XHSC 小华半导体
XIAOHUA SEMICONDUCTOR

HC32A4A0 系列 CAN 控制器

未定义波形检测及恢复

应用笔记

Rev1.0　2023 年 03 月

适用对象

产品系列	产品型号
HC32A4A0	HC32A4A0PITI-LQFP100 HC32A4A0SITI-LQFP176

声　明

★ 小华半导体有限公司（以下简称：XHSC）保留随时更改、更正、增强、修改小华半导体产品和/或本文档的权利，恕不另行通知。用户可在下单前获取最新相关信息。XHSC 产品依据购销基本合同中载明的销售条款和条件进行销售。

★ 客户应针对您的应用选择合适的 XHSC 产品，并设计、验证和测试您的应用，以确保您的应用满足相应标准以及任何安全、安保或其它要求。客户应对此独自承担全部责任。

★ XHSC 在此确认未以明示或暗示方式授予任何知识产权许可。

★ XHSC 产品的转售，若其条款与此处规定不同，XHSC 对此类产品的任何保修承诺无效。

★ 任何带有®或™标识的图形或字样是 XHSC 的商标。所有其他在 XHSC 产品上显示的产品或服务名称均为其各自所有者的财产。

★ 本通知中的信息取代并替换先前版本中的信息。

目　录

1 概述

在使用 HC32A4A0 的 CAN 控制器时，若总线受到干扰而产生错误，CAN 控制器可能发出未定义波形占用总线而使所有节点无法发送和接收数据。为了协助用户更好地使用 CAN 控制器，本文将介绍一种自动检测未定义波形的方法。

2 检测原理

本方法的原理是：在 CAN 发送过程中，用定时器计数单位时间 t 内 CAN_TXD 引脚下降沿产生的外部中断事件次数 n，计数次数 n 将通过 DMA 传输至 DCU 单元与预设区间进行比较，比较成功时 DCU 将产生中断。若计数次数 n 累计多次在预设区间内，则表示 CAN_TXD 在输出未定义波形。

由于未定义波形一个周期由 6 个 CAN 标称位（nominal bit）——5 个隐性位和 1 个显性位组成（见图 2-1），若 CAN_TXD 正在输出未定义波形，那么理论上单位时间 t 内其输出的下降沿个数是恒定的。一般情况下，只要单位时间 t 足够长，如至少 1 个数据帧的时间长度（在实现本方法的例程中，t 为 180 个 CAN 位时间），CAN_TXD 正常发送数据帧产生的下降沿次数（见图 2-2）是远大于发送未定义波形时产生的下降沿次数的。

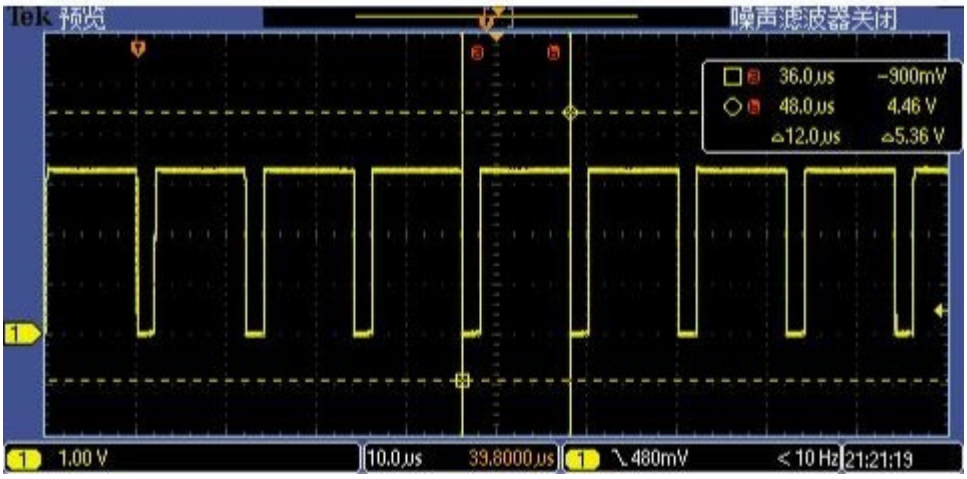

图 2-1 未定义波形

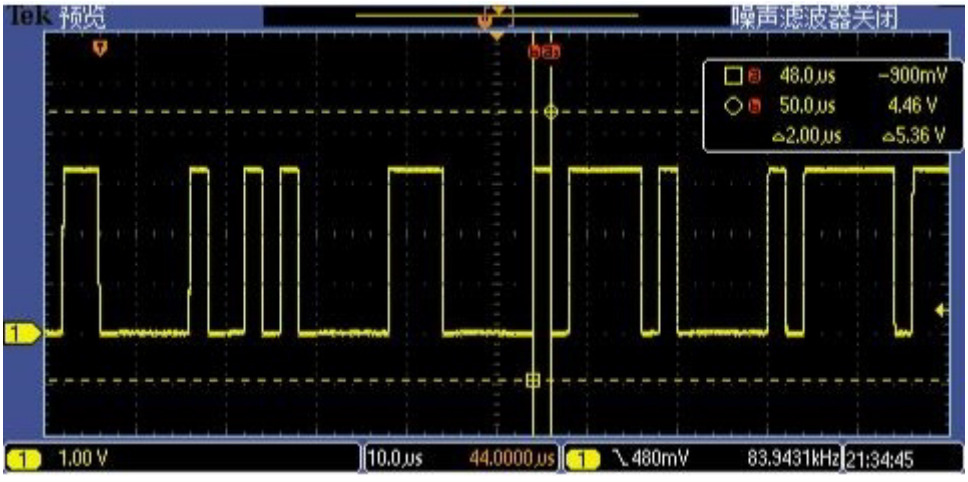

图 2-2 正常数据帧的一部分波形

2.1 辅助外设

要实现未定义波形的自动检测，需要定时器等他外设的辅助。

2.1.1 定时器

本方法需要用到两个定时器。

一个定时器用于计数 CAN_TXD 下降沿产生的外部中断事件（用 CW_TMR 表示）；一个定时器用于计时（用 CT_TMR 表示），计时的周期为 t。

CW_TMR 为硬件计数模式，以 CAN_TXD 的外部中断事件为硬件向上计数条件，并且以 CT_TMR 计数溢出事件为捕获和清零条件，即当 CT_TMR 计时 t 到达后，CW_TMR 捕获当前计数值 n，并对计数值寄存器清零。

2.1.2 DMA

DMA 用于将 CW_TMR 的在单位时间 t 内的计数值 n 传输到 DCU，与预设的 CAN_TXD 下降沿个数区间进行比较，其传输数据的触发事件为 CW_TMR 的捕获事件。

2.1.3 DCU

DCU 用于比较 CW_TMR 在单位时间 t 内计数的 CAN_TXD 下降沿个数 n 是否在预设区间内，若在预设区间内，则产生中断。

2.2 原理框图

由定时器、DMA 和 DCU 等辅助外设自动检测未定义波形的原理如图 2-3。

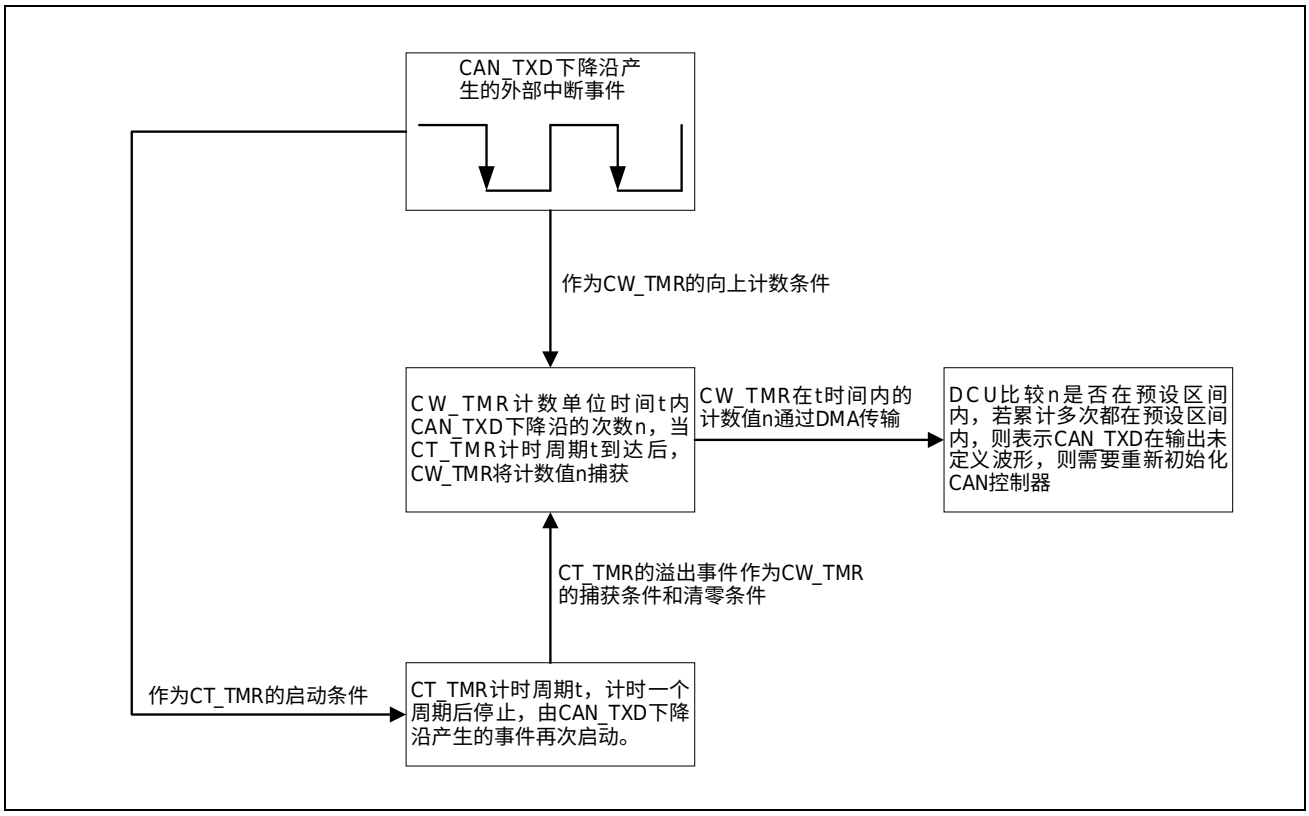

图 2-3 原理框图

3 应用举例

例程 can_undefined_wave_detect 用 Timer6 等辅助外设实现了该未定义波形的检测方法，实现方法在下文中详细介绍。

该例程的代码请访问小华官网（www.xhsc.com.cn），参考 HC32F4A0 产品页面下“技术文档”部分中《AN_HC32F4A0 系列 CAN 控制器未定义波形检测及恢复_Rev1.0.zip》内的样例工程。

3.1 CAN 控制器

例程用到的 CAN 控制器及引脚定义见表 3-1。

表 3-1 CAN 控制器及引脚

符号	定义	说明
CAN_UNIT	CM_CAN1	例程用的 CAN 控制器为 CAN1
CAN_TX_PORT	GPIO_PORT_D	CAN_TXD 为 PD5
CAN_TX_PIN	GPIO_PIN_05	
CAN_TX_PIN_EVT	EVT_SRC_PORT_EIRQ5	CAN_TXD 产生的事件为 EVT_SRC_PORT_EIRQ5，将用作 CW_TMR 的硬件向上计数条件以及 CT_TMR 的硬件启动条件
CAN_RX_PORT	GPIO_PORT_D	CAN_RXD 为 PD4
CAN_RX_PIN	GPIO_PIN_04	

3.2 辅助外设

例程用到的辅助外设见表 3-2。

表 3-2 辅助外设

符号	定义	说明
CW_TMR	CM_TMR6_5	Timer6 单元 5，用于计数 CAN_TXD 在单位时间 t 内输出的下降沿个数 n。其硬件向上计数条件为 CAN_TXD 下降沿产生的外部中断事件，例程中 CAN_TXD 引脚为 PD5，事件为 EVT_SRC_PORT_EIRQ5；其捕获和硬件清零条件为 CM_TMR6_2 的计数溢出事件 EVT_SRC_TMR6_2_OVF
CT_TMR	CM_TMR6_2	Timer6 单元 2，用于计时单位时间 t。其硬件启动条件为 CAN_TXD 下降沿产生的外部中断事件，例程中 CAN_TXD 引脚为 PD5，事件为 EVT_SRC_PORT_EIRQ5；每次启动只计时一个周期；计数溢出后产生的事件 EVT_SRC_TMR6_2_OVF 作为 Timer6 单元 5 的捕获和硬件清理条件
DMA_UNIT	CM_DMA2	DMA 单元 2，用于将 Timer6 单元 5 在单位时间 t 内计数的 CAN_TXD 输出的下降沿个数 n 传输至 DCU 进行比较
DCU_UNIT	CM_DCU1	DCU 单元 1，用于比较 CAN_TXD 在单位时间 t 内输出的下降沿个数 n 是否在预设区间内，若在预设区间内，则产生中断

3.3 重要的常量

一些重要的常量定义见表 3-3。

表 3-3 一些重要的常量定义

符号	定义	说明
CAN_UNDEF_WAVE_WIDTH	6	未定义波形一个周期由 5 个隐性位和 1 个显性位组成
CAN_UNDEF_WAVE_BIT_CNT	180	例程设定 CT_TMR 计时的单位时间 t 对应为 180 个 CAN 位时间，该定义用户可根据自己的实际应用需求进行修改。理论上，若 CAN_TXD 输出未定义波形，单位时间 t 内 CAN_TXD 输出的下降沿个数为 30
CAN_UNDEF_WAVE_LOWER_LIMIT	29	CAN_TXD 输出未定义波形时下降沿个数下限
CAN_UNDEF_WAVE_UPPER_LIMIT	31	CAN_TXD 输出未定义波形时下降沿个数上限
CAN_UNDEF_WAVE_DETECTED_CNT	5	CAN 每次发送启动后，累计 5 次检测到 CAN_TXD 输出的下降沿个数 n 满足：29 <= n <= 31，则表示 CAN_TXD 在输出未定义波形，则需要重新初始化 CAN 控制器。该常量数值可根据实际应用对帧率的要求，进行适当的减小或增加。根据例程配置的 CAN 波特率 500Kbps 及 CAN_UNDEF_WAVE_DETECTED_CNT 和 CAN_UNDEF_WAVE_BIT_CNT 定义，确定未定义波形的时间约为 2us*180*5=1.8ms

3.4 全局变量

例程用到的全局变量说明见表 3-4。

表 3-4 全局变量

变量	类型	说明
m_u8TxFlag	__IO uint8_t	发送成功标志位
m_u8UndefWaveCount	__IO uint8_t	发送过程中检测到未定义波形次数统计。例程中，当其大于等于 5 时表示检测到未定义波形，同时重新初始化 CAN 控制器，并置位标志位 m_u8TxFailFlag
m_u8TxFailFlag	__IO uint8_t	发送失败标志位，该标志位供用户使用，当检测到该标志位置位时，表示在发送当前帧时检测到了未定义波形，当前帧发送失败，需要重新填充并发送

3.5 时序图和流程图

例程用到的各辅助外设联动检测未定波形时序如图 3-1。

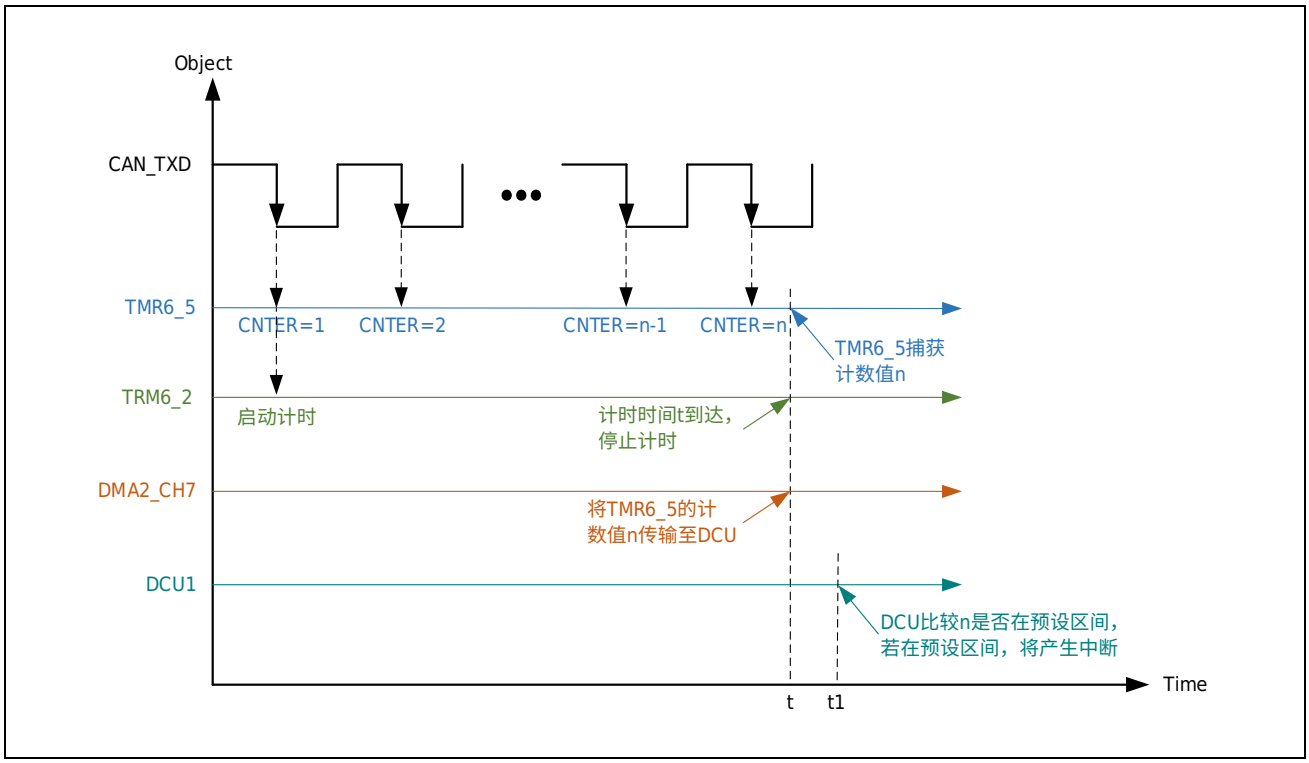

图 3-1 检测未定义波形的时序图

检测未定义波形只在发送时检测，其流程如图 3-2。

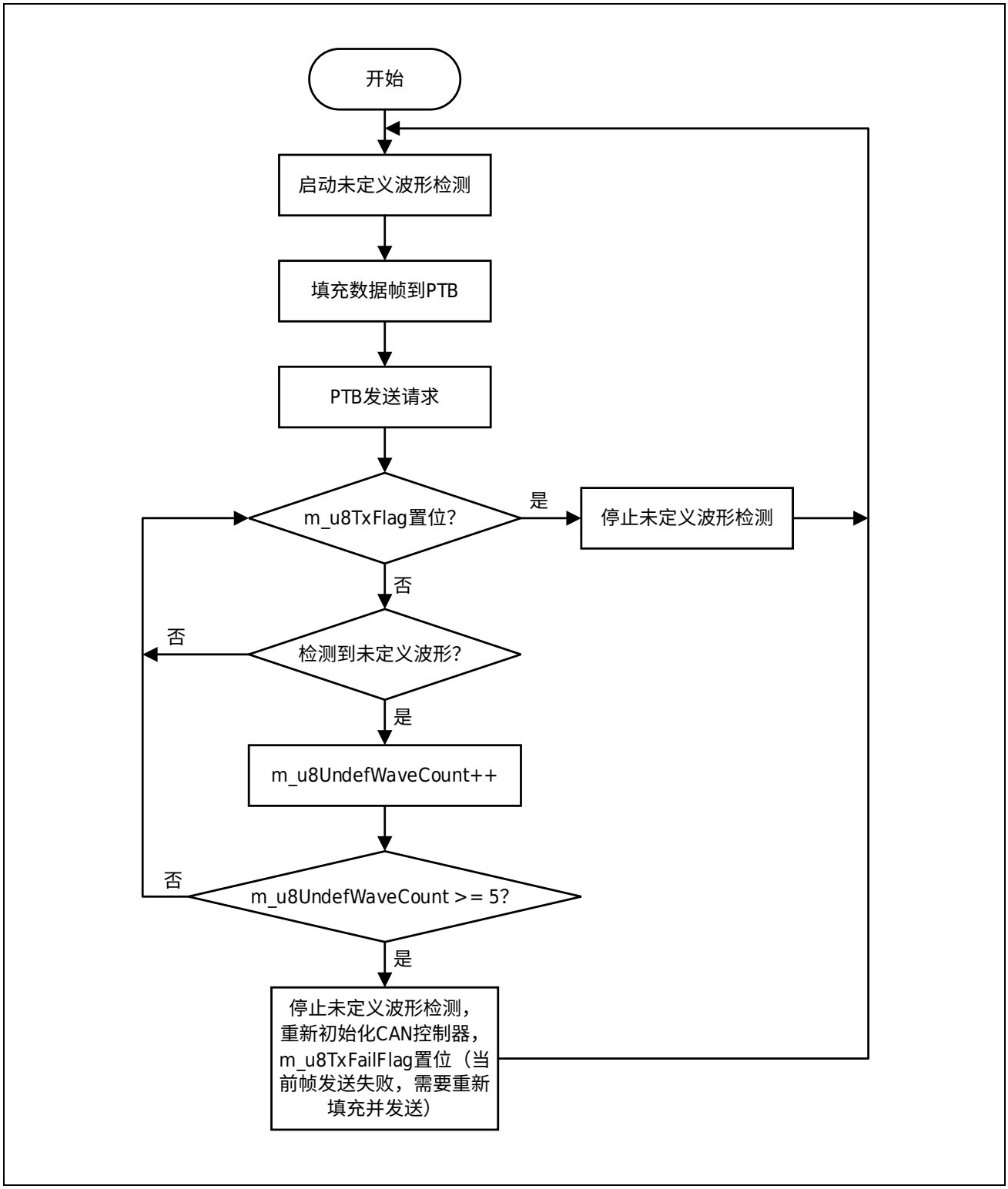

图 3-2 增加检测未定义波形的 CAN 发送流程

在 CAN 发送过程中，辅助外设自动检测未定义波形的流程如图 3-3。

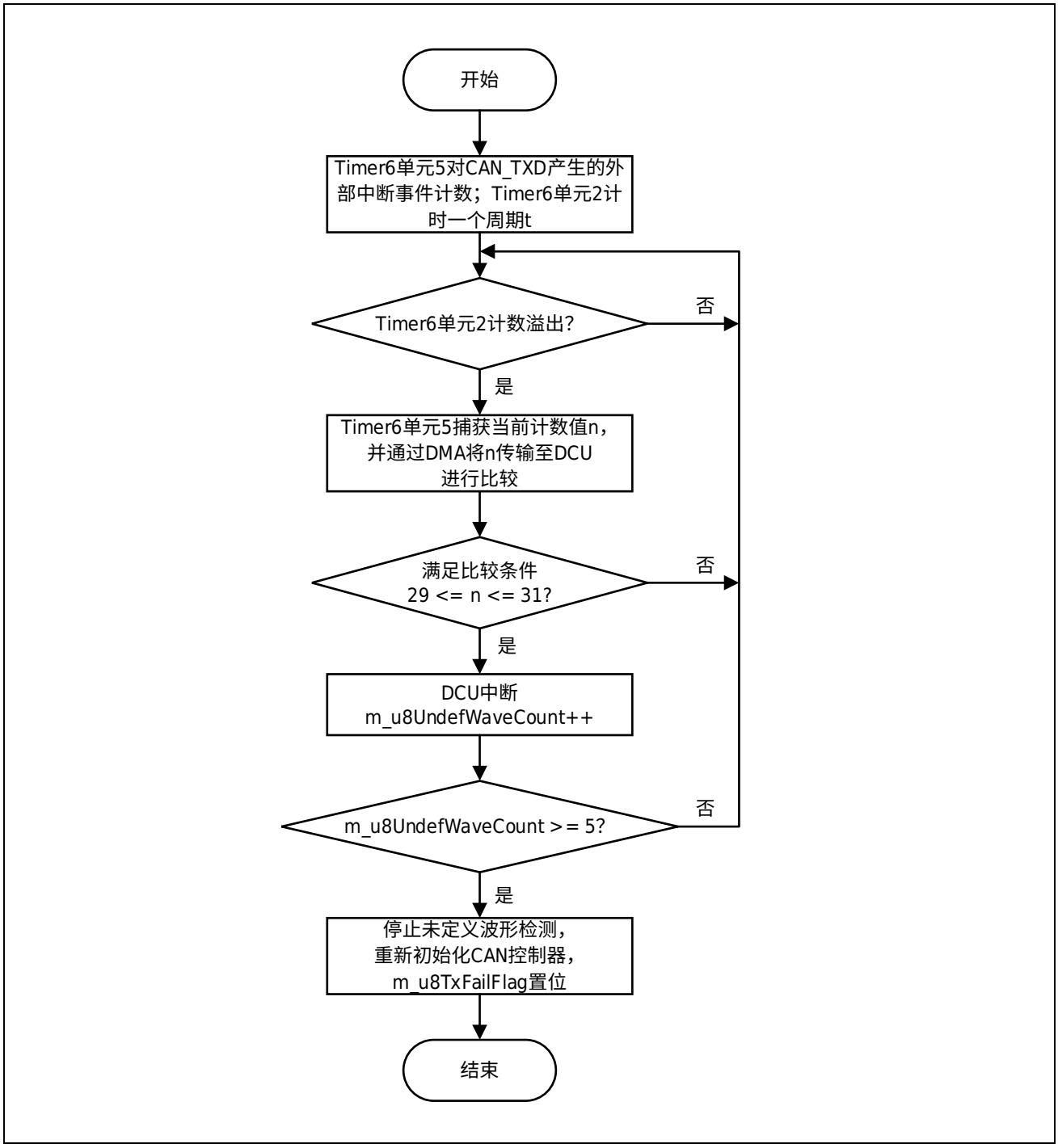

图 3-3 在 CAN 发送过程中辅助设备检测未定义波形的流程

3.6 应用代码说明

3.6.1 CAN 控制器初始化配置

例程使用系统时钟的 6 分频作为 CAN 通信时钟，配置 CAN 的波特率为 500Kbps。

```
/* CAN baudrate 500Kbps */
#define CAN_BAUDRATE                    (500000UL)
#define CAN_BIT_PRESC                   (4U)
#define CAN_BIT_TIME_SEG1               (16)
#define CAN_BIT_TIME_SEG2               (4)
#define CAN_BIT_TIME_SJW                (4)
```

```
static void CanInitConfig(void)
{
    stc_can_init_t stcCanInit;
    stc_can_filter_config_t astcFilter[CAN_FILTER_NUM] = {
        {CAN_FILTER1_ID, CAN_FILTER1_ID_MASK, CAN_FILTER1_ID_TYPE},
    };

    /* Initializes CAN. */
    (void)CAN_StructInit(&stcCanInit);
    stcCanInit.stcBitCfg.u32Prescaler = CAN_BIT_PRESC;
    stcCanInit.stcBitCfg.u32TimeSeg1  = CAN_BIT_TIME_SEG1;
    stcCanInit.stcBitCfg.u32TimeSeg2  = CAN_BIT_TIME_SEG2;
    stcCanInit.stcBitCfg.u32SJW       = CAN_BIT_TIME_SJW;
    stcCanInit.pstcFilter             = astcFilter;
    stcCanInit.u16FilterSelect        = CAN_FILTER_SEL;
    stcCanInit.u8WorkMode             = CAN_WORK_MD_NORMAL;

    /* Enable peripheral clock of CAN. */
    FCG_Fcg1PeriphClockCmd(CAN_PERIPH_CLK, DISABLE);
    FCG_Fcg1PeriphClockCmd(CAN_PERIPH_CLK, ENABLE);
    (void)CAN_Init(CAN_UNIT, &stcCanInit);
    /* Enable the interrupts, the status flags can be read. */
    CAN_IntCmd(CAN_UNIT, CAN_INT_ALL, DISABLE);
    /* Enable the interrupts that needed. */
    CAN_IntCmd(CAN_UNIT, CAN_INT_SEL, ENABLE);
}
```

3.6.2 定时器的初始化配置

配置 Timer6 单元 2 计时的单位时间 t 为 180 个 CAN nominal bit 时间，溢出后停止，由 CAN_TXD 下降沿产生的外部中断事件再次启动。

```
/* CAN baudrate 500Kbps */
#define CAN_BAUDRATE                    (500000UL)
//.....
/* CAN undefined wave bit count */
#define CAN_UNDEF_WAVE_BIT_CNT          (180U)
```

```
static void CtTmrInit(void)
{
    stc_clock_freq_t stcClockFreq;
    stc_tmr6_init_t stcTmr6Init = {
        .u8CountSrc = TMR6_CNT_SRC_SW,
        .sw_count.u32ClockDiv   = TMR6_CLK_DIV1,
        .sw_count.u32CountMode = TMR6_MD_SAWTOOTH,
        .sw_count.u32CountDir   = TMR6_CNT_UP,
        .u32PeriodValue = 0U,
```

```
        .u32CountReload = TMR6_CNT_RELOAD_OFF,
    };
    (void)CLK_GetClockFreq(&stcClockFreq);

    /* TMR6 clock source is PCLK0 */
    stcTmr6Init.u32PeriodValue = (stcClockFreq.u32Pclk0Freq/CAN_BAUDRATE) * CAN_UNDEF_WAVE_BIT_CNT;
    FCG_Fcg2PeriphClockCmd(CT_TMR_PERIPH_CLK, ENABLE);
    (void)TMR6_Init(CT_TMR, &stcTmr6Init);
    /* Use CAN_TX_PIN_EVT as the start condition */
    TMR6_HWStartCondCmd(CT_TMR, TMR6_START_COND_EVT1, ENABLE);
    TMR6_HWStartCmd(CT_TMR, ENABLE);
}
```

配置 Timer6 单元 5 为硬件向上计数，计数条件为 CAN_TXD 下降沿产生的外部中断事件，配置其功能模式为输入捕获，捕获条件为 Timer6 单元 2 的计数溢出事件；使能其硬件清零功能，清零条件为 Timer6 单元 2 的计数溢出事件。

```
static void CwTmrInit(void)
{
    stc_extint_init_t stcExtIntInit;
    stc_tmr6_init_t stcTmr6Init = {
        .u8CountSrc = TMR6_CNT_SRC_HW,
        /* Use CAN_TX_PIN_EVT as the count up condition */
        .hw_count.u32CountUpCond   = TMR6_CNT_UP_COND_EVT1,
        .hw_count.u32CountDownCond = 0U,
        .u32PeriodValue = 0xFFFFU,
        .u32CountReload = TMR6_CNT_RELOAD_ON,
    };

    /* Enable AOS peripheral clock */
    FCG_Fcg0PeriphClockCmd(PWC_FCG0_AOS, ENABLE);
    /* Enable TMR6 peripheral clock */
    FCG_Fcg2PeriphClockCmd(CW_TMR_PERIPH_CLK, ENABLE);
    (void)TMR6_Init(CW_TMR, &stcTmr6Init);

    TMR6_SetFunc(CW_TMR, TMR6_CH_B, TMR6_PIN_CAPT_INPUT);
    TMR6_HWCaptureCondCmd(CW_TMR, TMR6_CH_B, TMR6_CAPT_COND_EVT0, ENABLE);
    TMR6_HWClearCondCmd(CW_TMR, TMR6_CLR_COND_EVT0, ENABLE);
    TMR6_HWClearCmd(CW_TMR, ENABLE);

    /* TMR6 hardware count condition */
    AOS_SetTriggerEventSrc(AOS_TMR6_1, CAN_TX_PIN_EVT);
    /* Extint config */
    (void)EXTINT_StructInit(&stcExtIntInit);
    (void)EXTINT_Init(CAN_TX_PIN_EXTINT_CH, &stcExtIntInit);
    GPIO_ExIntCmd(CAN_TX_PORT, CAN_TX_PIN, ENABLE);

    /* TMR6 hardware capture and clear condition */
    AOS_SetTriggerEventSrc(AOS_TMR6_0, EVT_SRC_TMR6_2_OVF);
}
```

3.6.3 DMA 的初始化配置

例程使用 DMA 单元 2 的通道 7 来传输 Timer6 单元 5 捕获的计数值。

```
static void DmaInit(void)
{
    stc_dma_init_t stcDmaInit = {
        .u32IntEn          = DMA_INT_DISABLE,
        .u32SrcAddr        = DMA_SRC_ADDR,
        .u32DestAddr       = DMA_DEST_ADDR,
        .u32DataWidth      = DMA_DATA_WIDTH,
        .u32BlockSize      = DMA_BLOCK_SIZE,
        .u32TransCount     = DMA_TRANS_CNT,
        .u32SrcAddrInc     = DMA_SRC_ADDR_FIX,
        .u32DestAddrInc    = DMA_DEST_ADDR_FIX,
    };

    FCG_Fcg0PeriphClockCmd((DMA_PERIPH_CLK | FCG0_PERIPH_AOS), ENABLE);
    (void)DMA_Init(DMA_UNIT, DMA_CH, &stcDmaInit);
    AOS_SetTriggerEventSrc(AOS_DMA2_7, DMA_TRIG_SRC);
    DMA_Cmd(DMA_UNIT, ENABLE);
    (void)DMA_ChCmd(DMA_UNIT, DMA_CH, ENABLE);
}
```

3.6.4 DCU 的初始化配置

例程使用 DCU 单元 1 来比较 Timer6 单元 5 捕获的计数值是否在预设区间内。DCU 比较方式为：DATA2 <= DATA0 <= DATA1。

```
/* CAN undefined wave bit count */
#define CAN_UNDEF_WAVE_BIT_CNT              (180U)

/* Undefied wave width per period is 6 nominal bits */
#define CAN_UNDEF_WAVE_WIDTH                 (6U)

/* Undefied wave range */
#define CAN_UNDEF_WAVE_LOWER_LIMIT          ((CAN_UNDEF_WAVE_BIT_CNT / CAN_UNDEF_WAVE_WIDTH) - 1U)
#define CAN_UNDEF_WAVE_UPPER_LIMIT          ((CAN_UNDEF_WAVE_BIT_CNT / CAN_UNDEF_WAVE_WIDTH) + 1U)
```

```
static void DcuInit(void)
{
    stc_dcu_init_t stcDcuInit = {
        .u32Mode = DCU_MD_CMP,
        .u32DataWidth = DCU_DATA_WIDTH_8BIT,
    };
    stc_irq_signin_config_t stcIrq;

    FCG_Fcg0PeriphClockCmd(DCU_PERIPH_CLK, ENABLE);
    (void)DCU_Init(DCU_UNIT, &stcDcuInit);
    DCU_WriteData8(DCU_UNIT, DCU_DATA1_IDX, CAN_UNDEF_WAVE_UPPER_LIMIT);
    DCU_WriteData8(DCU_UNIT, DCU_DATA2_IDX, CAN_UNDEF_WAVE_LOWER_LIMIT);
    DCU_IntCmd(DCU_UNIT, DCU_CATEGORY_CMP_WIN, DCU_INT_CMP_WIN_INSIDE, ENABLE);
    DCU_GlobalIntCmd(DCU_UNIT, ENABLE);

    stcIrq.enIntSrc    = DCU_INT_SRC;
    stcIrq.enIRQn      = DCU_IRQn;
    stcIrq.pfnCallback = &UndefinedWaveDetectCallback;
    (void)INTC_IrqSignIn(&stcIrq);
    NVIC_ClearPendingIRQ(DCU_IRQn);
    NVIC_SetPriority(DCU_IRQn, DCU_IRQ_PRIO);
    NVIC_EnableIRQ(DCU_IRQn);
}
```

3.6.5 启动未定义波形检测

启动未定义波形的检测时，需要对各变量清零，对定时器的计数值进行清零等操作。

```
static void UndefinedWaveDetectStart(void)
{
    m_u8TxFlag = 0U;
    m_u8UndefWaveCount = 0U;
    m_u8TxFailFlag = 0U;
    TMR6_SetCountValue(CW_TMR, 0U);
    TMR6_SetCountValue(CT_TMR, 0U);
    TMR6_Start(CW_TMR);
    TMR6_HWStartCondCmd(CT_TMR, TMR6_START_COND_EVT1, ENABLE);
    TMR6_HWStartCmd(CT_TMR, ENABLE);
}
```

3.6.6 停止未定义波形检测

停止未定义波形的检测时，需要停止定时器，且禁止相关定时器的硬件启动条件。

```
static void UndefinedWaveDetectStop(void)
{
    m_u8UndefWaveCount = 0U;
    TMR6_Stop(CW_TMR);
    TMR6_Stop(CT_TMR);
    TMR6_HWStartCondCmd(CT_TMR, TMR6_START_COND_EVT1, DISABLE);
    TMR6_HWStartCmd(CT_TMR, DISABLE);
}
```

3.6.7 CAN 发送

标志位 m_u8TxFlag 置位表示 CAN 发送成功，其置位后可继续新的发送；标志位 m_u8TxFailFlag 置位表示 CAN 发送失败，当前帧未能成功发送，需要重新填充并发送。

```
static void CanTx(void)
{
    stc_can_tx_frame_t stcTx = {
        .u32ID     = CAN_TX_ID,
        .u32Ctrl = 0x0UL,
        .DLC        = CAN_TX_DLC,
        .IDE        = CAN_TX_IDE,
        .au8Data = {0U, 1U, 2U, 3U, 4U, 5U, 6U, 7U},
    };

    /* Start undefined wave detecting */
 UndefinedWaveDetectStart();
 /* Transmit one frame via PTB */
 (void)CAN_FillTxFrame(CAN_UNIT, CAN_TX_BUF_PTB, &stcTx);
    /* CAN Tx request */
 CAN_StartTx(CAN_UNIT, CAN_TX_REQ_PTB);
    /* Wait transmission complete */
    while (1U) {
        if (m_u8TxFlag != 0U) {
            /* Transmission completed. */
            break;
        }

        if (m_u8TxFailFlag != 0U) {
      /* Transmission fails, refill the Tx buffer and retransmit are needed. */
            break;
        }
    }
}
```

3.6.8 中断函数

例程用到了 CAN 中断和 DCU 中断。

CAN 中断使能了发送中断，在发送中断产生时，停止检测未定义波形，同时置位发送成功标志位 m_u8TxFlag。

```
static void CAN_IrqCallback(void)
{
    if (CAN_GetStatus(CAN_UNIT, CAN_FLAG_PTB_TX) == SET) {
        CAN_ClearStatus(CAN_UNIT, CAN_FLAG_PTB_TX);
        UndefinedWaveDetectStop();
        m_u8TxFlag = 1U;
    }

    if (CAN_GetStatus(CAN_UNIT, CAN_FLAG_RX) == SET) {
        CAN_ClearStatus(CAN_UNIT, CAN_FLAG_RX);
    }
}
```

DCU 中断产生，则表示 Timer6 单元 5 在 180 个 CAN 位时间内计数的 CAN_TXD 下降沿个数 n 满足比较条件 29<=n<=31，表示可能检测到未定义波形，若累计检测到 5 次，则重新初始化 CAN 控制器，当前帧标记为发送失败（m_u8TxFailFlag = 1U），需要重新填充并发送。

```
/* Undefined wave detected count */
#define CAN_UNDEF_WAVE_DETECTED_CNT        (5)
```

```
void UndefinedWaveDetectCallback(void)
{
    m_u8UndefWaveCount++;
    if (m_u8UndefWaveCount >= CAN_UNDEF_WAVE_DETECTED_CNT) {
        UndefinedWaveDetectStop();
        CanInitConfig();
        m_u8TxFailFlag = 1U;
        DDL_Printf("Undefined wave detected, and the last frame not successfully transmitted.\r\n");
    }
    /* Clear the flags. DATA2 <= DATA0 <= DATA1. DCU_UNIT->FLAGCLR = DCU_UNIT->FLAG */
    DCU_ClearStatus(DCU_UNIT, DCU_FLAG_DATA0_EQ_DATA2 | DCU_FLAG_DATA0_GT_DATA2 | \
                              DCU_FLAG_DATA0_LT_DATA1 | DCU_FLAG_DATA0_EQ_DATA1);
}
```

4 总结

本文档主要介绍了一种用定时器、DMA 和 DCU 来检测 CAN 控制器未定义波形的方法，该方法能较及时地检测出未定义波形。该方法在 CAN_TXD 发送启动时，定时器开始计时、开始计数下降沿的个数，即使 CAN_TXD 停止输出，当计时时间到达后，定时器也会将计数的 CAN_TXD 输出的下降沿个数传输至 DCU 进行比较，因此可能存在一定概率的误判，使用时请注意。

版本修订记录

版本号	修订日期	修订内容
Rev1.0	2023/03/24	初版发布。

若您在购买与使用过程中有任何意见或建议，请随时与我们联系。

邮箱：support@xhsc.com.cn

电话：021-68667000-7355

地址：上海市浦东新区中科路 1867 号 A 座 10 层

XIAOHUA SEMICONDUCTOR CO., LTD