24 控制器区域网络 (bxCAN)

除非特别说明，否则本节适用于整个 STM32F4xx 系列器件。

24.1 bxCAN 简介

基本扩展 CAN 外设又称 **bxCAN**，可与 CAN 网络进行交互。该外设支持 2.0A 和 B 版本的 CAN 协议，旨在以最少的 CPU 负载高效管理大量的传入消息，并可按需要的优先级实现消息发送。

在攸关安全性的应用中，CAN 控制器提供所有必要的硬件功能来支持 CAN 时间触发通信方案。

24.2 bxCAN 主要特性

- 支持 2.0 A 及 2.0 B Active 版本 CAN 协议
- 比特率高达 1 Mb/s
- 支持时间触发通信方案

发送

- 三个发送邮箱
- 可配置的发送优先级
- SOF 发送时间戳

接收

- 两个具有三级深度的接收 FIFO
- 可调整的筛选器组：
  - CAN1 和 CAN2 之间共享 28 个筛选器组
- 标识符列表功能
- 可配置的 FIFO 上溢
- SOF 接收时间戳

时间触发通信方案

- 禁止自动重发送模式
- 16 位自由运行定时器
- 在最后两个数据字节发送时间戳

管理

- 可屏蔽中断
- 在唯一地址空间通过软件实现高效的邮箱映射

双 CAN

- CAN1：主 bxCAN，用于管理 bxCAN 与 512 字节 SRAM 存储器之间的通信。
- CAN2：从 bxCAN，无法直接访问 SRAM 存储器。
- 两个 bxCAN 单元共享 512 字节 SRAM 存储器（请参见*图 224：双 CAN 框图*）。

24.3 bxCAN 一般说明

在如今的 CAN 应用中，网络节点数量日益增多，经常需要通过网关将数个网络连接在一起。系统中的消息数量（以及各个节点需要处理的消息）也有了显著增加。除应用程序消息外，还引入了网络管理和诊断消息。

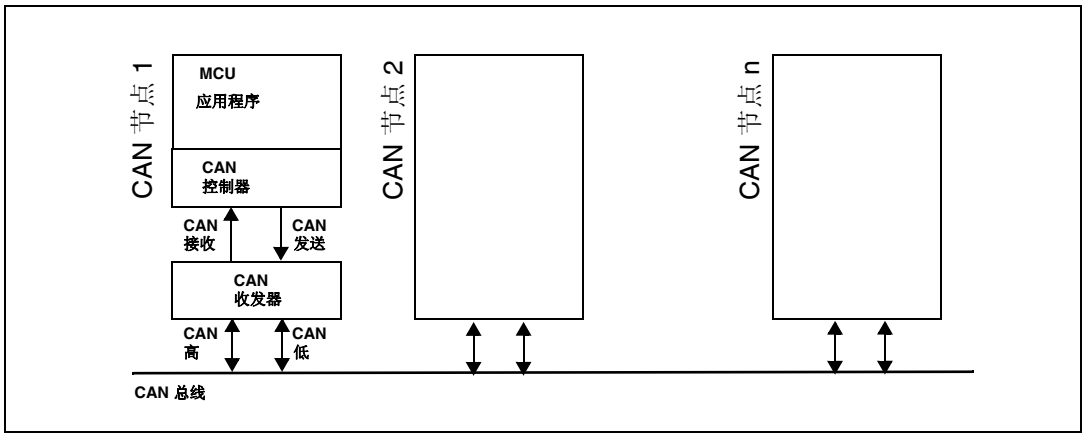
- 各种类型的消息需要一个增强的筛选机制进行处理。

此外，应用程序任务需要更多的 CPU 时间，因此必须减少因消息接收而对实时处理造成的限制。

- 接收 FIFO 方案使 CPU 能够长时间专门处理应用程序任务，而又不致丢失消息。

基于标准 CAN 驱动程序的标准 HLP（更高层协议）需要一个高效接口来与 CAN 控制器连接。

图 223. CAN 网络拓扑结构



24.3.1 CAN 2.0B 主动内核

bxCAN 模块可完全自主地处理 CAN 消息的发送和接收。标准标识符（11 位）和扩展标识符（29 位）完全由硬件支持。

24.3.2 控制、状态和配置寄存器

应用程序使用这些寄存器进行以下操作：

- 配置 CAN 参数，例如波特率
- 请求发送
- 处理接收
- 管理中断
- 获取诊断信息

24.3.3 发送邮箱

软件可通过三个发送邮箱设置消息。发送调度程序负责决定首先发送哪个邮箱的内容。

24.3.4 验收筛选器

bxCAN 提供了 28 个可调整/可配置的标识符筛选器组，用于选择软件所需的传入消息并丢弃其余消息。

bxCAN 提供了 28 个可调整/可配置的标识符筛选器组，用于选择软件所需的传入消息并丢弃其余消息。其他器件中，共有 14 个可调整/可配置的标识符筛选器组。

接收 FIFO

硬件使用两个接收 FIFO 来存储传入消息。每个 FIFO 中可以存储三条完整消息。FIFO 完全由硬件管理。

图 224. 双 CAN 框图

24.4 bxCAN 工作模式

bxCAN 有三种主要的工作模式：**初始化**、**正常**和**睡眠**。硬件复位后，bxCAN 进入睡眠模式以降低功耗，同时 CANTX 上的内部上拉电阻激活。软件将 CAN_MCR 寄存器的 INRQ 或 SLEEP 位置 1，以请求 bxCAN 进入**初始化**或**睡眠**模式。一旦进入该模式，bxCAN 即将 CAN_MSR 寄存器的 INAK 或 SLAK 位置 1，以确认该模式，同时禁止内部上拉电阻。如果 INAK 和 SLAK 均未置 1，则 bxCAN 将处于**正常**模式。进入**正常**模式之前，bxCAN 必须始终在 CAN 总线上实现**同步**。为了进行同步，bxCAN 将等待 CAN 总线空闲（即，已监测到 CANRX 上的 11 个隐性位）。

24.4.1 初始化模式

当硬件处于初始化模式时，可以进行软件初始化。为进入该模式，软件将 CAN_MCR 寄存器的 INRQ 位置 1，并等待硬件通过将 CAN_MCR 寄存器的 INAK 位置 1 来确认请求。

为退出初始化模式，软件将 INQR 位清零。一旦硬件将 INAK 位清零，bxCAN 即退出初始化模式。

在初始化模式下，所有从 CAN 总线传入和传出的消息都将停止，并且 CAN 总线输出 CANTX 的状态为隐性（高）。

进入初始化模式不会更改任何配置寄存器。

为初始化 CAN 控制器，软件必须设置位定时 (CAN_BTR) 和 CAN 选项 (CAN_MCR) 寄存器。

为初始化与 CAN 筛选器组相关的寄存器（模式、尺度、FIFO 分配、激活和筛选器值），软件必须将 FINIT 位 (CAN_FMR) 置 1。筛选器的初始化也可以在初始化模式之外进行。

注意： *FINIT=1 时，CAN 接收停用。*

筛选器值也可通过停用（CAN_FA1R 寄存器的）相关筛选器激活位来修改。

如果某个筛选器组未使用，建议将其保持未激活状态（将相应 FACT 位保持清零）。

24.4.2 正常模式

一旦初始化完成，软件必须向硬件请求进入正常模式，这样才能在 CAN 总线上进行同步，并开始接收和发送。

进入正常模式的请求可通过将 CAN_MCR 寄存器的 INRQ 位清零来发出。bxCAN 进入正常模式，并与 CAN 总线上的数据传输实现同步后，即可参与总线活动。执行这一步时，需要等待出现一个由 11 个连续隐性位（总线空闲状态）组成的序列。硬件通过将 CAN_MSR 寄存器的 INAK 位清零，来确认切换到正常模式。

筛选器值的初始化与初始化模式无关，但必须要在筛选器处于未激活状态（相应 FACTx 位清零）时进行。筛选器尺度和模式配置必须在进入正常模式之前完成。

24.4.3 睡眠模式（低功耗）

为降低能耗功耗，bxCAN 具有低功耗模式，称为睡眠模式。软件通过将 CAN_MCR 寄存器的 SLEEP 位置 1 而发出请求后，即可进入该模式。该模式下，bxCAN 时钟停止，但软件仍可访问 bxCAN 邮箱。

在 bxCAN 处于**睡眠**模式时，如果软件通过将 INRQ 位置 1 来请求进入**初始化**模式，则必须同时将 SLEEP 位清零。

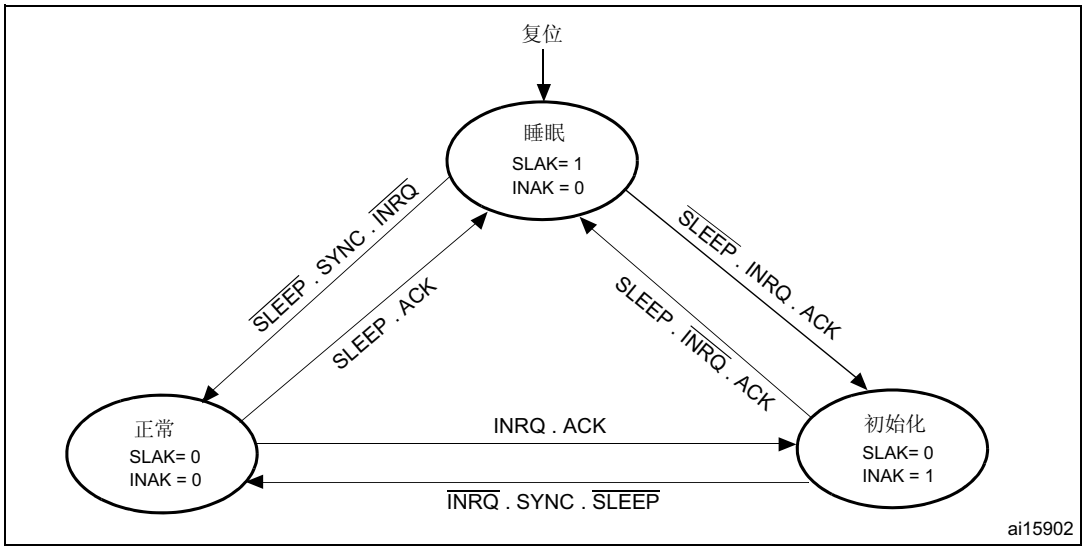
软件将 SLEEP 位清零或是检测到 CAN 总线活动时，bxCAN 即被唤醒（退出睡眠模式）。

检测到 CAN 总线活动后，如果 CAN_MCR 寄存器的 AWUM 位置 1，硬件将通过清零 SLEEP 位来自动执行唤醒序列。如果 AWUM 位清零，在发生唤醒中断时，软件必须将 SLEEP 位清零才能退出睡眠模式。

注意： *如果使能唤醒中断（CAN_IER 寄存器的 WKUIE 位置 1），即使 bxCAN 自动执行唤醒序列，一旦检测到 CAN 总线活动，也会发生唤醒中断。*

SLEEP 位清零后，一旦 bxCAN 与 CAN 总线同步，即会退出睡眠模式，请参见*图 225：bxCAN 工作模式*。一旦硬件将 SLAK 位清零，即会退出睡眠模式。

图 225. bxCAN 工作模式



1. ACK = 硬件通过将 CAN_MSR 寄存器的 INAK 或 SLAK 位置 1 来确认请求的等待状态
2. SYNC = bxCAN 等待 CAN 总线变为空闲（即在 CANRX 上监测到连续 11 个隐性位）的状态

24.5 测试模式

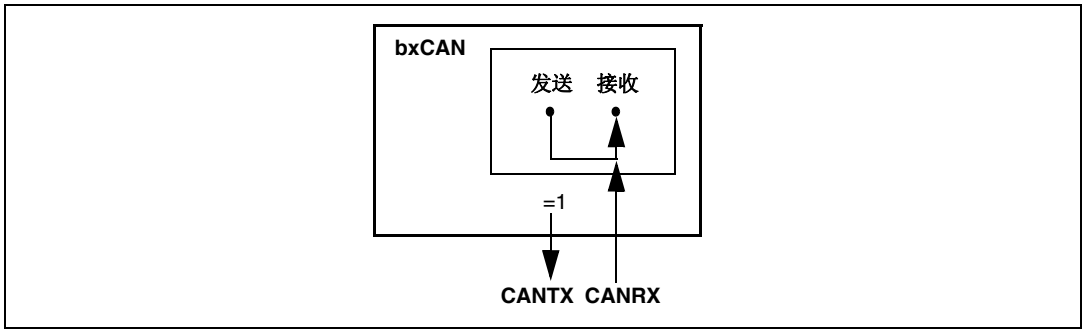
可以通过 CAN_BTR 寄存器中的 SILM 和 LBKM 位来选择测试模式。这些位必须在 bxCAN 处于初始化模式时进行配置。选择测试模式后，必须复位 CAN_MCR 寄存器中的 INRQ 位才能进入正常模式。

24.5.1 静默模式

可以通过将 CAN_BTR 寄存器的 SILM 位置 1，将 bxCAN 置于静默模式。

在静默模式下，bxCAN 可以接收有效数据帧和有效遥控帧，但仅在 CAN 总线上发送隐性位，并且无法启动发送。如果 bxCAN 必须发送一个显性位（ACK 位、溢出标志、活动错误标志），该位将在内部被改道发送，以便 CAN 内核可以监视该显性位，但 CAN 总线可以保持隐性状态。静默模式可用于分析 CAN 总线上的流量，同时又不会因发送显性位（确认位、错误帧）对其造成影响。

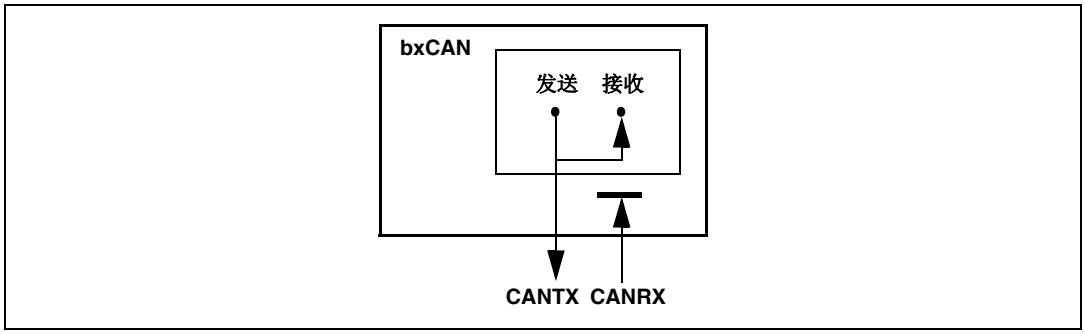
图 226. 静默模式下的 bxCAN



24.5.2 环回模式

可以通过将 CAN_BTR 寄存器的 LBKM 位置 1，将 bxCAN 置于环回模式。在环回模式下，bxCAN 将其自身发送的消息作为接收的消息来处理并存储（如果这些消息通过了验收筛选）在接收邮箱中。

图 227. 环回模式下的 bxCAN

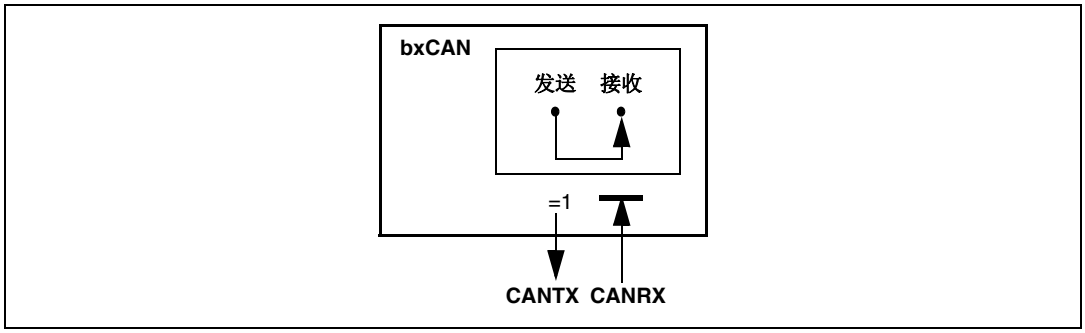


该模式为自检功能提供。为了不受外部事件的影响，CAN 内核在环回模式下将忽略确认错误（在数据/远程帧的确认时隙不对显性位采样）。在此模式下，bxCAN 将执行从发送输出到接收输入的内部反馈。bxCAN 将忽略 CANRX 输入引脚的实际值。从 CANTX 引脚可以监视发送的消息。

24.5.3 环回与静默组合模式

可以通过将 CAN_BTR 寄存器的 LBKM 和 SILM 位置 1，将环回模式和静默模式组合起来。该模式可用于“热自检”，也就是说，bxCAN 可以像在环回模式下一样进行检测，同时又不会影响与 CANTX 和 CANRX 引脚相连接的运行中的 CAN 系统。在此模式下，CANRX 引脚与 bxCAN 断开连接，CANTX 引脚则保持隐性。

图 228. 组合模式下的 bxCAN



24.6 调试模式

当微控制器进入调试模式（Cortex™-M4F 内核停止）时，bxCAN 可以继续正常工作，也可以停止工作，具体取决于如下条件：

- DBG 模块中用于 CAN1 的 DBG_CAN1_STOP 位或者用于 CAN2 的 DBG_CAN2_STOP 位。有关详细信息，请参见*第 33.16.2 节：对定时器、看门狗、bxCAN 和 I^2C 的调试支持*。
- CAN_MCR 中的 DBF 位。有关详细信息，请参见*第 24.9.2 节：CAN 控制和状态寄存器*。

24.7 bxCAN 功能说明

24.7.1 发送处理

为了发送消息，应用程序必须在请求发送前，通过将 CAN_TIxR 寄存器的相应 TXRQ 位置 1，选择一个**空**发送邮箱，并设置标识符、数据长度代码 (DLC) 和数据。一旦邮箱退出**空**状态，软件即不再具有对邮箱寄存器的写访问权限。TXRQ 位置 1 后，邮箱立即进入**挂起**状态，等待成为优先级最高的邮箱，请参见*发送优先级*。一旦邮箱拥有最高优先级，即被**安排**发送。CAN 总线变为空闲后，被安排好的邮箱中的消息即开始发送（进入**发送**状态）。邮箱一旦发送成功，即恢复**空**状态。硬件通过将 CAN_TSR 寄存器的 RQCP 和 TXOK 位置 1，来表示发送成功。

如果发送失败，失败原因将由 CAN_TSR 寄存器的 ALST 位（仲裁丢失）和/或 TERR 位（检测到发送错误）指示。

发送优先级

按标识符

当多个发送邮箱挂起时，发送顺序由邮箱中所存储消息的标识符来确定。根据 CAN 协议的仲裁，标识符值最低的消息具有最高的优先级。如果标识符值相等，则首先安排发送编号较小的邮箱。

按发送请求顺序

可以通过设置 CAN_MCR 寄存器中的 TXFP 位，将发送邮箱配置为发送 FIFO。在此模式下，优先级顺序按照发送请求顺序来确定。

该模式对分段发送非常有用。

中止

可以通过将 CAN_TSR 寄存器的 ABRQ 位置 1，来中止发送请求。在**挂起**或**已安排**状态下，邮箱立即中止。如果在邮箱处于**发送**状态时请求中止，则会出现两种结果。如果邮箱发送成功，将变为**空**状态，同时 CAN_TSR 寄存器的 TXOK 位置 1。如果发送失败，邮箱变为**已安排**状态，发送中止并变为**空**状态，同时 TXOK 位清零。在所有情况下，邮箱至少在当前发送结束时都会恢复**空**状态。

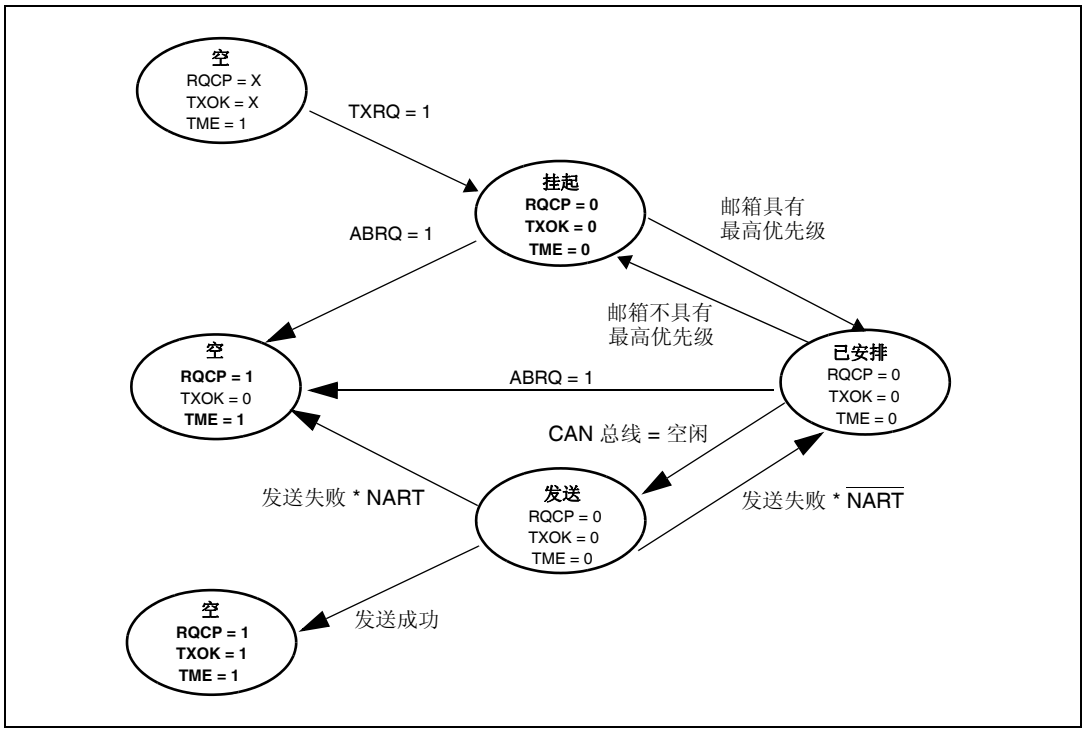
禁止自动重发送模式

该模式旨在满足 CAN 标准的时间触发通信方案的要求。要将硬件配置为此模式，必须将 CAN_MCR 寄存器的 NART 位置 1。

在此模式下，每个发送仅启动一次。如果第一次尝试失败，由于仲裁丢失或错误，硬件将不会自动重新启动消息发送。

第一次发送尝试结束时，硬件将认为请求已完成，并将 CAN_TSR 寄存器的 RQCP 位置 1。发送结果由 CAN_TSR 寄存器的 TXOK、ALST 和 TERR 位来指示。

图 229. 发送邮箱状态



24.7.2 时间触发通信模式

在此模式下，CAN 硬件的内部计数器激活，用于为接收和发送邮箱生成时间戳值，这些值分别存储在 CAN_RDTxR/CAN_TDTxR 寄存器中。内部计数器在每个 CAN 位时间递增（请参见*第 24.7.7 节：位时序*）。在接收和发送时，都会在帧起始位的采样点捕获内部计数器。

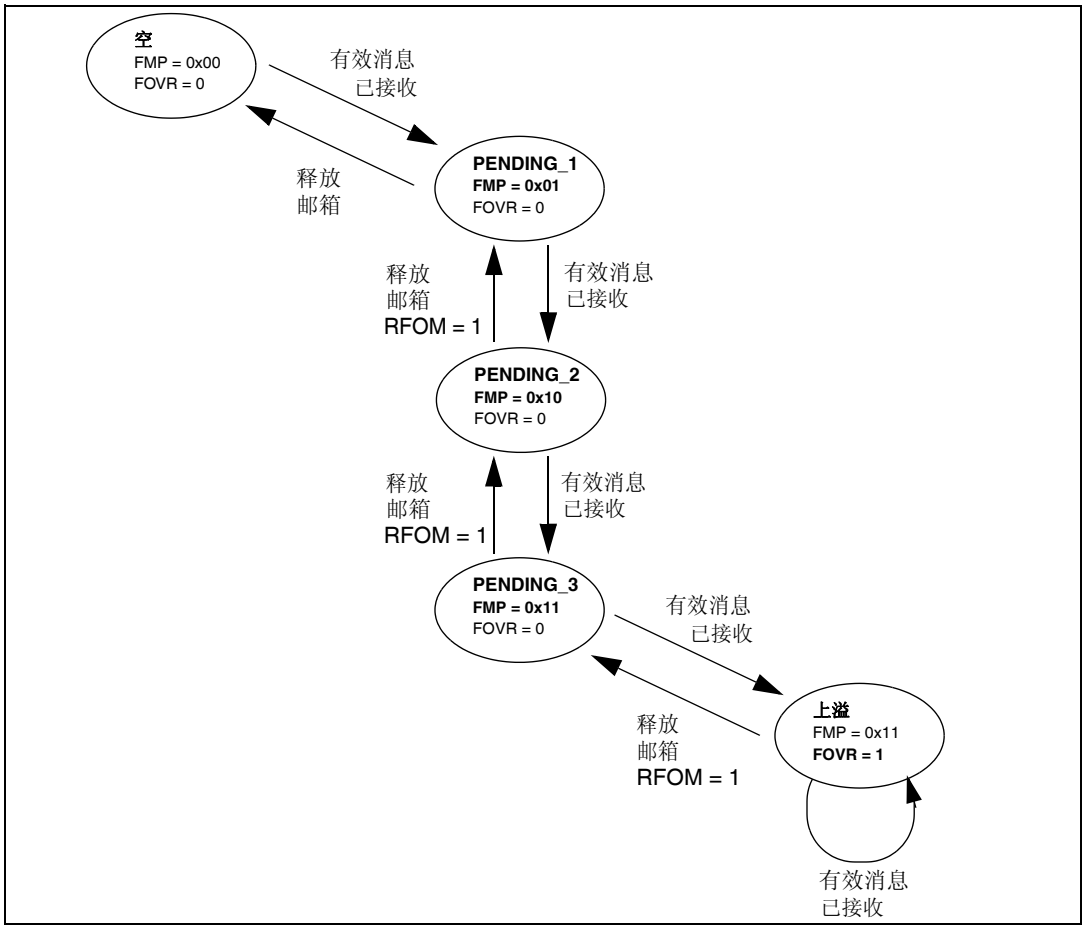
24.7.3 接收处理

为了接收 CAN 消息，提供了构成 FIFO 的三个邮箱。为了节约 CPU 负载，简化软件并保证数据一致性，FIFO 完全由硬件进行管理。应用程序通过 FIFO 输出邮箱访问 FIFO 中所存储的消息。

有效消息

当消息依据 CAN 协议正确接收（直到 EOF 字段的倒数第二位都没有发送错误）并且成功通过标识符筛选后，该消息将视为有效，请参见*第 24.7.4 节：标识符筛选*。

图 230. 接收 FIFO 状态



FIFO 管理

FIFO 开始时处于**空**状态，在接收的第一条有效消息存储在其中后，变为 **Pending_1** 状态。硬件通过将 CAN_RFR 寄存器的 FMP[1:0] 位置为 01b 来指示该事件。消息将在 FIFO 输出邮箱中供读取。软件将读取邮箱内容，并通过将 CAN_RFR 寄存器的 RFOM 位置 1，来将邮箱释放。FIFO 随即恢复**空**状态。如果同时接收到新的有效消息，FIFO 将保持 **Pending_1** 状态，新消息将在输出邮箱中供读取。

如果应用程序未释放邮箱，下一条有效消息将存储在 FIFO 中，使其进入 **Pending_2** 状态 (FMP[1:0] = 10b)。下一条有效消息会重复该存储过程，同时将 FIFO 变为 **Pending_3** 状态 (FMP[1:0] = 11b)。此时，软件必须通过将 RFOM 位置 1 来释放输出邮箱，从而留出一个空邮箱来存储下一条有效消息。否则，下一次接收到有效消息时，将导致消息丢失。

另请参见*第 24.7.5 节：消息存储*。

上溢

一旦 FIFO 处于 **Pending_3** 状态（即三个邮箱均已满），则下一次接收到有效消息时，将导致**上溢**并丢失一条消息。硬件通过将 CAN_RFR 寄存器的 FOVR 位置 1 来指示上溢状况。丢失的消息取决于 FIFO 的配置：

- 如果禁止 FIFO 锁定功能（CAN_MCR 寄存器的 RFLM 位清零），则新传入的消息将覆盖 FIFO 中存储的最后一条消息。在这种情况下，应用程序将始终能访问到最新的消息。
- 如果使能 FIFO 锁定功能（CAN_MCR 寄存器的 RFLM 位置 1），则将丢弃最新的消息，软件将提供 FIFO 中最早的三条消息。

与接收相关的中断

消息存储到 FIFO 中后，FMP[1:0] 位即会更新，如果 CAN_IER 寄存器的 FMPIE 位置 1，将产生中断请求。

FIFO 存满消息（即存储了第三条消息）后，CAN_RFR 寄存器的 FULL 位置 1，如果 CAN_IER 寄存器的 FFIE 位置 1，将产生中断。

出现上溢时，FOVR 位将置 1，如果 CAN_IER 寄存器的 FOVIE 位置 1，将产生中断。

24.7.4 标识符筛选

在 CAN 协议中，消息的标识符与节点地址无关，但与消息内容有关。因此，发送器将消息广播给所有接收器。在接收到消息时，接收器节点会根据标识符的值来确定软件是否需要该消息。如果需要，该消息将复制到 SRAM 中。如果不需要，则必须在无软件干预的情况下丢弃该消息。

为了满足这一要求，bxCAN 控制器为应用程序提供了 28 个可配置且可调整的筛选器组 (27-0)。在其他器件中，bxCAN 控制器为应用程序提供了 14 个可配置且可调整的筛选器组 (13-0)，以便仅接收软件需要的消息。此硬件筛选功能可以节省软件筛选所需的 CPU 资源。每个筛选器组 x 均包含两个 32 位寄存器，分别是 CAN_FxR0 和 CAN_FxR1。

可调整的宽度

为了根据应用程序的需求来优化和调整筛选器，每个筛选器组可分别进行伸缩调整。根据筛选器尺度不同，一个筛选器组可以：

- 为 STDID[10:0]、EXTID[17:0]、IDE 和 RTR 位提供一个 32 位筛选器。
- 为 STDID[10:0]、RTR、IDE 和 EXTID[17:15] 位提供两个 16 位筛选器。

请参见*图 231*。

此外，筛选器还可配置为掩码模式或标识符列表模式。

掩码模式

在**掩码**模式下，标识符寄存器与掩码寄存器关联，用以指示标识符的哪些位“必须匹配”，哪些位“无关”。

标识符列表模式

在**标识符列表**模式下，掩码寄存器用作标识符寄存器。这时，不会定义一个标识符和一个掩码，而是指定两个标识符，从而使单个标识符的数量加倍。传入标识符的所有位都必须与筛选器寄存器中指定的位匹配。

筛选器组尺度和模式配置

筛选器组通过相应的 CAN_FMR 寄存器进行配置。为了配置筛选器组，必须通过将 CAN_FAR 寄存器的 FACT 位清零而将其停用。筛选器尺度通过 CAN_FS1R 寄存器的相应 FSCx 位进行配置，请参见*图 231*。相应掩码/标识符寄存器的**标识符列表**或**标识符掩码**模式通过 CAN_FMR 寄存器的 FBMx 位进行配置。

要筛选一组标识符，应将掩码/标识符寄存器配置为掩码模式。

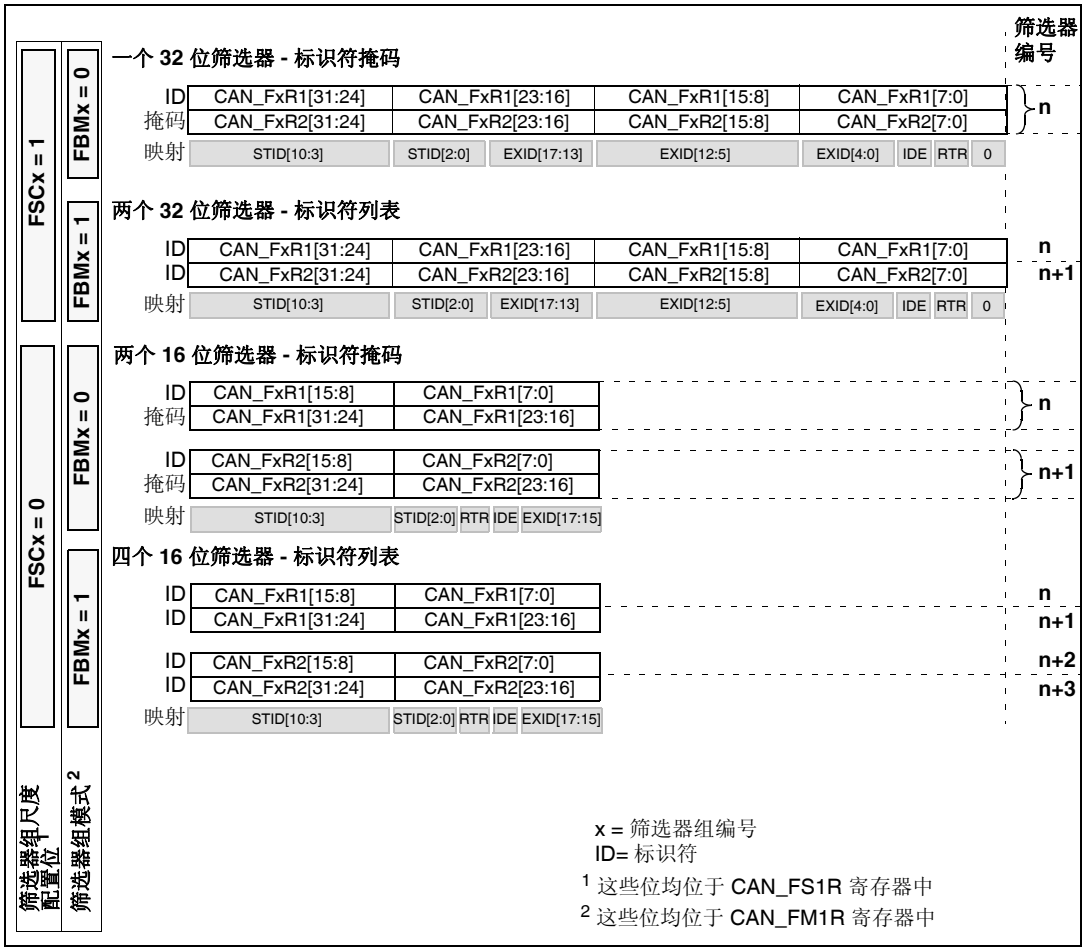
要选择单个标识符，应将掩码/标识符寄存器配置为标识符列表模式。

未由应用程序使用的筛选器应保持停用。

筛选器组中的每个筛选器将按从 0 到最大值的顺序进行编号（称为*筛选器编号*），具体取决于每个筛选器组的模式和尺度。

有关筛选器配置，请参见*图 231*。

图 231. 筛选器组尺度配置 - 寄存器构成



筛选器匹配索引

消息接收到 FIFO 中后，即可供应用程序使用。应用程序数据通常会复制到 SRAM 中的位置。为了将数据复制到正确的位置，应用程序必须通过标识符来识别数据。为了避免这种情况，方便访问 SRAM 位置，CAN 控制器提供了一个筛选器匹配索引。

该索引根据筛选器优先级规则与消息一同存储在邮箱中。因此，每条收到的消息都有相关联的筛选器匹配索引。

筛选器匹配索引的使用方法有两种：

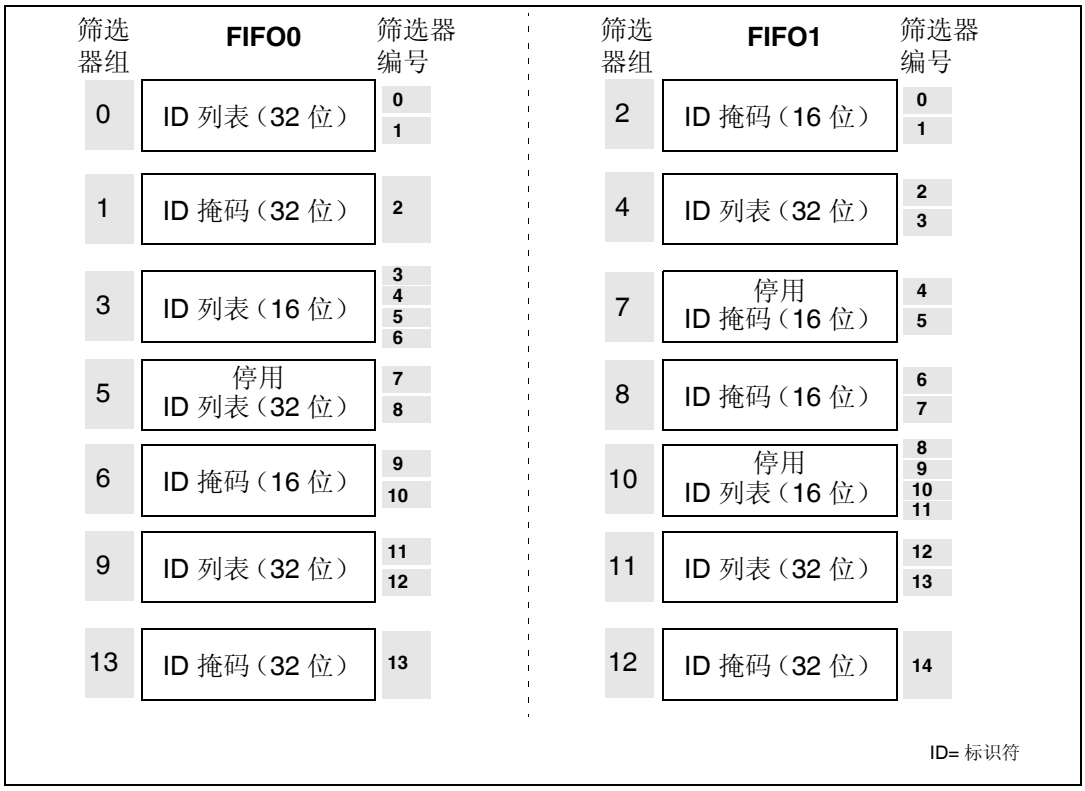
- 将筛选器匹配索引与预期值列表进行比较。
- 将筛选器匹配索引用作阵列索引，以访问数据目标位置。

对于非屏蔽筛选器，软件不再需要比较标识符。

如果筛选器有屏蔽，软件则只需比较屏蔽位。

筛选器编号的索引值与筛选器组的激活状态无关。此外，还将使用两个独立的编号方案，每个 FIFO 各一个。有关示例，请参见*图 232*。

图 232. 筛选器编号示例

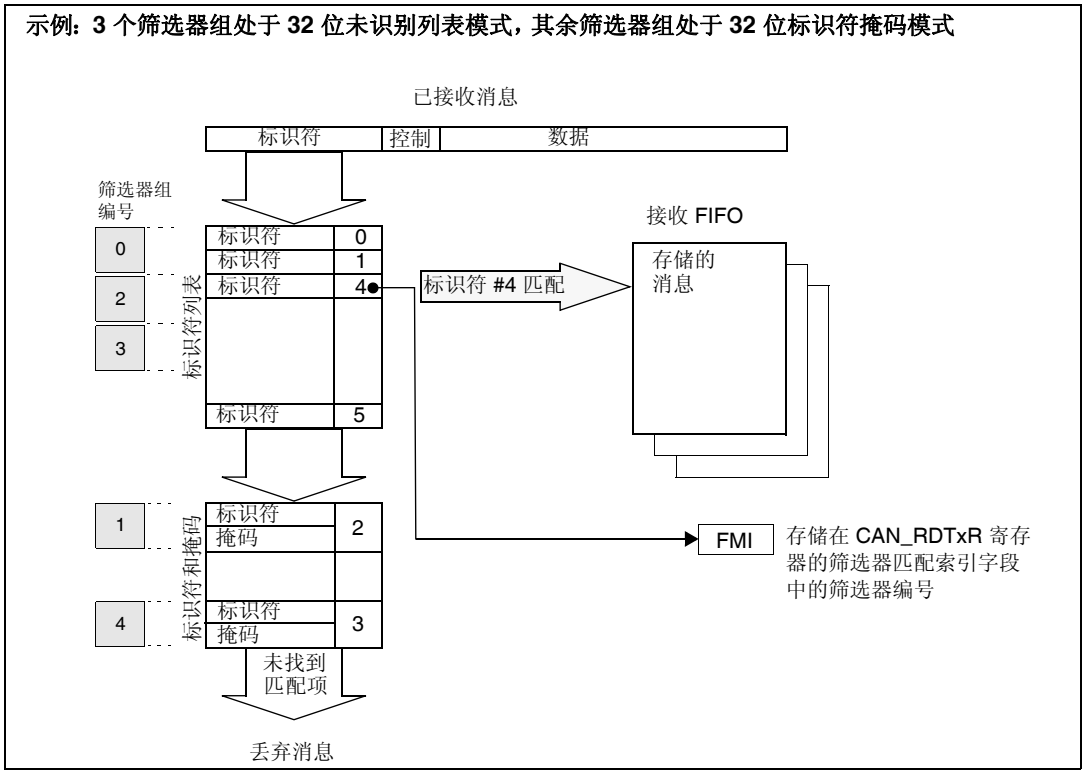


筛选器优先级规则

根据筛选器组合，可能会出现一个标识符成功通过数个筛选器的情况。这种情况下，将根据以下优先级规则选择接收邮箱中存储的筛选器匹配值：

- 32 位筛选器优先于 16 位筛选器。
- 对于尺度相等的筛选器，标识符列表模式优先于标识符掩码模式。
- 对于尺度和模式均相等的筛选器，则按筛选器编号确定优先级（编号越低，优先级越高）。

图 233. 筛选机制 - 示例



以上示例说明了 bxCAN 的筛选原则。接收到消息后，首先将标识符与标识符列表模式中配置的筛选器进行比较。如果匹配，消息将存储在相应 FIFO 中，匹配筛选器的索引则存储在筛选器匹配索引中。如本例所示，标识符与标识符 #2 匹配，因此消息内容和 FMI 2 存储到该 FIFO 中。

如果不匹配，则将传入标识符与掩码模式中配置的筛选器进行比较。

如果标识符与筛选器中配置的任何标识符均不匹配，硬件会在不干扰软件的情况下丢弃该消息。

24.7.5 消息存储

CAN 消息软件与硬件之间的接口通过邮箱实现。邮箱中包含所有与消息相关的信息：标识符、数据、控制、状态和时间戳信息。

发送邮箱

软件在空发送邮箱中设置将要发送的消息。发送状态由硬件在 CAN_TSR 寄存器中进行指示。

表 100. 发送邮箱映射

与发送邮箱基址之间的偏移	寄存器名称
0	CAN_TIxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

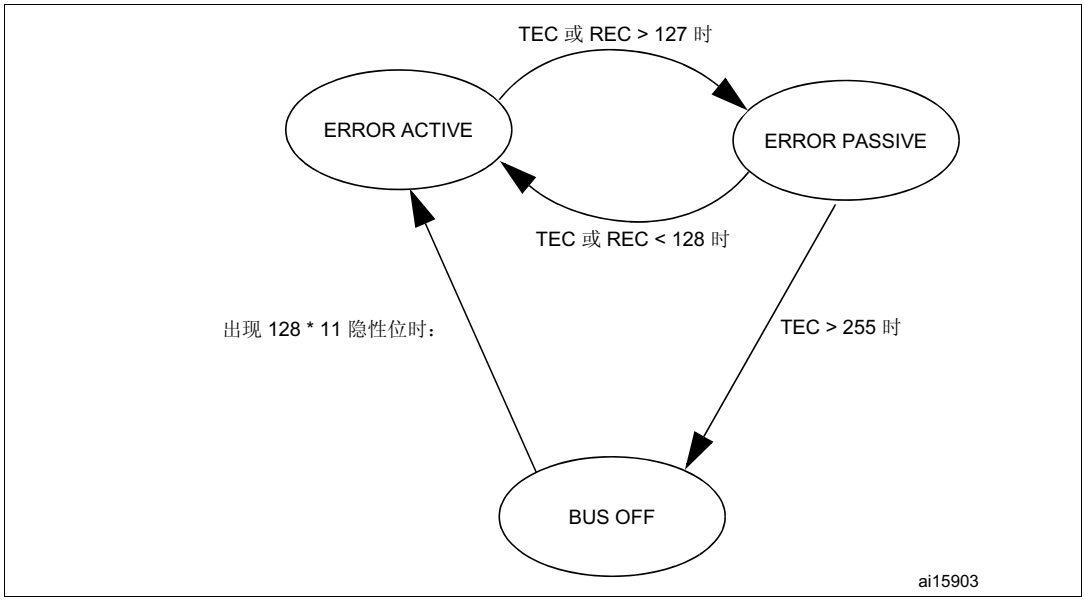
接收邮箱

消息在接收到后，将在 FIFO 输出邮箱中供软件使用。一旦软件对消息进行了处理（例如读取），则必须通过 CAN_RFR 寄存器的 RFOM 位释放 FIFO 输出邮箱，以接收下一条传入消息。筛选器匹配索引存储在 CAN_RDTxR 寄存器的 MFMI 字段中。16 位时间戳值则存储在 CAN_RDTxR 的 TIME[15:0] 字段中。

表 101. 接收邮箱映射

与接收邮箱基址之间的偏移（字节）	寄存器名称
0	CAN_RIxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

图 234. CAN 错误状态图



24.7.6 错误管理

如 CAN 协议所述，错误管理完全由硬件通过发送错误计数器（CAN_ESR 寄存器中的 TEC 值）和接收错误计数器（CAN_ESR 寄存器中的 REC 值）来处理，这两个计数器根据错误状况进行递增或递减。有关 TEC 和 REC 管理的详细信息，请参见 CAN 标准。

两者均可由软件读取，用以确定网络的稳定性。此外，CAN 硬件还将在 CAN_ESR 寄存器中提供当前错误状态的详细信息。通过 CAN_IER 寄存器（ERRIE 位等），软件可以非常灵活地配置在检测到错误时生成的中断。

总线关闭恢复

当 TEC 大于 255 时，达到总线关闭状态，该状态由 CAN_ESR 寄存器的 BOFF 位指示。在总线关闭状态下，bxCAN 不能再发送和接收消息。

bxCAN 可以自动或者应软件请求而从总线关闭状态中恢复（恢复错误主动状态），具体取决于 CAN_MCR 寄存器的 ABOM 位。但在两种情况下，bxCAN 都必须至少等待 CAN 标准中指定的恢复序列完成（在 CANRX 上监测到 128 次 11 个连续隐性位）。

如果 ABOM 位置 1，bxCAN 将在进入总线关闭状态后自动启动恢复序列。

如果 ABOM 位清零，则软件必须请求 bxCAN 先进入再退出初始化模式，从而启动恢复序列。

注意： *在初始化模式下，bxCAN 不会监视 CANRX 信号，因此无法完成恢复序列。**要进行恢复，bxCAN 必须处于正常模式**。*

24.7.7 位时序

位时序逻辑将监视串行总线，执行采样并调整采样点，在调整采样点时，需要在起始位边沿进行同步并后续的边沿进行再同步。

通过将标称位时间划分为以下三段，即可解释其工作过程：

- **同步段 (SYNC_SEG)**：位变化应该在此时间段内发生。它只有一个时间片的固定长度 (1 x t_{CAN})。
- **位段 1 (BS1)**：定义采样点的位置。它包括 CAN 标准的 PROP_SEG 和 PHASE_SEG1。其持续长度可以在 1 到 16 个时间片之间调整，但也可以自动加长，以补偿不同网络节点的频率差异所导致的正相位漂移。
- **位段 2 (BS2)**：定义发送点的位置。它代表 CAN 标准的 PHASE_SEG2。其持续长度可以在 1 到 8 个时间片之间调整，但也可以自动缩短，以补偿负相位漂移。

再同步跳转宽度 (SJW) 定义位段加长或缩短的上限。它可以在 1 到 4 个时间片之间调整。

有效边沿是指一个位时间内总线电平从显性到隐性的第一次转换（前提是控制器本身不发送隐性位）。

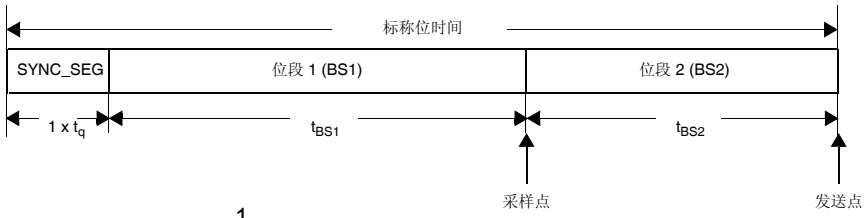
如果在 BS1 而不是 SYNC_SEG 中检测到有效边沿，则 BS1 会延长最多 SJW，以便延迟采样点。

相反地，如果在 BS2 而不是 SYNC_SEG 中检测到有效边沿，则 BS2 会缩短最多 SJW，以便提前发送点。

为了避免编程错误，位时序寄存器 (CAN_BTR) 只能在器件处于待机模式时进行配置。

注意： *有关 CAN 位时序和再同步机制的详细说明，请参见 ISO 11898 标准。*

图 235. 位时序



$$BaudRate = \frac{1}{NominalBitTime}$$

$$NominalBitTime = 1 \times t_q + t_{BS1} + t_{BS2}$$

其中：

$t_{BS1} = t_q$ x (TS1[3:0] + 1)，

$t_{BS2} = t_q$ x (TS2[2:0] + 1)，

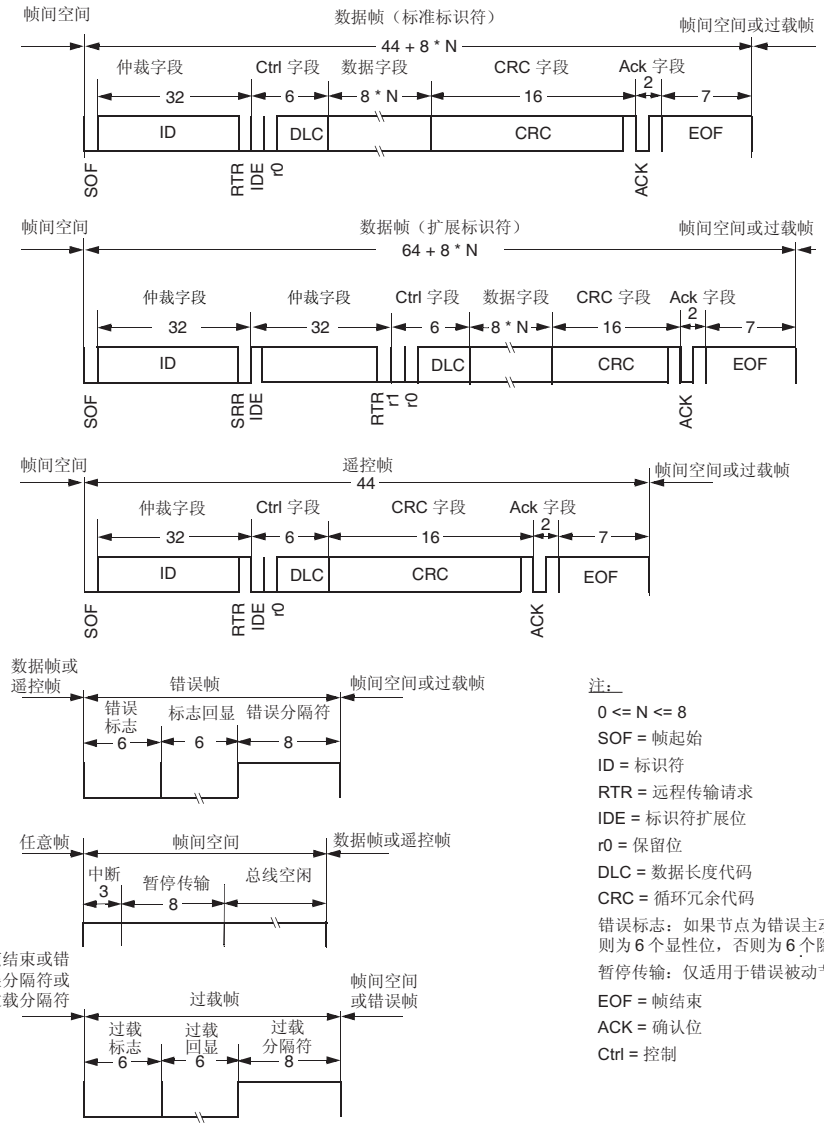
t_q = (BRP[9:0] + 1) x t_{PCLK}

其中 t_q 为时间片

t_{PCLK} = APB 时钟的时间周期，

BRP[9:0]、TS1[3:0] 和 TS2[2:0] 在 CAN_BTR 寄存器中定义。

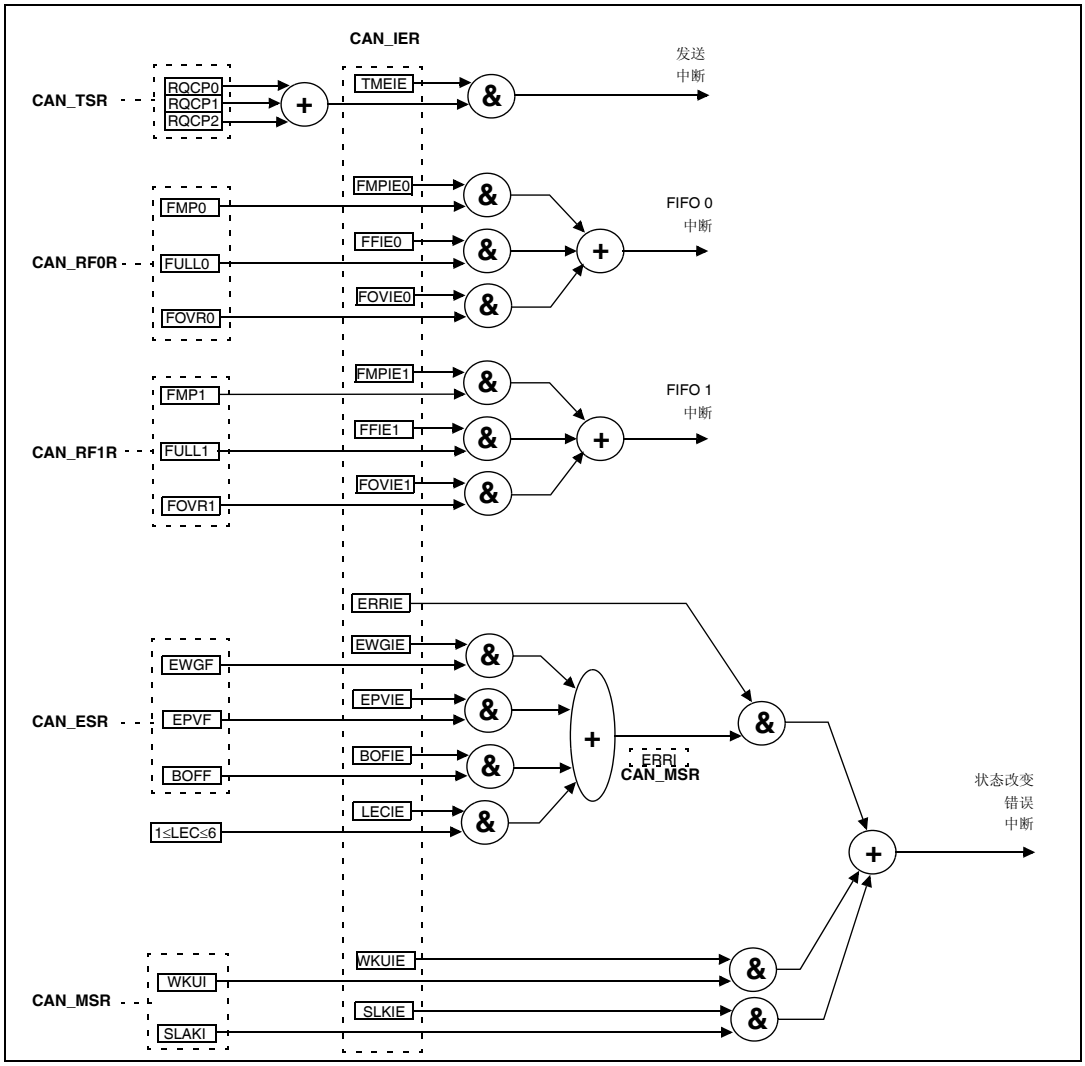
图 236. CAN 帧


帧间空间
数据帧（标准标识符）
帧间空间或过载帧
44 + 8 * N
仲裁字段
Ctrl 字段
数据字段
CRC 字段
Ack 字段
32
6
8 * N
16
2
7
ID
DLC
CRC
EOF
SOF
RTR
IDE
r0
ACK
帧间空间
数据帧（扩展标识符）
帧间空间或过载帧
64 + 8 * N
仲裁字段
仲裁字段
Ctrl 字段
数据字段
CRC 字段
Ack 字段
32
32
6
8 * N
16
2
7
ID
DLC
CRC
EOF
SOF
SRR
IDE
RTR
r1
r0
ACK
帧间空间
遥控帧
帧间空间或过载帧
44
仲裁字段
Ctrl 字段
CRC 字段
Ack 字段
32
6
16
2
7
ID
DLC
CRC
EOF
SOF
RTR
IDE
r0
ACK
数据帧或
遥控帧
错误帧
帧间空间或过载帧
错误
标志
标志回显
错误分隔符
6
6
8
任意帧
帧间空间
数据帧或遥控帧
中断
暂停传输
总线空闲
3
8
帧结束或错
误分隔符或
过载分隔符
过载帧
帧间空间
或错误帧
过载
标志
过载
回显
过载
分隔符
6
6
8
注：
0 <= N <= 8
SOF = 帧起始
ID = 标识符
RTR = 远程传输请求
IDE = 标识符扩展位
r0 = 保留位
DLC = 数据长度代码
CRC = 循环冗余代码
错误标志：如果节点为错误主动节点，
则为 6 个显性位，否则为 6 个隐性位。
暂停传输：仅适用于错误被动节点。
EOF = 帧结束
ACK = 确认位
Ctrl = 控制
ai15154

24.8 bxCAN 中断

bxCAN 共有四个专用的中断向量。每个中断源均可通过 CAN 中断使能寄存器 (CAN_IER) 来单独地使能或禁止。

图 237. 事件标志与中断产生



- **发送中断**可由以下事件产生：
  - 发送邮箱 0 变为空，CAN_TSR 寄存器的 RQCP0 位置 1。
  - 发送邮箱 1 变为空，CAN_TSR 寄存器的 RQCP1 位置 1。
  - 发送邮箱 2 变为空，CAN_TSR 寄存器的 RQCP2 位置 1。
- **FIFO 0 中断**可由以下事件产生：
  - 接收到新消息，CAN_RF0R 寄存器的 FMP0 位不是“00”。
  - FIFO0 满，CAN_RF0R 寄存器的 FULL0 位置 1。
  - FIFO0 上溢，CAN_RF0R 寄存器的 FOVR0 位置 1。

- **FIFO 1 中断**可由以下事件产生：
  - 接收到新消息，CAN_RF1R 寄存器的 FMP1 位不是“00”。
  - FIFO1 满，CAN_RF1R 寄存器的 FULL1 位置 1。
  - FIFO1 上溢，CAN_RF1R 寄存器的 FOVR1 位置 1。
- **错误和状态改变中断**可由以下事件产生：
  - 错误状况，有关错误状况的更多详细信息，请参见 CAN 错误状态寄存器 (CAN_ESR)。
  - 唤醒状况，CAN Rx 信号上监测到 SOF。
  - 进入睡眠模式。

24.9 CAN 寄存器

外设寄存器必须按字（32 位）进行访问。

24.9.1 寄存器访问保护

错误访问某些配置寄存器可能会导致硬件暂时性地干扰整个 CAN 网络。因此，只有在 CAN 硬件处于初始化模式时，才可以通过软件修改 CAN_BTR 寄存器。

尽管传输错误的数据不会引发 CAN 网络级别的故障，但可能会对应用程序造成严重干扰。发送邮箱只能在处于空状态时通过软件进行修改，请参见*图 229：发送邮箱状态*。

可以通过停用相应筛选器组或将 FINIT 位置 1 来修改筛选器值。此外，只有在 CAN_FMR 寄存器的筛选器初始化模式位置 1 (FINIT=1) 时，才能对 CAN_FMxR、CAN_FSxR 和 CAN_FFAR 寄存器中的筛选器配置（大小、模式和 FIFO 分配）进行修改。

24.9.2 CAN 控制和状态寄存器

有关寄存器说明中使用的缩写，请参见*第 1.1 节*。

CAN 主控制寄存器 (CAN_MCR)

CAN master control register

偏移地址：0x00
复位值：0x0001 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															DBF
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	Reserved							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

位 31:17 保留，必须保持复位值。

位 16 **DBF**：调试冻结 (Debug freeze)

0：调试期间 CAN 处于工作状态。

1：调试期间 CAN 处于接收/发送冻结状态。接收 FIFO 仍可正常访问/控制。

位 15 **RESET**：bxCAN 软件主复位 (bxCAN software master reset)

0：正常工作。

1：强制 bxCAN 进行主复位 -> 复位后激活睡眠模式（FMP 位和 CAN_MCR 寄存器初始化为复位值）。此位自动复位为 0。

位 14:8 保留，必须保持复位值。

位 7 **TTCM**：时间触发通信模式 (Time triggered communication mode)

0：禁止时间触发通信模式。

1：使能时间触发通信模式。

注意：有关时间触发通信模式的更多信息，请参见第 24.7.2 节：时间触发通信模式。

位 6 **ABOM**：自动的总线关闭管理 (Automatic bus-off management)

此位控制 CAN 硬件在退出总线关闭状态时的行为。

0：在软件发出请求后，一旦监测到 128 次连续 11 个隐性位，并且软件将 CAN_MCR 寄存器的 INRQ 位先置 1 再清零，即退出总线关闭状态。

1：一旦监测到 128 次连续 11 个隐性位，即通过硬件自动退出总线关闭状态。

有关总线关闭状态的详细信息，请参见*第 24.7.6 节：错误管理*。

位 5 **AWUM**：自动唤醒模式 (Automatic wakeup mode)

此位控制 CAN 硬件在睡眠模式下接收到消息时的行为。

0：在软件通过将 CAN_MCR 寄存器的 SLEEP 位清零发出请求后，退出睡眠模式。

1：一旦监测到 CAN 消息，即通过硬件自动退出睡眠模式。

CAN_MCR 寄存器的 SLEEP 位和 CAN_MCR 寄存器的 SLAK 位由硬件清零。

位 4 **NART**：禁止自动重发送 (No automatic retransmission)

0：CAN 硬件将自动重发送消息，直到根据 CAN 标准消息发送成功。

1：无论发送结果如何（成功、错误或仲裁丢失），消息均只发送一次。

位 3 **RFLM**：接收 FIFO 锁定模式 (Receive FIFO locked mode)

0：接收 FIFO 上溢后不锁定。接收 FIFO 装满后，下一条传入消息将覆盖前一条消息。

1：接收 FIFO 上溢后锁定。接收 FIFO 装满后，下一条传入消息将被丢弃。

位 2 **TXFP**：发送 FIFO 优先级 (Transmit FIFO priority)

此位用于控制在几个邮箱同时挂起时的发送顺序。

0：优先级由消息标识符确定

1：优先级由请求顺序（时间顺序）确定

位 1 **SLEEP**：睡眠模式请求 (Sleep mode request)

此位由软件置 1，用于请求 CAN 硬件进入睡眠模式。一旦当前 CAN 活动（发送或接收 CAN 帧）结束，即进入睡眠模式。

此位由软件清零时，将退出睡眠模式。

当 AWUM 位置 1 以及在 CAN RX 信号上检测到 SOF 位时，硬件即将此位清零。

复位后，此位将置 1 - CAN 启动睡眠模式。

位 0 **INRQ**：初始化请求 (Initialization request)

软件通过将此位清零，来将硬件切换到正常模式。一旦在 Rx 信号上监测到连续 11 个隐性位，CAN 硬件即完成同步并准备进行发送和接收。硬件通过将 CAN_MSR 寄存器的 INAK 位清零来指示此事件。

软件通过将此位置 1 来请求 CAN 硬件进入初始化模式。一旦软件将 INRQ 位置 1，CAN 硬件将等待当前 CAN 活动（发送或接收）结束，然后进入初始化模式。硬件通过将 CAN_MSR 寄存器的 INAK 位置 1 来指示此事件。

CAN 主状态寄存器 (CAN_MSR)

CAN master status register

偏移地址：0x04
复位值：0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved.				RX	SAMP	RXM	TXM	Reserved			SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

位 31:12 保留，必须保持复位值。

位 11 **RX**：CAN Rx 信号 (CAN Rx signal)
监视 **CAN_RX** 引脚的实际值。

位 10 **SAMP**：上一个采样点 (Last sample point)
上一个采样点的 RX 值（当前接收的位值）。

位 9 **RXM**：接收模式 (Receive mode)
CAN 硬件当前为接收器。

位 8 **TXM**：发送模式 (Transmit mode)
CAN 硬件当前为发送器。

位 7:5 保留，必须保持复位值。

位 4 **SLAKI**：睡眠确认中断 (Sleep acknowledge interrupt)
当 SLKIE=1 时，硬件将此位置 1，以指示 bxCAN 已经进入睡眠模式。如果 CAN_IER 寄存器的 SLKIE 位置 1，则当此位置 1 后将产生状态改变中断。
SLAK 清零后，此位由软件或硬件清零。
注意：SLKIE=0 时，无法对 SLAKI 进行轮询。在这种情况下，可以轮询 SLAK 位。

位 3 **WKUI**：唤醒中断 (Wakeup interrupt)
此位由硬件置 1，用于指示在 CAN 硬件处于睡眠模式期间检测到一个 SOF 位。如果 CAN_IER 寄存器的 WKUIE 位置 1，则当此位置 1 后将产生状态改变中断。
此位由软件清零。

位 2 **ERRI**：错误中断 (Error interrupt)
如果在检测到错误时 CAN_ESR 的一个位置 1，并且使能 CAN_IER 中的相应中断，则硬件将此位置 1。如果 CAN_IER 寄存器的 ERRIE 位置 1，则当此位置 1 后将产生状态改变中断。
此位由软件清零。

位 1 **SLAK**：睡眠确认 (Sleep acknowledge)
此位由硬件置 1，用于向软件指示 CAN 硬件此时处于睡眠模式。此位可确认软件的睡眠模式请求（CAN_MCR 寄存器的 SLEEP 位置 1）。
CAN 硬件退出睡眠模式（以在 CAN 总线上进行同步）时，此位由硬件清零。要进行同步，硬件必须在 CAN RX 信号上监测到由 11 个连续隐性位组成的一个序列。
注意：CAN_MCR 寄存器的 SLEEP 位清零时，将触发退出睡眠模式程序。有关 SLEEP 位清零的详细信息，请参阅 CAN_MCR 寄存器 AWUM 位的说明。

位 0 **INAK**：初始化确认 (Initialization acknowledge)
此位由硬件置 1，用于向软件指示 CAN 硬件此时处于初始化模式。此位可确认软件的初始化请求（CAN_MCR 寄存器的 INRQ 位置 1）。
CAN 硬件退出初始化模式（以在 CAN 总线上进行同步）时，此位由硬件清零。要进行同步，硬件必须在 CAN RX 信号上监测到由 11 个连续隐性位组成的一个序列。

CAN 发送状态寄存器 (CAN_TSR)

CAN transmit status register

偏移地址：0x08
复位值：0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]		ABRQ2	Reserved			TERR2	ALST2	TXOK2	RQCP2
r	r	r	r	r	r	r	r	rs				rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRQ1	Reserved Res.			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Reserved			TERR0	ALST0	TXOK0	RQCP0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

位 31 **LOW2**：邮箱 2 最低优先级标志 (Lowest priority flag for mailbox 2)
当多个邮箱挂起等待发送且邮箱 2 优先级最低时，此位由硬件置 1。

位 30 **LOW1**：邮箱 1 最低优先级标志 (Lowest priority flag for mailbox 1)
当多个邮箱挂起等待发送且邮箱 1 优先级最低时，此位由硬件置 1。

位 29 **LOW0**：邮箱 0 最低优先级标志 (Lowest priority flag for mailbox 0)
当多个邮箱挂起等待发送且邮箱 0 优先级最低时，此位由硬件置 1。
注意：当只有一个邮箱挂起时，LOW[2:0] 位置为零。

位 28 **TME2**：发送邮箱 2 空 (Transmit mailbox 2 empty)
当邮箱 2 没有挂起的发送请求时，此位由硬件置 1。

位 27 **TME1**：发送邮箱 1 空 (Transmit mailbox 1 empty)
当邮箱 1 没有挂起的发送请求时，此位由硬件置 1。

位 26 **TME0**：发送邮箱 0 空 (Transmit mailbox 0 empty)
当邮箱 0 没有挂起的发送请求时，此位由硬件置 1。

位 25:24 **CODE[1:0]**：邮箱代码 (Mailbox code)
如果至少一个发送邮箱空闲，代码值等于下一个空闲发送邮箱的编号。
如果所有发送邮箱均挂起，则代码值等于优先级最低的发送邮箱的编号。

位 23 **ABRQ2**：邮箱 2 中止请求 (Abort request for mailbox 2)
由软件置 1，用于中止相应邮箱的发送请求。
邮箱变为空后，此位由硬件清零。
邮箱未挂起等待发送时，将此位置 1 没有任何作用。

位 22:20 保留，必须保持复位值。

位 19 **TERR2**：邮箱 2 发送错误 (Transmission error of mailbox 2)
如果上一次发送因错误而失败，此位将置 1。

位 18 **ALST2**：邮箱 2 仲裁丢失 (Arbitration lost for mailbox 2)
如果上一次发送因仲裁丢失而失败，此位将置 1。

位 17 **TXOK2**：邮箱 2 发送成功 (Transmission OK of mailbox 2)
每次发送尝试后，硬件都将更新此位。
0：上一次发送失败
1：上一次发送成功
当邮箱 2 的发送请求成功完成时，此位由硬件置 1。请参见*图 229*。

位 16 **RQCP2**：邮箱 2 请求完成 (Request completed mailbox 2)
最后一个请求（发送或中止）执行完毕时，由硬件置 1。
由软件通过写入“1”清零，或是在发生发送请求（CAN_TMID2R 寄存器中的 TXRQ2 位置 1）时由硬件清零。
如果将此位清零，邮箱 2 的所有状态位（TXOK2、ALST2 和 TERR2）都将清零。

位 15 **ABRQ1**：邮箱 1 中止请求 (Abort request for mailbox 1)
由软件置 1，用于中止相应邮箱的发送请求。
邮箱变为空后，此位由硬件清零。
邮箱未挂起等待发送时，将此位置 1 没有任何作用。

位 14:12 保留，必须保持复位值。

位 11 **TERR1**：邮箱 1 发送错误 (Transmission error of mailbox 1)
如果上一次发送因错误而失败，此位将置 1。

位 10 **ALST1**：邮箱 1 仲裁丢失 (Arbitration lost for mailbox 1)
如果上一次发送因仲裁丢失而失败，此位将置 1。

位 9 **TXOK1**：邮箱 1 发送成功 (Transmission OK of mailbox 1)
每次发送尝试后，硬件都将更新此位。
0：上一次发送失败
1：上一次发送成功
当邮箱 1 的发送请求成功完成时，此位由硬件置 1。请参见*图 229*。

位 8 **RQCP1**：邮箱 1 请求完成 (Request completed mailbox 1)
最后一个请求（发送或中止）执行完毕时，由硬件置 1。
由软件通过写入“1”清零，或是在发生发送请求（CAN_TI1R 寄存器中的 TXRQ1 位）时由硬件清零。
如果将此位清零，邮箱 1 的所有状态位（TXOK1、ALST1 和 TERR1）都将清零。

位 7 **ABRQ0**：邮箱 0 中止请求 (Abort request for mailbox 0)
由软件置 1，用于中止相应邮箱的发送请求。
邮箱变为空后，此位由硬件清零。
邮箱未挂起等待发送时，将此位置 1 没有任何作用。

位 6:4 保留，必须保持复位值。

位 3 **TERR0**：邮箱 0 发送错误 (Transmission error of mailbox 0)
如果上一次发送因错误而失败，此位将置 1。

位 2 **ALST0**：邮箱 0 仲裁丢失 (Arbitration lost for mailbox 0)
如果上一次发送因仲裁丢失而失败，此位将置 1。

位 1 **TXOK0**：邮箱 0 发送成功 (Transmission OK of mailbox 0)
每次发送尝试后，硬件都将更新此位。
0：上一次发送失败
1：上一次发送成功
当邮箱 1 的发送请求成功完成时，此位由硬件置 1。请参见*图 229*。

位 0 **RQCP0**：邮箱 0 请求完成 (Request completed mailbox 0)
最后一个请求（发送或中止）执行完毕时，由硬件置 1。
由软件通过写入“1”清零，或是在发生发送请求（CAN_TI0R 寄存器的 TXRQ0 位置 1）时由硬件清零。
如果将此位清零，邮箱 0 的所有状态位（TXOK0、ALST0 和 TERR0）都将清零。

CAN 接收 FIFO 0 寄存器 (CAN_RF0R)

CAN receive FIFO 0 register

偏移地址：0x0C
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFOM0	FOVR0	FULL0	Res.	FMP0[1:0]	
										rs	rc_w1	rc_w1		r	r

位 31:6 保留，必须保持复位值。

位 5 **RFOM0**：释放 FIFO 0 输出邮箱 (Release FIFO 0 output mailbox)
由软件置 1，用于释放 FIFO 的输出邮箱。FIFO 中至少有一条消息挂起时，才能释放输出邮箱。FIFO 为空时，将此位置 1 没有任何作用。如果 FIFO 中至少有两条消息挂起，软件必须释放输出邮箱，才能访问下一条消息。
输出邮箱释放后，此位由硬件清零。

位 4 **FOVR0**：FIFO 0 上溢 (FIFO 0 overrun)
FIFO 填满时，如果接收到新消息并且通过筛选器，此位将由硬件置 1。
此位由软件清零。

位 3 **FULL0**：FIFO 0 满(FIFO 0 full)
FIFO 中存储三条消息后，由硬件置 1。
此位由软件清零。

位 2 保留，必须保持复位值。

位 1:0 **FMP0[1:0]**：FIFO 0 消息挂起 (FIFO 0 message pending)
这些位用于指示接收 FIFO 中挂起的消息数。
硬件每向 FIFO 存储一条新消息，FMP 即会增加。软件每次通过将 RFOM0 位置 1 来释放输出邮箱，FMP 即会减小。

CAN 接收 FIFO 1 寄存器 (CAN_RF1R)

CAN receive FIFO 1 register

偏移地址：0x10
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFOM1	FOVR1	FULL1	Res.	FMP1[1:0]	
										rs	rc_w1	rc_w1		r	r

位 31:6 保留，必须保持复位值。

位 5 **RFOM1**：释放 FIFO 1 输出邮箱 (Release FIFO 1 output mailbox)
由软件置 1，用于释放 FIFO 的输出邮箱。FIFO 中至少有一条消息挂起时，才能释放输出邮箱。FIFO 为空时，将此位置 1 没有任何作用。如果 FIFO 中至少有两条消息挂起，软件必须释放输出邮箱，才能访问下一条消息。
输出邮箱释放后，此位由硬件清零。

位 4 **FOVR1**：FIFO 1 上溢 (FIFO 1 overrun)
FIFO 填满时，如果接收到新消息并且通过筛选器，此位将由硬件置 1。
此位由软件清零。

位 3 **FULL1**：FIFO 1 满 (FIFO 1 full)
FIFO 中存储三条消息后，由硬件置 1。
此位由软件清零。

位 2 保留，必须保持复位值。

位 1:0 **FMP1[1:0]**：FIFO 1 消息挂起 (FIFO 1 message pending)
这些位用于指示接收 FIFO1 中挂起的消息数。
硬件每向 FIFO1 存储一条新消息，FMP1 即会增加。软件每次通过将 RFOM1 位置 1 来释放输出邮箱，FMP 即会减小。

CAN 中断使能寄存器 (CAN_IER)

CAN interrupt enable register

偏移地址：0x14
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														SLKIE	WKUIE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Reserved			LEC IE	BOF IE	EPV IE	EWG IE	Res.	FOV IE1	FF IE1	FMP IE1	FOV IE0	FF IE0	FMP IE0	TME IE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

位 31:18 保留，必须保持复位值。

位 17 **SLKIE**：睡眠中断使能 (Sleep interrupt enable)
0：SLAKI 位置 1 时不产生中断。
1：SLAKI 位置 1 时产生中断。

位 16 **WKUIE**：唤醒中断使能 (Wakeup interrupt enable)
0：WKUI 置 1 时不产生中断。
1：WKUI 位置 1 时产生中断。

位 15 **ERRIE**：错误中断使能 (Error interrupt enable)
0：CAN_ESR 中有挂起的错误状况时，不会产生中断。
1：CAN_ESR 中有挂起的错误状况时，会产生中断。

位 14:12 保留，必须保持复位值。

位 11 **LECIE**：上一个错误代码中断使能 (Last error code interrupt enable)
0：如果在检测到错误后硬件将 LEC[2:0] 中的错误代码置 1，则不会将 ERRI 位置 1。
1：如果在检测到错误后硬件将 LEC[2:0] 中的错误代码置 1，会将 ERRI 位置 1。

位 10 **BOFIE**：总线关闭中断使能 (Bus-off interrupt enable)
0：BOFF 置 1 时，不会将 ERRI 位置 1。
1：BOFF 置 1 时，将 ERRI 位置 1。

位 9 **EPVIE**：错误被动中断使能 (Error passive interrupt enable)
0：EPVF 置 1 时，不会将 ERRI 位置 1。
1：EPVF 置 1 时，将 ERRI 位置 1。

位 8 **EWGIE**：错误警告中断使能 (Error warning interrupt enable)
0：EWGF 置 1 时，不会将 ERRI 位置 1。
1：EWGF 置 1 时，将 ERRI 位置 1。

位 7 保留，必须保持复位值。

位 6 **FOVIE1**：FIFO 上溢中断使能 (FIFO overrun interrupt enable)
0：FOVR 置 1 时不产生中断。
1：FOVR 置 1 时产生中断。

位 5 **FFIE1**：FIFO 满中断使能 (FIFO full interrupt enable)
0：FULL 位置 1 时不产生中断。
1：FULL 位置 1 时产生中断。

位 4 **FMPIE1**：FIFO 消息挂起中断使能 (FIFO message pending interrupt enable)
0：FMP[1:0] 位的状态不是 00b 时，不产生中断。
1：FMP[1:0] 位的状态不是 00b 时，产生中断。

位 3 **FOVIE0**：FIFO 上溢中断使能 (FIFO overrun interrupt enable)
0：FOVR 位置 1 时不产生中断。
1：FOVR 位置 1 时产生中断。

位 2 **FFIE0**：FIFO 满中断使能 (FIFO full interrupt enable)
0：FULL 位置 1 时不产生中断。
1：FULL 位置 1 时产生中断。

位 1 **FMPIE0**:FIFO 消息挂起中断使能 (FIFO message pending interrupt enable)
0：FMP[1:0] 位的状态不是 00b 时，不产生中断。
1：FMP[1:0] 位的状态不是 00b 时，产生中断。

位 0 **TMEIE**：发送邮箱空中断使能 (Transmit mailbox empty interrupt enable)

0：RQCPx 位置 1 时不产生中断。

1：RQCPx 位置 1 时产生中断。

注意：请参见第 24.8 节：bxCAN 中断。

CAN 错误状态寄存器 (CAN_ESR)

CAN error status register

偏移地址：0x18

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7:0]								TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								LEC[2:0]				Res.	BOFF	EPVF	EWGF
								rw	rw	rw			r	r	r

位 31:24 **REC[7:0]**：接收错误计数器 (Receive error counter)

CAN 协议故障隔离机制的实施部分。如果接收期间发生错误，该计数器按 1 或 8 递增，具体取决于 CAN 标准所定义的错误状况。每次成功接收后，该计数器按 1 递减，如果其数值大于 128，则复位为 120。计数器值超过 127 时，CAN 控制器进入错误被动状态。

位 23:16 **TEC[7:0]**：9 位发送错误计数器最低有效字节 (Least significant byte of the 9-bit transmit error counter)

CAN 协议故障隔离机制的实施部分。

位 15:7 保留，必须保持复位值。

位 6:4 **LEC[2:0]**：上一个错误代码 (Last error code)

该字段由硬件置 1，其中的代码指示 CAN 总线上检测到的上一个错误的错误状况。如果消息成功传送（接收或发送）且未发生错误，该字段将清为“0”。

LEC[2:0] 位可由软件置为 0b111 值。这些位由硬件更新，以指示当前通信状态。

000：无错误
001：填充错误
010：格式错误
011：确认错误
100：位隐性错误
101：位显性错误
110：CRC 错误
111：由软件置 1

位 3 保留，必须保持复位值。

位 2 **BOFF**：总线关闭标志 (Bus-off flag)

此位由硬件在进入睡眠状态时置 1。TEC 上溢（超过 255）时，进入总线关闭状态，请参见*第 620 页的第 24.7.6 节*。

位 1 **EPVF**：错误被动标志 (Error passive flag)

达到错误被动极限（接收错误计数器或发送错误计数器 > 127）时，此位由硬件置 1。

位 0 **EWGF**：错误警告标志 (Error warning flag)

达到警告极限时，此位由硬件置 1（接收错误计数器或发送错误计数器≥96）。

CAN 位时序寄存器 (CAN_BTR)

CAN bit timing register

偏移地址：0x1C
复位值：0x0123 0000

只有 CAN 硬件处于初始化模式时，才能由软件访问此寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	Reserved				SJW[1:0]		Res.	TS2[2:0]			TS1[3:0]			
rw	rw					rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						BRP[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31 **SILM**：静默模式（调试）(Silent mode (debug))
0：正常工作
1：静默模式

位 30 **LBKM**：环回模式（调试）(Loop back mode (debug))
0：禁止环回模式
1：使能环回模式

位 29:26 保留，必须保持复位值。

位 25:24 **SJW[1:0]**：再同步跳转宽度 (Resynchronization jump width)
这些位定义 CAN 硬件在执行再同步时最多可以将位加长或缩短的时间片数目。
$t_{RJW} = t_{CAN} \times (SJW[1:0] + 1)$

位 23 保留，必须保持复位值。

位 22:20 **TS2[2:0]**：时间段 2 (Time segment 2)
这些位定义时间段 2 中的时间片数目。
$t_{BS2} = t_{CAN} \times (TS2[2:0] + 1)$

位 19:16 **TS1[3:0]**：时间段 1 (Time segment 1)
这些位定义时间段 1 中的时间片数目。
$t_{BS1} = t_{CAN} \times (TS1[3:0] + 1)$
有关位时序的详细信息，请参见*第 621 页的第 24.7.7 节：位时序*。

位 15:10 保留，必须保持复位值。

位 9:0 **BRP[9:0]**：波特率预分频器 (Baud rate prescaler)
这些位定义一个时间片的长度。
$t_q = (BRP[9:0]+1) \times t_{PCLK}$

24.9.3 CAN 邮箱寄存器

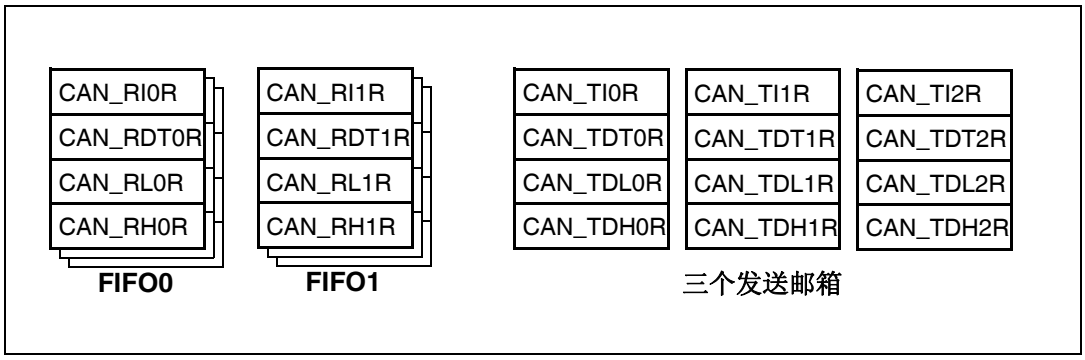
本章介绍发送和接收邮箱的寄存器。有关详细的寄存器映射，请参见*第 619 页的第 24.7.5 节：消息存储*。

除了以下情况外，发送邮箱和接收邮箱使用相同的寄存器：

- CAN_RDTxR 寄存器中的 FMI 字段。
- 接收邮箱始终处于写保护状态。
- 发送邮箱仅在为空时才处于可写状态（CAN_TSR 寄存器中的相应 TME 位置 1）。

发送邮箱共有 3 个，接收邮箱共有 2 个。每个接收邮箱允许访问一个深度为 3 级的 FIFO，访问仅限于 FIFO 中最早接收到的消息。

每个邮箱由 4 个寄存器组成。



CAN 发送邮箱标识符寄存器 (CAN_TIxR) (x=0..2)

CAN TX mailbox identifier register

偏移地址：0x180、0x190、0x1A0
复位值：0xXXXX XXXX（位 0 除外，TXRQ = 0）

当邮箱处于发送挂起状态（TMEx 复位）时，所有发送寄存器均为写保护状态。

该寄存器还会进行发送请求控制（位 0）- 复位值 0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	TXRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:21 **STID[10:0]/EXID[28:18]**：标准标识符或扩展标识符 (Standard identifier or extended identifier)
标准标识符或扩展标识符的 MSB（取决于 IDE 位的值）。

位 20:3 **EXID[17:0]**：扩展标识符 (Extended identifier)
扩展标识符的 LSB。

位 2 **IDE**：标识符扩展 (Identifier extension)
此位用于定义邮箱中消息的标识符类型。
0：标准标识符。
1：扩展标识符。

位 1 **RTR**：远程发送请求 (Remote transmission request)
0：数据帧
1：遥控帧

位 0 **TXRQ**：发送邮箱请求 (Transmit mailbox request)
由软件置 1，用于请求发送相应邮箱的内容。
邮箱变为空后，此位由硬件清零。

CAN 邮箱数据长度控制和时间戳寄存器 (CAN_TDTxR) (x=0..2)

CAN mailbox data length control and time stamp register

当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

偏移地址：0x184、0x194、0x1A4
复位值：0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							TGT	Reserved				DLC[3:0]			
							rw					rw	rw	rw	rw

位 31:16 **TIME[15:0]**：消息时间戳 (Message time stamp)
此字段包含在进行 SOF 发送时所捕获的 16 位定时器值。

位 15:9 保留，必须保持复位值。

位 8 **TGT**：发送全局时间 (Transmit global time)
只有硬件处于时间触发通信模式（CAN_MCR 寄存器的 TTCM 位置 1）时，此位才会激活。
0：不发送时间戳 TIME[15:0]。
1：在 8 字节消息的最后两个数据字节中发送时间戳 TIME[15:0] 的值。数据字节 7 对应 TIME[7:0]，数据字节 6 对应 TIME[15:8]，该值将替换 CAN_TDHxR[31:16] 寄存器（DATA6[7:0] 和 DATA7[7:0]）中写入的数据。DLC 必须编程为 8，才能通过 CAN 总线发送这两个字节。

位 7:4 保留，必须保持复位值。

位 3:0 **DLC[3:0]**：数据长度代码 (Data length code)
该字段定义数据帧或遥控帧请求中的数据字节数。
一条消息可以包含 0 到 8 个数据字节，具体取决于 DLC 字段的值。

CAN 邮箱数据低位寄存器 (CAN_TDLxR) (x=0..2)

CAN mailbox data low register

当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

偏移地址：0x188、0x198、0x1A8
复位值：0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 **DATA3[7:0]**：数据字节 3 (Data byte 3)
消息的数据字节 3。

位 23:16 **DATA2[7:0]**：数据字节 2 (Data byte 2)
消息的数据字节 2。

位 15:8 **DATA1[7:0]**：数据字节 1 (Data byte 1)
消息的数据字节 1。

位 7:0 **DATA0[7:0]**：数据字节 0 (Data byte 0)
消息的数据字节 0。
一条消息可以包含 0 到 8 个数据字节，从字节 0 开始。

CAN 邮箱数据高位寄存器 (CAN_TDHxR) (x=0..2)

CAN mailbox data high register

当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

偏移地址：0x18C、0x19C、0x1AC
复位值：0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:24 **DATA7[7:0]**：数据字节 7 (Data byte 7)
消息的数据字节 7。
注意：如果 TTCG 以及此消息的 TGT 为激活状态，则 DATA7 和 DATA6 将以时间戳值替换。

位 23:16 **DATA6[7:0]**：数据字节 6 (Data byte 6)
消息的数据字节 6。

位 15:8 **DATA5[7:0]**：数据字节 5 (Data byte 5)
消息的数据字节 5。

位 7:0 **DATA4[7:0]**：数据字节 4 (Data byte 4)
消息的数据字节 4。

CAN 接收 FIFO 邮箱标识符寄存器 (CAN_RIxR) (x=0..1)

CAN receive FIFO mailbox identifier register

偏移地址：0x1B0、0x1C0
复位值：0xXXXX XXXX

所有接收寄存器均受到写保护。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

位 31:21 **STID[10:0]/EXID[28:18]**：标准标识符或扩展标识符 (Standard identifier or extended identifier)
标准标识符或扩展标识符的 MSB（取决于 IDE 位的值）。

位 20:3 **EXID[17:0]**：扩展标识符 (Extended identifier)
扩展标识符的 LSB。

位 2 **IDE**：标识符扩展 (Identifier extension)
此位用于定义邮箱中消息的标识符类型。
0：标准标识符。
1：扩展标识符。

位 1 **RTR**：远程发送请求 (Remote transmission request)
0：数据帧
1：遥控帧

位 0 保留，必须保持复位值。

CAN 接收 FIFO 邮箱数据长度控制和时间戳寄存器 (CAN_RDTxR) (x=0..1)

CAN receive FIFO mailbox data length control and time stamp register

偏移地址：0x1B4、0x1C4
复位值：0xXXXX XXXX

所有接收寄存器均受到写保护。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMI[7:0]								Reserved				DLC[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

位 31:16 **TIME[15:0]**：消息时间戳 (Message time stamp)
此字段包含在进行 SOF 检测时所捕获的 16 位定时器值。

位 15:8 **FMI[7:0]**：筛选器匹配索引 (Filter match index)
该寄存器包含筛选器索引，邮箱中存储的消息需要经过该筛选器。有关标识符筛选的更多详细信息，请参见*第 616 页的第 24.7.4 节：标识符筛选*的**筛选器匹配索引**一节。

位 7:4 保留，必须保持复位值。

位 3:0 **DLC[3:0]**：数据长度代码 (Data length code)
该字段定义一个数据帧所包含的数据字节数（0 到 8）。如果是远程帧请求，则为 0。

CAN 接收 FIFO 邮箱数据低位寄存器 (CAN_RDLxR) (x=0..1)

CAN receive FIFO mailbox data low register

当邮箱未处于空状态时，该寄存器的所有位均为写保护状态。

偏移地址：0x1B8、0x1C8
复位值：0xXXXX XXXX

所有接收寄存器均受到写保护。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:24 **DATA3[7:0]**：数据字节 3 (Data byte 3)
消息的数据字节 3。

位 23:16 **DATA2[7:0]**：数据字节 2 (Data byte 2)
消息的数据字节 2。

位 15:8 **DATA1[7:0]**：数据字节 1 (Data byte 1)
消息的数据字节 1。

位 7:0 **DATA0[7:0]**：数据字节 0 (Data byte 0)
消息的数据字节 0。
一条消息可以包含 0 到 8 个数据字节，从字节 0 开始。

CAN 接收 FIFO 邮箱数据高位寄存器 (CAN_RDHxR) (x=0..1)

CAN receive FIFO mailbox data high register

偏移地址：0x1BC、0x1CC
复位值：0xXXXX XXXX

所有接收寄存器均受到写保护。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位 31:24 **DATA7[7:0]**：数据字节 7 (Data byte 7)
消息的数据字节 3。

位 23:16 **DATA6[7:0]**：数据字节 6 (Data byte 6)
消息的数据字节 2。

位 15:8 **DATA5[7:0]**：数据字节 5 (Data byte 5)
消息的数据字节 1。

位 7:0 **DATA4[7:0]**：数据字节 4 (Data byte 4)
消息的数据字节 0。

24.9.4 CAN 筛选器寄存器

CAN 筛选器主寄存器 (CAN_FMR)

CAN filter master register

偏移地址：0x200
复位值：0x2A1C 0E01

此寄存器的所有位均由软件置 1 和清零。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CAN2SB[5:0]						Reserved							FINIT
		rw	rw	rw	rw	rw	rw								rw

位 31:14 保留，必须保持复位值。

位 13:8 **CAN2SB[5:0]**：CAN2 起始存储区 (CAN2 start bank)

这些位将由软件置 1 和清零。它们为处于 0 到 27 范围内的 CAN2 接口（从模式）定义起始存储区。

注意：CAN2SB[5:0] = 28d 时，可以使用 CAN1 的所有筛选器。

CAN2SB[5:0] 设置为 0 时，不会为 CAN1 分配任何筛选器。

位 7:1 保留，必须保持复位值。

位 0 **FINIT**：筛选器初始化模式 (Filter init mode)

筛选器组的初始化模式

0：筛选器工作模式。

1：筛选器初始化模式。

CAN 筛选器模式寄存器 (CAN_FM1R)

CAN filter mode register

偏移地址：0x204

复位值：0x0000 0000

仅当 CAN_FMR 寄存器中设置了筛选器初始化模式 (FINIT=1) 时，才能对此寄存器执行写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FBM27	FBM26	FBM25	FBM24	FBM23	FBM22	FBM21	FBM20	FBM19	FBM18	FBM17	FBM16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBM15	FBM14	FBM13	FBM12	FBM11	FBM10	FBM9	FBM8	FBM7	FBM6	FBM5	FBM4	FBM3	FBM2	FBM1	FBM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

注意：请参见第 617 页的图 231：筛选器组尺度配置 - 寄存器构成。

位 31:28 保留，必须保持复位值。

位 27:0 **FBM*x***：筛选器模式 (Filter mode)

筛选器 x 的寄存器的模式

0：筛选器存储区 x 的两个 32 位寄存器处于标识符屏蔽模式。

1：筛选器存储区 x 的两个 32 位寄存器处于标识符列表模式。

CAN 筛选器尺度寄存器 (CAN_FS1R)

CAN filter scale register

偏移地址：0x20C
复位值：0x0000 0000

仅当 CAN_FMR 寄存器中设置了筛选器初始化模式 (FINIT=1) 时，才能对此寄存器执行写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FSC27	FSC26	FSC25	FSC24	FSC23	FSC22	FSC21	FSC20	FSC19	FSC18	FSC17	FSC16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSC15	FSC14	FSC13	FSC12	FSC11	FSC10	FSC9	FSC8	FSC7	FSC6	FSC5	FSC4	FSC3	FSC2	FSC1	FSC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 保留，必须保持复位值。

位 27:0 **FSCx**：筛选器尺度配置 (Filter scale configuration)
这些位定义了筛选器 13-0 的尺度配置。
0：双 16 位尺度配置
1：单 32 位尺度配置

注意： *请参见第 617 页的图 231：筛选器组尺度配置 - 寄存器构成。*

CAN 筛选器 FIFO 分配寄存器 (CAN_FFA1R)

CAN filter FIFO assignment register

偏移地址：0x214
复位值：0x0000 0000

仅当 CAN_FMR 寄存器中设置了筛选器初始化模式 (FINIT=1) 时，才能对此寄存器执行写操作。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FFA27	FFA26	FFA25	FFA24	FFA23	FFA22	FFA21	FFA20	FFA19	FFA18	FFA17	FFA16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FFA15	FFA14	FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 保留，必须保持复位值。

位 27:0 **FFA*x***：筛选器 x 的筛选器 FIFO 分配 (Filter FIFO assignment for filter x)
通过此筛选器的消息将存储在指定的 FIFO 中。
0：筛选器分配到 FIFO 0
1：筛选器分配到 FIFO 1

CAN 筛选器激活寄存器 (CAN_FA1R)

CAN filter activation register

偏移地址：0x21C
复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FACT27	FACT26	FACT25	FACT24	FACT23	FACT22	FACT21	FACT20	FACT19	FACT18	FACT17	FACT16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FACT15	FACT14	FACT13	FACT12	FACT11	FACT10	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 31:28 保留，必须保持复位值。

位 27:0 **FACT*x***：筛选器激活 (Filter active)

软件将此位置 1 可激活筛选器 x。要修改筛选器 x 的寄存器 (CAN_FxR[0:7])，必须将 FACTx 位清零或将 CAN_FMR 寄存器的 FINIT 位置 1。
0：筛选器 x 未激活
1：筛选器 x 激活

筛选器组 i 寄存器 x (CAN_FiRx)（i=0..27，x=1，2）

Filter bank i register x

偏移地址：0x240..0x31C
复位值：0xXXXX XXXX

共有 28 个筛选器组，i=0 ..27。每个筛选器组 i 由两个 32 位寄存器组成，即 CAN_FiR[2:1]。

仅当 CAN_FAxR 寄存器的 FACTx 位清零，或者 CAN_FMR 寄存器的 FINIT 位置 1 时，才能修改此寄存器。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

在所有配置中：

位 31:0 **FB[31:0]**：筛选器位 (Filter bits)

标识符
寄存器的每一位用于指定预期标识符相应位的级别。
0：需要显性位
1：需要隐性位
掩码
寄存器的每一位用于指定相关标识符寄存器的位是否必须与预期标识符的相应位匹配。
0：无关，不使用此位进行比较。
1：必须匹配，传入标识符的此位必须与筛选器相应标识符寄存器中指定的级别相同。

注意： *每个寄存器的功能可能因筛选器的尺度和模式配置而异。有关筛选器映射、功能说明和掩码寄存器关联的信息，请参见第 616 页的第 24.7.4 节：标识符筛选。*

掩码模式中，掩码/标识符寄存器的位映射与**标识符列表**模式中的相同。

有关筛选器组的寄存器映射/地址信息，请参见第 644 页的表 102。

24.9.5 bxCAN 寄存器映射

有关寄存器边界地址的信息，请参见*第 52 页的表 2*。寄存器仅在 CAN1 中位于偏移量为 0x200 到 31C 的地址处。

表 102. bxCAN 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**CAN_MCR**	Reserved															DBF	RESET	Reserved							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	Reset value																1	0								0	0	0	0	0	0	1	0
0x004	**CAN_MSR**	Reserved																				RX	SAMP	RXM	TXM	Reserved			SLAKI	WKUI	ERRI	SLAK	INAK
	Reset value																					1	1	0	0				0	0	0	1	0
0x008	**CAN_TSR**	LOW[2:0]			TME[2:0]			CODE[1:0]		ABRQ2	Reserved			TERR2	ALST2	TXOK2	RQCP2	ABRQ1	Reserved			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Reserved			TERR0	ALST0	TXOK0	RQCP0
	Reset value	0	0	0	1	1	1	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0
0x00C	**CAN_RF0R**	Reserved																										RFOM0	FOVR0	FULL0	Reserved	FMP0[1:0]	
	Reset value																											0	0	0		0	0
0x010	**CAN_RF1R**	Reserved																										RFOM1	FOVR1	FULL1	Reserved	FMP1[1:0]	
	Reset value																											0	0	0		0	0
0x014	**CAN_IER**	Reserved													SLKIE	WKUIE	ERRIE	Reserved				LECIE	BOFIE	EPVIE	EWGIE	Reserved	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
	Reset value														0	0	0					0	0	0	0		0	0	0	0	0	0	0
0x018	**CAN_ESR**	REC[7:0]								TEC[7:0]									Reserved							LEC[2:0]			Reserved	BOFF	EPVF	EWGF	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0		0	0	0	
0x01C	**CAN_BTR**	SILM	LBKM	Reserved				SJW[1:0]		Reserved	TS2[2:0]			TS1[3:0]					Reserved							BRP[9:0]							
	Reset value	0	0					0	0		0	1	0	0	0	1	1									0	0	0	0	0	0	0	0	0
0x020-0x17F		Reserved																															
0x180	**CAN_TI0R**	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x184	**CAN_TDT0R**	TIME[15:0]																	Reserved						TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x							x					x	x	x	x
0x188	**CAN_TDL0R**	DATA3[7:0]								DATA2[7:0]									DATA1[7:0]											DATA0[7:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

表 102. bxCAN 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18C	**CAN_TDH0R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x190	**CAN_TI1R**	STID[10:0]/EXID[28:18]								EXID[17:0]																					IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x194	**CAN_TDT1R**	TIME[15:0]																Reserved							TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x198	**CAN_TDL1R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x19C	**CAN_TDH1R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1A0	**CAN_TI2R**	STID[10:0]/EXID[28:18]								EXID[17:0]																					IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x1A4	**CAN_TDT2R**	TIME[15:0]																Reserved							TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x1A8	**CAN_TDL2R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1AC	**CAN_TDH2R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1B0	**CAN_RI0R**	STID[10:0]/EXID[28:18]								EXID[17:0]																					IDE	RTR	Reserved
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1B4	**CAN_RDT0R**	TIME[15:0]																FMI[7:0]								Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
0x1B8	**CAN_RDL0R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1BC	**CAN_RDH0R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1C0	**CAN_RI1R**	STID[10:0]/EXID[28:18]								EXID[17:0]																					IDE	RTR	Reserved
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1C4	**CAN_RDT1R**	TIME[15:0]																FMI[7:0]								Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
0x1C8	**CAN_RDL1R**	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

表 102. bxCAN 寄存器映射和复位值 （续）

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1CC	**CAN_RDH1R**	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1D0-0x1FF	Reserved																																
0x200	**CAN_FMR**	Reserved																		CAN2SB[5:0]						Reserved							FINIT
	Reset value																			0	0	1	1	1	0								1
0x204	**CAN_FM1R**	Reserved								FBM[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x208	Reserved																																
0x20C	**CAN_FS1R**	Reserved								FSC[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x210	Reserved																																
0x214	**CAN_FFA1R**	Reserved								FFA[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x218	Reserved																																
0x21C	**CAN_FA1R**	Reserved								FACT[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x220	Reserved																																
0x224-0x23F	Reserved																																
0x240	**CAN_F0R1**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x244	**CAN_F0R2**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x248	**CAN_F1R1**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x24C	**CAN_F1R2**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
. . . .	. . . .	.																															
0x318	**CAN_F27R1**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x31C	**CAN_F27R2**	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x