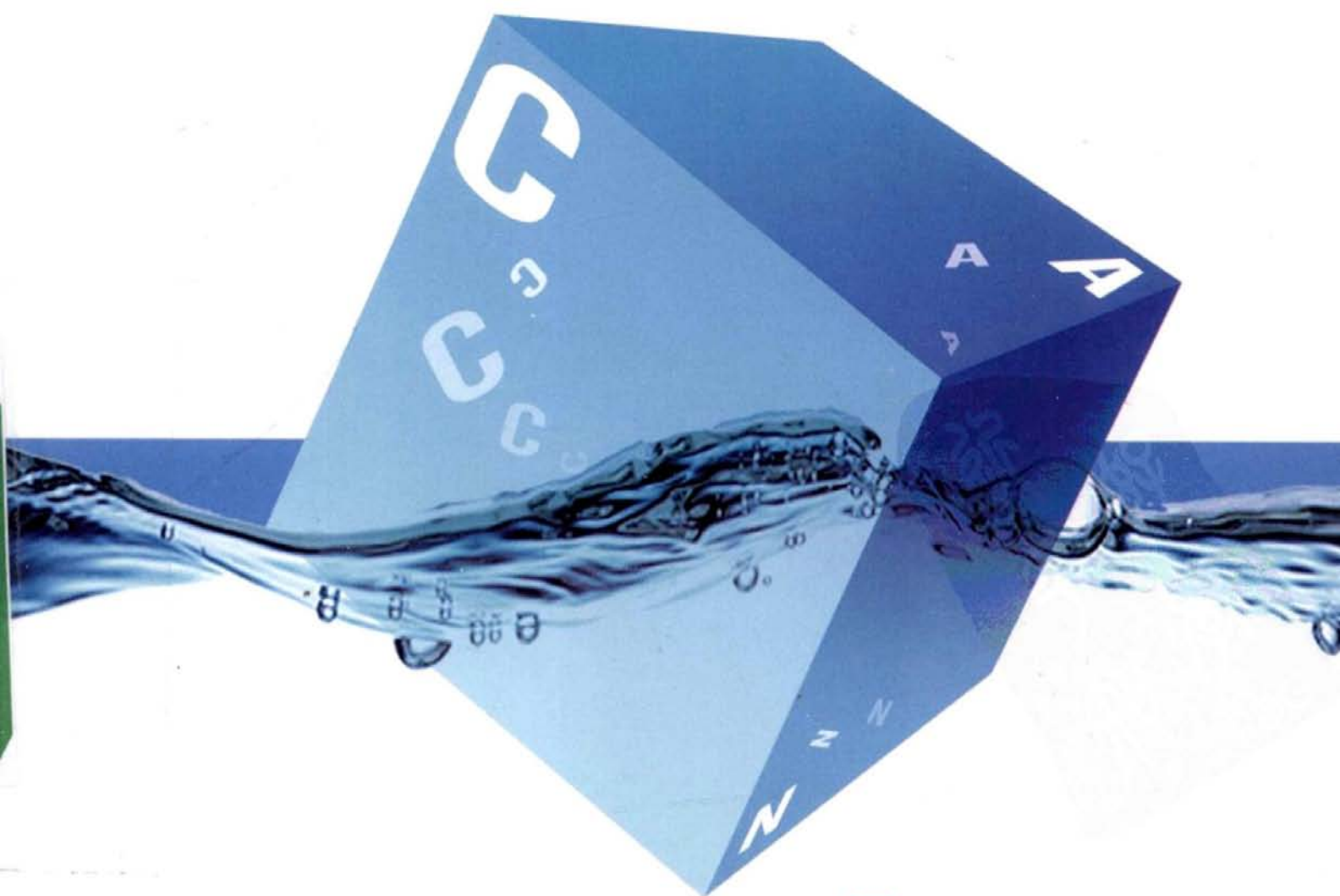


“十一五”高等院校规划教材

CAN

总线技术

杨春杰 王曙光 亢红波 编著



北京航空航天大学出版社

内容简介

本书系统地讲述了CAN总线的协议、常用器件及其使用方法, 辅以详细的实验指导和设计实例, 使读者能够逐步掌握CAN总线设备的基本设计方法。

全书共分8章, 第1章介绍了现场总线的基础知识, 第2章介绍了CAN协议, 第3章详细说明了SJA1000的原理和使用方法, 第4章简要介绍CAN总线收发器, 第5章简单介绍了几种具有CAN接口的处理器, 第6章介绍了CAN的应用层协议, 第7章是一个系统设计实例, 第8章是实验指导。

读者对象

本书系统性、实用性强, 简洁易懂, 可作为本科院校自动化、机电一体化、测控技术与仪器及其相关专业的教材, 也可供工程技术人员作入门参考。

上架建议: 计算机硬件/嵌入式系统

ISBN 978-7-81124-968-2



9 787811 249682 >

定价: 28.00元

“十一五”高等院校规划教材

CAN 总线技术

杨春杰 王曙光 亢红波 编著

北京航空航天大学出版社

内 容 简 介

本书系统地讲述了 CAN 总线的协议、常用器件及其使用方法,辅以详细的实验指导和设计实例,使读者能够逐步掌握 CAN 总线设备的基本设计方法。全书共分 8 章,第 1 章介绍了现场总线的基础知识,第 2 章介绍了 CAN 协议,第 3 章详细说明了 SJA1000 的原理和使用,第 4 章简要介绍 CAN 总线收发器,第 5 章简单介绍了几种具有 CAN 接口的处理器,第 6 章介绍了 CAN 的应用层协议,第 7 章是一个系统设计实例,第 8 章是实验指导。

本书系统性、实用性强,简洁易懂,可作为本科院校自动化、机电一体化、测控技术与仪器及其相关专业的教材,也可供工程技术人员作入门参考。

图书在版编目(CIP)数据

CAN 总线技术/杨春杰等编著. —北京:北京航空航天大学出版社,2010.2

ISBN 978-7-81124-968-2

I. C… II. 杨… III. 总线—技术 IV. TP336

中国版本图书馆 CIP 数据核字(2009)第 219205 号

© 2010,北京航空航天大学出版社,版权所有。

未经本书出版者书面许可,任何单位和个人不得以任何形式或手段复制本书内容。
侵权必究。

CAN 总线技术

杨春杰 王曙光 亢红波 编著

责任编辑 董立娟

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100191) 发行部电话:010-82317024 传真:010-82328026

<http://www.buaapress.com.cn> E-mail:emsbook@gmail.com

北京时代华都印刷有限公司印装 各地书店经销

*

开本:787×960 1/16 印张:15.25 字数:342 千字

2010 年 2 月第 1 版 2010 年 2 月第 1 次印刷 印数:4 000 册

ISBN 978-7-81124-968-2 定价:28.00 元

前 言

现场总线技术是工业控制系统发展的新方向。现场总线是工业控制与计算机网络相结合的产物,适应了分布式控制系统的发展特点,是一个开放的通信网络。现场总线技术的发展,将极大地改变控制系统的结构和功能。CAN 总线是一种很有发展前途的现场总线,已经在交通运输装备、医疗设备、先进制造系统等多个领域成功地应用。

本书适合自动控制、测控技术与仪器及相关专业本科生,也可作为技术人员学习 CAN 总线技术的入门书。作者根据近几年的教学经验,系统介绍了 CAN 总线的协议、CAN 通信控制器、CAN 总线节点的基本设计方法、CAN 简单系统的设计。本书内容简洁,并安排了专门的实验内容,可安排 32~48 学时的授课时间(包括实验)。

使用本书,需要有《计算机网络》的基础,另外需要学习过或同步学习过《单片机原理及接口技术》。实验部分所附参考程序大部分使用 C 语言开发,只作为参考,关键是掌握操作的流程。

全书分为 8 章。第 1 章介绍了现场总线的基本概念、特点和发展趋势,以及目前最具影响的几种现场总线。第 2 章介绍了 CAN 协议。第 3 章详细说明了 SJA1000 的原理和使用。第 4 章简要介绍 CAN 总线收发器。第 5 章简单介绍了几种具有 CAN 接口的处理器。第 6 章介绍了应用层协议。第 7 章介绍了一个基于 CAN 的监控系统设计。第 8 章是实验指导,包括 5 个典型实验,并附有参考程序。

其中,第 1、3、5.1、5.2、6 章(节)由杨春杰编写,第 2、5.3、7 章(节)由王曙光编写,第 4、5.4、8 章(节)由亢红波编写。杨春杰对全书进行统稿。本书在编写过程中参考了 NXP、SILICON、TI、新华龙等公司的数据手册。另外,王乔勇、张宁波、陈二阳等进行了部分实验验证和资料整理工作,特此感谢。

CAN 总线技术

由于编者学识有限,不足之处在所难免,敬请读者给予批评指正。

有兴趣的读者可以发送电子邮件到:yyccjj007@163.com,与作者进一步交流;也可以发送电子邮件到:xdhydcd5@sina.com,与本书策划编辑进行交流。

作 者

2009 年 12 月



目 录

第 1 章 现场总线技术概述	1
1.1 工业控制系统的发展	1
1.1.1 工业控制系统	1
1.1.2 现场总线的发展及定义	5
1.2 几种主要的现场总线标准	7
1.2.1 CAN 总线	7
1.2.2 Profibus 总线	8
1.2.3 LONWORKS	8
1.2.4 现场总线基金会 FF	9
1.2.5 HART 总线	10
1.3 现场总线的应用	10
1.4 现场总线技术的发展趋势	11
1.4.1 现场总线与计算机通信技术的关系	12
1.4.2 以太网与现场总线	13
1.4.3 现场总线应用工程的发展趋势	15
第 2 章 CAN 协议	16
2.1 CAN 的发展过程	16
2.1.1 CAN 起源	16
2.1.2 标准化过程	17
2.1.3 CAN 应用及前景展望	17
2.2 CAN 协议的基本定义与结构模型	19
2.3 帧结构	24
2.3.1 数据帧	24
2.3.2 远程帧	27
2.3.3 错误帧	28
2.3.4 过载帧	29

CAN 总线技术

2.3.5 帧间空间	30
2.4 错误界定及处理	30
2.4.1 错误类型	30
2.4.2 错误帧的输出	31
2.4.3 错误界定及规则	32
2.5 位定时与同步	34
2.5.1 基本概念	34
2.5.2 CAN 总线位定时与同步机制	34
第3章 SJA1000 的原理与使用	39
3.1 SJA1000 的结构与功能	39
3.1.1 概述	39
3.1.2 芯片引脚定义与说明	40
3.1.3 SJA1000 的结构及内部存储器分配	42
3.2 SJA1000 的主要寄存器	47
3.2.1 模式(控制)寄存器配置及使用方法	47
3.2.2 命令寄存器配置及使用方法	49
3.2.3 状态寄存器配置及使用方法	50
3.2.4 中断管理寄存器	51
3.2.5 总线定时寄存器配置及使用方法	52
3.2.6 输出控制寄存器	57
3.2.7 时钟分频寄存器	58
3.2.8 其他寄存器配置及使用方法	59
3.3 通信及滤波器原理	61
3.3.1 发送数据缓冲区	61
3.3.2 接收缓冲区	61
3.3.3 验收滤波器	62
3.4 SJA1000 基本功能的应用	66
3.4.1 SJA1000 典型应用接口电路	66
3.4.2 SJA1000 初始化程序设计	66
3.4.3 SJA1000 自检测	68
3.4.4 SJA1000 收发程序设计	70
第4章 常用 CAN 总线收发器	72
4.1 CAN 总线收发器 PCA82C250	72
4.1.1 概述	72

4.1.2 组成结构及功能描述	72
4.1.3 应用举例	75
4.2 高速 CAN 收发器 TJA1050	78
4.2.1 概 述	78
4.2.2 组成结构及功能描述	78
4.3 隔离 CAN 收发器 CTM1050	84
4.3.1 芯片概述	84
4.3.2 组成结构及功能描述	84
4.3.3 典型应用	86
第 5 章 具有 CAN 接口的处理器	88
5.1 C8051F040	88
5.1.1 C8051F040 的引脚	89
5.1.2 C8051F040 的 CAN 模块	92
5.1.3 CAN 寄存器配置	95
5.1.4 C8051F040 的 CAN 通信实例	96
5.2 TMS320F2812	100
5.2.1 TMS320F2812 概述	100
5.2.2 CAN 模块的结构	101
5.2.3 eCAN 配置	104
5.2.4 eCAN 中断	108
5.3 P8xC591	111
5.3.1 P8xC591 概述	111
5.3.2 P8xC591 引脚描述	112
5.3.3 P8xC591 的 CAN 模块	115
5.3.4 PeliCAN 寄存器和信息缓冲区描述	118
5.3.5 P8xC591 典型应用	122
5.4 带 CAN 控制器的 ARM 微控制器	123
5.4.1 LPC2000 系列 ARM 微控制器	123
5.4.2 LPC29xx 系列 ARM 微控制器	132
第 6 章 CAN 的应用层协议	139
6.1 简单的自定义应用层协议	139
6.1.1 标识符的分配	140
6.1.2 报文帧格式	142
6.1.3 通信实现方法	143

CAN 总线技术

6.2	CANopen 协议	143
6.2.1	CANopen 概述	143
6.2.2	CANopen 通信模型	146
6.3	DeviceNet	150
6.3.1	DeviceNet 概述	150
6.3.2	DeviceNet 报文组	151
6.3.3	对象模型	153
6.3.4	预定义主/从连接	155
第 7 章	基于 CAN 总线的监控系统设计	157
7.1	系统设计概述	157
7.2	系统网络拓扑结构及参数配置	158
7.2.1	系统网络拓扑结构	158
7.2.2	系统网络参数配置	159
7.2.3	系统通信协议	159
7.3	系统硬件设计	161
7.3.1	报警节点设计	161
7.3.2	转换模块设计	163
7.3.3	中继器模块设计	166
7.3.4	GSM 电路设计	166
7.4	系统软件设计	168
7.4.1	初始化模块设计	168
7.4.2	报警节点软件设计	170
7.4.3	CAN/RS485 模块软件设计	171
7.4.4	中继器模块软件设计	173
7.4.5	上位机软件设计	174
7.5	系统抗干扰措施	176
第 8 章	实验指导	179
8.1	实验开发平台	179
8.1.1	软件开发平台	179
8.1.2	硬件开发平台	186
8.2	课内实验	191
实验一	SJA1000 初始化实验	191
实验二	SJA1000 局部自检测实验	198
实验三	P8xC591 双节点通信实验	201

实验四 CAN 转 RS232 网桥模块的设计	204
实验五 CAN 中继器设计	207
附录 A 参考程序	209
实验一 SJA1000 初始化实验参考程序	209
实验二 SJA1000 局部自检测实验参考程序	211
实验三 P8xC591 双节点通信实验参考程序	214
实验四 CAN 转 RS232 网桥模块设计参考程序	217
实验五 CAN 中继器设计参考程序	222
附录 B CANopen 对象字典的详细结构	228
附录 C 常见调试错误分析	230
参考文献	231

第 1 章

现场总线技术概述

1.1 工业控制系统的发展

1.1.1 工业控制系统

自工业化大生产以来,为了提高生产效率,生产设备的控制操作逐步由人工手动控制发展为机器自动控制。随着认识的深入、综合科技水平的提高,工业控制也从简单到复杂、从单一设备的控制发展到控制系统。

控制系统大致经历了基地式仪表控制系统、集中式数字控制系统、集散控制系统、现场总线控制系统等几个主要阶段。每个阶段的控制系统在结构上都有明显的改进,都有一种标志性的设备。

1. 基地式仪表和继电器

20 世纪 40 年代,测控仪表和继电器进入工业生产领域。严格地说,这种控制方式还不能称为“系统”。它的规模很小,结构简单,或者所实现的功能很简单。基地式仪表主要对单台设备实现较为简单的控制,继电器控制电路主要完成顺序控制。它们是现今控制系统的前身,限于当时的技术条件,这种装置的控制精度和可靠性都不高。现在,这种“控制系统”已经难觅踪影。

2. 集中式数字控制系统

随着计算机的出现和发展,特别是早期对计算机神奇能力的超高预期,使人们自然想把它用到工业控制中去。但那时的计算机价格昂贵,性能也难以适应工业生产环境。直到 20 世纪 70 年代,随着集成电路技术的发展,研制出了微处理器和单片机。单片机侧重管理、控制功能的发展;微处理器主要发展数据处理能力。

单片机在工业控制、仪器仪表、军工、家电等诸多领域得到了广泛应用,至今不衰。但是以单片机的运算能力和速度,难以实现复杂的控制算法,不能组成大规模的控制系统;作为主要

CAN 总线技术

的控制器,它越来越不能满足实际需要。与此同时,微处理器也在快速发展,性价比不断提高,引入到工业控制系统的时机成熟了。以微处理器为核心的微型计算机经过特殊的抗干扰设计,以一种稍微特殊的形式——工控机出现在工业控制系统中。

无论是单片机,还是工控机,它们在控制系统中都是处于核心地位,系统所有的功能都由它来完成,如数据采集、数据处理、计算决策、控制输出等。典型的集中式控制系统的结构如图 1-1 所示。

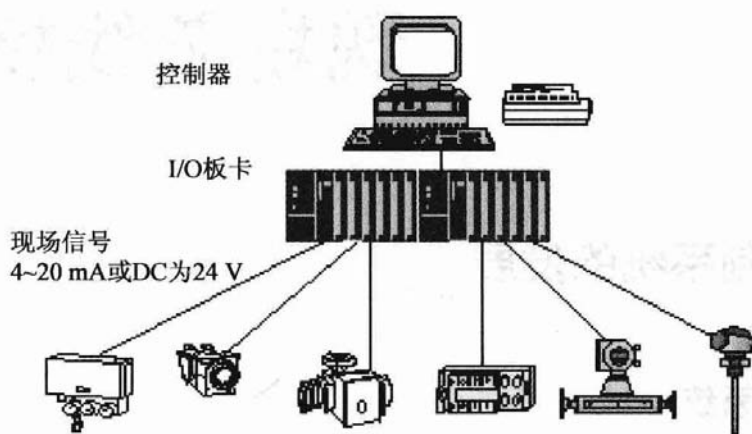


图 1-1 集中式控制系统的结构

集中式控制系统能够实现复杂的控制算法,也能达到很高的控制精度,但是它有两个主要的缺点无法克服,一是核心处理装置的负担太重,当系统规模扩大时,实时性不能保证,所以系统规模不能很大;二是系统功能集中,危险也集中,相当脆弱。

3. 集散控制系统

随着计算机技术、信号处理技术、测量技术、通信网络技术以及人机接口技术的发展,微处理器及网络器件价格大幅下降,出现了所谓的 DCS(Distributed Control System),又名分布式计算机控制系统。分布或分散是相对集中控制系统而言的,DCS 在系统结构上采用分级设计的思想,实现功能上分离,位置上分散,达到“分散控制,集中管理”的目的,对生产过程进行集中监测、操作、管理和分散控制。它既不同于分散的仪表控制,又不同于集中式计算机控制系统,具备通用性、系统组态灵活、控制功能完备、显示操作集中、人机界面友好、运行安全可靠的显著特点,对于提高生产过程的自动化水平、提高产品质量、提高劳动生产率具有重要意义。集中管理、分散控制,即管理与控制相分离,上位机用于集中监视管理,若干台下位机下放分散到现场实现相互之间的信息传递。

集散控制系统的结构模式为:操作站—控制站—现场仪表,典型的系统结构如图 1-2 所示。

DCS 系统大体可分为 3 个发展阶段:

第一阶段:1975—1980 年。这个阶段采用微处理器为基础的过程控制单元(Process Control Unit)实现了分散控制,各种控制单元具有多种控制算法,通过组态(Configuration)独

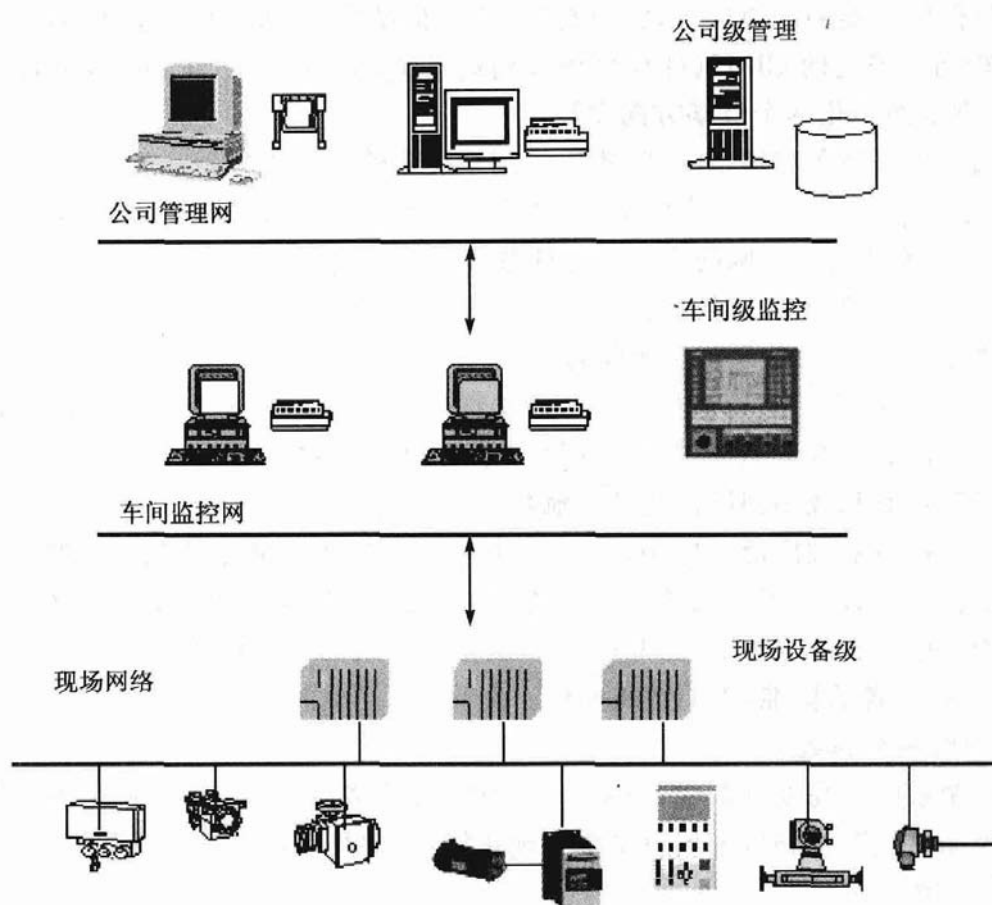


图 1-2 集散控制系统的结构

立完成回路控制；系统具有自诊断功能；在信号处理时，采用一定的抗干扰措施。操作站与过程控制单元的分离，采用冗余通信技术，将过程控制单元的信息送到操作站和上位计算机，从而实现了分散控制和集中管理。这一阶段分散控制系统在控制过程中成功地确立了地位。这一时期典型的产品有 HONEYWELL 公司的 TDC2000、FOXBORO 公司的 SPECTROM、西门子公司公司的 TELEPERM M、肯特公司的 P-4000 等。

第二阶段：1980—1985 年。这个阶段的主要技术重点表现为产品的换代周期愈来愈短。在过程控制单元增加了批量控制功能和顺序控制功能；在操作站及过程控制单元采用 16 位的微处理器，使系统性能增强；工厂级数据向过程级分散；提供更强的图画显示，报表生成和管理能力；强化系统功能，通过软件可组态规模不同的系统；强化了系统信息的管理，加强了通信系统。这一时期典型的产品有 HONEYWEL 的 TDC3000、BAILEY 的 NETWORK-90、西屋公司的 WDPF、ABB 公司的 MASTER 等。

第三阶段：1985 年以后。在这一时期中集散控制系统的技术特点是采用开放式系统网络。符合国际标准组织 ISOOSI 开放系统互联的参考模型；开发了中、小规模集散控制系

CAN 总线技术

统;采用 32 位微处理器和触摸屏等,完全实现 CRT 化操作;引入实时多用户多任务的操作系统。DCS 系统向大型化的 CIMS(计算机集成制造系统, Computer Integrated Manufacturing System)和小型及微型化两个极端方向发展。

根据采用的主要设备和通信方式,集散控制系统大致可分为如下几类:

① 模块化控制站+MAP 兼容的宽带、窄带局域网+信息综合管理系统。

② 分散过程控制站+局域网+信息管理系统。

③ 分散过程控制站+高速数据公路+操作站+上位机。

④ 单回路控制器+通信系统+操作管理站。

⑤ 编程逻辑控制器 PLC+通信系统+操作管理站,这是一种在制造业广泛应用的集散控制系统结构。现已有不少产品可以下挂各种厂家的 PLC,组成 PLC+DCS 的形式,应用于有实时要求的顺序控制和较多回路的连续控制场合。

集散控制系统目前被广泛地应用,取得了良好的效果,但是并未达到完美的程度。从结构上看,在系统的一个局部或者子系统基本上还是集中式控制,系统分散得不够彻底,集中式控制系统存在的问题没有从根本上得到解决。3 层甚至 4 层的系统结构方式,使成本较高;而且各公司的 DCS 各有各的标准,不能实现互联。

4. 现场总线控制系统

要实现控制系统的高度分散化,需要一种性能好、价格低的底层通信网络的连接现场仪表设备,称为“现场总线”。同时,现场设备要实现智能化,即具有通信、自诊断及保护、数据计算、测控输入输出等功能。

现场总线控制系统(FCS)就是用开放的现场总线网络将自动化系统最底层的现场设备互连的实时网络控制系统。它在结构上更加分散,可以分为两层,即现场控制网和管理协调网。

图 1-3 是现场总线控制系统的结构模型,图中的现场总线给出了多种通信介质,也不限定只用一个总线标准。

现场总线控制系统 FCS 在结构上与传统的集散控制系统 DCS 相比,产生了很大的变化,主要有以下几个方面:

① FCS 的信号传输实现了全数字化,从最底层的传感器和执行机构采用现场总线网络起,逐层向上直至最高层均为通信网络互联。

② FCS 系统结构是全分散式,废弃了 DCS 的输入/输出单元和控制站,由现场设备或现场仪表取代,即把 DCS 控制站的功能化整为零,分散地分配给现场仪表,从而构成虚拟的控制站,实现彻底的分散控制。

③ FCS 的现场设备具有互操作性,不同厂商的现场设备既可互连也可互换,并可以统一组态,彻底改变传统 DCS 控制层的封闭性和专用性。

④ FCS 的通信网络为开放式互联网络,既可与同层网络互联,也可与不同层网络互联,用户可极其方便地共享网络数据库。

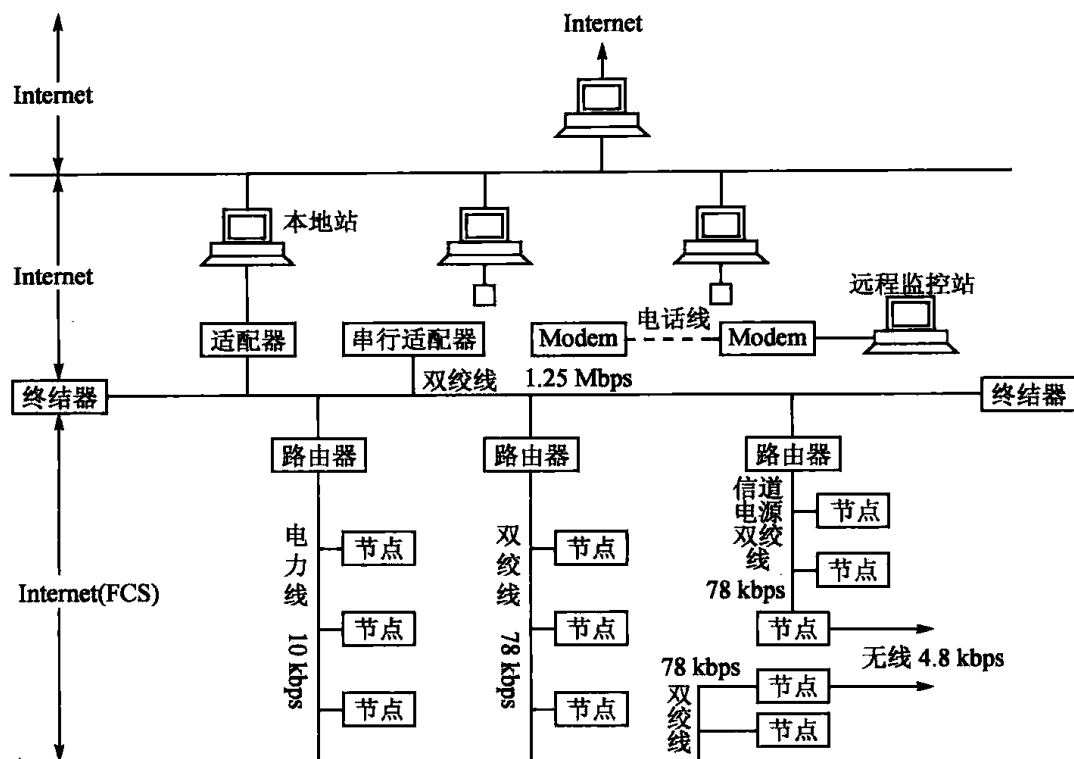


图 1-3 现场总线控制系统的结构模型

⑤ FCS 的技术和标准实现了全开放,从总线标准、产品检验到信息发布完全是公开的,面向世界任何一个制造商和用户。

现场总线控制系统的核心是现场总线。现场总线技术是计算机技术、通信技术和控制技术的综合与集成,它的出现将使传统的自动控制系统产生革命性变革:变革传统的信号标准、通信标准和系统标准,变革现有自动控制系统的体系结构、设计方法、安装调试方法和产品结构。

1.1.2 现场总线的发展及定义

工业测控设备和系统中长期采用 RS232/485 通信标准,这是一种低速率的数据传输标准,而且其协议并不完善,难以组成大规模的网络系统。控制系统的复杂性和规模的增大,如工业现场控制或生产自动化领域中需要使用传感器、控制器等分布广泛的设备,如果采用传统的星形网络拓扑结构或 LAN 组件及环形拓扑结构成本较高,所以在最底层需要设计一种造价低又适于现场环境的通信系统,这后来被称为现场总线。

1983 年, Honeywell 公司推出的智能化仪表,在 4~20 mA 直流信号上叠加了数字信号,使现场与控制室之间的信息交换由模拟信号向数字信号过渡。Rosemount 公司在此基础上制定了 HART 数字通信协议。此后的十几年间,各大公司都相继推出了各种智能仪表,基本上都是模拟数字混合仪表,它们克服了单一模拟信号仪表的技术缺陷,为现场总线的产生奠定

CAN 总线技术

了基础。

但是,不同公司的 DCS 系统不能互联。各种仪表通信标准也不统一,或者功能太简单(如 RS232、RS485),严重束缚了工厂底层网络的发展,从用户到制造商都强烈要求统一的标准,组成开放互连网络,即现场总线。

现场总线是用于过程自动化和制造自动化最底层的现场设备或现场仪表互连的通信网络,是现场通信网络与控制系统的集成。根据国际电工委员会 IEC(International Electrotechnical Commission)标准和现场总线基金会 FF(Fieldbus Foundation)的定义:现场总线是连接智能现场设备和自动化系统的数字式、双向传输、多分支结构的通信网络。现场总线是“在制造/过程现场和安装在生产控制室先进自动化装置中配置的主要自动化装置之间的一种串行数字通信链路”。

现场总线是在生产现场的测量控制设备之间实现双向串行多节点数字通信、完成测量控制任务的系统;是一种开放型的网络,使测控装置随现场设备分散化,被誉为自控领域的局域网。它在制造业、流程工业、交通、楼宇等处的自动化系统中具有广泛的应用前景。

现场总线的本质含义表现在以下几个方面:

1) 现场通信网络

现场总线的工作场所是以生产现场为主,是一种串行多节点数字通信系统。现场总线最基本的功能是连接生产现场的智能仪表或设备,一般的测量和控制功能将逐渐分散到现场的设备中来完成。采用现场总线的系统可以节约大量的电缆,通常费用较低,可以用低廉的造价组成一个系统,而且与上层网络连接的费用也不高。

2) 互操作性、互换性

不同厂家产品只要使用同样的总线标准,就能实现设备的互操作、互换,这使设备具有更好的可集成性。用户具有高度的系统集成主动权。

3) 分散功能模块

实现了现场通信网络与控制系统的集成,使控制系统在功能和地域上彻底分散化。现场设备智能化程度高,功能自治性强。

4) 通信线供电

5) 开放式互连网络

系统为开放式,可以让不同厂商将自己专长的技术,如控制算法、工艺流程、配方等集成到通用系统中去,使系统的组织更灵活、更有针对性。同时,开放式的系统给系统的升级扩容、维护检修也带来很大便利。

1984 年,ISA(美国仪表协会)最早开始制定现场总线标准。1985 年,国际电工委员会决定由 Proway working Group 负责现场总线体系结构与标准的研究制定工作。其后的 10 年间,欧美等国为主的自动化设备制造商组织制订了多个现场总线标准。从 OSI 网络模型的角度来看,现场总线网络一般只实现了第 1 层(物理层)、第 2 层(数据链路层)、第 7 层(应用层)。

1.2 几种主要的现场总线标准

按照现场总线技术的最初设想,实现开放式互连网络,并且要求设备具有互操作性、互换性,需要建立统一的现场总线技术标准,于是国际电工委员会于1984年提出了制定现场总线技术标准 IEC1158(即 IEC61158)。IEC1158 是一个面向整个工业自动化的现场总线标准。根据不同行业对自动化技术的不同需求,IEC1158 将自动化技术分为5个不同的行业。经过10余年努力,这个标准最终未获通过,取而代之的是将 IEC 61158(TS)+ Add. Protocols 作为 IEC61158 技术标准的方案;其中,Add. Protocols 包含 Control Net、PROFIBUS、P-Net、FF、Swift Net、WorldFIP 和 Interbus 等多种总线。自动化行业形成一个多种总线技术标准并存的现状。以下对几种影响力较大的现场总线做简单介绍。

1.2.1 CAN 总线

CAN 是 Controller Area Net 的缩写,即控制器局部网,是一种有效支持分布式控制或实时控制的串行通信网络。CAN 是德国 Bosch 公司为汽车的监测、控制系统而设计的,如控制发动机点火、注油及复杂的加速、刹车、抗锁定刹车系统等,已用于各种汽车上。由于 CAN 具有卓越的特性及极高的可靠性,因而非常适合工业过程监控设备互连。CAN 已经成为一种国际标准(ISO—11898),是最有前途的现场总线之一。在自动化电子领域的汽车发动机控制部件、传感器、抗滑系统等应用中,CAN 的速率可达到 1 Mbps。CAN 的信号传输介质为双绞线,具有现场总线的特点;目前,在国内的电力、石化、航天、冶金、空调等不同行业均有应用。用 CAN 做工程最大的特点就是启动成本低。

CAN 总线的特点如下:

- ① CAN 总线接口芯片支持 8 位、16 位 CPU,许多嵌入式微处理器都集成了 CAN 通信控制器。
- ② CAN 总线具有国际标准,即 ISO—11898。
- ③ CAN 可以多主方式工作,网络上任意一个节点均可以在任意时刻、主动地向网上其他节点发送信息而不分主从,通信方式灵活。利用这一特点,也可方便地构成(容错)多机备份系统。
- ④ CAN 网络上的节点可分成不同的优先级,满足不同的实时要求。
- ⑤ CAN 采用非破坏性总线仲裁技术。当两个节点同时向网络上传送信息时,优先级低的节点主动停止数据发送,而优先级高的节点可不受影响地继续传输数据,有效避免了总线冲突。
- ⑥ CAN 可以点对点、一点对多点及全局广播的方式传送和接收数据。
- ⑦ CAN 直接通信距离最远可达 10 km/5 kbps,通信速率最高可达 1 Mbps/40 m。CAN-

CAN 总线技术

BUS 上节点数据理论值为 2000 个,实际可达 110 个。

⑧ CAN 采用短帧结构,每一帧的有效字节数为 8 个。这样短的传输时间,受干扰的概率低,重新发送时间短。

⑨ CAN 节点在错误严重的情况下,具有自动关闭总线的功能,即切断它与总线的联系,以使总线上的其他操作不受影响。

⑩ CAN 每帧信息都有 CRC 校验及其他检错措施,保证了数据的出错率极低。

⑪ 通信介质采用廉价的双绞线,无特殊要求。

⑫ 用户接口简单,编程方便,很容易构成用户系统。开发系统廉价,OEM 用户容易操作。INTEL、NXP 等芯片厂商均生产具有 CAN 接口的 80C51 芯片。

1.2.2 Profibus 总线

过程现场总线——Profibus(Process Fieldbus)为德国标准,有 3 种改进型分别用于不同的场合,即 Profibus - PA、Profibus FMS、Profibus DP。

① Profibus PA 是 1994 年由 Profibus PON 用户组织提出的,引用了国际电工委员会 IEC 标准的物理链路层(ISO/OSI 模型的第一层),从而可以在有爆炸危险区域内连接本质安全型的现场仪表;它专为过程自动化而设计。

② Profibus FMS(Fieldbus Message Specification)用于车间级通用的控制及通信任务,是一个令牌环结构、实时多主网络。

③ Profibus DP 是一种高速且优化的通信方案,主要用于实现现场级控制系统与分布式 I/O 及其他现场级设备之间的通信。

Profibus 这 3 层协议使其成为能够提供制造业自动化、工程自动化、楼宇自动化以及电力自动化完整解决方案的唯一现场总线系统。

Profibus 引入功能模块的概念,不同的应用需要使用不同的模块。在一个确定的应用中,按照 Profibus 规范来定义模块,写明其硬件和软件的性能,规范设备功能与 Profibus 通信功能的一致性。

1.2.3 LONWORKS

LON 是 Local Operating Network 的缩写,即局部操作网络。LONWORKS 由美国 Echelon 公司推出,是一种功能全面的控制网络。对于工厂及车间的环境、安全、保卫、报警过程、动力分配、给水控制、库房或材料管理等,可以用 LONWORKS 组建一个综合性、分布式测控网络。

LONWORKS 的技术特点:

① 开放性。网络协议开放,对用户平等。

② 通信介质开放。可在任何通信介质下通信,包括双绞线、电力线、同轴电缆、射频电缆、

红外线等,并且多种介质可以在同一网络中混合使用。

③ 互操作性。LONWORKS 通信协议 LONTalk 是符合国际标准化组织(ISO)定义的开放系统互联(OSI)模型。任何制造商的产品都可以实现互操作。

④ LONWORKS 网络通信采用了“网络变量”,使网络通信的设计简化成为参数设置,增加了通信的可靠性。

⑤ 通信的每帧有效字节数,可以从 0~228 个字节定义。

⑥ 通信速度可达 1.25 Mbps,此时有效距离为 130 m。一个测控网络上的节点数,可达 32 000 个。直接通信距离长达 2 700 m(双绞线,78 kbps)。

⑦ LONWORKS 的基本元件叫做 Neuron(神经元芯片),这个芯片中有 3 个 8 位 CPU:第 1 个 CPU 为介质访问控制处理器;第二个 CPU 为网络处理器,处理 LONTalk 协议第 3~6 层;第 3 个 CPU 是应用处理器,执行用户编写的代码及用户的代码所调用的操作系统。Neuron 芯片具备通信与控制功能,并且固化了 ISO/OSI 的全部 7 层协议以及 34 种常见的 I/O 控制对象。

LONWORKS 目前在国内应用最多的领域是电力行业,在楼宇自动化领域,LONWORKS 也是主要的市场。

1.2.4 现场总线基金会 FF

现场总线基金会 FF(Fieldbus Foundation)有 120 多个成员,几乎覆盖了世界上著名的 DCS 与 PLC 的厂商。FF 的目标是产生一个单一的国际现场总线标准。

FF 在开放系统互联 OSI 模型基础上进行了部分改动,发布了低速总线 H1、高速总线 H2 两个标准。其中,物理层和链路层套用了 IEC/ISA 标准 IEC 1158-2。

FF 的用户层是在 OSI 模型之外增加了一个用户层,使该标准超过一项通信标准,成为一项系统标准,这也是使现场总线控制系统开放与可互操作性的关键。FF 的用户层规定了标准的“功能模块”供用户组态成为系统,不同厂商实现功能块的程序可能完全不同,但对功能块的特性描述、参数设定及相互连接的方法,却是公开统一的,即对分布在各现场设备中的分布式数据库有一个统一、公开的管理规则。

为了支持功能块模型的标准化,FF 中定义了 2 个工具:对象字典(Object Dictionary, OD)和设备描述语言(Device Description Language, DDL)。OD 是一个“基本方案”工具,用于定义字典以及设备和功能块的目录信息。设备应用的 OD 可以由设备描述(DD)来补足,而这些 DD 又是由设备描述语言(DDL)生成的。DDL 是一种解释语言、用于描述 AP(应用进程)对象到行为和操作接口。故而使众多 FF 成员生产厂家生产的设备描述方法标准化,由此,使得与生产厂家无关的设备互操作成为可能。

由 FF 设计的现场总线功能可以从自控系统最低层的简单传感器和执行器开始,直到单机自动化、系统自动化,并在上层与工业以太网功能交叉。

CAN 总线技术

FF 从自控系统的设计、安装、运行和维护的整个过程中都体现了优越性。FF 系统结构简单化,节省了控制设备。FF 高度分散性与现场设备的自治性,使系统运行性能提高,对现场设备的诊断可随时进行,使系统可靠性大大改善。FF 系统的开放性与互操作性,使得系统集成更加灵活便利,设备按功能配置,而不是按厂商配置。

1.2.5 HART 总线

HART(Highway Addressable Remote Transducer)可寻址远程传感器高速数据通道,是美国 Rosemount 公司提出的一种用于现场智能仪表和控制室设备之间的通信协议。HART 通信采用的是半双工的通信方式,特点是在现有模拟信号传输线上实现数字通信。HART 协议参照 ISO/OSI 模型的第 1、2、7 层,即物理层、数据链路层和应用层,主要有如下特征:

① 物理层:采用基于 Bell202 通信标准的 FSK 技术,即在 4~20 mA DC 模拟信号上叠加 FSK 数字信号,逻辑 1 为 1200 Hz,逻辑 0 为 2200 Hz,波特率为 1200 bps,调制信号为 ± 0.5 mA 或 0.25 V(250 Ω 负载)。用屏蔽双绞线单台距离 3000 m,而多台设备互连距离 1500 m。

② 数据链路层:数据帧长度不固定,最长 25 个字节。寻址范围 0~15,当地址为 0 时,则处于 4~20 mA DC 与全数字通信兼容状态;当地址为 1~15 时,则处于全数字通信状态。通信模式为“问答式”或“广播式”。

③ 应用层:规定了 3 类命令,第一类是通用命令,适用于遵守 HART 协议的所有产品;第二类是普通命令,适用于遵守 HART 协议的大部分产品;第三类是特殊命令,适用于遵守 HART 协议的特殊产品。另外,为用户提供了设备描述语言 DDL。

1.3 现场总线的应用

现场总线技术的应用已经非常广泛。凡是使用自动化系统或测控装置的,都将是现场总线技术的应用对象。在工业、农业、电力、公路、铁路、机场、船舶、医疗器械、国防等诸多行业,都有现场总线的存在。

纵观各种现场总线,它们一般是从某一个行业的应用开始,逐渐成熟,并推广到其他行业。在某些行业,现场总线技术占据了统治地位,几乎成为了标准的控制系统,如 CAN 总线在汽车电子中的应用。

现场总线在国内的应用比国外要滞后几年,大规模的应用还比较少,但未来的发展空间广阔。在国家“九五”、“十五”科技攻关项目或其他科技项目的支持下,国内企业完成了一批国产现场总线产品的开发,采用的现场总线有 CAN、HART、Foundation Fieldbus、LonWorks、DeviceNet、Profibus 等数种。

现场总线技术在开发应用中还存在一些问题:

(1) 现场总线技术的优势难以体现

以智能化现场仪表为基础的现场总线系统与传统系统相比,优势在自动管理和系统分散化两个方面。如果系统规模不大,管理自动化和远程诊断功能使用较少,与传统控制系统相比优点不明显,无法发挥现场总线系统降低运行维护费用的优势。而且系统规模太小,现场设备成本高,致使现场总线系统无法充分发挥现场总线节省线缆的优势。工程开销比预料得大。还可能应用的系统并不分散或者是利用原有布线进行改造,也难以体现现场总线的优势。

由于现场总线技术包含许多新的技术内容,有些现场总线技术本身也比较复杂,企业由于人才方面的原因,在使用时经常会遇到困难。

(2) 现场总线的技术问题

现场总线技术的发展面临4个主要的技术问题:

一是当总线切断时,现场总线网络断开,系统有可能产生不可预知的后果。目前,许多现场总线不能保证系统不会崩溃。对于有高可靠性要求的场合,目前只能采用双机双网络热备份的方式,尽量避免断线引起的系统崩溃。

二是在本安防爆应用中,现有的防爆规定限制总线的长度和总线上所挂设备的数量,这就限制了现场总线节省线缆优点的发挥。

三是系统组态参数过分复杂。现场总线的组态参数很多,不容易掌握,但组态参数设定得好坏,对系统性能影响很大。智能化组态软件的发展,使这个问题开始逐步解决。

四是现场总线标准的统一。现存的现场总线标准上百种,它们往往是在某个行业或领域发展成熟,进而向其他领域推广。在各自的传统应用领域,它们都有很强的生命力,但在推广时必然产生竞争。对用户来说,统一的标准当然最好,但现在看来,这是不现实的。可能的情况是随着市场竞争合并,出现几个统一的标准并存。

每种现场总线都有自己最适合的应用领域,如果能根据应用对象,将不同层次的现场总线组合使用,使系统的各部分都选择最合适的现场总线,这样既能充分展现现场总线的优越性,又能有效地降低成本。要搞现场总线的组合应用,就必须直接了解各种现场总线的技术细节和它们的最新发展,以便在组合应用中扬长避短。多种现场总线组合、智能仪表与传统模拟仪表组合的混合系统将长期存在。发展混合系统是推广现场总线技术的必要步骤。

现场总线并不是为解决传统控制系统不能解决的问题而出现的,它的主要优点是更灵活、更开放,并为采用新型的系统维护方式和企业管理模式提供了可能。因此,在以价格作为首选条件的场合,系统规模较小、控制对象分布比较集中的场合,以及没有扩展设备智能诊断和管理要求的场合,现场总线并不一定是最佳选择。

1.4 现场总线技术的发展趋势

现场总线技术应用研究的早期是标准之争,主要是 Profibus 与 FF 的竞争,标准之争的结

CAN 总线技术

果却是产生了包括 8 种现场总线的 IEC 标准。今后一段时间,竞争的焦点将是通信技术问题,具体说,是现场总线与 Ethernet 的用户层或 Ethernet 的整合方式。

1.4.1 现场总线与计算机通信技术的关系

现场总线技术最初的主要功能是将可编程逻辑控制器以一种比较简洁的方式连接起来。随着计算机技术逐渐融入 PLC,计算机通信技术也引入现场总线,从而大大增强了现场总线的功能,成为现场总线技术发展的主要趋势。

在分布式控制系统中,操作站间的通信普遍采用了局域网技术。随着半导体技术、电子技术的发展,许多操作站的功能已经可以在现场设备中实现,因此通信功能也逐渐延伸到现场。

在过程控制领域,曾经采用过许多通信协议。随着商用计算机领域的局域通信逐步被以太网垄断,过程控制领域中上层的通信也逐步统一到工业以太网。由于互联网的快速发展,通过互联网访问控制系统进行远程诊断、维护和服务从技术上成为可能,因此,TCP/IP 协议也进入过程控制领域。现场总线技术与现代计算机通信技术的结合是一项关键技术。

现代计算机通信技术具备延伸到工业现场的能力,现场总线技术在发展中也不断地融入计算机通信技术,但是计算机通信技术不会取代现场总线,因为现场总线与一般计算机通信在功能、性能指标和结构上都有很大的不同。

1. 功能的不同

计算机通信的基本功能是可靠地传递信息。现场总线专门用于控制系统,不但要求可靠地传递信息,还要及时处理并正确使用所传信息,同时尽可能降低成本。

成本是现场总线考虑的重要因素,如果总线能解决现场装置的供电问题更好。安全性主要是针对本安防爆应用。现场总线必须解决环境适应性问题,具有强抗干扰能力,包括电磁环境适应性、气候环境适应性(如耐温、防水、防尘)、机械环境适应性(如耐冲击、耐振动)等。

现场总线协议的严密性、完备性,保证网络上的设备能够互相理解所传递的信息,这些信息应该尽可能快地就地处理,从而提高系统的实时性。信息处理现场化是现场总线技术及智能仪表的重要发展趋势。

2. 性能指标的差异

我们对计算机通信系统,最关心的技术指标是传输速率。比如学习、工作中常用的互联网,我们总是关心“网速”是不是够快。对现场总线,不仅要求信息传输速度快,还要求系统响应快。

响应时间可认为是某事件发生后,从传感器检测到该事件并传输到网络上,直到执行器接收到该事件的信息并做出反应所需的时间。这个时间是由 4 个方面决定的:传感器或执行器的中断系统;信息在通信协议的应用层与物理层之间的传输时间;等待网络空闲的时间;避免信息在网络上碰撞的冲突解决时间。响应时间的长短与通信协议密切相关,大多数通信协议不能确定这个参数。过程控制系统通常并不要求这个时间达到最短,但它要求最大值是可预

估的,并在系统参数范围内。

此外,系统巡回时间也应足够短。巡回时间指系统与所有通信对象都至少完成一次通信所需的时间。过程控制系统的最长巡回时间一般根据工艺过程确定,并小于某极限值。

响应时间和巡回时间反映了系统的实时性,而实时性与通信协议有很密切的关系。现场总线采用两种技术来保证实时性:

一种是简化技术。将网络拓扑形式简化成线形;将通信模型简化为只有一、二层;将节点的信息简化到只有几比特。简化后节点的访问速度就非常快了。当然通过极大地提高通信传递速度也可以缩短节点访问时间。这时虽然理论上某些现场总线的节点访问时间还存在某种不确定性,但是反复发生不确定事件的概率很低,可以在一些非关键部位使用这种现场总线。系统的管理也可以简化为主—从方式轮询访问,只要限制网络轮询的规模,就可以将响应控制在指定的时间内。采用这种技术可大大降低总线的成本,大多数位式现场总线采用这种技术。

另一种是采用网络管理和数据链路调度技术来保证实时性。一般认为,分时式实时系统的响应具有可预知性,但资源利用率低;抢先式实时系统资源利用率高,但往往响应具有不可预知性。现在的现场总线往往采用两者结合的方式进行管理和调度,以达到效率与实时性的相对平衡。

改善现场总线的实时性、减少响应时间的不确定性,是现场总线系统研究的重要问题。

3. 网络结构的不同

计算机通信系统的结构是网状的,从一点到另外一点的通信路径可以是不固定的。大部分现场总线的结构是线状的,从一点到另外一点的通信路径是比较固定的。

线状结构的优点是:① 解决网络供电比较容易;② 解决本安防爆比较容易;③ 使通信协议中可以舍去与路径有关的几层,有利于改善实时性。

在线状结构时,一条现场总线支路的电源负载是确定的,沿总线电源电压的变化也是可以预料的。在网状结构中一定会出现多电源供电情况、各电源的负载平衡以及网络中各节点处的电压下降等难以预料的情况。

线状结构的明显缺点是当总线电缆切断后,网络联系就断开了,甚至导致瘫痪。这个问题是现场总线至今未用在最关键场合的主要原因之一,也是现场总线技术近期要研究解决的问题之一。

上述比较表明,现场总线不只是一项通信技术,也是通信技术、仪表智能化技术及自动控制技术的结合产物。虽然并不是所有的现场总线都满足了上述要求,但这些要求是用于过程控制的现场总线所追求的目标。虽然已经有一些直接通过因特网访问现场仪表的例子,但这都是一些对控制和实时性没有严格要求的检测系统或者是一些特殊的应用,没有普遍性。

1.4.2 以太网与现场总线

计算机通信网络对现场总线的影响主要体现在以太网技术。以太网技术的快速发展,逐

CAN 总线技术

渐渗透到工业控制管理系统中,形成了工业以太网。

工业以太网的核心采用以太网技术,与 IEEE802.3/802.3u 兼容,使用 ISO 和 TCP/IP 通信协议。工业以太网采取了一些特殊技术,比如双机双网冗余技术,以应对严酷的工业环境,确保工业应用的安全可靠。

以太网与现场总线之间有许多不同。物理层上:传输介质、连接件、总线供电及本安防爆功能、编码方式、传输速率等方面都存在较大不同。在媒体访问与控制方式方面,以太网采用的 CSMA/CD(载波监听多路访问/冲突检测)方式,不能满足工业网络通信的实时性和确定性要求。信息传输效率上,现场总线一般比以太网略好。

但由于以太网的标准统一,并且在办公、商务等网络应用中取得了极大成功,在工业自动化领域推广也是很自然的事。实际上,以太网已成功的应用于工业自动化诸多方面,如车间级生产信息集成及管理信息层网络;SCADA 系统:特别是一些区域广泛、含有计算机广域网技术、无线通信技术的 SCADA 系统,如城市供水或污水管网的 SCADA 系统、水利水文信息监测 SCADA 系统等;个别要求高可靠性和一定实时性的分布式控制系统也有采用以太网+TCP/IP 技术,并获得很好的效果,如水电厂的计算机监控系统。

以太网在工厂自动化管理层和车间监控层已得到广泛应用和用户认可,在设备层对实时性没有严格要求场合也有应用;如果以太网要全面进入工厂底层成为设备连接的主要网络技术,那么,以太网必须作出技术改进。

然而用一种技术覆盖各行业所有不同需求是困难的,或许也没有这个必要。把以太网和 TCP/IP 技术整合进来是现场总线今后发展的一个主要技术问题,而如何整合以太网与现场总线,对各种现场总线未来的发展是很关键的。

以太网与现场总线整合的方案分 3 种:包裹法、网关和代理服务器法、重建法。

包裹法的典型代表是 CIP、HSE 和 Modbus-TCP/IP。整合的方法是:将几种保持原样的现场总线报文作为“用户数据”嵌入 TCP/UDP 的数据帧,然后在 Ethernet 上发送。其优点是不需要花时间制定新的规范,不同现场总线的数据可以同时传输。缺点是报文的效率低,更适合传送长数据。例如,CIP 就是将 ControlNet、DeviceNet 和 Ethernet/IP 报文加上表示是哪一种现场总线的报文,由于总线报文通常很短,常常需要填入无效内容,降低报文效率。

InterBus 采用反向包裹法。它是将 TCP/IP 数据帧嵌入 InterBus 报文。由于 InterBus 报文短,TCP/IP 数据帧被拆开传送。这样做的优点是 InterBus 设备可以不受影响地用在现场,正常运行时的现场总线短报文信息传输效率大大提高。需要拆开传送的长报文主要用于传送程序和组态信息,这种传送出现的几率较低,而且对实时性的要求也低。

网关和代理服务器法是传统的跨协议转换方法,在现场总线领域中的典型代表是 Profi-Net。通过网关或代理服务器进行 Ethernet 与现场总线的信息转换的优点是已经在用的现场总线设备可以不受限制地用下去;缺点是不同现场总线同时使用时系统组态很复杂,实时性受到一定影响。

重建法是在 Ethernet 协议中建立新的实时通信用户层,从而建立以 Ethernet 为基础的现场总线,典型代表是 IDA(Interface for Distributed Automation)。

1.4.3 现场总线应用工程的发展趋势

(1) 充分发挥现场总线的优势

现场总线系统最明显优点是降低系统投资成本和减少运行费用。按照这一基本思想,在进行总线类型的选择和网络设计时,就会有明确的方向。在应用中合理地使用现场总线、充分发挥它的潜能。

(2) 不同类型的现场总线组合更有利于降低成本

按照数据格式,现场总线大致可分为位式、字节式和数据流总线 3 种类型;针对不同的情况选用不同的总线可以最大限度地降低系统成本。位式总线的能力很有限,不可能作为大型系统的信息传输主体,但是它在成本、速度等方面的优势又是其他高层次总线无法替代的,因此,常常与其他总线混合使用。

(3) 现场总线的本质是信息处理现场化

一个控制系统,无论是采用 DCS 还是采用现场总线,系统需要处理的信息量是差不多的。实际上,采用现场总线和智能仪表后,可以从现场得到更多的诊断、维护和管理信息。现场总线系统的信息量大大增加了,而传输信息的线缆却大大减少了。这就要求一方面要大大提高线缆传输信息的能力、减少多余信息的传递。另一方面要让大量信息在现场就地完成处理,减少现场与控制机房之间的信息往返。

如果仅仅把现场总线理解为省掉了几根电缆,是没有理解到它的实质的。信息处理的现场化才是智能化仪表和现场总线所追求的目标,也是现场总线不同与其他计算机通信技术的标志。

(4) 网络的设计

减少信息的往返传递是现场总线系统中网络设计和系统组态的一条重要原则。减少信息往返常常可带来改善系统响应时间的好处。因此,网络设计时应优先将相互间信息交换量大的节点,放在同一条支路里。

(5) 系统组态傻瓜化

目前,现场总线系统的组态是比较复杂的,需要组态的参数多,各参数之间的关系比较复杂,如果不是对现场总线非常熟悉,很难将系统设置到最佳状态。

第 2 章

CAN 协议

CAN 是 ISO 国际标准化的串行通信协议,是国际上应用最广泛的现场总线之一。与一般的总线通信相比,CAN 的数据通信具有突出的可靠性、实时性和灵活性的特点。

CAN 总线最初作为汽车环境中的微控制器通信,形成汽车电子控制网络。因其卓越的性能、极高的可靠性、独特灵活的设计和低廉的价格,现已广泛应用于工业现场控制、机器人、医疗仪器、环境监测等众多领域。

2.1 CAN 的发展过程

2.1.1 CAN 起源

在汽车产业中,出于对安全性、舒适性、方便性、低公害、低成本考虑,于是开发了各类电子控制系统,由于这些系统之间通信所用的数据类型及对可靠性的要求不尽相同,所以多条总线并存的情况很多,线束的数量也随之增加,没有一种现成的网络方案能够完全满足汽车工程师们的要求。为适应“减少线束的数量”、“通过多个 LAN 进行大量数据的高速通信”的需要,早在 1980 年初,Bosch 公司的工程师开始论证串行总线用于客车系统的可行性。在 1983 年初,Uwe Kiencke 开始研究一种新的串行总线,新总线的主要方向是增加新功能以及减少电气连接线,使其能够用于产品而非用于驱动技术。来自 Mercedes - Benz 的工程师较早制定了总线的状态说明,而 Intel 也准备作为半导体生产的主要厂商参与制定。德国 Applied Science 大学教授 Wolfhard Lawrenz 博士为新网络方案起名——Controller Area Network。

1986 年 2 月 CAN 诞生了。在该年的汽车工程人员协会年会上,Bosch 公司提出了新总线系统,称为汽车串行控制器局域网。Uwe Kiencke Siegfried Dais 和 Martin Litschel 分别介绍了这种多主网络方案,此方案基于非破坏性的仲裁机制,能够确保高优先级报文的无延迟传输,并且不需要在总线上设置主控制器。此外,Bosch 公司的 Wolfgang Borst、Wolfgang Botzenhard、Otto Karl、Helmut Schelling 和 Jan Unruh 提出并实现在 CAN 中的多种错误检

测、处理机制。该错误检测包括自动断开故障节点功能,以确保能继续进行剩余节点之间的通信传输。根据报文的内容识别节点地址,同时用于识别报文的标识符也规定了该报文在系统中的优先级。

1987 年中期 Intel 交付了首枚 CAN 控制器 82526,这是 CAN 方案首次通过硬件实现,仅仅用了 4 年的时间设想就变成了现实。不久,NXP 半导体推出了 82C200。这两枚最先的 CAN 控制器在验收滤波和报文控制方面有许多不同:一方面,由 Intel 主推的 FullCAN 比由 NXP 主推的 BasicCAN 占用较少的 CPU 时间;另一方面,FullCAN 器件所能接收的报文数目相对受到限制。如今的 CAN 控制器中,以同样的模式执行不同概念的验收滤波和报文处理。

今天,在欧洲几乎每一辆新客车均装配有 CAN 局域网,同样 CAN 也用于其他类型的交通工具,从火车到轮船或者用于工业控制,CAN 已经成为全球范围内最重要的总线之一,甚至领导着串行总线在 1999 年接近 6 千万个 CAN 控制器投入应用,2000 年市场销售超过 1 亿个 CAN 器件。

2.1.2 标准化过程

1990 年初,Bosch CAN 规范 CAN 2.0 版并提交给国际标准化组织。

1993 年 11 月正式出版了 CAN 的国际标准 ISO11898,这里应一些主要的法国汽车厂商要求增加了“交通工具局域网”(VAN,Vehicle Area Network)内容,并且也规定了物理层的波特率最高至 1 Mbps。目前,ISO—11898 追加新规约后,成为 ISO11898—1 新标准,是 CAN 高速通信标准。

1995 年在国际标准 ISO—11519—2 中规定了 CAN 数据传输中的容错方法。ISO—11519 是通信速度为 125 kbps 以下的 CAN 低速通信标准。

令人遗憾的是,所有出版的 CAN 规范均包含错误或者不完整,因此,为避免 CAN 应用出现不兼容,Bosch 公司一直在进行验证 CAN 芯片是否基于 Bosch CAN 参考模型。此外德国 Braunschweig/Wolfenbüttel 的 Applied Science 大学近年来根据国际标准测试规范 ISO—1684,对 CAN 的一致性进行测试。

CAN 规范正处在标准化过程中。ISO—11898—1 描述了 CAN 数据链路层;ISO—11898—2 定义了非容错 CAN 物理层;ISO—11898—3 规定了容错 CAN 物理层;国际标准 ISO—11992(卡车和拖车接口)和 ISO—11783(农业和森林机械)都在美国标准 J1939 的基础上定义了基于 CAN 应用的子协议,但并不完整。

2.1.3 CAN 应用及前景展望

CAN 协议已经有 15 年的历史,但它仍处在改进之中。为促进 CAN 以及 CAN 协议的发展,1992 在欧洲成立了 CiA(CAN in Automation)。在 CiA 的努力推广下,CAN 技术在汽车

CAN 总线技术

电子控制系统、电梯控制系统、安全监控系统、医疗仪器、纺织机械、船舶运输等方面均得到了广泛的应用。现已有 400 多家公司加入了 CiA, CiA 已成为全球应用 CAN 技术的权威。

生产 CAN 模块集成器件的 15 家半导体厂商主要面向汽车工业。1990 年中期起 Infineon 公司和 Motorola 公司已向欧洲的客车厂商提供了大量的 CAN 控制器。1990 年后期远东的半导体厂商也开始提供 CAN 控制器。1994 年, NEC 推出了 CAN 芯片 72005。从 1992 年起 Mercedes-Benz 奔驰开始在他们的高级客车中使用 CAN 技术: 使用电子控制器通过 CAN 对发动机进行管理, 使用控制器接收人们的操作信号, 这就使用了 2 个物理上独立的 CAN 总线系统, 它们通过网关连接。继 Volvo Saab Volkswagen BMW 之后, Renault 和 Fiat 也开始在他们的汽车上使用 CAN 总线。近几年内, 美国和远东的汽车厂商将在他们所生产汽车的串行部件上使用 CAN。

CAN 在全球市场上仍然处于起始点。CAN 总线在组网和通信功能上的优点以及它的高性能价格比, 决定了它在许多领域都有广阔的应用前景和发展潜力。大型仪器设备系统复杂, 对多种信息进行采集、处理、控制、输出等操作。如医疗器械 CT 断层扫描仪, 为保证其可靠工作, 在数据通信上要求功能块间可随意进行数据交换, 通信能以广播方式进行, 具有低成本的硬件接口, 通信线尽量少, 抗干扰能力强, 可靠性高并能自动进行故障识别和自动恢复。但是, 这些要求长时间未能得到很好的解决, 直至 CAN 总线技术的出现才提供了一个较好的解决方法。测控系统中离不开传感器, 由于各类传感器的工作原理不同, 其最终输出的电量形式也各不相同, 为了便于系统连接, 通常要考虑将传感器的输出变换成标准电压或电流信号。即便是这样, 在与计算机相连时, 必须增加 A/D 环节。如果传感器能以数字形式输出, 就可以方便地与计算机直接相连, 从而简化系统结构, 提高精度。这种传感器与计算机相连的总线可称为传感器总线。实际上传感器总线仍属于现场总线, 关键的问题在于如何将总线接口与传感器一体化。在广泛的工业控制领域, CAN 总线可作为现场设备级的现场总线, 与其他总线相比, 具有很高的可靠性和性价比。这必将是 CAN 技术开发应用的一个主要方向。在以往的国内测控领域, 由于没有更好的选择, 大多采用 BITBUS 或 RS485 作为通信总线, 其不足主要有: 一主多从, 无冗余; 数据通信为命令响应, 传输率低; 错误处理能力弱。采用 CAN 总线技术后即可解决上述问题。CAN 网络上任何一个节点均可作为主节点主动地与其他节点交换数据; CAN 网络节点的信息帧可以分出优先级, 这对于有实时性要求的控制提供了方便; CAN 的物理层及数据链路层有独特的设计技术, 使其在抗干扰以及错误检测等方面的性能均大大提高。CAN 的上述特点使其成为诸多工业测控领域中首选的现场总线之一。

另外, 大量潜在的新应用正在出现, CAN 不仅可用于客车, 也可用于家庭消费, 同时结合高层协议应用的特殊保安系统对 CAN 的需求也正在稳步增长。德国专业委员会 BIA 和德国安全标准权威 TÜV 已经对一些基于 CAN 的保安系统进行了认证, CANopen-Safety 是第一个获得 BIA 许可的 CAN 解决方案, DeviceNet-Safety 也会马上跟进。全球分级协会的领导者之一 (Germanischer Lloyd) 正在准备提议将 CANopen 固件应用于海事运输领域。在其他

事务中,规范定义了自动切换,可以将 CANopen 网络转换为冗余总线系统。据报道,CAN 技术已应用于家用电器和智能楼宇以及小区建设中,如安防系统、抄表系统、家电控制等。它投资少,每个节点可以随机访问,通信速度完全满足要求,且在这类应用中数据交换量都很少。适当的网关如 CAN 与 TCP/IP 协议的转换,可以使一个居室或一栋大楼的现场 CAN 信息转变为 Internet 的形式外传,或反过来通过这类网关把外部网传来的信息转换为 CAN 的形式,此即实现了所谓的远程控制。总之,CAN 的应用前景很广泛,被誉为“最有发展前景”的现场总线之一。

2.2 CAN 协议的基本定义与结构模型

正如人的语言交流一样,通信也是一种交流,是实体之间的交流;而通信协议就是规定交流形式是什么,交流符号是什么,怎么理解等。CAN 技术规范就是一种通信协议,即规定 CAN 节点之间如何完成通信。

1991 年,BOSCH 公司发布 CAN 2.0 规范。CAN2.0 规范分为 CAN 2.0A 与 CAN2.0B。CAN2.0A 支持标准的 11 位标识符,CAN2.0B 同时支持标准的 11 位标识符和扩展的 29 位标识符。CAN2.0 规范的目的是在任何两个基于 CAN-bus 的仪器之间建立兼容性,CAN2.0B 规范完全兼容 CAN2.0A 规范。由前所述,CAN 协议经 ISO 标准化后有 ISO11898 标准和 ISO—11519—2 两种标准,ISO—11898 和 ISO—11519—2 标准对于数据链路层的定义相同,但物理层不同。本章主要以 CAN2.0B 规范为主要讲解内容,同时也会涉及标准化的 CAN 协议。

CAN 的 ISO/OSI 参考模型的分层结构如图 2-1 所示。

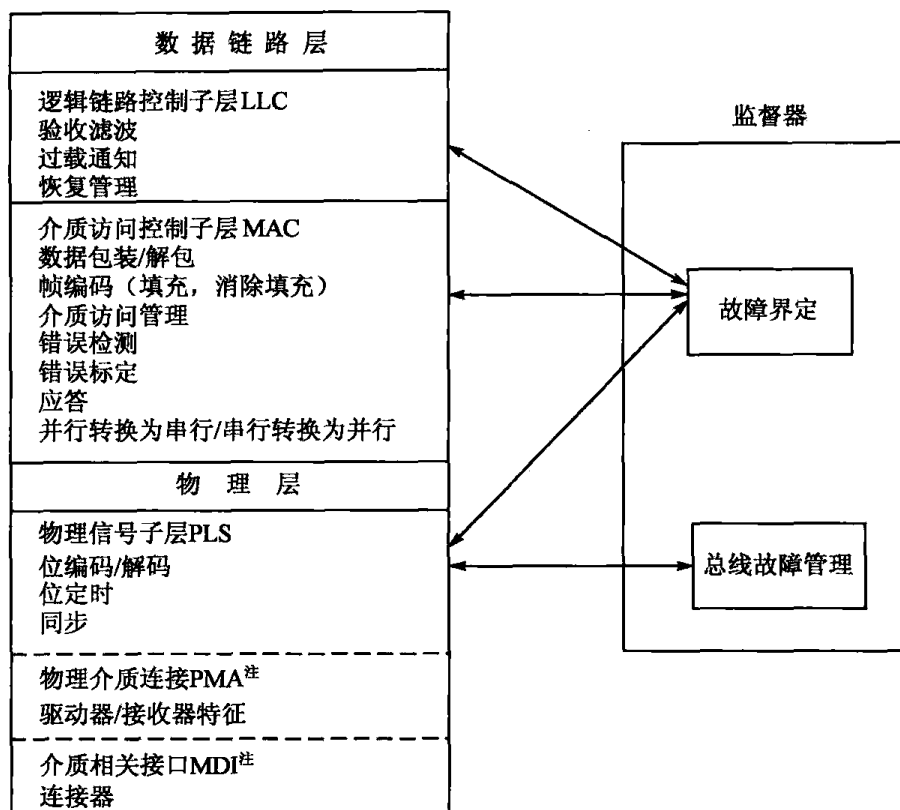
CAN 协议结构划分为两层:数据链路层和物理层。数据链路层又划分为逻辑链路控制子层和介质访问控制子层。物理层可分为物理信号子层 PLS、物理介质连接 PMA、介质相关接口 MDI。下面分别介绍相关子层功能。

1. 物理层

物理层定义信号是怎样进行传输的,因此涉及位定时、位编码/解码、同步的描述。CAN2.0 规范没有定义物理层的驱动器/接收器特性,以便允许根据它们的应用,对发送媒体和信号电平进行优化。ISO—11898 和 11519—2 对物理信号子层 PLS 的定义与 CAN2.0 规范是相同的,但其对物理介质连接 PMA、介质相关接口 MDI 进行了定义,并有所不同。物理层分为 3 层:

- 物理信号子层 PLS: 实现位编码/解码、定时和同步等功能。CAN 位流根据“不归零”NRZ 方式来编码。
- 物理介质连接 PMA: 实现总线发送/接收的功能电路并提供总线故障检测方法。
- 介质相关接口 MDI: 实现物理介质与媒体访问单元之间的机械和电气接口。

CAN 总线技术



注：CAN2.0B 规范对物理层中的驱动器、收发器、连接器、电缆等的形态没有规定，但 ISO11898 和 11519-2 对物理层的 PMA 层及 MDI 层都有定义，内容不相同。

图 2-1 CAN 的 ISO/OSI 参考模型的分层结构

(1) CAN 总线的位数值表示

CAN 的总线数值为两种互补逻辑数值：“显性”或“隐性”。“显性”(Dominant)数值表示逻辑“0”，而“隐性”(Recessive)表示逻辑“1”。CAN 能够使用多种物理介质，如双绞线光纤等，最常用的就是双绞线。信号使用差分电压传送，两条信号线称为 CAN_H 和 CAN_L。以 ISO11898 标准为例，其表示如图 2-2 所示。

以 $V_{\text{can-H}}$ 、 $V_{\text{can-L}}$ 和 V_{diff} 分别表示 CAN_H、CAN_L 和两者之间压差，其值如表 2-1 所列。

表 2-1 总线位的数值表示

名 称 \ 条 件	隐 性			显 性		
	Min	Nom	Max	Min	Nom	Max
$V_{\text{can-H}}/V$	2.00	2.50	3.00	2.75	3.50	4.50
$V_{\text{can-L}}/V$	2.00	2.50	3.00	0.50	1.50	2.25
V_{diff}/V	-0.5	0	0.05	1.5	2.0	3.0

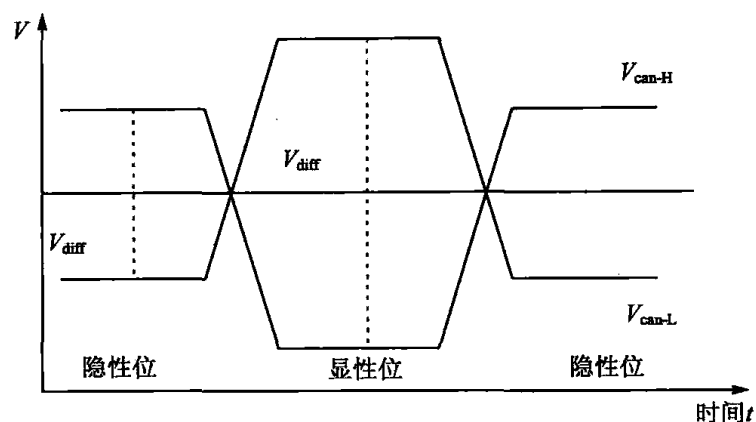


图 2-2 总线位的数值表示

(2) 最大传输距离与位速率

CAN 总线上任意两节点之间的最大传输距离与其速率的关系如图 2-3 所示。不同的系统,位速率不同。但在确定系统里,位速率是唯一且固定的。

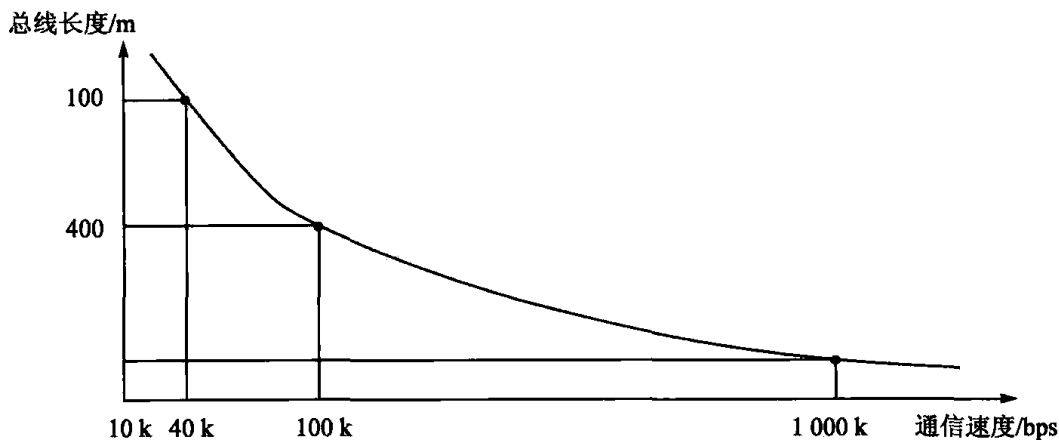


图 2-3 通信速率和最大总线长度

(3) 驱动器的选择

ISO—11898 与 ISO—11519—2 的物理层规格有所不同,每种规格需要有专门的驱动器与之相对应。表 2-2 列出了 ISO—11898 及 ISO—11519—2 所对应的主要驱动 IC。

表 2-2 ISO—11898 及 ISO—11519—2 所对应的驱动 IC

驱动 IC	ISO—11898	ISO—11519—2
	HA13721RPJE(RENESAS) PCA82C250(NXP) Si9200(Siliconix) CF15(Bosch) CTM1050(NXP)	PCA82C252(NXP) TJA1053(NXP) SN65LBC032(Texas Instruments)

CAN 总线技术

2. 数据链路层

(1) 逻辑链路控制子层 LLC

逻辑链路控制子层 LLC(Logical Link Control)的功能包括验收滤波、过载通知、恢复管理;为数据传送和远程数据请求提供服务,确认由 LLC 子层接收的报文实际已被接收;并为恢复管理和通知超载提供信息。在定义目标处理时,存在许多灵活性。

① 验收滤波:帧内容由标识符命名,标识符不能指明帧的目的地,但会描述数据的含义,每个接收器通过帧验收滤波确定此帧是否被接收。

② 过载通知:若接收器由于内部原因要求延迟下一个数据帧/远程帧,则发送过载帧。

③ 恢复管理:发送期间,对于丢失仲裁或被错误干扰的帧,LLC 子层具有自动重发功能,在成功发送完成之前,帧发送服务不被用户认可。如果不再出现新错误,则从检测出错误至下一报文的传送开始为止,恢复时间最多为 29 个位的时间。

(2) 介质访问控制子层 MAC

介质访问控制子层 MAC(Medium Access Control)的功能主要是传送规则,即控制帧结构、执行仲裁、错误检测、出错标定和故障鉴定。MAC 层还要确定是否马上开始接收、是否为新的发送开放总线等。MAC 子层不存在修改的灵活性。MAC 子层有“故障界定”(Fault Confinement)的逻辑电路,此故障界定为自检机制,以便把永久故障和短时扰动区别开来。

1) 功能描述

MAC 子层划分为完全独立工作的两个部分,即发送和接收部分,功能如图 2-4 所示。

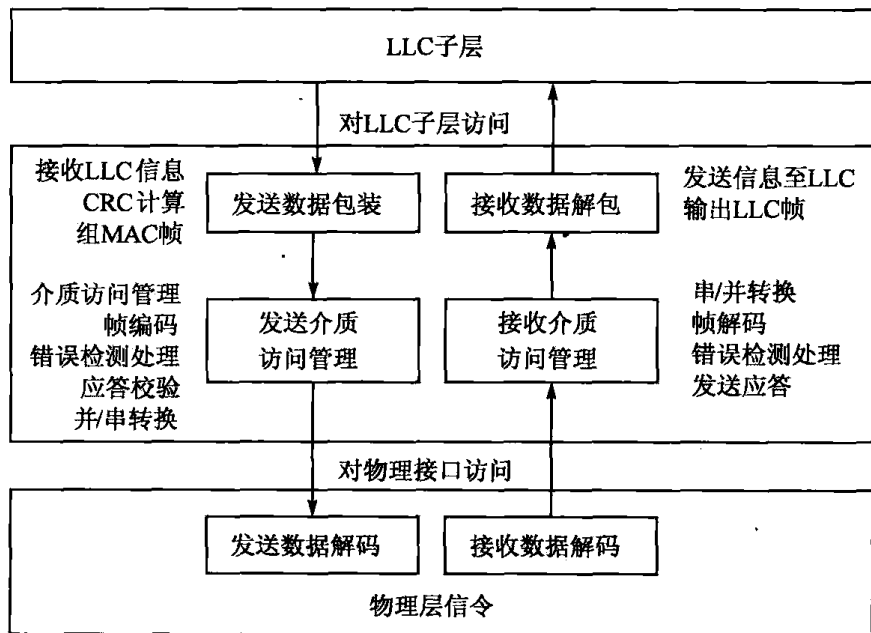


图 2-4 MAC 子层功能

2) MAC 帧编码

当发送器在发送位流中(帧起始、仲裁域、控制域、数据域和 CRC 序列)检测到 5 个数值相同的连续位(包括填充位)时,它在实际发送位流中,自动插入一个补码位,如图 2-5 所示。

未填充位流	100000xxx	011111xxxx
填充位流	1000001xxx	0111110xxx

其中, xxx 为 0 或者 1

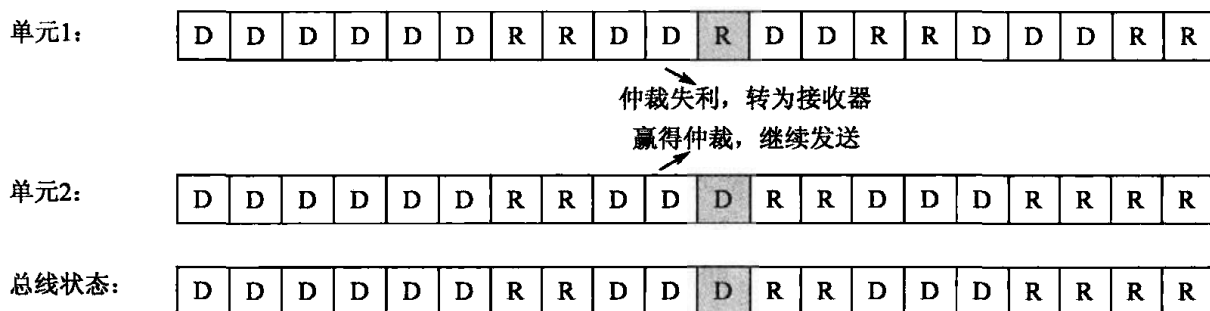
图 2-5 位填充

位填充对于发送单元和接收单元的工作是不同的:对于发送单元,在发送数据帧和远程帧时,SOE~CRC 段间的数据,相同电平如果持续 5 位,则在下一个位(第 6 个位)要插入 1 位与前 5 位反型的电平。对于接收单元,在接收数据帧和远程帧时,SOE~CRC 段间的数据,相同电平如果持续 5 位,需要删除下一个位(第 6 个位)再接收。如果这个第 6 个位的电平与前 5 位相同,则视为错误并发送错误帧。

3) 介质访问管理

如果总线处于空闲,则任何单元都可以开始发送报文。若是两个或两个以上的单元同时开始传送报文,那么就会有总线访问冲突,可以通过使用标识符的位仲裁形式可以解决这个冲突。标识符是 CAN 报文中所包含的信息,不指明报文的目的地,而系统中其他节点可以识别该标识符含义并判断是否接收该报文。

仲裁期间,每一个发送器都对发送位的电平与被监控的总线电平进行比较。如果电平相同,则这个单元可以继续发送。如果发送的是一个“隐性”电平而监控到一个“显性”电平,那么该单元就失去了仲裁,必须退出发送状态。仲裁过程如图 2-6 所示。



注: D 代表显性位, R 代表隐性位。

图 2-6 仲裁过程

所以,在总线访问期间,标识符定义静态的报文优先权,即标识符值越小的优先权越高。总线空闲时,任何单元都可以开始传送报文,具有较高优先权报文的单元可以获得总线访问权,这也是 CAN 的特点之一,即多主机(Multimaster)通信。

由此也可以得出 CAN 总线的信息路由方式(Information Routing):在 CAN 系统里,报

CAN 总线技术

文的寻址内容由其自带的标识符指定,网络上所有节点可以通过报文滤波确定是否应对该数据做出响应,所以节点不使用任何关于系统配置的信息,具有以下特点:

- ① 不需要应用层以及任何节点软硬件的任何改变,可以在 CAN 网络中直接添加节点。
- ② 由于报文滤波的作用,任何数目的节点都可以接收报文,并同时对此报文做出反应。
- ③ 网络里确保报文同时被所有的节点接收(或无节点接收)。系统的数据连贯性是通过多点传送和错误处理原理来实现的。

理论上,CAN 串行通信链路可以连接无数多的单元。但由于实际上受延迟时间以及总线线路上电气负载的影响,连接单元的数量是有限的。

2.3 帧结构

总线上的信息以不同的固定报文格式发送。CAN 通信协议约定了 4 种不同的报文格式:

- ① 数据帧(Data Frame): 数据帧携带数据从发送器至接收器。
- ② 远程帧(Remote Frame): 接收单元向发送单元请求发送具有相同标识符数据所用的帧。
- ③ 错误帧(Error Fram): 任何单元检测到一个总线错误就发出错误帧。
- ④ 过载帧(Overload Frame): 过载帧用以在先后的数据或远程帧之间提供一个附加的延时。

另外,数据帧和远程帧可以使用标准帧及扩展帧 2 种格式。它们用一个帧间隔与前面的帧分开。

2.3.1 数据帧

数据帧(Data Frame)由以下 7 个不同的位域组成: 帧起始、仲裁域、控制域、数据域、CRC 域、应答域及帧结尾,结构如图 2-7(标准帧)、图 2-8(扩展帧)所示。

1) 帧起始 SOF(标准格式和扩展格式): 标志数据帧和远程帧的起始

由一个显性位组成,只有在总线空闲时才允许站点开始发送信号,所有站必须同步于开始发送报文的站的帧起始前沿(硬同步),即检测到由隐性转到显性的边沿,为所有节点的采样点选择一个共同的位置,即同步。

2) 仲裁域

在 SOF 之后是仲裁域。

a. 标准帧: 由 12 个位组成,分别为 11 个识别位(ID)和一个远程发送请求(RTR)位。RTR 位用于区分报文是数据帧(RTR 位为显性)还是远程帧(RTR 位为隐性状态)。

b. 扩展帧: 由 11 位基本 ID、SRR 位、IDE 位和 18 位扩展 ID 组成。其中,SRR 位和 IDE 位皆为隐性。

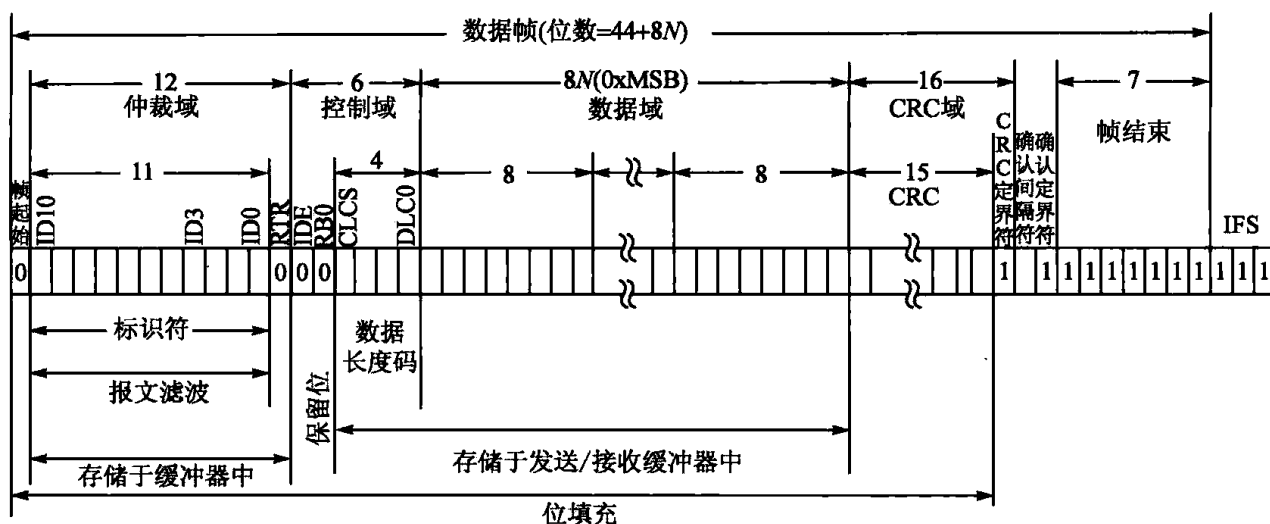


图 2-7 报文的数据结构帧(标准帧)

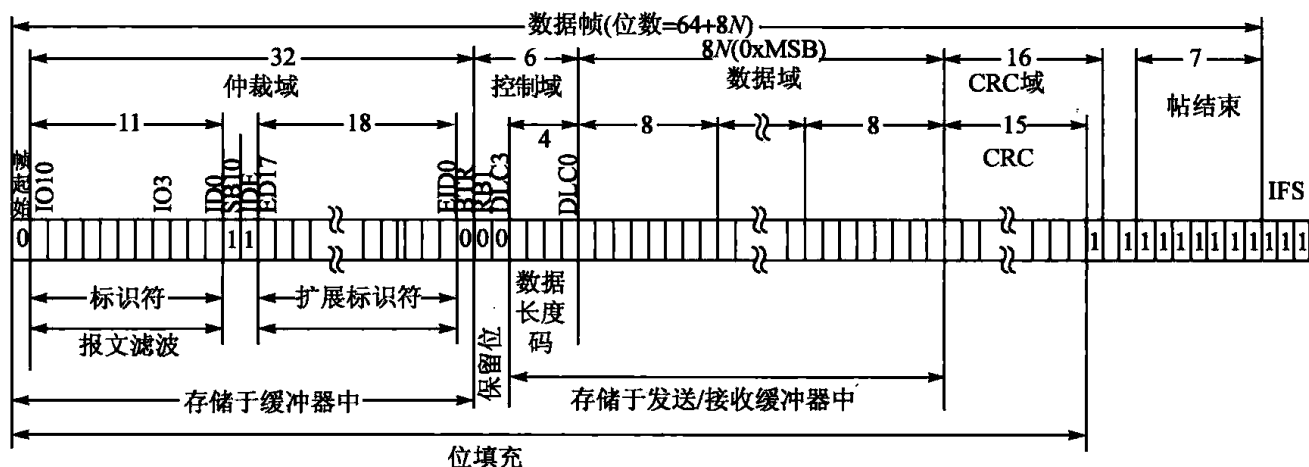


图 2-8 报文的数据结构帧(扩展帧)

3) 控制域

在仲裁域之后是控制域,由 6 个位组成。控制域的第一位为识别扩展(IDE)位,该位为显性状态时,说明这是标准帧。识别扩展位的下一位为零保留位(RB0),这一保留位将由 CAN 协议定义为显性位。控制域的其余 4 位为数据长度码(DLC),说明了报文中包含的数据字节数。具体定义见表 2-3。

4) 数据域

控制字段之后为数据域,包含正在发送的数据字节。数据域长度由上述数据长度码 DLC 定义(0~8 字节)。

5) 循环冗余码(CRC)域

数据域后为循环冗余校验码(CRC)域,用来检测报文传输错误。CRC 域包含一个 15 位

CAN 总线技术

的 CRC 序列,之后是 1 个隐性 CRC 定界位。

表 2-3 数据长度码和字节数的关系

数据字节的数目	数据代码长度			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

注: d:显性;r:隐性。

CRC 的原理:将每个比特串看作一个多项式。

通常它将比特串: $b_{n-1}b_{n-2}b_{n-3}\cdots b_2b_1b_0$

解释成多项式: $b_{n-1}x_{n-1}+b_{n-2}x_{n-2}+b_{n-3}x_{n-3}+\cdots+b_2x_2+b_1x_1+b_0$

比如,比特串: 1 0 0 1 0 1 0 1 1 1 0

被解释成: $x_{10}+x_7+x_5+x_3+x_2+x_1$

因为每个位或者是 0,或者是 1,这里只写出位为 1 的项,而不写出为 0 的项。

① 给定一个比特串,在其尾部追加几个 0 并把它叫 B,使 $B(x)$ 对应于 B。

② 将 $B(x)$ 除以一个事先约定的生成多项式 $G(x)$ (Generator Polynomial),求出余式 $R(x)$ 。

③ 定义 $T(x) = B(x) - R(x)$,并且减法运算可以通过与 $R(x)$ 对应的比特串替换成以前追加 0 的串而完成。

④ 传输与 $T(x)$ 对应的比特串 T。

⑤ 使 T' 代表接收方收到的比特流, $T'(x)$ 为相应的多项式。接收方将 $T'(x)$ 除以 $G(x)$ 。若余数为 0,则接收方认为 $T=T'$,传输未发生错误。否则,接收方认为传输发生了错误并要求重传。

在此,组成位流的成分是:帧起始、仲裁域、控制域、数据域,15 个最低位的系数是 0,将此多项式除以 $R(x)$ 多项式为: $x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+1$ 。CRC 校验是由硬件完成的。

6) 应答域(ACK Field)

所谓应答(Acknowledgment),即所有的接收器检查报文的一致性。对于接收正确的报文

接收器应答,对于不正确的报文接收器作出标志。应答域由应答间隙和应答定界符两个位组成。在应答间隙(ACK slot)期间,发送节点发出一个隐性位,任何接收到有效报文的节点会发回一个显性位(无论该节点是否配置为接收该报文与否),确认帧收到无误。应答定界符为1个隐性位。

7) 帧结尾

每一个数据帧和远程帧均由一个标志序列界定,这个标志序列由7个隐性位组成。

2.3.2 远程帧

一般情况下,数据传输是由数据源节点(如传感器发送数据帧)自主完成的,但也可能发生终节点向源节点请求发送数据的情况,即远程数据请求。要做到这一点,终节点须发送一个标识符与所需数据帧的标识符相匹配的远程帧。随后相应的数据源节点会发送一个数据帧以响应远程帧请求。远程帧也分为标准帧和扩展帧,由6个域组成。远程帧的构成如图2-9所示。

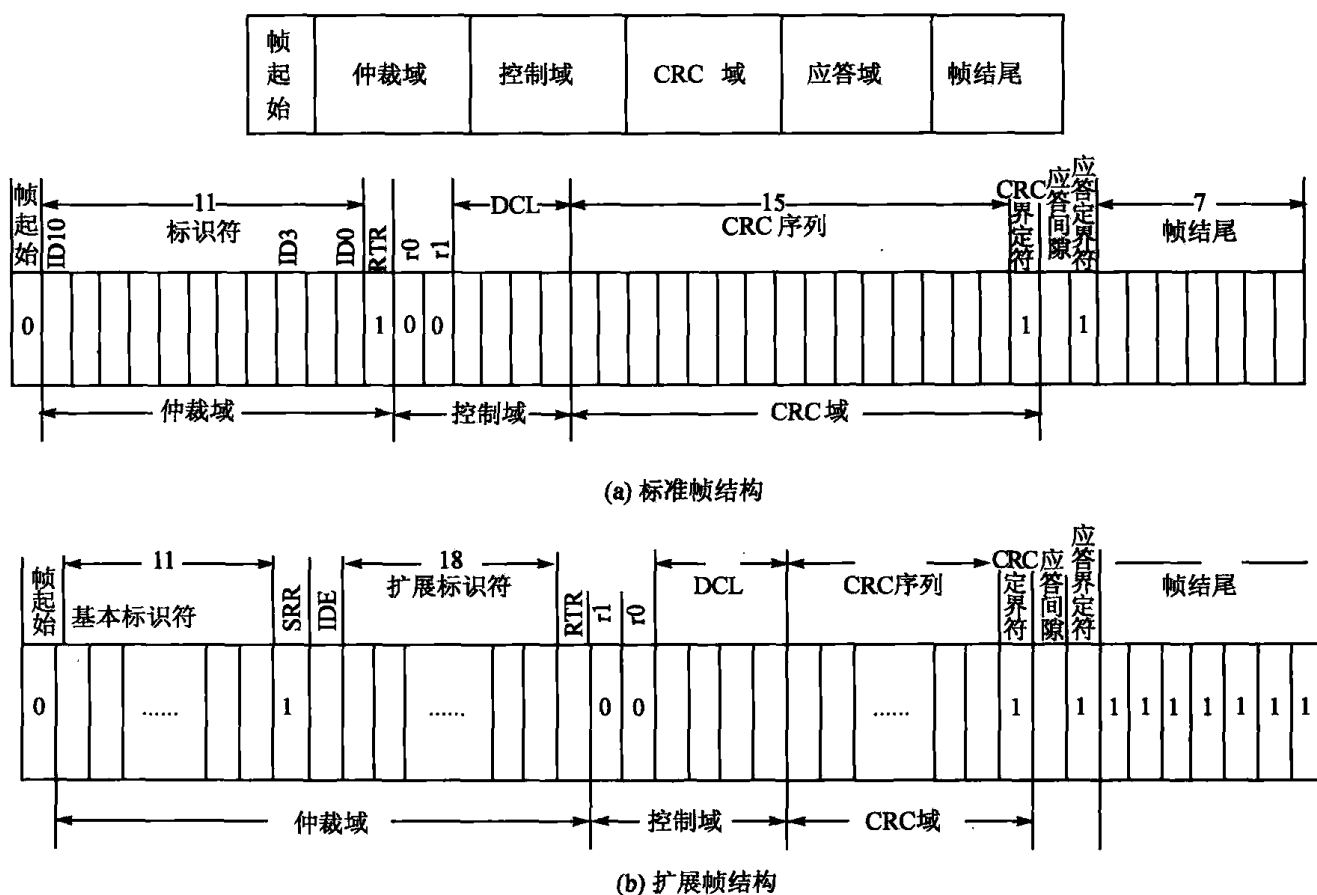


图 2-9 远程帧结构

CAN 总线技术

- ① 帧起始(SOF): 表示帧开始的段。
- ② 仲裁域: 表示该帧优先级的段。可请求具有相同标识符(ID)的数据帧。
- ③ 控制域: 表示数据的字节数及保留位的段。
- ④ CRC 域: 检查帧的传输错误的段。
- ⑤ ACK 域: 表示确认正常接收的段。
- ⑥ 帧结束: 表示远程帧结束的段。

远程帧与数据帧存在两点不同: 第一, 远程帧的 RTR 位为隐性状态; 第二, 远程帧没有数据字域, 所以数据长度代码的数值没有任何意义, 可以为 0~8 之间的任何数值。

当带有相同标识符的数据帧和远程帧同时发出时, 数据帧将赢得仲裁, 这是因为其紧随标识符的 RTR 位为显性。这样可使发送远程帧的节点立即收到所需数据。

2.3.3 错误帧

错误帧是由检测到总线错误的任一节点产生的。如图 2-10 所示, 错误帧包含两个域: 错误标志和错误定界符。

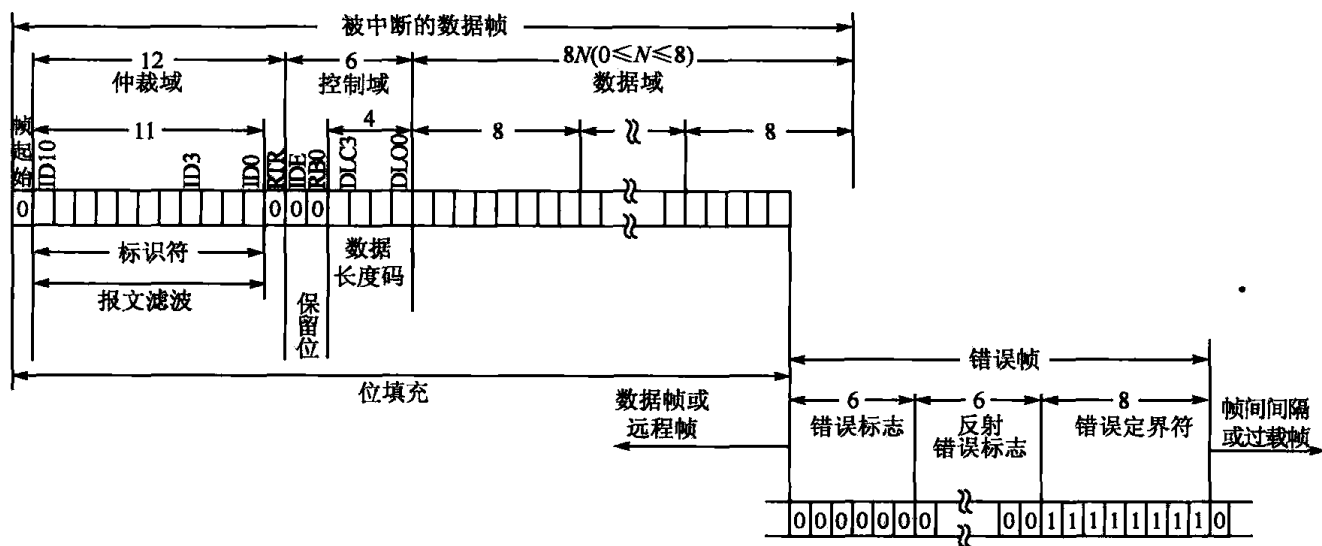


图 2-10 错误帧结构

1. 错误标志

错误标志包括激活错误标志和认可错误标志两种。节点发送哪种类型的出错标志, 取决于其所处的错误状态。

(1) 激活错误标志: 6 个连续的显性位

当节点处于错误激活状态且检测到一个总线错误时, 这个节点将产生一个激活错误标志, 中断当前的报文发送。激活错误标志由 6 个连续的显性位构成。这种位顺序打破了位填充规

则。所有其他节点在识别到所形成的位填充错误后自行产生错误帧,称为错误反射标志。错误标志域因此包含 6~12 个连续显性位(由 1 个或多个节点产生)。

(2) 认可错误标志: 6 个连续的隐性位

当节点处于错误认可状态且检测到一个总线错误时,该节点将发送一个认可错误标志。认可错误标志包含 6 个连续的隐性位。由此可知,除非总线错误被正在发送报文的节点检测到,否则错误认可节点错误帧的发送不会影响网络中任何其他节点。如果发送节点产生一个认可错误标志,那么由于位填充规则打破,将导致其他节点产生错误帧。错误帧发送完毕后,错误认可节点必须等待总线上出现 6 个连续隐性位后,认可错误标志才完成。

2. 错误定界符

由 8 个隐性位构成。传送了错误标志以后,每个节点开始发送错误定界符,先是发送一个隐性位,监视并检测到一个隐性位后,开始发送其余 7 个隐性位。错误定界符为错误帧划上了句号。在错误帧发送完毕后,总线主动恢复正常状态,被中断的节点会尝试重新发送被中止的报文。

2.3.4 过载帧

1. 过载帧的产生

过载帧只能在帧间间隔产生,因此可通过这种方式区分过载帧和错误帧(错误帧是在帧传输时发出的)。节点最多可产生两条连续过载帧来延迟下一条报文的发送,有 3 种情况会产生过载帧:

- ① 接收器由于内部原因需要延迟下一个数据帧或远程帧;
- ② 节点在帧空间检测到非法显性位;
- ③ 节点在错误界定符或过载界定符的第 8 位采样到一个显性位。

2. 过载帧格式

过载帧与激活错误帧具有相同的格式,如图 2-11 所示。

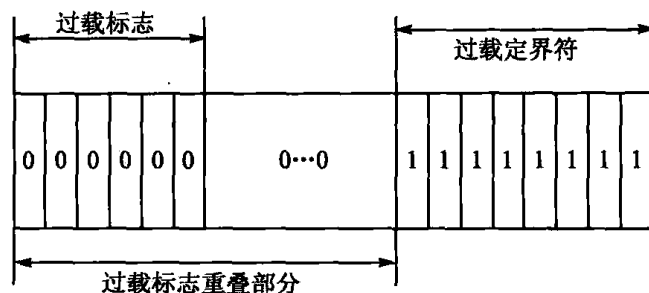


图 2-11 过载帧结构

过载帧由两个域组成,即过载标志和过载定界符。过载标志由 6 个显性位构成。因过载标志违反了“间歇”域的固定格式,其他节点检测到过载条件并发送过载标志,因此过载标志产

CAN 总线技术

生重叠(而错误帧包含最多 12 个显性位)。过载定界符包含 8 个隐性位。发送过载标志后,每个节点发送 1 个隐性位,然后均监控总线,直至检测到 1 个隐性位,此时,总线上每个节点完成了各自过载标志的发送,并开始同时发送其余 7 个隐性位。

2.3.5 帧间空间

帧间空间将前一条帧(无论何种类型)与其后的数据帧或远程帧分离开来。帧间空间由至少 3 个隐性位构成,又称为间断。间断使节点在发送下一条报文之前有时间进行内部处理。过载帧和错误帧前不能插入帧间隔。在间断之后,CAN 总线将保持隐性状态(总线空闲),直至下一条报文发送开始,其结构如图 2-12 所示。

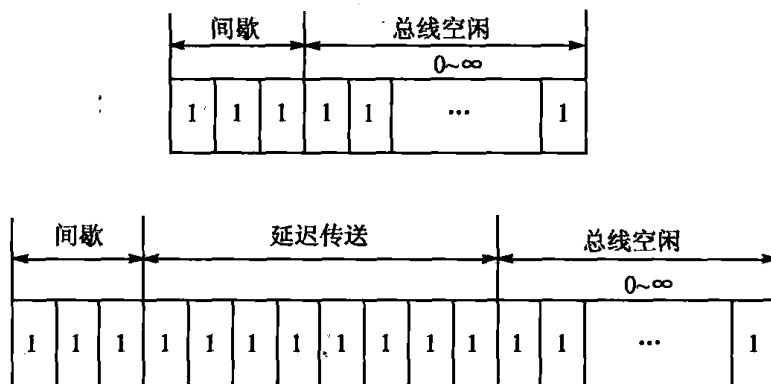


图 2-12 帧间空间结构

(1) 间 歇

3 个位的隐性位。间断期间,不允许传送数据帧或者远程帧,仅可以标识过载条件。

(2) 总线空闲

隐性电平,无长度限制(0 亦可)。总线认为是空闲时,任何等待发送报文的节点都可以访问总线。

(3) 延迟传送(发送暂时停止)

8 个位的隐性位。错误认可的节点发送报文后,在下一个报文开始传送之前或者确认总线空闲之前发出 8 个隐性位跟随在间断之后。其间,若有发送启动(由其他节点引起),则该节点将变为接收器。

2.4 错误界定及处理

2.4.1 错误类型

CAN 协议提供了完备的错误检测机制,可以检测到以下错误:

(1) 位错误

当发送节点发送了一个显性位但检测到一个隐性位,或发送一个隐性位但检测到一个显性位,同时该发送节点正在监控总线实际电平并将之与刚刚发送的位相比较时,产生一个位错误。下列情况将不产生位错误:

① 在仲裁域输出隐性电平但检测出显性电平时,视为仲裁失利,而不是位错误。

② 在仲裁域作为填充位输出隐性电平但检测出显性电平时,不视为位错误,而是填充错误。

③ 发送单元在应答域输出隐性电平但检测到显性电平时,判断为其他单元的 ACK 应答,而非位错误。

④ 发送节点输出认可错误标志时,打破位填充会导致其他节点发送错误帧,此时检测到显性电平,不视为位错误。

(2) 填充错误

在帧起始和 CRC 定界符之间,如果节点检测到 6 个连续且极性相同的位,则说明违反了位填充规则。此时节点将产生错误帧,以表明发生了位填充错误,报文将重新发送。

(3) CRC 错误

发送节点通过循环冗余校验(CRC)计算特殊校验位来确定从帧起始到数据域结束时的位序列。CRC 序列在 CRC 域发送。接收节点采用相同公式计算 CRC 序列,并将计算结果与收到的 CRC 序列相比较。如果两者不匹配,则接收节点产生错误帧,表明检测到 CRC 出错,出错报文将重新发送。

(4) 格式错误

如果一个节点在帧结尾、帧间空间、确认定界符或 CRC 定界符这 4 个位域中的任一位域检测到显性位,则产生出错帧表明检测到格式出错,报文将重新发送。以下情况例外:

① 即使接收单元检测出帧结尾(7 个位的隐性位)的后一位(第 8 个位)为显性电平,不视为格式错误。

② 即使接收单元检测出数据长度码(DLC)中 9~15 的值,也不视为格式错误。

(5) 应答错误

在报文的确认域,发送器检查确认间隙(已发送为隐性位)是否包含一个显性位。如果没有,则表明没有任何其他节点正确接收到报文。这时将发生确认错误,并产生一个错误帧,报文将重新发送。

2.4.2 错误帧的输出

检测出满足错误条件的单元则输出错误标志以通报错误。处于激活错误状态的单元输出的错误标志为激活错误标志;处于认可错误状态的单元输出的错误标志为认可错误标志。发送单元发送完错误帧后,再次发送数据帧或远程帧。错误标志输出时间如表 2-4 所列。

表 2-4 错误标志输出时序

错误的种类	输出时序
位错误 填充错误 格式错误 ACK 错误	从检测出错误后的下一位开始输出错误标志
CRC 错误	应答界定符后的下一位开始输出错误标志

2.4.3 错误界定及规则

1. 错误界定

CAN 具有错误分析功能。每个 CAN 通信单元能够在 3 个错误状态之一中工作：错误激活、错误认可、总线关闭。这些错误的区分取决于硬件自带错误计数器（接收错误寄存器、发送错误寄存器）的值。

① 错误激活状态：如果两个错误计数器的值都在 0~127 之间，则通信单元是“错误激活”的；一旦检测到错误，则产生激活错误标志（6 个显性位）。错误激活单元可以正常地参与总线通信。

② 错误认可状态：如果错误计数器值位于 128~255 之间，则通信单元是错误认可的；一旦检测到错误，则产生认可错误标志（6 个隐性位）。错误认可单元可以参与总线通信，只是在发送错误帧之后，再启动下一次发送之前处于等待状态。

③ 如果发送错误计数器高于 255，则到达总线关闭状态；在这种状态下复位请求位自动置位 CAN 节点，对总线没有影响。

2. 错误界定规则

CAN 总线上单元的 error 状态是依据错误计数器的数值而界定的，错误计数器按照下列规则进行计数（在给定报文发送期间，可应用不止 1 个规则）：

① 当接收器检测到一个错误时，接收错误计数器值加 1。

另外，在发送错误激活标志和过载标志期间所检测到的错误为位错误时，接收错误计数器值不变。

② 当接收器检测到一个显性位作为送出错误标志后的第一位时，接收错误计数器加 8。

③ 当发送器发送一错误标志时，发送错误计数器值加 8。

例如：a. 错误认可节点检测到应答错误且其送出认可错误标志但未检测到显性位时，发送计数器值不变。

b. 仲裁域作为填充位输出隐性电平但检测出显性电平时，发送错误计数器值不变。

- ④ 当发送激活错误标志或过载标志时,发送器检测到位错误,发送错误计数器值加 8。
- ⑤ 当发送激活错误标志或过载标志时,接收器检测到位错误,接收错误计数器值加 8。
- ⑥ 在以下情况下,接收器的接收错误计数器值和发送器的发送错误计数器值都加 8:
- 在激活错误标志或过载标志后检测到第 14 个连续的显性位。
 - 在认可错误标志后检测到第 8 个连续的显性位和每个附加的第 8 个连续的显性位。
- ⑦ 报文成功发送后,发送错误计数器值减 1。
- ⑧ 成功地接收到报文后,若接收错误计数器值小于 128,则接收错误计数器值减 1;若大于 127,则置一个介于 119 和 127 之间的值。
- ⑨ 当发送错误计数器值和接收错误计数器值都小于或等于 127 时,错误认可节点重新回到错误激活状态。
- ⑩ 总线关闭只能在主控制器使用“复位请求=0”命令退出,这将启动总线关闭恢复时间。发送错误计数器用于对 128 个总线空闲信号计数,恢复时间结束后两个错误计数器都是 0,设备处于错误激活状态。

CAN 系统各节点的具体错误状态及转换规则如图 2-13 所示。

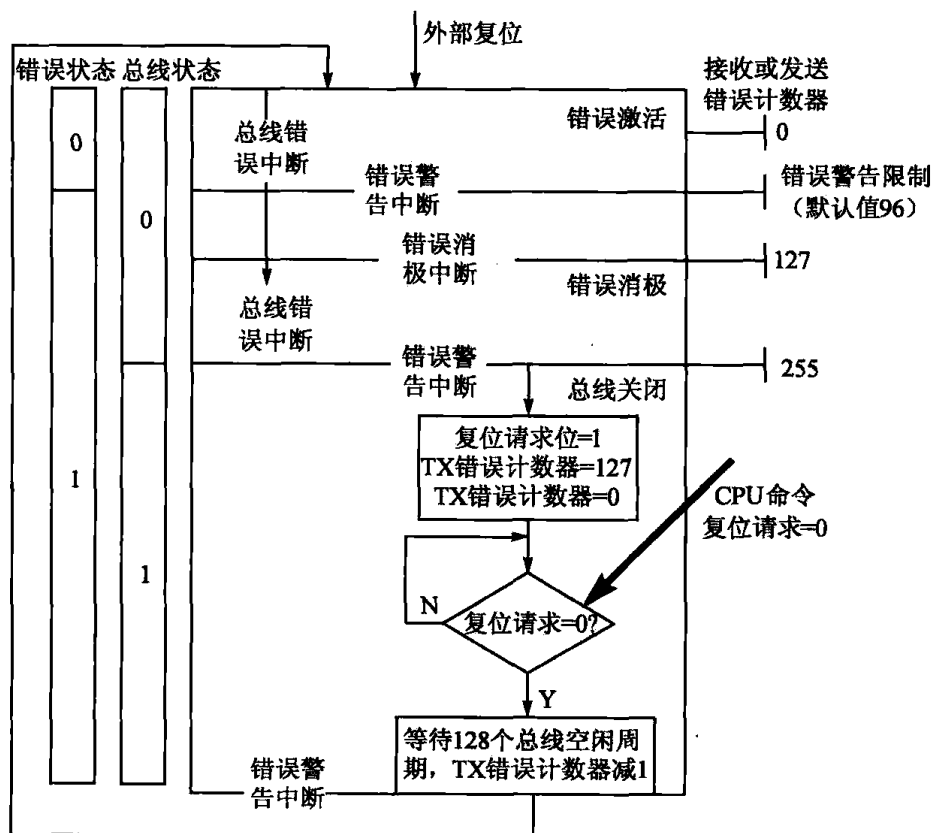


图 2-13 节点错误状态

CAN 总线技术

综上所述, CAN 具有极强的安全性, 每一个节点均可采取措施以进行错误检测(监视、循环冗余检查、位填充、报文格式检查)、错误标定及错误自检。由此可以检测到全局错误、发送器局部错误、报文中 5 个任意分布错误和长度低于 15 位的突发性错误、任一奇数个错误, 其遗漏错误的概率低于报文错误率 4.7×10^{-11} 。同时, CAN 具有很好的故障界定能力(Fault Confinement), 因为 CAN 节点能够把永久故障和短暂扰动区分开来, 永久故障的节点会关闭。

2.5 位定时与同步

2.5.1 基本概念

① 标称位速率(Nominal Bit Rate): 理想发送器在没有重新同步的情况下每秒发送的位数量。

② 标称位时间(Nominal Bit Time): 标称位时间 = $1/\text{标称位速率}$, 即数据发出时, 每个位值都要在总线上保持的一定时间。

③ 时间份额(T_q): 与振荡周期的固定时间单元有关的时间单位。

$$T_q = m \times \text{最小时间份额} \quad m \text{ 为预比例因子 } (m=1 \sim 32)$$

④ 信息处理时间: 一个以采样点为起始的时间段, 用于计算后续位的位电平(≤ 2 个 T_q)。

⑤ 时间段的长度: 根据具体网络情况而不同。一个位时间总的量程值可以编程为 $8 \sim 25T_q$ 之间。

⑥ 同步: 使系统的收发两端在时间上保持步调一致。

2.5.2 CAN 总线位定时与同步机制

CAN 是有效支持分布式实时控制的串行通信网络。从位定时的同步方式考虑, 它实质上属于异步通信协议, 每传输一帧, 以帧起始位开始, 以帧结束及随后的间歇场结束, 这就要求收/发双方从帧起始位开始必须保持帧内信息代码中的每一位严格的同步。从位定时编码考虑, 它采用的是非归零编码方式, 位流传输不像差分码那样可以直接用电平的变化来代表同步信号, 它属于自同步方式(接收端设法从收到的信号中提取同步信息的方式), CAN 节点从一个位值到另一个位值的转变中提取时钟信息。为保证同步质量, CAN 协议定义了自己的位同步方式, 即硬同步和重同步。

1. 位周期结构

CAN 总线的标准位时间(也称额定位时间)的结构如图 2-14 所示, 由 4 部分组成的: 同步段(SYNC_SEG)、传播段(PROP_SEG)、相位缓冲段 1(PSEG1)和相位缓冲段 2(PSEG2)。

所以额定位时间 $t_{\text{NBT}} = t_{\text{SYNC_SEG}} + t_{\text{PROP_SEG}} + t_{\text{PSEG1}} + t_{\text{PSEG2}}$ 。位周期中的这些段都是可以编程设置的,且都可以用整数个基本时间单位(时间份额) T_q 来表示。这个基本时间单位由振荡器分频而得,即 $T_q = \text{BRP} / f_{\text{SYS}}$,其中 BRP 为波特率预分频因子,也可以编程设置。

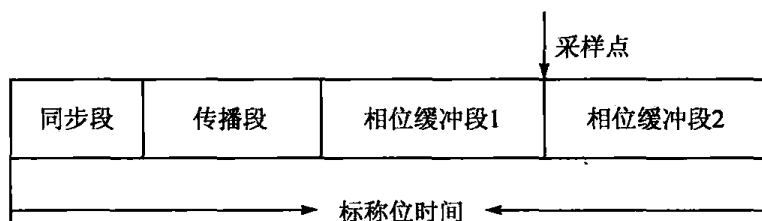


图 2-14 位时间结构

① 同步段(SYNC_SEG): 用于同步总线上不同的节点,是 CAN 总线位周期中每一位的起始部分。不管是发送节点发送一位还是接收节点接收一位都是从同步段开始的。此段期待一个跳变沿。

② 传播段(PROP_SEG): 用于补偿网络内的物理延时。由于发送节点和接收节点之间存在网络传输延迟以及物理接口延迟,发送节点发送一位之后,接收节点延迟一段时间才能接收到,因此,发送节点和接收节点对应同一位的同步段起始时刻就有一定的延时,记为 $t_{\text{PROP_SEG}}$ 。传播段的设置就是要补偿该段延时($t_{\text{PROP_SEG}}$)的。CAN 总线协议中的非破坏性仲裁机制以及帧内应答机制都要求那些正在发送位流的发送节点能够同时接收来自其他发送节点的“显性位”(逻辑 0);否则,会使得仲裁无效或者应答错误。传播段延迟那些可能较早采样总线位流的节点的采样点,保证由各个发送节点发送的位流到达总线上的所有节点之后才开始采样。

$$t_{\text{PROP_SEG}} = (\text{信号在总线上传播的时间} + \text{输入比较器延时} + \text{输出比较器延时}) \times 2$$

③ 相位缓冲段 1、2(PSEG1、PSEG2): 用于补偿边沿阶段的误差。可以通过重新同步来加长或缩短。

④ 采样点(Sample Point)读取总线电平并转换为一个对应的位置的一个时间点,位于相位缓冲段 1 结尾。

另外,重同步跳转宽度(SJW)并不是位周期里的一段,却是位定时计算的一个重要的指标;它定义了重同步时为补偿相位误差位时间中相位缓冲段 1 或者相位缓冲段 2 增长或缩短的最大时间单元数。

2. 同步机制

CAN 总线的位同步只有在节点检测到隐性位(逻辑 1)到显性位(逻辑 0)的跳变时才会产生,当跳变沿不位于位周期的同步段之内时将会产生相位误差,该相位误差就是跳变沿与同步段结束位置之间的距离。如果跳变沿发生在同步段之后采样点之前,则为正的相位误差;如果跳变沿位于同步段之前采样点之后,则为负的相位误差。相位误差源于节点的振荡器漂移、网

CAN 总线技术

络节点之间的传播延迟以及噪声干扰等。CAN 协议规定了两种类型的同步：硬同步和重同步。

(1) 硬同步

硬同步只在总线空闲时通过一个下降沿(帧起始)来完成,此时不管有没有相位误差,所有节点的位时间重新开始。强迫引起硬同步的跳变沿位于重新开始的位时间的同步段之内。硬同步后,位时间由每个位定时逻辑单元以 SYNC_SEG 重新启动。因此,硬同步强迫引起硬同步的边沿处于重新启动位时间的同步段内,如图 2-15 所示。

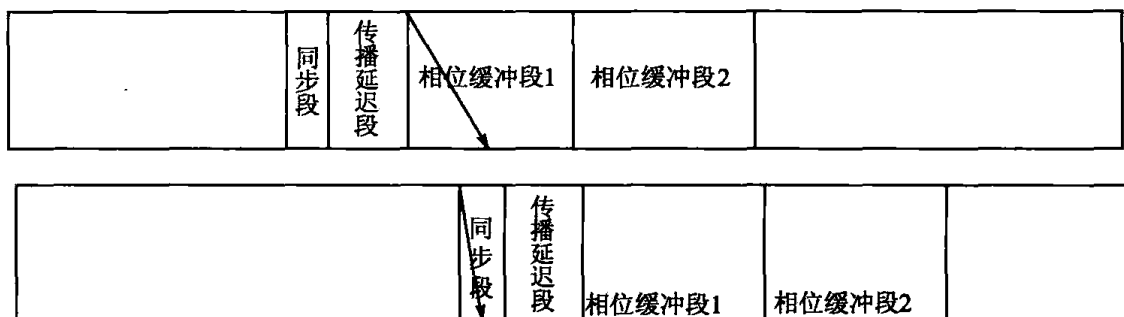


图 2-15 硬同步

(2) 重同步

在报文的随后位中,每当有从隐性位到显性位的跳变,并且该跳变落在了同步段之外时,就会引起一次重同步。重同步机制可以根据跳变沿增长或者缩短位时间以调整采样点的位置,保证正确采样。

(3) 同步边沿的相位误差

边沿的相位误差 e 由相对于 SYNC-SEG 边沿的位置给定,以时间份额量度。相位误差的符号定义如下:

- $e=0$, 边沿位于 SYNC_SEG 内;
- $e>0$, 边沿位于 SYNC_SEG 之后采样点前;
- $e<0$, 边沿位于 SYNC_SEG 之前。

(4) 重同步机制分析

1) 正相位误差重同步

如图 2-16 所示,跳变沿落在同步段之后采样点之前时,为正的相位误差,接收器会认为这是一个慢速发送器发送的滞后边沿。此时节点为了匹配发送器的时间,会增长自己的相位缓冲段 1(阴影部分)。增长的时间为相位差的绝对值,但是上限是重同步跳转宽度 SJW。

2) 负相位误差重同步

如图 2-17 所示,跳变沿落在采样点之后同步段之前时,为负的相位误差,接收器把它解释为一个快速发送器发送的下一个位周期的提前边沿。同样,节点为了匹配发送器的时间会

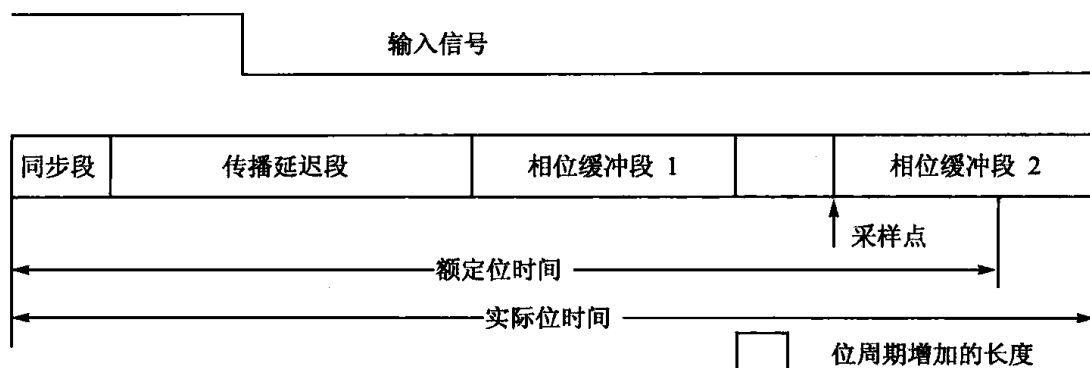


图 2-16 正相位误差时的重同步

缩短自己的相位缓冲段 2(阴影部分), 下一个位时间立即开始。缩短的时间也为相位差的绝对值, 上限是重同步跳转宽度 SJW。

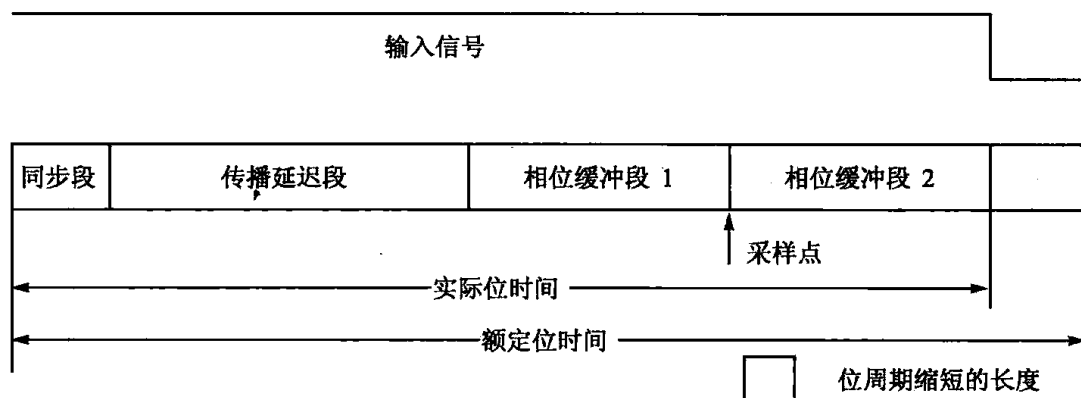


图 2-17 负相位误差时的重同步

注意: ① 相位缓冲段只在当前位周期内被增长或者缩短时, 对于接下来的位周期, 只要没有重同步, 各段就恢复为位时间的编程预设值。

② 当相位差的绝对值小于或者等于重同步跳转宽度 SJW 时, 重同步和硬同步的效果是相同的, 能实现相位差的补偿。

③ 如果相位差的绝对值相比重同步跳转宽度要大, 则由于补偿的最大值是重同步跳转宽度, 从而会使重同步不能完全补偿相位差。

CAN 协议的位填充机制除实现仲裁域、控制域、数据域和 CRC 序列的数据透明性外, 还增加了从隐性位到显性位跳变的机会, 也就是增多了重同步的数量, 提高了同步质量。在没有出错影响的情况下, 位填充原则保证了两次重同步跳转边沿之间不会多于 10 个位周期(即 5 个显性位, 5 个隐性位), 而实际的系统会有错误发生, 使得实际的两次重同步跳转边沿之间的间隔可能为 17~23 个位时间(激活错误标志及其叠加 6~12 个位时间, 错误界定符 8 个位时间, 间歇场 3 个位时间)。

CAN 总线技术

(5) 同步遵守下列规则

① 在一个位时间内仅允许一种同步。

② 只要先前采样点检测到的数值(先前读总线数值)不同于边沿后即现的总线数值,边沿即用于同步。

③ 总线空闲期间,当存在隐性至显性的跳变沿时即完成硬同步。

④ 所有满足规则①和②的其他隐性至显性的跳变沿(和在低位速率情况下一样,选择显性至隐性跳变沿)用于重同步;例外情况是,若只有隐性至显性沿用于重同步,则由于具有正相位误差的隐性至显性沿的结构,发送器将不完成重同步。

(6) 重同步跳转宽度

由于重同步的结构,PSEG1 可被延长或 PSEG2 可被缩短。延长和缩短相位缓冲器段的总和具有重同步跳转宽度给定的上限。重同步跳转宽度可在 $1 \sim \min(4, \text{PSEG1})$ 之间被编程。

时钟信息可由一位值至另一位值跃变提取。具有连续相同值的位的最大数目是固定的(由于位填充),这一特性提供了帧期间总线单元重同步于位流的可能性。可用于重同步的两个跳变之间的最大长度为 29 位时间。

当引起重同步的边沿相位误差的幅值小于或等于重同步跳转宽度的编程数值时,重同步导致位时间的缩短或延长使采样点处于正确位置。当相位误差幅值大于重同步跳转宽度时,如果相位误差 e 为正,则 PSEG2 缩短数值等于重同步跳转宽度。

在实际的系统设计中,用户可以根据振荡器时钟频率、总线波特率以及总线的最大传输距离等因素,对 CAN 控制器的位定时参数进行优化设置,协调影响位定时设置的两个主要因素:振荡器容差和最大总线长度,合理安排位周期中采样点的位置和采样次数,保证总线上位流的有效同步的同时,优化系统的通信性能,进一步推进 CAN 总线的广泛应用。

第 3 章

SJA1000 的原理与使用

当 CAN 协议规范化之后,就可以在集成电路上设计实现物理层和数据链路层的功能。实现这种功能的集成电路叫通信控制器。通信控制器与微处理器配合使用,在微处理器中实现应用层的功能以及用户需要的其他功能。

通信控制器可以单独封装,也可以嵌入到微处理器中,作为微处理器的一个外设。微处理器通过对通信控制器的设置和操作控制它的工作状态,进行数据的接收和发送,把应用层建立在它所实现的物理层和数据链路层基础上。独立封装的 CAN 控制器目前最常见的是 NXP 半导体公司生产的 SJA1000,它的功能和引脚与前期产品 PCA82C200 兼容。许多微控制器提供 CAN 通信接口,8051 系列单片机中,P87C591、8051F040 等都有 CAN 通信口,嵌入式处理器 ARM 系列也集成了 CAN 通信控制器。

本章主要介绍独立的 CAN 通信控制器 SJA1000。

3.1 SJA1000 的结构与功能

3.1.1 概 述

SJA1000 是 NXP 半导体 PCA82C200 CAN 控制器(BasicCAN)的替代产品,在引脚及电气特性上与 PCA82C200 控制器完全兼容。SJA1000 有两种不同的协议模式,即 BasicCAN 模式和 PeliCAN 模式。BasicCAN 模式是复位时的默认模式,这种模式与 PCA82C200 兼容,支持 CAN 2.0A 协议;PeliCAN 模式是新增加的工作模式,支持 CAN 2.0B 协议,具有一些新的特性。工作模式的切换是通过内部的时钟分频寄存器 CDR 的 CAN 模式位来选择的。SJA1000 可以完成物理层和数据链路层的所有功能。

SJA1000 可以支持多种微处理器的时序特性,有一个模式选择引脚,可以设置 Intel 模式或 NXP 模式。SJA1000 与微处理器的接口简单,微处理器以访问外部存储器的方式来访问 SJA1000。

CAN 总线技术

SJA1000 的时钟频率可以达到 24 MHz, 位速率可达 1 Mbps, 同时支持 11 位和 29 位标识符的数据帧。

PeliCAN 模式主要扩展了以下功能:

① 错误检测及处理。提供错误码寄存器, 可指示发生错误的位置。对错误计数器可以进行读/写访问, 错误报警界限值可编程设置。每次 CAN 总线错误都会产生错误中断, 新增仲裁丢失中断, 记录仲裁丢失位的位置。

② 可使用单次发送方式, 取消重发功能。

③ 监听模式(无应答, 无活动错误标志)。

④ 支持热插拔, 可由软件实现位速率自动检测。

⑤ 接收过滤器扩大为 4 字节长, 接收方式更加灵活。节点可以接收自发信息。

SJA1000 有两种工作模式, 分别是复位模式和操作模式。当硬件复位、控制器掉线或置位复位请求位时, SJA1000 进入复位模式; 当清除其内部控制寄存器的复位请求位时, SJA1000 进入工作模式。其内部寄存器在两种模式下的访问是有区别的, 有的寄存器只能在复位模式下访问, 有些寄存器只能在工作模式下访问, 而有些寄存器在这两种模式下都可以访问。

3.1.2 芯片引脚定义与说明

SJA1000 有 DIP28 和 SO28 两种封装形式, 引脚如图 3-1 所示, 工作温度范围 $-40 \sim +125^{\circ}\text{C}$ 。

AD0~AD7: 数据地址总线。

ALE/AS: Intel 模式下, 地址锁存信号; NXP 模式下, 地址选择信号。

$\overline{\text{CS}}$: 片选信号。

$\overline{\text{RD/E}}$: Intel 模式下, 读允许信号; NXP 模式下, 使能信号。

$\overline{\text{WR}}$: 写允许信号。

CLKOUT: 时钟输出信号, SJA1000 内部振荡器产生的时钟信号, 经过分频后输出。

V_{ss1} 、 V_{dd1} : 逻辑电路电源。

XTAL1、XTAL2: 时钟振荡器的输入、输出端, XTAL1 可接外部时钟。

MODE: 模式选择: 高电平, Intel 模式; 低电平, NXP 模式。

V_{ss2} 、 V_{dd2} : 输入比较器的电源。

V_{ss3} 、 V_{dd3} : 输出驱动电路电源。

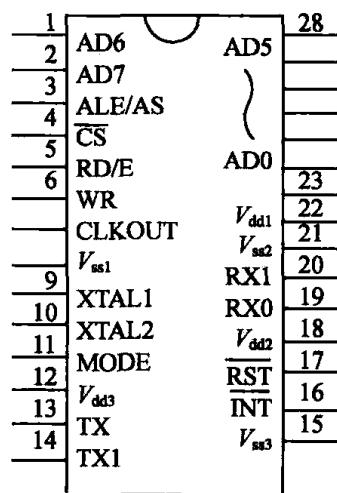


图 3-1 SJA1000 的引脚

TX0、TX1: 输出驱动器 0 和输出驱动器 1 的输出端。

RX0、RX1: 外部 CAN 总线接入 SJA1000 的输入端。

$\overline{\text{RST}}$: 复位。

$\overline{\text{INT}}$: 中断信号输出。

SJA1000 片上有 3 个独立电源, 分别给输入电路、输出电路以及内部逻辑管理电路供电, 这样可以把逻辑功能电路与外部总线更好地隔离, 减少外部干扰。

XTAL1、XTAL2 引脚的接法和一般单片机的接法相同。SJA1000 可以使用外部时钟, 也可以使用内部时钟。因为要与微处理器配合使用, SJA1000 和微处理器通常使用一个共同的时钟信号。图 3-2 是一种时钟电路方案, SJA1000 内部振荡器产生时钟, 微处理器从 SJA1000 的时钟输出引脚获得工作时钟。时钟也可以由微处理器的内部振荡器产生, 同时把时钟信号引出给 SJA1000, 如图 3-3 所示。其他的时钟处理方式也是可以的, 如微处理器和 SJA1000 单独使用各自的时钟或者接入同一个外部时钟。

SJA1000 的中断信号可以使 SJA1000 的某些事件触发微处理器的中断, 相对微处理器来说, 它是一个外部中断。

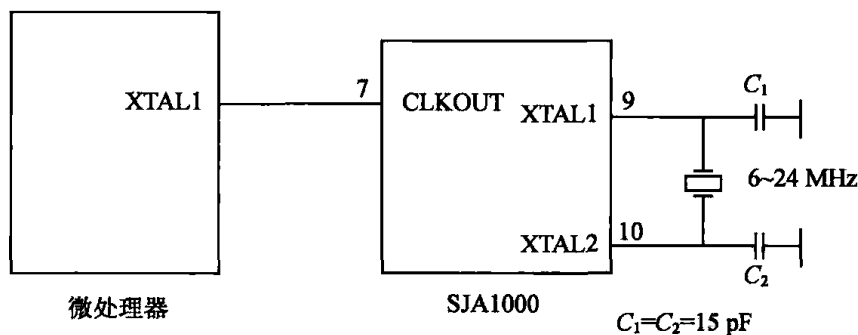


图 3-2 时钟电路 1

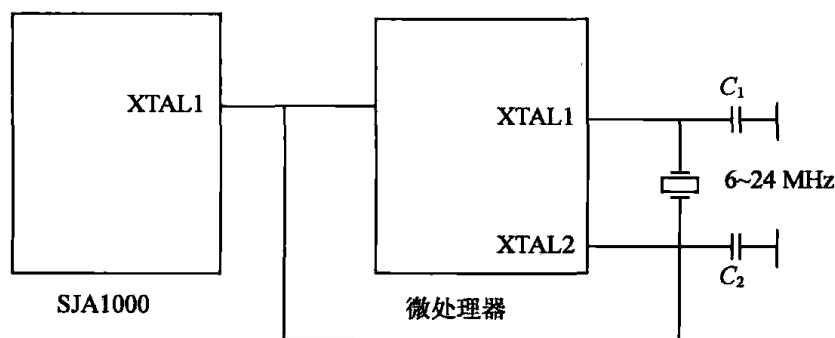


图 3-3 时钟电路 2

一个典型的 CAN 设备有如图 3-4 所示的结构。SJA1000 是其中 CAN 通信控制器, 它按照 CAN 协议进行数据的收发。总线驱动器可提高通信驱动能力, 抑制干扰, 并提供总线保护等。

CAN 总线技术

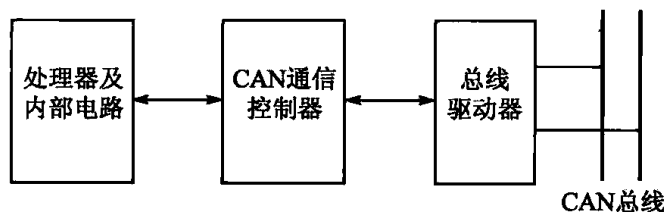


图 3-4 CAN 设备的典型结构

3.1.3 SJA1000 的结构及内部存储器分配

1. SJA1000 的结构和功能

SJA1000 的硬件结构框图如图 3-5 所示,主要包括接口管理逻辑、数据缓冲器、验收滤波器、位流处理器、位时序逻辑、错误管理逻辑等几个主要功能单元。各单元电路的功能简介如下:

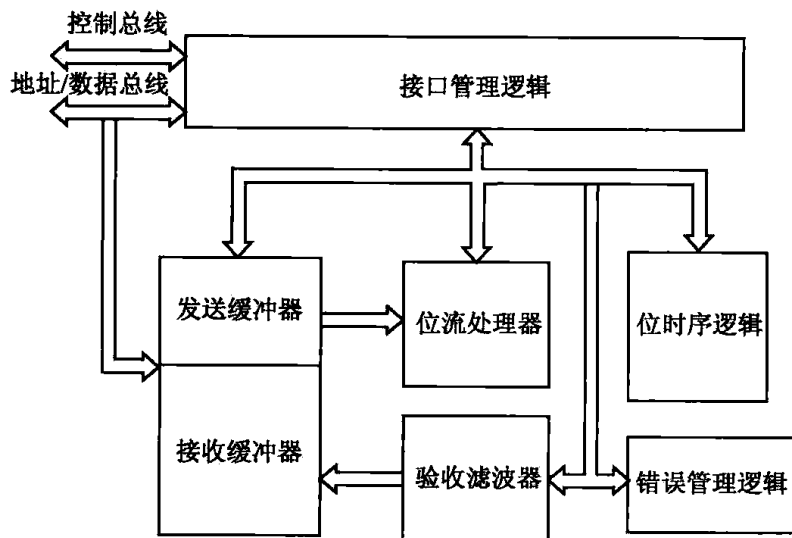


图 3-5 SJA1000 硬件结构框图

1) 接口管理逻辑

接口管理逻辑完成对外部处理器的连接,解释来自处理器的命令,控制 CAN 寄存器的寻址,向处理器提供中断信息和状态信息。该处理器可以是微型控制器或其他器件,经过复用的地址/数据总线访问 SJA1000 的内部寄存器,且控制总线的信号都在这里处理。

2) 数据缓冲区

数据缓冲区数据缓冲区包括接收缓冲区和发送缓冲区两部分,分别存储接收到的和要发送的数据。

处理器可直接将标识符和数据送入发送缓冲区,然后置位命令寄存器 CMR 中的发送请求位 TR,则启动 CAN 核心模块读取发送缓冲区中的数据,并按 CAN 协议封装成一个完整 CAN 信息帧通过收发器发往总线。

接收缓冲区位于验收滤波器和 CPU 之间,用来存储从 CAN 总线上接收并确认的信息。它采取先进先出(FIFO)的数据管理方式,共有 64 字节长度,提供给用户 13 字节的操作窗口,恰好是一个数据帧的长度。接收缓冲区中可以存储多个报文,储存报文的多少由工作模式决定,最多能存储 32 个。接收帧的数量可通过访问接收信息计数器 RMC 得知。一条报文读取后,下一条自动进入操作窗口等候读取。这使得数据溢出的可能性大大降低,从而使用户能更灵活地指定中断服务和中断优先级。

3) 验收滤波器

验收滤波器当 SJA1000 收到一条报文时,首先将串行位流转换成并行数据,然后送往验收滤波器。验收滤波器单元完成接收信息的滤波,通过这个可编程的滤波器 SJA1000 能确定哪些信息可以接收。只有验收滤波通过且无差错时,才把接收的信息帧送入接收 FIFO 缓冲区。

4) 位流处理器

位流处理器在数据缓冲区和 CAN 总线之间控制数据流的队列发生器,还执行总线上的错误检测、仲裁、填充和错误处理。

5) 位时序逻辑

位时序逻辑监视串行的 CAN 总线位时序,提供了可编程的时间段来补偿传播延时、相位偏移、定义采样点和每一位的采样次数。

6) 错误管理逻辑

错误管理逻辑负责处理传输层模块的错误。

2. SJA1000 的内部存储器分配

SJA1000 的内部 RAM 由功能寄存器和报文缓冲区组成,功能寄存器部分称为控制段。控制段在初始化载入时,被编程来配置通信参数。微控制器也是通过这个段来控制 CAN 总线上的通信的。

为了防止控制段的寄存器被随意改写而影响正常通信,SJA1000 设置了两种不同的操作模式——工作模式和复位模式。对控制段寄存器的初始化操作是在复位模式下进行的。当硬件复位或控制器掉线时,SJA1000 自动进入复位模式。正常的 CAN 通信是在工作模式下运行。这两种操作模式还可以通过控制寄存器的复位请求位切换。

初始载入后,验收代码、验收屏蔽、总线定时寄存器 0、1 以及输出控制寄存器就不能改变了,只有控制寄存器的复位位置高位时(进入复位模式)才可以访问这些寄存器。在 Basic-CAN 模式和 PeliCAN 模式下,SJA1000 内部 RAM 的分配略有不同。

CAN 总线技术

(1) BasicCAN 协议模式下的存储器分配

表 3-1 是 BasicCAN 模式下 SJA1000 内部 RAM 的分配情况,其中地址指 SJA1000 内部地址。

测试寄存器只用于产品测试,正常工作时使用这个寄存器可能会使设备产生不可预料的行为。

表 3-1 BasicCAN 模式下 SJA1000 内部 RAM 的分配

内部地址	段名	工作模式		复位模式	
		读	写	读	写
0	控制段	控制寄存器	控制寄存器	控制寄存器	控制寄存器
1		FFH	命令寄存器	FFH	命令寄存器
2		状态寄存器	—	状态寄存器	—
3		FFH	—	中断寄存器	—
4		FFH	—	验收代码寄存器	验收代码寄存器
5		FFH	—	验收屏蔽寄存器	验收屏蔽寄存器
6		FFH	—	总线定时器 0	总线定时器 0
7		FFH	—	总线定时器 1	总线定时器 1
8		FFH	—	输出控制寄存器	输出控制寄存器
9		测试寄存器	测试寄存器	测试寄存器	测试寄存器
10	发送缓冲 区	ID10~ID3	ID10~ID3	FFH	—
11		ID2~ID0、RTR、DLC	ID2~ID0、RTR、DLC	FFH	—
12		数据字节 1	数据字节 1	FFH	—
13		数据字节 2	数据字节 2	FFH	—
14		数据字节 3	数据字节 3	FFH	—
15		数据字节 4	数据字节 4	FFH	—
16		数据字节 5	数据字节 5	FFH	—
17		数据字节 6	数据字节 6	FFH	—
18		数据字节 7	数据字节 7	FFH	—
19		数据字节 8	数据字节 8	FFH	—

第3章 SJA1000 的原理与使用

续表 3-1

地 址	段 名	工作模式		复位模式	
		读	写	读	写
20	接 收 缓 冲 区	ID10~ID3	ID10~ID3	ID10~ID3	ID10~ID3
21		ID2~ID0,RTR,DLC	ID2~ID0,RTR,DLC	ID2~ID0,RTR,DLC	ID2~ID0,RTR,DLC
22		数据字节 1	数据字节 1	数据字节 1	数据字节 1
23		数据字节 2	数据字节 2	数据字节 2	数据字节 2
24		数据字节 3	数据字节 3	数据字节 3	数据字节 3
25		数据字节 4	数据字节 4	数据字节 4	数据字节 4
26		数据字节 5	数据字节 5	数据字节 5	数据字节 5
27		数据字节 6	数据字节 6	数据字节 6	数据字节 6
28		数据字节 7	数据字节 7	数据字节 7	数据字节 7
29		数据字节 8	数据字节 8	数据字节 8	数据字节 8
30		FFH	—	FFH	—
31		时钟分频寄存器	时钟分频寄存器	时钟分频寄存器	时钟分频寄存器

(2) PeliCAN 协议模式下的存储器分配

在 PeliCAN 模式下 SJA1000 的功能寄存器增加了一些新功能,能支持 CAN 2.0B 协议规定的所有功能。SJA1000 的内部寄存器对 CPU 来说是以外部寄存器形式存在的,而作为片内内存使用。在不同的操作模式,要区分不同的内部地址定义。从 CAN 地址 32 起,所有的内部 RAM(80 字节)映像为 CPU 的接口,具体分配见表 3-2。

表 3-2 PeliCAN 协议模式下的存储器分配

序 号	工作模式		复位模式	
	读	写	读	写
0	模式寄存器	模式寄存器	模式寄存器	模式寄存器
1	00	命令寄存器	00	命令寄存器
2	状态寄存器	—	状态寄存器	—
3	中断寄存器	—	中断寄存器	—
4	中断使能	中断使能	中断使能	中断使能
5	保留	—	保留	—
6	总线定时器 0	—	总线定时器 0	总线定时器 0

CAN 总线技术

续表 3-2

序 号	工作模式				复位模式	
	读		写		读	写
7	总线定时器 1		—		总线定时器 1	总线定时器 1
8	输出控制		—		输出控制	输出控制
9	测试寄存器		测试寄存器		测试寄存器	测试寄存器
10	保留		—		保留	—
11	仲裁丢失捕捉寄存器		—		仲裁丢失捕捉寄存器	—
12	错误代码捕捉寄存器		—		错误代码捕捉寄存器	—
13	错误报警上限寄存器		—		错误报警上限寄存器	—
14	RX 错误计数器		—		RX 错误计数器	RX 错误计数器
15	TX 错误计数器		—		TX 错误计数器	TX 错误计数器
16	标准帧	扩展帧	标准帧	扩展帧	验收代码 0	验收代码 0
17	RX 识别码 1	RX 识别码 1	TX 识别码 1	TX 识别码 1	验收代码 1	验收代码 1
18	RX 识别码 2	RX 识别码 2	TX 识别码 2	TX 识别码 2	验收代码 2	验收代码 2
19	RX 数据 1	RX 识别码 3	TX 数据 1	TX 识别码 3	验收代码 3	验收代码 3
20	RX 数据 2	RX 识别码 4	TX 数据 2	TX 识别码 4	验收屏蔽 0	验收屏蔽 0
21	RX 数据 3	RX 数据 1	TX 数据 3	TX 数据 1	验收屏蔽 1	验收屏蔽 1
22	RX 数据 4	RX 数据 2	TX 数据 4	TX 数据 2	验收屏蔽 2	验收屏蔽 2
23	RX 数据 5	RX 数据 3	TX 数据 5	TX 数据 3	验收屏蔽 3	验收屏蔽 3
24	RX 数据 6	RX 数据 4	TX 数据 6	TX 数据 4	保留	—
25	RX 数据 7	RX 数据 5	TX 数据 7	TX 数据 5	保留	—
26	RX 数据 8	RX 数据 6	TX 数据 8	TX 数据 6	保留	—
27	FIFO	RX 数据 7		TX 数据 7	保留	—
28	FIFO	RX 数据 8		TX 数据 8	保留	—
29	RX 信息计数器		—		RX 信息计数器	—
30	RX 缓冲器起始地址		—		RX 缓冲器起始地址	—
31	时钟分频器		时钟分频器		时钟分频器	时钟分频器
32	内部 RAM0(FIFO)		—		内部 RAM0	内部 RAM0
⋮	⋮		⋮		⋮	⋮
95	内部 RAM63(FIFO)		—		内部 RAM63	内部 RAM63

续表 3-2

序 号	工作模式		复位模式	
	读	写	读	写
96	内部 RAM64(TX 缓冲区)	—	内部 RAM64	内部 RAM64
⋮	⋮	⋮	⋮	⋮
108	内部 RAM76(TX 缓冲区)	—	内部 RAM76	内部 RAM76
109	内部 RAM77(空闲)	—	内部 RAM77	内部 RAM77
110	内部 RAM78(空闲)	—	内部 RAM78	内部 RAM78
111	内部 RAM79(空闲)	—	内部 RAM79	内部 RAM79
112	00	—	00	—
⋮	⋮	⋮	⋮	⋮
127	00	—	00	—

地址 16 用来表示数据帧的类型,可设置为标准帧或扩展帧。

数据发送缓冲区和数据接收缓冲区的地址是重叠的,但它们在物理上是独立的。这部分空间可统称为数据缓冲区,读这部分地址区,则访问数据接收缓冲区;而写这部分地址区,则访问数据发送缓冲区。

3.2 SJA1000 的主要寄存器

3.2.1 模式(控制)寄存器配置及使用方法

地址是 0 的寄存器,在 BasicCAN 模式中,称为控制寄存器(CR);在 Pelican 模式中,称为模式寄存器(MOD)。这个寄存器用于改变 CAN 控制器的行为,控制寄存器主要管理中断系统和复位模式;模式寄存器用于设置 SJA1000 的几种工作方式以及验收滤波器的模式。控制寄存器的功能见表 3-3。模式寄存器的功能见表 3-4。表中的复位值均指硬件复位值,后面讲到的寄存器也一样。对于某些功能位,由软件复位请求或因总线关闭引起的复位,其复位值与硬件复位值不同,若需要请查阅手册。

复位请求位是非常重要的一个功能位,在硬启动或总线状态位设置为 1(总线关闭时)时置 1。如果这些位被软件访问,则其值将发生变化而且会影响内部时钟(内部时钟的频率是外部晶振的 1/2)的下一个上升沿。在外部复位期间,微控制器不能把复位请求位置 0。如果把复位请求位设 0,则微控制器就必须检查该位,以保证外部复位引脚不保持低位。复位请求位的变化同内部分频时钟是同步的。

CAN 总线技术

表 3-3 控制寄存器的功能

位	符 号	名 称	功 能	复位值
CR.0	RR	复位请求	SJA1000 检测到 RR=1,则中止当前的通信任务,进入复位模式;RR=0,则回到工作模式	1
CR.1	RIE	接收中断使能	RIE=1,则开放接收中断;信息被无错接收后,SJA1000 发出一个接收中断信号到微控制器。RIE=0 则关闭此中断	×
CR.2	TIE	发送中断使能	TIE=1,则开放接收中断;当信息被成功发送或发送缓冲器又被访问时(如中止发送命令后),SJA1000 发出一个发送中断信号。TIE=0,则关闭此中断	×
CR.3	EIE	错误中断使能	EIE=1,则开放接收中断;如果出错或总线状态改变,则 SJA1000 发出接收错误中断信号。EIE=0,则关闭此中断	×
CR.4	OIE	溢出中断使能	OIE=1,则开放接收中断;接收数据溢出时,SJA1000 发出溢出中断信号。OIE=0,则关闭此中断	×
CR.5	—	—	保留	1
CR.6	—	—	保留	×
CR.7	—	—	保留	0

SJA1000 进入复位模式的方式不同,复位请求位被设为 0 后,等待的时间也不同:

① 如果是硬件复位或 CPU 初始复位,则等待一个总线空闲信号(11 个弱势位)。

② 如果是 CAN 控制器在重新进入总线开启模式前初始化总线造成的,则要等待 128 个连续的总线空闲。

模式寄存器的功能与控制寄存器相比,有较大变化,除了复位请求位的功能不变外,其余位的功能都不相同。模式寄存器不再管理中断系统,另设了中断寄存器,这里用来管理 Pelican 协议模式下新增的几种工作方式,如表 3-4 所列。

表 3-4 模式寄存器的功能

位	符 号	名 称	功 能	复位值
MOD.0	RM	复位模式	SJA1000 检测到 RR=1,中止当前的通信任务,进入复位模式;RR=0,回到工作模式	1
MOD.1	LOM	只听模式	LOM=1,只听模式,即使成功接收信息,CAN 控制器也不向总线发应答信号;错误计数器停止在当前值	0
MOD.2	STM	自检测模式	STM=1,进入自检测模式;节点使用自接收命令自发自收,即使没有应答 SJA1000 也会成功发送	0

续表 3-4

位	符 号	名 称	功 能	复位值
MOD. 3	AFM	验收滤波器模式	AFM=1, 为单验收滤波器; AFM=0, 为双验收滤波器	0
MOD. 4	SM	睡眠模式	当 SM=1, 没有 CAN 中断等待和总线活动时, SJA1000 进入睡眠模式; SM=0, 从睡眠状态唤醒	0
MOD. 5	—	—	保留	0
MOD. 6	—	—	保留	0
MOD. 7	—	—	保留	0

注意, 如果先进入复位模式, 则 MOD. 1~MOD. 3 是只写的。只听模式使 SJA1000 进入错误消极状态, 不能传送信息。以软件驱动的位速检测和热插时可使用只听模式, 所有其他功能都能像在正常工作模式中一样使用。

睡眠模式位设为 1, 则如果没有总线活动和中断等待, SJA1000 进入睡眠模式。当这 3 个条件不满足时, 则会导致 SM 产生唤醒中断, SJA1000 被唤醒。设置为睡眠模式后, CLKOUT 信号持续至少 15 位的时间, 以允许主微控制器在 CLKOUT 信号电平变低而被锁住之前进入准备模式。SM 电平设为低(唤醒)之后, 总线进入活动状态或 INT 被激活(变低)。唤醒后, 振荡器启动且产生一个唤醒中断。由于总线活动而唤醒的 SJA1000, 直到检测到 11 个连续的隐性位(总线空闲序列)后才能接收这条信息。在复位模式中不能设置 SM 位, 清除复位模式后再一次检测到总线空闲时, SM 的设置才开始有效。

3.2.2 命令寄存器配置及使用方法

命令寄存器控制 SJA1000 传输层的动作, 是只写的。BasicCAN 模式下读命令寄存器的结果总是 FFH, PeliCAN 模式下读命令寄存器的结果总是 00H。因处理的需要, 两条命令之间至少有一个内部时钟周期。表 3-5 是 BasicCAN 模式的命令寄存器(CMR)功能。

表 3-5 BasicCAN 模式下命令寄存器的功能

位	符 号	名 称	功 能	复位值
CMR. 0	TR	发送请求	TR=1, 信息被发送, TR=0, 无动作	1
CMR. 1	AT	中止发送	AT=1, 如果不是在处理过程中, 等待处理的发送请求被取消。 AT=0, 无动作	1
CMR. 2	RRB	释放接收缓冲器	RRB=1, 接收缓冲器中存放信息的内存空间将被释放。RRB=0, 无动作	1
CMR. 3	CDO	清除数据溢出	CDO=1, 清除数据溢出状态位, AFM=0, 无动作	1

CAN 总线技术

续表 3-5

位	符 号	名 称	功 能	复位值
CMR. 4	GTS	进入睡眠模式	当 GTS=1、没有 CAN 中断等待和总线活动时, SJA1000 进入睡眠模式; GTS=0, 无动作	1
CMR. 5	—	—	保留	1
CMR. 6	—	—	保留	1
CMR. 7	—	—	保留	1

PeliCAN 模式下, 命令寄存器 CMR. 4 的功能是自接收模式 (SRR), SRR = 1, 可使 SJA1000 进入自接收模式; SRR = 0, 无动作。其余的功能位与 BasicCAN 模式是相同的。

3.2.3 状态寄存器配置及使用方法

状态寄存器的内容反映了当前 SJA1000 的工作状态, 是只读的。BasicCAN 模式和 PeliCAN 模式下, 状态寄存器的功能相同。

表 3-6 状态寄存器的功能

位	符 号	名 称	功 能	复位值
SR. 0	RBS	接收缓冲器状态	RBS=1, 表示接收缓冲器满, RXFIFO 中有可用信息; RBS=0, 表示接收缓冲器空	0
SR. 1	DOS	数据溢出状态	DOS=1, 有溢出, 表示信息因为 RXFIFO 没有足够的空间而丢失; DOS=0, 没有溢出	0
SR. 2	TBS	发送缓冲器状态	TBS=1, 释放状态, CPU 可以向发送缓冲器写信息; TBS=0, 锁定状态, CPU 不能访问发送缓冲器, 有信息正在等待发送或正在发送	1
SR. 3	TCS	发送完毕状态	TCS=1, 最近一次发送请求被成功处理; TCS=0, 未完成当前发送	1
SR. 4	RS	接收状态	RS=1, SJA1000 在接收信息; RS=0, 未接收	1
SR. 5	TS	发送状态	TS=1, SJA1000 在传送信息; TS=0, 未发送	1
SR. 6	ES	出错状态	ES=1, 至少出现一个错误计数器满或超过 CPU 报警限制; ES=0, 两个错误计数器都在报警限制以下	0
SR. 7	BS	总线状态	BS=1, 总线关闭, SJA1000 退出总线活动; BS=0, 总线开启, SJA1000 加入总线活动	0

当发送错误计数器超过限制 255 时, 总线状态位置为 1 (总线关闭), SJA1000 将设置复位模式位为 1 且产生一个错误报警中断 (相应的中断允许时)。发送错误计数器被置为 127, 接

收错误计数器被清除,这种模式将保持直到 CPU 将复位模式位清除。完成这些之后, SJA1000 通过发送错误计数器的减 1 计数以等待协议规定的最少时间。之后,总线状态位清除(总线开启),错误状态位置为 0 且产生一个错误报警中断(中断允许时),这期间读 TX 错误计数器能给出关于总线关闭修复的状态信息。

如果接收状态位和发送状态位都是 0,则 CAN 总线是空闲的。如果这两位都是 1,则 SJA1000 正在等待下一次空闲。从硬件启动到空闲状态到来,必须检测到 11 个连续的隐性位。总线关闭后会产生 128 个 1 位的连续隐性位。

如果 CPU 在发送缓冲器状态位是 0 时向发送缓冲器写入信息,则写入的字节将丢失且没有提示。

当要接收的信息已经成功通过验收滤波器后, SJA1000 需要在 RXFIFO 中有足够的空间来存储信息描述符和每一个接收的数据字节。如果空间不足,信息就会丢失并表示数据溢出。但如果信息没有成功接收,则不会产生数据溢出。

读出接收缓冲器中的所有信息并用释放接收缓冲器命令内存空间后,接收缓冲器状态位被清除。

3.2.4 中断管理寄存器

关于中断管理的寄存器有中断寄存器和中断允许寄存器。BasicCAN 协议模式下,中断允许位在控制寄存器中定义,所以没有单独的中断允许寄存器。

中断寄存器提供中断源的识别,是只读存储器。当中断寄存器的一位或多位置位时, SJA1000 产生有效中断。微处理器读中断寄存器时,除了接收中断外还把所有位复位。表 3-7 是 Pelican 模式下中断寄存器的功能。BasicCAN 协议模式下,IR.5~IR.7 是保留位。

表 3-7 Pelican 模式下中断寄存器的功能

位	符 号	名 称	功 能	复位值
IR.0	RI	接收中断	当接收缓冲器满且接收中断使能时,RI 置位;接收缓冲器中无可用信息时,RI 复位	0
IR.1	TI	发送中断	发送缓冲器状态从 0 变为 1(释放)且发送中断使能时,TI 置位;对 TI 的读操作会将其清除	0
IR.2	EI	错误中断	错误中断使能时,错误状态位或总线状态位的变化使 EI 置位;对 EI 的读操作会将其清除	0
IR.3	DOI	数据溢出中断	当数据溢出中断使能时,数据溢出状态位 0→1 跳变使 DOI 置位;对 DOI 的读操作会将其清除	0
IR.4	WUI	唤醒中断	退出睡眠模式时,WUI 置位;对 WUI 的读操作会将其清除	0

CAN 总线技术

续表 3-7

位	符 号	名 称	功 能	复位值
IR. 5	EPI	错误认可中断	当 CAN 控制器到达错误认可状态或从错误认可状态又进入错误激活状态,且中断寄存器的 EPIE 位被置位时,EPI 置位;对 EPI 的读操作会将其清除	0
IR. 6	ALI	仲裁丢失中断	当 CAN 控制器丢失仲裁变为接收器,且中断激活寄存器的 ALIE 位置位时,ALI 置位;对 ALI 的读操作会将其清除	0
IR. 7	BEI	总线错误中断	当 CAN 控制器检测到总线错误且中断激活寄存器的 BEIE 置位时,BEI 置位;对 BEI 的读操作会将其清除	0

注意,当 SJA1000 参与总线活动或 CAN 中断正在等待时,微处理器置位睡眠模式位也能产生唤醒中断。

中断使能寄存器可以选择设置系统所用的中断源,它的各个位与中断寄存器的各个位对应相同的中断源。表 3-8 列出了中断激活寄存器的功能。

表 3-8 中断使能寄存器的功能

位	符 号	名 称	功 能	复位值
IER. 0	RIE	接收中断使能	置位时,允许接收中断;置 0 时,禁止接收中断	0
IER. 1	TIE	发送中断使能	置位时,允许发送中断;置 0 时,禁止发送中断	0
IER. 2	EIE	错误中断使能	置位时,允许错误中断;置 0 时,禁止错误中断	0
IER. 3	DOIE	数据溢出中断使能	置位时,允许溢出中断;置 0 时,禁止溢出中断	0
IER. 4	WUIE	唤醒中断使能	置位时,允许唤醒中断;置 0 时,禁止唤醒中断	0
IER. 5	EPIE	错误认可中断使能	置位时,允许错误认可中断;置 0 时,禁止错误认可中断	0
IER. 6	ALIE	仲裁丢失中断使能	置位时,允许仲裁丢失中断;置 0 时,禁止仲裁丢失中断	0
IER. 7	BEIE	总线错误中断使能	置位时,允许总线错误中断;置 0 时,禁止总线错误中断	0

3.2.5 总线定时寄存器配置及使用方法

1. 总线定时寄存器 0

总线定时寄存器 0(BTR0)定义了波特率预设值 BRP 和同步跳转宽度 SJW 的值,功能见表 3-9。复位模式时这个寄存器可以被读/写。PeliCAN 协议模式下,工作模式时还可以读 BTR0。

表 3-9 总线定时寄存器 0 的功能

位	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
位名称	SJW. 1	SJW. 0	BRP. 5	BRP. 4	BRP. 3	BRP. 2	BRP. 1	BRP. 0

① 波特率预置的计算 CAN 系统时钟 t_{SCL} 周期是可编程的,它决定了相应的位时序。CAN 系统时钟由如下公式计算:

$$t_{\text{SCL}} = 2t_{\text{CLK}} \times (32 \times \text{BRP}.5 + 16 \times \text{BRP}.4 + 8 \times \text{BRP}.3 + 4 \times \text{BRP}.2 + 2 \times \text{BRP}.1 + \text{BRP}.0 + 1) \quad (3.1)$$

即 $t_{\text{SCL}} = 2t_{\text{CLK}} \times (\text{BTR0 低 6 位数值} + 1)$, t_{CLK} 是振荡周期。

$$t_{\text{CLK}} = \text{XTAL 的振荡周期} = 1/f_{\text{XTAL}}$$

② 为了补偿在不同总线控制器的时钟振荡器之间的偏差,总线控制器需要在当前传送的相关信号边沿重新同步,这可能使位周期缩短或延长。同步跳转宽度定义了一个位周期可以缩短或延长的时钟周期最大数目。

$$t_{\text{SJW}} = t_{\text{SCL}} \times (2 \times \text{sjw}.1 + \text{sjw}.0 + 1) \quad (3.2)$$

2. 总线定时寄存器 1

总线定时寄存器 1(BTR1)定义了每个位周期的长度、采样点的位置和在每个采样点的采样次数,功能见表 3-10。在复位模式中,它可以被读/写访问。在 PeliCAN 协议模式的工作模式中,它是只读的,在 BasicCAN 模式中总是 FFH。

表 3-10 总线定时寄存器 1 的功能

位	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
位名称	SAM	TSEG2.2	TSEG2.1	TSEG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0

(1) 采样位

SAM=1: 3 次,总线采样 3 次;在低/中速总线上使用,能有效地过滤总线毛刺波。

SAM=0: 单次,总线采样 1 次;适用于高速总线上。

(2) 时间段 1 和时间段 2

时间段 1 和时间段 2 决定了每一个位时间的时钟数目和采样点的位置。

$$t_{\text{TSEG1}} = t_{\text{SCL}} \times (8 \times \text{TSEG1}.3 + 4 \times \text{TSEG1}.2 + 2 \times \text{TSEG1}.1 + \text{TSEG1}.0 + 1) \quad (3.3)$$

$$t_{\text{TSEG2}} = t_{\text{SCL}} \times (4 \times \text{TSEG2}.2 + 2 \times \text{TSEG2}.1 + \text{TSEG2}.0 + 1) \quad (3.4)$$

设同步段为 1 个系统时钟周期, $t_{\text{SYNCSEG}} = t_{\text{CLK}}$, 标称位周期 t_{bit} 为:

$$t_{\text{bit}} = t_{\text{SYNCSEG}} + t_{\text{TSEG1}} + t_{\text{TSEG2}} \quad (3.5)$$

举例: SJA1000 外接 16 MHz 晶振,若想获得 500 kbps 的波特率,则可选择 BTR0 = 0x00, BTR1 = 01C。关于总线时序寄存器与波特率的对应关系,现在又很多计算软件,用户只要将所需波特率输入即可由软件自动计算出配置参数。

定义位周期标量值 $\text{NBT} = t_{\text{bit}} / t_{\text{SCL}}$, 表示位周期包含系统时钟 t_{SCL} 的数目;同理定义 SYNC_SEG(同步段系统时钟周期数)、TSEG1(相位缓冲段 1 系统时钟周期数)和 TSEG2(相位缓冲

CAN 总线技术

段 2 系统时钟周期数)分别是同步段、相位缓冲段 1 和相位缓冲段 2 的标量值,它们的关系如下式:

$$\text{NBT} = t_{\text{bit}}/t_{\text{SCL}} = \text{SYNC_SEG} + \text{TSEG1} + \text{TSEG2} \quad (3.6)$$

$$\text{BRP} = 32\text{BRP}.5 + 16\text{BRP}.4 + 8\text{BRP}.3 + 4\text{BRP}.2 + 2\text{BRP}.1 + \text{BRP}.0 + 1 \quad (3.7)$$

$$\text{SJW} = 2\text{SJW}.1 + \text{SJW}.0 + 1 \quad (3.8)$$

$$\text{SYNC_SEG} = t_{\text{SYNCSEG}}/t_{\text{SCL}} = 1 \quad (3.9)$$

$$\text{TSEG1} = 8\text{TSEG1}.3 + 4\text{TSEG1}.2 + 2\text{TSEG1}.1 + \text{TSEG1}.0 + 1 \quad (3.10)$$

$$\text{TSEG2} = 4\text{TSEG2}.2 + 2\text{TSEG2}.1 + \text{TSEG2}.0 + 1 \quad (3.11)$$

可见,用 T_Q 代表 CAN 系统时钟 t_{SCL} 周期,则 NBT 的可变范围在 3~25 个 T_Q 之间,如表 3-11 所列。

表 3-11 位定时参数范围

单位: T_Q

参 数	BRP	SJW	SAM	SYNC_SEG	TSEG1	TSEG2	NBT
范 围	1~64	1~4	0~1	1	1~16	1~8	3~25

实际中在单次采样模式中可以使用的最小值是 $4T_Q$,在 3 次采样模式中最小值是 $5T_Q$ 。

3. 位时间参数计算规则

CAN 总线周期由 4 个部分组成:同步段(SYNC_SEG)、传播延时段、相位缓冲段 1 (PHASE_SEG1)和相位缓冲段 2 (PHASE_SEG2)。CAN 通信协议中规定通信波特率、位周期的取样点以及取样个数均可以自主设定,这样为用户在网络通信性能的优化上提供了空间。如果位周期取样点偏后,则可以接受较大的信号传输延迟,相应总线的传输距离可以延长;如果周期的取样点接近中间,则可以容忍系统节点间的参考时钟误差。这些矛盾直接影响了网络系统性能,所以总线位定时非常重要,合理的位定时可以提高系统的整体性能。

位时间参数计算规则是保证系统在极端恶劣条件的两个节点间,能够正确接受并解码网络上的信息帧。极端恶劣条件是指这两个节点的时钟振荡偏差在系统容忍偏差极限的两端,并且两个节点间具有最大的传输延迟。在没有噪音干扰的正常通信情况下,相位误差累计的最坏情况是:重同步边沿之间间隔有 10 个位周期(5 个显性位后面跟 5 个隐性位)。实际系统都运行在噪音环境中,这可能导致重同步边沿之间的间隔超过 10 个位周期,这时必须进行严格的采样,否则可能进入错误处理模式。下面对振荡器容差和传播延时因素进行分析。

(1) 振荡器容差

CAN 网络中的每个节点都有独立的参考时钟,并从振荡器基准取得位定时。实际应用中,振荡器基准频率 f_{CLK} 因为制造工艺、初始的容差偏移、运行时间和环境温度的变化而偏离了它的基准频率,最后这些偏离量之和就形成了振荡器容差 Δf ,导致不能精确同步。振荡器容差是一个相对的容差,表示和振荡器基准频率的偏离,标准化成额定的频率是

$$\Delta f = \left| \frac{f_{\text{CLK, max/min}} - f_{\text{CLK, nom}}}{f_{\text{CLK, nom}}} \right| \quad (3.12)$$

由于 CAN 系统时钟直接从振荡器基准频率取得, Δf 也表示系统时钟的相对容差, 相应的系统时钟周期也会有误差。因此, 系统时钟的最小值和最大值大约如下:

$$t_{\text{SCL, min}} = \frac{t_{\text{SCL, nom}}}{1 + \Delta f} \approx t_{\text{SCL, nom}} \times (1 - \Delta f) \quad (3.13)$$

$$t_{\text{SCL, max}} = \frac{t_{\text{SCL, nom}}}{1 - \Delta f} \approx t_{\text{SCL, nom}} \times (1 + \Delta f) \quad (3.14)$$

当 $\Delta f \ll 1$ 时, 式(3.13)和式(3.14)才成立。实际系统中使用的时钟基准, 像石英振荡器 $\Delta f < 0.1\%$ 、锁相环取得的频率 $\Delta f < 0.5\%$ 、陶瓷谐振器 $\Delta f < 1.2\%$, 都可以满足这个条件。

(2) 传输延迟

CAN 总线采用无破坏性的位仲裁机制, 因此, 传输延时非常重要, 若传输延时过长则导致无效的访问仲裁。CAN 系统中传输延迟来源于节点之间竞争访问网络时的非破坏性仲裁以及帧内应答。仲裁在标识符域产生, 表示有多个节点同时将自己的标识符位发送到总线上。由于节点在位的边沿同步, 系统的传输延迟太长就会导致仲裁无效。最终, CAN 系统的各种延迟在给定的位速率下限制了最大的网络总线长度。两个节点 A 和 B 之间的单方向传输延迟定义为 $t_{\text{prop(A,B)}}$ 。传输延时时间由物理总线延时 t_{bus} 、总线驱动器延时 t_{trc} 和其他设备传输延时 t_{oht} 共同决定, 其他设备包括 CAN 控制器、隔离光偶等, 用下式表示:

$$t_{\text{prop}} = 2(t_{\text{bus}} + t_{\text{trc}} + t_{\text{oht}}) \quad (3.15)$$

其中, t_{trc} 、 t_{oht} 为收发器的有效回路延时时间, 在器件参数表中可以查得。

定义传输延时的标量值 $\text{PROP} = \frac{t_{\text{prop}}}{t_{\text{SCL}}}$, 参与以后位值计算。

(3) 时间参数设定

位定时的计算读者可以参考相关文献, 在此做一总结:

1) 1 个采样点设置

$$\text{SJW}_{\min} = \text{MAX}\{(20 \times \text{NBT} \times \Delta f) / (1 - \Delta f),$$

$$(\text{NBT} \times (1 - 25 \times \Delta f) - \text{PROP}_{\max} - (1 - \Delta f) + \text{PROP}_{\min} / 2) / (1 - \Delta f)\} \quad (3.16)$$

$$\text{SJW}_{\max} = 4 \quad (3.17)$$

$$\text{TSEG2}_{\min} = \text{MAX}\{2, \text{SJW}\} \quad (3.18)$$

$$\begin{aligned} \text{TSEG2}_{\max} = & \text{MIN}\{8, \text{NBT} \times (1 - 25 \times \Delta f) - \text{PROP}_{\max}\} / (1 - \Delta f), \\ & (\text{NBT} \times (1 - 25 \times \Delta f) - \text{PROP}_{\max} - (1 - \Delta f) + \text{PROP}_{\min} / 2) / (1 - \Delta f) \} \end{aligned} \quad (3.19)$$

2) 3 个采样点设置

$$\text{SJW}_{\min} = \text{MAX}\{(20 \times \text{NBT} \times \Delta f) / (1 - \Delta f),$$

CAN 总线技术

$$(\text{NBT} \times (1 - 25 \times \Delta f) - \text{PROP}_{\max} - (1 - \Delta f) + \text{PROP}_{\min}/2)/(1 - \Delta f) \quad (3.20)$$

$$\text{SJW}_{\max} = 4 \quad (3.21)$$

$$\text{TSEG2}_{\min} = \text{MAX}\{3, \text{SJW}\} \quad (3.22)$$

$$\begin{aligned} \text{TSEG2}_{\max} = & \text{MIN}\{8, (\text{NBT} \times (1 - 25 \times \Delta f) - \text{PROP}_{\max} - 2 \times (1 - \Delta f))/(1 - \Delta f), \\ & (\text{NBT} \times (1 - 25 \times \Delta f) - \text{PROP}_{\max} - 3 \times (1 - \Delta f) + \text{PROP}_{\min})/2)/(1 - \Delta f)\} \end{aligned} \quad (3.23)$$

在计算 $\text{SJW}_{\min}$ 时,取大于计算数值的最小整数;在计算 $\text{TSEG2}_{\max}$ 时,取小于计算数值的最大整数。

(4) 参数计算步骤及举例

某 CAN 通信系统采用 1 个取样点模式,其他参数指标如表 3-12 所列。

表 3-12 CAN 通信系统参数

参 数	说 明	最小值	典型值	最大值
$f_{\text{bit}}/(\text{kbps})$	通信波特率		250	
$t_{\text{bit}}/\mu\text{s}$	位周期时间		4	
$f_{\text{CLK}}/\text{MHz}$	CAN 控制器的时钟频率			24
$\Delta f/\%$	时钟频率偏差			1.0
t_{trc}/ns	总线驱动器延时	30	75	157
t_{oth}/ns	其他设备延时	15		40
$\delta/(\text{ns} \cdot \text{m}^{-1})$	线路延时	5		6.5
L/m	节点间总线长度	3		95
t_{bus}/ns	计算得到线路延时 $t_{\text{bus}} = L \cdot \delta$	15		618
$t_{\text{prop}}/\text{ns}$	计算得到传输延时(式 3.15)	120		1 630

① 确定可能的 BRP、NBT 和 PROP。

由式(3.1)、式(3.5)、式(3.6)、式(3.7)得到

$$\text{NBT} = f_{\text{CLK}}/(2f_{\text{bit}} \cdot \text{BRP}), \text{即 } \text{NBT} \cdot \text{BRP} = f_{\text{CLK}}/2f_{\text{bit}} \quad (3.24)$$

$$\text{PROP} = \text{NBT} \times f_{\text{bit}} \times t_{\text{prop}} \quad (3.25)$$

将参数代入式 3.24,则得到 $\text{NBT} \cdot \text{BRP} = 250 \times 10^6 / 2 \times 250 \times 10^3 = 48$,而 NBT 取值为 3~25,所以 NBT 和 BRP 所有可能的组合如表 3-13 所列。

② 计算 NBT_{min} 和 NBT_{max}。

由式(3.16)得出:

$$\text{NBT} \leq \min\{\text{SJW}_{\max} \times (1 - \Delta f)/20\Delta f, (\text{SJW}_{\max} \times (1 + \Delta f) - 1 + \Delta f)/(20\Delta f - \text{PROP}_{\min}/(20\Delta f - f_{\text{bit}} \times t_{\text{propmin}}))\}$$

将参数代入计算得 $\text{NBT} = \min\{19.8, 17.94\}$,则 $\text{NBT} \leq 17.94$ 。

表 3-13 CAN 通信系统参数

f_{CLK}	NBT	BRP	$t_{SCL} = (2BRP/f_{CLK})$	$PROP_{max}$	$PROP_{min}$
24 MHz	4	12	1 000 ns	1.63	0.12
	6	8	666.6 ns	2.45	0.18
	8	6	500 ns	3.26	0.24
	12	4	333.3 ns	4.89	0.36
	16	3	250 ns	6.52	0.48
	24	2	166.6 ns	9.78	0.72

同理,将参数代入式(3.18)得出 $NBT \geq 8$ 。为了方便取采样点,取 $NBT=16$ 。

③ 根据式(3.16)计算 SJW_{min} ,如表 3-14 所列。

④ 根据式(3.18)计算 $TSEG2_{min}$,如表 3-14 所列。

⑤ 根据式(3.19)计算 $TSEG2_{max}$,如表 3-14 所列。

⑥ 确定寄存器设置数值,由公式可得 $RTR0=0xC2$, $RTR1=0x3A$ 。

表 3-14 计算 SJW、TSEG1、TSEG2

参 数	最小值	最大值	确定值
SJW	$\{3.23, 3.67\}_{max}$	4	4
TSEG2	$\{2, SJW\}_{max}$	$\{8, 5.54, 4.78\}_{min}$	4
TSEG1	$TSEG1 = NBT - TSEG2 - SYNC_SEG = 16 - 4 - 1$		11

以上为位时间参数的计算规则。在实际监控系统中,以此规则根据适当的方法进行位定时参数确定和优化,从而使系统性能达到最优。

3.2.6 输出控制寄存器

输出控制寄存器(OCR)控制 SJA1000 的发送电路,可以配置成不同输出驱动方式。在复位模式中此寄存器可被读/写访问。表 3-15 说明了输出控制寄存器的功能。

表 3-15 输出控制寄存器的功能

位	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
名 称	OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCMODE1	OCMODE0

OCTPx 和 OCTNx 可编程设置输出驱动器的特性,可设置为悬空、上拉、下拉、推挽这 4 种驱动方式。OCPOLx 控制输出端极性。

OCMODE1 和 OCMODE0 设置输出模式,取值为 00~11 时,分别对应双相输出模式、测

CAN 总线技术

试输出模式、正常输出模式、时钟输出模式。

正常模式中,位序列通过 TX0 和 TX1 输出,输出驱动引脚 TX0 和 TX1 的电平取决于驱动器的特性和输出端极性。

时钟输出模式中,TX0 引脚和正常模式中是相同的,但是 TX1 上的数据流被发送时钟 TXCLK 代替了,发送时钟的上升沿标志着一个位的开始,时钟脉冲宽度是一个系统时钟周期。

双相输出模式中的输出位随时间和位序变化而触发。如果总线控制器被发送器从总线上通电退耦,则位流不允许含有直流成分。在隐性位期间输出悬空,显性位轮流使用 TX0 或 TX1 发送。例如,第一个显性位在 TX0 上发送,第二个显性位在 TX1 上发送,第三个显性位在 TX0 上发送等,依此类推。

在测试输出模式中,RX 上的电平在下一个系统时钟的上升沿映射到 TXn 上,系统时钟与输出控制寄存器中定义的极性一致。

3.2.7 时钟分频寄存器

时钟分频寄存器控制输出时钟的频率。而且它还控制着 TX1 上的专用接收中断脉冲、接收比较通道以及 BasicCAN 模式与 PeliCAN 模式的选择。硬件复位后寄存器的默认状态是 NXP 模式 12 分频或 Intel 模式 2 分频。

软件复位时,时钟分频寄存器不受影响。时钟分频寄存器的功能如表 3-16 所列。

表 3-16 时钟分频寄存器的功能

位	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
名称	CAN 模式	CBP	RXINTEN	0	关闭时钟	CD. 2	CD. 1	CD. 0

CD. 2~CD. 0 用来定义外部 CLKOUT 引脚上的输出时钟频率,见表 3-17,其中, f_{osc} 是外部振荡器(XTAL)频率。

表 3-17 时钟分频寄存器的功能

CD. 2	CD. 1	CD. 0	时钟频率	CD. 2	CD. 1	CD. 0	时钟频率
0	0	0	$f_{osc}/2$	1	0	0	$f_{osc}/10$
0	0	1	$f_{osc}/4$	1	0	1	$f_{osc}/12$
0	1	0	$f_{osc}/6$	1	1	0	$f_{osc}/14$
0	1	1	$f_{osc}/8$	1	1	1	f_{osc}

在复位模式中可以把时钟关闭位置 1,以关闭外部引脚 CLKOUT 的时钟输出。如果置位此位,则 CLKOUT 引脚在睡眠模式中是低电平,而其他情况下是高电平。

RXINTEN 位允许 TX1 输出作为专用接收中断输出。当一条已接收的信息成功地通过验收滤波器时, TX1 引脚输出一个位时间长度的接收中断脉冲(帧的最后一位期间)。发送输出阶段, SJA1000 应该工作在正常输出模式。

在复位模式中可以置位 CDR. 6, 使输入数据不经过 CAN 输入比较器。此时内部延时减少, 总线长度得以增加。如果 CBP 置位, 则只有 RX0 激活。RX1 不使用时应连接到一个确定的电平, 如 V_{SS} 。

CDR. 7 定义了 CAN 模式, 如果 CDR. 7=0, 则 SJA1000 工作于 BasicCAN 模式; 否则, 工作于 PeliCAN 模式。只有在复位模式中此位才可以写。

3.2.8 其他寄存器配置及使用方法

除了上述寄存器, 在 PeliCAN 协议模式下, 还增加了几个功能寄存器, 以满足新增功能的需要。这些寄存器是仲裁丢失捕捉寄存器、错误代码捕捉寄存器、错误报警上限寄存器、接收错误计数寄存器 RXERR、发送错误计数寄存器 TXERR、接收信息计数器 RMC、接收缓冲器起始地址寄存器 RBSA。这几个寄存器功能比较简单, 这里只作简要介绍。

对于接收错误计数寄存器 RXERR 和发送错误计数寄存器 TXERR, 在 BasicCAN 模式, 它们不可访问; 在 PeliCAN 模式, 可以读/写。它们按照 CAN 协议的错误界定规则, 修改计数值, 表示 SJA1000 当前通信错误的严重程度。

接收信息计数器 RMC 指示接收缓冲区中存储的信息数目, 计数器每次接收后加 1, 每次释放后接收缓冲器减 1, 复位后该寄存器清 0。

接收缓冲器起始地址寄存器反映了当前可用来存储信息的内部 RAM 地址, 位于接收缓冲器窗口中的信息会存储到此地址开始的一段 RAM 中。CAN 地址 32 以后的内部 RAM 地址区可以被 CPU 读/写访问。例如, RBSA 被设置为 24, 则当前在接收缓冲器窗口中的可视信息被存储在内部 RAM 的起始地址是 24。因为 RAM 区在 CAN 地址空间的起始地址为 32 (相当于 RAM 地址 0), 所以这条信息存储在 CAN 地址 56 及随后的几个字节:

$$\text{CAN 地址} = \text{RBSA} + 32 = 24 + 32 = 56$$

如果信息超过 RAM 地址 63, 则从地址 0 继续。当 FIFO 中至少有一条可用信息时, 则执行释放接收缓冲器命令。RBSA 在下一条信息开始的时候更新。

错误报警上限寄存器用来设置出错的上限值, 默认值是 96。只有进入复位模式才有可能被改变, 直到退出复位模式, 新的寄存器内容引起的错误报警中断才有可能发生。

仲裁丢失捕捉寄存器描述了仲裁丢失的位置。仲裁丢失时会产生相应的仲裁丢失中断, 同时, 位流处理器的当前位位置被捕捉送入仲裁丢失捕捉寄存器, 这个值一直到用户通过软件读寄存器才会变。仲裁丢失捕捉寄存器被读一次之后, 新的仲裁丢失中断才有效。

仲裁丢失捕捉寄存器的低 5 位值有效, 高 3 位保留。对于扩展帧格式, 寄存器取值为 0~10, 分别表示仲裁丢失在识别码的第 1~11 位; 寄存器取值 11、12, 表示仲裁丢失在 SRTR 位、

CAN 总线技术

IDE 位;寄存器取值为 13~30,分别表示仲裁丢失在识别码的第 12~29 位;寄存器取值为 31,表示仲裁丢失在 RTR 位。对于标准帧格式,寄存器取值 0~10,分别表示仲裁丢失在识别码的第 1~11 位;寄存器取值 11、12,表示仲裁丢失在 RTR 位、IDE 位。

错误代码捕捉寄存器描述了总线错误的类型和位置信息。表 3-18 是寄存器各功能位的说明。

表 3-18 错误代码捕捉寄存器的位功能

位	符 号	名 称	功 能
ECC. 0	SEG0	段 0	这几个功能位的值描述错误发生的具体位置,见表 3-19
ECC. 1	SEG1	段 1	
ECC. 2	SEG2	段 2	
ECC. 3	SEG3	段 3	
ECC. 4	SEG4	段 4	
ECC. 5	DIR	方向	DIR=1,表示接收时发生的错误 DIR=0,表示发送时发生的错误
ECC. 6	ERRC0	错误代码 0	SEG1 和 SEG0 取值 00~11,对应的错误类型是位错误、格式错误、填充错误及其他错误
ECC. 7	ERRC1	错误代码 1	

总线发生错误时产生相应的错误中断,同时,位流处理器的当前位置被捕捉送入错误代码捕捉寄存器。该寄存器的内容直到用户通过软件读出时都是不变的,读出后捕捉机制又被激活了。访问中断寄存器期间,中断寄存器中相应的中断标志位被清除,新的总线中断直到捕捉寄存器被读出一次才能产生。

表 3-19 错误位置指示

SEG4	SEG3	SEG2	SEG1	SEG0	错误位置	SEG4	SEG3	SEG2	SEG1	SEG0	错误位置
0	0	0	1	1	帧开始	0	1	0	1	0	数据区
0	0	0	1	0	ID. 28~ID. 21	0	1	0	0	0	CRC 序列
0	0	1	1	0	ID. 20~ID. 18	1	1	0	0	0	CRC 界定符
0	0	1	0	0	SRTR 位	1	1	0	0	1	应答通道
0	0	1	0	1	IDE 位	1	1	0	1	1	应答定义符
0	0	1	1	1	ID. 17~ID. 13	1	1	0	1	0	帧结束
0	1	1	1	1	ID. 12~ID. 5	1	0	0	1	0	中止
0	1	1	1	0	ID. 4~ID. 0	1	0	0	0	1	活动错误标志
0	1	1	0	0	RTR 位	1	0	1	1	0	消极错误标志
0	1	1	0	1	保留位 1	1	0	0	1	1	显性位误差
0	1	0	0	1	保留位 0	1	0	1	1	1	错误界定符
0	1	0	1	1	数据长度代码	1	1	1	0	0	溢出标志

3.3 通信及滤波器原理

3.3.1 发送数据缓冲区

发送数据缓冲区是用来要 SJA1000 发送的信息的,分为描述符区和数据区。BasicCAN 模式下,描述符占 2 字节,见表 3-20。

表 3-20 BasicCAN 模式下描述符的结构

CAN 地址	位							
	7	6	5	4	3	2	1	0
10	ID. 10	ID. 9	ID. 8	ID. 7	ID. 6	ID. 5	ID. 4	ID. 3
11	ID. 2	ID. 1	ID. 0	RTR	DLC. 3	DLC. 2	DLC. 1	DLC. 0

描述符包括 11 位标识符,一个远程请求位,4 位数据长度码。数据长度码不应超过 8;如果选择的值超过 8,则按 8 处理。CAN 地址 12~19 是存储数据字节的存储单元。

在 PeliCAN 模式下,发送的每一帧都可分为描述符区和数据区。描述符区的第一个字节(CAN 地址 16)是帧信息字节,用来描述帧格式(标准或扩展)、帧类型(远程帧或数据帧)及数据长度。

帧信息字节中,FF=0,表示标准帧,FF=1,表示扩展帧。表 3-21 是扩展帧的描述符结构,标识符增加到了 29 位,占用 4 字节。对于标准帧的描述符,只用地地址 16~18 这 3 字节,标识符只用 11 位。

表 3-21 PeliCAN 模式下描述符的结构(扩展帧)

CAN 地址	位							
	7	6	5	4	3	2	1	0
16	FF	RTR	×	×	DLC. 3	DLC. 2	DLC. 1	DLC. 0
17	ID. 28	ID. 27	ID. 26	ID. 25	ID. 24	ID. 23	ID. 22	ID. 21
18	ID. 20	ID. 19	ID. 18	ID. 17	ID. 16	ID. 15	ID. 14	ID. 13
19	ID. 12	ID. 11	ID. 10	ID. 9	ID. 8	ID. 7	ID. 6	ID. 5
20	ID. 4	ID. 3	ID. 2	ID. 1	ID. 0	×	×	×

3.3.2 接收缓冲区

接收缓冲区共有 64 字节,一次可以存储多少条信息取决于数据的长度。如果接收缓冲区

CAN 总线技术

中没有足够的空间来存储新的信息, SJA1000 会产生数据溢出, 已部分写入接收缓冲区的信息将被删除, 这种情况可以通过状态寄存器和数据溢出中断表示出来。

在 BasicCAN 模式下, 接收缓冲区的结构和发送缓冲器类似。接收缓冲区是 RXFIFO 中可访问的部分, 位于 CAN 地址的 20~29, 即接收缓冲区的 64 字节只分配 10 个地址, 对应最早接收的那一条信息。一条信息读取后执行释放接收缓冲区命令, 则下一条信息进入 CAN 地址的 20~29, 等待读取。

在 PeliCAN 模式下, 接收缓冲区的结构和发送缓冲器结构相同, 并且逻辑地址也相同。读操作时访问接收缓冲区, 写操作时访问发送缓冲器。

3.3.3 验收滤波器

验收滤波器包括验收代码寄存器(ACR)和验收屏蔽寄存器(AMR)。信息的标识符和验收滤波器中预设值一致时, 才会被 SJA1000 接收。验收码寄存器定义所要接收信息标识符的值。验收屏蔽寄存器的某位值为 1, 则对应的标识符位为需要验收; 某位值为 0, 则对应的标识符位不需验收。

BasicCAN 模式下, 验收滤波器有 1 字节长度。验收代码位(AC. 7~AC. 0)和信息标识符的高 8 位(ID. 10~ID. 3)相等且与验收屏蔽位(AM. 7~AM. 0)的对应位相或为 1, 该信息可通过验收滤波器被接收。即满足以下方程:

$$[(ID. 10 \sim ID. 3) = (AC. 7 \sim AC. 0)] \vee (AM. 7 \sim AM. 0) = 11111111$$

PeliCAN 模式下, 验收滤波器长度是 4 字节。在模式寄存器中设置 MOD. 3(AFM)的值, 可选择两种不同的验收滤波模式:

AFM 位是 1, 配置为单滤波器模式;

AFM 位是 0, 配置为双滤波器模式。

1. 单滤波器配置

这种配置是设定长 4 个字节的长滤波器, 接收校验情况要根据当前所接收的帧格式来确定。

(1) 标准帧

如果接收到标准帧格式信息, 则信息的完整标识符(包括 RTR 位)和前两个数据字节都参与接收校验。如果是远程帧, 则数据长度为 0 或 1 的数据帧也可以接收。接收标准帧的校验配置如图 3-6 所示。

要使信息成功接收, 则每位比较结果必须是 1, 即信号接收。应注意的是, AMR1 和 ACR1 的最低 4 位未用, 为了与将来的产品兼容, 可通过将 AMR1. 3、AMR1. 2、AMR1. 1、AMR1 设为 1(即无关位), 对它们不验收。

(2) 扩展帧

如果接收的信息是扩展帧格式, 则全部识别码及 RTR 位都将接受滤波。接收扩展帧的校验配置如图 3-7 所示。AMR3. 1、AMR3. 0 必须设为 1, 以与将来的产品兼容。

第3章 SJA1000 的原理与使用

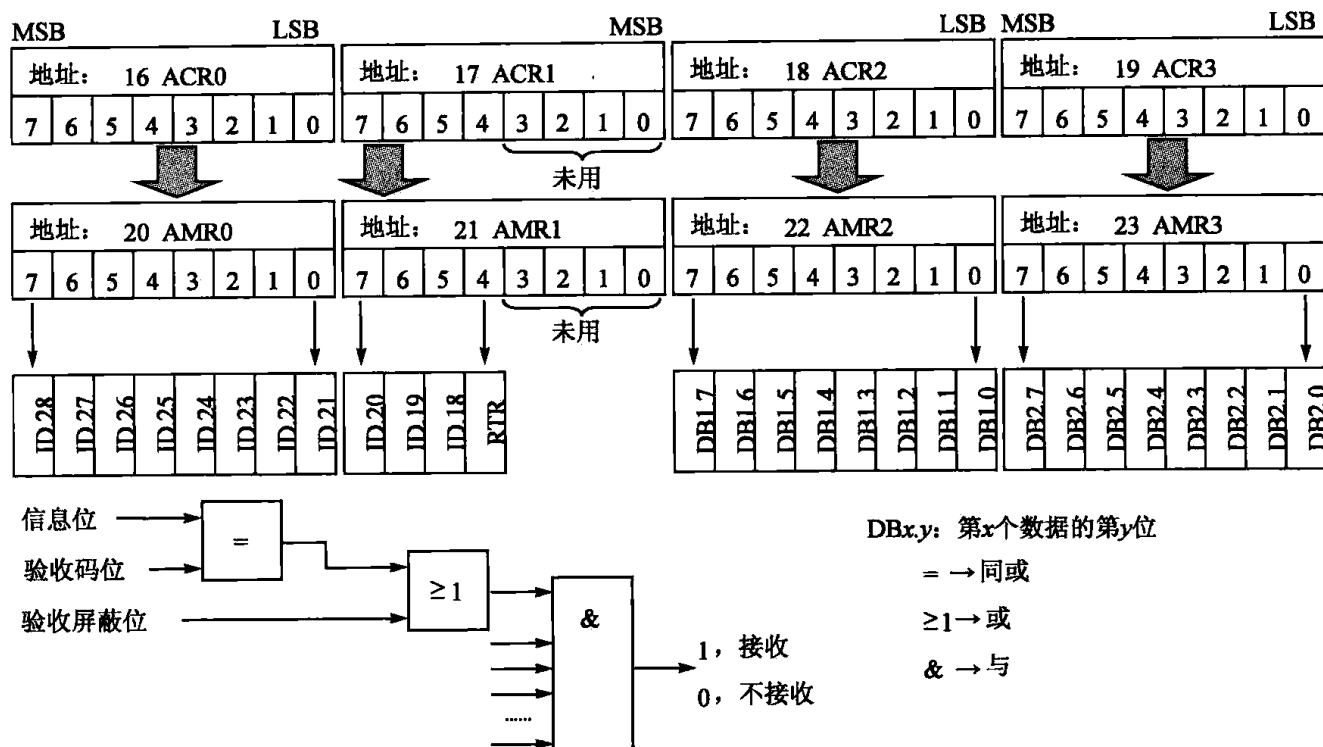


图 3-6 接收标准帧的单滤波器配置

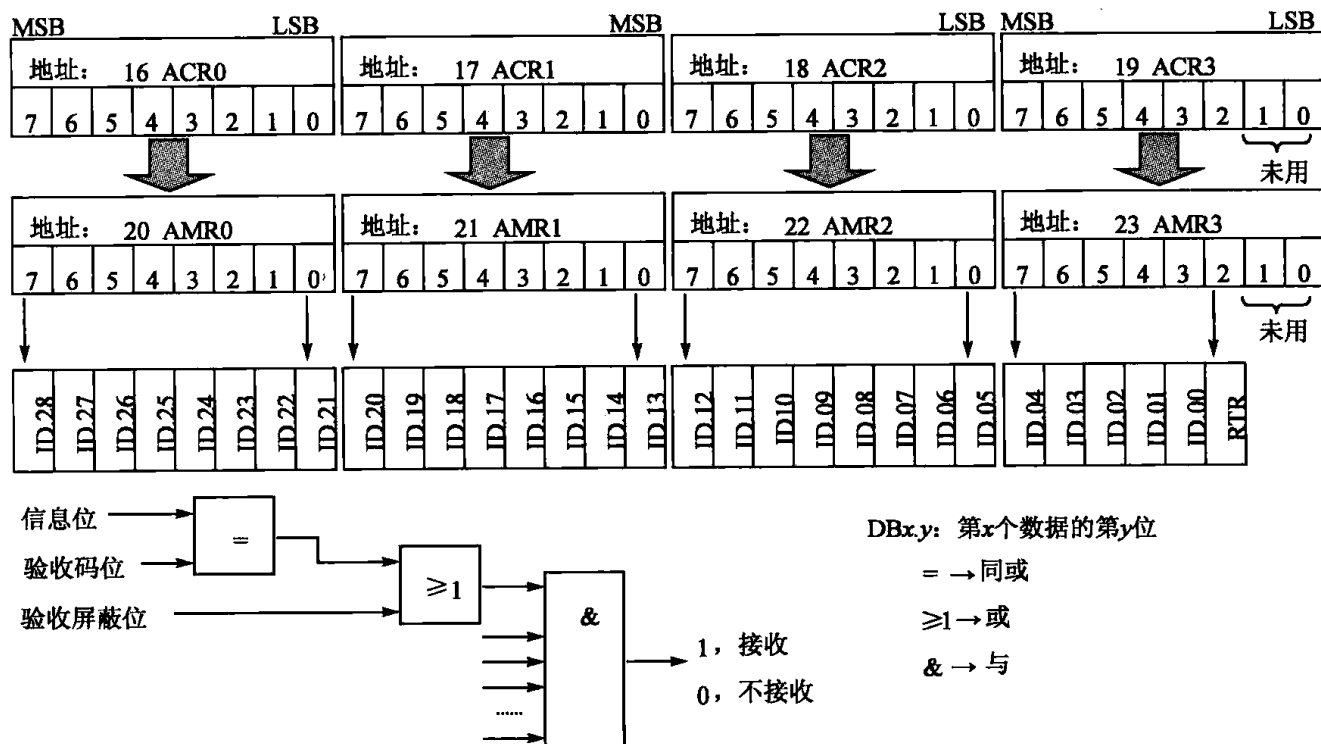


图 3-7 接收扩展帧的单滤波器配置

在这种配置中,定义了两个短滤波器。每条接收到的信息与两个滤波器都进行比较,以确定这条信息是否应该传送到接收缓冲区。只要两个滤波器有一个为接收,那么这条信息就是有效的。信息字节和验收滤波器字节直接的位关系取决于当前接收的帧格式。

如果接收到标准帧信息,则两个滤波器校验的范围不同。第一个滤波器用来比较包括 RTR 位在内的标准标识符,并且还有第一个数据字节。第二个滤波器只是比较包括 RTR 的标准标识符。接收标准帧的双滤波配置情况如图 3-8 所示。



为了成功地接收一条信息,信息的所有位必须至少被一个滤波器所接收。一旦 RTR 位置 1 或数据长度码为 0,即不存在数据位,则此时只要从 ID. 28 至 RTR 位被接收,此信息便可以通过滤波器 1。如果在滤波器 1 中不需要数据字节参与滤波,AMR1 和 AMR3 的最低 4 位必须置 1(无关),那么两个滤波器都工作在标准标识符滤波范围内。

(2) 扩展帧

当接收到扩展帧信息时,两个滤波器是相同的,都比较扩展标识符的前两个字节。只要通过一个滤波器的校验,那么此信息就可以成功地接收到。接收扩展帧的校验滤波配置如图 3-9 所示。

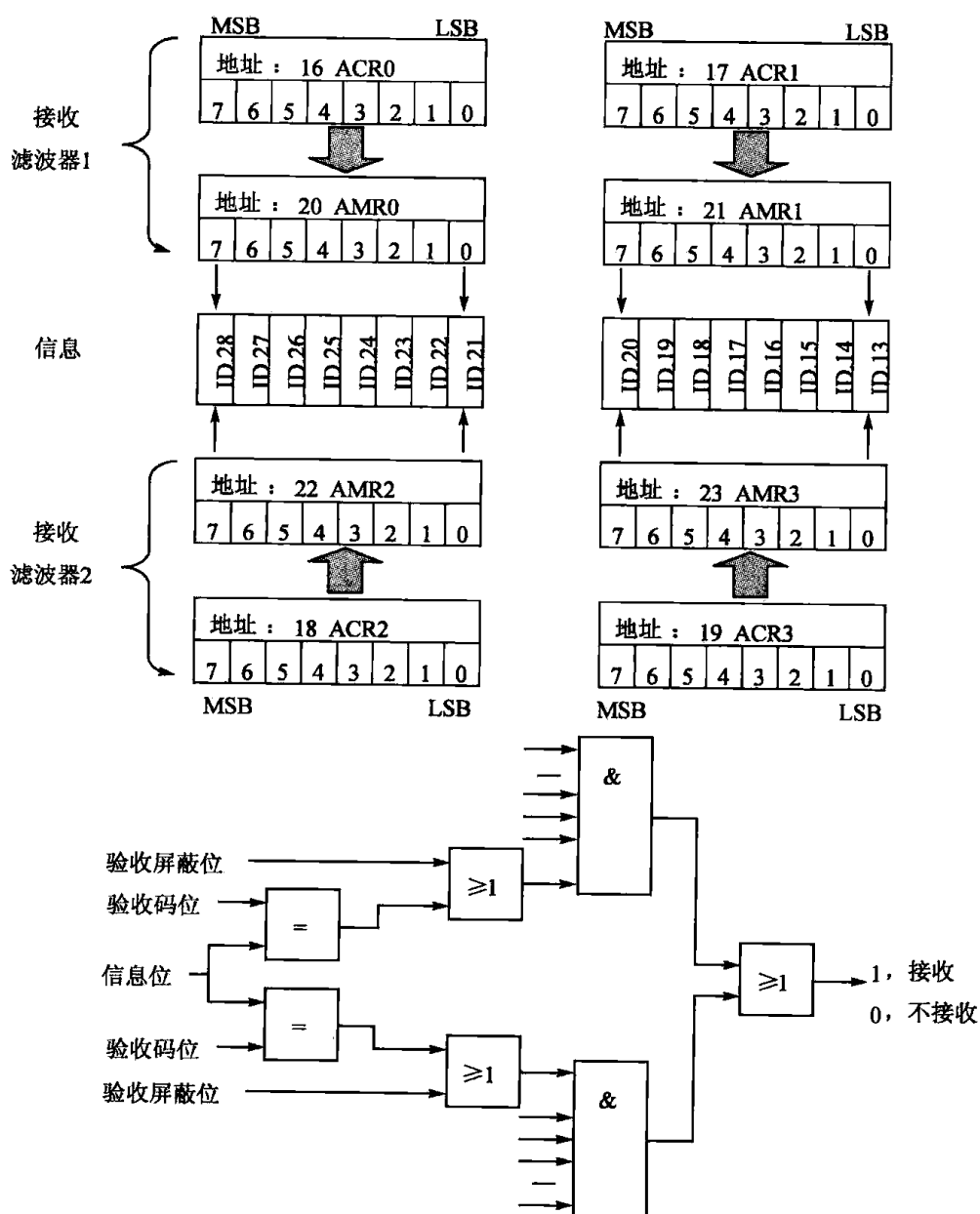


图 3-9 接收扩展帧的双滤波器配置

3.4.1 SJA1000 典型应用接口电路

80C51 的 P0 口接 SJA1000 的数据/地址总线, SJA1000 的片选信号接 P2 口任意一个口线, 这里接 P2.1。时钟电路采用 SJA1000 的内部时钟, CLKOUT 输出时钟给 80C51。读/写允许信号及地址锁存信号对应连接即可。

6N137 是高速光电耦合器,接在输入输出信号线上,从而把外部 CAN 总线与 SJA1000 隔离,减少干扰。82C250 是总线驱动器,可以提高信号驱动能力。

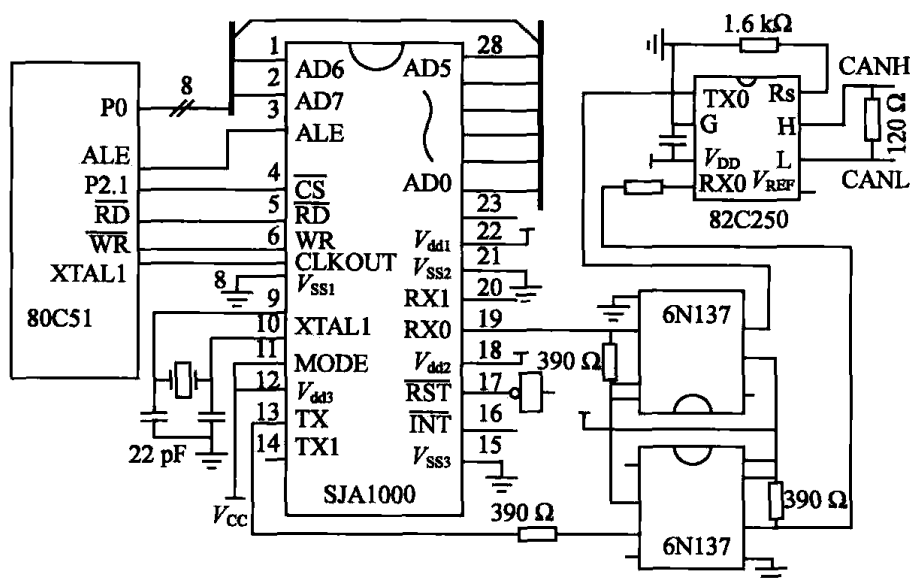


图 3-10 SJA1000 与 80C51 的接口电路

3.4.2 SJA1000 初始化程序设计

SJA1000 要完成正常的 CAN 通信,需要先进行必要的初始化参数设置。这些初始化参数包括中断系统设置、通信速率设置、验收滤波器设置以及输出方式等。这些设置实际上是对 SJA1000 内部相关寄存器的写操作。SJA1000 先进入复位模式,然后进行初始化工作;初始化完成后,退出复位模式,进入工作模式。图 3-11 说明了初始化的一般流程。

在上电或需要重新配置参数时, SJA1000 进入初始化程序。首先, 微处理器关闭 SJA1000 的中断(相对微处理器的一个外部中断), 然后写 SJA1000 的模式寄存器, 进入复位模式。接

下来应该访问时钟寄存器,在这里主要确定使用 BasicCAN 模式还是 PeliCAN 模式。根据程序设计的需要,在中断使能寄存器中选择要使用的中断。验收代码寄存器和验收屏蔽寄存器决定了报文的选择性接收,在 PeliCAN 模式下,它可以配置成单滤波模式或双滤波模式。通信系统的一个最基本参数是通信速率,总线定时寄存器 0 和总线定时寄存器 1 用来设置位时间长度、位采样等,从而决定了通信速率。输出控制寄存器配置输出位流的电平驱动形式。最后,将模式寄存器复位模式请求位清 0, SJA1000 进入工作模式,初始化完成。

针对图 3-10 的电路,设 SJA1000 工作于 PeliCAN 模式,典型的初始化程序如下:

```

ORG      0H
MOV      DPTR, #0           ;指向模式寄存器
MOV      A, #01H           ;进入复位模式,
MOVX     @DPTR, A
MOV      DPTR, #31         ;指向时钟分频寄存器
MOV      A, #88H           ;选择 PeliCAN 模式,关闭时钟输出
MOVX     @DPTR, A
MOV      DPTR, #4          ;指向中断允许寄存器
MOV      A, #07H           ;开放错误中断,
MOVX     @DPTR, A          ;接收、发送中断
MOV      DPTR, #20         ;指向验收屏蔽寄存器
MOV      A, #AMR0           ;验收屏蔽寄存器赋值
MOVX     @DPTR, A
INC      DPTR
MOV      A, #AMR1
MOVX     @DPTR, A
INC      DPTR
MOV      A, #AMR2
MOVX     @DPTR, A
INC      DPTR
MOV      A, #AMR3
MOVX     @DPTR, A
MOV      A, #ACR0           ;验收代码寄存器赋值
MOVX     @DPTR, A
INC      DPTR
MOV      A, #ACR1

```

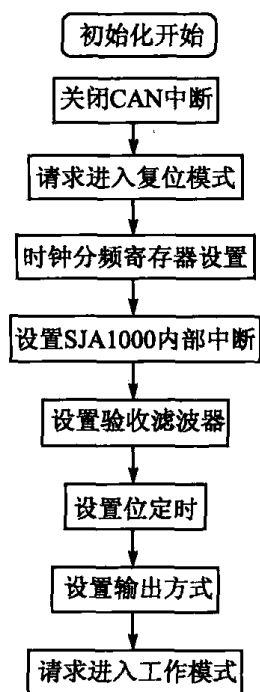


图 3-11 初始化流程

CAN 总线技术

```

MOVX    @DPTR,A
INC      DPTR
MOV      A, # ACR2
MOVX    @DPTR,A
INC      DPTR
MOV      A, # ACR3
MOVX    @DPTR,A
MOV      DPTR, # 6           ;总线定时寄存器 0
MOV      A, # 3
MOVX    @DPTR,A
MOV      DPTR, # 7           ;总线定时寄存器 1
MOV      A, # 0FFH
MOVX    @DPTR,A
MOV      DPTR, # 8           ;输出控制寄存器
MOV      A, # 0AAH
MOVX    @DPTR,A
MOV      DPTR, # 30          ;接收缓冲区起始地址寄存器
MOV      A, # 0              ;起始地址设为 0
MOVX    @DPTR,A
MOV      DPTR, # 12          ;错误代码捕捉寄存器清 0
MOV      A, # 0
MOVX    @DPTR,A
MOV      DPTR, # 14          ;接收错误计数器清 0
MOV      A, # 0
MOVX    @DPTR,A
MOV      DPTR, # 15          ;发送错误计数器清 0
MOV      A, # 0
MOVX    @DPTR,A
MOV      DPTR, # 0           ;指向模式寄存器
MOV      A, # 00H            ;进入正常操作模式,双验收滤波器模式
MOVX    @DPTR,A
END

```

其中, # AMR0~# AMR3 是给验收屏蔽寄存器所赋的值, # ACR0~# ACR3 是给验收代码寄存器所赋的值。可以读取上述寄存器的值,进行观察,看设置是否如预期。

3.4.3 SJA1000 自检测

为了方便通信调试,SJA1000 具有两种自检测方式:全局自检测和局部自检测。利用这两种自检测方式,可以测试物理层接口是否存在问题。全局自检测必须在一个运行的系统中

进行,因为被检测的 CAN 节点需要其他节点的接收应答。被检测的节点所发送的信息同时也被自身接收,所以全局自检测常用于测试系统中某个节点是否正常。局部自检测可用于单节点的测试,不需要其他节点的接收应答。局部自检测常用于对单个节点发送接收功能的测试。

执行全局自检测或局部自检测时,必须先进入复位模式,处理过程和正常的发送接收相似。

```

ORG      0H
MOV      DPTR, # 0           ;指向模式寄存器
MOV      A, # 01H           ;进入复位模式
MOVX     @DPTR, A
MOV      DPTR, # 31         ;指向时钟分频寄存器
MOV      A, # 88H           ;选择 PeliCAN 模式,关闭时钟输出
MOVX     @DPTR, A
MOV      DPTR, # 4          ;指向中断允许寄存器
MOV      A, # 03H           ;开放接收、发送中断
MOVX     @DPTR, A
MOV      DPTR, # 20         ;指向验收屏蔽寄存器
MOV      R3, # 4
MOV      A, # 0FFH          ;验收屏蔽寄存器赋值
AMRINI:  MOVX     @DPTR, A    ;接收所有信息
INC      DPTR
DJNZ     R3, AMRINI
MOV      DPTR, # 0          ;指向模式寄存器
MOV      A, # 04H           ;进入局部自检测模式,返回操作模式
MOVX     @DPTR, A
MOV      DPTR, # 16         ;发送缓冲区首地址
MOV      A, # 01H           ;标准帧,1 字节数据
MOVX     @DPTR, A
MOV      A, # 58H           ;任意配置一个标识符
INC      DPTR
MOVX     @DPTR, A
MOV      A, # 58H
INC      DPTR
MOVX     @DPTR, A
INC      DPTR              ;发送数据 05AH
MOV      A, # 05AH
MOVX     @DPTR, A
MOV      DPTR, # 1          ;指向命令寄存器

```

CAN 总线技术

```

MOV    A, #10H                ;发送自检测命令
MOVX   @DPTR, A
END

```

执行上述程序后,一个局部自检测发送完成。因为开放了发送和接收中断,测试数据发送完成后,会立即产生发送中断和接收中断。此时,可以读取接收缓冲区,观察是否接收到了发送的数据。

3.4.4 SJA1000 收发程序设计

1. 发送过程

发送过程是用户将要发送的数据按照 CAN 协议规定的帧格式组成数据帧,存入 SJA1000 的发送缓冲区,然后写发送命令。在把数据存入发送缓冲区前,要判断 SJA1000 当前的状态是否允许信息发送。发送一个数据帧的过程如图 3-12 所示。

发送前,一般检查 3 个状态位。第一个是接收状态。如果目前 SJA1000 正在接收信息,则不能发送,至少等本次接收完成后才能申请发送。第二个是发送完成状态,即检查 SJA1000 是否正在发送信息。如果正在发送,则要等一次发送完成才能启动新的发送任务。第三个是检查发送缓冲区是否被锁定。发送缓冲区处于不锁定状态时,才能发送信息。

因为远程帧没有数据字节,发送远程帧比发送数据帧还要简单一些,这里不再赘述。

2. 接收过程

接收过程的处理比发送过程要复杂些。在接收数据时,不仅要接收数据作出处理,还要对各种错误、数据溢出等进行判断和处理。接收报文的响应方式有查询接收方式和中断接收方式,中断方式适合实时性要求高的通信系统,其他可用查询接收方式。

图 3-13 是查询接收方式处理流程,中断接收方式处理过程与此类似。

该流程中,先读取状态寄存器,判断是否存在错误或其他异常情况。如果有异常情况,先读取中断寄存器把错误标志清除,然后进行相应的处理。一般,异常情况有下面几种:

➤ 总线关闭。此时该节点被隔离在系统之外,可将模式寄存器的复位请求位置 0,然后向

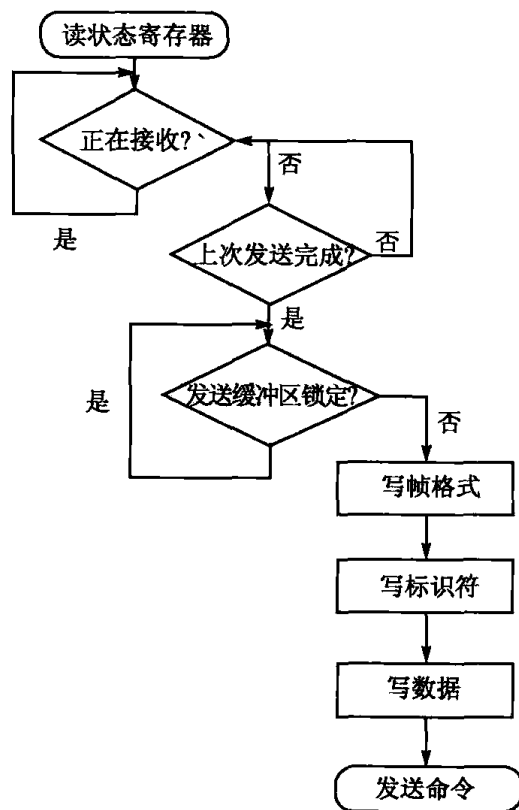


图 3-12 发送数据帧流程

用户报警。

- 数据溢出。当接收缓冲区已满,又接收到下一个报文时,会产生接收数据溢出,此时应该尽快读取数据,并释放接收缓冲区。用户应该及时把接收缓冲区中的数据读走,存放另外的数据存储空间,而不应该让 SJA1000 的接收缓冲区被逐渐堆积充满。
- 数据传输和接收期间,也可能发生错误;SJA1000 会将这些错误以错误帧的形式自动标注出来。
- 接收到数据,也可以算是一种“异常”。这时,接收缓冲区状态为“满”,说明有未被读取的数据。

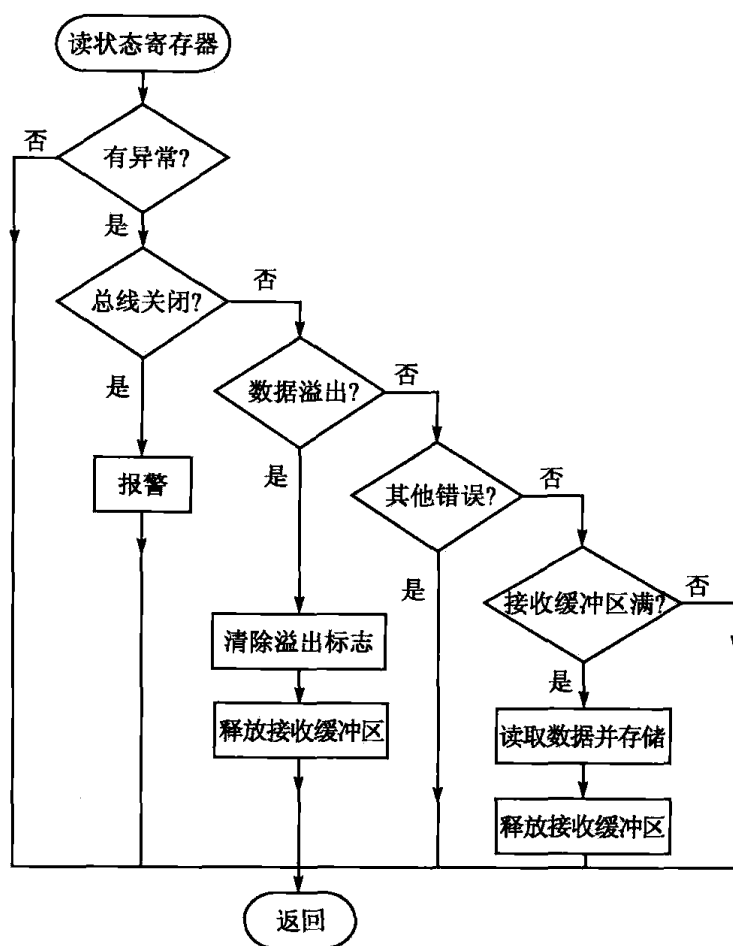


图 3-13 查询接收方式处理过程

第 4 章

常用 CAN 总线收发器

CAN 总线驱动器提供了 CAN 控制器与物理总线之间的接口,对总线提供差动发送能力,并对 CAN 控制器提供差动接收能力,是影响 CAN 网络通信的一个重要因素。

4.1 CAN 总线收发器 PCA82C250

4.1.1 概 述

PCA82C250 是 NXP 公司生产的 CAN 收发器,主要用于汽车中高速领域,是目前世界使用最广泛的 CAN 收发器,支持 ISO—11898 标准,具有以下特性:

- 符合和 ISO—11898 标准;
- 高速率(最高达 1 Mbps);
- 总线保护能力,具有抗汽车环境中的瞬间干扰;
- 斜率控制,降低射频干扰(RFI);
- 差分接收器,抗宽范围的共模干扰,抗电磁干扰(EMI);
- 热保护;
- 防止电池和地之间的发生短路;
- 低电流待机模式;
- 未上电节点对总线无影响;
- 可连接 110 个节点。

4.1.2 组成结构及功能描述

(1) 功能框图

PCA82C250 的结构方框图如图 4-1 所示。

(2) 引脚信息

PCA82C250 的外观及引脚信息如图 4-2 和表 4-1 所示。

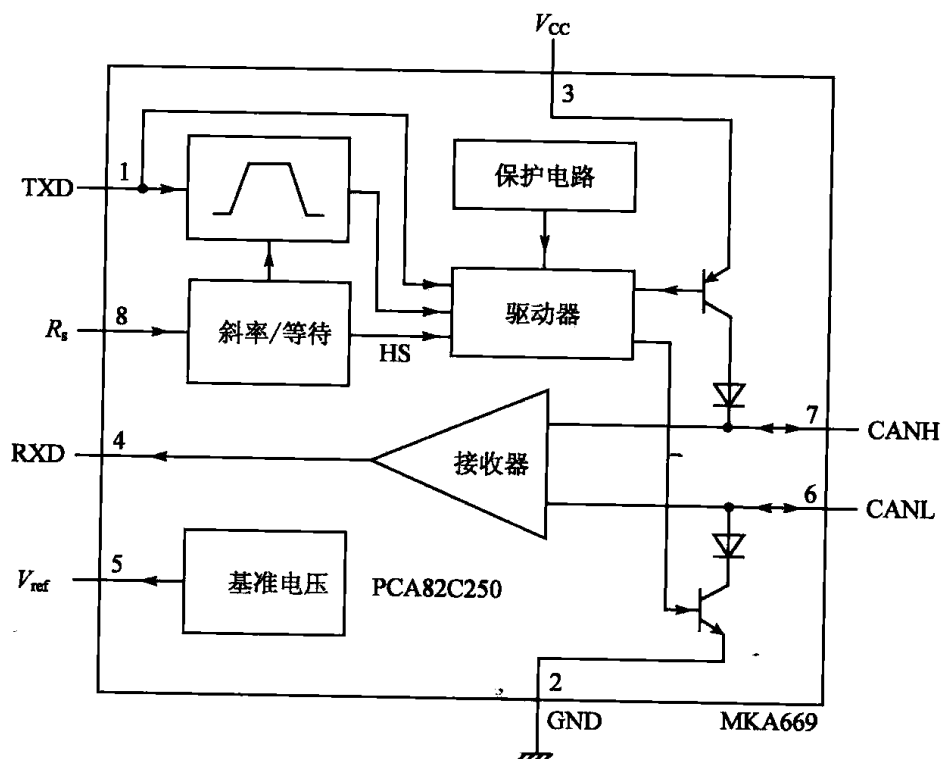


图 4-1 PCA82C250 框图

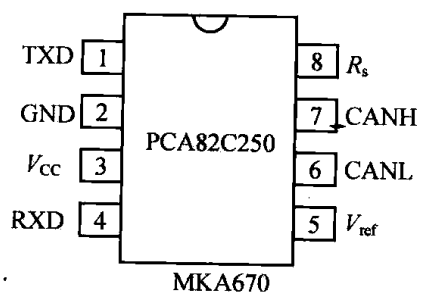


图 4-2 PCA82C250 外形图

表 4-1 引脚信息

符 号	引 脚	功能描述	符 号	引 脚	功能描述
TXD	1	发送数据输入	V_{ref}	5	参考电压输出
GND	2	地	CANL	6	低电平 CAN 电压输入或输出
V_{cc}	3	电源电压	CANH	7	高电平 CAN 电压输入或输出
RXD	4	接收数据输出	R_s	8	斜率电阻输入

CAN 总线技术

(3) 基本参数

PCA82C250 的基本参数如表 4-2 所列。

表 4-2 器件参考数据

符 号	参 数	条 件	最 小	最 大	单 位
V_{CC}	提供电压		4.5	5.5	V
I_{CC}	提供电流	待机模式	—	170	μA
$1/t_{bit}$	最大发送速度	非归零码	1	—	Mbaud
V_{CAN}	CANL、CANH 输入/输出电压		-8	+18	V
V_{diff}	差动总线电压		1.5	3.0	V
t_{pd}	传输延迟时间	高速模式	—	50	ns
T	工作环境温度		-40	+125	$^{\circ}C$

(4) 工作模式

引脚 8(R_S)的 3 种不同接法对应 3 种不同的工作模式：待机、斜率控制、高速，如表 4-3 所列。

表 4-3 R_S 选择的 3 种不同工作模式

在 R_S 引脚上强制条件	模 式	在 R_S 引脚上的电压和电流
$V_{RS} > 0.75V_{CC}$	待机	$I_{RS} < 10 \mu A $
$-10 \mu A < I_{RS} < -200 \mu A$	斜率控制	$0.3V_{CC} < V_{RS} < 0.6V_{CC}$
$V_{RS} < 0.3V_{CC}$	高速	$I_{RS} < -500 \mu A$

引脚 8 接地进入高速模式。发送器延时是影响发送速度的一个重要因素，高速工作模式下，发送器输出级晶体管最大限度地快速打开和关闭。在这种模式下，没有任何措施用于限制上升斜率和下降斜率。此时，最好使用屏蔽电缆，以避免射频干扰 RFI。

引脚 8 和地之间串入电阻进入斜率模式。对于较低速度或较短总线长度的应用场合，可使用非屏蔽双绞线或平行线作为总线。此时，为降低射频干扰 RFI，应限制发送器晶体管的上升斜率和下降斜率。上升斜率和下降斜率可由引脚 8 接至地的连接电阻进行控制，斜率正比于引脚 8 的电流输出。

引脚 8 接至高电平进入低电流待机模式。在此模式下，关闭发送器，接收器转至低电流。若在总线上检测到显性位(差动总线电压 $> 0.9 V$)，则 RXD 变为低电平。微控制器应将收发器转回至正常工作状态(通过引脚 8)，以对此信号作出响应。由于处在待机方式下，接收器是慢速的，因此，第一个报文将丢失。CAN 收发器的真值表如表 4-4 所列。

表 4-4 CAN 收发器真值表

电 源/V	TXD	CANH	CANL	总线状态	RXD
4.5~5.5	0	高	低	显性	0
4.5~5.5	1(或悬空)	悬空	悬空	隐性	1
<2(未上电)	X	悬空	悬空	隐性	X
$2 < V_{CC} < 4.5$	$> 0.75V_{CC}$	悬空	悬空	隐性	X
$2 < V_{CC} < 4.5$	X	若 $V_{RS} > 0.75V_{CC}$ 则悬空	若 $V_{RS} > 0.75V_{CC}$ 则悬空	隐性	X

注意：X=随意值。

(5) PCA82C250 与 PCA82C251

PCA82C251 同样是 NXP 公司生产的收发器,提供了控制器和物理传输线路之间的接口;使用 ISO—11898 标准,通信速率可以用高达 1 Mbps。它与 PCA82C250 有所不同,总体特性如表 4-5 所列。可见,PCA82C251 可以在额定电源是 12 V 和 24 V 的 CAN 总线系统中使用,主要用在汽车和普通工业应用上。从功能上讲,PCA82C251 和 PCA82C250 功能相同,可以互换,并可以在同一网络中互相通信。

表 4-5 PCA82C250 与 PCA82C251 比较

特 征	PCA82C250	PCA82C251
系统额定电源	12 V	12 V/24 V
最大的总线终端 DC 电压($0 \text{ V} < V_{CC} < 5.5V_{CC}$)	$-8 \text{ V} < V_{CANLH} < +18 \text{ V}$	$-40 \text{ V} < V_{CANLH} < +40 \text{ V}$
最大的瞬间总线终端电压(ISO7637)	$-150 \text{ V} < V_{tr} < +100 \text{ V}$	$-200 \text{ V} < V_{tr} < +200 \text{ V}$
扩展扇出应用时最小收发器电源电压($R_L = 45 \Omega$)	$V_{CC} > 4.9 \text{ V}$	$V_{CC} > 4.5 \text{ V}$

PCA82C251 有更高的击穿电压,并在这个电源电压范围内驱动低至 45Ω 的总线负载,所以在普通的工业应用中比较多。PCA82C251 在隐性状态下的拉电流更小,在掉电情况下的总线输出特性有一定改善。

4.1.3 应用举例

1. CAN 的收发器应用

CAN 的收发器典型应用电路如图 4-3 所示。CAN 控制器通过串行数据输出线(TX)和串行数据输入线(RX)连接到收发器上,收发器通过有差动发送和接收功能的两个总线终端 CANH 和 CANL 连接到总线电缆。输入 R_S 与 CPU 的 I/O 口连接,进行模式控制。参考电压输出 V_{REF} 的输出电压是额定 V_{CC} 的 0.5 倍。其中,收发器的额定电源电压是 5 V。

CAN 总线技术

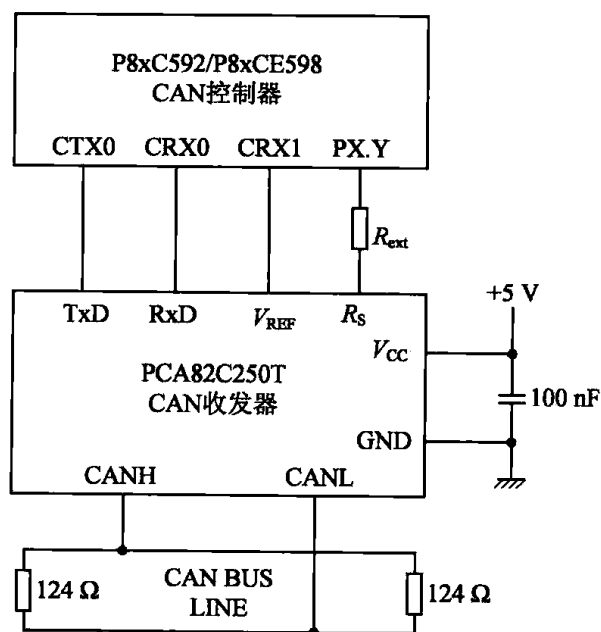
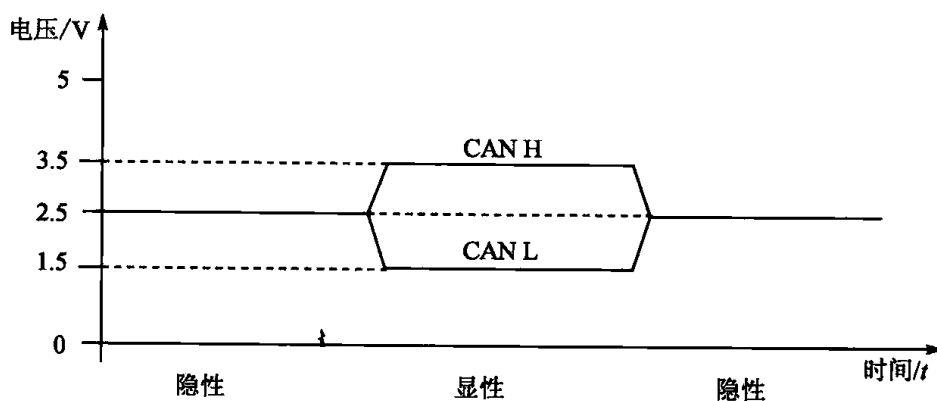


图 4-3 CAN 收发器的应用

CAN 控制器输出串行数据流至收发器的 TxD 引脚，收发器内部上拉功能将 TxD 输入拉至逻辑高电平，即总线输出驱动器默认是隐性的。ISO—11898 的额定总线电平如图 4-4 所示。在隐性状态中，CANH 和 CANL 差分输入通过典型内部阻抗是 17 kΩ 的接收器输入网络，偏置到 2.5 V 的额定电压。



注：有些资料中，显性用 Dominant 代表；隐性用 Recessive 代表。

图 4-4 根据 ISO—11898 的额定总线电平

若 TxD 是逻辑低电平，则总线的输出级激活，在总线电缆上产生一个显性的信号电平。输出驱动器由一个源输出级和一个下拉输出级组成。CANH 连接到源输出级，CANL 连接到下拉输出级。在显性状态中，CANH 的额定电压是 3.5 V，CANL 的额定电压是 1.5 V。

如果没有总线节点传输显性位，则总线处于隐性状态，即网络中所有 TxD 输入逻辑高电

平。另外,如果一个或更多的总线节点传输一个显性位,即至少一个 TxD 输入逻辑低电平,则总线从隐性状态进入显性状态,此即“线与”功能。

收发器中接收器的比较器将差分总线信号转换成逻辑信号电平,并在 RxD 输出。接收到的串行数据流传送到总线控制器进行译码。接收器的比较器总是工作的,即当总线节点传输一个报文时,它同时也监控总线,这就要求有诸如安全性和支持非破坏性逐位竞争等 CAN 策略。一些控制器提供一个模拟接收接口(RX0、RX1)。RX0 一般需要连接到 RxD 输出,RX1 需要偏置到一个相应的电压电平,这可以通过 V_{REF} 输出。

2. 有电气隔离的 CAN 收发电路

实际应用中,如果需要电气隔离,则可以在收发器和 CAN 控制器之间放置光耦,如图 4-5 所示。使用光耦时,要注意隔开的 CAN 控制器电路一边没有上电时的默认状态,在这种情况下,连接到 TxD 的光耦应该没有电流流过,是暗的,即 LED 关断。当光耦是断开/暗时,收发器的 TxD 输入逻辑高电平,可以达到隔离光耦电路的原边次边、自动防故障的目的。使用光耦还要考虑到将 Rs 模式控制输入连接到高电平有效的复位信号,如当本地收发器电源电压(在斜率上升和下降过程中)没有准备好的情况下,可以使能收发器。还有,当 CAN 控制器外接收发器时,RX1 应固定在一个固定电平;在图 4-5 中,RX1 偏置电压电平通过一个电阻电压分配器实现。

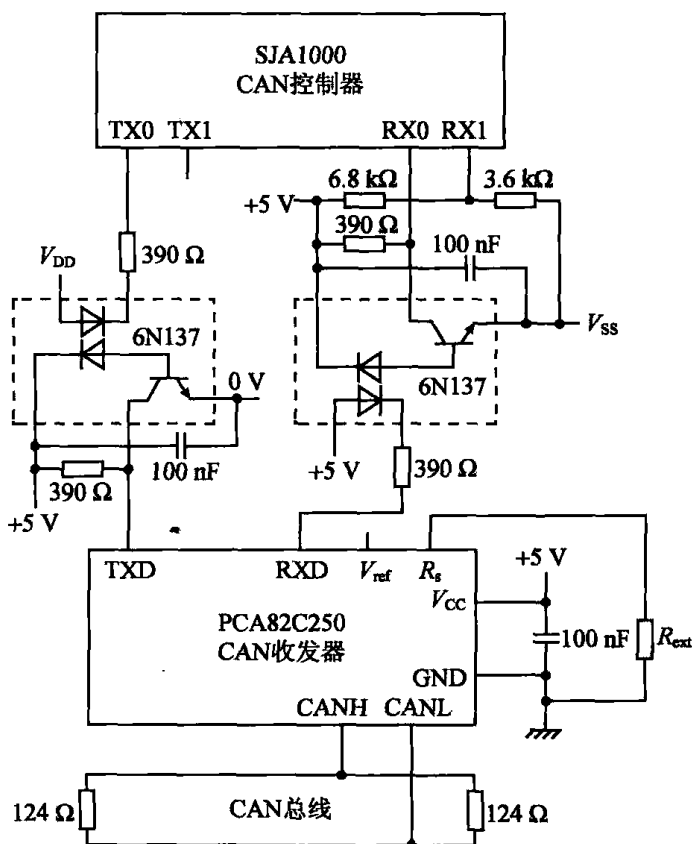


图 4-5 接口使用光耦进行电流隔离

CAN 总线技术

但是,在 CAN 控制器和收发器之间使用光耦,通常会增加总线节点的循环延迟。信号要从每个节点发送和接收电路上通过两次,这会降低位速率或者说减少位速率一定时可使用的最大总线长度,这在计算由于 CAN 网络中的传输延迟而造成限制可以使用的最大总线长度时要考虑。如果位速率很高,如高于 500 kbps,则应考虑使用延迟小于 40 ns 的高速光耦,比如 HCPL-7101、TLP113 等。

4.2 高速 CAN 收发器 TJA1050

4.2.1 概 述

TJA1050 是 CAN 协议控制器和物理总线之间的接口,是 PCA82C250 高速 CAN 收发器的后继产品,可以为总线提供不同的发送性能,为 CAN 控制器提供不同的接收性能。

TJA1050 的主要特性有:

- CANH 和 CANL 理想配合,降低电磁辐射;
- 网络中有不上电节点时,性能有所改进;
- 完全符合 ISO—11898 标准;
- 速率高达 1 Mbps;
- 电磁抗干扰 EMI 性极高;
- 不上电的节点不会对总线造成扰动;
- TxD 引脚有防止钳位在显性总线电平的超时检测功能;
- 静音模式中提供了只听模式和 Babbling Idiot 保护;
- 保护总线引脚,防止汽车环境中的瞬态干扰;
- 输入级兼容 3.3 V 和 5 V 器件;
- 输出驱动器具有温度保护;
- 防止电池对地的短路;
- 至少可以连接 110 个节点。

4.2.2 组成结构及功能描述

1. 功能框图

TJA1050 的组成方框图如图 4-6 所示。

2. 引脚信息

TJA1050 的引脚信息及外观如表 4-6 所列和引脚配置如图 4-7 所示。

3. 基本参数

TJA1050 的基本参数如表 4-7 所列。

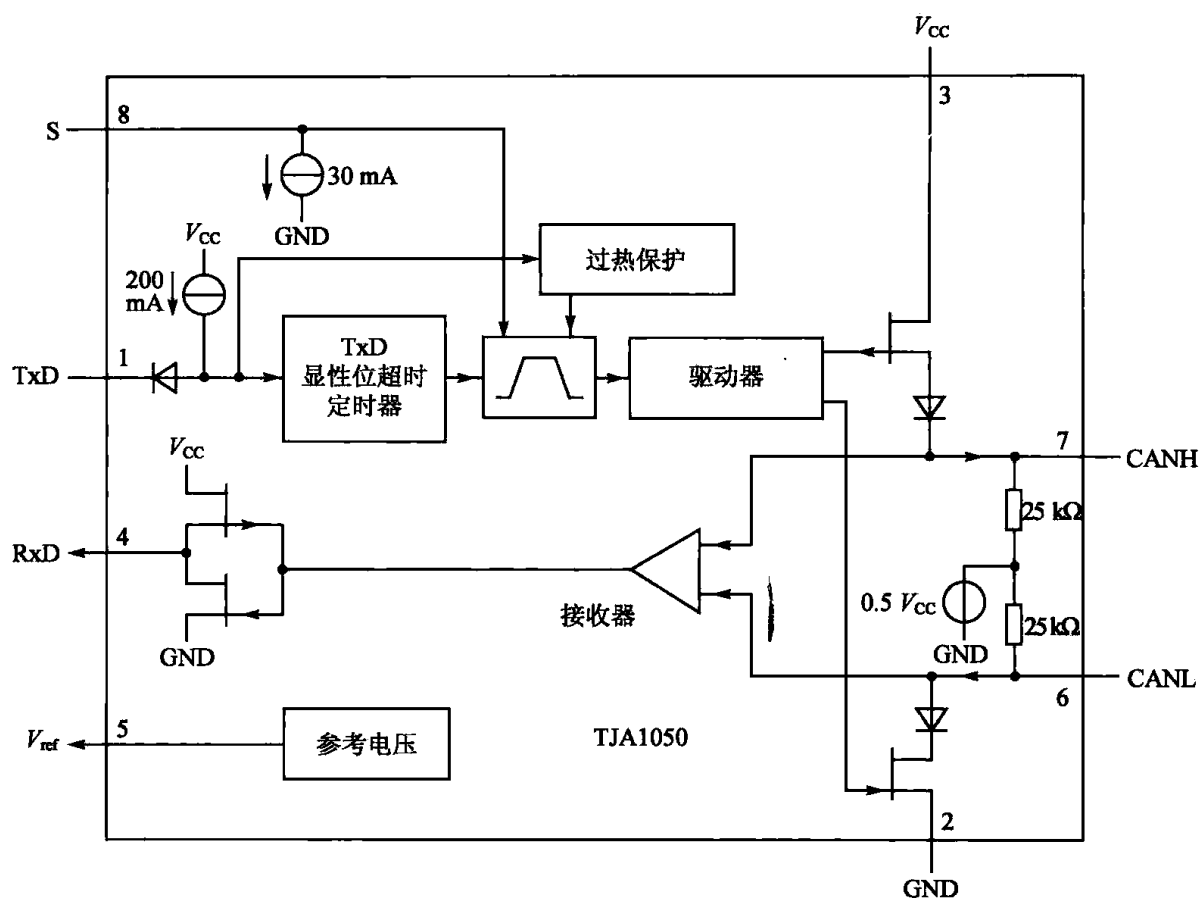


图 4-6 TJA1050 的功能方框图

表 4-6 TJA1050 引脚信息

符 号	引 脚	描 述
TxD	1	发送数据输入
GND	2	接地
V _{cc}	3	电源
RxD	4	接收数据输入
V _{ref}	5	参考电压输出
CANL	6	低电平 CAN 总线
CANH	7	高电平 CAN 总线
S	8	选择进入高速模式还是静音模式

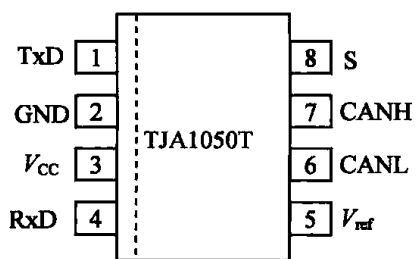


图 4-7 引脚配置图

CAN 总线技术

表 4-7 器件参考数据

符 号	参 数	条 件	最小值	最大值	单 位
V_{CC}	电源		4.75	5.25	V
V_{CANH}	引脚 CANH 的直流电压	$0 < V_{CC} < 5.25 \text{ V}$; 无时间限制	-27	+40	V
V_{CANL}	引脚 CANL 的直流电压	$0 < V_{CC} < 5.25 \text{ V}$; 无时间限制	-27	+40	V
$V_i(\text{dif})(\text{bus})$	不同的总线输入电压	控制	1.5	3	V
$T_{pd}(\text{TXD}-\text{RXD})$	TXD 到 RXD 的传播延迟时间	$V_S=0 \text{ V}$	—	250	ns
T	环境温度		-40	+125	°C

4. 功能描述

TJA1050 最初应用在波特率范围为 60k~1M bps 的高速自动化设备中。TJA1050 可以为总线提供不同的发送性能,为 CAN 控制器提供不同的接收性能,而且与 ISO—11898 标准完全兼容,收发器功能如表 4-8 所列。

表 4-8 CAN 收发器功能表

V_{CC}/V	TxD	S	CANH	CANL	总线状态	RxD
4.75~5.25	0	0(或悬空)	高	低	控制	0
4.75~5.25	X	1	$0.5 V_{CC}$	$0.5 V_{CC}$	隐性	1
4.75~5.25	1(或悬空)	X	$0.5 V_{CC}$	$0.5 V_{CC}$	隐性	1
<2(不加电)	X	X	$0 \text{ V} < V_{CANH} < V_{CC}$	$0 \text{ V} < V_{CANH} < V_{CC}$	隐性	X
$2 < V_{CC} < 4.75$	>2 V	X	$0 \text{ V} < V_{CANH} < V_{CC}$	$0 \text{ V} < V_{CANH} < V_{CC}$	隐性	X

注: X=不考虑。

TJA1050 设计有一个电流限制电路以保护发送器的输出级,当电源意外短路时,不会对 TJA1050 造成损坏(此时的功率消耗增加)。

TJA1050 还有一个温度保护电路,当与发送器的连接点的温度超过约 165°C 时,会断开与发送器的连接。芯片工作时,发送器消耗了大部分的功率,断开发送器使电路的功率消耗和温度变低,此时芯片的其他功能仍继续工作。当引脚 TxD 变高(电平)时,发送器由关闭状态复位。当总线短路时,温度保护电路尤其重要。

在汽车通电的瞬间,自动暂态测试电路如图 4-8 所示,引脚 CANH 和 CANL 也受到保护(根据 ISO—7637)。

通过引脚 S 可以选择两种工作模式:高速模式或静音模式。高速模式就是普通的工作模式,将引脚 S 接地可以进入这种模式。如果引脚 S 没有连接,则默认为高速模式。

将 S 引脚连接到 V_{CC} ,则可以进入静音模式。在静音模式中,发送器是禁止的,但芯片的

第4章 常用 CAN 总线收发器

其他功能可以继续使用。静音模式可以防止在 CAN 控制器不受控制时对网络通信造成堵塞。

当引脚 TXD 由于硬件和/或软件程序的错误而持久地为低(电平)时,则会阻塞所有网络通信。TXD 控制超时定时器电路可以防止总线进入这种持久的显性状态,是由引脚 TXD 的负跳变边沿触发。如果引脚 TXD 低电平的持续时间超过内部定时器的值,则发送器禁止,从而使总线进入隐性状态。定时器由引脚 TXD 的正跳变边沿复位。

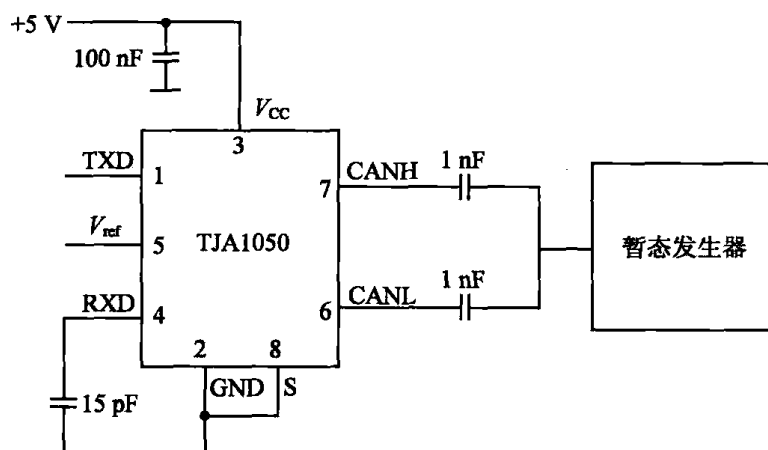


图 4-8 自动暂态测试过程

TJA1050 的各引脚必须满足一定的限值,如表 4-9 所列。

表 4-9 CAN 收发器功能表

助记符	参 数	条 件	最小值	最大值	单 位
V_{CC}	电源电压		-0.3	+6	V
V_{CANH}	引脚 CANH 的 DC 电压	$0 < V_{CC} < 5.25 \text{ V}$; 无时间限制	-27	+40	V
V_{CANL}	引脚 CANL 的 DC 电压	$0 < V_{CC} < 5.25 \text{ V}$; 无时间限制	-27	+40	V
V_{TXD}	引脚 TXD 的 DC 电压		-0.3	$V_{CC} + 0.3$	V
V_{RXD}	引脚 RXD 的 DC 电压		-0.3	$V_{CC} + 0.3$	V
V_{ref}	引脚 V_{ref} 的 DC 电压		-0.3	$V_{CC} + 0.3$	V
V_S	引脚 S 的 DC 电压		-0.3	$V_{CC} + 0.3$	V
$V_{tri}(\text{CANH})$	引脚 CANH 的瞬态电压	注	-200	+200	V
$V_{tri}(\text{CANL})$	引脚 CANL 的瞬态电压	注	-200	+200	V

注: 所有参数在连接温度范围内得到保证, 只有晶片级的电路板在 125℃ 的环境温度下测试过所有参数, 封装的产品在 25℃ 的环境温度下测试过。

CAN 总线技术

5. 工作模式

TJA1050 有两种工作模式,即高速模式和静音模式,它们都由引脚 S 来控制。

TJA1050 不支持 PCA82C250 具备的可变斜率控制,所以,TJA1050 有固定的斜率;尽管如此,其输出级良好的对称性使它的 EMC 性能比前面的产品更好。

(1) 高速模式

将引脚 S 连接到地可以进入高速模式,该模式是普通的工作模式。由于引脚 S 有内部下拉功能,所以当它没有连接时,高速模式也是默认的工作模式。

在这个模式中,总线输出信号有固定的斜率,并且以尽量快的速度切换。这种模式适合用于最大的位速率和/或最大的总线长度,而且此时它的收发器循环延迟最小。

(2) 静音模式

将引脚 S 接高电平,就可以进入静音模式。在静音模式中,发送器是禁止的,所以它不管 TxD 的输入信号。因此,收发器运行在非发送状态中时,它消耗的电流和在隐性状态时的一样。

静音模式中,节点可以设置成对总线绝对无源的状态,防止 CAN 控制器不受控制,占用总线无意识地发送报文(babbling idiot)。微控制器激活了静音模式后,微控制器不再直接访问 CAN 控制器,TJA1050 将释放总线;静音模式使系统有更高的可靠性。

在静音模式中,RxD 正常监控总线,因此,静音模式提供了具有诊断功能的只听模式,确保节点的显性位完全不会影响总线。

6. 应用举例

CAN 高速收发器的一般应用如图 4-9 所示。

CAN 协议控制器通过一条串行数据输出线(TxD)和一条串行数据输入线(RxD)连接到收发器,而收发器则通过它的两个有差动接收和发送能力的总线终端 CANH 和 CANL 连接到总线线路。引脚 S8 用于模式控制。参考输出电压 V_{ref} 提供一个 $V_{CC}/2$ 的额定输出电压,这个电压是作为带有模拟 Rx 输入的 CAN 控制器的参考电平。由于 SJA1000 具有数字输入,因此它不需要这个电压。收发器使用 5 V 的工作电压。

协议控制器向收发器的 TxD 引脚输出一个串行的数据流。收发器的内部上拉功能将 TxD 引脚置为逻辑高电平,即总线输出驱动器在开路时是无源的。如图 4-10 所示,在隐性状态中,CANH 和 CANL 输入通过典型内部阻抗为 25 k Ω 的接收器接入网络,偏置到 $V_{CC}/2$ 的电平电压。如果 TxD 是逻辑低电平,则激活总线的输出级,并在总线上产生一个显性信号电平。输出驱动 CANH 由 V_{CC} 提供一个源输出,而 CANL 则向 GND 提供一个下拉输出。

如果所有总线节点都不发送显性位,则总线处于隐性状态。如果一个或多个总线节点发送一个显性位,则总线进入显性状态。

接收器比较器将差动的总线信号转换成逻辑电平信号,并在 RxD 输出。总线协议控制器将接收到的串行数据流译码。接收器比较器总是激活的,即当总线节点发送一个报文时,它同

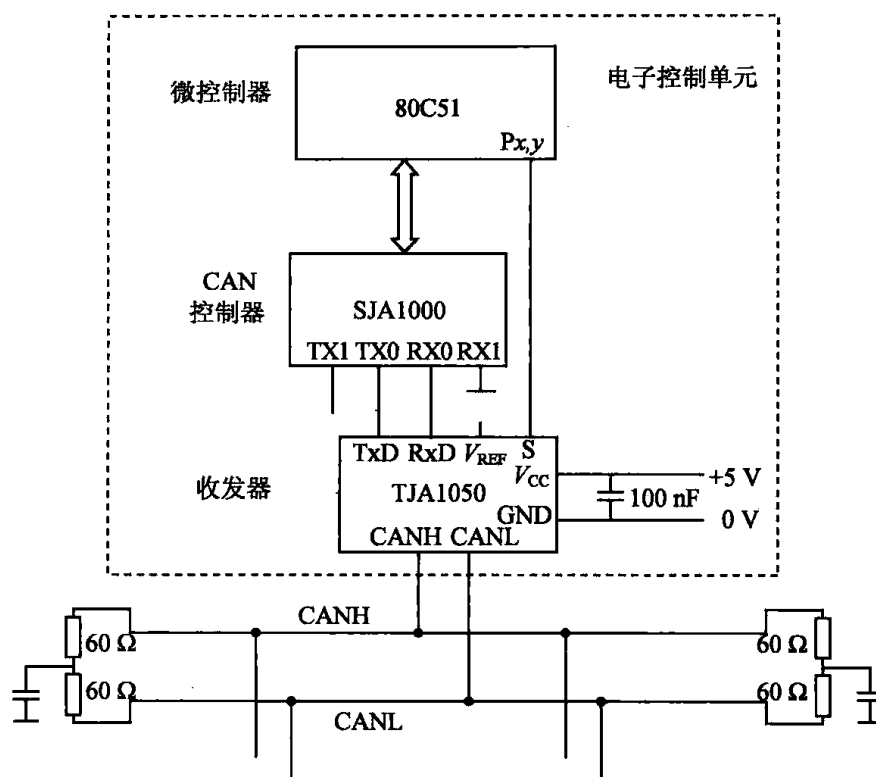


图 4-9 CAN 高速收发器的典型应用

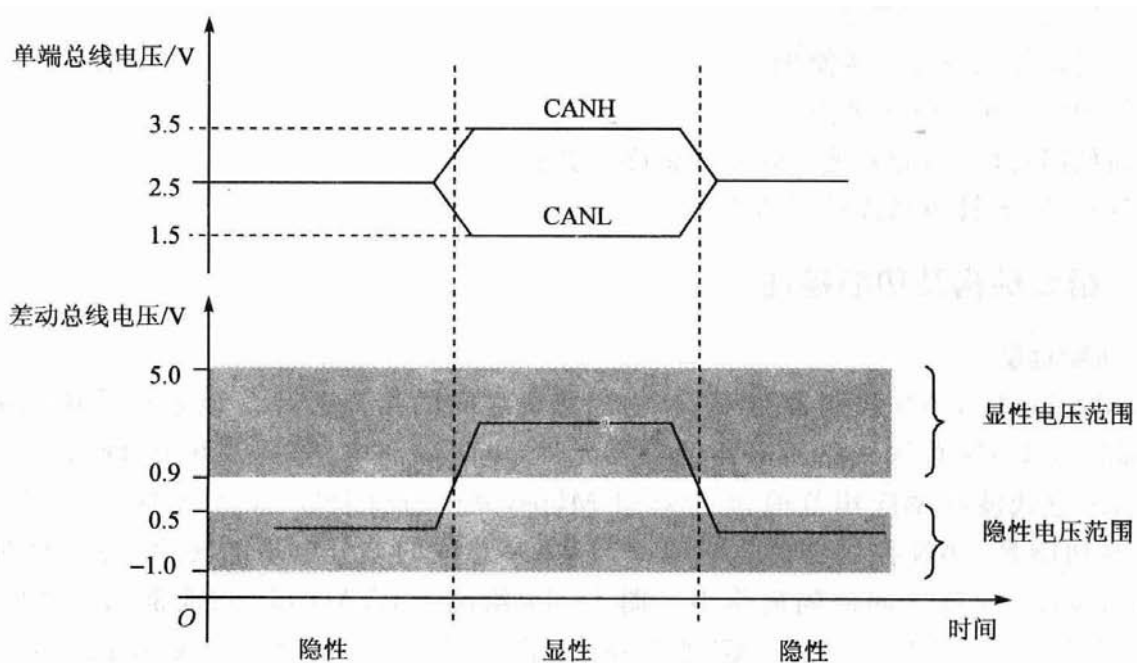


图 4-10 根据 ISO—11898 的额定总线电平

CAN 总线技术

时监控总线。这个功能可以用于支持 CAN 的非破坏性逐位仲裁方法。

典型的总线采用一对双绞线。按照 ISO—11898 中定义的线性拓扑结构,总线两端都接一个 $120\ \Omega$ 的额定电阻,这就要求总线额定负载是 $60\ \Omega$ 。终端电阻和电缆阻抗的紧密匹配确保了数据信号不会在总线的两端反射。

4.3 隔离 CAN 收发器 CTM1050

4.3.1 芯片概述

CTM1050 是一款新型的、带隔离的高速 CAN 收发器芯片,片内部集成了必需的 CAN 隔离及 CAN 收发器件。该芯片的主要功能仍是 CAN 控制器的逻辑电平与 CAN 总线的差分电平的转换,但同时具有 DC2500V 的隔离功能及 ESD 保护作用。

主要特性如下:

- 具有隔离、ESD 保护功能;
- 完全符合 ISO—11898 标准的 CAN 收发器;
- 通信速率最高达 1 Mbps;
- 隔离电压: DC 2500 V;
- 电磁辐射 EME 极低;
- 电磁抗干扰 EMI 性极高;
- 无需外加元件可直接使用;
- 至少可连接 110 个节点;
- 高低温特性好,能满足工业级产品技术要求;
- 具有 TVS 管防总线过压功能。

4.3.2 组成结构及功能描述

1. 功能框图

CTM1050 是 CAN 控制器与 CAN 物理总线之间的接口芯片。该芯片采用全灌封工艺,内部集成 CAN 总线线路所必需的收发元件,电气隔离电路(隔离电压 DC 2500 V);支持 CAN 总线波特率应用范围为 $40\text{k}\sim 1\text{ Mbps}$,完全符合 ISO—11898 标准。CTM1050 芯片主要功能是 CAN 控制器的逻辑电平与 CAN 总线的差分电平的转换,还具有对 CAN 控制器与 CAN 总线之间的隔离作用。图 4-11 给出了 CTM1050 功能框图。其中,此系列的 CTM1050T 在 CTM1050 基础上增加了保护 CAN-bus 总线的过压功能,如图 4-12 所示。

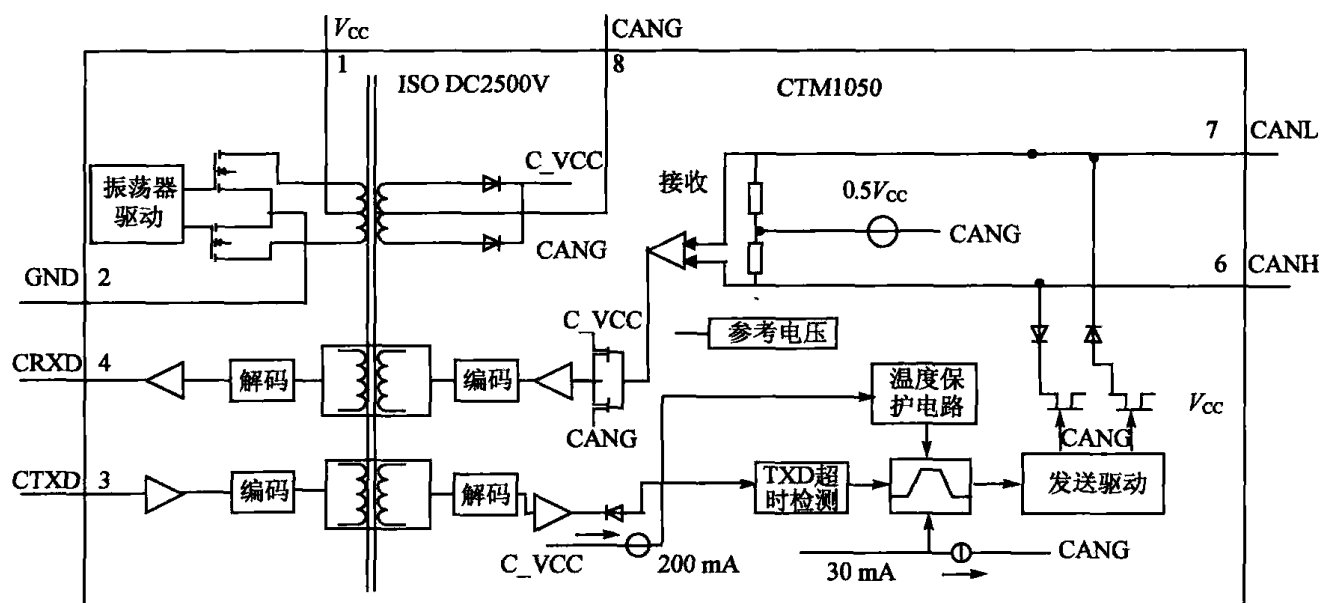


图 4-11 CTM1050 功能框图

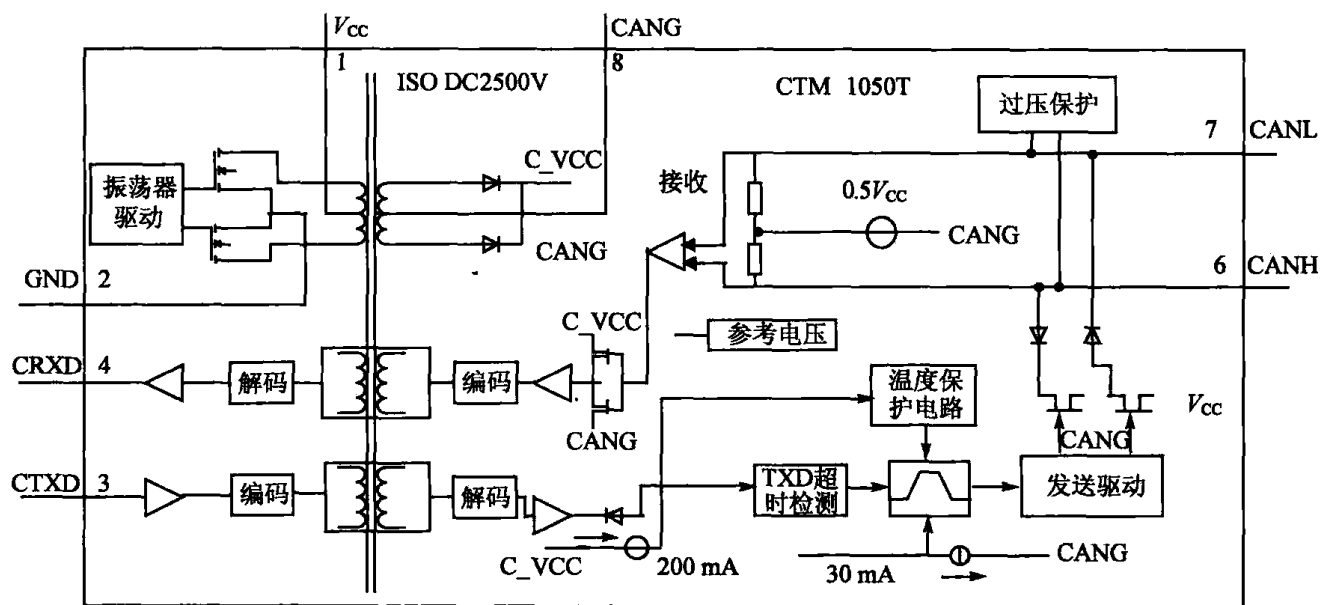


图 4-12 CTM1050T 功能框图

CAN 总线技术

2. 引脚信息及基本参数

引脚信息如表 4-10 所列。基本参数如表 4-11 所列。

表 4-10 引脚分配

引脚号	引脚名称	引脚含义	引脚号	引脚名称	引脚含义
1	V_{in}	+5 V 输入	6	CANH	CANH 信号线连接端
2	GND	电源地	7	CANL	CANL 信号线连接端
3	TXD	CAN 控制器发送端	8	CANG	隔离电源输出地
4	RXD	CAN 控制器接收端			

表 4-11 基本参数表

参 数	描 述
电 源	4.75~5.25 V 的 DC, 静态电流 35 mA, 最大电流 <60 mA
CAN 总线接口	符合 CAN 控制器接口, 支持各种 CAN 控制器
串行接口(3、4)引脚电流	小于 2 mA
输入数据比特率	40 kbps~1 Mbps
没上电的无源特性 ($V_{CC}=0$ 时的总线引脚漏电流)	<250 μ A ($V_{CANL}=5$ V)
总线引脚(6、7)的最大 DC 电压	-27~+40 V
湿度	5.95% 不结露
隔离电压	DC 2500 V
温度范围	-40~+85℃

4.3.3 典型应用

在实际应用中,CTM1050 构成的 CAN 通信线路外围器件少,简单可靠,如图 4-13 所示。

图 4-13 为 CTM1050 的典型应用实例。其中,CTM 系列模块是集成电源隔离、电气隔离、CAN 收发器、CAN 总线保护于一体的隔离 CAN 收发器模块,可以实现带隔离的 CAN 收发电路;隔离电压可以达到 DC 2500 V,并且能连接任何一款 CAN 协议控制器。该芯片 TxD、RxD 引脚兼容 +3.3 V 及 +5 V 的 CAN 控制器,不需要外接其他元器件直接将 +3.3 V 或 +5 V 的 CAN 控制器发送、接收引脚与 CTM 模块的发送、接收引脚相连接。可以看出,CTM1050 接口简单,使用方便,非常适合嵌入式系统的设计。

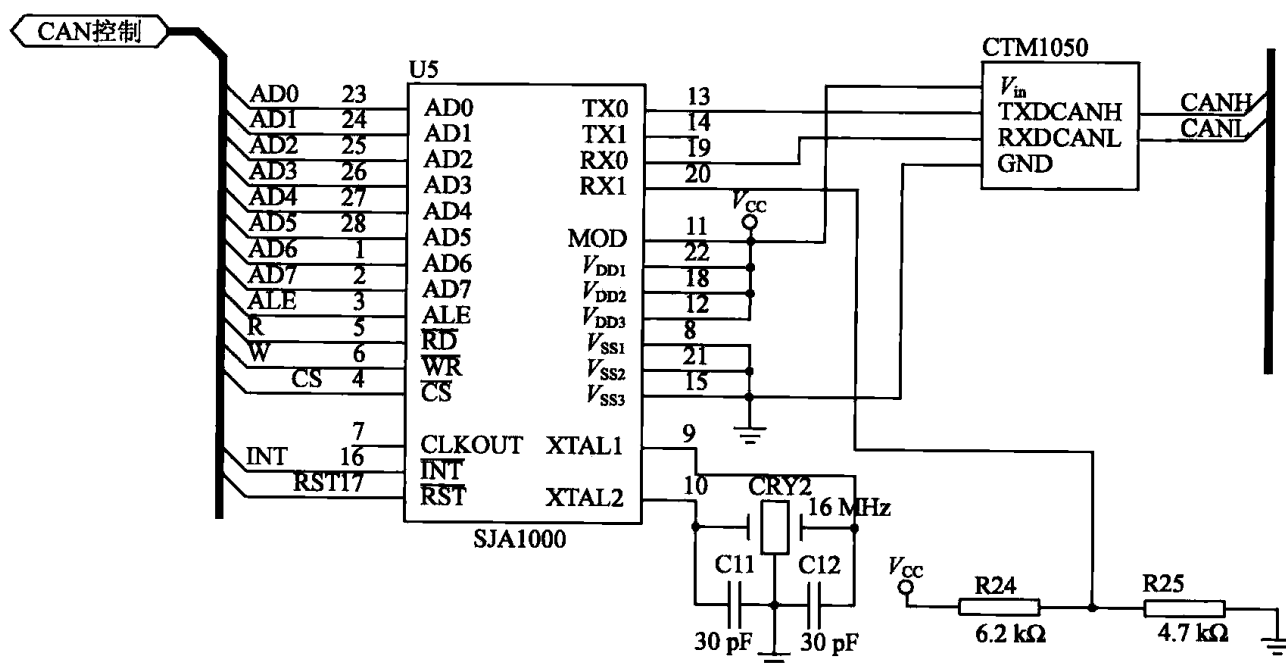


图 4-13 典型应用接口电路

第 5 章

具有 CAN 接口的处理器

本章介绍几种带有 CAN 接口的处理器,这种方案使 CAN 应用的电路设计更简单、更可靠。MCS-51 单片机拥有众多的使用者,本章主要介绍具有 CAN 接口的 51 系列单片机 C8051F040 和 P87C591,且对数字信号处理器 TMS320F2812 的 CAN 模块进行了简要说明。嵌入式系统设计近年发展也很迅速,本章介绍 ARM7、ARM9 系列微控制器,都是带有 CAN 接口的嵌入式控制器,使用灵活、方便。

5.1 C8051F040

C8051F 系列单片机是 MCS-51 单片机的一个改进系列,属于片上系统型(SOC)单片机。C8051F 系列单片机片内模拟与数字电路工作电压是 3 V(2.7~3.6 V);可编程的时钟系统可以保证系统在满足响应速度要求下,使系统的平均时钟频率最低;众多的复位源使系统在掉电方式下,可随意唤醒。这些特点使 C8051F 系列单片机具有极佳的低功耗特性。

C8051F 系列单片机仍采用原 80C51 指令系统,兼容性极好;它对 MCS-51CPU 内核进行了改进,称为 CIP-51,功能更强、速度更快。C8051F 指令系统的运行采用流水线结构,废除了机器周期的概念;指令以时钟周期为运行单位,大大提高了指令运行速度。与 8051 相比,在相同时钟下单周期指令运行速度为原来的 12 倍;整个指令集平均运行速度为原来的 9.5 倍。

C8051F 中采用开关网络以硬件方式实现 I/O 端口的灵活配置。在这种通过交叉开关配置的 I/O 端口系统中,单片机外部为通用 I/O 口,如 P0 口、P1 口和 P2 口;内有输入/输出的电路单元通过相应配置寄存器控制的交叉开关配置到所选择的端口上。C8051F 的所有 I/O 端口都可以接收 5 V 逻辑电平的输入,在选择开漏加上拉电阻到 5 V 后,也可驱动 5 V 的逻辑器件,可与 5 V 电路方便地连接。

I/O 端口的功能配置通过交叉开关配置寄存器 XBRx、输出方式寄存器 PnMDOUT、输入方式寄存器 PnMDIN 等来设置。

C8051F04x 系列单片机是完全集成的混合信号片上系统型 MCU, 具有 64(C8051F040/2/4/6)或 32 个(C8051F041/3/5/7)数字 I/O 引脚; 片内集成了一个 CAN2.0B 控制器; 片内具有 64 KB(C8051F040/1/2/3/4/5)或 32 KB(C8051F04/6/7)的、可在系统编程的 Flash, 4 352 字节的片内 RAM。其他外设主要有: 12/10 位或 8 位 ADC; 2 个 12 位 DAC; 3 个模拟比较器; 硬件实现的 SPI、SMBus/I²C 和 UART 串行接口; 5 个通用的 16 位定时器; 具有 6 个捕捉/比较模块的可编程计数器/定时器阵列; 片内具有看门狗定时器、VDD 监视器和温度传感器。对于不同的具体型号, 外设种类和数量有差别。

片内 JTAG 调试电路允许对 MCU 进行非侵入式(不占用片内资源)、全速、在系统调试。在使用 JTAG 调试时, 所有的模拟和数字外设都可全功能运行。

C8051F 单片机与标准型 MCS-51 单片机兼容性很好。标准型 51 单片机到 C8051F 单片机的转换, 基本不需要学习新的开发方法。使用 C8051F 甚至比标准型 51 单片机更加简便。

开发软件可以使用专用集成开发环境——Silicon Laboratories IDE; 也可以使用较为通用的 8051 单片机开发软件——KEIL μ Vision2 或 KEIL μ Vision3, 这时要在软件里安装一个补丁包。软件开发方法和其他的 8051 单片机没有什么差别。在源文件开始一般要有一个头文件包含声明, 定义所用单片机的特殊功能寄存器。

5.1.1 C8051F040 的引脚

C8051F040 的引脚定义及封装如图 5-1 所示。主要引脚功能说明见表 5-1。

表 5-1 C8051F040 的主要引脚功能说明

引脚名称	类 型	说 明
TMS	数字输入	JTAG 测试模式选择
TCK	数字输入	JTAG 测试时钟
TDI	数字输入	JTAG 测试数据输入, TDI 在 TCK 上升沿被锁存
TDO	数字输出	JTAG 测试数据输出, 数据在 TCK 的下降沿从 TDO 引脚输出。TDO 输出是一个三态驱动器
MONEN	数字输入	V _{DD} 监视器使能。该引脚接高电平时, 允许内部 V _{DD} 监视器工作; 当 V _{DD} 低于复位门限时, 强制系统复位。接低电平时, 内部 V _{DD} 监视器被关闭
VREF0 和 VREF2	模拟输入	ADC0 和 ADC2 的电压基准输入
V _{RED}	模拟输入	DAC 的电压基准输入
AIN0.0~AIN0.3	模拟输入	ADC0 输入通道 0~3
HVCAP	模拟 I/O	高压差分放大器电容

CAN 总线技术

续表 5-1

引脚名称	类 型	说 明
HVREF	模拟输入	高压差分放大器基准
HVAIN+	模拟输入	高压差分放大器正信号输入端
HVAIN-	模拟输入	高压差分放大器负信号输入端
CANTX	数字输出	CAN 发送端
CANRX	数字输入	CAN 接收端
DAC0 和 DAC1	模拟输出	数模转换器 0 和 1 的电压输出
P0.0~P0.4	数字 I/O	P0 口部分
ALE/P0.5	数字 I/O	复用信号。外部存储器地址总线 ALE 选通或 P0.5
$\overline{RD}/P0.6$	数字 I/O	复用信号。外部存储器接口的读选通或 P0.6
$\overline{WR}/P0.7$	数字 I/O	复用信号。外部存储器接口的写选通或 P0.7
P1.0~P1.7 AIN2.0~AIN2.7 A8~A15	模拟输入 数字 I/O	P1 口。还可作为 ADC2 的输入通道 0~7 非复用方式时,作为地址总线的高 8 位
P2.0~P2.7 A8m~A15m A0~A7	数字 I/O	P2 口。还可复用作为地址总线的高 8 位 非复用方式时,作为地址总线的低 8 位
P3.0~P3.7 AD0~AD7 D0~D7	数字 I/O	P3 口。还可复用作为地址/数据总线的低 8 位 非复用方式时,作为数据总线
P4.0~P4.4	数字 I/O	P4 口的一部分
ALE/P4.5	数字 I/O	复用信号。外部存储器地址总线 ALE 选通或 P4.5
$\overline{RD}/P4.6$	数字 I/O	复用信号。外部存储器接口的读选通或 P4.6
$\overline{WR}/P4.7$	数字 I/O	复用信号。外部存储器接口的写选通或 P4.7
P5.0~P5.7 A8~A15	数字 I/O	P5 口 非复用方式时,作为地址总线的高 8 位
P6.0~P6.7 A8m~A15m A0~A7	数字 I/O	P6 口。还可复用作为地址总线的高 8 位 非复用方式时,作为地址总线的低 8 位
P7.0~P7.7 AD0~AD7 D0~D7	数字 I/O	P7 口。还可复用作为地址/数据总线的低 8 位 非复用方式时,作为数据总线

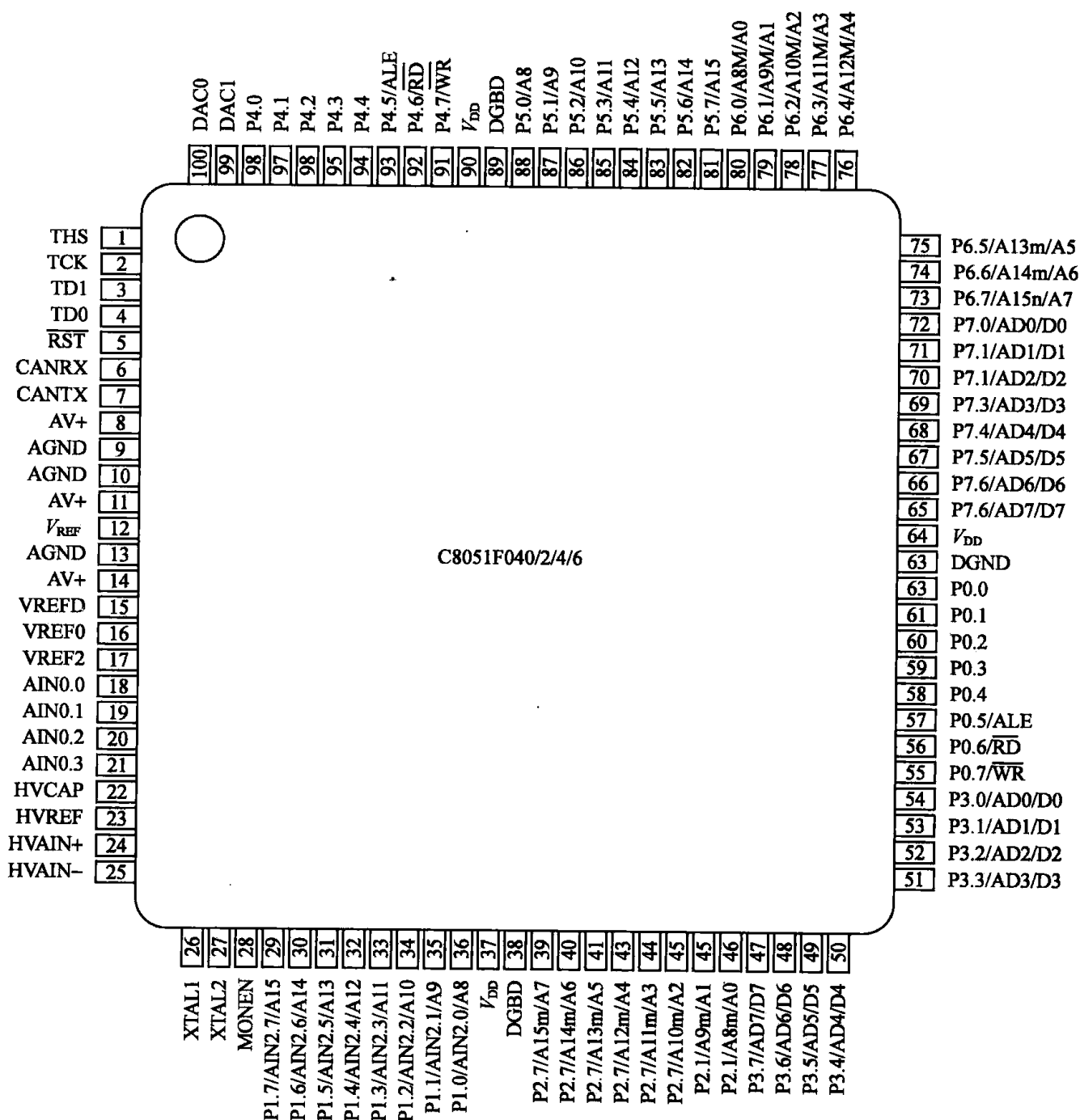


图 5-1 C8051F040 的引脚定义及封装

5.1.2 C8051F040 的 CAN 模块

C8051F040 中集成了一个 CAN 通信控制器,它支持 CAN2.0A(基本 CAN)和 CAN2.0B(全功能 CAN),工作位速率可达 1 Mbps。CAN 通信控制器包含一个 CAN 核、消息缓冲区、消息处理状态机和控制寄存器。CAN 模块的结构如图 5-2 所示。这个 CAN 通信控制器是一个协议控制器,不提供物理层驱动器功能。

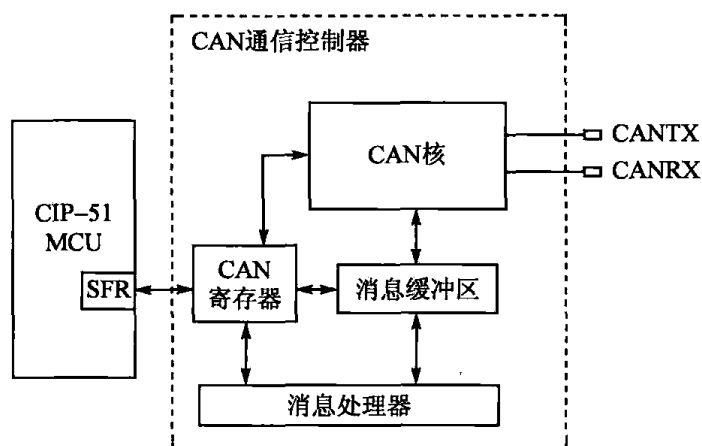


图 5-2 C8051F040 中 CAN 通信控制器的结构

CAN 通信控制器有 32 个消息对象,可以配置为发送或接收数据。输入数据、消息对象及其标识符存储在 CAN 消息缓冲区中,能实现 CAN 协议的数据链路层全部功能及物理层大部分功能。

CIP-51 CPU 对 CAN 模块的操作是通过访问 CAN 模块的寄存器来实现的。CAN 模块的寄存器中有 3 个寄存器 CIP-51 可以直接访问,分别是 CAN 控制寄存器 CAN0CN、CAN 状态寄存器 CAN0STA 和 CAN 测试寄存器 CANTST;其他的寄存器只能通过 CAN 地址寄存器 CAN0ADR、CAN 数据寄存器 CAN0DATH 和 CAN0DATL 以间接方式访问。为便于对消息对象编程,CAN0ADR 在地址范围 08H~12H(接口寄存器 1)和 20H~2AH(接口寄存器 2)内有自动加 1 功能。当 CAN0ADR 中的地址位于这两个范围之内时,CAN0ADR 在每次读/写 CAN0DATL 时自动加 1,指向下一个 CAN 寄存器。当配置消息对象时,这一特性可以加快对频繁访问的接口寄存器的访问速度。

CAN 控制器的寄存器按功能可分为 3 类,分别是 CAN 控制器协议寄存器、报文对象接口寄存器、报文处理寄存器。CAN 寄存器的配置如表 5-2 所列。

1. CAN 控制器协议寄存器

协议寄存器用来配置 CAN 控制器、处理各种中断、监控总线状态以及置控制器为测试模式。这部分的寄存器有 CAN 控制寄存器(CAN0CN)、CAN 状态寄存器(CAN0STA)、CAN 测试寄存器(CANTST)、错误计数寄存器、位定时寄存器和波特率预分频器扩展寄存器。

表 5-2 CAN 寄存器的配置

寄存器地址	寄存器名称	寄存器地址	寄存器名称
00H	控制寄存器	20H	IF2 命令请求
01H	状态寄存器	21H	IF2 命令掩码
02H	错误寄存器	22H	IF2 掩码 1
03H	位定时寄存器	23H	IF2 掩码 2
04H	中断寄存器	24H	IF2 仲裁 1
05H	测试寄存器	25H	IF2 仲裁 2
06H	BRP 扩展寄存器	26H	IF2 消息控制
08H	IF1 命令请求	27H	IF2 数据 A1
09H	IF1 命令掩码	28H	IF2 数据 A2
0AH	IF1 掩码 1	29H	IF2 数据 B1
0BH	IF1 掩码 2	2AH	IF2 数据 B2
0CH	IF1 仲裁 1	40H	发送请求 1
0DH	IF1 仲裁 2	41H	发送请求 2
0EH	IF1 消息控制	48H	新数据 1
0FH	IF1 数据 A1	49H	新数据 2
10H	IF1 数据 A2	50H	中断标志 1
11H	IF1 数据 B1	51H	中断标志 2
12H	IF1 数据 B2	58H	消息有效 1
		59H	消息有效 2

2. 报文对象接口寄存器

CAN 控制器中有两组报文对象接口寄存器,用于配置 32 个消息对象。消息对象可以被配置为发送或接收,并被分配消息标识,以便所有 CAN 节点进行接收过滤。所有的消息对象都存储在消息 RAM 里,通过报文对象寄存器对其访问和配置,但这些寄存器要通过 C8051F 的 CAN0ADR 和 CAN0DAT 寄存器,用间接寻址方式来访问。这部分寄存器有 IFX 命令请求寄存器、IFX 命令屏蔽寄存器、IFX 屏蔽寄存器 1、IFX 屏蔽寄存器 2、IFX 仲裁寄存器 1、IFX 仲裁寄存器 2、IFX 报文控制寄存器、IFX 数据寄存器 A1、IFX 数据寄存器 A2、IFX 数据寄存器 B1 和 IFX 数据寄存器 B2。

3. 报文处理寄存器

消息处理器寄存器用来提供中断、错误、发送/接收请求和新数据信息。通过读取它们的值可以实时地判断相应消息对象的状态,从而使 CAN 控制器正确运行。这部分寄存器包括中断寄存器、发送请求寄存器、新数据寄存器、中断队列寄存器和报文有效寄存器。

CAN 总线技术

表 5-2 中的寄存器地址,是指在 CAN 模块内的内部地址。每个寄存器都有一个内部地址,如果要访问某寄存器,只需将此寄存器的内部地址写入 CAN0ADR 寄存器,数据读/写操作通过 CAN0DATH 和 CAN0DATL 来完成。例如,如果需要对位定时寄存器重新配置,则只需向 CAN0ADR 寄存器中写入 03H,将新配置数据的低字节写入 CAN0DATL 中,高字节写入 CAN0DATH 中。

控制寄存器控制 CAN 模块的工作模式,对位定时和中断进行管理。表 5-3 是控制寄存器的位功能说明。状态寄存器的位功能如表 5-4 所列。对状态寄存器进行读操作,则清除所有的中断标志。

表 5-3 控制寄存器的位功能

位序号	符 号	名 称	说 明
7	Test	测试模式	置 1,进入测试模式;置 0,正常模式
6	CCE	配置改变允许	当 Init=1 时,CCE 位为 1,可以重新初始化位定时寄存器;CCE 位为 0,不能访问位定时寄存器
5	DAR	禁止自动重发	置 1,禁止自动重发功能;置 0,允许自动重发
4	CANIF	CAN 中断标志	0: 未发生 CAN 中断;1: 发生了 CAN 中断
3	EIE	错误中断允许	置 1,允许错误中断;置 0,禁止错误中断
2	SIE	状态改变中断允许	置 1 时,如果成功发送一条消息或检测到 CAN 总线错误,则产生一次中断;置 0 时,禁止这种中断
1	IE	中断允许	置 1,允许中断,中断期间,IRQ-B 保持低电平;置 0,禁止中断
0	Iint	初始化模式	置 1,进入初始化模式;置 0,进入正常模式

表 5-4 状态寄存器的位功能

位序号	符 号	名 称	说 明
7	BOff	总线状态	1: 总线断开;0: 总线未断开
6	EWarn	警告状态	1: 至少有一个错误计数器的值达到最低警告值 96; 0: 两个错误计数器的值都低于警告值 96
5	EPass	通信控制器状态	1: 错误认可状态;0: 错误激活状态
4	RxOK	接收成功	1: 成功接收一条消息;0: 上次复位后尚未成功接收
3	TxOK	发送成功	1: 成功发送一条消息;0: 上次复位后尚未成功发送
2	LEC	错误代码	0: 无错误;1: 填充错误;2: 格式错误
1			3: 应答错误;4: 位“1”监测错误
0			5: 位“0”监测错误;6: CRC 错误;7: 未用

错误代码中,LEC=4时,表示位“1”监测错误。它是指在发送一则消息(除仲裁段)过程中,发送的是隐性位而监测到的是显性位的情况。LEC=5时,表示位“0”监测错误。类似的,是指在发送一则消息(或应答位、活动错误标志、超载标志)过程中,发送显性位而监测到的是隐性位的情况。

测试寄存器提供CAN模块的基本收发功能测试。测试寄存器的位功能如表5-5所列。先把控制寄存器的测试模式控制位Test置1,然后可以访问测试寄存器。

表5-5 测试寄存器的位功能

位序号	符 号	名 称	说 明
7	Rx	引脚CAN-RX的值	1: CAN-RX=1;0: CAN-RX=0
6	Tx1	引脚CAN-TX控制	00: CAN-TX由CAN核控制;01: 在CAN-TX监视采样点; 10: CAN-TX输出0;11: CAN-TX输出1
5	Tx0		
4	LBack	自接收方式	1: 允许自接收;0: 禁止自接收
3	Silent	静音模式	1: 进入静音模式;0: 正常模式
2	Basic	基本模式	1: 进入基本模式,IF1寄存器组用作发送缓冲区,IF2寄存器组用作接收缓冲区;0: 禁止基本模式
1	res	保留	
0	res	保留	

其他的寄存器不做详细说明,请参考相关手册。

5.1.3 CAN寄存器配置

在使用CAN控制器时,要根据所完成的功能对相关的寄存器进行配置。这包括报文对象初始化配置、发送对象配置、接收对象配置、中断处理配置。另外,还有发送对象的更新、位定时寄存器等配置。

初始化CAN控制器的一般步骤如下:

- ① 将SFRPAGE寄存器设置为CAN0_PAGE。
- ② 将CAN0CN寄存器中的INIT和CCE位设置为1。
- ③ 设置位定时寄存器和BRP扩展寄存器中的时序参数。
- ④ 初始化每个消息对象或将其MsgVal位设置为NOT VALID(无效)。
- ⑤ 将INIT位清0。

(1) 报文对象初始化

报文对象在使用前必须由CPU来初始化为零或者设置为无效。报文对象的配置是通过相应的接口寄存器来设置其屏蔽码、仲裁场、控制场和数据场,这些设置过程由相应的IFX命

CAN 总线技术

令请求寄存器来完成。报文 RAM 中的报文对象(除 MsgVal、NewDat、IntPnd 和 TxRqst)配置不受芯片复位的影响。

当 CAN 控制寄存器中的 Init 位置零时,接收到的报文通过接收滤波后都存放在报文 RAM 中,而得到传输请求的报文都要移入 CAN 内核的移位寄存器中并通过 CAN 总线传出。

(2) 发送对象的配置

当报文对象作为发送对象时,仲裁寄存器定义报文的标识符和类型,如果使用 11 位标识符(标准帧),那么使用的是 ID28~ID18,而 ID17~ID0 将被忽视。如果 TxIE 位被置位,则 IntPnd 位在此报文对象成功发送后置位;如果 RmtEn 位被置位,则接收到对应的远程帧时 TxRqst 位被置位。屏蔽寄存器(Msk28-0、Umask、Mxtd 和 MDir 位)可以用来(UMask=1)允许相同识别符的远程帧组将 TxRqst 置位。

(3) 接收对象的配置

当报文对象作为接收对象时,仲裁寄存器定义报文的识别符和类型,如果使用 11 位标识符(标准帧),那么使用的是 ID28~ID18,而 ID17~ID0 将被忽视。当接收到带有 11 位标识符的数据帧,则 ID17~ID0 被置位。如果 RxIE 位被置位,则当一个接收到的数据帧接收并存储在报文对象中时,IntPnd 位将被置位。数据寄存器(DLC3~0)指示数据长度,当报文处理存储一个数据帧到报文对象时,它将存储接收到的数据长度代码和 8 字节数据。如果数据长度代码少于 8,则剩下的报文对象字节是不确定值。屏蔽寄存器(Msk28-0、Umask、MXtd 和 MDir 位)可以用来(UMask=1)允许相同识别符的数据帧组被接收。

(4) 中断处理

在所有中断中,状态中断具有最高优先级,报文对象的中断优先级随着报文编号的增大而减小。

状态中断通过读取状态寄存器来清除,报文中断通过清除报文对象的 IntPnd 位来清除。处于中断寄存器中的中断识别符 Intld 能表明中断的原因,如果这个寄存器的值为 0,则没有发生中断。

CPU 控制着状态寄存器的改变是否可以引起中断。CPU 有两种方式判断报文中断源,第一种是判断中断寄存器中的 Intld 位,另一种是顺序扫描中断发生寄存器。

5.1.4 C8051F040 的 CAN 通信实例

本小节给出了 CAN 模块初始化、中断方式的发送及接收程序。程序中使用的寄存器名称要进行预定义,如下:

```
#include <c8051f040.h>
#define CANCTRL      0x00           //控制寄存器
#define CANSTAT      0x01           //状态寄存器
.....
```

```
#define MSGVAL2          0x59
sfr16    CANODAT = 0xD8;
```

位定时计算:

系统时钟频率 $f_{\text{sys}} = 22.1184 \text{ MHz} / 2 = 11.0592 \text{ MHz}$

系统时钟周期 $t_{\text{sys}} = 1/f_{\text{sys}} = 90.422454 \text{ ns}$

CAN 时间份额 $t_q = t_{\text{sys}} (\text{at BRP} = 0)$

位速率设为 1 Mbps, 则位时间是 1000 ns。

实际位时间 $= 11 t_q = 996.65 \text{ ns} \sim 1000 \text{ ns}$

实际位速率 $= 1.005381818 \text{ Mbps} = \text{期望位速率} + 0.5381\%$

假设 CAN 总线长度是 10 m, 信号延迟时间按 5 ns/m 计算, 则

信号传播延时: $2 \times (\text{transceiver loop delay} + \text{bus line delay}) = 400 \text{ ns}$

传播段时间设为 Prop_Seg $= 5 t_q = 452 \text{ ns} (\geq 400 \text{ ns})$

同步段时间设为 Sync_Seg $= 1 t_q$

Phase_seg1 + Phase_Seg2 $= (11 - 6) t_q = 5 t_q$

Phase_seg1 $\leq$ Phase_Seg2, $\Rightarrow$ Phase_seg1 $= 2 t_q$ and Phase_Seg2 $= 3 t_q$

SJW $= (\min(\text{Phase_Seg1}, 4) t_q = 2 t_q$

TSEG1 $= (\text{Prop_Seg} + \text{Phase_Seg1} - 1) = 6$

TSEG2 $= (\text{Phase_Seg2} - 1) = 2$

SJW_p $= (\text{SJW} - 1) = 1$

所以, 位定时寄存器 $= \text{BRP} + \text{SJW_p} \times 0x0040 = \text{TSEG1} \times 0x0100 + \text{TSEG2} \times 0x1000 = 2640$

初始化程序:

```
void config_I0(void)
{
    WDTCN = 0xde;           //关闭看门狗
    WDTCN = 0xad;
    int n;
    SFRPAGE = CONFIG_PAGE; //SFR 设为配置页
    OSCXCN = 0x77;          //外部时钟 22.1 MHz, 系统时钟 11.05 MHz
    for (n = 0; n < 255; n++); //延时 1 ms
    while ((OSCXCN & 0x80) == 0); //时钟设置完成
    CLKSEL |= 0x01;         //设置为外部时钟
    XBR3 = 0x80;            //CAN-TX 设置为推挽数字输出方式
    XBR2 = 0x40;            //允许端口交叉开关
}
```

CAN 总线技术

消息对象清 0 程序:

```
void clear_msg_objects (void)
{
    SFRPAGE = CAN0_PAGE;           //SFR 设置为 CAN 寄存器页
    CAN0ADR = IF1CMDMSK;           //指向命令掩码寄存器 1
    CAN0DATL = 0xFF;               //对 IF 消息对象设为写入
    for (i = 1; i < 33; i++)
    {
        CAN0ADR = IF1CMDRQST;      //消息对象清 0
        CAN0DATL = i;
    }
}
```

消息对象初始化为接收对象:

```
void init_msg_object_RX (char MsgNum)
{
    SFRPAGE = CAN0_PAGE;
    CAN0ADR = IF1CMDMSK;
    CAN0DAT = 0x00B8;              //写入方式,除 ID MASK 和数据位外
    CAN0ADR = IF1ARB1;             //IF1ARB1 = 0
    CAN0DAT = 0x0000;
    CAN0DAT = 0x8004;              //Arb2 设置: 标准帧,接收
    CAN0DAT = 0x0480;              //禁止远程帧
    CAN0ADR = IF1CMDRQST;          //命令请求寄存器
    CAN0DATL = MsgNum;             //选择消息对象,初始化为写入
    // 3~6 CAN 时钟周期后 IF 的内容移入消息对象
}
```

初始化消息对象为发送对象:

```
void init_msg_object_TX (char MsgNum)
{
    SFRPAGE = CAN0_PAGE;
    CAN0ADR = IF1CMDMSK;
    CAN0DAT = 0x00B2;              //写入方式,除 ID MASK 外
    CAN0ADR = IF1ARB1;
    CAN0DAT = 0x0000;              //仲裁段 1 的标识符设为最高优先级
    CAN0DAT = 0xA000;              //Arb2 设置: 标准帧,写入
    CAN0DAT = 0x0081;              //消息控制: DLC = 1,禁止远程帧
    CAN0ADR = IF1CMDRQST;
```

```

CANODAT = MsgNum;           //选择消息对象,初始化为写入
//3~6 CAN 时钟周期后 IF 的内容移入消息对象
}

```

CAN 通信初始化:

```

void start_CAN (void)
{
    SFRPAGE = CAN0_PAGE;
    CANOCN |= 0x41;           //CCE 和 INIT 设置
    CANOADR = BITREG;         //位定时寄存器设置
    CANODAT = 0x2640;
    CANOADR = IF1CMDMSK;
    CANODAT = 0x0087;         //TX 配置: 数据写入 CAN RAM
    //RX - IF2 的操作可以中断 TX - IF1 的操作
    CANOADR = IF2CMDMSK;
    CANODATL = 0x1F;          //RX 配置: 从 CAN RAM 读取数据
    //清除 NewDat 和 IntPnd
    CANOCN |= 0x06;           //允许 IE、SIE
    CANOCN &= ~0x41;          //清除 CCE 和 INIT,准备 CAN 通信
}

```

发送程序: 把数据 55H 发给消息 "MsgNum":

```

void transmit_turn (char MsgNum)
{
    SFRPAGE = CAN0_PAGE;      //IF1 已经准备好发送
    CANOADR = IF1CMDMSK;
    CANODAT = 0x0087;         //TX 配置: 数据写入 CAN RAM
    //置位 TXrqst/NewDat,清除 IntPnd
    CANOADR = IF1DATA1;        //指向第一个数据字节
    CANODATL = 0x55;
    CANOADR = IF1CMDRQST;
    CANODATL = MsgNum;         //新数据发送给消息 "MsgNum"
}

```

接收程序:

```

void receive_data (char MsgNum)
{
    char recive;
    SFRPAGE = CAN0_PAGE;      //IF1 已经设置为接收
}

```

CAN 总线技术

```

CAN0ADR = IF2CMDRQST;
CAN0DATL = MsgNum;           //新数据从消息 "MsgNum"接收
CAN0ADR = IF2DATA1;          //数据存储
recive = CAN0DATL;
}

```

中断服务程序:

```

void ISRname (void) interrupt 19
{
    status = CAN0STA;
    if ((status&0x10) != 0)
    {
        //RxOk = 1,接收中断
        CAN0STA = (CAN0STA&0xEF)|0x07; //复位 RxOk, LEC 设为未变
        /* 读取消息 */
        receive_data (0x01);           //只接收到一条消息
    }
    if ((status&0x08) != 0)
    {
        //TxOk = 1,发送中断
        CAN0STA = (CAN0STA&0xF7)|0x07; //复位 TxOk, LEC 设为未变
    }
    if (((status&0x07) != 0)&&((status&0x07) != 7))
    {
        //错误中断, LEC 的值有变化
        /* 错误处理 */
        CAN0STA = CAN0STA|0x07;        //LEC 设为未变
    }
}

```

5.2 TMS320F2812

5.2.1 TMS320F2812 概述

数字信号处理器(DSP)芯片是实现数字信号处理技术的专门载体,在数字化时代中有着重要的地位。TI公司是目前最大的DSP研究生产厂商,生产的TMS320C28X系列数字信号处理器,是32位DSP控制器,内部集成了优化的电动机控制电路、丰富多事件管理功能,适合于数字电动机控制、数字电源等应用。TMS320C28X采用了优化的哈佛总线结构,结合哈佛总线和冯·诺依曼总线的优势,如能够从前序空间初始化数据空间,从数据空间将数据传输到程序空间等;这种能力使得C28X能够处理分时和多元化的任务,减少了系统成本,节省了数

据 ROM 而且增加了数据 RAM 的利用率。将程序空间和数据空间分开的好处是系统可以同时存取程序和数据。在一些比较复杂的运算中,相对于其他的处理器,这种能力极大地提高了运算速度。

TMS320F2812 具有 8 级指令流水线,时钟周期 6.67 ns,具有很强的运算能力。片内集成了多种存储器,程序存储器可加密。芯片内的主要外设:12 位 16 通道 AD 转换器,2 个事件管理器,3 个 CPU 定时器,56 个可编程输入/输出引脚。

TMS320F2812 具有多种通信接口,包括 2 个串行异步通信接口(SCI)、1 个串行外设接口(SPI)、1 个多通道缓冲串行接口(McBSP)及 1 个改进的 CAN 接口 eCAN。

F2812 有两种低功耗模式,在这两种模式下外设时钟都会被关闭。在 CAN 模块工作前,必须使能模块的时钟。与其他外设一样,复位后 CAN 模块的时钟被屏蔽。eCAN 有局部掉电模式,模块可以自己重新激活 CAN 内部时钟。

CAN 模块通过网络连接着多个节点,所以进入和退出低功耗模式必须特别谨慎,以保证所有节点接收到完整的数据包。而且,在掉电模式和自动唤醒过程中接收的第一个消息被丢弃。

TMS320F2812 的引脚较多,在此不再介绍,请参阅相关专业资料。下面只对有关 CAN 通信控制器的部分进行说明。

5.2.2 CAN 模块的结构

1. eCAN 的结构和工作原理

TMS320F2812 的 CAN 模块由 CAN 协议内核和消息控制器组成。消息控制器包括 CPU 接口、存储器管理单元、接收控制单元、定时器管理单元、控制和状态寄存器、消息邮箱存储器等部分,如图 5-3 所示。

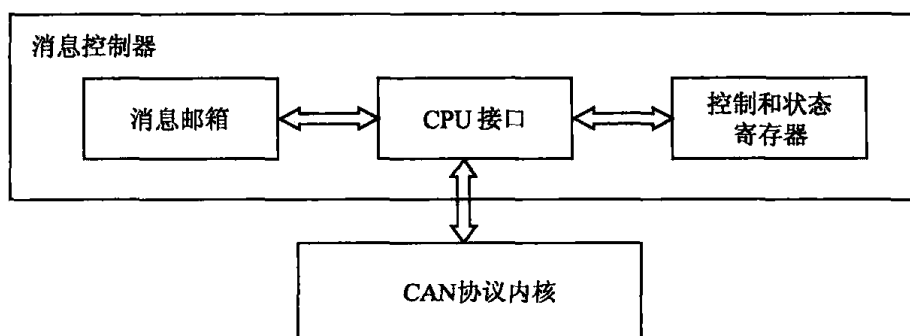


图 5-3 eCAN 模块的结构

消息邮箱共有 32 个,每个邮箱空间有 16 字节。CPU 接口部分包括存储器管理单元、接收控制单元以及定时器管理单元这 3 个主要功能电路。CAN 协议内核中有通信缓冲器,包括发送缓冲器和接收缓冲器,但用户不能访问。

CAN 总线技术

CAN 模块接收到一个有效的消息后,接收控制单元检验消息的状态、标识符和消息对象的屏蔽,经过接收过滤再确定是否把接收的消息存储到某个消息邮箱中。接收的消息存放在第一个邮箱中。如果接收控制单元不能找到任何一个存储接收消息的邮箱,则消息被丢掉。

这里的一个消息是指一个标准或扩展格式的数据帧。当要发送一个消息时,消息控制器把这个消息发送到发送缓冲器中,在下一个总线空闲状态发送消息。当有多个消息要发送时,最高优先级的消息将被消息控制器发送。如果两个邮箱具有相同的优先级,则大序号的邮箱首先发送消息。定时管理单元提供接收或发送消息的定时标志。在规定的时间内(超时值)如果不能完成接收或发送,则产生一个超时中断。

CAN 模块可以工作在标准 CAN 控制器(SCC)模式或者完整功能的 eCAN 模式。改变 SCB 位(CANMC.13)可在两种模式间切换。SCC 模式是 eCAN 的简化模式,在这种模式中只使用 16 个邮箱(0~15),并省略了时间标志特性,减少了可用的接收屏蔽的数目。在默认状态下,选择 SCC 模式。

2. eCAN 寄存器及存储器映射

eCAN 的寄存器被映射到外设帧 1(PF1)区域,有两个不同的地址映射段:第一个地址段用于访问控制寄存器、状态寄存器、接收屏蔽、时间标志和消息对象的超时寄存器,地址范围是 6000H~60FFH。其中,6000H~603FH 是控制和状态寄存器,6040H~607FH 是本地接收屏蔽寄存器,6080H~60BFH 是时间标志寄存器,60C0H~60FFH 是超时寄存器。CPU 用这些寄存器对 CAN 的消息对象进行配置和控制。控制和状态寄存器不支持字节和位操作,只允许 32 位存取。其他几个寄存器可以存取 8 位、16 位和 32 位宽度的数据。第二个地址段用于访问消息邮箱,邮箱映射地址从 6100H~61FFH,共 32 个。这个存储区域也可以存取 8 位、16 位和 32 位宽度的数据。这些寄存器的具体格式和功能不再详细介绍,请读者参阅相关资料。

eCAN 提供了 32 个具有 8 字节数据、29 位标识符和几个控制位的消息邮箱,每个邮箱都可以配置为发送或接收,并且每个邮箱都有自己的接收屏蔽位。

如果不使用 eCAN 功能,则接收屏蔽、时间标志、超时寄存器和邮箱 RAM 都可以用作通常目的的数据存储器。

3. 消息邮箱

eCAN 的数据发送和接收是通过消息邮箱完成的。消息邮箱是一块 RAM 区域,用来存储接收到的 CAN 消息或者存放等待发送的 CAN 消息。每个邮箱由 4 个 32 位寄存器构成,分别是消息标识符寄存器(MSGID)、消息控制寄存器(MSGCTRL)、消息数据低 4 个字节(MDL)和消息数据高 4 个字节(MDH)。

消息标识符寄存器存储消息标识符 ID,各位功能如表 5-6 所列。消息控制寄存器的功能如表 5-7 所列。

第5章 具有CAN接口的处理器

表 5-6 消息标志符寄存器的功能

位	名 称	说 明
31	IDE	标识符扩展位。位的功能根据接收屏蔽标志扩展位 AMI(CANGAM-31)的值改变： 当 AMI=1(可以接收标准帧和扩展帧)时，接收邮箱的 IDE 被发送消息的 IDE 覆盖，不起作用；为了能够接收到消息，必须符合滤波的规定；比较位的数目是发送消息的 IDE 值的一个函数 IDE=1 时，接收到消息有扩展标识符；IDE=0 时，接收到消息有标准标识符 当 AMI=0(邮箱中存放的标识符扩展位确定接收哪些消息)时，接收邮箱的 IDE 决定比较位的数目；不使用滤波，为了能够接收到消息，MSGID 必须各位都匹配 IDE=1 时，要被接收的消息必须有扩展标识符；IDE=0 时，要被接收的消息必须有标准标识符
30	AME	接收屏蔽使能位。此位只在接收邮箱中使用。若设置为自动应答，则邮箱不能确定邮箱的操作。消息接收不改变该位。 1：使用相应的接收屏蔽使能位；0：不接收屏蔽使能位，所有标志必须与接收到的消息匹配
29	AMM	自动应答模式位。此位只在发送邮箱中使用，对接收邮箱没有影响。邮箱总是配置为标准接收操作。消息接收不改变该位。 1：自动应答模式。如果接收到匹配的远程帧请求，CAN 模块通过发送邮箱中的内容应答远程帧请求 0：标准发送模式。邮箱不能应答远程请求，接收到远程帧请求对消息邮箱并没有影响
28~0	ID28~0	消息标识符 ID。标准消息模式。如果 IDE=0，则消息标识符存放在 ID28~18，ID17~0 无意义。扩展消息模式。如果 IDE=1，则消息标识符存放在 ID28~0

表 5-7 消息控制寄存器的功能

位	名 称	说 明
31~13		保留
12~8	TPL, 4~0	发送优先级：定义邮箱相对于其他 31 个邮箱的优先级，数越大优先级就越高。当 2 个邮箱优先级一样时，邮箱编号大的先发送。TPL 只对发送邮箱有效。在标准消息模式不使用 TPL
7~5		保留
4	RTR	远程发送请求位。1：对于接收邮箱，如果 TRS 置位，则发送远程帧，并且在相同邮箱中接收相应的数据帧。一旦发送远程帧，CAN 将邮箱的 TRS 位清零；对于发送邮箱，如果 TRS 置位，发送远程帧，但相应的数据帧必须在其他邮箱接收
3~0	DLC3~0	数据长度代码。说明发送或接收数据的字节数，0~8 有效

CAN 总线技术

低 4 字节数据和高 4 字节数据顺序由位 DBO(CANMC.10)确定。

消息邮箱可以配置为 4 种消息类型,分别是发送消息对象、接收消息对象、请求消息对象或应答消息对象。发送和接收消息对象用于在发送器和多个接收器之间进行数据交换,请求和应答消息对象用于建立一对一的通信连接。消息对象的功能配置方法如表 5-8 所列。

表 5-8 消息对象的功能配置

消息类型	邮箱方向寄存器 CANMD	自动应答位 AAM	远程请求位 RTR
发送消息对象	0	0	0
接收消息对象	1	0	0
请求消息对象	1	0	1
应答消息对象	0	1	0

CPIJ 把要发送的数据存储在发送邮箱中。先把发送邮箱的使能位置 1,然后将数据和标识符写到发送邮箱,再置位对应的 TRS 位,则邮箱中的消息就会发送出去。如果有多个邮箱设置为发送邮箱并且多个相应的 TRS 被置位,则按照邮箱优先级的高低,依次将消息发送出去。

在 SCC 模式下,邮箱的号码代表发送邮箱的优先级,号码越大优先级越高。

在 eCAN 模式下,发送邮箱的优先级由消息控制寄存器的 TPL 位组确定,TPL 值越大,则优先级越高。当邮箱的 TPL 值相同时,邮箱号较大的优先发送。

如果由于缺少仲裁或错误使发送失败,则自动重发。但是重新发送前,要再次对所有的待发送对象进行优先级排队。

在接收消息时,消息控制器开始从邮箱号最高的邮箱搜索标识符匹配的邮箱。接收邮箱首先比较输入消息的标识符和邮箱中存放的标识符,如果相等,则消息存入邮箱。同时,相应的接收消息挂起位 RMP 由内部逻辑自动置位。如果中断使能,则产生中断。如果两者的标识符不一致,则不存储接收的消息。

读取完数据后,CPU 必须将 RMP 复位。消息接收情况还受覆盖保护位 OPC 的影响。

如果接收邮箱的 RTR 位置位,则该邮箱可以发送一个远程帧。远程帧发送完成后,CAN 模块就会清除邮箱的 TRS 位。

5.2.3 eCAN 配置

1. eCAN 的初始化

使用 eCAN 模块先要进行必要的初始化过程,这需要进入初始化模式。检测到总线空闲状态后,可以完成初始化模式和正常操作模式之间的转换。设置 CCR(CAMMC.12)为 0,则可以使 CAN 模块处于工作模式。硬件复位后,CAN 处于初始化模式。

CCR(CANMC12)置1时,CAN模块工作在初始化模式,可以对寄存器进行设置;控制位CCE(CANES.4)=1时,CAN模块寄存器的配置值完成写操作,开始生效。

对eCAN的初始化步骤如下:

① 使能CAN模块时钟。

② 设置CANTX和CANRX为CAN通信引脚(CANTIOC.3:0=0x08;CANRIOC.3:0=0x08)。

③ 复位后CCR(CANMC.12)和CCE(CANES.4)为1,允许用户配置位时间寄存器(CANBTC);如果CCE位置1,进行下一步;否则,CCR置1,然后等待,直到CCE置1。

④ 配置CANBTC,确认TSEG1和TSEG2不等于0;如果两个值都等于0,则CAN模块无法退出初始化模式。

⑤ 对于标准CAN模式,此时可以对验收屏蔽寄存器编程。

⑥ 对主控制器CANMC编程:使CANMC.12~0=0x0000,CCR、PDR、DBO、WUBA、CDR、ABO、STM、SRES和MBNR等控制位为0。

⑦ 初始化MSGCTRLn寄存器的所有位为0。

⑧ 检查CCE是否清零,如果为0,则表示CAN模块已经配置完成。

一个初始化eCAN模块的程序如下:

```
#include"DSP28 - Device.h"
void InitECAN(void)
{
    long i;
    asm("EALLOW");
    SysCtrlRegs.WDCR = 0x006F           / 禁止 watchdog /
    SysCtrlRegs.PCLKCR.all = 0x4000;    / 使能 CAN 模块时钟 /
    / 设置 PLL 倍频系数为 5 倍,外部时钟 30 MHz,得到 150MHz 的 SYSCLKOUT /
    SysCtrlRegs.PLLCR.bit.DIV = 0x000A;
    for( i = 0; i < 100000; i++)        / 延时等待锁相环 PLL 稳定 /
    { ; }
    / 把 CANRX 和 CANTX 引脚设置为 CAN 功能引脚 /
    GpioMUXRegs.GPFMUX.bit.CANTXA - GPIOF6 = 1;
    GpioMUXRegs.GPEMUX.bit.CANRXA - GPIOF7 = 1;
    / 把 CANRX 和 CANTX 引脚设置为接收和发送引脚 /
    ECanaRegs.CANTIOC.bit.TXFUNC = 1;
    ECanaRegs.CANRIOC.bit.RXFUNC = 1;
    / eCAN 设为增强功能模式,32 个邮箱可以全部访问,支持定时邮箱功能 /
    ECanaRegs.CANMC.bit.SCB = 1;
    / 初始化 32 个邮箱的“主控制域”为 0,所有的 MCF 位都初始化为 0 /
    EcanaMboxes.MBOX0.MCF.all = 0x00000000;
```

CAN 总线技术

```

EcanaMboxes.MBOX1.MCF.all = 0x00000000;
.....
EcanaMboxes.MBOX31.MCF.all = 0x00000000;
/ 清除所有的 TAn 位 /
ECanaRegs.CANTA.all = 0xFFFFFFFF;
/ 清除所有的 RMPn 位 /
ECanaRegs.CANRMP.all = 0xFFFFFFFF;
/ 清除中断标志位 /
ECanaRegs.CANGIF0.all = 0xFFFFFFFF;
ECanaRegs.CANGIF1.all = 0xFFFFFFFF;
/ 位定时设置 /
ECanaRegs.CANRMC.bit.CCR = 1;
while(ECanaRegs.CANES.bit.CCE = 0;){}
ECanaRegs.CANBTC.bit.BRP = 9;
ECanaRegs.CANBTC.bit.TSEG1 = 10;
ECanaRegs.CANBTC.bit.TSEG2 = 2;
ECanaRegs.CANRMC.bit.CCR = 0;
while(ECanaRegs.CANES.bit.CCE = 1;){}
/ 邮箱全部静止 /
ECanaRegs.CANME.all = 0;

```

2. 邮箱配置

发送消息要把邮箱配置为发送邮箱,以邮箱 1 为例,配置步骤如下:

① 清除 CANTRS 寄存器中相应的位。

② 清除 CANTRS.1(写 0 到 TRS 没有影响,必须通过设置 TRR.1 为 1,等待 TRS 自动清零)。如果 RTR 置 1,则 TRS 位可以发送一个远程帧;发送完后,邮箱相应的 TRS 位清除。同样的节点可以向其他节点请求一个数据帧。

③ 清除邮箱使能寄存器 CANME 中的相应位,禁止邮箱 1(CANME.1=0)。

④ 设置邮箱的消息标识符寄存器 MSGID,清除 AME(MSGID.30)和 AAM(MSGID.29),把邮箱配置为标准发送邮箱。这个寄存器只能在邮箱禁止时修改,在操作过程中不能修改。

⑤ 数据长度写到消息控制寄存器的 DLC(MSGCTRL.3),通常清除 RTR 标志(MSGCTRL.4=0)。这个寄存器也只能在邮箱禁止时修改。

⑥ 清除邮箱方向寄存器 CANMD 中相应位,将邮箱设置为发送邮箱(CANMD.1=0)。

⑦ 通过设置邮箱使能寄存器 CANME 中相应位使能邮箱(CANME.1=1)。

使用发送邮箱发送消息的操作过程如下:

① 写消息到邮箱的数据区:配置时,DBO(MC.10)=0,MSGCTRL(1)=1;数据存放在 CANMDL(1)的两个高字节;写 CANMDL(1)=xxxx0000H。

② 设置发送请求寄存器中相应的标志位(CANTRS.1=1),启动消息发送。

③ 成功发送消息后,CAN模块将该位发送响应标志位置位(TA.1=1)。

④ 成功发送或中止发送后,模块将 TRS 标志复位为 0(TRS.1=0)。

⑤ 通过把 TA.1 设置为 1 而将发送响应标志清零。

⑥ 刷新邮箱的数据区,以便邮箱发送其他消息。写到邮箱中的数据可以是 16 位或 32 位,但模块总是返回 32 位数据,CPU 必须接收所有 32 位或 32 位中的一部分。

按照以下方法,把邮箱配置为接收邮箱。下面以邮箱 3 为例说明设置步骤。

① 清除邮箱使能寄存器 CANME 中相应位,禁止邮箱 3(ME.3=0)。

② 写消息标识符到相应的 MSGID 寄存器,并根据需要配置标识符扩展位。如果使用接收屏蔽,则接收屏蔽使能位 AME 必须置位(MSGID.30=1)。

③ 如果 AME=1,则相应的接收屏蔽必须进行编程。写 LAM(3)=0x03C0000。

④ 清除邮箱方向寄存器 CANMD 中相应位,将邮箱设置为接收邮箱(CANMD.3=1)。

⑤ 如果邮箱中的数据受保护,则要对覆盖控制寄存器 CANOPC 进行设置从而使消息不丢弃。当保护位 OPC 被置位时,必须确保有其他的邮箱存储“溢出”的消息,以免消息被丢弃。

⑥ 通过设置邮箱来使能寄存器 CANME 中的相应位,从而使能邮箱。必须采用读取然后写入的方式,这可以保证其他标志位不会被改变。

通过以上设置,邮箱 3 就设置为接收邮箱。当接收邮箱接收到消息时,则将接收到消息的标志寄存器 CANRMP 相应的标志位置 1,并产生一个中断。CPU 可以从邮箱 RAM 中读取消息。在读取消息之前 CPU 应该先清除 RMP(RMP.3)位,并检查消息丢弃标志 RML.3,根据具体应用,CPU 决定如何处理这些情况。

读取数据后,CPU 需要验证 RMP 没有被模块再次置位;如果 RMP 置位,则说明数据可能已经被破坏。

如果 CPU 不能及时地处理重要消息,则可以配置多个具有相同标识符的邮箱。具有相同标识符的消息对象共享一个屏蔽。对于标准 CAN 模式,使用 LAM(3)屏蔽;对于 eCAN 模式,每个消息对象有自己的屏蔽 LAM(3)、LAM(4)和 LAM(5),3 个屏蔽要使用相同的配置值。

为了保证不丢失消息,则可以将一部分消息对象设为保护,以防覆盖;也可以使消息对象产生中断,读取邮箱。

另外,节点还要处理远程帧。远程帧处理有两个方面:向其他节点发出数据请求和应答其他节点发出的数据请求。

为了从其他节点获取数据,消息对象配置为接收邮箱。以消息对象 3 为例,CPU 需要完成以下操作:

① 将 RTR 位置位,MSGCTRL(3)=0x12。

② 把消息标识符写到 MSGID,MSGID(3)=0x4F780000。

CAN 总线技术

③ 将相应邮箱的 CANTRS 标志置位, $CANTRS.3=1$ 。由于邮箱配置成接收邮箱,它只能向其他邮箱发送一个远程请求消息。

④ 接收到消息时,模块将应答消息存放在邮箱中,并置位 RMP。这样可以产生一个中断,且确认没有其他邮箱有相同的 ID。等待 $RMP.3=1$ 。

⑤ 读取接收到的消息。

应答一个远程请求,经过以下过程:

① 配置邮箱为发送邮箱。

② 将 MSGID 中的自动应答模式 AAM(MSGID-29)位置 1。

③ 刷新数据区 MDL 和 MDH。

④ 将邮箱使能寄存器 CANME 置位,使能邮箱。

当收到来自其他节点的远程请求时,TRS 自动置位并且数据被发送到那个节点。接收和发送消息的标识符相同。数据发动完后,TA 置位,CPU 可以刷新数据。

5.2.4 eCAN 中断

eCAN 模块的中断有两种:一种是同邮箱相关的中断,如接收到消息中断或中止响应中断;另一类是系统中断,用于处理错误或与系统相关的中断,如错误消息中断。

邮箱相关的中断包括接收到消息、成功发送消息、中止响应、接收消息丢失和邮箱超时中断。系统中断包括拒绝 CPU 写邮箱、唤醒、脱离总线、错误、警告级和定时器超时中断。

CAN 的每个邮箱可以在中断输出线 1 或 0 上产生一个中断,根据邮箱的配置不同,可以产生接收中断或发送中断。

eCAN 中断的配置主要包括邮箱中断屏蔽寄存器(CANMIM)、邮箱中断级别寄存器(CANMIL)和全局中断屏蔽寄存器(CANGIM)。通过设置这些寄存器可以确定使能哪些中断源通过哪条中断线中断。

如果满足中断条件,则相应的中断标志位就会置位。中断标志位的置位根据 GIL 位(CANGIM.2)的设置情况而定。如果 $GIL=1$,则全局中断标志寄存器 CANGIF1 置位;如果 $GIL=0$,全局中断标志寄存器 CANGIF0 置位。

中断标志位还根据产生中断邮箱的 MIL[n]的置位情况而定,如果 $MIL[n]=1$,则相应邮箱的中断标志 MIF[n]将寄存器 CANGIF1 中的 GMIF1 标志置位;如果 $MIL[n]=0$,则相应邮箱的中断标志 MIF[n]将寄存器 CANGIF0 中的 GMIF0 标志置位。

如果 GMIF0 或 GMIF1 标志置位,则邮箱中断向量 MIV0(CANGIF0.4~0)或 MIV1(CANGIF1.4~0)给出最高优先级的邮箱编号。

每个邮箱有一个专门的屏蔽位 MIM[n]和中断级选择位 MIL[n],MIM 必须置位才能够在接收或发送消息产生中断,接收邮箱接收到消息($RMP[n]=1$)或者发送邮箱发出消息($TA[n]=1$)就都会产生中断。如果邮箱配置成远程请求邮箱($CANMD[n]=1$,MSGCTRL.RTR=1),则

一旦接收到远程帧应答就会产生中断。远程应答邮箱在成功发送应答帧后(CANMD[n]=0, MSGID. AAM=1),产生一个中断。

CAN通过ECAN0INT和ECAN1INT向CPU申请中断,一旦有中断响应,CPU可以通过MIV0、MIV1或者相关的中断标志位判断出是哪个源产生的中断。

处理中断后,CPU要清除中断源和中断标志。向相应的标志位写1,可清除CANGIF0和CANGIF1寄存器中的大多数中断标志。但也有个别例外,如RM-LIFn由RMPn置位或清零;AAIFn由AAn置位或清零;GMIFn清零是通过写1到寄存器CANTA或CANRMP相应的位实现;MTOFn清零是通过写1到TOSn位实现的。

在中断服务程序中,一般进行如下处理:

- ① 清除CANGIF0/CANGIF1寄存器中引起中断的标志位;
- ② CAN模块相应的PIEACK位必须写1;
- ③ 使能CAN模块中使用的相应中断线;
- ④ 清除INTM,开放中断系统。

下面的程序是实现CAN模块的通信自检测,给出了CAN模块的简单使用方法。在自测试模式下,邮箱0~15分别发送数据给邮箱16~31。发送和接收连续进行,并对接收数据进行核对,计数错误次数。

```
# include "DSP28 - Device.h"
# define TXCOUNT 1000                                / 循环发送次数 /
long i;
int j;
long loopcount = 0;
long errorcount = 0;
unsigned long TextMbox1 = 0;
unsigned long TextMbox2 = 0;
unsigned long TextMbox3 = 0;
void InitECan(void);
void MBXcheck(long T1,long T2,long T3);
void MBXrd(int i);
main()
{
    InitECan();                                          / 初始化 CAN 模块 /
    EcanaMboxes.MBOX0.MSGID.all = 0x9555AAAO;          / 写发送邮箱的标识符 /
    .....
    EcanaMboxes.MBOX15.MSGID.all = 0x9555AAAF;
    EcanaMboxes.MBOX16.MSGID.all = 0x9555AAAO;          / 写接收邮箱的标识符 /
    .....
    EcanaMboxes.MBOX31.MSGID.all = 0x9555AAAF;
```

CAN 总线技术

```

EcanaMboxes.MBOX16.MDRL.all = 0;           / 接收邮箱的 RAM 区清 0 /
EcanaMboxes.MBOX16.MDRH.all = 0;
.....
EcanaMboxes.MBOX31.MDRL.all = 0;
EcanaMboxes.MBOX31.MDRH.all = 0;
EcanaRegs.CANMD.all = 0xFFFF0000;          / 邮箱 0~15 为发送, 邮箱 16~31 为接收 /
EcanaMboxes.MBOX0.MCF.bit.DLC = 8;          / 写主控制域 /
.....
EcanaMboxes.MBOX15.MCF.bit.DLC = 8;
EcanaMboxes.MBOX0.MDRL.all = 0x9555AAA0;    / 发送数据写入邮箱 0~15 /
EcanaMboxes.MBOX0.MDRH.all = 0x98ABCDEF;
.....
EcanaMboxes.MBOX15.MDRL.all = 0x9555AAAF;
EcanaMboxes.MBOX15.MDRH.all = 0x98ABCDEF;
EcanaRegs.CANME.all = 0xFFFFFFFF;           / 使能所有邮箱 /
EcanaRegs.CANMC.bit.STM = 1;                 / 设置自测试模式 /
for(i = 0; i < TXCOUNT; i++)               / 开始发送 /
{
    ECanaRegs.CANTA.all = 0xFFFFFFFF;
    ECanaRegs.CANTRS.all = 0x0000FFFF;        / 发送邮箱 TRS = 1 /
    while(ECanaRegs.CANTA.all != 0x0000FFFF) {}
    ECanaRegs.CANTA.all = 0x0000FFFF;
    loopcount++;
    / 从接收邮箱读取数据 /
    for(j = 0; j < 16; j++)                    / 读取 16 个邮箱 /
    {
        MBXrd(j);
        MBXcheck(TestMbox1, TestMbox2, TestMbox3); / 检测接收数据 /
    }
    asm("ESTOP0");                             / 停止 /
}
/ 从接收邮箱读取数据的函数 /
void MBXrd(void)
{
    volatile struct MBOX * Mailbox = (void *)0x6180; / 指向邮箱 16 /
    Mailbox = Mailbox + MBXnbr;
    TestMbox1 = Mailbox -> MDRL.all;             / = 0x9555AAAn, n 是邮箱号 /
    TestMbox2 = Mailbox -> MDRH.all;             / = 0x98ABCDEF /
    TestMbox3 = Mailbox -> MSGID.all;            / = 0x9555AAAn, n 是邮箱号 /
}
/ 发送和接收邮箱数据检查函数 /

```

```

void bfBXcheck(long T1,long T2,long T3)
{
    if((T1!= T3) || (T2!= 0x98ABCDEF))
    {
        errorcount ++;
    }
}

```

5.3 P8xC591

P8xC591 带有 CAN 控制器,内核为 8051,完全支持 CAN2.0B 规范,其增强型验收滤波器支持系统维护诊断系统优化以及接收 FIFO 特性,应用领域广泛。本节将着重介绍其与 CAN 相关部分。

5.3.1 P8xC591 概述

P8xC591 是一款 8 位高性能的微控制器,从 80C51 微控制器家族派生而来,集成了 P87C554(微控制器)和 SJA1000(独立的 CAN 控制器)的功能,采用了强大的 80C51 指令集,并成功地包括了 NXP 半导体 SJA1000CAN 控制器的 PeliCAN 功能。其方框图如图 5-4 所示。

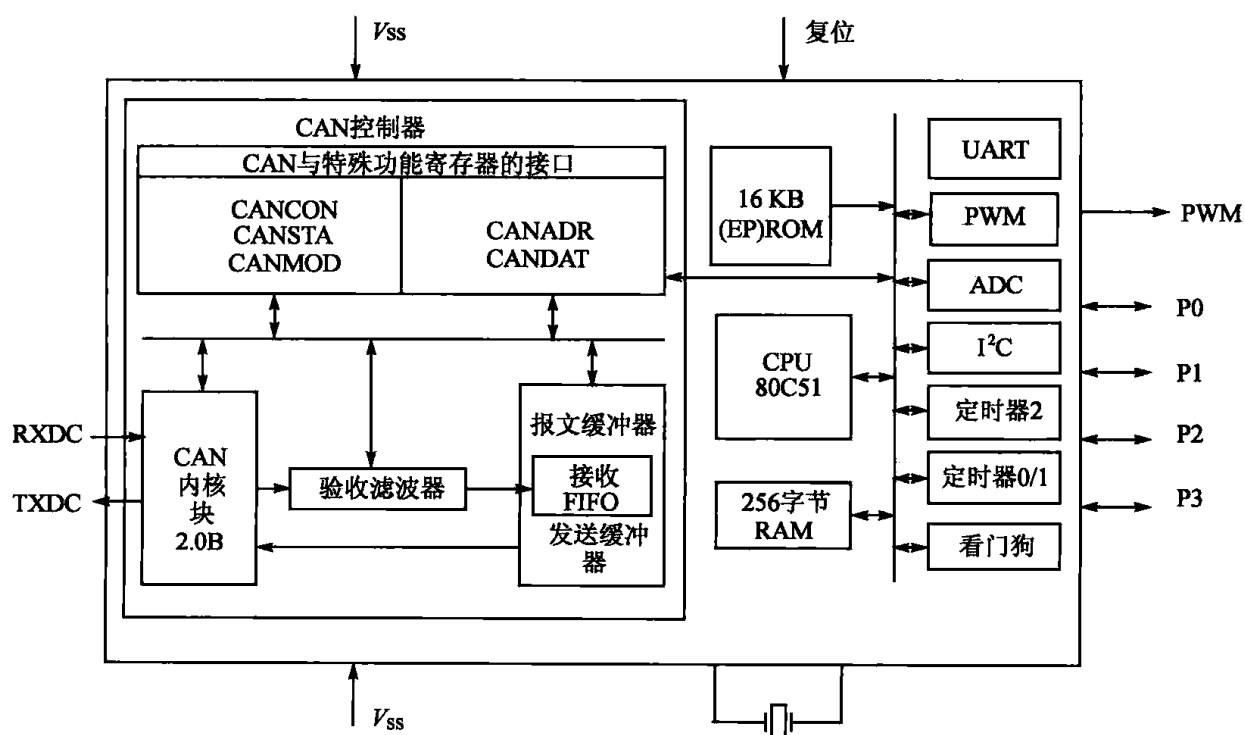


图 5-4 P8xC591 方框图

CAN 总线技术

CAN 内核根据 CAN2.0B 规范进行 CAN 帧的发送和接收。CAN 接口包含 5 个特殊功能寄存器(CPU 可以直接对其进行访问),实现 CPU 与 CAN 控制器的连接。芯片还支持自动增加的寻址特性实现对 CAN 寄存器的快速访问。CAN 控制器的发送缓冲区能够保存一个完整的 CAN 信息扩展或标准帧格式,CUP 写入命令字后即可启动发送。当接收一个信息时,只要 CAN 数据能够通过可编程验收滤波器或者缓冲区字节数目超过预设数目时即可引起接收标志位的改变。

P8xC591 还增加了新的 PeliCAN 特性:

- 4 个独立可配置的验收滤波器组;
- 每个组都有 4 个可选的验收滤波器配置;
- 所有滤波器都可在运行中改变;
- 支持更高层协议的验收滤波器;
- 只有达到 FIFO 接收中断级才产生接收中断;
- 在接收到高优先级数据帧时立即产生接收中断。

5.3.2 P8xC591 引脚描述

P8xC591 有 44 个引脚,常见封装有 PLCC 和 QFP 两种,如图 5-5 所示。主要引脚分配如表 5-9 所列。

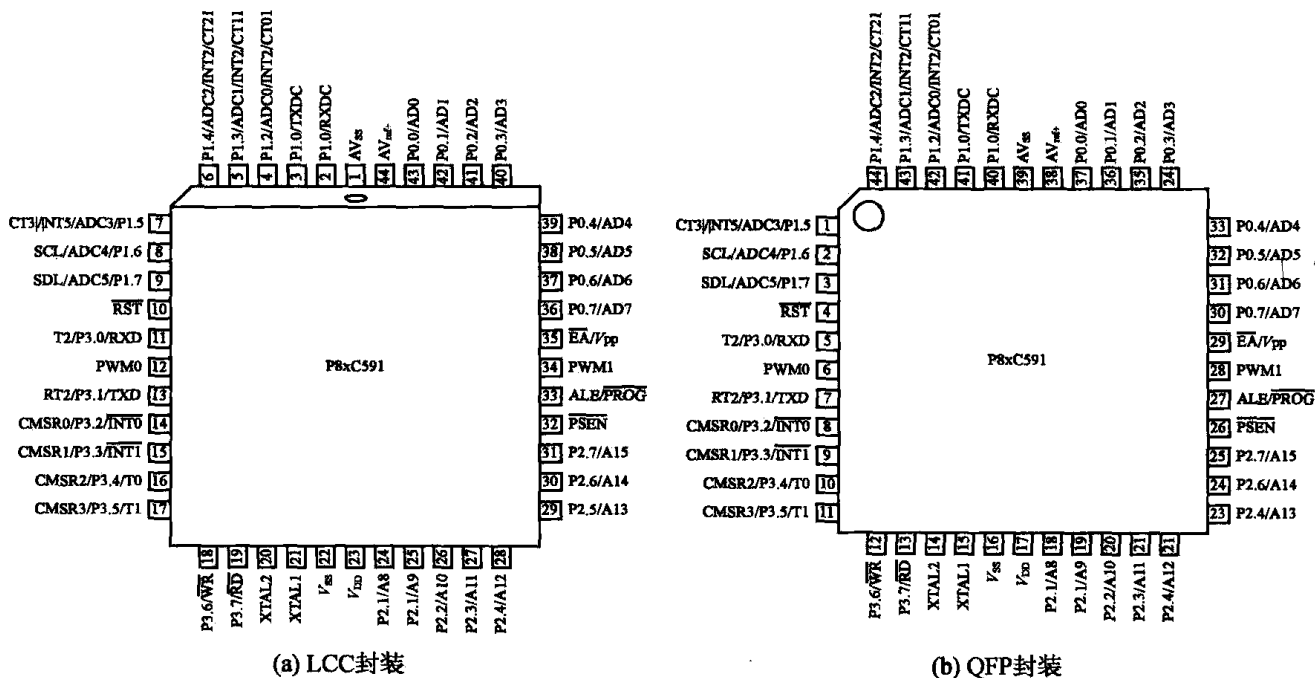


图 5-5 LCC 封装与 QFP 封装

表 5-9 引脚分配

符 号	引脚号		描 述
	QFP44	PLCC44	
RST	4	10	复位: P8xC591 复位输入。当定时器 T3 溢出时提供复位脉冲输出
P3.0~3.7			P3 口: 8 位可编程 I/O; P3 可驱动 4 个 LSTTL 输入
P3.0/RXD	5	11	RxD: 串行输入口 T2: 事件输出
P3.1/TXD	7	13	TxD: 串行输出口 RT2: T2 定时器复位信号。上升沿触发
P3.2/INT0/CMSR0	8	14	INT0: 外部中断 0 CMSR0: 定时器 T2 比较和设置/复位输出
P3.3/INT1/CMSR	19	15	INT1: 外部中断 1 CMSR1: 定时器 T2 比较和设置/复位输出
P3.4/T0/CMSR2	10	16	T0: 定时器 0 外部输入 CMSR2: 定时器 T2 比较和设置/复位输出
P3.5/T1/CMSR3	11	17	T1: 定时器 1 外部输入 CMSR3: 定时器 T2 比较和设置/复位输出
P3.6/WR	12	18	WR: 外部数据存储器写选通
P3.7/RD	13	19	RD: 外部数据存储器读选通 复位时, P3 异步驱动为高 通过 P3M1 和 P3M2 寄存器可将 P3 口设置为 4 种模式之一: P3M1.x/P3M2.x 模式描述 00 准双向(默认的标准 C51 配置) 01 推挽 10 高阻 11 开漏
XTAL2	14	20	晶振脚 2: 反相振荡放大器输出, 当使用外部振荡器时钟时开路
XTAL1	15	21	晶振脚 1: 反相振荡放大器输入和内部时钟发生电路输入。使用外部振荡器时钟时作为外部时钟信号的输入端
V _{SS}	22	16	地: 0 V 参考点
V _{CC}	44	38	电源: 提供正常、空闲和掉电工作电压

CAN 总线技术

续表 5-9

符 号	引脚号		描 述
	QFP44	PLCC44	
P2.0/A08~ P2.7/A15	8~25	24~31	P2 口:8 位可编程 I/O 口 A08~A15:外部存储器高地址,还具有以下功能: 外部存储器(A08~A15)高地址字节。还可作为 EPROM 编程和校验时的高地址 复位时,P2 异步驱动为高 通过 P2M1 和 P2M2 寄存器可将 P2 口设置为 4 种模式之一: P2M1.x/P2M2.x 模式描述 00 准双向(默认的标准 C51 配置) 01 推挽 10 高阻 11 开漏
PSEN	26	32	程序存储使能:外部程序存储器的读选通。当芯片从外部程序存储器读取程序时,PSEN 每个机器周期被激活两次。而在每次访问外部数据存储器时 PSEN 被忽略两次。对内部程序存储器访问时,PSEN 无效(保持为高)。PSEN 可驱动 8 个 LSTTL 输入。驱动 CMOS 不需要外部上拉
ALE/PROG	27	33	地址锁存使能:正常操作中,在访问外部存储器时锁存地址的低字节。每 6 个振荡器周期激活一次,但在访问外部数据存储器时例外。ALE 可驱动 8 个 LSTTL 输入。驱动 CMOS 不需要外部上拉。若想禁止 ALE 的翻转(降低 RFI 噪声),则必须通过软件置位 A0 (SFR:AUXR.0)。PROG 为编程脉冲输入
EA/V _{pp}	35	29	外部访问输入:如果复位时 EA 保持 TTL 高电平,则 CPU 执行内部程序存储器的程序。如果为 TTL 低电平,则 CPU 通过 P0 和 P2 执行外部程序存储器的程序。EA 不允许保持悬浮。在复位时锁存,复位后不用考虑 V_{pp}:提供编程电压
P0.0/AD0~ P0.7/AD7	30~37	36~43	P0 口:8 位开漏双向 I/O 口。复位时 P0 口为高阻态(三态) AD7~AD0:复用的数据和地址总线低地址。P0 口可驱动 8 个 LSTTL 输入
AVref+	38	44	A/D 转换参考电阻:高端
AVss	39	1	模拟地
P1.0~P1.4	40~44	2~6	P1 口:用户可配置输出类型的 8 位 I/O 口。P1 口作为输入或输出时的操作取决于所选择的配置。每个口都可独立配置
P1.5~P1.7	1~3	7~9	

第5章 具有CAN接口的处理器

续表 5-9

符 号	引脚号		描 述
	QFP44	PLCC44	
P1.0	40	2	RXDC: CAN 接收器输入脚
P1.1	41	3	TXDC: CAN 发送器输出脚 复位时, P1.0 和 P1.1 异步驱动为高, P1.2~P1.7 为高阻态(三态)
P1.5~P1.7	42~44	4~6	CT0I/INT2/CT1I/INT3/CT2I/INT4: T2 捕获定时器输入或外部中断输入
P1.5			ADC0~ADC2: 可选功能: ADC 输入通道
P1.6	1~3	7~9	ADC3~ADC5: ADC 输入通道
P1.7	1	7	CT3I/INT5: T2 捕获定时器输入或外部中断输入
28	2	8	SCL: I ² C 串行时钟线, 用于 I ² C 时不可使用推挽或准双向模式
6	3	9	SDA: I ² C 串行数据线, 用于 I ² C 时不可使用推挽或准双向模式。 通过 P1M1 和 P1M2 寄存器可将 P1 口设置为 4 种模式之一: P1M1. xP1M2. x 模式描述 00 准双向(默认的标准 C51 配置 ^注) 01 推挽 ^注 10 高阻 11 开漏
PWM06		12	脉宽调制: 输出 0
PWM128		34	脉宽调制: 输出

注: 该模式不适用于 P1.6 和 P1.7。

5.3.3 P8xC591 的 CAN 模块

1. PeliCAN 结构

PeliCAN 结构框图如图 5-6 所示, 共包括接口管理逻辑、发送缓冲器、接收缓冲器、验收滤波器、位流处理器、错误管理逻辑、位时序逻辑、发送管理逻辑等功能模块, 各模块介绍参见第 3 章。

2. PeliCAN 控制器与 CPU 之间的通信

PeliCAN 与 P8xC591 微控制器内部总线相连, 通过 5 个特殊功能寄存器(CANADR、CANDAT、CANMOD、CANSTA 和 CANCON)对 PeliCAN 寄存器和 RAM 区进行访问。CPU 与 CAN 的接口框图如图 5-7 所示。

CPU 支持两种方式对 PeliCAN 进行访问: 直接访问方式、间接访问方式。

(1) 直接访问

CPU 对 5 个特殊功能寄存器(CANADR、CANDAT、CANMOD、CANSTA 和 CAN-

CAN 总线技术

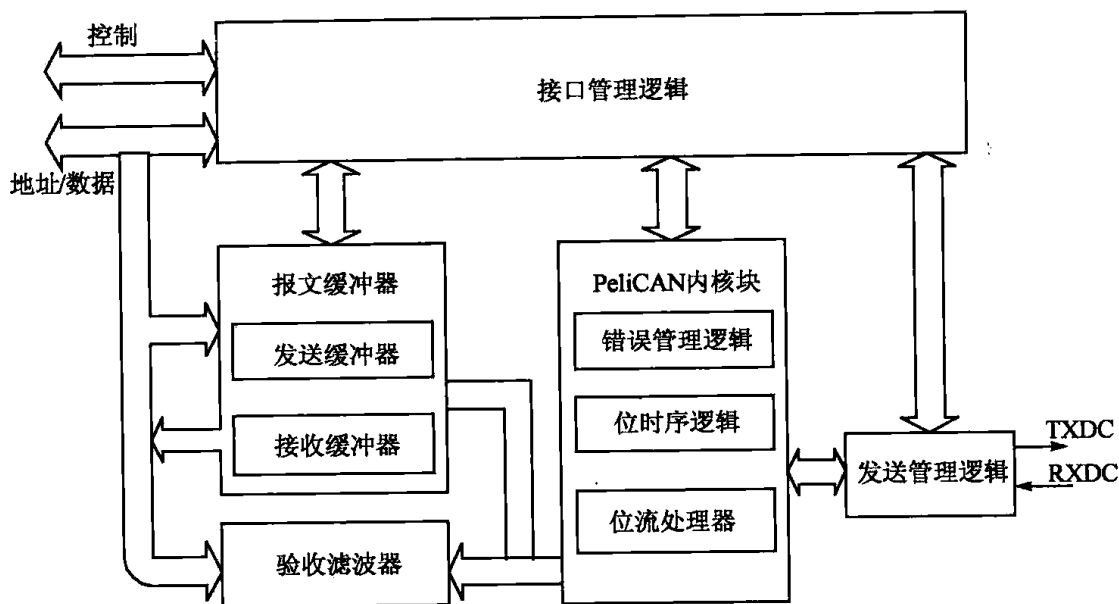


图 5-6 PeliCAN 结构框图

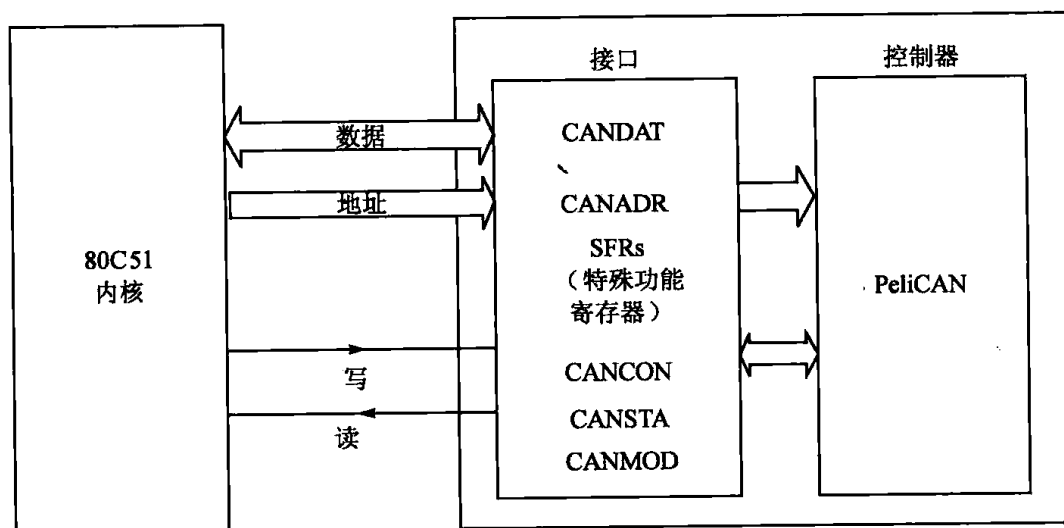


图 5-7 CPU 与 CAN 的接口

CON)直接访问。需要注意的是,CANCON 和 CANSTA 根据访问方向的不同而具有不同的寄存器功能。表 5-10 列出了特殊功能寄存器的功能和 RAM 地址。

1) CANADR

读/写寄存器,指向通过 CANDAT 访问的 PeliCAN 内部寄存器的地址;可以理解为一个指针,指向 PeliCAN 中的某个寄存器地址;对 PeliCAN 寄存器的读/写访问要通过 CANDAT 寄存器执行。

表 5-10 CAN 特殊功能寄存器

SFR(特殊功能寄存器)	访问	PELICAN寄存器	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0	SFR地址
CANADR	读/写	—	CANA7	CANA6	CANA5	CANA4	CANA3	CANA2	CANA1	CANA0	C1
CANDAT	读/写	—	CAND7	CAND6	CAND5	CAND4	CAND3	CAND2	CAND1	CAND0	C2
CANMOD	读/写	模式	TM	RIPM	RPM	SM	—	STM	LOM	RM	C4
CANSTA	读 写	状态 中断使能	BS BEIE	ES ALIE	TS EPIE	RS WUIE	TCS DOID	TBS EIE	DOS TIE	RBS RIE	C0
CANCON	读 写	中断 命令	BEI —	ALI —	EPI —	WUI SRR	DOI CDO	EI RRB	TI AT	RI TR	C3

2) CANDAT

读/写寄存器,可以理解为一个中转端口;对它的读/写就是对 CANADR 指针所指向的 PeliCAN 控制器内部寄存器的读/写。

3) CANMOD

对 PeliCAN 模式寄存器 CANMOD 是直接进行读/写访问的,模式寄存器位于 PeliCAN 模块中的地址 00H。

4) CANSTA

根据访问方向的不同,CANSTA 提供对 PeliCAN 的状态寄存器和中断使能寄存器的直接访问。对 CANSTA 的读操作是对 PeliCAN 的状态寄存器(地址 2)进行访问。对 CANSTA 的写操作是对中断使能寄存器(地址 4)进行访问。

5) CANCON

根据访问方向的不同,CANCON 提供对 PeliCAN 中断寄存器和命令寄存器的直接访问。对 CANCON 的读操作是对 PeliCAN 的中断寄存器(地址 3)进行访问,对 CANCON 的写操作是对命令寄存器(地址 1)进行访问。

(2) 间接访问

CPU 通过 CANADR 和 CANDAT 组成间接指针机制,对 PeliCAN 内部寄存器进行读/写。

[例 1] 初始化总线时序寄存器 BTR0、BTR1。

在读/写总线时序寄存器 BTR0、BTR1 时,先让 CANADR 指向 BTR0、BTR1 的 CAN 地址,然后再对 CANDAT 进行读/写,等效于对 BTR0、BTR1 的读/写。

```
#define CAN_BTR0 0x06
#define CAN_BTR0 0x07
.....
```

CAN 总线技术

```
CANADR = BTR0;
CANDAT = 0x00;
CANADR = BTR1;
CANDAT = 0x1C;
```

结果: (BTR0) = 0x00, (BTR0) = 0x1C。

[例 2] 向发送缓冲器写入数据。

```
#define CAN_TXB 0x70
.....
CANADR = CAN_TXB;
CANDAT = TranBuff[0];
CANDAT = TranBuff[1]; //地址自动加一,即 CANADR++
CANDAT = TranBuff[2]; //地址自动加一,即 CANADR++
.....
```

例 2 中只对 CANADR 操作了一次,之后每执行一次对 CANDAT 的操作,CANADR 自动加 1。规定,当 CANADR 中当前的地址大于等于 32 时,那么任何对 CANDAT 的访问将使 CANADR 自动增加。地址自动增加模式,为 CAN 控制器内部寄存器提供了快速的类栈读和写。CANADR 超过 FFH 后复位为 00H。如果 CANADR 小于 32,则不会执行自动地址增加。即使 CANDAT 执行读或写,CANADR 的值仍保持不变。这允许在 PeliCAN 控制器的低地址空间进行寄存器轮询。

5.3.4 PeliCAN 寄存器和信息缓冲区描述

1. 地址分布

PeliCAN 内部寄存器等效于 CPU 的一个片内存储器外围寄存器。由于 PeliCAN 可在不同的模式中操作(操作/复位),用户必须不同的内部寻址定义之间进行区分。从 CAN 地址 128 开始的所有内部 FIFO RAM 映射为 CPU 接口。地址分配如表 5-11 所列。

表 5-11 地址分配

地 址	工作模式		复位模式	
	读	写	读	写
0	模式寄存器	模式寄存器	模式寄存器	模式寄存器
1	00	命令寄存器	00	命令寄存器
2	状态寄存器	—	状态寄存器	—
3	中断寄存器	—	中断寄存器	—
4	中断使能	中断使能	中断使能	中断使能

续表 5-11

地 址	工作模式		复位模式	
	读	写	读	写
5	Rx 中断级	Rx 中断级	Rx 中断级	Rx 中断级
6	总线定时器 0	—	总线定时器 0	总线定时器 0
7	总线定时器 1	—	总线定时器 1	总线定时器 1
8	保留	—	—	—
9	Rx 报文计数器	—	Rx 报文计数器	—
10	Rx 缓冲区起始地址	—	Rx 缓冲区起始地址	—
11	仲裁丢失捕捉寄存器	—	仲裁丢失捕捉寄存器	—
12	错误代码捕捉寄存器	—	错误代码捕捉寄存器	—
13	错误报警上限寄存器	—	错误报警上限寄存器	—
14	RX 错误计数器	—	RX 错误计数器	RX 错误计数器
15	TX 错误计数器	—	TX 错误计数器	TX 错误计数器
16~28	保留(00)	—	保留(00)	—
29	ACF 模式	—	ACF 模式	ACF 模式
30	ACF 使能	—	ACF 使能	ACF 使能
31	ACF 优先级	—	ACF 优先级	ACF 优先级

其他地址分配如下:

20H~3FH: BANK1~BANK4 验收滤波器; 60H~6CH 接收缓冲器; 70H~7C 发送缓冲器; 80H~BFH 内部接收 FIFO。

2. 主要寄存器介绍

这部分内容和第3章大致相同, 所以这里只介绍几个区别比较大的主要寄存器, 其余请参见第3章。

(1) 模式寄存器

模式寄存器(MOD)的内容用于改变CAN控制器的状态。其中的位可由CPU置位或复位。保留位读出为0。表5-12列出了各位功能说明。

(2) Rx 中断级寄存器

Rx 中断级寄存器(RIL)用于定义RXFIFO的接收中断级。如果RXFIFO中的有效CAN信息字节的数目超过了该寄存器定义的值, 则产生一个接收中断。注意, 在完整的信息被接收之后才能产生接收中断。如果RIL设置为00, 则PeliCAN功能类似于SJA1000的接收中断状态。

CAN 总线技术

(3) 验收滤波器

PeliCAN 支持 4 个可编程的验收滤波器分组: BANK1~BANK4, 每个验收滤波器构成与 SJA1000 相似, 由验收代码寄存器 (ACRn) 和验收屏蔽寄存器 (AMRn) 构成。只要接收的报文能通过任一验收滤波器即可写入接收 FIFO; 若通过高优先级的 ACF, 则还可以直接引起接收中断。

表 5-12 模式寄存器 (MOD) 位描述说明 (CAN 地址 0)

位	符 号	名 称	功 能	复位值
MOD. 0	RM	复位模式	SJA1000 检测到 RR=1, 则中止当前的通信任务, 进入复位模式; RR=0, 回到工作模式	1
MOD. 1	LOM	只听模式	LOM=1, 只听模式, 这时即使成功接收信息, CAN 控制器也不向总线发应答信号; 错误计数器停止在当前值	0
MOD. 2	STM	自检测模式	STM=1, 进入自检测模式, 节点使用自接收命令自发自收; 即使没有应答, SJA1000 也会成功发送	0
MOD. 3	—	—	保留	0
MOD. 4	SM	睡眠模式	当 SM=1, 没有 CAN 中断等待和总线活动时, SJA1000 进入睡眠模式; SM=0, 从睡眠状态唤醒	0
MOD. 5	RPM	接收优先级模式	1: RXD 输入高有效; 0: RXD 输入低有效	0
MOD. 6	—	—	保留	0
MOD. 7	TM	—	在系统时钟的上升沿时, TXDC 引脚将映射 RXDC 脚检测到的位。RPM 位在此模式中不起作用	0

除了通常的 ACF 配置, PeliCAN 还支持通过 ACF 相关寄存器的配置来确定 ACF 的工作模式、优先级、使能。

1) 验收滤波器模式寄存器

验收滤波器模式寄存器 (ACFMOD) 用于规定 ACF 的工作模式, 如表 5-13 所列。只有在复位模式中才可对该寄存器进行写操作。

表 5-13 验收滤波器模式寄存器位说明 (CAN 地址: 29)

位	7	6	...	1	0
名 称	MFORMATB4	AMODEB4	...	MFORMATB1	AMODEB1

模式寄存器中每两位控制一组 ACF。以 BANK1 为例: ACFMOD. 0、ACFMOD. 1 位控制 BANK1 工作模式, 其中, MFORMATB1=1 表示 BANK1 用于扩展帧报文, MFORMATB1=0

时用于标准帧报文;AMODEB1=1 表示 BANK1 为单验收滤波模式,AMODEB1=0 为双验收滤波模式。

2) 验收滤波器使能寄存器(ACFEN)

每个验收滤波器由验收滤波器使能寄存器中的相应位使能或禁止,如表 5-14 所列。每两位控制 1 组 ACF。以 BANK1 为例: B1F1EN=1 表示 BANK1 的验收滤波器 1 使能,否则禁止;B1F2EN=1 表示 BANK1 的验收滤波器 2 使能,否则禁止。

表 5-14 验收滤波器使能寄存器位说明(CAN 地址: 30)

位	7	6	5	4	3	2	1	0
名 称	B4F2EN	B4F1EN	B3F2EN	B3F1EN	B2F2EN	B2F1EN	B1F2EN	B1F1EN

注:

① 如果选择单滤波器模式,则该单滤波器与对应的滤波器 1 使能位相关。滤波器 2 使能位在单滤波器模式中不起作用。

② 如果相应的滤波器被禁止,则允许在正常操作时改变验收滤波器的内容。禁止的验收滤波器不允许信息输入到接收缓冲区。如果所有的验收滤波器都被禁止(硬件复位后默认状态),则不会有信息输入到接收缓冲区。

3) 验收滤波器优先级寄存器(ACFPRIO)

每个可用的验收滤波器可定义为:

低优先级:一个报文通过验收滤波器即存入接收 FIFO 中,此时是否立即产生接收中断依靠 FIFO 中报文数目是否大于接收中断级寄存器的预设数目。

高优先级:一个报文通过高优先级验收滤波器立即引发接收中断,这允许将特定的验收滤波器用于报警信息识别并立即向主 CPU 发出中断信号。

ACFPRIO 的位分配如表 5-15 所列。

表 5-15 验收滤波器优先级寄存器位说明(CAN 地址: 31)

7	6	...	1	0
B4F2PRIO	B4F1PRIO	...	B1F2PRIO	B1F1PRIO

优先寄存器中每两位控制一组 ACF。以 BANK1 为例: ACFPRIO.0、ACFPRIO.1 位控制 BANK1 验收滤波器的优先级别,其中,B1F2PRIO=1 表示 BANK1 的验收滤波器 2 为高优先级,否则为低级;B1F1PRIO=1 表示 BANK1 的验收滤波器 1 为高优先级,否则为低级。

P8XC591 扩展的验收滤波器配置使 CAN 通信系统的配置更加灵活,用户可根据实际情况选取验收滤波器组,每组验收滤波器的内容也可以设置改变。值得一提的是,P8XC591 支持运行中改变 ACF 的配置,这样就可以在不中断节点运行的情况下改变 ACF 的配置。由于 ACF 配置的灵活性,用户在初始化时应注意对 ACF 的初始化。

CAN 总线技术

例：编写 P87C591 自检检测初始化程序。要求：

- ① 16 MHz 晶振, 500 kbps 波特率;
- ② 使用 BANK4(高优先级)和 BANK3(低优先级)单滤波方式, 不允许屏蔽滤波器;
- ③ FIFO 中有 20 个字节后就可以产生接收中断。

```
void CAN_Init(void)
{
    MODE = 0X01;           //设置方式寄存器,进入复位模式
    CDR = 0X88;            //设置时钟分频寄存器,选择 Pelican 模式关闭时钟输出(CLKOUT)
    IER = 0X01;            //设置中断允许寄存器,开放接收中断
    CANADR = RIL;
    CANDAT = 0X14;
    CANADR = ACR30;
    for(i = 0; i < 4; i++)
        CANDAT = Cpu_acr30[i];
    CANADR = ACR40;
    for(i = 0; i < 4; i++)
        CANDAT = Cpu_acr40[i];
    CANADR = ACFMOD;
    CANDAT = 0X50;
    CANADR = ACFPRIO;
    CANDAT = 0X40;
    CANADR = ACFEN;
    CANDAT = 0X50;
    BTR0 = 0X00;
    BTR1 = 0X1C;
    OCR = 0XAA;
    CANADR = IER;
    CANDAT = 0X01;
    MODE = 0X04;
}
```

注：出现的符号地址在头文件中应定义,参见第 8 章实验 3 说明部分。

5.3.5 P8xC591 典型应用

由于 P8xC591 系列单片机集成了 CAN 控制器,所以在最少数量的外部元件下工作。图 5-8 为使用 P8xC591 设计的 CAN 节点通信电路。所需要的外部电路为:复位电路、振荡器电路、收发器电路。

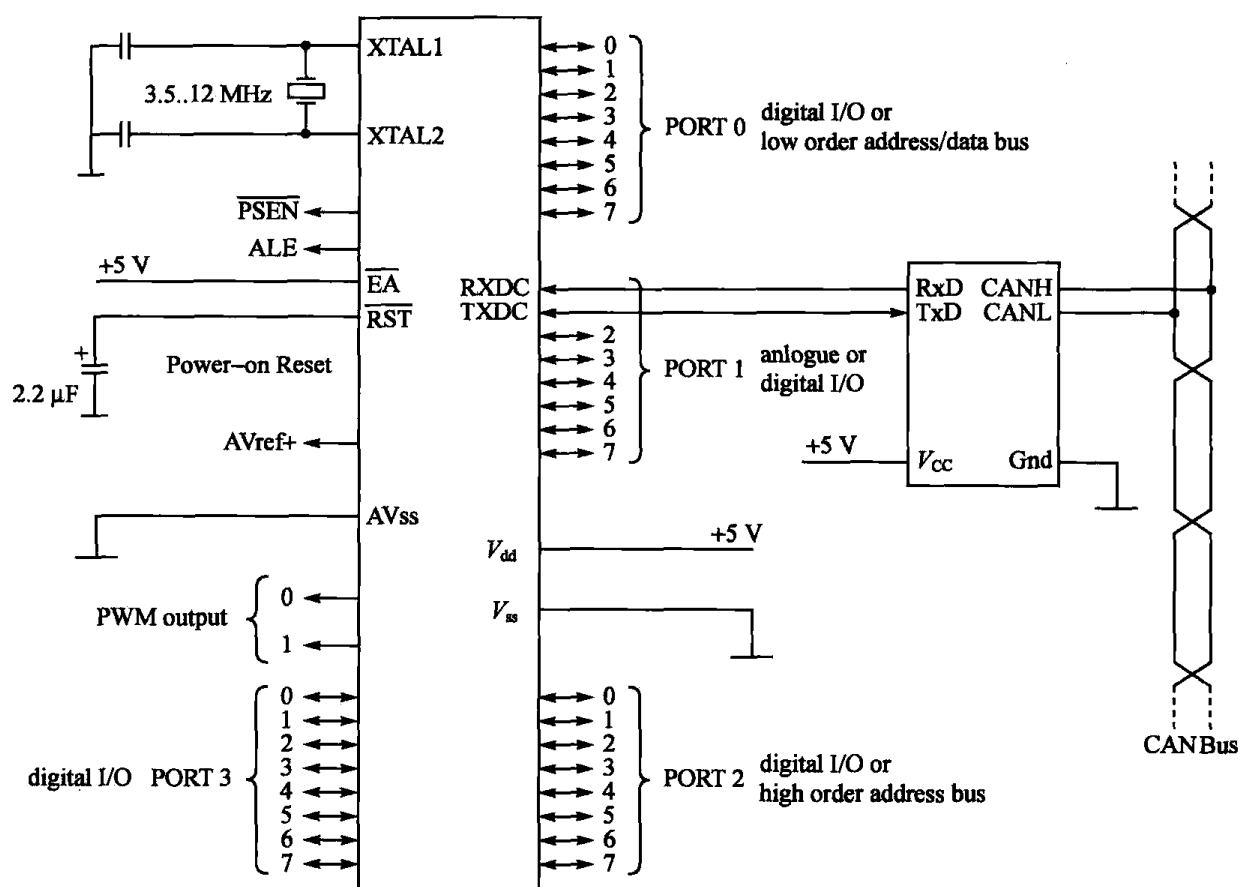


图 5-8 典型的 P8xC591 CAN 应用

图 5-8 是最基本的应用。在设计以 P8xC591 为主要器件的 CAN 节点通信电路时,为提高线路的抗干扰性,可以加入一些抗干扰措施,如光耦器件、滤波电容等。基于 P8xC591 的设计,在第 8 章实验 3 中给出了双节点通信实验,读者可以参照。

5.4 带 CAN 控制器的 ARM 微控制器

5.4.1 LPC2000 系列 ARM 微控制器

LPC2000 系列 32 位 ARM 微控制器可同时支持对多个 CAN 总线的操作,使器件可用作网关、开关或工业或汽车应用中多个 CAN 总线的路由器。微控制器集成有 2 个(LPC2119/2129/2290/2292)或 4 个(LPC2194/2294)CAN 控制器,每一个 CAN 控制器都与独立 CAN 控制器 SJA1000 寄存器结构相似,器件寄存器的访问由原来的 8 bit 字节访问转变成 32 bit 的双字访问。

CAN 总线技术

微控制器为所有的 CAN 控制器提供了全局的接收标识符查询功能。它包含一个 512×32 (2 KB) 的 RAM, 通过软件处理, 可在 RAM 中存放 1~5 个标识符表格。整个 AF RAM 可容纳 1024 个标准标识符或 512 个扩展标识符或者两种类型混合的标识符。由于允许的表格范围有 2 KB, 所以能容易地满足设计复杂的 ID 接收过滤要求。而在传统的 SJA1000 中, 接收过滤只能满足一些规律性较高的 ID 筛选过滤或个数较少的 ID (一般小于 10~15 个) 进行任意筛选过滤, 难以实现更复杂的任意 ID 进行筛选过滤, 这样会增加系统软件设计及运行时的负担。

1. CAN 特性

- 每个总线的数据波特率可达 1 Mbps;
- 可访问 32 位的寄存器和 RAM;
- 符合 CAN 规范 CAN2.0B, ISO—11898—1;
- 全局验收过滤器可识别几乎所有总线的 11 和 29 位 Rx 标识符;
- 验收过滤器为选择的标准标识符提供了 FullCAN—style 自动接收。

CAN 控制器输入/输出引脚如表 5-16 所列。

表 5-16 CAN 引脚描述

引脚名称	类 型	描 述
RX4、RX3 RX2、RX1	输 入	串行输入, 来自 CAN 收发器。其中, RX2 和 RX1 适用于所有含有 CAN 模块的器件, 但 RX4 和 RX3 只可用于 144 引脚含有 CAN 模块的某些器件
TX4、TX3 TX2、TX1	输 出	串行输出, 输出到 CAN 收发器。其中, TX2 和 TX1 适用于所有含有 CAN 模块的器件, 但 TX4 和 TX3 只可用于 144 引脚含有 CAN 模块的某些器件

2. CAN 模块的存储器映射

CAN 模块映射地址 (for LPC21xx_LPC22xx) 如表 5-17 所列。

表 5-18 列出了完整的 AF RAM 区的组成, 其中, LUT 存放 1~5 个标识符表格, 表格在接收过滤 RAM 区中是连续分布的。表格的大小和个数可以根据实际过滤需要进行裁减, 所以 AF RAM 是可裁减的。

3. 中 断

每个 CAN 控制器可产生 3 种中断请求: 接收、发送和其他状态。发送中断是 3 个 Tx 缓冲区发送中断“或”的结果。每个控制器的各个接收和发送中断请求在向量中断控制器 (VIC) 中分配有不同的通路并拥有各自的中断服务程序。所有 CAN 控制器的其他状态和验收滤波器 LUTerr 条件相“或”到一个 VIC 通道。

表 5-19 描述了根据用户手册得到的关于 CAN 中断通道分配。

第5章 具有CAN接口的处理器

表 5-17 CAN 模块的存储器映射

地址范围	用途
E003 8000~87FF	验收滤波器 RAM(2048 字节) 简称 AF RAM
E003 C000~C017	验收滤波器寄存器
E004 0000~000B	中央 CAN 寄存器
E004 4000~405F	CAN 控制器 1 寄存器
E004 8000~805F	CAN 控制器 2 寄存器
E004 C000~C05F	CAN 控制器 3 寄存器
E005 0000~005F	CAN 控制器 4 寄存器

注：① 验收滤波器 RAM 用于存放查找标识符表格
LUT 和 FullCAN(Lookup Table)模式下自动
接收 CAN 报文存储。

② 接收过滤寄存器用于控制过滤器工作模式及
LUT 格式。

表 5-18 AF RAM 组成

验收滤波器 RAM 区	地 址
FullCAN 标准标识符表格	E003 8000
独立标准标识符表格	
标准标识符范围表格	
独立扩展标识符表格	
扩展标识符范围表格	
FullCAN 自动接收存储区或 者剩余	E003 87FF

表 5-19 CAN 中断通道的描述

模 块	标志位	VIC 通道号
CAN	CAN and Acceptance Filter(1 ORed CAN, LUTerr int)	19
	CAN1 Tx	20
	CAN2 Tx	21
	CAN3 Tx(仅支持 LPC2194/2294 芯片)	22
	CAN4 Tx(仅支持 LPC2194/2294 芯片)	23
	CAN1 Rx	26
	CAN2 Rx	27
	CAN3 Tx(仅支持 LPC2194/2294 芯片)	28
	CAN4 Tx(仅支持 LPC2194/2294 芯片)	29

4. CAN 控制器寄存器

由于每个 CAN 控制器的组成、工作原理与单独的 SJA1000 相似,在此不再详述。寄存器的结构与 SJA1000 相似,只是 8 位和 32 位操作的不同,LPC2000 还把 SJA1000 中的某些 8 位 CAN 寄存器整合到一个寄存器中(比如总线时序寄存器),或者把某些位分布在不同的寄存器中,但功能基本一致,详细资料读者可以参考相关数据手册。表 5-20 列出了 CAN 模块使用的部分寄存器。

CAN 总线技术

表 5-20 CAN1、CAN2、CAN3 和 CAN4 控制器寄存器映射

名 称	描 述	访 问	CAN1 地址 & 名称	CAN2 地址 & 名称	CAN3 地址 & 名称	CAN4 地址 & 名称
CANMOD	控制 CAN 控制器的工作模式	R/W	0xE0044000 C1MOD	0xE0048004 C2MOD	0xE004C004 C3MOD	0xE0050004 C4MOD
CANCMR	影响 CAN 控制器状态的命令位	WO	0xE0044004 C1CMR	0xE0048004 C2CMR	0xE004C004 C3CMR	0xE0050004 C4CMR
CANGSR	全局控制器状态和错误计数器	RO	0xE0044008 C1GSR	0xE0048008 C2GSR	0xE004C008 C3GSR	0xE0050008 C4GSR
CANICR	中断状态, 仲裁丢失捕获, 错误码捕获	RO	0xE004400C C1ICR	0xE004800C C2ICR	0xE004C00C C3ICR	0xE005000C C4ICR
CANIER	中断使能	R/W	0xE0044010 C1IER	0xE0048010 C2IER	0xE004C010 C3IER	0xE0050010 C4IER
CANBTR	总线时序	R/W	0xE0044014 C1BTR	0xE0048014 C2BTR	0xE004C014 C3BTR	0xE0050014 C4BTR
CANEWL	出错警告界限	R/W	0xE0044018 C1EWL	0xE0048018 C2EWL	0xE004C018 C3EWL	0xE0050018 C4EWL
CANSR	状态寄存器	RO	0xE004401C C1SR	0xE004801C C2SR	0xE004C01 CC3SR	0xE005001 CC4SR
CANRFS	接收帧状态	R/W	0xE0044020 C1RFS	0xE0048020 C2RFS	0xE004C020 C3RFS	0xE0050020 C4RFS
CANRID	接收到的标识符	R/W	0xE0044024 C1RID	0xE0048024 C2RID	0xE004C024 C3RID	0xE0050024 C4RID
CANRDA	接收到的数据字节 1~4	R/W	0xE0044028 C1RDA	0xE0048028 C2RDA	0xE004C028 C3RDA	0xE0050028 C4RDA
CANRDB	接收到的数据字节 5~8	R/W	0xE004402C C1RDB	0xE004802C C2RDB	0xE004C02C C3RDB	0xE005002C C4RDB
CANTFI1	发送帧信息(1)	R/W	0xE0044030 C1TFI1	0xE0048030 C2TFI1	0xE004C030 C3TFI1	0xE0050030 C4TFI1
CANTID1	发送标识符(1)	R/W	0xE0044034 C1TID1	0xE0048034 C2TID1	0xE004C034 C3TID1	0xE0050034 C4TID1
CANTDA1	发送数据字节 1~4(1)	R/W	0xE0044038 C1TDA1	0xE0048038 C2TDA1	0xE004C038 C3TDA1	0xE0050038 C4TDA1

续表 5-20

名 称	描 述	访 问	CAN1 地址 & 名称	CAN2 地址 & 名称	CAN3 地址 & 名称	CAN4 地址 & 名称
CANTDB1	发送数据字节 5~8(1)	R/W	0xE004403C C1TDB1	0xE004803C C2TDB1	0xE004C03C C3TDB1	0xE005003C C4TDB1
CANTFI2	发送帧信息(2)	R/W	0xE0044040 C1TFI2	0xE0048040 C2TFI2	0xE004C040 C3TFI2	0xE0050040 C4TFI2
CANTID2	发送标识符(2)	R/W	0xE0044044 C1TID2	0xE0048044 C2TID2	0xE004C044 C3TID2	0xE0050044 C4TID2
CANTDA2	发送数据字节 1~4(2)	R/W	0xE0044048 C1TDA2	0xE0048048 C2TDA2	0xE004C048 C3TDA2	0xE0050048 C4TDA2
CANTDB2	发送数据字节 5~8(2)	R/W	0xE004404C C1TDB2	0xE004804C C2TDB2	0xE004C04C C3TDB2	0xE005004C C4TDB2
CANTFI3	发送帧信息(3)	R/W	0xE0044050 C1TFI3	0xE0048050 C2TFI3	0xE004C050 C3TFI3	0xE0050050 C4TFI3
CANTID3	发送标识符(3)	R/W	0xE0044054 C1TID3	0xE0048054 C2TID3	0xE004C054 C3TID3	0xE0050054 C4TID3
CANTDA3	发送数据字节 1~4(3)	R/W	0xE0044058 C1TDA3	0xE0048058 C2TDA3	0xE004C058 C3TDA3	0xE0050058 C4TDA3
CANTDB3	发送数据字节 5~8(3)	R/W	0xE004405C C1TDB3	0xE004805C C2TDB3	0xE004C05C C3TDB3	0xE005005C C4TDB3

5. 寄存器操作方法

同其他 CAN 控制器一样,对寄存器的正确读/写决定了 CAN 节点是否能正常工作,所以寄存器的操作方法应值得注意。本小节介绍 3 种访问寄存器的方法。

(1) 定义该寄存器在微处理器 VPB 空间的基地址

定义寄存器在 ARM 中 VPB 的基地址的方法清楚明了,可读性强。例:

```
#define CANMOD    0xE0044000    //定义 CAN1 模式寄存器
#define CANCMR    0xE0044004    //定义 CAN1 命令寄存器
```

(2) 用数据类型描述寄存器

可用数据结构 联合的方式定义寄存器,可以同时操作寄存器的 32 位或只操作其中的某个功能位。

例:对 CAN 模式寄存器数据类型定义。

CAN 总线技术

```
#define UINT32 unsigned int
...
typedef union _canmod_
{
    UINT32 Word;                //字操作定义
    struct{
        UINT32  RM_BIT      : 1;    //定义模式寄存器的复位控制位
        UINT32  LOM_BIT     : 1;    //定义模式寄存器的只听控制位
        ...
        UINT32  RSV_BIT24   : 24    //保留位
    } Bits;
} CANMod, * P_CANMOD;
```

(3) 统一定义 CAN 模块的功能寄存器

```
#define CANMOD_ADR      0xE0044000
#define CAN_OFFSET_ADR 0x4000    //CAN 各模块寄存器之间的差异
...
#define CANMOD(CanNum) (* ((volatile P_CANMOD)(CANMOD_BADR + CanNum * CAN_OFFSET_ADR)))
```

定义之后,可以对模式寄存器进行访问:

```
CANMOD(0).Bits. RM_BIT = 1;    //使 CAN1 模块进入复位状态
```

6. 验收滤波器

(1) 验收滤波器模块

该模块为 CAN 控制器提供了接收标识符的验收滤波功能,即类似 SJA1000 的验收滤波器;不同的是它包含一个 512×32 bit(2 KB)的 RAM(下文简称 AF RAM);通过软件编程,可在 RAM 中存放 1~5 个标识符表格。整个 RAM 可容纳 1024 个标准标识符或 512 个扩展标识符或两种类型混合的标识符,能够满足复杂的 ID 滤波要求。

LPC2000 的全局验收滤波器的工作流程与 SJA1000 是相似的:

- ① CAN 控制器接收到一个完整的标识符通知验收滤波器;
- ② 验收滤波器读出控制器编号、标识符长度(11 bit 或 29 bit),搜索 AF RAM 中的表格进行匹配,以决定接收或放弃这一帧信息。

(2) 验收滤波器寄存器

验收滤波寄存器的功能如表 5-21 所示。

通过这些寄存器的配置,可以完成验收滤波器模式配置、相关内容的读/写等,这里只对验收滤波器模式寄存器做简单的介绍。表 5-22 对模式寄存器各位做以说明。

表 5-21 CAN 验收过滤器和 CAN 集中寄存器

名 称	描 述	访 问	复位值	地 址
AFMR	验收过滤器寄存器	读/写	1	0xE003 C000
SFF_sa	标准帧单个起始地址寄存器	读/写	0	0xE003 C004
SFF_GRP_sa	标准帧组起始地址寄存器	读/写	0	0xE003 C008
EFF_sa	扩展帧起始地址寄存器	读/写	0	0xE003 C00C
EFF_GRP_sa	扩展帧组起始地址寄存器	读/写	0	0xE003 C010
ENDofTable	AF 表格终止寄存器	读/写	0	0xE003 C014
LUTerrAd	LUT 错误地址寄存器	只读	0	0xE003 C018
LUTerr	LUT 错误寄存器	只读	0	0xE003 C01C
CANTxSR	CAN 集中发送状态寄存器	只读	0x003F 3F00	0xE004 0000
CANRxSR	CAN 集中接收状态寄存器	只读	0	0xE004 0004
CANMSR	CAN 集中其他状态寄存器	只读	0	0xE004 0008

表 5-22 验收过滤器模式寄存器 (AFMR-0xE003 C000)

AFMR	名 称	值	功 能	复位值
0	AccOff	1	如果 AccBP 为 0, 则验收过滤器不工作。所有 CAN 总线上的 Rx 信息均被忽略	1
1	AccBP	1	所有 Rx 信息被使能的 CAN 控制器接收。软件必须先置位该位, 再修改以下各个寄存器的内容和查询表格 RAM 的内容(但不能置位或清零标准标识符行的禁能位)。如果该位和 AccOff 都为 0, 则验收过滤器就将接收到的 CAN 标识符屏蔽	0
2	eFCAN	0	与 SJA1000 类似, 通过软件从 CAN 控制器接收区内中读出信息(该信息的 ID 被激活)	0
		1	由验收过滤器自身来处理所选 CAN 总线上指定标准 ID 值信息的接收和保存	
31:3	保留位	—		—

(3) FullCAN 模式

LPC2000 系列支持 FullCAN 模式, 这是一种特殊的验收滤波模式。当 FullCAN 模式使能时, 验收滤波器自己在选定的总线上自动接收存储指定的标准标识符报文, 即验收滤波器以 FullCAN 控制器的身份来处理所选 CAN 总线上指定标准帧 ID 值的信息的接收和保存。

工作流程为: FullCAN 模式使能, 且 CAN 控制器收到的当前信息包含一个标准标识符, 则接验收滤波器首先查询 FullCAN 标准标识符表格。若 AF 未在 FullCAN 标准标识符表格

CAN 总线技术

中发现合适的匹配,则它接着依次查询下一个存在的标识符表格,以获取 CAN 控制器给出的标识符长度。只要发现匹配,AF 就通知 CAN 控制器保存信息,并提供一个 ID 索引值使之保存到接收帧状态寄存器(CANRFS)中;若在所有存在的标识符表格中都未能得到匹配,则 ACF 便通知 CAN 控制器丢弃/忽略接收到的信息。

自动保存的信息格式如 5-23 所列。

表 5-23 自动保存 Rx 信息的格式

地 址	31	30	29~26	25	24	23~20	19~16	15~11	10~0
0	FF	RTR	0000	SEM	0000	DLC	00000	ID	
+4	Rx Data 4					Rx Data 3		Rx Data2	Rx Data 1
+8	Rx Data 8					Rx Data 7		Rx Data 6	Rx Data 5

其中,FF、RTR 和 DLC 字段的描述与 SJA1000 相同。SEM 字用来确保访问的字来自同一信息,当信息更新时硬件将 SEM 字段设定为 01,信息更新结束后 SEM 字段被设定为 11。对信息进行访问时,软件应当清零 SEM 字段。

7. 典型应用

用 LPC2000 系列 ARM 微控制器开发 CAN,与其他带 CAN 的 8 位微控制器比较而言,硬件上是一样的,不同的是编程访问时,由 8 位操作变成 32 位操作,而且由于滤波功能的扩展,寄存器的数量和种类增多,编程相对复杂,但是流程是相似的。图 5-9 为应用时的典型连接框图。由于大多 NXP 的收发器是 3.3 V、5 V 兼容的,所以可以直接与微控制器引脚相连。这里选用了集成度高、可靠性高的收发器芯片 CTM1050。

由图可见,硬件连接简单。相对而言,软件开发比较复杂。在软件设计时,一般采用模块思想。这里介绍 CAN 的初始化模块 InitCAN 设计。

与其他 CAN 初始化类似,CAN 控制器初始化函数主要用来实现 CAN 工作时的参数设置,包括硬件使能 CAN、设置 CAN 报警界限、设置总线波特率、设置中断工作方式、设置 CAN 验收过滤器的工作方式、设置 CAN 控制器的工作模式等。流程如图 5-10 所示。

```

/***** 初始化程序 *****/
CANAFMR.Bits.AccBP_BIT = 1;          //硬件激活 CAN1 控制器
PCONP |= ((UINT32)0x01 << 13);
PINSEL1 &= ~(UINT32)0x03 << 18);
PINSEL1 |= ((UINT32)0x01 << 18);

CANMOD(0).Bits.RM_BIT = 1;           //软件复位 CAN1 控制器

CANBTR(0).Word = 0x0017C003;         //CAN1 的波特率 250Kbps

```

```

CANEWL(0).Bits.EWL_BIT = 0x60;    //报警限额为默认值 96

VICDefVectAddr = (UINT32)CANIntPrg; //初始化中断为非向量中断
VICIntEnable |= (1 << 19)|(1 << 20)|(1 << 26);
CANIER(0).Word = 0x7FF;

CANAFMR.Bits.AccBP_BIT = 1;        //配置验收滤波器(旁路状态)

CANMOD(0).Bits.TPM_BIT = 0x01;
CANMOD(0).Bits.LOM_BIT = 0x00;

CANMOD(0).Bits.RM_BIT = 0;        //启动 CAN1

```

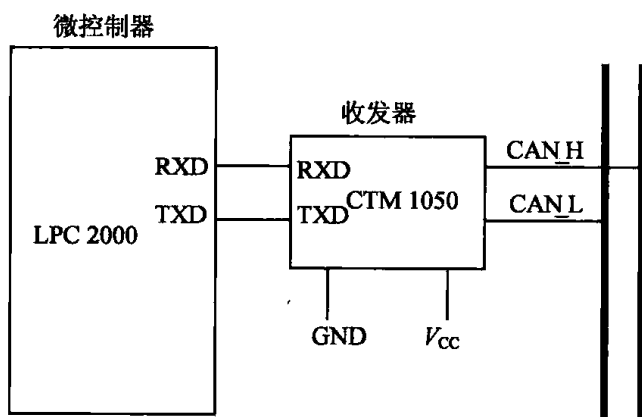


图 5-9 典型接口图

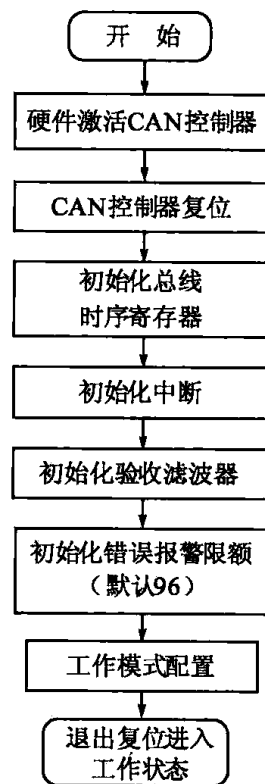


图 5-10 LPC2000CAN 初始化流程

发送、接收程序与其他控制器类似,这里不再详述。

LPC2000 系列 32 位 ARM 微控制器自带 2~4 个 CAN 控制器,开发方便,外围设备少,很适合用于网关、网桥的设计场合。

5.4.2 LPC29xx 系列 ARM 微控制器

LPC29xx 系列 ARM9 微控制器集成了 125 MHz 的 ARM968E-S CPU 核,具有全速 USB 2.0 OTG 接口、CAN 控制器、LIN 控制器、56 KB SRAM,高达 768 KB 的闪存,外部存储器接口、3 个 10 位 ADC 以及多种串行、并行接口;主要应用于消费、工业、医疗和通信领域。为了优化系统电源,LPC29xx 系列有一个非常灵活的时钟产生单元(CGU),能够提供动态时钟门限和比例。

LPC29xx 集成的 CAN 模块类似于 CAN 网络中继器(网关),由两路独立的 CAN 控制器构成,可以当作独立的 CAN 节点使用。CAN 报文信息经验收滤波器滤波后(ID 标识符查找表)可被 CAN 控制器接收。CAN 模块状态在中央状态寄存器组中描述。CAN 的结构框图如图 5-11 所示。可见,LPC29xx CAN 控制器与前文介绍过的其他类型 CAN 控制器结构类似,包含串行接口、接收缓冲器和发送缓冲器、验收滤波器等。其中,验收滤波器包扩一个 2 KB 的 RAM,存储“标识符查找表”。这里对主要的部分进行介绍。

1. CAN 发送缓冲区

CAN 控制区包含 3 个发送缓冲器,每个缓冲器的长度为 4×32 位字长,可以存储一个完整的 CAN 报文信息。CAN 控制器状态寄存器 CCSTAT 中的位 TBS1、TBS2、TBS3 反映了 3 个发送缓冲区是否被释放从而可以接收下一报文。

发送缓冲器位于 CAN 地址 030H~05CH,由报文信息、标识码和数据段组成。其中,报文信息除了包含常规帧信息外,还包含发送优先段;该段允许用户定义发送器的优先级。CAN 控制器采用了发送优先权自动保护机制。

由于发送缓冲区包含 3 个发送器,为了使报文顺利发送,CAN 控制器对发送器提供自动发送优先权检测。若模式寄存器的发送优选模式(TPM)位置位,则发送报文的优先权由用户设定 TXPRIO(发送缓冲器帧信息寄存器 0~7 位)的值决定,TXPRIO 值越小,优先级越高;若 TPM 清零,则报文的优先权由其 ID 号决定,ID 越小,优先级越高。在优先权和 ID 号都相同的情况下,报文的优先权由其所在缓冲器决定,缓冲器号小的优先级高。

标识码和数据段的结构和含义与其他控制器类似,不再赘述。

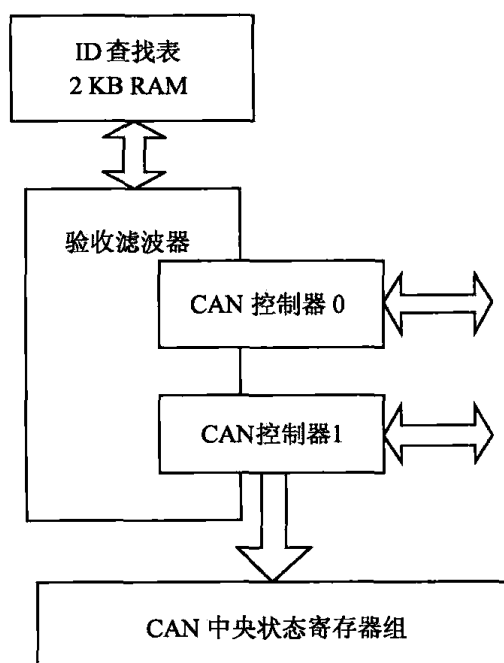


图 5-11 CAN 网关控制器结构框图

2. CAN 接收缓冲区

CAN 控制器有两个接收缓冲器,允许 CPU 读取报文的同时,下一个报文接收并存储在另一个缓冲器内。当两个缓冲器都没有剩余空间时,则 CAN 控制器产生数据溢出。在一个新的报文信息到达并通过验收滤波器之前,接收缓冲器处于锁定状态,当报文被 CPU 从缓冲器读取后,接收缓冲器释放。这些操作与以下位相关: DOS 位(全局状态寄存器)、DOI 位(中断寄存器)和 RRB 位(模式寄存器)。接收缓冲器的结构与发送缓冲器结构类似,区别如下:

(1) 标识符索引(IDI)

接收缓冲器的报文信息寄存器包含标 10 位标识符索引(IDI),标识了当前这一被接收报文标识符在标识符查找表(ID look-up table)中的位置;而用户也可通过此索引将有效的报文信息写入 CPU。

(2) 验收滤波旁路位(BP)

该位是一个状态位,反映了当前被接收的 CAN 报文所用的接收滤波器旁路模式。用户可通过设置验收滤波器模式寄存器中的 ACCBP 位使验收滤波器处于旁路状态,该状态支持运行中改变验收滤波器配置。

3. CAN 全局验收滤波器

CAN 全局验收滤波器对所有 CAN 控制器进行验收滤波——查找并验收所接收的标识符。验收滤波器包含一个标识符查找表存储区,用户可以将 1~5 个标识符段存储其间。查找表为 2 KB(512 字,32 位/字),最多可以存储 1024 个标准帧标识符或者 512 个扩展帧标识符,也可二者兼有。注意,此表只能按字访问。查找表分为 5 段,每段存储不同类型的标识符,如表 5-24 所列。

表 5-24 查找表结构

段 名	接收方式	CAN 报文格式	独立/群组 ID
标准帧格式 FullCAN 标识符段	独立存储	标准帧格式	独立
标准帧格式 独立标识符段	缓冲区存储	标准帧格式	独立
标准帧格式 群组标识符段	缓冲区存储	标准帧格式	群组
扩展帧格式 独立标识符段	缓冲区存储	扩展帧格式	独立
扩展帧格式 群组标识符段	缓冲区存储	扩展帧格式	群组

存储区中,这 5 个段的起始地址在相应的地址寄存器中设置,这些寄存器存储的地址为查找表中的偏移地址。其中,FullCAN 标识符段总是位于偏移地址的 00H,包含所以号为 0~

CAN 总线技术

(h-1)。图 5-12 给出了 CAN ID 查找表的存储结构。

h 条	11位标识符0	11 位标识符 1	标准帧格式 FullCAN 标识符 段
	11位标识符2	11 位标识符3	
	⋮	⋮	
	11位标识符(h-2)	11位标识符(h-1)	
CASFESA i 条	11 位标识符 h	11 位标识符(h+1)	标准帧格式 独立标识符段
	⋮	⋮	
	11位标识符(h+i-2)	11位标识符(h+i-1)	
CASFGSA j 条	11 位标识符(h+1)下限	11 位标识符(h+1)上限	标准帧格式 群组标识符段
	⋮	⋮	
	11位标识符(h+i+j-1)下限	11位标识符(h+i+j-1)上限	
CAEFESA k 条	29 位标识符(h+i+j)		扩展帧格式 独立标识符段
	29 位标识符(h+i+j+1)		
	⋮		
	29位标识符(h+i+j+k-1)		
CAEFGSA l 条	29 位标识符(h+i+j+k) 下限		扩展帧格式 群组标识符段
	29 位标识符(h+i+j+k) 上限		
	⋮		
	⋮		
	29 位标识符(h+i+j+k+l-1)下限		
	29 位标识符(h+i+j+k+l-1)上限		
CAEOTA	...		FullCAN 目标 报文段

图 5-12 CAN ID 查找表的存储结构

(1) 标准帧格式 FullCAN 标识符段

若 FullCAN 模式使能,则 FullCAN 标识符段使能;否则,验收滤波器忽略该段。每个 FullCAN 标识符必须按升序排列,如图 5-12 所示。因为每个 CAN 控制器有独立的寻址路径,所以表里的每个条目都包含起所属的 CAN 控制器号。FullCAN 标识符段起始于查找表的偏移地址 0 处,包含 0~h-1 条标识符,这部分的位分布如表 5-25 所列。

如果有新报文,则报文的标识符首先与 FullCAN 标识符段内的标识符比较,再与其他段内标识符比较。当发现匹配的标识符后,则目标报文被从 CAN 控制器的接收缓冲器移至相应的 FullCAN 目标报文段。表 5-26 列出该段的结构。目标报文段的首地址可以通过下式计算:

$$\text{目标报文段地址(Msg_ObjAddr + 0)} = \text{CAEOTA} + (12 \times i)$$

其中,CAEOTA 代表 CAN 验收滤波查找表地址寄存器;i 为查找表中 ID 的索引号。

表 5-25 FullCAN 标识符格式

位	描 述	说 明
31~29	SCC	偶索引;CAN 控制器标号
28	MDB	偶索引;=0,报文使能;=1,报文禁止
27	—	未用
26~16	ID[28:18]	偶索引;11 位 CAN 标识符
15~13	SCC	奇索引;CAN 控制器标号
12	MDB	奇索引;=0 报文使能;=1 报文禁止
11	—	未用
10~0	ID[28:18]	奇索引;11 位 CAN 标识符

表 5-26 FullCAN 目标报文段结构

地址	位	描 述	说 明
Msg_ObjAddr + 0	31	FF	CAN 帧格式
	30	RTR	远程请求位
	29~26	—	未用
	25~24	SEM[1:0]	状态标志位
	23	—	未用
	22~16	RXDLC[6:0]	数据长度描述
	15~11	—	未用
Msg_ObjAddr + 4	10~0	ID[28:18]	标识符
	31~24	RXDATA4[7:0]	接收数据第 4 字节
	23~16	RXDATA3[7:0]	接收数据第 3 字节
	15~8	RXDATA2[7:0]	接收数据第 2 字节
Msg_ObjAddr + 8	7~0	RXDATA1[7:0]	接收数据第 1 字节
	31~24	RXDATA8[7:0]	接收数据第 8 字节
	23~16	RXDATA7[7:0]	接收数据第 7 字节
	15~8	RXDATA6[7:0]	接收数据第 6 字节
	7~0	RXDATA5[7:0]	接收数据第 5 字节

注:

- ① SEM[1:0] = 01: 验收滤波器正在更新缓冲区;
 ② SEM[1:0] = 11: 验收滤波器已经完成缓冲区更新;
 ③ SEM[1:0] = 00: CPU 正在读缓冲区,或者未有数据更新。

CAN 总线技术

(2) 其他标识符段

查找表中其他标识符段的结构与 FullCAN 标识符段类似,读者可以参考数据手册,这里不再赘述。

4. FullCAN 模式

与 LPC2000 系列产品相似,LPC29xx 系列的产品也支持 FullCAN 模式,该模式可支持验收滤波器为两个 CAN 通道都提供滤波。FullCAN 模式适用于 BasicCAN 格式下的报文;该功能非常适合模块作为网关,在两个 CAN 通道间传送信息。

通常情况下,只要 CAN 报文信息被接收,则 BasicCAN 器件就会产生接收中断,用户用编写软件程序将该信息从 CAN 控制器读出并存储至用户的 RAM 区内。在某些应用场合,CAN 控制器需要同时处理几个 CAN 通道的报文,为了快速处理这些报文信息,CAN 进行了 FullCAN 模式的功能扩展。在这种模式下,CAN 模块可以将与 ID 查找表中标识符相匹配的报文信息自动接收并存储于 FullCAN 数据存储空间内,这样就大大简化了用户程序的编写和报文信息的处理。

5. CAN 验收滤波器匹配规则

验收滤波器的报文匹配顺序按如下顺序进行:

- ① 标准帧格式 FullCAN 标识符段;
- ② 标准帧格式独立标识符段;
- ③ 标准帧格式群组标识符段;
- ④ 扩展帧格式独立标识符段;
- ⑤ 扩展帧格式群组标识符段。

注意:① 只有被使能的标识符段参与匹配;

② 若相同的标识符出现在不同的段中,则第一次匹配成功后终止匹配。

6. 相关 CAN 寄存器

CAN 的相关寄存器如表 5-27 所列。

表 5-27 CAN 寄存器列表

偏移地址	访问方式	复位值	名称	描述
CAN 控制器: CAN0 基址 0xE008 0000; CAN1 基址 0xE008 1000				
00h	读/写	000000001H	CCMOD	CAN 控制器模式寄存器
04h	写	00000000H	CCCMD	CAN 控制器命令寄存器
08h	读/写	0000003CH	CCGS	CAN 控制器全局状态寄存器
0Ch	读	00000000H	CCIC	CAN 控制器捕获中断寄存器
10h	读/写	00000000H	CCIE	CAN 控制器中断使能寄存器

第5章 具有CAN接口的处理器

续表 5-27

偏移地址	访问方式	复位值	名称	描述
14h	读	001C0000H	CCBT	CAN 控制器总线时序寄存器
18h	读/写	00000060H	CCEWL	CAN 控制器错误警告寄存器
1Ch	读/写	003C3C3CH	CCSTAT	CAN 控制器状态寄存器
20h	读/写	00000000H	CCRXBMI	CAN 控制器接收缓冲寄存器
24h	读/写	00000000H	CCRXBID	CAN 控制器接收缓冲区标识符寄存器
28h	读/写	00000000H	CCRXBDA	CAN 控制器接收缓冲区的数据寄存器 A
2Ch	读/写	00000000H	CCRXBDB	CAN 控制器接收缓冲区的数据寄存器 B
30h	读/写	00000000H	CCTXB1MI	CAN 控制器发送缓冲 1 寄存器
34h	读/写	00000000H	CCTXB1ID	CAN 控制器接收缓冲区 1 标识符寄存器
38h	读/写	00000000H	CCTXB1DA	CAN 控制器接收缓冲区 1 的数据寄存器 A
3Ch	读/写	00000000H	CCTXB1DB	CAN 控制器接收缓冲区 1 的数据寄存器 B
40h	读/写	00000000H	CCTXB2MI	CAN 控制器发送缓冲 2 寄存器
44h	读/写	00000000H	CCTXB2ID	CAN 控制器接收缓冲区 2 标识符寄存器
48h	读/写	00000000H	CCTXB2DA	CAN 控制器接收缓冲区 2 的数据寄存器 A
4Ch	读/写	00000000H	CCTXB2DB	CAN 控制器接收缓冲区 2 的数据寄存器 B
50h	读/写	00000000H	CCTXB3MI	CAN 控制器发送缓冲 3 寄存器
54h	读/写	00000000H	CCTXB3ID	CAN 控制器接收缓冲区 3 标识符寄存器
58h	读/写	00000000H	CCTXB3DA	CAN 控制器接收缓冲区 3 的数据寄存器 A
5Ch	读/写	00000000H	CCTXB3DB	CAN 控制器接收缓冲区 3 的数据寄存器 B
CAN ID 查找表存储空间: 基址 0xE008 5000(CANAFM)				
000h to 7FCh	读/写	00000000H	CAFMEM	CAN ID 查找表存储器
CAN 验收滤波器: 基址 0xE008 7000(CAMODE)				
00h	读/写	00000001H	CAMODE	CAN 接收滤波器模式寄存器
04h	读/写	00000000H	CASFESA	CAN 验收滤波器标准帧开始地址寄存器
08h	读/写	00000000H	CASFGSA	CAN 验收滤波器标准帧组开始地址寄存器
0Ch	读/写	00000000H	CAEFESA	CAN 验收滤波器扩展帧开始地址寄存器
10h	读/写	00000000H	CAEFGSA	CAN 验收滤波器扩展帧组开始地址寄存器
14h	读/写	00000000H	CAEOTA	CAN 验收滤波器结束表地址寄存器
18h	读/写	00000000H	CALUTEA	CAN 验收滤波器查找表错误地址寄存器
1Ch	读/写	00000000H	CALUTE	CAN 验收滤波器查找表错误寄存器

CAN 总线技术

续表 5-27

偏移地址	访问方式	复位值	名 称	描 述
20h	读/写	00000000H	FCANIE	全局 FullCAN 接收寄存器
24h	读/写	00000000H	FCANIC0	FullCAN 中断和捕获寄存器 0
28h	读/写	00000000H	FCANIC1	FullCAN 中断和捕获寄存器 1
CAN 中央状态寄存器: 基址 0xE008 8000(CCCTS)				
0h	读/写	00030303H	CCCTS	CAN 控制器中央发送状态寄存器
4h	读/写	00000003H	CCCRS	CAN 控制器中央接收状态寄存器
8h	读/写	00000000H	CCCMS	CAN 控制器中央混合状态寄存器

各寄存器的具体使用可以参照相关数据手册。

LPC29xx 集成的 CAN 模块可以独立作为 CAN 节点使用;也可以方便地作为 CAN 网络中继器使用,连接两路 CAN 网络,开发简单,性价比较高。

第 6 章

CAN 的应用层协议

CAN 的协议最初只规定了物理层和数据链路层,这使得 CAN 协议能适应不同的应用对象。但是应用层协议在实际工程应用中是必需的,开发者可以根据工程需要,自己规定应用层的有关内容。由于没有统一的规范,这种应用层协议与设计者本身的习惯及思路有很大关系。协议内容不严谨、不统一会使开发人员在每个新项目中都要进行应用层协议设计,重复劳动大,产品维护问题也多。

随着 CAN 总线的推广应用,同时保证产品的互操作性,需要制定统一的应用层协议。目前,在不同的行业应用中已经形成了几种应用层协议,如 CANopen、DeviceNet、CAN Kingdom、SAE J1939 等,国内制定了 iCAN 协议。

CAN 应用层协议主要规定标识符的分配方案、过程数据交换方法、通信的实现方法等内容。本章先介绍简单的自定义应用层协议,它适用于小规模的应用系统;然后介绍 CANopen 和 DeviceNet 协议。

6.1 简单的自定义应用层协议

标识符是 CAN 的重要概念,可以说,CAN 的数据链路层协议的核心就是围绕标识符制定的。在应用层协议中,标识符的分配处理是基本内容之一。原则上,CAN 是按照数据分配标识符,而不像通常的按照物理节点分配地址。但实际上,节点和数据往往有密切的关系,技术人员也习惯于网络节点的思考方式,这可以使数据管理更加明晰。因此,“CAN 取消站地址编码”的特点在应用中并没有得到发扬光大。数据的标识符和报文验收滤波机制还有直接关系。

对于一个节点数量小、数据种类也不大的比较简单的系统,如果使用 CAN 作为通信网络,则可以自己定义一个简单的应用层协议;在这个协议中,可根据数据(或节点)的重要程度分配标识符,通信的实现方法可参考常用的 RS485 协议。

虽然自定义的应用层协议不规范,兼容性也很差,但对简单系统而言,不必使用复杂的上

CAN 总线技术

层协议,只需要定义信息帧的格式,应用 CAN 的多主站发送模式即可。这样的应用层协议更能发挥结构简单,易于实现,成本低廉的优点。

看一个简单系统的例子:基于 CAN 的纸浆绝干量测控系统。绝干量是指纸浆中含有的植物纤维量的多少。

纸张抄造的简单工艺过程如图 6-1 所示。叩打好的纸浆(中浓度)存放在浆池中。浆池中的浆料由浆泵泵出,与白水混合进行稀释后被泵至流浆箱;流浆箱的液位是通过控制浆泵的转速来保持恒定的,进而保持浆网速比的恒定。当正常生产时,网速恒定,浆网速比约为 1.0。因此,冲浆泵的转速保持恒定,即通过冲浆泵的纸浆流量恒定。通过定量阀控制浆的流量,通过定量扫描检测系统(图中的扫描部)配合可以控制纸的绝干量。但此系统的结构比较复杂,而且绝干量的测控过程具有很大的滞后,控制算法复杂,系统造价较高。对于较小规模的生产线,可以通过测量上网纸浆的浓度和流量计算出绝干量。应用表明,当控制精度要求不是非常高的时候,此系统能有效地控制纸张的绝干量(定量),控制效果比人工控制有很大改进。

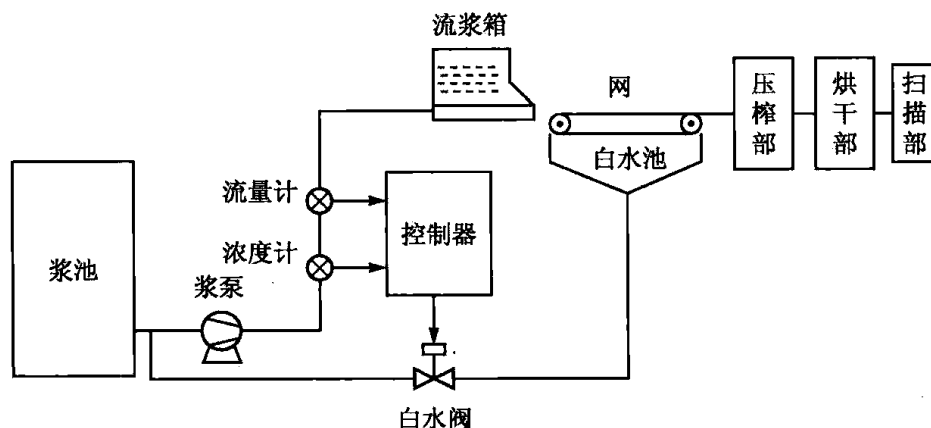


图 6-1 抄纸工艺及绝干量控制系统原理

在这个控制系统中,系统主要控制量是纸浆浓度,通过一只浓度计检测上网纸浆浓度;浓度的调节是靠加入白水的量来改变的,白水的加入量是本系统中唯一要进行控制的物理量。控制器计算、输出的控制信号控制白水管道上的电动调节阀,以调节白水的加入量。另外,在上浆管道再安装一台流量计,计量通过的纸浆流量。如果浓度控制能使纸浆浓度保持恒定,则按照流量及网速就能计算出纸张的绝干量。

对这个系统,自定义一个应用层协议,可以考虑以下几个方面:标识符的分配、报文帧格式、通信实现方法。下面分别介绍。

6.1.1 标识符的分配

在 CAN 协议中,报文的标识符决定了它的优先权,并且通过验收滤波器的设置决定通信的方式。也可以对标识符做出约定,以表示某种含义(如节点地址、自定义数据类型等)。应用

层协议的首要工作是根据系统的特点来分配标志符,在分配时要注意:

- ① 唯一性:系统中不同的报文具有不同的标识符。
- ② 标识符的高 7 位不能全部为 1,这是由位定时、帧结构决定的。

因为上面介绍的纸张绝干量控制系统较小,所以使用具有 11 位标识符的 BasicCAN 模式。

系统中,主要节点有控制器、流量计、浓度计、白水阀,另外还有几台浆泵、白水泵。总的节点数不超过 16 个。

系统中,有这样几类数据:

- 模拟量数据:包括模拟量输入/输出数据,模拟量输入数据有纸浆流量检测值、纸浆浓度检测值、白水阀开度检测值,模拟量输出数据是白水阀开度控制值。
- 数字量数据:包括数字量输入/输出数据,数字量输入主要是泵的开关状态,数字量输出主要是泵的启停控制。
- 设备参数:检测仪表的零度、满度调整值、工作状态信息等。

在上述数据中,设备参数关系到设备能否正常、准确地运行,所以它的优先级应该是最高的。模拟量数据是系统主要处理的数据,相对数字量数据,优先级应设为次高;数字量数据优先级最低。可以使用标识符的最高 2 位来表示数据类型,如图 6-2 所示。

假设节点数最大按 16 计算,采用面向节点的标识符配置方法时,节点地址需要使用 4 位标识符,如图 6-2 所示。

剩余的标识符还有 5 位,分配给具体的报文编号,这是足够的。

如果考虑到以后系统扩展,可以将此分配方案适当调整一下,比如节点地址用 5 位标识符,系统可扩展到 32 个节点。在此例中按 16 个节点设计,11 位标识符分成 3 段,如图 6-2 所示。

高位ID10~ID9	ID8~ID5	ID4~ID0低位
数据类型(2位)	节点地址(4位)	报文编号(5位)

图 6-2 11 位标识符的分配

图中,ID10 和 ID9 表示数据类型,若

- ID10、ID9 取值 00,表示设备参数;
- ID10、ID9 取值 01,表示模拟量数据;
- ID10、ID9 取值 10,表示数字量数据;
- ID10、ID9 取值 11,暂时不用,保留。

ID8~ID5 分配给各个节点作为节点地址编码。

ID4~ID0 是各种报文的分类编号,如设备参数中可以定义以下报文:

- ID4~ID0 取值 00000,表示设备故障;

CAN 总线技术

ID4~ID0 取值 00111,表示零点校正值;

ID4~ID0 取值 01011,表示满度校正值;

ID4~ID0 取值 01111,表示上限报警值;

ID4~ID0 取值 10011,表示下限报警值;

ID4~ID0 取值 10111,表示量程选择。

模拟量数据中,可以定义以下报文:

ID4~ID0 取值 00011,表示 8 位模拟量输入值;

ID4~ID0 取值 00111,表示 10 位模拟量输入值;

ID4~ID0 取值 01011,表示 16 位模拟量输入值;

ID4~ID0 取值 01111,表示 8 位模拟量输出值;

ID4~ID0 取值 10011,表示 10 位模拟量输出值;

ID4~ID0 取值 10111,表示 16 位模拟量输出值。

数字量数据比较简单,数值直接在标识符中表示,报文中不再携带数据字节,即数据长度码 DLC=0,可以按以下定义:

ID4~ID0 取值 00011,表示数字量输入 0;

ID4~ID0 取值 00111,表示数字量输入 1;

ID4~ID0 取值 01011,表示数字量输出 0;

ID4~ID0 取值 01111,表示数字量输出 1。

6.1.2 报文帧格式

应用层的报文格式基本上套用数据链路层所定义的数据帧、远程帧格式。根据标识符的分配方案,理解和使用各种报文。应用层帧格式如图 6-3 所示。

仲裁段				控制段	数据段
数据类型	节点地址	报文分类	RTR	数据长度	0~8 字节数据

图 6-3 应用层帧格式

可见,此格式与数据链路层的数据帧格式基本是一致的,只是对标识符的使用做了进一步的规范。用户程序根据数据的功能和类别,按照这个应用层帧格式把数据组装起来,准备发送。

同样,节点接收到报文后,按照应用层帧格式把数据拆解,正确理解信息功能,使用数据。对接收到的报文,先判断它的远程请求位 RTR,若 RTR=0,则是数据帧,然后判断此报文的具体意义并做处理;若 RTR=1,则是远程帧,根据报文提供的数据类型、报文分类等信息组织发送该报文。

6.1.3 通信实现方法

(1) 生产/消费方式

按照 CAN 的基本协议, CAN 网络是一个多主站网络, 任何一个节点都可以发起通信行为, 成为主站; 如果多个节点同时发送数据, 则按照仲裁规则, 确认优先权最高的节点成为主站, 这种通信方式是典型的生产/消费方式。生产/消费方式中, 各个节点间的联系是不固定的, 发送节点和接收节点直接面向网络开放, 发送者只需要把信息提供给网络, 并不关心哪些节点使用这些信息; 接收者同样向网络提出所需求的信息, 而这些信息可能来自多个提供者。CAN 进行生产/消费方式的通信, 不需要额外的协议规定, 只需要通过报文的标识符识别和理解数据, 效率很高。

检测节点(如浓度计、流量计)定时将检测到的纸浆浓度值以规定格式发送到网络上。控制器根据检测值及生产要求将计算后得出的控制信息按照控制周期的设定发送到网络。调节阀随时接收控制信息, 根据控制信息的指示调整阀门的开度, 并把阀门开度值反馈到网络。

(2) 远程请求方式

远程请求方式, 在第 2 章的远程帧中有解释, 这里不再赘述。

(3) 点对点通信方式

在系统调试和维护中, 操作员可能会对某些节点参数进行查阅和修改, 这些信息对系统通信至关重要。为保证操作的正确性, 最好采用点对点通信方式, 操作站作为主节点, 发起通信行为, 被访问的从节点必须做出应答, 如对从节点报文滤波器的读/写操作。

在标识符分配方案中, 数据类型 00 表示设备参数。这些参数在正常工作时很少修改, 可以考虑把这类数据作为点对点通信的特殊类。把报文滤波器的读/写操作等报文编号加入这类即可; 也可以把保留的数据类型 11 使用起来, 作为点对点通信类。

点对点通信时, 主节点通过发送数据帧、远程帧来和从节点进行通信。系统通信中, 每次传送 8 个字节数据, 基本能满足所有数据类型的要求。如果有超过 8 字节长度的数据, 就需要分为多次传送。分为多帧传送的数据块要有附加的协议信息, 以表示数据块的长度和各个字节的顺序。可以把第一帧中数据段的第一个字节定义成帧数和序号。例如, 高 4 位表示帧数, 低 4 位表示帧的序号。如果一个字节不够用, 就把第二个字节也用上。一般尽量不要传送很长的数据块; 在普通过程控制中也很少有较长的数据要传送。

6.2 CANopen 协议

6.2.1 CANopen 概述

6.1 节中的简单应用层协议基本上只规定了标识符的分配方案, 能实现的功能也很简单,

CAN 总线技术

对于开发小系统是合适的。规模较大的系统涉及网络管理功能,有更复杂的数据信息交换行为,需要定义更复杂、功能更强的应用层协议。对于设备制造厂商而言,要保证设备的互操作性和互换性,就必须基于这样一个标准化的应用层协议。

CANopen 是关于 CAN 的一项应用层协议,在机械制造、铁路、车辆、船舶、制药、食品加工等领域获得大量应用。CANopen 协议是建立在 CAL 协议(CAN Application Layer)基础上的一个子协议,是一个 CIA(CAN In Automation)标准,在欧洲被认为是基于 CAN 的工业系统中占领导地位的标准。

1. CAL 的核心内容

CAL 定义了 4 类应用层服务:

1) 通信规范 CMS(CAN-based Message Specification)

规定如何通过 CAN 接口访问节点的功能。

2) 网络管理 NMT(Network Management)

提供基于主从网络的网络管理服务,网络中只能有一个主节点。

3) 标识符分配 DBT(Distributor)

提供动态分配标识符的功能。

4) 层管理 LMT(Layer Management)

提供改变通信参数的功能。例如,与通信速率相关的位定时设置、节点地址等。

CAL 的协议标准以 11 位标识符的 CAN 标准帧为定义对象。CAL 报文分为 8 个优先级,每个优先级中分配 220 个标识符。另外,部分标识符分配给 NMT、DBT 及 LMT 使用。如果使用 29 位标识符的 CAN 扩展帧,则只规范 29 位标识符的高 11 位;每个优先级的标志符数量增多,就不仅仅是 220 个了。注意,每个优先级中的标识符,优先权是各不相同的,数值越小,优先权越高。报文标识符的分配如表 6-1 所列。

表 6-1 CAL 报文标志符的分配

标识符数值	功 能	标识符数值	功 能
0	NMT 启动、停止服务	1101~1320	CMS 对象,优先级 5
1~220	CMS 对象,优先级 0	1321~1540	CMS 对象,优先级 6
221~440	CMS 对象,优先级 1	1541~1760	CMS 对象,优先级 7
441~660	CMS 对象,优先级 2	1761~2015	NMT 节点监控
661~880	CMS 对象,优先级 3	2016~2031	NMT、DBT、LMT 服务
881~1100	CMS 对象,优先级 4	—	—

CANopen 设备可分为通信部分、对象字典和应用程序 3 个部分。通信部分提供协议规定的对象交换服务。对象字典是一个数据库,描述了设备使用的数据类型、通信对象和应用对

象。应用程序是用户编写的程序,通过操作对象字典来实现通信。对象字典是联系实际网络和应用实现的接口,是 CANopen 的重要概念。

2. CANopen 的标识符分配

CANopen 协议在 CAL 的基础上对 CMS 对象的具体内容进行了规定,支持设备参数直接访问以及实时性要求很强的过程数据交换。CANopen 针对分布式控制系统的特点,定义了通信对象、服务和协议。CANopen 定义了 4 种消息帧,分别是管理消息 NMT、过程数据对象 PDO(Press Data Object)、服务数据对象 SDO(Serve Data Object)、预定义消息及特殊功能对象。

CANopen 协议中规定了一套默认的标识符分配方案,称为预定义连接,用户可以直接使用。如果用户另有分配方法,可以对分配方案进行动态修改。

默认方案中,定义了 1 个紧急对象、4 个发送 PDO 及 4 个接收 PDO、1 个 SDO 和 1 个错误控制消息,其优先级由高到低。此方案还支持 NMT 控制服务、同步和时间戳。

11 位标识符分成两个部分,高 4 位 ID10~ID7 称为功能码,用来区分各种不同的消息帧;剩余的 7 位标识符 ID6~ID0 作为节点的编号。表 6-2 列出了默认的标识符分配方案。

表 6-2 默认的标识符分配方案

广播对象			
对 象	功能码	标识符数值	对象字典索引
NMT 控制服务	0000	0H	—
同步	0001	80H	1005H~1007H
时间戳	0010	100H	1012H、1013H
对等对象			
紧急事件	0001	81H~0FFH	1015H、1024H
PDO1(发送)	0011	181H~1FFH	1800H
PDO1(接收)	0100	201H~27FH	1400H
PDO1(发送)	0101	281H~2FFH	1801H
PDO1(接收)	0110	301H~37FH	1401H
PDO1(发送)	0111	381H~3FFH	1802H
PDO1(接收)	1000	401H~47FH	1402H
PDO1(发送)	1001	481H~4FFH	1803H
PDO1(接收)	1010	501H~57FH	1403H
SDO(发送/服务器)	1011	581H~5FFH	1200H
SDO(接收/客户端)	1100	601H~67FH	1200H
NMT 错误控制	1110	701H~7FFH	1016H、1017H

6.2.2 CANopen 通信模型

1. 对象字典

CANopen 标准最核心的部分是通过对象字典(Object Dictionary)对设备功能进行描述的。对象字典分为两部分,第一部分包括基本的设备信息,如设备 ID、制造商、通信参数等。第二部分描述了特殊的设备功能。大多数重要的设备类型,如数字和模拟的输入/输出模块、驱动设备、操作设备、控制器、可编程控制器等,其功能都在一个设备规范文件中描述。依靠 CANopen 协议的支持,可以对不同厂商的设备通过总线进行配置。

对象字典中的对象通过一个 16 位的索引来寻址,对于复杂的数据对象(如数组、结构)还有一个 8 位的子索引,用来寻址其内部成员。每一个索引指向了设备的一种属性,这些属性叫应用对象。通过对象字典的入口可以对设备的应用对象进行基本网络访问,设备的应用对象可以是输入/输出信号、设备参数、设备功能和网络变量等。对象字典的结构如表 6-3 所列。详细的结构参见附录 B。

表 6-3 对象字典的结构

索 引	对 象	索 引	对 象
0000H	保留	00A0H~0FFFH	保留
0001H~001FH	标准数据类型	1000H~1FFFH	通信对象
0020H~003FH	复杂数据类型	2000H~5FFFH	设备制造商信息
0040H~005FH	制造商定义的复杂数据类型	6000H~9FFFH	标准化设备规范
0060H~007FH	设备规范定义的标准数据类型	A000H~BFFFH	接口规范说明
0080H~009FH	设备规范定义的复杂数据类型	C000H~FFFFH	保留

每一个设备都有上述形式的对象字典,它是关联用户的应用程序和通信接口的纽带。用户通过对象字典可以访问设备的所有数据和配置参数。

对象字典的每个条目一般包括 6 个部分:索引、对象、名称、属性、类型、强制/可选。这里的对象是字典规定的符号名,名称是对对象的文字描述。对象类型有 7 种,如表 6-4 所列。属性定义了对对象的读/写性质,对象可以读/写(rw)、只读(ro)、只写(wo)、常量(const,只读)。强制对象必须实现,可选对象可以不实现。

对象字典的 1000H~1FFFH 部分是对通信对象的描述,通信对象的描述使用了索引、名称、对象代码、数据类型、强制/可选这几项。对象中的每个条目使用了子索引、名称、属性、类型、强制/可选、有无 PDO 映射、取值范围、默认值等几项参数描述,详细情况请参阅有关标准或手册。

CANopen 设备的功能及特性以电子数据文件 EDS(Electronic Data Sheet)的形式描述;

EDS 采用 ASCII 格式,由设备制造商提供,利用配置工具对节点进行配置。实际的设备设置通过设备配置文件(DCF)进行描述。EDS 和 DCF 可以从互联网上下载,并可以存储在设备之中。

表 6-4 对象字典的对象类型

对 象	说 明	对象代码
NULL	没有数据段的条目	0
DOMAIN	可执行的代码	2
DEFTYPE	定义的类型,如 Boolean、float	5
DEFSTRUCT	定义的结构	6
VAR	标准数据类型的变量	7
ARRY	数组,子索引 0 不是数组数据部分	8
RECORD	记录,子索引 0 不是记录数据部分	9

2. 管理消息

管理消息实现网络管理的功能,包括节点监护和 NMT 消息。节点监护是主节点周期性地向从节点发送远程请求报文,从节点响应此消息并返回表示节点状态的报文。节点监护的周期可以通过 SDO 进行设置。在这个周期内,从节点如果没有接收到主节点的查询,则产生一个生命监护事件;如果主节点没有接收到从节点的响应或从节点状态异常,则主节点产生一个节点监护事件。

上电后设备先进入初始化阶段,初始化正常后,发出 Boot-up 消息,表示设备准备好。接下来进入预处理阶段;在这个阶段,只有 SDO 可以通信。在接到启动命令后,设备进入工作状态;在这个阶段 PDO 可以进行通信,SDO 通信仍然有效。

(1) Boot-up 消息

正常的设备在上电完成基本初始化任务后,会向 NMT 主节点发送一个 Boot-up 消息。在这之后,就可以启动通信,启动步骤描述如下:

① 设备初始化完成,自动发送 Boot-up 消息。初始化包括硬件初始化、设备描述参数及通信参数初始化为上电值。

② 设备准备就绪,可以执行 NMT 服务、SDO 或特殊功能服务。

③ 启动节点通信,可以执行 NMT 服务、SDO、PDO 或特殊功能服务。

④ 节点通信结束,执行 NMT 服务,可以转入就绪状态或重启节点通信。

Boot-up 消息的格式如图 6-4 所示。Boot-up 消息的数据字节总是 0。

(2) Heartbeat 协议

Heartbeat 消息是从节点发送给 NMT 主节点的一种消息,用来表示从节点当前的工作状

CAN 总线技术

态。正常工作时,从节点总是按照一定的周期发送 Heartbeat 消息。Heartbeat 消息的格式和 Boot-up 消息是一样的,不同的是数据字节的状态值,状态值为 4,表示节点被停止;状态值为 5,表示节点正常运行;状态值为 127,表示节点处于准备就绪阶段。节点的心跳周期可以设置,设 0 时,禁止 Heartbeat 协议;设不为 0 的数值时,则 Heartbeat 协议开始执行。Boot-up 消息也可以看作一个特殊的 Heartbeat 消息,状态值是 0。

(3) NMT 消息

网络管理对象包括节点监护和 NMT 对象。网络管理的模式是主从式结构的,网络中只有一个 NMT 主节点,其余都是从节点。主节点通过 NMT 模式控制消息,按照协议来管理、监视从节点的工作,包括从节点的初始化、启动、停止、复位等状态的转换以及节点工作状态的监视。

NMT 消息包含 2 字节数据,标识符为 0,格式如图 6-5 所示。

标识符数值	数据字节
700H+节点地址	0

图 6-4 Boot-up 消息的格式

标识符数值	数据字节0	数据字节1
0H	命令标志	目标节点地址

图 6-5 NMT 模式控制消息的格式

其中,命令标志可以取以下值:

- 1 表示启动远程节点(从节点)
- 2 表示停止远程节点
- 128 表示进入就绪状态
- 129 表示复位远程节点
- 130 表示重启通信

NMT 模式控制消息只能由 NMT 主节点发送,所以从节点都监听。从节点的地址范围是 1~127。

3. 过程数据对象

过程数据对象 PDO 一般是传送实时数据的,这种数据比较短小,要求传送速度较高。PDO 分为发送 PDO 和接收 PDO,发送 PDO 是指某节点主动发送给其他节点的过程数据,通过直接发送数据帧来实现。接收 PDO 是指某节点需要其他节点提供的过程数据,通过发送远程请求来实现。

在对象字典中,每个 PDO 用两个对象来描述。一个是通信参数,规定标识符、传输类型、时间周期和禁用时间等内容;另一个是映射参数,说明 PDO 中的对象在对象字典中的位置和长度。

过程数据对象通常采用事件或定时触发、同步传输或远程请求方式发送;作为广播对象,它可以被多个节点接收。一个 PDO 就是一个 CAN 数据帧,上层并没有附加协议,也不需要

应答。

事件或定时触发、远程请求方式容易理解。同步传输方式是根据一个周期性的同步消息来传送 PDO。在网络中有一个固定节点(一般是主节点)发出同步信号。按照对同步信号的使用方式,同步传输方式又分为周期和事件触发两种方式。周期发送方式中,节点对同步信号进行计数,达到预定值(1~240)后就发送一个 PDO。事件触发方式是发生了指定的事件后,在接下来的一个同步信号时发送 PDO。表 6-5 列出了 PDO 的传输类型和传输条件。

表 6-5 PDO 的传输

传输类型	传输条件			说 明
	收到同步信号	收到 RTR 信息	事 件	
0	B	—	B	同步,事件触发
1~240	O	—	—	同步,周期
241~251	—	—	—	保留
252	B	B	—	同步,收到远程请求后发送
253	—	O	—	收到远程请求后发送
254	—	O	O	制造商指定的事件
255	—	O	O	设备规范指定的事件

注: B 表示条件必须同时满足, O 表示只需要一个条件满足。

发送节点按照对象字典的索引规定把标识符和数据组成帧发送出去。接收节点收到 PDO 后,按照 PDO 的标识符在对象字典中查找对应的索引,理解对象的内容。

4. 服务数据对象 SDO

SDO 提供客户端访问服务器的对象字典的功能。通过传输 SDO 可以实现可靠的数据传输,由两个 CAN 对象在两个网络节点间通过点对点的通信来实现这一过程。传输对象字典的索引以及子索引,可以定位相应的对象字典入口。SDO 传输的特点是客户端采取主动;可以传输任意长度的数据块,但需要附加协议;每个 SDO 请求必须要有应答。服务数据对象主要用于在设备配置过程中传输参数以及传输大数据块。

当传输的数据多于 8 字节时,必须使用多个帧组合成一个数据块,这时就需要用块传输协议。在第 1 帧中,要附加必要的控制信息,比如在第 1 字节中定义一个标志位,以防重复接收。接下来的 3 个字节中,给出访问对象的索引和子索引。随后的每帧第 1 个字节,都用作控制协议,标明帧的顺序。接收方对每个帧或整个数据块作出应答。

SDO 的通信参数描述在对象字典的 22H 处,其结构和 PDO 类似。

SDO 提供以下几种服务:

➤ SDO 块下载。SDO 客户端利用块下载协议,从 SDO 服务器下载数据。

CAN 总线技术

- SDO 块上传。SDO 客户端利用块上传协议,把数据上传到 SDO 服务器。
- 结束块 SDO 上传。在最后发送的一个帧中,说明总共发送的有效字节数目。
- 中止 SDO 传送。此服务可以由客户端或服务器在任何时候行使,停止当前的下载或上传。这项服务不需要应答。

5. 预定义消息及特殊功能对象

CANopen 定义了 3 种特殊功能对象:同步对象(SYNC)、时间标记(Time Stamp)、紧急对象(Emergency)。

同步对象。在前面有关 PDO 传输的内容中提到了同步信号,即同步对象。同步信号由 SYNC 主节点周期性产生,为保证周期的准确,同步消息往往使用最高优先级组的标识符,标识符是 28。同时,为了提高传送速度,此消息没有数据字节。

时间标记给网络中的所有设备提供统一的时间信息,标识符是 256。时间标记采用 Time-of-Day 的数据类型,占用 6 字节。

紧急对象是由设备内部错误触发的。CANopen 规定了几种错误代码,表示不同类型的错误。每个错误事件只产生一次紧急对象。紧急对象包含 8 字节数据,有 2 字节错误代码,1 字节错误寄存器,其余是制造商关于错误的信息。对紧急对象接受后的处理,CANopen 没有规定。

6.3 DeviceNet

6.3.1 DeviceNet 概述

DeviceNet 是一种基于 CAN 的低成本现场总线。它将简单工业设备(如开关、光电传感器、阀门、电动机、条形码读取器)和复杂设备(如智能仪表、变频器、控制器)连接成网络。DeviceNet 是一种简单的网络解决方案,规定的设备互操作性和互换性减少了配线和安装工业自动化设备的成本和时间。

DeviceNet 是一个开放的网络标准。它的应用行业包括汽车,半导体芯片制造,电子产品制造,食品和饮料生产,装配、包装和物流等。DeviceNet 在中国已有许多应用,而且随着技术的进一步完善和推广,DeviceNet 有很好的应用前景。

DeviceNet 网络最多允许 64 个节点,网络通信采用生产/消费者模型。DeviceNet 只支持 125 kbps、250 kbps、500 kbps 波特率,但并不要求 3 种波特率都支持。DeviceNet 的通信距离最大 500 m,在最高速 500 kbps 时,通信距离不超过 100 m。由此可见,DeviceNet 的系统规模相对较小。DeviceNet 支持总线供电,使设备连接更加简洁。DeviceNet 节点要求 CAN 的收发器部分从总线上获取电源,以保证连接到总线的 CAN 收发器不会因未加电而影响到总线的数据传送。

DeviceNet 是基于连接的网络,节点在通信前要先建立逻辑链路的连接。节点在连接过程中会生成一个连接标识 CID,随后的通信中,每次都要带上这个标识,以表示逻辑信道的连接关系。DeviceNet 定义了两类连接:I/O 连接和显式连接。I/O 连接主要用于传送实时性要求较高的现场数据信息。显式连接主要用于设备间组态、控制命令等信息的传递。显式连接是一对一的;I/O 连接可以一对一,也可以一对多。

CANopen 网络模型包括物理层、数据链路层和应用层,下面仅介绍应用层。

6.3.2 DeviceNet 报文组

1. 报文组

根据 DeviceNet 的连接方式,有两种形式的报文:I/O 报文和显式报文。

I/O 报文也称为隐性报文,用于在 DeviceNet 网络中传输应用和过程数据。相关的 I/O 数据从一个生产者传输到多个消费者。I/O 报文通常使用高优先级的报文标识符,连接标识符提供了 I/O 报文的相关信息。

I/O 报文的数据域一般不包含协议信息,完全用来传输过程数据。CAN 报文标识符表示过程数据的优先级和重要性。每个 I/O 报文使用 1 个 CAN 标识符。

显式报文用于 DeviceNet 网络中两个设备之间的一般性数据交换,如上载和下载程序、设备参数配置、故障诊断等,通常使用低优先级的报文标识符。显式报文请求指明了对象、实例、属性以及所要调用的特定分类服务,并由报文路由对象传递到相应的对象。

显式报文由标识符和附带的协议信息组成。CAN 标识符用作连接 ID,而不用于显式报文传输协议,所有协议都包含在 CAN 数据场当中。设备之间的每个显式连接通道需要 2 个 CAN 标识符,一个用于请求报文,另一个用于响应报文。标识符在连接建立时确定。

设备必须对显式报文做出解释并响应。

DeviceNet 对 11 位标识符进行了定义,把信息报文分成 4 组,即信息组 1~信息组 4。信息报文的格式如表 6-6 所列。

表 6-6 DeviceNet 的信息报文格式

标识符各位的含义											范 围	用 途
10	9	8	7	6	5	4	3	2	1	0		
0	报文组 1 标识				源 MAC ID 标识符						000~3FF	报文组 1
1	0	MAC ID 标识符						报文组 2 标识			400~5FF	报文组 2
1	1	报文组 3 标识			源 MAC ID 标识符						600~7BF	报文组 3
1	1	1	1	1	报文组 4 标识						7C0~7EF	报文组 4
1	1	1	1	1	1	1	x	x	x	x	7F0~7FF	无效标识

CAN 总线技术

11 位 CAN 标识符被划分为 4 个报文组,包括报文标识和设备标识(MAC ID)。报文标识表示某报文组中的一个报文。源 MAC ID 表示报文的发送者。源和目标地址都可作为 MAC ID。系统中报文的含义由报文 ID 确定。

由表 6-5 可见,报文组 1~报文组 4 的优先级依次降低。

报文组 1 分配了 1024 个 CAN 标识符,占有所有可用标识符的一半。该组中报文标识占 4 位标识符,即每个设备最多可拥有 16 个不同的报文,能建立 16 个优先级的报文系统。报文组 1 通常用于 I/O 报文交换应用数据。

报文组 2 分配了 512 个标识符。该组的 MAC ID 既可以是源 MAC ID,也可以是目的 MAC ID。这种方式适用于主从形式的网络通信,主节点发送报文时填写目的 MAC ID(从节点 MAC ID),从节点发送报文时填写源 MAC ID(自身 MAC ID)。

DeviceNet 利用报文组 2 和部分报文组 1 定义了一套预定义的主从连接标识符。其中,1 个报文 ID 定义为网络管理,以简化主从结构数据传递,避免使用复杂的动态连接配置。

报文组 3 分配了 448 个标识符,结构与报文组 1 相似;不同的是,它的优先级低。该组的主要用途是建立动态的显式连接。每个设备可有 7 个不同的报文,其中,2 个报文保留作为连接报文管理器端口(UCMMPort)。

报文组 4 分配了 48 个 CAN 标识符;标识符中不包含任何设备地址,只有报文 ID。该组的报文只用于网络管理。通常分配 4 个报文 ID 用于离线连接集。

2. I/O 报文

I/O 报文在发送前要设置好逻辑连接。通过标识符的分配,指出端点地址以及对象属性。I/O 报文和标准的数据帧格式是一样的。当报文长度大于 8 字节时,要使用分段协议,表示一个报文的多个数据帧。

分段协议占用数据场的第一个字节,由分段类型、分段计数器两部分构成,如图 6-6 所示。

7	6	5	4	3	2	1	0
分段类型		分段计数器					

图 6-6 分段协议格式

分段类型占用第一个字节的高两位,用来说明该帧数据是首段、中间段还是最后一段。分段类型取值为 0,说明是第一个分段,分段计数值必须是 0 或 3F。如果分段计数器的值为 0,那么这是分段系列中的第一段。如果分段计数器的值是 3F,那么这就是传输系列中的最后一个发送。当在 I/O 连接中建立一个大的连接长度、但当前只有少量的数据被发送时,接收器必须被告知这既是第一分段也是最后分段。分段类型取值为 1,表明这是一个中间分段。分段类型取值为 2,表明这是最后一个分段。分段类型取值为 3,是分段应答,用于确认分段的

接收。

分段计数器用来标志每一个单独的分段,每经过一个相邻连续分段,分段计数器加1,当计数器值达到64时,又从0值开始。

3. 显式报文

显式报文利用 CAN 数据帧的形式传递 DeviceNet 定义的信息。显式报文包括信息头和信息体两部分,如果报文长度大于8字节时,也要使用分段协议。图6-7和图6-8分别是单帧和多帧格式的显式报文。

字节/位	7	6	5	4	3	2	1	0
0	Frag	XID	MAC ID					
1	RR	服务代码						
2~7	服务变量							

图6-7 单帧格式的显式报文

字节/位	7	6	5	4	3	2	1	0
0	Frag	XID	MAC ID					
1	分段类型		分段计数器					
2	R/R	服务代码						
3~7	服务变量							

图6-8 多帧格式的显式报文

Frag 指示当前的报文是否为显式信息的一个分段,Frag=0,说明这个报文包含完整的显式信息,下一字节包含服务区。Frag=1,说明这个报文不包含完整的显式信息,下一字节包含分段协议。

XID 位用来判断报文请求和报文应答的一致性。

MAC ID 是与报文 ID 对应的设备 ID,如果报文 ID 中的 MAC ID 是源 ID,则此处的 MAC ID 是目的 ID;反之,则是源 ID。

R/R 位决定了这个信息是请求信息还是响应信息,R/R=0,是请求信息;R/R=1,是响应信息。

服务区字节低7位叫服务代码,表示传送服务的类型。

6.3.3 对象模型

DeviceNet 协议使用面向对象的方法来描述,将一个 DeviceNet 节点模拟为一个对象的集

CAN 总线技术

合,由此定义了节点外部的特性表现,而内部的所有参数全部由类—实例—属性的关系来表现。

对象就是对真实实体的一种抽象。在 DeviceNet 中,可以把一个对象看成一个封装的功能块,它具备对象的一切特点,如包含了数据结构和提供相关的行为等。

实例是指对象的一个特定的、具体的表现,如狗是哺乳动物分类对象中的一个实例。

属性是指对象的外部可见特征或特性的描述。属性提供了一个对象的状态信息及管理对象的工作。

例示是指建立一个对象的实例。对象定义中可以规定使用默认值,否则该对象所有实例属性都初始化为零。

类是指表现出相同类型系统成分的对象集合,对象实例是指在分类内某一特定对象的具体表示。类是一个对象的概括,某类内的所有对象在形式及行为上是相同的,但可能具有不同的属性值。一个分类的每个实例不但有一组相同的属性,也有自身的一组特定的属性值,在一个 DeviceNet 节点的一个特定分类中可以存在多种对象实例。

一个对象实例或对象分类都有自己的属性,都能提供服务完成一种特性。属性是一个对象或对象分类的特性。服务用来触发对象/分类实现一个任务对象。行为表示它如何响应特定的事件。例如,某个人可以抽象为在人这个分类中的一个实例,一般来说所有人都有一组相同的属性,如年龄、性别等,然而因为每一个属性的属性值不同,每一个人的具体表现都不同。

DeviceNet 的对象分为预定义对象和自定义对象。预定义对象描述 DeviceNet 节点必须具备的特性和服务,内容由 DeviceNet 规范定义;自定义对象则描述设备的特定功能,内容由生产商定义。

DeviceNet 定义了很多对象,而大部分的对象都是可以根据自己所开发的设备来选择的,但作为一个 DeviceNet 节点,协议中规定必需存在如下几个对象:

① 标识对象(Identity Object):DeviceNet 产品一般都有一个标识对象实例。此实例包含各种属性,如供货商 ID、设备类型、产品代码、版本、状态、序列号、产品名称和说明。

② 消息路由对象(Message Router Object):报文路由对象向其他对象传送显式报文。一般在 DeviceNet 网络中它不具有外部可视性。

③ DeviceNet 对象(DeviceNet Object):该对象实例具有的属性有节点地址或 MAC ID、波特率、总线关闭、总线关闭计数器、单元选择和主机的 MAC ID。

④ 连接对象(Connection Object):DeviceNet 产品一般至少包括两个连接对象。每个连接对象代表 DeviceNet 网络上两节点间虚拟连接中的一个端点。两种连接类型分别称为显式报文连接和 I/O 报文连接。显式报文包括属性地址、属性值和服务代码,以此描述所请求的行为。I/O 报文只包含数据。I/O 报文中,所有有关如何处理数据的信息都包含在与该 I/O 报文相关的连接对象中。

除此之外,还有组合对象、参数对象、应用对象等。

6.3.4 预定义主/从连接

DeviceNet 定义了非连接报文管理器(UCMM),可以建立动态显式报文连接,接收/发送和管理网络上的非连接显式报文。UCMM 提供两类服务,即建立、打开显式报文连接及关闭显式连接。设备间建立连接的一般步骤是先建立非连接显式报文,再建立显式报文连接,最后设置建立 I/O 连接。这种连接方法很灵活,可以动态修改,但是设置过程比较繁琐,同时需要设备具备灵活配置的能力。在实际应用中,大多数应用对象比较简单,可以主/从连接方式。DeviceNet 定义了预定义主/从连接报文组和仅使用报文组 2 的从站,以简化设备配置过程并降低成本。

预定义主/从连接报文组预先定义好了一些通信功能。主站通过主/从连接报文组,建立主/从通信关系,配置报文传输机制(位选通、轮询、显示等)。

表 6-7 说明了预定义主/从连接报文组的标识符分配情况。

表 6-7 预定义主/从连接报文组的标识符分配

标识符											报文类型
10	9	8	7	6	5	4	3	2	1	0	
0	报文组 1				源 MAC ID						报文组 1
0	1	1	0	0	源 MAC ID						从站 I/O 多点轮询响应报文
0	1	1	0	1	源 MAC ID						从站 I/O 状态改变/循环报文
0	1	1	1	0	源 MAC ID						从站 I/O 位选通响应报文
0	1	1	1	1	源 MAC ID						从站 I/O 轮询响应/状态改变/循环应答报文
1	0	MAC ID						报文组 2			报文组 2
1	0	源 MAC ID						0	0	0	主站 I/O 位选通命令报文
1	0	多点通信 MAC ID						0	0	1	主站 I/O 多点轮询命令报文
1	0	目标 MAC ID						0	1	0	主站状态改变/循环应答报文
1	0	源 MAC ID						0	1	1	从站显式/非连接响应报文
1	0	目标 MAC ID						1	0	0	主站显式响应/请求报文
1	0	目标 MAC ID						1	0	1	主站 I/O 轮询/状态改变/循环应答报文
1	0	目标 MAC ID						1	1	0	仅限组 2 的非连接显式响应/请求报文
1	0	目标 MAC ID						1	1	1	重复 MAC ID 检查报文

位选通命令是主站发送的具有多点发送功能的 I/O 报文。多个从站可以同时接收一个位选通命令,然后向主站发送位选通响应报文。在从站中,位选通命令和响应报文由同一个连接对象接收和发送。

CAN 总线技术

轮询命令是主站发送给某个特定从站的命令。主站向每个要查询的从站发送不同的查询命令。从站接收到轮询命令后,给主站发送一个轮询响应报文。在从站中,轮询命令和响应报文由同一个连接对象接收和发送。

主站和从站都可以发送状态改变/循环报文。这是一种点对点连接,接收和发送由同一个连接对象完成。

显式请求报文常用于读/写属性的操作。显式响应报文表示显式请求报文的服务结果。在从站中,显式请求和响应报文由同一个连接对象接收和发送。

非连接显式报文由 UCMM 管理。不具备 UCMM 功能的从站,称为仅限组 2 从站。这种从站不能接收非连接显式报文,只能通过仅限组 2 的非连接显式响应/请求报文实现预定义主/从连接的建立和删除。

仅限组 2 的非连接显式请求报文是预留的命令,用来分配/释放预定义主/从连接组。仅限组 2 的非连接显式响应报文是对仅限组 2 的非连接显式请求报文和发送设备监测脉冲/设备关闭报文。

重复 MAC ID 检查报文用来检测同一连接中是否存在具有相同 MAC ID 的模块。重复 MAC ID 检查报文的格式如图 6-9 所示。R/R 是请求/响应标志位。物理端口编号是不同的物理连接的识别号。

字节/位	7	6	5	4	3	2	1	0
0	R/R	物理端口编号						
1~2	制造商 ID							
3~6	序列号							

图 6-9 重复 MAC ID 检查报文的格式

第 7 章

基于 CAN 总线的监控系统设计

CAN 总线是一种有效支持分布式控制或实时控制的串行通信网络。由于 CAN 卓越的特性及极高的可靠性,所以非常适合过程监控设备互连。CAN 总线连线简单,可扩展性强,具有很高的性价比。CAN2.0B 协议只包括数据链路层和物理层,用户可自定义应用层协议,具有很大的灵活性。本章介绍了一种基于 CAN 的火灾报警系统设计。

7.1 系统设计概述

楼宇自动化是当今自动化技术应用的一个重要方面。随着智能技术的不断发展,对楼宇火灾报警控制系统的要求也越来越高,火灾报警系统要完成各节点的报警、联动等功能;还要具有开放性,能够把报警信息传送到其他网络或系统。本设计要求完成一项基于 CAN 总线分布式的智能火灾报警系统设计,系统由监控上位机、中继器、转换节点、分布式测量节点等部分组成,可以用于宾馆、商场、校园、家庭等场合进行火灾安全监控;并且各个模块具有较强的独立性,可以用于其他系统,具有工作可靠、报警及时、造价低廉等特点。系统组成机构框图如图 7-1 所示。

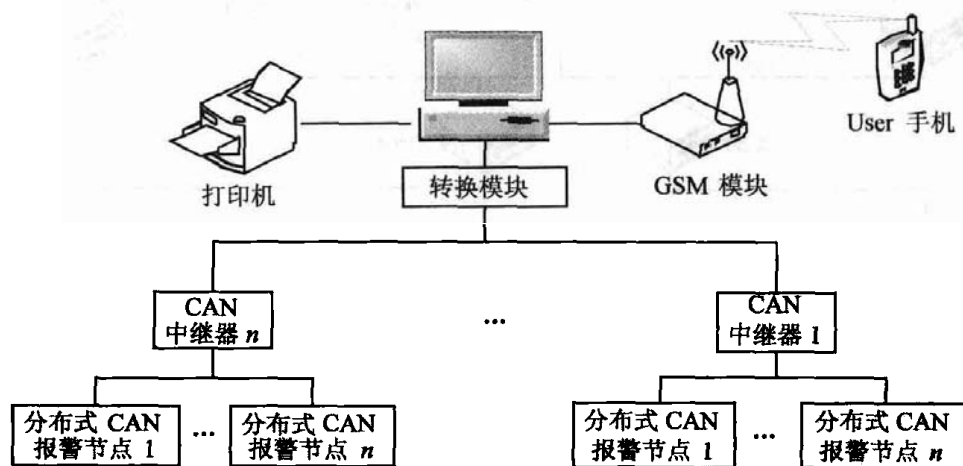


图 7-1 分布式火灾报警系统结构框图

CAN 总线技术

上位机用于系统监控,定期收集各分布式节点信息。当检测到异常时,产生报警提示,下发命令,并将报警信息通过 GSM 模块发至用户手机告知用户。同时,上位机应将数据存储在数据库,为事故分析作准备。

转换模块负责将 CAN 信号转换成上位机可识别接收的信号,可以根据情况选取,这里选择 CAN/RS485 设计。中继器是连接相同网络的物理转接模块,可以延伸 CAN 的通信距离,增加节点数目。各节点报警模块定期采集数据(温度、湿度、气体浓度等)给上位机,若发生异常也可以进行单独报警。

7.2 系统网络拓扑结构及参数配置

7.2.1 系统网络拓扑结构

以校园实验楼为应用对象,图 7-2 给出了监控系统的网络拓扑结构图。各个监控报警节点分布在不同楼层的重点监控区域内,如果需要扩充,可添加二级中继器进行节点扩展,整个

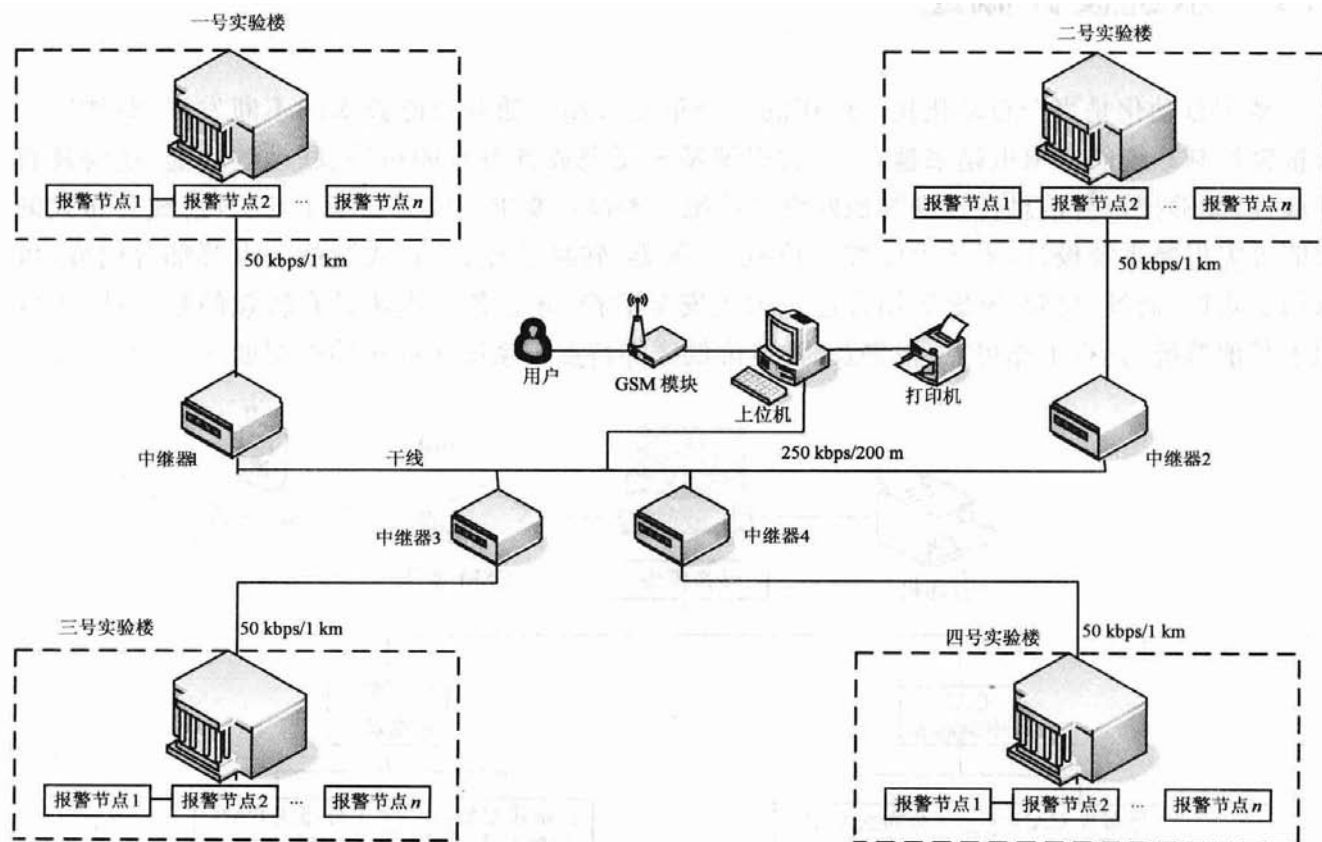


图 7-2 实验楼火灾报警系统网络拓扑结构图

系统呈树形结构。设计时,干线上通信速率采用 250 kbps,提高系统的响应速度;支线距离较远、信息量较少,可采用低速率长距离,通信速率为 60 kbps。

7.2.2 系统网络参数配置

出于对节点数量、通信速率、系统扩展的综合考虑,系统设计可考虑在中继器与上位机的主干线上使用 CAN2.0B 中的扩展帧,扩展帧支持 29 位标识符,系统将合理运用分配。分配如表 7-1 所列。

表 7-1 干线参数配置

ID. 28~ID. 21	ID. 20~ID. 18	ID. 17~ID. 12	ID. 11~ID. 6	ID. 5~ID. 0
命令/状态字	上位机 (可扩展)	一级中继器 (可扩展)	二级中继器 (预留)	监控报警节点

可以看出,ID. 28~ID. 21 可用于通信中区别状态、命令数据;ID. 20~ID. 18 用于上位机(即 CAN 网桥模块标识),系统可扩展 8 台上位机;ID. 17~ID. 12 用于一级中继器;ID. 11~ID. 6 为系统扩展二级中继器而预留;ID. 5~ID. 0 为各个现场报警节点,每个实验楼可安装 64 个报警节点。

各报警节点分布于实验楼不同区域,线路较长,所以节点与中继器之间的通信速率相对较低,此时采用短帧结构——标准帧结构,可以提高实时性。标准帧标识码分配如表 7-2 所列。

表 7-2 支线参数配置

ID. 10~ID. 9	ID. 8~ID. 3	ID. 2~ID. 0+保留位
预留	监控报警节点标识	屏蔽以做它用

可见,ID. 10~ID. 9 预留以备系统扩展需要;ID. 8~ID. 3 为该子网中各节点标识;ID. 2~ID. 0 和保留位可以被屏蔽,在具体应用中有不同的配置。

标识码的配置要依靠验收滤波器的正确配置,所以设计时要注意验收代码寄存器、验收屏蔽寄存器的合理搭配,使之能满足系统要求。

7.2.3 系统通信协议

1. 参数功能

① 上位机实时监控现场,处理异常情况:向 CAN 网络发送控制信息;发送异常时,通过 GSM 网络向区域负责人发出警告信息。

② 上位机在 CAN 总线上能实现差错控制、数据重传、单点呼叫等功能。

CAN 总线技术

③ 转换模块与上位机通信采用 32 位 CRC 校验。

④ CAN 总线采用 CAN2.0B 协议兼容 BASIC CAN 协议,接口面向 CAN 总线的参数,波特率可为 500 kbps,实现全双工多点通信。

⑤ 各中继器将所收数据过滤后转发给各子网的报警模块;每周期将模块上传数据发送给转换模块。

⑥ 各报警节点模块每周期采集现场信息上传至 CAN 总线;实时监听上位机所发命令并执行报警,紧急时触发联动装置(如喷头、机械手等)。

⑦ 数据发送均采用查询方式,数据的接收均采用中断方式。

2. 协议制定

(1) 上位机下发的命令帧格式/转换模块上传数据帧

上位机下发的命令帧格式/转换模块上传数据帧如表 7-3 所列。

表 7-3 上位机下发命令帧格式

帧 头	1 号模块命令		...	校 验	帧 尾
1B	3B	8B	...	8B	1B
0xFB	模块地址	下发命令/上传数据	...	CRC	0xFC

上位机下发命令采用不定长格式,即检测到异常模块进行命令发布。转换模块上传数据采用固定长度格式,即包括每个报警节点信息。

- 模块地址:一级中继器地址+二级中继器地址+监控节点地址;
- 命令:报警命令、触发联动装置命令;
- 校验:采用 32 位 CRC 校验。

(2) 各监控节点接收/发送帧格式

根据设计要求,本设计采用两种帧格式:数据帧格式、命令帧格式。数据帧是报警节点,用于传输节点状态信息。各模块节点实时采集温度、湿度、气体浓度等信息后组帧,发送数据帧至总线上。命令帧是报警节点接收由中继器转发的上位机控制命令,包括报警命令、触发联动装置命令。命令格式如表 7-4 所列。

表 7-4 命令帧格式

帧信息	ID. 10~ID. 9	ID. 8~ID. 3	ID. 2~ID. 0+保留位	数据 1	数据 2
标准帧	00H(预留)	目标 ID	00H(命令字)	命令 1	命令 2

各节点发至中继器的数据帧中,数据字节长度为 8 字节:Data0~Data2 为温度信息;Data3~Data4 为湿度信息;Data5~Data7 为气体浓度信息。如表 7-5 所列。

表 7-5 数据帧格式

帧信息	ID. 10~ID. 9	ID. 8~ID. 3	ID. 2~ID. 0+保留位	数据 1~数据 8
标准帧	00H(预留)	目标 ID	本报警节点 ID	温度、湿度、浓度信息

字符码制:数字采用 Bin(二进制),字母采用 ASCII 码,以方便编程。

7.3 系统硬件设计

系统硬件设计主要包括:报警节点温度采集模块、RS485/CAN 转换模块、中继器模块、GSM 通信模块等。下面就各个部分主要电路做一个简单的介绍。

7.3.1 报警节点设计

各分布式节点由微控制器、CAN 通信模块、信息采集模块、报警模块、显示模块和看门狗电路组成。分布式节点安置在环境现场(各房间内),负责采集现场的温度、湿度、气体浓度等信息,且周期性地通过 CAN 通信线路发送到 CAN 总线上,再由转换模块传至上位机;上位机由此进行实时现场监控,如有异常,则各节点接收命令进行报警。图 7-3 给出了分布式节点硬件框图。

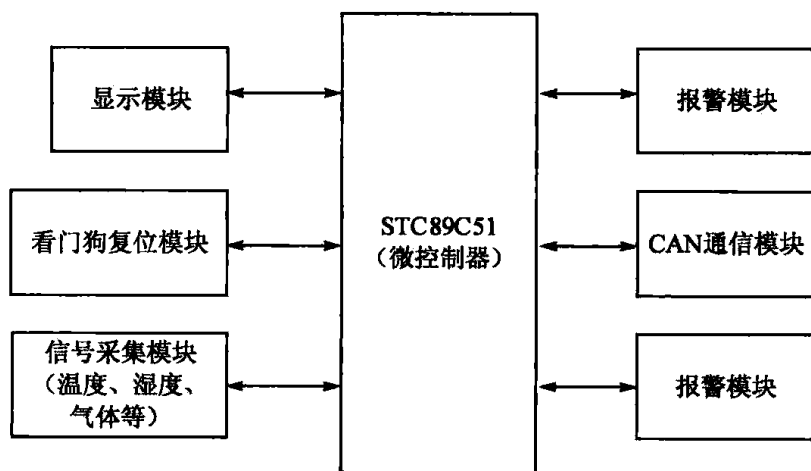


图 7-3 分布式节点硬件框图

主控制器采用 STC89C51 系列单片机,这是一款增强型单片机,处理速度、内部资源、I/O 口等都进行了扩展,指令代码完全兼容传统 8051 单片机,更易于开发。

信号采集模块:通过二氧化碳传感器(MG811)、湿度传感器(HF3223)、温度传感器(DS18B20)等采集现场气体浓度、湿度和温度。这里以温度和 CO₂ 测量为例进行介绍。

CAN 总线技术

1. 温度测量模块

(1) DS18B20 介绍

DS18B20 具有独特的单线接口, 仅需一个端口引脚进行通信; 简单的多点分布应用, 每个芯片唯一编码, 支持联网寻址; 无需外部器件、零功耗等待。图 7-4 为该传感器内部结构。

(2) 微控制器与 DS18B20 的连接

微控制器与 DS18B20 进行单线连接, 如图 7-5 所示。

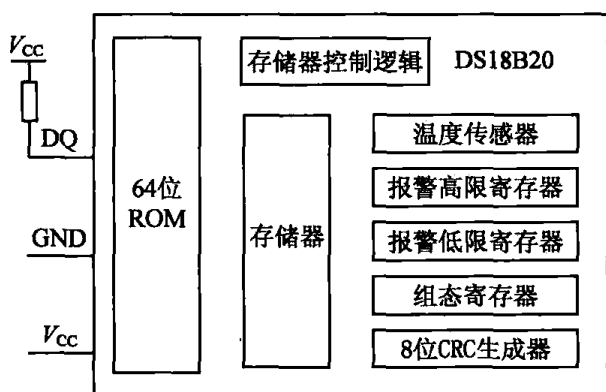


图 7-4 DS18B20 结构框图

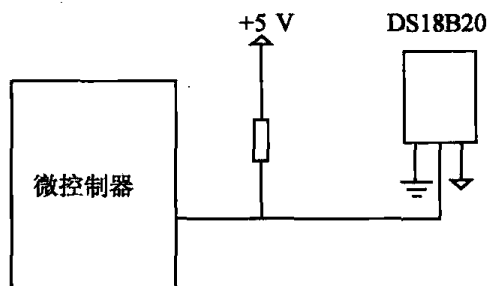


图 7-5 微控制器与 DS18B20 连接图

在单线端口条件下, 必须先建立 ROM 操作协议才能进行存储器和控制操作。因此, 控制器必须首先提供下面 5 个 ROM 操作命令之一: ① 读 ROM; ② 匹配 ROM; ③ 搜索 ROM; ④ 跳过 ROM; ⑤ 报警搜索。成功执行完一条 ROM 操作序列后即可进行存储器和控制操作, 控制器可以提供 6 条存储器和控制操作指令中的任一条。一条控制操作命令指示 DS18B20 完成一次温度测量。测量结果放在 DS18B20 的暂存器里, 用一条读暂存器内容的存储器操作命令可以读出暂存器中数据。

温度报警触发器 TH 和 TL 各由一个 EEPROM 字节构成。可以用一条存储器操作命令对 TH 和 TL 进行写入, 对这些寄存器的读出需要通过暂存器。所有数据都是以最低有效位在前的方式进行读/写。温度以 16 位带符号位扩展的二进制补码形式读出, 单片机可分高字节数据 Data_H 和低字节 Data_L 两个字节读出, 然后再做处理即可得到温度值。

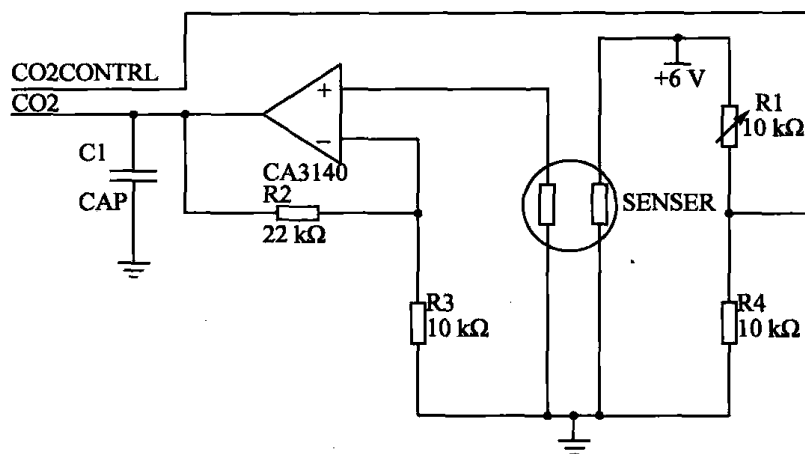
2. 气体浓度测量模块

MG811 传感器对 CO₂ 有良好的灵敏度和选择性, 受温湿度的变化影响较小, 具有良好的稳定性。

CO₂ 测量电路如图 7-6 所示。

3. CAN 通信模块

CAN 通信模块采用 SJA1000、TJ1050、TLP113 等芯片构成, 如图 7-7 所示。

图 7-6 CO₂ 测量电路原理图

为增加 CAN 总线节点的抗干扰能力, SJA1000 的 TX 和 RX 并不是直接与 TJA1050 的 TXD 和 RXD 相连, 而是通过高速光耦 TLP113 后与 TJA1050 相连, 这样就很好地实现了总线上各 CAN 节点间的电气隔离。

TJA1050 与 CAN 总线的接口部分也采用了一定的安全和抗干扰措施。TJA1050 的 CANH 和 CANL 引脚各自通过一个 $5\ \Omega$ 的电阻与 CAN 总线相连; 电阻可起到一定的限流作用, 保护 TJA1050 免受过流冲击。

CANH 和 CANL 与地之间并联了两个 $30\ \text{pF}$ 的小电容, 可起到滤除总线上的高频干扰和一定的防电磁辐射的作用。

另外, 两根 CAN 总线输入端与地之间分别接了一个防雷击管; 当两输入端与地之间出现瞬变干扰时, 防雷击管的放电可起到一定的保护作用。

在设计节点电路时, 还要注意下面几点:

- 光耦部分电路所采用的两个电源 V_{CC} 和 V_{DD} 必须完全隔离, 否则光耦失去意义;
- 当设置比较旁路为无效时, 引脚 RX1 须接两个分压电阻, 以给 RX1 提供比较电压, 否则不能正常通信。

4. 其他电路

显示模块采用 LCD 显示, 具体设计可参见第 8 章实验部分; 采用看门狗复位芯片设计看门狗复位电路; 报警模块采用声光报警。

7.3.2 转换模块设计

系统上位机采用 RS485 总线连接, 与 CAN 总线的通信通过 CAN/RS485 转换模块进行。该模块连接着不同类型的通信网络, 起着网桥的作用。一般情况下, 网桥要处理大量的数据, 所以对网桥模块的微控制器要求比较高, 要可靠性高、处理速度快、存储容量大。系统采用

CAN 总线技术

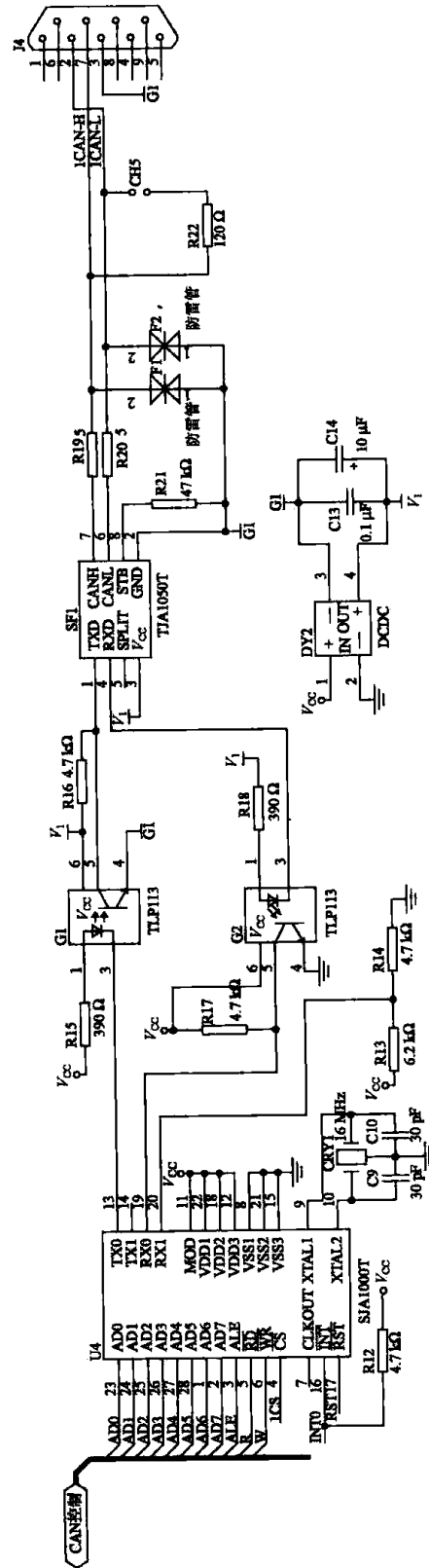


图 7-7 CAN通信电路原理图

ATmega128 是基于 AVR RISC 结构的 8 位低功耗 CMOS 微处理器。由于其先进的指令集以及单周期指令执行时间,ATmega128 的数据吞吐率高达 1 MIPS/MHz,从而可以缓减系统在功耗和处理速度之间的矛盾。AVR 内核具有丰富的指令集和 32 个通用工作寄存器。所有的寄存器都直接与逻辑运算单元(ALU) 相连接,使得一条指令可以在一个时钟周期内同时访问两个独立的寄存器。这种结构大大提高了代码效率,并且具有比普通的复杂指令集微处理器高 10 倍的数据吞吐率。

CAN/RS485 转换模块硬件包括:CAN 模块部分和串口电路部分,其中,CAN 模块部分的硬件电路跟其他区节点一样,这里只简叙串口电路部分(电路连接如图 7-8 所示)。为了提高通信可靠性,设计中采用两片 485 收发器(65HVD3082),一片用于接收,一片用于发送;读/写信号接 CPU 的异步收发器 0。为了增强系统的通用性,系统还预留了 RS232 接口;RS232 收发器(MAX232)占用了 CPU 的异步收发器 1。

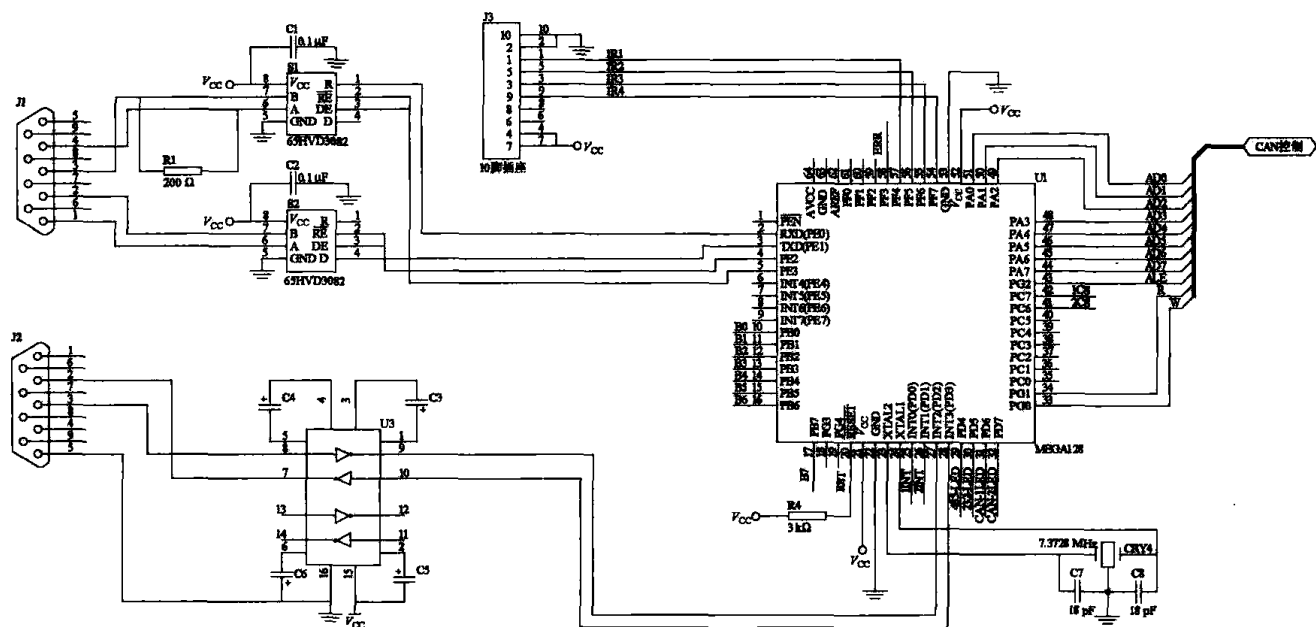


图 7-8 RS485/CAN 模块串口通信电路原理图

7.3.3 中继器模块设计

CAN 中继器扩大了 CAN 通信距离,增加了 CAN 节点数目。CAN 中继器模块包括微控制器电路、两路 CAN 通信模块、显示模块等,其结构框图如图 7-9 所示。

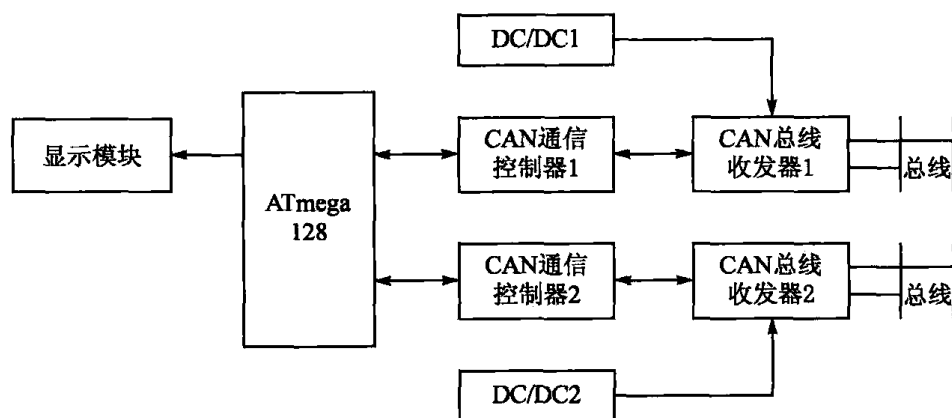


图 7-9 中继器结构框图

CAN 中继器的主要任务是在两个 CAN 网段之间实现数据的转发,所以对微控制要求也比较高。

7.3.4 GSM 电路设计

学校实验楼的管理一般由实验室管理员和保安负责。为了提高系统的智能化、人性化,系统利用 GSM 网络通过短信息方式进行数据传输,具有的预警功能,使得所用报警信息都以相应的短信息方式发送到手机和小灵通上、每个模块可以发送给 8 部手机或小灵通,提醒不在现场的实验室管理员,以配合相关部门进行意外处理。

1. GSM 介绍

GSM 即全球移动通信系统,是世界上主要的蜂窝系统之一。GSM 基于窄带 TDMA 制式,允许在一个射频同时进行 8 组通话。20 世纪 80 年代 GSM 兴起于欧洲,1991 年投入使用;到 1997 年底,已经在 100 多个国家运营,成为欧洲和亚洲实际上的标准。但 GSM 系统的容量是有限的,在网络用户过载时,就不得不构建更多的网络设施。GSM 在其他方面性能优异,除了提供标准化的列表和信令系统外,还开放了一些比较智能的业务(如国际漫游等)。GSM 手机的方便之处在于它提供了一个智能卡,人们称之为 SIM 卡;并且机卡可以分离,这样用户更换手机并且定制个人信息就十分便利了。GSM 手机还允许用户接收 160 字长度的短信息。

GSM 的基本特点:可以与各种公用通信网互连互通,尤其与 ISDN 的兼容性可提供更多的业务,各种接口规范明确,网络适合未来数字化发展的要求。GSM 组网结构灵活方便,能更有效地使用无线频率,抗干扰性强,通信质量高,能提供相当好的话音质量;采用了鉴权、语音

加密等技术使用户信息安全性得到保证。在采用 GSM 系统的所有国家范围内,可提供穿越国界的自动漫游功能。用户终端更小、更轻便,功能更强。

2. GSM 模块设计

系统选择西门子公司的 MC39I,接口与 TC35 系列通用,自带 RS232 通信接口,可以方便地与 PC 机、单片机联机通信。模块有 AT 命令集接口,支持文本和 PDU 模式的短消息、第三组的二类传真,GSM 单元也通过串口和控制器相连。GSM 模块电流变化非常大,空闲时电流小于 3.5 mA,而在通话期间电流最大可达 2.3 A,这就对供电电路提出了较高的要求。GSM 模块电源部分使用线性电压调节器把外部输入的电源电压 V_{batt+} 进行稳压处理后供 GSM 基带处理器和 GSM 射频部分使用;此外它还输出一个 2.5 V/70 mA 的电压供模块外的其他电路使用。GSM 射频部分的功率放大器对电源电压要求不高,所以直接使用外部的输入电压 V_{batt+} 。Flash 用来存储一些用户配置信息、电话本和其他信息。

3. GSM 模块与单片机通信硬件设计

GSM 模块主要由 4 部分组成:GSM 基带处理器、GSM 射频部分、电源、Flash。GSM 基带处理器是整个模块的核心,由一个 C166CPU 和一个 DSP 处理器内核控制着模块内各种信号的传输、转换、放大等处理过程。GSM 射频部分是一个单片收发器 SMARTi,由一个外差式接收器、上变频调制环路发送器、一个射频锁相环路和一个全集成中频合成器这 4 个功能块组成,共同完成对射频信号的接收和发送等处理。

图 7-10 给出了 GSM 模块与单片机通信电路。

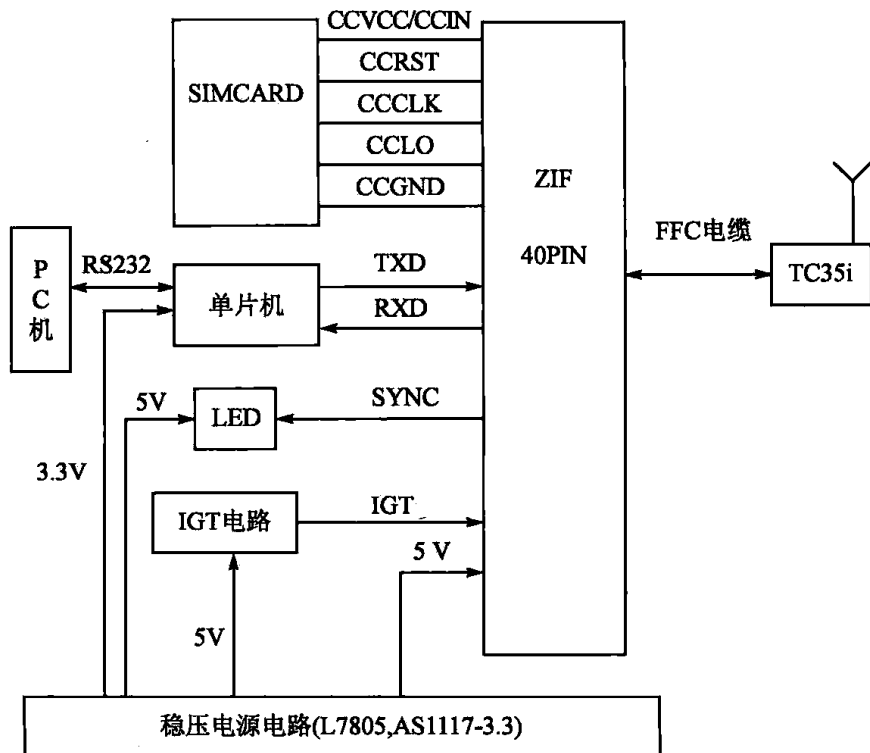


图 7-10 TC35i 外围硬件电路系统框图

7.4 系统软件设计

系统软件设计,包括上位机软件和硬件模块软件。即每一个功能程序模块都能完成某一明确的任务,实现具体的某个功能。采用模块化的程序设计方法,有下述优点:

- ① 单个模块结构的程序功能单一,因而易于编写、调试和修改;
- ② 便于分工,可由多个程序员同时进行编写、调试,加快了软件研制进度;
- ③ 程序可读性好,便于功能扩充和版本升级;
- ④ 程序的修改可局部进行,而其他部分则可以相对保持不变;
- ⑤ 使用频繁的子程序可以汇编成子程序库,以便于多个模块调用。

下面就系统软件设计进行介绍。

7.4.1 初始化模块设计

各硬件模块主程序中首先完成各种初始化操作。上电后进行初始化,包括:单片机初始化、CAN 控制器初始化、通信初始化以及其他中断源初始化。

1. 单片机初始化

- ① 单片机堆栈初始化。
- ② 端口初始化:设置各端口的输入/输出状态:“0”表示输入,“1”表示输出;设置输出 I/O 口 为安全侧。
- ③ 寄存器初始化:将各寄存器清零。
- ④ 数据区初始化:刷新程序中用到的数据区。
- ⑤ 初始化串行通信口的控制寄存器,设为接收状态。
- ⑥ 通信初始化完成的功能是设置波特率和 UART 通信参数,即 RX 完成中断使能、接受使能、发送使能。
- ⑦ 其他中断源初始化完成的功能是启动看门狗定时器、初始化外部中断、定时器中断等。

2. CAN 控制器 STJA1000 初始化配置

CAN 初始化程序即初始化 CAN 节点。根据协议正确的 CAN 初始化可以充分利用 CAN 总线的优势,保证 CAN 通信正确可靠工作。对 CAN 节点初始化只有在复位模式下才可以进行,初始化主要包括工作方式的设置、接收滤波方式的设置、接收屏蔽寄存器(AMR)、接收代码寄存器(ACR)的设置、波特率参数设置和中断允许寄存器(IER)的设置等。在完成 CAN 控制器的初始化设置以后,CAN 控制器就可以回到工作状态,执行正常的通信任务。初始化流程图详见第 8 章实验一。

3. 自检子模块

自检子模块包括固化程序机器码及常数自检(求和方式)程序、RAM 区中寄存器读/写自

检程序(非破坏性)及CAN自检程序等。自检子模块是每隔一定时间检查MCU内部程序是否运行正常,程序流程如图7-11所示。

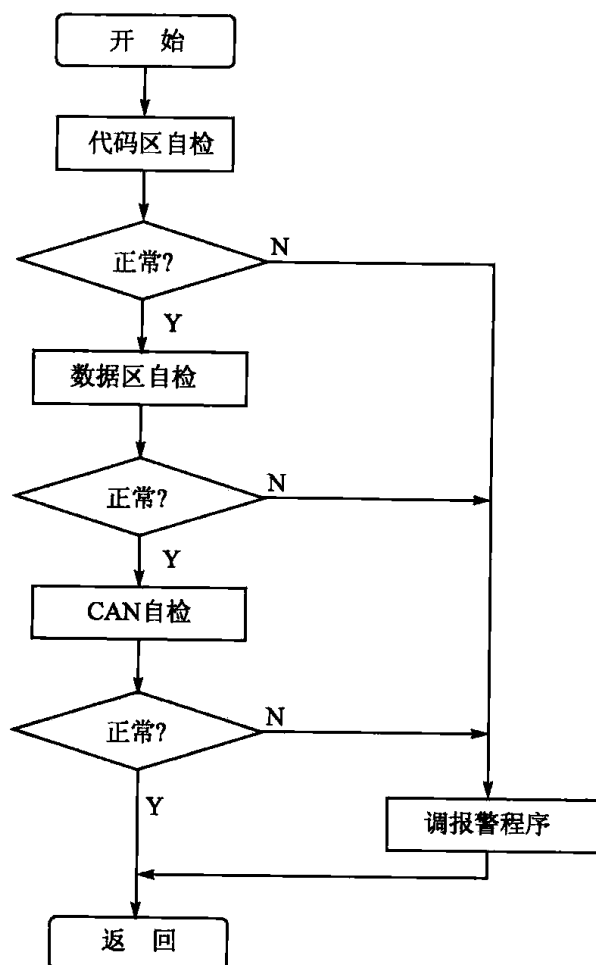


图 7-11 程序自检子模块流程图

(1) 数据区自检子模块

测试数据区读、写是否正常。首先将待测数据区的原始数据取出保存,然后将测试数据写入该数据区,再取出该数据区中写入的测试数据,判断特定数据区写入、读出是否正确,最后将待测数据区中的原始数据恢复。

(2) 程序区自检子模块

检查程序区运行是否正常。用间接寻址方式取出程序代码的字节进行累加,然后与代码区校验数据进行比较,若一致,则正常;若不一致,则调用MCU报警信息设置子模块。

(3) CAN控制器自检子模块

检查CAN控制器工作是否正常。首先读取总线关闭、错误状态、接收溢出等位。如果无错误,则退出该自检程序。如果有错误,判断总线是否开启。若总线开启,则判断数据是否溢

CAN 总线技术

出,如果数据没有溢出,则退出该程序;如果数据溢出,则清除数据溢出,调用命令执行程序,释放接收缓冲区。若总线关闭,出现总线脱离,则调用 CAN 退出复位模式、进入工作模式程序,判断 CAN 是否工作正常,正常则退出该程序;不正常则调用 MCU 报警信息设置子程序。

7.4.2 报警节点软件设计

1. CAN 通信模块

CAN 通信模块包括 CAN 的定时发送程序及 CAN 的中断接收程序。

(1) CAN 定时发送程序设计

系统规定各报警节点周期性上传数据,所以报警节点在定时中断中发送数据(现场信息),具体流程如图 7-12 所示。可见,发送之前须做些判断,是否正在接、先前发送是否成功、发送缓冲器是否锁定等,以确保数据发送可靠。

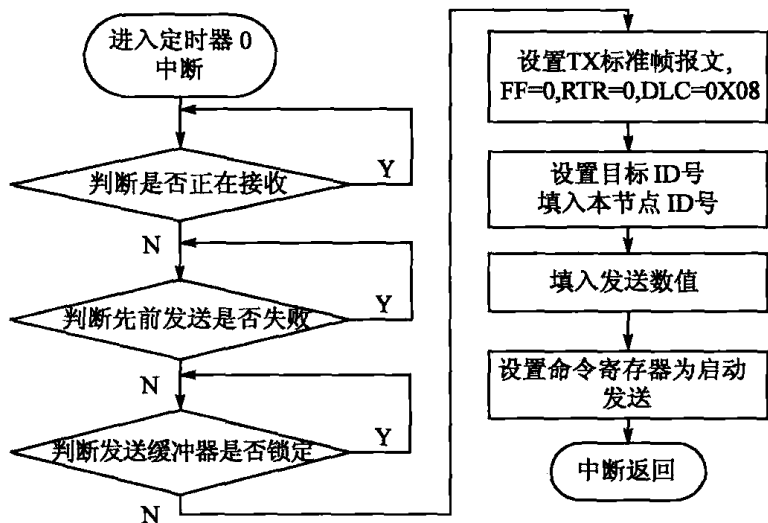


图 7-12 CAN 定时发送流程

值得指出的是,程序设计中应尽量避免死循环,如果 CAN 节点在发送过程中出现故障而使程序陷入死循环,则必须采取措施。可在发送判断过程中加入一些条件限制(比如发送次数计数)或者启动看门狗复位。本系统设计时使用了看门狗。

(2) CAN 中断接收程序设计

进入中断后要进行是否有数据的判断,以防止干扰误中断,具体流程如图 7-13 所示。

2. 现场信息采集模块设计

根据各传感器的工作方式,按照正确的时序进行读/写,实时采集现场信息。以温度采集为例,程序流程如图 7-14 所示。

温度采集每次都要先复位,再写命令读值。此程序已模块化,把它放在主循环中一直执行。各报警节点定时采集现场信息并将数据上传。

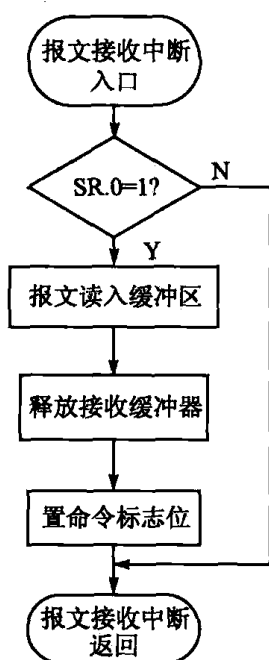


图 7-13 CAN 中断接收流程图

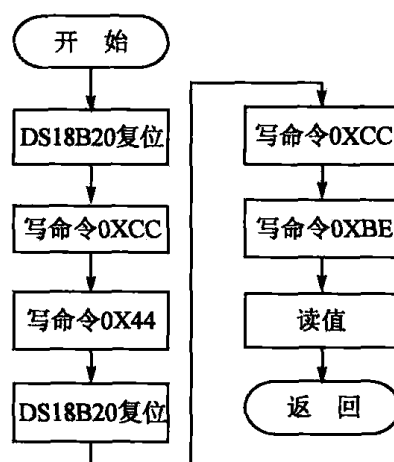


图 7-14 现场温度采集流程

7.4.3 CAN/RS485 模块软件设计

CAN/RS485 通信节点通过这部分程序接收从上位机串行接口发来的命令,并根据用户定义进行命令处理,采用中断方式接收计算机信息。另一方面,该部分软件还负责将经过 CAN/RS485 通信节点处理后的现场状态传送给上位机,实现整个模块进行监控;CAN 通信节点采用查询方式实现数据发送。本设计中设计了两个串行接口:RS232 和 RS485,分别接到 CPU 的两个串口。

该模块的 CAN 程序设计与 7.4.1 小节介绍的 CAN 通信类似,这里就不再赘述,只要注意根据协议对验收滤波器进行合理配置。

1. 串行通信模块

串行通信数据的发送采用查询方式,每个周期把从各个现场模块采集并打包的数据发送给联锁机;串行通信数据接收采用中断形式,流程如图 7-15 所示。在接收完成后,若发生传输错位或校验码错误,则这一包数据作废,并向上位机发出接收错误信息。

2. 数据处理子模块

数据处理模块完成两部分功能:

① 计算机命令处理。转换模块将从计算机接收到的命令根据用户协议进行分解、存储,等待下发到 CAN 总线上。

② CAN 数据处理。转换模块将从 CAN 总线接收到的报文根据用户协议进行打包,存

CAN 总线技术

储,等待上传给上位机。

在与上位机通信时,数据量大,为了保证通信可靠性,要采用有效的校验方式。随着技术的不断进步,各种数据通信的应用越来越广泛。由于传输距离、现场状况、干扰等诸多因素的影响,设备之间的通信数据常会发生一些无法预测的错误。为了降低错误所带来的影响,一般在通信时采用数据校验的办法,而循环冗余码校验(CRC)是常用的重要校验方法之一。CRC计算可以靠专用的硬件来实现,但是对于低成本的微控制器系统,在没有硬件支持下实现CRC检验,可以通过软件来完成。

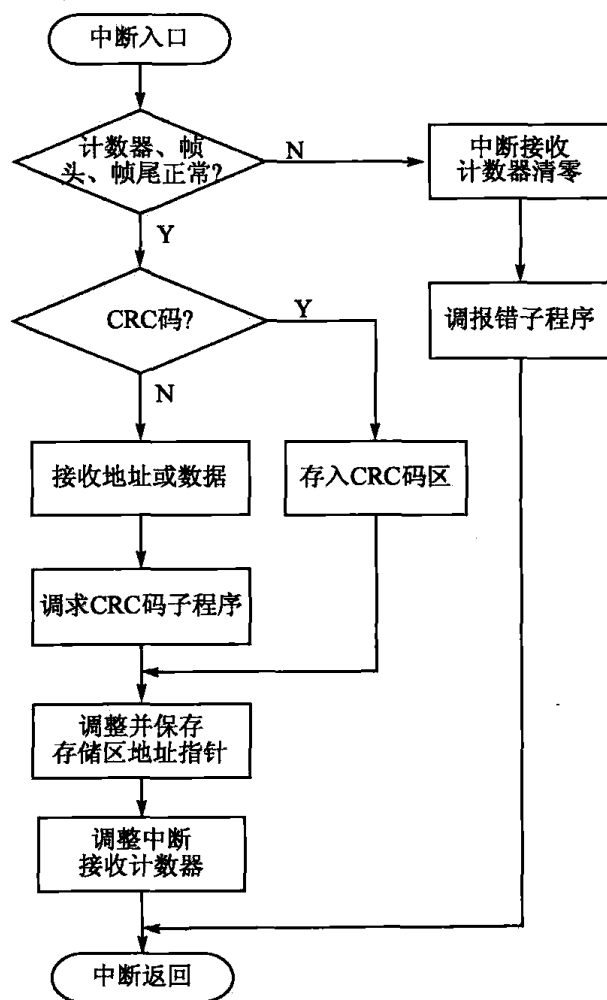


图 7-15 串口中断接收程序流程图

在 CCITT 推荐的高级数据链路控制规程 HDLC 的帧校验序列 FCS 中,使用 CITT-32, 即 $g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ 。

CRC-32 出错的概率比 CRC-16 低 10^{-5} 倍。由于其可靠性,CRC-32 非常适用于重要数据传输。在上位机与转换模块的通信中,根据 CRC 校验原理及算法,设计了 CRC-32 查表

法校验子程序,以提高通信的可靠性。图7-16是软件方法实现CRC校验的流程图。

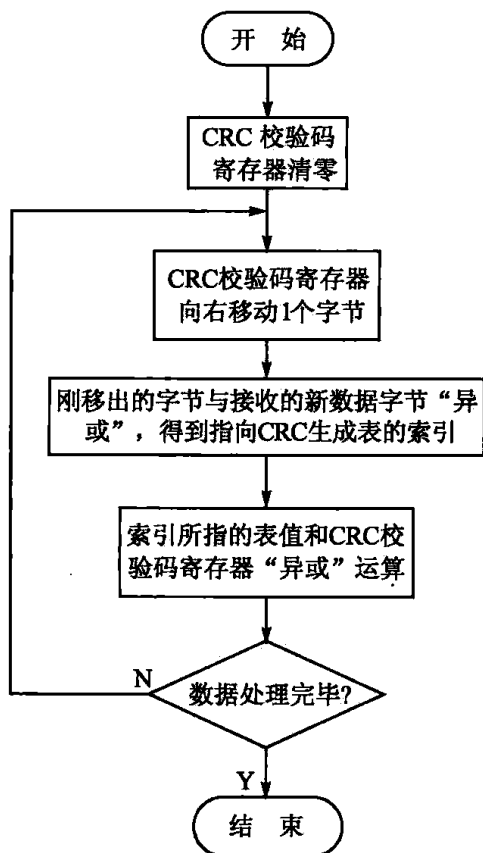


图7-16 32位CRC校验查表法流程图

图7-16所示为查表法生成32位CRC校验码的流程图。读者如果有兴趣可以参考相关资料。

另一方面,对于CAN总线而言,CAN的每帧信息都采用CRC校验及其他检错措施,具有良好的检错效果。所以,整个通信系统的数据通信可靠性有了很大的保证。

7.4.4 中继器模块软件设计

中继器模块软件负责将转换模块的数据(命令)下发给各报警节点,并将各报警节点数据(现场信息)上传给转换模块。如前所述,此处CAN通信采用扩展数据帧。在CPU中开辟两处固定长度存储区:REBUFF1、REBUFF2,依次存放各报警节点的信息和各节点的命令;存储区数据分别对应于CAN1接收中断、CAN2接收中断所接收的数据。数据区实时刷新,即只要接到数据就刷新存储区。模块定时将报警节点信息组帧发送至转换模块,只要收到上位机命令就组帧转发给各报警模块。

CAN通信采用中断方式接收,查询方式发送。具体流程与前述相似,不再赘述。

7.4.5 上位机软件设计

上位机软件要完成监控界面、数据判断处理、命令发布、手机报警短信发送、数据库存储更新等操作。上位机一般采用工控机或可靠性较好的计算机,带有两个串口,一个串口用以和模块转换器通信,另一个则用于 GSM 信息的发送。各节点的控制单片机也可以设计成自主进行 GSM 信息的发送模式,但这需要大量的 GSM 模块和占用大量的串口资源,利用上位机进行 GSM 通信则只需要一个 GSM 模块和多占用一个串口,这样降低了硬件开销;并且,用上位机进行 GSM 通信可以通过设计的软件自主选择通信对象,提高了其灵活性。

考虑到开发的方便性,这里采用 VB 图形软件工具。Visual Basic 是 Microsoft 公司开发的 Windows 应用程序开发工具。Visual 即“可视化的”,是一种开发图形用户界面(GUI)的方法,使得非计算机专业的人也可以开发出专业的 Windows 软件。Visual Basic 继承了 BASIC 语言简单易学的优点,且增加了许多新的功能;它采用面向对象与事件驱动的程序设计思想,使编程变得更加方便、快捷。使用 Visual Basic 既可以开发个人或小组使用的小型工具,又可以开发多媒体软件、数据库应用程序、网络应用程序等大型软件,是国内外最流行的程序设计语言之一。

数据库用于存储各报警节点信息,便于监控,也可为以后火灾发生趋势提供参考。由于整个监控系统数据比较多,为了节省存储,只有当节点数据超出一定范围时才更新数据库信息,从而大大减少了数据库负担。基于简单易学、应用广泛的考虑,本设计采用 Assecc。

Access 是微软公司推出的基于 Windows 的桌面关系数据库管理系统(RDBMS),是 Office 系列应用软件之一。它提供了表、查询、窗体、报表、页、宏、模块 7 种用来建立数据库系统的对象;提供了多种向导、生成器、模板,把数据存储、数据查询、界面设计、报表生成等操作规范化;为建立功能完善的数据库管理系统提供了方便,也使得普通用户不必编写代码,就可以完成大部分数据管理的任务。

综合以上分析,本设计采用 VB+Assecc 作为上位机软件。VB 软件与硬件的数据通信是通过其串口控件来完成的(需在“工程→部件”中选中 Microsoft comm control 6.0),数据流向如图 7-17 所示。

上位机软件部分分为 VB 人机界面部分和后台数据库部分。

为了保证数据处理的实时性,CAN 各节点只负责数据的采集,对数据的处理和报警则由上位机来完成,上位机可根据当前要求对报警上下限进行设置。当有报警发生时,报警命令发出以通知本地监控人员,并以短信的信息发送给相关管理人员;当报警处理后,可发送相应的报警解除命令以恢复正常工作状态。上位机依次轮流对各节点的数据进行采集,当在若干个周期不能采集到某些节点的数据时,则认为和该节点的通信发生中断,从而引发通信链路故障报警。

上位人机界面可分为主界面窗口(frmMain.vb)、火灾信息查询窗口(frmAlarmInfo.vb)。

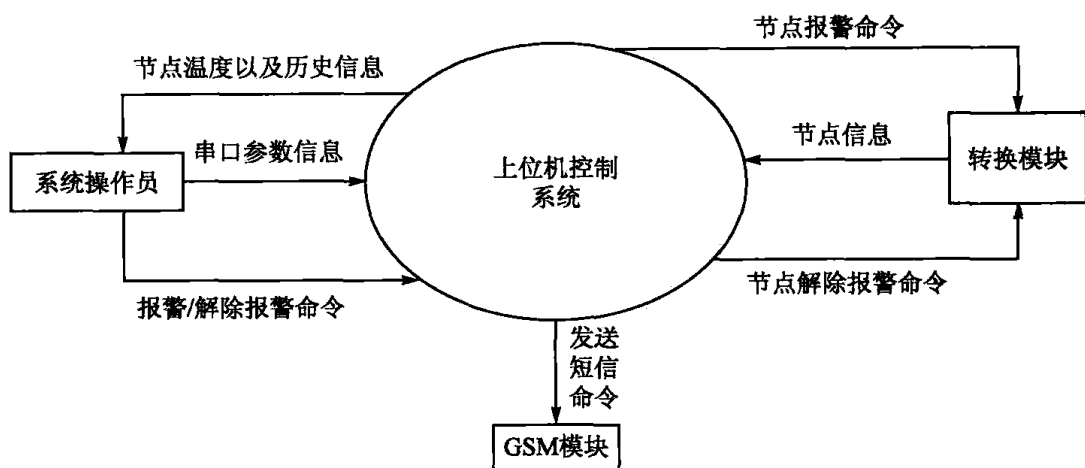


图 7-17 上位机数据流向图

frmMain 窗体中主要有串口控件(MSComm)、数据库连接控件(Adodc)、按键(Command)等。火灾信息查询窗体中主要有 DataGrid 控件,用来显示数据信息。而每个控件都有相应的事件,在事件中编写相应的处理程序。串口控件中有接收和发送事件,在这事件中处理接收、发送的数据。除了事件外,还有子过程、函数。在本设计中,创建标准模块和类模块。标准模块中的程序可在不同的应用程序中重用,故其中设置有常量模块(Const. bas)、变量模块(Variable. bas)、通用函数模块(GeneralFunc. bas)、数据库函数模块(DbFunc. bas);且可在类模块中编写代码建立新对象。

在数据库设计时,这里创建 TIME、ID、TEMP、HUMI、GAS 这 5 个字段,分别用来存储火灾时间、节点 ID、温度值、湿度值、气体浓度值。

(1) 主操作模块功能

主操作模块能够将实时的温度信息形象化地显示在电脑屏幕上。

- 能够调用和协调其他模块工作;
- 提供方便简洁的人机交互界面。

(2) 串口设置模块功能

串口设置模块功能:

- 能够选择温度节点和 GSM 模块使用哪个串口;
- 能够由操作员控制串口通信的参数,包括波特率、校验位、数据位和停止位;
- 能够设置报警的温度上限值。

(3) 串口通信模块功能

1) 温度节点信息接收子模块

温度节点信息接收子模块能够将温度节点的信息解包获得温度信息。

CAN 总线技术

2) 温度节点控制子模块

温度节点控制子模块功能如下:

- 能够根据传送来的节点温度值和所设置的温度上限值比较后自动向相应的节点发送命令;
- 能够向指定节点发送报警命令;
- 能够向指定节点发送解除报警命令。

3) GSM 控制子模块

GSM 控制子模块能够和 GSM 模块通信,控制 GSM 模块向指定手机发送短信。

(4) 数据分析模块功能

数据分析模块功能如下:

- 能够根据接收到的节点信息判断是否发送报警命令、是否发送联动命令、是否该解除报警;
- 当某节点信息变化至超过所设置的上限时,则通过串口通信模块向该节点发送报警命令并控制 GSM 模块发送报警短信;
- 当某节点信息下降到上限以下时,则通过串口通信模块向该节点发送解除报警命令并控制 GSM 模块发送“已经安全”短信。

(5) 数据保存模块功能

数据保存模块功能:

- 将接收到的节点温度数据保存入数据库;
- 能够将保存的历史信息随时显示给操作员。

7.5 系统抗干扰措施

对于一个完整的系统设计,抗干扰措施是必不可少的。抗干扰设计的基本原则是抑制干扰源,切断干扰传播路径,提高敏感器件的抗干扰性能。硬件抗干扰技术是设计系统时重要的抗干扰措施,能有效抑制干扰源、阻断干扰传输通道,以提高系统可靠性和安全性。在设计中主要采取以下措施抗干扰:

1. 电源抗干扰设计

电源中的干扰主要有高频干扰、感性负载产生的瞬变噪声、电网电压的短时间下降干扰、拉闸形成的高频干扰。针对电源干扰的特性,选择如下方式提高抗干扰能力:

① 采用瞬态电压抑制器(Transient Voltage Suppression Diode)简称 TVS,是一种二极管形式的高效能保护器件。当 TVS 二极管的两极受到反向瞬态高能量冲击时,它能以 10^{-12} 量级的速度将其两极间的高阻态变为低阻抗,吸收高达数千瓦的浪涌功率,使两极间的电压钳位于一个预定值,有效地保护电子线路中的精密元器件免受各种浪涌脉冲的损坏。设计中将

TVS加在信号线与地线间,从而避免数据及控制总线受到不必要的噪声影响。

② 充分考虑电源对系统的影响。采用滤波电容和高精度 AC/DC 转换器(YAS5-5-N),以减小电源噪声对单片机的干扰。采用滤波电容:在电源的进入处并联一个大容量的($>1000\ \mu\text{F}$)电解电容和电容($0.1\ \mu\text{F}$)。电解电容抑制低频波动,而来自电源侧的高频干扰由高频电容抑制。这里高频电容不可少,这是由于电解电容器在高频下工作时存在不可忽视的电感特性,因此高频干扰不能滤除掉。电解电容也改善了纹波。高频电容可以改善负载端的瞬态响应,抑制瞬变噪声干扰。

2. CAN 通信链路抗干扰设计

(1) 光电隔离

设计过程中的 CAN 链路的两个 SJA1000 的 TX 和 RX 通过高速光耦 TLP113 后与 TJA1050 相连,可实现总线上各 CAN 节点间的电气隔离。光耦两端电路所采用的两个电源必须完全隔离,这里采用小功率电源隔离模块(BO5O5LS-1W)进行电源隔离。高速光耦 TLP113 的输入端配置发光源,输出端配置受光器,因而在电气上输入/输出是完全隔离的。光耦合器的主要优点是单向传输信号,输入端与输出端完全实现了电气隔离,抗干扰能力强,使用寿命长,传输效率高。由于光电耦合器的输入阻抗与一般干扰源的阻抗相比较小,因此分压在光电耦合器输入端的干扰电压较小,它所能提供的电流并不大,不易使半导体二极管发光;光电耦合器的外壳是密封的,不受外部光的影响;光电耦合器的隔离电阻很大(约 $10^{12}\ \Omega$)、隔离电容很小(约几个 pF),所以能阻止电路性耦合产生的电磁干扰。光电耦合器是电流驱动型,需要足够大的电流才能使发光二极管导通;如果输入信号太小,发光二极管不会导通,其输出信号将失真。在开关电源尤其是数字开关电源中,利用线性光耦合器可构成光耦反馈电路,通过调节控制端电流来改变占空比,达到精密稳压目的。

(2) CAN 物理链路电路抗干扰设计

这里充分考虑总线抗干扰性能,采用了总线驱动器 TJA1050;TJA1050 本身电磁辐射极低、电磁抗干扰极高。CAN 物理链路采用双绞线;双绞线不仅具有较低的成本也具有较好的抗干扰性能。根据 ISO 11898 中定义的线性拓扑结构,总线两端都端接一个不低于 $120\ \Omega$ 的额定电阻,终端电阻和电缆阻抗的紧密匹配确保了数据信号不会在总线的两端反射。这里采用总线分离终端方法,有效减少线路辐射,提高抗干扰能力。

3. 软件抗干扰措施

监控系统对计算机运行的可靠性与安全性有很高的要求。窜入微机测控系统的干扰的频谱往往很宽且具有随机性,在提高硬件系统抗干扰能力的同时,软件抗干扰以其设计灵活、节省硬件资源、可靠性好的特点而越来越受到重视。

设计中主要针对程序运行混乱时怎样使程序重入正轨而采取了一些方法:

(1) 指令冗余

CPU 取指令过程是先取操作码,再取操作数。当 PC 受干扰出现错误时,程序便脱离正

CAN 总线技术

常轨道“乱飞”，当乱飞到某双字节指令时，若取指令时刻落在操作数上，从而误将操作数当作操作码，则程序将出错。若“飞”到了三字节指令，则出错机率更大。

在关键地方人为插入一些单字节指令或将有效单字节指令重写称为指令冗余。通常是在双字节指令和三字节指令后插入两个字节以上的 NOP，这样即使乱飞程序飞到操作数上，由于空操作指令 NOP 的存在，也可以避免了后面的指令被当作操作数执行，程序自动纳入正轨。

(2) 拦截技术

拦截是指将乱飞的程序引向指定位置，再进行出错处理。通常用软件陷阱来拦截乱飞的程序。因此先要合理设计陷阱，其次要将陷阱安排在适当的位置。软件陷阱就是一条引导指令，强行将捕获的程序引向一个指定的地址，在那里有一段专门对程序出错进行处理的程序。软件陷阱一般安排在未使用的中断向量区、未使用的大片 ROM 空间、表格、程序区。将捕获的乱飞程序引向复位入口地址的指令。

(3) 看门狗技术

若失控的程序进入死循环，则通常采用看门狗技术使程序脱离死循环。在正常情况下，程序启动 WDT 后周期性进行看门狗复位，这样 WDT 的定时溢出就不会发生。在受到干扰的异常情况下，CPU 时序逻辑破坏，程序逻辑执行混乱，不可能周期性地将 WDT 清零，这样当 WDT 的定时溢出时，其输出使系统复位；但 WDT 只是一种被动的抗干扰措施，只能在一定程度上减少干扰造成的损失。

(4) 设置自检程序

在程序设计的特定部位(如某些内存单元)设状态标志，则在运行中不断循环测试，以保证系统中信息存储、传输、运算的高可靠性。

由于单片机和 CAN 总线技术的日益发展、完善，二者构成监控系统可以很方便地应用于各种场合。将通用的 CAN 网络通信协议与嵌入式技术相结合时，需要对 CAN 网络通信协议有透彻的了解，同时要对嵌入式领域的特点有准确的认识。本章所设计的监控系统可以在较低成本的条件下实现对学校各实验楼的监控，还可以应用在智能楼宇、工业现场等场合。

第 8 章

实验指导

本章根据教学内容及实际教学情况,考虑到难易程度,介绍了 4 个基于 CAN 总线的课内实验,包括 3 个基本实验(分 4 次做)和 1 个综合实验。这些实验可以利用已有的实验箱进行开发验证,亦可自行设计开发硬件平台进行实验。本章以自制实验平台为硬件平台,结构简单,成本低廉,易于开发,既适应用于自行开发又适于移植到其他实验箱。

8.1 实验开发平台

8.1.1 软件开发平台

1. Keil 开发软件的介绍

Keil μ Vision2 的集成开发环境基于 8051 内核的微处理器软件开发平台,内嵌多种符合当前工业标准的开发工具,可以完成从工程建立和管理、编译、链接和目标代码的生成、软件仿真、硬件仿真等完整的开发流程。尤其编译工具在产生代码的准确性和效率方面达到了比较高的水平,而且可以附加灵活的控制选项,在开发大型项目时非常理想。Keil 本身是一个纯软件的东西,还不能直接进行硬件仿真,必须挂接类似 TKS 系列仿真器的硬件才可以进行仿真。

(1) μ Vision2 IDE

μ Vision2 IDE 包括一个工程管理器、一个功能丰富并有交互式错误提示的编辑器、选项设置生成工具以及在线帮助;可以使用 μ Vision2 创建源文件,并组成应用工程加以管理。 μ Vision2 可以自动完成编译和链接操作。

(2) C51 编译器和 A51 汇编器

由 μ Vision2 IDE 创建的源文件 Keil C51 编译器遵照 ANSI C 语言标准支持 C 语言的所有标准特性,另外,还增加了几个可以直接支持 80C51 结构的特性。Keil A51 宏汇编器支持 80C51 及其派生系列的所有指令集。

CAN 总线技术

(3) LIB51 库管理器

LIB51 库管理器可以由汇编器和编译器创建的目标文件建立目标库,这些库是按规定格式排列的目标模块,以后可被链接器使用。当链接器处理一个库时,仅仅使用了库中程序使用的目标模块而不是全部加以引用。

(4) BL51 链接器

BL51 链接器使用从库中提取出来的目标模块和编译器创建的目标文件建立目标库。这些库是按规定格式生成的目标模块,所有的代码和数据都固定在具体的存储器单元中,绝对地址目标文件可以存储于 EPROM 或其他存储器设备,由 μ Vision2 调试器对目标进行调试和模拟,使用在线仿真器进行程序测试。 μ Vision2 软件调试器能十分理想地进行快速可靠的程序调试。调试器包括一个高速模拟器,能模拟整个 80C51 系统,包括片上外围器件和外部硬件。从器件数据库选择器件时这个器件的属性会自动配置。

(5) μ Vision2 硬件件调试器

μ Vision2 调试器提供了几种在实际目标硬件上测试程序的方法。安装 MON51 目标监控器到目标系统,并通过 Monitor-51 接口下载程序。 μ Vision2 的人机交互环境使用高级 GDI 接口指挥连接的硬件完成仿真操作。

在 C51 编译器中集成了 RTX51 实时操作系统,使用非常简单。RTX51 实时操作系统是针对 80C51 微控制器系列的一个多任务内核,简化了需要对实事件进行反应的、复杂应用的系统设计、编程和调试。任务描述表和操作系统的一致性由 BL51 链接器定位器自动进行控制。

2. Keil 软件开发的流程

对于刚刚使用 Keil 的用户来讲,一般是按照下面的流程来完成开发任务的:

- 建立工程;
- 为工程选择目标器件,如选择 NXP 的 P87C591;
- 设置工程的配置参数;
- 打开/建立程序文件;
- 编译和链接工程;
- 纠正程序中的书写和语法错误并重新编译链接;
- 对程序中某些纯软件的部分使用软件仿真验证;
- 连接硬件仿真设备进行硬件仿真;
- 将生成的 Hex 文件烧写到 ROM 中运行测试。

3. Keil 软件的工作环境

安装完成后用户可以单击运行图标进入 IDE 环境。 μ Vision2 软件有菜单栏、工具栏、源代码文件窗口、对话框窗口、信息显示窗口;允许同时打开几个源程序文件。

菜单栏提供了各种操作菜单,比如编辑器操作、工程维护、开发工具、选项设置、程序调试、窗体选择和操作在线帮助。工具栏按钮可以快速执行 μ Vision2 命令。快捷键可以自己配置,

也可以执行 μ Vision2 命令。主菜单如图 8-1 所示。工程菜单如图 8-2 所示。视图菜单如图 8-3 所示。工具菜单如图 8-4 所示。

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

图 8-1 主菜单界面

Project 菜单	工具栏	快捷键	描述
New Project			创建一个新工程
Import μ Vision1 Project			输入一个 μ Vision1 工程文件
Open Project			打开一个已有的工程
Close Project			关闭当前工程

图 8-2 工程菜单界面

View 菜单	工具栏	快捷键	描述
Status Bar			显示或隐藏状态栏
File ToolBar			显示或隐藏文件工具栏
Buid Toolbar			显示或隐藏编译工具栏
Debug Toolbar			显示或隐藏调试工具栏
Project Window			显示或隐藏工程窗口
Output Window			显示或隐藏输出窗口
Source Browser			打开源（文件）浏览器窗口
Disassembly Window			显示或隐藏反汇编窗口
Watch & Call Stack Window			显示或隐藏观察和堆栈窗口
Memory Window			显示或隐藏存储器窗口
Code Coverage Window			显示或隐藏代码覆盖窗口
Performance Analyzer Window			显示或隐藏性能分析窗口
Symbol Window			显示或隐藏符号变量窗口
Serial Window #1			显示或隐藏串行窗口 1
Serial Window #2			显示或隐藏串行窗口 2
Toolbox			显示或隐藏工具箱
Period Window Update			在运行程序时，周期刷新调试窗口
Workbook Mode			显示或隐藏工作簿窗口的标签
Options			设置颜色、字体、快捷键和编辑器选项

图 8-3 视图菜单界面

CAN 总线技术

Tools 菜单	工具栏	快捷键	描述
Setup PC-Lint ...			配置 Gimpel Software 公司的 PC-Lint
Lint			在当前的编辑文件中运行 PC-Lint
Lint all C Source File			在工程的 C 源代码文件中运行 PC-Lint
Setup Easy-Case			配置 Siemens Easy-Case
Start/Stop Easy-Case			启动或停止 Siemens Easy-Case
Show File(Line)			在当前的编辑文件中运行 Easy-Case
Customize Tools Menu...			将用户程序加入工具菜单

图 8-4 工具菜单界面

通过工具菜单可以配置、运行 Gimpel PC-Lint、Siemens Easy-Case 和用户程序。执行 Customize Tools Menu, 可以将用户程序添加到菜单中。调试菜单如图 8-5 所示。外围器件菜单如图 8-6 所示。

Debug 菜单	工具栏	快捷键	描述
Start/Stop Debugging		Ctrl+F5	启动或停止 μ Vision2 调试模式
Go		F5	运行 (执行), 直到下一个有效的断点
Step		F11	跟踪运行程序
Step Over		F10	单步运行程序
Step out of current function		Ctrl+F11	执行到当前函数的程序
Run to Cursor Line		Ctrl+F10	执行到当鼠标所在行
Stop Running		ESC	停止程序运行
Breakpoints			打开断点对话框
Insert/Remove Breakpoints			在当前行设置/清除断点
Enable/Disable Breakpoints			使能/停止当前断点
Disable All Breakpoints			禁用程序中所有的断点
Kill All Breakpoints			清除所有的断点
Show Next Statement			显示下一条执行的语句/指令
Enable/Disable Trace Recording			使能跟踪记录, 可以显示程序运行轨迹
View Trace Records			显示以前执行的命令
Memory Map...			打开存储器空间配置对话框
Performance Analyzer...			打开性能分析器的设置对话框
Inline Assembly			对某一行重新汇编, 可以修改汇编代码
Function Editor			编辑调试函数和调试配置文件

图 8-5 调试菜单界面

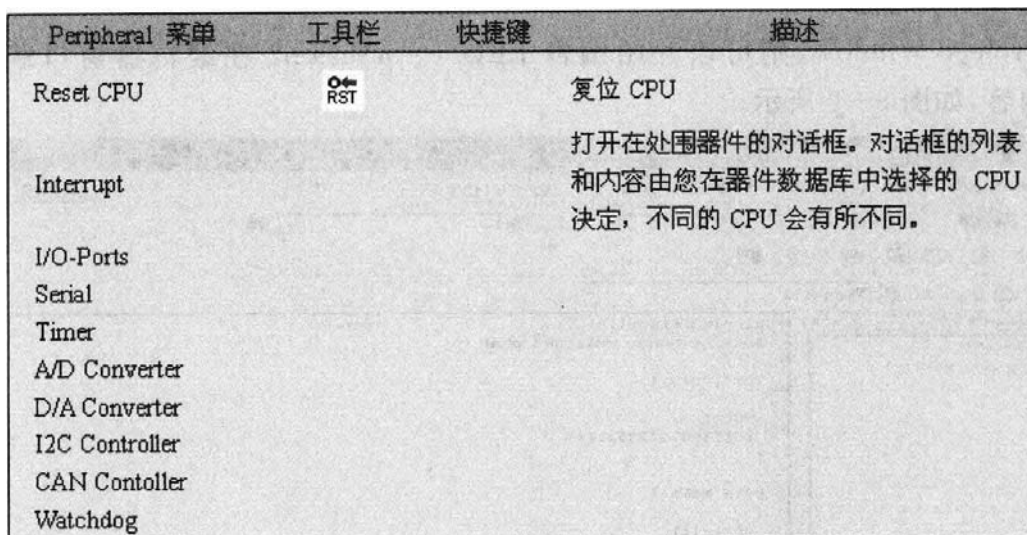


图 8-6 外围器件菜单界面

例：编程实现由 P1 口控制 LED 灯，使其闪烁。简单步骤如下：

1) 建立工程

选择 project→new project 菜单项，建立新的工程 Shiyan. uv2，如图 8-7 所示。

2) 选择器件

工程建立后自动进入选择器件界面，或者选择 project→select device for 菜单项，如图 8-8 所示。

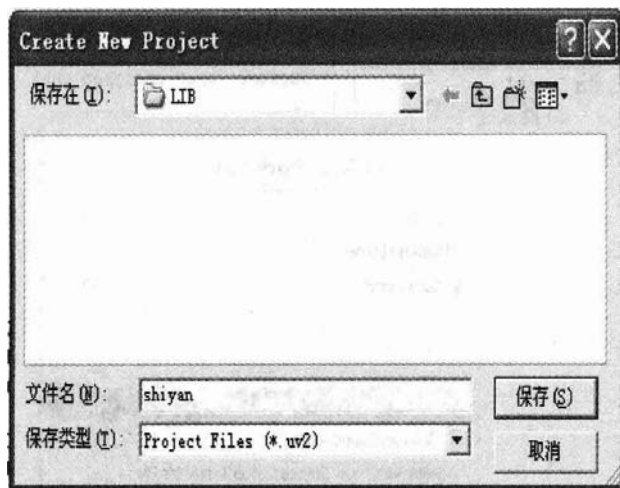


图 8-7 新建工程

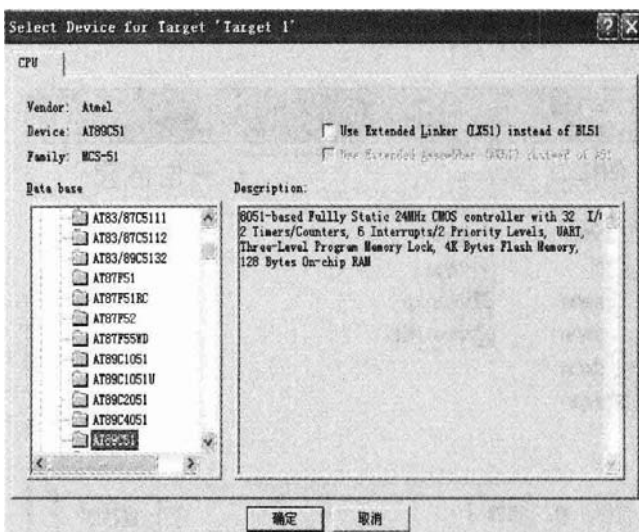


图 8-8 选择器件

CAN 总线技术

3) 编辑文件并添加

双击 Project Window, 则可以开始编辑 LED. c。μVision2 在编辑器窗口载入和显示 LED. c 的内容, 如图 8-9 所示。

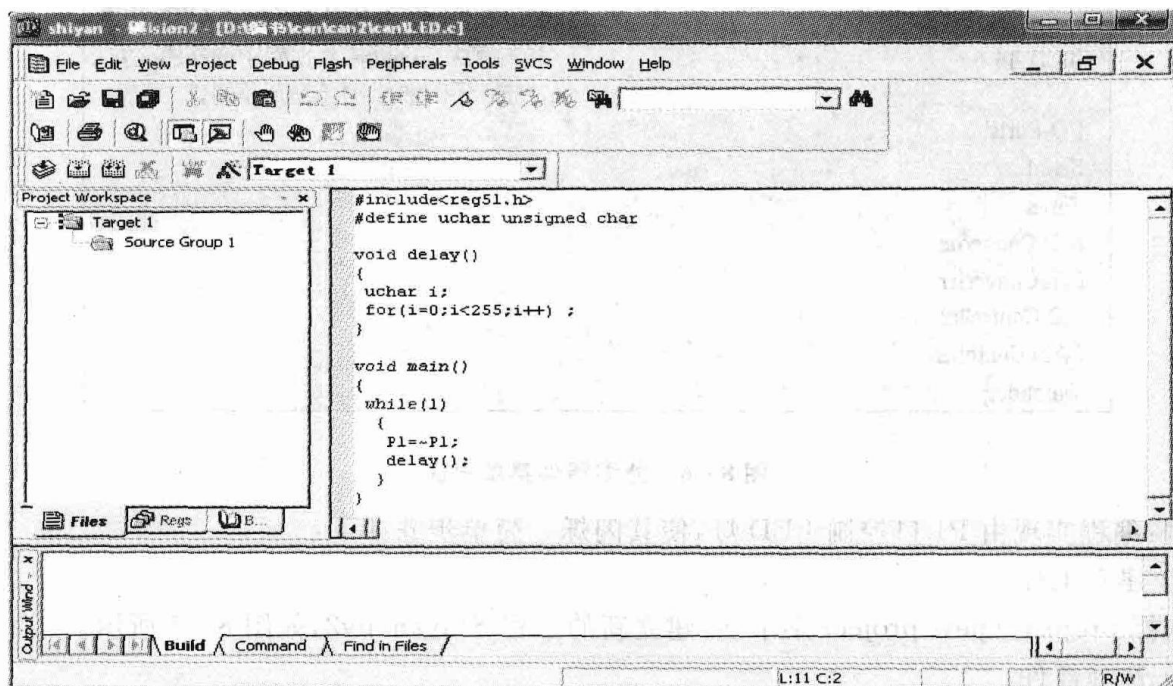


图 8-9 Shiyen 工程的编辑/编译界面

选择 File→New 菜单项新建 LED. c 文件, 并保存添加到 shiyen 工程下, 如图 8-10、图 8-11 所示。

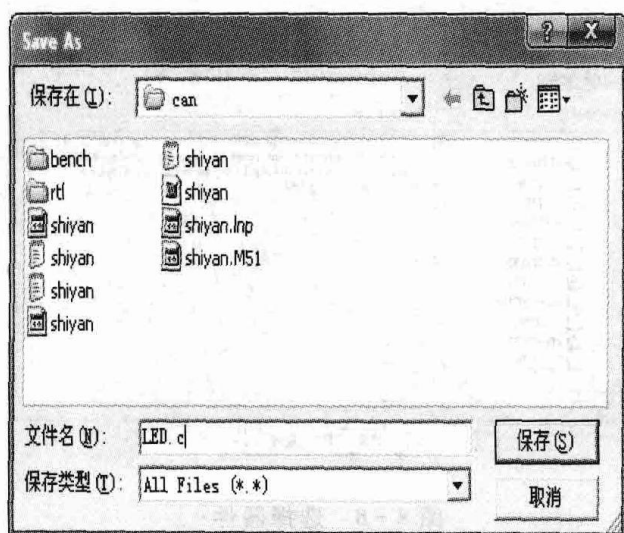


图 8-10 新建工程

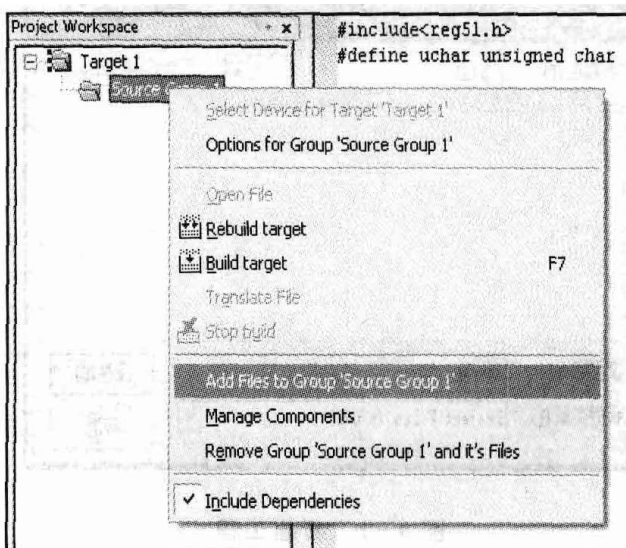


图 8-11 选择器件

4) 编译/链接 Shiyan

选择 Project→Build Target 菜单项,则 μ Vision2 开始编译和链接源文件,并创建一个可以载入到 μ Vision2 调试器调试的绝对目标模块。编译结果列在 Output Window 的 Build 选项卡上,如图 8-12 所示。

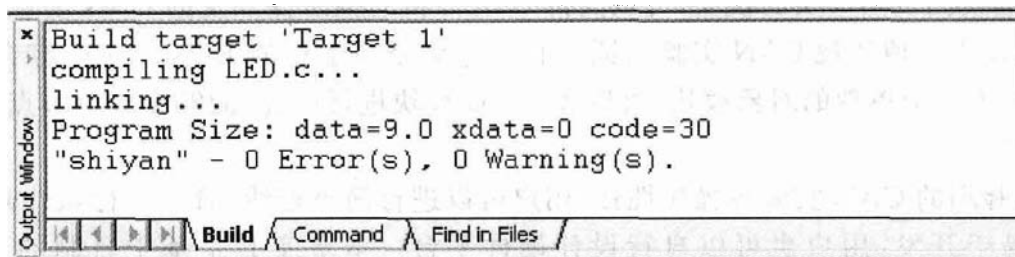


图 8-12 shiyan 工程的编译结果

5) 调试 Shiyan

LED.c 程序被编译和链接后,用 μ Vision2 调试器对它进行测试。选择 Debug 菜单项或单击工具栏 Start/Stop Debug Session 按钮可以开始调试 Shiyan。打开外围器件菜单,观测 P1 口变化,如图 8-13 所示。

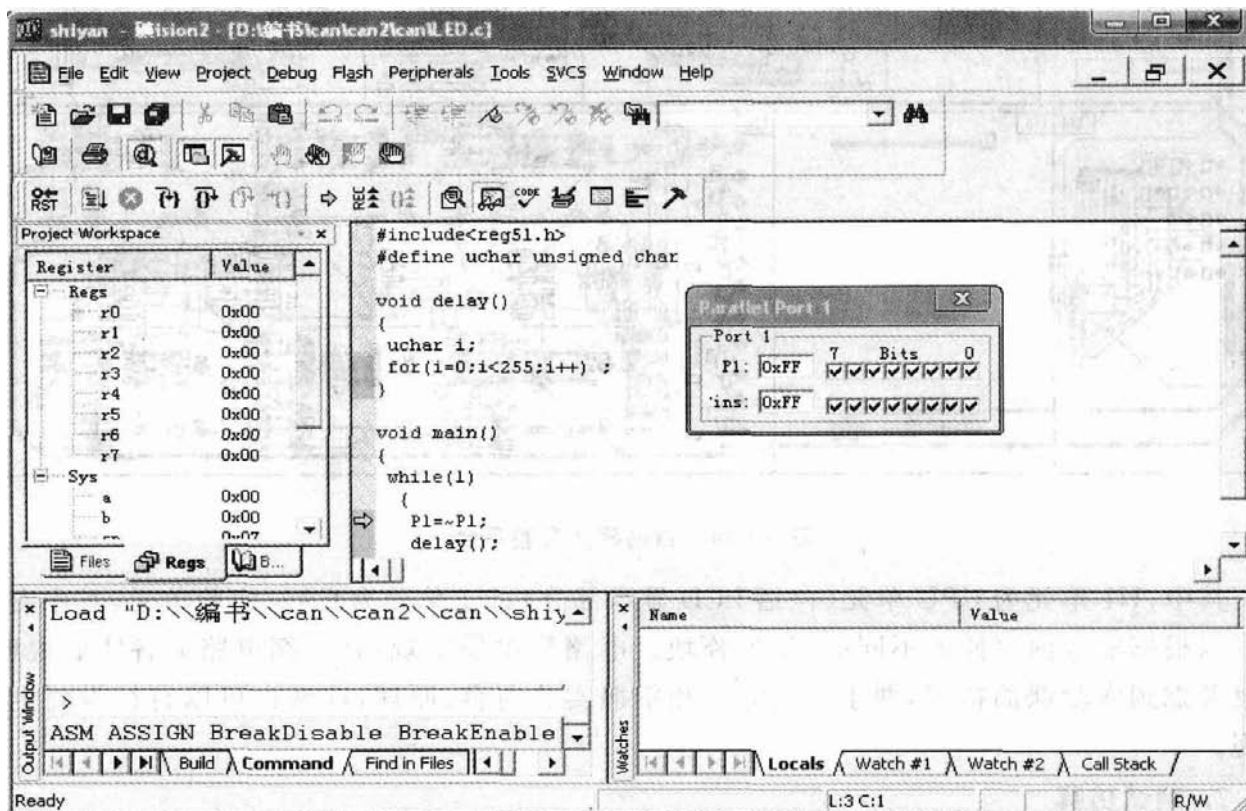


图 8-13 Shiyan 工程的编译结果输出

调试过程中可以用到单步、设置断点等调试手段,也可以通过观测窗口观测变量等。

8.1.2 硬件开发平台

1. 硬件结构

目前,一些单片机实验箱集成 CAN 实验模块,如周立功公司生产的 DP-51PRO 单片机综合实验仪集成了 CAN 实验模块,可以支持 C 语言和汇编语言开发的 CAN 实验,学生可以通过简单连线方便地实现 CAN 实验验证。北京达盛公司生产的 ELN-2000 综合实验台将 CAN 模块作为一个单独的对象模块,可以和 MCU 模块进行组合,能够完成单节点、多节点实验,简单方便。

基于实验箱的 CAN 实验可操作性强,用户可以进行简单连线,着重进行软件开发。由于 CAN 实验易于开发,用户也可以自行设计硬件平台。本章实验是基于自制的实验板,如图 8-14 所示。

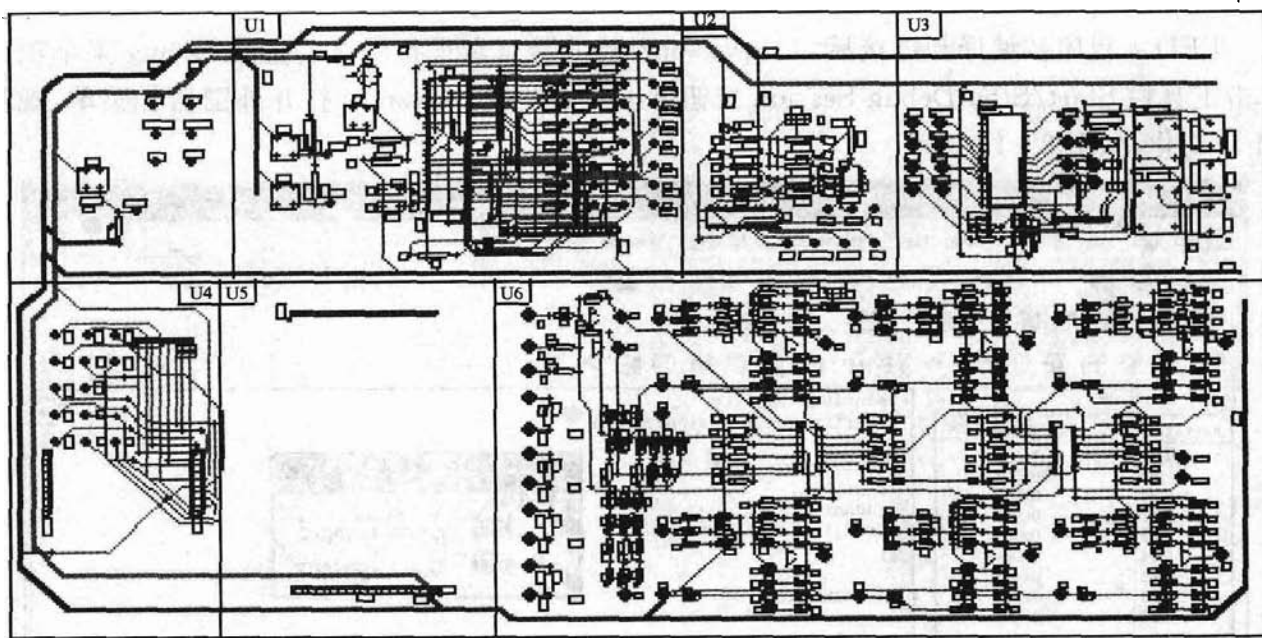


图 8-14 自制硬件实验平台

其中,U1 单元为 CPU 单元(包括 LCD 显示部分),U4 单元为 CAN 实验单元,其中有插槽可以根据实验内容插入不同的 CAN 模块。电路简单易实现,其详细电路见各实验说明。这里考虑到实验课的特点,便于学生完成和掌握实验内容、原理,读者也可以自行设计电路实现。

2. 调试仿真

使用监控程序进行调试仿真。Keil C 自带 Keil Monitor-51 驱动程序,可将该监控程序下载到 51 系列的芯片上实现实时在线调试。实验中选择 P89V51RD2FN 芯片进行在线调试

仿真,下面做一个简单的介绍。

(1) P89V51RD2FN 概述

P89V51RB2/RC2/RD2 是一款 80C51 微控制器,包含 16/32/64 KB Flash 和 1 024 字节的数据 RAM。P89V51RB2/RC2/RD2 的典型特性是它的时钟频率具有 2 种选择方式。利用该特性,设计工程师可使应用程序以传统的 80C51 时钟频率(每个机器周期包含 12 个时钟)或 X2 方式(每个机器周期包含 6 个时钟)的时钟频率运行,选择 X2 方式可在相同时钟频率下获得 2 倍的吞吐量。从该特性获益的另一种方法是将时钟频率减半而保持特性不变,这样可以极大地降低电磁干扰(EMI)。

Flash 程序存储器支持并行和串行在系统编程(ISP)。并行编程方式提供了高速的分组编程(页编程)方式,可节省编程成本和上市时间。ISP 允许在软件控制下对成品中的器件进行重复编程。应用固件的产生/更新能力实现了 ISP 的大范围应用。P89V51RB2/RC2/RD2 也可采用在应用中编程(IAP),允许随时对 Flash 程序存储器重新配置,即使是应用程序正在运行也不例外。

(2) P89V51RD2EN 在线下载 Bootloader 程序

1) P89V51RD 单片机 ISP 典型电路

P89V51RD 单片机 ISP 典型电路如图 8-15 所示。

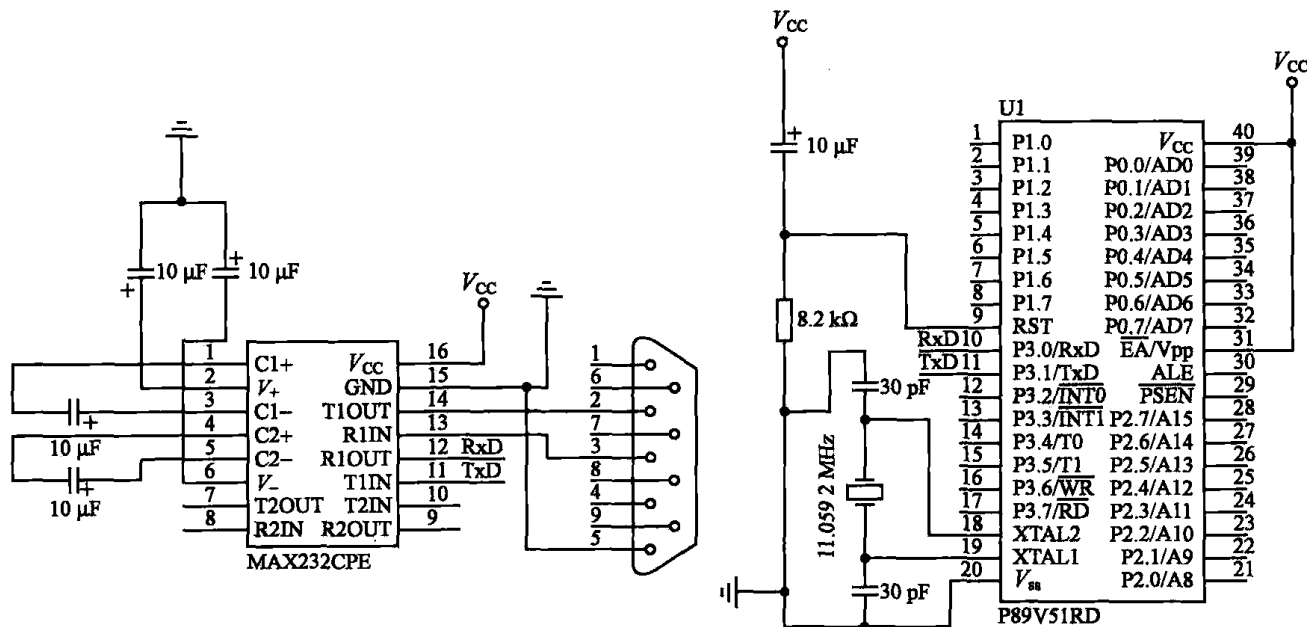


图 8-15 P89V51RD 单片机 ISP 典型电路

电路中要注意几个要点: V_{CC} 是电源,必须保证稳定可靠;EA 引脚不可悬空,必须连到 V_{CC} ,或者通过上拉电阻接到 V_{CC} ;复位电路建议采用传统的 RC 复位;不可接看门狗复位电路,否则,在 ISP 期间会引起复位,导致下载失败;晶振使用 11.059 2 MHz 或 22.118 4 MHz;

CAN 总线技术

PSEN/ALE 引脚悬空处理。

2) ISP 驱动程序 Flash Magic

Flash Magic 是支持众多 NXP 单片机 ISP 下载的驱动程序,其中就包括对 P89(L) V51RB2、RC2、RD2 的支持,该软件可到相关网站上下载安装。

3) ISP 操作步骤

② 连接串行口电缆

用 RS232 串行电缆将 PC 机与目标板连接起来,注意用带有真正串行口的 PC 机;如果是 USB 虚拟的串口,则可能出现下载失败的情况。

③ 运行 Flash Magic 软件

第一步:

COM Port: 选择实际使用的串行口,通常为 COM1;

Baud Rate: 波特率不可设置得过高,推荐用 9600 或 19200;

Device: 请选择正确的型号;

Interface: 选择 None(ISP)。

设置界面如图 8-16 所示。

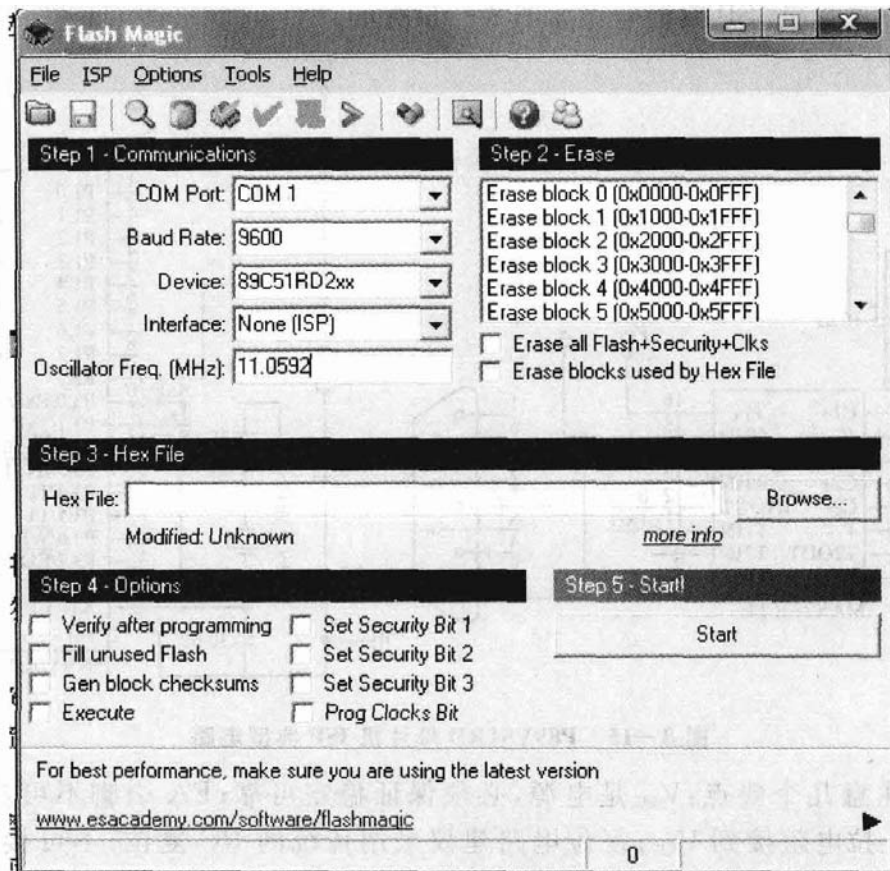


图 8-16 Flash Magic 设置界面

第二步:

选择 ISP→Start Bootloader 菜单项或单击快捷键 , 则出现如图 8-17 所示界面。

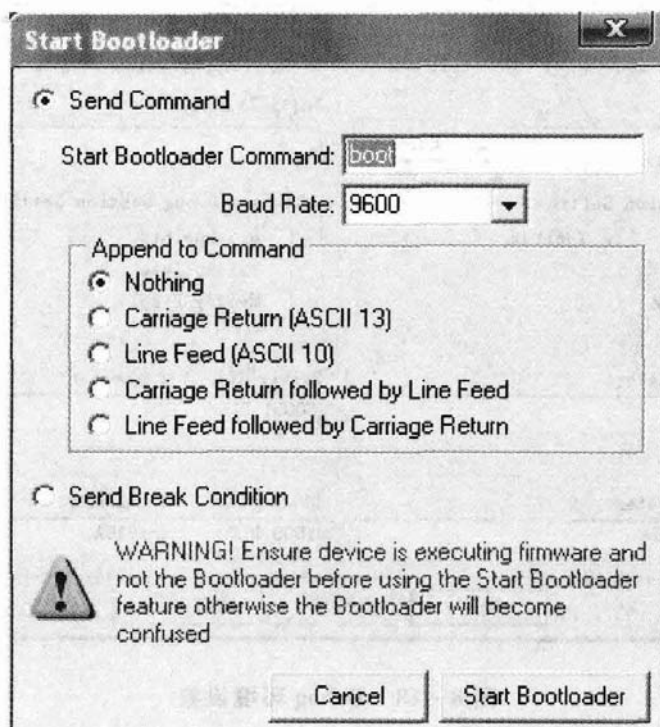


图 8-17 Start Bootloader 界面

单击 Start Bootloader 进行 boot 程序的下载。

第三步:

ISP 下载结束后,按一次复位键或者重新上电程序即开始运行。

下面介绍一下调试前准备工作。

④ 硬件环境

- ① 用通用 RS232 串口线将 PC 串口 (COM1 或 COM2) 机和实验箱串口相连;
- ② 打开实验板工作电源,进入调试准备。

⑤ 软件调试环境

① Debug 环境设置。进入 Keil C51 集成开发环境,选择 Project→Option for target 'target 1' 菜单项,在弹出的对话框中选择 Debug 选项,则弹出如图 8-18 所示的调试环境设置界面。在 Use 下拉列表框中选择 Keil Monitor-51 Driver,且选择其下两项。

② 通信环境设置。单击 **Settings** 进行波特率设置,如图 8-19 所示。此时和 PC 机还未建立起通信,在界面下提示“Monitor-51 not connected!”。

③ 其他环境设置:按默认值进行设置即可。

CAN 总线技术

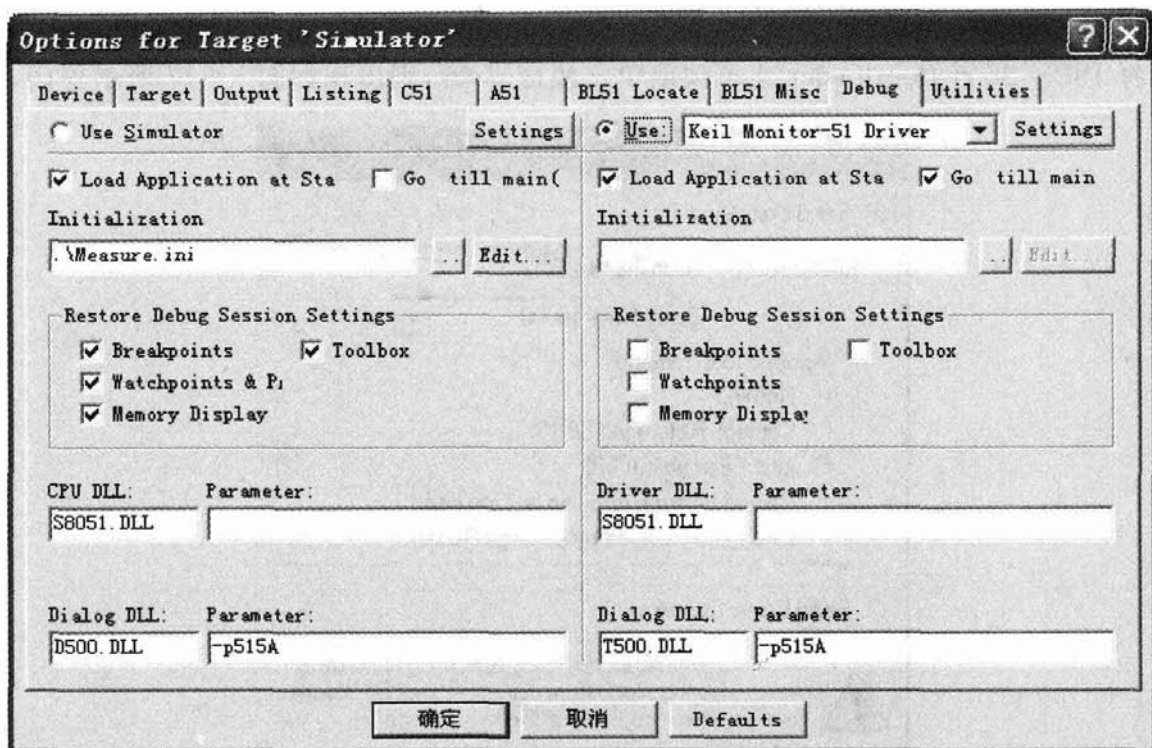


图 8-18 Debug 环境设置

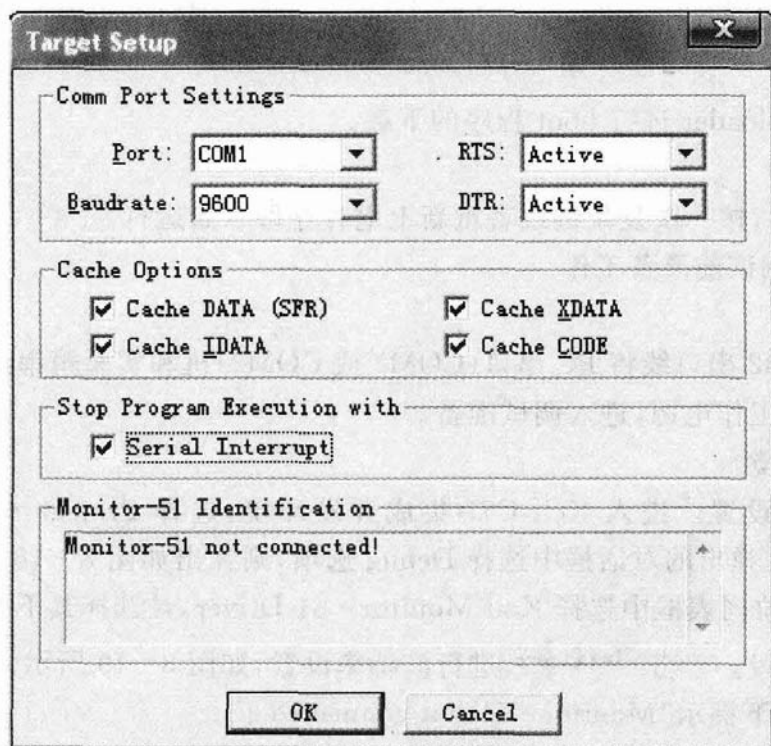


图 8-19 通信环境设置

④ 单击 Debug, 将程序下载到目标芯片上, 如图 8-20 所示, 则左下侧进度条显示下载进度。

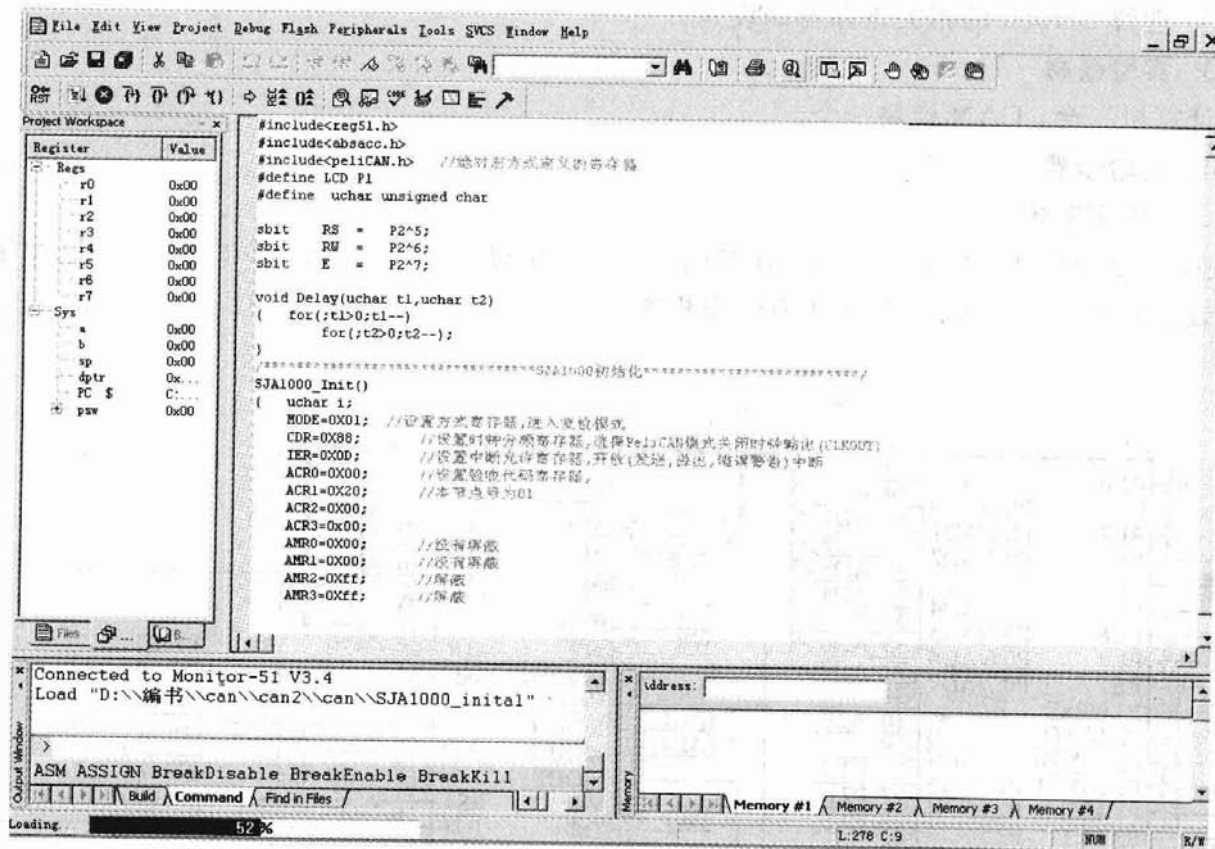


图 8-20 通信环境设置

程序下载后就可以进行调试了, 调试的方法有: 单步调试、 宏单步调试(每个子程序按照一步完成)、 全速运行, 程序复位。在调试的过程中可以监控各个变量值和内存单元。

8.2 课内实验

实验一 SJA1000 初始化实验

1. 实验要求

正确完成对 SJA1000 初始化, 初始化成功后用 LCD 显示出“SUCCESS”成功信号; 否则, 显示“ERROR”。

① 实现 SJ1000 的初始化设置。

② 理解 SJ1000 的相关寄存器的设置。

计算机一台，CAN 模块一个。

(1) 硬件设计

图 8-21 是 89C51 与 SJA1000 连接图。MCU 与 SJA1000 通过地 P0 口实现地址/数据复用。读、写、地址锁存依次与 51 相应引脚相连。

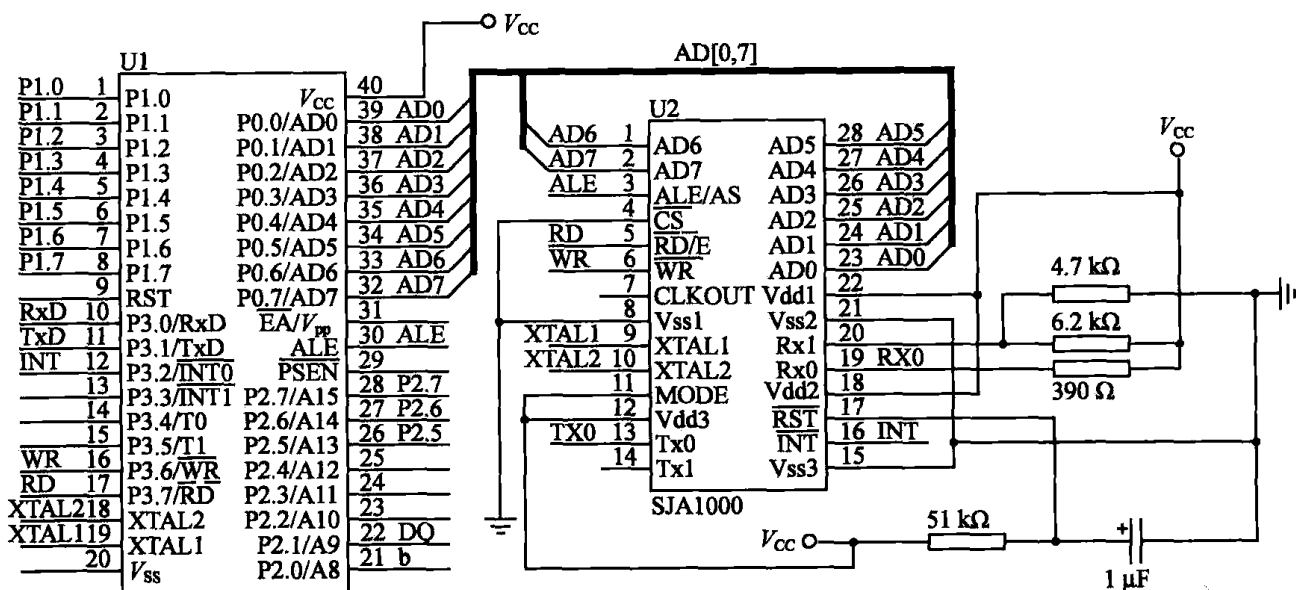


图 8-21 实验 1 硬件连接图

设计时应注意以下问题:

① SJA1000 工作模式的选择。由于 89C51 支持 Intel 模式,所以 SJA1000 的 MODE 引脚应接高电平。

② 应该注意 SJA1000 的复位引脚的接法。SJA1000 正常工作前,只有通过复位引脚对其进行可靠的硬件复位,才能对 SJA1000 中的寄存器进行正确的读/写操作。使 SJA1000 可靠复位的电平持续最小时间为 $0.1\ \mu\text{s}$,经常会出现微控制器的上电复位时间和 SJA1000 的复位时间的偏差,导致芯片不能可靠复位从而无法正常工作。这里介绍 3 种常用的方法:上电复位、I/O 复位以及按键复位方式。

1) 上电复位

选择适合的电阻和电容。此实验选择了 $51\text{ k}\Omega$ 电阻与 $1\text{ }\mu\text{F}$ 电容,开机后给电容充电,电

容电压由 0 V 升至 5 V, SJA1000 可靠复位。

2) I/O 复位

如图 8-22 所示, 由单片机某一 I/O 引脚控制 SJA1000 复位引脚, 使单片机在可靠复位之后完成对 SJA1000 的复位, 避免时间偏差。

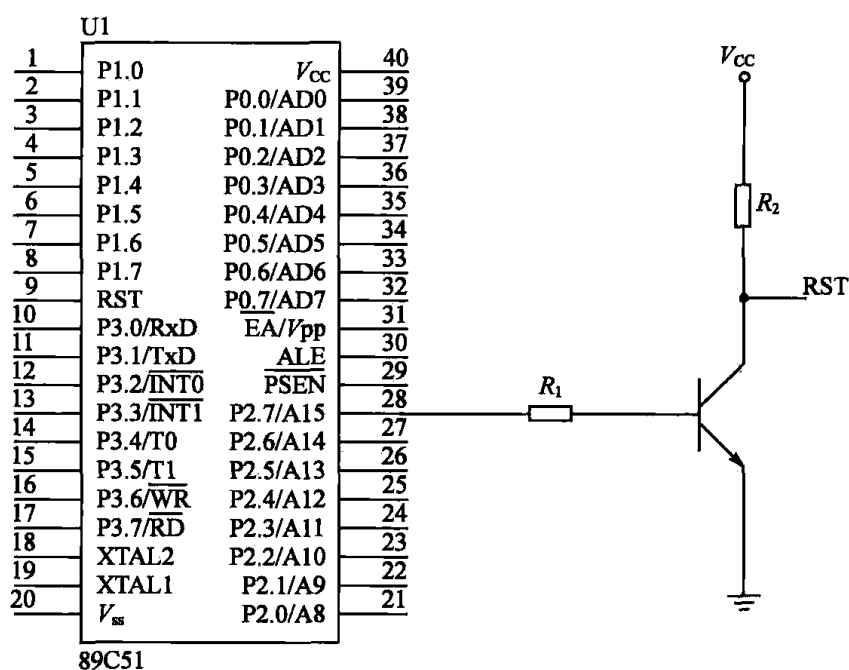


图 8-22 I/O 复位电路

3) 芯片复位

可以通过外围芯片进行硬件复位, 如看门狗芯片 MAX705。图 8-23 为 MAX705 控制的复位电路。

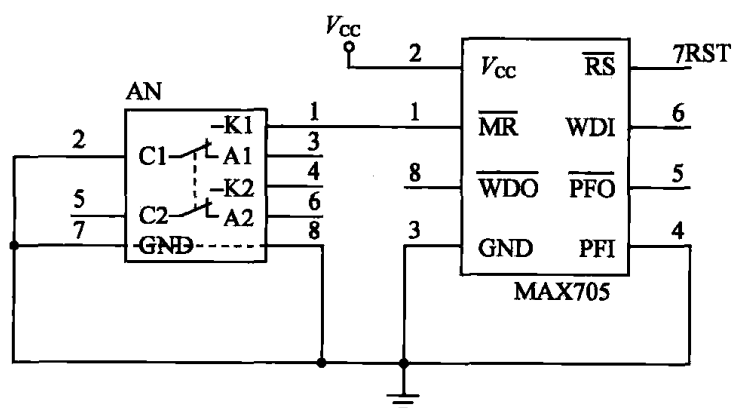


图 8-23 芯片复位电路

CAN 总线技术

(2) 软件设计

独立的 CAN 控制器 SJA1000 必须在上电或硬件复位以后设置。在复位模式中,主控制器必须配置以下寄存器(以 PeliCAN 为例),选择合适的配置:

① 模式寄存器配置:

选择验收滤波器模式;

选择自测模式;

选择只听模式。

② 时钟分频器配置:

选择 BasicCAN 或 PeliCAN 模式;

选择是否使能 CLKOUT 引脚;

选择是否旁路输入比较器。

③ 验收滤波器设置:

定义接收代码寄存器(本节点的标识码);

定义与代码寄存器相对应的验收滤波器的位值(是否屏蔽)。

④ 总线时序寄存器:

定义总线位速率;

定义采样点及采样速率。

⑤ 输出寄存器:

定义输出引脚输出模式(正常、时钟、双向和测试输出模式等);

定义输出引脚配置(悬空、上拉、下拉、推挽、极性)。

读者可以根据实际情况进行其他寄存器设置。本实验的流程如图 8-24 所示。

5. 调 试

① 编写源代码:新建工程 SJA1000_intial,新建文件 SJA1000_intial.c、PeliCAN.h 等文件,并加入工程中。选择 P80/87C51X2,如图 8-25 所示。

② 编译:对工程进行编译,通过后进行调试。

③ 调试:如前所述,选择 project→Debug 选项为硬件仿真(使用 Keil Monitor-51 driver),并进行相关设置。单击 Debug,将程序下载到目标芯片上。

打开存储器窗口,在“Adress:”文本框中输入 SJA1000 的基址 0x6000,单击  进行单步调试,注意观测外部存储区 0x6000(CAN 基址)后的存储内容,如图 8-26 所示。

6. 思 考

① 正常情况下状态寄存器的值为多少?

② 退出复位模式后,能否改写状态、模式、总线时序、输出寄存器等值?

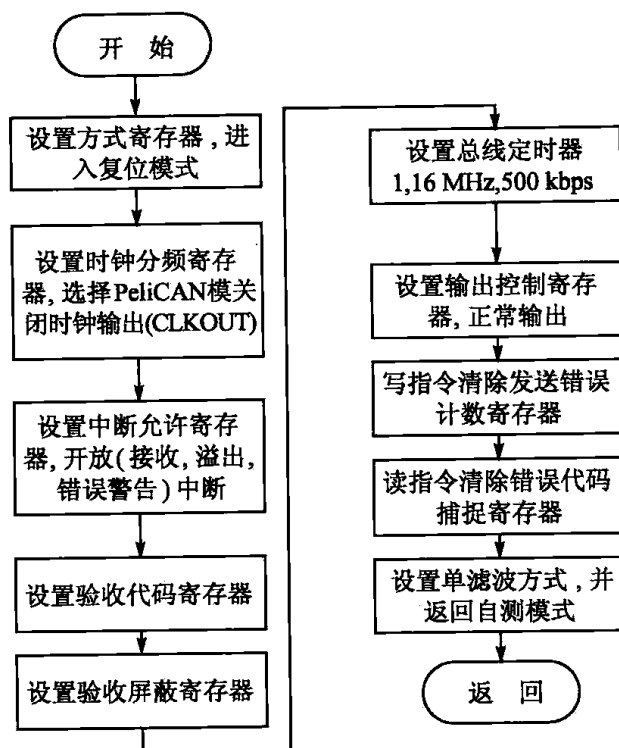


图 8-24 初始化流程图

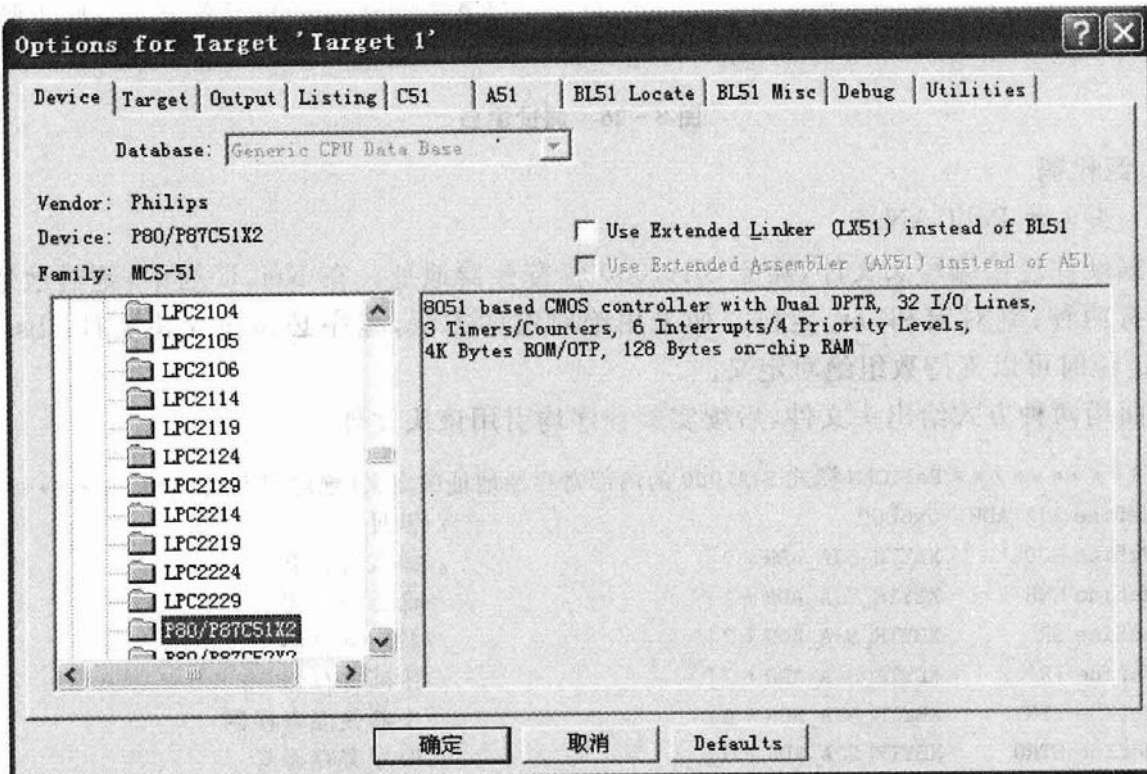


图 8-25 器件选择

CAN 总线技术

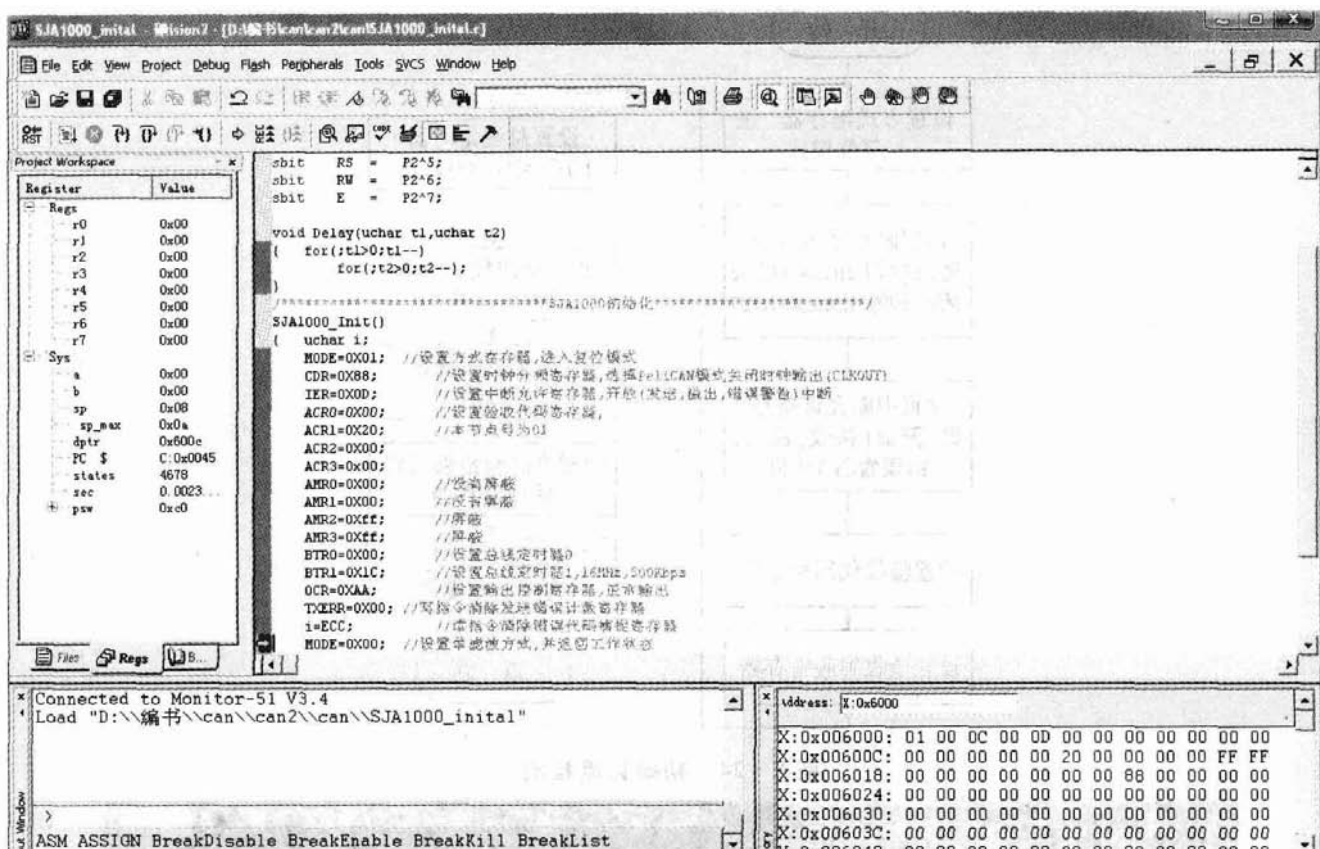


图 8-26 调试窗口

7. 源代码

(1) 头文件 PeliCAN.h

根据硬件连接编写头文件,确定 SJA1000 各寄存器地址。在 Keil C 当中,绝对地址的访问方式有两种:绝对宏和_at_定位。如果用绝对宏的方法,程序必须包含头文件 absacc.h,而_at_定位时可以支持数组绝对定义。

下面用两种方式给出头文件,后续实验程序均引用该头文件。

```

/***** PeliCAN 模式 SJA1000 的内部寄存器地址的定义(绝对宏方式) *****/
#define SJA_ADR 0x6000 //基址
#define MODE XBYTE[SJA_ADR + 0] //模式寄存器
#define CMR XBYTE[SJA_ADR + 1] //命令寄存器
#define SR XBYTE[SJA_ADR + 2] //状态寄存器
#define IR XBYTE[SJA_ADR + 3] //中断寄存器
#define IER XBYTE[SJA_ADR + 4] //中断使能寄存器
#define BTR0 XBYTE[SJA_ADR + 6] //时序寄存器 0
#define BTR1 XBYTE[SJA_ADR + 7] //时序寄存器 1
#define OCR XBYTE[SJA_ADR + 8] //输出控制寄存器

```

```

#define TEST      XBYTE[SJA_ADR + 9]           //测试寄存器
#define ALC        XBYTE[SJA_ADR + 11]         //仲裁丢失捕捉寄存器
#define ECC        XBYTE[SJA_ADR + 12]         //错误代码捕捉寄存器
#define ELWR       XBYTE[SJA_ADR + 13]         //错误报警限额寄存器
#define RXERR      XBYTE[SJA_ADR + 14]         //RX 错误计数器
#define TXERR      XBYTE[SJA_ADR + 15]         //TX 错误计数器
#define TXSFF      XBYTE[SJA_ADR + 16]         //TX 报文缓冲区(操作模式下)
#define TXID1      XBYTE[SJA_ADR + 17]         //TX ID1      (操作模式下)
#define TXID2      XBYTE[SJA_ADR + 18]         //TX ID2(操作模式下)
#define TXDATA1    XBYTE[SJA_ADR + 19]         //TX DATA1(操作模式下)
#define RXDATA1    XBYTE[SJA_ADR + 19]         //RX DATA1(操作模式下)
#define ACRO       XBYTE[SJA_ADR + 16]         //验收代码寄存器(复位模式下)
#define ACR1       XBYTE[SJA_ADR + 17]         //验收代码寄存器(复位模式下)
#define ACR2       XBYTE[SJA_ADR + 18]         //验收代码寄存器(复位模式下)
#define ACR3       XBYTE[SJA_ADR + 19]         //验收代码寄存器(复位模式下)
#define AMR0       XBYTE[SJA_ADR + 20]         //验收屏蔽寄存器
#define AMR1       XBYTE[SJA_ADR + 21]         //验收屏蔽寄存器
#define AMR2       XBYTE[SJA_ADR + 22]         //验收屏蔽寄存器
#define AMR3       XBYTE[SJA_ADR + 23]         //验收屏蔽寄存器
#define CDR        XBYTE[SJA_ADR + 31]         //时钟分频器

```

/* ***** Pelican 模式 SJA1000 的内部寄存器地址的定义(_at_定义) ***** */

```

unsigned char xdata MODE _at_ 0x6000;
unsigned char xdata CMR _at_ 0x6001;
unsigned char xdata SR _at_ 0x6002;
unsigned char xdata IR _at_ 0x6003;
unsigned char xdata IER _at_ 0x6004;
unsigned char xdata BTR0 _at_ 0x6006;
unsigned char xdata BTR1 _at_ 0x6007;
unsigned char xdata OCR _at_ 0x6008;
unsigned char xdata TEST _at_ 0x6009;
unsigned char xdata ALC _at_ 0x600B;
unsigned char xdata ECC _at_ 0x600C;
unsigned char xdata ELWR _at_ 0x600D;
unsigned char xdata RXERR _at_ 0x600E;
unsigned char xdata TXERR _at_ 0x600F;
unsigned char xdata TXB[13] _at_ 0x6010;
unsigned char xdata RXB[13] _at_ 0x6010;
unsigned char xdata ACR[4] _at_ 0x6010;
unsigned char xdata AMR[4] _at_ 0x6014;
unsigned char xdata CDR _at_ 0x601F;

```

代码见附录 A。

实验二 SJA1000 局部自检测实验

进行单节点自测实验,要求 CAN 节点自发自收标准数据帧(数据可自定义),并将所收数据显示到 LCD 上,观察与发送数据是否一致。

① 实现 CAN 节点的初始化及 CAN 节点的自发自收。

② 学会对 CAN 节点的基本操作,理解 CAN 通信的基本流程。

计算机一台,CAN 模块一个。

(1) 硬件设计

CAN 网络进行通信时,发送器需要得到接收器的应答信号,但是,节点局部自检测时不需要其他节点配合,接终端电阻即可。实验采用中断接收,将 SJA1000 的中断引脚与单片机的外部中断 0 相连。本实验选择 $120\ \Omega$ 的终端电阻,具体电路如图 8-27 所示。

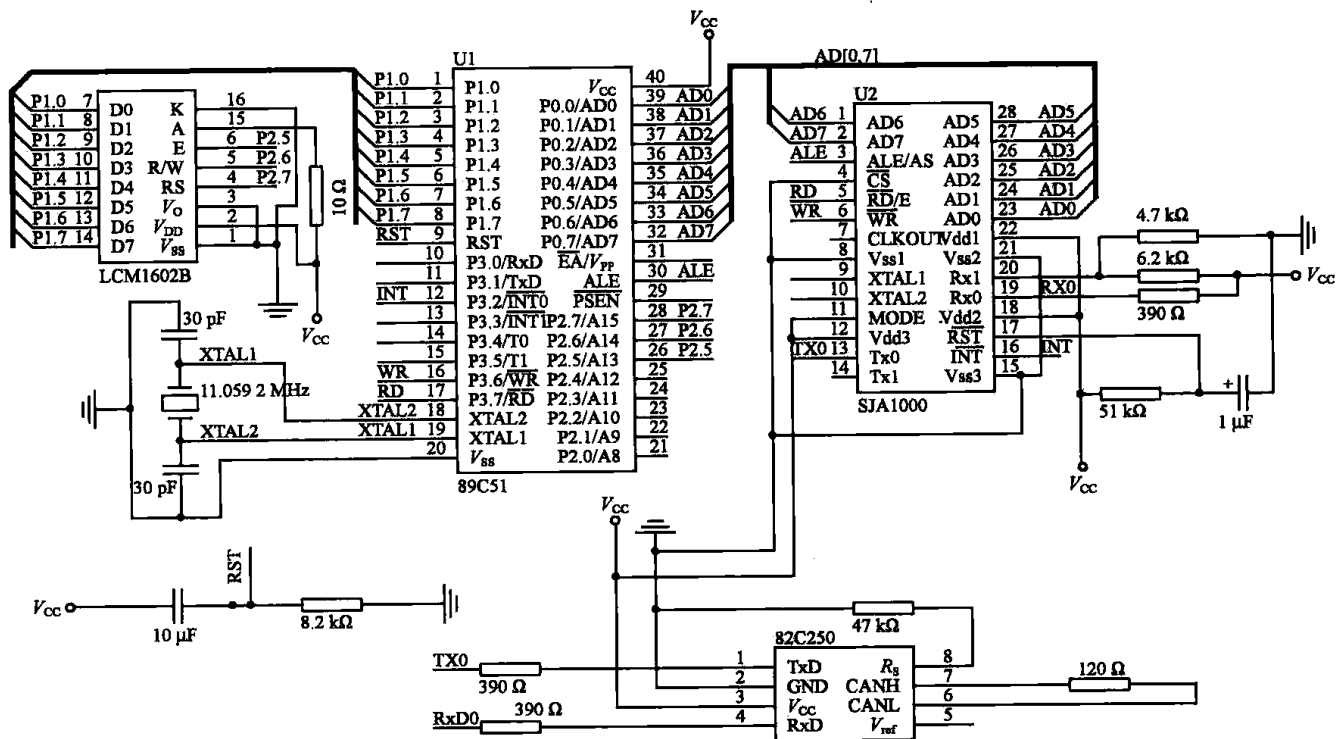


图 8-27 SJA1000 自检测硬件连接图

(2) 软件设计

这里采用分模块设计的方法,由主模块、发送模块、中断接收模块、显示模块等构成。主程序流程如图 8-28 所示。

5. 调 试

① 新建工程,编辑并添加文件,编译,进入调试状态。

② 由于实验程序较复杂,调试时可采用分模块调试的方法。

a. SJA1000 初始化模块调试,参见实验一,注意观察状态寄存器值。

b. 发送模块调试:单击  进行单步调试,如图 8-29 所示。

在发送命令执行语句行设置断点,注意观测程序运行到断点处时,如果发送缓冲器的值分别为“08 00 20 00 01...07”,则说明单片机对 SJA1000 发送缓冲器的写访问正常,数据已经写入缓冲器内。如果此步出现问题,观测状态寄存器,检查 CPU 与 SJA1000 数据地址线的连接,查找问题。

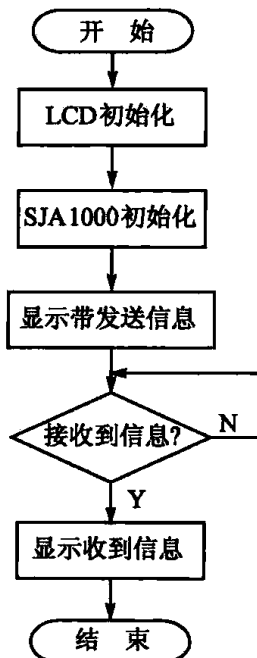


图 8-28 SJA1000 自测主程序流程图

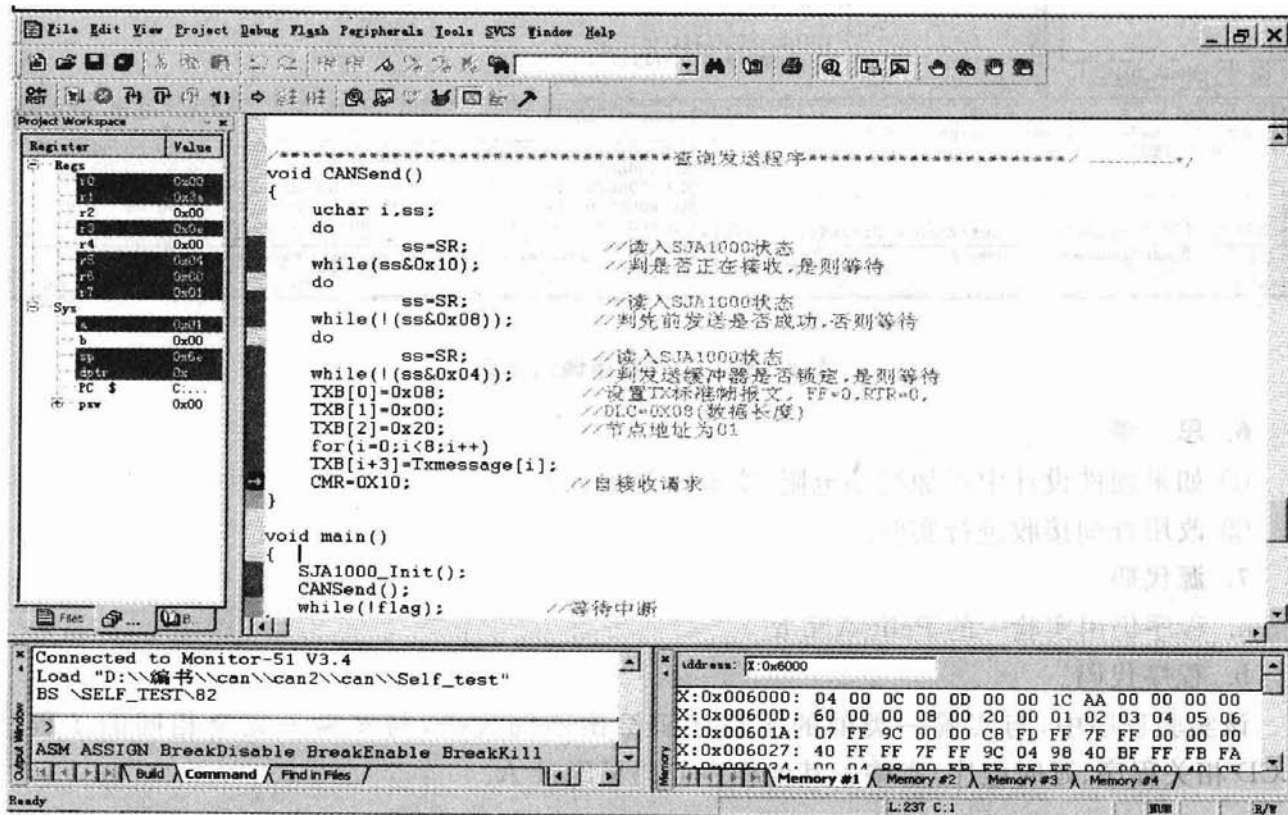


图 8-29 发送程序调试窗口

CAN 总线技术

c. 接收模块调试。可在接收中断处设置断点,启动“发送”命令后单击  全速运行,进入断点后注意观察状态寄存器值若变为 0x0D,则表示接收缓冲区有有效报文。此时,注意观察接收缓冲器里的数值实验是否与发送的相符,如图 8-30 所示。

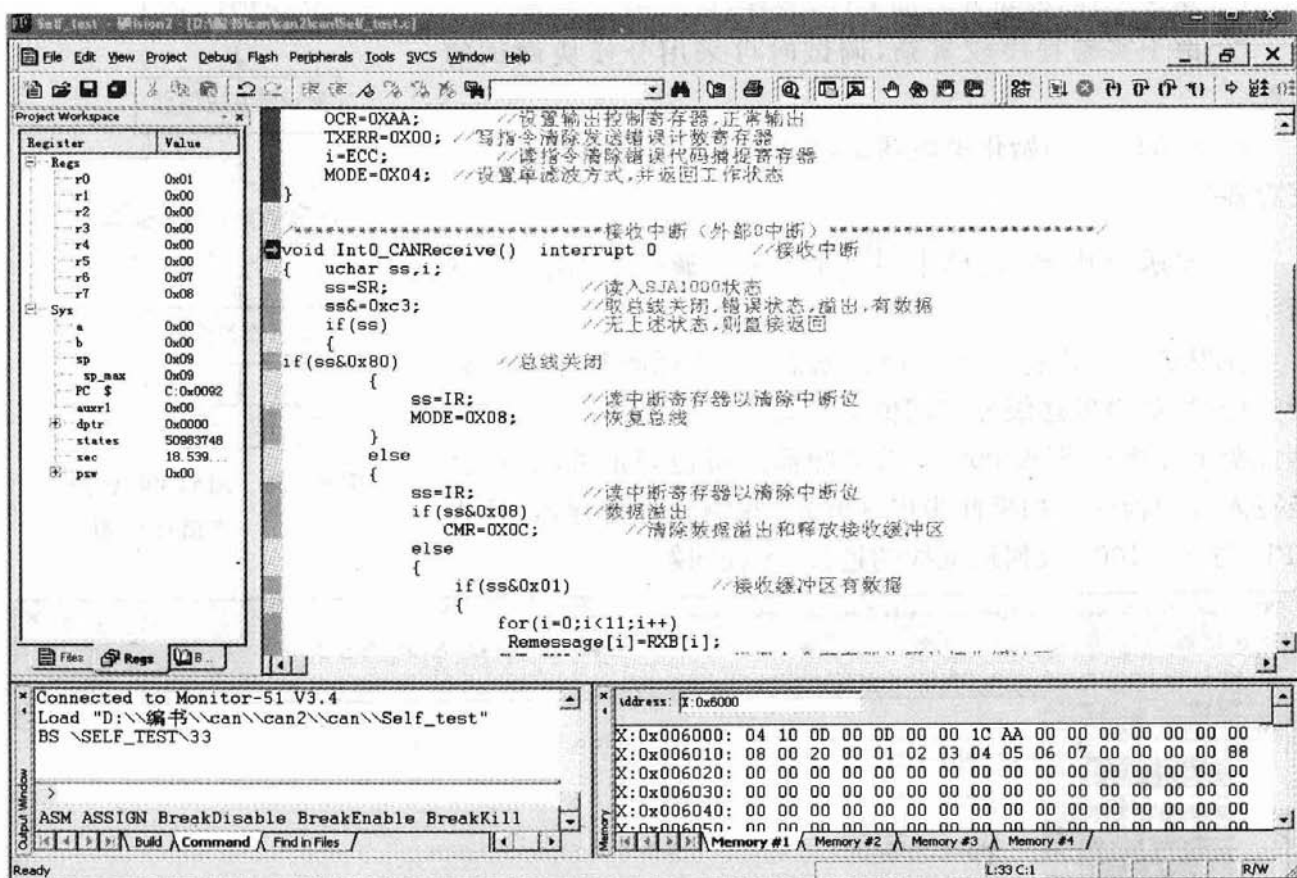


图 8-30 接收程序调试窗口

6. 思考

- ① 如果硬件设计中不加终端电阻,实验结果如何?
- ② 改用查询接收进行实验。

7. 源代码

- a. 程序仍用实验一的 PeliCAN.h。
- b. 程序代码。

该实验程序中,与实验一类似的子程序只给出不同代码,与实验一完全相同的子程序(LCD 相关程序、延时程序)省略。其他程序代码见附录 A。

实验三 P8xC591 双节点通信实验

1. 实验要求

选 P8xC591 为主要芯片构成 CAN 节点 A (ID:0x00)、B (ID:0x01), 要求 A、B 节点可以进行通信 (标准数据帧), 并将接收到的数据进行显示。

2. 实验目的

- ① 熟悉带 CAN 控制器芯片的结构、CAN 通信原理。
- ② 掌握双节点通信的原理及设计。

3. 实验器材

计算机一台, TKS 仿真器一个, P8xC591 节点模块二个。

4. 实验步骤

(1) 硬件链接

由于 P87C591 内部含有 SJA1000 控制器, 所以硬件连接时只要外接光偶、驱动器和外围接口元件即可。图 8-31 为双节点通信时单节点的硬件结构。

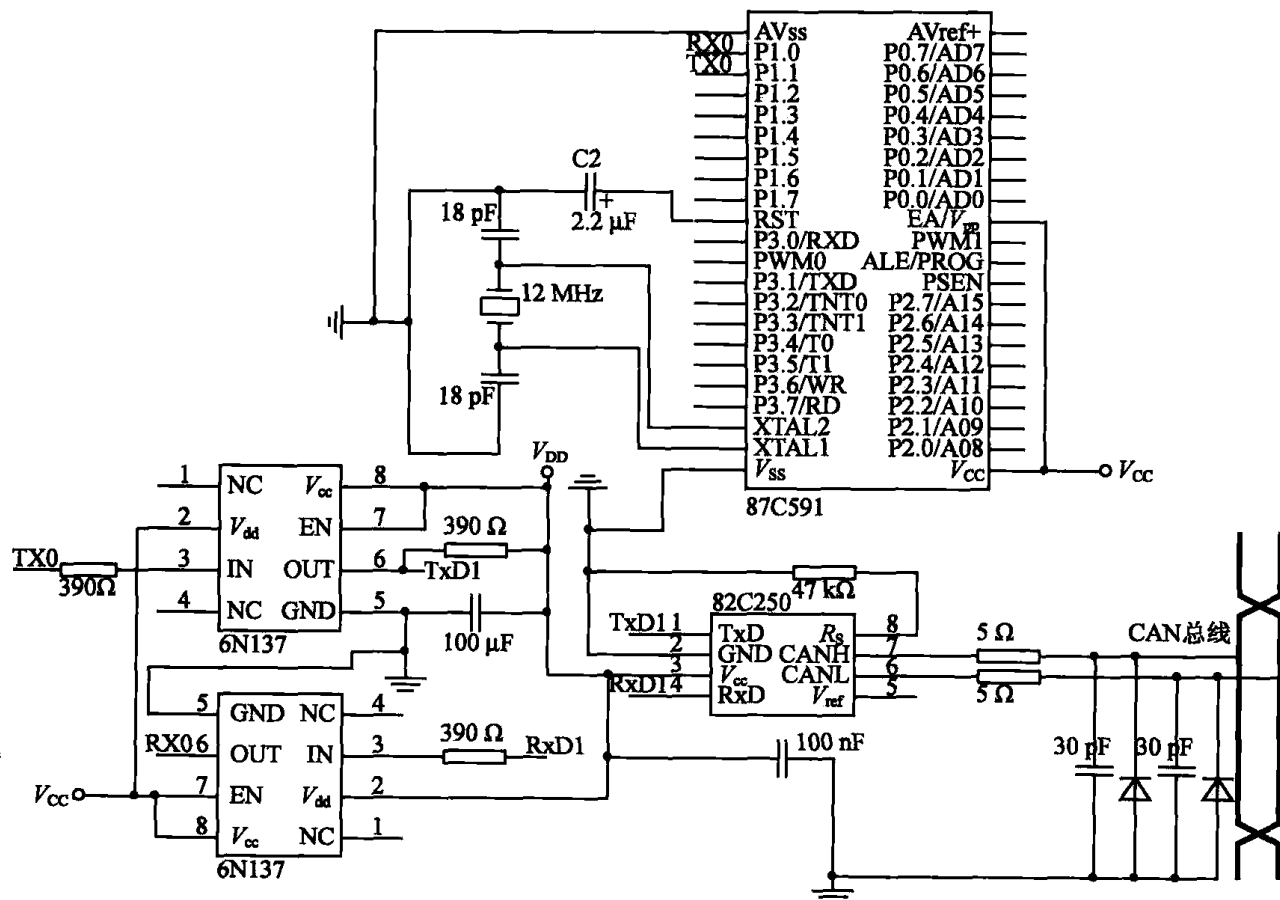


图 8-31 节点通信电路图

CAN 总线技术

(2) 软件设计

1) 软件流程

两个节点程序类似,每个节点采用分模块方法设计程序,分为初始化模块、发送模块、接收模块和显示模块等。主程序流程如图 8-32 所示。

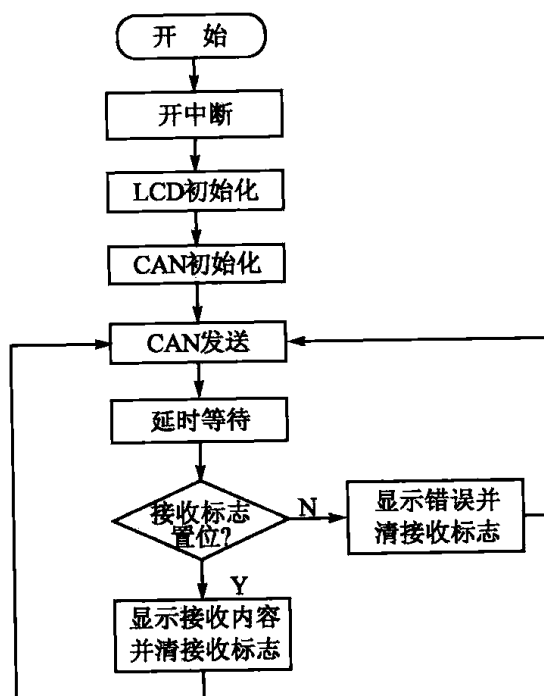


图 8-32 主流程图

2) 头文件编制

头文件编制包括 P87C591.h 和 can591reg.h 头文件的编制。

(a) P87C591.h 设计。用 keilC 扩展的特殊变量类型 sfr 进行各寄存器定义：

```

sfr P0 = 0x80;
sfr P1 = 0x90;
sfr P2 = 0xA0;
sfr P3 = 0xB0;
...
sfr CANMOD = 0xC4;
sfr CANCON = 0xC3;
sfr CANDAT = 0xC2;
sfr CANADR = 0xC1;
sfr CANSTA = 0xC0;
...
  
```

(b) can591reg.h 设计。与 peliCAN.h 编写类似,按照 CAN 地址对各个寄存器进行定义,这里只给出部分定义：

```

#define CAN_MOD      0x00      //内部控制寄存器
#define CAN_CMR      0x01      //命令寄存器
#define CAN_SR        0x02      //状态寄存器
#define CAN_IR        0x03      //中断寄存器
#define CAN_IER       0x04      //中断使能寄存器
#define CAN_RIL       0x05      //中断级寄存器
#define CAN_BTR0      0x06      //总线定时器 0 寄存器
#define CAN_BTR1      0x07      //总线定时器 1 寄存器
#define CAN_RMC        0x09      //RX 信息计数器寄存器
#define CAN_RBSA      0x0a      //RX 缓冲区起始地址 寄存器
#define CAN_ALC        0x0b      //仲裁丢失捕捉寄存器
#define CAN_ECC        0x0c      //错误捕捉寄存器
#define CAN_EWLR       0x0d      //错误报警限制寄存器
#define CAN_RXERR      0x0e      //RX 错误计数器寄存器
#define CAN_TXERR      0x0f      //TX 错误计数器寄存器
#define CAN_ACFMOD     0x1d      //验收滤波器模式寄存器
#define CAN_ACFEN      0x1e      //验收滤波器使能寄存器
#define CAN_ACFPRIO    0x1f      //验收滤波器优先级寄存器
#define CAN_ACF1_ACR0   0x20      //验收代码 0 寄存器
#define CAN_ACF1_ACR1   0x21      //验收代码 1 寄存器
#define CAN_ACF1_ACR2   0x22      //验收代码 2 寄存器
#define CAN_ACF1_ACR3   0x23      //验收代码 3 寄存器
...
#define CAN_RXBUF0     0x60      //RX 帧信息(标准帧、扩展帧)寄存器
...
#define CAN_RXBUF12    0x6c      //RX 帧信息(扩展帧数据 8)寄存器
...
#define CAN_TXB0       0x70      //TX 帧信息(标准帧、扩展帧)寄存器
...
#define CAN_TXBUF12    0x7c      //TX 帧信息(扩展帧数据 8)寄存器
...

```

5. 调 试

① 对每一节点新建工程,选择器件为 P87C591。编写、编译过程与前相同。

② 进入调试阶段。

本次实验采用 TKS 系列仿真器,该仿真器是广州致远电子有限公司推出的实时在线仿真器,硬件上采用了 NXP 授权的 HOOKS/Bondout 仿真技术并加以改进,支持大部分 80C51 系列单片机的实时在线仿真。

(1) 仿真器连接

仿真器电源为仿真器主机提供工作电源,仿真头带有 40 脚 DIP 针状插座,可以插入用户目标板的插座中;它的另外一端通过仿真电缆连接到仿真主机,这样把仿真主机和用户目标板

CAN 总线技术

连接起来。RS232 串口通信电缆用于连接仿真器主机和计算机。

(2) 仿真器驱动程序声明

将 TKS 系列仿真器驱动文件 TKS_DEB_B.DLL 复制到 Keil 的安装目录中(假如 Keil 的安装目录为 C:\Keil,则将 TKS_DEB_B.DLL 复制到 C:\Keil\C51\bin 目录下)。打开 C:\Keil 目录下的 Tools.ini 文件,在几个分类中找到“C51”类别加入下列描述:

```
TDRV5 = C:\Keil\C51\bin\TKS_DEB_B.DLL ("TKS Debugger B")
```

其中,TDRV5 是驱动 DLL 的序号。TKS-591B 仿真器仍保留了同 Keil 软件的接口,使用户可以在 Keil 平台上编译调试程序。

(3) 操作步骤

① 按(1)所述进行仿真器连线,并将仿真器插头插入目标板(实验板)。

② 确保仿真器内部为 P87C591 的仿真芯片,确保仿真器内跳线连接正确,确保仿真器驱动已安装完成,将 TKS591B 仿真器上电。

③ 在 Keil 编译环境打开所建工程,选择 Project→Options for Target 'Target 1'→Debug 菜单项,并选择硬件仿真,配置仿真器串口。配置时须选择[Bus Option]为[Use No Bus],设置晶振为“Internal OSC 20KHz-50MHz”,为 12 MHz。

④ 按照其他实验方式进行在线调试。

6. 思考

① 如果启用两组滤波器,一组定义如本实验并设为低优先级,另一组可自定义并设为高优先级,程序如何修改?

② 实验时,两个节点上电时间进行改变:同时上电、依次上电,观察实验结果是什么?

7. 参考代码

见附录 A。

实验四 CAN 转 RS232 网桥模块的设计

1. 实验要求

该实验完成 CAN 和 RS232 通信。CAN/RS232 模块将采集的 CAN 节点数据定时通过 RS232 串口向计算机发送(计算机通过“串口调试助手”接收)。同时,CAN/RS232 模块将接收到的数据(计算机通过“串口调试助手”发送)发送至 CAN 节点上,该节点将数据通过 LCD 显示。

2. 实验目的

① 了解 CAN/RS232 网桥模块的工作原理;

② 掌握模块的设计。

3. 实验器材

计算机一台(串口调试助手),CAN/RS232 模块一个,CAN 模块一个。

(1) 硬件设计

图 8-33 CAN/RS232 转换模块电路图

将计算机与模块通过串行线相连。计算机上运行“串口调试助手”，如图 8-34 所示。

在发送区域内将要发给 CAN 的数据按照 CAN 格式写入“08 00 20 11 22 33 44 55 66”，在接收区域等待接收 CAN 模块通过 CAN/RS232 模块上传的数据。

CAN 总线技术

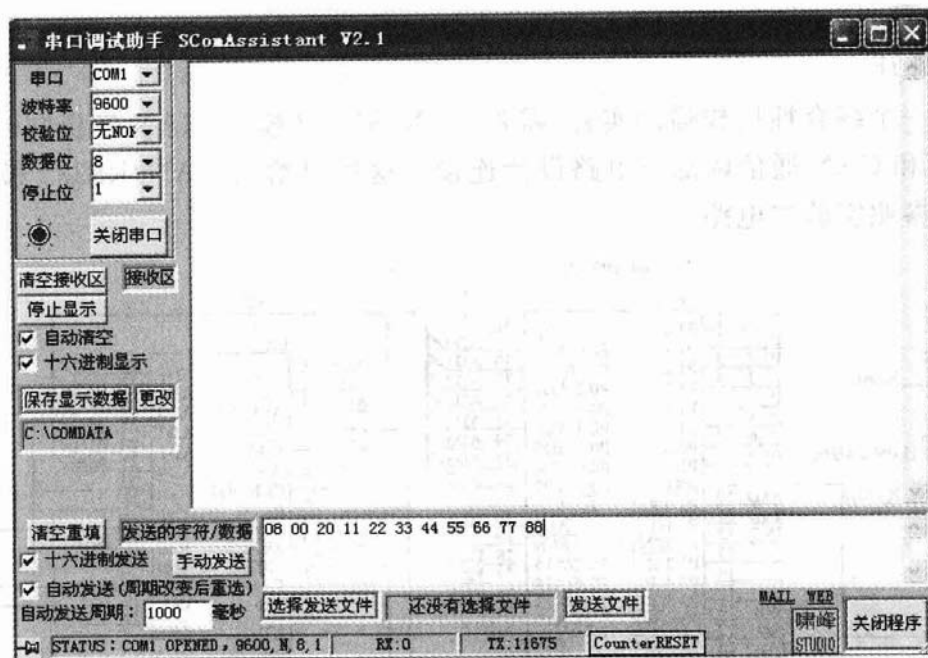


图 8-34 串口调试助手界面

(2) 软件设计

CAN/RS232 网桥模块起着连接 RS232 与 CAN 网络的作用。程序设计注意完成 RS232 通信与 CAN 通信的转换,即 RS232 数据信息、CAN 信息的组帧与转发。其中,CAN 接收、串口接收使用中断方式,发送使用查询方式。程序流程如图 8-35 所示。

5. 调 试

① 分别建立 CAN/RS232、CAN 节点两个工程,进行程序编写、编译和调试。

② 调试时注意 CAN/RS232 模块与计算机的连接,应选用交叉线。

6. 思 考

- ① 改用查询方式进行串口、CAN 接收,怎样修改程序?
- ② 若模块交换数据量比较大,如何进行数据存储?

7. 源代码

CAN/RS232 转换模块的程序代码(C 语言,汇编语言)见附录 A。注意,参考代码中假设 CAN 节点所采集数据为温度信息。

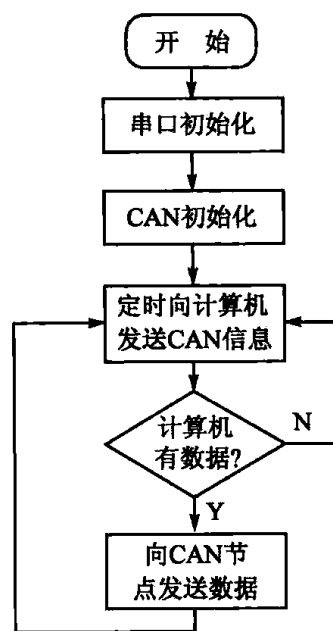


图 8-35 主程序流程图

实验五 CAN 中继器设计

1. 实验要求

设计以 89C52 与 SJA1000 为主要芯片的 CAN 中继器,能够连接两个 CAN 网络(由两路 CAN 节点代表)并进行数据转发,每个 CAN 节点将收到的数据进行显示。考虑实验的复杂度,可建议分 2 次(4 学时)完成。

2. 实验目的

- ① 掌握中继器的工作原理;
- ② 熟悉中继器的软硬件设计。

3. 实验器材

计算机一台,中继器模块一个,CAN 节点模块两个。

4. 实验步骤

(1) 硬件设计

中继器连接相同类型的两个网络,实验中中继器模块连接两个 CAN 节点模块(模拟两个 CAN 网络)。在设计中注意 MCU 对两路 CAN 控制器 SJA1000 的控制,将两个 SJA1000 的 $\overline{CS}$ 端分别连接到 89C51 的 P2.0 和 P2.1 口,使之对应于不同的地址(CAN1 基址 0x100H, CAN2 基址 0x200H),并且地址不能重叠,要进行可靠的片选。图 8-36 给出了两片 SJA1000 的通信电路,MCU 电路、CAN 节点模块电路设计与前述实验类似,这里不再赘述。

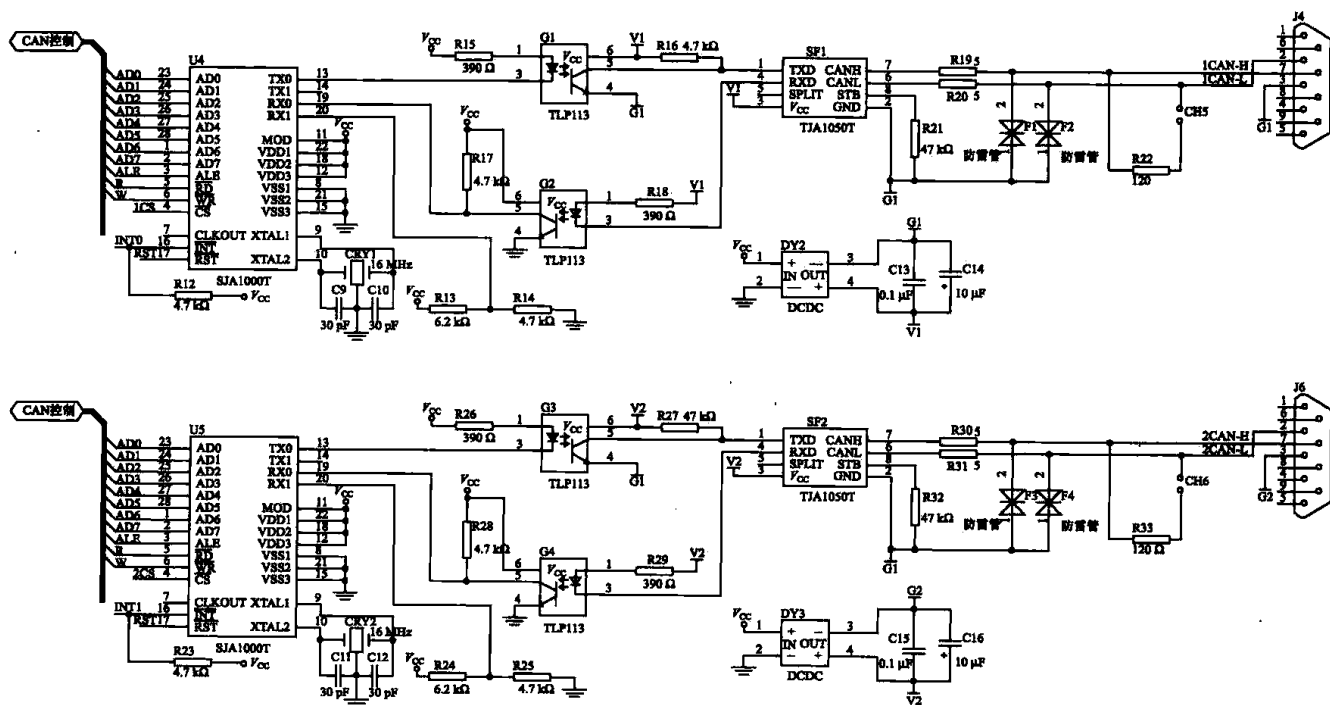


图 8-36 CAN 通信部分电路图

CAN 总线技术

(2) 软件设计

中继器模块的 MCU 通过控制两路 CAN 控制器(CAN1、CAN2)通信连接两个 CAN 网络,采用中断方式接收来自 CAN1、CAN2 的数据,将收到的数据存入数组(缓存区)并进行转发。主程序流程图如图 8-37 所示。

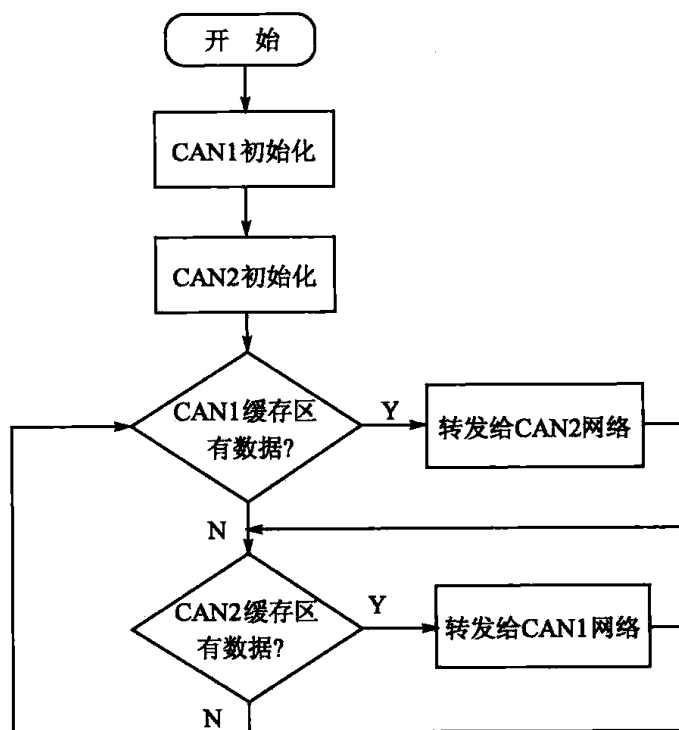


图 8-37 CAN 中继器主程序流程

5. 调 试

该实验调试比较复杂,应分模块调试。

- ① 建立 CAN1 节点工程,程序编写、编译、调试。
- ② 建立 CAN2 节点工程,程序编写、编译、调试。
- ③ 建立中继器模块工程,程序编写、编译。
 - (a) 进行中继器模块与 CAN1 节点的通信调试;
 - (b) 进行中继器模块与 CAN2 节点的通信调试。
- ④ 3 个模块进行联调。

6. 思 考

如果转发数据量比较大,采用什么方式存储数据?

7. 源代码

附录 A 只给出中继器模块程序,其他程序参见其他实验源代码。

附录 A

参考程序

实验一 SJA1000 初始化实验参考程序

```
#include < reg51.h >
#include < absacc.h >
#include < peliCAN.h >          //绝对宏方式定义的寄存器
#define LCD P1
#define uchar unsigned char
sbit RS = P2^5;
sbit RW = P2^6;
sbit E = P2^7;
void Delay(uchar t1,uchar t2)
{   for(;t1>0;t1--)
        for(;t2>0;t2--);
}
/***** SJA1000 初始化 *****/
SJA1000_Init()
{   uchar i;
    MODE = 0X01;          //设置方式寄存器,进入复位模式
    CDR = 0X88;           //设置时钟分频寄存器,选择 PeliCAN 模式关闭时钟输出(CLKOUT)
    IER = 0X0D;           //设置中断允许寄存器,开放(发送,溢出,错误警告)中断
    ACR0 = 0X00;          //设置验收代码寄存器
    ACR1 = 0X20;          //本节点号为 01
    ACR2 = 0X00;
    ACR3 = 0x00;
    AMR0 = 0X00;          //没有屏蔽
    AMR1 = 0X00;          //没有屏蔽
```

CAN 总线技术

```

    AMR2 = 0Xff;           //屏蔽
    AMR3 = 0Xff;           //屏蔽
    BTR0 = 0X00;           //设置总线定时器 0
    BTR1 = 0X1C;           //设置总线定时器 1,16MHz,500Kbps
    OCR = 0XAA;            //设置输出控制寄存器,正常输出
    TXERR = 0X00;          //写指令清除发送错误计数寄存器
    i = ECC;                //读指令清除错误代码捕捉寄存器
    MODE = 0X00;            //设置单滤波方式,并返回工作状态
}

/ ***** LCD 写数据 *****/
void Lcd_WD(uchar da)
{
    RS = 1;
    RW = 0;
    E = 1;
    LCD = da;
    Delay(1,100);
    E = 0;
}

/ ***** LCD 写命令 *****/
void Lcd_WC(uchar c)
{
    RS = 0;
    RW = 0;
    E = 1;
    LCD = c;
    Delay(1,100);
    E = 0;
}

/ ***** LCD 初始化 *****/
void Lcd_Init()
{
    Delay(15,125);
    Lcd_WC(0x38);
    Delay(5,125);
    Lcd_WC(0x38);
    Delay(5,125);
    Lcd_WC(0x38);
    Delay(5,125);

```

```

    Lcd_WC(0x08);
    Lcd_WC(0x01);
    Lcd_WC(0x06);
    Lcd_WC(0x0c);
}
/***** LCD 写字符串 *****/
void WriteString(uchar *s)
{
    uchar i;
    if(*s == '\0')           //遇到字符串结束
        return;
    for(i = 0;; i++)
    {
        if(*(s+i) == '\0')
            break;
        Lcd_WD(*(s+i));
    }
}
/***** 主程序 *****/
void main()
{
    uchar *s1 = "susecss";
    uchar *s2 = "error";
    Delay(255,255);           //延时等待 CAN 可靠复位
    Lcd_Init();
    SJA1000_Init();
    Lcd_WC(0x80);             //写入第一行
    if(SR == 0x0C) WriteString(s1);
    else WriteString(s2);
    while(1);
}

```

实验二 SJA1000 局部自检测实验参考程序

```

#include < reg51.h >
#include < peliCAN.h >       //绝对宏方式定义的寄存器
#define LCD P1
#define uchar unsigned char
uchar Remessage[11];

```

CAN 总线技术

```

uchar Txmessage[8] = {0,1,2,3,4,5,6,7};
uchar flag = 0;
/***** SJA1000 初始化 *****/
SJA1000_Init()
{
    uchar i;
    MODE = 0X01;           //设置方式寄存器,进入复位模式
    CDR = 0X88;            //设置时钟分频寄存器,选择 Pelican 模式关闭时钟输出(CLKOUT)
    IER = 0X0D;            //设置中断允许寄存器,开放(发送,溢出,错误警告)中断
    ACR[0] = 0X00;         //设置验收代码寄存器,
    ACR[1] = 0X20;         //本节点号为 01
    ACR[2] = 0X00;
    ACR[3] = 0x00;
    AMR[0] = 0X00;         //没有屏蔽
    AMR[1] = 0X00;         //没有屏蔽
    AMR[2] = 0Xff;         //屏蔽
    AMR[3] = 0Xff;         //屏蔽
    BTRO = 0X00;           //设置总线定时器 0
    BTR1 = 0X1C;           //设置总线定时器 1,16MHz,500Kbps
    OCR = 0XAA;            //设置输出控制寄存器,正常输出
    TXERR = 0X00;          //写指令清除发送错误计数寄存器
    i = ECC;               //读指令清除错误代码捕捉寄存器
    MODE = 0X04;           //设置单滤波方式,并返回工作状态
}
/*****接收中断(外部 0 中断) *****/
void Int0_CANReceive() interrupt 0 //接收中断
{
    uchar ss,i;
    ss = SR;               //读入 SJA1000 状态
    ss&= 0xc3;             //取总线关闭,错误状态,溢出,有数据
    if(ss)                 //无上述状态,则直接返回
    {
        if(ss&0x80)        //总线关闭
        {
            ss = IR;        //读中断寄存器以清除中断位
            MODE = 0X08;     //恢复总线
        }
        else
        {
            ss = IR;        //读中断寄存器以清除中断位
            if(ss&0x08)      //数据溢出
                CMR = 0X0C; //清除数据溢出和释放接收缓冲区
        }
    }
}

```

```

        else
        {
            if(ss&0x01)    //接收缓冲区有数据
            {
                for(i = 0; i < 11; i++)
                Remessage[i] = RXB[i];
                CMR = 0X04; //设置命令寄存器为释放接收缓冲区
            }
            ss = ALC;      //读操作以释放仲裁丢失捕捉寄存器
            ss = ECC;      //读操作以释放错误丢失捕捉寄存器
            flag = 1;
        }
    }
}

/***** 查询发送程序 *****/
void CANSend()
{
    uchar i, ss;
    do
    {
        ss = SR;          //读入 SJA1000 状态
        while(ss&0x10);    //判是否正在接收,是则等待
        do
        {
            ss = SR;      //读入 SJA1000 状态
            while(!(ss&0x08)); //判先前发送是否成功,否则等待
            do
            {
                ss = SR;      //读入 SJA1000 状态
                while(!(ss&0x04)); //判发送缓冲器是否锁定,是则等待
                TXB[0] = 0x08; //设置 TX 标准帧报文, FF = 0, RTR = 0, DLC = 0X08(数据长度)
                TXB[1] = 0x00; //
                TXB[2] = 0x20; //节点地址为 01
                for(i = 0; i < 8; i++)
                TXB[i + 3] = Txmessage[i];
                CMR = 0X10; //自接收请求
            }
        }
    }
}

void main()
{
    uchar i;
    IT0 = 1;

```

CAN 总线技术

```

EA = 1;
EX0 = 1;
Delay(255,255);           //延时等待 CAN 可靠复位
Lcd_Init();
SJA1000_Init();
Lcd_WC(0x80);              //欲发送信息写入第一行
for(i = 0; i < 8; i++)
WriteString (Txmessage[i] + 0x30);
CANSend();
while(!flag);              //等待中断
Lcd_WC(0xC0)      ;        //接收信息写入第二行
for(i = 0; i < 8; i++)
WriteString (Rmessage[i] + 0x30);
while(1);
}

```

实验三 P8xC591 双节点通信实验参考程序

```

#include < P87C591.h >
#include < can591reg.h >
#define uchar unsigned char
uchar Txmessage[11];
uchar Rxmessage[8];
uchar Re_flag = 0;
/***** 初始化程序 *****/
void CAN_Init(void)
{
    P1M1 = 0xC0;           //设置 P10 为准双向口,设置 P11 为推挽输出
    P1M2 = 0xC2;           //设置 P12、P13 为开漏口
    CAN_MOD = 0X01;        //设置方式寄存器,进入复位模式
    CAN_CDR = 0X88;        //设置时钟分频寄存器,选择 Pelican 模式关闭时钟输出
    CAN_IER = 0X01;        //设置中断允许寄存器,开放接收中断
    CANADR = RIL;
    CANDAT = 0x00;         //收到有效报文即可产生接收中断
    CANADR = CAN_ACF1_ACR0; //配置验收滤波器组 1 验收代码寄存器
    CANDAT = 0x00;
    CANDAT = 0x00;
    CANADR = CAN_ACF1_AMR0; //配置验收滤波器组 1 验收屏蔽寄存器
}

```

```

    CANDAT = 0x00;
    CANDAT = 0x00;
    CANDAT = 0xff;
    CANDAT = 0xff;
    CANADR = ACFMOD;           //单边滤波
    CANDAT = 0x01;
    CANADR = ACFEN;           //允许使用验收滤波器组 1
    CANDAT = 0x01;
    CANADR = CAN_BTR0;
    CANDAT = 0x00;           //500kbps 波特率
    CANADR = CAN_BTR1;
    CANDAT = 0x1C;
    CANADR = IER;
    CANDAT = 0x01;
    MODE = 0x01;             //推出复位模式,进入正常操作模式
}

/***** Can 中断服务程序 *****/
void Can_ISR(void) interrupt 13
{
    uchar st;
    if(ss&0x01)              //接收缓冲区有数据
    {
        CANADR = CAN_RXBUF3;
        for(i = 0; i < 8; i++)
            Remessage[i] = CANDAT;
        CMR = 0x04;          //设置命令寄存器为释放接收缓冲区
        uchar Re_flag = 1;
    }
}

/***** 查询发送程序 *****/
void Can_send()
{
    uchar i;
    while(CAN_SR&0x10);      //判是否正在接收,是则等待
    while(!(CAN_SR&0x08));    //判先前发送是否成功,否则等待
    while(!(CAN_SR&0x04));    //判发送缓冲器是否锁定,是则等待
    CANADR = CAN_TXB0;

```

CAN 总线技术

```

    CANDAT = 0x08;           //设置 TX 标准帧报文, FF = 0, RTR = 0, DLC = 0x08(数据长度)
    CANDAT = 0x00;           //对方节点地址为 02
    CANDAT = 0x40;
    for(i = 0; i < 8; i++)
    CANDAT = i;
    CAN_CMR = 0x01;          //发送
}

/***** 主程序 *****/
void mian()
{
    uchar i;
    uchar * s1 = "error";
    ECAN = 1;                 //使能 CAN 中断
    EA = 1;                   //使能全局中断
    Lcd_Init();               //见实验 1 源代码
    while(1);
    {
        Can_send();
        Delay(255,255);       //见实验 1 源代码
        Delay(255,255);
        Delay(255,255);
        if(Re_flag == 1)
        {
            for(i = 0; i < 8; i++)
            { Lcd_WC(0x80);      //写入第一行
              WriteString (Rxmessage[i] + 0x30);
              Re_flag = 0;
            }
        }
        else
        {
            Lcd_WC(0xC0);       //写入第二行
            WriteString (* s1);
            Re_flag = 0;
        }
    }
}

```

实验四 CAN 转 RS232 网桥模块设计参考程序

```

/***** 宏定义 *****/
#define uchar unsigned char
#define uint unsigned int
#define Enable() EA = 1
#define Disable() EA = 0

/***** 声明 *****/
void UART_Init(void);
void UART_Send(void);
void UART_Receive(void);

/***** 引脚定义 *****/
//串口波特率选择开关,P1.0,P1.1,P1.2
sbit LED_CANTX = P2^0;           //接收为黄灯
sbit LED_CANRX = P2^2;           //发送为绿灯

/***** 变量定义 *****/
uchar code
ASCII[] = {0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3a,0x20}; //ASCII code
uchar uBuf_CANSend[2];
uchar uBuf_UARTSend[8];
bit bFlag_CANSend;               //can 发送标志

/***** 延时 *****/
void Delay(uchar t1,uchar t2)
{
    for(;t1>0;t1--)
        for(;t2>0;t2--);
}

/***** 引脚初始化 *****/
void PIN_Init(void)
{
    LED_CANTX = 1;               //不亮
    LED_CANRX = 1;
}

/***** SJA1000 初始化 *****/
void SJA1000_Init(void)
{
    MODE = 0X01;                 //设置方式寄存器,进入复位模式
    CDR = 0X88;                  //设置时钟分频寄存器,选择 Pelican 模式关闭时钟输出
                                //(CLKOUT)

```

CAN 总线技术

```

    IER = 0X01;           //设置中断允许寄存器,开放接收中断
    ACR0 = 0X00;          //设置验收代码寄存器 0,
    ACR1 = ID << 5;       //本机号为 00
    ACR2 = 0X00;
    ACR3 = 0x00;
    AMR0 = 0X00;          //设置验收屏蔽寄存器
    AMR1 = 0X00;          //不屏蔽
    AMR2 = 0XFF;          //设置验收屏蔽寄存器
    AMR3 = 0XFF;          //屏蔽
    BTR0 = 0X00;          //设置总线定时器 0
    BTR1 = 0X1C;          //设置总线定时器 1,11.0592MHz,500Kbps
    OCR = 0XAA;           //设置输出控制寄存器,正常输出
    MODE = 0X08;          //设置单滤波方式,并返回工作状态
}

/ ***** 串口初始化 *****/
void UART_Init(void)
{
    TMOD = TMOD | 0x20;   //定时器 1,方式 2
    TH1 = 0xfd;           //9600Bps,11 059.2 晶振
    TL1 = 0xfd;
    PCON = 0x00;          //波特率不倍增
    SCON = 0x50;          //串行方式 1
    PS = 1;               //串行优先级高
    TR1 = 1;              //启动计时
    ES = 1;               //开串口中断
}

/ ***** 定时器初始化 *****/
void Timer_Init(void)
{
    TMOD = TMOD | 0x01;   //定时器 0,方式 1
    TH0 = 0X3C;
    TL0 = 0XB0;           //方式 1,50 ms 定时
    ET0 = 1;              //开定时中断 0
}

/ ***** 外部中断初始化 *****/
void Int_Init(void)
{
    IT0 = 1;

```

```

    EX0 = 1;                                //开外部中断 0
}
/***** SJA1000_Send *****/
void SJA1000_Send(void)
{
    TXSFF = 0x08;                            //设置 TX 标准帧报文, FF = 0, RTR = 0, DLC = 0x08(数据长度)
    TXID1 = 0x00;
    TXID2 = (uBuf_CANSend[0] - 0x30) << 5; //目标机号为 00
    TXDATA1 = uBuf_CANSend[1];              //发送命令
    CMR = 0x01;                             //设置命令寄存器为启动发送
    LED_CANTX = ~LED_CANTX;                 //发送标志
}
/***** 主程序 *****/
main()
{
    Disable();
    Delay(255, 255);                         //延时等待 CAN 可靠复位
    PIN_Init();
    UART_Init();
    Timer_Init();
    Int_Init();
    SJA1000_Init();
    Enable();
    uBuf_UARTSend[0] = '$';
    uBuf_UARTSend[5] = '.';
    uBuf_UARTSend[7] = '*';
    while(1)
    {
        if(bFlag_CANSend)
        {
            bFlag_CANSend = 0;
            SJA1000_Send();
        }
    }
}
/***** 外部 0 中断 *****/
void Int0_CANReceive() interrupt 0          //接收中断
{

```

CAN 总线技术

```

uchar ss;
ss = SR; //读入 SJA1000 状态
ss&= 0xc3; //取总线关闭,错误状态,溢出,有数据
if(ss) //无上述状态,则直接返回
{
    if(ss&0x80) //总线关闭
    {
        ss = IR; //读中断寄存器以清除中断位
        MODE = 0x08; //恢复总线
    }
    else
    {
        ss = IR; //读中断寄存器以清除中断位
        if(ss&0x08) //数据溢出
            CMR = 0x0C; //清除数据溢出和释放接收缓冲区
        else
        {
            if(ss&0x01) //接收缓冲区有数据
            {
                uBuf_UARTSend[1] = ASCII[RXDATA1]; //目标机 ID
                uBuf_UARTSend[2] = ASCII[RXDATA2]; //温度的千位
                uBuf_UARTSend[3] = ASCII[RXDATA3]; //温度的百位
                uBuf_UARTSend[4] = ASCII[RXDATA4]; //温度的个位
                uBuf_UARTSend[6] = ASCII[RXDATA5]; //温度的十分位
                CMR = 0x04; //设置命令寄存器为释放接收缓冲区
                LED_CANRX = ~LED_CANRX; //接收标志
                TFO = 1; //进入定时中断
            }
            ss = ALC; //读操作以释放仲裁丢失捕捉寄存器
            ss = ECC; //读操作以释放错误丢失捕捉寄存器
        }
    }
}

/***** 定时 0 中断 *****/
void Time0_CANSend() interrupt 1 //定时发送
{
    static uchar uCount_50ms;
    TR0 = 0;

```

```

    TH0 = 0X3C;
    TL0 = 0XB0;
    TR0 = 1;
    uCount_50ms++;
    if(uCount_50ms == 2)                //100ms//SJA1000 发送
    {
        uCount_50ms = 0;
        UART_Send();
    }
}
/***** 串口发送 *****/
void UART_Send(void)
{
    static uchar uCount_Send;
    if(uCount_Send < 8)
    {
        SBUF = uBuf_UARTSend[uCount_Send];
        uCount_Send++;
    }
    else
    {
        uCount_Send = 0;
        TR0 = 0;
    }
}
/***** 串口接收 *****/
void UART_Receive(void)
{
    static uchar uCount_Receive;
    static bit bFlag_FF;
    uchar uBuf;
    uBuf = SBUF;
    if(uBuf == '$')
        bFlag_FF = 1;
    else if(uBuf == '*')
        bFlag_FF = 0;
    else
        if(bFlag_FF)
        {

```

CAN 总线技术

```

        uBuf_CANSend[uCount_Receive] = uBuf;
        uCount_Receive++;
        if(uCount_Receive > 1)
        {
            uCount_Receive = 0;
            bFlag_CANSend = 1;
        }
    }

}

/***** 串口中断 *****/
void UART_INT(void) interrupt 4    //串口通信用 ASCII
{
    if(TI == 1)
    {
        TI = 0;
        LED_CANTX = ~LED_CANTX;
    }
    if(RI == 1)
    {
        RI = 0;
        UART_Receive();
        LED_CANRX = ~LED_CANRX;
    }
}

```

实验五 CAN 中继器设计参考程序

```

#include < reg51.h >
#include < peliCAN1.h >           //绝对宏方式定义的寄存器
#include < peliCAN2.h >
#define LCD P1
#define uchar unsigned char
uchar Remessage1[11];
uchar Remessage2[11];
uchar flag1 = 0, flag2 = 0;
/***** CAN1 初始化 *****/
SJA1000_1_Init()
{
    uchar i;

```

```

MODE1 = 0X01;           //设置方式寄存器,进入复位模式
CDR1 = 0X88;           //设置时钟分频寄存器,选择 PeliCAN 模式关闭时钟输出(CLKOUT)
IER1 = 0X0D;           //设置中断允许寄存器,开放(发送,溢出,错误警告)中断
ACR1[0] = 0X00;         //设置验收代码寄存器,
ACR1[1] = 0X20;         //节点号为 01
ACR1[2] = 0X00;
ACR1[3] = 0x00;
AMR1[0] = 0X00;         //没有屏蔽
AMR1[1] = 0X00;         //没有屏蔽
AMR1[2] = 0Xff;         //屏蔽
AMR1[3] = 0Xff;         //屏蔽
BTR0_1 = 0X00;         //设置总线定时器 0
BTR1_1 = 0X1C;         //设置总线定时器 1,16MHz,500Kbps
OCR1 = 0XAA;           //设置输出控制寄存器,正常输出
TXERR1 = 0X00;         //写指令清除发送错误计数寄存器
i = ECC1;              //读指令清除错误代码捕捉寄存器
MODE1 = 0X04;         //设置单滤波方式,并返回工作状态
}

/ ***** CAN2 初始化 ***** /
SJA1000_2_Init()
{
    uchar i;
    MODE2 = 0X01;       //设置方式寄存器,进入复位模式
    CDR2 = 0X88;       //设置时钟分频寄存器,选择 PeliCAN 模式关闭时钟输出(CLKOUT)
    IER2 = 0X0D;       //设置中断允许寄存器,开放(发送,溢出,错误警告)中断
    ACR2[0] = 0X00;     //设置验收代码寄存器,
    ACR2[1] = 0X20;     //本节点号为 01
    ACR2[2] = 0X00;
    ACR2[3] = 0x00;
    AMR2[0] = 0X00;     //没有屏蔽
    AMR2[1] = 0X00;     //没有屏蔽
    AMR2[2] = 0Xff;     //屏蔽
    AMR2[3] = 0Xff;     //屏蔽
    BTR0_2 = 0X00;     //设置总线定时器 0
    BTR1_2 = 0X1C;     //设置总线定时器 1,16MHz,500Kbps
    OCR2 = 0XAA;       //设置输出控制寄存器,正常输出
    TXERR2 = 0X00;     //写指令清除发送错误计数寄存器
    i = ECC2;          //读指令清除错误代码捕捉寄存器
    MODE2 = 0X04;     //设置单滤波方式,并返回工作状态
}

```

CAN 总线技术

```

}

/***** CAN1 接收中断(外部 0 中断) *****/
void Int0_CANReceive() interrupt 0 //接收中断
{
    uchar ss,i;
    ss = SR1; //读入 SJA1000 状态
    ss&= 0xc3; //取总线关闭,错误状态,溢出,有数据
    if(ss) //无上述状态,则直接返回
    {
        if(ss&0x80) //总线关闭
        {
            ss = IR1; //读中断寄存器以清除中断位
            MODE1 = 0X08; //恢复总线
        }
        else
        {
            ss = IR1; //读中断寄存器以清除中断位
            if(ss&0x08) //数据溢出
                CMR1 = 0X0C; //清除数据溢出和释放接收缓冲区
            else
            {
                if(ss&0x01) //接收缓冲区有数据
                {
                    for(i = 0;i<11;i++)
                        Remessage1[i] = RXB1[i];
                    CMR1 = 0X04; //设置命令寄存器为释放接收缓冲区
                }
                ss = ALC1; //读操作以释放仲裁丢失捕捉寄存器
                ss = ECC1; //读操作以释放错误丢失捕捉寄存器
                flag1 = 1;
            }
        }
    }
}

/***** CAN2 接收中断(外部 1 中断) *****/
void Int1_CANReceive() interrupt 2 //接收中断
{
    uchar ss,i;
    ss = SR2; //读入 SJA1000 状态
    ss&= 0xc3; //取总线关闭,错误状态,溢出,有数据

```

```

        if(ss)                                //无上述状态,则直接返回
        {
if(ss&0x80)                                //总线关闭
        {
            ss = IR2;                        //读中断寄存器以清除中断位
            MODE2 = 0X08;                    //恢复总线
        }
        else
        {
            ss = IR2;                        //读中断寄存器以清除中断位
            if(ss&0x08)                      //数据溢出
                CMR2 = 0X0C;                //清除数据溢出和释放接收缓冲区
            else
            {
                if(ss&0x01)                  //接收缓冲区有数据
                {
                    for(i = 0; i < 11; i++)
                        Remessage2[i] = RXB2[i];
                    CMR2 = 0X04;              //设置命令寄存器为释放接收缓冲区
                }
                ss = ALC2;                    //读操作以释放仲裁丢失捕捉寄存器
                ss = ECC2;                    //读操作以释放错误丢失捕捉寄存器
                flag2 = 1;
            }
        }
    }
}

/ ***** 中继器查询发送程序 1 ***** /
void CANSend1()
{
    uchar i,ss;
    do
    {
        ss = SR1;                            //读入 SJA1000 状态
        while(ss&0x10);                       //判是否正在接收,是则等待
        do
        {
            ss = SR1;                            //读入 SJA1000 状态
            while(!(ss&0x08));                  //判先前发送是否成功,否则等待
        }
        do

```

CAN 总线技术

```

        ss = SR1;                //读入 SJA1000 状态
while(!(ss&0x04));              //判发送缓冲器是否锁定,是则等待
TXB1[0] = 0x08;                //设置 TX 标准帧报文, FF = 0, RTR = 0,
TXB1[1] = 0x00;                //DLC = 0x08(数据长度)
TXB1[2] = 0x20;                //节点地址为 01
for(i = 0; i < 8; i++)
TXB1[i + 3] = Remessage1[i];
CMR1 = 0x01;                  //发送请求
flag1 = 0;
}

/***** 中继器查询发送程序 2 *****/
void CANSend2()
{
    uchar i, ss;
    do
        ss = SR2;                //读入 SJA1000 状态
    while(ss&0x10);              //判是否正在接收,是则等待
    do
        ss = SR2;                //读入 SJA1000 状态
    while(!(ss&0x08));            //判先前发送是否成功,否则等待
    do
        ss = SR2;                //读入 SJA1000 状态
    while(!(ss&0x04));            //判发送缓冲器是否锁定,是则等待
    TXB2[0] = 0x08;              //设置 TX 标准帧报文, FF = 0, RTR = 0,
    TXB2[1] = 0x00;              //DLC = 0x08(数据长度)
    TXB2[2] = 0x40;              //节点地址为 02
    for(i = 0; i < 8; i++)
    TXB2[i + 3] = Remessage2[i];
    CMR2 = 0x01;                //发送请求
    flag2 = 0;
}

/***** 主程序 *****/
void main()
{
    IT0 = 1;
    IT1 = 1;
    EA = 1;
    EX0 = 1;

```

```
EX1 = 1;
SJA1000_1_Init();
SJA1000_2_Init();
while(1)
{
    if(flag1 == 1) CANSend1();
    else if(flag2 == 1) CANSend2();
}
}
```

附录 B

CANopen 对象字典的详细结构

表 B-1 CANopen 对象字典的详细结构列表

索 引	对 象	名 称
0000H	NULL	保留
0001H	DEFTYPE	Boolean
0002H	DEFTYPE	Integer8
0003H	DEFTYPE	Integer16
0004H	DEFTYPE	Integer32
0005H	DEFTYPE	Unsigned8
0006H	DEFTYPE	Unsigned16
0007H	DEFTYPE	Unsigned32
0008H	DEFTYPE	Float
0009H	DEFTYPE	Visible String
000AH	DEFTYPE	Octet String
000BH	DEFTYPE	Date
000CH	DEFTYPE	Time of Day
000DH	DEFTYPE	Time Difference
000EH	DEFTYPE	Bit String
000FH	DEFTYPE	Domain
0010H~001FH	NULL	保留
0020H	DEFSTRUCT	PDO 通信参数

附录 B CANopen 对象字典的详细结构

续表 B-1

索 引		对 象	名 称
0020 的子索引	0	04H	对象数量,也可设为 2、3
	1	07H	PDO 标识符, Unsigned32
	2	05H	传输方式, Unsigned8
	3	06H	禁止时间, Unsigned16
	4	05H	CMS 优先级, Unsigned8
0021H		DEFSTRUCT	PDO 映射参数
0021H 的子索引	0		对象数量, Unsigned8
	1		第 1 个映射对象, Unsigned32
	2		第 2 个映射对象, Unsigned32
	⋮	⋮	⋮
	40H		第 64 个映射对象, Unsigned32
0022H		DEFSTRUCT	SDO 参数
0022H 的子索引	0		对象数量, Unsigned8
	1		客户端发给服务器的标识符, Unsigned32
	2		服务器发给客户端的标识符, Unsigned32
	3		响应服务器的客户端的节点 ID, Unsigned8
0023H~003FH		NULL	保留
0040H~005FH		DEFTYPE	设备制造商信息
0060H~007FH		DEFTYPE	标准化设备规范
0080H~009FH		DEFSTRUCT	接口规范说明
00A0H~0FFFH		NULL	保留

附录 C

常见调试错误分析

常见调试错误有：

- ① 不能正确初始化 CAN 控制器；
- ② CAN 控制器收发不成功；
- ③ CAN 控制器不能正常工作。

可针对常见错误采取如下措施：

① 上电后不能正确初始化 CAN 控制器。在上电时，从 CAN 地址内容中观测到 CAN 没有正确初始化，这是因为 CPU 工作电压较低（为 2.7 ~ 5.5 V），上电即工作，当 CPU 运行初始化程序时，CAN 控制器还未准备好，所以出现该现象。此时，可以在主程序开始加一段延时程序等待 CAN 控制器。

② CAN 发送不成功。此时 CAN 的状态寄存器中 SR. 3 为 0，表明先前请求的发送还未完成。这里原因较多，主要因为硬件发送线路出现问题，如光隔次边不导通、发送接口接触不良等。

③ 在 CAN 通信程序中尽量避免死循环，设置软件看门狗复位，使 CAN 适时复位，重新恢复正常工作状态。

④ 在发送过程中，由于发送数据过快，会产生总线错误，可适当调整帧之间的发送数据时间参数，如可通过设置延时调整发送数据间隔。

⑤ 收发线路出现断路或短路，总线关闭（状态寄存器 = F4）。一般情况下，存在收发电路上的光电隔离器容易损坏、CAN 控制器或 CAN 收发器虚焊等问题。

⑥ 总线上的差分信号经收发器转换后成为电平信号，此时的信号易受外界干扰，这样会导致 CAN 控制器发生接收错误，接收错误寄存器满后总线自动关闭，无法通信。这时应查找干扰源，予以排除。

通过调试还可以发现很多实际问题，这些问题必须现场根据情况一一解决，随着总线技术与单片机技术的不断成熟，相信以后的调试工作将越来越方便快捷。

参 考 文 献

- [1] <http://www.dndev.com>
- [2] 史久根. CAN 现场总线系统设计技术[M]. 北京:国防工业出版社,2004.
- [3] 饶运涛. 现场总线 CAN 原理与应用技术(第 2 版)[M]. 北京:北京航空航天大学出版社,2007.
- [4] TI. TMS320X281X,280XDSP Enhanced Controller Area Network(eCAN)Reference Guide. 2007.
- [5] 宁改娣. DSP 控制器原理及应用(第 2 版)[M]. 北京:科学出版社,2009.