STM32 器件上的 FDCAN 外设

引言

本文档的具体目标如下：

- 提供采用灵活数据率（CAN-FD）协议的控制器域网（CAN）的概述。
- 描述 CAN-FD 相较于传统 CAN（CAN2.0）的改进和优势。
- 展示下表所示 STM32 微控制器和微处理器中的 CAN-FD 实现。
- 描述 FDCAN 外设的各种模式和特性。

本应用笔记适用于下表中列出的产品。在本文档中，将这组适用产品统称为 *STM32 器件*。

表 1. 适用产品

类型	产品系列
微控制器	STM32G0 系列、STM32G4 系列、STM32H7 系列、STM32L5 系列
微处理器	STM32MP1 系列

1 概述

本应用笔记提供了表 1 中所列 STM32 微控制器和微处理器中嵌入的 FDCAN 外设的概述，它们是基于 Arm® Cortex® 内核的器件。

提示 *Arm 是 Arm Limited（或其子公司）在美国和/或其他地区的注册商标。*

2 CAN-FD 协议概述

CAN-FD 协议（具有灵活数据率的 CAN）是传统 CAN（CAN 2.0）协议的扩展。CAN-FD 是 CAN 2.0 的后续协议。它高效地支持分布式实时控制，具有极高的安全性。CAN-FD 由博世开发，并被标准化为 ISO 11898-1:2015（适合工业、汽车和一般嵌入式通信）。

2.1 CAN-FD 功能

CAN-FD 协议的主要特性如下：

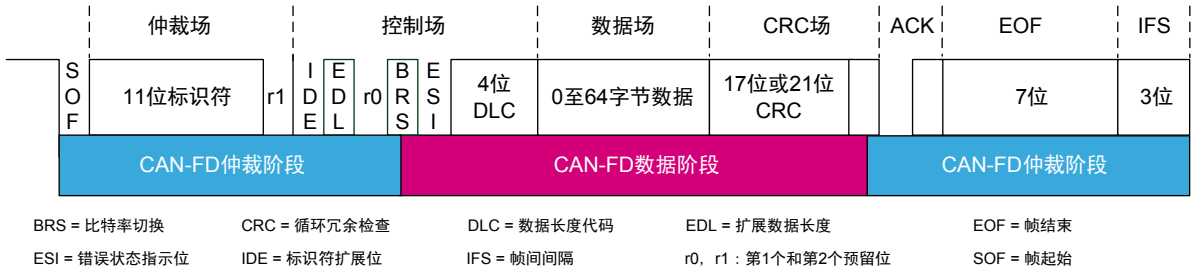
- 与 CAN 协议兼容：CAN-FD 节点能够发送/接收 CAN 消息（依据 ISO 11898-1）
- 错误校验改进，基于最多 21 位 CRC 校验和域
- 消息优先次序
- 延迟时间保证
- 配置灵活性
- 时间同步多播接收
- 系统范围内的数据一致性，每条信息最多 64 个字节
- 多主机
- 错误检测和信号传递
- 区分节点的暂时性错误和永久性故障并自动关闭问题节点

2.2 CAN-FD 格式

发送的数据被封装在一条消息中，如下图所示。CAN-FD 消息可划分为三个阶段：

1. 第一仲裁阶段
2. 数据阶段
3. 第二仲裁阶段

图 1. 标准 CAN-FD 帧



第一仲裁阶段是一条包含下列内容的消息：

- 帧起始 (SOF)
- ID 号和其他位，指示消息用途（提供或请求数据）、速度和格式配置（CAN 或 CAN-FD）

数据发送阶段包含：

- 数据长度代码（DLC），指示消息包含的数据字节数
- 用户想要发送的数据
- 校验循环冗余序列（CRC）
- 显性位

第二仲裁阶段包含：

- 由总线上的其他节点发送的接受方的应答(ACK)（如果至少有一个成功接收到消息）

- 帧结束（EOF）
  IFS 期间不发送消息：目的是将当前帧与下一帧分开。

提示　　如果在第一仲裁阶段的 IDE 位后添加 18 位标识符，则 29 位标识符帧类似于标准 CAN-FD 帧。

3 CAN-FD 相较于 CAN 2.0 的改进和优势

开发 CAN-FD 是为了响应对更高带宽的通信网络的需求。为了满足此需求，使用每帧最多有 64 个字节的 CAN-FD，并在数据阶段将比特率增加至最多快八倍，然后在第二仲裁阶段恢复正常比特率。

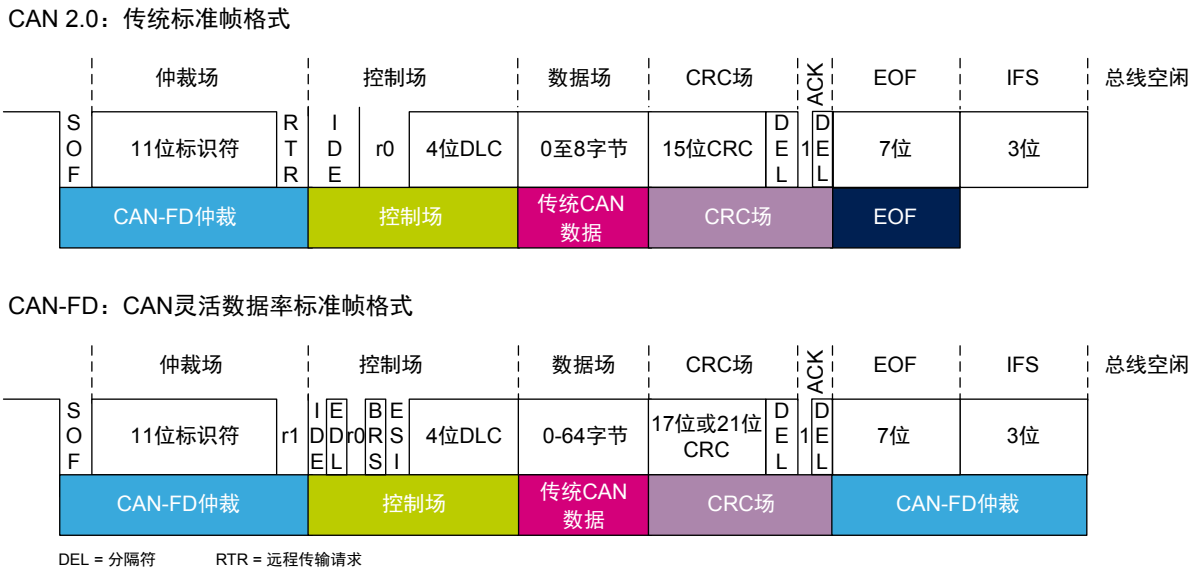
通过以下方式确保数据传输完整性：

- 使用 CRC 基于 17 阶多项式校验最多 16 字节的有效负载
- 使用 21 阶多项式校验 16 至 64 字节的有效负载

3.1 CAN-FD 与 CAN 2.0 的帧架构比较

下图显示了 CAN-FD 与 CAN 2.0 相比在帧架构上的主要差异。

图 2. CAN-FD 与 CAN 2.0 的帧架构比较



在标识符之后，CAN 2.0 和 CAN-FD 有一个不同的操作：

- CAN 2.0 发送 RTR 位明确帧类型：数据帧（RTR 为显性）或远程帧（RTR 为隐性）。
- CAN-FD 只支持数据帧，因此总是发送显性 RRS（预留）。

IDE 位保存在相同位置，并用相同的操作来区分基本格式（11 位标识符）。请注意，对于扩展格式（29 位标识符），以显性或隐性形式发送 IDE 位。

相较于 CAN 2.0，CAN-FD 帧中的控制场增加了三个新的位：

- 扩展数据长度（EDL）位：若为隐性，表示帧为 CAN-FD 帧；否则，此位为显性（称为 R0）（在 CAN 2.0 帧中）。
- 比特率切换（BRS）：指示是否启用了两个比特率（例如，在用不同于仲裁阶段的比特率发送数据阶段时）。
- 错误状态指示位（ESI）：指示节点处于错误主动模式还是错误被动模式。

控制场的最后一部分是数据长度代码（DLC），CAN 2.0 和 CAN-FD 的 DLC 具有相同位置和相同长度（4 位）。在 CAN-FD 和 CAN 2.0 中，DLC 的功能相同，CAN-FD 的细微不同之处在于有效负载数据长度代码（详细信息见下表）。CAN-FD 允许在一条消息中发送包含最多 64 个数据字节的扩展帧，而 CAN 2.0 的有效负载数据最多包含 8 个字节。

表 2. 有效负载数据长度代码（字节）

DLC (Dec)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CAN 2.0	0	1	2	3	4	5	6	7	8	8	8	8	8	8	8	8
CAN-FD	0	1	2	3	4	5	6	7	8	12	16	20	24	32	48	64

承载有效负载数据的数据场的增加改善了网络带宽，原因在于对多包处理的需求减少。因此，通过增加 CRC 字段的位数，可以提高消息完整性：

- 如果有效负载数据最多为 16 个字节，则 CRC 编码为 17 位。
- 如果有效负载数据多于 20 个字节，则 CRC 编码为 21 位。

另外，为确保 CAN-FD 帧的稳健性，CRC 字段支持位填充机制。

下表总结了 CAN-FD 与 CAN 2.0 之间的主要差异。CAN-FD 优于 CAN 2.0 的主要特性是增加了数据有效负载，并通过 CAN-FD 中的 BRS、EDL 和 ESI 位确保更高速度。

表 3. CAN-FD 与 CAN 2.0 之间的主要差异

特征	CAN 2.0	CAN-FD
兼容性	不支持 CAN-FD	支持 CAN 2.0 A/B
最大比特率（Mbit/s）	帧比特率：最高 1	仲裁比特率：最高 1s 数据比特率：最高 8
DLC 字段（4 位）编码	编码为 0 至 8	编码为 0 至 64
一条消息中的最大数据类型	8 字节数据	64 字节数据
支持 BRS	无	有
支持 EDL	无	有
支持 ASI	无	有
CRC 位校验码	CRC 计算中不包括位	CRC 计算中包括位
远程帧支持	有	无

提示 *关于 CAN 2.0 和 CAN-FD 的更多信息，请参见博世网站上提供的相关文档。*

4 STM32 器件中 CAN-FD 的实现

表 1 中定义的 STM32 器件内置支持 CAN-FD 协议（依据 ISO 11898-12015）的 FDCAN 外设。大多数 STM32 器件支持一个以上的 CAN 实例（请参见产品数据手册了解特定设备的可用实例数量）。

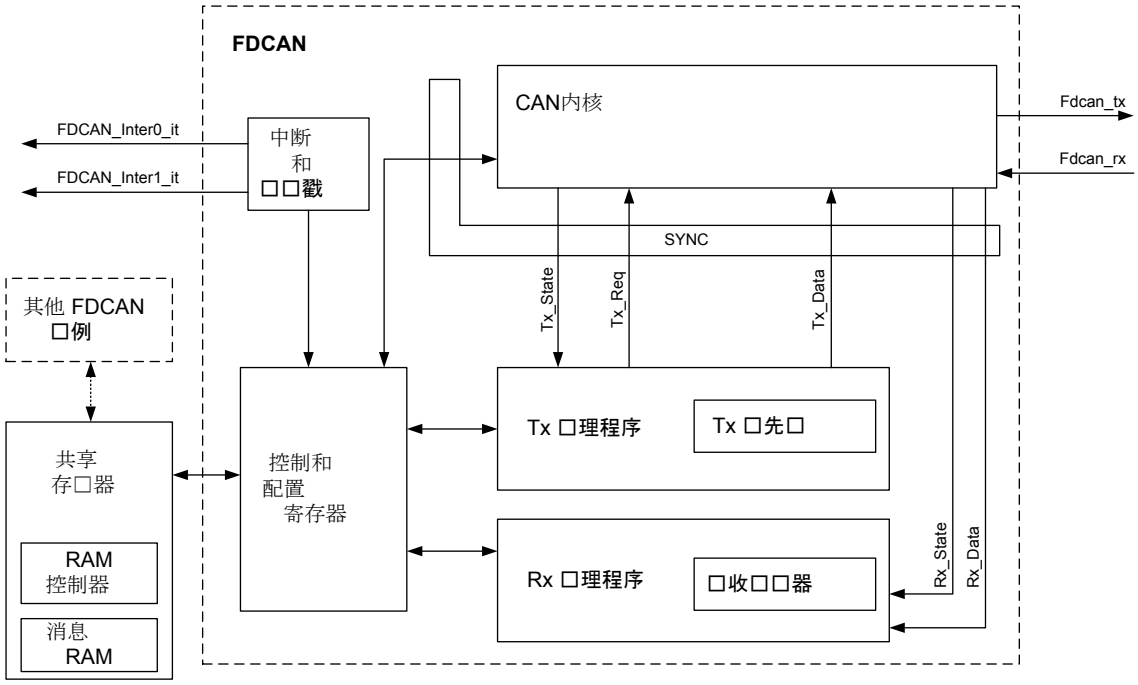
4.1 FDCAN 外设的主要特性

STM32 器件上的 FDCAN 的特性如下：

- 符合 CAN 协议第 2.0 版 A、B 部分和 ISO 11898-1: 2015, -4
- 可访问的 10-KB RAM 存储器，可存储最多 2560 字
- 改进了接受过滤
- 两个可配置接收 FIFOs
- 多达 64 个专用接收缓冲区
- 接收到高优先级消息时单独发出信号指示
- 多达 32 个专用发送缓冲区
- 可配置*发送 FIFO* 和*发送队列*
- 可配置*发送事件 FIFO*
- 时钟校准单元
- 收发器延迟补偿

下图所示为 FDCAN 框图。

图 3. FDCAN 框图



FDCAN 框图体现了下列特性：

- 所有 FDCAN 实例共享一个存储器。
- 每个 FDCAN 实例都包含 CAN 内核。
- CAN 内核提供协议控制器和接收/发送移位寄存器。
- Tx 处理程序控制从 CAN 消息 RAM 到 CAN 内核的消息传输。
- Rx 处理程序控制从 CAN 内核到外部 CAN 消息 RAM 的已接收消息的传输。

4.2 RAM 管理

发送和接收的所有消息均存储在 CAN 消息 RAM 中。在 CAN 消息 RAM 初始化期间，用户必须定义 11 位滤波器、29 位滤波器、接收到的消息和要发送的消息的存储位置。

4.2.1 RAM 组织

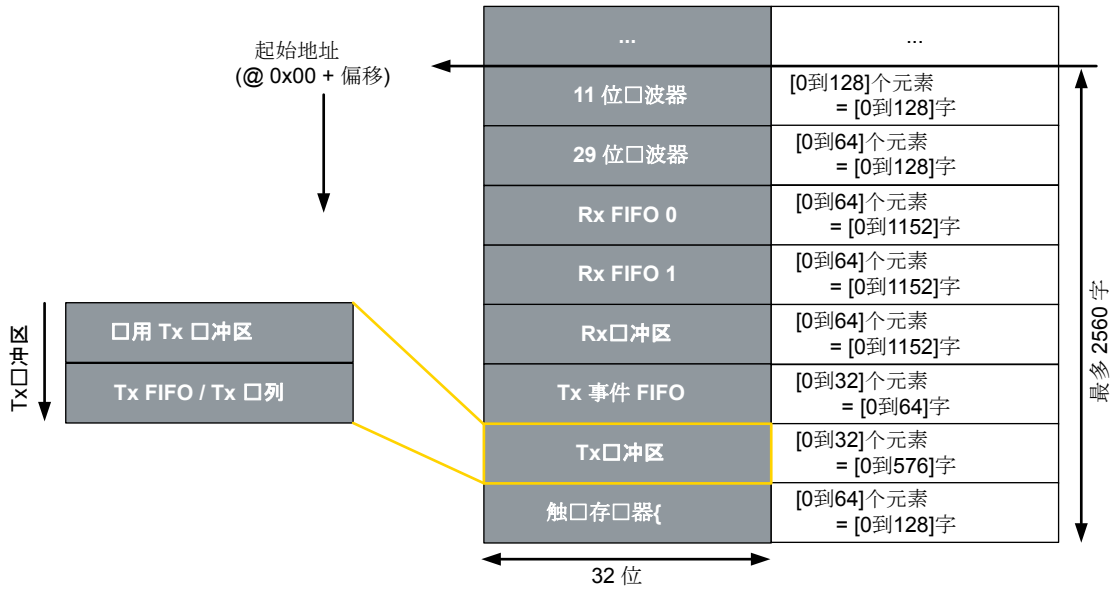
为了确定每条消息需要的存储空间，必须配置每条消息的数据字节数。CAN-FD 上的有效负载增加会提高存储器的使用效率，还能在分配的存储空间内存储更多消息。

使用 STM32 器件上为 FDCAN 预留的专用 RAM，可分配最多 2560 字（32 位）。此预留 RAM 空间使 CPU 更高效。

如下图所示，CAN 消息 RAM 分为四个不同区域：

- 滤波区（11 位滤波器、29 位滤波器）
- 接收区（Rx FIFO 0、Rx FIFO 1 和 Rx 缓冲区）
- 发送区（Tx 事件 FIFO 和 Tx 缓冲区）
- 触发存储器区（触发存储器）

图 4. CAN 消息 RAM 映射



用户可以配置 FDCAN 外设的所有区。所有区的所有元素之和不得超过 CAN 消息 RAM 的总大小。由于能够清除未使用的区并为其他区扩展足够的存储空间，此 RAM 提供了更高的灵活性和性能。

在 CAN 消息 RAM 中，以动态且连续的方式分配每个区配置的元素（按照上图中的顺序）；但是，为避免超出 RAM 的风险并出于可靠性考虑，没有为每个区分配特定的起始和结束地址。

FDCAN 外设可以配置三种发送机制：Tx 缓冲区和/或 Tx 队列和/或 Tx FIFO，可以通过 Rx 缓冲区和/或 Rx FIFO 接收。它们配置有起始地址偏移和要存储的存储元素数。在配置中预定义起始地址（0 至 2560 之间的数），由用户负责确保每个存储空间的元素数不会导致重叠。

提示 *消息的接收和发送意味着有必要在 RAM 层面存储“元素”。此“元素”只包含 DLC、控制位（ESI、XTD、RTR、BRS 和 FDF）、数据场和用于控制的特定发送/接收位字段。CAN 消息的剩余位由硬件自动处理且不保存在 RAM 中。*

用于控制接收的特定位字段是滤波器索引、接受的不匹配帧和 Rx 时间戳。用于发送的特定位字段是消息标志和事件 FIFO 控制位。

为了保留以下内容，无论是 Tx 缓冲区、Tx FIFO、Tx 队列还是 Rx 缓冲区，都将计算为每个元素分配的 32 位字的数量：

- 头文件消息（预留的两个 32 位字），用于分配标识符、DLC 字段、控制位和特定发送/接收位字段
- 数据（足够数量的 32 位字），包含每个数据场的字节数

使用下面的公式确定为每个元素分配的 32 位字数量：

元素大小（以字为单位）= 头文件消息（2 字）+ 数据（数据场/4）

其中，数据场是指每条消息的数据字节数。

提示 *如果数据场处于 0 至 8 的范围内，则为每个元素的数据分配 2 字。*

下表详细提供了取决于数据场范围的必要“元素”大小。

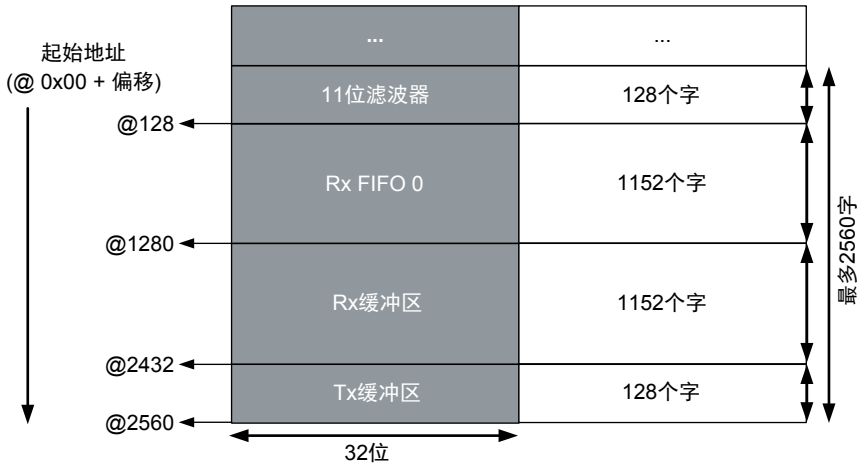
表 4. 取决于数据场范围的“元素”大小

数据场（字节）	元素大小（RAM 字）
0 至 8	4
12	5
16	6
20	7
24	8
32	10
48	14
64	18

下图所示为一个高效使用 CAN 消息 RAM 的示例。该示例假定应用配置了 FDCAN 外设：

- 用专用 Tx 缓冲区发送 32 条消息（每条消息的数据场包含 8 字节）
- 提供 128 个 11 位滤波器用于接受消息
- 用专用 Rx 缓冲区接收 64 条消息，每条消息的数据场包含 64 字节
- 用 Rx FIFO 0 接收 64 条消息，每条消息的数据场包含 64 字节

图 5. CAN 消息 RAM 的 RAM 映射示例和高效利用



在本例中，按下列顺序完成 RAM 中的分配：

1. 在 11 位 ID 部分分配 128 字。
2. 在 Rx FIFO 0 区预留 1152 字用于接收元素。
3. 在 Rx 缓冲区预留 1152 字用于接收元素。
4. 在 Tx 缓冲区为发送的元素预留 128 字。

得益于动态分配，以及不对未使用区域进行任何分配，RAM 的总存储量得到高效利用：在该应用中，对全部 2560 字进行了分配。

提示　　配置后，分配的地址范围初始化为零。

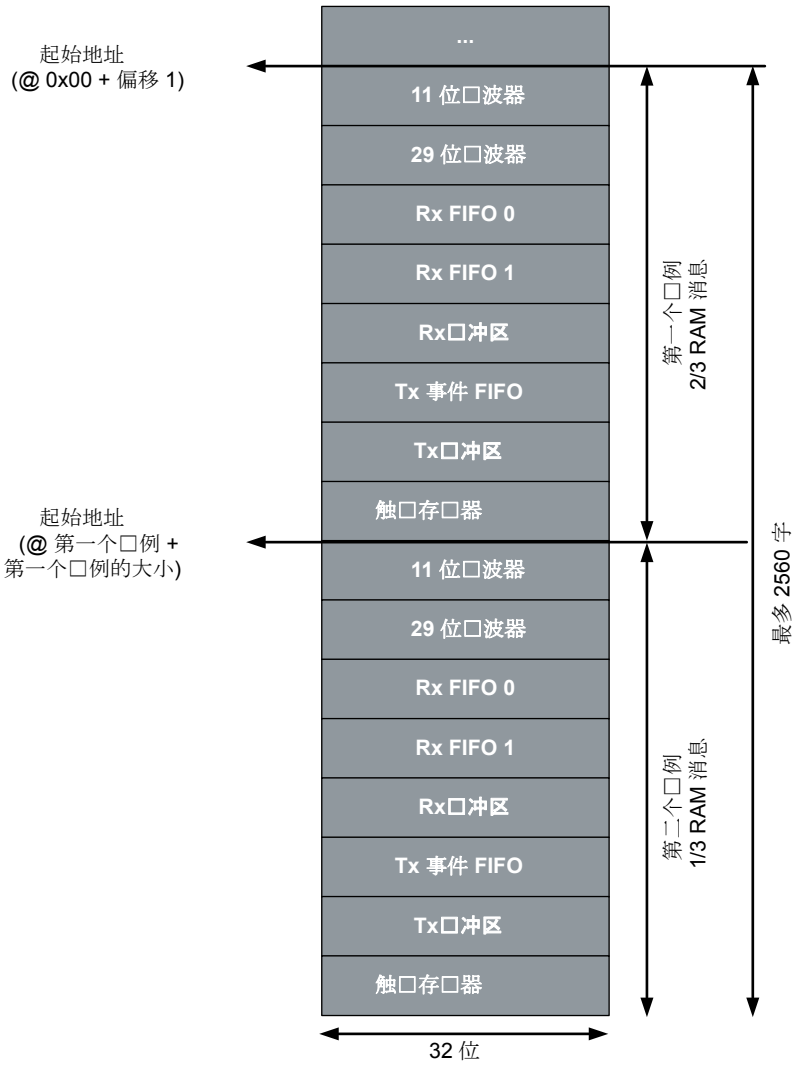
4.2.2 多个 FDCAN 实例

为了满足所有应用要求，大多数 STM32 器件支持一个以上的 FDCAN 实例（请参见产品数据手册了解实例数量）。

在这种情况下，不同实例共享 RAM 十分重要。用户可根据各种不同实例划分 RAM（通过指定起始偏移地址选择每个实例的大小）。

下图所示为使用多个 FDCAN 实例的 CAN 消息 RAM 的示例。该示例假定用户将 CAN 消息 RAM 划分为两个实例：第一个实例的大小是第二个实例的两倍。

图 6. 具有多个 FDCAN 实例的 CAN 消息 RAM 的示例



4.3 RAM 区

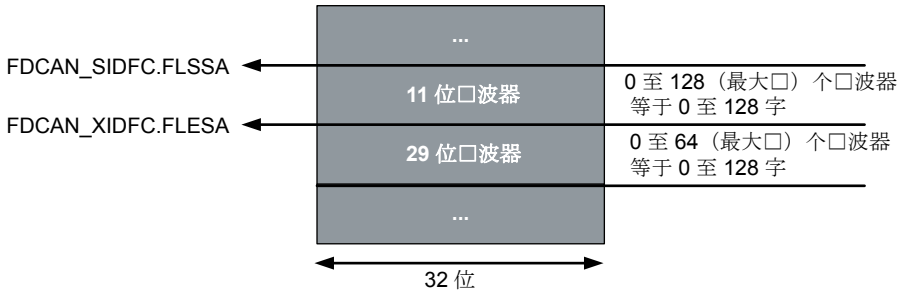
4.3.1 RAM 过滤区

FDCAN 外设可以配置两组接受滤波器：一组用于标准标识符，一组用于扩展标识符，以便存储或拒绝接收到的消息。可以为 11 位标准 ID 配置最多 128 个滤波器元素，可以为 29 位扩展 ID 配置最多 64 个滤波器元素。

通过 FDCAN_SIDFC 寄存器中的 FLSSA[13:0] 位配置 11 位滤波器区的起始地址，通过 FDCAN_XIDFC 寄存器中的 FLESA[13:0] 位配置 29 位滤波器区的起始地址。

下图显示了 CAN 消息 RAM 的一个区域，以及滤波器元素数及其起始地址。

图 7. CAN 消息 RAM 滤波器区



这些滤波器可以分配给 Rx FIFO 0/1 或专用 Rx 缓冲区。FDCAN 在执行接受过滤时，总是从滤波器元素 #0 开始，然后遍历滤波器列表以找到匹配的元素。接受过滤会在第一个匹配元素处停止，不为此消息评估后续滤波器元素。因此，配置的滤波器元素的顺序对过滤过程的执行影响极大。

用户选择启用或禁用每个滤波器元素，并且可以将每个元素配置用于接受或拒绝过滤。每个滤波器元素可配置为：

- 范围滤波器：该滤波器匹配标识符处于两个 ID 定义的范围之内的所有消息。
- 专用 ID 的滤波器：该滤波器可配置用于匹配一个或两个特定标识符。
- 传统位屏蔽滤波器：通过对所接收标识符的位进行屏蔽来匹配标识符组。第一个 ID 配置为消息 ID 滤波器，第二个 ID 为滤波器屏蔽。滤波器屏蔽的每个零位会屏蔽所配置 ID 滤波器的相应位。

提示 *如果所有位均为 1，仅当接收到的消息 ID 与消息 ID 滤波器一致时才匹配。如果所有屏蔽位均为 0，则所有消息 ID 均匹配。*

4.3.1.1 高优先级消息

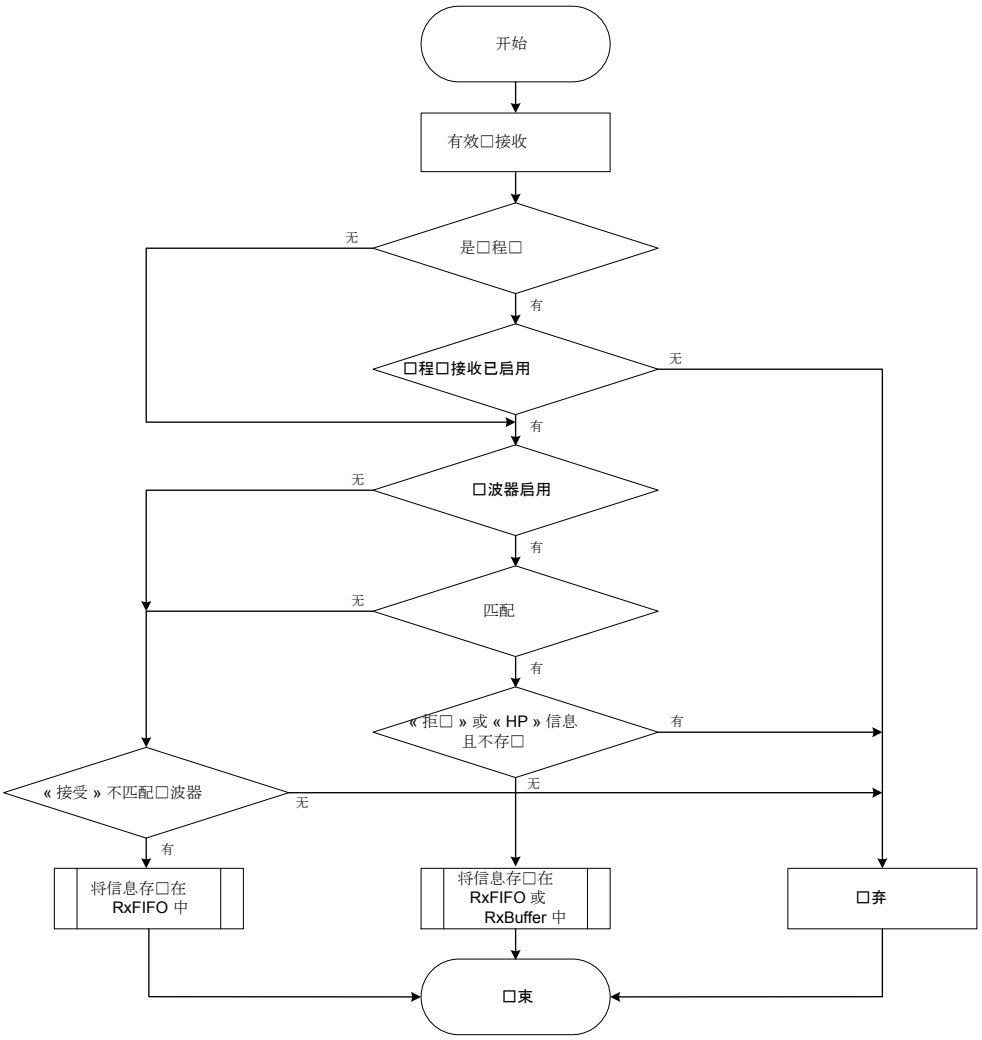
FDCAN 可以在接收到高优先级消息时通知用户。此通知功能可用于监控传入高优先级消息的状态，并支持对这些消息的快速访问。

FDCAN 利用消息滤波器检测高优先级消息。滤波器元素提供下列与高优先级消息相关的设置：

- 如果滤波器匹配，设置优先级并存储在 FIFO 0/1 中：如果此消息滤波器匹配，FDCAN 会通知用户有高优先级消息到达，并将元素存储在 Rx FIFO 0/1 中。
- 如果滤波器匹配，设置优先级：如果此消息滤波器匹配，FDCAN 会通知用户有高优先级消息到达，但不存储元素。

下面的流程图解释了接受滤波器的全局机制。

图 8. 接受滤波器的全局流程图



接受过滤的说明示例

为了说明可以使用的不同滤波器类型以及每种类型的结果，假定用户想要配置 FDCAN：

- 拒绝所有标识符在 [0x16 到 0x20] 范围内的消息
- 接受所有标识符等于 0x15 或 0x120 的消息并将其存储在 FIFO 1 中
- 接受标识符等于 0x130 的消息并将其存储在 Rx 缓冲区索引 4 中

- 接受具有与以下内容相对应的标识符的消息：
  - 位 [10..6] = 0b111 00
  - 位 [5..4] = *任意值*
  - 位 [3..0] = 0b00000

  在这种情况下，必须将滤波器配置为*传统位屏蔽滤波器*，因为接受的标识符对应于 0b11100XX0000（其中 x 可以是 0 或 1 中的任何值）。接受的标识符为：
  - 0b111 0000 0000 (0x700)
  - 0b111 0001 0000 (0x710)
  - 0b111 0010 0000 (0x720)
  - 0b111 0011 0000 (0x730)

  基础滤波器 ID 可以是 0x700、0x710、0x720 和 0x730 中的任何值。屏蔽滤波器 ID 等于 0b111 1100 1111 (0x7CF)。

下表列出了上述示例中的标准 11 位消息 ID 滤波器的不同配置。每个标准滤波器元素均包含：

- SFT 位（标准滤波器类型）
- SFEC 位（标准滤波器元素配置）
- SFID1 位（标准滤波器 ID1）
- SFID2 位（标准滤波器 ID2）

表 5. 标准滤波器元素配置

滤波器	标准滤波器类型 SFT [31:30]	标准滤波器元素配置 SFEC [29:27]	标准滤波器 ID 1 SFID1 [26:16]	标准滤波器 ID 2 SFID2 [15:0]
第一个	00 - 范围滤波器	011 - 拒绝	0x16	0x20
第二个	01 - 双 ID	010 - 存储在 FIFO 1 中	0x15	0x120
第三个	xx - 无关	111 - 存储在 Rx 缓冲区中	0x130	0x04（缓冲区索引）
第四个	10 - 传统位屏蔽滤波器	001 - 存储在 FIFO 0 中	0x700	0x7CF

第一个滤波器配置为拒绝 ID 在 范围 [0x16...0x20] 以内的消息。

第二个滤波器配置为将 ID 等于双 ID 0x15 或 0x120 的消息存储在 Rx FIFO 1 中。

第三个滤波器配置为将 ID 等于 0x130 的消息存储在 Rx 缓冲区索引 4 中。

提示 *如果将 SFEC 配置为“存储到 Rx 缓冲区”，将忽略 SFT 的配置。接受滤波器会在第一个匹配元素处停止。因此，滤波器的顺序很重要。*

本示例采用与用户配置扩展滤波器相同的方式配置标准滤波器（请参见产品数据手册了解更多信息）。

FDCAN 的众多滤波器选择允许在硬件中进行复杂的消息过滤，不仅提供了软件过滤冗余，还节省了 CPU 资源。

4.3.2 接收区

4.3.2.1 Rx FIFO 0 与 Rx FIFO 1

可以在 CAN 信息 RAM 中配置两个 Rx FIFO。每个 Rx FIFO 区可存储最多 64 个元素。每个 Rx FIFO 元素中存储一个元素。

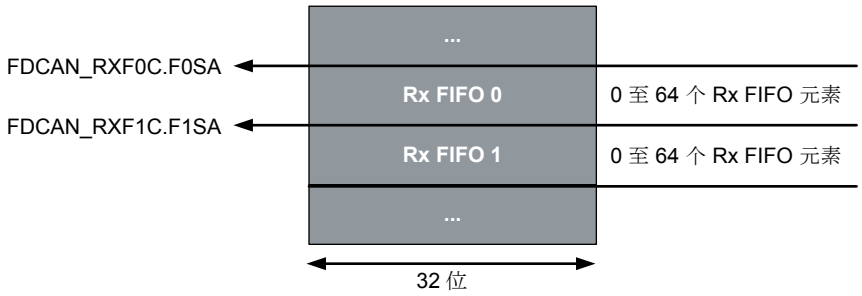
可通过 FDCAN_RXESC 寄存器分别为每个 Rx FIFO 配置 Rx FIFO 元素的大小。Rx FIFO 元素的大小定义了可存储的已接收元素的数据场字节数。通过第 4.2.1 节 RAM 组织中指定的公式定义 Rx FIFO 元素的大小。

头文件信息包含标识符、DLC 字段、控制位和位字段（滤波器索引、接受的不匹配帧和 Rx 时间戳）。

在通过 FDCAN_RXESC 寄存器中的 F1DS[2:0] 字段配置元素大小后，必须通过 FDCAN_RXF1C 寄存器中的 F1S[6:0] 和 F1SA[13:0] 字段分别配置 Rx FIFO 1 的元素数量和起始地址。

下图所示为 CAN 信息 RAM 中的 Rx FIFO 区，以及每个区可支持的元素数量和起始地址。

图 9. CAN 信息 RAM 中的 Rx FIFO 区



Rx FIFO 的起始地址是第一个 Rx FIFO 元素的第一个字的地址。根据匹配的滤波器元素，将接收到的通过接受过滤的元素存储在合适的 Rx FIFO 中。

如果 Rx FIFO 已满，可根据两种不同模式处理新到达的元素：

- 拦截模式：这是 Rx FIFO 的默认工作模式，在至少有一个元素被读出前，不再向 Rx FIFO 写入元素。
- 覆盖模式：Rx FIFO 中接收的新元素会覆盖 Rx FIFO 中最早的元素，并且 FIFO 的*放入和获取*索引增加 1。

为了从 Rx FIFO 读取元素，CPU 必须执行下列步骤：

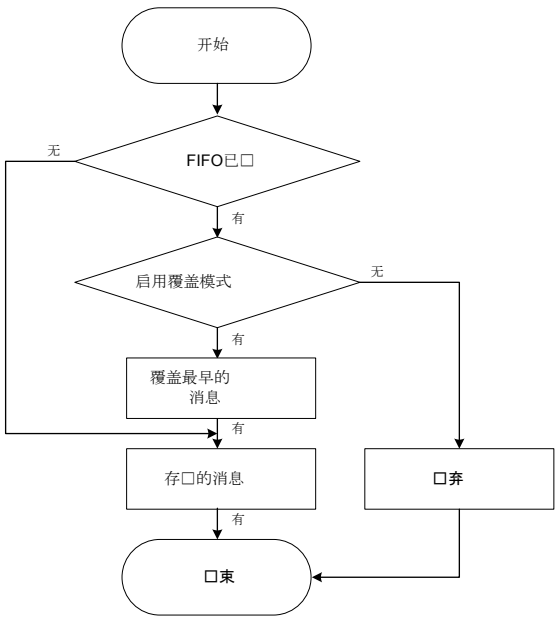
1. 读取 FDCAN_RXF1S 寄存器以了解 Rx FIFO 的状态。
2. 使用下列公式，计算 RAM 中最早元素的地址：

   最早元素的地址 = CAN_message_RAM_base_address + FDCAN_RXF1C.F1SA（起始地址）+ FDCAN_RXF1S.F1GI（获取索引）x Rx FIFO_element_size。
3. 从计算所得地址读取元素。

在 CPU 从 Rx FIFO 读取一个或一系列元素后，CPU 必须对读取应答。应答后，FDCAN 可将相应 Rx FIFO 缓冲区重新用于新元素。为了确认一个或多个元素，CPU 必须将从 Rx FIFO 读取的最后一个元素的缓冲区索引写入 FDCAN_RXF1A 寄存器。然后，FDCAN 将更新 FIFO 使用率和*获取索引*。

下图所示为简化后的 Rx FIFO 操作。

图 10. 简化后的 Rx FIFO 操作



提示 *Rx FIFO 0 和 Rx FIFO 1 具有相同寄存器，它们的名称有含义，并随 FIFO 数量而变化。*

4.3.2.2 *专用 Rx 缓冲区*

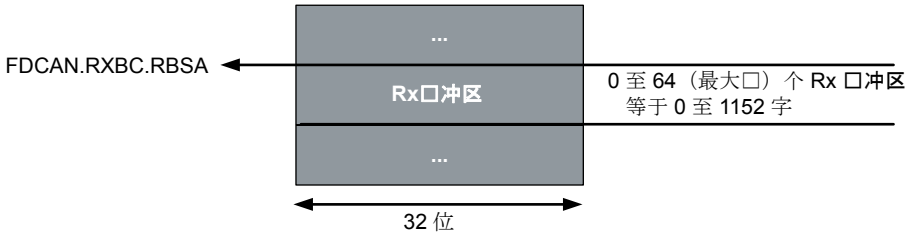
FDCAN 支持最多 64 个专用 Rx 缓冲区。每个专用 Rx 缓冲区可存储一个元素。

专用 Rx 缓冲区的大小可通过 FDCAN_RXESC 寄存器进行配置。Rx 缓冲区的大小定义了可存储的已接收元素的数据场字节数。通过第 4.2.1 节 RAM 组织中描述的公式定义专用 Rx 缓冲区的大小。

在通过 FDCAN_RXESC 寄存器中的 RBDS[2:0] 字段配置元素大小后，必须通过 FDCAN_RXBC 寄存器中的 RBSA[13:0] 字段配置起始地址。

下图所示为 CAN 消息 RAM 上具有可支持的最大专用 Rx 缓冲区元素数量的 Rx 缓冲区和起始地址。

图 11. CAN 消息 RAM 上的 Rx 缓冲区



当元素存储在专用 Rx 缓冲区中时，FDCAN 通过 FDCAN_IR 寄存器中的 DRX 位和新数据标记 FDCAN_NDAT1 或 FDCAN_NDAT2 寄存器中的相应位将中断标记置位。当 FDCAN_NDAT1/2 寄存器中的位置位时，对应 Rx 缓冲区被锁定（不会被新元素覆盖）且相应滤波器不匹配。在读取元素后，CPU 必须将 FDCAN_NDAT1/2 中的相应位复位，以便解锁对应 Rx 缓冲区。

为了从专用 Rx 缓冲区读取元素，CPU 必须执行下列步骤：

1. 检查 FDCAN_NDAT1/2 中的位，以便了解专用 Rx 缓冲区是否接收到新元素。
2. 使用下列公式，计算 CAN 消息 RAM 中的元素地址：

   参考 Rx 缓冲区地址 = CAN_message_RAM_base_address + FDCAN_RXBC.RBSA（起始地址）+ 专用 Rx 缓冲区索引 x Rx_Buffer_element_size。
3. 从计算所得地址读取消息。

滤波器元素可引用 Rx 缓冲区索引（0 至 63）作为所接收元素的目标位置。如果相应滤波器匹配，则 FDCAN 将只对引用的 Rx 缓冲区位置执行写入。换句话说，FDCAN 不对未引用的 Rx 缓冲区位置执行写入。

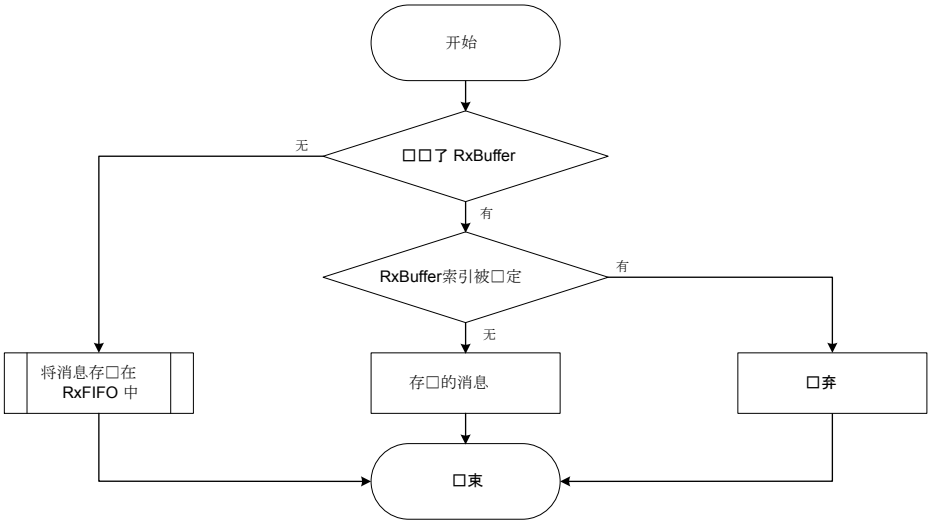
Rx 缓冲区编号对 Rx 缓冲区索引的相对配置示例

为了配置滤波器元素以便引用 Rx 缓冲区索引 60，必须配置至少 61 个 Rx 缓冲区。

提示 *为避免浪费 RAM，用户必须选择最佳配置。*

下图所示流程图是简化后的 Rx 缓冲区操作。

图 12. 简化后的 Rx 缓冲区操作



4.3.2.3 *专用 Rx 缓冲区与 Rx FIFO 之间的差异*

如前文所述，FDCAN 有两种机制：专用 Rx 缓冲区或 Rx FIFO 0/1 都可以配置用于存储接收到的元素。

下表中描述了专用 Rx 缓冲区与 Rx FIFO 之间的差异。

表 6. 专用 Rx 缓冲区与 Rx FIFO 之间的差异

特征	专用 Rx 缓冲区	Rx FIFO
要配置的区	可配置 64 个专用 Rx 缓冲区。	可配置 2 个 Rx FIFO。
每个区的元素	配置为每个缓冲区仅包含一个元素	每个缓冲区可包含一个或多个元素（最多 64 个元素）
在 RAM 中的位置	用户选择缓冲区索引。	自动并动态地管理在 RAM 中的位置（使用获取/放入索引的递增机制）。
丢弃新到达元素的配置	在缓冲区被锁定时丢弃新到达的元素。 *注意：用户必须将 FDCAN_NDAT1/2 中的相应位复位。*	在 Rx FIFO 已满（默认拦截模式）时丢弃新到达的元素。 *注意：覆盖模式选项用于接收新元素并覆盖最早的元素。*

4.3.3 发送部分

4.3.3.1 *Tx 事件 FIFO 区*

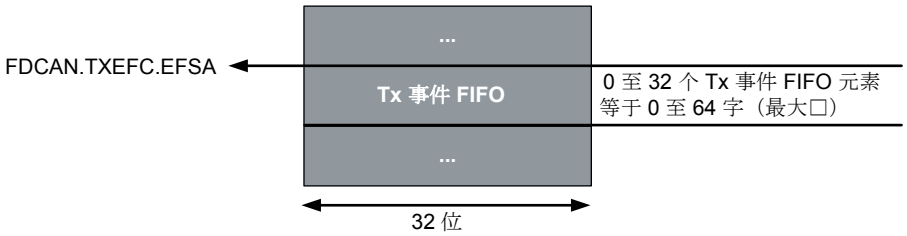
CPU 使用 Tx 事件 FIFO 获取所发送元素的下列信息：

- 元素的发送顺序
- 每个元素的帧的本地发送时间

FDCAN 提供 Tx 事件 FIFO。用户可以选择是否使用此 Tx 事件 FIFO。在 FDCAN 通过 CAN 总线成功发送元素后，可以在 Tx 事件 FIFO 元素中存储信息 ID 和时间戳。Tx 事件 FIFO 元素是一种数据结构，用于存储所发送信息的相关信息。Tx 事件 FIFO 可通过 FDCAN_TXEFC 寄存器进行配置（Tx 事件 FIFO 配置）。FIFO 最多可存储 32 个元素。

下图显示了存储 Tx 事件 FIFO 元素的 CAN 消息 RAM 的示例，EFSA[13:0] 字段包含起始地址。

图 13. CAN 信息 RAM 中的 Tx 事件 FIFO 区



使用下面的公式计算 CAN 消息 RAM 中 Tx 事件 FIFO 元素的地址：

Tx 事件 FIFO 元素地址 = CAN_message_RAM_base_address + FDCAN_TXEFC.EFSA（起始地址）+ FDCAN_TXEFS.EFGI（获取索引）x Tx_event_FIFO_element_size

为了将 Tx 事件关联到 Tx 事件 FIFO 元素，已发送的 Tx 缓冲区中的信息标志会被复制到 Tx 事件 FIFO 元素中。

仅当 Tx 缓冲区元素中的 EFC 位（存储 Tx 事件）等于 1 时，事件才会存储在 Tx 事件 FIFO 中。

当 Tx 事件 FIFO 已满时，不再向 Tx 事件 FIFO 中写入元素，直至至少一个元素被读出且 Tx 事件 FIFO 获取索引增加。如果在 Tx 事件 FIFO 已满时发生 Tx 事件，该事件会被丢弃。为避免 Tx 事件 FIFO 溢出，可使用 Tx 事件 FIFO 水印。

在 CPU 从 Tx 事件 FIFO 读取一个或一系列元素后，CPU 必须对读取应答。为此，它通过 FDCAN_TXEFA 寄存器的 EFAI[4:0] 字段写入从 Tx 事件 FIFO 读取的最后一个元素的索引。

4.3.3.2 *Tx 缓冲区*

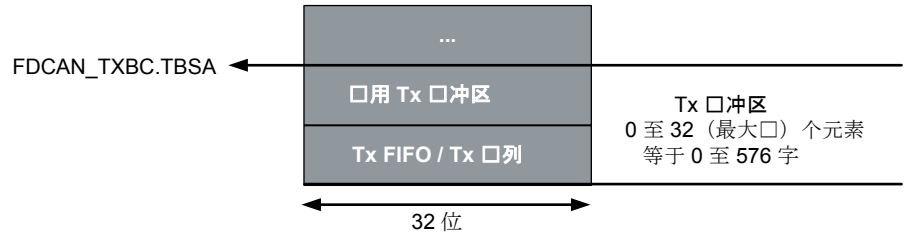
针对用于发送元素模块，在指定的存储空间内形成元素并开始发送。发送的元素存储在 Tx 缓冲区，用户可以选择其中使用的机制：专用 Tx 缓冲区和/或 Tx 队列或 Tx FIFO。

FDCAN 支持最多 32 个元素。每个元素都存储了一条消息的标识符、DLC、控制位（ESI、XTD、RTR、BRS 和 FDF）、数据域、位字段消息标志和事件 FIFO 控制位。

按照下列顺序完成在 RAM 上的分配：如果应用使用专用 Tx 缓冲区，则将其分配在 Tx FIFO 和 Tx 队列之前。在一个应用中，用户只能选择 Tx 队列或 Tx FIFO：FDCAN 不支持二者的组合。

如下图所示，通过 FDCAN_TXBC 寄存器中的 TBSA[13:0] 字段配置 Tx 缓冲区的起始地址。

图 14. CAN 消息 RAM 中的 Tx 缓冲区



提示 *如前文所述，RAM 中的分配是以动态且连续的方式进行的，若用户没有配置专用 Tx 缓冲区机制，则 Tx 缓冲区只包含配置的 Tx 队列或 Tx FIFO，并存储在起始地址中。*

专用发送缓冲区

通过 FDCAN_TXBC 寄存器中的 NTDB[5:0] 字段配置专用 Tx 缓冲区的数量。为每个专用 Tx 缓冲区配置了特定标识符，以便只存储一个元素。通过 FDCAN_TXBAR 寄存器的*添加请求*请求发送。请求的消息用 CAN 总线上的消息进行外部仲裁，并按最小标识符（最高优先级）发送。

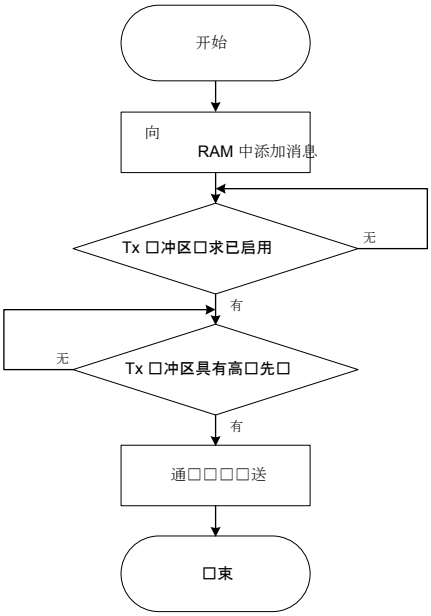
专用 Tx 缓冲区的存储要求取决于 Tx 缓冲区的元素大小。Tx 缓冲区的元素大小定义了 Tx 缓冲区包含的数据字节数。

使用下面的公式计算 CAN 消息 RAM 中专用 Tx 缓冲区的地址：

专用 Tx 缓冲区地址 = CAN_message_RAM_base_address + FDCAN_TXBC[TBSA]（起始地址）+ Tx_buffer_index x Tx_buffer_element_size

利用下面的流程图来简化 Tx 缓冲区机制的发送操作。

图 15. 采用 Tx 缓冲区机制发送



提示 *如果用相同 ID 配置了多个专用 Tx 缓冲区，则首先发送第一个发送请求的 Tx 缓冲区。*

Tx FIFO

通过向 FDCAN_TXBC 中的 TFQM 位写入 0 来配置 Tx FIFO 操作。从 FDCAN_TXFQS 中的 TFG1[4:0] 字段指示的*获取索引*引用的元素开始，发送 Tx FIFO 中存储的元素。在每次发送后，*获取索引*循环递增，直至 Tx FIFO 缓冲区为空。

Tx FIFO 可按照元素写入 Tx FIFO 的顺序发送元素。由于会首先发送 FIFO 中的第一个/最早的元素，因此发送顺序与各标识符的优先级无关。

FDCAN 通过 FDCAN_TXFQS 中的 TFFL[5:0] 字段计算 Tx FIFO 缓冲区的空闲水平，即*获取*与*放入*索引（*获取*与*放入*索引会随每个事务而递增，以指示 RAM 上要读取或写入元素的下一个缓冲区位置）之间的差值。该值表示可用（空闲）的 Tx FIFO 元素数。

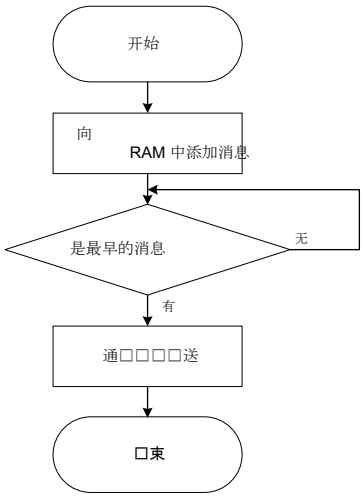
将新的发送元素写入 Tx FIFO 时，必须从*放入*索引引用的 Tx 缓冲区开始，由 FDCAN_TXFQS 中的 TFAQPI [4:0] 字段指示。

使用下面的公式计算 CAN 消息 RAM 中下一个空闲 Tx FIFO 缓冲区的地址：

下一个空闲 Tx FIFO 缓冲区地址 = CAN_message_RAM_base_address + FDCAN_TXBC.TBSA（起始地址）+ FDCAN_TXFQS.TFQPI（获取索引）x Tx_FIFO_element_size

下面的流程图展示了采用 Tx FIFO 机制时的简化版发送操作。

图 16. 采用 Tx FIFO 机制发送



Tx 队列

通过向 FDCAN_TXBC 中的 TFQM 位写入 1 来配置 Tx 队列操作。从具有最小标识符（最高优先级）的 Tx 队列缓冲区开始发送 Tx 队列中存储的元素。

不同于专用 Tx 缓冲区，RAM 上的位置管理是自动且动态的，因此，消息标识符不是固定的预定义 Tx 缓冲区索引。

新消息必须写入*放入*索引引用的 Tx 队列缓冲区。*添加请求*会将*放入*索引循环递增到下一空闲 Tx 队列缓冲区。通过将 FDCAN_TXFQS 中的 TFQF 位置为 1 来指示 Tx 队列已满。在发送至少一个被请求元素或挂起的发送请求被取消之前，不得再向 Tx 队列中写入元素。

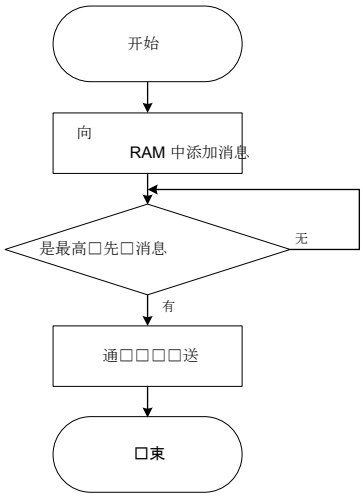
Tx 队列缓冲区的存储要求取决于 Tx 队列元素包含的数据字节数。

可使用下面的公式计算 CAN 消息 RAM 中下一个空闲 Tx 队列缓冲区的地址：

下一个空闲 Tx 队列缓冲区地址 = CAN_message_RAM_base_address + FDCAN_TXBC.TBSA（起始地址）+ FDCAN_TXFQS.TFQPI（放入索引）x Tx_Buffer_element_size

下面的流程图展示了采用 Tx 队列机制时的简化版发送操作。

图 17. 采用 Tx 队列机制发送



专用 Tx 缓冲区、Tx FIFO 和 Tx 队列之间的差异

下表中描述了专用 Tx 缓冲区、Tx FIFO 和 Tx 队列之间的差异。

表 7. 专用 Tx 缓冲区、Tx FIFO 和 Tx 队列之间的差异

特征	专用 Tx 缓冲区	Tx FIFO	Tx 队列
可配置的区	可配置 32 个专用 Tx 缓冲区。	可配置 1 个 Tx FIFO。	可配置 1 个 Tx 队列。
每个区的元素	Tx 缓冲区被配置为每个缓冲区仅包含一个元素	每个区可包含一个或多个元素（最多 32 个）	每个区可包含一个或多个元素（最多 32 个）
将首先发送的元素	首先发送 ID 最小的元素。	按照 FIFO 顺序发送元素	首先发送 ID 最小的元素。
多个元素具有相同 ID 时的处理方式	首先发送第一个发送请求。	标识符无优先次序。	FIFO 顺序
在 RAM 中的位置	用户选择缓冲区索引。	自动并动态地管理（使用*获取*和*放入*索引递增机制）	
挂起元素管理	在添加到 RAM 后或启用 Tx 缓冲区请求后元素开始挂起。	在添加到 RAM 后元素开始挂起（自动启用请求）。	

灵活发送配置

高效的 FDCAN 支持混合配置，使发送更加灵活，并能充分利用每种机制的优势。支持的混合配置为专用 Tx 缓冲区 + Tx FIFO 和专用 Tx 缓冲区 + Tx 队列。

混合配置：专用 Tx 缓冲区和 Tx FIFO

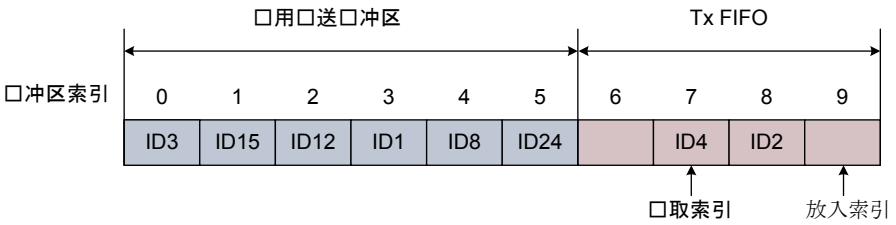
CAN 消息 RAM 的 Tx 缓冲区可采用混合配置。在混合配置中，CAN 消息 RAM 中的 Tx 缓冲区被细分为一组专用 Tx 缓冲区和一个 Tx FIFO。

通过 FDCAN_TXBC 中的 NDTB[5:0] 字段配置专用 Tx 缓冲区数。通过 FDCAN_TXBC 中的 TFQS[5:0] 字段配置分配给 Tx FIFO 的 Tx 缓冲区数。

Tx 处理程序扫描所有具有激活的发送请求的专用 Tx 缓冲区和被*获取*索引引用的最早的挂起 Tx FIFO 缓冲区。具有最小标识符的缓冲区获得最高优先级，并在下一个发送。

使用专用 Tx 缓冲区与 Tx FIFO 的混合配置的用例如下图所示。

图 18. 专用 Tx 缓冲区和 Tx FIFO 的混合配置



在本例中，按照下列顺序发送元素（假定启用了所有专用 Tx 缓冲区请求）：

1. Tx 缓冲区 3（标识符 = 1：所有其他专用 Tx 缓冲区中的最高优先级，其优先级高于最早的挂起 Tx FIFO：Tx 缓冲区 7）
2. Tx 缓冲区 0（标识符 = 3：所有其他专用 Tx 缓冲区中的最高优先级，其优先级高于最早的挂起 Tx FIFO：Tx 缓冲区 7）
3. Tx 缓冲区 7（因为它是最早的挂起 Tx FIFO，标识符 = 4，在所有专用 Tx 缓冲区中具有较高优先级）
4. Tx 缓冲区 8（因为它是最早的挂起 Tx FIFO，标识符 = 2，在所有专用 Tx 缓冲区中具有最高优先级）
5. Tx 缓冲区 4（标识符 = 8：所有其他专用 Tx 缓冲区中的最高优先级，并且 Tx FIFO 为空）
6. Tx 缓冲区 2（标识符 = 12：所有其他专用 Tx 缓冲区中的最高优先级，并且 Tx FIFO 为空）
7. Tx 缓冲区 1（标识符 = 15：所有其他专用 Tx 缓冲区中的最高优先级，并且 Tx FIFO 为空）
8. Tx 缓冲区 5（因为它是唯一的挂起专用 Tx 缓冲区）

混合配置：专用 Tx 缓冲区和 Tx 队列

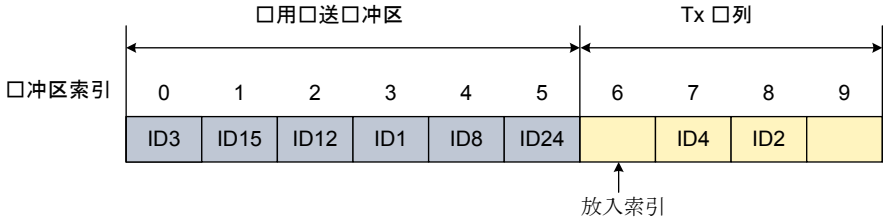
CAN 消息 RAM 的 Tx 缓冲区可采用混合配置。在混合配置中，CAN 消息 RAM 中的 Tx 缓冲区被细分为一组专用 Tx 缓冲区和一个 Tx 队列。

通过 FDCAN_TXBC 中的 NDTB[5:0] 字段配置专用 Tx 缓冲区数，并通过 FDCAN_TXBC 中的 TFQS[5:0] 字段配置 Tx 队列缓冲区数。

Tx 处理程序扫描所有具有激活的发送请求的专用 Tx 缓冲区，以及 Tx 队列缓冲区。具有最小标识符的缓冲区获得最高优先级，并在下一个发送。

下图描述了使用专用 Tx 缓冲区与 Tx 队列 的混合配置的用例。

图 19. 专用 Tx 缓冲区和 Tx 队列的混合配置



在本例中，按照下列顺序发送消息（假定启用了所有专用 Tx 缓冲区请求）：

1. Tx 缓冲区 3（标识符 = 1：所有其他 Tx 缓冲区中的最高优先级）
2. Tx 缓冲区 8（标识符 = 2：所有其他 Tx 缓冲区中的最高优先级）
3. Tx 缓冲区 0（标识符 = 3：所有其他 Tx 缓冲区中的最高优先级）
4. Tx 缓冲区 7（标识符 = 4：所有其他 Tx 缓冲区中的最高优先级）
5. Tx 缓冲区 4（标识符 = 8：所有其他 Tx 缓冲区中的最高优先级）
6. Tx 缓冲区 2（标识符 = 12：所有其他 Tx 缓冲区中的最高优先级）
7. Tx 缓冲区 1（标识符 = 15：所有其他 Tx 缓冲区中的最高优先级）
8. Tx 缓冲区 5（因为它是唯一的挂起 Tx 缓冲区）

提示 *不支持 Tx FIFO 与 Tx 队列的混合配置。*

混合配置 Tx 缓冲区 + Tx FIFO 与混合配置 Tx 缓冲区 + Tx 队列之间的差异

下表列出了混合配置 Tx 缓冲区 + Tx 队列与 Tx 缓冲区 + Tx FIFO 之间的主要差异。

表 8. 混合配置 Tx 缓冲区 + Tx FIFO 与混合配置 Tx 缓冲区 + Tx 队列之间的差异

特征	混合配置：Tx 缓冲区 + Tx 队列	混合配置：Tx 缓冲区 + Tx FIFO
说明	专用 Tx 缓冲区与 Tx 队列的组合	专用 Tx 缓冲区与 Tx FIFO 的组合
将首先发送的元素	所有专用 Tx 缓冲区和 Tx 队列中具有最小 ID 的元素	所有专用 Tx 缓冲区中具有最小 ID 的元素和 Tx FIFO 中最早的元素
具有相同 ID 的多个元素的管理	首先发送第一个请求。	顺序请求 FIFO

提示 *在 Tx FIFO + Tx 队列中，在“添加到 RAM”操作后元素开始挂起。*

4.3.4 测试模式

在 FDCAN 的工作模式中，除了普通工作模式，还提供了几种测试模式。必须对校准单元使用这些测试模式，且只能用于生产测试或自检。

为了启用对 FDCAN 测试寄存器的写访问以及测试模式和功能的配置，必须将 FDCAN_CCCR 中的 TEST 位置为 1。

FDCAN 以下列模式之一工作：

- 受限工作模式

- 总线监控模式
- 外部环回模式
- 内部环回模式

4.3.4.1 *受限工作模式*

在受限工作模式下，FDCAN 能够：

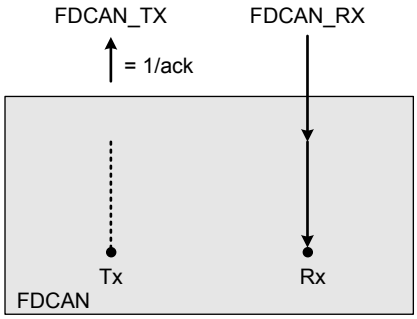
- 接收数据帧
- 接收远程帧
- 确认有效帧

该模式不支持：

- 数据帧发送
- 远程帧发送
- 活动错误帧或过载帧发送

通过 FDCAN_CCCR 寄存器中的 ASM 位将 FDCAN 置为受限工作模式。
当 Tx 处理程序无法及时从 CAN 信息 RAM 读取数据时或时钟校准激活时，将自动进入受限工作模式。
在该模式下，应用会测试不同比特率，并在应用收到有效帧后退出受限工作模式。
下图所示为受限工作模式下 FDCAN_TX 与 FDCAN_RX 引脚的连接。

图 20. 受限工作模式下的引脚控制



提示 *只要 FDCAN 处于受限工作模式，FDCAN_TX 即为隐性。发送显性位以确认有效帧的接收。*

4.3.4.2 *总线监控模式*

在分析总线上的通信量时，为避免受到显性位发送的影响，用户可通过 FDCAN_CCCR 中的 MON 位将 FDCAN 设置为总线监控模式。
在总线监控模式下，FDCAN 能够：

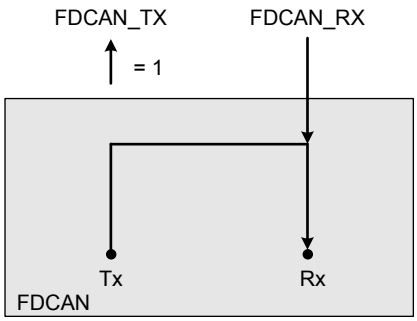
- 接收有效数据帧
- 接收有效远程帧

该模式不支持：

- 发送开始
- 确认有效帧（不同于受限工作模式）

在总线监控模式下，FDCAN 只通过总线发送隐性位。下图所示为总线监控模式下 FDCAN_TX 与 FDCAN_RX 引脚的连接。

图 21. 总线监控模式下的引脚控制



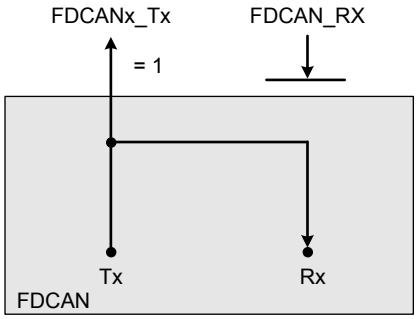
4.3.4.3 *外部环回模式*

该模式为硬件自检提供。通过向 FDCAN_TEST 中的 LBCK 位写入 1 并向 FDCAN_CCCR 中的 MON 位写入 0，用户可以将 FDCAN 设置为外部环回模式。FDCAN 将自身发送的信息视为接收到的信息，如果它们通过了接受过滤，则将它们存储在 Rx FIFO 中。

为了不受外部激励的影响，FDCAN 会忽略回执错误（在回执时隙中采样的隐性位）。FDCAN 执行从其“发送”输出到其“接收”输入的内部反馈。

下图所示为外部环回模式下 FDCAN_TX 和 FDCAN_RX 引脚的连接。

图 22. 外部环回模式下的引脚控制



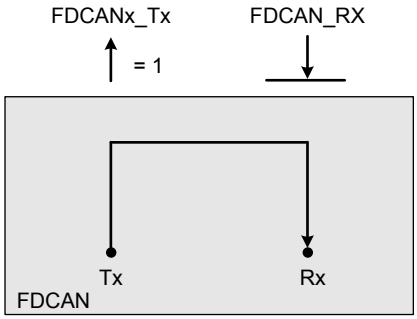
4.3.4.4 *内部环回模式*

该模式为硬件自检提供。通过向 FDCAN_TEST 中的 LBCK 位写入 1 并向 FDCAN_CCCR 中的 MON 位写入 1，用户可以将 FDCAN 设置为内部环回模式。

可在不影响正在运行的连接到 FDCAN_TX 和 FDCAN_RX 引脚的 CAN 系统的情况下，测试 FDCAN。FDCAN_RX 引脚与 FDCAN 断开连接，FDCAN_TX 引脚则保持隐性。

下图所示为内部环回模式下 FDCAN_TX 和 FDCAN_RX 引脚的连接。

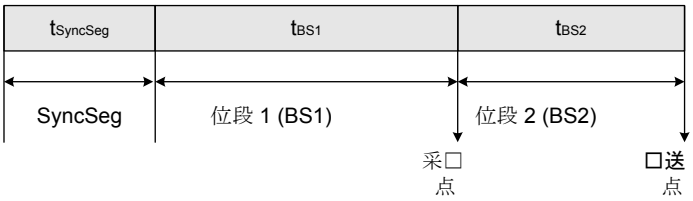
图 23. 内部环回模式下的引脚控制



4.3.5 收发器延迟补偿

在采样点，所有发送器都会检查，之前发送的位是否经过正确采样。需使用此机制检查是否存在问题，并检测其他节点错误帧。由于发送器检测到收发器回路延迟导致其自身发送的位延迟，此延迟为 TSEG1 设置了下限（采样点前的时间段），如下图所示，同时也是数据比特率的上限。由此引入了发送器延迟补偿机制（TDC）的原因。

图 24. 位时序



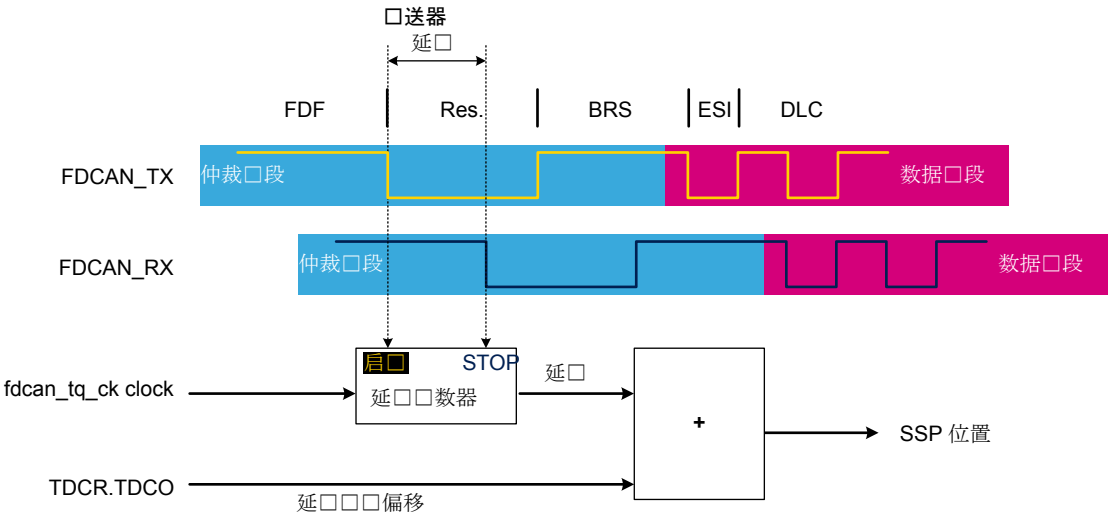
为了在检查位错误时补偿此回路延迟，定义了第二采样点（SSP）。在 SSP 而非采样点检查发送的位。保存检查结果，直至到达下一个采样点。

在数据阶段为发送的每个位生成 SSP。对于 SSP 位置，必须考虑收发器不对称和总线上的振铃效应，但由于收发器监控其自身的位流，因此没有时钟容差。

通过向 FDCAN_DBTP 中的 TDC 位写入 1 启用收发器延迟补偿。在发送的每个 FDCAN 帧的数据阶段开始前开始测量（在 FDF 位与预留位之间的下降沿）。当在发送器的“接收”输入引脚 FDCAN_RX 看到此边沿时，测量停止。此类测量的分辨率为 1 mtq（最小时间量子）。

下图所示为回路延迟的测量。

图 25. 回路延迟测量



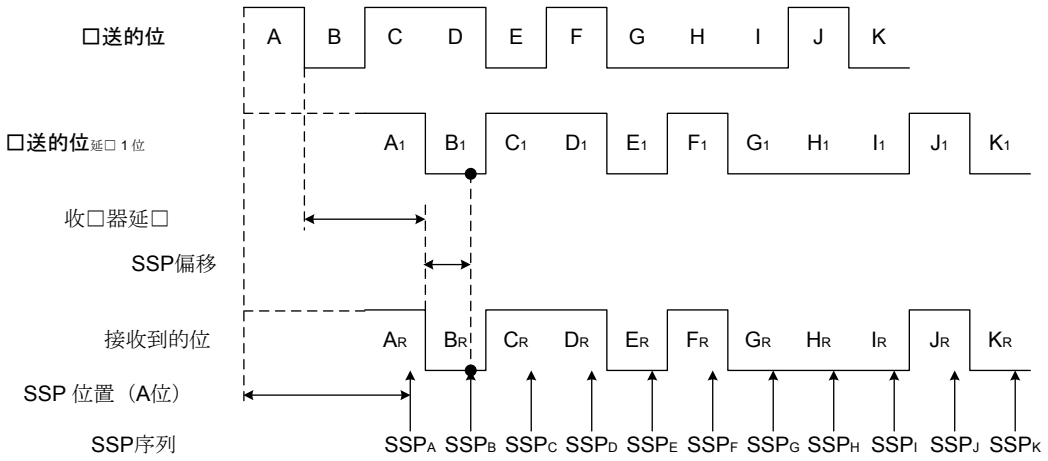
提示 *在仲裁阶段，始终禁用延迟补偿。*

SSP 位置定义为从 FDCAN_TX 引脚到 FDCAN_RX 引脚测得的延迟与发送器延迟补偿偏移（通过 FDCAN_TDCR 中的 TDCO[6:0] 字段配置）之和。

提示 *发送器延迟补偿偏移用于调整接收到的位内部的 SSP 位置。*

存储发送的位的值，直至到达 SSP，然后将其与实际接收到的位值进行比较，如下图所示，图中显示了发送的位序列 A 至 K 和接收到的位序列 AR 至 KR，以及 SSP 序列 SSPA 至 SSPK。在 SSPB 检查接收到的 BR 位，方法是将其与延迟的 B1 位进行比较。SSPB 的位置位于发送的 B 位开始后的特定时间。此特定时间是测得的收发器延迟与配置的 SSP 偏移之和。

图 26. 发送延迟补偿中的 SSP 位置



根据博世提供的文档，对于 FDCAN 中实现的发送器延迟补偿，必须考虑下列边界条件：

- 在数据阶段，从 FDCAN_Tx 到 FDCAN_Rx 测得的延迟与配置的发送器延迟补偿偏移之和必须小于六个位的时间。
- 从 FDCAN_Tx 到 FDCAN_Rx 测得的延迟与配置的发送器延迟补偿偏移之和必须小于或等于 127 mtq。

提示 *如果二者之和超过 127 mtq，则对发送器延迟补偿使用最大值（127 mtq）。*

- 数据阶段在 CRC 分隔符的采样点结束，此时会停止检查在 SSP 接收到的位。

时钟校准单元及其效用和操作见本文档的下一部分。

4.3.6 FDCAN 时钟校准

FDCAN 支持时钟校准单元（CCU）功能。该功能允许用户通过 FDCAN 发送器（主机）校准 FDCAN 接收器（设备）。例如，当 FDCAN 设备用主机的最新比特率进行通信时，

该功能允许用户在总线中添加新实例，并且比特率的存在是未知的。当 FDCAN 接收器没有精密石英（可能导致时间误差）时，该功能也十分有用。

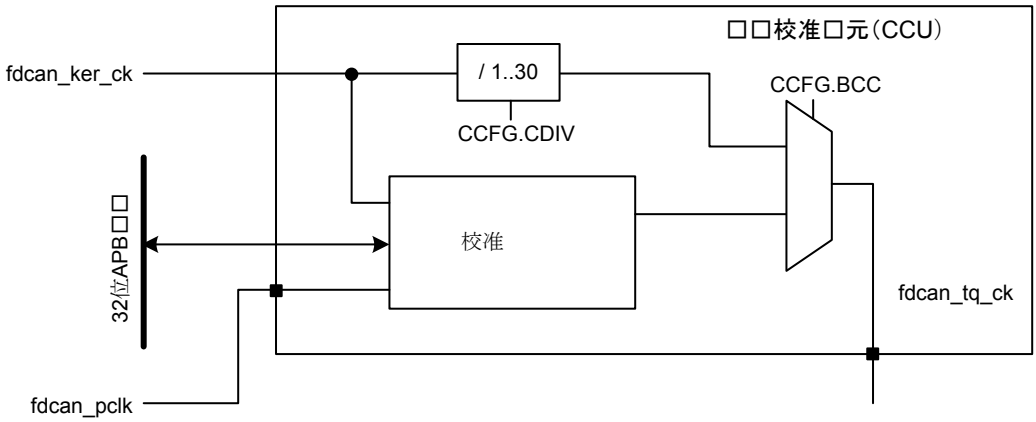
4.3.6.1 CCU 描述

时钟校准单元通过 FDCAN_CCU_CCFG 寄存器进行初始化，该寄存器只能在 FDCAN_CCCR 中的 CCE 和 INIT 位均置为 1 时写入。

仅当 FDCAN 以 CAN 2.0 模式工作时，才能使用 CCU。

当 FDCAN_CCU_CCFG 中的 BCC = 1 时，会对时钟校准进行旁路。下图显示了旁路操作。

图 27. CCU 的旁路操作



4.3.6.2 CCU 工作条件

CCU 仅在 FDCAN 比特率介于 125 Kbit/s 和 1 Mbit/s 之间时工作。

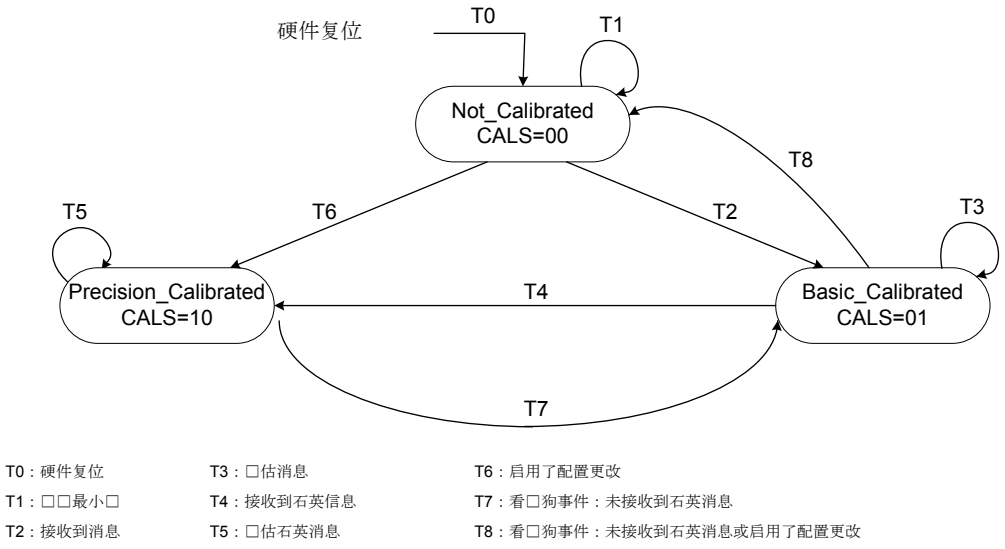
4.3.6.3 CCU 功能说明

通过适配时钟分频器，可以通过 CAN 消息对 fdcan_tq_ck（量子时钟）进行校准，时钟分频器从 fdcan_ker_ck 时钟生成 CAN 协议时间量 tq。

状态机的校准

功能状态机的校准如下图所示。

图 28. 功能状态机的校准



基础校准

测量从隐性到显性的两个连续下降沿之间的最小距离。此次测量假设以 PLL 时钟周期为单位计算两个 CAN 位时间。每当新的测量发现更小距离（两个下降沿之间）时，通过 FDCAN_CCU_CCFG 中的 CDIV [3:0] 字段更新时钟分频器。当 CAN 协议控制器检测到有效 CAN 信息时，则基础校准完成。

精确校准

校准状态机通过计算 fdcan_ker_ck 周期数来测量 CAN 帧中较长的位序列长度。此位序列的长度可通过 FDCAN_CCU_CCFG 寄存器中的 CFL 位配置为 32 或 64 位。精确校准基于新的时钟分频器值，是从更长位序列的测量值计算得来的。

校准帧会被 FDCAN 接受过滤检测到。必须在 FDCAN 中配置过滤器元素和 Rx 缓冲区，以便识别和存储校准信息。

如果 fdcan_ker_clk 校准是通过软件完成的（从 FDCAN_CCU_CSTAT 寄存器的 CALS[1:0] 字段评估校准状态），则 FDCAN 必须设置为受限工作模式，直至校准处于 Precision_Calibrated 状态（无帧，且无错误或过载标记发送，无错误计数）。

提示 *接收到校准信息后，必须将 Rx 缓冲区的新数据标记复位，以便传递下一条校准信息的信号。*

为确保设备节点可以进入 Basic_Calibrated 状态以及主机节点信息得到确认，校准信息的数据场须至少为 1010 二进制序列。

只能对使用石英控制的稳定时钟的主机节点发送的有效 CAN 帧执行精确校准。必须在由校准看门狗监视的预定义最大时间间隔内重复执行精确校准。

校准看门狗

校准看门狗是递减计数器，它从非 Not_Calibrated 状态开始，并监控接收到的信息。

当处于 Basic_Calibrated 状态时，每接收到一条信息，校准看门狗就会重新启动。

提示 *如果在校准看门狗递减计数到零之前未接收到任何信息，则 FSM 校准保持 Not_Calibrated 状态。计数器被重载，重新开始基础校准。*

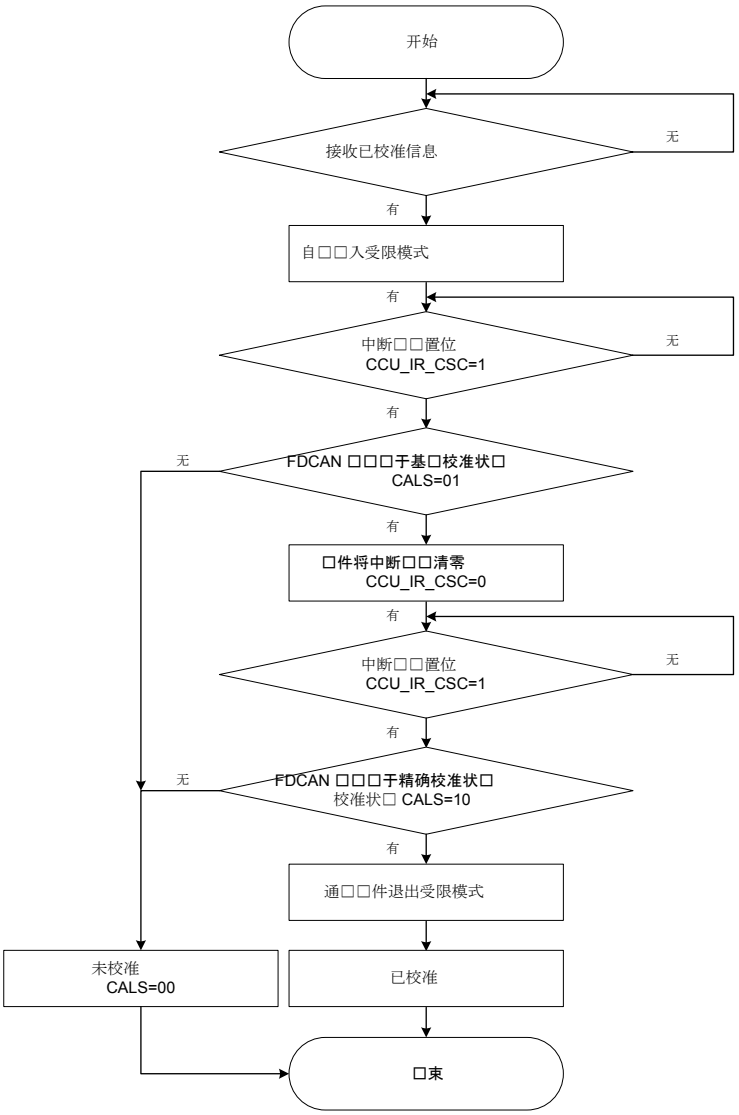
当进入 Precision_Calibrated 状态时，重新启动校准看门狗。在该状态下，校准看门狗会监控接收到的石英信息。

提示 *如果 FDCAN 直到校准看门狗递减计数达到零都没有收到来自石英控制的节点的任何信息，则 FSM 校准将回到“Basic_Calibrated”状态。*

4.3.6.4 *校准示例*

该示例展示了一个通过 FDCAN 主机（发送器）校准 FDCAN 设备（接收器）的用例。下面的流程图描述了校准 FDCAN 设备的步骤。

图 29. 校准 FDCAN 设备的步骤



在成功完成校准后，FDCAN 设备准备就绪，可以用新的 FDCAN 主机比特率接收和发送信息。

5 FDCAN 相较于 BxCAN 实现的改进

下表可帮助用户简化 STM32 器件中 CAN 2.0 协议到 CAN-FD 协议的升级。此表还具体说明了 FDCAN 的改进之处。

与传统的 BxCAN（基本扩展 CAN）相比，FDCAN 具有许多优势，包括更快的数据率和数据字节数的扩展（减少了帧开销）。此外，还可以减少总线负载。处于发送和接收状态的消息数量的增加要求改进 RAM 存储器。

表 9. FDCAN 相较于 BxCAN 的改进

特征		BxCAN	FDCAN
兼容性	与 BxCAN 兼容	有	有
	与 CAN-FD 兼容	无	无
	远程帧	无	支持，以便与 BxCAN 兼容
帧	仲裁比特率 /数据率（Mbit/s）	最高 1（仅每帧的比特率）	多达 1 /最高 8
	每帧的数据长度（字节）	0 至 8	0 至 64
RAM	RAM	512 字节	10 KB
	可存取 RAM	无	有
传输	专用 Tx 缓冲区/Tx 队列/Tx FIFO	否/是/是 最多可以有 3 个元素用作队列或 FIFO。	是/是/是 最多 32 个元素：用户选择发送机制。
	发送暂停	无	有
接收	专用 Rx 缓冲区/Rx FIFO	否/2 每个 Rx FIFO 中最多 3 个元素 最多 6 个元素	64/2 每个 Rx FIFO 中最多 64 个元素 最多 192 个元素
	覆盖模式选项	覆盖 FIFO 中接收到的最后一个元素	覆盖 FIFO 中最早的元素
	接受过滤器的改进	传统 CAL（Can 2.0）接受过滤器	增加了一些特性： • 丢弃匹配过滤器。 • 接受不匹配过滤器。
其它	受限测试模式	无	有
	收发器延迟补偿	无	有
	时钟校准单元	无	有

鉴于 BxCAN 的兼容性，无需对整个系统设计进行修订即可实现 FDCAN，因此 BxCAN 开发人员可以轻松地迁移到 FDCAN。FDCAN 包含 BxCAN 的所有特性并进行了改进，并满足新应用的要求。

版本历史

表 10. 文档版本历史

日期	版本	变更
2019 年 10 月 4 日	1	初始版本。

目录

表一览

图一览