文章编号：1674-8085（2022）06-0081-06

基于 VS.NET 及串口通讯的线控制动操控系统开发与验证

*张利芬 [1,2]，时培成 [1]，贾慧利 [2]

（1.安徽工程大学机械工程学院，安徽，芜湖 241000；2.芜湖职业技术学院汽车与航空学院，安徽，芜湖 241006）

摘 要：线控制动系统中信号传输处理的速度及稳定性会对整个制动操控效率产生决定性的影响，是影响制动距离及其恒定性的主要因素之一。选择不同的开发平台和通讯类型，其效果也不一样。本研究基于 VS.NET 平台及串口通讯方式，设计开发线控制动的操控系统，围绕制动操控效率和可靠性两个目标元素，从生效速度、操控效率的稳定性两个方面对线控制动操控系统进行性能验证，包括制动需求分析、开发背景介绍、系统设计以及基于紧急持续制动、点动缓刹、常规持续限速三种典型工况下的性能验证。

关键词：VS.NET 平台；C#；线控制动；操控系统；测试验证

中图分类号：TP311.51 **文献标识码**：A DOI：10.3969/j.issn.1674-8085.2022.06.013

DEVELOPMENT AND VERIFICATION OF BRAKE-BY-WIRE CONTROL SYSTEM BASED ON VS.NET AND SERIAL COMMUNICATION

*ZHANG Li-fen [1,2], SHI Pei-cheng [1], JIA Hui-li [2]

（1. School of Mechanical engineering, Anhui Polytechnic University, Wuhu, Anhui 241000, China;

2. School of Automobile and Aaviation, Wuhu Institute and Technology, Wuhu, Anhui 241006, China）

Abstract：The speed and stability of signal transmission and processing in the brake-by-wire control system has a decisive impact on the whole braking control efficiency, also is one of the main factors affecting the braking distance and constancy. Different development platforms and communication types have different effects. Based on VS.NET platform and serial port communication mode, the brake-by-wire control system was designed and developed, focusing on the two objective elements of braking control efficiency and reliability, the performance verification of the line control brake control system was carried out, including braking demand analysis, development background introduction, system design, and performance verification under three typical working conditions as emergency continuous braking, inching braking and conventional continuous speed limit.

Key words: VS.NET platform; c#; brake-by-wire; control system; test verification

收稿日期：2021-08-09；修改日期：2022-02-10

基金项目：安徽省优秀青年骨干教师国内访学研修项目(gxgnfx2021189)；安徽高校自然科学研究重点项目（KJ2020A0909)；芜湖职业技术学院校级自然重点科学研究项目(wzyzrzd202003)

作者简介：*张利芬(1989-)，女，湖北黄石人，讲师，硕士，主要从事新能源汽车电控方向、汽车设计与制造技术等教学与研究(E-mail:zhanglf@whit.eddu.cn).

0 前言

顺应的汽车电动化和智能化的两大发展趋势，x-by-wire（线控操控）成为了最理想的操控方式，目前处于广泛研发试验阶段。制动效能及效能的恒定性是评价机动车制动安全的两个核心指标，对于线控制动而言，亟待解决的是制动生效的灵敏性和可靠性问题，这些取决于整个制动操控实现过程中的软件通讯和解码，而软件通讯和解码的速度、准确度和实时性取决于软件模块的开发平台、通讯类型、信号传输、转码速度及性能稳定性等[1]。

目前汽车x-by-wire系统开发，通讯主要依赖于CAN和FlexRay总线技术[2-3]，FlexRay可以更好地满足制动操控高精读高灵敏的要求，但在汽车上的应用不成熟，成本高，开发工作量大。CAN通讯为目前车载网络的主要通信方式，技术成熟，应用成本较低，但实时性和容错性不足。目前在线控软件开发中主要有Java/go和.NET/C#两种模式，两者搭建的应用程序有类似的性能，但在面向对象控制方面，C#使得一些在Java中过于复杂的特性变得简单，另外据TechEmpower测试，.NET Core框架允许在一台服务器上每秒处理的请求数量远远领先于任何Java开发的web框架[4]。本项目根据线控制动系统的常规结构、常规制动规律，基于VS.NET平台和CAN串口通信模式，拟借VS.NET平台及C#开发语言组合开发模式能够简化公共组件、减少繁杂代码堆砌以及处理速度快的优势，来弥补CAN串口通讯实时性和容错性的不足，对制动要求的达成度进行探究。结合程序逻辑的优化，开发独立线控制动操控系统，并对不同制动工况下的制动生效速度和可靠性进行测试验证。

1 线控制动需求分析

如图 1 为驾驶员接受紧急制动信号后的制动过程示意图，其中 τ 为制动时间，F_P为制动踏板力，a_b为制动减速度。

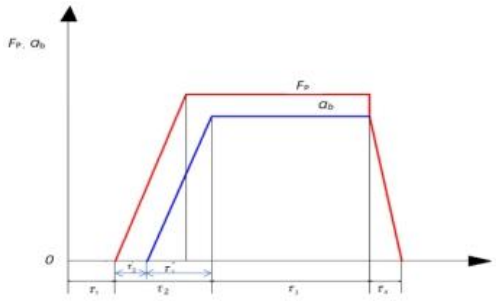

图 1 汽车制动过程

Fig. 1 vehicle braking process

制动距离=反应时间距离（此处不考虑驾驶员反应时间）+刹车距离，即[5]

$$S=\frac{1}{3.6}+\left(\tau_2^{'}+\frac{\tau_2^{''}}{2}\right)\mu_{a0}+\frac{\mu_{a0}^2}{25.92a_{b\max}}$$

在线控制动中，踏板只提供模拟信号，所以F_P可以等效为踏板行程，τ_2为制动器作用时间，可以看出，除了制动初速度和制动减速度（线控制动中主要取决于踏板行程、执行电机和执行制动器）之外，影响制动距离很重要的一个因素为制动器起作用时间$\tau_2^{'}$，具体包括消除自由行程、信号传输处理、消除制动间隙所占的时间。由于线控系统不用考虑主缸的复位问题，踏板自由行程只起到缓冲保护的作用，所以设置值较小，且为标准值。另外制动间隙一般为避免制动拖滞而设置，加入自调装置后也基本为标准值。因此，要想通过$\tau_2^{'}$来优化距离及制动效能的恒定性必须从信号传输处理方面加以改善。

2 基于 VS.NET 平台的系统设计及功能实现

2.1 系统设计

线控系统主要由刹车踏板、踏板位移传感器、ECU（电控单元）、数据总线、伺服电动机和制动执行机构组成。

采用PWM信号发生器模拟油门踏板，踏板位移传感器采用线性函数模拟，ECU采用PC模拟，数据总线使用USB_CAN模块的双向通讯模拟，刹车的

执行单元由.NET进行仿真。软件程序在VS2019 平台进行开发，核心代码如下：

```
private void stay_up1(object sender, MouseEventArgs e)
{
stay1.Stop();
uptime1 = stay1.Duration;
stay1.ClearTimer();
//信号生效时间公式
total += 1;
//textBox1.Text = (uptime1 +
Convert.ToDouble(Effective_itinerary.Text) *
Convert.ToDouble(total_trip_time.Text)).ToString();
//持续刹车信号持续时间=刹车踩下总时间-2*刹车生效行程
*刹车踩到底耗时*当前工况刹车系数
textBox1.Text = (uptime1-
Convert.ToDouble(Effective_itinerary.Text)*2*
Convert.ToDouble(total_trip_time.Text)*
Convert.ToDouble(textBox11.Text)).ToString();
textBox12.Text = uptime1.ToString();
//下面仿真数字信息转换成帧信息换算所需要的时间
textBox3.Text = set_message(9, 0).ToString();
//下面模拟帧信息从发送到ECU收到所消耗的时间
if (!ECU_port.IsOpen)
{
ECU_port.PortName = ECU_port_name.Text;
ECU_port.Open();
}
time_to_ECU.Start();
ECU_port.WriteLine("send");
label33.Text = total.ToString ();
}
```

系统可视化操作界面如图 2 所示。

图 2　软件操作界面

Fig. 2　software operation interface

2.2　功能实现

在所开发的线控制动操控系统中，主要目的为：根据实际制动需求，模拟不同的制动工况，对制动器起作用时间 τ_2' 中跟软件信号传输转码相关的部分进行仿真测试。共包括以下几个部分：

1）信号生效时间。主要是指刹车从开始踩下到产生刹车信号所使用的时间，即：

信号生效时间=开始踩下踏板时间－（预设点动生效行程/踏板总行程）×预设人体刹车移动至最终行程耗时；

2）信号持续时间。主要是指在踏板持续踩踏时，刹车信号真正的持续时间，即：

信号持续时间=从踏板开始踩下到最后抬起的耗时－信号生效耗时；

3）信号编码耗时。从踏板的电信号转换成CAN通信帧所需要的的时间；

4）ECU收到消息耗时。从踏板发送CAN信号到收到ECU的应答信号的耗时，使用modus协议转码来仿真信号转换过程，使用串口线 2、串口线 3 短路的方式模拟信号收发；

5）刹车模块接收到ECU消息耗时。从ECU发送执行指令到刹车模块确认收到信号的耗时，测试原理同上。

3　试验平台搭建及多工况试验验证

3.1　试验平台搭建

基于所开发操控系统的控制逻辑及功能目标所搭建的试验平台逻辑框图如图 3 所示。

串口 1 通过短路模块在软件仿真中模拟点到点的车内通讯。

PC通过串口 2 实现IO的开关控制，这里IO板模拟刹车的执行机构（继电器结构）。

CAN模块通过仪器仿真车内的点到点通讯，不考虑线长的影响，此处的仿真是为了获得更精细的数据，可以替换软件仿真中的“ECU收到消息帧耗时”和“刹车收到消息帧耗时”。

ECU仿真端口号：用来模拟ECU的串口模块所在的串口号。

IO板端口号：用来模拟执行器电信号转换模块

通讯所使用的串口号。

CAN1 波特率：CAN1 口通讯使用的频率。

CAN2 波特率：CAN2 口通讯使用的频率，需将CAN2 和CAN1 频率设置为一致，再点击打开设备。

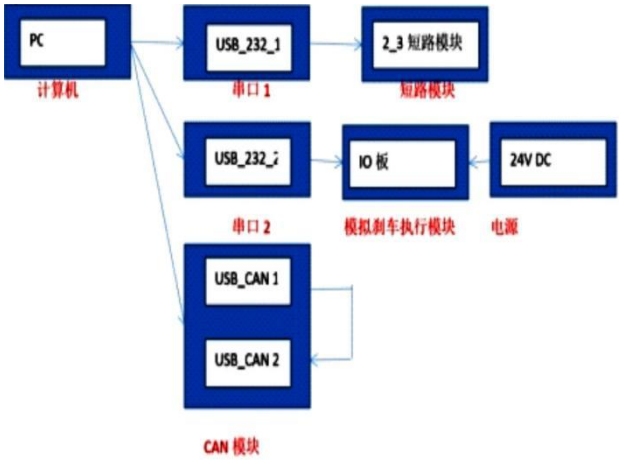

图 3　系统逻辑框图

Fig.3　system logic block

基于以上逻辑所搭建的试验系统物理结构如图 4 所示：1）计算机 1 台；2）绿联USB转 232 通讯线 2 根；3）232 控制继电器模块 1 个（4 路 LH-04C）；4）双通道USB转CAN盒 1 个（广成科技 USBCAN-II C）；5）手工制作 232 2_3 短接模块，使形成回路系统；6）易达源电子 3-24V可调直流电源 1 个。

图 4　系统物理结构

Fig. 4　system physical structure

3.2　测试方法及数据采集

3.2.1　预设生效行程

用小数表示刹车开始反馈有效信号的行程位置，例如：该数字为 0.08 时，即为当刹车踩下总行程的 8%时，刹车开始反馈给ECU刹车信号。

3.2.2　预设刹车踩到底耗时

预设的刹车从完全释放状态至踩到底所需要的总耗时，这个时间相当于设置了踩刹车的速度，给的时间越少，踩刹车速度越快。

3.2.3　本次工况刹车行程

当前刹车的踩踏程度，该数据只使用在持续信号测试中，当设置为 0.8 时，即为当前工况设置中，刹车最多踩至 80%的位置。

CAN传输时间：点击CAN传输模拟按钮后，该数值开始出现，表示刹车信号从刹车踏板模块到ECU模块的耗时（包括转码时间）以及其从ECU到刹车执行模块的耗时，因此在总耗时计算中该部分需要计算两次。

命令发出耗时：点击发送命令至IO板按钮后，该数值开始出现，表示刹车模块从收到刹车信号到发出执行指令所耗时间。

3.3　试验结果分析

根据城市道路使用汽车实际制动需求试验选择：1）紧急全行程持续制动；2）多次部分行程点制动；3）常规部分行程持续制动三种典型工况，对制动时间及性能的稳定性进行测试。具体结果如下（单位均为毫秒）：

（1）紧急全行程持续制动试验结果如表 1 所示。

表 1　紧急全行程持续制动测试

Table 1　Test results of emergency full stroke continuous braking

信号生效用时/信号持续时间	刹车总时间	信号编码用时	ECU 收到消息用时	刹车模块收到消息用时	时间	测试类型
1633.9661	1793.9661	0.01	1.2159	1.2159	18:29:30	持续
2137.0218	2297.0218	0.0058	1.2343	1.2343	18:29:33	持续
2952.768	3112.768	0.0059	1.2622	1.2622	18:29:37	持续
2772.1138	2932.1138	0.0058	1.3147	1.3147	18:29:41	持续
2251.3903	2411.3903	0.0052	1.2143	1.2143	18:29:44	持续
2011.5605	2171.5605	0.0059	1.1973	1.1973	18:29:46	持续
1786.2685	1946.2685	0.0105	1.2151	1.2151	18:29:49	持续
1770.0894	1930.0894	0.0068	1.2582	1.2582	18:29:52	持续
1698.6882	1858.6882	0.0057	1.1931	1.1931	18:29:54	持续
1561.5987	1721.5987	0.0057	1.2632	1.2632	18:29:56	持续

多次部分行程点制动试验结果如表 2 所示。

表 2 多次部分行程点制动测试

Table 2 Test results of multiple partial stroke point braking

信号生效用时/信号持续时间	刹车总时间	信号编码用时	ECU 收到消息用时	刹车模块收到消息用时	时间	测试类型
393.0284	553.0284	0.0092	1.2391	1.2391	18:39:39	点动
369.3934	529.3934	0.0131	1.2968	1.2968	18:39:40	点动
362.0963	522.0963	0.0064	1.2514	1.2514	18:39:42	点动
394.5845	554.5845	0.0061	1.221	1.221	18:39:44	点动
419.2306	579.2306	0.006	1.2354	1.2354	18:39:45	点动
434.7465	594.7465	0.0053	1.2441	1.2441	18:39:47	点动
475.6106	635.6106	0.0061	1.2373	1.2373	18:39:48	点动
514.1831	674.1831	0.0093	1.2413	1.2413	18:39:50	点动
610.6527	770.6527	0.0059	1.2195	1.2195	18:39:51	点动
313.3019	473.3019	0.0057	1.2513	1.2513	18:39:52	点动
409.1276	569.1276	0.0058	1.2559	1.2559	18:39:54	点动
411.2074	571.2074	0.0093	1.3088	1.3088	18:39:55	点动
489.9303	649.9303	0.0056	1.2208	1.2208	18:39:57	点动
521.9173	681.9173	0.0081	1.2379	1.2379	18:39:58	点动
337.352	497.352	0.006	1.2254	1.2254	18:40:00	点动

3.2.3 常规部分行程持续制动

1）预设行程参数为 0.5 持续制动试验结果如表 3 所示。

表 3 预设 0.5 行程制动测试

Table 3 Test results of preset 0.5 stroke braking

信号生效用时/信号持续时间	刹车总时间	信号编码用时	ECU 收到消息用时	刹车模块收到消息用时	时间	测试类型
2079.7449	2159.7449	0.0056	1.2465	1.2465	18:58:59	持续
2395.0368	2475.0368	0.0056	1.3759	1.3759	18:59:03	持续
3006.7336	3086.7336	0.0057	1.2834	1.2834	18:59:07	持续
2689.9571	2769.9571	0.0057	1.3114	1.3114	18:59:10	持续
2953.1701	3033.1701	0.0054	1.311	1.311	18:59:14	持续

预设行程参数为 0.6 持续制动试验结果如表 4 所示。

表 4 预设 0.6 行程制动测试

Table 4 Test results of preset 0.6 stroke braking

信号生效用时/信号持续时间	刹车总时间	信号编码用时	ECU 收到消息用时	刹车模块收到消息用时	时间	测试类型
2602.213	2698.213	0.0057	1.2839	1.2839	18:51:56	持续
2242.014	2338.014	0.0057	1.2945	1.2945	18:51:59	持续
4818.026	4914.026	0.0106	1.2308	1.2308	18:52:05	持续
4651.437	4747.437	0.0061	1.2785	1.2785	18:52:11	持续
3826.113	3922.113	0.0085	1.2214	1.2214	18:52:15	持续

预设行程参数为 0.7 持续制动试验结果如表 5 所示。

表 5 预设 0.7 行程制动测试

Table 5 Test results of preset 0.7 stroke braking

信号生效用时/信号持续时间	刹车总时间	信号编码用时	ECU 收到消息用时	刹车模块收到消息用时	时间	测试类型
2917.407	3029.407	0.2444	1.7582	1.7582	18:56:53	持续
2386.298	2498.298	0.0057	1.2059	1.2059	18:57:17	持续
1970.032	2082.032	0.0059	1.2126	1.2126	18:57:20	持续
2162.13	2274.13	0.0062	1.2559	1.2559	18:57:23	持续
2810.067	2922.067	0.0057	1.2244	1.2244	18:57:31	持续

在测试数据中：刹车总时间=信号持续时间+信号传输编码时间（信号编码用时+ECU收到信号用时+刹车模块收到信息用时），从测试结果可以打得出以下三点结论（除去个别测试操作抖动情况）：

1）按机动车制动效能标准，制动器作用耗时 τ_2 和制动力增长耗时 τ_2'' 分别为0.2 - 0.9 s和0.05 - 0.1 s之间[5]，踏板空行程耗时一般为 0.05 - 0.1 s，所以所允许的信号传输编码时间= τ_2' －踏板空行程耗时 = τ_2 － τ_2'' －踏板空行程耗时，为 0.1 ~ 0.7 s之间。而所有工况的测试结果均显示，信号传输转码用时为 0.002 ~ 0.003 s范围内，远小于所允许的 0.1 ~ 0.7 s 范围。对制动效能及效能恒定的要求达成度高。展示了所开发操控系统完全能保证操控效率要求，且远远高于一般标准。

2）持续制动测试结果显示，在信号持续时间内，信号传输稳定，不会出现任何中断或跳动现象，且信号传输转码时间基本与信号持续时间成正比，不受踏板行程影响。另外从点制动结果也可以看出，在连续多次踩踏的情况下，信号传输转码速度依然保持原有的规律，不受踩踏频率影响，展示了所开发系统操控性能的高稳定性。

3）按照机动车制动性能恒定性要求，车辆在所允许的任何工况下，都能保持灵敏、准确的操控性能[5]，本实验研究基于不同制动工况的各项测试结果显示，信号传输转码时间按照统一规律，满足各种制动工况下均能稳定发挥制动操控性能的需

求，展示了所开发系统操控性能的高实用性。

4 结束语

线控操控用计算机手段来代替机械液压手段，智能化程度、操控效率等都显著提高，但其稳定性、可靠性还是需要继续深入研究。本研究在VS.NET平台搭建的线控操控系统上，这两方面都得到了验证。但不同的操控系统，不同的通信方法，不同的解码方式均有待进一步探究，在后续研究还需从在不同温度湿度等环境因素干扰下操控性能的稳定性，以及在与动力模块、防滑模块、车身稳定模块协同控制中，进一步提高智能化程度等方面继续展开。

参考文献：

[1] 周国宪.基于多物理量的制动性能测试系统研究[D].昆明:昆明理工大学,2018.

[2] 杨甲丰.基于 CAN__FD 总线的线控制动系统设计[J].计算机次测量与控制,2021(8):9.

[3] 刘杰.基于 FlexRay 总线的汽车线控制动系统仿真研究[J].内燃机与配件,2021(10):205-206.

[4] 吴睿.基于 VS.NET 平台的第二炼钢厂生产执行系统设计与实现[D].大连:大连理工学报.2016.

[5] 余志生.汽车理论[M].北京:机械工业出版社,2009:97-99.

[6] 孟宪罗. 基于串口通信的 PTB220 气压传感器误差自动调整系统设计[J].气象科技,2017,45(5):811-817.

[7] 袁莹静.基于 VS.NET 的研究生教学管理系统的设计与实现[J].软件,2020,41(3):278-282.