文章编号:1671-4598(2014)07-2076-04　　中图分类号:TP273　　文献标识码:A

基于 CANopen 协议的水下机器人控制系统设计

金　洋[1,2]，李　硕[1]，曾俊宝[1]
(1. 中国科学院 沈阳自动化研究所机器人学国家重点实验室，沈阳　110016；2. 中国科学院大学，北京　100049)

摘要：为了提高 AUV 控制系统的可靠性、灵活性，采用分布式网络代替之前的集中式网络；并通过 CAN 和 CANopen 协议搭建分布式网络，主节点采用 ARM－Linux，从节点采用 AVR－μCOS；分别针对主从节点进行设计和优化，并给出了具体的实现方法，最后对 CAN 网络进行了优化；最终提高了 AUV 的控制系统设计的灵活性，同时提高了可靠性和实时性。

关键词：AUV；CAN；CANopen；分布式控制系统

AUV Control System Design Based on CANopen Protocol

Jin Yang[1,2], Li Shuo[1], Zeng Junbao[1]
(1. State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang　110016, China; 2. University of Chinese Academy of Sciences, Beijing　100049, China)

Abstract: In order to improve the reliability and flexibility of the SAUV (Small Autonomous Underwater Vehicle) control system, distributed instead of centralized control system structure is used. This system is designed according to CAN and CANopen protocol, and the master node is based on ARM－Linux, slave node based on AVR－μCOS. To build this system, both master and slave nodes are designed and optimized respectively, then the implementation is given and CAN network is optimized. Finally the flexibility of the AUV control system is improved and also the reliability, performance of real－time.

Keywords: AUV; CAN; CANopen; distributed control system

0　引言

以往 AUV 多采用集中式控制系统结构，通过主控计算机实现对所有传感器和设备的数据采集和控制。然而集中式有如下不足：1）主控计算机需要负责与所有节点的通讯，任务集中，主机负荷较大，效率低；2）主控计算机故障将影响整个网络，可靠性低；3）可扩展节点受到主控计算机接口的限制，扩展性差；4）通讯与控制系统耦合较高，不便于调试；5）系统成本较高，维护困难[1-2]。为了解决集中式控制系统所带来的不足，在设计 AUV 时采用模块化设计，控制系统时采用分布式。

设计分布式系统时，CAN 总线在 AUV 中得到了越来越广泛的应用。CAN 总线最初是由德国 BOSCH 公司为解决汽车监控系统中的诸多复杂技术和难题而设计的，属于总线式串行通信网络[3]。后来，由于其成本低、可靠性高、抗干扰能力和实时性强等特点使其应用范围逐渐扩大[4]。

然而 CAN 协议只定义了物理层和数据链路层，对于一些复杂的应用问题需要一个更高层次的协议来实现。目前许多厂商制定了自己的应用层协议，其中有 DeviceNet、CANopen、CANKingdom、SDS、CAL 等[5]。相比而言 CANopen 协议完全公开，采用面向对象的思想，具有很好的模块化特性和较高的适应性[6]，得到了广泛的应用。

收稿日期:2014-01-22；　修回日期:2014-03-22。
基金项目:863 计划(2011AA09A105)。
作者简介:金　洋(1988-)，男，山东济南人，硕士，学生，主要从事水下机器人控制技术方向的研究。

1　CANopen 协议简介

如图 1 所示[7]，CAN 协议只定义了物理层和数据链路层，CANopen 协议位于 ISO/OSI 参考模型的应用层。CANopen 协议实际包含了多个协议模块，其中 DS－301 为基本的模块，对 CAN 报文进行了划分，包括了网络管理报文（NMT）、同步报文（SYNC）、紧急报文（EMERGENCY）、时间戳（TIME）、过程数据对象（PDO）、服务数据对象（SDO）和节点状态报文（NMT Err Control）；CiA－401 定义了基本 I/O 模块；CiA－404 定义了测量设备和闭环控制器；CiA－406 定义了编码设备等[8]。

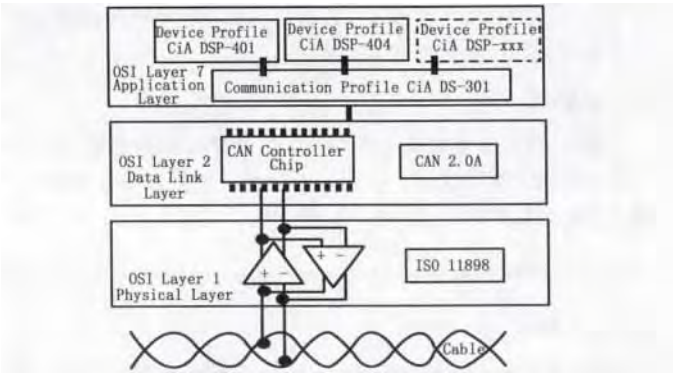

图 1　CAN/CANopen 协议参考模型

CANopen 协议可以支持 127 个节点，使用 CAN2.0A 协议，但是可以兼容 CAN2.0B。另外，在设计 AUV 的控制系统时采用 CAN2.0A 协议，这是因为对于 AUV 的设计，127 个节点可以满足要求，而且使用扩展模式要付出一些代价，主

要包括了：1）总线等待时间较长（最小 20 个位时间）；2）具有扩展格式的消息要求更多的带宽（大约 20%）；3）错误检测的性能较低，因为 15 位 CRC 是针对大约 80 位的帧长进行优化的。

2　AUV 控制系统设计

在 AUV 上实现的 CANopen 控制系统主要由 4 个节点组成，包括电源节点、舵机控制节点、主推电机节点和主控制节点。主节点采用 ARM 芯片，运行的是 Linux 操作系统；从节点采用 AVR 单片机，运行的是 μCOS 实时操作系统。

其中电源节点主要用于对电源和载体部分状态的信息采集，包括电源的电压、电流、电池舱的温度和电池舱的漏水检测；舵机控制节点主要用于舵机的控制和相关的信息采集，包括了控制 4 个舵机控制量的输出、对推进舱的电流和温度的数据采集、对推进舱的漏水检测；主节点主要用于数据采集、控制量的计算、对 CAN 网络的管理和保护；主推节点主要用于控制主推电机并返回电机的工作状态，该节点为非标准的 CANopen 节点，其它 3 个节点为符合 CANopen 协议。整个 CANopen 控制系统的拓扑结构如图 2 所示。

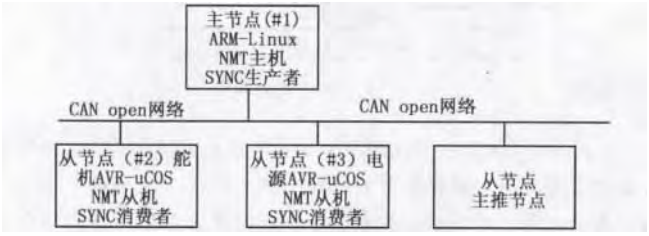

图 2　CANopen 控制系统拓扑结构

2.1　从节点设计

舵机和电源从节点属于标准的 CANopen 从节点，主推节点通过 CAN 总线进行通讯，但只需要周期性的接收速度信息，并返回电机的状态信息。因此，对于主推节点只需要通过 SYNC 周期性的发送电机控制量，并实时监控电机的状态即可。在介绍从节点的设计时，主要针对的是舵机和电源从节点，而且以舵机从节点为例。

从节点主要采用 AVR 单片机，为了提高系统的可扩展性和编程的灵活性，使用小型的实时操作系统 μCOS。μCOS 是一个可靠性高、资源消耗少、可配置性高的实时操作系统内核[9]，在完成基本的通讯模块后可以对从节点进行自我检测和容错控制的设计，此时不会影响原有的功能。

电源和舵机从节点的部分对象字典如图 3 所示。其中 TPDO 用于向主控节点发送数据；RPDO 用于接收主节点发送的控制量；SDO 可以用于修改从节点的对象字典；HB 用于提供心跳报文，通过该报文可用于主节点监控从节点。

TPDO1	TPDO2	SDO-s	HB-p
183h	283h	603h-rx	703h
0x1800	0x1801	583h-tx	0x1017
0x03	0x04	0x1200	1000ms
从节点（#3）电源 NMT从机 SYNC消费者			

TPDO1	RPDO1	SDO-s	HB-p
182h	202h	602h-rx	702h
0x1800	0x1400	582h-tx	0x1017
0x03	0xFF	0x1200	1000ms
从节点（#2）舵机 NMT从机 SYNC消费者			

图 3　电源/舵机从节点的对象字典设计

舵机从节点的功能模块如图 4 所示，主要包括了采集舵机舱段的电压、电流、温度，漏水检测，对 4 路舵机进行控制。

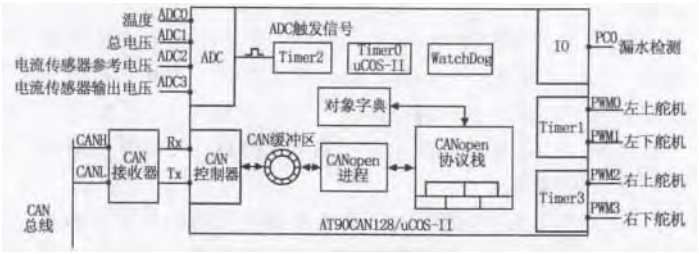

图 4　舵机从节点功能模块图

CAN 控制器利用中断接收 CAN 报文，并保存在循环队列中，通过消息邮箱通知 CANopen 堆栈。CANopen 堆栈接着根据报文类型对报文进行处理，处理后的数据保存在对象字典中。CANopen 堆栈最后会调用初始化时注册的钩子函数，调用用户定义的相应处理函数。

2.1.1　硬件设计

从节点采用 AVR 单片机，该单片机内含有 CAN 控制器，可以用于接收和发送 CAN 报文，支持 CAN 协议最高的 1Mbps 通讯速率；支持 CAN 的基本模式和扩展模式；总共有 15 个消息对象（Message Object，MOb）可供配置，都可以配置成发送或接收模式。

图 5 为从节点的硬件结构示意图，CAN 驱动电路采用独立 CAN 接收器 TJA1050。为了减少干扰，通过 ADUM1201 进行隔离，CAN 控制器使用 AVR 内集成的 CAN 的控制器。信号调理电路主要用于采集模拟量信息，如电源电压、电源电流、温度等信息。PWM 驱动电路主要是通过 AVR 产生的 PWM 信号驱动电机。

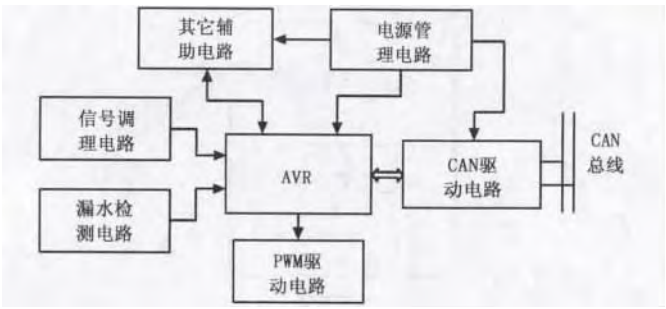

图 5　CANopen 网络从节点硬件结构示意图

2.1.2　μCOS 中断响应

为了加快从节点的响应速度，在从节点中的大部分功能都采用中断进行处理。在 μCOS 中，中断响应函数可以分为如下的 3 类：

1）对于正常的中断服务，可能会使较高优先级的任务进入就绪状态、会调用 C 语言函数并且允许中断嵌套。此时，在保存现场时，需要保存所有寄存器，包括所有的通用寄存器和状态寄存器，并将堆栈指针保存在 TCB 中。

而且需要调用 OSIntEnter（）函数通知 μCOS 进入了中断服务中，在退出中断服务时需要通过

OSIntExit（）通知 μCOS 中断服务已经完成。在退出中断服务时，检测是否有高优先级的任务就绪，如果有则会进行任务切换。

2）如果不需要中断嵌套，且不会使高优先级的任务就绪，但是仍需要调用 C 函数。此时只需要保存现场，以防止在 C 函数会改变寄存器的值；但不需要通知 μCOS 已经进入中断服务，在退出时同样不需要通知 μCOS，只需要在退出时恢复现场即可。

3）如果不允许中断嵌套，不会使高优先级的任务就绪且不调用 C 函数，此时只需要保存在汇编语言中改变的寄存器即可，不需要保存所有的现场。中断服务中同样可以不通知 μCOS。

在上述的中断服务类型中，响应速度 1＜2＜3。在从节点的设计中，时钟中断函数为 μCOS 提供时钟信号，精度为 5 ms，采用第一种。对于 CAN 接收中断属于第二种情况，不允许中断嵌套，但会调用 C 语言函数。对于 PWM 占空比的改变属于第三种情况，中断响应采用汇编语言，此时只需要保存在汇编中用到的寄存器即可。

2.1.3 ADC 设计

在 AVR 中采用 10 位的逐次逼近型 ADC，选择内部 2.56 V 作为 AD 转换的参考电压，为了确定其采样的周期，采用自动触发和手动触发相结合，同样为了加快处理速度，采用中断服务。中断服务的执行流程图如图 6 所示。

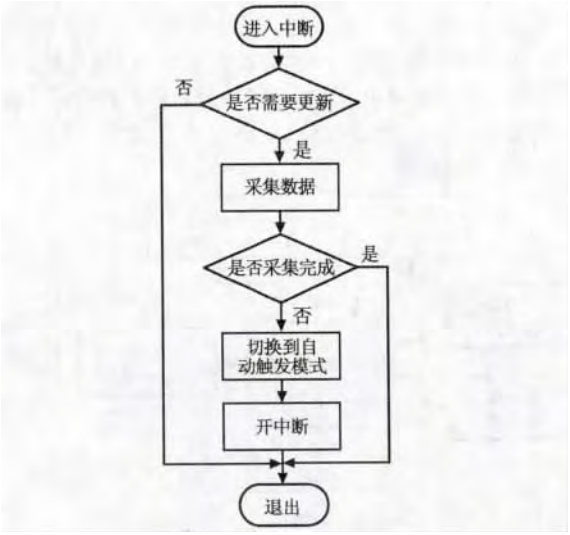

图 6 ADC 中断服务处理流程图

由于 ADC 的转换时间相对而言时间较长，因此每个一段时间更新一次。在需要读取时直接读取变量即可，而非在接到读取的请求命令时再进行采样。为了防止读取数据和采样发生冲突，通过全局变量检测是否需要更新，此时对全局变量的操作应该是原子操作。

在 AVR 中总共有 3 个定时器，分别用于系统时钟、应用定时、和产生 4 路的 PWM。因此 ADC 的时钟触发源需要与其它应用共用，此处选择与 PWM 的定时器共享，PWM 的周期为 30 ms。而 ADC 每一个采样周期内，需要进行 4 次采样，因此必须满足如下条件。

$$t_{adc} = 4 \times 13 \times T_{adc} = 52 \times \frac{Hz_{avr}}{P_{adc}} < t_{pwm}$$

t_{adc} 表示 ADC 采样总时间，Hz_{avr} 表示 AVR 的主频，P_{adc} 表示 ADC 分频。也就是说应该保证 ADC 的采样时间小于 PWM 的周期，否则每个周期内 ADC 采样将会失败。

2.2 主节点设计

相比从节点而言，主节点拥有较多的资源，设计主节点时考虑更多的是软件设计的灵活性。主节点采用 ARM 芯片，使用 Linux 操作系统，其结构框图如图 7 所示。与 AVR 单片机类似，使用独立的 CAN 接收器，且使用 ARM 芯片内部的 CAN 控制器。

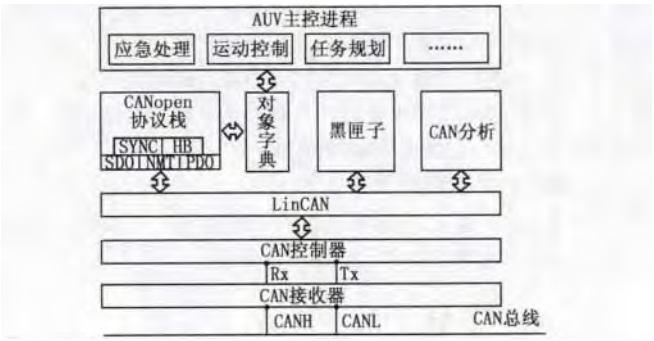

图 7 主节点结构框图

Linux 的 CAN 驱动采用 LinCAN，相比其它 CAN 驱动程序来说，LinCAN 有着独有的特性。如 CANLinux 支持一个进程，那么如果 CANopen 协议栈打开该设备后，那么其它进程将不能访问；SocketCAN 使用了 Socket 编程的接口，虽然支持多个进程同时打开，但 CAN 总线和以太网有着很多不同的特性。使用 LinCAN，不但支持多个进程打开，而且可以支持 CAN 总线独有的特性，不会受限于 Socket 架构。

AUV 的主控进程将通过对象字典与 CANopen 协议栈进行通讯，在 CANopen 协议栈接收数据的同时，可以通过另一个进程同时存储 CAN 报文，或者进行分析。

2.2.1 主节点对象字典

主节点包括了网络管理系统的主机、同步消息的生产者、心跳报文的消费者。NMT 主机用于控制各个从节点的工作状态，进而可以管理控制整个网络的工作状态。SYNC 生产者主要用于产生同步报文，对于 PDO 进行数据的同步。心跳报文消费者主要用于监控从节点。CANopen 主节点同样是通过对象字典对节点的特性进行配置，主节点的对象字典如表 1 所示。

表 1 主节点的对象字典设计

对象	COB－ID	OD 中的索引	传输类型
RPDO1	0x182	0x1400	0xff
TPDO1	0x202	0x1800	0x03
SDO1	0x602/0x582	0x1280	—
RPDO2	0x183	0x1401	0xff
RPDO3	0x283	0x1402	0xff
SDO2	0x603/0x583	0x1281	—
TPDO2	0x1f5	0x1801	0x05
RPDO4	0x1ff	0x1403	0xff
Heartbeat	—	0x1016	—

在初始化阶段主节点可以通过服务数据对象对从节点进行配置，配置的对象包括了几乎所有的 CANopen 配置参数。主节点还包括了 4 个 RPDO 对象和 2 个 TPDO 对象，用于发送和接收数据对象。总共包含了 6 个 PDO 数据链，其中两个对应于舵机从节点，一个 RPDO 用于接收推进舱的工作电流、温度和漏水检测，一个 TPDO 用于发送舵机的控制量；两个对应于电源从节点，均为 RPDO，用于接收温度、电源电压、电流和漏水检测等数据；另有两个对应于主推电机，用于发送控制量和接收返回数据。

对从节点的保护。主节点可通过 RPDO、心跳报文和紧急报文对 3 个从节点进行监控。其中心跳报文可以监控电源和舵机从节点，两个从节点每隔 1 s 向主节点发送一次心跳报文，如果在 3 s 内主节点没有接收到从节点的心跳报文，则主节点会采取相应的措施。

2.2.2　CANopen 网络优化

CANopen 网络在开始启动时，首先会通过 SDO 报文对各个节点进行配置，此时是网络的初始化阶段，这时我们需要保证各个节点配置的功能性。

当 CANopen 网络完成配置并运行时，我们需要考虑总线负载是否合理。在设计网络时应该尽量保证特定事件响应时间，如管理报文、紧急报文等，这些报文通常是事件驱动的因此最好使总线的负载率小于 80%。在本文的网络中，节点较少，总线负载率较低。

为了提高对总线中数据通讯的可预测性，所有的 PDO 类型均采用同步模式。此时在每个通讯周期中可以确定每个数据报文的顺序。同时为了减少同步数据传输时出现的浪涌现象，在使用 SYNC 报文时，同时含有计数信息，并利用该信息将 PDO 报文平均分配。

同时为了提高网络的可靠性，通过心跳报文对各个节点进行监控。同时各个节点会监控自己的状态，在将要发生错误时(如错误计数值到达一个阈值后)，将该节点的状态信息发送给主节点。

最终 CAN 总线的数据报文传输如图 8 所示。在每个同步窗口中，各个 PDO 类型数据进行发送和接收。同时主节点通过心跳报文对从节点进行监控，同时也会有异步报文的传输，如 EMCY 报文。

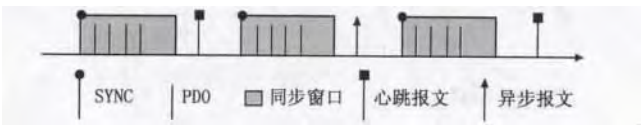

图 8　CAN 总线数据报文

2.2.3　实验结果

在完成网络组建后，对其运行的实际效果进行了测试。主节点发送同步报文，从节点在接收到同步报文后，根据各自的传输类型决定是否在同步窗口中发送报文。可以检测到在同步窗口内，会有大量的报文发送，总线的负载增加；这主要是因为在同步窗口中有大量的同步报文传输，此时不会影响异步报文和紧急报文的传输。当从节点发生故障时，可以向主节点发送紧急报文；如果是严重的故障，主节点同时可以通过心跳报文检测到，实现了对从节点的有效监控。

3　结论

本文实现了基于 CAN/CANopen 协议的 AUV 控制系统，并进行了测试。CAN 协议通过物理层和数据链路层保证了数据传输的实时性和可靠性，通过 CANopen 应用层提高了网络设计的灵活性，利用 CANopen 协议，在设计节点时只需要少量修改即可。分别针对主从设备设计了相应的 CANopen 节点。主节点含有丰富的资源，主要进行复杂的数值运算，为了保证 CANopen 网络的可靠性，可以设计监控进程；而从节点资源较少，为了提高实时性，尽量采用中断响应。采用分布式网络符合 AUV 的模块化设计，提高了网络的可靠性，便于添加其它功能模块。

参考文献：

[1] 王　立，王世强. 基于 CAN 总线的 AUV 分布式控制系统 [J]. 水雷战与舰艇防护，2011，19 (2)：22-25.

[2] 黄时佺，刘　健，王国权. AUV 内部通讯总线设计 [J]. 机器人，2004，26 (4)：342-345.

[3] International Organization for Standardization，ISO－11898 Road vehicles－Controller area network [S]. Switzerland，2003.

[4] 饶运涛，邹继军，郑勇芸. 现场总线 CAN 原理与应用技术 [M]. 北京：北京航空航天大学出版社，2003.

[5] 王桂荣，钱剑敏. CAN 总线和基于 CAN 总线的高层协议 [J]. 计算机测量与控制. 2006，27 (1)：24-30.

[6] CAN in Automation，CiA－301 CANopen application layer and communication profile [S]. Nuremberg，2011.

[7] Boterenbrood H. CANopen high－level protocol for CAN－bus [R/OL]，www. nikhef. nl/pub/departments/ct/po/doc/CANopen30. pdf？ 2000.

[8] Holger Zeltwanger. 现场总线 CANopen 设计与应用 [M]. 周立功，黄晓清，严寒亮译. 北京：北京航空航天大学出版社，2011.

[9] Jean J. Labrosse. 嵌入式实时操作系统 μCOS－II [M]. 邵贝贝，等译. 北京：北京航空航天大学出版社，2003.

(上接第 2072 页)

参考文献：

[1] 庄衍平. 通信基站用环保低噪声柴油发电机组 (5kW～24kW) [A]. 通信电源新技术论坛——2008 通信电源学术研讨会论文集 [C]，中国通信学会通信电源委员会，2008：5.

[2] 徐　明. 基于 DSP 的柴油发电机组的控制器设计 [D]. 南昌：南昌大学，2005.

[3] 黄智伟. 基于 STM32F4 的图像压缩技术研究 [J]. 南华大学学报 (自然科学版)，2012，(3)：58-63.

[4] 胡恒生，王　慧，赵徐成，等. 蓄电池充电方法的分析和探讨 [J]. 电源技术应用，2009，(8)：1-4.

[5] 陈建国，黄　河. PID 数字控制器多指标优化模拟设计方法研究 [J]. 计算机测量与控制，2012，(6)：1530-1534.

[6] 庞丽萍，曲洪斌，王　浚. 电加热模糊 PID 控制及仿真研究 [J]. 计算机应用研究，2004，(9)：225-226.