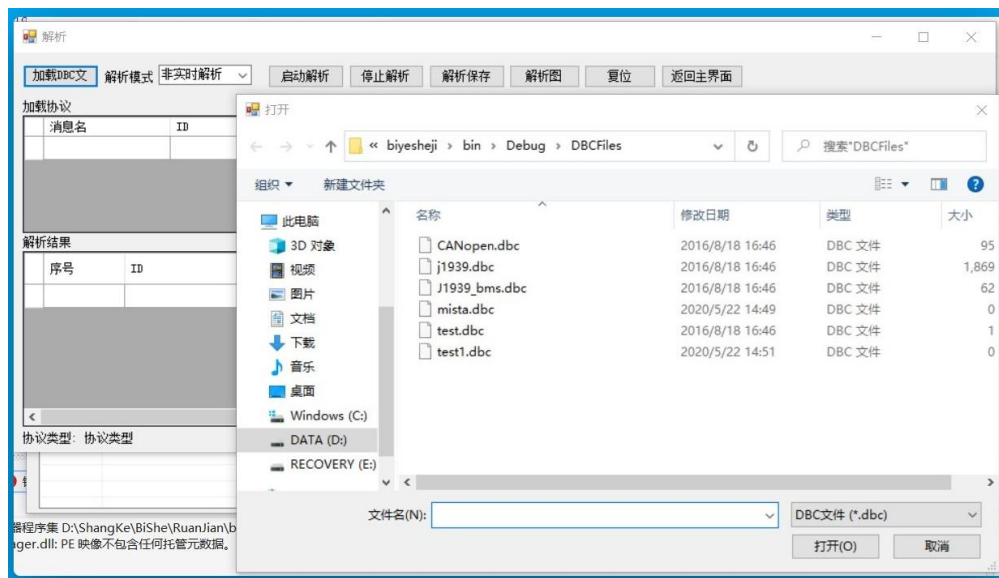


图 4.7 打开文本对话框



图 4.8 加载 DBC 库函数

4.3.2 解析模式

解析模式有两个选项：“实时解析”和“非实时解析”。前者的实现为采集界面显示区的选中行的序号（SelectedItems[0].Index）在解析完一条报文后自动加一来实现。而后者需要手动选中准备解析的那一行报文。

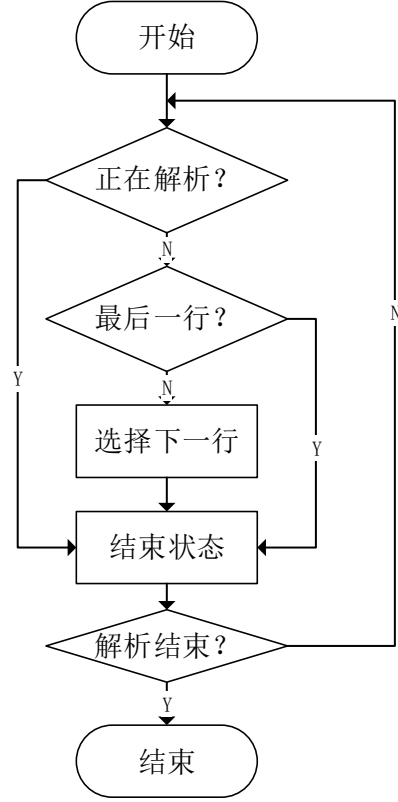


图 4.9 实时解析流程图

4.3.3 解析报文

以解析 J1939 报文为例，首先用待解析的报文帧 ID 遍历 J1939 协议中的所有帧 ID，直至找到帧 ID 相同的那一条消息；然后将报文的数据转化为二进制数，按照对应信号的起始位和信号长度提取二进制数中对应长度的数据；最后将这些数据乘上转换率再加上偏移量便得到了真实值，解析完毕。

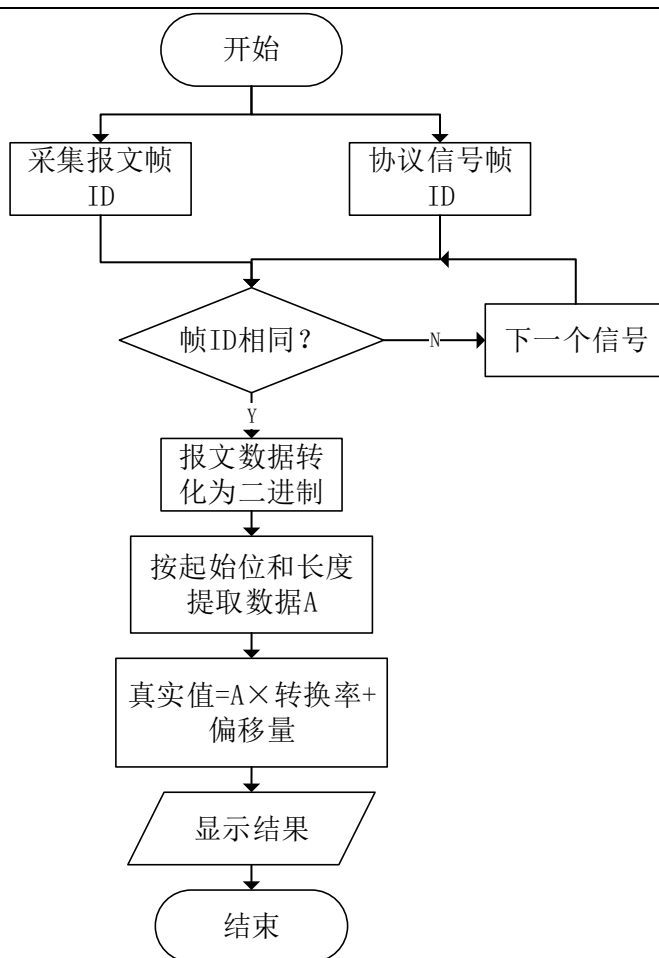


图 4.10 J1939 报文解析流程图

4.3.4 显示解析结果并绘图

解析结果显示采用的是数据网络视图（DataGridView）控件，直接向该控件每一格赋值便能显示。如 `dataGridView2.Rows[0].Cells[0].Value=0`，就能使数据网络视图控件的第一个单元格显示数字 0。

而绘制结果图时，以 J1939 协议为例，绘制解析结果图时只要将解析结果区的真实值那一列作为因变量，序号那一列作为自变量便能绘制出真实值随录入序号变化的结果图。该图可以反应传感器采集到数据的变化情况。

4.4 保存模块设计

4.4.1 保存方式及类型

保存方式分为自动保存和手动保存两类。手动保存是仅在用户点击保存或者另存为时触发保存的流程。自动保存则是采集到一条新报文时触发保存的后续内容。

保存类型分为 Excel 和 txt 两类。不同的类型仅在创建文件和写入数据这两步有所不同。

4.4.2 保存窗体（另存为）

普通的手动保存和自动保存直接获得了创建或写入文件的地址和文件类型，而另存为可以由用户利用保存文本框控件自行选择地址。

4.4.3 创建 txt 文件

在软件获得了保存地址(localFilePath)后 StreamWriter 类的 StreamWriter(localFilePath,false)重载，可以实现目标地址存在文件名相同的文件时，仅修改该文件写入数据而不是重新创建新文件的方法。为保存 txt 文件，先将数据列表中的内容保存在字符串中，而 StreamWriter 类的 Write（字符串）方法可以实现将字符串的内容写入文档中。

4.4.4 创建 Excel 文件

为创建 Excel 文档，软件下载并依赖了微软公司的官方托管 DLL 代码。故软件使用了它的 Microsoft.Office.Interop.Excel.Application 名称空间。其中向 worksheet 类的 Cells 填充采集界面显示列表（listview）控件的内容即可，然后用该名称空间的 workbook 类保存上文 Cells 中的数据即可创建完文档。

4.5 其他模块设计

4.5.1 调用函数（接口）库

保存 Excel 文档用到了微软公司的托管动态链接库（DLL），而接收模块和解析模块则用到了立功电子和广成科技的非托管动态链接库。对于前者的调用，直接点击 VS2019 界面的右上角，解决方案-引用-管理 NuGet 程序包（鼠标右键），然后搜索下载该 DLL 即可。而至于后者，则需要将 DLL 文件放在主程序的根目录（bin/Debug）下，然后在程序内用 DllImport 方法调用该文件并声明其中的库函数，调用数据采集的库函数如图 4.5 所示。

4.5.2 多线程操作

在用户操作界面（主函数）中，创建一个 AutoResetEvent（false）的实例，参数 false 表

示初始状态为非终止。然后直到用户点击“启动录入”按钮，开启接收定时器并创建并启动一个子线程，在子线程中，通过调用 `AutoResetEvent` 的 `WaitOne` 方法，使子线程等待指定事件（采集数据）的发生。最后主线程的定时器每隔一定时间调用 `AutoResetEvent` 的 `Set` 方法，使状态由非终止变为终止，重新激活子线程。

这样做使得接收线程不会一直占用电脑的 CPU，提高了软件处理的效率，防止了多个模块工作时占太多内存卡死电脑的现象。

4.6 软件容错性设计

当硬件设备和录入端口设置不匹配时或者未连接硬件设备时，点击打开设备，会弹出错误窗口提醒用户。

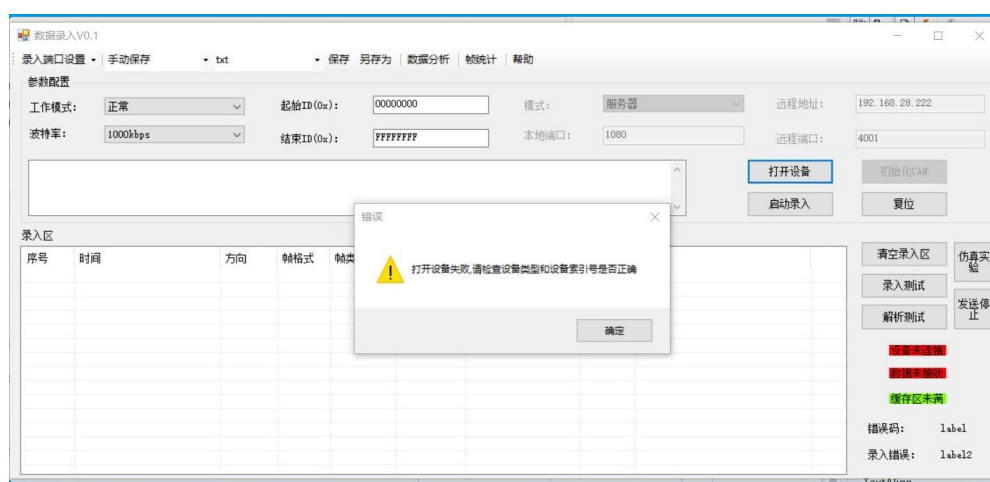


图 4.11 打开错误

当用户打开了错误的 DBC 文件并点击确认按钮时，软件会弹出错误窗口提醒用户。

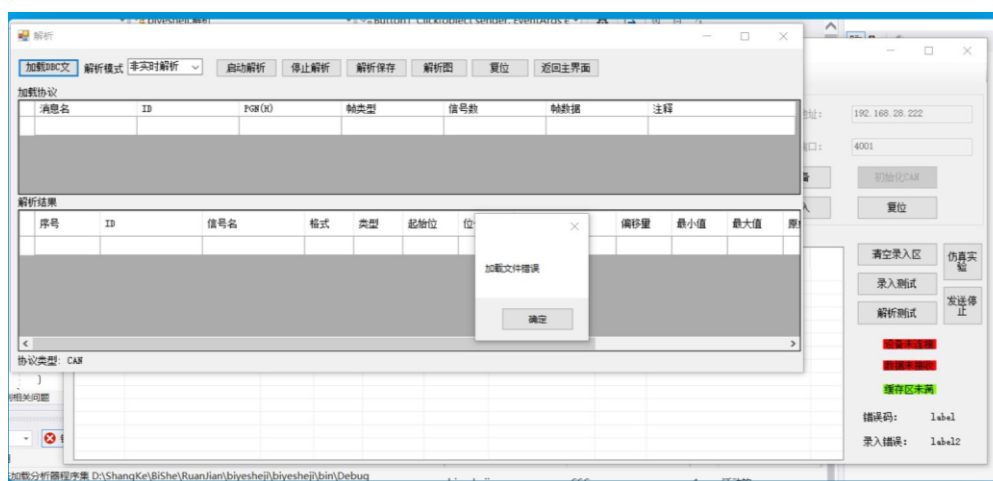


图 4.12 加载错误

当用户失误点击采集界面的清空按钮或者解析界面的复位按钮时，软件会弹出警告窗口提醒用户。



图 4.13 清除误操作

4.7 本章小结

本章首先介绍了用户界面设计，以从采集界面、解析界面到绘图界面的顺序，将每个界面的直观图形和用户使用流程都详细的展现了出来。然后按照使用顺序详细地介绍了各个模块，最开始是采集模块，它也按用户正常使用的流程开始叙述，由硬件的开启到软件的采集显示，层层深入；之后是解析模块，介绍了模式选择、如何解析等功能；最后介绍保存模块和其他模块这些非主要功能，让软件的详细设计更加的完整细致。另外，本章还提及了软件安全性问题，考虑到了用户的误操作，设计了许多弹窗用以保护采集到的数据。

本章为下一章软件测试打下了基础，下一章就是本章节详细设计的成果评估。

5 软件功能测试与验证分析

5.1 单元测试

该软件采用面向对象的 C#语言进行编程，故单元测试的对象为类。而单元测试是指对软件中的最小可测试单元进行检查和验证，故该项测试需将测试单元与程序的其他部分隔离之后再进行测试。

5.1.1 显示测试

测试方法：点击测试区的解析测试按钮，如图 2.3 所示。

测试结果：录入区的显示列表控件则会按照序号、时间、方向、帧格式、帧类型、数据长度（DLC）、帧 ID、通道和数据的顺序，依次显示报文的各种信息。

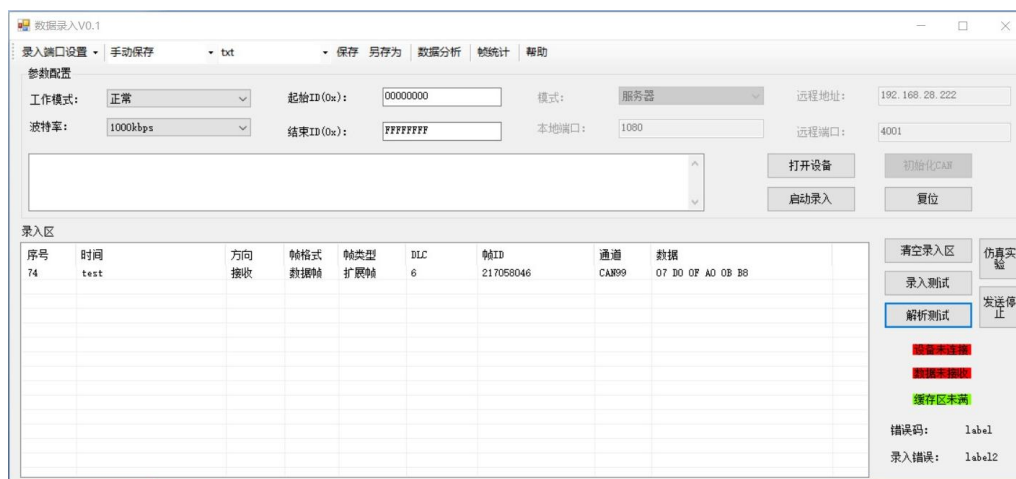


图 5.1 显示测试

注意事项：采集数据时，只要为显示区的显示列表控件赋值的顺序正确，那么显示的报文信息便正确。如何正确赋值的代码在第四章 4.2.3 小节已详述。

5.1.2 调用库函数测试

测试方法：点击采集界面菜单栏数据分析按钮，软件弹出解析窗体。点击加载 DBC 文件按钮，选择 J1939.dbc 文件。

测试结果：此时解析窗体的加载协议区会正确显示协议的内容，如图 5.2 所示。



图 5.2 调用 DLL 测试

注意事项：该段功能的部分代码如图 5.3 所示，调用了 LibDBCManager.dll 文件的代码。原函数由 C/C++ 开发，调用时定义需用 C# 的 IntPtr 类型替代 C/C++ 的指针类型，且库函数原来的 int 型数据应用 C# 的 Int16 型替换。

```
9 个引用
public static class DBCjixi
{
    [DllImport("LibDBCManager.dll")]
    1 个引用
    static public extern int DBC_Init();
    [DllImport("LibDBCManager.dll")]
    3 个引用
    static public extern void DBC_Release(Int32 hDBCHandle);
    [DllImport("LibDBCManager.dll")]
    1 个引用
    static public extern bool DBC_LoadFile(Int32 hDBC, ref GCKJ.FileInfo fileinfo);

    [DllImport("LibDBCManager.dll")]
    1 个引用
    static public extern bool DBC_GetFirstMessage(Int32 hDBC, IntPtr pMsg);
    //static extern bool DBC_GetFirstMessage(Int32 hDBC, ref DBCMessage pMsg);
    [DllImport("LibDBCManager.dll")]
    //static extern bool DBC_GetNextMessage(Int32 hDBC, ref DBCMessage pMsg);
    1 个引用
    static public extern bool DBC_GetNextMessage(Int32 hDBC, IntPtr pMsg);
    [DllImport("LibDBCManager.dll")]
    0 个引用
    static public extern bool DBC_GetMessageById(Int32 hDBC, UInt32 nID, IntPtr pMsg);
    [DllImport("LibDBCManager.dll")]
    2 个引用
    static public extern UInt32 DBC_GetMessageCount(Int32 hDBC);
}
```

图 5.3 调用代码

5.1.3 保存测试

测试方法：点击菜单栏保存按钮。

测试结果：显示区的所有数据信息都按照各自的通道保存在相应的通道文件中，且用户界面弹出保存成功窗体。

注意事项：保存为 txt 文档时软件进程迅速，保存为 Excel 文档时，软件会出现卡顿现象。预估原因可能为调用的微软的 Excel 类库函数的方法占用了一定的内存。

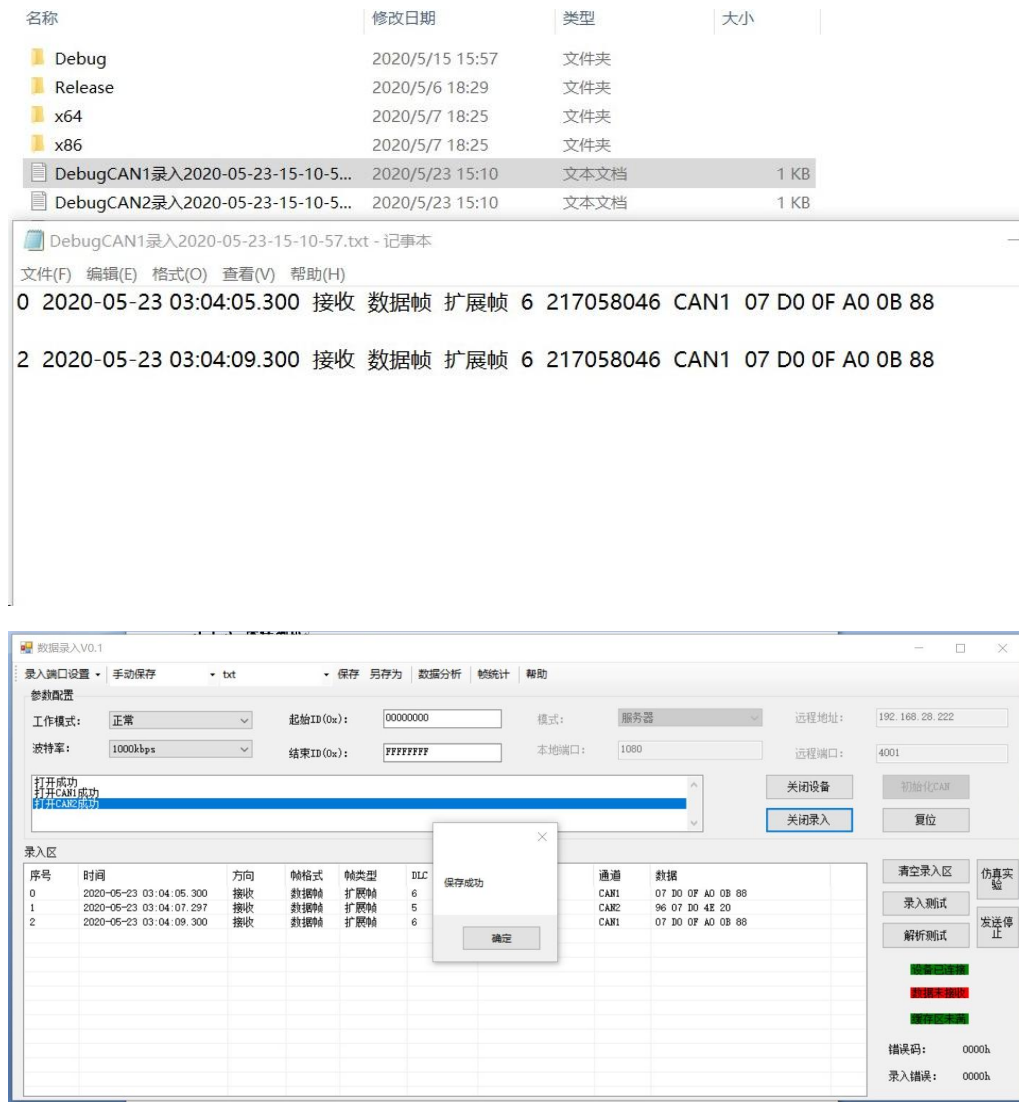


图 5.4 保存测试

5.2 集成测试

集成测试也被称为联合测试，它是在单元测试的基础上，将所有模块按照设计要求组装成为系统，然后进行集中的测试。该项测试主要为解决一些模块虽然能够单独地工作，却并不能和其他模块联系起来之后正常工作的 bug。该项测试的原理主要在于某些部分反映不出来的问题，在全局上很可能得到暴露。

5.2.1 采集测试

测试方法：设备正常打开，软件初始化，并且点击启动录入按钮后，点击录入测试按钮。

测试结果：两个通道都能互相正常的收发数据，即实际工程上两个通道都能采集数据并准确的显示出来。

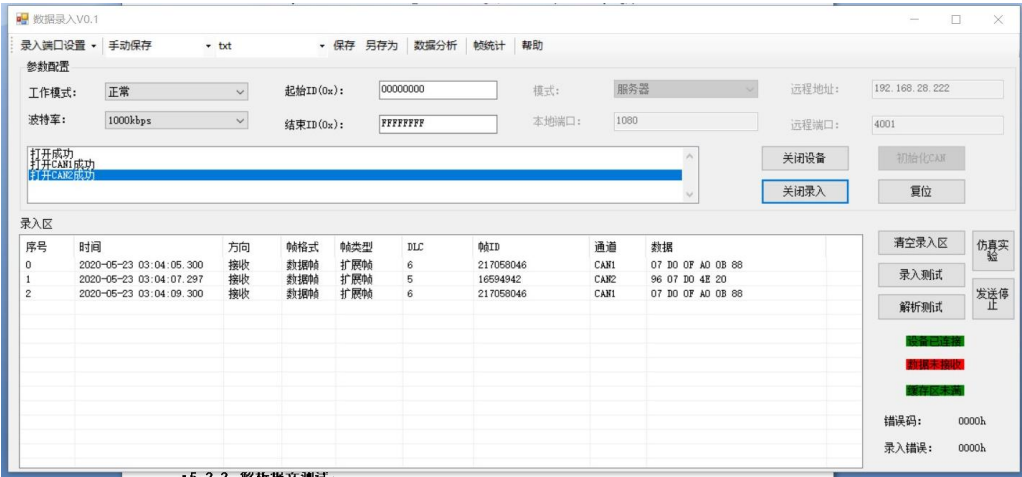


图 5.5 采集测试

注意事项：仿真方法是用某一个通道模拟传感器发送报文，另一个通道则为该软件的有效端口采集报文。发送报文中时间标志的值可以让保存的时间项显示硬件设备的时间或者软件系统的时间。

5.2.2 解析测试

测试方法：在采集测试后，点击数据分析并加载 J1939 文件。1、采集界面选中某条报文，且解析界面选择非实时解析，点击启动解析；2、解析界面选择实时解析，点击启动解析。

测试结果：1、 解析界面结果区域显示该条报文的信息，读取该信息可知该条报文有三条信号，内容为化学反应加入催化剂的转化率、进气量、出气量。真实值如图 5.6 所示，分

别是 60%、2000L/h、1000L/h。2、软件默认从第一条报文开始解析直至最后一条报文，报文全部解析后，解析界面弹出解析完毕提示窗体，解析的结果在结果区显示。

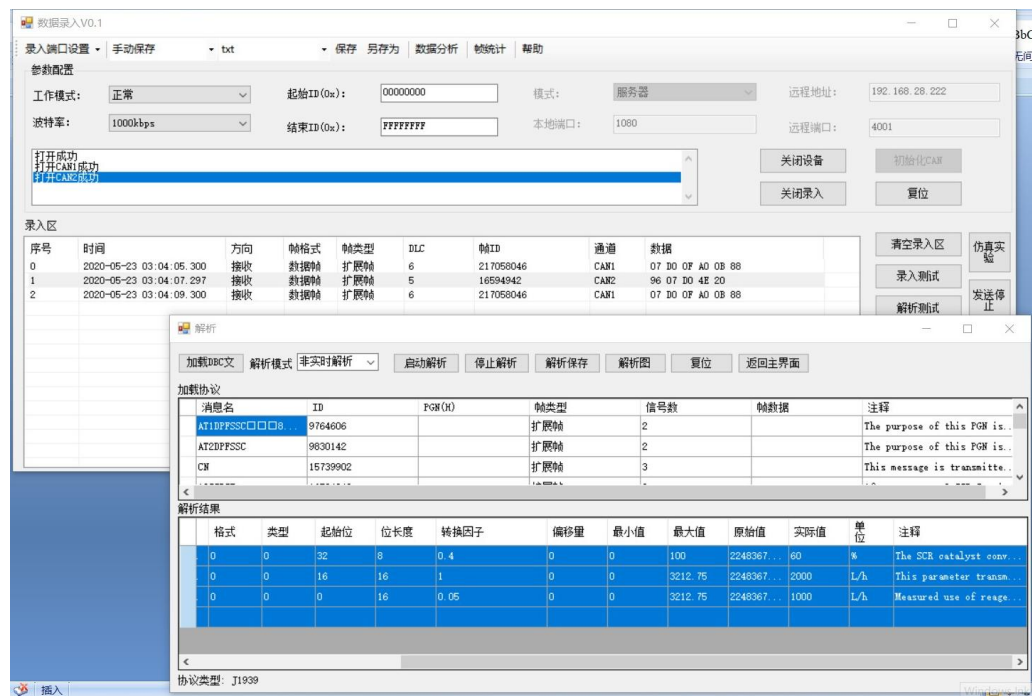


图 5.6 单次解析

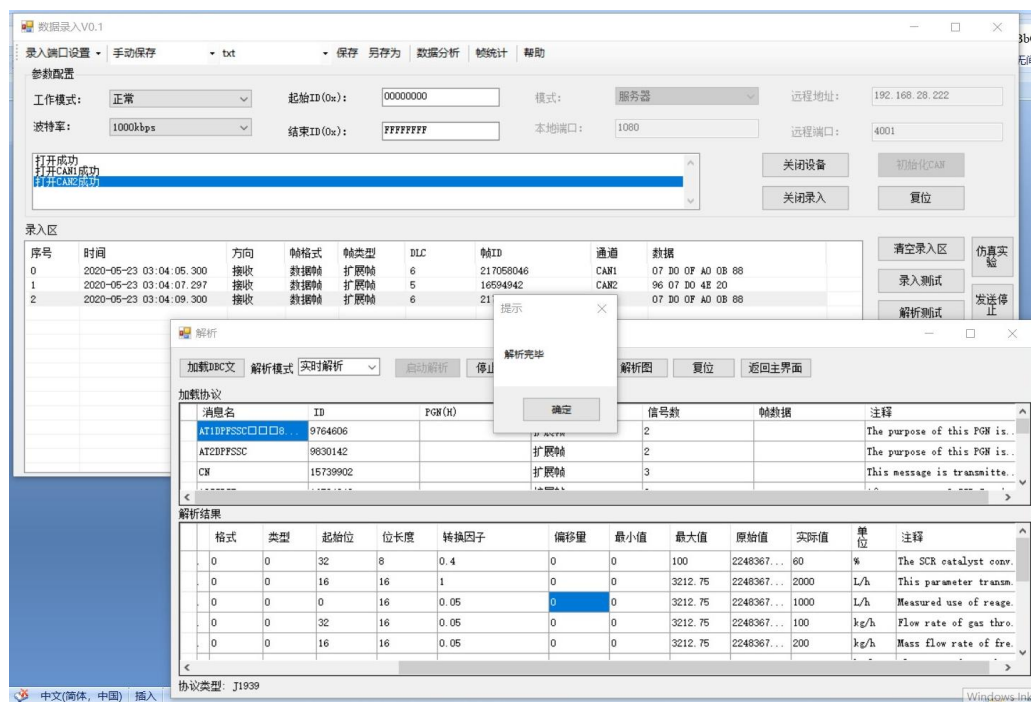


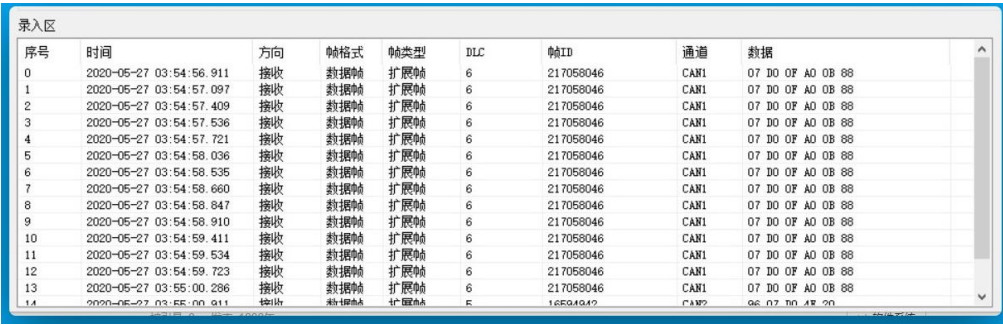
图 5.7 实时解析

注意事项：实时解析模式下，用户不需要选中报文，软件默认从第一条解析到最新的一条，然后跳出解析完毕界面，若要从某一条报文开始解析，则在启动解析前选中那一条报文，之后不要误点击采集界面报文区域。

5.2.3 实时性测试

测试方法：在设备某一通道正常录入数据时，手动利用另一通道向正在采集数据的通道发送固定的代码，可在短时间内点击多次。

测试结果：根据显示结果的时间栏可知，软件能准确的实时采集数据，相隔一毫秒的数据都能准确采集不丢帧，这证明了采集时间能精确到毫秒级别。



序号	时间	方向	帧格式	帧类型	DLC	帧ID	通道	数据
0	2020-05-27 03:54:56.911	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
1	2020-05-27 03:54:57.097	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
2	2020-05-27 03:54:57.409	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
3	2020-05-27 03:54:57.536	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
4	2020-05-27 03:54:57.721	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
5	2020-05-27 03:54:58.036	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
6	2020-05-27 03:54:58.535	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
7	2020-05-27 03:54:58.660	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
8	2020-05-27 03:54:58.847	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
9	2020-05-27 03:54:58.910	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
10	2020-05-27 03:54:59.411	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
11	2020-05-27 03:54:59.534	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
12	2020-05-27 03:54:59.723	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
13	2020-05-27 03:55:00.286	接收	数据帧	扩展帧	6	217058046	CAN1	07 D0 0F A0 0B 88
14	2020-05-27 03:55:00.911	接收	数据帧	扩展帧	6	16594947	CAN2	06 07 D0 A8 20

图 5.8 实时性测试

5.2.4 可靠性测试

测试方法：软件每次使用时记录各个模块的状态，如采集模块是否正常采集，解析模块是否报错等，以及软件报错的错误信息，使用后开关软件一次而后下次使用时继续记录，重复一定次数。

测试结果：经过了上百次的录入、解析和保存实验，数据并未出现过错录、解析乱码、保存失败等现象，可见该软件在实验室内使用具备一定的可靠性。

5.3 数据采集案例

仿真：用 MATLAB 模拟了小车的一系列运动状况并编写了其四轮速度的 CAN 报文。其中小车先做直线的加速运动再做左转弯减速运动，最后减速停止。

测试方法：本次实验用硬件设备的 CAN2 通道发送，CAN1 通道接收进行仿真。在启动录入后，点击采集界面的仿真实验按钮即可。

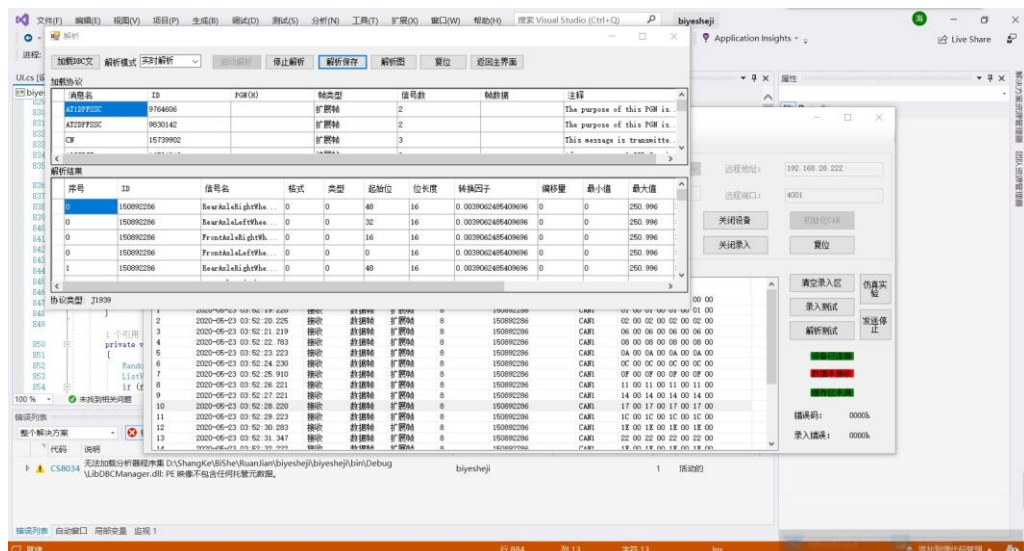


图 5.9 实验采集图

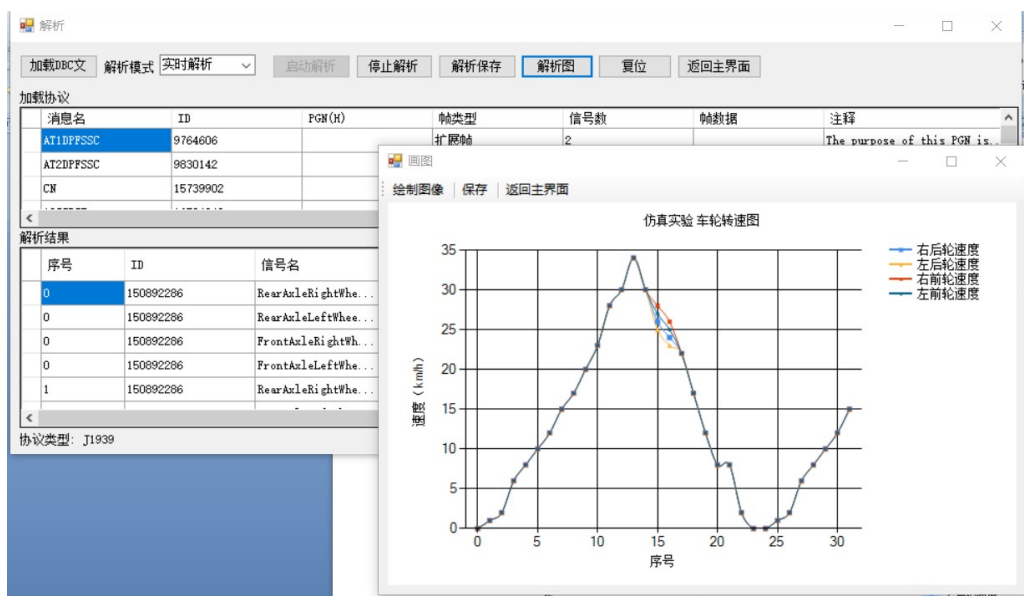


图 5.10 解析结果图

测试结果：从采集界面看，报文全部都采集到了，不存在丢包丢帧现象且报文信息未出现错误，可见采集功能完善。从解析界面和解析图看，每条报文的真实值都正确的解析了出来且解析图直观清晰地绘制了解析结果（真实值），该结果和实际速度一致，实际速度表如表 5.1 所示。这说明本次仿真实验已经成功，软件的采集功能、解析功能和保存功能都正常。

表 5.1 小车四轮速度表

右后速度 (km/h)	左后速度 (km/h)	右前速度 (km/h)	左前速度 (km/h)
0	0	0	0
1	1	1	1
2	2	2	2
6	6	6	6
8	8	8	8
10	10	10	10
12	12	12	12
15	15	15	15
17	17	17	17
20	20	20	20
23	23	23	23
28	28	28	28
30	30	30	30
34	34	34	34
30	30	30	30
26	25	28	27
24	23	26	25
22	22	22	22
17	17	17	17
12	12	12	12
8	8	8	8
2	2	2	2
0	0	0	0

注意事项：仿真的速度报文一共 24 条，软件仿真时发送完毕会从头重复发送，故解析图右侧出现了重复的折线，但该部分的折现依然和左侧开始的那几组数据折线相同，故长时间采集和解析数据并未对软件的性能产生影响，软件各模块功能正常。

5.4 遇到问题及解决思路

5.4.1 需求分析和界面布局问题

由于此前从未使用过类似的软件和硬件设备，我对课题研究的问题很疑惑，如该软件的使用群体、具体功能等。但是在老师的细心讲解和指导下，我查阅了大量的相关资料，最终列出了需求分析表初稿并设计了用户使用界面。

对于用户界面的布局，大多数商用成熟的软件采用了菜单栏+显示区的布局，用户通过点击软件上部分菜单栏中的按钮，即可实现软件的各种功能。而我考虑到该软件不仅需要录入数据而且需要解析数据，单独的菜单栏装在太多的按钮会显得臃肿，故采用了菜单栏和按钮区组合的方式，将各个功能按钮合理的分散开来。这使得该软件的用户设计更加的合理，提高了用户的使用舒适性。

5.4.2 数据流设计问题

由于需求分析得到的结果，该软件并未专用的数据流，如 files 表等，而是采用了外部数据及时保存为文件，内部可传递数据定义为全局静态量的方式。这种方式能尽可能满足数据的存储与调用的情况下，降低软件的代码复杂度。

5.4.3 软件的代码问题

刚开始时不清楚是否需要写底层调用硬件的汇编代码和软件的驱动程序，故代码仅停留在了软件界面设计这一步。买到了硬件设备后，通过学习 C#语言的调用动态链接库的方式，可以直接调用厂商的函数库来操控硬件，此时软件代码的可读性和规范性都极大地提高了。举以下三个问题为例。

1、无法正常调用库函数，调用后显示“未找到对应的 DLL 文件”。

生成项目文件的时候要注意是 X86 还是 X64 还是 Any CPU。某些库函数用 X86 编写的话，其对应的 DLL 也应该用 X86 版本的，否则无法调用 DLL。解决方案平台选择 Any CPU，则调用的两类函数库统一用 X86 版本，不能用 X64 版本。

2、采用类重构的方式在窗体间传递数据，导致类的类型膨胀，无效代码增多。

将字符串等数据的属性定义为公用、静态类型即可在三个窗体间自由传递。

3、C#的二进制数据受空格影响，解析结果错误。

C#语言将二进制数据的类型定义为字符串类型，故进行计算和操作是需要用字符串的替换方法将空格（“ ”）替换为无（“”），然后才能正常计算和按位操作。

结 论

由于数据采集技术和 CAN 总线技术经历了几十年的发展，它们已经在各个领域都被广泛地应用，基于 CAN 总线的数据采集系统普遍存在于实验室、汽车等场所之中，而本文所设计的多路实时采集软件则是在 CAN 总线工程师们创建的软、硬件库基础上进行的二次开发。它有着实时采集、多通道采集、显示信息、保存报文和解析数据等功能，可以采集用 USB-CAN2 C 硬件设备录入的信息。为实现这些功能，本次设计运用了多线程、调用动态链接库和数据流的读写等技术。经由这些技术，本次软件设计的绝大部分问题都已解决，而匹配更多的硬件采集设备成了下一个待解决的问题。

致 谢

感谢指导老师在软件设计前为我指明了方向，并在设计过程中纠正我的错误。在和老师的交流讨论之后，本次软件设计为今后进行的软件设计或者软件测试的工作打下了基础。另外感谢室友帮助本人解决了本文的部分格式问题。

参 考 文 献

- [1] 王琳, 商周, 王学伟. 数据采集系统的发展与应用[J]. 电测与仪表, 2004, 41(8): 4-8.
- [2] 王海山. 基于 USB-CAN 总线的车辆监控数据采集系统[D]. 成都: 电子科技大学, 2018.
- [3] 毛春奎, 张颖超. 基于 CAN 总线的气象数据采集系统[J]. 南京信息工程大学学报, 2009, 1(2): 165-168.
- [4] 鲁力, 张波. 嵌入式 TCP / IP 协议的高速电网络数据采集系统[J]. 仪器仪表学报, 2009, 30(2): 405-409.
- [5] 杜鹏, 晏亮, 高保成. 基于电力调度数据网的广域数据采集方案[J]. 电力系统自动化, 2019, 43(13): 156-161.
- [6] 郭世友, 辛海华, 沈志军. CAN 总线在数据采集系统中的应用[C]// 全国遥测遥控技术年会, 2003: 246-252.
- [7] 王毅峰. 基于 CAN 总线的分布式数据采集与控制系统[J]. 工业控制计算机, 2001, 13(5): 34-38.
- [8] 朱敏, 张崇巍, 谢震. CAN 总线在数据采集与控制系统中的应用[J]. 合肥工业大学学报(自然科学版), 2002, 25(3): 345-349.
- [9] 陈敏, 汤晓安. 虚拟仪器软件 LabVIEW 与数据采集[J]. 小型微型计算机系统, 2001, 22(04): 501-503.
- [10] 游雪峰, 文玉梅, 李平. 以太网分布式数据采集同步和实时传输研究[J]. 仪器仪表学报, 2006, 27(04): 345-349.
- [11] 周振安, 范良龙, 王秀英等. 数据采集系统的设计与实践[M]. 北京: 地震出版社, 2005.
- [12] Huang R W, Huang H W. Development of Real Time Monitoring System using Controller Area Network[J]. Materials Science Forum, 2006, (505-507): 475-480.
- [13] Woody D P, Wiitala B, Scott S L, et al. Controller-Area-Metwork bus control and Monitor System for a radio astronomy interferometer[J]. Review of Scientific Instruments, 2007, 78(9): 094-501.
- [14] 陶楚良. 数据采集系统及其器件[M]. 北京: 北京工业大学出版社, 1988.
- [15] 韩冰, 李芬华. GPRS 技术在数据采集与监控系统中的应用[J]. 电子技术, 2003, (8): 26-29.
- [16] 罗义军, 陈松. 基于 PCIe 接口的高速数据采集系统[J]. 仪表技术与传感器, 2019, (5): 91-95.
- [17] 徐聪辉, 李彩, 张振昭. ADS1262 多通道数据采集系统设计[J]. 中国测试, 2019, 45(9): 112-117.

附录 A 软件实际使用图

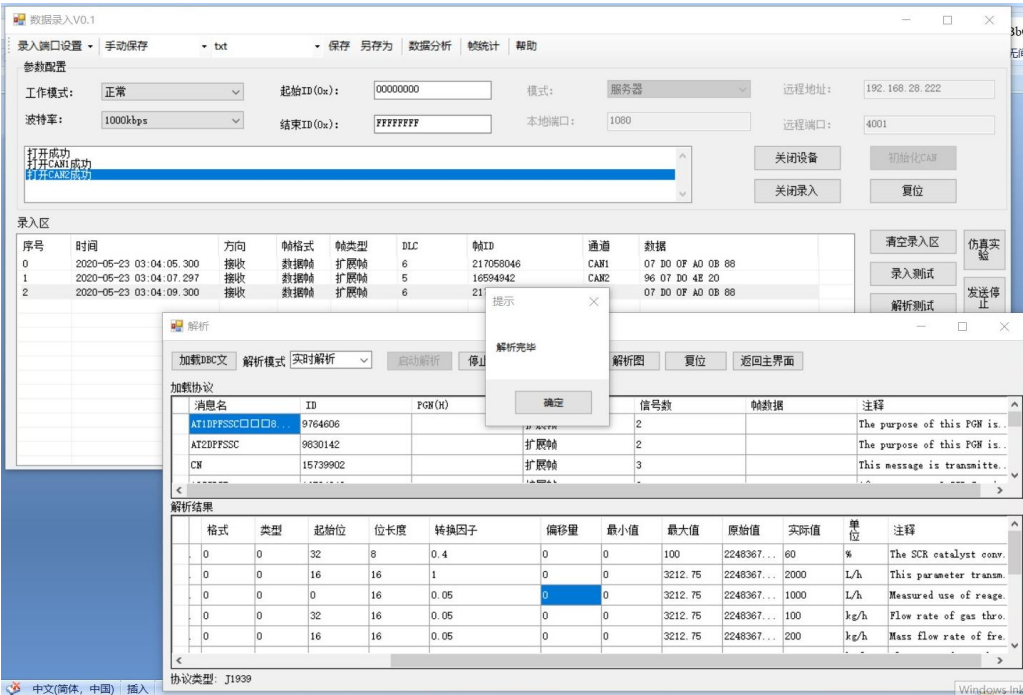


图 1 实时解析

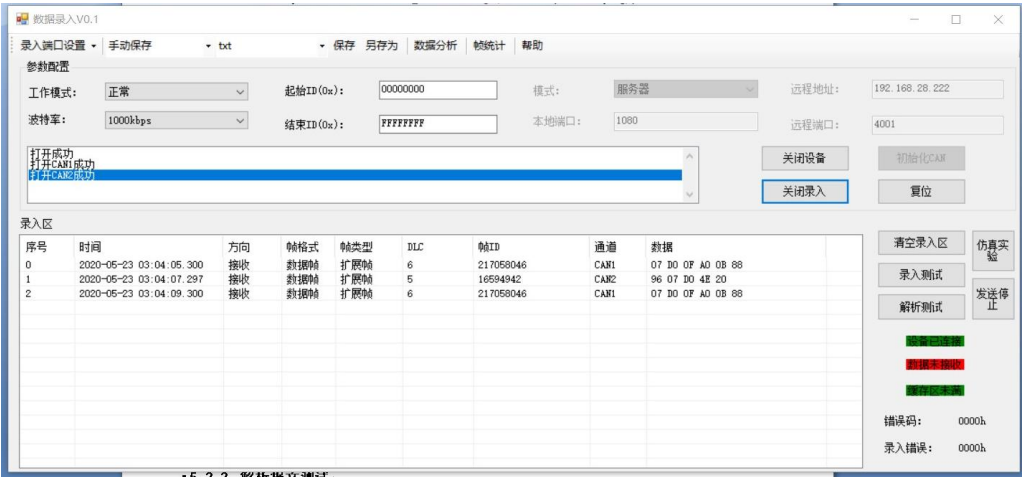


图 2 多路采集数据

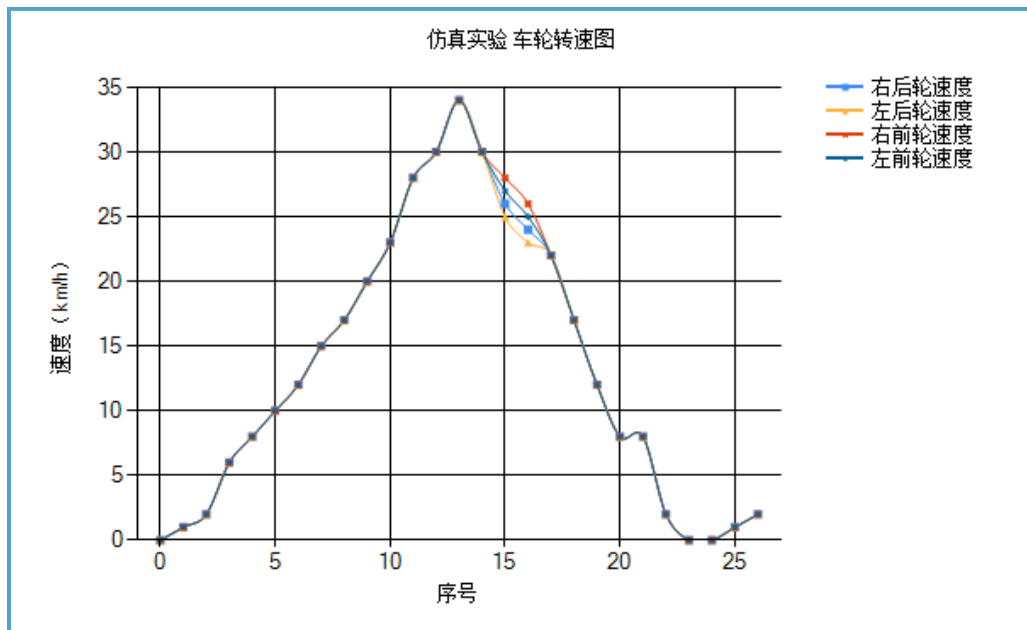


图 3 仿真实验解析 (结果图)

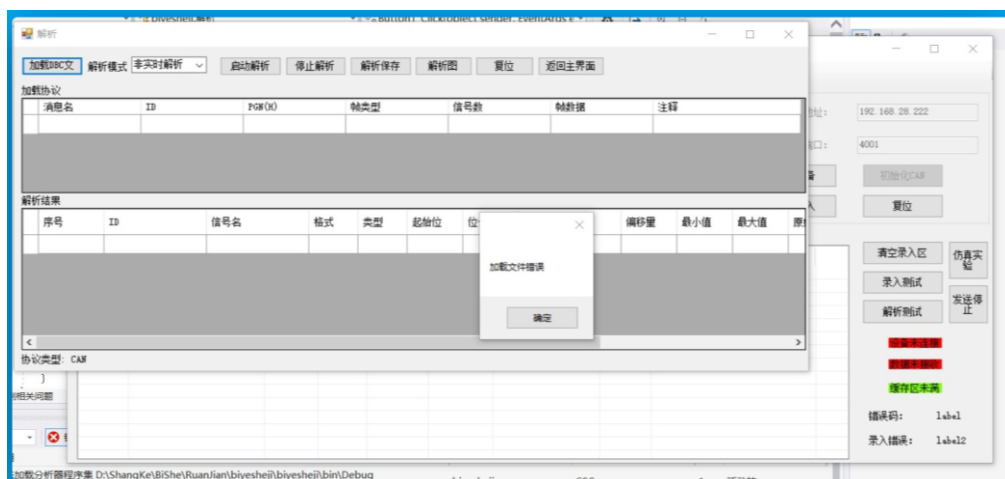


图 4 打开错误的 DBC 文件