****

****

**CanTool App详细设计说明书**

Team 24组员

魏卿、苗旭、吴虹、汪嫱

2017年10月

目录

[1．引言 1](#_Toc497578414)

[1.1编写目的 1](#_Toc497578415)

[1.2项目背景 1](#_Toc497578416)

[1.3定义 1](#_Toc497578417)

[1.4参考资料 2](#_Toc497578418)

[2.总体设计 3](#_Toc497578419)

[2.1需求概述 3](#_Toc497578420)

[2.1.1 CanTool Windows App系统功能需求 3](#_Toc497578421)

[2.1.2 CanTool Windows App系统用例图 3](#_Toc497578422)

[2.1.3 CanTool Windows App系统功能模块图 4](#_Toc497578423)

[2.2系统界面设计 6](#_Toc497578424)

[3．程序描述 6](#_Toc497578425)

[3.1数据解析模块 6](#_Toc497578426)

[3.1.1模块类图 6](#_Toc497578427)

[3.1.2详细程序流程 7](#_Toc497578428)

[3.1.3流程活动图 7](#_Toc497578429)

[3.2数据传输模块 8](#_Toc497578430)

[3.2.1模块类图 8](#_Toc497578431)

[3.3数据存储模块 9](#_Toc497578432)

[3.3.1模块类图 9](#_Toc497578433)

[3.5数据显示模块 10](#_Toc497578434)

[3.5.1模块类图 10](#_Toc497578435)

### 

# 1．引言

## 1.1编写目的

本详细设计说明书是对CanTool Windows App项目进行详细设计，在概要设计的基础上进一步明确系统结构，详细地介绍系统的各个模块，为进行后面的实现和测试做准备。本详细设计说明书的预期读者为本项目小组的成员以及对该系统感兴趣，在以后想对系统进行扩展和维护的人员。

## 1.2项目背景

在现代汽车控制技术中，汽车中会使用多个电子控制装置（ECU：Electronic Control Unit）对整车进行控制。而ECU之间的信息交换更多地依赖于CAN(Controller Area Network)总线的网络连接方式来完成。为了检测和控制CAN bus的信息内容，需要使用CAN bus检测设备。CanTool装置是完成CANbus检测的工具。

为了实现CAN数据的显示及控制，需要使用本文提出的CanToolApp软件。该软件需要将连接在CAN总线上的CanTool装置采集的CAN信息发送到上位机（移动终端Android、iOS、Windows PC），并由运行在上位机中的CanToolApp软件接收这些信息，显示在用户图形界面上。同时在CanToolApp的界面上还可以设定CAN信息，通过GUI按钮将信息发送给CanTool装置，CanTool装置将按照规定的信息格式，将信息发送的CAN总线上。此外，CanToolApp可以设定CAN总线的通信参数，并通过相应的命令设置CanTool装置的CAN通信参数，以使CanTool装置能够与CAN总线上的其他被测ECU进行正常的通信。

## 1.3定义

|  |  |
| --- | --- |
| **名称** | **说明** |
| CAN bus | （Controller Area Network）控制器区域网络的简称。 控制器区域网络（CAN总线）是一个强大的车辆总线标准，旨在允许微控制器和设备在没有主机的情况下相互通信。它是一种基于消息的协议，最初设计用于汽车内的多路电线以节省铜，但也用于许多其他情况。 |
| ECU | （Electronic Control Unit）车辆中使用的控制装置。 |
| CanTool装置 | 用于CANBUS的CAN信息采集与发送的装置，CanToolApp用于上位机与CanTool装置进行通信，并完成CAN信息、信号的显示与设定。 |
| CAN message | CAN message 由CAN id，DLC，data构成。 |
| CAN signal | Can Signal是分布在CAN message中的CAN信号。具有一定物理意义。 |
| CAN信息和信号数据库 | 用于存储CAN信息的组成信息，和CAN信号的相关参数设置。 |
| Little endian ／Big Endian | 数据在存储空间中保存的方式。 |
| WebAPI | Web API是用于Web服务器或Web浏览器的应用程序编程接口（API）。 这是一个Web开发概念，通常仅限于Web应用程序的客户端（包括使用的任何Web框架），因此通常不包括Web服务器或浏览器实现细节，如SAPI或Web浏览器引擎API，除非公开远程网络应用程序可以访问。 |

## 1.4参考资料

1.本文参考项目组的《CanTool-Problem-Statement》。

2.本文参考项目组的《CanTool-Glossary》。

3.本位参考项目组的《单元测试说明书》。

4.本文参考项目组的《需求规格说明书》。

# 2.总体设计

文档中主要对于CanTool Windows App系统的总体逻辑、物理、技术结构进行了描述，对于功能模块进行了划分，及一些系统的总体设计进行了描述。此份《详细设计说明书》是基于《概要设计说明书》文档的一些软件、概念细节进行进一步说明与补充。因此，在进入第三章程序描述之前，先对总体设计的重要部分进行详细说明。

本章总共分为三个部分，分别是：对于整个项目的功能需求划分模块、软件整体结构图及其说明、系统总体界面设计。

## 2.1需求概述

### 2.1.1 CanTool Windows App系统功能需求

阅读需求文档，我们将CanTool App的总体需求进行了总结，并讲问题初步进行了划分，如图1。

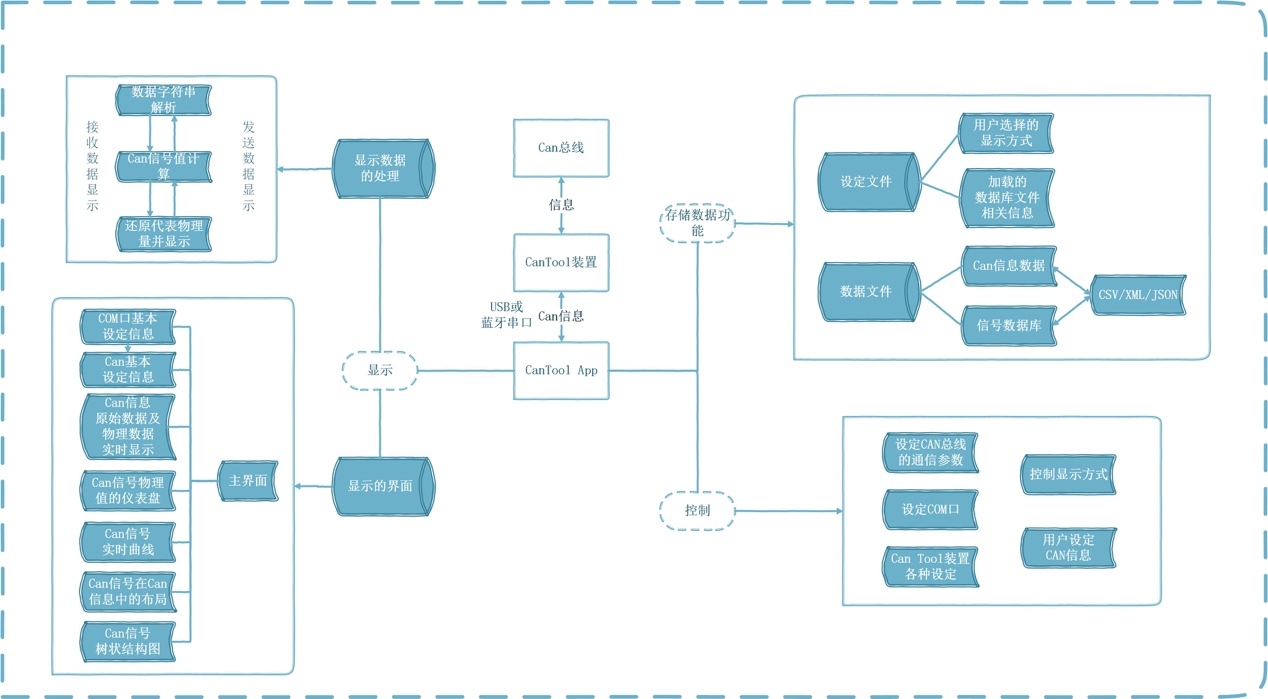


图 1 CanTool需求划分

连接在CAN总线上的CanTool装置采集的CAN信息发送到上位机（移动终端Android、iOS、Windows PC），并由运行在上位机中的CanTool App软件接收这些信息，显示在用户图形界面上。CanTool装置与上位机通过USB串口或蓝牙RFComm实现UART串口通信。系统整体的业务逻辑重心在于Can信息数据的解析与Can信号物理值的反解析，进而将解析结果显示在GUI界面上。于此衍生的功能包括数据的存储与控制。

### 2.1.2 CanTool Windows App系统用例图

用例图是指由参与者（Actor）、用例（Use Case），边界以及它们之间的关系构成的用于描述系统功能的视图。用例图（User Case）是外部用户（被称为参与者）所能观察到的系统功能的模型图。用例图是系统的蓝图。用例图呈现了一些参与者，一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行建模。

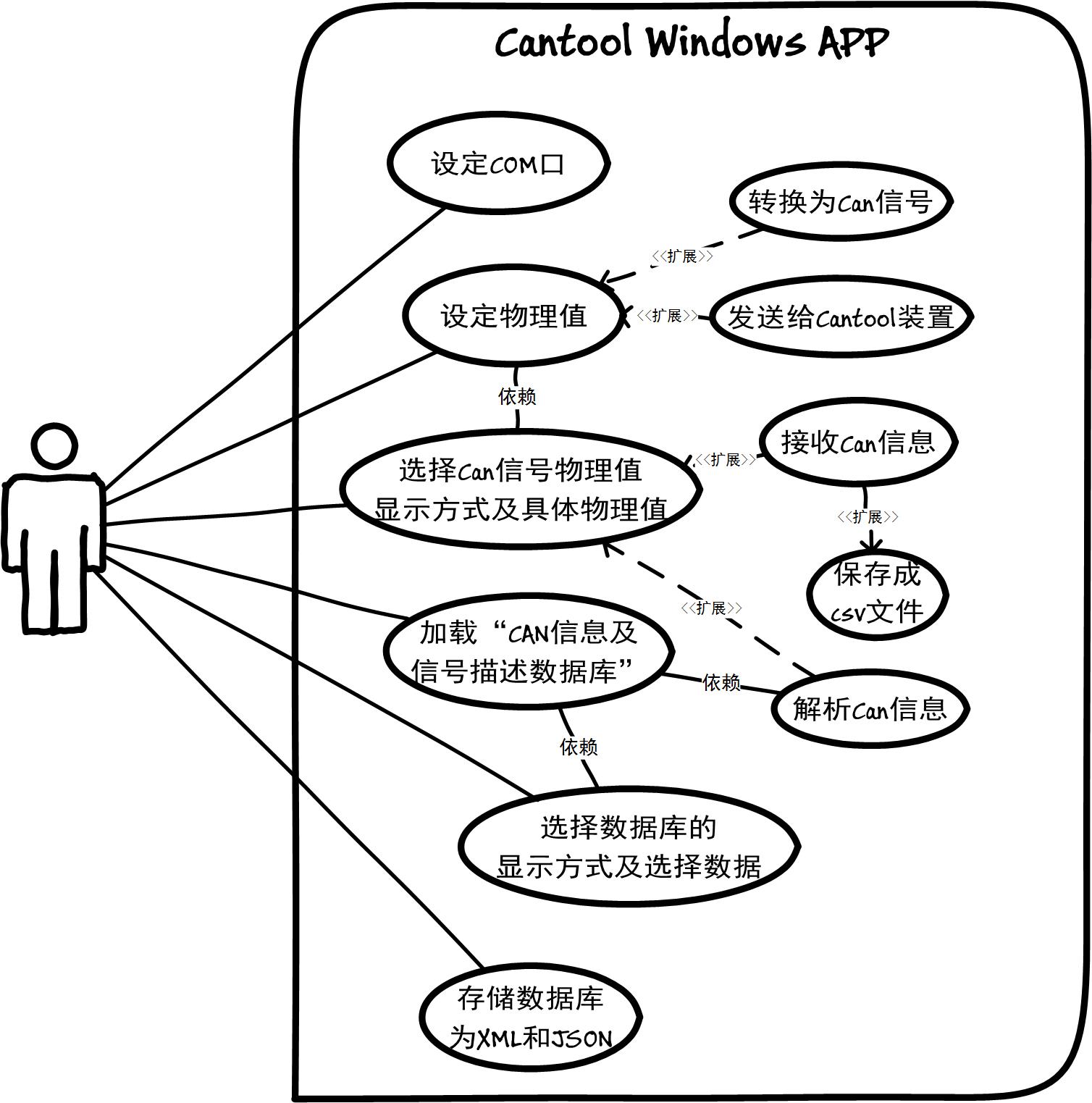


图 2 CanTool App用例图

CanTool Windows App主要是用户与App进行交互。

扩展关系的基本含义和泛化关系类似，但在扩展关系中，对于扩展用例有更多的规则限制，基本用例必须声明扩展点，而扩展用例只能在扩展点上增加新的行为和含义。与包含关系一样，扩展关系也是依赖关系的版型。在用户解析Can信息以及显示数据库之前，必须先加载数据库，在接收和发送数据库之前，必须先设置Com口。扩展关系是对基用例的扩展，基用例是一个完整的用例，即使没有子用例的参与，也可以完成一个完整的功能，例如，用户设置完物理值，首先由App转换为Can信号，再发送给CanTool装置。

### 2.1.3 CanTool Windows App系统功能模块图

概括了CanTool Windows App系统的需求后，按照功能需求将系统划分为若干功能模块。

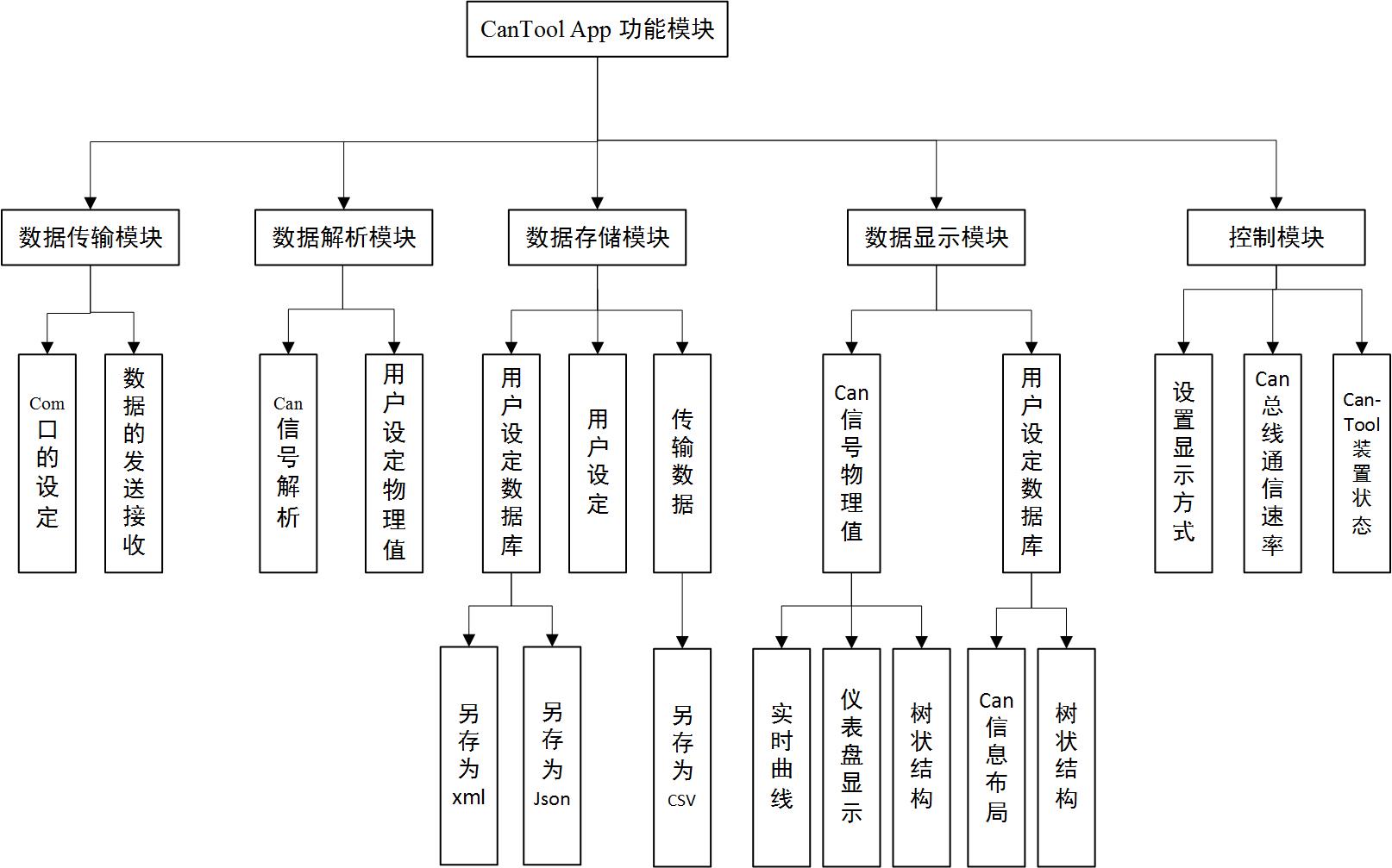


图3 CanTool Windows App系统功能模块图

1）. 数据传输模块

数据传输模块主要针对于App与CanTool装置通过虚拟串口的数据传输。在这个模块，首先要进行Com口的设定，能够搜索到本机所有可使用的COM口，选择可用的com口进行通信，并在弹出式ComboBox中以列表方式让用户选择CanTool装置在上位机中映射的COM口。与上位机连接的串口号需要用户选择，并设置相应COM口波特率115200、数据位数8、停止位数1。

打开Com口后，需要通过串口进行通信，进行数据的发送与接收。

2）. 数据解析模块

能够对接收到的多个CAN信息，根据用户提供的CAN信息和信号数据库，通过CAN信息及CAN信号数据库进行解析，然后得到CAN信息中包含的各种CAN信号值，将此CAN信号值进一步进行计算，还原该信号所代表的物理量的信息，并显示在GUI界面上。；将用户设定的物理值转换为CAN信号值，将CAN信息中包含的所有CAN信号合成完整的CAN信息后，发送给CanTool装置，发送到CAN总线上。要发送的CAN信息也采用同种方式将用户输入的物理值转换为CAN信号，并依据CAN信号描述数据库将属于同一个CANID的信号合成为字符串发送给CanTool装置。

3）. 数据存储模块

对于用户提供的CAN信息和信号数据库另存为xml和JSON (JavaScript Object Notation)格式。也可以已将xml或Json格式的数据库，转换为CAN信息和信号数据库格式。

对于设定文件，能够实现CanTool装置的CAN速率设置、进入CAN工作状态（Open）、进入CAN初始化状态（Close）。设定内容可保存到CanToolApp设定文件中，供下次使用。用户选择的显示方式也可保存到CanToolApp设定文件中，供下次使用。

对于接收到的所有CAN信息数据，实时保存为数据文件，格式为CSV格式，或自定义。

4）. 数据显示模块

对于Can信号物理值，让用户选择某些接收到的CAN信号，显示其变化的实时物理值曲线；显示时可以让用户选择仪表盘方式显示接收到CAN信号物理值；

对于用户提供的CAN信息和信号数据库，完成CAN信号数据的解析以及CAN发送信息的组装，可以显示CAN信号在CAN信息的布局；加载用户提供的CAN信息和信号数据库，可以树状结构显示在GUI界面中；解析过后的数据，也可以树状结构显示在GUI界面中。

5）. 控制模块

用户可以控制选择显示方式；可以指定CAN信息的发送周期（0-65535ms即0x0000-0xFFFF）；可以设置CAN总线的通信速率；可设置CANtool装置的CAN速率设置、进入CAN工作状态（Open）、进入CAN初始化状态（Close）。

## 2.2系统界面设计

CanTool App运行在电脑上，因此可模仿一般电脑软件的界面，设置菜单栏，以及界面的跳转。针对CanTool的内部业务功能划分，最终的界面导航图如图4。

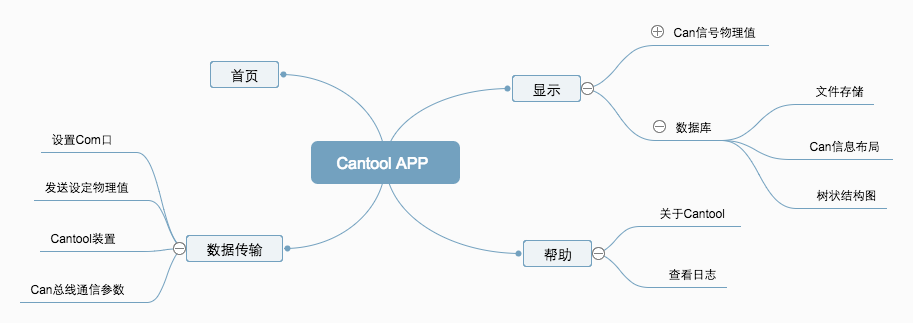


图4 CanTool App界面导航图

# 3．程序描述

第三章详细设计针对整个App：通过模块的类图以及对应的活动图，对于该模块程序逻辑代码的细节进行不同方式的表现；根据测试场景列出对应结果及测试过程中的测试要点。

## 3.1数据解析模块

### 3.1.1模块类图

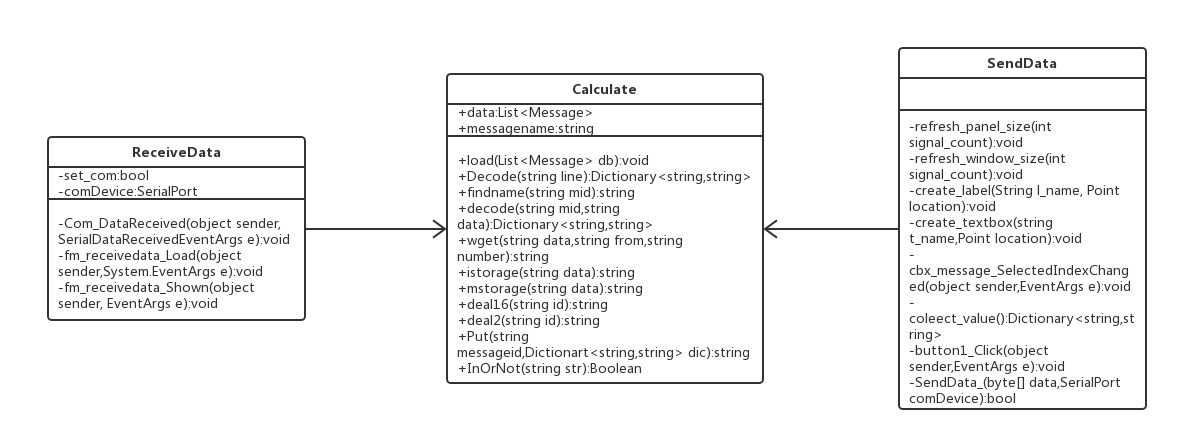


图5 数据解析模块主要类图

Calculate类的编写，这个类里边主要实现了诸多方法，分模块来说：

1） 对来自CanTool装置发送的信号进行解析，涉及的方法有：

Private Dictionary<string,string> Decode(string line)

Private Dictionary<string,string> decode(string mid, string data)

Private string wget(string data, string from, string number)

Private string istorage(string data)

Private string mstorage(string data)

Private string deal16(string id)

Private string deal2(string id)

2) 对用户设置的物理值放入到Can信息里

Private string Put(string messageid, Dictionary<string, string> dic)

3) 加载数据库

Private void load(List<Message> db)

### 3.1.2详细程序流程

首先加载数据库，获得数据库的信息，以便后续解析设置功能的使用。然后是对发送过来的数据进行解析，将传过来的字符串输入进Decode方法里，在这个方法里先对信息进行拆分，获得十六进制的id信息以及数据信息。吧这两个得到的值传入到decode这个方法里。进入到decode这个方法里之后，通过循环遍历数据库信息，找到与id匹配的message，并对该message的signal进行遍历且依次进行解析。把data根据存储类型放入到intel处理或者Motorola处理。处理完的数据再放入到wget方法中从而得到指定位数的数据并通过公式计算出该signal的物理值。

设置信息是通过用户设置物理值，然后根据该signal的信息确定放入can信息的位置，依次循环遍历每个设置的信息然后组合起来，发送到CanTool装置。

### 3.1.3流程活动图

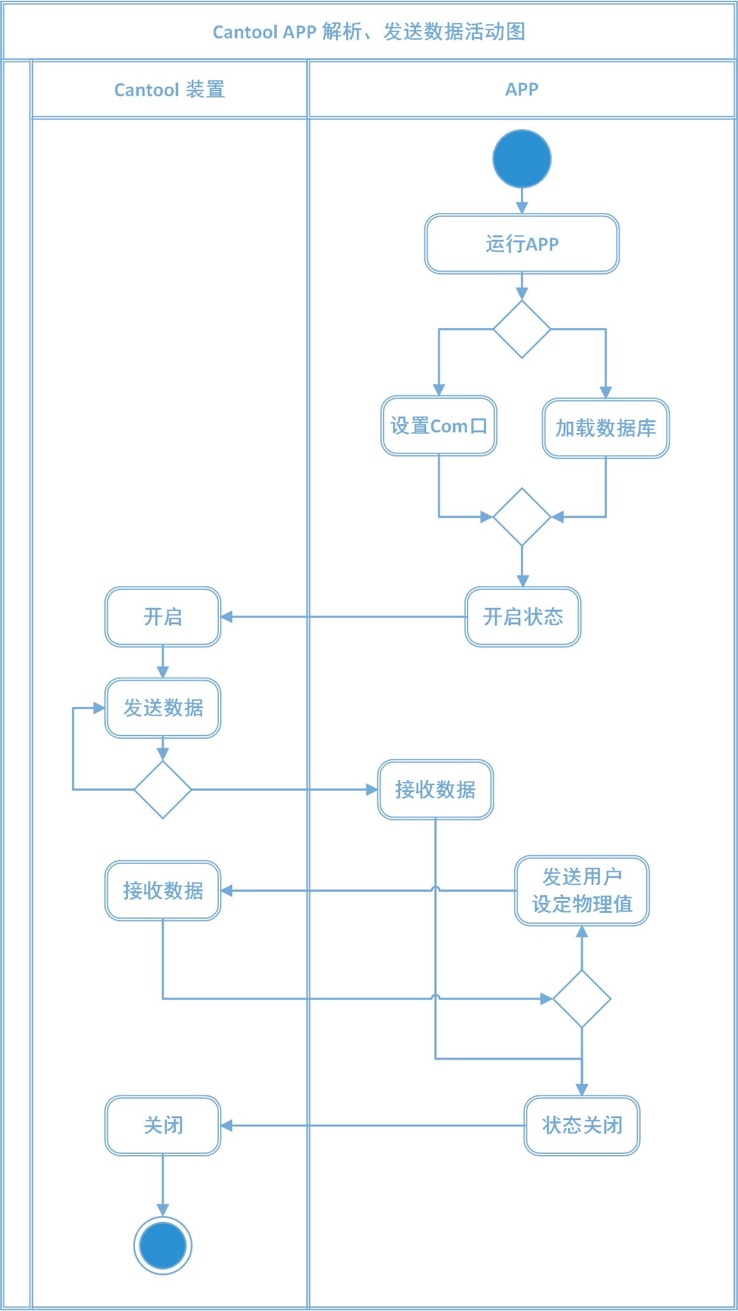
****

图 6 数据解析活动图

运行App之后，为了保证解析过程的顺利进行，需要先设置Com口的相关参数，选择可使用的Com口，然后加载用户提供的“CAN信息和信号数据库”，随后可通过串口向CanTool装置发送开启状态的命令，当成功接收到命令，开启状态的同时，向CanTool App发送成功开启的握手消息。紧接着可以开始循环发送数据；接收数据也是如此。发送或者接受过程结束时，由CanTool App向CanTool装置发送状态关闭命令。

## 3.2数据传输模块

### 3.2.1模块类图

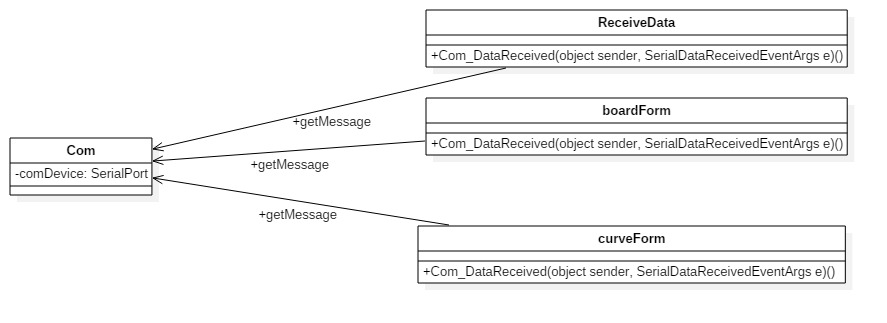
****

图 7 Com口设定相关类图

利用23个设计模式中的观察者Observer模式，在Com类中，订阅了其他相关的接口，当它发生改变时，通过getMessage方法会通知旗下订阅他消息的几个类，分别为ReceiveData类、BoardForm类、CurveForm类，绘制、显示不同的数据出来，达到实时显示Can信息数据的目的。

观察者模式（有时又被称为发布（publish）-订阅（Subscribe）模式、模型-视图（View）模式、源-收听者(Listener)模式或从属者模式）是软件设计模式的一种。在此种模式中，一个目标物件管理所有相依于它的观察者物件，并且在它本身的状态改变时主动发出通知。这通常透过呼叫各观察者所提供的方法来实现。此种模式通常被用来实现事件处理系统。

## 3.3数据存储模块

### 3.3.1模块类图

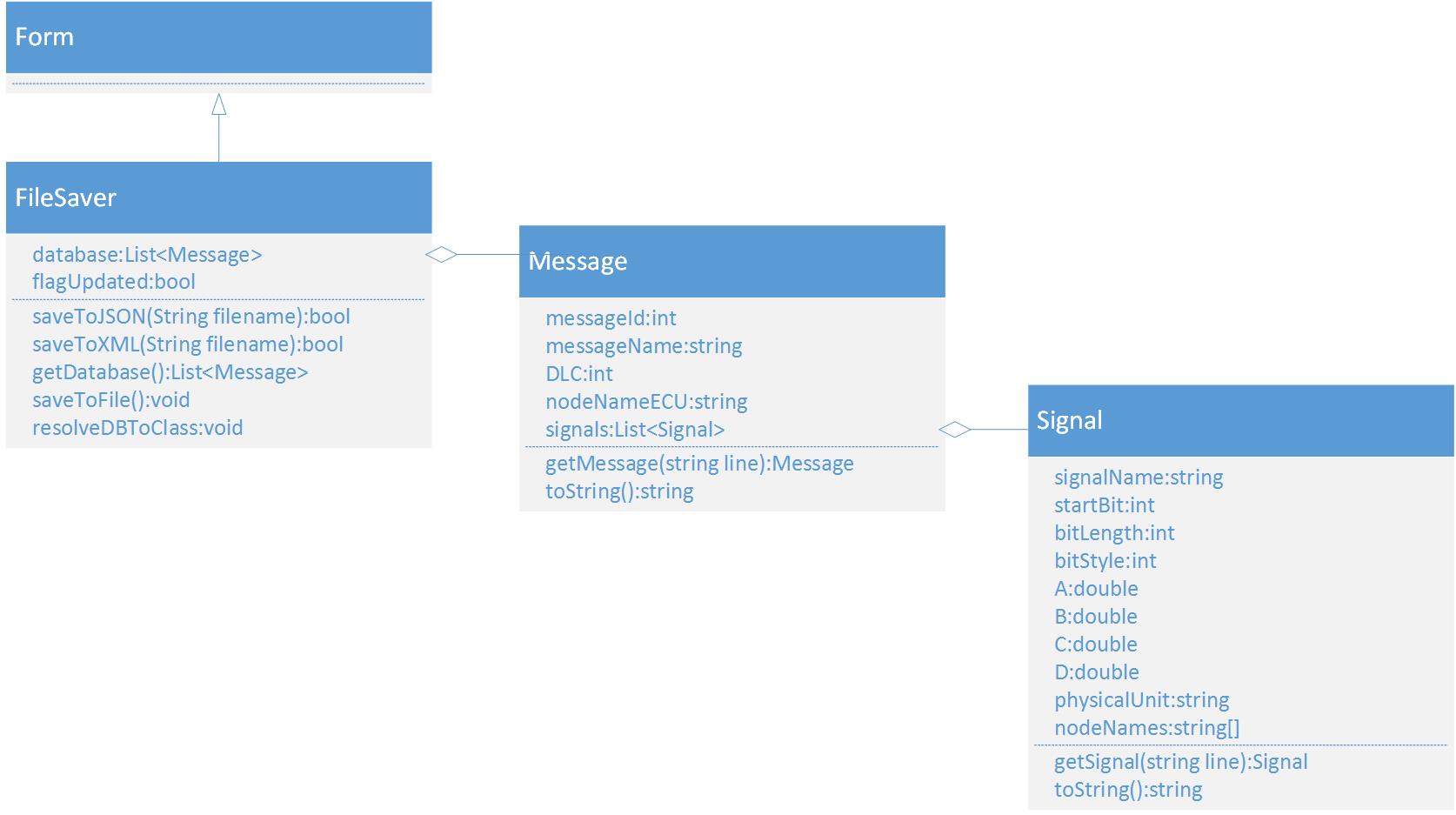


图 8 数据库解析存储相关类图

使用面向对象的编者原则，对于数据库解析之后，存储到类Signal类，由Signal类再组装到Message类中，进而使用列表List来存储整个数据库的对象集合。而对于文件存储的功能则都集中到FileSaver类中，其中有将dbc文件转存成对应的XML文件和Json文件，并可直接读取，再次加载到内存中。

## 3.5数据显示模块

### 3.5.1模块类图

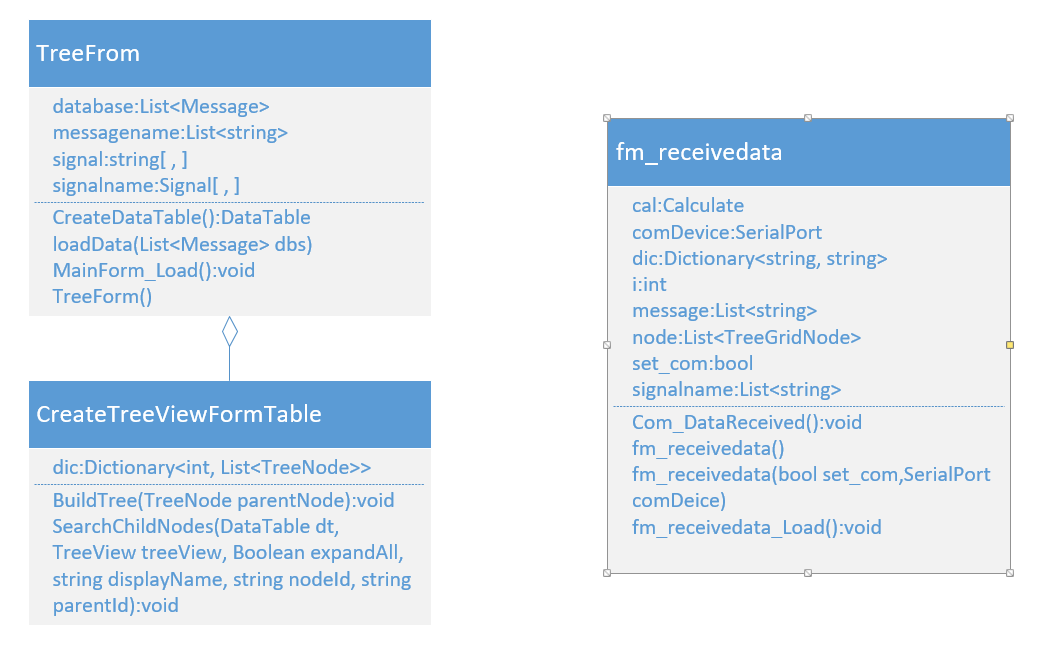


图 9 树形结构显示类图

左图中TreeForm是父类树结构，CreateTreeViewFormTable继承TreeForm类，对于树形结构进行进一步的包装，根据表格的形式生成最终Can Meaasage与Can Signal的树状结构显示出来。