中国科学技术大学

专业硕士学位论文

（专业学位类型）

硬件在环仿真测试系统的设计与实现

作者姓名：　胡闯闯

专业领域：　软件工程

校内导师：　华蓓　樊彦恩

企业导师：　睢佳彩

完成时间：　2021 年 07 月 xx 日

University of Science and Technology of China

A dissertation for master's degree

（Professional degree type）

Title

Author：

Speciality：　　Software Engineering

Supervisor：

Advisor：

Finished time:

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文,是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：_____ 签字日期：_____

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入《中国学位论文全文数据库》等有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

□公开 □保密（____年）

作者签名：_____ 签字日期：_____

摘 要

随着汽车大量涌入人们的生活中，人们对汽车的安全性、舒适性、经济性等各个方面性能的要求不断提高，使得电子控制单元（ECU）的控制功能越来越复杂。传统的测试必须等汽车样机开发出来之后才能进行实车测试，这样的流程容错率低、流程繁琐，且极端测试非常危险，给 ECU 的开发测试带来了困难。而 HIL 仿真测试可重现与 ECU 交互的物理系统，为 ECU 软件算法及硬件功能的测试和验证提供了良好的环境，提高测试效率，增加测试覆盖率和灵活性，以及降低测试成本。本文以硬件在环测试的思想为基础，自主研发硬件在环仿真测试系统，并进行了 ECU 的功能测试。

本文设计并实现了一个硬件在环仿真测试系统。系统的主要功能是实现对现有的 ECU 的功能和控制算法进行测试。系统主要包括三个核心部分：上位机监控软件、下位机实时仿真平台以及 IO 接口。其中上位机监控软件基于 Qt 实现，主要功能在于为用户提供监控和测试命令的可视化界面。下位机实时仿真平台分为内核空间和用户空间，内核空间用于模型任务的调度，IO 板卡的驱动实现，用于保证实时性；用户空间主要是运行模型任务，与上位机进行数据通信，以及以 IO 接口之间进行数据交换。IO 接口部分介于 ECU 与模型之间，提供与被测部件交互的模拟、数字和总线信号。

本文详细介绍了本硬件在环仿真测试系统的概要设计、详细设计与实现以及系统测试。目前，本系统的各个环节均已打通，整个系统的三大核心部分均已实现。测试人员可以在上位机监控软件对待测 ECU 进行基本的功能测试。为后续持续的开发打下坚实的基础。

关键词：电子控制单元；HIL 测试；实时性；IO 接口

ABSTRACT

With the influx of automobiles into people's lives, people's requirements for the safety, comfort, economy and other aspects of automobiles are constantly increasing, making the control functions of the electronic control unit (ECU) more and more complicated. The traditional test must wait for the development of the car prototype before the actual vehicle test can be carried out. Such a process has a low fault tolerance rate, a cumbersome process, and extreme testing is very dangerous, which brings difficulties to the development and testing of ECU. The HIL simulation test can reproduce the physical system that interacts with the ECU, providing a good environment for the testing and verification of ECU software algorithms and hardware functions, improving test efficiency, increasing test coverage and flexibility, and reducing test costs. Based on the idea of hardware-in-the-loop testing, this paper independently develops a hardware-in-the-loop simulation test system and conducts ECU functional testing.

This paper designs and implements a hardware-in-the-loop simulation test system. The main function of the system is to test the functions and control algorithms of the existing ECU. The system mainly includes three core parts: upper computer monitoring software, lower computer real-time simulation platform and IO interface. The host computer monitoring software is implemented based on Qt, and its main function is to provide users with a visual interface for monitoring and testing commands. The real-time simulation platform of the lower computer is divided into kernel space and user space. The kernel space is used for the scheduling of model tasks, and the driver implementation of the IO board is used to ensure real-time performance; the user space is mainly for running model tasks and communicating data with the upper computer. And to exchange data between IO interfaces. The IO interface part is between the ECU and the model, providing analog, digital and bus signals that interact with the component under test.

This article introduces in detail the outline design, detailed design and implementation, and system testing of the hardware-in-the-loop simulation test system. At present, all links of the system have been opened up, and the three

core parts of the entire system have been realized. Testers can perform basic functional tests on the ECU to be tested in the host computer monitoring software. Lay a solid foundation for subsequent continuous development.

Key Words: **electronic control unit; HIL test; real-time; IO interface**

目 录

目录

第 1 章 绪论

1.1 选题背景及研究意义

在当今社会，随着经济的发展、工业技术的进步、人类生活水平的大幅度提高，国际间愈加紧密的技术交流，汽车成为了人们生活工作中必不可少的交通工具。在国内国际市场需求和政府的相关扶持政策推动下，近几十年来汽车工业更是得到了飞速发展，特别是科技的超越性发展，给汽车工业的提升提供了不可缺少的技术支持，更进一步刺激到汽车制造业的扩张发展。

同时，随着汽车大量涌入人们的生活中，人们对汽车的安全性、舒适性、经济性等各个方面性能的要求不断提高，使得电子控制单元（ECU）的控制功能越来越复杂[1]。因此对电子控制单元进行测试迫在眉睫。然而传统的测试必须等汽车样机开发出来之后才能进行实车测试，这样的流程容错率低、流程繁琐、对技术人员的经验积累要求比较高、开发人员之间交流沟通难以进行以及成本高、周期长[2]等，而且还可能因为控制算法的错误导致设备的损坏甚至发生危险。这就给 ECU 的开发测试带来了困难。

基于模型的现代开发流程 —— V 模式开发流程[3]是现代汽车电子控制系统最重要的开发方法，其中提出采用硬件在环（HIL）仿真测试[4]的方法来对 ECU 进行测试和验证。硬件在环仿真测试系统是以实时处理器运行仿真模型来模拟受控对象的运行状态，通过 I/O 接口与被测的 ECU 连接，对被测 ECU 进行全方面的、系统的测试。从安全性、可行性和合理的成本上考虑，硬件在环仿真测试已经成为 ECU 开发流程中非常重要的一环，减少了实车路试的次数，缩短开发时间和降低成本的同时提高 ECU 的软件质量，降低汽车厂的风险[5]。

本文基于此需求，旨在开发一套测试工具 —— 硬件在环仿真测试系统，实现对现有 ECU 的功能测试。本系统主要由上位机、下位机、硬件板卡、ECU 等几个部分组成。上位机进行 UI 开发，用于从模型中实时读取反馈数据或修改模型相应参数。下位机主要包括实时软件系统框架的开发，用于上下位机之间的数据交互、与模型之间的信息交互；下位机与板卡相连接，进行数据交互；以及板卡与 ECU 之间的信息交互。在系统运行期间，用户可通过上位机实时监控或修改模型变量，达到对 ECU 测试的目的。

1.2 国内外研究现状

1.2.1 国外研究现状

国外硬件在环测试技术起步比较早，经过这些年的发展，目前技术比较成熟，已研究开发出了多种通用型的、商业化的硬件在环仿真测试系统，已经在汽车行业取得显著成就[6]。目前，比较知名的硬件在环测试系统包括德国的 dSPACE 公司开发的 dSPACE 硬件在环仿真测试系统、德国 ETAS 公司开发的 LabCar 硬件在环仿真测试系统、美国国家仪器（NI）开发的 Labview RT 硬件在环仿真测试系统[7]以及 vector 公司的基于 VT 的仿真测试系统。这几种系统均是为了 HIL 仿真测试所研发的软硬件设备，在汽车 HIL 仿真及测试领域有着非常重要的地位，也是各大汽车厂、零部件企业和研发机构在汽车 ECU 研发中应用的核心系统。下面分别对应用较多的两种系统进行介绍：

dSPACE 实时仿真系统[8]是由德国 dSPACE 公司开发的一套基于 MATLAB/Simulink 的控制系统在实时环境下的开发及测试工作平台，实现了和 MATLAB/Simulink 的无缝连接[9]。 dSPACE 实时系统由两大部分组成，一是硬件系统，二是软件环境。硬件系统主要包括实时处理器和 I/O 接口等。其中处理器具有强大的计算能力，能够运行计算复杂的模型，保证了系统的实时性要求；丰富的 I/O 支持则保证用户可以根据测试需要自行进行组合调用。软件环境主要由两大部分组成，一部分是实时代码的生成和下载，可以实现从 Simulink 模型到 dSPACE 实时硬件代码的自动下载。另一部分为测试软件，可以完成实验和调试工作。正是由于 dSPACE 的以上优越性，使得 dSPACE 得到了众多科研人员和工程技术人员的广泛关注，并得到迅速的推广应用。目前，dSPACE 已在汽车、航空航天、电力机车、机器人、驱动及工业控制等领域得到广泛应用[10]。但是一套完整的 dSPACE 的价格十分昂贵，加上大部分技术保密，因此大部分主机厂都不愿意选择 dSAPCE 来实现硬件在环仿真测试系统[11]。

Labcar 仿真测试系统是由德国 ETAS 公司开发的汽车 ECU 硬件在环仿真测试系统，其功能强大，二次开发和扩展能力极大[12]。Labcar 由硬件、软件和项目数据三方面组成[13]。其中软件部分主要用来创建汽车的 Simulink 模型，以及相关信号进行关联以驱动执行器；硬件部分用来产生控制单元的输入信号和控制信号。项目数据包含待运行相关车型的数据信息，包括排量大小、车轮尺寸、驱动方式等。因此，LabCar 可以看做是一部虚拟的但是能够接收并响应 ECU 控制信号的实验室汽车，可应用于 ECU 开发的各个环节，包括 ECU 模块的测试、软件功能的验证以及初期控制策略的开发等，有效缩短开发周期，提高 ECU 开发效率。但是 LabCar 硬件在环仿真测试系统也存在着不足，比如配套使用的 Vector 设备价格不菲，而且实现也比较复杂，还有高昂的技术服务费，这些都制

约着 LabCar 仿真测试系统的进一步发展。

1.2.2 国内研究现状

相比国外，国内汽车电子工业起步较晚，硬件在环测试技术也相对落后，基本处于被国外大公司垄断的地步。另外，相关产品种类较少且实际效果不太理想，如硬件 I/O 板卡等，所以目前国内尚没有自主研发的成熟的或商业化的硬件在环仿真测试系统。

目前，国内许多汽车企业及相关 ECU 供应商，如一汽、上汽、中汽研等。虽已成功应用硬件在环测试技术进行 ECU 的开发和测试，并取得相对满意的测试结果，但令人遗憾的是，这些公司应用产品的基础基本是基于国外成熟产品的应用，而并未在硬件在环技术研发上投入太多精力以及开展相关深入的研发。而国内的高校、研究院所侧重于硬件在环技术理论研究、实验研究以及简单应用。浙江大学基于 Simulink RT 软件平台和部分国产 IO 板卡，研发出了半自主的 HIL 仿真测试平台。其他如吉林大学、天津大学、北京理工大学[14]等高校也都对汽车硬件在环仿真测试系统进行了一些研究[15]。尽管国内对硬件在环测试技术进行了大量的研究，并取得一定的科研成果，但是国内目前的硬件在环仿真测试系统仍然存在着一些不足[16]，具体表现在以下几点：

1）系统实用性差。国内大部分的硬件在环仿真测试系统均是针对特定的车型和 ECU,往往是为了节省成本，根据项目实际需求而进行开发的，这样导致系统的实用性差，难以进行大规模的商业推广。其次，对于技术人员的专业知识要求较高，进一步制约了系统的迭代发展。

2）系统成本较高。由于国外存在已经成熟的硬件在环仿真测试系统及完善的配套服务，因此国内绝大多数的硬件在环测试系统都是基于国外成熟的产品进行二次开发，但是国外的这些产品往往价格高昂且后期维护费用较高，导致系统的成本较高。

此外，近十年来我国汽车电子产业取得了突飞猛进的发展，汽车 ECU 技术的研发取得突破式发展，对硬件在环仿真测试系统的需求越来越多，而现有的硬件在环仿真测试系统或在技术层面或在成本上都一定程度上限制了产业的发展，因此开发一套实用的硬件在环仿真测试系统是一项重要而又极富应用意义的课题。

1.3 本文的主要工作

本文内容主要包含上位机和下位机两大部分。其中上位机主要是 UI 界面的

制作，主要用于控制或读取模型数据信息，因此其需要解决一下几个关键技术：XML 文件的解析、变量树的生成、数据存储与处理、网络通信等。下位机主要包含两大方面：内核模块和用户空间模块。其中，内核模块作为上位机与用户空间模块之间的中转站，需要解决实时调度、网络通信、内存映射等。用户空间模块需要解决模型与板卡的数据交互、模型与内核的交互等。此外，还需要进行上下位机之间的通信协议的制定，以便上下位机之间保证正常通信。

整个项目过程中，本人的主要工作如下：

1）根据项目的实际需求，进行上位机 UI 的开发。

2）下位机内存模块的开发。

3）内核各功能模块的集成。

4）上下位机通信协议的制定，以及上下行数据通信的开发。

1.4 本文的组织结构

本文主要分为七个章节，每个章节的具体内容安排如下：

第一章：绪论：

第二章第二章：相关技术简介：

第三章第三章：需求分析：

第四章：概要设计：

第五章：详细系统设计与实现：

第六章：系统测试与分析：

第七章：总结与展望。在本章，笔者对该篇论文进行简单的总结，同时对系统存在的不足进行描述分析，并对 xxx 提出了未来的展望。

第 2 章 相关技术简介

2.1 硬件在环

硬件在环（Hardware-in-the-Loop，HIL ）是一种用于复杂设备控制器的开发与测试技术。通过 HIL 测试，机器或系统的物理部分被仿真器所代替，并被广泛运用于汽车控制器开发过程中。从安全性、可行性和合理的成本上考虑，硬件在环测试已经成为 ECU 开发流程中非常重要的一环，减少了实车路试的次数，缩短开发时间和降低成本的同时提高 ECU 的软件质量，降低汽车厂的风险。

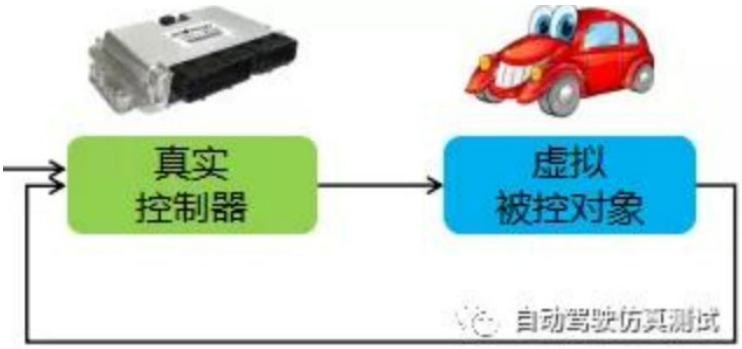
在新能源汽车这个全新的领域中，硬件在环测试对于三大核心电控系统:整车控制系统、BMS 电池管理系统、MCU 电机控制器是非常重要的。但其高精度的实时性要求、大电压大电流的安全性、信号接口的特殊属性、以及系统的可扩展性都使得传统汽车电控系统的 HiL 硬件在环测试系统无法解决。

硬件在环测试系统主要由三部分组成：实时处理器、I/O 接口和操作界面，其中实时处理器是 HIL 测试系统的核心。

实时处理器主要用于运行仿真模型以及信号处理。在实时硬件上需要运行实时操作系统，以保证模拟的实时性；被控对象的行为模型运行在实时操作系统之上；另外根据各执行机构、传感器的特性，需要建立接口模型以达到逼真的仿真效果。

操作界面与实时处理器通信，提供测试命令和可视化。在大多数情况下，这个部件也提供配置管理、测试自动化、分析和报告任务。

I/O 接口是与被测部件交互的模拟，数字和总线信号。可以用它们来产生激励信号，获取用于记录和分析的数据，并提供被测的电子控制单元 (ECU)与模型仿真的虚拟环境之间的传感器/执行器交互。



2.2 Model/View 框架

Model-View-Controller(MVC)， 是从 Smalltalk 发展而来的一种设计模式，常被用于构建用户界面。MVC 由三种对象组成，Model 负责维护数据（如管理数据库），View 负责显示与用户交互（如各种界面），Controller 将控制业务逻辑。如果把 View 与 Controller 结合在一起，结果就是 Model/View 框架。Model/View 框架依然是把数据存储与数据表示进行了分离，与 MVC 都基于同样的思想，但更简单。数据存储与数据显示的分离使得在几个不同的 View 上显示同一个数据成为可能，也可以重新实现新的 View,而不必改变底层的数据结构。

Model 里面并不真正存储数据,只是负责与数据源通讯，并提供接口给结构中的别的组件使用。通讯的实质依赖于数据源的类型与 Model 实现的方式。

View 从 Model 获取模型索引,模型索引是数据项的引用。通过把模型索引提供给 Model,View 可以从数据源中获取数据。

Models、Views 之间通过信号-槽机制来进行通讯：从 Model 发出的信号通知 View 关于数据源中的数据发生的改变。从 View 发出的信号提供了有关被显示的数据项与用户交互的信息 Model/View 框架结构如图 2-2 所示：

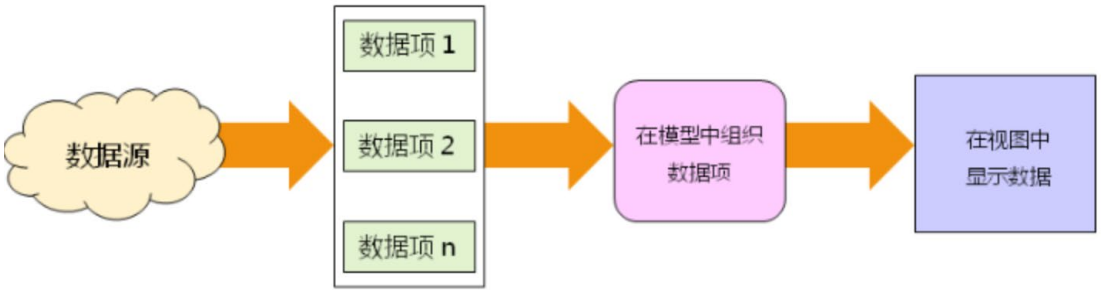


图 2-2 Model/View 框架结构图

在本系统中，上位机 UI 需要接收处理来自下位机发送的数据，并展示在不同窗口中，因此 Model/View 的框架很适合上位机 UI 的开发。其中下位机发送的数据经解析之后作为数据源并存入 QStandarItem 数据项。不同的窗口的视图展示通过在模型中组织数据项而显示数据。

2.3 TCP 网络通信

本系统上下位机之间采用基于 TCP 协议的 Socket 通信。基于 TCP 协议的 Socket 通信主要是 C/S 模式，即服务器(S)、客户端(C)模式。服务器模式创建一个服务程序， 等待客户端用户的连接， 接收到用户的连接请求后， 根据用户

的请求进行处理；客户端模式则根据目的服务器的地址和端口进行连接， 向服务器发送请求并对服务器的响应进行数据处理。通信流程如图 2-3 所示：

1）服务器模式的程序设计流程

1.套接字初始化过程中， 根据用户对套接字的需求来确定套接字的选项。这个过程中的函数为 socket(), 它按照用户定义的网络类型、协议类型和具体的协议标号等参数来定义。系统根据用户的需求生成一个套接字文件描述符供用户使用。

2.套接字与端口的绑定过程中， 将套接字与一个地址结构进行绑定。绑定之后， 在进行网络程序设计的时候， 套接字所代表的 IP 地址和端口地址及协议类型等参数按照绑定值进行操作。

3.由于一个服务器需要满足多个客户端的连接请求， 而服务器在某个时间仅能处理有限个数的客户端连接请求， 所以服务器需要设置服务端排队队列的长度。服务器侦听连接会设置这个参数， 限制客户端中等待服务器处理连接请求的队列长度。

4.在客户端发送连接请求之后， 服务器需要接收客户端的连接， 然后才能进行其他的处理。

5.在服务器接收客户端请求之后， 可以从套接字文件描述符中读取数据或者向文件描述符发送数据。接收数据后服务器按照定义的规则对数据进行处理，并将结果发送给客户端。

6.当服务器处理完数据， 要结束与客户端的通信过程的时候， 需要关闭套接字连接。

2）客户端模式的程序设计流程

客户端模式主要分为套接字初始化 socket()， 连接服务器 connect() ，写网络数据 read() 、write() 并进行数据处理和最后的套接字关闭 close() 过程。客户端程序设计模式流程与服务器端的处理模式流程类似，二者的不同之处是客户端在套接字初始化之后可以不进行地址绑定， 而是直接连接服务器端。

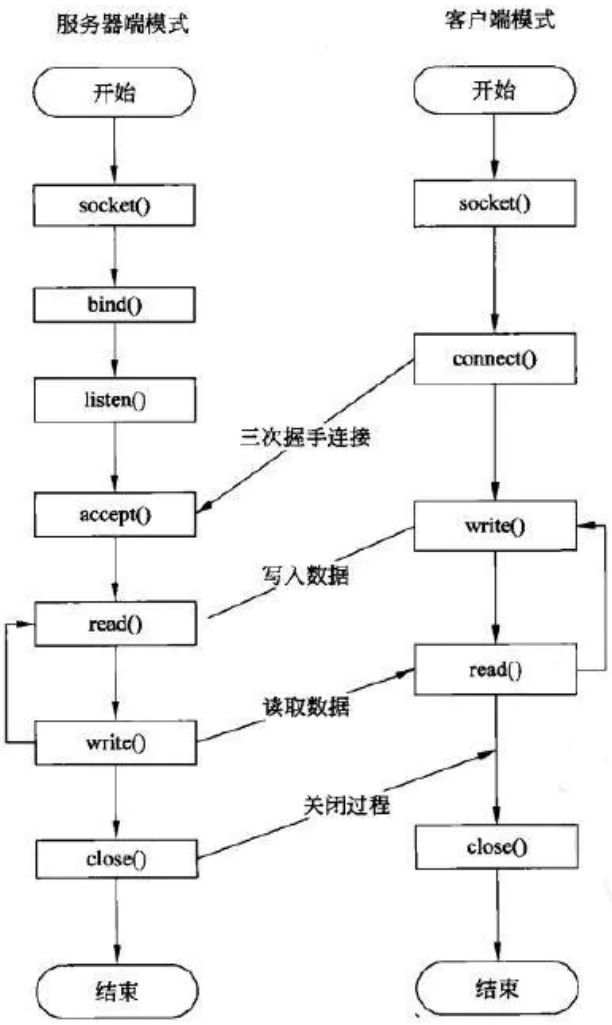
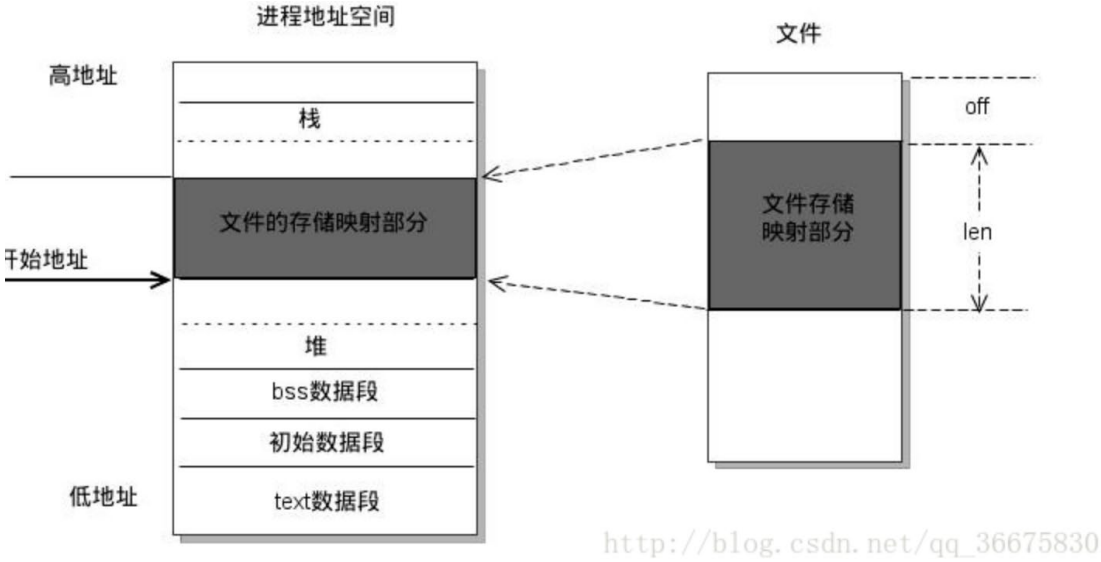


图 2-3 基于 TCP 的 Socket 通信流程图

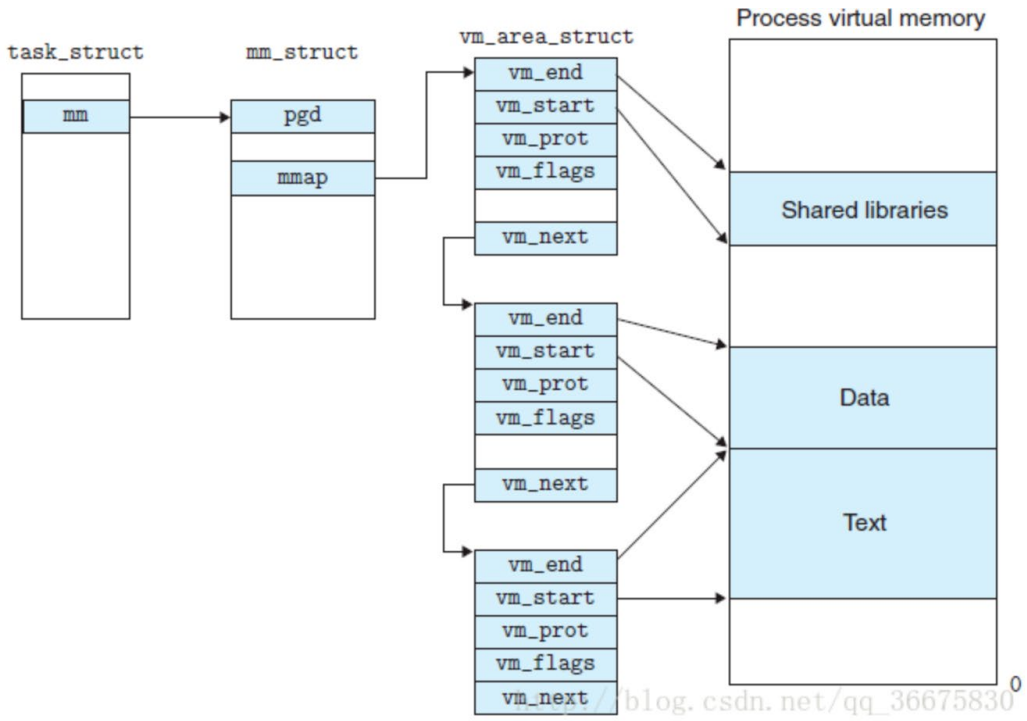
2.4 Mmap 原理

mmap 是一种内存映射文件的方法，即将一个文件或者其它对象映射到进程的地址空间，实现文件磁盘地址和进程虚拟地址空间中一段虚拟地址的一一对映关系。实现这样的映射关系后，进程就可以采用指针的方式读写操作这一段内存，而系统会自动回写脏页面到对应的文件磁盘上，即完成了对文件的操作而不必再调用 **read,write** 等系统调用函数。相反，内核空间对这段区域的修改也直接反映用户空间，从而可以实现不同进程间的文件共享。如下图所示：



由上图可以看出，进程的虚拟地址空间，由多个虚拟内存区域构成。虚拟内存区域是进程的虚拟地址空间中的一个同质区间，即具有同样特性的连续地址范围。上图中所示的 text 数据段（代码段）、初始数据段、BSS 数据段、堆、栈和内存映射，都是一个独立的虚拟内存区域。而为内存映射服务的地址空间处在堆栈之间的空余部分。

linux 内核使用 vm_area_struct 结构来表示一个独立的虚拟内存区域，由于每个不同质的虚拟内存区域功能和内部机制都不同，因此一个进程使用多个 vm_area_struct 结构来分别表示不同类型的虚拟内存区域。各个 vm_area_struct 结构使用链表或者树形结构链接，方便进程快速访问，如下图所示：



vm_area_struct 结构中包含区域起始和终止地址以及其他相关信息，同时也包含一个 vm_ops 指针，其内部可引出所有针对这个区域可以使用的系统调用函数。这样，进程对某一虚拟内存区域的任何操作需要用要的信息，都可以从 vm_area_struct 中获得。mmap 函数就是要创建一个新的 vm_area_struct 结构，并将其与文件的物理磁盘地址相连。

mmap 内存映射的实现过程，总的来说可以分为三个阶段：

（一）进程启动映射过程，并在虚拟地址空间中为映射创建虚拟映射区域

1、进程在用户空间调用库函数 mmap，原型：void *mmap(void *start, size_t length, int prot, int flags, int fd, off_t offset);

2、在当前进程的虚拟地址空间中，寻找一段空闲的满足要求的连续的虚拟地址

3、为此虚拟区分配一个 vm_area_struct 结构，接着对这个结构的各个域进行了初始化

4、将新建的虚拟区结构（vm_area_struct）插入进程的虚拟地址区域链表或树中

（二）调用内核空间的系统调用函数 mmap（不同于用户空间函数），实现文件物理地址和进程虚拟地址的一一映射关系

5、为映射分配了新的虚拟地址区域后，通过待映射的文件指针，在文件描述符表中找到对应的文件描述符，通过文件描述符，链接到内核“已打开文件集”中该文件的文件结构体（struct file），每个文件结构体维护着和这个已打开文件

相关各项信息。

6、通过该文件的文件结构体，链接到 file_operations 模块，调用内核函数 mmap，其原型为：int mmap(struct file *filp, struct vm_area_struct *vma)，不同于用户空间库函数。

7、内核 mmap 函数通过虚拟文件系统 inode 模块定位到文件磁盘物理地址。

8、通过 remap_pfn_range 函数建立页表，即实现了文件地址和虚拟地址区域的映射关系。此时，这片虚拟地址并没有任何数据关联到主存中。

（三）进程发起对这片映射空间的访问，引发缺页异常，实现文件内容到物理内存（主存）的拷贝

注：前两个阶段仅在于创建虚拟区间并完成地址映射，但是并没有将任何文件数据的拷贝至主存。真正的文件读取是当进程发起读或写操作时。

9、进程的读或写操作访问虚拟地址空间这一段映射地址，通过查询页表，发现这一段地址并不在物理页面上。因为目前只建立了地址映射，真正的硬盘数据还没有拷贝到内存中，因此引发缺页异常。

10、缺页异常进行一系列判断，确定无非法操作后，内核发起请求调页过程。

11、调页过程先在交换缓存空间（swap cache）中寻找需要访问的内存页，如果没有则调用 nopage 函数把所缺的页从磁盘装入到主存中。

12、之后进程即可对这片主存进行读或者写的操作，如果写操作改变了其内容，一定时间后系统会自动回写脏页面到对应磁盘地址，也即完成了写入到文件的过程。

注：修改过的脏页面并不会立即更新回文件中，而是有一段时间的延迟，可以调用 msync()来强制同步, 这样所写的内容就能立即保存到文件里了。

参考：https://blog.csdn.net/qq_36675830/article/details/79283113

2.5 本章小结

第 3 章 需求分析

3.1 系统概述

通常情况下，对 ECU 进行测试需要搭建专业测试台架，或者直接进行实车测试，而采用硬件在环的方法，执行单元并不需要是实物，取而代之的是建立相应的模型。通过仿真测试系统，在测试 ECU 的功能时，测试人员通过上位机的 UI 界面，可以实时修改模型的参数，方便快捷，同时也可以实时读取模型反馈的变量信息，然后进行观测，并验证 ECU 相应功能。

综上所述，该仿真测试系统的设计要求实现的功能包括有：测试人员可以通过上位机的 UI 控制仿真测试系统，能够在 UI 界面上方便快捷的对各种参数进行修改；实时读取模型反馈的变量数据进行存储和分析；下位机实时运行模型，并向相应板卡发送信号，进而激励 ECU 工作使其发出控制信号，转而修改模型的参数，进而测试 ECU 功能。

3.2 功能性需求 （下可合并）

3.2.1 上位机 UI

上位机 UI 基于 QT 进行开发，主要服务对象是测试者，测试者在 UI 界面可以控制整个实时仿真测试系统。通过 socket 远程连接到下位机，进行 socket 通信；monitor 窗口和 scope 窗口可拖拽变量且按自行定制的协议向下位机发送变量信息、控制命令，或者接收、解析、处理下位机上传的数据；同时支持一键恢复功能。因此上位机主要包含五个子功能模块：模型变量树模块、socket 通信模块、monitor 模块、scope 模块、一键恢复模块。用户用例图如图 1 所示。

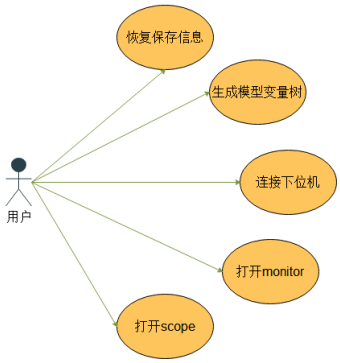


图 1 上位机用例图

3.2.1.1 socket 通信功能模块

本系统的目的是对 ECU 进行功能测试，用户需要在本地 UI 界面与下位机实时仿真测试模块进行通信，以达到测试 ECU 功能的目的。同时为了保证数据的可靠性，需要建立基于 TCP 的 socket 通信连接，因此上位机需要搭建 TCP 客户端模块。

用户通过该模块可以选择下位机并建立通信，控制下位机实时仿真测试模块的启停，同时接收、解析处理下位机发送的数据。该模块需求内容以及说明如表 3-1 所示：

表 3-1 TCP 客户端模块需求内容及说明

需求项	说明
远程 IP 地址及端口输入	用户进行下位机连接
启动模型	启动下位机仿真模块的运行
暂停模型	暂停下位机仿真模块的运行
停止模型	停止下位机仿真模块的运行

3.2.1.2 模型变量树模块

该模块主要用于测试者操作变量，要求有如下功能：

1）解析模型提供的 XML 文件中的变量及其属性，以便保持上下位机变量一致性。

2）将所有解析出来的变量按照树形结构的形式在 UI 子窗口中展示。

3）子窗口中的变量可拖拽到 UI 其它子窗口且拖拽时必须保证变量原有属性不变。

3.2.1.4 monitor 功能模块

该模块用于测试者实时观测下位机上传的数据，以及测试者更改模型变量的值。本功能模块要求有如下功能：

1）允许变量从其它窗口拖入并放置到本窗口。

2）允许变量从本窗口删除。

3）变量的 value 定期变化。

4）测试者可以对参数类型变量的 value 进行修改，且变量处于编辑状态时 value 不可变。

3.2.1.5 scope 功能模块

该模块用于测试者查看变量的波形信息。本功能模块功能如下：

1）允许变量从其他窗口拖入并放置到本窗口。

2）允许变量在本窗口删除。

3）变量的 value 定期变化。

4）窗口中变量可勾选，且绘制勾选变量的波形，且允许显示多条波形。

5）可修改变量波形的颜色。

3.2.1.6 一键恢复模块（可合并）

应用程序创建的各个窗口的位置、尺寸、窗口内容，导入的模型文件的路径，用户的配置等信息，保存为一个独立的文件。下次启动程序后，导入该文件，即可恢复上一次的结果。

3.2.2 下位机实时仿真模块

下位机实时仿真模块基于 linux 进行开发。该模块负责与上位机建立 TCP 连接，用于接收上位机发送的控制命令并写入到指定内存区域，或者从指定内存区域读取数据并发送给上位机；该模块能周期性调度模型运行，在每个周期内，读取 I/O 卡接收的数据并写入模型，读取 can 卡接收的数据并写入模型，从指定内存区域读取上位机下发的命令并写入模型。在模型计算之后，将数据写入到 can 卡、写入指定区域发送给上位机、写入 IO 卡。因此主要包含五个子功能模块：socket 通信模块、内存模块、模型任务运行模块、can 卡驱动模块、I/O 板卡驱动模块。其中后两个模块不在本课题工作内容之中。

3.2.2.1 socket 通信模块

该模块用于和上位机建立 socket 通信连接，向上位机发送数据或者接收并存储上位机发送的数据。该功能模块要求如下：

1）与上位机建立远程连接，进行 socket 通信；

2）接收上位机发送的数据，并写入到指定内存区域。

3）根据系统调度，向上位机发送数据。

3.2.2.2 内存模块

该模块主要用于存储数据，便于上下位机进行数据通信。该模块功能如下：

1）开辟内存区域，用于上下位机交换数据。

2）允许 socket 通信模块和用户态程序对所开辟的内存进行读写操作。

3.2.2.3 信号配置与初始化模块

该模块主要读取信号的硬件配置 EXCEL 表，获取信号配置信息，包块板卡槽位、通道、信号类型，初始配置等硬件信息。根据获取结果，完成信号配置和硬件初始化工作。

3.2.2.4 模型任务运行模块

该模块主要负责模型的调度运行以及数据的读写。该模块功能如下：

1）模型计算。

2）读取 can 卡和 I/O 卡的数据并写入模型。

3）从指定内存区域读取上位机发送的数据并进行处理。

4）模型计算结束后，将相应数据写到指定内存区域。

5）将相应数据写入到 CAN 卡和 I/O 卡。

3.3 非功能性需求

3.3.1 模型任务性能需求

在本系统中，下位机运行的模型每个周期的步长在 1ms 之内。在每个周期内，完成对板卡的读写工作、模型的读写工作以及模型计算。若步长超过 1ms，将影响到后续周期的运行，导致系统仿真失去原本的目的。

3.3.2 系统健壮性和稳定性

健壮性也称为系统的坚固性或坚实性，表示系统在不正常输入或不正常外部环境下仍能表现正常的程度。在本系统中，测试者修改模型变量时，需要对测试

者输入的数据进行合理的判断或者给出错误提示信息。因此本系统需要具有较好的健壮性。同时，本系统可能长时间运行，因此要保证系统具有稳定性，避免意外中断或提前结束等情况发生。

3.5 本章小结

第 4 章 概要设计

4.1 系统概述

本仿真测试系统主要包含上位机 UI、下位机实时仿真模块、I/O 板卡、CAN 卡以及待测 ECU。其中，考虑到系统实时性，下位机实时仿真模块又细分为内核模块和用户空间模块。本章节主要介绍整个系统的整体结构以及各个模块的功能设计。

4.2 系统总体结构设计

本仿真测试系统的总体架构如图 2 所示。上下位机之间通过 TCP 进行数据通信。上位机包含 socket 通信模块、monitor 模块、scope 模块、模型变量树模块、

一键恢复模块。下位机包含实时仿真模块和硬件板卡，其中实时仿真模块又细分为内核模块和用户空间模块，硬件板卡主要是 I/O 板卡和 CAN 卡。内核模块主要包含 socket 通信模块、内存模块、can 卡驱动模块、I/O 板卡驱动模块；用户空间模块包含模型任务运行模块，负责进行模型信息配置、模型计算、将计算后的数据一部分写入指定内存区域上传到上位机，另一部分写入到相应的 can 卡和 I/O 卡。内核空间与用户空间通过共享内存实现数据交换，硬件板卡与待测 ECU 之间通过线束进行连接。备注：该部分整体架构由华为项目组设计完成，论文继承该成果。

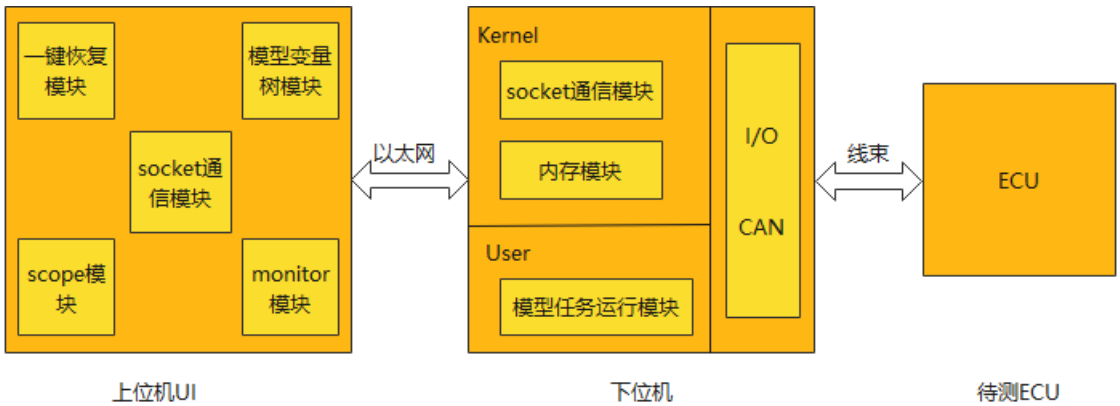


图 2 系统架构图

4.2.1 上位机 UI 开发架构

4.2.1.1 UI 整体架构设计

上位机 UI 基于 QT 进行开发设计。其中，UI 一方面与用户进行交互，提供良好的用户界面和简单便捷的操作，另一方面与下位机进行数据交换。按照功能需求设计，上位机 UI 模块架构设计如图 3 所示，socket 通信模块负责与下位机进行通信，进行数据收发；模型变量树模块中的变量可以拖拽到 monitor 模块和 scope 模块；monitor 模块和 scope 模块检测到变量拖入后，通过 socket 通信模块向下位机发送数据，同时展示变量的相关信息；一键恢复模块负责程序启动后是否恢复到上一次保存之后的界面。

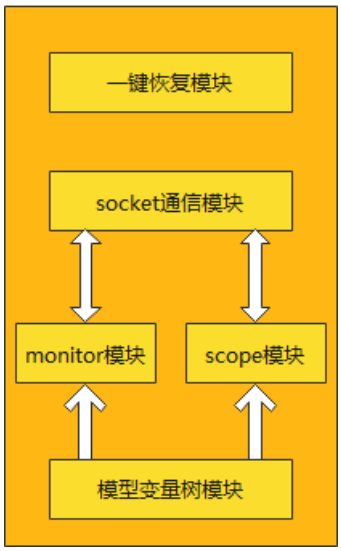


图 3 上位机架构图

4.2.1.2 Model/View 框架设计

新加 model 设计：数据项组织与基本功能拆分

功能视图

{

上合

}

根据需求分析，用户可以将变量树界面上的变量拖拽到其他页面上，因此需要自定义 Model 类 。考虑到变量为树形结构，因此选用 Qt 提供的 QStandardItemModel 类作为父类进行自定义 Model 设计。该类用于管理复杂的树型结构数据项，每项都可以包含任意数据。通过重

写该类部分 API，实现变量拖拽功能。而变量树模块、monitor 模块、scope 模块通过 Qt 提供的界面类完成模型数据项的组织，进行数据视图显示。

4.2.1.3 UI 界面设计（待删除）

4.2.1.4 信号与槽设计（待删除）

4.2.2 下位机实时仿真模块架构

下位机实时仿真模块基于 linux 系统进行开发设计。该模块一方面通过 socket 通信模块与上位机进行通信，一方面通过 IO 卡和 CAN 卡与 ECU 通信。按照功能需求设计，下位机设计架构如图 4 所示。用户空间主要负责模型的运行。用户空间和内核空间通过共享内存进行数据交换。备注：该部分整体架构由华为项目组设计完成，论文继承该成果。

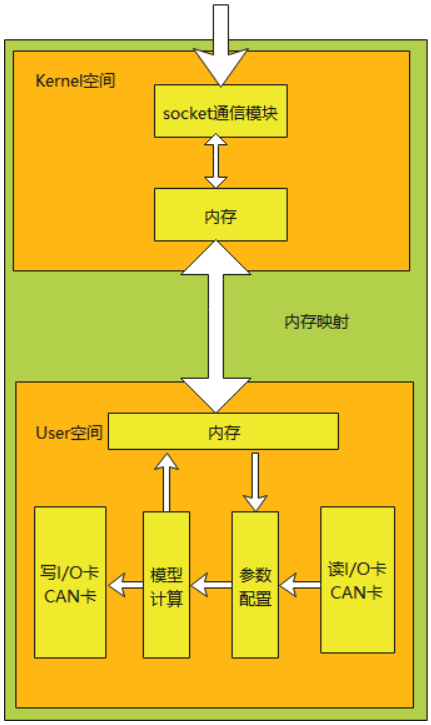


图 4 下位机架构图

4.3 上位机功能模块设计

本章节主要详细介绍上位机 UI 各个模块的结构设计或进行算法描述。

4.3.1 模型变量树模块

用户首先加载 XML 文件，通过 QT 提供的方法将 XML 文件中的所有变量及其属性解析并保存。根据解析之后的变量信息，构造 N 叉树。其次通过重写 QT 自身提供的标准模型视图类实现变量拖放功能。最终通过 N 叉树给变量的树形视图添加条目，完成该模块的功能需求。

构造 N 叉树的算法思路：树的每个节点应该包含两个元素，第一是用当前节点的名称来标记节点，第二是一个字典，用来保存它的所有子节点，然后依次遍历每一个变量的路径信息，将每个节点挂接到树上。若为叶子节点，则该节点保存变量的属性信息。

算法正确性验证：对新建的 N 叉树进行深度优先遍历，记录所有从根节点到叶子节点的路径，将每一条路径信息与变量经过处理后的路径信息进行对照，即可验证之。

构建变量的树形视图的解决思路：构建变量树形视图的过程本质上也是在构建一棵 N 叉树，因此由 N 叉树构建树形视图有两种思路，通过深度优先遍历或广度优先遍历生成变量树形模型，但考虑到界面最终呈现给用户的变量按照字典序排列，因此采用广度优先遍历的方式最终构建完成变量的树形视图。其中，在构建树形视图每个条目的子节点之前，需要对该条目对应节点的孩子节点进行字典序排列。

4.3.2 monitor 模块

本模块主要用于用户修改模型变量信息，向下位机模型发送控制信号。其次还用于显示变量的属性信息。该模块工作流程图如图 6 所示。

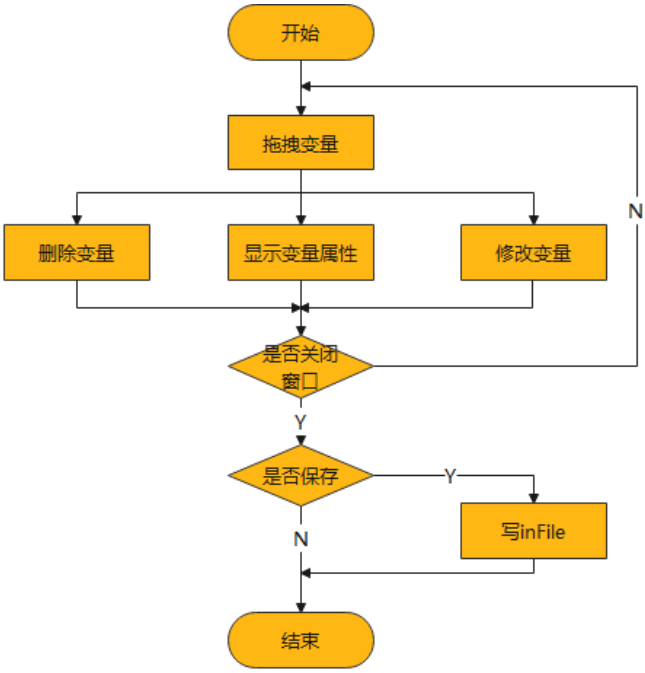


图 6 monitor 模块流程图

4.3.3 scope 模块

本模块主要用于绘制勾选变量的波形绘制，便于用户观测模型反馈的信息。其次变量的 value 定期进行显示更新。该模块工作流程图如图 7 所示：

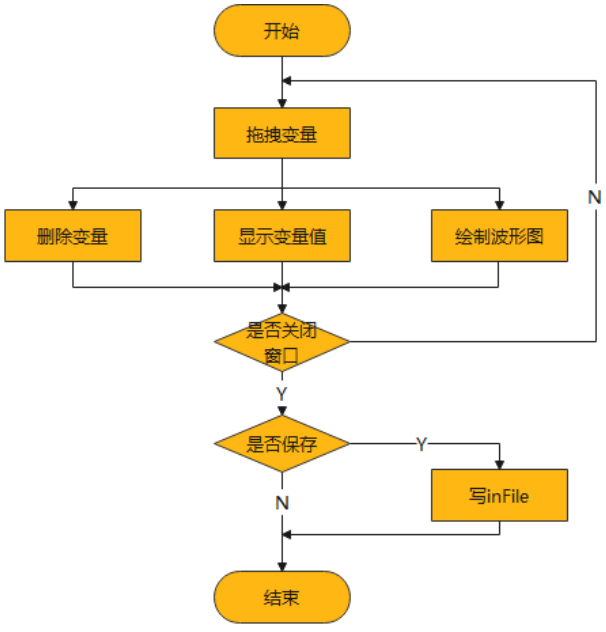


图 7 scope 模块波形图

4.3.4 恢复模块（上合 4.2.1.2）

本模块即读取 iniFile 文件，恢复上一次 UI 关闭前保存的工作内容。

4.4 下位机功能模块设计

本章节主要介绍下位机各个子模块的结构设计。备注：本部分内容由本人和华为项目组共同完成。

4.4.1 共享内存模块

Linux 系统通过内存映像机制来提供用户程序对内存直接访问的能力。内存映像的意思是把内核中特定部分的内存空间映射到用户程序的内存空间去。也就是说，用户空间和内核空间共享一块相同的内存。本模块即通过 mmap 进行基于文件的映射，实现内核空间与用户空间之间共享内存。设计思路如下：

用户空间通过读取文件获取三块物理地址，分别调用 mmap 启动文件映射过程，并在虚拟地址空间为映射创建虚拟映射区域。

内核模块则需要实现待映射文件 file_operations 结构体中的 mmap 函数（不同于用户空间函数），实现文件物理地址与进程虚拟地址的一一映射关系。之后即可对该映射区域进行读写操作。

4.4.2 信号配置与初始化模块

本模块作为下位机用户程序的基础模块之一，主要是进行综合 IO 板卡的初始化工作，即 IO 信号的配置与初始化。根据实际的物理布线情况以及板卡负载能力，配置每个 IO 信号所属的板卡槽位、通道信息。同时根据不同的信号类型，进行信号初始化，如初始化参考电压、信号初始值、配置模式等。

在本模块中，首先编写接口函数，解析 Excel 表中的信号配置信息，实现从 Excel 表格到下位机程序所需数据结构的转换，完成 IO 信号的获取工作。

其次，根据获取的信号类型，打开相应的板卡设备文件，将信号与相应板卡的句柄进行绑定。

最后，根据获取的信号配置信息，进行信号的初始配置工作，完成综合 IO 板卡的初始化工作。

4.4.3 模型任务运行模块

根据需求分析，在本模块中，任务运行前，首先进行初始化工作，包括模型初始化、内存初始化、注册模型任务等，然后等待任务管理模块发送信号。当任务接收到信号后，开始一个周期的运行。模型任务在每个周期内的执行包括三个阶段：

模型读入阶段：读 CAN 卡和 IO 卡，获得相应的信号数据信息并写入模型；读下行内存区域，若有上位机发送的数据，则根据指令类型进行相应处理。如修

改上传信号，修改模型信号值等。

模型运行阶段：模型根据写入的信号值进行仿真计算；

模型写出阶段：模型计算结束之后，将相应信号信息写入到 CAN 卡和 IO 卡，经板卡处理后发向待测 ECU；根据用户订阅的信号表，从模型读取信号值并写入上行内存中。该模块工作流程图如图 9 所示，其中 begin 指模型每个步长开始执行，end 指模型结束。

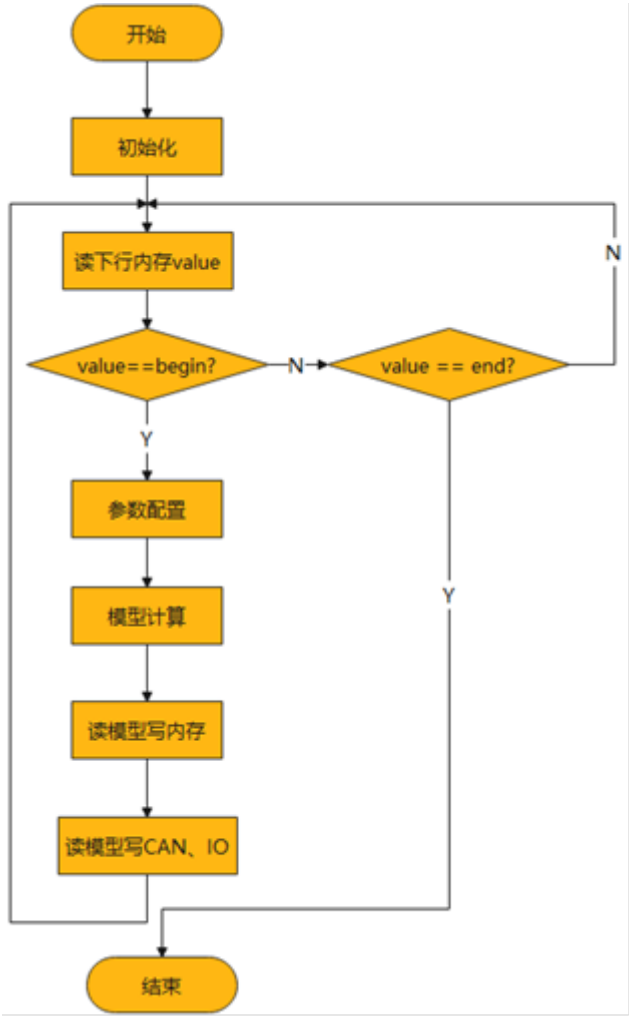


图 9 模型任务流程图

4.5 上下位机通信设计

4.5.1 通信协议设计

1）包格式

包头（10 字节）	数据包

2）包头格式

类型	字节数
控制字段	1 字节
数据类型	1 字节
变量 ID	4 字节
数据包大小	4 字节

其中，控制字段类别包括添加变量、修改变量、删除变量、启动模型、停止模型。数据类型即下发的数据类型。数据包大小即所发送数据包的字节数。

4.5.2 TCP 客户端设计

上位机 socket 通信模块主要是创建 TCP 客户端，通过配置下位机服务器端的 IP 地址和端口号，与下位机建立 TCP 连接，之后实现与下位机之间数据通信该模块工作流程如图 5 所示：

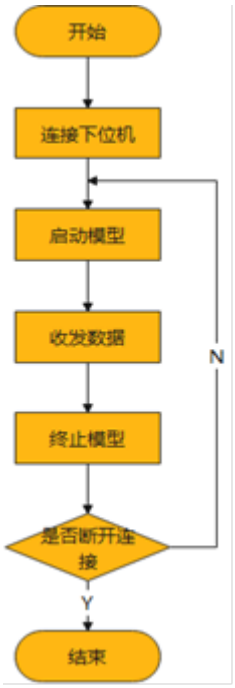


图 5 TCP 客户端流程图

4.5.3 TCP 服务端设计

下位机 socket 通信模块主要是建立 TCP 服务器端，当接收到客户端的连接

请求后，建立连接，创建收发进程，并进入休眠状态。定时器时间到，唤醒数据收发进程。接收上位机下发的控制命令或数据信息写入到下行内存供模型使用；从上行内存读取数据并上传给上位机。其中，下行内存指上位机下发数据的存储区域，上行内存指下位机上传数据的存储区域。该模块功能流程图如图 8 所示：

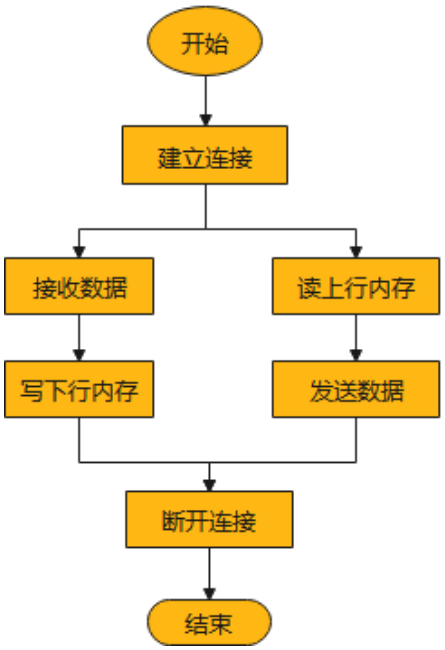


图 8 下位机 socket 模块流程图

4.6 本章小结

第 5 章 详细设计与实现

本章节是在第四章概要设计的基础上，给出本课题所实现的硬件在环仿真测试系统的详细设计与实现，主要内容包括有前端部分核心类的设计与实现、上位机 UI 各个功能模块的设计与实现以及下位机部分功能模块的设计与实现。

5.1 上位机 UI 功能模块的设计与实现

5.1.1 Model 类的设计与实现

QT 提供了类 QStandardItemModel，该类是 QAbstractItemModel 的派生类，用于在 Model/View 架构中存储自定义数据的通用模型，可以用于在任何支持 QAbstractItemModel 接口的 view（例如 QListView、QTableView 和 QTreeView，以及自定义视图）中作为数据存储。本 Model 类继承于类 QStandardItemModel，需要实现变量拖拽功能，其中，拖放操作分为拖动(Drag)和放置(Drop)两种操作，当拖动时，需要把拖动的数据进行存储，存储数据这一步骤称为编码，当执行放下操作时，需要对放置数据进行处理，若需要使用这些数据，则需要把存储的数据读取出来(即解码)，然后进行处理；当然，若不需要这些放置数据，也可以直接丢弃。下面为具体实现逻辑：

1）重新实现 flags()函数，设置合适的标志用于指示哪些数据项可以被拖动，哪些可以接受放置。

2）重新实现 mimeData()函数，将拖动的变量属性信息通过编码存储在该函数返回的 QMimeData 对象中。

3）重新实现 dropMimeData()函数来处理放置数据，此时需要对放置的数据进行解码（即读出 QMimeData 对象存储的数据的内容），并将其插入模型的底层数据结构中，同时还要发出相应的信号。

4）提供 ItemIsDrag()方法，保证变量拖拽到同一个窗口时，能且仅能拖拽一次。除非变量在窗口被删除。

5.1.2 TCP 客户端设计与实现

根据第四章概要设计，在该模块中，基本功能即与下位机建立 Socket 通信，具体原理见第二章。核心功能是进行数据收发操作。以下部分详细介绍收发实现：

所谓数据发送，包含两种含义：其一，用户在 TCP 客户端界面发送指令，该指令面向模型任务，包括启动、暂停以及终止模型；其二，用户在 Monitor 界面发送数据指令，包括添加订阅信号、删除订阅信号、修改订阅信号值。指令数据结构定义如下：

```
struct Head{
    unsigned char cmdType;
    unsigned char dataType;
    unsigned int vr;
    unsigned int dataNum;
};
```

其中，cmdType 字段用于标识指令类型；dataType 字段用于标识数据类型；vr 字段即信号的 id，仅当用户修改信号值时有效；dataNum 即要发送的数据个数。

数据接收，即用户在上位机完成信号订阅之后，下位机每个周期都会按照一定的数据格式发送数据。由于 Qt 客户端接收到下位机发送的数据并非刚好是一个周期的数据量，因此需要设置一个缓冲 buffer。当接收到的数据不足约定的报文大小后，等待；否则将收到的数据按照一定的帧格式进行数据解析，之后分别向 Monitor 模块和 Scope 模块发送信号。该模块流程图如下：

5.1.3 变量树模块设计与实现

变量树模块是上位机 UI 的基础模块，用户对运行在下位机上的模型数据的读写操作均在本模块的基础上进行。本模块的作用是向用户展示下位机已经完成配置和初始化的信号。

在该模块中，用户首先需要解析 XML 文件（该文件囊括了模型代码中的参数变量，输入输出变量）获得变量属性信息，以保持与下位机模型变量的一致性。之后根据变量相关信息构造 N 叉树，进而根据 N 叉树在 Model 类中组织数据项，达到在窗口显示变量树的目的。该模块类图如图 5-2 所示：

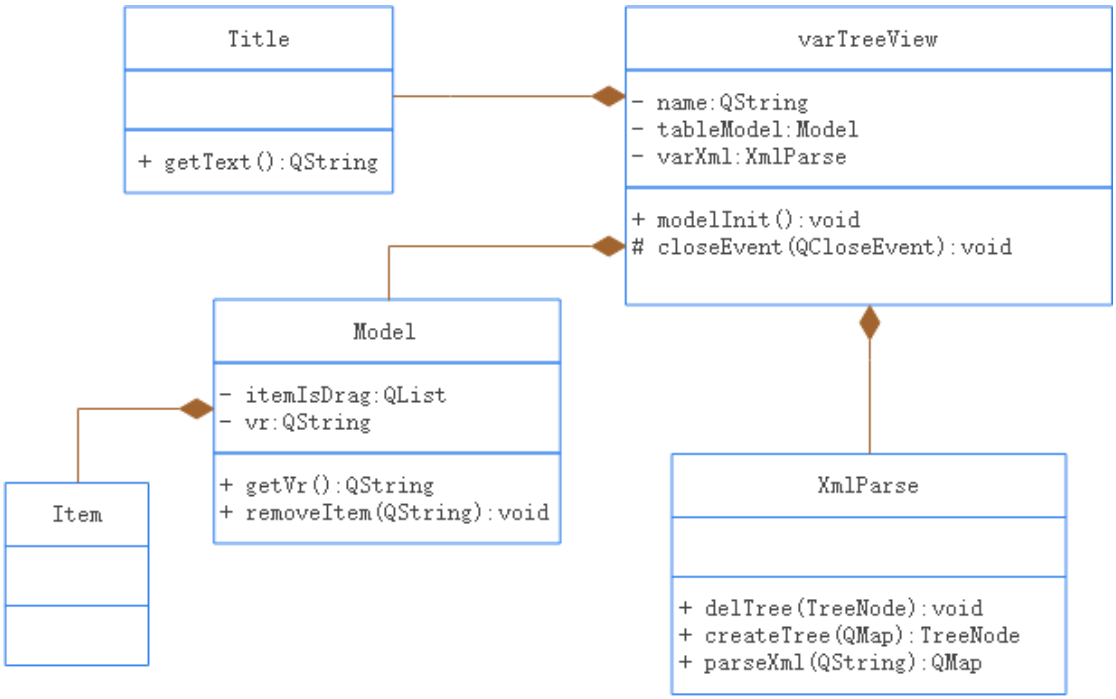


图 5-2 变量树模块类图

图 5-2 是根据该模块需要的类以及类之间的关系构造的类图。在该模块中，Model 类功能如上一节所述，XmlParse 类负责解析并存储有模型变量信息的 XML 文件以及根据解析之后的变量属性建立 N 叉树，varTreeView 类负责该模块与 UI 之间的逻辑控制以及模型的数据项组织。

在模型变量树模块的 UI 树形视图名字的设置通过 Title 类实现；树形视图中的变量通过 XmlParse 类解析 XML 文件得到；变量的树形展示通过 Model 类进行数据项重新组织得到，其中每一个数据项都是一个 Item 对象。因此 Model 类与 Item 类之间是组合关系，varTreeView 类与 Model 类、XmlParse 类、Title 类之间是组合关系。

在该模块中，首先，varTreeView 调用 modelInit()函数，在该函数中通过调用 XmlParse 类的 parseXml()函数完成 XML 文件的解析，获得变量属性信息 data，之后根据变量的部分属性调用 createTree(data)函数完成 N 叉树的构建。最后，该模块调用 createModel()函数，根据上一步生成的 N 叉树进行模型中数据项的组织，最终展示在 UI 界面中。

5.1.4 Monitor 模块设计与实现

本模块是上位机 UI 的核心模块之一，用户主要通过本模块来与下位机进行数据交互。一方面，用户可以在本界面修改变量的 value 然后下发给下位机模型，修改模型每个步长运行过程中的参数变量的 value。另一方面，再接收到 TCP 客

户端发送的信号后，显示下位机模型反馈回来的变量 value 值。该模块类图如图 5-3 所示：

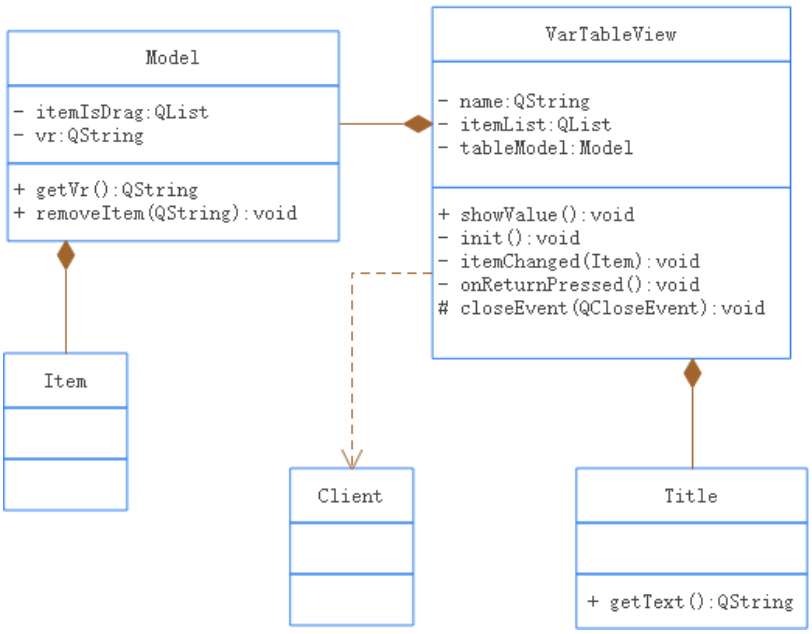


图 5-3 Monitor 模块类图

根据前文概要设计，该模块与上一小节类似，均为模型视图架构，主要有两点区别：首先该模块需要与下位机进行通信以及捕获 TCP 客户端发送的信号，因此该模块需要依赖于 TCPClient 模块；其次，该模块视图采用的是 QT 提供的 QTableView 类，不同于上一小节的所用的 QTreeView 类。

用户打开 Monitor 窗口时，该模块首先调用 init()函数，完成界面的基本属性的设置、信号与槽函数的关联。其次当用户拖拽变量到 Monitor 窗口内时，调用槽函数 itemChanged(item)，完成变量相关属性信息（除了 value 值）的显示。特别需要说明的是，用户的 value 项处于复用状态：一方面当用户修改 value 后回车，触发 onReturnPressed()槽函数，向下位机发送变量 ID 及 value,达到修改下位机模型变量的目的；另一方面当接收到 TCPClient 模块发送的信号后，触发槽函数 showValue()，在窗口显示下位机模型反馈的变量值。最终，为实现窗口信息保存功能，需重写 QT 提供的 closeEvent()函数，在窗口关闭的时候将所有内容保存到配置文件中。

5.1.5 Scope 模块设计与实现

本模块是上位机 UI 的核心模块之一，用户主要通过本模块来实时观测下位机模型所反馈的变量 value，进而进行 ECU 功能测试。在本模块中，用户可以自行勾选所要观测的变量，然后该模块根据 TCPClient 传递的数据在窗口完成变量

波形的绘制。

其中本模块对应的 UI 界面上，左侧显示变量名及其 value 信息，依然采用 model/view 架构。右侧需要绘制变量的波形信息，采用 QT 提供的 QtCharts 类完成变量波形的简易绘制。该模块类图如图 5-4 所示：

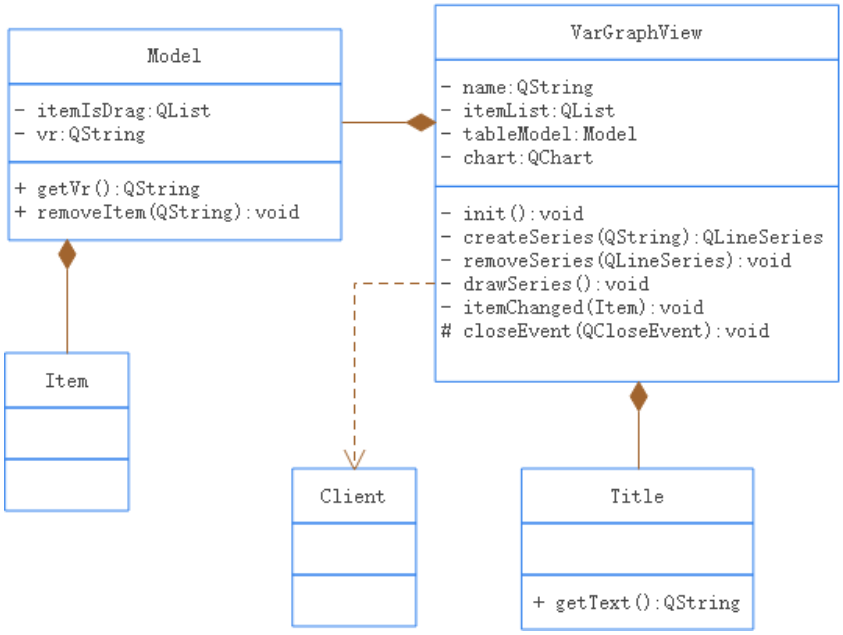


图 5-4 Scope 模块类图

本模块的整体结构与 Monitor 模块比较类似，与 Model 类和 Titile 类依然是组合关系，同时因为需要获得 TCPClient 传递的信号，因此依赖于 TCPClient 类。

用户打开 Scope 窗口时，该模块首先调用 init()函数，完成界面的基本属性的设置、信号与槽函数的关联。其次当用户拖拽变量到窗口内时，调用槽函数 itemChanged(item)，为该变量创建曲线、设置复选框等。

当该模块接收到 TCPClient 模块发送的信号后，触发槽函数 drawSeries()，进行变量波形的绘制。只有用户在窗口勾选变量之后，才能显示变量的波形曲线最终，为实现窗口信息保存功能，需重写 QT 提供的 closeEvent()函数，在窗口关闭的时候将所有内容保存到配置文件中。

5.2 下位机实时仿真模块设计与实现

5.2.1 TCP 服务端模块的设计与实现

本模块作为下位机与上位机的通信接口，主要与上位机建立通信连接，进行数据收发处理。本模块包含两个进程，其中一个进程用于监听客户端连接，另外一个进程用于数据收发功能。其中监听进程完成套接字初始化 socket()，套接字与端口的绑定 bind()，设置服务器的侦听连接 listen() ,当有客户端申请连接时，调用 accept()接受客户端连接，并创建收发进程。收发进程功能流程如下图所示：

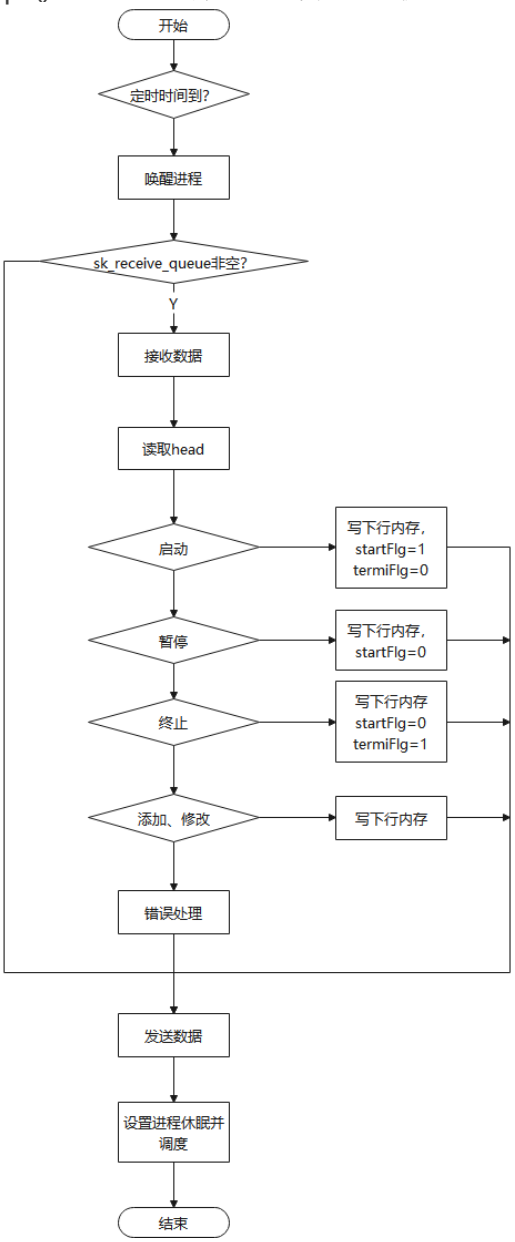


图 5-5 数据收发流程图

如上图所示，当定时器信号到达，唤醒收发进程。首先调用 skb_queue_empty()

函数判断上位机是否有发送数据。若没有数据，则调用 period_Send()函数：该函数首先从上行内存读取模型写入的数据，获得每帧的类型号。同时根据 startFlg 和该帧类型组织数据，累计够 N 个周期后，上发给上位机。若有数据到来，则读取数据，获取 head,根据之前规定的协议获取上位机发送的命令类型，并进行相应处理，然后调用 period_Send()。数据发送结束后，将当前进程阻塞并切换其他进程。

5.2.2 内存模块设计与实现

在 linux 系统下，常规的文件操作为了提高读写效率和保护磁盘，使用了页缓存机制。用户程序读文件时需要先将文件页从磁盘拷贝到页缓存中，由于页缓存处在内核空间，不能被用户进程直接寻址，所以还需要将页缓存中数据页再次拷贝到内存对应的用户空间中。这样，通过了两次数据拷贝过程，才能完成进程对文件内容的获取任务。写操作也是一样，待写入的 buffer 在内核空间不能直接访问，必须要先拷贝至内核空间对应的主存，再写回磁盘中（延迟写回），也是需要两次数据拷贝。考虑到本系统的实时性，采用 mmap 操作文件，实现了用户空间和内核空间的数据直接交互而省去了空间不同数据不通的繁琐过程，提高效率，节省时间。本模块分为用户空间和内核空间两部分。

在内核空间中，主要是创建三个文件。其中文件 A 作为信息文件，用于告诉用户层上下行内存的起始地址以及内存的大小，因此需要实现 file_operations 中的 open()函数，保证用户程序打开该文件时可以获取到所需信息。文件 B、C 分别作为上行内存、下行内存的映射文件，因此需要实现 file_operations 中的 mmap()函数，在该函数中通过 remap_pfn_range()函数建立页表，实现文件地址和虚拟地址区域的映射关系。在用户空间中，用户程序首先读取文件 A 获得上下行内存的起始地址以及内存的大小，然后调用内存初始化函数，启动映射过程。

5.2.3 信号配置与初始化模块的设计与实现

本模块是下位机用户程序的基础模块之一。根据概要设计，主要是建立起信号与 IO 板卡之间的一一映射关系，并完成信号的初始化配置。本模块流程图如下：

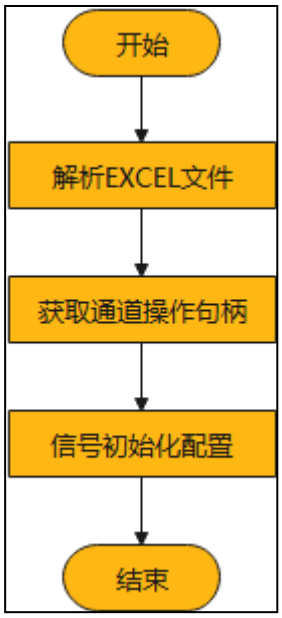


图 5-5 信号配置与初始化流程图

根据上图所示，第一步先解析硬件配置 Excel 表获得需要配置的信号与硬件之间的映射关系，以及初始配置信息。信号类型有 DI、DO、AI、AO、PWMI、PWMO、RES 总共 7 种。

第二步，根据上一步获取的信号—硬件映射信息，打开设备文件，获得通道句柄，将信号与句柄一一绑定。

第三步，根据获取到的配置信息，进行相应初始化配置。如 Digital Out 类型的信号需要设置指定输出通道电平配置（输出阻抗和外部参考电压配置），设置指定输出通道以高电平模式/低电平模式。以及配置通道使能。配置信息如下表 5-1 所示：

表 5-1 信号初始化配置

信号类型	初始化配置
DI	设置输入通道阈值电压
PWMI	设置输入通道 PWM 波形采集
DO	设置通道输出模式，参考电压、推拉模式，设置通道输出电平配置
PWMO	设置通道输出模式，参考电压、推拉模式，波形频率、占空比
AI	设置指定输入通道电压采样频率
AO	设置通道电压输出

RES	设置指定通道失调电阻值

5.2.4 模型任务运行模块的设计与实现

本模块是本文整个系统的核心模块，模型与 ECU 之间构成闭合回路。模型任务在系统调度下周期性的运行，在每个周期运行过程中涉及到与用户、ECU 之间的交互。模型运行之前需要从下行内存读取用户下发的指令信息并写入模型、读取来自 IO 板卡、CAN 卡采集的数据并写入模型。之后模型开始进行计算，计算结束之后模型需要将数据发送给 IO 卡和 CAN 卡，经 CAN 卡和 IO 卡转换之后发给相应的 ECU，同时还要上传给用户订阅的数据。本模块的流程图如图 5-5 所示：

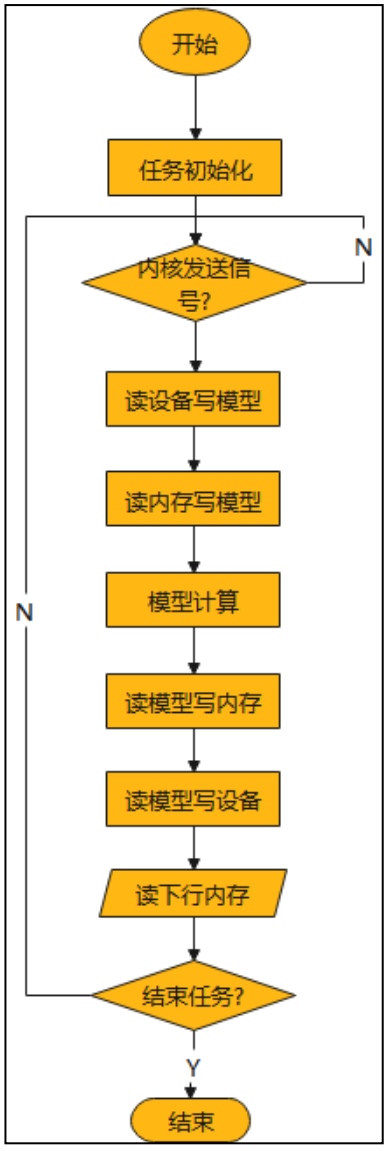


图 5-5 模型任务流程图

从流程图中可以看出，模型任务首先进行初始化工作，包括信号的硬件配置信息、综合 IO 板卡、CAN 卡、共享内存的初始化。其次注册模型任务，等待调度模块进行周期性调度。当接收到调度模块发送的信号之后，模型任务需要读取用户下发的指令信息（如订阅、删除、修改等），并做相应处理，接着读取 CAN 卡和 IO 卡采集数据并写入模型。此时读取工作结束后进行模型计算。模型计算结束后，将计算后的数据中的一部分上传给上位机展示给用户，另一部分数据经 CAN 卡和综合 IO 卡转换后反馈给待测 ECU。

5.3 本章小结

第 6 章 系统测试与分析

6.1 测试环境与工具

6.1.1 测试硬件环境

上位机部署环境：Windows10 操作系统，处理器为 Inter® Xeon® Gold 6134 CPU @ 3.20GHZ 3.19GHz,系统内存为 128GB,磁盘为 3TB。

下位机部署环境：工控机，欧拉操作系统。

软件测试工具：QtTest , GoogleTest。

综合 IO 板卡测试工具：示波器、信号发生器、6 位半高精度万用表、程控电源。

6.1.2 测试方案

根据前文需求分析，存在功能性需求以及非功能性需求，因此本章节内容分为功能性测试以及非功能性测试。功能性测试比较简单，主要包括对应各个模块的功能以及系统整体功能是否符合需求。非功能性测试主要在于下位机模型任务每个步长中模型、CAN 卡、IO 卡计算所占时间。

6.2 功能性测试

在本模块中根据第三章的需求分析，对各个功能模块进行测试。同时，因全部功能测试点较多，只选取其中重要的几个功能模块进行测试。主要有上位机 TCP 客户端模块的测试、变量树模块的测试、Monitor 模块的测试、Scope 模块的测试、下位机 TCP 服务端的测试以及模型任务模块的测试、综合 IO 板卡的测试。

6.2.1 TCP 客户端模块的测试

TCP 客户端模块，依据设计说明，主要包含通信连接、控制下位机、数据解析处理的功能。下面首先描述本模块各测试用例的测试前置条件：

通信连接的测试，前置条件是用户已经进入 TCP 客户端界面，下位机内核程序已加载，用户程序已执行。

启动模型运行的测试，前置条件是用户已经进入 TCP 客户端界面，上下位机已建立通信连接。

暂停模型的测试，前提条件是用户已经进入 TCP 客户端界面，下位机用户

程序中模型正在周期性运行。

终止模型的测试，前提条件是用户已经进入 TCP 客户端界面，下位机用户程序中模型正在周期性运行。

数据解析处理的测试，前提条件是上下位机已建立通信连接，下位机已向上位机发送数据。

下表是本模块的测试用例：

表 6-1 TCP 客户端模块测试用例

项目名称	ECU 在环仿真测试系统		
测试用例 ID	HIL_Test_001	测试方法	黑盒测试
模块名称	TCP 客户端模块	测试日期	2021.6.1
用例描述	测试 TCP 客户端的功能是否符合需求		
测试项目	输入	期望结果	实际结果
通信连接	输入下位机 IP,端口号,点击 Connect 按钮	Connect 按钮变绿,该，按钮内容变为 DisConnect	Connect 按钮变绿,该，按钮内容变为 DisConnect
启动模型	点击启动按钮	下位机模型开始周期性运行	下位机模型开始周期性运行
暂停模型	点击暂停按钮	下位机模型暂停运行	下位机模型暂停运行
终止模型	点击终止按钮	模型运行结束，用户程序终止	模型运行结束，用户程序终止
数据解析	下位机发送报文	窗口显示每帧类型，步长信息	窗口显示每帧类型，步长信息

6.2.2 变量树模块的测试

变量树模块，依据设计说明，主要包含 XML 解析、变量拖拽的功能。主要采用基于业务的场景设计法进行测试用例设计，覆盖上述功能点。下面分别描述本模块各功能的测试前置条件：

XML 解析的测试，前置条件是用户已经进入变量树界面，与模型配套的 XML 文件已准备好。

变量拖拽的测试，前置条件是 XML 已经解析完成，变量以树形结构在界面上显示。

下表是本模块的测试用例：

表 6-2 变量树模块测试用例

项目名称	ECU 在环仿真测试系统		
测试用例 ID	HIL_Test_002	测试方法	黑盒测试
模块名称	变量树模块	测试日期	2021.6.1
用例描述	测试变量树模块的功能是否符合需求		
测试项目	输入	期望结果	实际结果
XML 解析	点击 loadXml 按钮，选择 XML 文件	变量树形展示	变量树形展示
变量拖拽	鼠标推拽变量	变量可被拖拽到 Monitor、Scope 窗口	变量可被拖拽到 Monitor、Scope 窗口

6.2.3 Monitor 模块的测试

Monitor 模块，依据设计说明，主要包含变量拖放、修改变量、定时显示变量值、删除变量的功能。主要采用基于业务的场景设计法进行测试用例设计，覆盖上述功能点。下面分别描述本模块各功能的测试前置条件：

变量拖放的测试，前置条件是变量树窗口已打开，Monitor 窗口已打开。

修改变量的测试，前置条件是下位机用户程序中模型正在周期性运行，且该变量在下位机已完成硬件初始化配置工作。

定时显示变量值的测试，前置条件是 TCP 客户端模块向本模块发送信号，触发槽函数。

删除变量的测试，前置条件是窗口存在变量。

下表是本模块的测试用例：

表 6-3 Monitor 模块测试用例

项目名称	ECU 在环仿真测试系统		
测试用例 ID	HIL_Test_003	测试方法	黑盒测试
模块名称	变量树模块	测试日期	2021.6.1
用例描述	测试 Monitor 模块的功能是否符合需求		
测试项目	输入	期望结果	实际结果

变量拖放	从变量树窗口拖拽变量到本窗口	变量属性信息在窗口显示	变量属性信息在窗口显示
修改变量	修改变量的 value 字段，按回车下发	下位机模型获取变量 ID 及其 value	下位机模型获取变量 ID 及其 value
定时显示变量值	TCP 客户端发送信号，自行取数据	变量的 value 字段数据定时变化	变量的 value 字段数据定时变化
删除变量	选中变量，点击删除	变量删除	变量删除

6.2.4 Scope 模块的测试

Scope 模块，依据设计说明，主要包含变量拖放、定时显示变量值、绘制波形、修改波形颜色、删除变量的功能。主要采用基于业务的场景设计法进行测试用例设计，覆盖上述功能点。下面分别描述本模块各功能的测试前置条件：

变量拖放的测试，前置条件是变量树窗口已打开，Scope 窗口已打开。

定时显示变量值的测试，前置条件是 TCP 客户端模块向本模块发送信号，触发槽函数。

绘制波形的测试，前置条件 TCP 客户端模块向本模块发送信号，触发槽函数。

修改波形颜色的测试，前置条件是变量被勾选。

删除变量的测试，前置条件是窗口存在变量。

下表是本模块的测试用例：

表 6-3 Scope 模块测试用例

项目名称	ECU 在环仿真测试系统		
测试用例 ID	HIL_Test_004	测试方法	黑盒测试
模块名称	Scope 模块	测试日期	2021.6.1
用例描述	测试 Scope 模块的功能是否符合需求		
测试项目	输入	期望结果	实际结果
变量拖放	从变量树窗口拖拽变量到本窗口	在窗口显示变量名及其 value 值	在窗口显示变量名及其 value 值
定时显示变量值	TCP 客户端发送信号，读取数据	变量的 value 字段数据定时变化	变量的 value 字段数据定时变化
绘制波形	勾选变量	显示波形	显示波形
修改波形颜色	双击变量选择颜	波形颜色变化	波形颜色变化

	色		
删除变量	选中变量，点击删除	变量删除	变量删除

6.2.5 下位机 TCP 服务端的测试（待删除）

下位机 TCP 服务端模块，依据设计说明，主要包含通信连接、接收数据、发送数据的功能。主要采用基于业务的场景设计法进行测试用例设计，覆盖上述功能点。下面分别描述本模块各功能的测试前置条件：

通信连接的测试，前置条件是用户在上位机请求建立连接。

接收数据的测试，前置条件上位机发送数据。

发送数据的测试，前置条件是模型任务向共享内存已经写入数据。

下表是本模块的测试用例：

表 6-5 TCP 服务端模块测试用例

项目名称	ECU 在环仿真测试系统		
测试用例 ID	HIL_Test_005	测试方法	黑盒测试
模块名称	TCP 服务端模块	测试日期	2021.6.1
用例描述	测试 TCP 服务器端的功能是否符合需求		
测试项目	输入	期望结果	实际结果
通信连接	上位机请求建立通信连接	建立通信	建立通信
接收数据	上位机发送数据	收到数据	收到数据
发送数据	调度器唤醒发送进程	发送数据	发送数据

6.2.6 信号设定与分配模块的测试。

信号设定与分配模块模块，依据设计说明，根据物理布线情况以及板卡安装槽位，进行信号设置并将信号分配到相应的板卡及通道上。本功能的前置条件是经由 EXCEL 导出的硬件信号配置表已完成，板卡已成功安装：

下表是本模块的测试用例：

表 6-6 TCP 信号设定与分配模块测试用例

项目名称	ECU 在环仿真测试系统		
测试用例 ID	HIL_Test_006	测试方法	黑盒测试
模块名称	信号设定与分配模块	测试日期	2021.6.1
用例描述	测试信号设置与分配是否符合预期		
测试项目	输入	期望结果	实际结果
信号设置与分配	信号配置文件	硬件初始化成功	硬件初始化成功

6.2.7 综合 IO 板卡功能测试

综合 IO 板卡测试用例众多，功能性测试只选取 AI、AO、DO 和 PWM 的冒烟测试用例。用例的前置条件为：AIO 子卡正确插入工控机；DIO 子卡正确插入工控机；工控机上电；

下表是本模块的的测试用例：

表 6-7 综合 IO 板卡功能测试用例

项目名称	ECU 在环仿真测试系统		
测试用例 ID	HIL_Test_007	测试方法	黑盒测试
模块名称	综合 IO 板卡功能测试	测试日期	2021.6.1
用例描述	对 IO 板卡的 AIO 和 DIO 进行冒烟测试		
测试项目	输入	期望结果	实际结果
DI 输入范围测试：TC-DI-0011	1，配置每个通道 DI 输入；门限电压 4V； 2，同时喂给每个通道电压 0V，12V，19V，30V	2，观测测量到逻辑 0,1,1，1	符合预期
DO 输出范围测试：TC-DO-001	1，配置每个 DO 通道为 pull&push; 外部参考到 5V,12V,19V,30V； 2，同时设置全部通道逻辑 1 输出；	2. 万用表测试电压 5V,12V,19V,30V; 3．万用表测试电压 0V；	符合预期

	3，同时设置全部通道逻辑 0 输出；		
AI 输入范围测试: TC-AI-001	1，配置每个通道为 AI 输入； 2，依次喂给每个通道 AI+/AI- 电压：1V,6V,9V,12V,19V,24V,29.99V	2,观测得到每个通道的电压值误差小于 0.1%±5mV;	符合预期
AO 波形输出测试: TC-AO-008	1，同时配置每个通道输出 5*sin(2*pi*f)波形，其中 f=1Hz,10Hz,99Hz,500Hz, 1kHz,2kHz,3kHz,4kHz,5kHz；	1,示波器测量通道输出波形，观测正弦波频率和幅值在误差之内	符合预期

6.2.8 系统整体测试

本部分进行系统的整体功能测试，测试整个系统的功能是否符合预期结果。本模块以四个信号的测试为例进行说明：包块 ECU 的唤醒与休眠、车窗玻璃的升降。下表是本模块的测试用例：

表 6-7 系统整体测试用例

项目名称	ECU 在环仿真测试系统		
测试用例 ID	HIL_Test_007	测试方法	黑盒测试
模块名称	整个系统	测试日期	2021.6.1
用例描述	测试系统整体功能是否符合预期		
测试项目	输入	期望结果	实际结果
ECU 上电	在 monitor 窗口为唤醒参数赋初值	继电器上电，唤醒 ECU	继电器上电，唤醒 ECU
车窗玻璃上升	在 monitor 窗口为车窗上升参数赋初值	Scope 窗口绘制车窗上升波形	Scope 窗口绘制车窗上升波形
车窗玻璃下降	在 monitor 窗口为车窗上升参数赋初值	Scope 窗口绘制车窗下降波形	Scope 窗口绘制车窗下降波形
ECU 下电	在 monitor 窗口为	继电器下电，ECU	继电器下电，ECU

	下电参数赋初值	休眠	休眠

6.3 非功能性测试

6.3.1 模型任务性能测试

对于本系统来讲，下位机需要保模型任务尽可能在每个周期之内都能完成。因此需要测试每个周期内 CAN，IO 计算所占的系统资源，消耗时间以及模型计算所消耗的时间。测试思路如下：

1) 挂载内核模块，启动用户程序；
2) 上位机 UI 发送信号，启动模型任务。
3) 在任务中每 500ms 打印出系统资源消耗最多的一次时间（us）。

测试结果如表 6-1 所示：

表 6-1 模型任务运行耗时图

线程	耗时（us）
CAN	200-300
IO	200-300
Model	200-300

从上表可以清楚看出：每 500ms 收集到的最大一次 CAN 计算资源时间在 200– 300us 之间，模型计算耗时在 200-300us 之间，IO 计算耗时在 200-300us 之间符合预期。

6.3.2 综合 IO 板卡性能测试

本模块的测试只选取综合 IO 板卡的 DO 和 DI 的性能测试。测试用例的前置条件为：DIO 子卡正确插入工控机；工控机上电；测试用例如下表 6-1 所示：

表 6-1 综合 IO 板卡性能测试

项目名称	ECU 在环仿真测试系统		
测试用例 ID	HIL_Test_007	测试方法	黑盒测试

模块名称	综合 IO 板卡性能测试	测试日期	2021.6.1
用例描述	对 IO 板卡的 AIO 和 DIO 进行冒烟测试		
测试项目	输入	期望结果	实际结果
DI 输入精度测试： TC-DI-006	1，配置每个通道为 PWMIN，门限电压 2.5V； 2，用信号发生器同时喂给通道 5V PWM 波，频率 f=1kHz，占空比=0,1,10,49,50.2,99,100； 3，用信号发生器同时喂给通道 5V PWM 波，频率 f=200kHz，占空比=40,48,52,60；	2，观测采集的频率误差<A，占空比误差<B 3，观测采集的频率误差<A，占空比误差 B;	符合预期
DO 输出与输入：全部通道 TC-DO-023	1，连接全部 DO 输出 PIN 到 DI 输入 PIN 2，配置通道 DO，pull&push，外部参考到 12V；配置通道 DI，门限电压 6V； 3，同时设置通道 DO 逻辑 0 输出； 4，同时设置通道 DO 逻辑 1 输出；记录 DO 设置逻辑 1 到 DI 采集到逻辑 1 的时间；	3，DI 采集到逻辑 0； 4，DI 采集到逻辑 1；从 DO 设置逻辑 1 到 DI 采集到逻辑 1 的时间小于 A；	符合预期
AO 到 AI 的响应时间：全部 15 个通道： TC-AO-062	1,配置全部 AO 通道 AO+/-连接到对应的 AI+/-; 2,配置通道 AO 先输出 4V；测量 AO 输出 4V 到 AI 采集到 4V 的时间；	2,AI 采集到 4V；AO 输出 4V 到 AI 采集到 4V 的时间小于 A;	符合预期

根据实际的测试结果分析，综合 IO 板卡的性能符合实际预期。

6.4 本章小结

第 7 章 总结与展望

7.1 总结

本文以实习中参与开发的 mini-HIL 项目为课题，课题目标旨在如何进行硬件在环仿真测试系统的设计与实现。

本问题从开题调研开始，到整个系统的的实现，主要完成的工作有：

1）根据项目的需求分析对整个系统进行梳理和划分。根据实际的工作需求划分不同的模块，分为上位机监控软件、下位机实时仿真平台以及 IO 接口。

2）主要参与完成了上位机监控软件的开发，下位机部分模块的开发。并针对上下位机的数据通信，实现通信协议的规定，保证上下位机数据一致性。

3）对上下位机的软件功能以及 IO 板卡进行功能性测试以及对下位机实时仿真任务进行性能测试。功能性测试中，每个模块对应的功能都给出了相应的测试用例。非功能性测试中，从下位机模型任务的实时性入手，保证模型任务的运行周期在要求范围之内；从 IO 板卡的 DO 和 PWMO 为例，验证通道的性能是否在误差允许范围之内。 根据最终的测试结果来看，系统较好地满足了功能性需求以及非功能性需求。最后，通过本文所搭建的系统，可以实现对现有的 ECU 的功能测试。

7.2 展望

由于硬件在环仿真测试系统的复杂性以及开发时间的紧迫性，目前还有部分功能尚未来得及实现，并且该系统仍有许多可以进行优化的地方：

1）上下位机通信部分，数据下行通信协议后续可以进行优化，以支持兼容更多的业务需求。数据上行协议根据后续开发仍需要进行扩展或者进一步的优化。除此之外，数据传输使用的是 TCP 协议，后续将在 UDP 协议的基础上再开发而进行数据通信。

2）上位机监控软件仅仅做了部分基础功能，诸如类似示波器绘制的波形，波形的游标、放大缩小功能尚未实现。除此之外，信号的自动化配置管理，测试自动化等功能将在后续工作中设计实现。

3）随着下位机模型的复杂性增加，对硬件板卡也有更高的要求，现行初步搭建的系统仅具备综合 IO 和 CAN 卡，诸如 LIN 卡、ETH 卡在后续工作中仍需进行开发实现。

参考文献

写作参考：

详细设计/概要设计：https://www.jianshu.com/p/a52fe78962f0

Mmap 参考:https://juejin.cn/post/6956031662916534279

用例图：https://blog.csdn.net/zxsydyq/article/details/6967308

致 谢