**AN2401**
**应用笔记**

# UM32x42x CANFD 配置指南

版本：V1.0




广芯微电子（广州）股份有限公司

http://www.unicmicro.com/

# 条款协议



# 目录

# 1 摘要

本篇应用笔记主要介绍 UM32x42x CANFD 的功能以及配置方法。

本篇应用笔记主要包括：

- CANFD 概述
- CANFD 配置流程
- 使用参考例程

**注：**具体功能及寄存器的操作等相关事项请以用户手册为准。

# 2 CANFD 概述

## 2.1 什么是 CANFD

CANFD，全称为 CAN with Flexible Data rate。继承了 CAN 的主要特性，弥补了 CAN 的数据长度和带宽的限制。CANFD 相较于 CAN 只升级了协议，物理层未改变，可简单认为 CANFD 就是 CAN 的升级版。它们主要区别在于：

1. 传输速率不同。

   CAN 最大传输速率 1Mbps，CANFD 速率可变，仲裁波特率最高可到 1Mbps，数据波特率最高 8Mbps。

2. 数据长度不同。

   CAN 一帧数据最长 8 字节，CANFD 一帧数据最长 64 字节。

3. 帧格式不同。

   CANFD 没有远程帧，报文结构新增了 FDF、BRS、ESI 位。

## 2.2 CANFD 帧结构

CANFD 数据帧结构与 CAN 一样，一共有 7 个部分：帧起始 SOF，仲裁段，控制段，数据段，CRC 段，ACK 段，帧结束。



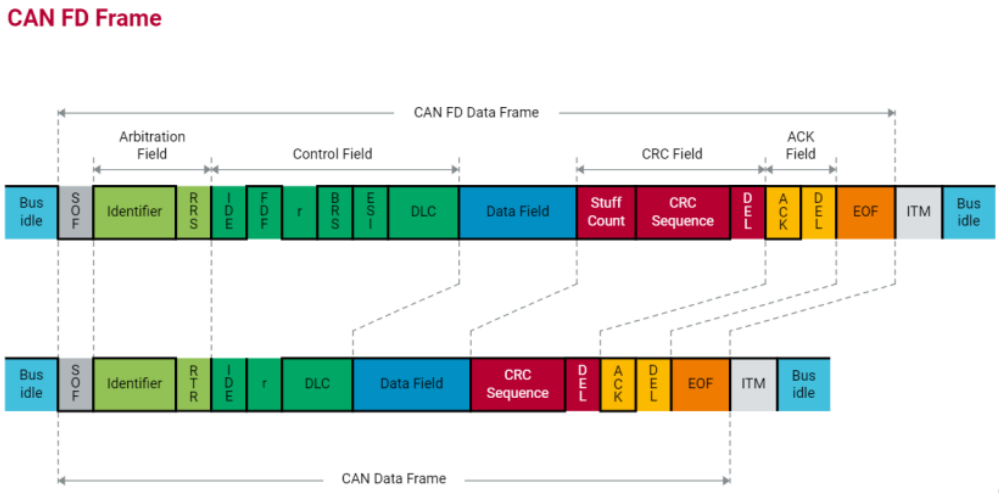



图 2-1：CANFD 与 CAN 帧格式

## 2.2.1 帧起始

CANFD 与 CAN 使用相同的 SOF 标志位来标志报文的起始。帧起始由 1 个显性位构成，标志着报文的开始，并在总线上起着同步的作用。

## 2.2.2 仲裁段

与传统 CAN 相比，CANFD 取消了远程帧，用 RRS 位替换了 RTR 位，为常显性。IDE 位仍为标准帧和扩展帧标志位，若标准帧与扩展帧具有相同的前 11 位 ID，那么标准帧将会由于 IDE 位为 0，优先获得总线。

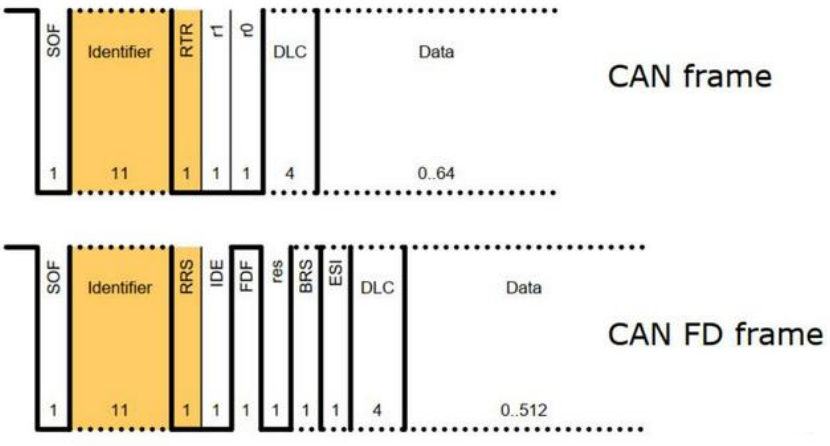



图 2-2：CAN 和 CANFD 仲裁段的对比

### 2.2.3 控制段

控制段中 CANFD 与 CAN 有着相同的 IDE，RES，DLC 位。同时增加了三个控制位：FDF、BRS 及 ESI。

1. FDF（Flexible Data Rate Format）：原 CAN 数据帧中的保留位。FDF 常为隐性，表示 CANFD 报文。
2. BRS（Bit Rate Switch）：位速率转换开关，当 BRS 为显性位时数据段的位速率与仲裁段的位速率一致，当 BRS 为隐性位时数据段的位速率高于仲裁段的位速率。
3. ESI（Error State Indicator）：错误状态指示，主动错误时发送显性位，被动错误时发送隐性位。
4. DLC：同样是 4-bit 表示数据段的长度。

| Frames | Data Length Code | | | | Number of Data Bytes |
|---|---|---|---|---|---|
| | DLC3 | DLC2 | DLC1 | DLC0 | |
| Classical Frame and FD Frame | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 1 |
| | 0 | 0 | 1 | 0 | 2 |
| | 0 | 0 | 1 | 1 | 3 |
| | 0 | 1 | 0 | 0 | 4 |
| | 0 | 1 | 0 | 1 | 5 |
| | 0 | 1 | 1 | 0 | 6 |
| | 0 | 1 | 1 | 1 | 7 |
| | 1 | 0 | 0 | 0 | 8 |
| FD Frame | 1 | 0 | 0 | 1 | 12 |
| | 1 | 0 | 1 | 0 | 16 |
| | 1 | 0 | 1 | 1 | 20 |
| | 1 | 1 | 0 | 0 | 24 |
| | 1 | 1 | 0 | 1 | 32 |
| | 1 | 1 | 1 | 0 | 48 |
| | 1 | 1 | 1 | 1 | 64 |

图 2-3：DLC 取值和数据长度的关系

### 2.2.4 数据段

CANFD 不仅能支持传统的 0～8 字节报文，同时最大还能支持 12、16、20、24、32、48、64 字节。

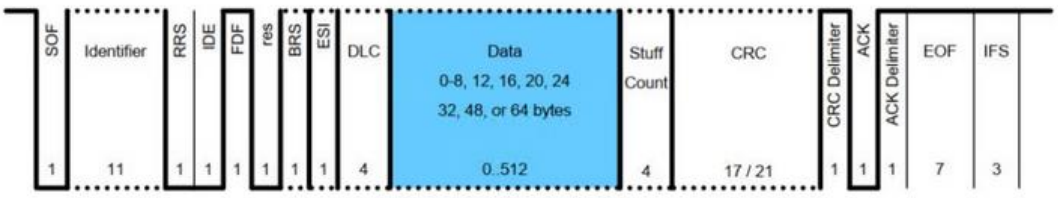



图 2-4：Data 数据长度

### 2.2.5 CRC 段

为了避免位填充对 CRC 的影响，CANFD 在 CRC 场中增加了 Stuff Count，记录填充位数对应

8 的模，并用格雷码表示，还增加了奇偶校验位。FSB（fixed stuff-bit）固定为前一位的补码。Stuff Count 由以下两个元素组成：

- 格雷码计算：CRC 区域之前的填充位数除以 8，得到的余数（Stuff bit count modulo 8）进行格雷码计算得到的值（bit0 ~ 2）。
- 奇偶校验（parity）：通过格雷码计算后的值的奇偶校验（偶校验）。

CANFD 对 CRC 算法进行了改进，CRC 对填充位也加入了计算。在校验和部分为避免有连续位超过 6 个，确定在第一位及以后每 4 位添加一个填充位加以分割，这个填充位的值是上一位的反码，作为格式检查，如果填充位不是上一位的反码，进行出错处理。

CAN 的 CRC 的位数是 15 位，而在 CANFD 中，CRC 场扩展到了 21 位，如下：

当传输数据为 0 ~ 8 字节或更少时：CRC 15 位。

当传输数据为 9 ~ 16 字节或更少时：CRC 17 位。

当传输数据超过 17 ~ 64 个字节时：CRC 21 位。

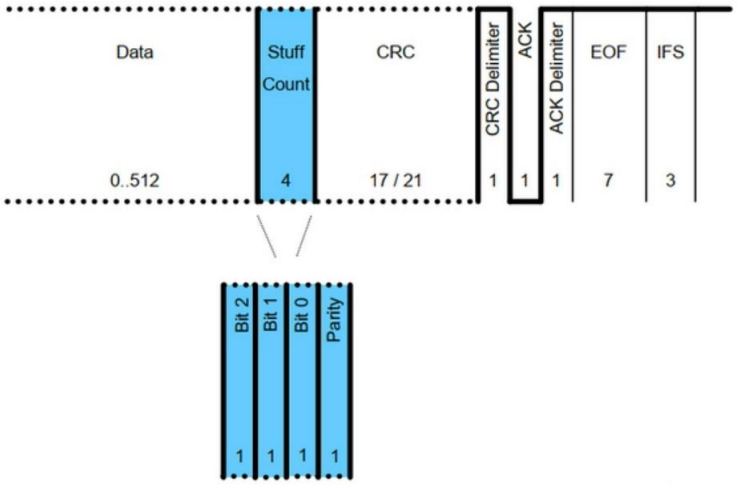



图 2-5：CRC 段

## 2.2.6 ACK 段

与 CAN 相比，在 CANFD 中最多可接受 2 个位时间有效的 ACK，允许 1 个额外的位时间来补偿收发器相移和传播延迟。

由高速的数据场到慢速的仲裁场时，时钟切换会引起收发器相移和总线传播延迟。为了补偿其相移和延迟，相比传统的 CAN，在 CANFD 中多加了额外的 1 位时间。

在 ACK 之后，发送 ACK 界定符。这是一个表示 ACK 结束的分隔符，是 1 位隐性位。

## 2.2.7 帧结束

与 CAN 一样，CANFD 的帧结尾为连续 7 位的隐性位。

# 3 CANFD 配置流程

## 3.1 发送 CANFD 数据帧

1. 开启 CANFD 时钟，释放复位，CAN 功能管脚复用。
2. 配置 CAN_CONFIG0[2]，CAN 工作在复位模式。
3. 配置总线时序寄存器 CAN_CONFIG1[31:16]。
4. 配置 CAN_CONFIG2[0]，CANFD 帧格式。
5. 寄存器 CAN_CONFIG0[31:24]清除错误标志位/中断标志位。
6. 配置 CAN_CONFIG1[7:0]，使能 TI 中断（可选）。
7. 配置 CAN_CONFIG0[2:1]，CAN 工作在其他模式，CAN 进入正常模式。
8. 配置 CAN_RTCONFIG[9:8]，选择 TX 缓冲区。
9. 配置发送缓存寄存器 CAN_TXBUF，根据定义的格式写入 CAN 数据帧内容，按发送的先后顺序写入，每次写入 32 位数据。
10. 配置指令寄存器 CAN_CONFIG0[14:9]，TXB0/1/2 启动发送。
11. 等待状态寄存器的 CAN_CONFIG0[18]置 1 后（若使能 TI 中断，此处可选择等待 TI 中断触发），数据发送完毕。

## 3.2 接收 CANFD 数据帧

1. 开启 CAN 时钟，释放复位，CAN 功能管脚复用。
2. 配置 CAN_CONFIG0[2]，CAN 工作在复位模式。
3. 配置总线时序寄存器 CAN_CONFIG1[31:16]。
4. 配置 CAN_CONFIG2[0]，CANFD 帧格式。
5. 寄存器 CAN_CONFIG0[31:24]清除错误标志位/中断标志位。
6. 配置 CAN_CONFIG1[7:0]，使能 TI 中断（可选）。
7. 设置接收过滤器配置，若使用单过滤器，CAN_CONFIG0[0]置 1。CAN_ACR 寄存器配置用户需要过滤筛选的内容，CAN_AMR 寄存器选择需要与 CAN_ACR 寄存器进行对比的位。若不需要进行比对，CAN_AMR 寄存器所有位置 1。
8. 配置 CAN_CONFIG0[2:1]，CAN 工作在其他模式，CAN 进入正常模式。
9. 等待状态寄存器 CAN_CONFIG0[23]置 1 后（若使能 RI 中断，此处可选择等待 RI 中断触发），读取接收缓存寄存器 CAN_RXBUF 数据，多次读取直到取出所有数据。



# 4 使用参考例程

本例程 CANFD 在 UM32x42x 平台上实现，具体可参考 UM32x42x_SDK Examples 中的 CAN 应用例程。

## 4.1 CANFD 的发送配置

以下简单介绍如何使用 hal 库对 CANFD 的发送进行配置.

1. 配置 CANFD 基本内容。其中需要配置 CAN 帧格式为 FD，开启波特率切换后，在 CANFD 帧传输过程中波特率会进行切换，传输完仲裁段后，数据帧的波特率会切换。

```
static void CANFD_Transmit(void)
{
    /*##-1- Configure the CAN peripheral ######################################*/
    CanHandle.Instance = CAN;
    CanHandle.Init.Mode = CAN_NORMAL_MODE;
    CanHandle.Init.CanFrameFormat = CAN_FRAME_FORMAT_FD;
    CanHandle.Fd.CanfdBaudRateSwitch = CANFD_BAUDRATE_SWITCH;
    CanHandle.Fd.CanfdExtbtSelect = CANFD_EXTBT_SELECT_NBT;
    CanHandle.Fd.CanfdFormatSelect = CANFD_FORMAT_SELECT_BOSCH;
```

选择FD帧格式
开启波特率切换
使用NBT寄存器配置波特率
帧格式符合博世CANFD规范

2. 配置波特率。

CANFD 波特率与 CAN2.0 的计算遵循统一公式：

**BaudRate = PCLK / [( TSEG1 + TSEG2 + 3 ) * (BPR + 1)]**

常见配置的仲裁波特率有：125K、250K，500K、1M，数据段波特率有 125K、500K、1M、2M、4M。

其中 PCLK 为系统时钟，根据公式计算可得出 TSEG1、TSEG2、BPR 的值，总体配置遵循 SEG1>SEG2。

同步跳转宽度（SJW）：该值直接影响到重同步时相位缓冲段的可调节的范围，SJW 的值可以在 1~4 之间选择，选择 3、4 可以使总线获得更宽的波。若在 FD 基础配置中开启了波特率切换和使用 NBT 寄存器配置波特率，则需要配置数据位时序寄存器 CAN_DBTCR。

在此例程中系统时钟为 96M，仲裁波特率为 1M，数据段波特率为 4M，具体波特率参数配置参考如下：

```
/*Baud rate calculation formula:
BaudRate = PCLK / ( ( TSEG1 + TSEG2 + 3 )  * (BPR + 1)
*  Arbitration baud = 1M
*  Data segment baud = 4M   */
CanHandle.Fd.CanfdNsjm = 2;
CanHandle.Fd.CanfdNbrp = 12;
CanHandle.Fd.CanfdNseg1 = 3;
CanHandle.Fd.CanfdNseg2 = 2;

CanHandle.Fd.CanfdDsjm = 1;
CanHandle.Fd.CanfdDbrp = 2;
CanHandle.Fd.CanfdDseg1 = 8;
CanHandle.Fd.CanfdDseg2 = 1;
HAL_CAN_Init(&CanHandle);
```

同步跳转宽度
仲裁段波特率
波特率预分频
采样点1
采样点2
数据段波特率
将上面的配置参数传入初始化

3. 配置发送帧格式。

```
CanHandle.TxHeader.DLC = 0xf;
CanHandle.TxHeader.IDE = CAN_ID_STD;
CanHandle.TxHeader.StdId =  0x592;
CanHandle.TxHeader.SELTX = CAN_SETTX_BUFFER0;

/* Load Send Data*/
HAL_CAN_AddTxMessage(&CanHandle, CAN_Tx_buf);
```

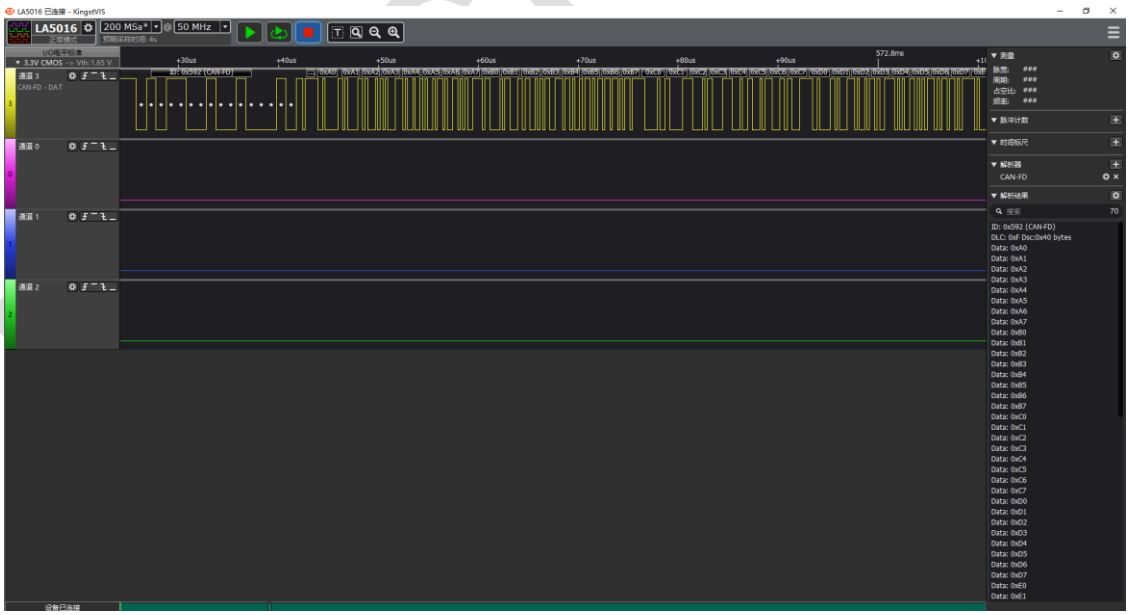
数据个数
ID格式
帧ID
选择发送buff
发送数据buff

4. 调用 HAL_CAN_Transmit 函数数据并启动发送。

```
if(HAL_OK == HAL_CAN_Transmit(&CanHandle ,0xFFFF))
{
    printf("CAN Send complete! \n");
    Can_Tx_Message_Display(&CanHandle);
}
else
{
    printf("CAN Send data fail!!! \r\n");
}
```

超时时间设定

5. 使用支持 CANFD 的逻辑分析仪，设置 BRS Wdith 为 67.5%+32.5%，即可解析出数据。



## 4.2 CANFD 的接收配置

CANFD 接收的基本配置与发送配置基本一致，需要注意的是接收端有过滤器功能需要配置。配置了过滤器后只有当接收到的消息的标识符位等于验收代码寄存器中的预定义位时，CAN 控制

器中的接收过滤器才有可能将接收到的消息传递给 RX FIFO。过滤器共有四种配置：标准单过滤、拓展单过滤、标准双过滤、拓展双过滤。

1. 标准单过滤：第一个过滤器与 ID1 和 rtr 位对比，第二个过滤器与数据 1、数据 2 对比。

```
CAN_FilterTypeDef Filter_Hdr;

Filter_Hdr.FilterMode = HAL_CAN_FilterMode_SINGLE;
Filter_Hdr.FilterStdId1 = 0x592;
Filter_Hdr.FilterData1 = 0xa0;
Filter_Hdr.FilterData2 = 0xa1;
Filter_Hdr.FilterIde = CAN_ID_STD;
Filter_Hdr.FilterMaskType = CAN_Filter_MASK_NONE;
```

配置单过滤器
对比ID
对比数据1
对比数据2
启用所有位对比

2. 拓展单过滤：过滤器比较扩展标识符范围的前两个字节。

```
Filter_Hdr.FilterMode = HAL_CAN_FilterMode_SINGLE;
Filter_Hdr.FilterExtId1 = 0x11E8E977;
Filter_Hdr.FilterIde = CAN_ID_EXT;
Filter_Hdr.FilterMaskType = CAN_Filter_MASK_NONE;
```

3. 标准双过滤：第一个过滤器与 ID1 和 rtr 对比和第一个数据，或者第二个过滤器与 ID2 对比。

```
Filter_Hdr.FilterMode = HAL_CAN_FilterMode_DOUBLE;
Filter_Hdr.FilterStdId1 = 0x592;
Filter_Hdr.FilterStdId2 = 0x592;
Filter_Hdr.FilterData1 = 0xa0;
Filter_Hdr.FilterIde = CAN_ID_STD;
Filter_Hdr.FilterMaskType = CAN_Filter_MASK_NONE;
```

4. 拓展双过滤：双过滤器模式接收到扩展帧，则两个过滤器比较扩展标识符范围的前两个字节。

```
Filter_Hdr.FilterMode = HAL_CAN_FilterMode_DOUBLE;
Filter_Hdr.FilterExtId1 = 0x11E8E977;
Filter_Hdr.FilterExtId2 = 0x11E8E977;
Filter_Hdr.FilterIde = CAN_ID_EXT;
Filter_Hdr.FilterMaskType = CAN_Filter_MASK_NONE;
```

5. 调用 HAL_CAN_ConfigFilter 函数即可对 CAN 过滤器完成配置。

```
HAL_CAN_ConfigFilter(&CanHandle,&Filter_Hdr);  // 配置过滤器
```

# 5 注意事项

在使用 CANFD 通信时，系统时钟源最好使用外部晶振，否则当通信数据量大时，总线上波形会出现偏差，可能导致从机接收错误数据，或者逻辑分析仪解析的数据会出错。

# 6 版本维护

| 版本 | 日期 | 描述 |
| --- | --- | --- |
| V1.0 | 2023.09.11 | 初始版 |
| | | |
| | | |
| | | |
| | | |