第37卷 第15期 计 算 机 工 程 2011年8月
Vol.37 No.15 Computer Engineering August 2011

·工程应用技术与实现· 文章编号：1000—3428(2011)15—0232—03 文献标识码：A 中图分类号：TP311

CAN总线网络层协议栈开发测试

韩 鑫[1,2]，鲍可进[1]

(1. 江苏大学计算机科学与通信工程学院，江苏 镇江 212013；2. 上海华普汽车有限公司，上海 201501)

摘 要：对ISO15765协议进行分析，依据协议在基于MC9S12DP512芯片的整车控制器上完成CAN网络层协议栈的开发，给出开发过程。在VC环境下开发测试程序，对协议栈进行测试。结果表明，实现的CAN网络层协议栈符合ISO15765协议的规定，可满足车辆故障诊断系统等具体应用中系统对CAN网络层通信的需求。

关键词：控制局域网；网络层；协议栈；整车控制器

Development and Test of CAN Bus Network Layer Protocol Stack

HAN Xin[1,2], BAO Ke-jin[1]

(1. School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang 212013, China;
2. Shanghai Maple Automobile Co., Ltd., Shanghai 201501, China)

【Abstract】The ISO15765 protocol is analyzed, the development of CAN network layer protocol stack is achieved by the platform of vehicle control unit which is based on MC9S12DP512, meanwhile, the development process is given in detail. Under the VC environment, test procedures are developed for the test of the protocol stack. The results show that the protocol stack is in accord with ISO15765, and meets CAN network layer communications needs in variety applications, such as vehicle fault diagnosis system.

【Key words】Control Area Network(CAN); network layer; protocol stack; Vehicle Control Unit (VCU)

DOI: 10.3969/j.issn.1000-3428.2011.15.075

1 概述

CAN总线具有结构简单、成本低、可靠性高、抗干扰能力强等优点，已广泛应用于车辆通信与控制系统中，在其他工业领域也得到广泛应用[1]。ISO15765是一种基于CAN总线的车辆故障诊断协议，在其网络层协议中规定了CAN总线上不同节点间进行数据交换的网络层通信需求，为保证数据在CAN总线上的可靠传输提供了完善的网络层管理机制。除车辆故障诊断系统外，ISO15765的网络层协议可应用到其他采用CAN总线进行通信的工业系统中，满足系统对CAN网络层协议的需求[2]。

本文介绍ISO15765协议体系结构，对协议中的CAN网络层通信机制进行深入分析。在基于MC9S12DP512芯片的整车控制器上开发符合ISO15765协议的CAN网络层协议栈。在PC机的VC环境下开发协议栈测试程序与控制器进行通信，通过分析测试程序记录的数据对协议栈功能进行验证。所实现的网络层协议栈已应用到新型电容混合动力轿车故障诊断系统中，可用于实现诊断系统上位机与控制器之间的诊断通信。

2 ISO15765协议分析

CAN2.0规范在开放系统互连模型OSI上定义了CAN的物理层和数据链路层标准，国际标准化组织ISO通过对CAN2.0规范的进一步标准化制订了ISO11898协议。在实际应用中，CAN通信的实现通常需要更高层协议的支持[3]。比如ASAM组织制定的CAN标定协议CCP，SAE组织制定的J1939协议等。ISO15765协议是ISO针对车辆故障诊断系统制定的CAN高层协议。

ISO15765协议的目标是为基于CAN总线实现的车辆故障诊断系统定义一组统一的需求。如图1所示，ISO15765依据OSI模型的分层结构在ISO11898协议规定的CAN物理层与数据链路层基础之上，定义了故障诊断系统网络层通信和应用层服务相关的内容。ISO15765-3诊断服务层映射到OSI模型的应用层，规定了应用层定时参数、网络层接口、诊断服务格式等信息；ISO15765-2网络层服务映射到OSI模型的网络层和传输层[4]，规定了网络层协议数据单元N_PDU与底层CAN数据帧、应用层协议数据单元A_PDU之间的映射关系，通过网络层可对长报文进行分段传输，网络层为分段传输过程提供了时间管理、流控制和错误处理机制。

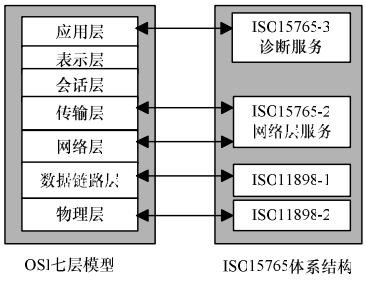

图1 ISO15765体系结构在OSI模型中的映射关系

IS015765采用分层结构，各层之间通过约定的服务接口进行数据交换，各层对其内部的操作机制进行完全的封装。这使得在进行协议栈开发时只需要按协议规定为相邻层设计服务接口，各层内部操作的实现完全不受其他层的制约，可

基金项目：国家“863”计划基金资助项目“新型电容混合动力轿车整车产品研发”(2006AA11A128)

作者简介：韩 鑫(1984－)，男，硕士，主研方向：嵌入式系统；鲍可进，教授

收稿日期：2011-03-17 **E-mail：**crazyxiaoxin@163.com

单独进行各层协议栈的开发和测试。

3 ISO15765网络层协议分析

3.1 网络层服务接口

网络层向下层传输的是网络层协议数据单元N_PDU，向应用层提供应用层协议数据单元A_PDU，一个A_PDU可映射为一个或多个N_PDU[5]。N_PDU对应于CAN帧，两者之间的对应关系由寻址方式确定。网络层负责完成N_PDU与A_PDU之间转换的组织和管理，应用层不必关心分段信息重组和分段的过程，只需访问网络层提供的接口即可请求网络层服务、获取请求服务的执行情况。如表1所示，网络层为应用层提供4类服务接口。

表1 网络层提供的服务接口及功能

服务名称	功能
N_USData.request	应用层请求网络层传输数据
N_USData.confirm	网络层通知应用层所请求的服务已执行完成/失败
N_USData_FF.indication	网络层通知应用层开始接收分段信息
N_USData.indication	网络层通知应用层单帧(分段)信息接收完成/失败

3.2 网络层数据传输方式

由于CAN数据帧最多只能传送8个字节的数据，为适应不同长度数据的传输需求，ISO15765网络层协议提供了非确信不分段传输(UUDT)和非确信分段传输(USDT) 2种数据传输机制。为实现这两种传输方式，协议中通过网络层协议控制信息N_PCI定义了4种类型的网络层协议数据单元N_PDU。

3.2.1 网络层协议数据单元N_PDU

N_PDU包含网络层地址信息N_AI、N_PCI和网络层数据N_Data三部分。N_AI存放于CAN标识符中(不需要扩展地址时)，N_PCI和N_Data存放于CAN帧的数据域中。如图2所示，不同类型N_PDU通过N_PCI第1个字节中的前4位进行区分，表中单帧数据长度SF_DL记录单帧传输数据的长度；首帧数据长度FF_DL在多帧传输中记录分段数据的总长度；连续帧序号SN保存数据分段的顺序；流控制状态FS，在分段传输中用来控制发送方数据发送的过程；N_Data存放于CAN帧数据域除N_PCI之外的其他字节中。

N_PDU名称	协议控制信息N_PCI中的字节			
	字节#1		#2	#3
	Bits 7-4	Bits 3-0		
单帧(SF)	N_PCItype=0	SF_DL	N/A	N/A
首帧(FF)	N_PCItype=1	FF_DL		N/A
连续帧(CF)	N_PCItype=2	SN	N/A	N/A
流控制帧(FC)	N_PCItype=3	FS	BS	STmin

图2 N_PCI在不同类型N_PDU中的描述

3.2.2 非确信不分段数据传输

非确信不分段数据传输即单帧传输，该方式下网络层使用单帧N_PDU传送数据。由于单帧N_PCI占用CAN帧数据域中的一个字节，单帧传送方式可传输的最大数据长度为7个字节，如果采用远程诊断方式，由于扩展地址AE需要占用数据域中的一个字节，可传输的最大数据长度变为6个字节。

3.2.3 非确信分段数据传输

非确信分段传输是指多帧传输，当网络层传送数据中的字节个数超出单帧传输方式的传送能力时，网络层将数据分段为多个N_PDU进行发送；接收时，网络层将接收到的多个N_PDU进行重组。分段数据包括一个首帧N_PDU、一个或多个连续帧N_PDU。为解决通信双方的数据同步问题，分段传输过程应当按照协议规定流控制管理机制执行。

分段传输的过程如图3所示，发送方网络层检查发送数据的长度，如果数据不能通过单帧传输方式发送，发送方向接收方发送首帧N_PDU。接收方网络层在接收到首帧N_PDU后，将其包含的FF_DL与自身网络层缓冲区大小进行比较，如果FF_DL大于接收方网络层缓冲区的容量，接收方中止接收过程并向发送方发送FS=OVFLW的流控制帧。

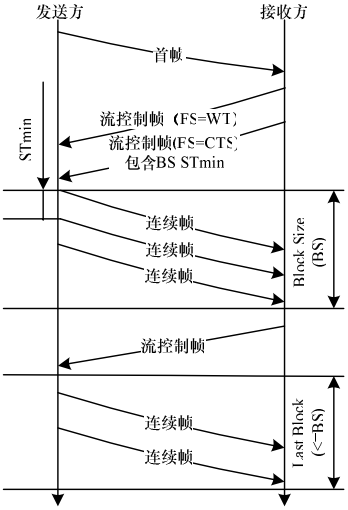

图3 分段传输示意图1

如果接收方暂时无法接收数据，接收方暂停数据接收向发送方发送FS=WAIT的流控制帧。若接收方可接收数据，向发送方发送FS=CTS的流控制帧，其中包含发送方可连续发送CF N_PDU的最大个数BS(Block Size)和发送间隔时间STmin。发送方接收到FC N_PDU后，根据N_PCI中的BS和STmin向接收方发送分段数据，并记录当前已发送CF N_PDU的个数。当发送方发送连续帧个数等于BS后暂停数据发送，等待接收方发送新的流控制信息，重复以上的操作，直到所有数据传输完成或者传输过程出错进行错误处理。接收方的数据接收能力受到网络层缓冲区容量和数据帧处理速度的限制，可通过BS与STmin进行衡量。

3.3 网络层时间管理及错误处理机制

网络层时间管理机制是为防止CAN网络上的通信节点因持续等待而被永久挂起，从而造成整个CAN网络通信能力的瘫痪。CAN网络上不同节点间进行通信时，需要根据网络层协议设置超时参数，在超时时间内未完成规定的操作时触发网络层超时错误。除超时错误外与网络层相关的错误还包括N_PCI错误、非预期N_PDU、流控制等待错误。错误处理机制在网络层发生错误时进行错误处理，保证CAN网络的正常通信。

4 ISO15765网络层协议的实现

下面介绍在CodeWarrior 4.7下使用C语言进行协议栈开发的详细过程。以新型电容混合动力轿车整车控制器为开发平台，控制器采用Freescale公司的16位控制芯片MC9S12DP512，拥有五路MSCAN控制器，14 KB的RAM和512 KB的Flash等硬件资源。外接晶振频率为16 MHz，控制器中的主程序采用前后台系统[6]。

通过以上对ISO15765网络层协议的分析，网络层协议

栈的功能可通过网络层接口、数据分段与重组、时间管理、错误处理 4 个模块实现，协议栈实现的原理如图 4 所示。其中网络层接口的实现比较简单，首先根据协议定义四类网络层服务接口的结构体类型，然后定义相应服务接口的结构体变量即可，重点讲述其他 3 个模块的实现过程。

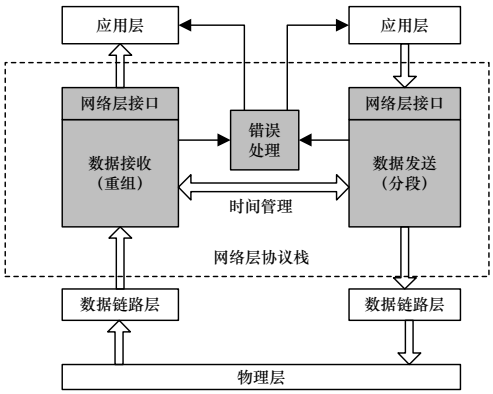

图 4 网络层协议栈实现原理

4.1 数据分段与重组

在多帧传输时，网络层负责将传输的数据进行分段和重组，分段和重组的过程中需要设置网络层缓冲区来暂时存放数据。缓冲区的设置可以考虑 2 种方案，一种是设置双缓冲区，分别用作接收缓冲区和发送缓冲区。该方案优点是数据的传输过程可实现全双工通信，通信速率、总线负载率较高，不需要单独的缓冲区管理程序；缺点是占用 RAM 空间大。另一种方案是设置单缓冲区，接收和发送过程中网络层互斥使用该缓冲区。该方案只能进行半双工通信，通信速率较低，实现过程中需要专门的程序对缓冲区进行管理；优点是系统 RAM 的占用率低。考虑到开发环境中硬件条件的限制，本研究采用单缓冲区方案。

如图 5 所示，为充分利用缓冲区空间，克服“假溢出”现象，将缓冲区设计为单向循环队列。在多帧传输过程中接收到首帧时，将首帧中的 FF_DL 与缓冲区大小进行比较，如果缓冲区的容量大于 FF_DL，允许向缓冲区中写入数据。在数据重组完成后只需要将数据在循环队列中的开始位置和数据的总长度提供给应用层，应用层便可从缓冲区中读取重组后的数据。在分段发送数据时，先比较发送数据的长度与缓冲区容量，如果数据的长度小于缓冲区容量将发送数据拷入缓冲区中，同时保存数据的开始位置与长度。启动数据分段，将数据依次从缓冲区中取出填入相应的 N_PDU 中，数据链路层根据接收到的 N_PDU 填充 CAN 帧，调用发送函数发送数据。

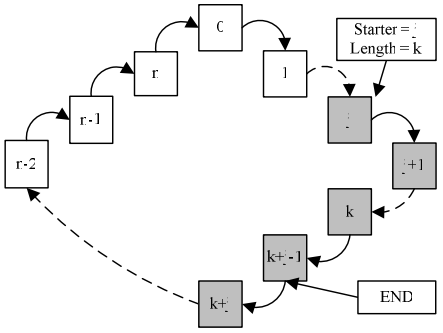

图 5 分段传输示意图 2

分段和重组的过程中互斥使用缓冲区，只有循环队列为空时才允许新的缓冲区使用请求。为适应不同应用中的数据对网络层缓冲区容量的要求，缓冲区长度采用宏定义，最大可达 4 095 Byte。

4.2 时间管理

时间管理包括延时发送和超时处理两部分功能。同一块中的连续帧发送或流控制等待帧连续发送时，连续发送的两帧之间需要一定的间隔时间，来保证接收方有足够的时间处理接收到的数据，即发送方前一帧发送结束后需要延时固定时间后再调用发送函数发送新的帧。此外，在通信过程中通信双方每发送完一帧或接收到一帧后，都需要记录时间进行超时判断。

网络层通信过程中使用系统时钟进行计时。通过配置寄存器，利用系统时钟产生一个长度为 1 ms 的实时中断基准时间(RTICTL = 0x1f)。在实时时钟中断函数中定义一个 16 位的无符号计数器进行加一操作，当需要进行延时操作时采集计数器的当前值作为计时开始时间，当计时器达到设定的时间条件后进行下一步操作。需要进行超时判断时将计时器值、超时参数与需要触发的事件是否发生一起作为发生超时错误的判断条件。

4.3 错误处理

如图 3 所示，在网络层通信过程中，发生任何类型错误时通信过程都会被中断，并转入错误处理模块进行错误处理。错误处理模块将接收到的错误指示信息与自身 case 语句中的错误类型进行匹配，匹配成功后执行相应的错误处理操作，并将错误告知应用层。错误处理完成后应当及时清空网络层缓冲区，已备网络层接收新的数据。

5 协议栈测试

为验证协议栈功能的可靠性与正确性，在 PC 机的 VC6.0 环境下开发测试程序对实现的协议栈进行测试。测试程序通过周立功公司的 USBCANI 接口卡与控制器进行 CAN 通信。测试程序的运行界面如图 6 所示，通信的整个过程可在测试程序中实时显示，通过读取显示数据分析协议栈运行的正确性与可靠性。设定上位机地址为 0x00，控制器地址为 0x10，CAN 总线的通信速率为 500 Kb/s。

图 6 测试程序界面

测试过程分为 2 个部分：正常通信功能测试和错误处理功能测试。正常通信功能测试通过单帧发送(测试程序)单帧响应(控制器)、单帧发送分段响应、分段发送分段响应、分段发送单帧响应 4 类测试实例来测试协议栈在正常通信情况下的数据分段与重组等功能是否正常。错误处理功能测试根据故障类型编写错误实例，验证网络层协议栈是否能够对错误进行识别，并通过错误处理模块对出现的错误进行容错处理，保证通信正常。

(下转第 237 页)