# 华北电力大学毕业设计(论文)附件外文文献翻译

# 目录

[32 控制器局域网路（CAN）](#header-c538)  
 [32.1 简介](#header-c539)  
 [32.2 嵌入式的特点](#header-c552)  
 [32.3 框图](#header-c603)  
 [32.4 应用框图](#header-c608)  
 [32.5 I/O 总线简介](#header-c629)  
 [32.6 产品依赖](#header-c645)  
 [32.6.1 I/O 总线](#header-c646)  
 [32.6.2 资源管理器](#header-c677)  
 [32.6.3 中断资源](#header-c682)  
 [32.7 CAN 控制器的特色](#header-c697)  
 [32.7.1 CAN 协议概述](#header-c698)  
 [32.7.2 邮箱的组织](#header-c720)  
 [32.7.2.1 信息接受过程](#header-c725)  
 [32.7.2.2 接收邮箱](#header-c749)  
 [32.7.2.3 传输邮箱](#header-c771)

# 32 控制器局域网路（CAN）

## 32.1 简介

爱特梅尔控制器局域网络（CAN）提供了由罗伯特博世有限公司起草的串行通信协议要求实现的所有功能，CAN 为实现高速部分援引的是 ISO/11898A(2.0 A部分 2.0 B部分)规范，为实现低速部分援引的是 ISO/11519-2。CAN 控制器能够处理所有类型的帧（数据、远程、错误和过载）并且能达到 1 Mbit/s 的比特率。

可以通过配置寄存器来访问 CAN 控制器。CAN 控制器实现了 8 个独立的消息对象（邮箱）。

任何一个邮箱都能被程序设定为接收缓冲块（甚至非连续缓冲区）。为了定义消息的接收，一个或多个消息对象能被掩盖不参与缓冲特性。当缓冲区满的时候会生成一个中断。通过的配置选项，第一个被接收的消息可以被锁存在 CAN 控制器中直到应用承认它，或者这个消息能被新接收的消息丢弃。

任何一个邮箱也能被程序设定为变速器。允许同时设置多个传送邮箱。每个独立的邮箱都可以设定优先级。

每一次接收和发送的消息都会被一个内部的 16 位的时间戳标记。当 CAN 控制器一可用这个时间戳就开始计数，这个计数器能够被应用重置，或者在接收最后一个时间触发模式的邮箱之后自动地计数。

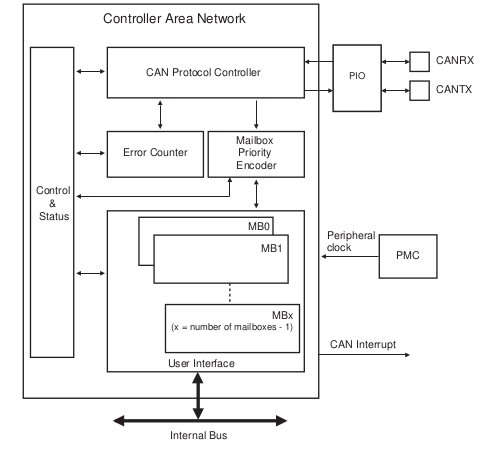
CAN 控制器提供最佳化的特性去支持时间触发通信（TTC）协议。

## 32.2 嵌入式的特点

* 完全兼容 CAN 2.0 A 部分和 2.0 B 部分
* 比特率高于 1 Mbit/s
* 8 个面向对象邮箱具有以下特性：
  + 每个消息 可以根据 CAN 规范 2.0 的 A 部分 或者 B 部分编程
  + 在接收（有复写接收或者非复写接收）或传送模式对象可配置
  + 独立的 29 为标识符，并且每个邮箱可以掩藏定义
  + 可以 32 位寻址每个邮箱数据对象的数据寄存器
  + 可以在接收或者传送的消息上使用 16 位时间戳
  + 隐藏 ID 位域的硬件链接可以加速家庭 ID 进程
* 16 位的内部定时器可以保证时间戳和网络的同步
* 可编程的接收器缓冲区长达 8 个邮箱对象
* 可以用权重管理传送邮箱
* 有自适应和听力模式
* 低功耗模式和可编程唤醒总线活动或者通过应用
* 可处理数据，远程，错误和过载帧
* 寄存器写保护

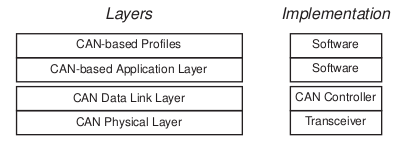
## 32.3 框图

图 32-1，CAN 框图



## 32.4 应用框图

图 32-2，应用框图



|  |  |
| --- | --- |
| 层 | 实现 |
| CAN 基础描述 | 软件 |
| CAN 基础应用层 | 软件 |
| CAN 数据链路层 | CAN 控制器 |
| CAN 物理层 | 收发器 |

## 32.5 I/O 总线简介

表 32-1, I/O 总线简介

|  |  |  |
| --- | --- | --- |
| 名称 | 描述 | 类型 |
| CANRX | CAN 串行接收数据 | 输入 |
| CANTX | CAN 串行发送数据 | 输出 |

## 32.6 产品依赖

### 32.6.1 I/O 总线

使用于 CAN 接口的插槽可能是多路复用的 PIO 线。程序员必须首先编写 PIO 控制器程序将外部设备赋值给他们期望的 CAN 插槽。如果应用没有使用 CAN 的 I/O 总线，它们也能被用于其他用于的 PIO 控制器。

表 32-2, I/O 总线

|  |  |  |  |
| --- | --- | --- | --- |
| 实例 | 标识符 | I/O 总线 | 外设 |
| CAN0 | CANRX0 | PB３ | A |
| CAN0 | CANRX0 | PB３ | A |
| CAN1 | CANRX1 | PC12 | C |
| CAN1 | CANRX1 | PC15 | C |

### 32.6.2 资源管理器

程序员在使用 CAN 之前必须首先在资源管理器控制器中开启 CAN 锁

为 CAN 定义了低功耗模式。如果应用不需要 CAN 运算，当不需要时 CAN 锁能被停，并且 稍后 CAN 锁能被重启。在关闭 CAN 锁之前，CAN 控制器必须处于低功耗模式下去完成 CAN 锁状态的切换。CAN 锁在重启之后，应用必须禁止 CAN 控制器的低功耗模式。

### 32.6.3 中断资源

CAN 中段线被连接在一个内部高级终端控制器的内部资源上。使用 CAN 中断需求时中断控制器必须首先被编程。注意：不建议在边缘敏感模式下使用 CAN 中断线。

表 32-3 外设 ID

|  |  |
| --- | --- |
| 实例 | ID |
| CAN0 | 37 |
| CAN1 | 38 |

## 32.7 CAN 控制器的特色

### 32.7.1 CAN 协议概述

控制器局域网络（CAN）是一个多主串行通信协议，以非常高的安全性和极高的传输速度（比特率高达 1Mbit/s）有效地支持实时控制。

CAN 协议支持四种不同类型的帧：

* 数据帧：他们携带数据从发射机节点到接收器节点。整体的最大数据帧长度为 108 位的标准帧和 128 位扩展帧。
* 远程帧：目标节点可以通过发送与所需数据帧标识符一致的标识符发送远程帧来请求数据。然后适当的数据源节点发送数据帧作为此节点请求的响应。
* 错误帧：由检测总线错误的任何节点生成错误帧。
* 过载帧：它们提供了前和连续数据帧或远程帧之间的额外延迟。

爱特梅尔 CAN 控制器提供 CAN 协议第二版 A 部分和 第二版 B 部分的全功能的 CPU。它最大限制的减少了 CPU 的通信开销。数据链路层和物理层的一部分由 CAN 控制器本身自动处理。

CPU 通过 CAN 控制器的消息盒读写数据或信息。每一个消息和被赋予一个标识符。CAN 控制器通过封装或解码数据消息去生成或解码总线数据帧。远程帧，错误帧和过载帧被在用户软件应用监视下的 CAN 控制器自动处理。

### 32.7.2 邮箱的组织

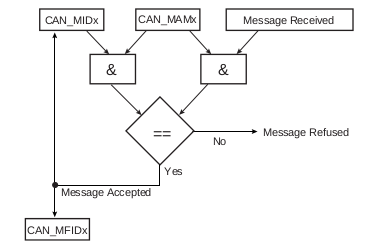
CAN 模块有 8 个缓冲区，也被叫做频道或者邮箱。为每个活动邮箱定义对应于CAN标识符的标识符。消息标识符可以与标准帧标识符或扩展帧标识符匹配。此标识符在初始化过程中第一次被定义，但稍后可以动态重新配置，以便邮箱可以处理新的消息族。几个邮箱可以配置相同的ID。

每个邮箱都能独立的被配置为接收或传送模式，邮箱对象的类型在 CAN\_MMRx 的 MOT 字段。

#### 32.7.2.1 信息接受过程

如果将 CAN\_MIDx 寄存器设置为 MIDE 类型，邮箱就能处理额外定义的格式；否则，邮箱只能处理标准格式。一旦一个消息被接收，那么它 ID 将被 CAN\_MAMx 的值或者准备的 CAN\_MIDx 的值掩盖。如果被接受，消息的 ID 将被复制到 CAN\_MIDx 寄存器。

图 32-3 信息接受过程



如果一个邮箱是专门接收一个族有着不同 ID 的多个消息，定义在 CAN\_MAMx 寄存器的验收掩码必须被掩藏在 ID 族的变量部分。一旦消息被接收，应用必须编码在 CAN\_MIDs 寄存器的掩码位。为了高速的解码，掩码位被分为不同的 DI 族（CAN\_MEIDx）。

下面是一组消息 ID 被同一个邮箱处理的例子：

ID0 101000100100010010000100 0 11 00b  
ID1 101000100100010010000100 0 11 01b  
ID2 101000100100010010000100 0 11 10b  
ID3 101000100100010010000100 0 11 11b  
ID4 101000100100010010000100 1 11 00b  
ID5 101000100100010010000100 1 11 01b  
ID6 101000100100010010000100 1 11 10b  
ID7 101000100100010010000100 1 11 11b

邮箱 x 的 CAN\_MIDx 和 CAN\_MAMx 必须被初始化为相应的值：

CAN\_MIDx = 001 101000100100010010000100 x 11 xxb  
CAN\_MAMx = 001 111111111111111111111111 0 11 00b

如果邮箱 x 接收 ID6 的消息，那么 CAN\_MIDx 和 CAN\_MFIDx 将被置为：

CAN\_MIDx = 001 101000100100010010000100 1 11 10b  
CAN\_MFIDx = 00000000000000000000000000000110b

如果应用程序为每个消息关联处理程序，它可能定义一个指向相应处理行数的数组指针：

void (\*pHandler[8])(void);

当一个消息被接收，相应的处理程序可以挂载在 CAN\_MFIDx 寄存器，并且不需要检查掩码位：

unsigned int MFID0\_register;  
MFID0\_register = Get\_CAN\_MFID0\_Register();  
// Get\_CAN\_MFID0\_Register() returns the value of the CAN\_MFID0 register  
pHandler[MFID0\_register]();

#### 32.7.2.2 接收邮箱

当 CAN 模块接收一个消息时，它首先会寻找编号最小的那个邮箱和准备用于接收消息的邮箱。如果发现了这样的邮箱，消息将会被存储在它的数据寄存器中。根据配置，只要消息没有被应用程序接收（仅仅接收），或者，如果一个接收了一个相同的消息，那么它就会复写以前的消息（接收与复写）,最终邮箱会被禁用。

邮箱也可以被配置为消费者模式，在这个模式下，每发送一个请求之后，一个远程帧被自动发送。收到的第一个答案被存储在相应的邮箱数据寄存器。

多个邮箱可以被串联起来去接收缓冲区。它们在接收模式中必须被配置为相同的 ID，除了最后一个,它可以被配置为接收或复写模式。最后一个邮箱能被用来检测缓冲区溢出。

|  |  |
| --- | --- |
| 对象类型 | 描述 |
| 接收 | 接收的第一个消息被存储在邮箱数据寄存器，数据保持可用直到下一个传输请求 |
| 接收与复写 | 接收到的最后一个消息被存储在邮箱数据寄存器，下一个消息一直复写前一个。应用可以检查一个新消息是否可以被当前从邮箱数据寄存器读取的消息复写 |
| 消费者 | 一个远程帧被邮箱发送。发送的答案被存储在邮箱数据寄存器。这扩展了接收邮箱的特性，数据保持可用知道下一次传输请求 |

表 32-4 接收邮箱对象

#### 32.7.2.3 传输邮箱

当传输一个消息时，消息的长度和数据以正确的格式被写入传输邮箱。每一个传输邮箱都有一个权重标识符。控制器首先会自动发送权重高的（在 CAN\_MMRx 设置 PRIOR 字段）

也可以配置邮箱为生产者模式，在这种模式下，当一个远程帧被接收，邮箱数据会被自动发送。通过启用此模式，生产者可以只使用一个邮箱，而不是两个：一个检测远程帧和一个发送答案。

表 32-5 传输邮箱对象

|  |  |
| --- | --- |
| 对象类型 | 描述 |
| 传输 | 存储在邮箱数据寄存器的消息将努力赢得总线仲裁 |
| 生产者 | 在邮箱数据寄存器中编写的消息将在接收下一个远程帧后发送。这扩展传输邮箱功能 |