# 第 34 章 控制器局域网 （CAN）

## 目录

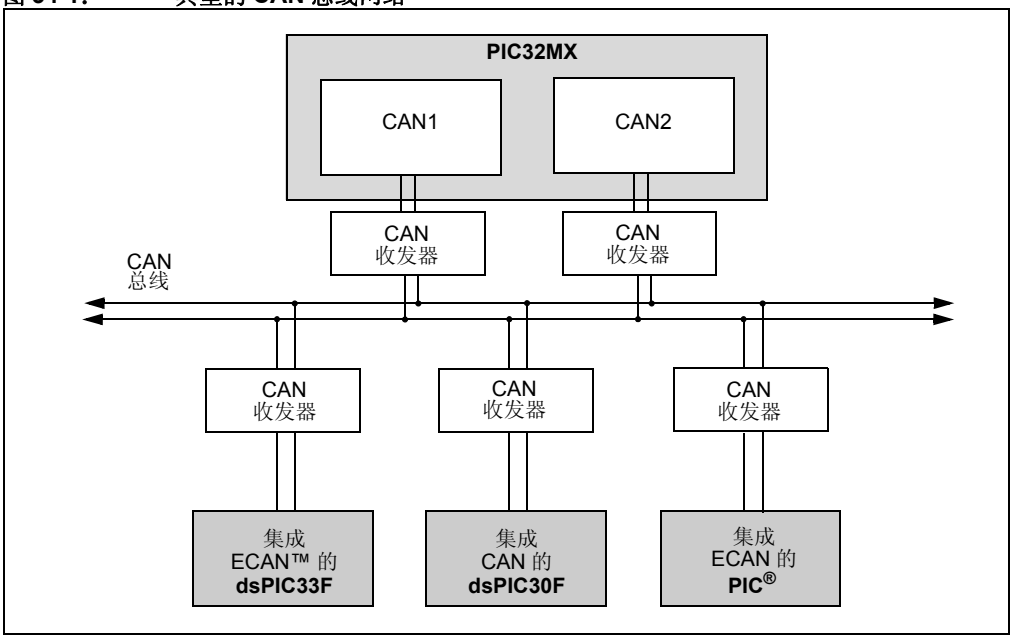
本章包括下列主题：


34.1 简介 ..... 34-2
34.2 CAN 报文格式 ..... 34-4
34.3 CAN 寄存器 ..... 34-9
34.4 使能和禁止 CAN 模块 ..... 34-52
34.5 CAN 模块工作模式 ..... 34-53
34.6 CAN 报文处理 ..... 34-55
34.7 发送 CAN 报文 ..... 34-62
34.8 CAN 报文过滤 ..... 34-74
34.9 接收 CAN 报文 ..... 34-80
34.10 位时序 ..... 34-88
34.11 CAN 错误管理 ..... 34-91
34.12 CAN 中断 ..... 34-94
34.13 CAN 接收报文时间标记 ..... 34-98
34.14 低功耗模式 ..... 34-98
34.15 相关应用笔记 ..... 34-100
34.16 版本历史 ..... 34-101

## 34.1 简介

PIC32MX 控制器局域网（Controller Area Network，CAN）模块实现了 CAN 2.0B 协议，该协议主要用于工业和汽车应用。该异步串行数据通信协议能在电气噪声环境下提供可靠的通信。PIC32MX 器件系列可以集成最多两个 CAN 模块。图 34-1 给出了典型 CAN 总线拓扑的图示。

**图 34-1： 典型的 CAN 总线网络**



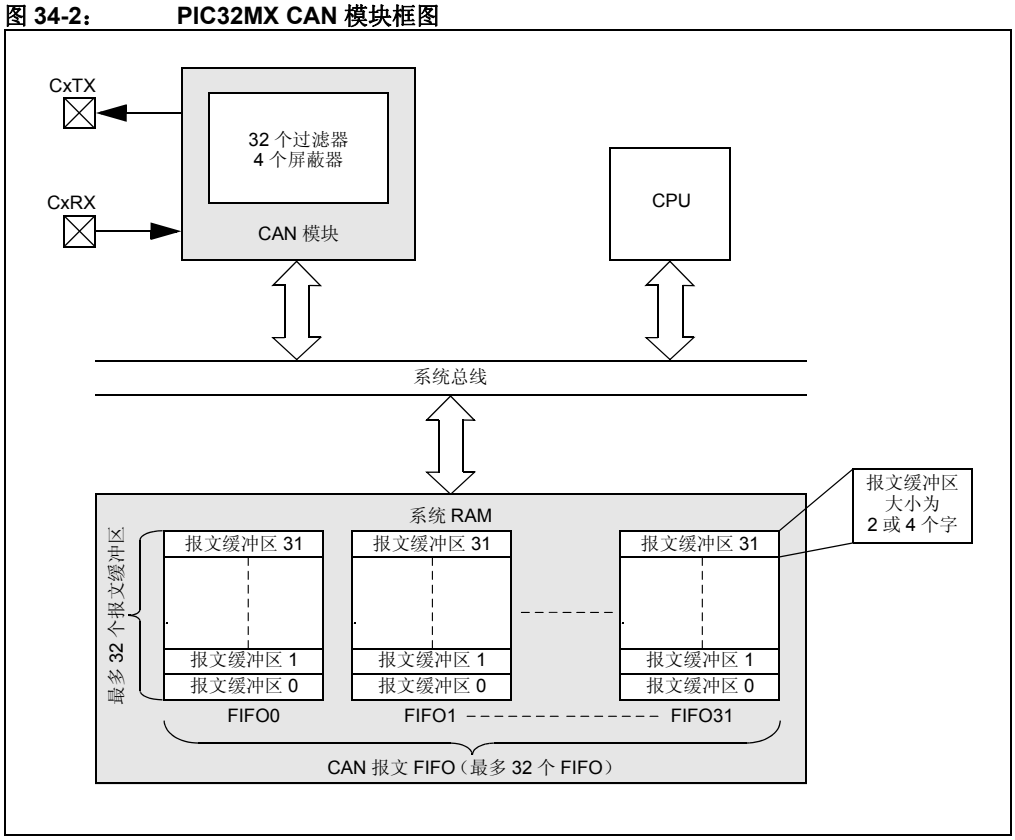

CAN 模块支持以下主要特性：

- 符合标准：
  - 完全符合 CAN 2.0B 协议
  - 最高 1 Mbps 的可编程比特率
- 报文接收和发送：
  - 32 个报文 FIFO
  - 每个 FIFO 最多可包含 32 个报文，总共可包含 1024 个报文
  - FIFO 可以作为发送报文 FIFO 或接收报文 FIFO
  - 用户可为用于发送的报文 FIFO 定义优先级
  - 32 个用于报文过滤的接收过滤器
  - 4 个用于报文过滤的接收过滤器屏蔽器寄存器
  - 自动响应远程发送请求
  - 支持 DeviceNet™ 寻址
- 其他特性：
  - 环回、监听所有报文和监听模式，用于自检、系统诊断和总线监视
  - 低功耗工作模式
  - CAN 模块是 PIC32MX 系统总线上的总线主机
  - 不需要使用 DMA
  - 专用的时间标记定时器
  - 仅数据报文接收模式

图 34-2 显示了 CAN 模块的大体结构。

**图 34-2： PIC32MX CAN 模块框图**




CAN 模块由协议引擎、报文接收过滤器和报文组合缓冲区组成。协议引擎通过 CAN 总线（按照 CAN 总线 2.0B 协议）发送和接收报文。所接收的报文先接收到接收报文组合缓冲区中。然后，接收到的报文通过报文接收过滤器进行过滤。发送报文组合缓冲区在协议引擎进行处理时存放待发送的报文。

CAN 报文缓冲区驻留在系统 RAM 中。CAN 模块中没有 CAN 报文缓冲区。因此，所有报文都存储在系统 RAM 中。CAN 模块是 PIC32MX 系统总线上的总线主机，它会根据需要向系统 RAM 读写数据。CAN 模块执行操作时不使用 DMA。CAN 模块可以在无需 DMA 或 CPU 干预的情况下，从系统 RAM 中取报文。

## 34.2 CAN 报文格式

CAN 总线协议使用异步通信。信息以数据帧的形式从发送器传递到接收器，数据帧由定义数据帧内容的字节字段组成，如图 34-3 所示。

每一帧都以帧起始（Start-of-Frame，SOF）位开始，以帧结束（End-of-Frame，EOF）位字段结束。SOF 位后面跟随仲裁和控制字段，它们标识报文类型、格式、长度和优先级。该信息使 CAN 总线上的每个节点都可以适当地响应报文。数据字段用于传送报文内容，其长度可变，范围为 0 至 8 字节。错误保护通过循环冗余校验（Cyclic Redundancy Check，CRC）和应答（Acknowledgement，ACK）字段来实现。

**图 34-3： CAN 总线报文帧**


CAN 总线协议支持 5 种帧类型：

- **数据帧**——包含发送器向接收器传送的数据
- **远程帧**——由总线上的某个节点发送，用于请求从另一个节点发送具有相同标识符的数据帧
- **错误帧**——由任意节点在检测到错误时发送
- **过载帧**——在连续的数据帧或远程帧之间提供额外的延时
- **帧间间隔**——在连续的帧之间提供间隔

CAN 2.0B 规范还定义了两种额外的数据格式：

- **标准数据帧**——用于使用 11 个标识符位的标准报文
- **扩展数据帧**——用于使用 29 个标识符位的扩展报文

CAN 总线规范有三种版本：

- **2.0A**——将 29 位标识符视为错误
- **2.0B Passive**——忽略 29 位标识符报文
- **2.0B Active**——处理 11 位和 29 位标识符

PIC32MX CAN 模块符合 CAN 2.0B Active 规范，同时增强了报文过滤功能。

| **注：** 关于 CAN 协议的详细信息，请参见 Bosch CAN 总线规范。 |
|---|

### 34.2.1 标准数据帧

标准数据帧报文以帧起始位开始，后面跟随一个 12 位的仲裁字段，如图 34-4 所示。仲裁字段包含一个 11 位的标识符和一个远程发送请求（Remote Transmit Request，RTR）位。标识符定义报文中包含的信息的类型，每个接收节点都通过它来确定报文是否属于自己感兴趣的内容。RTR 位用于区分数据帧和远程帧。对于标准数据帧， RTR 位清零。

仲裁字段之后是一个 6 位的控制字段，提供关于报文内容的更多信息。控制字段中的第一位是标识符扩展（Identifier Extension，IDE）位，用于区分报文是标准数据帧还是扩展数据帧。在发送 IDE 位期间，标准数据帧使用显性状态（逻辑电平 0）指示。控制字段中的第二位是保留（RB0）位，该位处于显性状态（逻辑电平 0）。控制字段中的最后 4 位表示数据长度编码（Data Length Code， DLC），它规定了报文中包含的数据字节数。

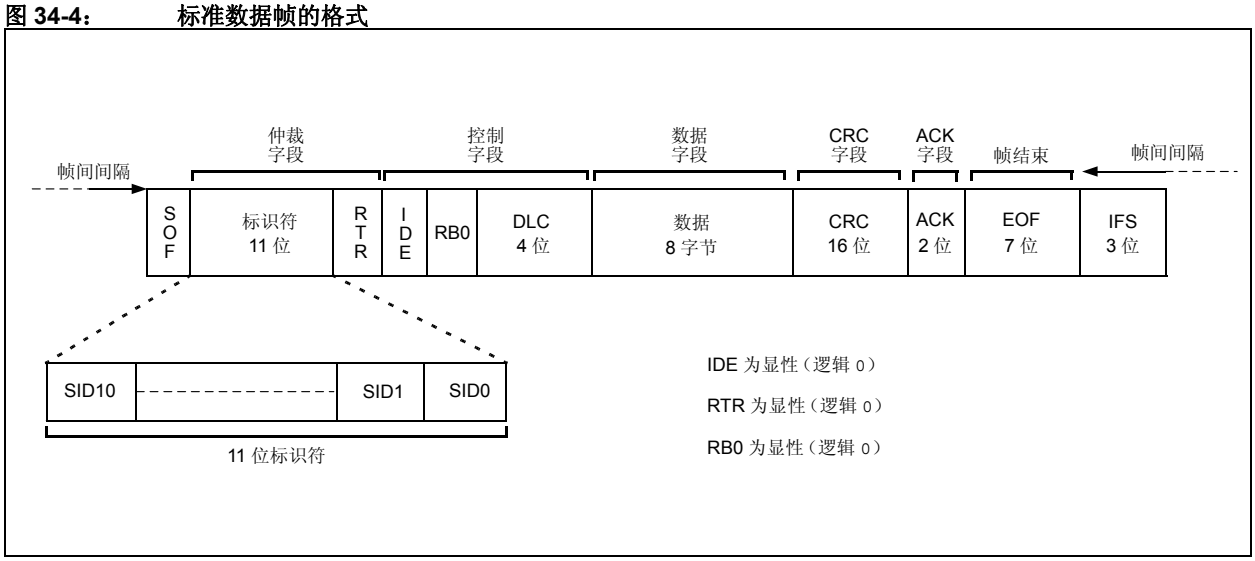
控制字段之后是数据字段。该字段承载报文数据——数据帧的实际有效载荷。该字段的长度可变，范围为 0 至 8 字节。字节数可由用户选择。

数据字段之后是循环冗余校验 （CRC）字段，由一个 15 位的 CRC 序列和一个定界符位组成。

应答（ACK）字段以隐性位（逻辑电平 1）发送，会被已正确接收数据的任意接收器改写为显性位。无论接收过滤器比较的结果如何，接收器总是会应答报文。

最后一个字段是帧结束字段，由 7 个隐性位组成，指示报文结束。

**图 34-4： 标准数据帧的格式**



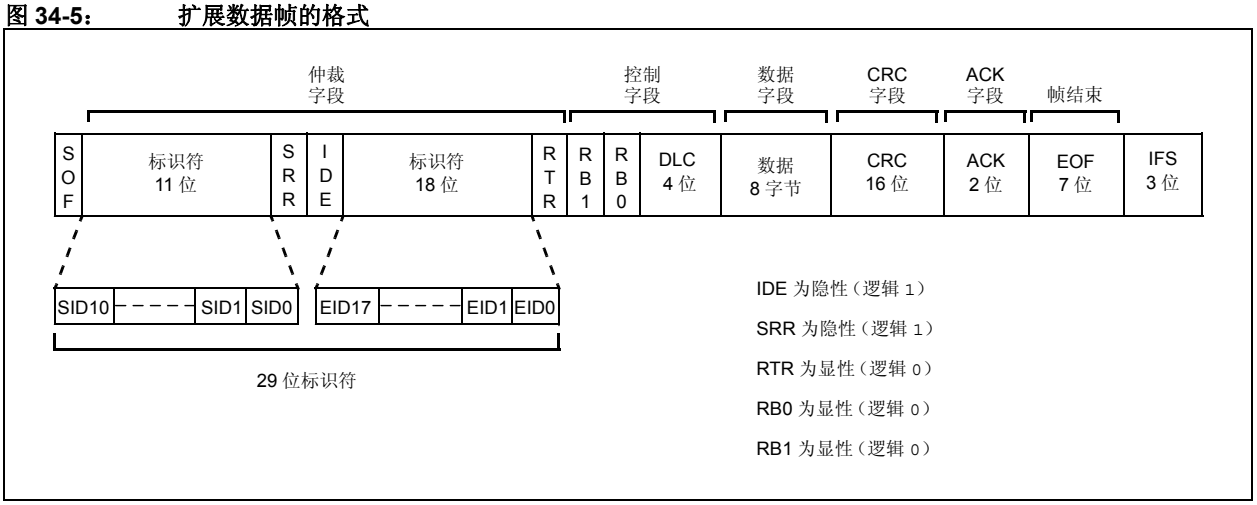
### 34.2.2 扩展数据帧

扩展数据帧以 SOF 位开始，后面跟随一个 31 位的仲裁字段，如图 34-5 所示。扩展数据帧的仲裁字段包含 29 个标识符位，这些位分布在由替代远程请求（Substitute Remote Request，SRR）位和 IDE 位隔开的两个字段中。 SRR 位决定报文是否是远程帧。对于扩展数据帧， SRR = 1。IDE 位指示数据帧类型。对于扩展数据帧， IDE = 1。

扩展数据帧的控制字段由 7 位组成。第一位是 RTR。对于扩展数据帧， RTR = 0。接下来两位 RB1 和 RB0 是保留位，处于显性状态（逻辑电平 0）。控制字段中的最后 4 位是数据长度编码，它规定了报文中包含的数据字节数。

扩展数据帧中的其余字段在结构上与标准数据帧相同。

**图 34-5： 扩展数据帧的格式**



### 34.2.3 远程帧

希望从另一个节点接收数据的节点可以通过发送远程帧来启动源节点发送相应数据。远程帧可以是标准格式 （图 34-6）或扩展格式 （图 34-7）。

除以下各项外，远程帧类似于数据帧：

- RTR 位为隐性 （RTR = 1）
- 没有数据字段 （DLC = 0）

**图 34-6： 标准远程帧的格式**

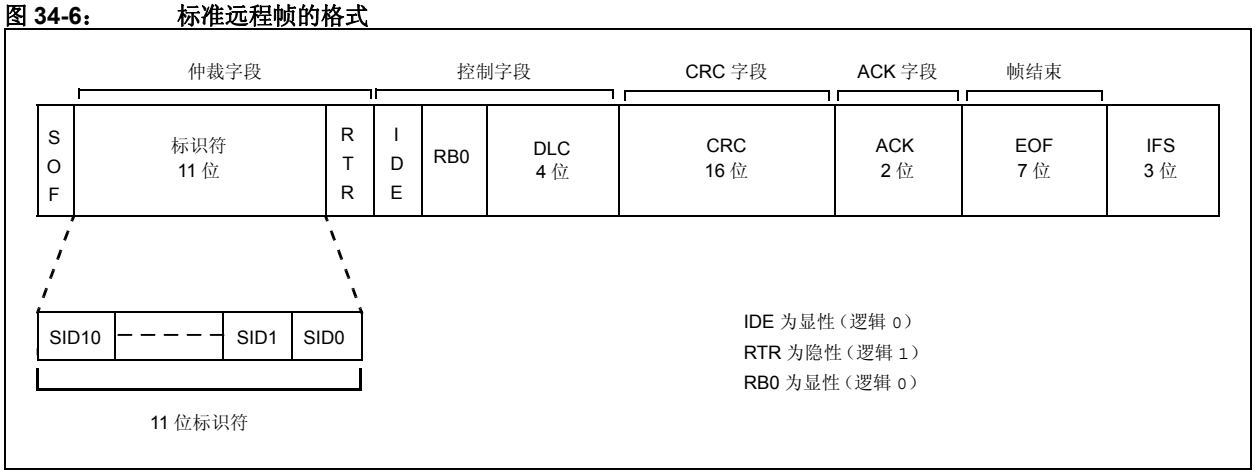
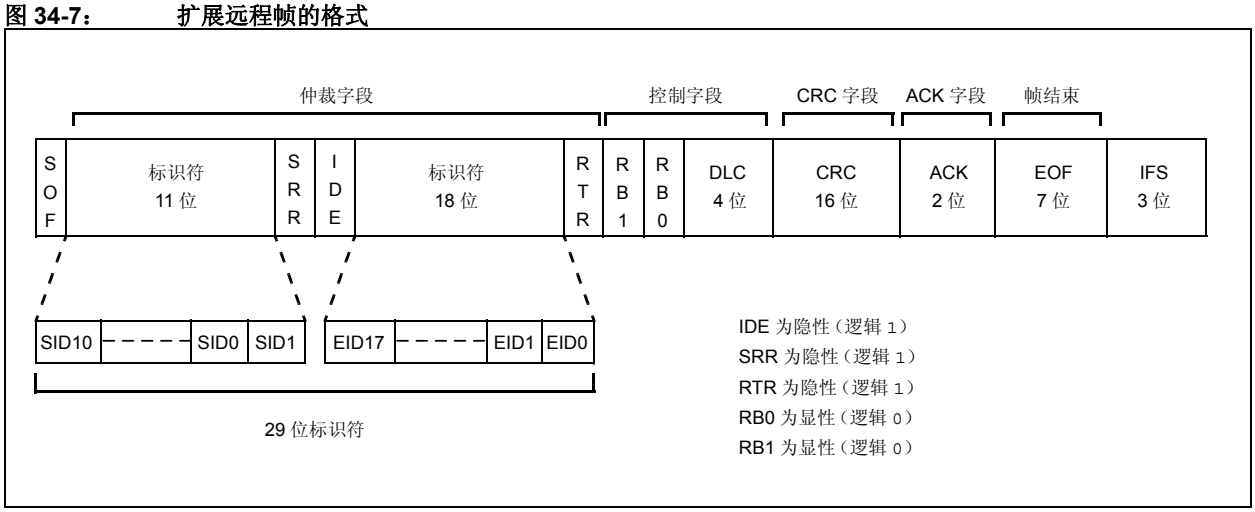



**图 34-7： 扩展远程帧的格式**



### 34.2.4 错误帧

错误帧是由检测到总线错误的任一节点产生的。错误帧由一个错误标志字段和其后的一个错误定界符字段组成。错误定界符由 8 个隐性位组成，允许总线节点在错误发生后重新启动总线通信。有两种错误标志字段，具体是哪一种取决于检测到错误的节点的错误状态：

- **错误主动标志**——包含 6 个连续的显性位，强制网络上的所有其他节点产生错误回应标志，从而在总线上产生 6 至 12 个连续显性位。
- **错误被动标志**——包含 6 个连续的隐性位，因此除非总线错误被正在发送的节点检测到，否则错误被动标志的发送将不会影响网络上任何其他节点的通信。

### 34.2.5 过载帧

节点可以在以下情况下产生过载帧：在帧间间隔中检测到显性位，或者节点尚未准备好接收下一个报文（例如，如果它仍在读取先前接收到的报文）。过载帧与带主动错误标志的错误帧具有相同的格式，但过载帧只能在帧间间隔中产生。它由包含 6 个显性位的过载标志字段和其后的包含 8 个隐性位的过载定界符字段组成。节点最多可产生两个连续过载帧来延迟下一条报文的接收。

### 34.2.6 帧间间隔

帧间间隔用于分隔开在 CAN 总线上发送的连续帧。它由至少 3 个隐性位组成，又称为间断（Intermission）。帧间间隔允许节点在开始下一个帧之前有时间对先前接收到的报文进行内部处理。如果发送节点处于错误被动状态，则会在帧间间隔中另外插入 8 个隐性位，然后该节点才会发送任何其他报文。该时段称为暂停发送字段，允许其他发送节点有时间获得总线的控制权。

## 34.3 CAN 寄存器

CAN 模块寄存器可以按照它们的功能分为以下几组：

- 模块和 CAN 比特率配置寄存器
- 中断和状态寄存器
- 屏蔽器和过滤器配置寄存器
- FIFO 控制寄存器

### 34.3.1 模块和 CAN 比特率配置寄存器

| 注： | 寄存器标识符中的 “i” 表示 CAN1 或 CAN2。 |
|---|---|

- **CiCON：CAN 控制寄存器**

  该寄存器用于设置 CAN 模块工作模式和 DeviceNet 寻址。

- **CiCFG：CAN 波特率配置寄存器**

  该寄存器包含一些控制位，它们用于设置每个时间量的周期 （通过使用波特率预分频比），并指定以时间份额数表示的同步跳转宽度 （Synchronization Jump Width， SJW）。该寄存器还用于设定每个CAN位时间段（包括传播时间段以及相位缓冲段1和2）中的时间份额数。

### 34.3.2 中断和状态寄存器

- **CiINT：CAN 中断寄存器**

  该寄存器可用于使能和禁止各种 CAN 模块中断源。它还包含了一些中断状态标志。

- **CiVEC：CAN 中断编码寄存器**

  该寄存器提供了一些状态位，这些状态位可提供关于 CAN 模块中断源和选中报文过滤器的信息。这些值可用于实现用以处理不同情况的跳转表。

- **CiTREC：CAN 发送 / 接收错误计数寄存器**

  该寄存器提供关于发送和接收错误计数器值的信息。它还具有一些指示各种警告状态的位。

- **CiFSTAT：CAN FIFO 状态寄存器**

  该寄存器包含所有 FIFO 的中断状态标志。

- **CiRXOVF：CAN 接收 FIFO 溢出状态寄存器**

  该寄存器包含所有 FIFO 的溢出中断状态标志。

- **CiTMR：CAN 定时器寄存器**

  该寄存器包含 CAN 报文时间标记定时器和预分频器。

### 34.3.3 屏蔽器和过滤器配置寄存器

- **CiRXMn：CAN 接收过滤器屏蔽器 n 寄存器 （n = 0、 1、 2 或 3）**

  这些寄存器可用于配置过滤器屏蔽器。共有 4 个屏蔽器可供使用。

- **CiFLTCONn：CAN 接收过滤器控制寄存器 n （n = 0 至 7）**

  这些寄存器可用于将 FIFO 和屏蔽器与过滤器相关联。过滤器可以与任一屏蔽器关联。它也包含了过滤器使能 / 禁止位。

- **CiRXFn：CAN 接收过滤器 n 寄存器 （n = 0 至 31）**

  这些寄存器用于指定要应用于接收报文的过滤器。共有 32 个过滤器可供使用。

### 34.3.4 FIFO 控制寄存器

- **CiFIFOBA：CAN 报文缓冲区基址寄存器**

  该寄存器保存 CAN 报文缓冲区的基址 （起始地址）。该地址是物理地址。

- **CiFIFOCONn：CAN FIFO 控制寄存器 n （n = 0 至 31）**

  这些寄存器可用于控制和配置 CAN 报文 FIFO。

- **CiFIFOINTn：CAN FIFO 中断寄存器 n （n = 0 至 31）**

  这些寄存器可用于使能或禁止各个 FIFO 中断源。它们还包含了一些中断状态位。

- **CiFIFOUAn：CAN FIFO 用户地址寄存器 n （n = 0 至 31）**

  这些寄存器提供 CAN 报文 FIFO 中用于读取下一个报文或写入下一个报文的存储单元的地址。

- **CiFIFOCIn：CAN 模块报文索引寄存器 n （n = 0 至 31）**

  这些寄存器提供 CAN 模块将发送的下一个报文的报文缓冲区索引，或者下一个接收报文的保存位置的报文缓冲区索引 （在报文 FIFO 中）。

表 34-1 汇总了所有与 CAN 相关的寄存器。该汇总表之后列出了相应的寄存器，并且每个寄存器均附有详细的说明。所有未实现的寄存器和 / 或寄存器中的位都读为 0。

表 34-1： CAN 控制器寄存器汇总

| 地址偏移 | 名称 | 位范围 | Bit 31/2315/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CiCON(1,2,3) | 31:24 | — | — | — | — | ABAT | REQOP<2:0> | | |
| | | 23:16 | OPMOD<2:0> | | | CANCAP | — | — | — | — |
| | | 15:8 | ON | — | SIDLE | — | CANBUSY | — | — | — |
| | | 7:0 | — | — | — | DNCNT<4:0> | | | | |
| 0x10 | CiCFG(1,2,3) | 31:24 | — | — | — | — | — | — | — | — |
| | | 23:16 | — | WAKFIL | — | — | — | SEG2PH<2:0> | | |
| | | 15:8 | SEG2PHTS | SAM | SEG1PH<2:0> | | | PRSEG<2:0> | | |
| | | 7:0 | SJW<1:0> | | BRP<5:0> | | | | | |
| 0x20 | CiINT(1,2,3) | 31:24 | IVRIE | WAKIE | CERRIE | SERRIE | RBOVIE | — | — | — |
| | | 23:16 | — | — | — | — | MODIE | CTMRIE | RBIE | TBIE |
| | | 15:8 | IVRIF | WAKIF | CERRIF | SERRIF | RBOVIF | — | — | — |
| | | 7:0 | — | — | — | — | MODIF | CTMRIF | RBIF | TBIF |
| 0x30 | CiVEC(1,2,3) | 31:24 | — | — | — | — | — | — | — | — |
| | | 23:16 | — | — | — | — | — | — | — | — |
| | | 15:8 | — | — | — | FILHIT<4:0> | | | | |
| | | 7:0 | — | ICOD<6:0> | | | | | | |
| 0x40 | CiTREC(1,2,3) | 31:24 | — | — | — | — | — | — | — | — |
| | | 23:16 | — | — | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN |
| | | 15:8 | TEC<7:0> | | | | | | | |
| | | 7:0 | REC<7:0> | | | | | | | |
| 0x50 | CiFSTAT(1,2,3) | 31:24 | FIFOIP31 | FIFOIP30 | FIFOIP29 | FIFOIP28 | FIFOIP27 | FIFOIP26 | FIFOIP25 | FIFOIP24 |
| | | 23:16 | FIFOIP23 | FIFOIP22 | FIFOIP21 | FIFOIP20 | FIFOIP19 | FIFOIP18 | FIFOIP17 | FIFOIP16 |
| | | 15:8 | FIFOIP15 | FIFOIP14 | FIFOIP13 | FIFOIP12 | FIFOIP11 | FIFOIP10 | FIFOIP9 | FIFOIP8 |
| | | 7:0 | FIFOIP7 | FIFOIP6 | FIFOIP5 | FIFOIP4 | FIFOIP3 | FIFOIP2 | FIFOIP1 | FIFOIP0 |
| 0x60 | CiRXOVF(1,2,3) | 31:24 | RXOVF31 | RXOVF30 | RXOVF29 | RXOVF28 | RXOVF27 | RXOVF26 | RXOVF25 | RXOVF24 |
| | | 23:16 | RXOVF23 | RXOVF22 | RXOVF21 | RXOVF20 | RXOVF19 | RXOVF18 | RXOVF17 | RXOVF16 |
| | | 15:8 | RXOVF15 | RXOVF14 | RXOVF13 | RXOVF12 | RXOVF11 | RXOVF10 | RXOVF9 | RXOVF8 |
| | | 7:0 | RXOVF7 | RXOVF6 | RXOVF5 | RXOVF4 | RXOVF3 | RXOVF2 | RXOVF1 | RXOVF0 |
| 0x70 | CiTMR(1,2,3) | 31:24 | CANTS<15:8> | | | | | | | |
| | | 23:16 | CANTS<7:0> | | | | | | | |
| | | 15:8 | CANTSPRE<15:8> | | | | | | | |
| | | 7:0 | CANTSPRE<7:0> | | | | | | | |
| 0x80 | CiRXM0(1,2,3) | 31:24 | SID<10:3> | | | | | | | |
| | | 23:16 | SID<2:0> | | | -— | MIDE | — | EID<17:16> | |
| | | 15:8 | EID<15:8> | | | | | | | |
| | | 7:0 | EID<7:0> | | | | | | | |
| 0x90 | CiRXM1(1,2,3) | 31:24 | SID<10:3> | | | | | | | |
| | | 23:16 | SID<2:0> | | | -— | MIDE | — | EID<17:16> | |
| | | 15:8 | EID<15:8> | | | | | | | |
| | | 7:0 | EID<7:0> | | | | | | | |
| 0xA0 | CiRXM2(1,2,3) | 31:24 | SID<10:3> | | | | | | | |
| | | 23:16 | SID<2:0> | | | -— | MIDE | — | EID<17:16> | |
| | | 15:8 | EID<15:8> | | | | | | | |
| | | 7:0 | EID<7:0> | | | | | | | |

**注 1：** 该寄存器具有关联的清零寄存器，位于0x4字节偏移处。这些清零寄存器的命名方式是在关联寄存器的名称末尾附加CLR（例如，CiCONCLR）。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置1寄存器，位于0x8字节偏移处。这些置1寄存器的命名方式是在关联寄存器的名称末尾附加SET（例如，CiCONSET）。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器，位于 0xC 字节偏移处。这些取反寄存器的命名方式是在关联寄存器的名称末尾附加 INV（例如，CiCONINV）。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

表 34-1： CAN 控制器寄存器汇总 （续）

| 地址偏移 | 名称 | 位范围 | Bit 31/2315/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xB0 | CiRXM3(1,2,3) | 31:24 | SID<10:3> | | | | | | | |
| | | 23:16 | SID<2:0> | | | -— | MIDE | — | EID<17:16> | |
| | | 15:8 | EID<15:8> | | | | | | | |
| | | 7:0 | EID<7:0> | | | | | | | |
| 0xC0 | CiFLTCON0(1,2,3) | 31:24 | FLTEN3 | MSEL3<1:0> | | FSEL3<4:0> | | | | |
| | | 23:16 | FLTEN2 | MSEL2<1:0> | | FSEL2<4:0> | | | | |
| | | 15:8 | FLTEN1 | MSEL1<1:0> | | FSEL1<4:0> | | | | |
| | | 7:0 | FLTEN0 | MSEL0<1:0> | | FSEL0<4:0> | | | | |
| 0xD0 | CiFLTCON1(1,2,3) | 31:24 | FLTEN7 | MSEL7<1:0> | | FSEL7<4:0> | | | | |
| | | 23:16 | FLTEN6 | MSEL6<1:0> | | FSEL6<4:0> | | | | |
| | | 15:8 | FLTEN5 | MSEL5<1:0> | | FSEL5<4:0> | | | | |
| | | 7:0 | FLTEN4 | MSEL4<1:0> | | FSEL4<4:0> | | | | |
| 0xE0 | CiFLTCON2(1,2,3) | 31:24 | FLTEN11 | MSEL11<1:0> | | FSEL11<4:0> | | | | |
| | | 23:16 | FLTEN10 | MSEL10<1:0> | | FSEL10<4:0> | | | | |
| | | 15:8 | FLTEN9 | MSEL9<1:0> | | FSEL9<4:0> | | | | |
| | | 7:0 | FLTEN8 | MSEL8<1:0> | | FSEL8<4:0> | | | | |
| 0xF0 | CiFLTCON3(1,2,3) | 31:24 | FLTEN15 | MSEL15<1:0> | | FSEL15<4:0> | | | | |
| | | 23:16 | FLTEN14 | MSEL14<1:0> | | FSEL14<4:0> | | | | |
| | | 15:8 | FLTEN13 | MSEL13<1:0> | | FSEL13<4:0> | | | | |
| | | 7:0 | FLTEN12 | MSEL12<1:0> | | FSEL12<4:0> | | | | |
| 0x100 | CiFLTCON4(1,2,3) | 31:24 | FLTEN19 | MSEL19<1:0> | | FSEL19<4:0> | | | | |
| | | 23:16 | FLTEN18 | MSEL18<1:0> | | FSEL18<4:0> | | | | |
| | | 15:8 | FLTEN17 | MSEL17<1:0> | | FSEL17<4:0> | | | | |
| | | 7:0 | FLTEN16 | MSEL16<1:0> | | FSEL16<4:0> | | | | |
| 0x110 | CiFLTCON5(1,2,3) | 31:24 | FLTEN23 | MSEL23<1:0> | | FSEL23<4:0> | | | | |
| | | 23:16 | FLTEN22 | MSEL22<1:0> | | FSEL22<4:0> | | | | |
| | | 15:8 | FLTEN21 | MSEL21<1:0> | | FSEL21<4:0> | | | | |
| | | 7:0 | FLTEN20 | MSEL20<1:0> | | FSEL20<4:0> | | | | |
| 0x120 | CiFLTCON6(1,2,3) | 31:24 | FLTEN27 | MSEL27<1:0> | | FSEL27<4:0> | | | | |
| | | 23:16 | FLTEN26 | MSEL26<1:0> | | FSEL26<4:0> | | | | |
| | | 15:8 | FLTEN25 | MSEL25<1:0> | | FSEL25<4:0> | | | | |
| | | 7:0 | FLTEN24 | MSEL24<1:0> | | FSEL24<4:0> | | | | |
| 0x130 | CiFLTCON7(1,2,3) | 31:24 | FLTEN31 | MSEL31<1:0> | | FSEL31<4:0> | | | | |
| | | 23:16 | FLTEN30 | MSEL30<1:0> | | FSEL30<4:0> | | | | |
| | | 15:8 | FLTEN29 | MSEL29<1:0> | | FSEL29<4:0> | | | | |
| | | 7:0 | FLTEN28 | MSEL28<1:0> | | FSEL28<4:0> | | | | |
| 0x140 | CiRXFn(1,2,3) （n = 0 至 31） | 31:24 | SID<10:3> | | | | | | | |
| | | 23:16 | SID<2:0> | | | -— | EXID | — | EID<17:16> | |
| | | 15:8 | EID<15:8> | | | | | | | |
| | | 7:0 | EID<7:0> | | | | | | | |
| 0x340 | CiFIFOBA(1,2,3) | 31:24 | CiFIFOBA<31:24> | | | | | | | |
| | | 23:16 | CiFIFOBA<23:16> | | | | | | | |
| | | 15:8 | CiFIFOBA<15:8> | | | | | | | |
| | | 7:0 | CiFIFOBA<7:0> | | | | | | | |
| 0x350 | CiFIFOCONn(1,2,3) （n = 0 至 31） | 31:24 | — | — | — | — | — | — | — | — |
| | | 23:16 | — | — | — | FSIZE<4:0> | | | | |
| | | 15:8 | — | FRESET | UINC | DONLY | — | — | — | — |
| | | 7:0 | TXEN | TXABAT | TXLARB | TXERR | TXREQ | RTREN | TXPRI<1:0> | |

注 1： 该寄存器具有关联的清零寄存器，位于0x4字节偏移处。这些清零寄存器的命名方式是在关联寄存器的名称末尾附加CLR（例如，CiCONCLR）。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置1寄存器，位于0x8字节偏移处。这些置1寄存器的命名方式是在关联寄存器的名称末尾附加SET（例如，CiCONSET）。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器，位于 0xC 字节偏移处。这些取反寄存器的命名方式是在关联寄存器的名称末尾附加 INV（例如，CiCONINV）。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

表 34-1： CAN 控制器寄存器汇总 （续）

| 地址偏移 | 名称 | 位范围 | Bit 31/2315/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x360 | CiFIFOINTn(1,2,3)（n = 0 至 31） | 31:24 | — | — | — | — | — | TXNFULLIE | TXHALFIE | TXEMPTYIE |
| | | 23:16 | — | — | — | — | RXOVFLIE | RXFULLIE | RXHALFIE | RXNEMPTYIE |
| | | 15:8 | — | — | — | — | — | TXNFULLIF | TXHALFIF | TXEMPTYIF |
| | | 7:0 | — | — | — | — | RXOVFLIF | RXFULLIF | RXHALFIF | RXNEMPTYIF |
| 0x370 | CiFIFOUAn(1,2,3)（n = 0 至 31） | 31:24 | CiFIFOUA<31:24> | | | | | | | |
| | | 23:16 | CiFIFOUA<23:16> | | | | | | | |
| | | 15:8 | CiFIFOUA<15:8> | | | | | | | |
| | | 7:0 | CiFIFOUA<7:0> | | | | | | | |
| 0x380 | CiFIFOCIn(1,2,3)（n = 0 至 31） | 31:24 | — | — | — | — | — | — | — | — |
| | | 23:16 | — | — | — | — | — | — | — | — |
| | | 15:8 | — | — | — | — | — | — | — | — |
| | | 7:0 | — | — | — | CiFIFOCI<4:0> | | | | |

**注 1：** 该寄存器具有关联的清零寄存器，位于0x4字节偏移处。这些清零寄存器的命名方式是在关联寄存器的名称末尾附加CLR（例如，CiCONCLR）。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置1寄存器，位于0x8字节偏移处。这些置1寄存器的命名方式是在关联寄存器的名称末尾附加SET（例如，CiCONSET）。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器，位于 0xC 字节偏移处。这些取反寄存器的命名方式是在关联寄存器的名称末尾附加 INV（例如，CiCONINV）。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**寄存器 34-1： CiCON：CAN 控制寄存器 (1,2,3)**

| U-0 | U-0 | U-0 | U-0 | S/HC-0 | R/W-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | ABAT | REQOP<2:0> | | |
| bit 31 | | | | | | | bit 24 |

| R-1 | R-0 | R-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| OPMOD<2:0> | | | CANCAP | — | — | — | — |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | r-0 | R/W-0 | U-0 | R-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| ON(4) | — | SIDLE | — | CANBUSY | — | — | — |
| bit 15 | | | | | | | bit 8 |

| U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | DNCNT<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| **图注：** | HC = 硬件清零位 | S = 可置 1 位 | r = 保留 |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-28 **未实现：** 读为 0

bit 27 **ABAT：** 中止所有等待处理的发送位
1 = 通知所有发送缓冲区中止发送
0 = 模块将在所有发送中止时清零该位

bit 26-24 **REQOP<2:0>：** 请求工作模式位
111 = 设置监听所有报文模式
110 = 保留——不使用
101 = 保留——不使用
100 = 设置配置模式
011 = 设置监听模式
010 = 设置环回模式
001 = 设置禁止模式
000 = 设置正常工作模式

**注 1：** 该寄存器具有关联的清零寄存器 （CiCONCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器 （CiCONSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（CiCONINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 如果用户应用程序清零该位，则可能需要先经过一定周期之后，CAN 模块才会完成当前事务并响应该请求。用户应用程序应通过查询 CANBUSY 位来确认是否已履行请求。

**寄存器 34-1： CiCON：CAN 控制寄存器 (1,2,3) （续）**

bit 23-21 **OPMOD<2:0>**：工作模式状态位
111 = 模块工作在监听所有报文模式下
110 = 保留
101 = 保留
100 = 模块工作在配置模式下
011 = 模块工作在监听模式下
010 = 模块工作在环回模式下
001 = 模块工作在禁止模式下
000 = 模块工作在正常工作模式下

bit 20 **CANCAP**：CAN 报文接收时间标记定时器捕捉使能位
1 = 在接收到有效报文时 CANTMR 值随报文一起存储
0 = 禁止 CAN 报文接收时间标记定时器捕捉，并停止 CANTMR 以降低功耗

bit 19-16 **未实现**：读为 0

bit 15 **ON**：CAN 使能位 (4)
1 = 使能 CAN 模块
0 = 禁止 CAN 模块

bit 14 **保留**：不使用

bit 13 **SIDLE**：CAN 空闲模式停止位
1 = 当系统进入空闲模式时， CAN 停止工作
0 = 当系统进入空闲模式时， CAN 继续工作

bit 12 **未实现**：读为 0

bit 11 **CANBUSY**：CAN 模块忙状态位
1 = CAN 模块处于活动状态
0 = 已完全禁止 CAN 模块

bit 10-5 **未实现**：读为 0

bit 4-0 **DNCNT<4:0>**：DeviceNet 过滤器位编号位
10011-11111 = 无效选择 （最多可将数据的 18 位与 EID 作比较）
10010 = 最多可将数据字节 2 的 bit 6 与 EID17 （CiRXFn<17>）作比较
•
•
•
00001 = 最多可将数据字节 0 的 bit 7 与 EID0 （CiRXFn<0>）作比较
00000 = 不比较数据字节

**注 1**：该寄存器具有关联的清零寄存器 （CiCONCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器 （CiCONSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器（CiCONINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：如果用户应用程序清零该位，则可能需要先经过一定周期之后，CAN 模块才会完成当前事务并响应该请求。用户应用程序应通过查询 CANBUSY 位来确认是否已履行请求。

**寄存器 34-2： CiCFG：CAN 波特率配置寄存器 (1,2,3,4)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 31 | | | | | | | bit 24 |

| U-0 | R/W-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | WAKFIL | — | — | — | SEG2PH<2:0>(5,8) | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| SEG2PHTS(5) | SAM(6) | SEG1PH<2:0> | | | PRSEG<2:0> | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| SJW<1:0>(7) | | BRP<5:0> | | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-23 **未实现：**读为 0

bit 22 **WAKFIL：**CAN 总线线路滤波器使能位
1 = 使用 CAN 总线线路滤波器来唤醒
0 = 不使用 CAN 总线线路滤波器来唤醒

bit 21-19 **未实现：**读为 0

bit 18-16 **SEG2PH<2:0>：**相位缓冲段 2 位 (5,8)
111 = 长度为 8 x TQ
•
•
•
000 = 长度为 1 x TQ

**注 1：**该寄存器具有关联的清零寄存器（CiCFGCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：**该寄存器具有关联的置 1 寄存器（CiCFGSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：**该寄存器具有关联的取反寄存器（CiCFGINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：**仅当 CAN 模块处于配置模式 （OPMOD<2:0> （CiCON<23:21>） = 100）时，才能修改该寄存器。

**5：**SEG2PH ≤ SEG1PH。如果 SEG2PHTS 清零，则 SEG2PH 将自动置 1。

**6：**BRP < 2 时不允许 3 次位采样。

**7：**SJW ≤ SEG2PH。

**8：**每位的时间份额必须大于 7 （即， TQBIT > 7）。

**寄存器 34-2： CiCFG：CAN 波特率配置寄存器 (1,2,3,4) （续）**

bit 15 **SEG2PHTS**：相位缓冲段 2 时间选择位 (5)
1 = 可自由编程
0 = SEG1PH 的最大值与信息处理时间中的较大值

bit 14 **SAM**：CAN 总线线路采样位 (6)
1 = 在采样点对总线线路采样三次
0 = 在采样点对总线线路采样一次

bit 13-11 **SEG1PH<2:0>**：相位缓冲段 1 位 (8)
111 = 长度为 8 x TQ
•
•
•
000 = 长度为 1 x TQ

bit 10-8 **PRSEG<2:0>**：传播时间段位 (8)
111 = 长度为 8 x TQ
•
•
•
000 = 长度为 1 x TQ

bit 7-6 **SJW<1:0>**：同步跳转宽度位 (7)
11 = 长度为 4 x TQ
10 = 长度为 3 x TQ
01 = 长度为 2 x TQ
00 = 长度为 1 x TQ

bit 5-0 **BRP<5:0>**：波特率预分频比位
111111 = TQ = (2 x 64)/FSYS
111110 = TQ = (2 x 63)/FSYS
•
•
•
000001 = TQ = (2 x 2)/FSYS
000000 = TQ = (2 x 1)/FSYS

**注 1**：该寄存器具有关联的清零寄存器（CiCFGCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiCFGSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器（CiCFGINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当 CAN 模块处于配置模式 （OPMOD<2:0> （CiCON<23:21>） = 100）时，才能修改该寄存器。

**5**：SEG2PH ≤ SEG1PH。如果 SEG2PHTS 清零，则 SEG2PH 将自动置 1。

**6**：BRP < 2 时不允许 3 次位采样。

**7**：SJW ≤ SEG2PH。

**8**：每位的时间份额必须大于 7 （即， TQBIT > 7）。

**寄存器 34-3： CiINT：CAN 中断寄存器 (1,2,3)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| IVRIE | WAKIE | CERRIE | SERRIE | RBOVIE | — | — | — |
| bit 31 | | | | | | | bit 24 |

| U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | MODIE | CTMRIE | RBIE | TBIE |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| IVRIF | WAKIF | CERRIF | SERRIF(4) | RBOVIF | — | — | — |
| bit 15 | | | | | | | bit 8 |

| U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | MODIF | CTMRIF | RBIF | TBIF |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31 **IVRIE**：收到无效报文中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 30 **WAKIE**：CAN 总线活动唤醒中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 29 **CERRIE**：CAN 总线错误中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 28 **SERRIE**：系统错误中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 27 **RBOVIE**：接收缓冲区溢出中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 26-20 **未实现**：读为 0

**注 1**：该寄存器具有关联的清零寄存器（CiINTCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiINTSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器（CiINTINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：只能通过将 ON 位 （CiCON<15>）清零或置 1，从而关闭或打开 CAN 模块的方式来清零该位。

**寄存器 34-3： CiINT：CAN 中断寄存器 (1,2,3) （续）**

bit 19 **MODIE**：模式改变中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 18 **CTMRIE**：CAN 时间标记定时器中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 17 **RBIE**：接收缓冲区中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 16 **TBIE**：发送缓冲区中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 15 **IVRIF**：收到无效报文中断标志位
1 = 发生了无效报文中断
0 = 未发生无效报文中断

bit 14 **WAKIF**：CAN 总线活动唤醒中断标志位
1 = 发生了总线唤醒活动中断
0 = 未发生总线唤醒活动中断

bit 13 **CERRIF**：CAN 总线错误中断标志位
1 = 发生了 CAN 总线错误
0 = 未发生 CAN 总线错误

bit 12 **SERRIF**：系统错误中断标志位 (4)
1 = 发生了系统错误 （通常是向系统总线送入了非法地址）
0 = 未发生系统错误

bit 11 **RBOVIF**：接收缓冲区溢出中断标志位
1 = 发生了接收缓冲区溢出
0 = 未发生接收缓冲区溢出

bit 10-4 **未实现**：读为 0

bit 3 **MODIF**：CAN 模式改变中断标志位
1 = 发生了 CAN 模块模式改变 （OPMOD<2:0> 已改变，以反映 REQOP）
0 = 未发生 CAN 模块模式改变

bit 2 **CTMRIF**：CAN 定时器溢出中断标志位
1 = 发生了 CAN 定时器 （CANTMR）溢出
0 = 未发生 CAN 定时器 （CANTMR）溢出

bit 1 **RBIF**：接收缓冲区中断标志位
1 = 有一个接收缓冲区中断等待处理
0 = 没有待处理的接收缓冲区中断

bit 0 **TBIF**：发送缓冲区中断标志位
1 = 有一个发送缓冲区中断等待处理
0 = 没有待处理的发送缓冲区中断

**注 1**：该寄存器具有关联的清零寄存器（CiINTCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiINTSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器（CiINTINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：只能通过将 ON 位 （CiCON<15>）清零或置 1，从而关闭或打开 CAN 模块的方式来清零该位。

**寄存器 34-4： CiVEC：CAN 中断编码寄存器 (1,2,3)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 31 | | | | | | | bit 24 |

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 23 | | | | | | | bit 16 |

| U-0 | U-0 | U-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | FILHIT<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| U-0 | R-1 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| — | ICOD<6:0>(4) | | | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-13 **未实现：**读为 0

bit 12-8 **FILHIT<4:0>：**选中过滤器的编号位

11111 = 过滤器 31
11110 = 过滤器 30
•
•
•
00001 = 过滤器 1
00000 = 过滤器 0

bit 7 **未实现：**读为 0

**注 1：**该寄存器具有关联的清零寄存器（CiVECCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：**该寄存器具有关联的置 1 寄存器（CiVECSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：**该寄存器具有关联的取反寄存器（CiVECINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：**只有对于已允许的中断，才会更新相应的位。

**寄存器 34-4： CiVEC：CAN 中断编码寄存器 (1,2,3) （续）**

bit 6-0 **ICOD<6:0>**：中断标志编码位 (4)

1001000-1111111 = 保留
1001000 = 收到无效报文 （IVRIF）
1000111 = CAN 模块模式改变 （MODIF）
1000110 = CAN 时间标记定时器 （CTMRIF）
1000101 = 总线带宽错误 （SERRIF）
1000100 = 地址错误中断 （SERRIF）
1000011 = 接收 FIFO 溢出中断 （RBOVIF）
1000010 = 唤醒中断 （WAKIF）
1000001 = 错误中断 （CERRIF）
1000000 = 无中断
0100000-0111111 = 保留
0011111 = FIFO31 中断 （CiFSTAT<31> 置 1）
0011110 = FIFO30 中断 （CiFSTAT<30> 置 1）
•
•
•
0000001 = FIFO1 中断 （CiFSTAT<1> 置 1）
0000000 = FIFO0 中断 （CiFSTAT<0> 置 1）

**注 1：** 该寄存器具有关联的清零寄存器（CiVECCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（CiVECSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（CiVECINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 只有对于已允许的中断，才会更新相应的位。

**寄存器 34-5： CiTREC：CAN 发送 / 接收错误计数寄存器 (1,2,3)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 31 | | | | | | | bit 24 |

| U-0 | U-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| — | — | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN |
| bit 23 | | | | | | | bit 16 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| TEC<7:0> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| REC<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **图注：** | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-22 **未实现：**读为 0

bit 21 **TXBO：**发送器处于错误状态总线关闭位 （TEC ≥ 256）

bit 20 **TXBP：**发送器处于错误状态总线被动位 （TEC ≥ 128）

bit 19 **RXBP：**接收器处于错误状态总线被动位 （REC ≥ 128）

bit 18 **TXWARN：**发送器处于错误状态警告位 （128 > TEC ≥ 96）

bit 17 **RXWARN：**接收器处于错误状态警告位 （128 > REC ≥ 96）

bit 16 **EWARN：**发送器或接收器处于错误状态警告位

bit 15-8 **TEC<7:0>：**发送错误计数器位

bit 7-0 **REC<7:0>：**接收错误计数器位

**注 1：** 该寄存器具有关联的清零寄存器（CiTRECCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（CiTRECSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（CiTRECINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**寄存器 34-6： CiFSTAT：CAN FIFO 状态寄存器 (1,2,3)**

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| FIFOIP31 | FIFOIP30 | FIFOIP29 | FIFOIP28 | FIFOIP27 | FIFOIP26 | FIFOIP25 | FIFOIP24 |
| bit 31 | | | | | | | bit 24 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| FIFOIP23 | FIFOIP22 | FIFOIP21 | FIFOIP20 | FIFOIP19 | FIFOIP18 | FIFOIP17 | FIFOIP16 |
| bit 23 | | | | | | | bit 16 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| FIFOIP15 | FIFOIP14 | FIFOIP13 | FIFOIP12 | FIFOIP11 | FIFOIP10 | FIFOIP9 | FIFOIP8 |
| bit 15 | | | | | | | bit 8 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| FIFOIP7 | FIFOIP6 | FIFOIP5 | FIFOIP4 | FIFOIP3 | FIFOIP2 | FIFOIP1 | FIFOIP0 |
| bit 7 | | | | | | | bit 0 |

| **图注：** | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-0 **FIFOIP<31:0>**：FIFOn 中断待处理位
1 = 有一个或多个已允许的 FIFO 中断等待处理
0 = 没有待处理的 FIFO 中断

**注 1**：该寄存器具有关联的清零寄存器（CiFSTATCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFSTATSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器（CiFSTATINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**寄存器 34-7： CiRXOVF：CAN 接收 FIFO 溢出状态寄存器 (1,2,3)**

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| RXOVF31 | RXOVF30 | RXOVF29 | RXOVF28 | RXOVF27 | RXOVF26 | RXOVF25 | RXOVF24 |
| bit 31 | | | | | | | bit 24 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| RXOVF23 | RXOVF22 | RXOVF21 | RXOVF20 | RXOVF19 | RXOVF18 | RXOVF17 | RXOVF16 |
| bit 23 | | | | | | | bit 16 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| RXOVF15 | RXOVF14 | RXOVF13 | RXOVF12 | RXOVF11 | RXOVF10 | RXOVF9 | RXOVF8 |
| bit 15 | | | | | | | bit 8 |

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| RXOVF7 | RXOVF6 | RXOVF5 | RXOVF4 | RXOVF3 | RXOVF2 | RXOVF1 | RXOVF0 |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-0 **RXOVF<31:0>**：FIFOn 接收溢出中断待处理位
1 = FIFO 已溢出
0 = FIFO 未溢出

**注 1**：该寄存器具有关联的清零寄存器（CiRXOVFCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiRXOVFSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器（CiRXOVFINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**寄存器 34-8： CiTMR：CAN 定时器寄存器 (1,2,3,4,5)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CANTS<15:8> | | | | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CANTS<7:0> | | | | | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CANTSPRE<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CANTSPRE<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **图注：** | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-16 **CANTS<15:0>**：CAN 时间标记定时器位
它是一个自由运行定时器，在 CANCAP 位 （CiCON<20>）置 1 时，它会每经过一个 CANTSPRE 系统时钟递增一次。

bit 15-0 **CANTSPRE<15:0>**：CAN 时间标记定时器预分频比位
65535 = CAN 时间标记定时器 （CANTS）每隔 65,535 个系统时钟递增一次
•
•
•
0 = CAN 时间标记定时器 （CANTS）每经过一个系统时钟递增一次

**注 1**：该寄存器具有关联的清零寄存器（CiTMRCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器 （CiTMRSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器（CiTMRINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：当 CANCAP = 0 时， CiTMR 将被冻结。

**5**：每次写入 CiTMR 时， CiTMR 预分频器计数都会复位 （CANTSPRE 不受影响）。

**寄存器 34-9： CiRXMn：CAN 接收过滤器屏蔽器 n 寄存器 （n = 0、 1、 2 或 3）(1,2,3,4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| SID<10:3> | | | | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | U-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| SID<2:0> | | | — | MIDE | — | EID<17:16> | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| EID<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| EID<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **图注：** | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-21 **SID<10:0>**：标准标识符位
1 = 过滤器比较操作包含 SIDx 位
0 = 过滤器比较操作与 SIDx 位无关

bit 20 **未实现**：读为 0

bit 19 **MIDE**：标识符接收模式位
1 = 只匹配与过滤器中 EXID 位对应的报文类型 （标准 / 扩展地址）
0 = 如果过滤器匹配 （即，如果 （过滤器 SID） = （报文 SID）或如果 （过滤器 SID/EID） = （报文 SID/EID）），则与标准或扩展地址报文匹配

bit 18 **未实现**：读为 0

bit 17-0 **EID<17:0>**：扩展标识符位
1 = 过滤器比较操作包含 EIDx 位
0 = 过滤器比较操作与 EIDx 位无关

**注 1**：该寄存器具有关联的清零寄存器（CiRXMnCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiRXMnSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器（CiRXMnINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当 CAN 模块处于配置模式 （OPMOD<2:0> （CiCON<23:21>） = 100）时，才能修改该寄存器。

**寄存器 34-10： CiFLTCON0：CAN 过滤器控制寄存器 0(1,2,3,4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN3 | MSEL3<1:0> | | FSEL3<4:0> | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN2 | MSEL2<1:0> | | FSEL2<4:0> | | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN1 | MSEL1<1:0> | | FSEL1<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN0 | MSEL0<1:0> | | FSEL0<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31 **FLTEN3**：过滤器 3 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 30-29 **MSEL3<1:0>**：过滤器 3 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 28-24 **FSEL3<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
•
•
•
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 23 **FLTEN2**：过滤器 2 使能位
1 = 使能过滤器
0 = 禁止过滤器

**注 1**：该寄存器具有关联的清零寄存器（CiFLTCON0CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFLTCON0SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器 （CiFLTCON0INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当对应的过滤器使能 （FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-10： CiFLTCON0：CAN 过滤器控制寄存器 0(1,2,3,4) （续）**

bit 22-21 **MSEL2<1:0>**：过滤器 2 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 20-16 **FSEL2<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 15 **FLTEN1**：过滤器 1 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 14-13 **MSEL1<1:0>**：过滤器 1 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 12-8 **FSEL1<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 7 **FLTEN0**：过滤器 0 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 6-5 **MSEL0<1:0>**：过滤器 0 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 4-0 **FSEL0<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

**注 1：** 该寄存器具有关联的清零寄存器（CiFLTCON0CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（CiFLTCON0SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（CiFLTCON0INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 仅当对应的过滤器使能（FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-11： CiFLTCON1：CAN 过滤器控制寄存器 1(1,2,3,4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN7 | MSEL7<1:0> | | FSEL7<4:0> | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN6 | MSEL6<1:0> | | FSEL6<4:0> | | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN5 | MSEL5<1:0> | | FSEL5<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN4 | MSEL4<1:0> | | FSEL4<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31 **FLTEN7**：过滤器 7 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 30-29 **MSEL7<1:0>**：过滤器 7 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 28-24 **FSEL7<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
•
•
•
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 23 **FLTEN6**：过滤器 6 使能位
1 = 使能过滤器
0 = 禁止过滤器

**注 1**：该寄存器具有关联的清零寄存器（CiFLTCON1CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFLTCON1SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器 （CiFLTCON1INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当对应的过滤器使能 （FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-11： CiFLTCON1：CAN 过滤器控制寄存器 1(1,2,3,4) （续）**

bit 22-21 **MSEL6<1:0>：**过滤器 6 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 20-16 **FSEL6<4:0>：**FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 15 **FLTEN5：**过滤器 5 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 14-13 **MSEL5<1:0>：**过滤器 5 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 12-8 **FSEL5<4:0>：**FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 7 **FLTEN4：**过滤器 4 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 6-5 **MSEL4<1:0>：**过滤器 4 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 4-0 **FSEL4<4:0>：**FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

**注 1：**该寄存器具有关联的清零寄存器（CiFLTCON1CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：**该寄存器具有关联的置 1 寄存器（CiFLTCON1SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：**该寄存器具有关联的取反寄存器 （CiFLTCON1INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：**仅当对应的过滤器使能 （FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-12： CiFLTCON2：CAN 过滤器控制寄存器 2(1,2,3,4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN11 | MSEL11<1:0> | | FSEL11<4:0> | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN10 | MSEL10<1:0> | | FSEL10<4:0> | | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN9 | MSEL9<1:0> | | FSEL9<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN8 | MSEL8<1:0> | | FSEL8<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31 **FLTEN11**：过滤器 11 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 30-29 **MSEL11<1:0>**：过滤器 11 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 28-24 **FSEL11<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
•
•
•
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 23 **FLTEN10**：过滤器 10 使能位
1 = 使能过滤器
0 = 禁止过滤器

**注 1**：该寄存器具有关联的清零寄存器（CiFLTCON2CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFLTCON2SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器 （CiFLTCON2INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当对应的过滤器使能 （FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-12： CiFLTCON2：CAN 过滤器控制寄存器 2(1,2,3,4) （续）**

bit 22-21 **MSEL10<1:0>**：过滤器 10 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 20-16 **FSEL10<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 15 **FLTEN9**：过滤器 9 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 14-13 **MSEL9<1:0>**：过滤器 9 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 12-8 **FSEL9<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 7 **FLTEN8**：过滤器 8 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 6-5 **MSEL8<1:0>**：过滤器 8 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 4-0 **FSEL8<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

**注 1**：该寄存器具有关联的清零寄存器（CiFLTCON2CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFLTCON2SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器（CiFLTCON2INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当对应的过滤器使能（FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-13： CiFLTCON3：CAN 过滤器控制寄存器 3(1,2,3,4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN15 | MSEL15<1:0> | | FSEL15<4:0> | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN14 | MSEL14<1:0> | | FSEL14<4:0> | | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN13 | MSEL13<1:0> | | FSEL13<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN12 | MSEL12<1:0> | | FSEL12<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31 **FLTEN15**：过滤器 15 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 30-29 **MSEL15<1:0>**：过滤器 15 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 28-24 **FSEL15<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
•
•
•
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 23 **FLTEN14**：过滤器 14 使能位
1 = 使能过滤器
0 = 禁止过滤器

**注 1**：该寄存器具有关联的清零寄存器（CiFLTCON3CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFLTCON3SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器 （CiFLTCON3INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当对应的过滤器使能 （FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-13： CiFLTCON3：CAN 过滤器控制寄存器 3(1,2,3,4) （续）**

| | |
|---|---|
| bit 22-21 | **MSEL14<1:0>**：过滤器 14 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 20-16 | **FSEL14<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |
| bit 15 | **FLTEN13**：过滤器 13 使能位<br>1 = 使能过滤器<br>0 = 禁止过滤器 |
| bit 14-13 | **MSEL13<1:0>**：过滤器 13 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 12-8 | **FSEL13<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |
| bit 7 | **FLTEN12**：过滤器 12 使能位<br>1 = 使能过滤器<br>0 = 禁止过滤器 |
| bit 6-5 | **MSEL12<1:0>**：过滤器 12 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 4-0 | **FSEL12<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |

**注 1：** 该寄存器具有关联的清零寄存器（CiFLTCON3CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（CiFLTCON3SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（CiFLTCON3INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 仅当对应的过滤器使能（FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-14： CiFLTCON4：CAN 过滤器控制寄存器 4(1,2,3,4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN19 | MSEL19<1:0> | | FSEL19<4:0> | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN18 | MSEL18<1:0> | | FSEL18<4:0> | | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN17 | MSEL17<1:0> | | FSEL17<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN16 | MSEL16<1:0> | | FSEL16<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31 **FLTEN19**：过滤器 19 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 30-29 **MSEL19<1:0>**：过滤器 19 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 28-24 **FSEL19<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
•
•
•
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 23 **FLTEN18**：过滤器 18 使能位
1 = 使能过滤器
0 = 禁止过滤器

**注 1**：该寄存器具有关联的清零寄存器（CiFLTCON4CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFLTCON4SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器 （CiFLTCON4INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当对应的过滤器使能 （FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-14： CiFLTCON4：CAN 过滤器控制寄存器 4(1,2,3,4) （续）**

| | |
|---|---|
| bit 22-21 | **MSEL18<1:0>**：过滤器 18 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 20-16 | **FSEL18<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |
| bit 15 | **FLTEN17**：过滤器 17 使能位<br>1 = 使能过滤器<br>0 = 禁止过滤器 |
| bit 14-13 | **MSEL17<1:0>**：过滤器 17 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 12-8 | **FSEL17<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |
| bit 7 | **FLTEN16**：过滤器 16 使能位<br>1 = 使能过滤器<br>0 = 禁止过滤器 |
| bit 6-5 | **MSEL16<1:0>**：过滤器 16 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 4-0 | **FSEL16<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |

**注 1：** 该寄存器具有关联的清零寄存器（CiFLTCON4CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（CiFLTCON4SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（CiFLTCON4INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 仅当对应的过滤器使能（FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-15： CiFLTCON5：CAN 过滤器控制寄存器 5(1,2,3,4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN23 | MSEL23<1:0> | | FSEL23<4:0> | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN22 | MSEL22<1:0> | | FSEL22<4:0> | | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN21 | MSEL21<1:0> | | FSEL21<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN20 | MSEL20<1:0> | | FSEL20<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31 **FLTEN23**：过滤器 23 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 30-29 **MSEL23<1:0>**：过滤器 23 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 28-24 **FSEL23<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
•
•
•
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 23 **FLTEN22**：过滤器 22 使能位
1 = 使能过滤器
0 = 禁止过滤器

**注 1**：该寄存器具有关联的清零寄存器（CiFLTCON5CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFLTCON5SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器 （CiFLTCON5INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当对应的过滤器使能 （FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-15： CiFLTCON5：CAN 过滤器控制寄存器 5(1,2,3,4) （续）**

| | |
|---|---|
| bit 22-21 | **MSEL22<1:0>**：过滤器 22 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 20-16 | **FSEL22<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |
| bit 15 | **FLTEN21**：过滤器 21 使能位<br>1 = 使能过滤器<br>0 = 禁止过滤器 |
| bit 14-13 | **MSEL21<1:0>**：过滤器 21 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 12-8 | **FSEL21<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |
| bit 7 | **FLTEN20**：过滤器 20 使能位<br>1 = 使能过滤器<br>0 = 禁止过滤器 |
| bit 6-5 | **MSEL20<1:0>**：过滤器 20 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 4-0 | **FSEL20<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |

**注 1：** 该寄存器具有关联的清零寄存器（CiFLTCON5CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（CiFLTCON5SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（CiFLTCON5INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 仅当对应的过滤器使能（FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-16： CiFLTCON6：CAN 过滤器控制寄存器 6(1,2,3,4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN27 | MSEL27<1:0> | | FSEL27<4:0> | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN26 | MSEL26<1:0> | | FSEL26<4:0> | | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN25 | MSEL25<1:0> | | FSEL25<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN24 | MSEL24<1:0> | | FSEL24<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31 **FLTEN27**：过滤器 27 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 30-29 **MSEL27<1:0>**：过滤器 27 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 28-24 **FSEL27<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
•
•
•
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 23 **FLTEN26**：过滤器 26 使能位
1 = 使能过滤器
0 = 禁止过滤器

**注 1**：该寄存器具有关联的清零寄存器（CiFLTCON6CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFLTCON6SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器 （CiFLTCON6INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当对应的过滤器使能 （FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-16： CiFLTCON6：CAN 过滤器控制寄存器 6(1,2,3,4) （续）**

| | |
|---|---|
| bit 22-21 | **MSEL26<1:0>**：过滤器 26 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 20-16 | **FSEL26<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |
| bit 15 | **FLTEN25**：过滤器 25 使能位<br>1 = 使能过滤器<br>0 = 禁止过滤器 |
| bit 14-13 | **MSEL25<1:0>**：过滤器 25 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 12-8 | **FSEL25<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |
| bit 7 | **FLTEN24**：过滤器 24 使能位<br>1 = 使能过滤器<br>0 = 禁止过滤器 |
| bit 6-5 | **MSEL24<1:0>**：过滤器 24 屏蔽器选择位<br>11 = 选择接收屏蔽器 3<br>10 = 选择接收屏蔽器 2<br>01 = 选择接收屏蔽器 1<br>00 = 选择接收屏蔽器 0 |
| bit 4-0 | **FSEL24<4:0>**：FIFO 选择位<br>11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中<br>11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中<br>.<br>.<br>.<br>00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中<br>00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中 |

**注 1：** 该寄存器具有关联的清零寄存器（CiFLTCON6CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（CiFLTCON6SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（CiFLTCON6INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 仅当对应的过滤器使能（FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-17： CiFLTCON7：CAN 过滤器控制寄存器 7(1,2,3,4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN31 | MSEL31<1:0> | | FSEL31<4:0> | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN30 | MSEL30<1:0> | | FSEL30<4:0> | | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN29 | MSEL29<1:0> | | FSEL29<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FLTEN28 | MSEL28<1:0> | | FSEL28<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31 **FLTEN31**：过滤器 31 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 30-29 **MSEL31<1:0>**：过滤器 31 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 28-24 **FSEL31<4:0>**：FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
•
•
•
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 23 **FLTEN30**：过滤器 30 使能位
1 = 使能过滤器
0 = 禁止过滤器

**注 1**：该寄存器具有关联的清零寄存器（CiFLTCON7CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFLTCON7SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器 （CiFLTCON7INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：仅当对应的过滤器使能 （FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-17： CiFLTCON7：CAN 过滤器控制寄存器 7(1,2,3,4) （续）**

bit 22-21 **MSEL30<1:0>：**过滤器 30 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 20-16 **FSEL30<4:0>：**FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 15 **FLTEN29：**过滤器 29 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 14-13 **MSEL29<1:0>：**过滤器 29 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 12-8 **FSEL29<4:0>：**FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

bit 7 **FLTEN28：**过滤器 28 使能位
1 = 使能过滤器
0 = 禁止过滤器

bit 6-5 **MSEL28<1:0>：**过滤器 28 屏蔽器选择位
11 = 选择接收屏蔽器 3
10 = 选择接收屏蔽器 2
01 = 选择接收屏蔽器 1
00 = 选择接收屏蔽器 0

bit 4-0 **FSEL28<4:0>：**FIFO 选择位
11111 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 31 中
11110 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 30 中
.
.
.
00001 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 1 中
00000 = 与过滤器匹配的报文被存储在 FIFO 缓冲区 0 中

**注 1：** 该寄存器具有关联的清零寄存器（CiFLTCON7CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器（CiFLTCON7SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（CiFLTCON7INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 仅当对应的过滤器使能（FLTENn）位为 0 时，才能修改该寄存器中的位。

**寄存器 34-18： CiRXFn：CAN 接收过滤器 n 寄存器 （n = 0 至 31） (1,2,3,4)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| SID<10:3> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-x | R/W-x | R/W-x | U-0 | R/W-0 | U-0 | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| SID<2:0> | | | — | EXID | — | EID<17:16> | |
| bit 23 | | | | | | | bit 16 |

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| EID<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| EID<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **图注：** | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-21 **SID<10:0>：**标准标识符位
1 = 报文地址位 SIDx 必须为 1 才能与过滤器匹配
0 = 报文地址位 SIDx 必须为 0 才能与过滤器匹配

bit 20 **未实现：**读为 0

bit 19 **EXID：**扩展标识符使能位
1 = 只与具有扩展标识符地址的报文匹配
0 = 只与具有标准标识符地址的报文匹配

bit 18 **未实现：**读为 0

bit 17-0 **EID<17:0>：**扩展标识符位
1 = 报文地址位 EIDx 必须为 1 才能与过滤器匹配
0 = 报文地址位 EIDx 必须为 0 才能与过滤器匹配

**注 1：**该寄存器具有关联的清零寄存器 （CiRXFnCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：**该寄存器具有关联的置 1 寄存器 （CiRXFnSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：**该寄存器具有关联的取反寄存器 （CiRXFnINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：**仅当过滤器被禁止 （FLTENn = 0）时，才能修改该寄存器。

**寄存器 34-19： CiFIFOBA：CAN 报文缓冲区基址寄存器 (1,2,3,4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CiFIFOBA<31:24> | | | | | | | |
| bit 31 | | | | | | | bit 24 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CiFIFOBA<23:16> | | | | | | | |
| bit 23 | | | | | | | bit 16 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CiFIFOBA<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0(5) | R-0(5) |
|---|---|---|---|---|---|---|---|
| CiFIFOBA<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-0 **CiFIFOBA<31:0>**：CAN FIFO 基址位

定义所有报文缓冲区的基址。每个报文缓冲区的位置取决于前一个报文缓冲区的大小。该地址是物理地址。请注意， bit <1:0> 是只读位且读为 0，这会使报文在系统 RAM 中按 32 位字对齐。

**注 1：** 该寄存器具有关联的清零寄存器（CiFIFOBACLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器 （CiFIFOBASET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器（CiFIFOBAINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 仅当 CAN 模块处于配置模式 （OPMOD<2:0> （CiCON<23:21>） = 100）时，才能修改该寄存器。

**5：** 该位未实现，总是读为 0，这会使报文按字对齐。

**寄存器 34-20： CiFIFOCONn：CAN FIFO 控制寄存器 n （n = 0 至 31）(1,2,3)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 31 | | | | | | | bit 24 |

| U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | FSIZE<4:0>(4) | | | | |
| bit 23 | | | | | | | bit 16 |

| U-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | FRESET | UINC | DONLY(4) | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| TXEN | TXABAT(5) | TXLARB(6) | TXERR(6) | TXREQ | RTREN | TXPR<1:0> | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-21 **未实现：**读为 0

bit 20-16 **FSIZE<4:0>：**FIFO 大小位 (4)

11111 = FIFO 为 32 个报文深
•
•
•
00010 = FIFO 为 3 个报文深
00001 = FIFO 为 2 个报文深
00000 = FIFO 为 1 个报文深

bit 15 **未实现：**读为 0

bit 14 **FRESET：**FIFO 复位位

1 = 该位置 1 时 FIFO 会复位， FIFO 复位之后硬件会将该位清零。在执行任何操作之前，用户应先查询该位是否清零
0 = 无影响

**注 1：** 该寄存器具有关联的清零寄存器 （CiFIFOCONnCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器 （CiFIFOCONnSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器 （CiFIFOCONnINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 仅当 CAN 模块处于配置模式 （OPMOD<2:0> （CiCON<23:21>） = 100）时，才能修改这些位。

**5：** 该位在报文完成 （或中止）时或 FIFO 复位时更新。

**6：** 该位在每次读取该寄存器时或 FIFO 复位时复位。

**寄存器 34-20： CiFIFOCONn：CAN FIFO 控制寄存器 n （n = 0 至 31）(1,2,3) （续）**

bit 13 **UINC**：递增头部 / 尾部位

TXEN = 1：（FIFO 被配置为发送 FIFO）
当该位置 1 时， FIFO 头部会按单个报文进行递增

TXEN = 0：（FIFO 被配置为接收 FIFO）
当该位置 1 时， FIFO 尾部会按单个报文进行递增

bit 12 **DONLY**：仅存储报文数据位 (1)

TXEN = 1：（FIFO 被配置为发送 FIFO）
不使用该位，该位不起任何作用。

TXEN = 0：（FIFO 被配置为接收 FIFO）
1 = 仅在 FIFO 中存储数据字节
0 = 存储完整报文，包括标识符

bit 11-8 **未实现**：读为 0

bit 7 **TXEN**：发送 / 接收缓冲区选择位
1 = FIFO 为发送 FIFO
0 = FIFO 为接收 FIFO

bit 6 **TXABAT**：报文中止位 (5)
1 = 中止报文
0 = 成功完成报文发送

bit 5 **TXLARB**：报文仲裁失败位 (6)
1 = 报文在发送过程中仲裁失败
0 = 报文在发送过程中没有仲裁失败

bit 4 **TXERR**：在发送过程中检测到错误位 (6)
1 = 报文发送时发生总线错误
0 = 报文发送时未发生总线错误

bit 3 **TXREQ**：报文发送请求位

TXEN = 1：（FIFO 被配置为发送 FIFO）
将该位设置为 1 请求发送报文。
在成功发送 FIFO 中排队的所有报文之后，该位会自动清零。
在该位置 1 的情况下清零该位，将请求中止报文。

TXEN = 0：（FIFO 被配置为接收 FIFO）
该位不起任何作用。

bit 2 **RTREN**：自动 RTR 使能位
1 = 当接收到远程发送时，将 TXREQ 置 1
0 = 当接收到远程发送时， TXREQ 不受影响

bit 1-0 **TXPR<1:0>**：报文发送优先级位
11 = 最高报文优先级
10 = 中高报文优先级
01 = 中低报文优先级
00 = 最低报文优先级

**注 1：** 该寄存器具有关联的清零寄存器 （CiFIFOCONnCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器 （CiFIFOCONnSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器 （CiFIFOCONnINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 仅当 CAN 模块处于配置模式 （OPMOD<2:0> （CiCON<23:21>） = 100）时，才能修改这些位。

**5：** 该位在报文完成 （或中止）时或 FIFO 复位时更新。

**6：** 该位在每次读取该寄存器时或 FIFO 复位时复位。

**寄存器 34-21： CiFIFOINTn：CAN FIFO 中断寄存器 n （n = 0 至 31）(1,2,3)**

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | TXNFULLIE | TXHALFIE | TXEMPTYIE |
| bit 31 | | | | | | | bit 24 |

| U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | RXOVFLIE | RXFULLIE | RXHALFIE | RXNEMPTYIE |
| bit 23 | | | | | | | bit 16 |

| U-0 | U-0 | U-0 | U-0 | U-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | TXNFULLIF(4) | TXHALFIF | TXEMPTYIF(4) |
| bit 15 | | | | | | | bit 8 |

| U-0 | U-0 | U-0 | U-0 | R/W-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | RXOVFLIF | RXFULLIF(4) | RXHALFIF(4) | RXNEMPTYIF(4) |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-27 **未实现：** 读为 0

bit 26 **TXNFULLIE：** 发送 FIFO 未满中断允许位
1 = 允许 FIFO 未满中断
0 = 禁止 FIFO 未满中断

bit 25 **TXHALFIE：** 发送 FIFO 半满中断允许位
1 = 允许 FIFO 半满中断
0 = 禁止 FIFO 半满中断

bit 24 **TXEMPTYIE：** 发送 FIFO 为空中断允许位
1 = 允许 FIFO 为空中断
0 = 禁止 FIFO 为空中断

bit 23-20 **未实现：** 读为 0

bit 19 **RXOVFLIE：** 接收 FIFO 溢出中断允许位
1 = 允许 FIFO 溢出事件中断
0 = 禁止 FIFO 溢出事件中断

bit 18 **RXFULLIE：** 接收 FIFO 已满中断允许位
1 = 允许 FIFO 已满中断
0 = 禁止 FIFO 已满中断

**注 1：** 该寄存器具有关联的清零寄存器 （CiFIFOINTnCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器 （CiFIFOINTnSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器 （CiFIFOINTnINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 该位是只读位，用于反映 FIFO 的状态。

**寄存器 34-21： CiFIFOINTn：CAN FIFO 中断寄存器 n （n = 0 至 31）(1,2,3) （续）**

bit 17 **RXHALFIE**：接收 FIFO 半满中断允许位
1 = 允许 FIFO 半满中断
0 = 禁止 FIFO 半满中断

bit 16 **RXNEMPTYIE**：接收 FIFO 非空中断允许位
1 = 允许 FIFO 非空中断
0 = 禁止 FIFO 非空中断

bit 15-11 **未实现**：读为 0

bit 10 **TXNFULLIF**：发送 FIFO 未满中断标志位 (4)
TXEN = 1：（FIFO 被配置为发送缓冲区）
1 = FIFO 未满
0 = FIFO 已满
TXEN = 0：（FIFO 被配置为接收缓冲区）
未使用，读为 0

bit 9 **TXHALFIF**：发送 FIFO 半空中断标志位 (4)
TXEN = 1：（FIFO 被配置为发送缓冲区）
1 = FIFO ≤ 半满
0 = FIFO > 半满
TXEN = 0：（FIFO 被配置为接收缓冲区）
未使用，读为 0

bit 8 **TXEMPTYIF**：发送 FIFO 为空中断标志位 (4)
TXEN = 1：（FIFO 被配置为发送缓冲区）
1 = FIFO 为空
0 = FIFO 非空，至少有 1 个报文在排队等待发送
TXEN = 0：（FIFO 被配置为接收缓冲区）
未使用，读为 0

bit 7-4 **未实现**：读为 0

bit 3 **RXOVFLIF**：接收 FIFO 溢出中断标志位
TXEN = 1：（FIFO 被配置为发送缓冲区）
未使用，读为 0
TXEN = 0：（FIFO 被配置为接收缓冲区）
1 = 发生了溢出事件
0 = 未发生溢出事件

bit 2 **RXFULLIF**：接收 FIFO 已满中断标志位 (4)
TXEN = 1：（FIFO 被配置为发送缓冲区）
未使用，读为 0
TXEN = 0：（FIFO 被配置为接收缓冲区）
1 = FIFO 已满
0 = FIFO 未满

**注 1**：该寄存器具有关联的清零寄存器（CiFIFOINTnCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器（CiFIFOINTnSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器（CiFIFOINTnINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：该位是只读位，用于反映 FIFO 的状态。

**寄存器 34-21： CiFIFOINTn：CAN FIFO 中断寄存器 n （n = 0 至 31）(1,2,3) （续）**

bit 1 **RXHALFIF**：接收 FIFO 半满中断标志位 (4)

TXEN = 1：（FIFO 被配置为发送缓冲区）
未使用，读为 0

TXEN = 0：（FIFO 被配置为接收缓冲区）
1 = FIFO ≥ 半满
0 = FIFO < 半满

bit 0 **RXNEMPTYIF**：接收缓冲区非空中断标志位 (4)

TXEN = 1：（FIFO 被配置为发送缓冲区）
未使用，读为 0

TXEN = 0：（FIFO 被配置为接收缓冲区）
1 = FIFO 非空，至少有 1 个报文
0 = FIFO 为空

**注 1**：该寄存器具有关联的清零寄存器 （CiFIFOINTnCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2**：该寄存器具有关联的置 1 寄存器 （CiFIFOINTnSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3**：该寄存器具有关联的取反寄存器 （CiFIFOINTnINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4**：该位是只读位，用于反映 FIFO 的状态。

**寄存器 34-22： CiFIFOUAn：CAN FIFO 用户地址寄存器 n （n = 0 至 31）(1,2,3,4)**

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| CiFIFOUAn<31:24> | | | | | | | |
| bit 31 | | | | | | | bit 24 |

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| CiFIFOUAn<23:16> | | | | | | | |
| bit 23 | | | | | | | bit 16 |

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| CiFIFOUAn<15:8> | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R-x | R-x | R-x | R-x | R-x | R-x | R-0(5) | R-0(5) |
|---|---|---|---|---|---|---|---|
| CiFIFOUAn<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-0 **CiFIFOUAn<31:0>**：CAN FIFO 用户地址位

TXEN = 1：（FIFO 被配置为发送缓冲区）

读取该寄存器将返回用于写入下一个报文的地址 （FIFO 头部）。

TXEN = 0：（FIFO 被配置为接收缓冲区）

读取该寄存器将返回用于读取下一个报文的地址 （FIFO 尾部）。

**注 1：** 该寄存器具有关联的清零寄存器 （CiFIFOUAnCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：** 该寄存器具有关联的置 1 寄存器 （CiFIFOUAnSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：** 该寄存器具有关联的取反寄存器 （CiFIFOUAnINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

**4：** 在配置模式下，无法保证可以正确读取该寄存器，应当在模块不处于配置模式时访问它。

**5：** 该位总是读为 0，这会使报文按字节对齐。

**寄存器 34-23： CiFIFOCIn：CAN 模块报文索引寄存器 n （n = 0 至 31）(1,2,3)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 31 | | | | | | | bit 24 |

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 23 | | | | | | | bit 16 |

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| U-0 | U-0 | U-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | CiFIFOCI<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| **图注：** | | | |
|---|---|---|---|
| R = 可读位 | W = 可写位 | U = 未实现位，读为 0 | |
| -n = POR 时的值 | 1 = 置 1 | 0 = 清零 | x = 未知 |

bit 31-5 **未实现：**读为 0

bit 4-0 **CiFIFOCIn<4:0>：**CAN 端 FIFO 报文索引位

TXEN = 1：（FIFO 被配置为发送缓冲区）
读取该寄存器将返回一个索引，该索引指向 FIFO 下一次尝试发送的报文。

TXEN = 0：（FIFO 被配置为接收缓冲区）
读取该寄存器将返回一个索引，该索引指向 FIFO 用于保存下一个报文的报文位置。

**注 1：**该寄存器具有关联的清零寄存器 （CiFIFOCInCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

**2：**该寄存器具有关联的置 1 寄存器 （CiFIFOCInSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

**3：**该寄存器具有关联的取反寄存器 （CiFIFOCInINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

## 34.4 使能和禁止 CAN 模块

当 CAN 模块关闭时，整个 CAN 模块会保持在复位状态，用户只能访问 CAN 模块的特殊功能寄存器（SFR）。可以通过将 CAN 控制寄存器中的 ON（CiCON<15>）位置 1 来使能 CAN 模块。当 ON 位置 1 时，模块处于活动状态， CAN 模块对于 CAN 发送 （CiTX）和接收 （CiRX）引脚具有优先使用权。

关闭模块会将模块置为复位状态，并释放对 CiTX 和 CiRX 引脚的器件控制。当 CAN 模块关闭时，所有报文 FIFO 都会复位。

CAN 模块完全关闭可能需要一定的时钟周期。 CANBUSY 位 （CiCON<11>）可以指示模块的状态。用户应通过查询 CANBUSY 位来确保模块已关闭。此外，在关闭模块之前，需要确保模块处于配置模式，这一点很重要（见**第 34.5 节 “CAN 模块工作模式”**）。这可以防止由于在 CAN 模块发送报文过程中关闭模块而导致总线错误。

例 34-1 给出了关闭 CAN1 模块的必需步骤。

**例 34-1： 禁止 CAN1 模块**

```
/* Place the CAN module in configuration mode. */

C1CONbits.REQOP = 4;
while(C1CONbits.OPMOD != 4);

/* Switch the CAN module off. */
C1CONCLR = 0x00008000;        /*Clear the ON bit */
while(C1CONbits.CANBUSY == 1);
```

## 34.5 CAN 模块工作模式

用户应用程序可以选择 CAN 模块工作在以下几种模式之一。这些模式包括：

- 配置模式
- 正常工作模式
- 监听模式
- 监听所有报文模式
- 环回模式
- 禁止模式

应用程序通过写入 CAN 控制寄存器中的请求工作模式位 REQOP<2:0> （CiCON<26:24>）来请求所需的工作模式。 CAN 模块通过工作模式位 OPMOD<2:0> （CiCON<23:21>）确认进入所请求的模式。模式转换与 CAN 网络同步执行。

通过允许 CAN 中断寄存器中的模式改变中断位 MODIE （CiINT<19>），应用程序可以选择在所请求的模式改变发生时产生中断。在成功应用新模式之后，将会产生 CAN 中断。或者，用户也可以选择通过查询 OPMOD<2:0> 位来确定 CAN 模块是否成功切换了模式 （当前工作模式与所请求的工作模式匹配）。

### 34.5.1 配置模式

在器件复位之后， CAN 模块处于配置模式（OPMOD<2:0>（CiCON<23:21>）= 100）。错误计数器被清零，所有寄存器均包含复位值。CAN 模块只能在配置模式下进行配置或初始化。请求进入配置模式的方法是将 REQOP<2:0> 位设置为 100。应用程序应等到 CAN 模块实际处于配置模式之后才能执行。这通过查询 OPMOD<2:0> 位的值是否为 100 来实现。以下寄存器和位只能在配置模式下进行修改：

- CAN 配置 （CiCFG）寄存器
- CAN FIFO 基址 （CiFIFOBA）寄存器
- CAN 接收过滤器屏蔽器 （CiRXMn）寄存器
- CAN FIFO 控制 （CiFIFOCONn）寄存器中的 FIFO 大小 （FSIZE）位和 DONLY 位

这可以防止用户因为编程错误而意外违反 CAN 协议。

### 34.5.2 正常工作模式

在正常工作模式下， CAN 模块会出现在 CAN 总线上，可以发送和接收 CAN 报文。正常工作模式在初始化之后请求，方法是将 REQOP<2:0> 位设置为 000。当 OPMOD<2:0> = 000 时，模块会开始正常工作。

### 34.5.3 监听模式

监听模式是正常工作模式的一种变化形式。如果监听模式被激活，则模块会出现在 CAN 总线上，但处于被动状态。它会接收报文，但不会发送报文。CAN 模块不会产生错误标志，也不会应答信号。在该状态下，错误计数器不再工作。监听模式可用来检测 CAN 总线上的波特率。要使用此功能，必须至少有另外两个互相通信的节点。波特率可以通过测试一些不同值以实证方式检测。该模式也可用作总线监视器，因为 CAN 总线不会影响数据通信。

### 34.5.4 监听所有报文模式

监听所有报文模式是正常工作模式的一种变化形式。如果监听所有报文模式被激活，则发送和接收操作与正常工作模式下相同，只是在接收到带有错误的报文时，仍然将它传输到接收缓冲区中，好像没有错误时一样。接收缓冲区中将包含在出现错误时已接收的所有数据。在该模式下仍然需要使能和配置过滤器。

### 34.5.5 环回模式

环回模式用于进行自检，让 CAN 模块接收它自己的报文。在该模式下， CAN 模块发送路径在内部与接收路径相连接。该模式下会提供“假”应答，从而不需要另一个节点来提供应答位。CAN 报文不会实际发送到 CAN 总线上。

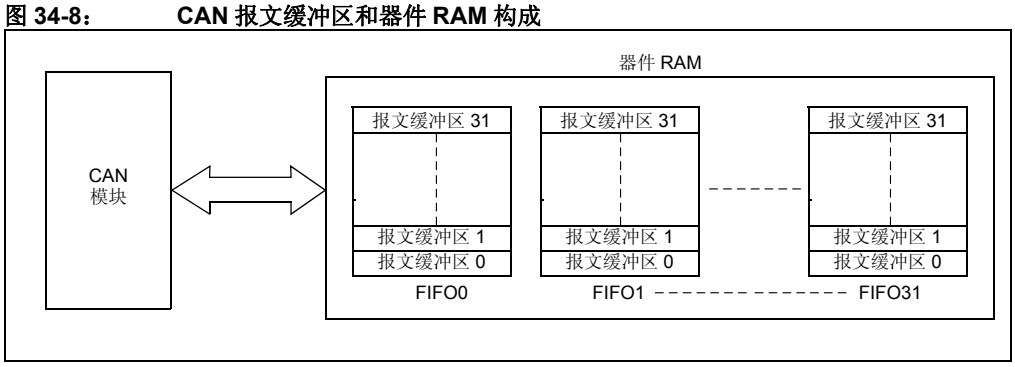
### 34.5.6 禁止模式

禁止模式类似于配置模式，只是 CAN 模块错误计数器不会复位。通过将 REQOP<2:0> 位设置为 001，可以将模块置为禁止模式。在禁止模式下，除非模块在接收或发送报文，否则 CAN 模块的内部时钟会停止。在进行发送或接收时，将不允许 CAN 模块进入禁止模式，以防止模块在系统总线上导致错误。模块会在当前报文完成时进入禁止模式。OPMOD<2:0>（CiCON<23:21>）位指示模块是否已成功进入禁止模式。在模块处于禁止模式时， CAN 模块发送 （CiTX）引脚将保持在隐性状态，以防止意外的 CAN 总线错误。

## 34.6 CAN 报文处理

CAN 模块使用器件 RAM 来存储需要发送或接收的 CAN 报文。模块自身没有用户可访问的报文缓冲区。图 34-8 显示了 CAN 模块缓冲区存储器在器件 RAM 中的构成。

**图 34-8： CAN 报文缓冲区和器件 RAM 构成**




如图 34-8 中所示，CAN 模块采用 FIFO 形式来组织报文缓冲区。共有 32 个不同 FIFO 可供使用，它们具有以下特性：

- 最小为一个 CAN 报文缓冲区，最大为 32 个 CAN 报文缓冲区
- 大小可独立配置
- 可配置为发送报文或接收报文 FIFO
- 用户可读的头部和尾部指针
- 可独立配置中断
- 在发送或接收报文时，状态位可提供 FIFO 的状态
- 可以设置为发送或接收 FIFO，但不能同时设置为两者 （因此，如果将某个 FIFO 配置为用于发送操作，则认为该 FIFO 中的所有报文都用于发送）
- 可以相互独立地配置为发送或接收 FIFO，并且可以使用任意顺序配置

32 个报文 FIFO 中的每一个都具有以下关联寄存器：

- CAN FIFO 控制 （CiFIFOCONn）寄存器 （寄存器 34-20）
- CAN FIFO 中断 （CiFIFOINTn）寄存器 （寄存器 34-21）
- CAN FIFO 用户地址 （CiFIFOUAn）寄存器 （寄存器 34-22）
- CAN FIFO 报文索引 （CiFIFOCIn）寄存器 （寄存器 34-23）

CAN 模块只需要知道 FIFO 中第一个报文缓冲区的起始地址。 CAN FIFO 基址 （CiFIFOBA）寄存器应指向该报文缓冲区的起始地址。CAN 模块会根据各个 FIFO 的配置自动计算每个 FIFO 中报文缓冲区的地址。每个 FIFO 连续排列，从而将所有报文缓冲区聚集在一起。这可以产生不带任何空隙的 CAN 报文缓冲区存储器。 CiFIFOUAn 寄存器提供应用程序必须使用的下一个读 / 写 CAN 报文缓冲区的地址，而 CiFIFOCIn 寄存器则提供要由 CAN 模块发送或写入的下一个报文缓冲区对应的 CAN 报文缓冲区索引。

**第 34.12 节 “CAN 中断”**中讨论了与 FIFO 相关的中断，以及其他一些 CAN 模块中断。

### 34.6.1 报文 FIFO 配置

用户应用程序必须分配用于 CAN 报文缓冲的器件 RAM 空间。每种报文缓冲区的要求如下：

- CAN 发送报文缓冲区需要 4 个字 （16 字节）的存储器
- 如果要存储整个报文 （时间标记加上数据再加上报文 ID，即完整接收报文），则 CAN 接收报文缓冲区需要 4 个字 （16 字节）的存储器
- 如果仅存储数据 （仅数据接收报文），则 CAN 接收报文缓冲区需要 2 个字 （8 字节）

应用程序必须设计每个报文 FIFO 的大小。FIFO 大小通过相应 CAN FIFO 控制寄存器中的 FIFO 大小位 FSIZE<4:0>（CiFIFOCONn<20:16>）进行控制。所有 FIFO 的默认大小为至少一个报文缓冲区。然后，通过计算 FIFO 中的缓冲区总数，以及每个缓冲区的存储器大小，可以得到要分配的存储器总量。

为了以最优方式使用 CAN FIFO 报文缓冲模型，用户应用程序应从 FIFO0 开始配置所需的 FIFO 数量，然后配置 FIFO1、 FIFO2 和 FIFO3，按此顺序类推。

例如，某个应用程序需要 29 个 FIFO，并配置了 FIFO0、FIFO1 和 FIFO5 至 FIFO31。FIFO2、FIFO3 和 FIFO4 未配置。FIFO2、FIFO3 和 FIFO4 虽然未配置，但仍然会各占用 4 字的 RAM。CAN 模块不会使用这些 FIFO，但在计算 FIFO5 的地址时会考虑它们的存在。在该示例中，用户应用程序必须为全部 32 个 FIFO 分配存储器。应用程序可以使用由 FIFO2、FIFO3 和 FIFO4 占用的存储器。

实现该目的的最优方式是配置 FIFO0 至 FIFO28，而不配置 FIFO29、FIFO30 和 FIFO31。使用这种方式时，用户应用程序只需要为 29 个 FIFO 分配空间。图 34-9 给出了一个示例。

**图 34-9： CAN 报文缓冲区 FIFO 配置示例 （包括仅数据接收报文缓冲区）**

| | FIFO 编号 | 报文缓冲区起始地址 | 报文缓冲区 |
|---|---|---|---|
| A | FIFO（发送 FIFO） | 0x00001000 | MB0 |
| | | 0x00001010 | MB1 |
| | | 0x00001020 | MB2 |
| | | 0x00001030 | MB3 |
| B | FIFO1（仅数据接收报文） | 0x00001040 | MB0 |
| | | 0x00001048 | MB1 |
| C | FIFO3（完整接收报文） | 0x00001050 | MB0 |
| | | 0x00001060 | MB1 |
| | | 0x00001070 | MB2 |
| | | 0x00001080 | MB3 |

**CAN1FIFO 配置**
C1FIFOBA = 0x00001000
（FIFO0 中 MB0 的起始地址）

A
C1FIFOCON0.TXEN = 1
C1FIFOCON0.FSIZE = 3

B
C1FIFOCON1.TXEN = 0
C1FIFOCON1.DONLY = 1
C1FIFOCON1.FSIZE = 1

C
C1FIFOONC3.TXEN = 0
C1FIFOCON3.DONLY = 0
C1FIFOCON3.FSIZE = 3

在图 34-9 给出的 CAN1 报文缓冲区 FIFO 配置中，FIFO0 具有 4 个发送缓冲区，共需要 16 个字（4 个缓冲区 x 每个缓冲区 4 个字）。 FIFO1 具有 2 个仅数据接收缓冲区，共需要 4 个字 （2 个缓冲区 x 每个缓冲区 2 个字）。FIFO3 具有 4 个完整接收缓冲区，共需要 16 个字（4 个缓冲区 x 每个缓冲区 4 个字）。因此，总共需要分配 36 个字 （16 + 4 + 16）的存储器。

假设存在图 34-10 中给出的另一个示例应用程序，它总共需要 4 个 FIFO。

**图 34-10： CAN FIFO 配置示例**

| | FIFO 编号 | 报文缓冲区起始地址 | 报文缓冲区 |
|---|---|---|---|
| A | FIFO0 | 0x00002000 | MB0 |
| | | 0x00002010 | MB1 |
| B | FIFO1 | 0x00002020 | MB0 |
| | | 0x00002030 | MB1 |
| C | FIFO2 | 0x00002040 | MB0 |
| | | 0x00002050 | MB1 |
| | | 0x00002060 | MB2 |
| D | FIFO3 | 0x00002070 | MB0 |
| | | 0x00002080 | MB1 |
| | | 0x00002090 | MB2 |

**CAN1FIFO 配置**
C1FIFOBA = 0x00002000

A C1FIFOCON0.TXEN = 1
C1FIFOCON0.SIZE = 1

B C1FIFOCON1.TXEN = 1
C1FIFOCON1.SIZE = 1

C C1FIFOCON2.TXEN = 0
C1FIFOCON2.SIZE = 2

D C1FIFOCON3.TXEN = 0
C1FIFOCON3.SIZE = 2

FIFO0 和 FIFO1 各具有 2 个报文缓冲区，并且均配置为 CAN 报文发送 FIFO。 FIFO2 和 FIFO3 各具有 3 个报文缓冲区，并且均配置为 CAN 报文接收 FIFO。 FIFO0 和 FIFO1 共需要 16 个字（2 个 FIFO x 每个 FIFO 具有 2 个报文缓冲区 x 每个报文缓冲区 4 个字）的存储器。 FIFO2 和 FIFO3 共需要 24 个字 （2 个 FIFO x 每个 FIFO 具有 3 个报文缓冲区 x 每个报文缓冲区 4 个字）的存储器。因此，总共需要分配 40 个字 （16 + 24）的存储器。

以下步骤可用于配置 CAN 模块 FIFO：

1. 为 CAN 报文缓冲区 FIFO 分配存储器。
2. 将模块置为配置模式 （OPMOD<2:0> = 100）。
3. 使用 FIFO 基址更新 CAN FIFO 基址 （CiFIFOBA）寄存器。它应为 FIFO0 的报文缓冲区 0 的物理起始地址。
4. 使用 FIFO 大小 （CiFIFOCONn 中的 FSIZE<4:0>）更新 FIFO 控制寄存器。
5. 选择 FIFO 是发送还是接收 FIFO （CiFIFOCONn 中的 TXEN 位）。
6. 将模块置为正常工作模式 （OPMOD<2:0> = 000）。

这些步骤如例 34-2 中的代码所示。该代码示例给出了用于设置例 34-10 中所示的 CAN 报文 FIFO 的编码步骤。

**例 34-2： 设置 CAN 报文 FIFO**

```
/* This code snippet illustrates the steps required to configure the */
/* PIC32 CAN Message FIFOs.The FIFO configuration example shown in */
/* Figure 34-5 will be used in this example. */

/* FIFO0 - Transmit - 2 MESSAGE BUFFERS */
/* FIFO1 - Transmit - 2 MESSAGE BUFFERS */
/* FIF02 - Receive - 3 MESSAGE BUFFERS */
/* FIF03 - Receive - 3 MESSAGE BUFFERS */
/* FIFO4 - FIFO31 - Not used */

/* Allocate a total of 40 words.

unsigned int CanFifoMessageBuffers[40];

/* Request CAN to switch to configuration mode and wait until it has switched */

C1CONbits.REQOP = 100
while(C1CONbits.OPMOD != 100);

/* Initialize C1FIFOBA register with physical address of CAN message Buffer */

C1FIFOBA = KVA_TO_PA(CanFifoMessageBuffers); ;

/* Configure FIFO0 */
C1FIFOCON0bits.FSIZE = 1;
C1FIFOCON0SET = 0x80;         /* Set the TXEN bit */

/* Configure FIFO1 */
C1FIFOCON1bits.FSIZE = 1;
C1FIFOCON1SET = 0x80;         /* Set the TXEN bit */

/* Configure FIFO2 */
C1FIFOCON2bits.FSIZE = 2;
C1FIFOCON2CLR = 0x80;         /* Clear the TXEN bit */

/* Configure FIFO3 */
C1FIFOCON3bits.FSIZE = 2;
C1FIFOCON3CLR = 0x80;         /* Clear the TXEN bit */

/* The CAN module can now be placed into normal mode if no further */
/* configuration is required. */

C1CONbits.REQOP = 0;

while(C1CONbits.OPMOD != 0);
```

### 34.6.2 将待发送报文装入 FIFO

为了使用 FIFO 来发送数据，必须将其配置为发送 FIFO。这通过将 CAN FIFO 控制器寄存器中的发送 / 接收缓冲区选择位 TXEN （CiFIFOCONn<7>）置 1 来实现。

CAN FIFO 用户地址 （CiFIFOUAn）寄存器提供在 FIFO 中应用程序用于存储报文的下一个报文缓冲区的物理地址 （也称为 FIFO 头部位置）。 CAN 报文应从 CiFIFOUAn 寄存器指定的位置开始装入。

在FIFO的一个报文缓冲区中装入报文之后，应用程序应将递增头部/尾部位UINC（CiFIFOCONn<13>）置 1。这会导致 CAN 模块将 CiFIFOUAn 寄存器中包含的地址递增 16 （从而指向下一个报文缓冲区）。此时，报文已准备好发送。在 CiFIFOUAn 寄存器达到 FIFO 结束位置时，它会计满返回到 FIFO 起始位置。

因此，应用程序不需要跟踪 FIFO 头部位置，可以使用 CiFIFOUAn 寄存器来实现该目的。以下编码步骤可用于将待发送报文装入发送 FIFO：

1. 读取 CiFIFOUAn 寄存器，并在该地址处存储一个报文 （16 字节）。
2. 将 UINC （CiFIFOCONn<13>）位置 1，以更新 CiFIFOUAn 寄存器。
3. 将 TXREQ （CiFIFOCONn<3>）位置 1，以发送报文。
4. 重复执行步骤 2 和 3，直到达到要排队报文数量和 FIFO 大小为止。

例 34-3 中的代码给出了这些编码步骤。在该示例中，有 4 个报文装入 CAN1 模块的 FIFO0。

**例 34-3： 装入发送报文 FIFO**

```
/* This code snippet illustrates the steps to load a transmit message FIFO. */
/* This example uses the CAN1 module. */

/* Four messages to be transmitted using transmit FIFO0 */

int message;   /* Tracks the message buffer */
unsigned int * currentMessageBuffer; /* Points to message buffer to be written */

message = 0;

for(message = 0; message <= 3; message ++)
{
    /* Get the address of the message buffer to write to from the C1FIFOUA0 */
    /* register.Convert this physical address to virtual address. */

    currentMessageBuffer = PA_TO_KVA1(C1FIF0UA0);

    /* This procedure will load the message
     * buffer with the message to be transmitted. */

    LoadMessageBuffer(currentMessageBuffer);

    C1FIFOCON0SET = 0x2008; /* Set the UINC and TXREQ bit */
}

/* At this point the messages are loaded in FIFO0 */
```

### 34.6.3 访问 FIFO 中的接收报文

为了使用FIFO来接收数据，必须将FIFO配置为接收FIFO。这通过清零TXEN（CiFIFOCONn<7>）位来实现。接收到报文之后，应用程序应从 CiFIFOUAn 寄存器中获取待读取的第一个报文缓冲区的物理起始地址 （也称为 FIFO 尾部指针）。然后，可以从该地址开始读取报文。

处理 FIFO 中的报文之后，应用程序应将 UINC 位 （CiFIFOCONn<13>）置 1，向 CAN 模块指示报文已进行了处理，可以覆盖相应的缓冲区。这将会递增尾部指针，使 CiFIFOUAn 指向的地址增大 4 个字或 2 个字，具体取决于 CiFIFOCONn 寄存器中 DONLY 位的值。在 CiFIFOUAn 寄存器达到 FIFO 结束位置时，它会计满返回到 FIFO 起始位置。

因此，应用程序不需要跟踪 FIFO 尾部位置，可以使用 CiFIFOUAn 寄存器来实现该目的。以下编码步骤可用于处理接收 FIFO 中的报文：

1. 使用任意可用中断来确定 FIFO 中是否存在报文。
2. 读取 CiFIFOUAn 寄存器，并处理该地址处的一个报文 （16 字节）。
3. 将 UINC 位 （CiFIFOCONn<13>）置 1，以更新 CiFIFOUAn 寄存器。
4. 重复执行步骤 1 和 2，直到中断条件清除为止。

例 34-4 中的代码演示了以上步骤。在该代码示例中，CAN1 模块的 FIFO1 配置为用于接收操作。该示例会一直读取 FIFO，直到它为空。

**例 34-4： 从 FIFO 中读取接收到的报文**

```
/* This code snippet illustrates the steps to read messages from receive */
/* message FIFO.This example uses the CAN1 module.*/

/* FIFO1 size is 4 messages and each message is 4 words long. */

unsigned int * currentMessageBuffer; /* Points to message buffer to be read */

while(1)
{
    /* Keep reading till the FIFO is empty. */
    while(C1FIFOINT1bits.RXNEMPTYIF == 1)
    {
        /* Get the address of the message buffer to read from the C1FIFOUA1 */
        /* register.Convert this physical address to virtual address. */

        currentMessageBuffer = PA_TO_KVA1(C1FIF0UA1);

        ProcessReceivedMessage(currentMessageBuffer);

        /* Set the UINC bit to tell the CAN module that
         * a message has been read. */

        C1FIFOCON0SET = 0x2000;
    }
}
```

### 34.6.4 复位 FIFO

FIFO 可以通过以下任意方法复位：

- 将 FRESET （CiFIFOCONn<14>）位置 1
- 将 CAN 模块置为配置模式 （OPMOD<2:0> （CiCON<23:21>） = 100）
- 关闭 CAN 模块 （CiCON<15> = 0）

复位 FIFO 将会复位头部和尾部指针、中断标志，以及 CiFIFOCONn 寄存器中的 TXABAT、TXLARB 和 TXERR 状态位。

在使用 FRESET 位复位 FIFO 之前，应当考虑以下方面：

- 如果 FIFO 是发送 FIFO，则不应有任何待处理的数据发送
- 如果 FIFO 是接收 FIFO，则不应有任何活动过滤器指向它

应用程序通常可以在退出禁止模式之后复位 FIFO。使用 FRESET 位复位 FIFO 时，应用程序必须对该位进行查询，以确保复位操作已完成。例 34-5 显示了此过程。

**例 34-5： 复位报文 FIFO**

```
/* This code snippet shows how to reset a message FIFO.This example */
/* uses FIFO0 of the CAN1 module. */

C1FIFOCON0SET = 0x00004000;           /* Set the FRESET bit */
while(C1FIFOCON0bits.FRESET == 1);
```

## 34.7 发送 CAN 报文

CAN 模块会发送装入发送 FIFO 的报文。关于将 FIFO 配置为用于发送操作的详细说明，请参见**第 34.6.1 节 “报文 FIFO 配置”**和**第 34.6.2 节 “将待发送报文装入 FIFO”**。

### 34.7.1 发送报文缓冲区的格式

发送报文 FIFO 最多可以具有 32 个 CAN 发送报文缓冲区。发送报文缓冲区为 4 字 （16 字节）长，具有表 34-2 中所述的固定格式。它包含 4 个字：CMSGnSID、CMSGnEID、CMSGnDATA0 和 CMSGnDATA1。根据 CAN 总线规范的定义，这些寄存器中的位与 CAN 报文中的位域一一对应。应用程序必须确保发送 FIFO 中的每个报文缓冲区都遵从图 34-11 至图 34-14 给出的格式。

**表 34-2： 系统存储器中存储的发送报文缓冲区格式**

| 地址偏移 | 名称 | | Bit 31/2315/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 00 | CMSGSID | 31:24 | -— | -— | -— | -— | -— | -— | -— | -— |
| | | 23:16 | -— | -— | -— | -— | -— | -— | -— | -— |
| | | 15:8 | -— | -— | -— | -— | -— | SID<10:8> | | |
| | | 7:0 | SID<7:0> | | | | | | | |
| 04 | CMSGEID | 31:24 | -— | -— | SRR | IDE | EID<17:14> | | | |
| | | 23:16 | EID<13:6> | | | | | | | |
| | | 15:8 | EID<5:0> | | | | | | RTR | RB1 |
| | | 7:0 | -— | -— | --— | RB0 | DLC<3:0> | | | |
| 08 | CMSGDATA0 | 31:24 | 发送缓冲区数据字节 3 | | | | | | | |
| | | 23:16 | 发送缓冲区数据字节 2 | | | | | | | |
| | | 15:8 | 发送缓冲区数据字节 1 | | | | | | | |
| | | 7:0 | 发送缓冲区数据字节 0 | | | | | | | |
| 0C | CMSGDATA1 | 31:24 | 发送缓冲区数据字节 7 | | | | | | | |
| | | 23:16 | 发送缓冲区数据字节 6 | | | | | | | |
| | | 15:8 | 发送缓冲区数据字节 5 | | | | | | | |
| | | 7:0 | 发送缓冲区数据字节 4 | | | | | | | |

**图注：** 阴影位应设置为 0。
**注 1：** CAN 发送报文存储在系统 RAM 中，没有与之关联的置 1/ 清零 / 取反寄存器。

**图 34-11： CMSGSID 的格式**

| SID<10:0> |
|---|

bit 10-0 **SID<10:0>：**标准标识符

**图 34-12： CMSGEID 的格式**

| — | SRR | IDE | EID<17:0> | RTR | RB1 | — | RB0 | DLC<3:0> |
|---|---|---|---|---|---|---|---|---|

bit 31-30 **未实现：**读为 0

bit 29 **SRR：**替代远程请求
如果是标准报文格式 （IDE = 0），则该位是无关位。
如果是扩展报文格式 （IDE = 1），则该位应总是置 1。

bit 28 **IDE：**扩展标识符
1 = 报文将发送扩展标识符
0 = 报文将发送标准标识符

bit 27-10 **EID<17:0>：**扩展标识符

bit 9 **RTR：**远程发送请求
1 = 报文是远程发送请求
0 = 报文不是远程发送请求

bit 8 **RB1：**保留的 bit 1
根据 CAN 总线规范，用户应用程序必须将该位设置为 0。

bit 7-5 **未实现：**读为 0

bit 4 **RB0：**保留的 bit 0
根据 CAN 总线规范，用户应用程序必须将该位设置为 0。

bit 3-0 **DLC<3:0>：**数据长度编码
1xxx = 模块将发送 8 字节
0111 = 将发送数据的 7 字节
•
•
•
0001 = 将发送数据的 1 字节
0000 = 将发送数据的 0 字节

**注 1：**该寄存器位于 CAN 模块外的系统存储器中。

**图 34-13： CMSGDATA0 的格式**

| DATA3<7:0> | DATA2<7:0> | DATA1<7:0> | DATA0<7:0> |
|---|---|---|---|

bit 31-24 **DATA3<7:0>：**数据字节 3
bit 23-16 **DATA2<7:0>：**数据字节 2
bit 15-8 **DATA1<7:0>：**数据字节 1
bit 7-0 **DATA0<7:0>：**数据字节 0

**注 1：**该寄存器位于 CAN 模块外的系统存储器中。

**图 34-14： CMSGDATA1 的格式**

| DATA7<7:0> | DATA6<7:0> | DATA5<7:0> | DATA4<7:0> |
|---|---|---|---|

bit 31-24 **DATA7<7:0>：**数据字节 7
bit 23-16 **DATA6<7:0>：**数据字节 6
bit 15-8 **DATA5<7:0>：**数据字节 5
bit 7-0 **DATA4<7:0>：**数据字节 4

**注 1：**该寄存器位于 CAN 模块外的系统存储器中。

例 34-6 中的代码给出了在存储器中实现 CAN 报文缓冲区的示例数据结构。例 34-7 给出了一个使用该结构的示例。

**例 34-6： 在存储器中实现 CAN 报文缓冲区**

```
/* This code snippet shows an example of data structure to implement a CAN */
/* message buffer. */

/* Define the sub-components of the data structure as specified in Table 34-2 */

/* Create a CMSGSID data type. */
typedefstruct
{
    unsigned SID:11;
    unsigned :21;
}txcmsgsid;

/* Create a CMSGEID data type. */
typedef struct
{
    unsigned DLC:4;
    unsigned RB0:1;
    unsigned :3;
    unsigned RB1:1;
    unsigned RTR:1;
    unsigned EID:18;
    unsigned IDE:1;
    unsigned SRR:1;
    unsigned :2;
}txcmsgeid;

/* Create a CMSGDATA0 data type. */
typedef struct
{
    unsigned Byte0:8;
    unsigned Byte1:8;
    unsigned Byte2:8;
    unsigned Byte3:8;
}txcmsgdata0;

/* Create a CMSGDATA1 data type. */
typedefstruct
{
    unsigned Byte4:8;
    unsigned Byte5:8;
    unsigned Byte6:8;
    unsigned Byte7:8;
}txcmsgdata1;

/* This is the main data structure. */

typedef union uCANTxMessageBuffer {

    struct
    {
        txcmsgsid CMSGSID;
        txcmsgeid CMSGEID;
        txcmsgdata0 CMSGDATA0;
        txcmsgdata0 CMSGDATA1;
    };
    int messageWord[4];
}CANTxMessageBuffer;
```

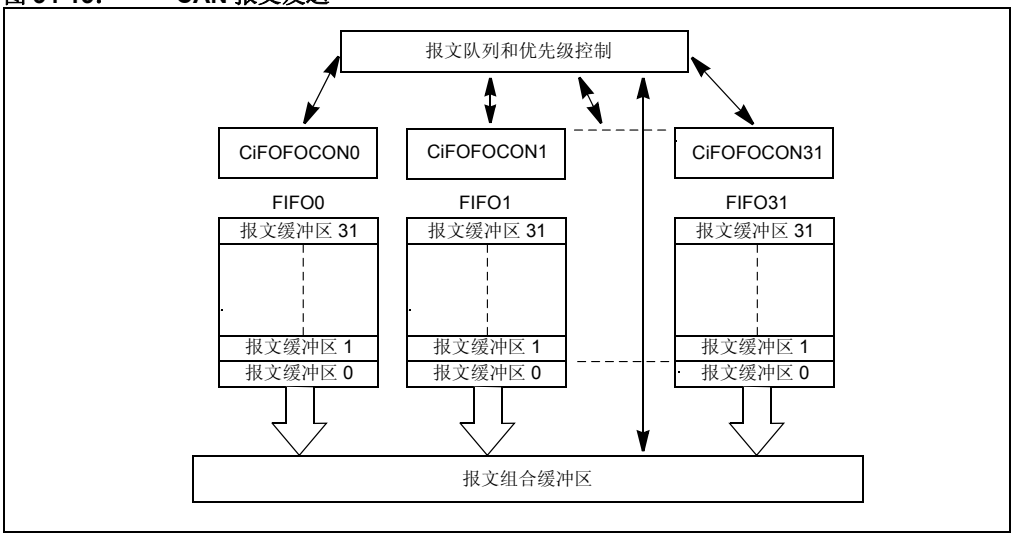
**例 34-7： 数据结构的示例用法**

```
/* Example usage of data structure shown in Example 34-6.This example sets */
/* up a message buffer in FIFO1 */

CANTxMessageBuffer * buffer;

buffer = (CANTxMessageBuffer *)(PA_TO_KVA1(C1FIFOUA1));

/* Clear the message buffer. */
buffer->messageWord[0] = 0;
buffer->messageWord[1] = 0;
buffer->messageWord[2] = 0;
buffer->messageWord[3] = 0;

buffer->CMSGSID.SID = 0x100;/* Message SID */
buffer->CMSGEID.DLC = 0x2;/* Data Length is 2 */
buffer->CMSGDATA0.Byte0 = 0xAA;/* Byte 0 Data */
buffer->CMSGDATA0.Byte1 = 0xbb;/* Byte 1 Data */
```

### 34.7.2 请求发送报文

图 34-15 给出了 CAN 模块发送过程。应用程序必须将 FIFO 配置为用于发送操作。报文发送和优先级由 CiFIFOCONn 寄存器 （寄存器 34-20）中的位控制。然后， CAN 模块会根据就绪状态和优先级选择要发送的下一个报文，并处理该报文使之可用于发送。

**图 34-15： CAN 报文发送**




将报文装入 FIFO 并准备好发送之后，用户应用程序可以通过将 CAN FIFO 控制寄存器中的报文发送请求位 TXREQ（CiFIFOCONn<3>）置 1 来启动报文发送。当该位置 1 时， FIFO 的内容将根据报文优先级排入发送队列， CAN 模块将会发送 FIFO 中的所有报文。发送所有报文之后，TXREQ 位将清零。应用程序可以装入所有发送 FIFO，并将所有这些 FIFO 的 TXREQ 位置 1。或者，应用程序也可以装入一个发送 FIFO、将关联的 TXREQ 位置 1，然后在前一个 FIFO 进行处理时装入下一个 FIFO。

在发送一个报文时，可以向 FIFO 末尾附加其他报文。应用程序应使用 CiFIFOUAn 寄存器来获取 FIFO 头部指针，并将报文存储在该地址处。附加的报文将排入发送队列。

| **注：** | 建议在每个报文装入FIFO之后（在UINC位置1之后），应用程序才将TXREQ位置1。 |
|---|---|

以下编码步骤可用于发送 CAN 报文：

1. 配置发送操作所需的 FIFO 数量。
2. 如果需要，设置各个 FIFO 的优先级。
3. 使用表 34-2 中给出的格式创建发送报文。
4. 读取 CiFIFOUAn 寄存器，并将报文存储在所提供的地址处。
5. 将 UINC 位置 1。
6. 将 TXREQ 位置 1。
7. 重复步骤 3 至 6，直到待发送报文数量达到所需 FIFO 数量为止。
8. 发送 FIFO 中断可用于监视 FIFO 的状态。

例 34-8 中的代码给出了使用 CAN1 模块发送 4 个报文时，发送一个 CAN 报文所需的步骤。报文 0 和 1 是标准 ID CAN 报文。报文 2 和 3 是扩展 ID 报文。使用 FIFO1，其长度为 3。

**例 34-8： 发送标准和扩展 ID 报文**

```
/* This code example illustrates how to transmit standard and extended */
/* ID messages with the PIC32 CAN module. */

/* The code example uses CAN1, FIFO0.FIFO0 size is 4 */

/* Four CAN message have to be transmitted.                    */
/* Msg 0:SID - 0x100 Data - 0x12BC1245                         */
/* Msg 1:SID - 0x102 Data - 0x12BC124512BC1245                 */
/* Msg 2:SID - 0x100 EID - 0xC000 Data - 0x12BC1245            */
/* Msg 3:SID - 0x102 EID - 0xC000 Data - 0x12BC124512BC1245    */

/* Pointer to CAN Message Buffer */
CANTxMessageBuffer * transmitMessage;

/* This array is the CAN FIFO and message buffers.FIF0 has 4 message */
/* buffers.All other buffers are default size. */

unsigned int CANFIFO[16];

/* Place CAN module in configuration mode */

C1CONbits.REQOP = 4;
while(C1CONbits.OPMOD != 4);

/* Initialize the C1FIFOBA with the start address of the CAN FIFO message */
/* buffer area. */

C1FIFOBA = KVA_TO_PA(CANFIFO);

/* Set FIFO0 size to 3 messages.All other FIFOs at default size. */
/* Configure FIFO0 for transmit.*/

C1FIFOCON0bits.FSIZE = 2
C1FIFOCON0SET = 0x00000080; /* Set the TXEN bit - Transmit FIFO */

/* Place the CAN module in Normal mode. */

C1CONbits.REQOP = 0;
while(C1CONbits.OPMOD != 0);

/* Get the address of the message buffer to write to.Load the buffer and */
/* then set the UINC bit.Set the TXREQ bit next to send the message. */

/* Message 0 SID - 0x100 Data - 0x12BC1245*/

transmitMessage = (CANTxMessageBuffer *)(PA_TO_KVA1(C1FIFOUA1));
transmitMessage->CMSGSID.SID = 0x100;            /* CMSGSID */
transmitMessage->CMSGEID.IDE = 0;
transmitMessage->CMSGEID.DLC = 0x4;
transmitMessage->messageWord[2] = 0x12BC1245;    /* CMSGDAT0 */
C1FIFOCON1SET = 0x00002000;                      /* Set the UINC bit */
C1FIFOCON1SET = 0x00000008;                      /* Set the TXREQ bit */

/* Message 1 SID - 0x102 Data - 0x12BC124512BC1245 */

transmitMessage = (CANTxMessageBuffer *)(PA_TO_KVA1(C1FIFOUA1));
transmitMessage->CMSGSID.SID = 0x102;            /* CMSGSID */
transmitMessage->CMSGEID.IDE = 0;
transmitMessage->CMSGEID.DLC = 0x8;
transmitMessage->messageWord[2] = 0x12BC1245;    /* CMSGDAT0 */
transmitMessage->messageWord[3] = 0x12BC1245;    /* CMSGDAT1 */
C1FIFOCON1SET = 0x00002000;                      /* Set the UINC bit */
C1FIFOCON1SET = 0x00000008;                      /* Set the TXREQ bit */
```

**例 34-8： 发送标准和扩展 ID 报文 （续）**

```
/* Message 2 SID - 0x100 EID - 0xC000 Data - 0x12BC1245*/

transmitMessage = (CANTxMessageBuffer *)(PA_TO_KVA1(C1FIFOUA1));
transmitMessage->CMSGSID.SID = 0x100;            /* CMSGSID */
transmitMessage->CMSGEID.SID = 0xC000;           /* CMSGEID */
transmitMessage->CMSGEID.IDE = 1;
transmitMessage->CMSGEID.DLC = 0x4;
transmitMessage->messageWord[2] = 0x12BC1245;    /* CMSGDAT0 */
C1FIFOCON1SET = 0x00002000;                      /* Set the UINC bit */
C1FIFOCON1SET = 0x00000008;                      /* Set the TXREQ bit */

/* Message 3 SID - 0x102 EID - 0xC000 Data - 0x12BC124512BC1245*/

transmitMessage = (CANTxMessageBuffer *)(PA_TO_KVA1(C1FIFOUA1));
transmitMessage->CMSGSID.SID = 0x102;            /* CMSGSID */
transmitMessage->CMSGEID.SID = 0xC000;           /* CMSGEID */
transmitMessage->CMSGEID.IDE = 1;
transmitMessage->CMSGEID.DLC = 0x8;
transmitMessage->messageWord[2] = 0x12BC1245;    /* CMSGDAT0 */
transmitMessage->messageWord[3] = 0x12BC1245;    /* CMSGDAT1 */
C1FIFOCON1SET = 0x00002000;                      /* Set the UINC bit */
C1FIFOCON1SET = 0x00000008;                      /* Set the TXREQ bit */

}
```

### 34.7.3 发送报文优先级

CAN 模块允许应用程序控制发送报文缓冲区的优先级。在发送报文 SOF 之前，CAN 模块会比较所有准备好发送的缓冲区的优先级。优先级最高的发送报文缓冲区最先发送。例如，如果发送 FIFO0 的优先级高于发送 FIFO1，则会先发送 FIFO0 中的所有报文。如果在处理一个 FIFO 时，另一个 FIFO 的优先级发生改变， CAN 模块会在当前报文完成发送之后，重新评估所有 FIFO 的优先级。这使 FIFO 优先级变化可以尽早生效。

优先级由 TXPRI<1:0> （CiFIFOCONn<1:0>）位控制。 TXPRI 位可定义 4 种发送优先级。选择如下：

- 11 = 该 FIFO 具有最高优先级
- 10 = 该 FIFO 具有中高优先级
- 01 = 该 FIFO 具有中低优先级
- 00 = 该 FIFO 具有最低优先级

如果两个 FIFO 的优先级相同，则会发送自然顺序最高的 FIFO。在所有 FIFO 优先级相同时，发送的自然顺序为：

- FIFO0——最低自然优先级
- FIFO1——较高自然优先级
  .
  .
  .
- FIFO31——最高自然优先级

### 34.7.4 中止已排队报文的发送

通过清零 TXREQ 位 （CiFIFOCONn<3>），可以中止已排入发送队列的报文。如果 TXREQ 清零，模块会尝试中止报文发送。

如果报文成功中止，硬件会将 TXABT （CiFIFOCONn<2>）位置 1。 TXREQ 将一直保持置 1，直到报文中止或发送成功为止。

如果报文成功发送，FIFO 指针将按正常情况更新。如果报文成功中止，FIFO 指针将不改变。然后如果需要，用户可以使用内部索引 （CFIFOCI）来确定哪些报文已发送。

要复位 FIFO 指针并擦除所有待发送报文，用户应用程序可以将 FRESET（CiFIFOCONn<14>）位置 1。然后，可以重新使能 FIFO 并装入要发送的新报文。

### 34.7.5 远程发送请求

如**第 34.7.2 节 “请求发送报文”**中所述， CAN 总线系统提供了让一个节点向另一个节点请求数据的方法。请求节点会发送 RTR 位置 1 的报文。报文中不包含任何数据，仅包含用于触发过滤器匹配的地址。

配置为用于响应远程发送请求的过滤器将指向配置为用于发送的 FIFO。此外，还必须通过设置 RTREN = 1 使能 FIFO 来答复远程发送请求。

远程发送请求可以在无需 CPU 干预的情况下进行处理。如果发送 FIFO 进行了正确配置，则在过滤器发生匹配，并且过滤器指向 FIFO 时，缓冲区将会排入发送队列。 FIFO 必须配置如下：

1. 通过将 TXEN （CiFIFOCONn<7>）位设置为 1，将 FIFO 设置为发送模式。
2. 必须使能一个过滤器，并在其中装入匹配的报文标识符。
3. 该过滤器的缓冲区指针寄存器必须指向发送缓冲区 （请注意，虽然过滤器通常指向接收 FIFO，但在这种情况下它必须指向发送 FIFO）。
4. RTREN （CiFIFOCONn<2>）位必须设置为 1，以使能 RTR。
5. FIFO 中必须预先装入至少一个要发送的报文。

当接收到远程发送请求报文，并且它与一个指向已正确配置发送缓冲区的过滤器匹配时，TXREQ（CiFIFOCONn<3>）位会自动置 1。然后， FIFO 中的报文缓冲区将按照优先级顺序发送。如果在出现远程发送请求时，发送缓冲区中没有报文可用，则该事件会被视为 FIFO 溢出事件， CAN FIFO 中断寄存器中的接收 FIFO 溢出中断标志位 RXOVFLIF（CiFIFOINTn<3>）会置 1。例 34-9 中的代码给出了说明一个节点如何向远程节点请求数据的示例。例 34-10 中的代码给出了说明如何将一个节点配置为响应远程发送请求的示例。

**例 34-9： 远程发送请求**

```
/* This code snippet shows an example of a Remote Transmit Request.The */
/* node in this case will request a transmission from an application (or */
/* node) with an SID of 0x100. */

CANMessageBuffer * rtrMessage;
rtrMessage = (CANMessageBuffer *)(PA_TO_KVA1(C1FIFOUA1));

/* Clear the message. */
rtrMessage->messageWord[0] = 0x0;
rtrMessage->messageWord[1] = 0x0;
rtrMessage->messageWord[2] = 0x0;
rtrMessage->messageWord[3] = 0x0;

rtrMessage->CMSGSID.SID = 0x100;
rtrMessage->CMSGEID.IDE = 0;
rtrMessage->CMSGEID.RTR = 1;
rtrMessage->CMSGEID.DLC = 0;
C1FIFOCON SET = 0x00002000; /* Set the UINC bit */
/* The Remote Transmit Request message (rtrMessage) is now ready to be sent.*/
```

**例 34-10： 响应远程发送请求**

```
/* This code example shows how to configure the CAN1 module to respond */
/* to a remote transmit request. */

/* In this case, FIFO1 is configured to respond to the a remote request */
/* on SID = 0x100. */

/* Allocate CAN FIFO memory. */
unsigned int CANFIFO[140];

/* This is the pointer to the reply message. */
CANMessageBuffer * rtrReply;

/* Place CAN Module in configuration mode.*/

C1CONbits.REQOP = 4;
while(C1CONbits.OPMOD != 4);

/* Configure FIFO1 for transmit operation, 4 message buffers and enable */
/* Auto Remote Transmit. */

C1FIFOCON1SET = 0x00000080; /* Set the TXEN bit */
C1FIFOCON1SET = 0x00000004; /* Enable RTR */
C1FIFOCON1bits.FSIZE = 4;

/* Configure a filter to accept a message with SID = 0x100.Refer to */
/* Section 34.8 "CAN Message Filtering" for more details on configuring */
/* filters.In this case, Filter 0 and Mask0 are used. */

C1FLTCON0bits.FSEL0 = 1;                /* Point to FIF01 */
C1FLTCON0bits.MSEL0 = 0;                /* Select Mask 0 */

C1RXF0bits.SID = 0x100;                 /* Configure Filter 0. */
C1RXF0bits.EXID = 0;

C1RXM0bits.SID = 0x1FF;                 /* Configure Mask 0. */
C1RXM0bits.MIDE = 1;

C1FLTCON0SET = 0x00000080;              /* Enable the filter. */

/* Assign FIFO memory to CAN module */
C1FIFOBA = KVA_TO_PA(CANFIFO);

/* Place CAN Module in normal mode.*/

C1CONbits.REQOP = 0;
while(C1CONbits.OPMOD != 0);

/* Form the remote reply message. */

rtrReply = (CANMessageBuffer *)(PA_TO_KVA1(C1FIFOUA1));
rtrReply->messageWord[0] = 0;
rtrReply->messageWord[1] = 0;
rtrReply->messageWord[2] = 0;
rtrReply->messageWord[3] = 0;

rtrReply->CMSGSID.SID = 0x100;          /* CMSGSID */
rtrReply->CMSGEID.IDE = 0;
rtrReply->CMSGEID.DLC = 0x4;
rtrReply->messageWord[2] = 0x12BC1245;  /* CMSGDAT0 */

C1FIFOCON1bits.UINC = 1;

/* FIFO is now ready to respond to RTR. */
```

### 34.7.6 报文发送 FIFO 行为示例

假设有一个使用 CAN1 模块的两个 FIFO（FIFO0 和 FIFO1）的应用程序示例。FIFO0 具有 4 个配置为用于 CAN 报文接收的报文缓冲区。 FIFO1 具有 7 个报文缓冲区，配置为用于报文发送。FIFO2 至 FIFO31 保留为默认状态。 CAN 报文缓冲区从物理地址 0x00000100 开始。该值装入 C1FIFOBA 寄存器中。图 34-16 显示了该应用程序配置。

**图 34-16： FIFO 行为示例的 FIFO 配置**

| | FIFO 编号 | 报文缓冲区起始地址 | 报文缓冲区 | CAN1FIFO 配置<br>C1FIFOBA = 0x00000100 |
|---|---|---|---|---|
| Ⓐ | FIFO0（完整接收报文） | 0x00000100 | MB0 | Ⓐ C1FIFOCON0.TXEN = 1<br>C1FIFOCON0.FSIZE = 3 |
| | | 0x00000110 | MB1 | |
| | | 0x00000120 | MB2 | |
| | | 0x00000130 | MB3 | |
| Ⓑ | FIFO1（发送 FIFO） | 0x00000140 | MB0 | Ⓑ C1FIFOCON1.TXEN = 1<br>C1FIFOCON1.SIZE = 6 |
| | | 0x00000150 | MB1 | |
| | | 0x00000160 | MB2 | |
| | | 0x00000170 | MB3 | |
| | | 0x00000180 | MB4 | |
| | | 0x00000190 | MB5 | |
| | | 0x000001A0 | MB6 | |
| Ⓒ | FIFO2 | 0x000001B0 | MB0 | Ⓒ FIFO2 - FIFO31 未配置 |
| | FIFO4 | 0x000001C0 | MB0 | |
| | • | • | • | |
| | • | • | • | |
| | • | • | • | |
| | FIFO31 | 0x00000380 | MB0 | |

FIFO1 的起始地址可以根据 CAN 报文缓冲区基址 （0x00000100）和 FIFO0 的字节大小 （4 个报文缓冲区 * 16 = 0x40）计算。FIFO1 的物理起始地址为 0x00000140。剩下的讨论将集中于发送 FIFO1。

图34-17显示了FIFO1为空且尚未写入任何报文的情形。FIFO发送未满中断标志（TXNFULLIF）、FIFO 发送半空中断标志 （TXHALFIF）和 FIFO 发送为空中断标志 （TXEMPTYIF）均置 1。

**图 34-17： 开始时的 FIFO1**

CFIFOBA = 0x00000100
CFIFOUA = 0x00000140
CFIFOCI = 0x00000000

TXNFULLIF - 1
TXHALFIF - 1
TXEMPTYIF - 1
TXREQ - 0

| |
|---|
| MB6 |
| MB5 |
| MB4 |
| MB3 |
| MB2 |
| MB1 |
| MB0 |

图 34-18 显示了第一个报文装入 FIFO 之后的 FIFO1。可以注意到， FIFO1 中的第一个报文缓冲区 MB0 包含了数据。应用程序会将 UINC （C1FIFOCON1<13>）位置 1，这会导致 FIFO 头部（C1FIFOUA1）前移并指向下一个空报文缓冲区 MB1。由于 FIFO 非空，TXEMPTYIF 标志会清零。此时，应用程序已通过将 TXREQ （C1FIFOCON1<3>）位置 1 请求了报文发送。

**图 34-18： FIFO1——第 1 次写入**

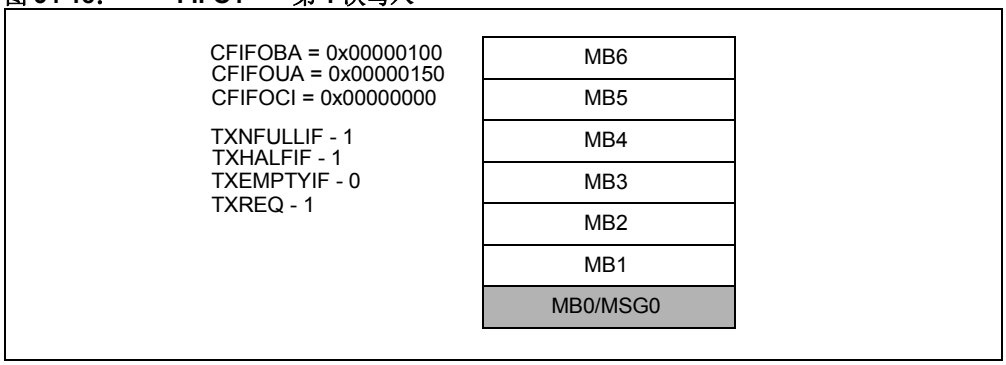



图 34-19 显示了从 MB0 中发送第一个报文之后的 FIFO1。TXREQ 已清零，TXEMPTYIF 再次置 1。可以注意到， CAN 模块报文索引寄存器 C1FIFOCI1 现在指向第二个报文缓冲区 MB1 （地址 0x00000150 处）。

**图 34-19： FIFO1——发送第一个报文**

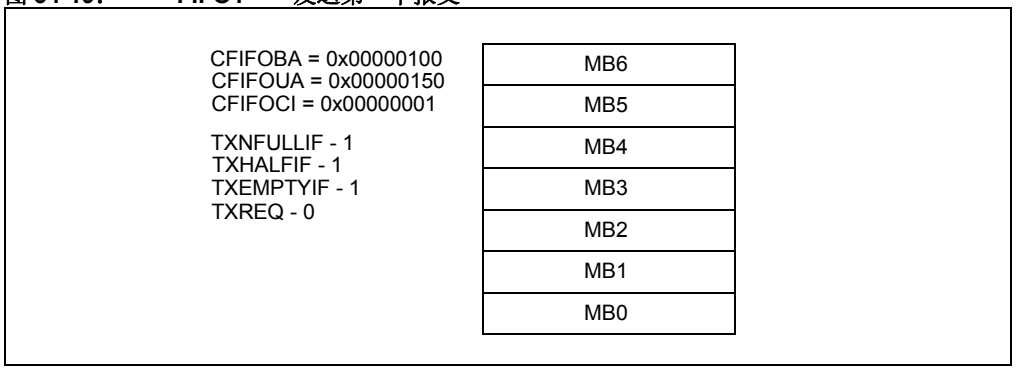



图 34-20 显示了再次在 MB1 至 MB6 中装入 6 个报文之后的 FIFO。在装入报文时，应用程序需要读取 C1FIFOUA1，以获取要写入的下一个报文缓冲区的地址位置。 TXHALFIF 标志已清零，应用程序尚未请求发送数据 （TXREQ = 0）。

**图 34-20： FIFO1——第 7 次写入 （即将填满）**

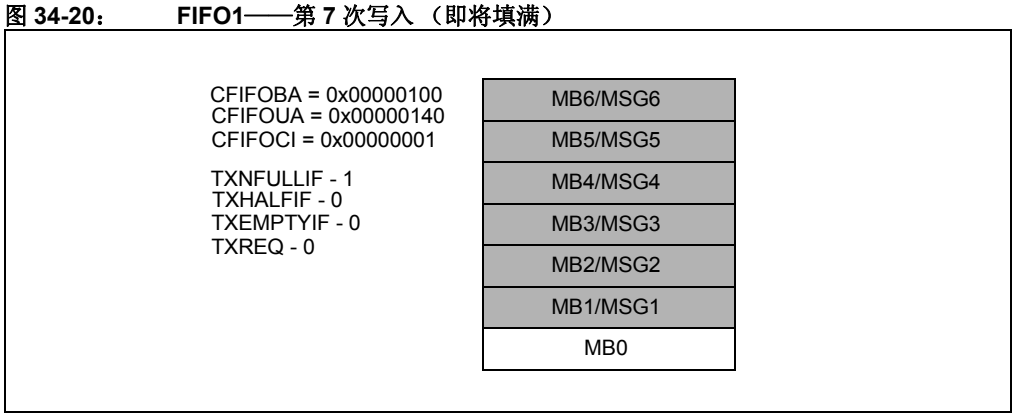


图 34-21 显示了装入第 8 个报文（此次装入 MB0）之后的 FIFO。 TXNFULLIF 和 TXHALFIF 标志均已清零。FIFO 头部现在指向 MB1。FIFO 在此阶段变为全满。应用程序已请求要发送该报文（TXREQ = 1）。

**图 34-21： FIFO1——第 8 次写入 （缓冲区已满）**

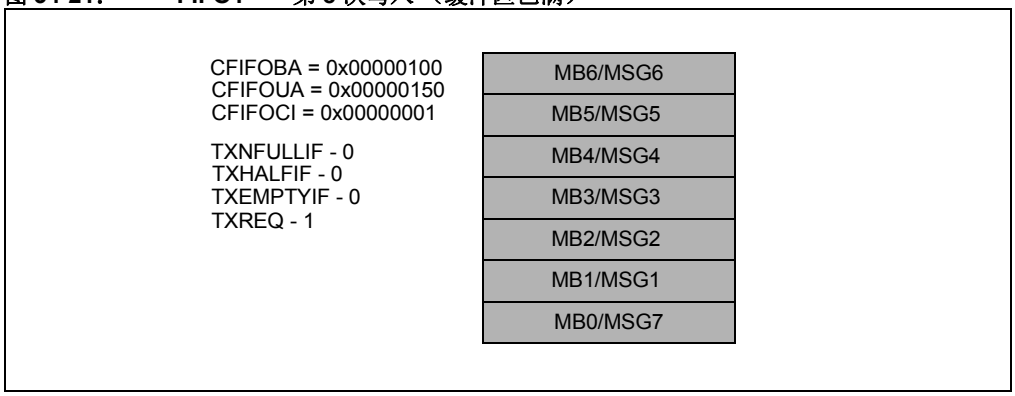
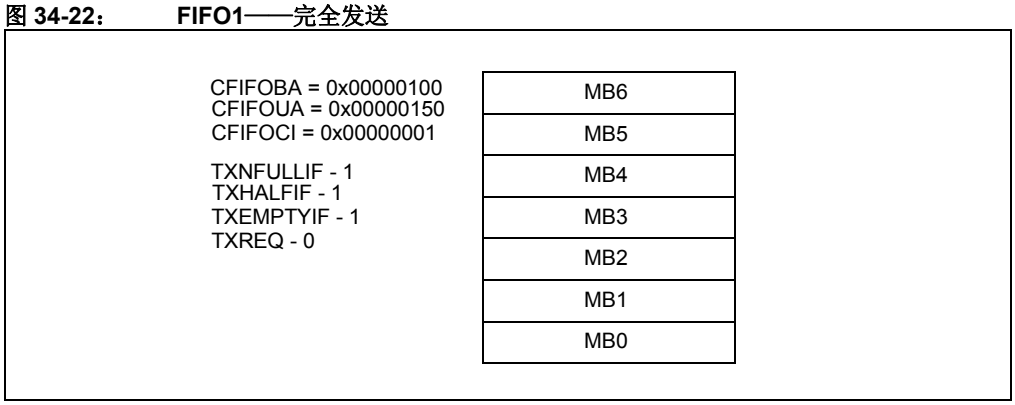



图 34-22 显示了 FIFO 再次变为空。 CAN 模块已发送了 FIFO 中的全部 7 个报文。 TXEMPTYIF 再次置 1， TXREQ 已由硬件清零。 C1FOCI1 已发生折回，再次指向 MB1。

**图 34-22： FIFO1——完全发送**



## 34.8 CAN 报文过滤

CAN 网络是一种广播式网络。一个节点发送报文时，网络上的所有节点都会接收到报文。每个 CAN 节点需要通过过滤机制来接收感兴趣的报文。这种过滤机制通过 CAN 报文接收过滤器和屏蔽器寄存器来实现。过滤根据 CAN 报文的 ID 字段来执行。

PIC32MX CAN 模块总共有 32 个接收过滤器和 4 个屏蔽器寄存器。应用程序需要将特定过滤器配置为接收带有给定标识符的报文，配置方法是将过滤器设置为匹配要接收报文的标识符。

每个过滤器都由自己的 CAN 过滤器控制（CiFLTCONn）寄存器控制。对于每个过滤器，该寄存器具有以下位：

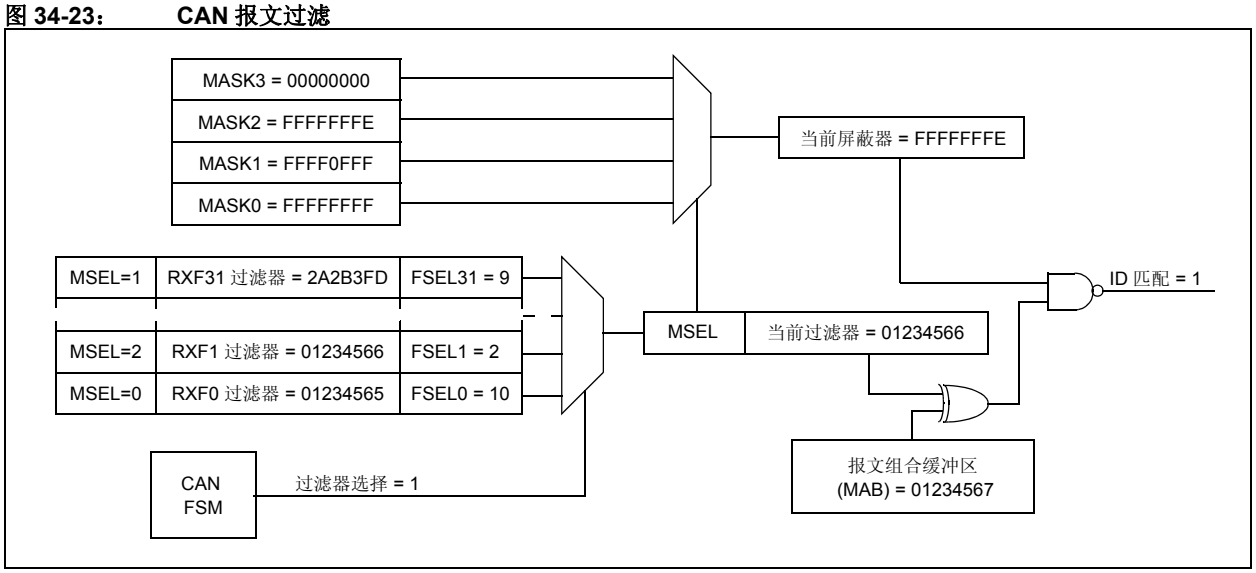
- FLTENn （n = 0 至 31）——过滤器使能位，用于使能 / 禁止过滤器
- MSELn<1:0> （n = 0 至 31）——屏蔽器选择，定义要在过滤器比较中应用的屏蔽器
- FSELn<4:0> （n = 0 至 31）——缓冲区指针，定义 ID 与过滤器匹配的报文的目标 FIFO

在模块接收到报文时，会将报文标识符与过滤器中的相应位进行比较。如果标识符与用户配置的过滤器匹配，报文会被存储到 CAN 过滤器控制 n 寄存器中的 FIFO 选择位 FSELn<4:0>（CiFLTCONn<4:0>）所指向的 FIFO 中。

接收屏蔽器可用于在接收时忽略标识符的选定位。在接收报文时，这些位将不与过滤器中的位进行比较。例如，如果用户希望接收带有标识符 0、1、2 和 3 的所有报文，用户需要屏蔽掉标识符的低 2 位。共有 4 个屏蔽器寄存器可供使用。过滤器可以与 4 个屏蔽器中的任意一个关联。

图 34-23 给出了说明过滤如何工作的原理图。接收到报文之后， CAN 模块会循环测试所有过滤器。在选择每个过滤器时，也会使用MSELn<1:0>位选择相应的屏蔽器。报文组合缓冲区（MAB）中的报文将与过滤器进行比较 （异或运算）。需要排除过滤的任意位都使用选定屏蔽器来进行排除。如果所有条件均匹配，则会发生 ID 匹配， CAN 模块会将 MAB 的内容移入 FSELn<4:0> 位指向的相应 FIFO 中。

**图 34-23： CAN 报文过滤**




在图 34-23 给出的示例中，接收报文的 ID 为 01234567，匹配度最高的过滤器是过滤器 1，它的 ID 为 01234566。可以注意到，除了最低有效位 （LSb），它们两个完全匹配。由于选定屏蔽器（屏蔽器 2）设置为忽略报文的最低有效位，所以仍然会发生匹配。由于过滤器 1 的 FSELn 位设置为 2，之后 CAN 模块会将报文存储到 FIFO2 中。

### 34.8.1 使能和修改过滤器

可以使用 CiFLTCONn 寄存器中的过滤器使能 （FLTENn）位开启或关闭过滤器。清零该位会关闭相应的过滤器，将它置1则会开启过滤器。当CAN模块处于配置模式（OPMOD<2:0> = 100）、正常工作模式（OPMOD<2:0> = 000）或禁止模式（OPMOD<2:0> = 001）时，可以修改过滤器。

在处理接收报文时，CAN 模块会延迟禁止过滤器。FLTENn 位将一直保持置 1，直到过滤完成为止。如果在过滤期间报文与该过滤器匹配，接收到的报文会被写入过滤器指向的缓冲区中。在处理完报文之后，过滤器会被禁止。

在修改过滤器之前，用户应用程序应先查询 FLTENn 位，以确保过滤器已被禁止。当过滤器使能时，对过滤器寄存器的写操作会被阻止。

### 34.8.2 扩展和标准标识符

CAN报文可以具有11位的标准标识符或29位的扩展标识符。CAN接收过滤器屏蔽器n（CiRXFn）寄存器中的 SID<11:0> 位和 EID<17:0> 位应配置为与所需报文的 SID 或 SID 和 EID 匹配。将扩展标识符使能 （EXID）位 （CiRXFn<19>）置 1 时，过滤器将匹配带有扩展标识符的报文。清零 CiRXFn 寄存器中的 EXID 位时，过滤器将匹配带有标准标识符的报文。如果屏蔽器标识符接收模式 （MIDE）位 （CiRXMn<19>）清零，则会忽略该报文类型，并接受与过滤器匹配的所有报文类型。

### 34.8.3 接收屏蔽器

CAN 模块中具有 4 个专用的屏蔽器寄存器：屏蔽器寄存器 0、1、2 和 3。所有过滤器都可以选择以下 4 个屏蔽器选项之一：

- 屏蔽器寄存器 0
- 屏蔽器寄存器 1
- 屏蔽器寄存器 2
- 屏蔽器寄存器 3

接收过滤器的屏蔽器选择通过相应 CiFLTCONn 寄存器中对应于过滤器的 MSELm<1:0> 位进行控制。在使能使用屏蔽器的过滤器时，将无法修改屏蔽器寄存器。在修改屏蔽器之前，需要先禁止过滤器。如果给定过滤器不需要进行屏蔽，则相应的屏蔽器位应设置为 1。这会导致过滤逻辑在执行过滤操作时考虑所有的过滤器位。表 34-3 列出了 CAN 报文中的 IDE 位、CiRXFn 寄存器中的 EXID 位和 CiRXMn 寄存器中的 MIDE 位的各种组合的真值表。

| **注：** | 只有在 CAN 模块处于配置模式时，才能修改 CiRXMn 寄存器。 |
|---|---|

**表 34-3： 标准 / 扩展报文接收真值表**

| 报文 IDE 位 | 过滤器 EXID 位 | 屏蔽器 MIDE 位 | 备注 | 接收报文 |
|---|---|---|---|---|
| 0 | 0 | 1 | 标准报文，过滤器指定仅标准， SID 匹配 | 是 |
| | | | 标准报文，过滤器指定仅标准， SID 不匹配 | 否 |
| 1 | 1 | 1 | 扩展报文，过滤器指定仅扩展， SID 和 EID 匹配 | 是 |
| | | | 扩展报文，过滤器指定仅扩展， SID 或 EID 不匹配 | 否 |
| 1 | 0 | 1 | 扩展报文，过滤器指定仅标准 | 否 |
| 0 | 1 | 1 | 标准报文，过滤器指定仅扩展 | 否 |
| 0 | x | 0 | 标准报文，过滤器指定忽略类型， SID 匹配 | 是 |
| | | | 标准报文，过滤器指定忽略类型， SID 不匹配 | 否 |
| 1 | x | 0 | 扩展报文，过滤器指定忽略类型， SID 和 EID 匹配 | 是 |
| | | | 扩展报文，过滤器指定忽略类型， SID 或 EID 不匹配 | 否 |

**图注：** x = 无关位

### 34.8.4 缓冲区指针

与接收过滤器的控制寄存器关联的是 FIFO 指针 （FSELn）位。它用作相应过滤器的地址指针。在标识符进入时，如果发生匹配，则对应于该过滤器的FSELn输出会被使能，并锁存到接收FIFO存储器的地址指针中。在报文组合到 MAB 中之后，它会被写入选定 FIFO。与报文匹配的过滤器的地址会被装入接收缓冲区的 FILHIT 位。

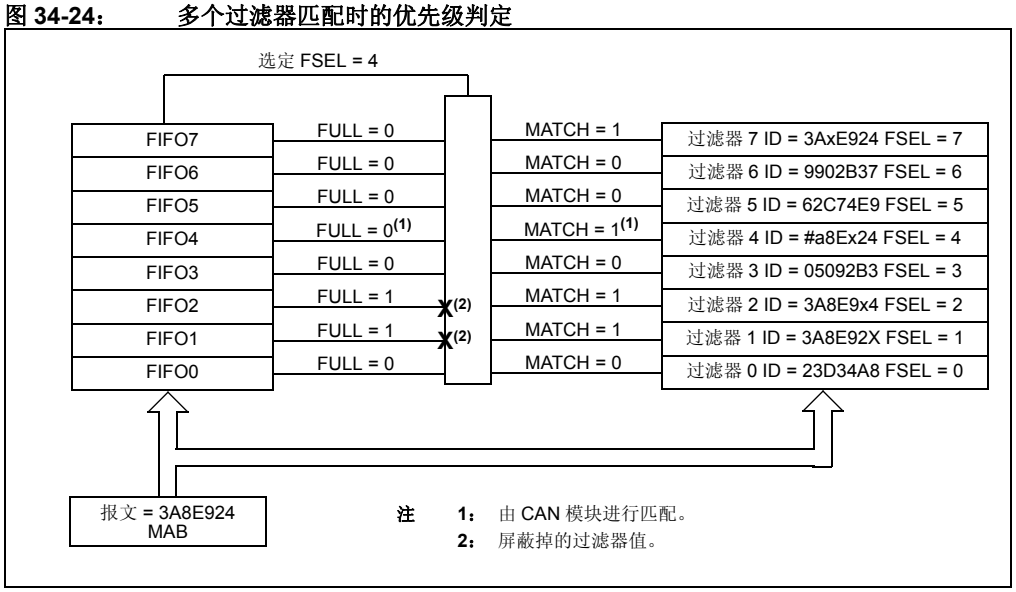
使用缓冲区指针意味着可以将任意过滤器设置为指向任意 FIFO，任意 FIFO 可以有多个过滤器指向它。此示例可以说明如何在系统中使用这种功能：将所有低优先级的接收报文放入一个很大的 FIFO （CPU 很少进行处理），高优先级报文放入第二个 FIFO （在报文到达时即进行处理）。

### 34.8.5 处理具有相同 ID 的报文

由于模块中提供了报文过滤和屏蔽选项，有可能发生一个报文与多个过滤器匹配的情况。如果有两个或更多过滤器与接收到的报文匹配，并且它们指向不同的接收缓冲寄存器，则只会装入一个接收缓冲寄存器。装入报文的接收缓冲寄存器将由过滤器编号和缓冲区可用性决定。编号较低的过滤器总是优先于编号较高的过滤器。例如，过滤器 0 将总是优先于过滤器 1。如果过滤器 0 指向的接收缓冲寄存器已满，那么报文会被装入与匹配过滤器关联的下一个可用 FIFO。

图 34-24 给出了多个过滤器匹配时的优先级判定示例。 MAB 中的报文与过滤器 1、2、4 和 7 匹配。过滤器 1 的自然优先级最高，但关联 FIFO 已满，过滤器 2 也是如此。过滤器 4 的优先级仅次于它们，并且有空间。报文会被存储到 FIFO4 中。

**图 34-24： 多个过滤器匹配时的优先级判定**



### 34.8.6 配置过滤器

要为标准 ID 的 CAN 报文配置过滤器，应执行以下步骤：

1. 使用所需 CAN 报文的 SID 更新 CiRXFn 寄存器（n 代表所需的过滤器）的 SID<10:0> 位。
2. 清零 CiRXFn 寄存器中的 EXID 位。这会导致过滤器仅匹配标准 ID 报文。
3. 创建过滤器的屏蔽器。在 CiRXMn 寄存器中装入相应值。将 MIDE 位置 1，以仅匹配标准 ID 报文。
4. 确定控制选定过滤器的 CiFLTCONn 寄存器。设置 FSELn 位的值，使之指向接收 FIFO。设置 MSELn 位，以使用在步骤 3 中配置的屏蔽器。
5. 通过将 CiFLTCONn 寄存器中的 FLTENn 位置 1 来使能过滤器。

例 34-11 中的代码给出了为标准 ID 配置 CAN 模块报文接收过滤器的示例。

**例 34-11： 配置过滤器来匹配 SID CAN 报文**

```
/* This code example shows how to configure a filter to match a Standard */
/* Identifier (SID) CAN message. */

/* In this example, Filter 3 is set up to accept messages with a SID range */
/* of 0x100 to 103. Accepted messages will be stored in FIFO5.Mask 1 is */
/* used to implement the filter address range. */

C1FLTCON0bits.FSEL3 = 5;  /* Store messages in FIFO5 */
C1FLTCON0bits.MSEL3 = 1;  /* Use Mask 1 */

C1RXF3bits.SID = 0x100;   /* Filter 3 SID */
C1RXF3bits.EXID = 0;      /* Filter only SID messages */

C1RXM1bits.SID = 0x7FC;   /* Ignore last 2 bits in comparison */
C1RXM1bits.MIDE = 1;      /* Match only message types. */

C1FLTCON0bits.FLTEN3 = 1; /* Enable the filter */

/* Filter is now configured. */
```

要为扩展 ID 的 CAN 报文配置过滤器，应执行以下步骤：

1. 使用所需 CAN 报文的 SID 更新 CiRXFn 寄存器（n 代表所需的过滤器）的 SID<10:0> 位。使用所需 CAN 报文的 EID 更新 CiRXFn 寄存器的 EID<17:0> 位。
2. 将 CiRXFn 寄存器中的 EXID 位置 1。这会导致过滤器仅匹配扩展 ID 报文。
3. 创建过滤器的屏蔽器。在 CiRXMn 寄存器中装入相应值。将 MIDE 位置 1，以仅匹配扩展 ID 报文。
4. 确定控制选定过滤器的 CiFLTCONn 寄存器。设置 FSELn 位的值，使之指向接收 FIFO。设置 MSELn 位，以使用在步骤 3 中配置的屏蔽器。
5. 通过将 CiFLTCONn 寄存器中的 FLTENn 位置 1 来使能过滤器。

例 34-12 中的代码给出了为扩展 ID 配置 CAN 模块报文接收过滤器的示例。

**例 34-12： 配置过滤器来匹配 EID CAN 报文**

```
/* This code example shows how to configure a filter to match an */
/* Extended Identifier CAN message. */

/* In this example, Filter 3 is set up to accept messages with SID = 0x53 */
/* and EID = 0x1C498.Accepted message will be stored in FIFO5.Mask 1 is */
/* used. */

C1FLTCON0bits.FSEL3 = 5;       /* Store messages in FIFO5 */
C1FLTCON0bits.MSEL3 = 1;       /* Use Mask 1 */

C1RXF3bits.SID = 0x100;        /* Filter 3 SID */
C1RXF3bits.EID = 0x1C498;      /* Filter 3 EID */
C1RXF3bits.EXID = 1;           /* Filter only SID messages */

C1RXM1bits.SID = 0x7FF;        /* Consider all bits in */
C1RXM1bits.EID = 0x3FFFF;      /* in the filter comparison. */
C1RXM1bits.MIDE = 1;           /* Match only message types. */

C1FLTCON0bits.FLTEN3 = 1;      /* Enable the filter */

/* Filter is now configured. */
```

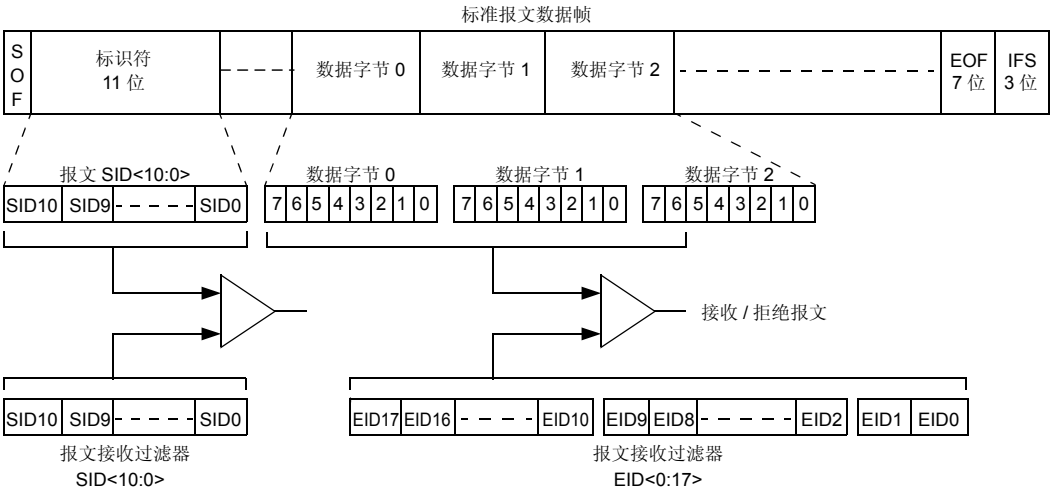
### 34.8.7 DeviceNet™ 过滤

DeviceNet 过滤功能基于 CAN 2.0A 协议，在该协议中，除了标准标识符 （SID）之外，还可以将最多 18 位数据字段与报文接收过滤器的扩展标识符 （EID）进行比较。

DeviceNet 功能通过 CAN 控制寄存器中的 DeviceNet 过滤器位编号控制位 DNCNT<4:0>（CiCON<4:0>）进行使能或禁止。在 DNCNT 位域中指定的值决定用于与报文接收过滤器的 EID 位进行比较的数据位的编号。如果 CiCON（DNCNT<4:0>）位清零，则 DeviceNet 功能被禁止。

要使报文被接收， 11 位 SID 必须与报文接收过滤器中的 SID<10:0> 位匹配，并且报文中的前“n”个数据位应与报文接收过滤器中的 EID<17:0> 位匹配。例如，如图 34-25 所示，接收报文数据有效载荷的前 18 个数据位与报文接收过滤器的相应扩展标识符位（CiRXFn<17:0>）进行比较。接收报文的 IDE 位必须为 0。

**图 34-25： 使用 DeviceNet™ 过滤的 CAN 操作**




**注：** 对于 EID 位，所显示的 DeviceNet™ 过滤配置为 DNCNT<4:0> = 10010。

#### 34.8.7.1 过滤器比较

表 34-4 显示了通过 CiCON（DNCNT<4:0>）控制位配置的过滤器比较。例如，如果 DNCNT<4:0> = 00011，则只有 11 位标准标识符与 SID 接收过滤器（SID<10:0>）匹配，且数据字节 0 的 bit 7、bit 6 和 bit 5 与扩展标识符过滤器 （EID<0:2>）匹配的报文才会被接收。

**表 34-4： DeviceNet™ 过滤器位配置**

| DeviceNet™ 过滤器配置（DNCNT<4:0>） | 要进行比较的接收报文数据位（字节 < 位 >） | 用于接收过滤器的 EID 位 |
|---|---|---|
| 00000 | 不比较 | 不比较 |
| 00001 | 数据字节 0<7> | EID<17> |
| 00010 | 数据字节 0<7:6> | EID<17:16> |
| 00011 | 数据字节 0<7:5> | EID<17:15> |
| 00100 | 数据字节 0<7:4> | EID<17:14> |
| 00101 | 数据字节 0<7:3> | EID<17:13> |
| 00110 | 数据字节 0<7:2> | EID<17:12> |
| 00111 | 数据字节 0<7:1> | EID<17:11> |
| 01000 | 数据字节 0<7:0> | EID<17:10> |
| 01001 | 数据字节 0<7:0> 和数据字节 1<7> | EID<17:9> |
| 01010 | 数据字节 0<7:0> 和数据字节 1<7:6> | EID<17:8> |
| 01011 | 数据字节 0<7:0> 和数据字节 1<7:5> | EID<17:7> |
| 01100 | 数据字节 0<7:0> 和数据字节 1<7:4> | EID<17:6> |
| 01101 | 数据字节 0<7:0> 和数据字节 1<7:3> | EID<17:5> |
| 01110 | 数据字节 0<7:0> 和数据字节 1<7:2> | EID<17:4> |
| 01111 | 数据字节 0<7:0> 和数据字节 1<7:1> | EID<17:3> |
| 10000 | 数据字节 0<7:0> 和数据字节 1<7:0> | EID<17:2> |
| 10001 | 字节 0<7:0>、字节 1<7:0> 和字节 2<7> | EID<17:1> |
| 10010 | 字节 0<7:0>、字节 1<7:0> 和字节 2<7:6> | EID<17:0> |
| 10011 至 11111 | 无效选择 | 无效选择 |

#### 34.8.7.2 特殊情形

可能会存在一些特殊情形，报文中包含的数据位少于 DeviceNet 过滤器配置所要求的位数。

- **情形 1**——如果 DNCNT <4:0> 大于 18，指示用户应用程序选择的位数大于 EID 位总数，则过滤器比较在数据的第 18 位 （数据字节 2 的 bit 6）处结束。如果 SID 和全部 18 个数据位都匹配，则接收报文。
- **情形 2**——如果 DNCNT<4:0> 大于 16，且接收报文数据长度编码 （DLC）为 2 （指示有效载荷为两个数据字节），则过滤器比较在数据的第 16 位 （数据字节 1 的 bit 0）处结束。如果 SID 和全部 16 个位都匹配，则接收报文。
- **情形 3**——如果 DNCNT<4:0> 大于 8，且接收报文的 DLC = 1 （指示有效载荷为一个数据字节），则过滤器比较在数据的第 8 位 （数据字节 0 的 bit 0）处结束。如果 SID 和全部 8 个位都匹配，则接收报文。
- **情形 4**——如果 DNCNT<4:0> 大于 0，且接收报文的 DLC = 0 （指示无数据有效载荷），则过滤器比较在标准标识符处结束。如果 SID 匹配，则接收报文。

## 34.9 接收 CAN 报文

CAN 模块会不断地监视 CAN 总线上的报文。在 CAN 模块接收到报文时，会将报文标识符与当前配置的过滤器 / 屏蔽器组合进行比较。如果发生匹配，模块会将报文存储到 FSELn 指向的 FIFO 中。接收报文并存储到 FIFO 中之后，以下位会发生更新：

- CiFIFOCIn 寄存器中的 CFIFOCI<4:0> 位会发生更新
- CiFIFOINTn 寄存器中的接收 FIFO 溢出中断标志 （RXOVFLIF）、接收 FIFO 已满中断标志 （RXFULLIF）、接收 FIFO 非空中断标志 （RXNEMPTYIF）和接收 FIFO 半满中断标志 （RXHALFIF）将进行相应更新
- CiFSTAT 寄存器中的 FIFO 中断待处理标志 （FIFOIPn）将进行相应更新
- 如果允许中断，将产生中断

CAN 中断编码 （CiVEC）寄存器中的 CAN 模块中断编码位 ICOD<6:0>（CiVEC<6:0>）和选中过滤器的编号位 FILHIT<4:0> （CiVEC<12:8>）也会发生更新。

**注：** 为了接收报文，应用程序必须至少使能一个报文接收过滤器和一个屏蔽器寄存器。

接收的 CAN 报文将使用表 34-5 中所示的格式存储到报文缓冲区中。CAN 模块会使用图 34-26 至图 34-29 所示的格式。

**表 34-5： RAM 中存储的接收报文格式——CiCON.CANCAP = 1， CFIFOCON.DONLY = 0**

| 地址偏移 | 名称 | | Bit 31/2315/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CMSGSID | 31:24 | CMSGTS<15:8> | | | | | | | |
| | | 23:16 | CMSGTS<7:0> | | | | | | | |
| | | 15:8 | FILHIT<4:0> | | | | | SID<10:8> | | |
| | | 7:0 | SID<7:0> | | | | | | | |
| 0x04 | CMSGEID | 31:24 | -— | -— | SRR | IDE | EID<17:14> | | | |
| | | 23:16 | EID<13:6> | | | | | | | |
| | | 15:8 | EID<5:0> | | | | | | RTR | RB1 |
| | | 7:0 | -— | -— | --— | RB0 | DLC<3:0> | | | |
| 0x08 | CMSGDATA0 | 31:24 | 接收缓冲区数据字节 3 | | | | | | | |
| | | 23:16 | 接收缓冲区数据字节 2 | | | | | | | |
| | | 15:8 | 接收缓冲区数据字节 1 | | | | | | | |
| | | 7:0 | 接收缓冲区数据字节 0 | | | | | | | |
| 0x0C | CMSGDATA1 | 31:24 | 接收缓冲区数据字节 7 | | | | | | | |
| | | 23:16 | 接收缓冲区数据字节 6 | | | | | | | |
| | | 15:8 | 接收缓冲区数据字节 5 | | | | | | | |
| | | 7:0 | 接收缓冲区数据字节 4 | | | | | | | |

**图注：** 阴影位读为 0。

**注 1：** CAN 接收报文存储在系统 RAM 中，没有与之关联的置 1/ 清零 / 取反寄存器。

**图 34-26： CMSGSID 的格式**

| CMSGTS<15:0> | FILHIT<4:0> | SID<10:0> |
|---|---|---|

bit 31-16 **CMSGTS<15:0>**：CAN 报文时间标记
成功接收到报文时的 CANTMR 值。如果 CANCAP = 0，这些位将读为 0。

bit 15-11 **FILHIT<4:0>**：选中过滤器的编码
对导致写入该缓冲区的过滤器的编号进行编码。
11111 = 通过过滤器 31 接收报文
•
•
•
00000 = 通过过滤器 0 接收报文

bit 10-0 **SID<10:0>**：接收报文的标准标识符

**图 34-27： CMSGEID 的格式**

| — | SRR | IDE | EID<17:0> | RTR | RB1 | — | RB0 | DLC<3:0> |
|---|---|---|---|---|---|---|---|---|

bit 31-30 **未实现**：读为 0

bit 29 **SRR**：替代远程请求
如果是标准报文帧 （IDE = 0），则该位是无关位。
如果是扩展报文帧 （IDE = 1），则该位总是为 1。

bit 28 **IDE**：扩展标识符
1 = 接收到的报文具有扩展标识符
0 = 接收到的报文具有标准标识符

bit 27-10 **EID<17:0>**：扩展标识符

bit 9 **RTR**：远程发送请求
1 = 报文发送方请求远程发送
0 = 接收到正常报文

bit 8 **RB1**：保留的 bit 1
根据 CAN 总线规范，用户应用程序必须将该位设置为 0。

bit 7-5 **未实现**：读为 0

bit 4 **RB0**：保留的 bit 0
根据 CAN 总线规范，用户应用程序必须将该位设置为 0。

bit 3-0 **DLC<3:0>**：数据长度编码
1xxx = 接收到的报文包含数据的 8 字节
0111 = 接收到的报文包含数据的 7 字节
•
•
•
0001 = 接收到的报文包含数据的 1 字节
0000 = 接收到的报文包含数据的 0 字节

**注 1**：该寄存器位于 CAN 模块外的系统存储器中。

**图 34-28： CMSGDATA0 的格式**

| DATA3<7:0> | DATA2<7:0> | DATA1<7:0> | DATA0<7:0> |
|---|---|---|---|

bit 31-24 **DATA3<7:0>：**数据字节 3
bit 23-16 **DATA2<7:0>：**数据字节 2
bit 15-8 **DATA1<7:0>：**数据字节 1
bit 7-0 **DATA0<7:0>：**数据字节 0

**注 1：**该寄存器位于 CAN 模块外的系统存储器中。

**图 34-29： CMSGDATA1 的格式**

| DATA7<7:0> | DATA6<7:0> | DATA5<7:0> | DATA4<7:0> |
|---|---|---|---|

bit 31-24 **DATA7<7:0>：**数据字节 7
bit 23-16 **DATA6<7:0>：**数据字节 6
bit 15-8 **DATA5<7:0>：**数据字节 5
bit 7-0 **DATA4<7:0>：**数据字节 4

**注 1：**该寄存器位于 CAN 模块外的系统存储器中。

例 34-13 中的代码给出了实现 CAN 完整接收报文缓冲区的示例数据结构。例 34-14 给出了一个使用该结构的示例。

**例 34-13： 在存储器中实现 CAN 完整接收报文缓冲区**

```
/* This code snippet shows an example of data structure to implement a CAN */
/* full receive message buffer.*/

/* Define the sub-components of the data structure as specified in Table 34-5 */

/* Create a CMSGSID data type. */
typedefstruct
{
    unsigned SID:11;
    unsigned FILHIT:5;
    unsigned CMSGTS:16;
}rxcmsgsid;

/* Create a CMSGEID data type. */
typedef struct
{
    unsigned DLC:4;
    unsigned RB0:1;
    unsigned :3;
    unsigned RB1:1;
    unsigned RTR:1;
    unsigned EID:18;
    unsigned IDE:1;
    unsigned SRR:1;
    unsigned :2;
}rxcmsgeid;

/* Create a CMSGDATA0 data type. */
typedef struct
{
    unsigned Byte0:8;
    unsigned Byte1:8;
    unsigned Byte2:8;
    unsigned Byte3:8;
}rxcmsgdata0;

/* Create a CMSGDATA1 data type. */
typedefstruct
{
    unsigned Byte4:8;
    unsigned Byte5:8;
    unsigned Byte6:8;
    unsigned Byte7:8;
}rxcmsgdata1;

/* This is the main data structure. */

typedef union uCANRxMessageBuffer {

    struct
    {
        rxcmsgsid CMSGSID;
        rxcmsgeid CMSGEID;
        rxcmsgdata0 CMSGDATA0;
        rxcmsgdata0 CMSGDATA1;
    };
    int messageWord[4];
}CANRxMessageBuffer;
```

**例 34-14： 数据结构的示例用法**

```
/* Example usage of code provided in Example 34-13. */

CANRxMessageBuffer * buffer;

/* When a message have been received and read, the individual fields of */
/* the received message can be queried as such. */

buffer = (CANRxMessageBuffer *)(PA_TO_KVA1(C1FIFOUA1));

if(buffer->CMSGEID.DLC == 4)
{
    /* If the length of the received message is 4 then do something. */
}
```

### 34.9.1 仅数据的接收报文

PIC32 CAN 模块提供了一种特殊的接收模式，在该模式下，只会在报文缓冲区中存储接收报文的数据有效载荷部分。模块会丢弃标识符、保留位和 DLC 位。即使使能了时间标记，也不会存储时间标记值。该模式可以通过将 DONLY 位 （CiFIFOCONn<12>）置 1 来使能。

请注意，无论 CAN 报文中 DLC 字段的值如何，模块都会存储 8 字节的数据。未用字节会填入 0x00。该模式的一种可能用途是连接数据跨越多个报文的报文。例如，通过 CAN 总线发送一个字符串。

**表 34-6： RAM 中存储的仅数据接收报文格式**

| 地址偏移 | 名称 | | Bit 31/2315/7 | Bit 30/22/14/6 | Bit 29/21/13/5 | Bit 28/20/12/4 | Bit 27/19/11/3 | Bit 26/18/10/2 | Bit 25/17/9/1 | Bit 24/16/8/0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | CMSG0DATA0 | 31:24 | 接收缓冲区数据字节 3 | | | | | | | |
| | | 23:16 | 接收缓冲区数据字节 2 | | | | | | | |
| | | 15:8 | 接收缓冲区数据字节 1 | | | | | | | |
| | | 7:0 | 接收缓冲区数据字节 0 | | | | | | | |
| 0x04 | CMSG0DATA1 | 31:24 | 接收缓冲区数据字节 7 | | | | | | | |
| | | 23:16 | 接收缓冲区数据字节 6 | | | | | | | |
| | | 15:8 | 接收缓冲区数据字节 5 | | | | | | | |
| | | 7:0 | 接收缓冲区数据字节 4 | | | | | | | |

**注 1：** 报文被存储在系统存储器中。
**2：** 因此，不存在与这些寄存器关联的置 1/ 清零 / 取反寄存器。

### 34.9.2 处理接收到的报文

用户应用程序可以使用查询方法或使用 CAN 模块中断来跟踪报文接收。CAN 模块提供了关于不同接收 FIFO 事件的通知。应用程序可以在接收 FIFO 非空、半满或全满时接收到通知。应用程序应当经常读取接收 FIFO，以确保不会发生 FIFO 溢出。无论使用何种技术（查询或中断），以下步骤都可用于从 FIFO 中读取报文：

1. 读取 CiFIFOUAn 寄存器的值。这可以提供指向应用程序应读取的接收报文缓冲区的 32 位物理地址指针。
2. 对于仅数据接收报文类型，需要处理报文缓冲区中的两个字。
3. 对于完全接收报文类型，需要处理报文缓冲区中的 4 个字。
4. 处理报文缓冲区之后，将关联的 CiFIFOCONn 寄存器中的 UINC 位置 1。

例 34-15 中的代码给出了复制 CAN 报文的示例。

**例 34-15： 从接收 FIFO 中复制报文**

```
/* This code example shows how to process a message from a receive FIFO. */
/* In this example CAN1 and FIF01 are used. */

CANRxMessageBuffer rxMessage;
unsigned int * bufferToRead;

/* Check if there is a message available to read. */
if(C1FIFOINT1bits.RXNEMPTYIF == 1)
{
    /* Get the address of the buffer to read */
    bufferToRead = PA_TO_KVA1(C1FIFOUA1);

    /* This is an example process (Copying the buffer).The application */
    /* could process this buffer in place. */
        bufferToRead[0] = rxMessage->messageWord[0];
        bufferToRead[1] = rxMessage->messageWord[1];
        bufferToRead[2] = rxMessage->messageWord[2];
        bufferToRead[3] = rxMessage->messageWord[3];

    /* Update the message buffer pointer. */
    C1FIFOCON1bits.UINC = 1;
}
```

### 34.9.3 接收 FIFO 行为示例

假设有一个应用程序，它将 CAN1 模块的 FIFO0 配置为接收报文 FIFO。图 34-30 显示了在第一个报文到达并放入 FIFO 之前的 FIFO 状态。CAN FIFO 基址寄存器 C1FIFOBA 设置为 0x00000100。接收到的报文将被放入系统 RAM（物理地址 0x00000100）。可以注意到，CFIFOUA 最初指向 FIFO 的起始位置。该示例会跟踪接收 FIFO 溢出中断标志 （RXOVFLIF）、接收 FIFO 已满中断标志 （RXFULLIF）、接收 FIFO 半满中断标志 （RXHALFIF）和接收 FIFO 非空中断标志 （RXNEMPTYIF）。图 34-30 中显示了该配置。

**图 34-30： 开始时的 FIFO**

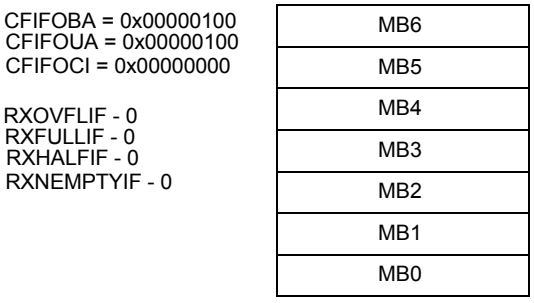


图 34-31 显示了在接收到第一个报文并写入 FIFO 之后的 FIFO 状态。请注意，虽然 FIFO 中的第一个报文缓冲区 （MB0）包含了数据，但 CFIFOUA 仍然指向 FIFO 的基址，因为用户尚未读取该位置。请注意， CAN 索引 （CFIFOCI）已经递增，指示下一个接收报文将被放入 MB1。RXNEMPTYIF 标志已置 1，指示 FIFO 非空。

**图 34-31： FIFO——第 1 次写入**

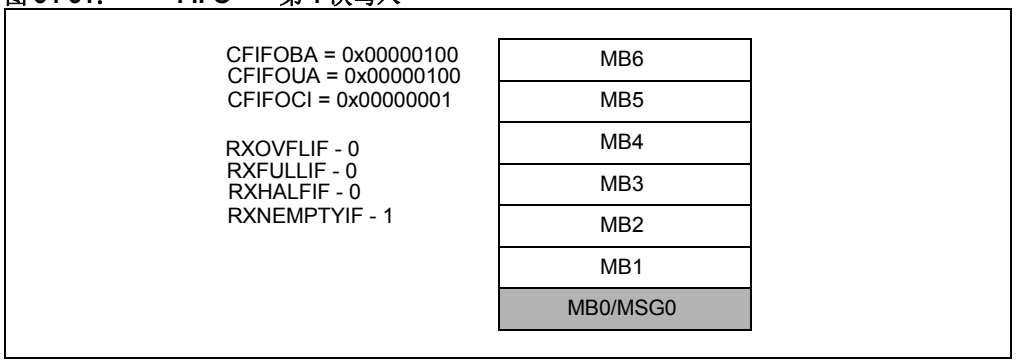



图 34-32 显示了又接收到 5 个报文之后的 FIFO 状态。与前面一样， CFIFOUA 尚未被修改。在接收第 4 个报文之后，缓冲区半满标志 （RXHALFIF）会置 1。

**图 34-32： FIFO——第 6 次写入**

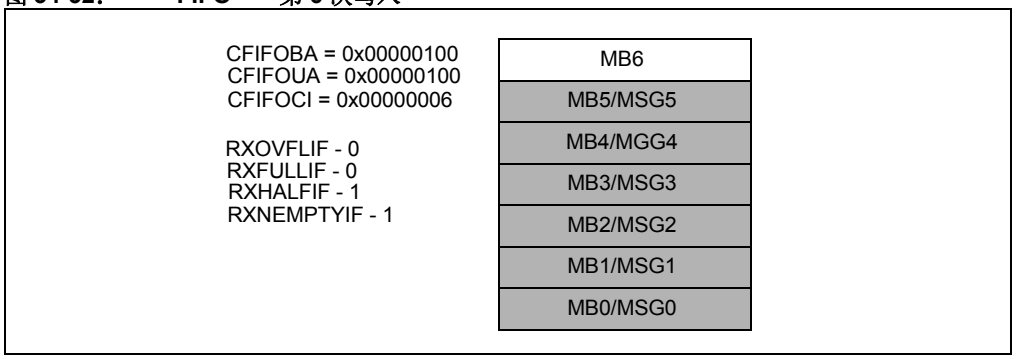



图 34-33 显示了从 MB0 中读取第一个报文之后的 FIFO 状态。在应用程序读取该报文之后，应将 CiFIFOCONn 寄存器中的 UINC 位置 1。这会导致 CFIFOUA 寄存器更改为 0x0000110（这是应用程序必须读取的下一个报文缓冲区的地址）。

**图 34-33： FIFO——第 1 次读取**

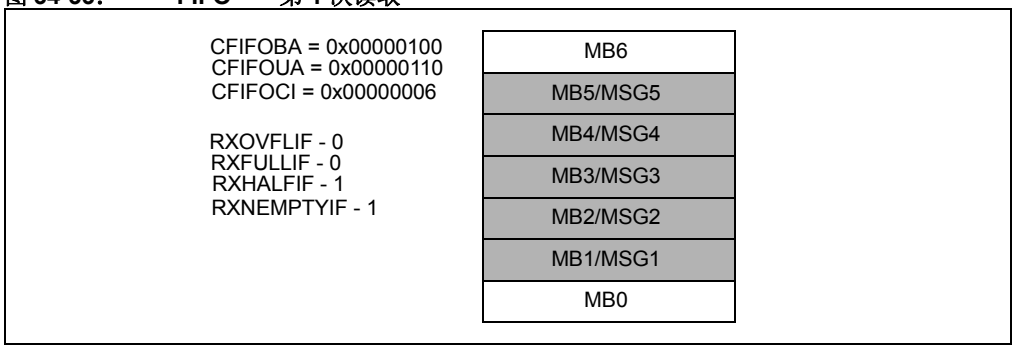


图 34-34 显示了接收到第 7 个报文之后的 FIFO 状态。

**图 34-34： FIFO——第 7 次写入 （即将填满）**

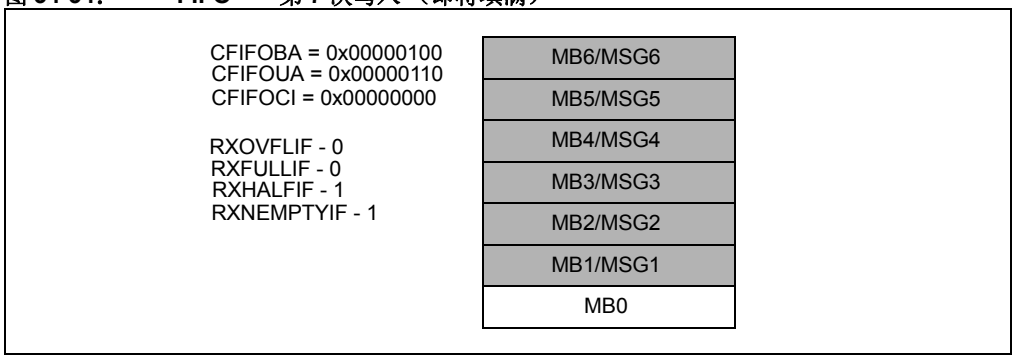



图 34-35 显示了处于填满状态的 FIFO。可以注意到， RXFULLIF 标志已置 1。

**图 34-35： FIFO——第 8 次写入 （FIFO 已满）**

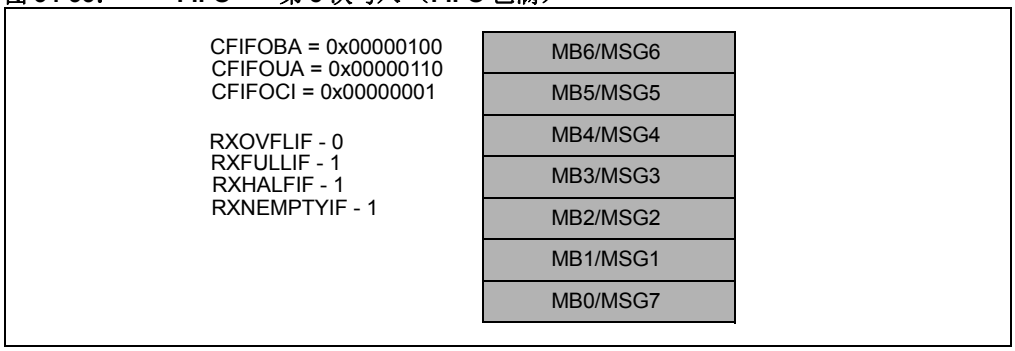
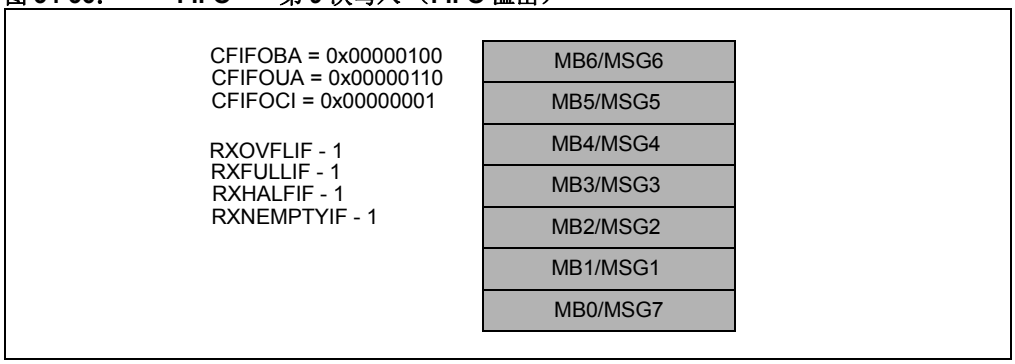



图 34-36 显示了在 FIFO 填满时 FIFO 又接收到一个报文，使 FIFO 发生溢出。可以注意到，RXOVFLIF 标志已置 1。请注意，CAN 索引（CiCIFOCI）未发生移动，报文（MSG8）已丢失。

**图 34-36： FIFO——第 9 次写入 （FIFO 溢出）**



## 34.10 位时序

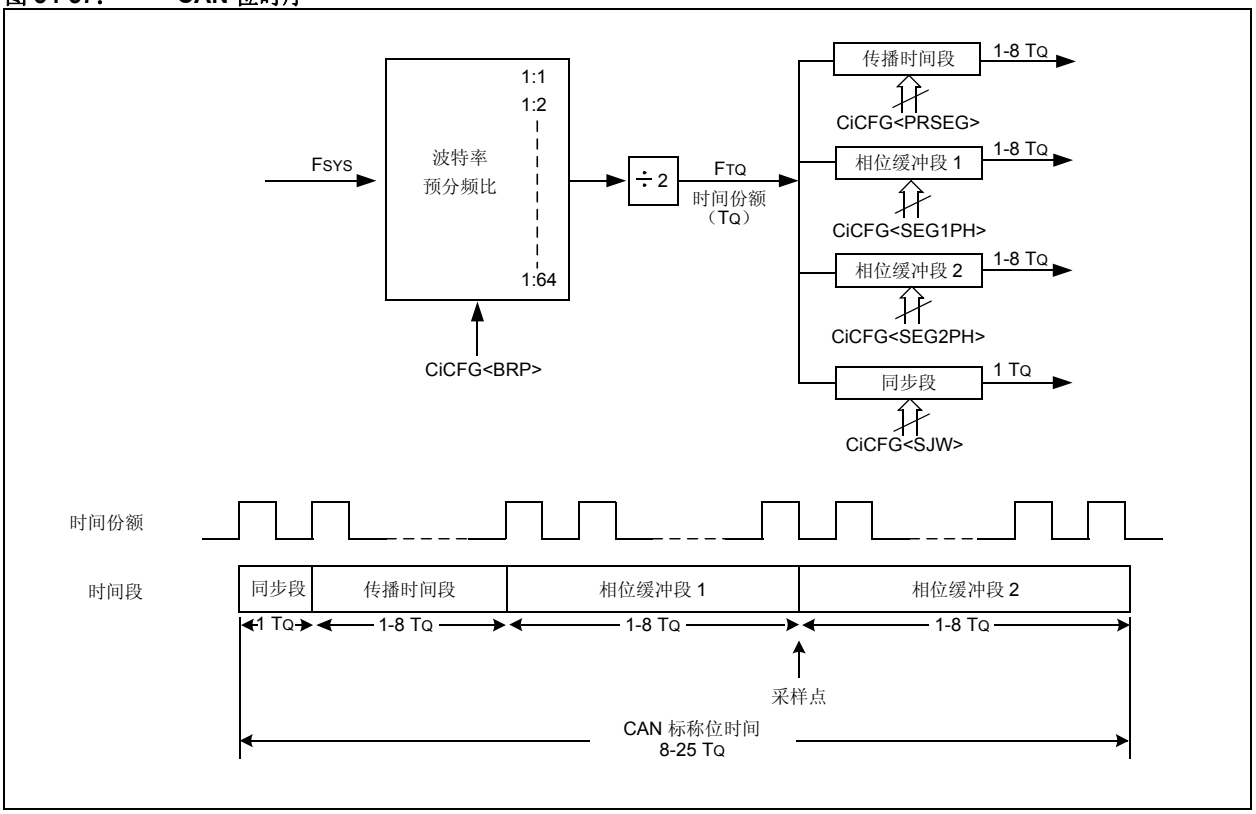
标称比特率是指每秒在 CAN 总线上发送的位数。

标称位时间 = 1 ÷ 标称比特率。

在一个位时间中共有 4 个时间段，用以补偿由于振荡器漂移或传播延时引起的任何相移。这些时间段彼此不重叠，以时间份额 （Time Quantum， TQ）来表示。一个 TQ 是基于振荡器时钟产生的固定时间单位。标称位时间的时间份额总数必须设定为 8 至 25 个 TQ 之间。

图 34-37 显示了如何基于系统时钟得到时间份额时钟 （FTQ），以及如何设定不同的时间段。

**图 34-37： CAN 位时序**




### 34.10.1 位时间段

每个位发送时间由 4 个时间段组成。

- **同步段**——该时间段用于同步与 CAN 总线连接的不同节点。位边沿需要处于该时间段内。根据 CAN 协议，同步段假设为一个时间份额。
- **传播时间段**——该时间段用于补偿由于总线线路或由于与该总线连接的各种收发器而可能导致的任何延时。
- **相位缓冲段 1**——该时间段用于补偿由于边沿中的相移而可能产生的误差。该时间段可以在重新同步期间延长，以补偿相移。
- **相位缓冲段 2**——该时间段用于补偿由于边沿中的相移而可能产生的误差。该时间段可以在重新同步期间缩短，以补偿相移。相位缓冲段 2 时间可以配置为可编程或由相位缓冲段 1 时间指定。

### 34.10.2 采样点

采样点是指在 CAN 位时间间隔中获取采样并读取和解释总线状态的时间点。它介于相位缓冲段 1 和相位缓冲段 2 之间。CAN 总线可以在采样点采样一次或三次，这由 CAN 波特率配置寄存器中的采样 CAN 总线线路位 SAM （CiCFG<14>）配置。

- 如果 CiCFG<SAM> = 1，则在采样点对 CAN 总线采样三次。三次采样中出现次数最多的采样值将决定位值。
- 如果 CiCFG<SAM> = 0，则在采样点仅对 CAN 总线采样一次。

### 34.10.3 同步

使用的同步方式有两种——硬同步和重新同步。硬同步在帧起始时进行一次。重新同步在帧内进行。

- **硬同步**在启动位从隐性变为显性时进行。位时间从该边沿重新开始。
- **重新同步**在某个位边沿不是出现在报文的同步段内时进行。根据信号中的相位误差，有一个相位缓冲段会被缩短或延长一定时间量。可使用的最大时间量由同步跳转宽度参数（CiCFG<SJW>）决定。

相位缓冲段 1 和相位缓冲段 2 的长度可以根据发送和接收节点的振荡器容差进行更改。重新同步可以补偿由于发送和接收节点使用不同振荡器而可能产生的任何相移。

- **位延长**——如果 CAN 中的发送节点的振荡器速度比接收节点慢，则可以通过在位时间中延长相位缓冲段 1 来延迟下一个下降沿，从而延迟采样点。
- **位缩短**——如果 CAN 中的发送节点的振荡器速度比接收节点快，则可以通过在位时间中缩短相位缓冲段 2 来推前下一个下降沿，从而推前下一位的采样点。
- **同步跳转宽度 （SJW）**——CAN 波特率配置寄存器中的 SJW <1:0> 位 （CiCFG<7:6>）通过限制可应用于相位缓冲段 1 和相位缓冲段 2 时间间隔的延长或缩短时间量来决定同步跳转宽度。该时间段的时间不应大于相位缓冲段 2 时间。宽度可以是 1-4 TQ。

### 34.10.4 CAN 位时间计算

用户应用程序配置 CAN 模块的位时间需要执行的步骤使用如下示例进行说明：

#### 34.10.4.1 步骤 1：计算时间份额频率 （FTQ）

- 选择 CAN 总线的波特率 （FBAUD）
- 根据系统需求，选择位时间中的时间份额数。时间份额频率（FTQ）根据公式 34-1 计算。

**公式 34-1：**

$$FTQ = N * FBAUD$$

**注 1：** 标称位时间的时间份额总数必须设定在 8 TQ 和 25 TQ 之间。因而，时间份额频率（FTQ）介于 8-25 乘以波特率 （FBAUD）之间。

**2：** 请确保 FTQ 是 FBAUD 的整数倍，以获得精确的位时间。如果不是整数倍，则可能需要更改振荡器输入频率或波特率。

#### 34.10.4.2 步骤 2：计算波特率预分频比 （CiCFG<BRP>）

波特率预分频比根据公式 34-2 计算。

**公式 34-2：**

$$CiCFG<BRP> = (FSYS/(2 * FTQ)) - 1$$

#### 34.10.4.3 步骤 3：选择各个位时间段

各个位时间段使用 CiCFG 寄存器进行选择。
位时间 = 同步段 + 传播时间段 + 相位缓冲段 1 + 相位缓冲段 2

**注 1：**（传播时间段 + 相位缓冲段 1）必须大于或等于相位缓冲段 2 的长度。
**2：**相位缓冲段 2 必须大于同步跳转宽度。

**例 34-16： CAN 位时序计算示例**

**步骤 1：**计算时间份额频率。

- 如果 FBAUD = 1 Mbps，且时间份额数 N = 10，则 FTQ = 20 MHz。

**步骤 2：**计算波特率预分频比 （如果 FSYS = 80000000）。

- CiCFG1<BRP> = ( 80000000/(2 * FTQ) ) - 1 = 3。

**步骤 3：**选择各个位时间段。

- 同步段 = 1 TQ （常量）。
- 基于系统特性，假设传播延时 = 3 TQ。
- 假设采样点位于标称位时间的 70% 位置处。相位缓冲段 2 = 标称位时间的 30% = 3 TQ。
- 从而，相位缓冲段 1 = 10 TQ - (1 TQ + 3 TQ + 3 TQ) = 3 TQ。

例 34-17 给出了配置 CAN 位时序的代码示例。

**例 34-17： 配置 CAN 模块来获得特定比特率**

```
/* This code example shows how to configure the CAN module to obtain a */
/* specific bit rate. */

/* This example implements the configuration shown in Example 34-16. */

/* Fsys = System Clock Frequency = 80000000;     */
/* Fbaud = CAN bit rate = 1000000;               */
/* N = Time Quanta (Tq) per bit = 10;            */
/* Prop Segment = 3Tq                            */
/* Phase Seg 1 = 3Tq                             */
/* Phase Seg 2 = 3Tq                             */
/* Sync Jump Width = 2Tq                         */

/* Ensure the CAN module is in configuration mode.*/

C1CONbits.REQOP = 4
while(C1CONbits.OPMOD != 4);

C1CFGbits.SEG2PHTS = 1;/* Phase seg 2 is freely programmable */
C1CFGbits.SEG2PH = 2;/* Phase seg 2 is 3Tq. */
C1CFGbits.SEG1PH = 2;/* Phase seg 1 is 3Tq. */
C1CFGbits.PRSEG = 2;/* Propagation seg 2 is 3Tq. */
C1CFGbits.SAM = 1;/* Sample bit 3 times. */
C1CFGbits.SJW = 2;/* Sync jump width is 2 Tq */
C1CFGbits.BRPVAL = 3;/* BRP value as calculated in example 34-11 */

/* CAN bit rate configuration complete. */
```

## 34.11 CAN 错误管理

### 34.11.1 CAN 总线错误

CAN 协议定义了 5 种不同的错误检测方式。

- 位错误
- 应答错误
- 格式错误
- 填充错误
- CRC 错误

位错误和应答错误出现在位级别；其他三种错误出现在报文级别。

#### 34.11.1.1 位错误

在总线上发送一个位的节点同时也会监视总线。当监测到的位值不同于所发送位值时，即表示检测到位错误。但存在一种例外情况，即在仲裁字段的填充位流期间或在 ACK 时隙期间发送隐性位。在这种情况下，监测到显性位时不会出现位错误。发送被动错误帧并检测到显性位的发送器不会将这种情况解释为位错误。

#### 34.11.1.2 应答错误

在报文的应答字段中，发送器会检查应答时隙（发送器将它以隐性位发送）是否包含显性位。如果不包含，则表示其他节点没有正确接收到该帧。这表示发生了应答错误，因此，必须重复发送报文。这种情况下不会产生错误帧。

#### 34.11.1.3 格式错误

当固定格式的位字段 （帧结束、帧间间隔、应答定界符或 CRC 定界符）包含一个或多个非法位时，即表示检测到格式错误。对于接收器，帧结束的最后一位期间的显性位不视为格式错误。

#### 34.11.1.4 填充错误

在应按照位填充方法进行编码的报文字段中，在第 6 个连续相等位电平的位时间会检测到填充错误。

#### 34.11.1.5 CRC 错误

发送报文的节点需要计算并发送对应于所发送报文的 CRC。总线上的每个接收器都会执行与发送器相同的 CRC 计算。如果计算的结果与从接收报文中获得的 CRC 值不同，则表示检测到 CRC 错误。

### 34.11.2 错误限制

总线上的每个 CAN 控制器都会尝试检测每个报文中的上述错误。如果发现错误，则发现错误的节点会发送一个错误帧，从而破坏总线通信。其他节点会检测到由错误帧导致的错误（如果它们尚未检测到原始错误）并执行相应操作 （即，丢弃当前报文）。

CAN 模块会维持两个错误计数器：

- 发送错误计数器 （TEC）——CiTREC<15:8>
- 接收错误计数器 （REC）——CiTREC<7:0>

存在一些控制这些计数器递增和 / 或递减方式的规则。实质上，检测错误的发送器递增其发送错误计数器的速度会比监听节点递增其接收错误计数器的速度快。这是因为发送器出错的机率较大。

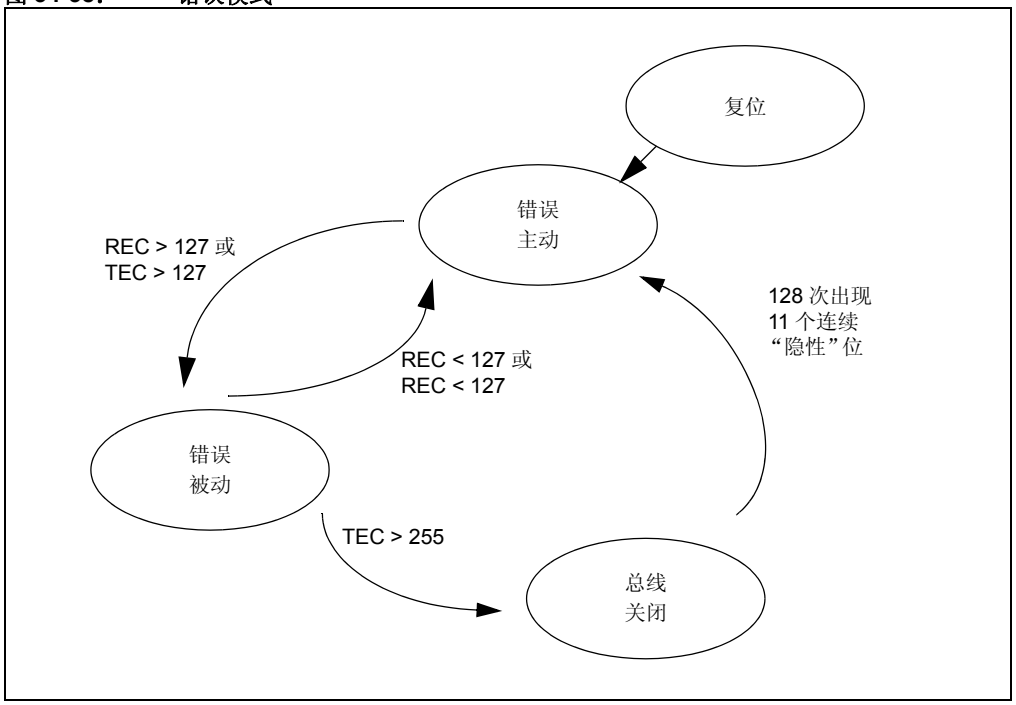
| **注：** | 错误计数器根据 CAN 2.0B 规范进行修改。 |
|---|---|

节点在开始时处于错误主动模式。当两个错误计数器值有任意一个大于等于 127 时，节点进入称为 “错误被动”的状态。当发送错误计数器的值超出 255 时，节点进入总线关闭状态。

- 错误主动节点在检测到错误时会发送主动错误帧。
- 错误被动节点在检测到错误时会发送被动错误帧。
- 处于总线关闭状态的节点不会在总线上发送任何数据。

此外，CAN 模块采用了错误警告功能，该功能会在节点进入错误被动状态之前警告用户应用程序（在发送错误计数器大于等于 96 时），如图 34-38 所示。

**图 34-38： 错误模式**




### 34.11.2.1 发送器处于错误被动状态

当发送错误计数器大于等于 128 时，发送器错误被动 （TXBP）位 （CiTREC<20>）会置 1，并在进入错误被动状态时产生错误中断 （CiINT<CERRIF>）。如果发送错误计数器变为小于 128，则硬件会自动将发送错误被动标志清零。

### 34.11.2.2 接收器处于错误被动状态

当接收错误计数器大于等于 128 时，接收器错误被动 （RXBP）位 （CiTREC<19>）会置 1，并在进入错误被动状态时产生错误中断 （CiINT<CERRIF>）。如果接收错误计数器变为小于 128，则硬件会自动将接收错误被动标志清零。

### 34.11.2.3 发送器处于总线关闭状态

当发送错误计数器大于等于 256 时，发送器总线关闭 （TXBO）位 （CiTREC<21>）会置 1，并产生错误中断 （CiINT<CERRIF>）。

### 34.11.2.4 发送器处于错误警告状态

当发送错误计数器大于等于 96 时，发送器错误警告 （TXWARN）位 （CiTREC<18>）会置 1，并在进入错误警告状态时产生错误中断（CiINT<CERRIF>）。如果发送错误计数器变为小于 96，则硬件会自动将发送错误警告标志清零。

#### 34.11.2.5 接收器处于错误警告状态

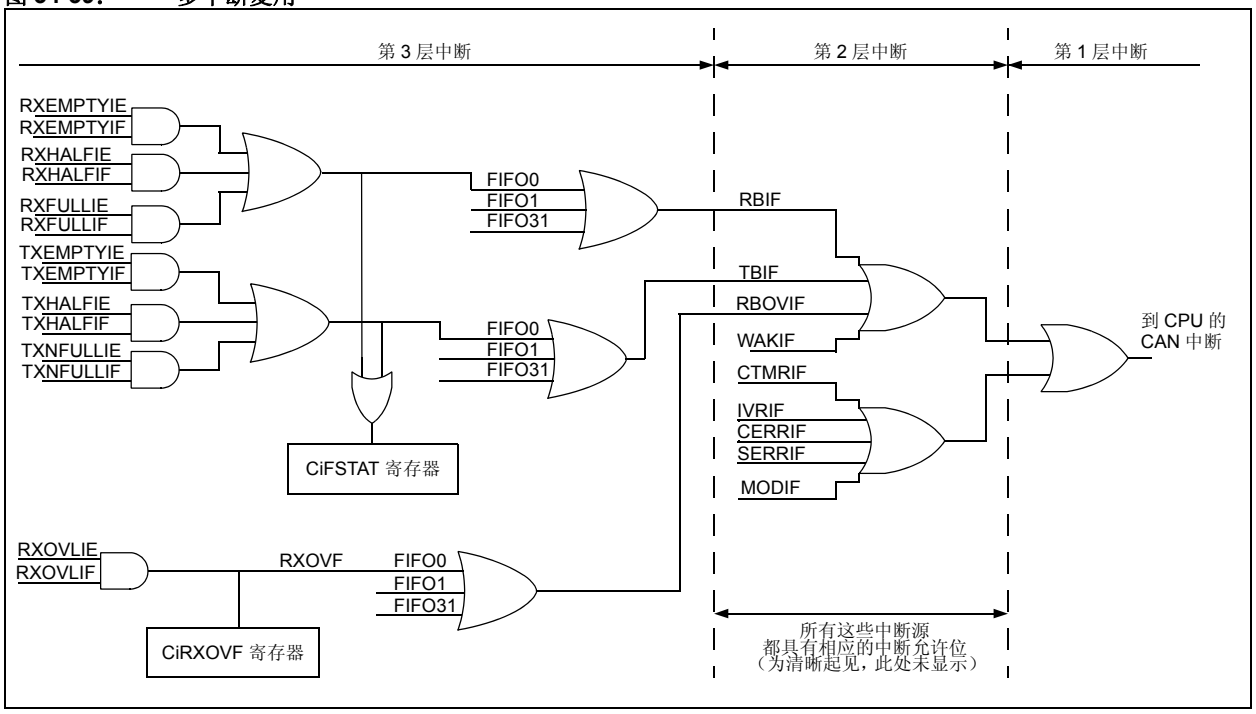
当接收错误计数器大于等于 96 时，接收器错误警告 （RXWARN）位 （CiTREC<17>）会置 1，并在进入错误警告状态时产生错误中断（CiINT<CERRIF>）。如果接收错误计数器变为小于 96，则硬件会自动将接收错误警告标志清零。

此外，还有一个错误状态警告标志 （EWARN）位 （CiTREC<16>），如果至少有一个错误计数器大于等于错误警告限制值 96，则该位会置 1。如果两个错误计数器都小于错误警告限制值，则 EWARN 会复位。

## 34.12 CAN 中断

PIC32 CAN 模块中的中断可以分为三层。较低层中断会被传递到较高层，它们在较高层复用为单个中断。因此，第 1 层中断是第 2 层中多个中断的复用结果。一些第 2 层中断是第 3 层中断的复用结果 （见图 34-39）。

**图 34-39： 多中断复用**




在第 1 层， CAN 模块会产生一个 CPU 中断，称为 CANx 中断。中断由第 2 层中任意中断产生。CANx 中断可以通过 PIC32 中断控制器进行允许或禁止。关于在 PIC32 器件上允许和配置 CPU 中断的更多详细信息，请参见 《PIC32MX 系列参考手册》中的**第 8 章 “中断”**（DS61108）。

每个第 2 层和第 3 层中断都具有与之关联的中断允许位 （IE）和中断状态标志位 （IF）。在发生中断事件时， CAN 模块会将 IF 位置 1。如果应用程序已将 IE 位置 1，则事件会被传播到下一个中断层。应用程序可以通过查询 IF 位或允许中断的方式来检测 CAN 模块事件。

中断编码（ICOD<6:0>）位将包含最新中断的编码。这些位可以与跳转表联合使用，用于高效地处理中断。

### 34.12.1 第 2 层中断

CAN 模块在第 2 层具有 9 个中断。

- TBIF——发送 FIFO 中断
- RBIF——接收 FIFO 中断
- CTMRIF——CAN 时间标记定时器计满返回中断
- MODIF——CAN 工作模式改变中断
- RBOVIF——接收 FIFO 溢出中断
- SERRIF——CAN 系统错误中断
- CERRIF——CAN 总线错误中断
- WAKIF——CAN 唤醒中断
- IVRIF——无效 CAN 报文中断

在以上中断中，收到无效报文 （IVRIF）、唤醒 （WAKIF）、 CAN 定时器 （CTMRIF）和 CAN 模式改变（MODIF）中断具有单个中断源。FIFO（TBIF/RBIF）、FIFO 溢出（RBOVIF）、CAN 总线错误 （CERRIF）和系统错误 （SERRIF）中断具有多个中断源。

#### 34.12.1.1 收到无效报文中断 （IVRIF）

在上一个发送或接收报文中发生某种错误时，会发生收到无效报文中断 （IVRIF） 。发生的特定错误对于应用程序软件不可用。要清除中断，必须将标志位清零。

#### 34.12.1.2 唤醒中断 （WAKIF）

在 CAN 模块处于休眠模式时，如果检测到 CAN 总线活动，则会发生唤醒中断（WAKIF）。要清除中断，必须将标志位清零。

#### 34.12.1.3 CAN 总线错误中断 （CERRIF）

在 CAN 总线上出现错误时，会发生 CAN 总线错误中断 （CERRIF）。该位在每次 CAN 模块的当前错误状态发生与 CAN 网络有关的变化时置 1。错误状态通过 CiTREC 寄存器进行跟踪。要清除中断，必须将标志位清零。每次发送错误计数器 （TEC<7:0>）位或接收错误计数器（REC<7:0>）位跨过阈值时， CERRIF 位会置 1。例如，如果 TEC<7:0> 从 95 变为 96，则 CERRIF 会置1。如果清零该位，则即使TEC<7:0>保持大于96，它也会保持清零。如果TEC<7:0> 降至小于 96 并再次升到大于 95，或者如果 TEC<7:0> 升到大于 127，它会再次置 1。

#### 34.12.1.4 系统错误中断 （SERRIF）

系统错误中断 （SERRIF）会由于两种系统错误 （寻址错误和带宽错误）而发生。这两种错误都是严重错误， CAN 模块会被停止。

发生错误之后，用户应通过查询 CiCON 寄存器中的 BUSY 位来等待 CAN 模块停止。在模块停止之后，可以通过清零 CiCON 寄存器中的 ON 位来清零 SERRIF。

| | |
|---|---|
| **注：** | 导致系统错误中断 （SERRIF）的错误条件只能通过关闭 CAN 模块 （通过清零 CiCON 寄存器中的 ON 位）来解除。清零 SERRIF 位并不会清除该错误条件。 |

#### 34.12.1.5 寻址错误

寻址错误有两种来源。它们具有相同的 ICOD 值 （ICOD = 100 0100）。

- 如果 FIFO 配置了无效的基址，则会发生寻址错误。最常出现该错误的情形是 FIFO 指向未实现的地址。
- 如果报文目标非法，则会发生寻址错误；例如，尝试将接收报文写入不可直接写入的程序闪存。

#### 34.12.1.6 总线带宽错误

如果 CAN 模块无法在下一个 CAN 报文到达之前将接收到的 CAN 报文写入系统存储器中的位置，则会发生总线带宽错误。该条件通过 ICOD = 100 0101 表示。如果某个优先级高于 CAN 模块的其他 PIC32 系统总线发起器长时间占用系统总线，从而导致 CAN 模块无法存储接收到的 CAN 报文，则可能发生该错误。相应 FIFO 中断寄存器中的溢出位会置 1。

#### 34.12.1.7 接收 FIFO 溢出中断 （RBOVIF）

在 CAN 模块接收到报文，但指定的接收 FIFO 已满时，会发生接收 FIFO 溢出中断。 CAN 模块会将对应于受影响的接收FIFO的CiFIFOINTn寄存器中的RXOVFLIF标志置1。如果CiFIFOINTn寄存器中的接收 FIFO 溢出中断允许 （RXOVFLIE）位置 1，则 CAN 接收 FIFO 溢出状态（CiRXOVF）寄存器中也会反映该状态。

CiINT 寄存器中的 RBOVIF 位是 CiRXOVF 寄存器中所有位的或运算。 RBOVIF 位和 CiRXOVF 寄存器中的 RXOVIFn 位不能直接清零。它们需要通过清零 CiFIFOINTn 寄存器中相应的 RXOVFLIF 来清零。

#### 34.12.1.8 CAN 模式改变中断 （MODIF）

当 CAN 模块 OPMOD<2:0> 位发生改变，指示 CAN 模块的工作模式发生改变时，会发生 CAN 模式改变中断 （MODIF）。 ICOD 位将指示模式发生了更改。要清除中断，必须将标志位清零。

#### 34.12.1.9 CAN 时间标记定时器中断 （CTMRIF）

当 CAN 时间标记定时器溢出时，会发生 CAN 时间标记定时器中断 （CMTRIF）。 ICOD 位将指示时间标记定时器溢出。要清除中断，必须将标志位清零。

#### 34.12.1.10 CAN 发送 FIFO 中断 （TBIF）

当发送 FIFO 的状态发生改变时，会发生 CAN 发送 FIFO 中断（TBIF）。该中断具有多个第 3 层中断源。

- 发送 FIFO 未满中断 （TXNFULLIF）
- 发送 FIFO 半满中断 （TXHALFIF）
- 发送 FIFO 为空中断 （TXEMPTYIF）

TBIF 中断不能由应用程序清除。在所有源中断条件终止时，该中断会被清除。

#### 34.12.1.11 CAN 接收 FIFO 中断 （RBIF）

当接收 FIFO 的状态发生改变时，会发生 CAN 接收 FIFO 中断（RBIF）。该中断具有多个第 3 层中断源。

- 接收 FIFO 已满中断 （RXFULLIF）
- 接收 FIFO 半满中断 （RXHALFIF）
- 接收 FIFO 非空中断 （RXNEMPTYIF）

RBIF 中断不能由应用程序清除。在所有源中断条件终止时，该中断会被清除。

### 34.12.2 第 3 层中断

CAN 模块具有以下第 3 层中断。

- 发送 FIFO 未满中断 （TXNFULLIF）
- 发送 FIFO 非半满中断 （TXNHALFIF）
- 发送 FIFO 非空中断 （TXNEMPTYIF）
- 接收 FIFO 已满中断 （RXFULLIF）
- 接收 FIFO 半满中断 （RXHALFIF）
- 接收 FIFO 为空中断 （RXEMPTYIF）
- 接收 FIFO 溢出中断 （RXOVLIF）

这些中断反映发送和接收 FIFO 的当前状态。

### 34.12.3 中断持久性

CAN 模块提供了一些可指示报文 FIFO 工作状态的中断。这些中断是持久中断，即在导致相应条件的中断清除之前，它们会一直存在。中断标志本身不能由应用程序直接清零。例如，只要接收 FIFO 中至少有一个报文，接收 FIFO 非空中断（RXNEMPTYIF）就会保持置 1。以下 CAN 中断是持久中断：

- 发送 FIFO （TBIF）中断 （CiINT<0>）
- 接收 FIFO （RBIF）中断 （CiINT<1>）
- FIFO 状态 （FIFOIPx）中断 （CiFSTAT<31:0>）
- 发送 FIFO 未满 （TXNFULLIF）中断 （CiFIFOINTn<10>）
- 发送 FIFO 半满 （TXHALFIF）中断 （CiFIFOINTn<9>）
- 发送 FIFO 为空 （TXNEMPTYIF）中断 （CiFIFOINTn<8>）
- 接收 FIFO 已满 （RXFULLIF）中断 （CiFIFOINTn<10>）
- 接收 FIFO 半满 （RXHALFIF）中断 （CiFIFOINTn<9>）
- 接收 FIFO 非空 （RXNEMPTYIF）中断 （CiFIFOINTn<8>）

请注意，接收 FIFO 溢出（RXOVFLIF）中断（CiFIFOINTn<11>）是粘性中断，可由用户清除。

### 34.12.4 CAN FIFO 状态寄存器 （CiFSTAT）

CAN FIFO 状态（CiFSTAT）寄存器是一个指示寄存器。如果在 CiFIFOINTn 寄存器中允许中断，则在发生以下任意中断条件时， CiFSTAT 寄存器中相应的 FIFOIPn 位会置 1：

- 发送 FIFO 未满 （TXNFULLIF）中断 （CiFIFOINTn<10>）
- 发送 FIFO 半满 （TXHALFIF）中断 （CiFIFOINTn<9>）
- 发送 FIFO 为空 （TXNEMPTYIF）中断 （CiFIFOINTn<8>）
- 接收 FIFO 已满 （RXFULLIF）中断 （CiFIFOINTn<10>）
- 接收 FIFO 半满 （RXHALFIF）中断 （CiFIFOINTn<9>）
- 接收 FIFO 非空 （RXNEMPTYIF）中断 （CiFIFOINTn<8>）

FIFOIPn 位本身不是中断源。 ICOD 位 （CiVEC<6:0>）会反映已置 1 的 FIFOIPn 位的值。

### 34.12.5 CAN 接收 FIFO 溢出寄存器 （CiRXOVF）

如果接收 FIFO 溢出中断允许位 RXOVFLIE（CiFIFOINTn<19>）置 1，则在发生接收 FIFO 溢出时，CAN 接收 FIFO 溢出寄存器（CiRXOVF）中的 RXOVFn 位会置 1。RXOVFn 位是 RBOVIF（CiINT<11>）中断的源中断。CiRXOVF 寄存器中的位不能直接清零，可以通过清零 RXOVFLIF（CiFIFOINTn<3>）位来清零。

### 34.12.6 中断编码 （ICOD）位

CAN 中断编码 （CiVEC）寄存器中的中断编码位 ICOD （CiVEC<6:0>）会反映 CAN 模块中仍然等待处理的最高中断。ICOD 值越高，待处理中断的自然优先级就越高。ICOD 事件是持久的。如果发生自然优先级较高的中断，屏蔽了 ICOD 寄存器中的较低优先级中断，较低优先级中断将在较高优先级中断清除时立即显现。

例如，假设有以下应用情形：

- FIFO0 导致中断——ICOD = 000 0000。
- 缓冲区 5 也具有导致中断的事件，导致 ICOD 设置为 000 0101。
- 用户进入 CAN 中断处理程序，处理缓冲区 5 并清除中断。
- ICOD 现在将读为 000 0000，指示缓冲区 0 仍然有待处理中断，可以在从中断服务程序 （Interrupt Service Routine， ISR）中返回之前处理它。

### 34.12.7 CAN 选中过滤器位 （FILHIT）

CAN 中断编码 （CiVEC）寄存器中的 CAN 选中过滤器位 FILHIT （CiVEC<12:8>）包含上次与接收报文匹配的过滤器的值。不同于 ICOD 位 （CiVEC<6:0>）， FILHIT 位没有历史记录，只会在接收到报文时更新。

## 34.13 CAN 接收报文时间标记

CAN 模块可以提供接收到报文时的时间值。如果 CAN 控制寄存器中的 CAN 接收报文定时器捕捉事件使能位 CANCAP（CiCON<20>）置 1，则该时间标记将随接收报文一起存储。CAN 模块会将时间标记写入接收报文缓冲区中 CMSGOSID 字段 （CMSG0SID<31:16>）的 CMSGTS 位中。每次接收到有效帧时， CAN 模块会通过捕捉 CiTMR 寄存器中的 CAN 时间标记定时器位 CANTS（CiTMR<31:16>）的值来获取时间标记。因为 CAN 规范规定，如果在成功发送 EOF 字段之前未发生任何错误，则表示帧是有效的，所以将在 EOF 之后立即捕捉 CANTS 的值。

模块会对所有接收报文进行时间标记，即使是模块自身发送和接收的报文。

应用程序可以通过 CiTMR 寄存器中的 CAN 时间标记定时器预分频比位 CANSTPRE（CiTMR<15:0>），以系统时钟为单位配置 CAN 时间标记的时间间隔。

## 34.14 低功耗模式

### 34.14.1 休眠模式

当 CPU 进入休眠模式时，用户应用程序必须确保模块不处于活动状态。为了保护 CAN 总线系统，避免由于违反以上规则而导致严重后果，在休眠时，模块会将 TXCAN 引脚驱动为隐性状态。建议的步骤是，先让模块进入禁止模式，然后再将 CPU 置于休眠模式。

### 34.14.2 空闲模式

当 CPU 处于空闲模式时，如果 CAN 控制寄存器中的空闲模式停止位 SIDLE（CiCON<13>）为 1，则模块会关闭电源。用户应用程序必须确保在 CPU 进入空闲模式且 SIDLE 位置 1 时，模块不处于活动状态。为了保护 CAN 总线系统，避免由于违反以上规则而导致严重后果，在 CPU 处于空闲模式且 SIDLE 位置 1 时，模块会将 CiTX 引脚驱动为隐性状态。

### 34.14.3 唤醒功能

在模块休眠时，CAN模块会监视CAN接收（CiRX）引脚上是否有活动。以下情况下，模块会休眠：

- 器件处于休眠模式
- 器件处于空闲模式且 SIDLE 位置 1

处于该模式时，如果 CAN 中断（CiINT）寄存器中的总线唤醒活动中断允许位 WAKIE（CiINT<30>）置 1 并且检测到总线活动，则 CAN 模块会产生中断。

在禁止模式或休眠 / 空闲模式下，模块可以设定为对 CiRX 线应用低通滤波器功能。该功能可以用于保护模块，避免由于 CAN 总线线路上的短暂毛刺而导致唤醒。CAN 配置（CiCFG）寄存器中的 CAN 总线线路滤波器使能位 WAKFIL（CICFG<22>）用于在模块休眠时使能或禁止滤波器。

请注意，在 CAN 模块产生中断时， CAN 模块自身的状态不会受中断影响。

以下步骤描述了对 PIC32 器件休眠模式使用唤醒中断功能的建议过程。

1. 为了使能总线活动唤醒功能，应将 WAKIE 位置 1。
2. 在将器件置为休眠模式之前，先将 CAN 工作模式切换为禁止模式，并等待 CAN 模块进入禁止模式。
3. 将 PIC32 器件置为休眠模式。
4. 在休眠期间，如果存在 CAN 总线活动， PIC32 器件会从休眠模式唤醒。如果允许了 CAN CPU 中断， CPU 将会执行 CAN 中断服务程序。
5. 在 ISR 中，检查 WAKIF 位是否置 1，并将 CAN 工作模式切换为正常模式。在切换完毕时，CAN模块将能够接收报文（清零WAKIF位）。请注意，唤醒器件的CAN报文将会丢失。

## 34.15 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 PIC32MX 器件系列而编写的，但其概念是相近的，通过适当修改并受到一定限制即可使用。当前与控制器局域网（CAN）模块相关的应用笔记有：

| 标题 | 应用笔记编号 |
|---|---|
| 目前没有相关的应用笔记。 | N/A |

**注：** 如需获取更多 PIC32MX 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

## 34.16 版本历史

### 版本 A （2009 年 7 月）

这是本文档的初始版本。

注：

**请注意以下有关 Microchip 器件代码保护功能的要点：**

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了 《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。



ISBN：978-1-60932-531-2

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

*Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2002 认证。公司在 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。*

# 全球销售及服务网点

## 美洲


**公司总部 Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

**亚特兰大 Atlanta**
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

**波士顿 Boston**
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

**芝加哥 Chicago**
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

**克里夫兰 Cleveland**
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

**达拉斯 Dallas**
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

**底特律 Detroit**
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

**科科莫 Kokomo**
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

**洛杉矶 Los Angeles**
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

**圣克拉拉 Santa Clara**
Santa Clara, CA
Tel: 1-408-961-6444
Fax: 1-408-961-6445

**加拿大多伦多 Toronto**
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509


## 亚太地区


**亚太总部 Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

**中国 – 北京**
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

**中国 – 成都**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**中国 – 重庆**
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

**中国 – 香港特别行政区**
Tel: 852-2401-1200
Fax: 852-2401-3431

**中国 – 南京**
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

**中国 – 青岛**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**中国 – 上海**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**中国 – 沈阳**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**中国 – 深圳**
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

**中国 – 武汉**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**中国 – 西安**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

**中国 – 厦门**
Tel: 86-592-238-8138
Fax: 86-592-238-8130

**中国 – 珠海**
Tel: 86-756-321-0040
Fax: 86-756-321-0049

**台湾地区 – 高雄**
Tel: 886-7-213-7830
Fax: 886-7-330-9305

**台湾地区 – 台北**
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102


## 亚太地区


**台湾地区 – 新竹**
Tel: 886-3-6578-300
Fax: 886-3-6578-370

**澳大利亚 Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**印度 India - Bangalore**
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

**印度 India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**印度 India - Pune**
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

**日本 Japan - Yokohama**
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

**韩国 Korea - Daegu**
Tel: 82-53-744-4301
Fax: 82-53-744-4302

**韩国 Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

**马来西亚 Malaysia - Kuala Lumpur**
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

**马来西亚 Malaysia - Penang**
Tel: 60-4-227-8870
Fax: 60-4-227-4068

**菲律宾 Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**新加坡 Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**泰国 Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350


## 欧洲


**奥地利 Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**丹麦 Denmark-Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**法国 France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**德国 Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**意大利 Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**荷兰 Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**西班牙 Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**英国 UK - Wokingham**
Tel: 44-118-921-5869
Fax: 44-118-921-5820


07/15/10