

面向对象的CAN通信软件设计

邵 晖

上海聚星仪器有限公司

1 前言

汽车电子大多以控制局域网络(controller area network, CAN)总线通信。允许网络上多个带有微控制器的电子设备之间交换信息而不需要主控计算机介入。这个协议最早是1983年博世(Robert Bosch GmbH)开发,并于1986年在汽车工程师学会(society of automotive engineers)底特律会议发表。1988年宝马8系列车应用了CAN总线。1993年国际标准化组织(ISO)接纳CAN为标准ISO11898。其后博世持续演进CAN技术,发明了高速、低速容错、单线CAN等等协议,并且将其说明和白皮书公开,而11898标准也有了多个分项标准ISO11898-1到-5。目前,美国和欧洲都强制汽车具备车载诊断系统(OBD-II),而CAN是支持OBD-II的5个协议之一。由于CAN总线的汽车应用,实现了可靠、经济的局域网通讯,使之在汽车之外的工业领域也得到广泛应用。

面向对象软件设计是1990年发展起来2000年之后由于JAVA、C#等计算机语言支持下发展起来的现代软件

设计方法。目前排名前10位的计算机语言当中统计,有56%的程序员使用面向对象编程语言。面向对象语言使得软件更加可靠、灵活并提升代码重利用率。本文在汽车收音机测试背景下,介绍面向对象的CAN通信软件设计。

2 汽车收音机测试

汽车收音机是娱乐系统最基本的组成部分。由于汽车电子产业对于出厂合格率极高的要求,汽车收音机的生产测试成为严谨的射频、音频综合测试。表1典型收音机测试项目表展示了收音机测试的部分典型项目。为满足这些测试项目建立的汽车收音机的功能测试原理如图1所示。有时候为了适应双信号测试,需要2台信号源合路输送给收音机。也有时候需要测试极小的休眠电流,需要在电源输入端配置电流表。

其中绝大多数测试的步骤就是通过CAN给收音机发送指令,用信号源给出测试信号,然后用音频测试仪测量。限于篇幅关系,本文集中讲解CAN的通信编程设计。

表1 典型收音机测试项目

AM、FM	FM
内部干扰抑制	信噪比
噪声限制灵敏度	无台噪声
搜索停止电平	频偏适应性
电台搜索时间	AM抑制
中频抑制	FM立体声
镜频抑制	通道一致性
选择性和阻塞	单声道-立体声切换
大信号能力	导频和子载波抑制
自动增益控制	...

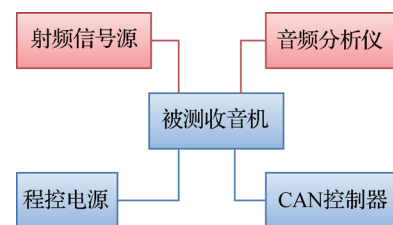


图1 汽车收音机测试系统原理

3 CAN通信模式

首先常见的CAN有3种协议,高速、低速容错,和单线。你首先要搞清楚协议类型和接口阻抗匹配要求。其中接口阻抗大多是120 Ω ,在控制器这一端需要并联一个120 Ω 电阻。有的CAN控制器板上自带,但是需要用跳接端子选择,有点控制器需要在电缆端子上焊接一个。极其偶然,也有不是120 Ω 的,需要和厂商沟通。端接电阻不匹配会导致通信不可靠。

物理上接通了,就进入软件设计。汽车电子CAN控制器通信模式主

要有3种：只发不收、先发后收，和先收后发。如图2所示，汽车网络通常需要一些包括单次的和持续广播的信息，通知网络上的设备进入某种模式，或者通知汽车系统正常，大家可以保持工作。前者包括钥匙到位，点火等指令。后者信号常被称为心跳信号(Heartbeat)。在整个测试过程中，CAN控制器需要向被测件单向定时广播这些心跳数据帧。这种广播可能是100 ms~1 s重复的数据帧。一旦这个心跳数据广播停止了，被测收音机可能很快就自动关机了。

先发后收模式是典型的指令-响应模式。控制器发送指令给被测件，被测件给出回应，控制器根据当前任务逻辑和回应决定终止还是继续下一个指令发送。

先收后发模式是控制器对被测件请求的响应。有的被测件会主动提出请求，需要测试系统的控制器给予回应，不然被测件会自动断开连接。所以CAN控制器必须有较快实时应答能力。

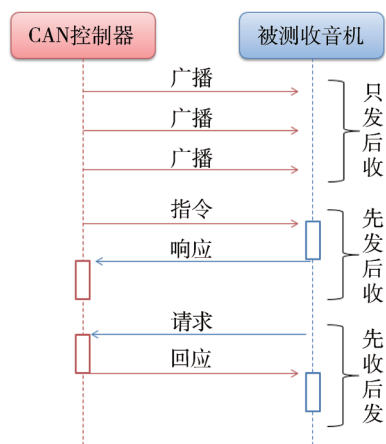


图2 CAN通信模式序列

4 CAN通信对象类设计

传统上我们要设计一批子函数，专门针对收音机CAN控制。现在基于面向对象原理设计，可以设计成两个对象，收音机和CAN控制器。不妨称我们的收音机为Radio101，CAN控制器为CANController101。他们的功能设计如图3所示。对于测试序列程序员来说，只用到收音机Radio101对象。他关心的是如何打开收音机，配置调谐波段和频率，然后通过信号源发信号，音频分析仪测量音频。对于收音机对象驱动开发程序员来说，他要用CAN控制器对象实现这些方法。

也就是Radio101对象的方法是调用了CANController101对象的方法和属性的。而CANController101内部调用了CAN控制器的驱动，完成了诸如导入CAN识别号和地址数据库，发送指令，收发长数据等功能。

其中“导入数据库”方法根据DBC文件或者NCD文件，解析出各个CAN网络变量的ID编号、变量数据比特在该ID号的数据帧里面的位置、以及变量量化的增益和零偏等信息。

“写网络变量”方法将特定的数据根据数据库规则规定转换成比特序列，写入相应ID编号的数据帧。由于写变量往往是位于一个64字节的数据帧里面，所以软件会使用读-修改-写的流程，保证仅仅修改需要写入的变量。汽车钥匙插入、点火、心跳等信息，都可能采用这样的方法设置。“读网络变量”方法根据软件要求从数据库

规定的帧读取变量比特，并转换为变量值返回。“发送指令”方法将软件规定的指令帧发送到被测件，并读取被测件响应。这个方法在测试流程中使用最为普遍。诸如设定波段和频率、设定CD曲目大多是用这样的指令完成。“发送长指令”方法通过多次指令发送和应答，传输较大数据给被测件。通常在写入Flash数据时使用。“接收长数据”方法通过多次指令发送和应答，命令被测件传回较大数据，可以用来读回Flash数据，进行校验。

Radio101	CANController101
+ 初始化连接 + 电源开关 + 收音机调谐 + CD机选曲 + 声音设置 + 引擎状态 + 点火状态 + 读属性 + 写属性 + 调用功能 + 关闭连接	+ 通信速率 + 数据库文件路径 + 初始化 + 导入数据库 + 写网络变量 + 读网络变量 + 发送指令 + 发送长指令 + 接收长数据 + 关闭

图3 收音机和CAN控制器类图

5 范例

例如某收音机接收的汽车点火状态变量heartbeat在数据库里面定义为ID: AD80BD0, 起始比特55, 长度1比特。软件将相应数据库文件导入后, 就可以通过写网络变量实现:

```
string signalName =
    "heartbeat";
byte heartbeat=1;
CANController101.
WriteSignal(signalName,
    heartbeat);
```

这个方法自动到数据库设置里面找到心跳信号所在数据帧ID和位置,读取这个数据帧,根据变量heartbeat修改数据帧里面第55比特的数值,再写回。

又例如设置收音机接收波段和频率,调用发送指令方法:

```
byte[] command = { 0x07,
0x31, 0x01, 0x21, 0x02, 0x00,
0x00, 0x00 };
byte band = 1; //1:FM,
2:MW, 3:SW;
byte[] frequency = { 98, 01 };
byte[] response = newbyte[8];
command[5] = band;
command[6] = frequency[0];
command[7] = frequency[1];
CANController101.
Send(radioID, command, ref
response);
```

其中, response是被测件返回数据。可以根据返回信息判断是否设置成功。

6 增加一个被测型号

软件工程需要可靠性和灵活性。客户经常会增加被测件型号,而且型号间指令会有差异。在汽车电子行业,往往采用平台化开发。比如大众的途观、途安、高尔夫都属于同一个平台。各种车型包括高档、低档、汽油、柴油等等,可能有好多种类。同一个汽车电子厂可能会在短时期内小批量多批次地生产大量的汽车收音机。这就要求产线软件能够迅速可靠地升级适应不同指令的被测件。

在我们这个设计当中,你不需要修改CANController101,只要建立另一个收音机对象,不妨称Radio102,将Radio101对象的代码复用在这个新对象里面,稍作修改,就可以适应新的收音机。而上层的测试序列软件,也不需要修改,只要将收音机对象实例化类从Radio101更换为Radio102就可以了。

7 换一个CAN通信卡

另一方面,当工厂生产若干年,测试模块的技术更新,使得老的CAN控制器型号被淘汰,就需要采用新的CAN控制器。在我们的设计中,更换CAN控制器型号导致控制器驱动变化,不需要修改收音机对象,只要生成新的控制器对象。不妨假设新的控制器叫102,对象名CANController102。你只要按照CANController101的属性、方法定义完成CANController102的类库就可以。大多数情况,硬件驱动接口有相当延续性。你可以较多复用原来

CNController101的代码。由于我们设计对象的公开属性和方法保持不变,你在Radio101和Radio102里面只要实例化CANController102就可以,而不用修改任何主体程序。

8 总结

传统的结构化编程用一系列子函数来实现通信控制。很容易造成子函数庞杂混乱,数据定义域和作用域过于泛化增加可靠性风险。面向对象的编程将软件当中和硬件对应的对象包装成被测件和控制器对象,每个对象的内部数据和操作对外不可见,大大提高了软件模块内聚度和降低了模块间的耦合度。我们科学地将被测件和控制器对象拆分开,使得灵活性和代码重利用率进一步提高。聚星基于这种架构,研制了多媒体并行测试系统,如图4所示。

这个设计案例体现了科学的面向对象软件设计的优势,使得软件灵活可靠重利用率高,而且模块开发相对独立,便于团队软件工程。



图4 聚星多媒体并行测试系统