

**GigaDevice Semiconductor Inc.**

**GD32A5x3 CAN 位时间问题的软件规避**

**应用笔记**

**AN222**

1.0 版本

(2024 年 8 月)

## 目录

目录.....	1
图索引 .....	2
表索引 .....	3
1. 前言.....	4
2. 受影响芯片型号.....	5
3. CAN 模块限制.....	6
4. 解决方案 .....	7
4.1. 方案描述.....	7
4.1.1. 相关公式.....	7
4.1.2. 解决步骤.....	8
4.2. 方案示例.....	8
5. 版本历史 .....	13

## 图索引

图 2-1. MCU 产品版本号 .....	5
图 3-1. CAN 帧位时间问题发送示例 .....	6
图 4-1. 收发器延迟参数查询 .....	8

## 表索引

表 1-1.CAN 帧位时间问题总结 .....	4
表 2-1.适用 MCU 产品.....	5
表 3-1.CAN 帧位时间问题配置示例.....	6
表 4-1. 发送端参考配置.....	8
表 4-2. 发送测试配置 .....	10
表 4-3. 接收端参考配置.....	11
表 5-1. 版本历史.....	13

# 1. 前言

GD32A5x3xx 系列 MCU 在使用 CAN 模块时有一个限制。

当 MCU 作为 CAN 发送端时，如果发送端总延迟大于一定值，会触发芯片的位时间重同步机制，可能导致 CAN 发送端的位时间变长。此时若接收节点重同步能力不足以补偿位时间时，有一定概率会出现信号错误，通信失败。

对于发送 CAN 常规帧或者 FD 帧，情形总结如下：

**表 1-1.CAN 帧位时间问题总结**

序号	发送帧	发送问题
1	常规帧	发送节点会进行重同步，触发重同步后，会导致显性位时间拉长
2	FD帧+波特率切换禁能	发送节点会进行重同步，触发重同步后，会导致显性位时间拉长
3	FD帧+波特率切换使能	发送节点数据段不会进行重同步，控制段会进行重同步

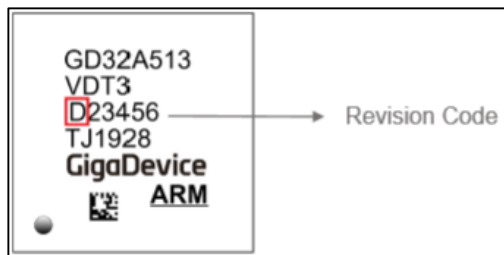
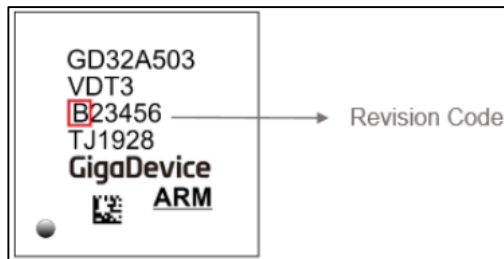
## 2. 受影响芯片型号

表 2-1. 适用 MCU 产品

类型	型号	产品版本
MCU	GD32A503xx系列	B版
	GD32A513xx系列	D版

MCU 产品版本号查看可参考下图：

图 2-1. MCU 产品版本号



### 3. CAN 模块限制

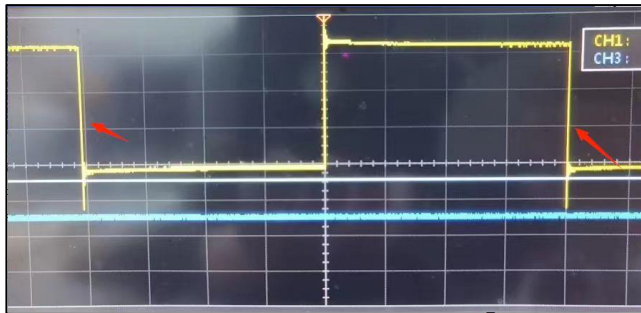
MCU CAN 模块的限制会引起以下的现象：

(1) MCU 发送的 CAN 帧实际位时间长度大于设定的波特率所代表的位时间长度。如下配置情况与实际发送情况所示，CAN 帧发送的显性位比隐性位长；

表 3-1.CAN 帧位时间问题配置示例

波特率 (bps)	位包含位时间单元数量	显性电平时间 (us)	隐性电平时间 (us)	位时间单元
500K	25	2.04	2.00	80ns

图 3-1. CAN 帧位时间问题发送示例



- (2) MCU CAN 帧有一定概率发送不成功，一直重发；
- (3) MCU CAN 发送节点有一定概率发生 ACK 错误；
- (4) 接收节点有一定概率会发生帧格式错误，CRC 校验错误，出现信号错误，通信失败。

由[表 1-1.CAN 帧位时间问题总结](#)可得知对发送节点的仲裁段配置进行分析即可。因此 CAN 模块限制的触发需满足以下情况：

- (1) MCU 作为 CAN 发送节点；
- (2) MCU CAN 发送节点在发送 CAN 帧的隐性位到显性位时；
- (3) MCU CAN 发送节点总延迟  $T_{\text{delay\_total\_tx}}$  (MCU 端收发器延迟+内部处理的 2 个 CAN 时钟延迟)  $> T_{\text{ss\_tx}}$  同步段时间 (1 个  $T_{\text{q\_tx}}$ ) 时；
- (4) 当接收节点重同步能力  $\text{SJW}_{\text{rx}} * T_{\text{q\_rx}}$  小于 MCU CAN 发送端补偿时间时，补偿时间为  $\min\{\text{ceil}((T_{\text{delay\_total\_tx}} - T_{\text{ss\_tx}})/T_{\text{CANCLK\_tx}}), \text{SJW}_{\text{tx}} * T_{\text{q\_tx}}/T_{\text{CANCLK\_tx}}\} * T_{\text{CANCLK\_tx}}$ 。

注意：ceil()函数为向上取整函数。

## 4. 解决方案

### 4.1. 方案描述

#### 4.1.1. 相关公式

$$T_{\text{delay\_total\_tx}} = T_{\text{delay\_phy\_tx}} + 2 \times T_{\text{CANCLK\_tx}}$$

$$T_{\text{ss\_tx}} = T_{\text{q\_tx}} = \frac{1}{\text{Baudrate} \times N_{\text{q\_tx}}}$$

$$\text{发送节点补偿时间} = \min\{\text{ceil}((T_{\text{delay\_total\_tx}} - T_{\text{ss\_tx}}) / T_{\text{CANCLK\_tx}}), \text{SJW}_{\text{tx}} \times T_{\text{q\_tx}} / T_{\text{CANCLK\_tx}}\} \times T_{\text{CANCLK\_tx}}$$

$$\text{接收节点重同步能力} = \text{SJW}_{\text{rx}} \times T_{\text{q\_rx}}$$

其中各名词解释如下：

**$T_{\text{delay\_total\_tx}}$**

为发送端总延迟。

**$T_{\text{delay\_phy\_tx}}$**

为发送端收发器延迟。

**$T_{\text{CANCLK\_tx}}$**

为发送端 CAN 时钟时间。

**$T_{\text{ss\_tx}}$**

为发送端同步段时间。

**$T_{\text{q\_tx}}$**

为发送端位时间单元。

**$N_{\text{q\_tx}}$**

为发送端位时间所占的时间单元数量。

**$\text{SJW}_{\text{tx}}$**

为发送端的再同步补偿占用的时间单元数量。

**$T_{\text{q\_rx}}$**

为接收端位时间单元。

**$N_{\text{q\_rx}}$**

为接收端位时间所占的时间单元数量。



## SJW<sub>rx</sub>

为接收端的再同步补偿占用的时间单元数量。

### 4.1.2. 解决步骤

为了规避这个问题：

1. CANCLK 需要配置为最大 CAN 时钟，在 GD32A5x3xx 产品中，系统时钟最大为 100M，配置 CANCLK 为 PCLK2 时钟 100M，则  $T_{CANCLK\_tx}$  等于 10ns。

2. 查询发送端收发器数据手册，如下图，以 MCP2561 为例，需查看 TXD 到 RXD 的时间延迟参数。 $T_{delay\_phy\_tx}$  为最大 235ns 时， $T_{delay\_total\_tx}$  等于 255ns。

图 4-1.收发器延迟参数查询

2.3 AC Characteristics							
Electrical Characteristics: Extended (E): T <sub>AMB</sub> = -40°C to +125°C and High (H): T <sub>AMB</sub> = -40°C to +150°C; V <sub>DD</sub> = 4.5V to 5.5V, V <sub>IO</sub> = 1.8V to 5.5V (Note 2), R <sub>L</sub> = 60Ω; unless otherwise specified.							
Param. No.	Sym.	Characteristic	Min.	Typ.	Max.	Units	Conditions
1	tBIT	Bit Time	1	—	69.44	μs	
2	fBIT	Bit Frequency	14.4	—	1000	kHz	
3	tTXD-BUSON	Delay TXD Low to Bus Dominant	—	—	70	ns	
4	tTXD-BUSOFF	Delay TXD High to Bus Recessive	—	—	125	ns	
5	tBUSON-RXD	Delay Bus Dominant to RXD	—	—	70	ns	
6	tBUSOFF-RXD	Delay Bus Recessive to RXD	—	—	110	ns	
7	tTXD - RXD	Propagation Delay TXD to RXD	—	—	125	ns	Negative edge on TXD
8			—	—	235	ns	Positive edge on TXD
9	tFLTR(WAKE)	Delay Bus Dominant to RXD (Standby mode)	0.5	1	4	μs	Standby mode

3.  $T_{ss\_tx}$  需要尽可能的大，假设 Baudrate 为 500Kbps,  $N_{q\_tx}$  为 10，则  $T_{ss\_tx} = T_{q\_tx}$  等于 200ns。

4. SJW<sub>tx</sub> 配置为 2，则发送端具有调整 400ns 重同步的能力，适用于绝大多数场景。

5. 发送端在以上配置下时，隐性位到显性位时引入了  $\min\{\text{ceil}((T_{delay\_total\_tx} - T_{ss\_tx})/T_{CANCLK\_tx}), SJW_{tx} * T_{q\_tx} / T_{CANCLK\_tx} * T_{CANCLK\_tx}\}$  等于  $\min\{\text{ceil}((255ns - 200ns)/10ns), 2 * 200ns / 10ns\} * 10ns = \min\{\text{ceil}(5.5), 40\} * 10ns = 6 * 10ns = 60ns$  的延迟。

6. 接收节点重同步能力  $SJW_{rx} * T_{q\_rx}$  需要尽可能大，至少满足  $SJW_{rx} * T_{q\_rx} \geq 60ns$ ，由于收发两端波特率需要相同，当  $N_{q\_rx}$  数目固定不可配时，也即  $T_{q\_rx}$  值不可调整时，需按照  $SJW_{rx} \geq 60ns * \text{Baudrate} * N_{q\_rx}$ ，即  $SJW_{rx} \geq 0.03 * N_{q\_rx}$  进行配置。

## 4.2. 方案示例

遵循以上分析，系统时钟为 100M，通信波特率位 500Kbps 时，发送端参考配置如下：

表 4-1. 发送端参考配置

```
void can_gpio_config(void)
{
    /* enable CAN clock */
    rcu_can_clock_config(CAN0, RCU_CANSRC_PCLK2); // CANCLK_tx = 100M, T_CANCLK_tx=10ns
    rcu_periph_clock_enable(RCU_CAN0);
    /* enable CAN port clock */
}
```

```
rcu_periph_clock_enable(RCU_GPIOB);

/* configure CAN0 GPIO */
gpio_output_options_set(GPIOB, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_13);
gpio_mode_set(GPIOB, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_13);
gpio_af_set(GPIOB, GPIO_AF_6, GPIO_PIN_13);

gpio_output_options_set(GPIOB, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_14);
gpio_mode_set(GPIOB, GPIO_MODE_AF, GPIO_PUPD_NONE, GPIO_PIN_14);
gpio_af_set(GPIOB, GPIO_AF_6, GPIO_PIN_14);
}

void can_config(void)
{
    can_parameter_struct can_parameter;

    /* initialize CAN register */
    can_deinit(CAN0);
    /* initialize CAN */
    can_struct_para_init(CAN_INIT_STRUCT, &can_parameter);

    /* initialize CAN parameters */
    can_parameter.internal_counter_source = CAN_TIMER_SOURCE_BIT_CLOCK;
    can_parameter.self_reception = DISABLE;
    can_parameter.mb_tx_order = CAN_TX_HIGH_PRIORITY_MB_FIRST;
    can_parameter.mb_tx_abort_enable = ENABLE;
    can_parameter.local_priority_enable = DISABLE;
    can_parameter.mb_rx_ide_rtr_type = CAN_IDE_RTR_FILTERED;
    can_parameter.mb_remote_frame = CAN_STORE_REMOTE_REQUEST_FRAME;
    can_parameter.rx_private_filter_queue_enable = DISABLE;
    can_parameter.edge_filter_enable = DISABLE;
    can_parameter.protocol_exception_enable = DISABLE;
    can_parameter.rx_filter_order = CAN_RX_FILTER_ORDER_MAILBOX_FIRST;
    can_parameter.memory_size = CAN_MEMSIZE_32_UNIT;
    /* filter configuration */
    can_parameter.mb_public_filter = 0x0;
    can_parameter.resync_jump_width = 2; //SJWtx=2*Tq_tx=400ns
    can_parameter.prop_time_segment = 2;
    can_parameter.time_segment_1 = 4;
    can_parameter.time_segment_2 = 3; //Nq_tx=1+PTS+PBS1+PBS2=10
    /* 500Kbps*/
}
```

```
can_parameter.prescaler = 20; //Baudrate= CANCLK_tx/(prescaler*Nq_tx)=500Kbps
                                //Tss_tx=Tq_tx=1/(Baudrate*Nq_tx)=prescaler/CANCLK_tx=200ns
/* initialize CAN */
can_init(CAN0, &can_parameter);

can_operation_mode_enter(CAN0, CAN_NORMAL_MODE);
}
```

发送邮箱配置、测试数据参考如下，建议使用 8 个字节 0x55 数据进行测试：

表 4-2. 发送测试配置

```
int main(void)
{
    const uint8_t tx_data[8] = {0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55};
    can_mailbox_descriptor_struct transmit_message;
    FlagStatus can_tx_state;
    uint8_t i = 0;
    ... ..

    can_struct_para_init(CAN_MDSC_STRUCT, &transmit_message);
    /* initialize transmit message */
    transmit_message.rtr = 0;
    transmit_message.ide = 0;
    transmit_message.code = CAN_MB_TX_STATUS_DATA;
    transmit_message.brs = 0;
    transmit_message.fdf = 0;
    transmit_message.prio = 0;
    transmit_message.data_bytes = 8;
    /* tx message content */
    transmit_message.data = (uint32_t*)(tx_data);
    transmit_message.id = 0x55;
    can_tx_state = RESET;
    ... ..
    while(1) {
        ... ..
        if((RESET == can_tx_state) || (SET == can_flag_get(CAN0, CAN_FLAG_MB1))){
            can_tx_state = SET;
            can_flag_clear(CAN0, CAN_FLAG_MB1);
            /* transmit message */
            can_mailbox_config(CAN0, 1, &transmit_message);

            printf("\r\nCAN0 transmit data: \r\n");
            for(i = 0; i < 8; i++) {
                printf("%02x\r\n", tx_data[i]);
            }
        }
    }
}
```

```
    }  
    }  
    .....  
    }  
}
```

接收端参考配置如下：

表 4-3. 接收端参考配置

```
void can_gpio_config(void)  
{  
    /* enable CAN clock */  
    rcu_can_clock_config(CAN1, RCU_CANSRC_PCLK2); // CANCLK_rx = 100M, T_CANCLK_rx=10ns  
    rcu_periph_clock_enable(RCU_CAN1);  
    /* enable CAN port clock */  
    rcu_periph_clock_enable(RCU_GPIOD);  
  
    /* configure CAN1 GPIO */  
    gpio_output_options_set(GPIOD, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_6);  
    gpio_mode_set(GPIOD, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_6);  
    gpio_af_set(GPIOD, GPIO_AF_6, GPIO_PIN_6);  
  
    gpio_output_options_set(GPIOD, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_7);  
    gpio_mode_set(GPIOD, GPIO_MODE_AF, GPIO_PUPD_NONE, GPIO_PIN_7);  
    gpio_af_set(GPIOD, GPIO_AF_6, GPIO_PIN_7);  
}  
  
void can_config(void)  
{  
    can_parameter_struct can_parameter;  
  
    /* initialize CAN register */  
    can_deinit(CAN1);  
    /* initialize CAN */  
    can_struct_para_init(CAN_INIT_STRUCT, &can_parameter);  
  
    /* initialize CAN parameters */
```

```
can_parameter.internal_counter_source = CAN_TIMER_SOURCE_BIT_CLOCK;
can_parameter.self_reception = DISABLE;
can_parameter.mb_tx_order = CAN_TX_HIGH_PRIORITY_MB_FIRST;
can_parameter.mb_tx_abort_enable = ENABLE;
can_parameter.local_priority_enable = DISABLE;
can_parameter.mb_rx_ide_rtr_type = CAN_IDE_RTR_FILTERED;
can_parameter.mb_remote_frame = CAN_STORE_REMOTE_REQUEST_FRAME;
can_parameter.rx_private_filter_queue_enable = DISABLE;
can_parameter.edge_filter_enable = DISABLE;
can_parameter.protocol_exception_enable = DISABLE;
can_parameter.rx_filter_order = CAN_RX_FILTER_ORDER_MAILBOX_FIRST;
can_parameter.memory_size = CAN_MEMSIZE_32_UNIT;
/* filter configuration */
can_parameter.mb_public_filter = 0x0;
can_parameter.resync_jump_width = 4; //SJWrx=4*Tq_rx=320ns
can_parameter.prop_time_segment = 6;
can_parameter.time_segment_1 = 11;
can_parameter.time_segment_2 = 7; // Nq_rx=1+PTS+PBS1+PBS2=25
/* 500Kbps */
can_parameter.prescaler = 8; //Baudrate= CANCLK_rx/(prescaler*Nq_rx)=500Kbps
//Tss_rx=Tq_rx=1/(Baudrate*Nq_rx)=prescaler/CANCLK_rx=80ns
/* initialize CAN */
can_init(CAN1, &can_parameter);

/* configure CAN1 NVIC */
nvic_irq_enable(CAN1_Message_IRQn, 0, 0);

/* enable CAN MBO interrupt */
can_interrupt_enable(CAN1, CAN_INT_MB0);

can_operation_mode_enter(CAN1, CAN_NORMAL_MODE);
}
```

## 5. 版本历史

表 5-1. 版本历史

版本号	说明	日期
1.0	首次发布	2024 年 08 月 05 日

## Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the "Product"), is owned by the Company under the intellectual property laws and treaties of the People's Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as its suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.