CAN寄存器操作

部分客户为了适配之前的软件结构，或是想要简化代码复杂度，需要采取纯寄存器操作。其中，由于RAM的寄存器定义在手册中并未体现，因此操作难点主要体现在RAM配置。

这里以HA01为对象做一个简单的示例。

1. 型号及版本

EVB: HA01 EVB

SDK: 1_4_0

YCT: 2_7_8

2. 配置

a. 控制器初始化

初始化主要包括以下几个方面：

i. 选择CAN时钟源，在CTRL1.CLKSRC进行设置，设置时钟源时CAN需处于DISABLE状态；

ii. 通过MCR.MDIS使能CAN模块；

iii. 设置RXMGMASK、RX14MASK、RX15MASK和RXFGMASK，默认初始化为全F。以上四个寄存器只有在freeze模式下才配置生效（如果需要使用独立掩码，即MCR.IRMQ=1，则RXIMR也需要初始化为全F）；

iv. 初始化RAM，RAM寄存器数量为邮箱数*4，即如果有32个邮箱，则RAM[0]~RAM[127]需要被初始化为0，具体看使用邮箱数量。如果使用了enhance fifo，那么enhance fifo的RAM所在区域都需要初始化；

v. 设置波特率。通常情况下，可以在CTRL1中设置PRESDIV、PROPSEG、PSEG1、PSEG2和RJW，但当CTRL2.BTE=1或者FD模式下CTRL1中的配置失效，此时需要在CBT或者ENCBT中进行波特率配置，寄存器中的值实际计算时需要加1。

其中，同步跳转宽度设置规则为≤min（Phase_Seg1，4），如果太大会使采样点提前，进而影响通信时序；

vi. 通过IMASKn开启对应邮箱的中断。

ps：如果使用了FD、FIFO、Enhance FIFO或DMA等功能时，还需要单独进行相应设置。

b. RAM设置

上面有提到，一个MB占用4个RAM寄存器空间（Normal 模式下），那么以MB1为例，必须进行的步骤如下：

i. 设置ID。

MB1所在区域为RAM[4]~RAM[7]，因此ID需要在RAM[5]中进行设置，若ID为0x32，那么配置如下：

代码块

```
base->RAM[5] &= ~CAN_ID_STD_MASK;
base->RAM[5] |= (frame.msgId << CAN_ID_STD_SHIFT) & CAN_ID_STD_MASK;
```

ii. 写入数据。

一个单位RAM寄存器为32bit，最多可以存储4byte数据，因此MB1的RAM[6]、RAM[7]都是用来存储数据的，配置如下：

代码块

```
base->RAM[6] = (((uint32_t) frame.data[0]) << 24) \
                                  | (((uint32_t) frame.data[1]) << 16) \
                                  | (((uint32_t) frame.data[2]) << 8) \
                                  | frame.data[3];
base->RAM[7] = (((uint32_t) frame.data[4]) << 24) \
                                  | (((uint32_t) frame.data[5]) << 16) \
                                  | (((uint32_t) frame.data[6]) << 8) \
                                  | frame.data[7];
```

iii. 设置邮箱状态/激活邮箱

若数据长度为8，那么配置如下：

代码块

```
base->RAM[0] &= ~(CAN_CS_IDE_MASK | CAN_CS_SRR_MASK);
base->RAM[0] |= ((uint32_t)8 << CAN_CS_DLC_SHIFT) & CAN_CS_DLC_MASK;
base->RAM[0] |= (FLEXCAN_TX_DATA << CAN_CS_CODE_SHIFT) &
CAN_CS_CODE_MASK;
```

c. 清除中断标志。在封装的send函数中，或者每次调用send之前，必须通过IFLAGx清除对应邮箱中断标志。

Flexcan_Demo_Register_Option.zip
2.23MB