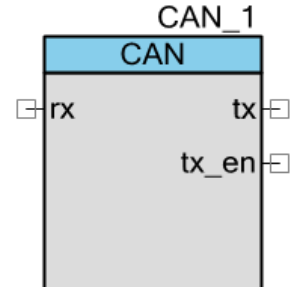


Controller Area Network (CAN)

2.30

特性

- CAN 2.0A 和 CAN 2.0B 协议实现，符合 ISO 11898-1 标准
- 在 8 MHz（BUS_CLK）时高达 1 Mbps 的可编程比特率
- 连接外部收发器的两线或三线接口（Tx、Rx 和 Enable（使能））
- 扩展硬件信息过滤器，涵盖“数据字节 1”和“数据字节 2”字段
- 可编程发送优先级：轮循和固定



概述

控制器区域网络（CAN）控制器符合 Bosch 规范中定义的 CAN 2.0A 和 CAN 2.0B 规范，并符合 ISO-11898-1 标准。

何时使用 CAN

CAN 协议最初是针对汽车应用设计的，侧重于高阶的故障检测。能够确保以较低的成本实现高度的通信可靠性。由于在汽车应用中取得了巨大成功，CAN 被用作运动机械控制网络（CANOpen）和工厂自动化应用（DeviceNet）的标准通信协议。CAN 控制器具有丰富的功能，能够高效实现更高级的协议，而不会影响微控制器 CPU 的性能。

输入/输出接口

本节介绍了 CAN 组件的各种输入和输出接口。I/O 列表中的星号（*）表示，在 I/O 说明部分中所列出的情况下，该 I/O 可能不可见。

rx — 输入

CAN 总线接收信号（连接到外部收发器的 CAN Rx 总线）。

tx — 输出

CAN 总线传输信号（连接到外部收发器的 CAN Tx 总线）。

tx_en — 输出*

外部收发器使能信号。当在 **Configure**（配置）对话框中选择 **Add Transceiver Enable Signal**（添加收发器使能信号）选项时，将显示此输出。

中断 — 输出*

此中断输出由CAN硬件中的配置的中断源驱动。对所有源进行“或”运算以创建最终输出信号。中断的源包括：

- 信息已发送
- 接收到信息
- 接收缓冲区已满
- 总线关闭状态
- 检测到 CRC 错误
- 检测到信息格式错误
- 信息应答，检测到错误
- 检测到位填充错误
- 检测到误码
- 接收到过载帧
- 检测到仲裁失败

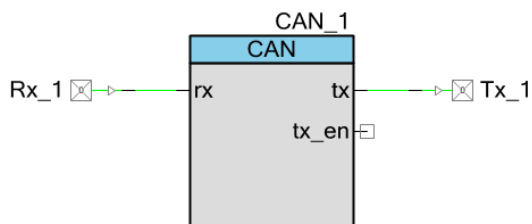
当 **Configure**（配置）对话框中 **Interrupt**（中断）选项卡的 **Advanced Interrupt Configuration...**（高级中断配置...）窗口中选择了 **Enable External Interrupt Line**（使能外部中断线）选项时，显示此输出。

在启动之前，CAN 组件不能识别错误，也不能向更高级别的软件记录错误（时间为 0 时，RX 短路接地导致该错误）。如果在 RX 上检测不到下降沿，CAN 状态机处于闲置状态。

启动 CAN 组件之前，更高级别的软件需要确定时间为 0 时的总线短路。

原理图宏的信息

组件目录中的默认 CAN 是使用带默认设置的 CAN 组件的原理图宏。CAN 组件连接到输入和输出引脚组件。除了输入引脚组件中的 Input Synchronized（输入同步）被设为假以外，引脚组件也被配置为默认设置。



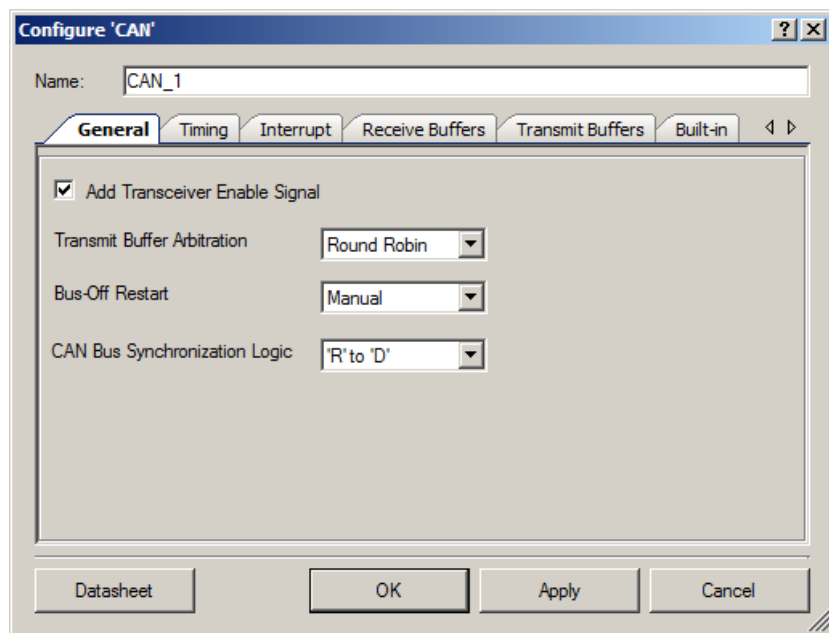
组件参数

将 CAN 组件拖入设计中，双击它以打开 **Configure**（配置）对话框。该对话框有许多选项卡，可引导您完成 CAN 组件的设置过程。

组件更新笔记

如果从之前的版本更新 CAN 组件，很多参数采用新的格式并必须进行转换。为此，打开 **Configure**（配置）对话框，至少更改一个参数选项，并单击 **OK**（确定）以保存更改。

General 选项卡



General 选项卡包含以下设置：



添加收发器使能信号

为外部 CAN 收发器使能或禁用 tx_en 信号的使用。默认使能。

发送缓冲区仲裁

定义信息发送仲裁方案：

- **轮循**（默认） — 缓冲区以所定义的顺序运行：**0-1-2 ... 7-0-1**。特定的缓冲区仅在设置了其 TxReq 标志时可选择。此方案保证了所有缓冲区具有同等的信息发送概率。
- **固定优先级** — 缓冲区 **0** 具有最高优先级。这样，可将缓冲区 **0** 指定为错误信息的缓冲区，保证最先发送错误信息。

总线关闭重新启动

用于配置复位类型：

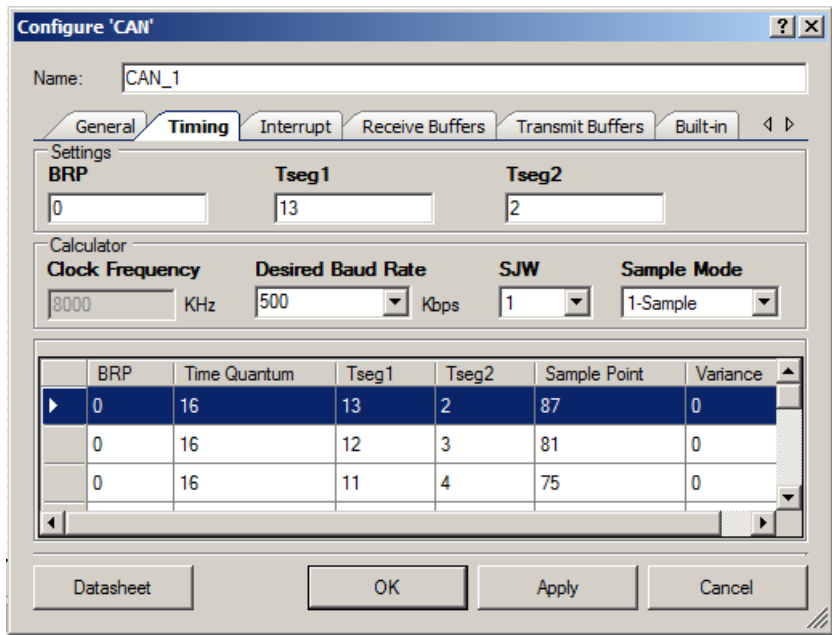
- **手动**（默认） — 总线关闭之后，必须重新启动 **CAN**。这是推荐设置。
- **自动** — 总线关闭之后，在 **128** 组 **11** 个“空闲”位之后，自动重新启动 **CAN** 控制器。

CAN 总线同步逻辑

用于配置边沿同步：

- **‘R’ 到 ‘D’**（默认） — 从 **‘R’**（空闲）到 **‘D’**（占有）的边沿用于进行同步。
- **双边沿** — 双边沿用于进行同步。

Timing 选项卡

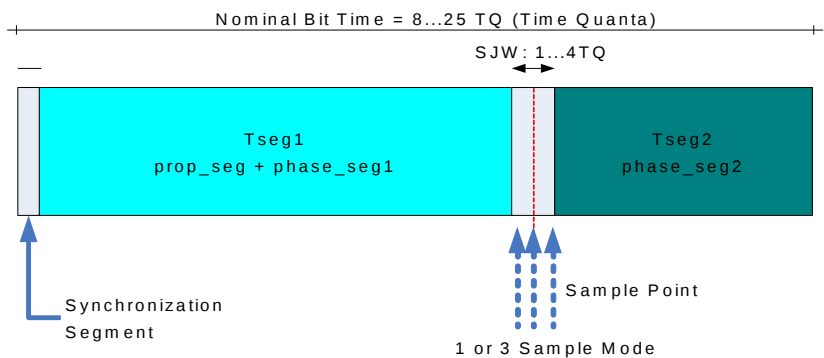


Timing（定时）选项卡包含以下设置：

Settings（设置）

- **BRP** — 用于生成时间段的比特率预分频器值。位定时计算器用于计算此值。0 表示 1 个时钟周期；7FFFh 表示 32768 个时钟周期，15 个位。
- **Tseg1** — 时间段 1 的值。
- **Tseg2** — 时间段 2 的值。不允许值 0 和 1；仅在将 **Sample Mode**（样本模式）设置为直接采样（**1-Sample**）时才允许值 2。

下面显示了 CAN 位定时图示例：



计算器

- **时钟频率**（单位：MHz） — 系统时钟频率等于 BUS_CLK。
- **需要的波特率**（单位：Kbps） — 选项为：**10、20、62.5、125、250、500、800 或 1000**。
- **SJW** — 同步跳转宽度的配置（2 位）。值不得大于 Tseg1 且不得大于 Tseg2。选项为：**1、2、3 或 4**。
- **样本模式** — 采样模式的配置。选项为：**1-Sample 或 3-Sample**。

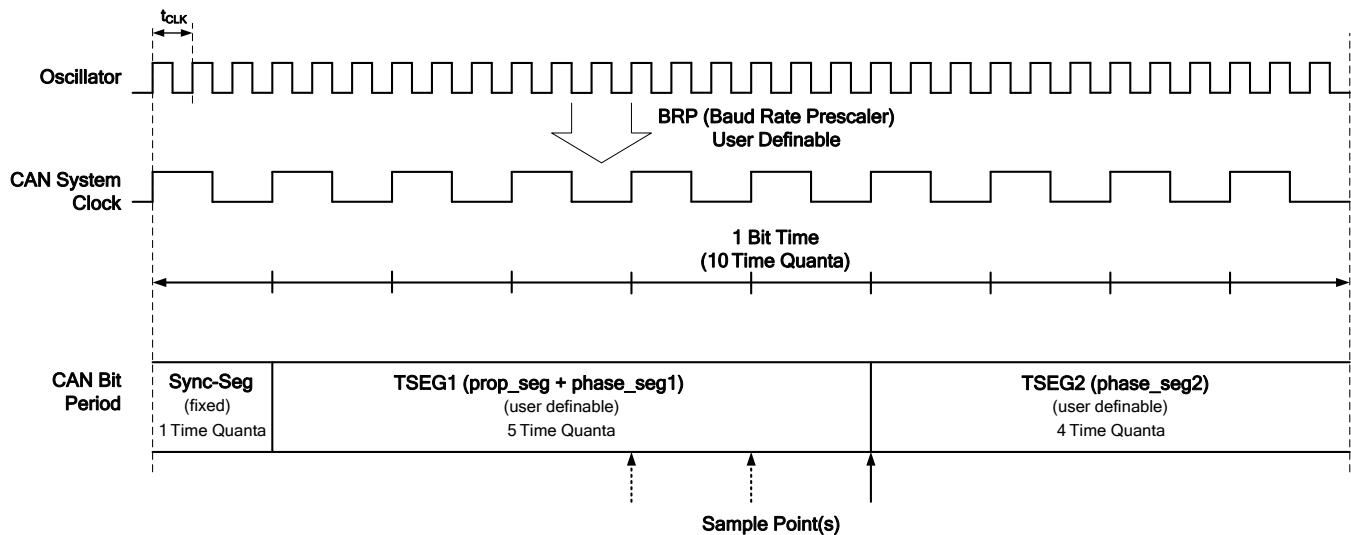
表格

在计算了位定时之后，参数表中显示了时间段（Tseg1 和 Tseg2）和 BRP 的推荐寄存器设置。可通过双击相应的行选择要加载的值。选择的值显示在顶部的 **Settings**（设置）输入框。

也可以选择提供的输入框手动输入 Tseg1、Tseg2 和 BRP 的值。

注意：错误的位定时设置会导致 CAN 控制器保留在错误状态。

下图显示了所有定时是如何衍生自振荡器的示例。



位时间段

SYNC SEG（同步段）

此位时间部分使总线上的各个节点保持同步。一个边沿预计位于此段中。

PROP SEG（传输时间段）

此位定时部分用于补偿网络中的物理延迟时间。它是总线上信号的传播时间、输入比较器延迟和输出驱动器延迟的总和的两倍。

PHASE SEG1、PHASE SEG2（相位缓冲区间 1/2）

这些相位缓冲区间用于补偿边沿相位错误。可通过重新同步延长或缩短这些区间。

样本点

样本点是总线级别读取并解释为对应位的值的时间点。它位于 PHASE_SEG1 结尾处。

信息处理时间

信息处理时间是开始于为计算后续位级别而保留的样本点的时间段。

时间段

时间段是从震荡周期中衍生的固定时间单位。有一个可编程预分频器（BRP），整数值范围为1到32768。从最小时间段开始，时间段长度可为

$$\text{时间段} = m \times \text{最小时间段}$$

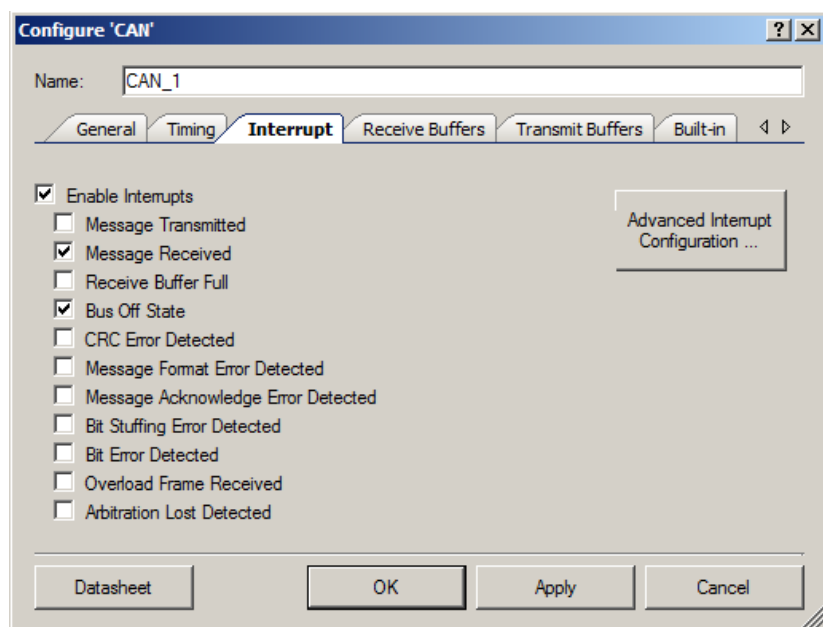
其中 m 是预分频器的值。

时间段长度

- SYNC_SEG 的长度为 1 个时间段。
- PROP_SEG 的长度可编程为 1、2、...、8 个时间段。
- PHASE_SEG1 的长度可编程为 1、2、...、8 个时间段。
- PHASE_SEG2 是 PHASE_SEG1 的最大值和信息处理时间。

中断选项卡

基本中断配置



Basic Interrupt Configuration（基本中断配置）选项卡包含以下设置：

使能中断

从 CAN 控制器中使能或禁用全局中断。默认使能。

- **使能** — 在使用 `CAN_1_Start()` 启动 CAN 组件时，使能全局中断。
- **禁用** — 在使用 `CAN_1_Start()` 启动 CAN 组件时，不使能全局中断。全局中断使能位设置前不会进入 CAN ISR。您应使用 `CAN_1_GlobalIntEnable()` 或 `CAN_1_GlobalIntDisable()` 在主代码中使能或禁用全局中断。

信息已传输

使能或禁用信息传输中断。默认禁用。表示已发送了一个信息。禁用信息传输中断选项时，CAN 显示以下信息：**是否禁用所有传输缓冲区中断？**

- **是** — 取消选中 **Message Transmitted**（信息已传输）复选框，并取消选中 **Transmit Buffers**（传输缓冲区）选项卡上的所有单独传输缓冲区中断。
- **否（默认）** — 取消选中 **Message Transmitted**（信息已传输）复选框，并保持 **Transmit Buffers**（传输缓冲区）选项卡上的所有单独传输缓冲区中断不变。
- **取消** — 无更改。

已接收到信息

使能或禁用信息接收中断。默认使能。表示已接收了一个信息。禁用信息接收中断选项时，CAN 显示以下信息：**是否禁用所有接收缓冲区中断？**

- **是**（默认）— 取消选中 **Message Received**（信息已接收）复选框，并取消选中 **Receive Buffers**（接收缓冲区）选项卡上的所有单独接收缓冲区中断。
- **否** — 取消选中 **Message Received**（信息已接收）复选框，并保持 **Receive Buffers**（接收缓冲区）选项卡上的所有单独接收缓冲区中断不变。
- **取消** — 无更改。

接收缓冲区已满

使能或禁用信息丢失中断。表示在上一条信息未被确认时，接收到一条新信息。默认禁用。

总线关闭状态

使能或禁用总线关闭中断。表示 CAN 节点已达到总线关闭状态。默认使能。

检测到 CRC 错误

使能或禁用 CRC 错误中断。表示检测到 CAN CRC 错误。默认禁用。

检测到信息格式错误

使能或禁用信息格式错误中断。表示检测到 CAN 信息格式错误。默认禁用。

检测到信息确认错误

使能或禁用信息确认错误中断。表示检测到 CAN 信息确认错误。默认禁用。

检测到位填充错误

使能或禁用位填充错误中断。表示检测到位填充错误。默认禁用。

检测到位错误

使能或禁用位错误中断。表示检测到位错误。默认禁用。

接收到过载帧

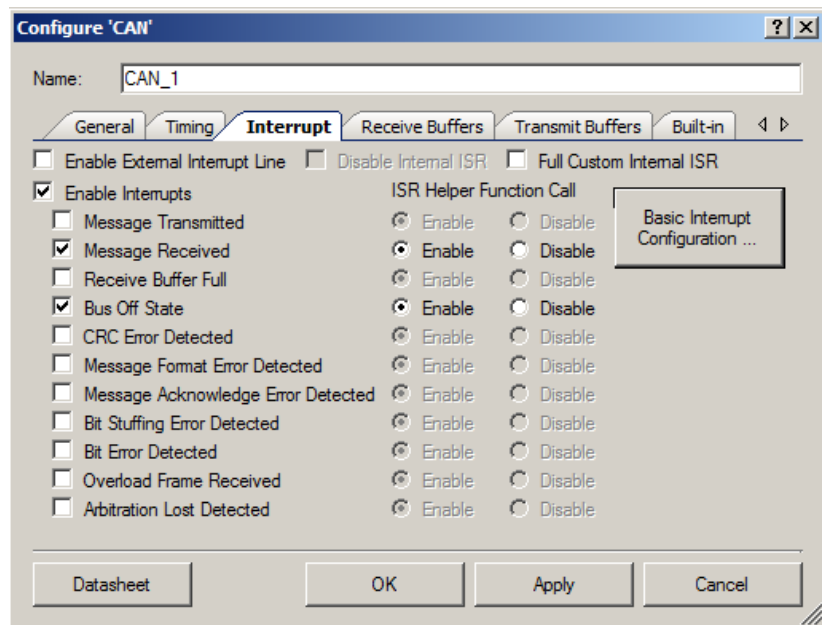
使能或禁用过载中断。表示接收到一个过载帧。默认禁用。



检测到仲裁失败

使能或禁用管理已排队信息的仲裁和取消。表示发送信息时丢失仲裁。默认禁用。

Advanced Interrupt Configuration选项卡



Advanced Interrupt Configuration（高级中断配置）选项卡包含以下设置：

使能外部中断线

使能 CAN 模块中断线的外部可视性和连接。默认取消（在 CAN 组件符号实例中，外部中断线不可见）。

禁用内部 ISR

禁用或旁路内部 ISR 组件。如果禁用了内部 ISR，相关 CAN API 不会处理 ISR 启动/停止流程。默认取消（使能内部 ISR）。仅在选定了 **Enable External Interrupt Line**（使能外部中断线）时，此复选框才可用（未变灰）。仅在有备用方式（外部中断线）处理中断时，才可禁用内部 ISR。

完全自定义内部 ISR

使能使用带完全自定义代码的内部 ISR。当选定此选项时，CAN_1_ISR 不包含代码。将您的自定义代码置于行间：

```
/* Place your Interrupt code here. */
/* `#START CAN_ISR` */

/* `#END` */
```

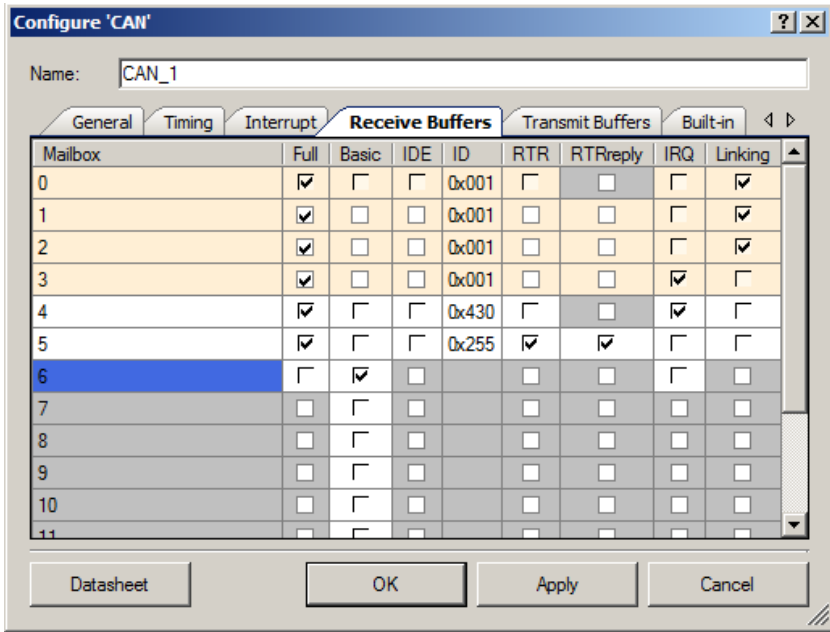
默认不选择（默认为 CAN v1.50 ISR 处理）。只在选定了 **Disable Internal ISR**（禁用内部 ISR），才能使用该复选框（非变灰）。

ISR 助手功能调用

如果您仅选择了基本中断设置（例如，如 CAN v1.50 中），当发生中断时，CAN ISR 基于已使能的中断调用相关用户可定制功能（ISR 助手）。

这些选项允许您使能或禁用 ISR 助手调用，从而可在硬件和固件中都可实施对特定中断的自定义处理。默认**使能**。仅在使能相关中断事件，未选定 **Full Custom Internal ISR**（完全自定义内部 ISR），且未选定 **Disable Internal ISR**（禁用内部 ISR）时，这些选项才可用（非灰色）。

Receive Buffers 选项卡



Receive Buffers（接收缓冲区）选项卡包含以下设置：

邮箱

接收邮箱被禁用，直至选择了 **Full**（为满）或 **Basic**（基本）。所有已禁用邮箱的 **IDE**、**ID**、**RTR**、**RTRreply** 和 **IRQ** 字段都被锁定。

对于为满邮箱，**Mailbox**（邮箱）字段可编辑，您可输入一个唯一的信息名称。用于处理各个邮箱的 API 将附加邮箱字符串。可用符号有：**A–Z**、**a–z**、**0–9** 和 **_**。如果您输入错误的名称，将显示错误信息，且 **Mailbox**（邮箱）字段返回默认值。



Full（为满）

选定了 **Full**（为满）时，可修改 **Mailbox**（邮箱）、**IDE**、**ID**、**RTR**、**RTRreply**、**IRQ** 和 **Linking**（链接）字段。默认选择中包括下列各选项：

- **邮箱** = 邮箱编号 0 到 15
- **IDE** = 已清除
- **ID** = 0x001
- **RTR** = 已清除
- **RTRreply** = 已清除和已锁定（仅在选择了 **RTR** 时使能）
- **IRQ** = 已选定，仅在选择了 **Message Received**（**Interrupt** 选项卡）中断时可用
- **链接** = 已清除

基本

如果选择了 **Basic**（基本），选项 **IDE**、**ID**、**RTR** 和 **RTRreply** 不可用。默认选择内容包括下列选项：

- **IDE** = 已清除（不可用）
- **ID** = <全部>（不可用）
- **RTR** = 已清除（不可用）
- **RTRreply** = 已清除（不可用）
- **IRQ** = 已清除，仅在选择了 **Message Received**（信息已接收）中断时可用
- **链接** = 已清除

IDE

当 **IDE** 复选框被清除时，标识符长度限制为 11 位（0x001 到 0x7FE）。选择了 **IDE** 时，标识符长度限制为 29 位（0x00000001 到 0x1FFFFFFE）。

RTR — 远程传输请求

仅适用于设置为接收完全 **CAN** 信息的邮箱。选择此选项后，将配置接受过滤设置，以仅允许接收已设置了 **RTR** 位的信息。

RTRreply — 远程传输请求自动回复

仅适用于设置为接收完全 CAN 信息，并设置了 RTR 位的邮箱。选择此选项后，将用接收缓冲区的内容自动回复 RTR 请求。

IRQ

针对邮箱使能 IRQ 时，如果清除了 **Interrupt**（中断）选项卡中的 **Message Received Interrupt**（信息接收中断），将显示以下信息：全局“信息接收中断”已禁用。是否使能它？

- **是** — 选择 **IRQ** 复选框，并选择 **Interrupt**（中断）选项卡上的 **Message Received Interrupt**（信息接收中断）复选框。
- **否或取消** — 选择 **IRQ** 复选框，并保持 **Interrupt**（中断）选项卡上的 **Message Received Interrupt**（信息接收中断）复选框不变。

链接

选择 **Linking**（链接）复选框，可链接多个连续的接收邮箱，以创建接收邮箱阵列。此阵列与接收 FIFO 的功能类似。同一阵列的所有邮箱必须具有相同的信息过滤设置，也就是说，验收屏蔽寄存器（AMR）和验收编码寄存器（ACR）一致。

- 阵列的最后一个邮箱不可设置链接标志。
- 最后一个邮箱 15 不可设置链接标志。
- 所有已链接的邮箱用同一种颜色突出显示。
- 链接中只有第一个邮箱可编辑。所有参数自动应用到同一阵列中所有链接的邮箱。
- 针对所有已链接邮箱生成一个函数。

接收信息函数

每个为满 RX 邮箱都有一个预定义的 API。*CAN_1_TX_RX_func.c* 项目文件中有函数列表。这些函数是按照条件编译的，这取决于接收邮箱的设置。只有定义为为满的邮箱才编译各自的函数。

宏观标识符 **CAN_1_RXx_FUNC_ENABLE** 定义是否编译函数。定义在 *CAN_1.h* 项目文件中显示。

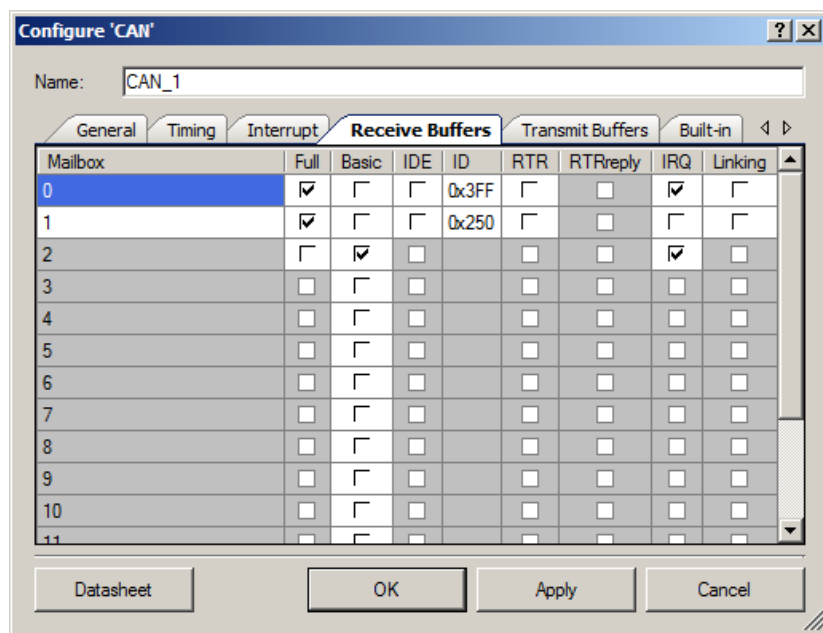
- 当发生信息接收中断时，将调用 **CAN_1_MsgRXIsr()** 函数。此函数依次通过所有接收邮箱，并检查它们各自的“信息可用标志”（**MsgAv** — 读取：0 无新信息可用；1 有新信息可用）和“中断使能”（接收中断使能：0 禁用中断生成；1 使能中断生成），以成功接收 CAN 信息。
- 如果使能了 **Message Receive**（信息接收）中断，当接收到信息时，将调用 **CAN_1_ReceiveMsgX()** 函数，其中 X 表示完全 CAN 邮箱编号或用户定义的名称。



- 对于所有基于中断的基本 CAN 邮箱，将调用 `CAN_1_ReceiveMsg(uint8 rxMailbox)` 函数，其中，`rxMailbox` 参数表示接收到信息的邮箱的编号。

接收缓冲区配置

以下示例说明接收信息 API 的用法。



CAN_1.h 文件:

```
...
#define CAN_1_RX0_FUNC_ENABLE 1
#define CAN_1_RX1_FUNC_ENABLE 1
#define CAN_1_RX2_FUNC_ENABLE 0
...
```

CAN_1_TX_RX_func.c 文件:

```
#if (CAN_1_RX0_FUNC_ENABLE)

/* ... */
void CAN_1_ReceiveMsg0(void)
{
    /* `#START MESSAGE_0_RECEIVED` */

    /* `#END` */

    CAN_1_RX[0u].rxcmd.byte[0u] |= CAN_1_RX_ACK_MSG;
}

#endif /* CAN_1_RX0_FUNC_ENABLE */
```

```

#if (CAN_1_RX1_FUNC_ENABLE)

    /* ... */
    void CAN_1_ReceiveMsg1(void)
    {
        /* `#START MESSAGE_1_RECEIVED` */

        /* `#END` */

        CAN_1_RX[1u].rxcmd.byte[0u] |= CAN_1_RX_ACK_MSG;
    }

#endif /* CAN_1_RX1_FUNC_ENABLE */

```

针对 CAN 接收信息 2 调用以下函数，配置为基本 CAN 并使能中断。

```

void CAN_1_ReceiveMsg(uint8 rxMailbox)
{
    if ((CAN_1_RX[rxMailbox].rxcmd.byte[0u] & CAN_1_RX_ACK_MSG) ==
CAN_1_RX_ACK_MSG)
    {
        /* `#START MESSAGE_BASIC_RECEIVED` */

        /* `#END` */

        CAN_1_RX[rxMailbox].rxcmd.byte[0u] |= CAN_1_RX_ACK_MSG;
    }
}

```

CAN_1_INT.c 文件

```

void CAN_1_MsgRXIsr(void)
{
    ...
    /* RX Full mailboxes handler */
    switch(i)
    {
        case 0 : CAN_1_ReceiveMsg0();
        break;
        case 1 : CAN_1_ReceiveMsg1();
        break;
        default:
        break;
    }
    ...
}

```

如果实施了链接，条件编译应用于设置了 IRQ 标志的邮箱。所有接收函数通过清除信息可用 (MsgAv) 标志来确认信息接收。

如何设置 AMR 和 ACR 在可接受 ID 范围

以下是 ACR/AMR 寄存器示例：



[31:3] - Identifier(ID[31:21] - identifier when IDE = 0, ID[31:3] - identifier when IDE = 1), [2] - IDE, [1] - RTR, [0] - N/A;

验收屏蔽寄存器 (AMR) 定义是否针对验收编码寄存器 (ACR) 检查输入位。

AMR: ‘0’ 针对各个 ACR 检查输入位。当输入位不匹配各自的 ACR 位时，不接受信息。
‘1’ 输入位无需关注。

例如，要设置邮箱以接收在 0x180-187、IDE = 0（已清除）、RTR = 0（已清除）、邮箱 5 范围中的 ID，执行以下附加操作：

1. 采用 ID 的较低范围，例如 0x180，以及相应的 IDE 和 RTR。对于此 ID、IDE 和 RTR 值，AMR 和 ACR 寄存器设置为以下值：

- ACR[31:21] = 0x180 AMR[31:21] = 0x0
- ACR[20:3] = 0x0（无需关注） AMR[20:3] = 0x3FFFF（全一）
- ACR[2] = 0 AMR[2] = 0
- ACR[1] = 0 AMR[1] = 0
- ACR[0] = 0 AMR[0] = 0

2. 定义范围 ID（11 位）的共用部分：

- 0x180 = 0`b001 1000 0000
- 0x187 = 0`b001 1000 0111
- Mask = 0`b001 1000 0XXX
- AMR[31:21] = 0`b000 0000 0111

您必须在 AMR 寄存器中用 1 替代“XXX”及空字符而非共用位。这样，后续值为：

- ACR[31:21] = 0x180 AMR[31:21] = 0x7
- ACR[20:3] = 0x0（无需关注） AMR[20:3] = 0x3FFFF（全一）
- ACR[2] = 0 AMR[2] = 0
- ACR[1] = 0 AMR[1] = 0
- ACR[0] = 0 AMR[0] = 0

3. 使用 CAN_1_RXRegisterInit() 函数写入邮箱编号 5 的 AMR 寄存器：

```
uint8 result = CAN_1_FAIL;
uint32 temp_amr;
uint32 temp_acr;
```

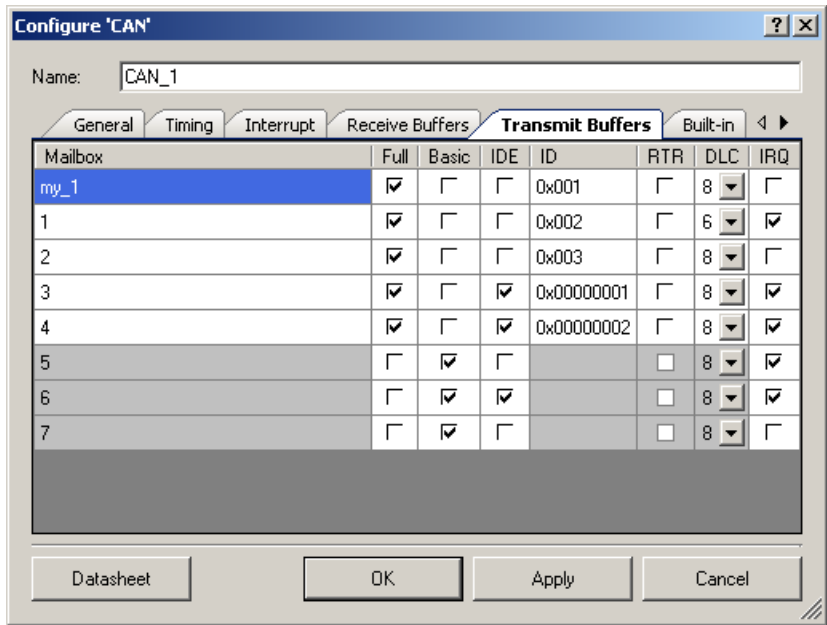
```
/* Upper address value, so address is shifted */
temp_amr = ((uint32)0x7u << 21u) | ((uint32)0x3FFFFu << 3u); /* obtain
necessary value to put in AMR */
temp_acr = ((uint32)0x180u << 21u) | ((uint32)0x3FFFFu << 3u); /* obtain
necessary value to put in ACR */

if (CAN_1_RXRegisterInit((reg32 *)&CAN_1_RX[5].rxamr, temp_amr) == CYRET_SUCCESS)
{
    if (CAN_1_RXRegisterInit((reg32 *)&CAN_1_RX[5].rxacr, temp_acr) ==
CYRET_SUCCESS)
    {
        result = CYRET_SUCCESS;
    }
}

if (result == CAN_1_FAIL)
{
    /* error */
}
```

有关AMR和ACR配置的更多详细信息，请参见[技术参考手册](#)中的“控制器区域网络（CAN）”章节。

Transmit Buffers 选项卡



Transmit Buffers（传输缓冲区）选项卡包含以下设置：

Mailbox（邮箱）

对于为满邮箱，**Mailbox**（邮箱）字段可编辑，您可输入一个唯一的邮箱名称。处理该邮箱的函数也将有一个唯一的名称。接受的符号有：**A–Z**、**a–z**、**0–9** 和 **_**。如果输入错误名称，**Mailbox**（邮箱）字段恢复为默认值。



Full（为满）

选定了 **Full**（为满）时，可修改 **Mailbox**（邮箱）、**IDE**、**ID**、**RTR**、**RTRreply**、**IRQ** 和 **Linking**（链接）字段。默认选择内容包括下列选项：

- **邮箱** = 编号 0 到 7
- **IDE** = 已清除
- **ID** = 0x01
- **RTR** = 已清除
- **DLC** = 8
- **IRQ** = 已清除

Basic（基本）

默认情况下，所有邮箱都选择了 **Basic**（基本）复选框。如果选择了 **Basic**（基本），选项 **ID**、**RTR** 和 **DLC** 不可用。如果选择了 **Basic**，使用代码输入所需的 CAN 信息字段。默认选择内容包括下列选项：

- **IDE** = 已清除
- **ID** = 无（不可用）
- **RTR** = 已清除（不可用）
- **DLC** = 8（不可用）
- **IRQ** = 已清除

IDE

如果 **IDE** 复选框已清除，标识符长度不可超过 11 位（从 0x001 到 0x7FE）。如果选择了 **IDE**，允许使用 29 位标识符（从 0x00000001 到 0x1FFFFFFE）。不可选择 0x000 或 0x7FF（11 位）或 0x1FFFFFFF（29 位）标识符。

ID

信息标识符。

RTR

此信息是返回传输请求信息。

DLC

信息包含的字节数。

IRQ

IRQ 位取决于 **Message Transmitted**（信息已传输）（**Interrupt** 选项卡）。

如果取消选择 **Message Transmitted**（信息已传输）复选框，选择 **IRQ** 时，将显示以下信息：全局“信息传输中断”已禁用。是否使能它？

- **是** — 选择 **IRQ** 和 **Interrupt** 选项卡中的 **Message Transmitted Interrupt**（信息传输中断）复选框。
- **否或取消** — 选择 **IRQ**。**Interrupt** 选项卡中的 **Message Transmitted Interrupt**（信息传输中断）复选框保持已清除状态。

CAN TX 函数

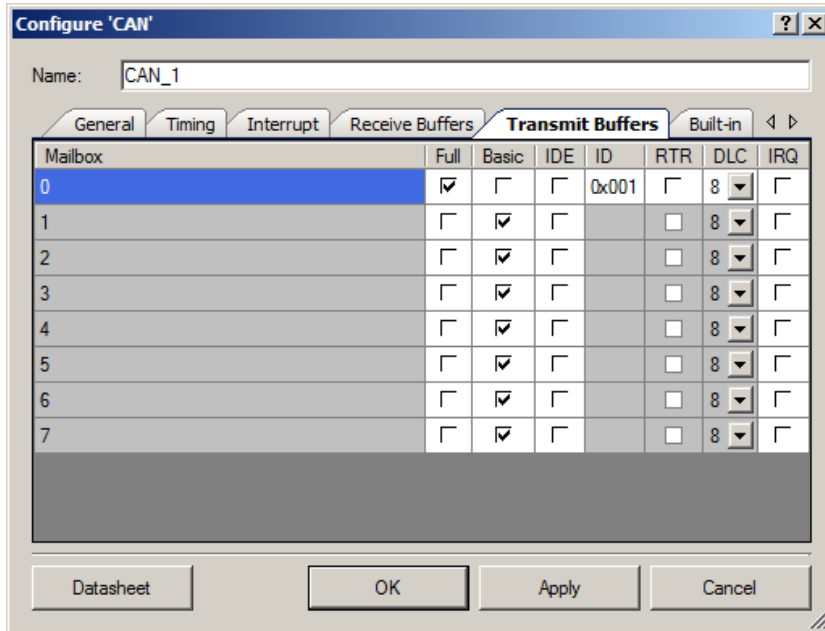
每个为满 TX 邮箱都有一个预定义的 API。*CAN_1_TX_RX_func.c* 项目文件中有函数列表。这些函数是按照条件编译的，这取决于传输邮箱的设置。只有定义为为满的邮箱才编译各自的函数。

宏标识符 *CAN_1_TXx_FUNC_ENABLE* 定义是否编译函数。定义在 *CAN_1.h* 项目文件中显示。

CAN_1_SendMsgX() 函数适用于所有配置为全部的 Tx 邮箱，其中 *X* 表示全部 CAN 邮箱编号或用户定义的名称。

传输缓冲区配置

以下范例说明传输 API 的用法。



CAN_1.h 文件

```
#define CAN_1_TX0_FUNC_ENABLE 1
#define CAN_1_TX1_FUNC_ENABLE 0
```

CAN_1_TX_RX_func.c 文件

```
#if (CAN_1_TX0_FUNC_ENABLE)

/* ... */
uint8 CAN_1_SendMsg0(void)
{
    uint8 result = CYRET_SUCCESS;

    if ((CAN_1_TX[0u].txcmd.byte[0u] & CAN_1_TX_REQUEST_PENDING) ==
        CAN_1_TX_REQUEST_PENDING)
    {
        result = CAN_1_FAIL;
    }
    else
    {
        /* `#START MESSAGE_0_TRANSMITTED` */

        /* `#END` */

        CY_SET_REG32((reg32 *) &CAN_1_TX[0u].txcmd, CAN_1_SEND_MESSAGE);
    }

    return(result);
}
```

```

    }

#endif /* CAN_1_TX0_FUNC_ENABLE */

```

为所有基本传输邮箱提供的共用函数为：

```
uint8 CAN_1_SendMsg(const CAN_1_TX_MSG *message)
```

为用于收集 CAN 传输信息所需数据的应用程序定义的一般结构：

- ID — 如果 ID 槽包括以下内容时的限制：
 - 对于标准信息（IDE = 0），标识符长度限制为 11 位（0x001 到 0x7FE）。
 - 对于扩展信息（IDE = 1），标识符长度限制为 29 位（0x00000001 到 0x1FFFFFFE）。
- RTR（0 — 标准信息；1 — 0xFF：在信息中设置的 RTR 位）
- IDE（0 — 标准信息；1 — 0xFF：扩展信息）
- DLC（定义数据字节 0 到 8，9 到 0xFF 的数量等于 8 个数据字节）
- IRQ（0 — IRQ 使能，1 — 0xFF：IRQ 禁用）
- DATA_BYTES（指针指向代表传输数据的 8 字节的结构）

调用 CAN_1_SendMsg()函数时，函数遍历被指定为基本 CAN 邮箱的传输信息邮箱，并寻找第一个可用邮箱：

- 当找到一个闲置的基本 CAN 邮箱时，通过 CAN_1_TX_MSG 结构的数据被复制到相应的 CAN 传输邮箱中。当信息进入传输队列时，将返回“成功”指示到应用程序。
- 未找到闲置的基本邮箱时，函数将最多再进行三次尝试。如果所有尝试都失败了，则向应用程序返回“失败”指示。

CAN_1_TX_MSG 结构包含传输信息所需的所有信息：

```

/* Stuct for BASIC CAN mailbox to send messages */
typedef struct
{
    uint32 id;
    uint8 rtr;
    uint8 ide;
    uint8 dlc;
    uint8 irq;
    CAN_1_DATA_BYTES_MSG *msg;
} CAN_1_TX_MSG;

```

在一条信息中，CAN_1_DATA_BYTES 结构包含八个数据字节。

```
/* Stuct for DATA of BASIC CAN mailbox */
```



```
typedef struct
{
    uint8 byte[8u];
} CAN_1_DATA_BYTES_MSG;
```

时钟选择

CAN 组件连接到 BUS_CLK 时钟信号。需要最小 8 MHz 以支持所有高达 1 Mbps 的标准 CAN 波特率。

应用编程接口

通过应用编程接口（API）子程序，您可以使用软件对组件进行配置。下表列出并说明了每个函数的接口。以下各节将更详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“CAN_1”分配给指定设计中组件的第一个实例。您可以将该实例重新命名为符合标识符语法规则的唯一一个任意值。实例名称会成为每个全局函数名称、变量和常量符号的前缀。出于可读性考虑，下表中使用的实例名称为“CAN”。

函数	说明
CAN_Start()	设置initVar变量，调用CAN_Init()函数，然后调用CAN_Enable()函数。
CAN_Stop()	禁用CAN。
CAN_GlobalIntEnable()	从CAN组件中使能全局中断。
CAN_GlobalIntDisable()	从CAN组件中禁用全局中断。
CAN_SetPreScaler()	设置从BUS_CLK生成时间段的预分频器。
CAN_SetArbiter()	设置传输缓冲区的仲裁类型
CAN_SetTsegSample()	配置：时间段1、时间段2、同步跳转宽度和采样模式。
CAN_SetRestartType()	设置复位类型。
CAN_SetEdgeMode()	设置沿模式。
CAN_RXRegisterInit()	仅写入接收CAN寄存器。
CAN_SetOpMode()	设置操作模式。
CAN_GetTXErrorflag()	返回标志，说明传输错误数是否超过0x60。
CAN_GetRXErrorflag()	返回标志，说明接收错误数是否超过0x60。
CAN_GetTXErrorCount()	返回传输错误数。
CAN_GetRXErrorCount()	返回接收错误数。
CAN_GetErrorState()	返回CAN组件的错误状态。

函数	说明
CAN_SetIrqMask()	设置以使能或禁用特定的中断源。
CAN_ArbLostIsr()	清除仲裁丢失中断标志。
CAN_OvrLdErrorIsr()	清除过载错误中断标志。
CAN_BitErrorIsr()	清除位错误中断标志。
CAN_BitStuffErrorIsr()	清除位填充错误中断标志。
CAN_AckErrorIsr()	清除确认错误中断标志。
CAN_MsgErrorIsr()	清除格式错误中断标志。
CAN_CrcErrorIsr()	清除CRC错误中断标志。
CAN_BusOffIsr()	清除总线关闭中断标志。将CAN组件设为停止模式。
CAN_MsgLostIsr()	清除信息丢失中断标志。
CAN_MsgTXIsr()	清除传输信息中断标志。
CAN_MsgRXIsr()	清除接收信息中断标志，并针对基于基本和完全中断的邮箱调用正确的处理程序。
CAN_RxBufConfig()	为特定邮箱配置所有接收寄存器。
CAN_TxBufConfig()	为特定邮箱配置所有传输寄存器。
CAN_SendMsg()	从一个基本邮箱中发送信息。
CAN_SendMsg0-7()	查看邮箱0-7是否有未传输的信息等待仲裁。
CAN_TxCancel()	取消传输已排队等候传输的信息。
CAN_ReceiveMsg0-15()	确认收到新信息。
CAN_ReceiveMsg()	清除接收特定信息中断标志。
CAN_Sleep()	使CAN组件做好睡眠准备。
CAN_Wakeup()	使CAN组件做好唤醒准备。
CAN_Init()	根据Configure对话框设置，初始化或恢复CAN。
CAN_Enable()	使能CAN。
CAN_SaveConfig()	保存当前配置。
CAN_RestoreConfig()	恢复配置。

对于返回执行指示的函数：0 表示“成功”，1 表示“失败”，2 表示“超出范围”。

全局变量

变量	说明
CAN_initVar	说明CAN是否已初始化。变量将初始化为0，并在第一次调用CAN_Start()时设置为1。这样，在第一次调用CAN_Start()子程序后，组件不用重新初始化即可重启。 如需重新初始化组件，可在CAN_Start()或CAN_Enable()函数前调用CAN_Init()函数。

uint8 CAN_Start(void)

说明： 设置initVar变量，调用CAN_Init()函数，然后调用CAN_Enable()函数。此函数将CAN组件设为运行模式，并在轮询邮箱可用时启动计数器。

参数： 无

返回值： uint8：指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响： 如果已设置initVar变量，则该函数仅调用CAN_Enable()函数。

uint8 CAN_Stop(void)

说明： 此函数将CAN组件设为停止模式，并在轮询邮箱可用时停止计数器。

参数： 无

返回值： uint8：指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响： 无

uint8 CAN_GlobalIntEnable(void)

说明: 此函数从CAN组件中使能全局中断。

参数: 无

返回值: uint8: 指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响: 无

uint8 CAN_GlobalIntDisable(void)

说明: 此函数从CAN组件中禁用全局中断。

参数: 无

返回值: uint8: 指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响: 无

uint8 CAN_SetPreScaler(uint16 bitrate)

说明: 此函数设置从BUS_CLK生成时间段的预分频器。0x0和0x7FFF之间的值有效。

参数: uint16 bitrate: 预分频器值
 时间段长度 = (比特率 + 1) 时钟周期

返回值: uint8: 指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。
CAN_OUT_OF_RANGE	函数参数超过了范围。

其他影响: 无

uint8 CAN_SetArbiter(uint8 arbiter)

说明: 此函数设置传输缓冲区的仲裁类型。仲裁器类型是轮循和固定优先级。

参数: uint8 arbiter: 仲裁器类型

数值	说明
CAN_ROUND_ROBIN	轮循仲裁。
CAN_FIXED_PRIORITY	固定优先级仲裁。

返回值: uint8: 指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响: 无

uint8 CAN_SetTsegSample(uint8 cfgTseg1, uint8 cfgTseg2, uint8 sjw, uint8 sm)

说明: 该函数配置：时间段1、时间段2、同步跳转宽度和采样模式。

参数: uint8 cfgTseg1: 时间段1的长度，0x2和0xF之间的值有效

uint8 cfgTseg2: 时间段2的长度，0x1和0x7之间的值有效

uint8 sjw: 同步跳转宽度，0x0和0x3之间的值有效。

uint8 sm: 采样模式。

数值	说明
CAN_ONE_SAMPLE_POINT	使用一个采样点
CAN_THREE_SAMPLE_POINTS	使用三个采样点。

返回值: uint8: 指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。
CAN_OUT_OF_RANGE	函数参数超过了范围。

其他影响: 无

uint8 CAN_SetRestartType(uint8 reset)

说明: 此函数设置复位类型。复位类型为自动和手动。手动复位为推荐设置。

参数: uint8 reset: 复位类型

数值	说明
CAN_MANUAL_RESTART	总线关闭后，必须‘手动’重启CAN。这是推荐设置。
CAN_AUTO_RESTART	总线关闭后，在128组11个“空闲”位之后，将自动重新启动CAN。

返回值: uint8: 指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响: 无

uint8 CAN_SetEdgeMode(uint8 edge)

说明: 此函数设置沿模式。模式为‘R’到‘D’（空闲到占有），同时使用两种沿。

参数: uint8 edge: 边沿模式

数值	说明
CAN_EDGE_R_TO_D	从‘R’到‘D’的边沿用于进行同步。
CAN_BOTH_EDGES	同时使用两种边沿。

返回值: uint8: 指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响: 无

uint8 CAN_RXRegisterInit(uint32 *regAddr, uint32 config)

说明: 此函数仅写入CAN接收寄存器。

参数: uint32 * regAddr: 指向CAN接收寄存器的指针
uint32 configuration: 将要写入到寄存器中的值

返回值: uint8: 指示寄存器是否已被写入及验证

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。
CAN_OUT_OF_RANGE	函数参数超过了范围。

其他影响: 无

uint8 CAN_SetOpMode(uint8 opMode)

说明: 此函数设置操作模式。操作模式为“活动”或“仅侦听”。

参数: uint8 opMode: 操作模式值

数值	说明
CAN_ACTIVE_MODE	活动模式。
CAN_LISTEN_ONLY	CAN仅侦听: 该输出保持为‘R’电平。

返回值: uint8: 指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响: 无

uint8 CAN_GetTXErrorflag(void)

说明: 此函数返回标志, 说明传输错误数是否超过0x60。

参数: 无

返回值: uint8: 说明传输错误数是否超过0x60

其他影响: 无

uint8 CAN_GetRXErrorflag(void)

说明: 此函数返回标志，说明接收错误数是否超过0x60。

参数: 无

返回值: uint8: 说明接收错误数是否超过0x60

其他影响: 无

uint8 CAN_GetTXErrorCount(void)

说明: 此函数返回传输错误数。

参数: 无

返回值: uint8: 传输错误数

其他影响: 无

uint8 CAN_GetRXErrorCount(void)

说明: 此函数返回接收错误数。

参数: 无

返回值: (uint8)接收错误数

其他影响: 无

uint8 CAN_GetErrorState(void)

说明: 此函数返回CAN组件的错误状态。

参数: 无

返回值: uint8: 错误状态

其他影响: 无

uint8 CAN_SetIrqMask(uint16 mask)

说明： 此函数使能或禁用特定中断源。中断屏蔽直接写入到CAN中断使能寄存器。通过将特殊中断源的相应标志设置为1可实现使能操作。

参数： uint8 request: 中断使能或禁用请求。每个中断源一位。

数值	说明
CAN_GLOBAL_INT_ENABLE	全局中断使能标志。
CAN_ARBITRATION_LOST_ENABLE	仲裁丢失中断使能。
CAN_OVERLOAD_ERROR_ENABLE	过载中断使能。
CAN_BIT_ERROR_ENABLE	位错误中断使能。
CAN_STUFF_ERROR_ENABLE	填充错误中断使能。
CAN_ACK_ERROR_ENABLE	确认错误中断使能。
CAN_FORM_ERROR_ENABLE	格式错误中断使能。
CAN_CRC_ERROR_ENABLE	CRC错误中断使能。
CAN_BUS_OFF_ENABLE	总线状态中断使能。
CAN_RX_MSG_LOST_ENABLE	Rx Msg丢失的中断使能。
CAN_TX_MESSAGE_ENABLE	Tx Msg发送的中断使能。
CAN_RX_MESSAGE_ENABLE	Msg接收到的中断使能。

返回值： uint8: 指示寄存器是否已被写入及验证。

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响： 无

void CAN_ArbLostIsr(void)

说明： 此函数是仲裁丢失中断的入口点。它清除仲裁丢失中断标志。仅在仲裁丢失中断参数使能时才生成此函数。

参数： 无

返回值： 无

其他影响： 无

void CAN_OvrLdErrorIsr(void)

说明: 此函数是过载错误中断的入口点。它清除过载错误中断标志。仅在过载错误中断参数使能时才生成此函数。

参数: 无

返回值: 无

其他影响: 无

void CAN_BitErrorIsr(void)

说明: 此函数是位错误中断的入口点。它清除位错误中断标志。仅在位错误中断参数使能时才生成此函数。

参数: 无

返回值: 无

其他影响: 无

void CAN_BitStuffErrorIsr(void)

说明: 此函数是位填充错误中断的入口点。它清除位填充错误中断标志。仅在位填充错误中断参数使能时才生成此函数。

参数: 无

返回值: 无

其他影响: 无

void CAN_AckErrorIsr(void)

说明: 此函数是确认错误中断的入口点。它清除确认错误中断标志。仅在确认错误中断参数使能时才生成此函数。

参数: 无

返回值: 无

其他影响: 无

void CAN_MsgErrorIsr(void)

说明: 此函数是格式错误中断的入口点。它清除格式错误中断标志。仅在格式错误中断参数使能时才生成此函数。

参数: 无

返回值: 无

其他影响: 无

void CAN_CrcErrorIsr(void)

说明: 此函数是CRC错误中断的入口点。它清除CRC错误中断标志。仅在CRC错误中断参数使能时才生成此函数。

参数: 无

返回值: 无

其他影响: 无

void CAN_BusOffIsr(void)

说明: 此函数是总线关闭中断的入口点。它将CAN组件设为停止模式。仅在总线关闭中断参数使能时才生成此函数。建议使能此中断。

参数: 无

返回值: 无

其他影响: 停止CAN组件操作

void CAN_MsgLostIsr(void)

说明: 此函数是信息丢失中断的入口点。它清除信息丢失中断标志。仅在信息丢失中断参数使能时才生成此函数。

参数: 无

返回值: 无

其他影响: 无

void CAN_MsgTXIsr(void)

说明: 此函数是传输信息中断的入口点。它清除传输信息中断标志。仅在传输信息中断参数使能时才生成此函数。

参数: 无

返回值: 无

其他影响: 无

void CAN_MsgRXIsr(void)

说明: 此函数是接收信息中断的入口点。它清除接收信息中断标志，并针对基本和完全中断邮箱调用正确的处理程序。仅在接收信息中断参数使能时才生成此函数。建议使能此中断。

参数: 无

返回值: 无

其他影响: 无

uint8 CAN_RxBufConfig(const CAN_RX_CFG *rxConfig)

说明: 此函数为特定邮箱配置所有接收寄存器。邮箱编号包含CAN_RX_CFG结构。

参数: const CAN_RX_CFG * rxConfig: 指向某结构的指针，该结构包含为特定邮箱配置所有接收寄存器所需的所有值

返回值: uint8: 说明是否已接收或拒绝了特定配置

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响: 无

uint8 CAN_TxBufConfig(const CAN_TX_CFG *txConfig)

说明: 此函数为特定邮箱配置所有传输寄存器。邮箱编号包含CAN_TX_CFG结构。

参数: const CAN_TX_CFG * txConfig: 指向某结构的指针，该结构包含为特定邮箱配置所有传输寄存器所需的所有值

返回值: uint8: 说明是否已接收或拒绝了特定配置

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响: 无

uint8 CAN_SendMsg(const CANTXMsg *message)

说明: 该函数从一个基本邮箱中发送信息。此函数依次通过所有指定为基本CAN邮箱的传输信息缓冲器。它查找第一个闲置可用邮箱，并从该邮箱发送信息。重试次数最多为3。

参数: const CAN_TX_MSG * message: 指针，指向包含发送信息所需的数据的结构

返回值: uint8: 说明是否已发送信息

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响: 无

uint8 CAN_SendMsg0-7(void)

说明: 这些函数是传输信息0-7的入口点。查看邮箱0-7是否有等待仲裁的未传输的信息。如果有，它启动信息传输。仅针对被指定为“为满”的传输邮箱生成这些函数。

参数: 无

返回值: uint8: 说明是否已发送信息

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响: 无

void CAN_TxCancel(uint8 bufferId)

说明: 此函数取消已排队等候传输的信息的传输。有效值范围为0到15。

参数: uint8 bufferId: Tx邮箱的数量。

返回值: 无

其他影响: 无

void CAN_ReceiveMsg0-15(void)

说明: 此函数是接收信息0-15中断的入口点。它们清除接收信息0-15中断标志。仅针对指定为“完全”中断的接收邮箱生成这些函数。

参数: 无

返回值: 无

其他影响: 无

void CAN_ReceiveMsg(uint8 rxMailbox)

说明: 此函数是基本邮箱的接收信息中断的入口点。它清除接收特殊信息中断标志。仅在一个接收邮箱被指定为“基本”时，才生成此函数。

参数: uint8 rxMailbox: 触发接收信息中断的邮箱编号

返回值: 无

其他影响: 无

void CAN_Sleep(void)

说明: 这是组件准备进入睡眠模式时的首选子程序。CAN_Sleep()子程序保存当前组件的状态。然后它调用CAN_Stop()函数，并调用CAN_SaveConfig()函数保存硬件配置。

在调用CyPmSleep()或CyPmHibernate()函数之前调用CAN_Sleep()函数。有关电源管理函数的详细信息，请参考PSoC Creator *系统参考指南*。

参数: 无

返回值: 无

其他影响: 无

void CAN_Wakeup(void)

说明： 该函数是将组件恢复到调用CAN_Sleep()时状态的首选子程序。CAN_Wakeup()函数调用CAN_RestoreConfig()函数以恢复用户配置。如果组件在调用CAN_Sleep()函数前已使能，则CAN_Wakeup()函数还将重新使能组件。

参数： 无

返回值： 无

其他影响： 调用CAN_Wakeup()函数前未调用CAN_Sleep()或CAN_SaveConfig()函数可能会产生不可预计行为。

uint8 CAN_Init(void)

说明： 根据自定义程序“Configure”对话框设置初始化或恢复组件。无需调用CAN_Init()，因为CAN_Start()子程序会调用该函数并是开始组件操作的优选方法。

参数： 无

返回值： uint8: 说明是否已接受或拒绝配置

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响： 所有寄存器将复位为初始值。这将重新初始化组件，不同的是，它不清除邮箱中的数据。打开CAN内核的电源。

uint8 CAN_Enable(void)

说明： 激活硬件，开始组件操作。无需调用CAN_Enable()，因为CAN_Start()子程序会调用该函数，这是开始组件操作的优选方法。

参数： 无

返回值： uint8: 说明是否已接受或拒绝配置

数值	说明
CYRET_SUCCESS	函数已成功完成。
CAN_FAIL	函数执行失败。

其他影响： 无

void CAN_SaveConfig(void)

- 说明：** 此函数会保存组件配置和非保留寄存器。此函数还将保存当前“Configure”（配置）对话框中定义的或通过相应API修改的组件参数值。该函数由CAN_Sleep()函数调用。
- 参数：** 无
- 返回值：** 无
- 其他影响：** 无

void CAN_RestoreConfig(void)

- 说明：** 该函数会恢复组件配置和非保留寄存器。该函数还会将组件参数值恢复为调用CAN_Sleep()函数之前的值。
- 参数：** 无
- 返回值：** 无
- 其他影响：** 调用此函数前未调用CAN_Sleep()或CAN_SaveConfig()函数可能会产生不可预计行为。以下寄存器将恢复为它们的默认值：CAN_INT_SR、CAN_INT_EN、CAN_CMD和CAN_CFG。

MISRA 合规性

本节介绍了 MISRA-C:2004 合规性和本器件的偏差情况。定义了下面两种类型的偏差：

- 项目偏差 — 适用于所有 PSoC Creator 组件的偏差
- 特定偏差 — 仅适用于该组件的偏差

本节提供了有关组件特定偏差的信息。系统参考指南的“MISRA 合规性”章节中介绍了项目偏差以及有关 MISRA 合规性验证环境的信息。

此 CAN 组件具有如下特定偏差：

MISRA-C:2004 规则	规则类别 (必须(R) / 建议(A))	规则说明	偏差说明
11.4	A	不能对不同对象指针类型进行转换。	将指向8位寄存器的指针转换为指向16位寄存器的指针，以访问硬件中的连续字节。
13.7	R	不允许结果不变的Boolean运算。	在某些组件配置中，控制表达式的值始终不变。
15.5	R	每个switch语句至少有一个case子句。	未选择RX全邮箱中断处理器时，switch语句没有case子句。

MISRA-C:2004 规则	规则类别 (必须(R)/ 建议(A))	规则说明	偏差说明
17.4	R	数组索引是唯一允许的指针运算形式。	适用于一个指针类型对象的数组下标。CAN组件将指针用于映射到硬件寄存器的结构。
19.7	A	函数应比类函数宏优先使用。	使用于应用二进制掩码的宏语句规定该情况，可以使用一个函数代替该语句，但这样会降低其性能。

该组件具有以下的嵌入式组件：时钟和中断。MISRA 合规性与特定偏差的相关信息，请参见相应的组件数据手册。

示例固件源代码

PSoC Creator 在“Find Example Project”（查找示例项目）对话框中提供了多种包括原理图和代码示例的示例工项目。要查看特定组件示例，请打开“Component Catalog”中的对话框或原理图中的组件实例。要查看通用示例，请打开“Start Page”或“File”菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options** 选项来限定可选的项目列表。

更多有关信息，请参考《PSoC Creator 帮助》部分中主题为“查找示例项目”中的内容。

中断服务子程序

有多个 CAN 组件中断源：

- 仲裁丢失检测 — 发送信息时仲裁丢失
- 过载错误 — 接收到一个过载帧
- 位错误 — 检测到位错误
- 位填充错误 — 检测到位填充错误
- 确认错误 — 检测到 CAN 信息确认错误
- 格式错误 — 检测到 CAN 信息格式错误
- CRC 错误 — 检测到 CAN CRC 错误
- 总线关闭 — CAN 达到总线关闭状态
- 信息丢失 — 新信息到达，但无处放置
- 传输信息 — 队列中的信息被发送

■ 接收信息 — 接收到信息

所有这些中断源都有入口点（函数），所以可将代码置于其中。这些函数是按照条件编译的，这取决于定制器。

接收信息中断有一个特殊处理程序，该程序针对为满和基本邮箱调用相应的函数。

中断输出使用案例

以下是 CAN 组件中硬件中断输出线路的示范用例。

中断事件上的逻辑的硬件控制

硬件中断线可用于执行简单的任务，例如评估 CAN 总线负载。通过在 CAN 组件定制器中使能“信息已传输”和“信息已接收”中断，并将中断线连接到计数器，可评估特定时间间隔中总线上的信息数。同时，如果信息速率超过特定值，可直接在硬件中采取措施。

中断输出与 DMA 的互动

CAN 组件内部不支持 DMA 操作，但您可将 DMA 组件连接到外部中断线（如果已使能）。您负责进行 DMA 配置和操作。同时，您应记住必须管理代码中的一些日常工作（例如，确认信息和清除中断标志），以正确处理 CAN 中断。

利用硬件 DMA 触发器，您可在发生信息接收中断时处理寄存器和数据传输，而无需 CPU 中的任何固件进行操作。处理 RTR 信息时此触发器也很有用。信息传输中断可用于触发 DMA 传输，以用新数据重新加载信息缓冲区，而无需 CPU 干预。

自定义外部中断服务子程序

自定义外部 ISR 可用于辅助或替代内部 ISR。当外部 ISR 用于辅助内部 ISR 时，可设置中断优先级以确定哪个 ISR 先执行（内部或外部），从而在内部 ISR 中编码的操作之前或之后执行操作。当外部 ISR 用于替代内部 ISR 时，您承担正确处理 CAN 寄存器和事件的所有责任。

中断输出与中断子系统的互动

CAN 组件中断输出设置允许您执行以下操作：

- 使能或禁用外部中断线（定制器选项）
- 禁用或旁路内部 ISR（定制器选项）
- 完全自定义内部 ISR（定制器选项）
- 使能或禁用当使能相关事件中断时，内部 ISR 中的特定中断处理函数调用（定制器选项）。可在 CAN 组件定制器中使能或禁用个别中断（信息已传输、信息已接收、接收缓冲区满、



总线关闭状态等)。一旦使能,将在内部 `CAN_ISR` 中执行相关函数调用。这样,您可禁用(移除)此类函数调用。

只有在定制器中使能了外部中断线时,外部中断线才可见。

如果连接了外部中断组件,外部中断组件不作为 `CAN_Start()` API 的一部分启动,且必须在该子程序外启动。

如果连接了外部中断组件,且内部 `ISR` 未禁用或跳过,两个中断组件连接到同一线路。在这种情况下,您有两个单独的中断组件处理同一中断事件。这是一个特定的情况,通常也是不受欢迎的情况。

如果内部 `ISR` 被禁用或跳过(使用定制器选项)时,内部中断组件将在构建过程中被移除。

若选择在内部中断子程序中禁用个别中断函数调用(针对已使能的中断事件,通过使用定制器选项),`CAN` 模块中断将触发(当相关事件发生时),但内部 `CAN_ISR` 子程序中不执行任何内部函数调用。一个示例使用情况是,当您要通过不同的路径而不是标准用户函数调用(例如,通过 `DMA`)处理特定事件(例如,信息接收)时。

如果选择完全自定义内部 `ISR`(使用自定义选项),`CAN_ISR` 函数将不包含任何函数调用。

功能说明

有关完整的说明,请参见 [技术参考手册](#) 中的“控制器区域网络(CAN)”章节。

框图和配置

有关框图和配置信息,请参见 [技术参考手册](#) 中的“控制器区域网络(CAN)”章节。

参考

1. ISO-11898: 公路车辆—控制器区域网络(CAN):

- 第一部分: 数据链接层和物理信号
- 第二部分: 高速媒体访问单元
- 第三部分: 低速、容错、媒体相关接口
- 第四部分: 时间触发的通信
- 第五部分: 高速媒体访问单元,具有低功耗模式

2. CAN 规范版本 2 BOSCH

3. Inicore CANmodule-III-AHB 数据手册

资源

CAN 组件使用芯片中的专用 CAN 硬件模块。

API 存储器的使用情况

根据编译器、器件、所使用的 API 数量以及组件的配置不同，组件对存储资源的占用也不一样。下表提供了在某种器件配置中所有 API 占用存储器的大小。

数据是在将编译器设置为 Release 模式并将优化等级设置为 Size 的情况下测得的。对于特定的设计，分析完编译器生成的映射文件后可以确定组件占用存储器的大小。

配置	PSoC 3 (Keil_PK51)		PSoC 5LP (GCC)	
	闪存大小 (字节)	SRAM大小 (字节)	闪存大小 (字节)	SRAM大小 (字节)
默认值	3331	18	2060	21
其他接收缓冲区使用情况	+ 8	–	+ 16	–
其他中断处理器使能	+ 7	–	+ 12	–

直流和交流电气特性

除非另有说明，否则这些规范的适用条件是： $-40^{\circ}\text{C} \leq T_A \leq 85^{\circ}\text{C}$ 且 $T_J \leq 100^{\circ}\text{C}$ 。除非另有说明，否则这些规范的适用范围为 1.71 V 到 5.5 V。

直流规范

参数	说明	条件	最小值	典型值	最大值	单位
I _{DD}	模块的电流消耗		–	–	200	μA

交流规范

参数	说明	条件	最小值	典型值	最大值	单位
	比特率	时钟的最低频率为8 MHz	–	–	1	Mbit



组件勘误表

本节列出了组件的已知问题。

赛普拉斯ID	组件版本	问题	解决方案
191257	v2.30	在没有修正PSoC Creator 3.0 SP1中的版本编号时进行更改这组件。更多有关信息，请参见基础知识文章KBA94159（网页地址： www.cypress.com/go/kba94159 ）。	解决方案并非必要的。不会对设计产生影响。

组件更改

本节列出了各版本的主要组件更改内容。

当前版本	更改说明	更改原因/影响
2.30.a	更新了数据手册。	将组件勘误章节添加到被更改组件的文档，但设计不受任何影响。
		移除了已停产的PSoC 5器件的参考内容。
2.30	添加了MISRA合规性章节。	此组件具有特定的描述偏差。
	更新了API函数参数说明。	未完成API函数参数说明。
2.20	CAN_Start()函数内部更新。	如果之前已经启动并停止了该组件，则CAN_Start()函数不能使能该组件。
2.10	添加了PSoC 5LP器件支持。	
	对所有CAN API添加了CYREENTRANT关键词（当它们包含在.cyre文件中时）。	并非所有API都是真正可重入的函数。组件API源文件中的注释指出了适用的函数。 对于采用了安全方式并且是不可重入的函数，则需要该项变更，这样可以消除编译器警告：通过标志或关键节防止同时调用。
	将组件名称修改为“CAN_1”，并更新了实例中的代码段，用于说明接收和传输信息API的使用情况。	为了满足一致性。
	更新了直流和交流电气特性章节。	
2.0.a	已替换时序框图并向数据手册添加了描述性文本。	
	对数据手册进行了少量编辑和更新。	
2.0	已向组件符号添加了中断输出。	已更新PSoC 3 CAN组件的营销要求文档MRD。

当前版本	更改说明	更改原因/影响
	已在组件符号中添加了 ConnectExtInterruptLine、IntISRDisable、FullCustomIntISR、AdvancedInterruptTab 参数和 ISR 助手参数。	已更新 PSoC 3 CAN 组件的营销要求文档 MRD。
	已在 CAN_Init() 函数开头添加了停止语句。	要确保 CAN 在初始化时停止。
	已更新了中断处理程序函数。在所有案例中，在用户代码之前清除中断标志。	要以同样的方式清除中断标志。
	已移除过期定义。	
1.50.a	向数据手册中添加了特性数据。	
	数据手册文本编辑。	
1.50	已添加睡眠/唤醒 API。	这些 API 支持低功耗模式。
	已添加 CAN_Init() 和 CAN_Enable() API。	为了符合公司标准，并提供 API 以便无需启动组件即可初始化/恢复组件。

赛普拉斯半导体公司，2013-2016 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用者应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担任何全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，WICED，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财

