零死角玩转STM32

CAN—通讯实验

淘宝：firestm32.taobao.com

论坛： www.chuxue123.com

扫描进入淘宝店铺

主讲内容

01 CAN协议简介

02 STM32的CAN外设简介

03 CAN控制的相关结构体

04 CAN—通讯实验

参考资料:《零死角玩转STM32》

“CAN—通讯实验”章节

CAN—通讯实验

CAN结构体

从STM32的CAN外设我们了解到它的功能非常多，控制涉及的寄存器也非常丰富，而使用STM32标准库提供的各种结构体及库函数可以简化这些控制过程。跟其它外设一样，STM32标准库提供了CAN初始化结构体及初始化函数来控制CAN的工作方式，提供了收发报文使用的结构体及收发函数，还有配置控制筛选器模式及ID的结构体。

- 初始化结构体：CAN_InitTypeDef
- 发送及接收结构体：CanTxMsg及CanRxMsg
- 筛选器结构体：CAN_FilterInitTypeDef

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM;  /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;   /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;   /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;   /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;   /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;   /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

配置完这些结构体成员后，调用库函数CAN_Init即可把这些参数写入到CAN控制寄存器中，实现CAN的初始化。

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM; /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;  /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;  /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;  /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;  /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;  /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_Prescaler

本成员设置CAN外设的时钟分频，它可控制时间片Tq的时间长度，这里设置的值最终会减1后再写入BRP寄存器位，即前面介绍的Tq计算公式：

$$Tq = (BRP[9:0]+1) \times T_{PCLK}$$

等效于：$Tq = CAN\_Prescaler \times T_{PCLK}$

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM; /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;  /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;  /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;  /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;  /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;  /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_Mode

本成员设置CAN的工作模式，可设置为正常模式(CAN_Mode_Normal)、回环模式(CAN_Mode_LoopBack)、静默模式(CAN_Mode_Silent)以及回环静默模式(CAN_Mode_Silent_LoopBack)。

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM; /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;  /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;  /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;  /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;  /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;  /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_SJW

本成员可以配置SJW的极限长度，即CAN重新同步时单次可增加或缩短的最大长度，它可以被配置为1-4Tq(CAN_SJW_1/2/3/4tq)。

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM; /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;  /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;  /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;  /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;  /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;  /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_BS1

本成员用于设置CAN位时序中的BS1段的长度，它可以被配置为1-16个Tq长度(CAN_BS1_1/2/3…16tq)。

CAN—通讯实验

CAN初始化结构体

```
1  /**
2    * @brief  CAN 初始化结构体
3    */
4  typedef struct {
5      uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
6      uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
7      uint8_t CAN_SJW;           /*配置 SJW 极限值 */
8      uint8_t CAN_BS1;           /*配置 BS1 段长度*/
9      uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM; /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;  /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;  /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;  /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;  /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;  /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_BS2

本成员用于设置CAN位时序中的BS2段的长度，它可以被配置为1-8个Tq长度(CAN_BS2_1/2/3…8tq)。

SYNC_SEG、BS1段及BS2段的长度加起来即一个数据位的长度，即前面介绍的原来计算公式：

$$T_{1bit} = 1Tq + T_{S1} + T_{S2} = 1 + (TS1[3:0] + 1) + (TS2[2:0] + 1)$$

等效于：$T_{1bit} = 1Tq + CAN\_BS1 + CAN\_BS2$

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM; /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;  /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;  /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;  /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;  /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;  /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_TTCM

本成员用于设置是否使用时间触发功能(ENABLE/DISABLE)，时间触发功能在某些CAN标准中会使用到。

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM;  /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;   /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;   /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;   /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;   /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;   /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_ABOM

本成员用于设置是否使用自动离线管理(ENABLE/DISABLE)，使用自动离线管理可以在节点出错离线后适时自动恢复，不需要软件干预。

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM; /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;  /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;  /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;  /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;  /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;  /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_ AWUM

本成员用于设置是否使用自动唤醒功能(ENABLE/DISABLE)，使能自动唤醒功能后它会在监测到总线活动后自动唤醒。

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM; /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;  /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;  /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;  /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;  /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;  /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_NART

本成员用于设置是否使用自动重传功能(ENABLE/DISABLE)，使用自动重传功能时，会一直发送报文直到成功为止。

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM; /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;  /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;  /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;  /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;  /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;  /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_RFLM

本成员用于设置是否使用锁定接收FIFO(ENABLE/DISABLE)，锁定接收FIFO后，若FIFO溢出时会丢弃新数据，否则在FIFO溢出时以新数据覆盖旧数据。

CAN—通讯实验

CAN初始化结构体

```
1 /**
 2   * @brief  CAN 初始化结构体
 3   */
 4 typedef struct {
 5     uint16_t CAN_Prescaler;    /*配置 CAN 外设的时钟分频，可设置为 1-1024*/
 6     uint8_t CAN_Mode;          /*配置 CAN 的工作模式，回环或正常模式*/
 7     uint8_t CAN_SJW;           /*配置 SJW 极限值 */
 8     uint8_t CAN_BS1;           /*配置 BS1 段长度*/
 9     uint8_t CAN_BS2;           /*配置 BS2 段长度 */
10     FunctionalState CAN_TTCM;  /*是否使能 TTCM 时间触发功能*/
11     FunctionalState CAN_ABOM;   /*是否使能 ABOM 自动离线管理功能*/
12     FunctionalState CAN_AWUM;   /*是否使能 AWUM 自动唤醒功能 */
13     FunctionalState CAN_NART;   /*是否使能 NART 自动重传功能*/
14     FunctionalState CAN_RFLM;   /*是否使能 RFLM 锁定 FIFO 功能*/
15     FunctionalState CAN_TXFP;   /*配置 TXFP 报文优先级的判定方法*/
16 } CAN_InitTypeDef;
```

- CAN_TXFP

本成员用于设置发送报文的优先级判定方法(ENABLE/DISABLE)，使能时，以报文存入发送邮箱的先后顺序来发送，否则按照报文ID的优先级来发送。

CAN—通讯实验

CAN发送及接收结构体

```
1  /**
2    * @brief  CAN Tx message structure definition
3    * 发送结构体
4    */
5  typedef struct {
6      uint32_t StdId;   /*存储报文的标准标识符 11 位，0-0x7FF. */
7      uint32_t ExtId;   /*存储报文的扩展标识符 29 位，0-0x1FFFFFFF. */
8      uint8_t IDE;      /*存储 IDE 扩展标志 */
9      uint8_t RTR;      /*存储 RTR 远程帧标志*/
10     uint8_t DLC;      /*存储报文数据段的长度，0-8 */
11     uint8_t Data[8]; /*存储报文数据段的内容 */
12 } CanTxMsg;
13
14 /**
15   * @brief  CAN Rx message structure definition
16   * 接收结构体
17   */
18 typedef struct {
19     uint32_t StdId;   /*存储了报文的标准标识符 11 位，0-0x7FF. */
20     uint32_t ExtId;   /*存储了报文的扩展标识符 29 位，0-0x1FFFFFFF. */
21     uint8_t IDE;      /*存储了 IDE 扩展标志 */
22     uint8_t RTR;      /*存储了 RTR 远程帧标志*/
23     uint8_t DLC;      /*存储了报文数据段的长度，0-8 */
24     uint8_t Data[8]; /*存储了报文数据段的内容 */
25     uint8_t FMI;      /*存储了 本报文是由经过筛选器存储进 FIFO 的，0-0xFF */
26 } CanRxMsg;
```

在发送或接收报文时，需要往发送邮箱中写入报文信息或从接收FIFO中读取报文信息，利用STM32标准库的发送及接收结构体可以方便地完成这样的工作。

CAN—通讯实验

CAN发送及接收结构体

```
1  /**
2    * @brief  CAN Tx message structure definition
3    * 发送结构体
4    */
5  typedef struct {
6      uint32_t StdId;   /*存储报文的标准标识符 11 位，0-0x7FF. */
7      uint32_t ExtId;   /*存储报文的扩展标识符 29 位，0-0x1FFFFFFF. */
8      uint8_t IDE;      /*存储 IDE 扩展标志 */
9      uint8_t RTR;      /*存储 RTR 远程帧标志*/
10     uint8_t DLC;      /*存储报文数据段的长度，0-8 */
11     uint8_t Data[8]; /*存储报文数据段的内容 */
12 } CanTxMsg;
```

- StdId

本成员存储的是报文的11位标准标识符，范围是0-0x7FF。

- ExtId

本成员存储的是报文的29位扩展标识符，范围是0-0x1FFFFFFF。ExtId与StdId这两个成员根据下面的IDE位配置，只有一个是有效的。

CAN—通讯实验

CAN发送及接收结构体

```
1  /**
2    * @brief  CAN Tx message structure definition
3    * 发送结构体
4    */
5  typedef struct {
6      uint32_t StdId;   /*存储报文的标准标识符 11 位，0-0x7FF. */
7      uint32_t ExtId;   /*存储报文的扩展标识符 29 位，0-0x1FFFFFFF. */
8      uint8_t IDE;      /*存储 IDE 扩展标志 */
9      uint8_t RTR;      /*存储 RTR 远程帧标志*/
10     uint8_t DLC;      /*存储报文数据段的长度，0-8 */
11     uint8_t Data[8]; /*存储报文数据段的内容 */
12 } CanTxMsg;
```

- IDE

本成员存储的是扩展标志IDE位，当它的值为宏CAN_ID_STD时表示本报文是标准帧，使用StdId成员存储报文ID；当它的值为宏CAN_ID_EXT时表示本报文是扩展帧，使用ExtId成员存储报文ID。

CAN—通讯实验

CAN发送及接收结构体

```
1  /**
2    * @brief  CAN Tx message structure definition
3    * 发送结构体
4    */
5  typedef struct {
6      uint32_t StdId;   /*存储报文的标准标识符 11 位，0-0x7FF. */
7      uint32_t ExtId;   /*存储报文的扩展标识符 29 位，0-0x1FFFFFFF. */
8      uint8_t IDE;      /*存储 IDE 扩展标志 */
9      uint8_t RTR;      /*存储 RTR 远程帧标志*/
10     uint8_t DLC;      /*存储报文数据段的长度，0-8 */
11     uint8_t Data[8]; /*存储报文数据段的内容 */
12 } CanTxMsg;
```

- RTR

本成员存储的是报文类型标志RTR位，当它的值为宏CAN_RTR_Data时表示本报文是数据帧；当它的值为宏CAN_RTR_Remote时表示本报文是遥控帧，由于遥控帧没有数据段，所以当报文是遥控帧时，下面的Data[8]成员的内容是无效的。

CAN—通讯实验

CAN发送及接收结构体

```
1  /**
2    * @brief  CAN Tx message structure definition
3    * 发送结构体
4    */
5  typedef struct {
6      uint32_t StdId;   /*存储报文的标准标识符 11 位，0-0x7FF. */
7      uint32_t ExtId;   /*存储报文的扩展标识符 29 位，0-0x1FFFFFFF. */
8      uint8_t IDE;      /*存储 IDE 扩展标志 */
9      uint8_t RTR;      /*存储 RTR 远程帧标志*/
10     uint8_t DLC;      /*存储报文数据段的长度，0-8 */
11     uint8_t Data[8]; /*存储报文数据段的内容 */
12 } CanTxMsg;
```

- DLC

本成员存储的是数据帧数据段的长度，它的值的范围是0-8，当报文是遥控帧时DLC值为0。

Data[8]

本成员存储的就是数据帧中数据段的数据。

CAN—通讯实验

CAN发送及接收结构体

```
14 /**
15   * @brief  CAN Rx message structure definition
16   * 接收结构体
17   */
18 typedef struct {
19     uint32_t StdId;   /*存储了报文的标准标识符 11 位，0-0x7FF. */
20     uint32_t ExtId;   /*存储了报文的扩展标识符 29 位，0-0x1FFFFFFF. */
21     uint8_t IDE;      /*存储了 IDE 扩展标志 */
22     uint8_t RTR;      /*存储了 RTR 远程帧标志*/
23     uint8_t DLC;      /*存储了报文数据段的长度，0-8 */
24     uint8_t Data[8]; /*存储了报文数据段的内容 */
25     uint8_t FMI;      /*存储了 本报文是由经过筛选器存储进 FIFO 的，0-0xFF */
26 } CanRxMsg;
```

- FMI

本成员只存在于接收结构体，它存储了筛选器的编号，表示本报文是经过哪个筛选器存储进接收FIFO的，可以用它简化软件处理。

CAN—通讯实验

CAN发送及接收结构体

当需要使用CAN发送报文时，先定义一个上面发送类型的结构体，然后把报文的内容按成员赋值到该结构体中，最后调用库函数CAN_Transmit把这些内容写入到发送邮箱即可把报文发送出去。

接收报文时，通过检测标志位获知接收FIFO的状态，若收到报文，可调用库函数CAN_Receive把接收FIFO中的内容读取到预先定义的接收类型结构体中，然后再访问该结构体即可利用报文。

CAN—通讯实验

CAN筛选器结构体

```
1 /**
 2   * @brief  CAN filter init structure definition
 3   * CAN 筛选器结构体
 4   */
 5 typedef struct {
 6     uint16_t CAN_FilterIdHigh;          /*CAN_FxR1 寄存器的高 16 位 */
 7     uint16_t CAN_FilterIdLow;           /*CAN_FxR1 寄存器的低 16 位*/
 8     uint16_t CAN_FilterMaskIdHigh;      /*CAN_FxR2 寄存器的高 16 位*/
 9     uint16_t CAN_FilterMaskIdLow;       /*CAN_FxR2 寄存器的低 16 位 */
10     uint16_t CAN_FilterFIFOAssignment; /*设置经过筛选后数据存储到哪个接收 FIFO
*/
11     uint8_t CAN_FilterNumber;           /*筛选器编号，范围 0-27*/
12     uint8_t CAN_FilterMode;             /*筛选器模式 */
13     uint8_t CAN_FilterScale;            /*设置筛选器的尺度 */
14     FunctionalState CAN_FilterActivation; /*是否使能本筛选器*/
15 } CAN_FilterInitTypeDef;
```

- CAN_FilterIdHigh

CAN_FilterIdHigh成员用于存储要筛选的ID，若筛选器工作在32位模式，它存储的是所筛选ID的高16位；若筛选器工作在16位模式，它存储的就是一个完整的要筛选的ID。

CAN—通讯实验

CAN筛选器结构体

```
1 /**
 2   * @brief  CAN filter init structure definition
 3   * CAN 筛选器结构体
 4   */
 5 typedef struct {
 6     uint16_t CAN_FilterIdHigh;          /*CAN_FxR1 寄存器的高 16 位 */
 7     uint16_t CAN_FilterIdLow;           /*CAN_FxR1 寄存器的低 16 位*/
 8     uint16_t CAN_FilterMaskIdHigh;      /*CAN_FxR2 寄存器的高 16 位*/
 9     uint16_t CAN_FilterMaskIdLow;       /*CAN_FxR2 寄存器的低 16 位 */
10     uint16_t CAN_FilterFIFOAssignment; /*设置经过筛选后数据存储到哪个接收 FIFO
*/
11     uint8_t CAN_FilterNumber;           /*筛选器编号，范围 0-27*/
12     uint8_t CAN_FilterMode;             /*筛选器模式 */
13     uint8_t CAN_FilterScale;            /*设置筛选器的尺度 */
14     FunctionalState CAN_FilterActivation; /*是否使能本筛选器*/
15 } CAN_FilterInitTypeDef;
```

- CAN_FilterIdLow

类似地，CAN_FilterIdLow成员也是用于存储要筛选的ID，若筛选器工作在32位模式，它存储的是所筛选ID的低16位；若筛选器工作在16位模式，它存储的就是一个完整的要筛选的ID。

CAN—通讯实验

CAN筛选器结构体

```
1 /**
 2   * @brief  CAN filter init structure definition
 3   * CAN 筛选器结构体
 4   */
 5 typedef struct {
 6     uint16_t CAN_FilterIdHigh;          /*CAN_FxR1 寄存器的高 16 位 */
 7     uint16_t CAN_FilterIdLow;           /*CAN_FxR1 寄存器的低 16 位*/
 8     uint16_t CAN_FilterMaskIdHigh;      /*CAN_FxR2 寄存器的高 16 位*/
 9     uint16_t CAN_FilterMaskIdLow;       /*CAN_FxR2 寄存器的低 16 位 */
10     uint16_t CAN_FilterFIFOAssignment; /*设置经过筛选后数据存储到哪个接收 FIFO
*/
11     uint8_t CAN_FilterNumber;           /*筛选器编号，范围 0-27*/
12     uint8_t CAN_FilterMode;             /*筛选器模式 */
13     uint8_t CAN_FilterScale;            /*设置筛选器的尺度 */
14     FunctionalState CAN_FilterActivation; /*是否使能本筛选器*/
15 } CAN_FilterInitTypeDef;
```

- CAN_FilterMaskIdHigh

CAN_FilterMaskIdHigh存储的内容分两种情况，当筛选器工作在标识符列表模式时，它的功能与CAN_FilterIdHigh相同，都是存储要筛选的ID；而当筛选器工作在掩码模式时，它存储的是CAN_FilterIdHigh成员对应的掩码，与CAN_FilterIdLow组成一组筛选器。

CAN—通讯实验

CAN筛选器结构体

```
1 /**
 2   * @brief  CAN filter init structure definition
 3   * CAN 筛选器结构体
 4   */
 5 typedef struct {
 6     uint16_t CAN_FilterIdHigh;          /*CAN_FxR1 寄存器的高 16 位 */
 7     uint16_t CAN_FilterIdLow;           /*CAN_FxR1 寄存器的低 16 位*/
 8     uint16_t CAN_FilterMaskIdHigh;      /*CAN_FxR2 寄存器的高 16 位*/
 9     uint16_t CAN_FilterMaskIdLow;       /*CAN_FxR2 寄存器的低 16 位 */
10     uint16_t CAN_FilterFIFOAssignment; /*设置经过筛选后数据存储到哪个接收 FIFO
*/
11     uint8_t CAN_FilterNumber;           /*筛选器编号，范围 0-27*/
12     uint8_t CAN_FilterMode;             /*筛选器模式 */
13     uint8_t CAN_FilterScale;            /*设置筛选器的尺度 */
14     FunctionalState CAN_FilterActivation; /*是否使能本筛选器*/
15 } CAN_FilterInitTypeDef;
```

- CAN_FilterMaskIdLow

类似地，CAN_FilterMaskIdLow存储的内容也分两种情况，当筛选器工作在标识符列表模式时，它的功能与CAN_FilterIdLow相同，都是存储要筛选的ID；而当筛选器工作在掩码模式时，它存储的是CAN_FilterIdLow成员对应的掩码，与CAN_FilterIdLow组成一组筛选器。

CAN—通讯实验

CAN筛选器结构体

不同模式下各结构体成员的内容：

模式	CAN_FilterIdHigh	CAN_FilterIdLow	CAN_FilterMaskIdHigh	CAN_FilterMaskIdLow
32位列表模式	ID1的高16位	ID1的低16位	ID2的高16位	ID2的低16位
16位列表模式	ID1的完整数值	ID2的完整数值	ID3的完整数值	ID4的完整数值
32位掩码模式	ID1的高16位	ID1的低16位	ID1掩码的高16位	ID1掩码的低16位
16位掩码模式	ID1的完整数值	ID2的完整数值	ID1掩码的完整数值	ID2掩码完整数值

CAN—通讯实验

CAN筛选器结构体

```
1 /**
 2   * @brief  CAN filter init structure definition
 3   * CAN 筛选器结构体
 4   */
 5 typedef struct {
 6     uint16_t CAN_FilterIdHigh;          /*CAN_FxR1 寄存器的高 16 位 */
 7     uint16_t CAN_FilterIdLow;           /*CAN_FxR1 寄存器的低 16 位*/
 8     uint16_t CAN_FilterMaskIdHigh;      /*CAN_FxR2 寄存器的高 16 位*/
 9     uint16_t CAN_FilterMaskIdLow;       /*CAN_FxR2 寄存器的低 16 位 */
10     uint16_t CAN_FilterFIFOAssignment; /*设置经过筛选后数据存储到哪个接收 FIFO
*/
11     uint8_t CAN_FilterNumber;           /*筛选器编号，范围 0-27*/
12     uint8_t CAN_FilterMode;             /*筛选器模式 */
13     uint8_t CAN_FilterScale;            /*设置筛选器的尺度 */
14     FunctionalState CAN_FilterActivation; /*是否使能本筛选器*/
15 } CAN_FilterInitTypeDef;
```

- CAN_FilterFIFOAssignment

本成员用于设置当报文通过筛选器的匹配后，该报文会被存储到哪一个接收FIFO，它的可选值为FIFO0或FIFO1(宏CAN_Filter_FIFO0/1)。

CAN—通讯实验

CAN筛选器结构体

```
1 /**
 2   * @brief  CAN filter init structure definition
 3   * CAN 筛选器结构体
 4   */
 5 typedef struct {
 6     uint16_t CAN_FilterIdHigh;          /*CAN_FxR1 寄存器的高 16 位 */
 7     uint16_t CAN_FilterIdLow;           /*CAN_FxR1 寄存器的低 16 位*/
 8     uint16_t CAN_FilterMaskIdHigh;      /*CAN_FxR2 寄存器的高 16 位*/
 9     uint16_t CAN_FilterMaskIdLow;       /*CAN_FxR2 寄存器的低 16 位 */
10     uint16_t CAN_FilterFIFOAssignment; /*设置经过筛选后数据存储到哪个接收 FIFO
*/
11     uint8_t CAN_FilterNumber;           /*筛选器编号，范围 0-27*/
12     uint8_t CAN_FilterMode;             /*筛选器模式 */
13     uint8_t CAN_FilterScale;            /*设置筛选器的尺度 */
14     FunctionalState CAN_FilterActivation; /*是否使能本筛选器*/
15 } CAN_FilterInitTypeDef;
```

- CAN_FilterNumber

本成员用于设置筛选器的编号，即本过滤器结构体配置的是哪一组筛选器，CAN一共有28个筛选器，所以它的可输入参数范围为0-27。

CAN—通讯实验

CAN筛选器结构体

```
1 /**
 2   * @brief  CAN filter init structure definition
 3   * CAN 筛选器结构体
 4   */
 5 typedef struct {
 6     uint16_t CAN_FilterIdHigh;          /*CAN_FxR1 寄存器的高 16 位 */
 7     uint16_t CAN_FilterIdLow;           /*CAN_FxR1 寄存器的低 16 位*/
 8     uint16_t CAN_FilterMaskIdHigh;      /*CAN_FxR2 寄存器的高 16 位*/
 9     uint16_t CAN_FilterMaskIdLow;       /*CAN_FxR2 寄存器的低 16 位 */
10     uint16_t CAN_FilterFIFOAssignment; /*设置经过筛选后数据存储到哪个接收 FIFO
*/
11     uint8_t CAN_FilterNumber;           /*筛选器编号，范围 0-27*/
12     uint8_t CAN_FilterMode;             /*筛选器模式 */
13     uint8_t CAN_FilterScale;            /*设置筛选器的尺度 */
14     FunctionalState CAN_FilterActivation; /*是否使能本筛选器*/
15 } CAN_FilterInitTypeDef;
```

- CAN_FilterMode

本成员用于设置筛选器的工作模式，可以设置为列表模式(宏CAN_FilterMode_IdList)及掩码模式(宏CAN_FilterMode_IdMask)。

CAN—通讯实验

CAN筛选器结构体

```
1 /**
 2   * @brief  CAN filter init structure definition
 3   * CAN 筛选器结构体
 4   */
 5 typedef struct {
 6     uint16_t CAN_FilterIdHigh;          /*CAN_FxR1 寄存器的高 16 位 */
 7     uint16_t CAN_FilterIdLow;           /*CAN_FxR1 寄存器的低 16 位*/
 8     uint16_t CAN_FilterMaskIdHigh;      /*CAN_FxR2 寄存器的高 16 位*/
 9     uint16_t CAN_FilterMaskIdLow;       /*CAN_FxR2 寄存器的低 16 位 */
10     uint16_t CAN_FilterFIFOAssignment; /*设置经过筛选后数据存储到哪个接收 FIFO
*/
11     uint8_t CAN_FilterNumber;           /*筛选器编号，范围 0-27*/
12     uint8_t CAN_FilterMode;             /*筛选器模式 */
13     uint8_t CAN_FilterScale;            /*设置筛选器的尺度 */
14     FunctionalState CAN_FilterActivation; /*是否使能本筛选器*/
15 } CAN_FilterInitTypeDef;
```

- CAN_FilterScale

本成员用于设置筛选器的尺度，可以设置为32位长(宏CAN_FilterScale_32bit)及16位长(宏CAN_FilterScale_16bit)。

CAN—通讯实验

CAN筛选器结构体

```
1 /**
 2   * @brief  CAN filter init structure definition
 3   * CAN 筛选器结构体
 4   */
 5 typedef struct {
 6     uint16_t CAN_FilterIdHigh;          /*CAN_FxR1 寄存器的高 16 位 */
 7     uint16_t CAN_FilterIdLow;           /*CAN_FxR1 寄存器的低 16 位*/
 8     uint16_t CAN_FilterMaskIdHigh;      /*CAN_FxR2 寄存器的高 16 位*/
 9     uint16_t CAN_FilterMaskIdLow;       /*CAN_FxR2 寄存器的低 16 位 */
10     uint16_t CAN_FilterFIFOAssignment; /*设置经过筛选后数据存储到哪个接收 FIFO
*/
11     uint8_t CAN_FilterNumber;           /*筛选器编号，范围 0-27*/
12     uint8_t CAN_FilterMode;             /*筛选器模式 */
13     uint8_t CAN_FilterScale;            /*设置筛选器的尺度 */
14     FunctionalState CAN_FilterActivation; /*是否使能本筛选器*/
15 } CAN_FilterInitTypeDef;
```

- CAN_FilterActivation

本成员用于设置是否激活这个筛选器(宏ENABLE/DISABLE)。

THANKS

论坛： www.chuxue123.com

淘宝：firestm32.taobao.com

扫描进入淘宝店铺