Application Note

SPC11x8/SPD11x8 SIO_CAN 使用指南

Revision 1.4 – September 2019

目录

表格列表

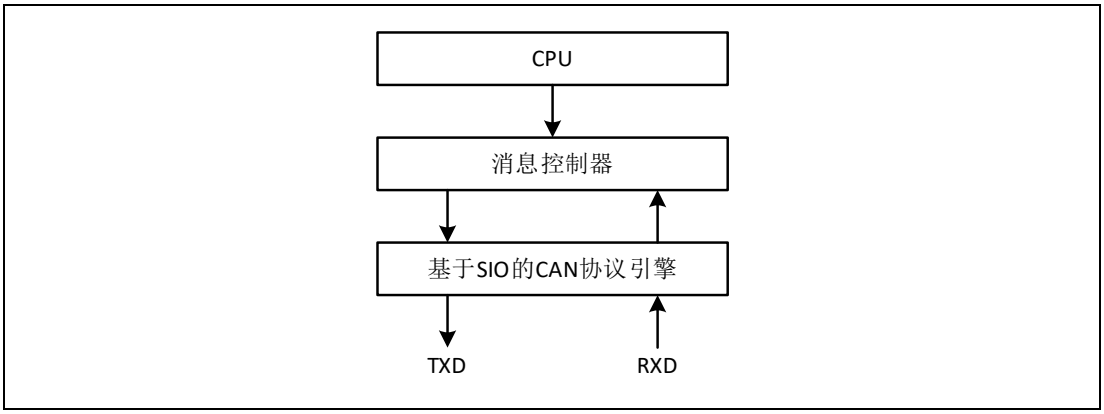
图片列表

1 基于 SIO 的控制器局域网（CAN）模块

1.1 系统概述

如图 1-1 所示，控制器局域网（CAN）系统主要由消息控制器和 CAN 协议引擎组成。

图 1-1: 基于 SIO 的 CAN 系统架构



在发送时，将 CAN 消息写入消息控制器，然后由 CAN 协议引擎发送，发送完成后，触发数据发送完成中断。CAN 协议引擎负责监控总线状态，发送消息到总线，以及检测应答。

在接收时，CAN 协议引擎在被正确配置后开始监控总线上的数据接收，当接收到数据 CRC 校验正确，发送 ACK 信号。接收完整的帧后，根据掩码和过滤器设置，接收符合过滤要求的帧并触发接收到数据中断。

由于 SIO 实现的 CAN 系统只具有一个接收信箱和一个发送信箱，因此需要软件实现信箱 FIFO 功能。

TXD 和 RXD 分别为发送通道输出信号和接收通道输入信号。

注意: 需要外接 CAN 收发器完成 TXD/RXD 逻辑电平到 CAN 总线差分电平的互相转换。

1.2 特性

基于 SIO 所实现的 CAN 系统具备如下特性：

- 支持 CAN2.0B 协议标准；
- 支持标准帧和扩展帧；
- 支持自动总线恢复；
- 支持自动重发；
- 最大数据率 500Kbps；
- 支持 1 个发送信箱和 1 个接收信箱；
- 支持掩码和过滤，4 组掩码和过滤器；
- 支持数据接收和数据发送完毕中断。

1.3 消息对象以及软件 FIFO 定义

1.3.1 接收消息对象

每个接收消息对象定义如下：

接收消息定义

```
typedef struct
{
  uint32_t ID  : 29; /*!< 帧 ID。
                          标准帧取值范围：[0:0x7FF];
                          扩展帧取值范围：[0:0x1FFFFFFF] */
  uint8_t  IDE : 1 ; /*!< 扩展帧使能。数据类型为：CAN_IDEnumDef */
  uint8_t  RTR : 1 ; /*!< 远程帧使能。数据类型为：CAN_RTREnumDef */
  uint8_t  DLC : 4 ; /*!< 数据长度。取值范围：[0:4]。 */
  uint8_t  DATA[8] ; /*!< 数据。 */
  uint32_t FilterMatchIndex; /*!< 匹配的过滤器索引。取值范围[0:3]。 */
} CAN_RxFrameTypeDef;
```

1.3.2 发送消息对象

每个发送消息对象定义如下：

发送消息定义

```
typedef struct
{
    uint32_t ID  : 29; /*!< 帧 ID。
                            标准帧取值范围：[0:0x7FF];
                            扩展帧取值范围：[0:0x1FFFFFFF] */
    uint8_t  IDE : 1 ; /*!< 扩展帧使能。数据类型为：CAN_IDEnumDef */
    uint8_t  RTR : 1 ; /*!< 远程帧使能。数据类型为：CAN_RTREnumDef */
    uint8_t  DLC : 4 ; /*!< 数据长度。取值范围：[0:4]。 */
    uint8_t  DATA[8] ; /*!< 数据。 */
} CAN_TxFrameTypeDef
```

1.3.3 接收 FIFO 数据类型定义

每个接收消息对象定义如下：

接收消息定义

```
typedef struct
{
  CAN_RxFrameTypeDef * FRAME ;  /*!< 接收帧缓存指针 */
  uint16_t  DEPTH         ;     /*!< FIFO 深度。 */
  uint16_t  MASK          ;     /*!< 掩码。*/
  uint16_t  BEGIN         ;     /*!< FIFO 头位置。 */
  uint16_t  END           ;     /*!< FIFO 尾位置。 */
  uint16_t  LEVEL         ;     /*!< 当前 FIFO 深度。 */
  uint16_t  OVERFLOW      ;     /*!< FIFO 溢出。 */
} CAN_RxFIFOTypeDef;
```

1.3.4 发送 FIFO 数据类型定义

每个发送消息对象定义如下：

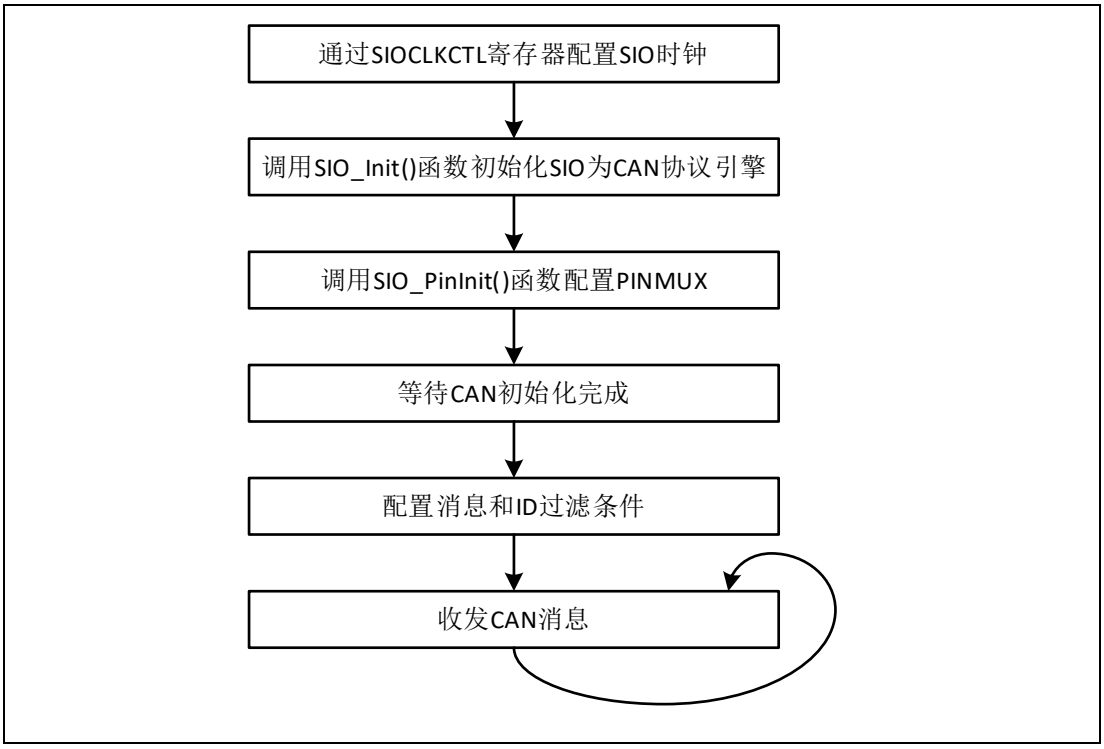
发送消息定义

```
typedef struct
{
  CAN_TxFrameTypeDef * FRAME ;  /*!< 发送帧缓存指针 */
  uint16_t  DEPTH         ;     /*!< FIFO 深度。 */
  uint16_t  MASK          ;     /*!< 掩码。*/
  uint16_t  BEGIN         ;     /*!< FIFO 头位置。 */
  uint16_t  END           ;     /*!< FIFO 尾位置。 */
  uint16_t  LEVEL         ;     /*!< 当前 FIFO 深度。 */
  uint16_t  OVERFLOW      ;     /*!< FIFO 溢出。 */
} CAN_RxFIFOTypeDef;
```

2 操作方式

图 2-1 给出了基于 SIO 的 CAN 系统在使用时的具体操作流程。Spintrol 提供了相应的软件库来简化该系统的使用。

图 2-1: 基于 SIO 的 CAN 系统操作流程



2.1 配置 SIO 时钟

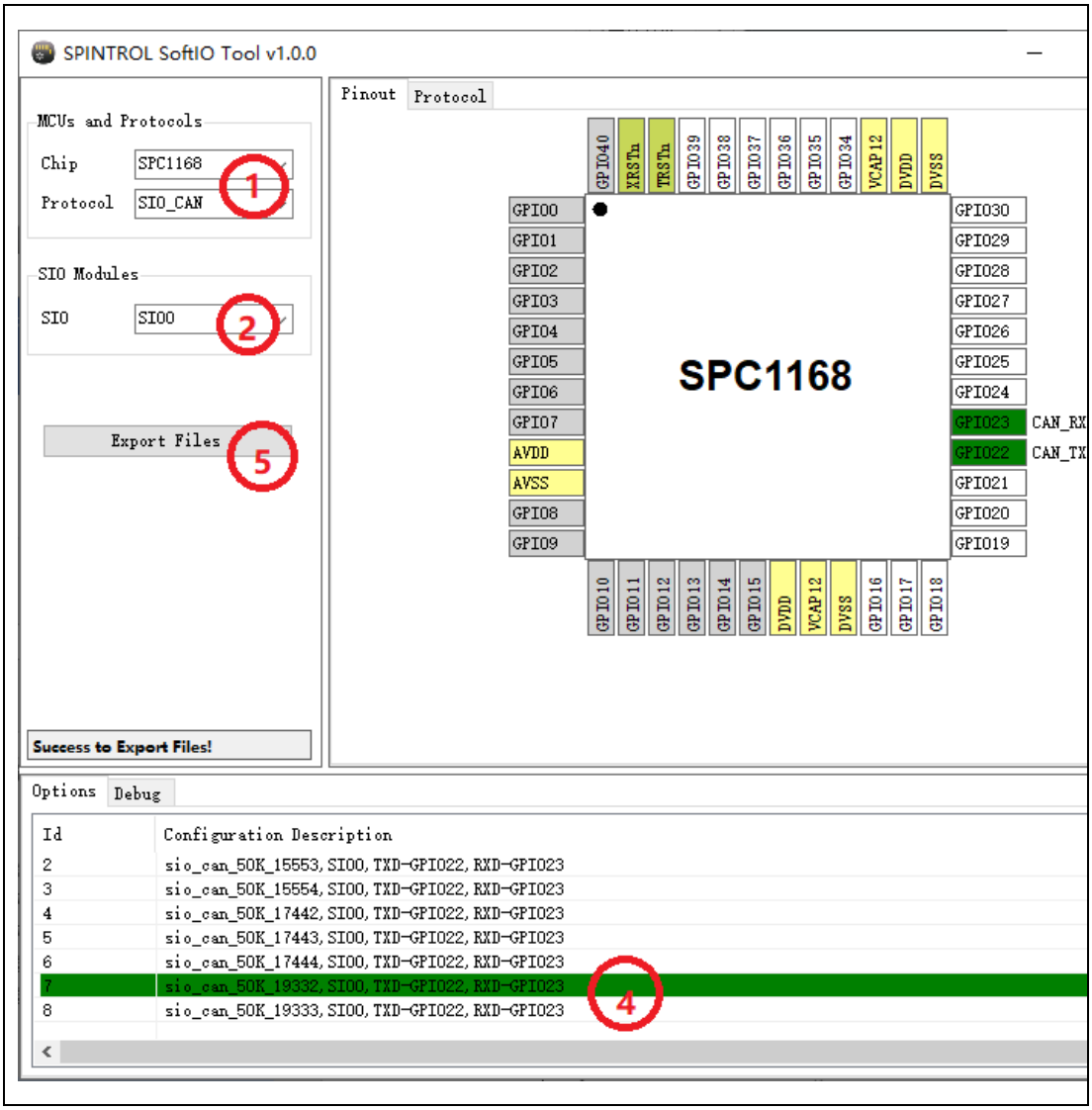
用户可以通过 SIOCLKCTL 寄存器来配置 SIO 时钟，包括时钟的使能和分频比。具体可参见《SPC11X8/SPD11X8 Technical Reference Manual》的第 3 章。当 SIO 被配置用作 CAN 协议引擎时，所允许的 SIO 时钟最高频率为 100MHz。

2.2 初始化 SIO 为 CAN 协议引擎

SPC11X8/SPD11X8 SDK 中提供了 SIO_CAN_Init()函数，下载固件到 SIO 模块，将其初始化成 CAN 协议引擎。用户只需要在代码中直接调用该函数即可。SIO_CAN_Init()同时配置了 PINMUX，将引脚切换至 SIO 的通道，可参见《SPC11X8/SPD11X8 Technical Reference Manual》的第 4.3 章节和第 4.4 章节。SIO_CAN_Init()最后还等待 SIO_CAN 初始化完成。

注意，SIO_CAN 不支持波特率和时序的动态调整。配置 CAN 波特率和时序需要使用配置工具，如图 2-2。可参考《SIO 配置工具使用指南》。

图 2-2:　基于 SIO 的 CAN 配置工具使用



2.3 配置消息

SPC11X8/SPD11X8 SDK 中提供了一系列函数来配置发送和接收的消息。详见 API 函数列表。

2.4 中断处理

当 CAN 引擎完成一个消息的收发后，会向 CPU 发送一个 SIO0A 中断请求。CPU 在拿到 TX 数据以后，会产生一个软件中断。用户可以参考代码示例，在中断服务代码中添加消息队列的处理。

3 API 函数

表 3-1: API 函数列表

函数名称	说明
void SIO_PinInit(void);	初始化 SIO 管脚
void SIO_Init(void);	初始化 SIO 为 SIO_CAN
ErrorStatus CAN_TxFIFOPush(CAN_TxFIFOTypeDef * pCanTxFIFO, CAN_TxFrameTypeDef * pCanTxFrame) ;	TXFIFO 入列。 pCanTxFIFO：TXFIFO 地址指针。 pCanTxFrame：发送帧数据指针。 TXFIFO 满则返回 ERROR，否则返回 SUCCESS。
ErrorStatus CAN_TxFIFOPop(CAN_TxFIFOTypeDef * pCanTxFIFO, CAN_TxFrameTypeDef * pCanTxFrame) ;	TXFIFO 出列。 pCanTxFIFO：TXFIFO 地址指针。 pCanTxFrame：发送帧数据指针。 TXFIFO 空则返回 ERROR，否则返回 SUCCESS。
Void CAN_TxFIFOClear(CAN_TxFIFOTypeDef * pCanTxFIFO) ;	TXFIFO 清空。 pCanTxFIFO：TXFIFO 地址指针。
ErrorStatus CAN_RxFIFOPush(CAN_RxFIFOTypeDef * pCanRxFIFO, CAN_RxFrameTypeDef * pCanRxFrame) ;	RXFIFO 入列。 pCanRxFIFO：RXFIFO 地址指针。 pCanRxFrame：接收帧数据指针。 RXFIFO 满则返回 ERROR，否则返回 SUCCESS。
ErrorStatus CAN_RxFIFOPop(CAN_RxFIFOTypeDef * pCanRxFIFO, CAN_RxFrameTypeDef * pCanRxFrame) ;	RXFIFO 出列。 pCanRxFIFO：RXFIFO 地址指针。 pCanRxFrame：接收帧数据指针。 RXFIFO 空则返回 ERROR，否则返回 SUCCESS。
Void CAN_RxFIFOClear(CAN_RxFIFOTypeDef * pCanRxFIFO) ;	RXFIFO 清空。 pCanRxFIFO：RXFIFO 地址指针。
Void SIO_CAN_Init(void) ;	SIO_CAN 初始化并等待 CAN 初始化完成。
Void SIO_CAN_SetMaskFilter(uint32_t u32Index, CAN_IDEnumDef eFrameType, uint32_t u32FilterRTR, uint32_t u32Mask, uint32_t u32Filter	SIO_CAN 设置掩码和过滤器。 u32Index：取值范围 0 到 3。 eFrameType：选择过滤标准/扩展帧。 u32FilterRTR：选择远程/数据帧过滤。 u32Mask：ID 掩码。 u32Filter：ID 过滤码。 注意：收取的 ID 需要满足（u32Mask & ID）

) ;	== （u32Mask & u32Filter）)
Void SIO_CAN_DisableMaskFilter(uint32_t u32Index) ;	停用过滤功能。 u32Index：取值范围 0 到 3。
Void SIO_CAN_SendMessage(CAN_TxFrameTypeDef * pCanTxFrame) ;	发送 CAN 消息。 pCanTxFrame：发送帧数据指针。
ErrorStatus SIO_CAN_GetMessage(CAN_RxFrameTypeDef * pCanRxFrame) ;	读取 CAN 消息。 pCanRxFrame：接收帧数据指针。 读取接收溢出寄存器置位，则返回 ERROR，否则返回 SUCCESS。
ErrorStatus SIO_CAN_PushTxMessage(CAN_TxFIFOTypeDef * pCanTxFIFO, CAN_TxFrameTypeDef * pCanTxFrame) ;	发送消息入列。 pCanTxFIFO：TXFIFO 地址指针。 pCanTxFrame：发送帧数据指针。 TXFIFO 满则返回 ERROR，否则返回 SUCCESS。
ErrorStatus SIO_CAN_PopTxMessage(CAN_TxFIFOTypeDef * pCanTxFIFO) ;	发送消息出列。 pCanTxFIFO：TXFIFO 地址指针。 pCanTxFrame：发送帧数据指针。 TXFIFO 空则返回 ERROR，否则返回 SUCCESS。
ErrorStatus SIO_CAN_PushRxMessage(CAN_RxFIFOTypeDef * pCanRxFIFO, CAN_RxFrameTypeDef * pCanRxFrame) ;	读取消息入列。 pCanRxFIFO：RXFIFO 地址指针。 pCanRxFrame：接收帧数据指针。 RXFIFO 满则返回 ERROR，否则返回 SUCCESS。
ErrorStatus SIO_CAN_PopRxMessage(CAN_RxFIFOTypeDef * pCanRxFIFO, CAN_RxFrameTypeDef * pCanRxFrame) ;	读取消息出列。 pCanRxFIFO：RXFIFO 地址指针。 pCanRxFrame：接收帧数据指针。 RXFIFO 空则返回 ERROR，否则返回 SUCCESS。
void SIO_CAN_IRQHandler(void) ;	SIO_CAN 函数。
void SIO_CAN_RXCompleteCallback(void) ;	SIO_CAN 接收完成回调函数。
void SIO_CAN_TXCompleteCallback(void) ;	SIO_CAN 发送完成回调函数。
SIO_CAN_IsReady()	SIO_CAN 初始化完成。 注意，SIO_CAN 启动以及总线自动恢复后需要等待初始化完成。
SIO_CAN_IsRxDone()	SIO_CAN 有数据接收到。
SIO_CAN_ClearRxDone()	清除数据接收到状态位。
SIO_CAN_IsBusOff()	SIO_CAN 掉线。
SIO_CAN_IsRxOverflow()	SIO_CAN 接收数据溢出，该位置位，已经接收到

	的数据会被破坏。
SIO_CAN_ClearRxOverflow()	清除接收数据溢出状态位。
SIO_CAN_GetRXDLC()	读取接收的数据帧中的 DLC。
SIO_CAN_GetRXFilterIndex()	读取接收的数据帧匹配的过滤索引。
SIO_CAN_IsTxBusy()	SIO_CAN 发送数据忙标志位。
SIO_CAN_IsPassive()	SIO_CAN 被动节点状态位。
SIO_CAN_GetTEC()	读取 TEC。
SIO_CAN_GetREC()	读取 REC。

4 代码示例

该例子中，SIO_CAN 发送数据帧，然后接收数据帧。若接收到的数据帧是远程帧，则发送相同 ID 的数据帧。

4.1 全局变量定义及部分应用函数定义

示例代码 4-1：全局变量定义及部分应用函数定义

```
#include "spc1168.h"
#include "sio_can0.h"
#include <stdio.h>
#include <stdlib.h>

#define CAN_BASIC_TX_TEST
#define CAN_BASIC_RX_TEST

#define CAN_NODE_NUM                    (1)

#define PRINT(fmt, args...)             printf(fmt, ##args)
#define PRINT_INFO(fmt, args...)        printf("[Info ][%2d][%s] "
fmt, CAN_NODE_NUM, __FUNCTION__, ##args)
#define PRINT_ERR(fmt, args...)         printf("[Error][%2d][%s][%d]
" fmt, CAN_NODE_NUM, __FUNCTION__, __LINE__, ##args)

ErrorStatus gIsPass = SUCCESS;

int i32RxFrameCnt = 0;
int i32TxFrameCnt = 0;

int i32PushTFIFOCnt = 0;
int i32PushTFIFOFailedCnt = 0;

int i32RxErrorCnt = 0;
int i32TxErrorCnt = 0;

int i32RxOverloadCnt = 0;

FunctionalState EnableFileter   = DISABLE;
uint32_t gu32EnableSendInISE    = DISABLE;
uint32_t gu32EnableReceiveInISE = DISABLE;

void print_can_txframe(CAN_TxFrameTypeDef *CanTxFrame)
{
```

```
    int i;
    uint8_t u8DLen = CanTxFrame->RTR ? 0 : ( CanTxFrame->DLC > 8 ) ?
8 : CanTxFrame->DLC ;

    if( CanTxFrame->IDE )
    {
        PRINT("TB0: %d, %d'h%0*X, %d, %d\t", 1, 29, 8, CanTxFrame-
>ID, CanTxFrame->RTR, CanTxFrame->DLC);
    }
    else
    {
        PRINT("TB0: %d,     %d'h%0*X, %d, %d\t", 0, 11, 3,
CanTxFrame->ID, CanTxFrame->RTR, CanTxFrame->DLC);
    }
    for( i = 0; i < u8DLen; i++ )
    {
        PRINT("| 0x%02X ", CanTxFrame->DATA[i]);
    }
    PRINT("\n\n");
}

void print_can_rxframe(CAN_RxFrameTypeDef *CanRxFrame)
{
    int i;
    uint8_t u8DLen = CanRxFrame->RTR ? 0 : ( CanRxFrame->DLC > 8 ) ?
8 : CanRxFrame->DLC ;

    if( CanRxFrame->IDE )
    {
        PRINT("RB0-%d: %d, %d'h%0*X, %d, %d\t", CanRxFrame-
>FilterMatchIndex, 1, 29, 8, CanRxFrame->ID, CanRxFrame->RTR,
CanRxFrame->DLC);
    }
    else
    {
        PRINT("RB0-%d: %d,     %d'h%0*X, %d, %d\t", CanRxFrame-
>FilterMatchIndex, 0, 11, 3, CanRxFrame->ID, CanRxFrame->RTR,
CanRxFrame->DLC);
    }
    for( i = 0; i < u8DLen; i++ )
    {
        PRINT("| 0x%02X ", CanRxFrame->DATA[i]);
    }
    PRINT("\n");
```

```
}

void send_one_frame(void)
{
    /* Send one message from TXFIFO if TXFIFO not empty */
    if( SUCCESS == SIO_CAN0_PopTxMessage(&CanTxFIFO) )
    {
        i32TxFrameCnt++;
    }

    i32TxErrorCnt += SIO_CAN0_GetTEC();
}
```

4.2 初始化 SIO_CAN

示例代码 4-2：初始化 SIO_CAN

```
ErrorStatus isPass = SUCCESS;
int i;
int test_FIFO_depth = CAN_TXFIFO_DEPTH;
int test_cnt = CAN_TXFIFO_DEPTH;

uint8_t  u8DLen = 0;

CAN_TxFrameTypeDef CanTxFrame;
CAN_RxFrameTypeDef CanRxFrame;

/* Config Flash Timing for 200 MHz */
FLASH_WALLOW();
FLASH_SetTiming(200000000);
FLASH_WDIS();

/* Clock Init */
CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

/* Delay Init */
Delay_Init();

/* UART Init */
PINMUX->GPIO34.bit.MUXSEL = GPIO34_BIT_MUXSEL_UART_TXD;
PINMUX->GPIO35.bit.MUXSEL = GPIO35_BIT_MUXSEL_UART_RXD;
UART_Init(UART, 256000);

/* Configure SIO Clock, maximum clock is 100MHz */
CLOCK_SetModuleDiv(SIO0_MODULE, 2);
CLOCK_EnableModule(SIO0_MODULE);

PRINT("SYSCLK0CTL.bit.SRC: %d\n", CLOCK->SYSCLK0CTL.bit.SRC );
PRINT("SYSCLK1CTL.bit.SRC: %d\n", CLOCK->SYSCLK1CTL.bit.SRC );
PRINT("SIO Clock: %d\n\n", CLOCK_GetModuleClock(SIO0_MODULE));

/* SIO_CAN0 Init */
SIO_CAN0_Init();
PRINT_INFO("SIO Init Finished.\n\n");

/* SIO_CAN0 RX Filter Init */
// EnableFileter = ENABLE;
if( ENABLE == EnableFileter )
```

```
{
    // matched id: ( mask & id ) ^~ ( mask & filter )
    SIO_CAN0_SetMaskFilter(0, CAN_ID_STD, CAN_RTR_FILTER_DATA   ,
0x3FF, 0x0004) ;
    SIO_CAN0_SetMaskFilter(1, CAN_ID_STD, CAN_RTR_FILTER_REMOTE ,
0x3FF, 0x0001) ;

    SIO_CAN0_SetMaskFilter(2, CAN_ID_EXT, CAN_RTR_FILTER_DATA   ,
0x1FFFFFFF, 0x0002) ;
    SIO_CAN0_SetMaskFilter(3, CAN_ID_EXT, CAN_RTR_FILTER_REMOTE ,
0x1FFFFFFF, 0x0000) ;
}
else
{
    SIO_CAN0_DisableMaskFilter(0) ;
    SIO_CAN0_DisableMaskFilter(1) ;
    SIO_CAN0_DisableMaskFilter(2) ;
    SIO_CAN0_DisableMaskFilter(3) ;
}

/* Enable SIO0A Interrupt */
NVIC_EnableIRQ(SIO0A_IRQn);
```

4.3 发送数据测试

示例代码 4-3：发送数据测试

```
gu32EnableSendInISE    = ENABLE;
gu32EnableReceiveInISE = DISABLE;

test_FIFO_depth = CAN_TXFIFO_DEPTH;
test_cnt = CAN_TXFIFO_DEPTH;

i32PushTFIFOCnt = 0;

while(test_cnt)
{
    CanTxFrame.IDE = rand() & 1;
    CanTxFrame.ID  = CanTxFrame.IDE ? rand() & 0x1FFFFFFF : rand() &
0x7FF;
    CanTxFrame.RTR = rand() & 1;
    CanTxFrame.DLC = rand() % 9;

    u8DLen = ( CanTxFrame.DLC > 8 ) ? 8 : CanTxFrame.DLC ;

    for( i = 0; i < u8DLen; i++ )
    {
        CanTxFrame.DATA[i] = rand();
    }

    if(1) print_can_txframe(&CanTxFrame);

    if( CanTxFIFO.LEVEL < test_FIFO_depth )
    {
        if( SUCCESS == SIO_CAN0_PushTxMessage(&CanTxFIFO,
&CanTxFrame ) )
        {
            i32PushTFIFOCnt++;
        }
        else
        {
            i32PushTFIFOFailedCnt++;
        }
    }
    else
    {
        // Send First Frame
        send_one_frame();
```

```
        /* Wait All Frame Send Done */
        while( CanTxFIFO.LEVEL )
        {
            if( ( SIO_CAN0_GetTEC() > 0 ) || ( SIO_CAN0_GetREC() >
0 ) )
            {
                isPass = ERROR;
            }

            if( SIO_CAN0_IsBusOff() )
            {
                PRINT("\n[Fatal]Bus off, wait bus off recover and exit
transmit test\n\t\t\t*********\n");
                isPass = ERROR;

                while( SIO_CAN0_IsBusOff() ) ;

                goto RX_TEST;
            }
            else
            {
                PRINT("FIFO LV: %d, %d\n\n", CanTxFIFO.LEVEL,
CanRxFIFO.LEVEL);
            }
        }

        while( SIO_CAN0_IsTxBusy() );
        PRINT("\nLoop %d\n\n", test_cnt);

        if( --test_cnt )
        {
            if( SUCCESS == SIO_CAN0_PushTxMessage(&CanTxFIFO,
&CanTxFrame ) )
            {
                i32PushTFIFOCnt++;
            }
            else
            {
                i32PushTFIFOFailedCnt++;
            }
        }
    }
```

```
    PRINT("FIFO LV: %d, %d\n\n", CanTxFIFO.LEVEL, CanRxFIFO.LEVEL);
}


RX_TEST:
if(isPass==ERROR) { PRINT("^_^ failed\n\n"); }
else              { PRINT("p_q success\n\n"); }

PRINT("TEC/REC: %3d, %3d; %d, %d\n", SIO_CAN0_GetTEC(),
SIO_CAN0_GetREC(), i32TxErrorCnt, i32RxErrorCnt);
PRINT("FIFO LV: %d, %d\n", CanTxFIFO.LEVEL, CanRxFIFO.LEVEL);
PRINT(">> %5d, %5d, %5d, (%5d, %5d)\n\n", i32TxFrameCnt,
i32RxFrameCnt, i32RxFrameCnt + i32TxFrameCnt, i32PushTFIFOCnt,
i32PushTFIFOFailedCnt);
PRINT("\n\n\n");

gu32EnableSendInISE    = DISABLE;
gu32EnableReceiveInISE = DISABLE;
```

4.4 收发数据测试

示例代码 4-4：收发数据测试

```
gu32EnableSendInISE    = DISABLE;
gu32EnableReceiveInISE = ENABLE ;

test_cnt = 25;

while(test_cnt)
{
    if( ( SIO_CAN0_GetTEC() > 0 ) || ( SIO_CAN0_GetREC() > 0 ) )
    {
        isPass = ERROR;
    }

    if( SIO_CAN0_IsBusOff() )
    {
        PRINT("\n[Fatal]Bus off, wait bus off recover and exit
transmit test\n\t\t\t*********\n");
        isPass = ERROR;

        while( SIO_CAN0_IsBusOff() ) ;

        goto END_OF_TEST;
```

```
    }

    /* Get RX message from RXFIFO */
    if( SUCCESS == SIO_CAN0_PopRxMessage(&CanRxFIFO, &CanRxFrame) )
    {
        // test_cnt--;

        if(1)
        {
            print_can_rxframe(&CanRxFrame);
            PRINT("TEC/REC: %3d, %3d; %d, %d\n", SIO_CAN0_GetTEC(),
SIO_CAN0_GetREC(), i32TxErrorCnt, i32RxErrorCnt);
            PRINT("FIFO LV: %d, %d\n", CanTxFIFO.LEVEL,
CanRxFIFO.LEVEL);
            PRINT(">> %5d, %5d, %5d, (%5d, %5d)\n\n", i32TxFrameCnt,
i32RxFrameCnt, i32RxFrameCnt + i32TxFrameCnt, i32PushTFIFOCnt,
i32PushTFIFOFailedCnt);
        }

        // Remote Frame Response
        if( CanRxFrame.RTR == CAN_RTR_REMOTE )
        {
            CanTxFrame = *(CAN_TxFrameTypeDef *) & CanRxFrame;

            u8DLen = ( CanTxFrame.DLC > 8 ) ? 8 : CanTxFrame.DLC ;

            for( i = 0; i < u8DLen; i++ )
            {
                CanTxFrame.DATA[i] = rand();
            }

            CanTxFrame.RTR = CAN_RTR_DATA ;

            if(1) print_can_txframe(&CanTxFrame);

            /* Add new message to TXFIFO */
            if( SUCCESS == SIO_CAN0_PushTxMessage(&CanTxFIFO,
&CanTxFrame ) )
            {
                i32PushTFIFOCnt++;
            }
            else
            {
                i32PushTFIFOFailedCnt++;
```

```
            }
        }
    }
    if( CanTxFIFO.LEVEL )
    {
        /* Send one message from TXFIFO if TXFIFO not empty */
        if( SUCCESS == SIO_CAN0_PopTxMessage(&CanTxFIFO) )
        {
            i32TxFrameCnt++;
        }
        i32TxErrorCnt += SIO_CAN0_GetTEC();
    }

    if( i32RxOverloadCnt )
    {
        PRINT("[Error] Receiver Overload Occured, exit test\n");
        isPass = ERROR;
        goto END_OF_TEST;
    }
}

END_OF_TEST:
while( CanRxFIFO.LEVEL )
{
    /* Get RX message from RXFIFO */
    SIO_CAN0_PopRxMessage(&CanRxFIFO, &CanRxFrame);
    if(1)
    {
        print_can_rxframe(&CanRxFrame);
        PRINT("TEC/REC: %3d, %3d; %d, %d\n", SIO_CAN0_GetTEC(),
SIO_CAN0_GetREC(), i32TxErrorCnt, i32RxErrorCnt);
        PRINT("FIFO LV: %d, %d\n", CanTxFIFO.LEVEL, CanRxFIFO.LEVEL);
        PRINT(">> %5d, %5d, %5d, (%5d, %5d)\n\n", i32TxFrameCnt,
i32RxFrameCnt, i32RxFrameCnt + i32TxFrameCnt, i32PushTFIFOCnt,
i32PushTFIFOFailedCnt);
    }
}
while( CanTxFIFO.LEVEL )
{
    if( SIO_CAN0_IsTxBusy() )
    {
    }
    else
    {
```

```
        /* Send one message from TXFIFO if TXFIFO not empty */
        if( SUCCESS == SIO_CAN0_PopTxMessage(&CanTxFIFO) )
        {
            i32TxFrameCnt++;
        }
        i32TxErrorCnt += SIO_CAN0_GetTEC();
    }
}

if(isPass==ERROR) { PRINT("^_^ failed\n\n"); }
else              { PRINT("p_q success\n\n"); }

PRINT("TEC/REC: %3d, %3d; %d, %d\n", SIO_CAN0_GetTEC(),
SIO_CAN0_GetREC(), i32TxErrorCnt, i32RxErrorCnt);
PRINT("FIFO LV: %d, %d\n", CanTxFIFO.LEVEL, CanRxFIFO.LEVEL);
PRINT(">> %5d, %5d, %5d, (%5d, %5d)\n\n", i32TxFrameCnt,
i32RxFrameCnt, i32RxFrameCnt + i32TxFrameCnt, i32PushTFIFOCnt,
i32PushTFIFOFailedCnt);
PRINT("\n\n");

gu32EnableSendInISE    = DISABLE;
gu32EnableReceiveInISE = DISABLE;
```

4.5 中断函数

示例代码 4-5：中断函数

```
void SIO_CAN0_IRQHandler(void)
{
    if ( SIO_CAN0_IsRxDone() )
    {
        SIO_CAN0_RXCompleteCallback() ;
    }

    if ( ! SIO_CAN0_IsTxBusy() )
    {
        SIO_CAN0_TXCompleteCallback() ;
    }
}
```

4.6 接收消息队列处理

示例代码 4-6：接收消息队列处理

```
void SIO_CAN0_RXCompleteCallback(void) // 被 SIO_CAN0_IRQHandler 调用
{
    CAN_RxFrameTypeDef CanRxFrame;

    if( gu32EnableReceiveInISE )
    {
        /* Get message from SIO */
        if( ERROR == SIO_CAN0_GetMessage(&CanRxFrame) )
        {
            // Receive frame corrupted
            i32RxOverloadCnt++;
            PRINT("\n[Fatal] Receive frame
corrupted!\n\t\t\t*********\n");
        }
        else
        {
            /* Add new message to RXFIFO */
            if( ERROR == SIO_CAN0_PushRxMessage(&CanRxFIFO,
&CanRxFrame) )
            {
                // RFIFO is full
                PRINT("\n[Fatal] RFIFO FUll!\n\t\t\t*********\n");
            }
            else
            {
                i32RxFrameCnt++;
            }
            i32RxErrorCnt += SIO_CAN0_GetREC();
        }
    }
}
```

4.7 发送消息队列处理

示例代码 4-7：发送消息队列处理

```
void SIO_CAN0_TXCompleteCallback(void) // 被 SIO_CAN0_IRQHandler 调用
{
    if( gu32EnableSendInISE )
    {
        /* Send one message from TXFIFO if TXFIFO not empty */
        if( SUCCESS == SIO_CAN0_PopTxMessage(&CanTxFIFO) )
        {
            i32TxFrameCnt++;
        }
        i32TxErrorCnt += SIO_CAN0_GetTEC();
    }
}
```

5 寄存器

5.1 SIO_CAN 寄存器表

表 5-1: SIO_CAN 模块基地址

外设模块	基地址
SIO_CAN	0x4000 B000

表 5-2: SIO_CAN 模块寄存器表

寄存器	偏移地址	说明	默认值
STS0	0x0	Status Register 0	0x00000000
IDF0L	0x8	ID Filer 0 Low 16 bit Register	0x00000000
TXD0	0xC	Transmit Data Register 0	0x00000000
TXD1	0x10	Transmit Data Register 1	0x00000000
TXD2	0x14	Transmit Data Register 2	0x00000000
TXD3	0x18	Transmit Data Register 3	0x00000000
TXEID0	0x1C	Transmit Extended ID Register 0	0x00000000
TXEID1	0x20	Transmit Extended ID Register 1	0x00000000
TXSID	0x20	Transmit Standard ID Register	0x00000000
CTL	0x24	Control Register	0x00000000
IDF0H	0x28	ID Filer 0 High 16 bit Register	0x00000000
IDM0L	0x2C	ID Mask 0 Low 16 bit Register	0x00000000
IDM0H	0x30	ID Mask 0 High 16 bit Register	0x00000000
IDF1L	0x34	ID Filer 1 Low 16 bit Register	0x00000000
IDF1H	0x38	ID Filer 1 High 16 bit Register	0x00000000
IDM1L	0x3C	ID Mask 1 Low 16 bit Register	0x00000000
IDM1H	0x40	ID Mask 1 High 16 bit Register	0x00000000
RXD0	0x44	Receive Data Register 0	0x00000000
RXD1	0x48	Receive Data Register 1	0x00000000
RXD2	0x4C	Receive Data Register 2	0x00000000
RXD3	0x50	Receive Data Register 3	0x00000000
RXEID0	0x54	Receive Extended ID Register 0	0x00000000
RXEID1	0x58	Receive Extended ID Register 1	0x00000000
RXSID	0x58	Receive Standard ID Register	0x00000000
IDF2L	0x5C	ID Filer 2 Low 16 bit Register	0x00000000
IDF2H	0x60	ID Filer 2 High 16 bit Register	0x00000000
IDM2L	0x64	ID Mask 2 Low 16 bit Register	0x00000000
IDM2H	0x68	ID Mask 2 High 16 bit Register	0x00000000
IDF3L	0x6C	ID Filer 3 Low 16 bit Register	0x00000000
IDF3H	0x70	ID Filer 3 High 16 bit Register	0x00000000
IDM3L	0x74	ID Mask 3 Low 16 bit Register	0x00000000

IDM3H	0x78	ID Mask 3 High 16 bit Register	0x00000000
STS1	0x7C	Status Register 1	0x00002101

5.2 SIO_CAN 寄存器

表 5-3 到表 5-46 列举了 SIO_CAN 寄存器的所有细节。

表 5-3: Status Register 0 (ST0) Layout

STS0 (Status Register 0) Offset: 0x0 Default: 0x00000000 Access: SIO_CAN -> STS0.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
REC							
7	6	5	4	3	2	1	0
PASSIVE	TEC						

表 5-4: Status Register 0 (ST0) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:8	REC	RO	0x0	RX error counter
7	PASSIVE	RO	0x0	Node passive error status 0: CAN is in active mode 1: CAN is in passive mode
6:0	TEC	RO	0x0	TX error counter

表 5-5: ID Filter 0 Low 16 bit Register (IDF0L) Layout

IDF0L (ID Filer 0 Low 16 bit) Offset: 0x8 Default: 0x00000000 Access: SIO_CAN -> IDF0L.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-6: ID Filter 0 Low 16 bit Register (IDF0L) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-7: Transmit Data Register 0 (TXDATA0) Layout

TXD0 (Transmit Data Register 0) Offset: 0xC Default: 0x00000000 Access: SIO_CAN -> TXD0.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
BYTE1							
7	6	5	4	3	2	1	0
BYTE0							

表 5-8: Transmit Data Register 0 (TXDATA0) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:8	BYTE1	RW	0x0	Data byte 1
7:0	BYTE0	RW	0x0	Data byte 0

表 5-9: Transmit Data Register 1 (TXDATA1) Layout

TXD1 (Transmit Data Register 1) Offset: 0x10 Default: 0x00000000 Access: SIO_CAN -> TXD1.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
BYTE3							
7	6	5	4	3	2	1	0
BYTE2							

表 5-10: Transmit Data Register 1 (TXDATA1) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:8	BYTE3	RW	0x0	Data byte 3
7:0	BYTE2	RW	0x0	Data byte 2

表 5-11: Transmit Data Register 2 (TXDATA2) Layout

TXD2 (Transmit Data Register 2) Offset: 0x14 Default: 0x00000000 Access: SIO_CAN -> TXD2.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
BYTE5							
7	6	5	4	3	2	1	0
BYTE4							

表 5-12: Transmit Data Register 2 (TXDATA2) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:8	BYTE5	RW	0x0	Data byte 5
7:0	BYTE4	RW	0x0	Data byte 4

表 5-13: Transmit Data Register 3 (TXDATA3) Layout

TXD3 (Transmit Data Register 3) Offset: 0x18 Default: 0x00000000 Access: SIO_CAN -> TXD3.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
BYTE7							
7	6	5	4	3	2	1	0
BYTE6							

表 5-14: Transmit Data Register 3 (TXDATA3) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:8	BYTE7	RW	0x0	Data byte 7
7:0	BYTE6	RW	0x0	Data byte 6

表 5-15: Transmit Extended ID Register 0 (TXEID0) Layout

TXEID0 (Transmit Extended ID Register 0) Offset: 0x1C Default: 0x00000000 Access: SIO_CAN -> TXEID0.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
ID14T0							
7	6	5	4	3	2	1	0
ID14T0							RTR

表 5-16: Transmit Extended ID Register 0 (TXEID0) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:1	ID14T0	RW	0x0	Transmit extended ID[14:0]
0	RTR	RW	0x0	Transmit extended RTR

表 5-17: Transmit Extended ID Register 1 (TXEID1) Layout

TXEID1 (Transmit Extended ID Register 1) Offset: 0x20 Default: 0x00000000 Access: SIO_CAN -> TXEID1.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
ID28T18							
7	6	5	4	3	2	1	0
ID28T18			SRR	IDE	ID17T15		

表 5-18: Transmit Extended ID Register 1 (TXEID1) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:5	ID28T18	RW	0x0	Transmit extended ID[28:18]
4	SRR	RW	0x0	Transmit extended RTR
3	IDE	RW	0x0	Transmit extended IDE
2:0	ID17T15	RW	0x0	Transmit extended ID[17:15]

表 5-19: Transmit Standard ID Register (TXSID) Layout

TXSID (Transmit Standard ID Register) Offset: 0x20 Default: 0x00000000 Access: SIO_CAN -> TXSID.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
ID							
7	6	5	4	3	2	1	0
ID			RTR	IDE	RESERVED_2_0		

表 5-20: Transmit Standard ID Register (TXSID) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:5	ID	RW	0x0	Transmit standard ID[10:0]
4	RTR	RW	0x0	Transmit standard RTR
3	IDE	RW	0x0	Transmit standard IDE
2:0	RESERVED_2_0	RW	0x0	Reserved.

表 5-21: Control Register (CTL) Layout

CTL (Control Register) Offset: 0x24 Default: 0x00000000 Access: SIO_CAN -> CTL.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
TXRUN	RESERVED_14_4						
7	6	5	4	3	2	1	0
RESERVED_14_4				TXDLC			

表 5-22: Control Register (CTL) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15	TXRUN	RW	0x0	Transmit engine running control and indicator 0: Write 0 is not permitted. Read 0 indicates the engine is idle and ready to take new message. 1: Write 1 will start the engine, which will transmit the message once the bus is available. NOTE: All information on the transmitting message should be completed before set this bit. Read 1 indicates the message is pending for transmitting.
14:4	RESERVED_14_4	RW	0x0	Reserved.
3:0	TXDLC	RW	0x0	Transmit Data length counter

表 5-23: ID Filter 0 High 16 bit Register (IDF0H) Layout

IDF0H (ID Filer 0 High 16 bit) Offset: 0x28 Default: 0x00000000 Access: SIO_CAN -> IDF0H.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-24: ID Filter 0 High 16 bit Register (IDF0H) Filed Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-25: ID Mask 0 Low 16 bit Register (IDM0L) Layout

IDM0L (ID Mask 0 Low 16 bit) Offset: 0x2C Default: 0x00000000 Access: SIO_CAN -> IDM0L.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-26: ID Mask 0 Low 16 bit Register (IDM0L) Filed Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-27: ID Mask 0 High 16 bit Register (IDM0H) Layout

IDM0H (ID Mask 0 High 16 bit) Offset: 0x30 Default: 0x00000000 Access: SIO_CAN -> IDM0H.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-28: ID Mask 0 High 16 bit Register (IDM0H) Filed Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-29: ID Filer 1 Low 16 bit Register (IDF1L) Layout

IDF1L (ID Filer 1 Low 16 bit) Offset: 0x34 Default: 0x00000000 Access: SIO_CAN -> IDF1L.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-30: ID Filer 1 Low 16 bit Register (IDF1L) Filed Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-31: ID Filer 1 High 16 bit Register (IDF1H) Layout

IDF1H (ID Filer 1 High 16 bit) Offset: 0x38 Default: 0x00000000 Access: SIO_CAN -> IDF1H.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-32: ID Filer 1 High 16 bit Register (IDF1H) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-33: ID Mask 1 Low 16 bit Register (IDM1L) Layout

IDM1L (ID Mask 1 Low 16 bit) Offset: 0x3C Default: 0x00000000 Access: SIO_CAN -> IDM1L.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-34: ID Mask 1 Low 16 bit Register (IDM1L) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-35: ID Mask 1 High 16 bit Register (IDM1H) Layout

IDM1H (ID Mask 1 High 16 bit) Offset: 0x40 Default: 0x00000000 Access: SIO_CAN -> IDM1H.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-36: ID Mask 1 High 16 bit Register (IDM1H) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-37: Receive Data Register 0 (RXD0) Layout

RXD0 (Receive Data Register 0) Offset: 0x44 Default: 0x00000000 Access: SIO_CAN -> RXD0.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
BYTE1							
7	6	5	4	3	2	1	0
BYTE0							

表 5-38: Receive Data Register 0 (RXD0) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:8	BYTE1	RW	0x0	Receive byte 1
7:0	BYTE0	RW	0x0	Receive byte 0

表 5-39: Receive Data Register 1 (RXD1) Layout

RXD1 (Receive Data Register 1) Offset: 0x48 Default: 0x00000000 Access: SIO_CAN -> RXD1.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
BYTE3							
7	6	5	4	3	2	1	0
BYTE2							

表 5-40: Receive Data Register 1 (RXD1) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:8	BYTE3	RW	0x0	Receive byte 3
7:0	BYTE2	RW	0x0	Receive byte 2

表 5-41: Receive Data Register 2 (RXD2) Layout

RXD2 (Receive Data Register 2) Offset: 0x4C Default: 0x00000000 Access: SIO_CAN -> RXD2.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
BYTE5							
7	6	5	4	3	2	1	0
BYTE4							

表 5-42: Receive Data Register 2 (RXD2) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:8	BYTE5	RW	0x0	Receive byte 5
7:0	BYTE4	RW	0x0	Receive byte 4

表 5-43: Receive Data Register 3 (RXD3) Layout

RXD3 (Receive Data Register 3) Offset: 0x50 Default: 0x00000000 Access: SIO_CAN -> RXD3.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
BYTE7							
7	6	5	4	3	2	1	0
BYTE6							

表 5-44: Receive Data Register 3 (RXD3) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:8	BYTE7	RW	0x0	Receive byte 7
7:0	BYTE6	RW	0x0	Receive byte 6

表 5-45: Receive Extended ID Register 0 (RXEID0) Layout

RXEID0 (Receive Extended ID Register 0) Offset: 0x54 Default: 0x00000000 Access: SIO_CAN -> RXEID0.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
ID14T0							
7	6	5	4	3	2	1	0
ID14T0							RTR

表 5-46: Receive Extended ID Register 0 (RXEID0) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:1	ID14T0	RW	0x0	Receive extended ID[14:0]
0	RTR	RW	0x0	Receive extended RTR

表 5-47: Receive Extended ID Register 1 (RXEID1) Layout

RXEID1 (Receive Extended ID Register 1) Offset: 0x58 Default: 0x00000000 Access: SIO_CAN -> RXEID1.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
ID28T18							
7	6	5	4	3	2	1	0
ID28T18			SRR	IDE	ID17T15		

表 5-48: Receive Extended ID Register 1 (RXEID1) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:5	ID28T18	RW	0x0	Receive extended ID[28:18]
4	SRR	RW	0x0	Receive extended RTR
3	IDE	RW	0x0	Receive extended IDE
2:0	ID17T15	RW	0x0	Receive extended ID[17:15]

表 5-49: Receive Standard ID Register (RXSID) Layout

RXSID (Receive Standard ID Register) Offset: 0x58 Default: 0x00000000 Access: SIO_CAN -> RXSID.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
ID							
7	6	5	4	3	2	1	0
ID			RTR	IDE	RESERVED_2_0		

表 5-50: Receive Standard ID Register (RXSID) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:5	ID	RW	0x0	Receive standard ID[10:0]
4	RTR	RW	0x0	Receive standard RTR
3	IDE	RW	0x0	Receive extended IDE
2:0	RESERVED_2_0	RO	0x0	Reserved.

表 5-51: ID Filer 2 Low 16 bit Register (IDF2L) Layout

IDF2L (ID Filer 2 Low 16 bit) Offset: 0x5C Default: 0x00000000 Access: SIO_CAN -> IDF2L.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-52: ID Filer 2 Low 16 bit Register (IDF2L) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-53: ID Filer 2 High 16 bit Register (IDF2H) Layout

IDF2H (ID Filer 2 High 16 bit) Offset: 0x60 Default: 0x00000000 Access: SIO_CAN -> IDF2H.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-54: ID Filer 2 High 16 bit Register (IDF2H) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-55: ID Mask 2 Low 16 bit Register (IDM2L) Layout

IDM2L (ID Mask 2 Low 16 bit) Offset: 0x64 Default: 0x00000000 Access: SIO_CAN -> IDM2L.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-56: ID Mask 2 Low 16 bit Register (IDM2L) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-57: ID Mask 2 High 16 bit Register (IDM2H) Layout

IDM2H (ID Mask 2 High 16 bit) Offset: 0x68 Default: 0x00000000 Access: SIO_CAN -> IDM2H.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-58: ID Mask 2 High 16 bit Register (IDM2H) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-59: ID Filer 3 Low 16 bit Register (IDF3L) Layout

IDF3L (ID Filer 3 Low 16 bit) Offset: 0x6C Default: 0x00000000 Access: SIO_CAN -> IDF3L.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-60: ID Filer 3 Low 16 bit Register (IDF3L) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-61: ID Filer 3 High 16 bit Register (IDF3H) Layout

IDF3H (ID Filer 3 High 16 bit) Offset: 0x70 Default: 0x00000000 Access: SIO_CAN -> IDF3H.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-62: ID Filer 3 High 16 bit Register (IDF3H) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-63: ID Mack 3 Low 16 bit Register (IDM3L) Layout

IDM3L (ID Mask 3 Low 16 bit) Offset: 0x74 Default: 0x00000000 Access: SIO_CAN -> IDM3L.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-64: ID Mack 3 Low 16 bit Register (IDM3L) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-65: ID Mack 3 High 16 bit Register (IDM3H) Layout

IDM3H (ID Mask 3 High 16 bit) Offset: 0x78 Default: 0x00000000 Access: SIO_CAN -> IDM3H.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 5-66: ID Mack 3 High 16 bit Register (IDM3H) Field Description

Bits	Field Name	Type	Reset	Description
31:16	RESERVED_31_16	RO	0x0	Reserved.
15:0	VAL	RW	0x0	

表 5-67: Status Register 1 (STS1) Layout

STS1 (Status Register 1) Offset: 0x7C Default: 0x00002101 Access: SIO_CAN -> STS1.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
RXDONE	BUSOFF	RDY	RXOVLD	RESERVED_11_8			
7	6	5	4	3	2	1	0
RXDLC				MFI		RESERVED_1_0	

表 5-68: Status Register 1 (STS1) Field Description

Bits	Field Name	Type	Reset	Description
15	RXDONE	RW	0x0	RX complete receive a message 0: Read 0 indicates the data in the receive registers are not valid. Write 0 will clear the bit. 1: Read 1 indicates the engine has received a complete message. Write 1 is not permitted.
14	BUSOFF	RO	0x0	CAN bus off indicator. Once CAN TEC reaches its limit of 256, the CAN will enter BusOff mode and BusOff indicator will become 1. The CAN will monitor the bus and once it sees 128 occurrences of 11 consecutive 'recessive' bot, the error counter will be reset to 0 and it goes back to active mode and the indicator will become 0. 0: Bus is normal. 1: Bus off status
13	RDY	RO	0x1	CAN initial ready 0: CAN is not ready. 1: CAN is ready.
12	RXOVLD	RO	0x0	Receive new frame when RXDONE is 1 0: RX message box is not corrupted. 1: RX message box is corrupted.
11:8	RESERVED_11_8	RW	0x1	Reserved.
7:4	RXDLC	RO	0x0	Receive data length counter
3:2	MFI	RO	0x0	Matched filter index
1:0	RESERVED_1_0	RW	0x1	Reserved.

6 修订记录

表 6-1: 文档修订记录

日期	版本	修改内容
2019-09-28	1	初始版本
2021-01-04	1.4	修正 SIO 模块最高始终速率为 100MHz