第 48 章 CAN（FlexCAN）

FlexCAN 模块是一个通信控制器，该模块实现了 CAN 协议即 CAN2.0B 协议规范。下图为一个常用的框图，介绍了 FlexCAN 模块的的字模块，包括了用来存储消息缓冲的相关联的内存区域，Rx 全局掩码寄存器、Rx 私有掩码寄存器、Rx 先进先出队列以及 Rx 队列标识过滤器。各个子模块的功能将在随后的章节介绍。

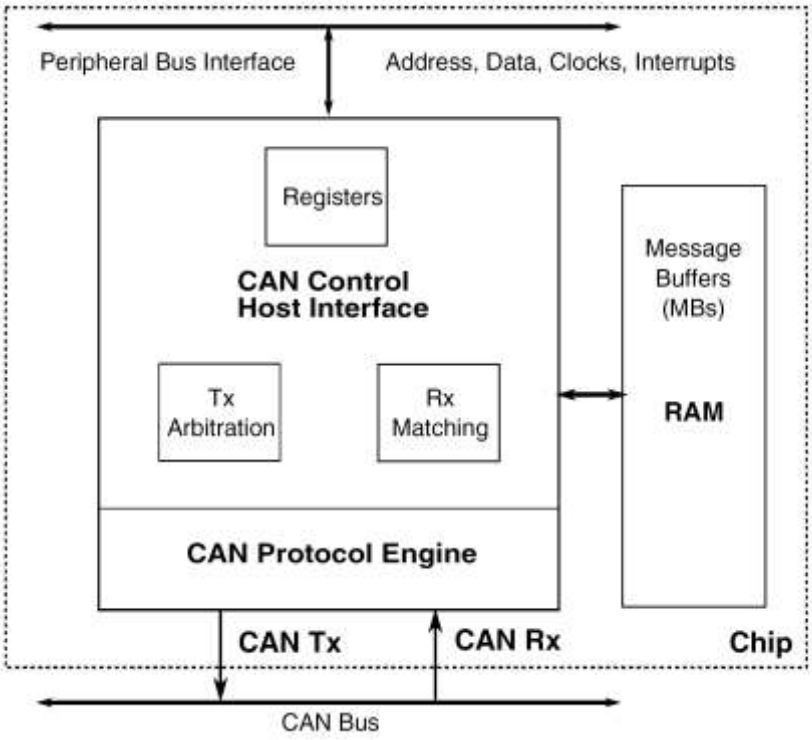

图 48-1 FlexCAN 框图

48.1.1 概述

CAN 协议主要（但不是唯一）是设计用于汽车串口总线使用，在以下这些方面满足了规范的要求：实时处理、汽车在电磁干扰环境下的可靠操作、成本效益以及有带宽要求。FlexCAN 模块完全实现了 CAN 协议规范 V2.0B 版本，该版本支持标准的与扩展的消息帧。消息缓冲区存储在一个专用于 FlexCAN 模块的 RAM 区。可以参考芯片配置细节来了解在 MCU 内配置的消息缓冲区的数目。

CAN 协议引擎(PE)子模块管理 CAN 总线上的串行通信，请求存取 RAM 接收和传输消息帧，验证接收到的消息以及进行错误处理。控制器主机接口子模块用来选择接收和传输的消息缓冲区，使用仲裁与 ID 匹配算法，以建立同 CPU 或者其他模块的连接。时钟、地址、数据总线或者中断输出以及测试信号通过总线接口单元都可以进行访问。

48.1.2 FlexCAN 模块特征

FlexCAN 模块具有以下鲜明的传统特征：

1.完全支持 CAN2.0B 协议：标准数据帧；扩展数据帧；0～8 字节的数据长度；可编程控制的波特率，数据传输速率可达 1Mbps；与内容相关的寻址防止。

2.0 到 8 字节长度报文缓冲区；

3.每个报文缓冲区都可以配置成发送缓冲区或者接受缓冲区，支持标准和扩展帧格式；

4.每个消息缓冲区都有自己的接受掩码控制寄存器；

5.全功能的接受队列，该队列可以存储最多 6 个帧，并且自动进行内部指针处理；

6.传输中止能力；

7.可编程的 CAN 协议接口的时钟源，可以是总线时钟也可以是外部晶振；

8.没有使用的结构空间可以当成普通的 RAM 空间使用；

9.可编程的回环模式支持自测试；

10.可编程的传输优先级机制：最低 ID、最少缓冲区数目或者最高优先级；

11.基于 16 位自由运行定时器的时间戳机制；

12.全局网络时间，通过一个特殊的帧来同步；

13.中断掩码；

14.独立的传输媒介（假定一个外部收发器）；

15.对于高优先权信息具有短的相应时间；

16.低功耗模式，当总线激活时可以编程实现从低功耗模式唤醒。

此外，下面的最新的主要特征也添加到了先前的 FlexCAN 版本中：

1.远程请求帧可以被自动或者软件处理；

2.正常模式下 ID 过滤配置的安全机制；

3.只有在“Freeze”模式下进行的 CAN 比特时间设置与配置比特；

4.发送缓冲区状态（最低优先级缓冲区或者空缓冲区）；

5.用于接收帧的 IDHIT 寄存器；

6.SYNC 比特状态位用来指示模块已经与 CAN 总线同步；

7.调试寄存器

8.用于报文的 CRC 状态；

9.接收队列全局掩码寄存器；

10.在匹配过程中接受缓冲区与接受队列可选择的优先级；

11.强大的接受队列 ID 过滤功能，能够匹配接受的 ID 是扩展的 128 字节、标准的 256 字节或者 512 部分（8 比特）ID，最多 32 个匹配能力。

12.100%兼容 FlexCAN 以前版本。

48.1.3 操作模式

FlexCAN 模块一共具有四种不同的功能模式：正常模式（用户和管理员）、冻结模式、监听模式以及闭环模式。同样具有三种不同的低功耗模式：禁止模式、睡眠模式以及停止模式。

1.正常模式（用户与管理员）

在正常模式下，CAN 模块收发数据帧、处理错误，CAN 协议的所有功能全部开启。对于一些控制比较严格的寄存器，在用户模式和管理员模式下访问时又区别的。

2.冻结模式

如果 MCR 寄存器的 FRZ 位被置位，那么将开启 CAN 模块的冻结模式。当 MCR 的 HALT 位置位时或者在 MCU 级请求调试模式（Debug Mode）并且 MCR 寄存器的 FRZ_ACK 位被置位时，CAN 模块将进入到冻结模式。在该模式下，将停止收发帧，并且将丢失与 CAN 总线的同步。有关该模式的更多信息可以参见“冻结模式”节。

3.监听模式

当控制 1 寄存器的 LOM 位置位时，模块将进入到监听模式。在该模式下，CAN 模块禁止数据收发，所有的错误计数器都被冻结，且该模块工作在 CAN 被动错误模式。只有被其他 CAN 节点应答了的报文才可以被监听的节点接受。如果 FlexCAN 检测到一个还没有被应答的报文，则标记为一个 BIT0 错误（不会改变 REC），将如同它试图应答报文一些样。

4.回环模式

如果控制 1 寄存器的 LPB 位被置位，模块将进入到回环模式。在该模式下，FlexCAN 工作在内部闭环模式用于自测。从发送器发送出的比特流输出回内部的接收器输入。输入引脚将会被忽略，并且输出引脚将处于逻辑 1 状态。发送报文时，FlexCAN 模块和正常模式一样，而接收器则认为接受它自己的报文与接受远程节点的报文相同。FlexCAN 为保证能正确接受到自己发送的报文，将忽略应答间隙内的应答字段。报文接受发送时，如果中断使能则 FlexCAN 将向 CPU 产生中断。

5.模块禁止模式

当 MCR 寄存器的 MDIS 位被 CPU 置位并且 LPM_ACK 位被 FlexCAN 模块置位时，模块将会进入该低功耗模式。如果模块被禁止，那么模块将会请求停止 CAN 协议引擎的时钟并且请求禁止控制器主机接口子模块。通过忽略 MCR 寄存器 MDIS 位可以退出该模式。有关该模块的更多信息参考“模块禁止模式”节。

6.睡眠模式

当 MCR 寄存器的 DOZE 位被置位、在 MCU 级请求睡眠模式并且由 FlexCAN 置位 MCR 寄存器的 LPM_ACK 位时模块将进入到该低功耗模式。当模块处于睡眠模式时，FlexCAN 将会请求禁止 CAN 协议引擎的时钟并且请求禁止 CAN 控制器主机接口子系统。当 MCR 寄存器的 DOZE 位被忽略（negated）、当 MCU 离开睡眠模式时或者当检测到 CAN 总线上有活动并且自醒机制开启时模块将退出睡眠模式。更多信息可以参考“睡眠模式”这一节的内容。

7.停止模式

在 MCU 级请求模式并且由 FlexCAN 模块置位 MCR 寄存器的 LPM_ACK 位时，模块将会进入到该低功耗模式。在停止模式中，模块将会将自己置于不活动状态，然后通知 CPU 可以关闭所有的时钟。当请求离开停止模式或者当检测到 CAN 总线上有或者并且开启了自醒即使时，FlexCAN 将会退出该模式。有关更多信息可以参考“停止模式”这一节的内容。

48.2 FlexCAN 信号模式

FlexCAN 一共有两个 IO 引脚用来连接外部的 MCU 引脚。下表将列出这两个引脚，在随后的章节将会有这两个引脚的介绍。

表 48-1 FlexCAN 模块引脚描述

引脚	描述	I/O
CAN Rx	CAN接受引脚	输入
CAN Tx	CAN输出引脚	输出

48.2.1 CAN Rx 引脚

该引脚为来自于 CAN 总线收发器的接受数据引脚。显性状态表示逻辑‘0’；隐形状态表示逻辑‘1’。

48.2.2 CAN Tx 引脚

该引脚为来自于 CAN 总线收发器的发送引脚。显性状态代表逻辑‘0’，隐形状态代表逻辑‘1’。

48.3 内存映射/寄存器定义

本节介绍 FlexCAN 模块中的寄存器以及数据结构。模块的基址取决于 MCU 部分内存映射。

48.3.1 FlexCAN 内存映射

下表中为 FlexCAN 模块全部的内存映射。

从模块基址开始一共有 128 字节的地址空间用于 FlexCAN 模块的寄存器定义，模块基址开始与 0x0080。

每一个独立的寄存器都通过其唯一的名字及助记符来表示。访问权限可以是管理员身份或者无限制。大部分的寄存器都可以通过对 MCR 寄存器的 SUPV 位编程配置成管理员身份或者无限制访问。表 48-2 的访问权限一列定义了这些寄存器的访问权限 S/U。

寄存器 IFLAG2 与 IMASK2 位预留空间，这依赖于设备可以用的消息缓冲区个数。

表 48-2 模块内存映射

寄存器	访问权限	硬件复位 是否受影响	软件复位 是否受影响
模块配置寄存器（MCR）	S	是	是
控制寄存器 1（CTRL1）	S/U	是	否
自由运行定时器（TIMER）	S/U	是	是
接受全局掩码寄存器（RXMGMASK）	S/U	否	否
接受缓冲 14 掩码寄存器（RX14MASK）	S/U	否	否
接受缓冲 15 掩码寄存器（RX15MASK）	S/U	否	否
错误计数寄存器（ECR）	S/U	是	是
错误和状态寄存器 1（ESR1）	S/U	是	是
中断掩码寄存器 2（IMASK2）	S/U	是	是
中断掩码寄存器 1（IMASK1）	S/U	是	是
中断标志寄存器 2（IFLAG2）	S/U	是	是
中断标志寄存器 1（IFLAG1）	S/U	是	是
控制寄存器 2（CTRL2）	S/U	是	否
错误和状态寄存器 2（ESR2）	S/U	是	是
个别匹配元素更新寄存器（IMUER）	S/U	是	是
丢失接受帧寄存器（LRFR）	S/U	是	是

CRC 寄存器（CRCR）	S/U	是	是
接受队列全局掩码寄存器（RXFGMASK）	S/U	否	否
接受队列信息寄存器（RXFIR）	S/U	否	否
报文缓冲	S/U	否	否
接受私有掩码寄存器	S/U	否	否

FlexCAN 模块可以使用邮箱和 Rx 接受队列结构存储 CAN 报文以用于传输与接受。

FlexCAN 模块的内存映射包括 16 个 128 比特的报文缓冲（MBs），该缓冲区占据了偏移地址的 0x80 到 0x17F。

CAN 内存映射

绝对地址	寄存器名字	宽度（Bit）	访问权限	复位值	章节
4002_4000	模块配置寄存器（CAN0_MCR）	32	R/W	D890_000Fh	48.3.2
4002_4004	控制寄存器1（CAN0_CTRL1）	32	R/W	0000_0000h	48.3.3
4002_4008	自由运行定时器（CAN0_TIMER）	32	R/W	0000_0000h	48.3.4
4002_4010	接受邮箱全局掩码寄存器（CAN0_RXMGMASK）	32	R/W	FFFF_FFFFh	48.3.5
4002_4014	接收14掩码寄存器（CAN0_RX14MASK）	32	R/W	FFFF_FFFFh	48.3.6
4002_4018	接收15掩码寄存器（CAN0_RX15MASK）	32	R/W	FFFF_FFFFh	48.3.7
4002_401C	错误计数寄存器（CAN0_ECR）	32	R/W	0000_0000h	48.3.8
4002_4020	错误与状态寄存器1（CAN0_ESR1）	32	R/W	0000_000h	48.3.9
4002_4024	中断掩码寄存器2（CAN0_IMASK2）	32	R/W	0000_0000h	48.3.10
4002_4028	中断掩码寄存器1（CAN0_IMASK1）	32	R/W	0000_0000h	48.3.11
4002_402C	中断标志寄存器2（CAN0_IFLAG2）	32	R/W	0000_0000h	48.3.12
4002_4030	中断标志寄存器1（CAN0_IFLAG1）	32	R/W	0000_0000h	48.3.13
4002_4034	控制寄存器2（CAN0_CTRL2）	32	R/W	0048_0000h	48.3.14
4002_4038	错误与状态寄存器2（CAN0_ESR2）	32	R/W	0000_0000h	48.3.15
4002_4044	CRC寄存器（CAN0_CRCR）	32	R	0000_0000h	48.3.16
4002_4048	接收队列全局掩码寄存器（CAN0_RXFGMASK）	32	R/W	FFFF_FFFFh	48.3.17
4002_404C	接收队列信息寄存器（CAN0_RXFIR）	32	R	0000_000Xh	48.3.18
4002_4880	接收个别掩码寄存器（CAN0_RXIMR0）	32	R/W	0000_000Xh	48.3.19
4002_4884	接受个别掩码寄存器（CAN0_RXIMR1）	32	R/W	0000_000Xh	48.3.19
4002_4888	接受个别掩码寄存器（CAN0_RXIMR2）	32	R/W	0000_000Xh	48.3.19
4002_488C	接受个别掩码寄存器（CAN0_RXIMR3）	32	R/W	0000_000Xh	48.3.19

4002_4890	接受个别掩码寄存器（CAN0_RXIMR4）	32	R/W	0000_000Xh	48.3.19
4002_4894	接受个别掩码寄存器（CAN0_RXIMR5）	32	R/W	0000_000Xh	48.3.19
4002_4898	接受个别掩码寄存器（CAN0_RXIMR6）	32	R/W	0000_000Xh	48.3.19
4002_489C	接受个别掩码寄存器（CAN0_RXIMR7）	32	R/W	0000_000Xh	48.3.19
4002_48A0	接受个别掩码寄存器（CAN0_RXIMR8）	32	R/W	0000_000Xh	48.3.19
4002_48A4	接受个别掩码寄存器（CAN0_RXIMR9）	32	R/W	0000_000Xh	48.3.19
4002_48A8	接受个别掩码寄存器（CAN0_RXIMR10）	32	R/W	0000_000Xh	48.3.19
4002_48AC	接受个别掩码寄存器（CAN0_RXIMR11）	32	R/W	0000_000Xh	48.3.19
4002_48B0	接受个别掩码寄存器（CAN0_RXIMR12）	32	R/W	0000_000Xh	48.3.19
4002_48B4	接受个别掩码寄存器（CAN0_RXIMR13）	32	R/W	0000_000Xh	48.3.19
4002_48B8	接受个别掩码寄存器（CAN0_RXIMR14）	32	R/W	0000_000Xh	48.3.19
4002_48BC	接受个别掩码寄存器（CAN0_RXIMR15）	32	R/W	0000_000Xh	48.3.19
400A_4004	控制寄存器1（CAN1_CTRL1）	32	R/W	0000_0000h	48.3.3
400A_4008	自由运行定时器（CAN1_TIMER）	32	R/W	0000_0000h	48.3.4
400A_4010	接受邮箱全局掩码寄存器（CAN1_RXMGMASK）	32	R/W	FFFF_FFFFh	48.3.5
400A_4014	接收14掩码寄存器（CAN1_RX14MASK）	32	R/W	FFFF_FFFFh	48.3.6
400A_4018	接收15掩码寄存器（CAN1_RX15MASK）	32	R/W	FFFF_FFFFh	48.3.7
400A_401C	错误计数寄存器（CAN1_ECR）	32	R/W	0000_0000h	48.3.8
400A_4020	错误与状态寄存器1（CAN1_ESR1）	32	R/W	0000_000h	48.3.9
400A_4024	中断掩码寄存器2（CAN1_IMASK2）	32	R/W	0000_0000h	48.3.10
400A_4028	中断掩码寄存器1（CAN1_IMASK1）	32	R/W	0000_0000h	48.3.11
400A_402C	中断标志寄存器2（CAN1_IFLAG2）	32	R/W	0000_0000h	48.3.12
400A_4030	中断标志寄存器1（CAN1_IFLAG1）	32	R/W	0000_0000h	48.3.13
400A_4034	控制寄存器2（CAN1_CTRL2）	32	R/W	0048_0000h	48.3.14
400A_4038	错误与状态寄存器2（CAN1_ESR2）	32	R/W	0000_0000h	48.3.15
400A_4044	CRC寄存器（CAN1_CRCR）	32	R	0000_0000h	48.3.16

400A_4048	接收队列全局掩码寄存器（CAN1_RXFGMASK）	32	R/W	FFFF_FFFFh	48.3.17
400A_404C	接收队列信息寄存器（CAN1_RXFIR）	32	R	0000_000Xh	48.3.18
400A_4880	接收私有掩码寄存器（CAN1_RXIMR0）	32	R/W	0000_000Xh	48.3.19
400A_4884	接受私有掩码寄存器（CAN1_RXIMR1）	32	R/W	0000_000Xh	48.3.19
400A_4888	接受私有掩码寄存器（CAN1_RXIMR2）	32	R/W	0000_000Xh	48.3.19
400A_488C	接受私有掩码寄存器（CAN1_RXIMR3）	32	R/W	0000_000Xh	48.3.19
400A_4890	接受私有掩码寄存器（CAN1_RXIMR4）	32	R/W	0000_000Xh	48.3.19
400A_4894	接受私有掩码寄存器（CAN1_RXIMR5）	32	R/W	0000_000Xh	48.3.19
400A_4898	接受私有掩码寄存器（CAN1_RXIMR6）	32	R/W	0000_000Xh	48.3.19
400A_489C	接受私有掩码寄存器（CAN1_RXIMR7）	32	R/W	0000_000Xh	48.3.19
400A_48A0	接受私有掩码寄存器（CAN1_RXIMR8）	32	R/W	0000_000Xh	48.3.19
400A_48A4	接受私有掩码寄存器（CAN1_RXIMR9）	32	R/W	0000_000Xh	48.3.19
400A_48A8	接受私有掩码寄存器（CAN1_RXIMR10）	32	R/W	0000_000Xh	48.3.19
400A_48AC	接受私有掩码寄存器（CAN1_RXIMR11）	32	R/W	0000_000Xh	48.3.19
400A_48B0	接受私有掩码寄存器（CAN1_RXIMR12）	32	R/W	0000_000Xh	48.3.19
400A_48B4	接受私有掩码寄存器（CAN1_RXIMR13）	32	R/W	0000_000Xh	48.3.19
400A_48B8	接受私有掩码寄存器（CAN1_RXIMR14）	32	R/W	0000_000Xh	48.3.19
400A_48BC	接受私有掩码寄存器（CAN1_RXIMR15）	32	R/W	0000_000Xh	48.3.19

48.3.2 模块配置寄存器（CANx_MCR）

该寄存器用来定义全局系统配置

地址：CAN0_MCR-4002_4000h（基址）+0h（偏移地址） = 4002_4000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

读	MDIS	FRZ	REFN	HALT	NOTRDY	WAKMSK	SOFTRST	FRZACK	SUPV	SLFWAK	WRNEN	LPMACK	预留	DOZE	SRXDIS	IRMQ
写					无效			无效				无效				
复位	1	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0

Bit	15-14		13	12	11-10		9-8		7	6-0						
读	0		LPRIOEN	AEN	0		IDMA		0	MAXMB						
写	无效				无效				无效							
复位	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

CANx_MCR 字段描述

字段	说明
31（MDIS）	模块禁止位。该位用来控制是否禁止FlexCAN模块。当禁止时，FlexCAN模块将禁止CAN协议引擎与控制器主机接口子系统的时钟。该位是MCR寄存器中唯一一位不受软件复位影响的位。 0 开启FlexCAN模块 1 禁止FlexCAN模块
30（FRZ）	冻结使能位。当MCR寄存器的HALT位置位或者当在MCU级请求调试模式时，该位用来指定FlexCAN模块的行为。当FRZ被置位，FlexCAN将会进入到冻结模式。该位置0将使FlexCAN模块退出冻结模式。 0 不开启进入到冻结模式 1 开启进入到冻结模式
29（RFEN）	开启接收队列。该位用来控制是否开启接收队列。当该位被置位时，MBs0到MBs5不能用于正常的接受与传输，因为其相应的内存空间（0x80～0xDC）被队列引擎以及MBs（最多32个，这取决于CTRL2[RFEN]位的设置）用于接收队列ID过滤器表元素。RFEN同样影响每一个CAN的外设时钟的最小个数的定义，正如在表“外设时钟频率与CAN比特率之间的最小比率”中描述的一样。该位只有在冻结模式下才能被写入，在其他模式下该位被通过硬件的方法锁定。 0 没有开启接受队列 1 开启了接受队列

28（HALT）	暂停FlexCAN模块。该位置位会导致FlexCAN模块进入到冻结模式。CPU在初始化报文缓冲区与控制寄存器之后应该将该位清0。当FlexCAN在任何一种低功耗模式下时，都不可能进入到冻结模式 0 没有冻结模式请求； 1 如果FRZ位置位那么进入到冻结模式。
27（NOTRDY）	FlexCAN模块未准备好。该位只读，用来表示FlexCAN现在处于以下几种模式：禁止模式、睡眠模式、停止模式或者冻结模式。该位清0可以使得FlexCAN模块退出以上模式。 0 FlexCAN模块处于正常模式、监听模式或者回环模式 1 FlexCAN模块处于禁止模式、睡眠模式、停止模式或者冻结模式
26（WAKMSK）	唤醒中断掩码。该位用来使能唤醒中断。 0 禁止唤醒中断 1 开启唤醒中断
25（SOFTRST）	软件复位。该位置位，FlexCAN模块将复位其内部状态机以及一些内存映射寄存器。一下寄存器将会被复位：MCR（除了MDIS位）、TIMER、ECR、ESR1、ESR2、IMASK1、IMASK2、IFLAG1、IFLAG2以及CRCR。用来控制与CAN总线接口的配置寄存器不受软件复位影响。以下寄存器不受影响：CTRL1、CTRL2、RXIMR0- RXIMR63、RXMGMASK、RX14MASK、RX15MASK、RXFGMASK、RXFIR以及所有的报文缓冲区。当CPU写MCR寄存器时，CPU可以直接将SOFTRST位置位，当在MCU级请求全局软件复位时，该位也同样会被置位。因为软件复位是同步的并且必须伴随一个跨时钟域的请求与应答过程，因此可能会需要一些时间来全部显示其效果。如果复位没有完成，那么SOFTRST位将会一直置位，且它会在复位完成之后自动取反。因此可以轮询该位以判断软件复位是否完成。 在任何一种低功耗模式下，如果时钟被关闭那么软件复位将不会起作用，模块应该首先退出低功耗模式，然后软件复位才会起作用。 0 没有复位请求 1 受软件复位影响复位寄存器
24（FRZACK）	冻结模式确认位。该位只读，用来指示FlexCAN模块现在处于冻结模式，并且它的预分频器是停止的。冻结模式请求将不会被执行直到当前传输或者接受过程完成。因此软件可以轮询该位以了解FlexCAN模块是否已确切进入到冻结模式。如果冻结模式请求被忽略，那么一旦FlexCAN模块预分频器又重新运行，该位也会取反。如果FlexCAN正处于任何一种低功耗模式时请求冻结模式，那么只有当模块退出低功耗模式时FRZACK位才会被置位。 0 FlexCAN模块没有处于冻结模式，预分频器处于运行之中 1 FlexCAN模块处于冻结模式，预分频器停止。
23（SUPV）	管理员模式。该位用来配置FlexCAN模块是处于管理员模式还是用户模式。模块内存映射表的访问权限一些标有S/U的寄存器受该位影响。该位复位之后的值为‘1’，因此受此位影响的寄存器将仅能以管理员的权限访问。该位只有在冻结模式下才能被写入，在其他模式下该位通过硬件的方式被锁住。 0 FlexCAN处于用户模式。所有受影响的寄存器可以以管理员或者无限制的方式访问； 1FelxCAN处于管理员模式，所有受影响的寄存器只能以管理员的身份访问。无限制访问行为就像访问一个没有实现的寄存器地址。

22（SLFWAK）	自唤醒。当FlexCAN模块处于睡眠模式或者停止模式时，该位可以使能自唤醒功能。当FlexCAN模块进入到睡眠模式或者停止模式的时候该位被置位，那么FlexCAN模块在这些模式下会在CAN总线上寻找一个从隐性到显性的转变。在睡眠模式下，如果检测到一个从隐性到显性的转变，那么FlexCAN将会请求重新恢复其时钟，并且如果使能了如此功能，将会向CPU产生一个唤醒中断。在停止模式下，如果检测到一个从隐性到显性的转变，使能了该功能，那么FlexCAN将会向CPU发出一个唤醒中断，因此它可以设置退出停止模式并且FlexCAN模块可以请求重新恢复时钟。在睡眠模式或者停止模式下该位不能被写入就好象该位被硬件锁住一样。 0 禁止FlexCAN模块自唤醒功能 1 开启FlexCAN模块自唤醒功能
21（WRNEN）	使能警告中断。如果该位被置位，将使能错误与状态寄存器中的TWRNINT与RWRNINT中断。如果该位被忽略，那么TWRNINT与RWRNINT标志位将会一直为0，独立于错误计数器的值，并且不会产生任何警告中断。该位只能在冻结模式下被写入，在其他模式下该位被硬件锁定。 0 TWRNINT与RWRNINT位为0，独立于错误计数器 1 当相应的错误计数器从小于96达到96或者比96大时，TWRNINT与RWRNINT位被置位。
20（LPMACK）	低功耗模式确认位。该位只读，用来表示FlexCAN是否处于静止模式、睡眠模式或者停止模式。FlexCAN模块不会进入到任何低功耗模式直到当前所有的传输与接受操作都完成。因此CPU可以轮询该位以了解FlexCAN模块是否已经确切地进入到了低功耗模式。
19（预留）	该位预留
18（DOZE）	使能睡眠模式。该位用来定义当在CPU级请求睡眠模式时是否允许FlexCAN进入到低功耗模式。当一旦检测到CAN总线上有活动（使能自我唤醒），FlexCAN模块将会从睡眠模式中唤醒时，该位会被自动复位。 0 当请求请求睡眠模式时，FlexCAN模块禁止进入到低功耗模式 1 当请求请求睡眠模式时，使能FlexCAN模块进入到低功耗模式
17（SRXDIS）	禁止自接受。该位用来定义是否允许FlexCAN模块接收由自己传送的帧。如果该位被置位，由模块传输出去的帧将不会存储在MB内，不管MB是否与传输帧ID匹配，并且由于帧的接受没有中断标志或者将不会产生中断信号。该位只有在冻结模式下才能被写入，在其他模式下该位被硬件锁定。 0 只能自接收 1 禁止自接收
16（IRMQ）	使能私有接收掩码与队列。该位用来表明匹配过程将基于私有掩码与队列或者基于由RXMGMASK、RX14MASK、RX14MASK及RXFGMASK构成的掩码机制。该位只能在冻结模式下写入，在其他模式下该位被硬件锁定。 0 禁止个别接收掩码和队列功能，为了向后兼容，C/S字的读取将锁定MB即使MB位空。 1 开启个别接收掩码和队列功能。
15-14（预留）	这两位只读，读出总为0.

13（LPRIOEN）	使能本地优先。该位是为了向后兼容的原因而提供的。该位用来控制是否开启本地优先的功能。主要用于在仲裁阶段扩展ID。有了扩展ID的概念，仲裁过程将会在基于32位的情况下完成，但是实际上传输的ID仍然为标准帧的11比特的ID以及扩展帧的29比特的ID。该位只能在冻结模式下写入，在其他模式下该位被硬件锁定。 0 禁止本地优先功能 1 开启本地优先功能
12（AEN）	启用中止。该位是为了向后兼容的原因而提供的，当该位被置位，它将开启传输中止功能。该功能提供了一个安全的程序来中止正在进行的传输，因此在CAN总线上如果没有通知将不会有帧传输。该位只能在冻结模式下写入，在其他模式下该位被硬件锁定。 警告：写中止代码到接受邮箱当MCR[AEN]位置位时，将导致不可预见的结果。 0 禁止中止功能 1 开启中止功能
11-10（预留）	这两位只读，读出总为0.
9-8（IDAM）	身份接受模式，这两位用来定义接受队列过滤器表元素的格式。主语该表的所有元素会被在配置该字段时被配置（它们具有相同的格式）。该字段只能在冻结模式下写入，在其他模式下该位被硬件锁定。 00 格式A：每一个ID过滤表元素有一个全ID（标准的以及扩展的）； 01 格式B：每一个ID过滤表元素都有两个全标准IDs或者两个14比特（标准的及扩展的）的IDs； 10 格式C：每一个过滤表元素有四个8比特标准IDs； 11 格式D：拒绝所有的帧。
7（预留）	该位只读，读出总为0。
6-0（MAXMB）	最大报文缓冲区个数，这7位定义了将参与匹配及仲裁过程的最大报文缓冲区的个数。复位值为0x0F，即最大报文缓冲区个数为16MB。该字段只能在冻结模式下写入，在其他模式下该位被硬件锁定。 注意：MAXMB设置的值必须小于参数NUMBER_OF_MB，不然最大有效报文个数将会是NUMBER_OF_MB-1。 另外，MAXMB的值必须包括由CTRL2[RFEN]定义的队列大小。MAXMB同样也影响由表“外设时钟与CAN比特率之间比值”所描述的每一个CAN比特外设时钟的最小个数的定义。

48.3.3 控制寄存器 1（CANx_CTRL1）

该寄存器用来定义和 CAN 总线相关的具体的 FlexCAN 模块的控制功能。

地址：CAN0_ CTRL1-4002_4000h（基址）+4h（偏移地址） = 4002_4004h

CAN1_ CTRL1-400A_4000h（基址）+4h（偏移地址） = 400A4004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	PRESDIV								RJW		PSEG1			PSEG2		
写																

复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	BOFFMSK	ERRMSK	CLKSRC	LPB	TWRNMSK	RWRNMSK	0		SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
写							无效									
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_CTRL1 字段描述

字段	说明
31-24（PRESDIV）	预分频器分频系数。这8位用来定义PE时钟频率与串行时钟频率（Sclock）之间的比率。Sclock周期定义了CAN协议的最小时间份额，对于复位之后的值，Sclock频率等于PE的时钟频率。该寄存器的最大值为0xFF，位Sclock的最小时钟频率，等于PE时钟频率除以256的值。该字段只能在冻结模式下写入，在其他模式下该位被硬件锁定。 Sclock频率=PE时钟频率/（PRESDIV+1）
23-22（RJW）	再同步补偿位。这两位定义了在一次在同步过程中允许补偿误差的最大值。（1个最小时间份额为Sclock时钟周期）。有效值为0～3，该字段只能在冻结模式下写入，在其他模式下该位被硬件锁定。 再同步补偿宽度=RJW+1。
21-19（PSEG1）	相位段1。该字段用来定义相位段1位时间长度。有效值为0～7，该字段只能在冻结模式下写入，在其他模式下该位被硬件锁定。 相位段1=（PSEG1+1）×Time-Quanta。
18-16（PSEG1）	相位段2。该字段用来定义相位段2位时间长度。有效值为0～7，该字段只能在冻结模式下写入，在其他模式下该位被硬件锁定。 相位段2=（PSEG2+1）×Time-Quanta。
15（BOFFMSK）	总线关闭掩码。该位提供了总线关闭中断的掩码。 0 禁止总线关闭中断； 1 开启总线关闭中断。
14（ERRMASK）	错误中断掩码。该位提供了错误中断的掩码。 0 禁止错误中断； 1 开启错误中断。
13（CLKSRC）	CAN引擎时钟源。该位用来为CAN协议引擎（PE）选择时钟源，可以是设备外设时钟（有PLL驱动）也可以是外部晶体振荡器时钟。所选择的时钟源供预分频使用以产生串行时钟（Sclock）。为了保证可靠的运行，该位只有在禁止模式下才能被写入，在其他模式下该位被硬件锁定。

12（LPB）	闭环模式选择位。该位用来配置FlexCAN工作在闭环模式下。在该模式下，FlexCAN模式执行内部循环用于自测。发送器的输出比特流返回到内部接收器的输入引脚。Rx CAN输入将会被忽略并且Tx CAN输出将会处于逻辑‘1’状态。发送报文时，FlexCAN模块和正常模式一样，而接收器则认为接受它自己的报文与接受远程节点的报文相同。 FlexCAN为保证能正确接受到自己发送的报文，将忽略应答间隙内的应答字段。报文接受发送时，如果中断使能则FlexCAN将向CPU产生中断。该位只有在冻结模式下才能被写入，在其他模式下该位被硬件锁定。 注意：在该模式下MCR[SLFDIS]位不能被置位，因为这将阻碍发送报文的自接收。 0 禁止闭环模式； 1 开启闭环模式。
11（TWRNMSK）	发送警告中断掩码。该位提供了与错误与状态寄存器的TWRNINT中断标志有关的发送警告中断掩码。当MCR[WRNEN]位为0时，该位读出为0。只有当MCR[WRNEN]位置位时该位才能够被写入。 0 禁止发送警告中断； 1 开启发送警告中断。
10（RWRNMSK）	接收警告中断掩码。该位提供了与错误与状态寄存器的TWRNINT中断标志有关的的接收警告中断掩码。当MCR[WRNEN]位为0时，该位读出为0。只有当MCR[WRNEN]位置位时该位才能够被写入。 0 禁止接收警告中断； 1 开启接收警告中断。
9-8（预留）	这两位为只读位，并且读出总为0。
7（SMP）	采样模式选择位。该位定义了接收引脚（Rx pin）的采样模式。该位只有在冻结模式下才能被写入，在其他模式下该位被硬件锁定。0 仅采样一次来决定比特的值；1 通过三次采样来决定接收到比特的值：常规的一次（采样点）以及两次之前的采样，使用了多数裁定原则（少数服从多数原则）。
6（BOFFREC）	总线关闭恢复模式。该位定义了FlexCAN如何从总线关闭状态中返回。如果该位被清0，则自动根据CAN2.0B协议规定从总线关闭状态中恢复。如果该位被置位，总线关闭自动恢复被禁止。在这种情况下，总线关闭时，除非通过用户复位，否则该模块将保持总线关闭状态。如果在CAN总线上监测到128次连续11个1位之前该位被清0，则总线关闭回复，就像BOFFREC从没有被置位过一样。如果该位在CAN总线上监测到128次连续11个1位之后该位被清0，则FlexCAN通过等待连续11个1位来重新同步总线。该位清0后，在总线关闭该时该位可以被再次置位，但是它只有在下一次进入总线关闭状态时才生效。如果在模块进入总线关闭状态时BOFFREC被清0，在监测到128次出现11个1之后，允许的总线关闭恢复节点将会到主动错误状态。 0 当总线关闭时，自动根据CNA2.0B协议规范从总线关闭状态中恢复； 1 禁止从总线关闭状态中自动恢复。

5（TSYN）	计数器同步模式位。该位定义计数器的同步模式。使能MB0每接收到一个报文就复位自由运行计数器的机制。该机制为同步多个FlexCAN结点提供特定的SYNC报文（全局网络时间）。如果MCR寄存器的RFEN位置位（启用接收队列），第一个可用的邮箱（通过CTRL2[RFEN]的设置）而不是MB0被用于计数器的同步。该位只有在冻结模式下才能被写入，在其他模式下该位被硬件锁定。 0 禁止计数器同步； 1 使能计数器同步。
4（LBUF）	报文缓冲区的发送次序位。该位定义了报文缓冲区传输的排序机制。当该位被置位时，LPRIOEN位将不会影响优先级的裁定。该位只有在冻结模式下才能被写入，在其他模式下该位被硬件锁定。 0 具有高优先级的缓冲先发送； 1 缓冲区序号最小的先发送。
3（LOM）	监听模式位。该位配置FlexCAN工作在监听模式。在该模式下， FlexCAN模块禁止数据发送，所有的错误计数器都被冻结，且该模块工作在CAN被动错误模式。只有被其他CAN结点应答了的报文才可以被监听的结点接收。如果FlexCAN检测到一个还没有被应答的报文，则标记为一个BIT0错误，就好像它试图应答报文一样。监听模式应答通过ESR1位的状态可知，开启监听模式时，该位是一个被动错误。该位只有在冻结模式下才能被写入，在其他模式下该位被硬件锁定。0 停用监听模式；1 FlexCAN模块处于监听模式。
2-0（PROPSEG）	传播时间段，这3位定义了发送阶段位时间长度，有效值为0～7，该位只有在冻结模式下才能被写入，在其他模式下该位被硬件锁定。传播时间=（PROPSEG+1）× Time-Quanta；Time-Quantum=一个Sclock周期。

48.3.4 自由运行计数器计数器（CANx_TIMER）

该寄存器内有一个 16 位自由运行的计数器，可以通过 CPU 进行读写。该寄存器复位值为 0x0000，计数到 0xFFFF 后回到 0x0000 重新开始计数。该计数器通过 FlexCAN 位时钟（位时钟决定 CAN 总线波特率）来计数。在报文发送/接收期间，每进行一次发送/接收，计数器加一。当总线上没有报文时，它按照编程设置的波特率来计数。在冻结、禁止、睡眠以停止模式下定时器将不再增加。

在 CAN 总线上，不管是收到什么类型的帧，该计数值总是在 ID 字段的起始处被捕获。捕获的值在报文被正确发送或接收后将被写入到的报文缓冲区的时间戳（Time Stamp）字段。

如果 CTRL1[TSYN]位被置位，那么每当一个报文接收到第一个可用的邮箱时（根据 CTRL2[RFEN]的设置）计时器将会被复位。

CPU 可以在任何时候写该寄存器。但是如果写操作发生在第一个邮箱的接受操作而导致定时器复位的时刻，那么写入的值将会被丢弃。

查看该寄存器影响邮箱的解锁过程参阅“消息缓冲区锁机制”章节。

地址：CAN0_TIMER-4002_4000h（基址+8h（偏移地址）=4002_4008h

CAN1_TIMER-400A_4000h（基址+8h（偏移地址）=400A_4008h

Bit	31-16								15-0							
读	0								TIMER							
写	无效															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_TIMER 字段描述	
字段	说明
31-16（预留）	只读位，读出总为0
15-0	计数器值。包含自由运行计数器的值。

48.3.5 接受邮箱全局掩码寄存器（CANx_RXMGMASK）

该寄存器位于 RAM 区。

提供该寄存器是为了对传统的支持。当 MCR[IRMQ]位清 0 时，RXMGMASK 总是全部有效。当 MCR[IRMQ]位置位时，RXMGMASK 无效。

RXMGMASK 用于给所有接受 MB 提供过滤字段，除了 MBx14-15，因为他们有自己的掩码寄存器。

该寄存器只有在冻结模式下才能被写入，在其他模式下该位被硬件锁定。

地址：CAN0_RXMGMASK-4002_4000h（基址）+10h（偏移地址）=4002_4010h

CAN1_RXMGMASK-400A_4000h（基址）+10h（偏移地址）=400A_4010h

Bit	31-0															
读	MG[31:0]															
写																
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CANx_RXMGMASK 字段描述	
字段	说明

<table>
<tr><td>31-0（MG[31:0]）</td><td>接收邮箱全局掩码位。这些位用来作为掩码匹配邮箱。注意邮箱ID字的对其并不如两个最重要的MG比特字段RTR与IDE那么完善。这两个字段位于邮箱的控制与状态寄存器。下表详细介绍了MG比特是如何掩码每个邮箱的过滤字段的。
<table>
<tr><td rowspan="2">SMB [RTR]1</td><td rowspan="2">CTRL2 [RRS]</td><td rowspan="2">CTRL2 [EACEN]</td><td colspan="4">邮箱过滤字段</td></tr>
<tr><td>MB[RTR]</td><td>MB[IDE]</td><td>MB[ID]</td><td>预留</td></tr>
<tr><td>0</td><td>-</td><td>0</td><td>note2</td><td>note3</td><td>MG[28:0]</td><td>MG[31:29]</td></tr>
<tr><td>0</td><td>-</td><td>1</td><td>MG[31]</td><td>MG[30]</td><td>MG[28:0]</td><td>MG[29]</td></tr>
<tr><td>1</td><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td><td>MG[31:0]</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>-</td><td>-</td><td>MG[28:0]</td><td>MG[31:29]</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>MG[31]</td><td>MG[30]</td><td>MG[28:0]</td><td>MG[29]</td></tr>
</table>
1.接受帧的RTR比特位。其被存放在附加的MB，即接受串行报文缓冲区（Rx SMB）；
2.如果CTRL2[EACEN]位被清0，邮箱的RTR比特将永不会和接收到帧的RTR比特进行比对。
3. 如果CTRL2[EACEN]位被清0，邮箱的IDE比特将永远和接收到帧的IDE比特进行比对。
0 相应的比特在过滤器中为“无关”
1 相应的比特在过滤器中为检查</td></tr>
</table>

48.3.6 接收 14 掩码寄存器（CANx_RX14MASK）

该寄存器位于 RAM 区。

提供该寄存器是为了对传统的支持。当 MCR[IRMQ]位被置位时，RX14MASK 无效。

RX14MASK 用来给报文缓冲区 14 的过滤字段提供掩码。

该寄存器只有在冻结模式下才能被写入，在其他模式下该位被硬件锁定。

地址：CAN0_RX14MASK-4002_4000h（基址）+14h（偏移地址）=4002_4014h

CAN1_RX14MASK-400A_4000h（基址）+14h（偏移地址）=400A_4014h

Bit	31-0															
读	RX14M[31:0]															
写																
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CANx_ RX14MASK 字段描述

字段	说明
31-0（RX14M [31:0]）	接受缓冲去14掩码位。每一个掩码比特用作邮箱14相应过滤字段的掩码，与RXMGMASK寄存器被设置为邮箱的过滤器的方式一样。详细信息可以参考CAN_RXMGMASK寄存器。0 相应的过滤器的相应位“不受影响”； 1 检查过滤器相应的比特。

48.3.7 接收 15 掩码寄存器（CANx_RX15MASK）

寄存器位于 RAM 区。

提供该寄存器是为了对传统的支持。当 MCR[IRMQ]位被置位时，RX15MASK 无效。

RX15MASK 用来给报文缓冲 15 的过滤字段提供掩码。

该寄存器只有在冻结模式下才能被写入，在其他模式下该位被硬件锁定。

地址：CAN0_RX15MASK-4002_4000h（基址）+18h（偏移地址）=4002_4018h

CAN1_RX15MASK-400A_4000h（基址）+18h（偏移地址）=400A_4018h

Bit	31-0															
读	RX15M[31:0]															
写																
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CANx_ RX15MASK 字段描述

字段	说明
31-0（MG[31:0]）	接受缓冲区15掩码比特。每一个掩码比特用作邮箱14相应过滤字段的掩码，与RXMGMASK寄存器被设置为邮箱的过滤器的方式一样。详细信息可以参考CAN_RXMGMASK寄存器。0 相应的过滤器的相应位“不受影响”； 1 检查过滤器相应的比特。

48.3.8 错误计数器（CANx_ECR）

该寄存器有两个 8 位的字段可以影响两个 FlexCAN 模块的错误计数器:即传输错误计数器（TXERRCNT 字段）以及接受错误计数器（RXERRCNT 字段）。这些寄存器自加及自减的规则在 CAN 协议中描述并且在 FlexCAN 模块中完全实现。两个计数器只读除了在冻结模式下，在该模式下这些计数器可以被 CPU 写入。

FlexCAN 可以响应协议中所描述的任何的总线状态，例如传输“主动错误”或“被动错误”标志，延迟其传输开始时间（“被动错误”）以及避免在“总线关闭”状态下在总线上的任何影响。以下为用于 FlexCAN 总线状态传输的基本规则。

1.如果 TXERRCNT 或者 RXERRCNT 增加到大于等于 128 时，错误与状态寄存器的 FLTCONF 字段将会更新以表示“被动错误模式”状态；

2.如果 FlexCAN 模块的状态为“被动错误”状态，并且 TXERRCNT 或者 RXERRCNT 自减到一个小于等于 127 的值并且其他的都已经满足条件,错误与状态寄存器的 FLTCONF 字段将会更新以表示“主动状态”;

3.如果 TXERRCNT 的值增加到大于 255，错误与状态寄存器的 FLTCONF 字段将会更新以表示“总线关闭”状态，并且可能会产生一个中断。TXERRCNT 的值会复位到 0。

4.如果 FlexCAN 模块处于“总线关闭”状态，然后 TXERRCNT 与另外一个内部计数器级联在 CAN 总线上达到第 128 个连续 11 个 1 位事件时，TXERRCNT 将会复位到 0 并且以这种方式计数,在内部计数器计数 11 个如此位且此后环绕递增 TXERRCNT 的值。当 TXERRCNT 达到 128 时，错误与状态寄存器的 FLTCONF 字段会更新到“主动错误”并且两个错误计数器都会复位到 0。

在少于 11 个连续隐形位数据流之后出现显性位，内部计数器将会复位自己为 0 并且不会影响 TXERRCNT 值。

5.在系统启动时刻，只能有一个节点工作，然后作为一个确认错误的结果（通过错误与状态寄存器的 ACKERR 为来表征），TXERRCNT 会在试图发送每一个报文时增加 1。当转变到“被动错误”状态之后，TXERRCNT 将不会再因为确认错误而增加。因此设备将不会在进入到“总线关闭”状态。

6.如果 RXERRCNT 增加到一个大于 127 的值，它将不会在增加，即使在接受时检测到再多的错误。在下一个报文正确接收时，计数器将会被设置一个在 119 到 127 之间的值用来恢复“主动错误”状态。

Bit	31-16				15-8						7-0					
读	0				RXERRCNT						TXERRCNT					
写	无效															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_ ECR 字段描述

字段	说明
31-16（预留）	这些位只读，且读出总为0.
15-8（RXERRCNT）	接受错误计数器
7-0（TXERRCNT）	传输错误计数器

48.3.9 错误与状态寄存器 1（CANx_ESR1）

该寄存器影响多种错误条件、设备的一些一般状态并且它是 CPU 的中断源。

CPU 的读操作将会清除 15-10 位，因此所报告的错误条件为上次 CPU 读取该寄存器之后发生的。第 9 到第 3 位是状态位。

下表显示了 FlexCAN 模块的状态变量以及他们的含义。其他的没有出现在表中的组合为预留。

SYNCH	IDLE	TX	RX	FlexCAN状态
0	0	0	0	不同步到CAN总线
1	1	X	X	Idle
1	0	1	1	传输
1	0	0	0	接受

地址：CAN0_ESR1-4002_4000（基址）+20h（便宜地址）=4002_4020h

CAN1_ESR1-400A_4000（基址）+20h（便宜地址）=400A_4020h

Bit	31-19				18	17	16
读	0				SYNCH	TWRNINT	RWRNINT
写	无效						
复位	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	BIT1ERR	BIT0ERR	ACKERR	CRCERR	FRMERR	STFERR	TXWRN		IDLE	TX	FLTCONF		RX	BOFFINT	ERRINT	WAKINT
写	无效															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_ ESR1 字段描述

字段	说明
31-19（预留）	这些位只读，且读出总为0.
18（SYNCH）	CAN同步状态位。该位只读用来指示FlexCAN模块是否已经同步到CAN总线，并且能够参与到通信过程。该位由FlexCAN模块置位与清除。参见CAN_ESR1寄存器介绍表。 0 FlexCAN模块不同步到CAN总线；1 FlexCAN模块同步到CAN总线
17（TWRNINT）	发送警告中断标志位。如果MCR寄存器的WRNEN位被置位，当TXWRN从0变为1时，TWRNINT位置位，意味着发送错误计数器达到了96。如果控制寄存器的相应掩码位（TWRNMSK）被置位，将会向CPU产生一个中断。通过向该位写1，可以清除该位。当WREN为被清0，该标志将会被屏蔽.CPU必须在禁止该位之前清除该位。否则，当WRNEN位再次被置位时该位也会被置位。向该位写0没有影响。在“总线停止”状态下，不会产生该标志。在冻结模式下该位不会被更新。 0 不发生此事件；1 发送错误计数器的值将会从小于96转换到大于等于96.
16（RWRNINT）	接收警告中断标志位。如果MCR寄存器的WRNEN位被置位，当RXWRN标志位从0转变为1时RWRNINT位被置位，这意味着接收错误计数器达到了96。如果控制寄存器的相应掩码为置位，那么将会向CPU产生一个中断。通过向该位写1清除该位。当WRNEN位被清0，该标志将会被屏蔽。CPU必须在禁止该位之前清除该标志位。不然该位会在WRNEN被再次置位时被置位。向该位写0没有影响。在冻结模式下该位不会被更新。 0 没有如此事件；1接收错误计数器的值将会从小于96转变到大于等于96。
15（BIT1ERR）	Bit1错误标志。该位表示在一个报文内传输与接收比特不一致。注意：该位不会被发送器置位如果在仲裁阶段或者ACK间隙，如果一个节点发送一个被动错误标志，该标志表示检测到一个显性比特。0 没有如此事件；1 至少有一个发送的隐形比特被接收为显性比特。
14（BIT0ERR）	Bit0错误标志。该位表示报文的传输与接收比特不一致。 0 没有如此事件；1 至少有一个发送的显形比特被接收为显性比特。
13（ACKERR）	确认错误标志。该位标志传输节点检测到一个确认错误。例如，在ACK间隙没有检测到一个显性比特。0 没有如此事件；1 从上次读取该寄存器之后发送了一个ACK错误。

12（CRCERR）	CRC校验错误。该位表示一个接收节点检测到CRC校验错误。例如，计算的CRC值与接收到的不一样。0 没有CRC校验错误事件；1 从上次读取该寄存器之后发生了一个CRC校验错误。
11（FRMERR）	格式错误。该位表示一个接收节点检测到一个格式错误。例如，一个固定格式的比特字段至少包含一个非法比特。0 没有格式错误事件；1 从上次读取该寄存器之后发生了一个格式错误。
10（SRFERR）	填充错误。该位表示检测到一个填充错误。0 没有填充错误事件；1从上次读取该寄存器之后发生了一个填充错误。
9（TXWRN）	发送错误警告。该位表示在报文传输阶段发生了一个重复误差。该位不会在冻结模式下更新。0 没有发送错误警告；1 TXERRCNT的值大于等于96；
8（RXWRN）	接收错误警告。该位表示在报文接受阶段重复出现一个错误。该位不会在冻结模式下更新。0 没有接收错误警告；1 RXERRCNT的值大于等于96；
7（IDLE）	该位用来表示CAN总线进入到IDLE状态。参看CAN_ESR1寄存器的描述。0 CAN总线没有进入IDLE状态；1 CAN总线进入IDLE状态。
6（TX）	FlexCAN正在传输数据。该位用来表示FlexCAN正在传输报文。参看CAN_ESR1寄存器的描述。0 FlexCAN模块没有传输报文；1 FlexCAN正在传输报文。
5-4（FLTCONF）	错误界定状态。这两位用来表示FlexCAN模块的界定模式。如果控制寄存器的LOM位被置位，一些延迟之后（该延迟时间决定于CAN位时间）FLTCONG字段会标志为“被动错误”。同样的延迟将会影响FLTCONF位，使CPU更新ECR寄存器。可能需要一个CAN比特时间来使得他们再一次一致。 因为控制寄存器不受软件复位的影响，如果LOM位被置位那么FLTCONF位也不会受软件复位的影响。 00 主动错误； 01 被动错误； 1X 总线关闭
3（RX）	FlexCAN模块正在接收报文。该位用来指示FlexCAN模块正在接收报文。参看CAN_ESR1寄存器的描述。0 FlexCAN模块没有接收报文；1 FlexCAN正在接收报文。
2（BOFFINT）	总线关闭中断。当FlexCAN模块进入到总线关闭状态时该位被置位。如果控制寄存器相应的掩码（BOFFMSK）置位，将会向CPU产生一个中断。通过向该位写1清除该标志位，写0无影响。0 没有总线关闭中断事件；1 FlexCAN模块进入到总线关闭状态。
1（ERRINT）	错误中断。该位标志至少有一位错误位（bits15-10）被置位。如果相应的掩码（CTRL1[ERRMSK]）被置位，那么会向CPU产生一个中断。过向该位写1清除该标志位，写0无影响。0 没有错误中断事件；1 表示在错误与状态寄存器的任何一个错误位被置位。
0（WAKINT）	唤醒中断。当FlexCAN处于睡眠模式或者停止模式时，如果在总线检测到一个从隐形到显性的转变时，并且MCR[WAKMSK]被置位，那么会向CPU产生一个中断。向该位写1清除该标志位。当SLFWAK被清0时，该位被屏蔽。CPU在禁止该位之前必须清除该位，不然当SLFWAK被再次置位时该位会被置位。写0无影响。0没有唤醒中断事件；1表示当FlexCAN模块在睡眠模式或者停止模式时，在CAN总线收到一个从隐形位到显性位的转变。

48.3.10 中断掩码寄存器 2（CANx_IMASK2）

该寄存器允许 32 个报文缓冲区中断的任何一个禁止或者开启。每一个缓冲区在该寄存器中都有一个相应的屏蔽位，允许 CPU 决定当成功接受或者发送时缓冲区是否产生中断。例如，当 IFLAG2 位被置位时，那么当缓冲区 2 完成收发过程时，会向 CPU 产生中断。

地址：CAN0_IMASK2-4002_4000h（基址）+24h（偏移地址）=4002_4024h

CAN1_IMASK2-400A_4000h（基址）+24h（偏移地址）=400A_4024h

Bit	31-0															
读	BUFHM															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_ IMASK2 字段描述

字段	说明
31-0（BUFHM）	缓冲区MBi掩码.用来使能或者禁止相应FlexCAN报文缓冲区中断。注意：如果相应的相应的IFLAG2位置位，那么置位或者清0IMASK2寄存器中的一位可以用来允许或者屏蔽一个中断请求。0 禁止相应的缓冲区中断；1开启相应的缓冲区中断。

48.3.11 中断掩码寄存器 1（CANx_IMASK1）

寄存器允许 32 个报文缓冲区中断的任何一个禁止或者开启。每一个缓冲区在该寄存器中都有一个相应的屏蔽位，允许 CPU 决定当成功完成接受或者发送时缓冲区是否产生中断。例如，当 IFLAG1 位被置位时，那么当缓冲区 1 完成收发过程时，会向 CPU 产生中断。

地址：CAN0_IMASK1-4002_4000h（基址）+28h（偏移地址）=4002_4028h

CAN1_IMASK1-400A_4000h（基址）+28h（偏移地址）=400A_4028h

Bit	31-0															
读	BUFLM															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_ IMASK1 字段描述

字段	说明
31-0（BUFLM）	缓冲区MBi掩码.用来使能或者禁止相应FlexCAN报文缓冲区中断。注意：如果相应的相应的IFLAG1位置位，那么置位或者清0IMASK1寄存器中的一位可以允许或者屏蔽一个中断请求。0 禁止相应的缓冲区中断；1开启相应的缓冲区中断。

48.3.12 中断标志寄存器 2（CANx_IFLAG2）

该寄存器用来定义 32 个报文缓冲区中断的标志位。每一个缓冲区都有一个中断标志位。每一次成功地传输与接受都会置位相应的 IFLAG2 位。如果相应的 IMASK2 被置位，那么将会产生一个中断。中断标志为必须通过向该位写 1 来清除，写 0 无影响。

在更新 MCR[MAXMB]字段之前，CPU 必须首先处理值大于 MCR[MAXMB]的中断服务程序。不然他们会仍然置位并且与可用的 MBs 的个数不一致。

地址：CAN0_ IFLAG2-4002_4000h（基址）+2Ch（偏移地址）=4002_402Ch

CAN1_IMASK1-400A_4000h（基址）+2Ch（偏移地址）=400A_402Ch

Bit	31-0															
读	BUFHi															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_ IFLAG2 字段描述

字段	说明
31-0（BUFHi）	缓冲区MB_i中断标志位。每一个FlexCAN报文缓冲中断都有相应的比特位。0 相应的缓冲区没有成功地完成传输或者发送报文；1相应的缓冲区已成功地完成传输或者接受报文

48.3.13 中断标志寄存器 1（CANx_IFLAG1）

该寄存器为 32 个报文缓冲区中断定义了标志位。每一个缓冲区都有一个中断标志位。每一次成功地传输或者接受都会设置相应的 IFLGA1 比特。如果相应的 IMASK1 寄存器位置位，那么就会产生一个中断。向相应的中断标记为写 1 可以清除该位，写 0 无影响。

当接受队列FIFO使能时,BUF7I与BUF5I标志也可以用来表示队列FIFO中断。当MCR[FREN]位置位时，最重要的 8 个中断标志 BUF[7:0]I 的功能该会改变：BUF7I、BUF5I 表明队列 FIFO 的执行条件。BUF4I 到 BUF0I 预留。

在使能 RFEN 之前，CPU 必须在 FILAG 在接受队列 FIFO 区域中处理中断，参见“接受队列 FIFO”章节。不然，FLAG 将会不正确地显示现在属于 FIFO 队列的 MBs 个数，这些队列有用于服务的内容。当 RFEN 位清 0 时，FIFO 标志必须被清除。当 RFFN 的值用来选择扩展接受队列 FIFO 过滤器超过 7 个时，也需要注意相同的问题。例如当 RFFN 的值为 0x8 时，MB0 到 MB23 范围被接受队列 FIFO 所使用并且相应的 IFLAG 比特位必须被清除。

在更新 MCR[MAXMB]字段之前，CPU 必须处理 MB 值大于 MCR[MAXMB]的 IFLAG1 比特位。不然他们会仍然置位，并且将会使得可用的 MBs 的个数与寄存器中显示的出现不一致的情况。

地址：CAN0_IFLAG1-4002_4000h（基址）+30h（偏移地址）=4002_4030h

CAN1_IFLAG1-400A_4000h（基址）+30h（偏移地址）=400A_4030h

Bit	31-8	7	6	5	4-0
读	BUF31TO8I	BUF7I	BUF6I	BUF5I	BUF4TO0I
写					
复位	0	0	0	0	

CANx_ IFLAG1 字段描述

字段	说明
31-8 （BUF31TO8I）	缓冲区MBi标志位。每一个比特表示FlexCAN报文缓冲区的中断标志。0 相应的缓冲区没有成功地完成传输或者接受操作；1相应的缓冲区成功地完成了传输或者接受。
7（BUF7I）	缓冲区MB7中断或者“接受队列溢出”。当MCR寄存器中的RFEN位被清0时（接受队列FIFO被禁止），该位用来表示MB7的中断标志位。 注意：当MCR[RFEN]位被CPU写入而改变时，该位被FlexCAN清空。当MCR[RFEN]位被置位，那么该位用来表示“接收队列FIFO溢出”标志。在这种情况下，该标志位标志一个报文将会因为接受队列满而丢失。注意当接受队列FIFO满时该位不会被置位，并且报文被邮箱捕捉。0 MB7缓冲区没有完成接收或者发送（当MCR[RFEN]=0）或者接收队列FIFO溢出（当MCR[RFEN]=1）；1 MB7完成发送或者接收（当MCR[RFEN]=0）或者接收队列FIFO溢出（当MCR[RFEN]=1）。
6（BUF6I）	缓冲区MB6中断或者“接收队列警告”。当MCR的RFEN位为0时（禁止接收队列FIFO），该位为MB6中断标志位。注意：当MCR[RFEN]位被CPU写入而改变时，该位被FlexCAN模块清除。当MCR[RFEN]位置位时，该位表示“接受队列警告”标志。在这种情况下，该位用来表示当在接收队列FIFO接收到一个新的报文时，未读报文数目从4增加到5，这意味着接收队列FIFO几乎已经满了。注意：如果当接收报文的数目大于4时该位被清除时，那么直到没有接受队列FIFO的未读报文减少到小于等于4时，该位将会不再被置位。0 没有发生MB6完成传输或者接受事件（当MCR[RFEN]=0）或者接受队列FIFO几乎满（当MCR[RFEN]=1）；1 MB6完成传输或者接受（当MCR[RFEN]=0）或者接收队列FIFO几乎满。
5（BUF5I）	缓冲区MB5中断或者“接收队列FIFO中可用的帧”。当MCR寄存器的RFEN位被清0时，该位用来表示MB5缓冲区中断。注意：当MCR[RFEN]位被CPU写入而改变时，该位被FlexCAN模块清除。当MCR[RFEN]位置位时，该位表示“接受队列FIFO中可用的帧”。在这种情况下该位用来表示在接收队列FIFO内至少一个帧可读。0 MB5没有发生传输或者接受操作（当MCR[RFEN]=0）或者在接收队列FIFO中可用的帧（当MCR[RFEN]=1）；1 MB5完成传输或者接受操作（当MCR[RFEN]=0）或者接收队列FIFO中可用帧（当MCR[RFEN]=1）。
4-0（BUF4TO0I）	缓冲区MBi中断标志位或者预留。当MCR寄存器的RFEN位清除时（禁止接收FIFO队列），这些位用来表示MB4到MB0的中断标志位。注意：当MCR[RFEN]位被CPU写入时，FlexCAN将会清除这些位标志。当MCR寄存器的RFEN位置位时，缓冲区4到缓冲区0的标志位被保留。0 相应的缓冲区没有发生发送或者接收事件（当MCR[RFEN]=0）；1 相应的缓冲区成功地完成了传输或者接收（当MCR[RFEN]=0）。

48.3.14 控制寄存器 2（CANx_CTRL2）

该寄存器包含 CAN 错误、FIFO 特征以及模式选择的控制位。

地址：CAN0_CTRL2-4002_4000h（基址）+34h=4002_4034h

CAN1_CTRL2-400A_4000h（基址）+34h=400A_4034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	无效	0		WRMFRZ	RFEN				TASD					MRP	RRS	EACEN
写	0	无效														
复位	1	1	0	1	1	0	0	0	0	1	0	0	1	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0															
写	无效															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_ CTRL2 字段描述

字段	说明
31（预留）	该位预留
30-29（预留）	只读位，并且读出总为0
28（WRMFRZ）	在冻结模式下对内存的写访问。在冻结模式下使能无限制写访问FlexCAN内存。该位只能在冻结模式下被写入并且在除了冻结模式下没有影响。 0 维持写限制访问；1使能无限制写访问FlexCAN内存。
27-24（RFEN）	接收队列FIFO过滤器的数目。这4位用来定义了下表中所示的接收队列过滤器的数目。最大的可选的过滤器数目由MCU来定义。该字段只能在冻结模式下被写，在其他模式下被硬件锁定。该字段不能被设置为大于接收队列及邮箱ID过滤器的报文缓冲区的数目，这些由MCR[MAXMB]所定义。注意：每组的8个过滤器拥有一个2个报文缓冲区大小的内存空间，这意味着过滤器使用的越多，满足条件的邮箱越少。考虑到有接收队列FIFO所用的内存空间最初保留用于MB0-5，RFEN的值必须设置为不大于可用内存空间的数的过滤器的值，可以按照以下方式来计算：（SETUP_MB-6）×4，SETUP_MB为NUMBER_OF_MB与MAXMB之间最小的。仍然可用的邮箱数目为：（SETUP_MB-8）-（RFEN×2）.如果接收队列FIFO的值被设置为超过SETUP_MB的值（可以使用的内存空间）（由RFEN所定义），那么超过的数目将不会起作用。0

RFEN	接收队列过滤器数	接受队列和ID过滤器表所占有的消	仍然可用的邮箱	由接收队列私有掩码所影响	由接收队列全局掩码所影响的接受队列ID

<table>
<tr><td></td><td>
<table>
<tr><td></td><td></td><td>息缓冲区</td><td></td><td>的接收队列ID过滤器元素</td><td>过滤器元素</td></tr>
<tr><td>0x01</td><td>8</td><td>MB0-7</td><td>MB8-63</td><td>元素0-7</td><td>没有</td></tr>
<tr><td>0x01</td><td>16</td><td>MB0-9</td><td>MB10-63</td><td>元素0-9</td><td>元素10-15</td></tr>
<tr><td>0x02</td><td>24</td><td>MB0-11</td><td>MB12-63</td><td>元素0-11</td><td>元素12-23</td></tr>
<tr><td>0x03</td><td>32</td><td>MB0-13</td><td>MB14-63</td><td>元素0-13</td><td>元素14-31</td></tr>
<tr><td>0x04</td><td>40</td><td>MB0-15</td><td>MB16-63</td><td>元素0-15</td><td>元素16-39</td></tr>
<tr><td>0x05</td><td>48</td><td>MB0-17</td><td>MB18-63</td><td>元素0-17</td><td>元素18-47</td></tr>
<tr><td>0x06</td><td>56</td><td>MB0-19</td><td>MB20-63</td><td>元素0-19</td><td>元素20-55</td></tr>
<tr><td>0x07</td><td>64</td><td>MB0-21</td><td>MB22-63</td><td>元素0-21</td><td>元素22-63</td></tr>
<tr><td>0x08</td><td>72</td><td>MB0-23</td><td>MB24-63</td><td>元素0-23</td><td>元素24-71</td></tr>
<tr><td>0x09</td><td>80</td><td>MB0-15</td><td>MB26-63</td><td>元素0-25</td><td>元素26-79</td></tr>
<tr><td>0xA</td><td>88</td><td>MB0-27</td><td>MB27-63</td><td>元素0-27</td><td>元素28-87</td></tr>
<tr><td>0xB</td><td>96</td><td>MB0-29</td><td>MB30-63</td><td>元素0-29</td><td>元素30-95</td></tr>
<tr><td>0xC</td><td>104</td><td>MB0-31</td><td>MB32-63</td><td>元素0-31</td><td>元素32-103</td></tr>
<tr><td>0xD</td><td>112</td><td>MB0-33</td><td>MB34-63</td><td>元素0-33</td><td>元素32-111</td></tr>
<tr><td>0xE</td><td>120</td><td>MB0-35</td><td>MB36-63</td><td>元素0-31</td><td>元素32-119</td></tr>
<tr><td>0xF</td><td>128</td><td>MB0-37</td><td>MB38-63</td><td>元素0-31</td><td>元素32-127</td></tr>
</table>
1. 最后剩余可用的邮箱数目是NUMBER_OF_MB减1和MCR寄存器的MAXMB字段之间的数
2. 如果接收队列私有掩码寄存器不可用则所有的接收队列过滤器受接收队列全局掩码的影响。
</td></tr>
<tr><td>23-19（TASD）</td><td>发送仲裁开始延迟。这5位用来定义在总线上从第一个CRC字段比特开始发送仲裁过程开始点所能持续的CAN比特时间。该字段只能在冻结模式下被写入，在其他模式下该字段被硬件锁定。该字段用来优化基于以下因素的传输性能：外设/串行时钟比率、CAN比特时间以及MBs个数。仲裁过程的持续时间（即CAN位时间）正比于可用的MB数和CAN波特率反比于外设时钟频率。优化的仲裁时间为在CAN数据帧的间隔字段的第一个比特之前最后一个MB被扫描的时间。因此，如果有很少的MBs、系统与串行时钟的比率很高并且CAN波特率很低，那么仲裁可以延迟反之亦然。
如果TASD为0，那么仲裁开始将不会延迟。因此CPU只有很少的时间来为下一次仲裁配置接收MB，但是可以有很长的时间用来仲裁。另一方面，如果TASD的值为24那么CPU可以推迟配置发送MB但是只会有很少的时间用于仲裁。
如果用于仲裁的时间太少，FlexCAN可能不会及时地找到CAN总线上相对于其他节点优先的MBs。如果在间隔字段的第一个比特之前结束仲裁的话，CPU就有机会来重新配置一些发送MBs，仲裁优先的MBs可能不是最适合传输的。TASD的最优配置可以按以下方式来计算：TASD=25-{f_{CANCLK}×[MAXB+3-(RFEN×8)-(RFEN×RFEN×2)]×2]}/
{f_{SYS}×[1+(PSEG1+1)+(PSEG2+1)+(PROPSEG+1)]×(PRESDIV+1)}。
其中：fCANCLK为协议引擎（PE）时钟，Hz；fSYS为外设时钟，Hz；MAXMB位字段CTRL1[MAXMB]的值；RFEN位字段CTRL1[RFEN]的值；RFEN位字段CTRL2[RFEN]的值；PSEG1位CTRL1[PSEG1]的值；PSEG2位CTRL1[PSEG2]的值；</td></tr>
</table>

	PROPSEG为CTRL1[PROPSEG]的值；PRESDIV为CTRL1[PRESDIV]的值；
18（MRP）	邮箱接收优先权。如果该位置位，匹配过程从邮箱开始，并且如果没有匹配成功，那么在接收队列FIFO中继续匹配。该位只有在冻结模式下被写入，在其他模式下该位被硬件锁定。0 首先在接收队列FIFO中匹配然后在邮箱中匹配；1 首先从邮箱开始匹配然后在接收队列FIFO中匹配。
17（RRS）	远程请求存储。如果该位被置位，远程请求将会开始匹配过程并且以和数据帧同样的方式存储在相应的报文缓冲区。不会自动产生远程回应帧。如果该位被清0，远程请求帧会被提交以开始匹配过程并且如果一个消息缓冲区的CODE=0b1010含有同样的ID那么会产生一个自动远程回应帧。该位只有在冻结模式下被写入，在其他模式下该位被硬件锁定。0 产生远程回应帧；1 远程请求帧被存储。
16（EACEN）	位接收邮箱开启整帧仲裁字段比较。在匹配过程中该位控制接收邮箱过滤器的IDE和RTR位与接收到的帧的相应位比较。该位不会影响接收队列FIFO的匹配。该位只有在冻结模式下被写入，在其他模式下该位被硬件锁定。0接收邮箱过滤器的IDE位总是比较并且RTR从来不比较不管其掩码位是什么；1 开启接收邮箱过滤器的IDE和RTR与接受帧的相应位进行比较。屏蔽位不起作用。
15-0（预留）	只读位，并且读出总为0。

48.3.15 错误与状态寄存器 2（CANx_ESR2）

该寄存器反应不同的中断标志与一些常用的状态。

地址：CAN0_ESR2-4002_4000h（基址）+38h（偏移地址）=4002_4038h

CAN1_ESR2-400A_4000h（基址）+38h（偏移地址）=400A_4038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0									LPTM						
写	无效									无效						
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0	VPS	IMB	0												
写	无效															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_ ESR2 字段描述

字段	说明
31-23（预留）	只读位，该位读出总为0.

22-16（LPTM）	最低优先权的发送邮箱。如果ESR2[VPS]位被置位，该位用来说明最低数目的非激活邮箱（参考IMB位的描述）。如果没有非激活邮箱，那么指示的邮箱依赖于CTRl1[LBUF]比特的值。如果CTRl1[LBUF]位被清0，那么邮箱是具有最大仲裁值的邮箱（参见“最高优先权优先”章节）。如果CTRL1[LBUF]位置位，那么邮箱会是最大数字的激活的发送邮箱。如果一个发送邮箱正在进行传输操作，那么在LPTM的计算将不会被考虑。如果ESR2[IMB]清0，并且一帧数据被成功地传输，LPTM将会以邮箱号来更新。
15（预留）	只读位，该位读出总为0。
14（VPS）	有效优先权状态。该位用来指示IMB和LPTM的内容当前是否有效。每一次成功地完成发送仲裁过程时VPS置位，除了在发送仲裁阶段，CPU写已经扫描过的邮箱的控制与状态寄存器，例如位于发送仲裁指针之后。如果有没有激活的邮箱并且只有一个正在发送的发送邮箱，那么该位被清0.在每一个发送仲裁过程开始或者写任何邮箱的控制与状态寄存器VPS都会被清0。注意ESR2[VPS]不会因为任何的CPU写一个被退出机制而锁定的MB的控制状态（C/S）而受到影响。当MCR[AEN]位置位时，写入到正在传输的MBC/S的中止代码或者任何企图进入到一个发送MB将会被锁定。0 IMB和LPTM的内容无效；1 IMB和LPTM的内容有效。
13（IMB）	非激活邮箱。如果ESR2[VPS]位置位，该位标识了是否有非激活的邮箱（CODE字段为0b1000或者0b0000）.在下列情况下该位被置位：1仲裁阶段，如果发送了一个LPTM并且它处于非激活状态；2如果IMB没有置位并且成功传输一个帧。在所有仲裁的开始该位被清0。 注意：LPTM机制具有如下的行为：如果一个MB被成功地传输并且ESR2[IMB]=0（没有非激活邮箱），那么ESR2[VPS]和ESR2[IMB]位都被置位并且刚刚被传输的MB相关索引被写入到ESR2[LPTM]。0 如果ESR2[VPS]被置位，ESR2[LPTM]是非激活邮箱；1 如果ESR2[VPS]被置位，那么会至少有一个非激活邮箱。LPTM为邮箱的第一位。
12-0（预留）	只读位，并且读出总为0.

48.3.16 CRC 寄存器（CANx_CRCR）

该寄存器提供了所传输报文的 CRC 信息。

地址：CAN0_CRCR-4002_4000h（基址）+44h=4002_4044h

CAN1_CRCR-400A_4000h（基址）+44h=400A_4044h

Bit	31-23	22-16	15	14-0
读	0	MBCRC	0	TXCRC
写	无效			
复位	0			

CANx_ CRCR 字段描述

字段	说明
31-23（预留）	只读位，并且读出总为0.
22-16（MBCRC）	CRC邮箱。该字段用来指示相对于TXCRC字段的邮箱号

15（预留）	只读位，并且读出总位0
14-0（TXCRC）	传输的CRC。该字段用来指示最后一个传输的报文的CRC值。在发送中断标志位被置位的同时该字段的值被更新。

48.3.17 接收队列 FIFO 全局掩码寄存器（CANx_RXFGMASK）

该寄存器位于 RAM 区。

如果接收 FIFO 队列被使能，RXFGMASK 被用来屏蔽接收队列 FIFO 过滤器表元素，这些表元素由于 CTRL2[RFEN]字段的设定并没有相应的 RXIMR。

该寄存器只能在冻结模式下被写入，在其他模式下该字段被硬件锁定。

地址：CAN0_RXFGMASK-4002_4000h（基址）+48h（偏移地址）=4002_4048h

CAN1_RXFGMASK-400A_4000h（基址）+48h（偏移地址）=400A_4048h

Bit	31-0
读	FGM[31:0]
写	
复位	0xFFFF_FFFF

CANx_ RXFGMASK 字段描述

字段	说明
31-0 （FGM[31:0]）	接收队列FIFO全局掩码位。这些比特位以完美对齐的方式用来屏蔽ID过滤器表元素。下表详细地描述了FGM位是如何对应IDAF的每一位的。

MCR [IDAM]	验证过滤器字段标识符					
	RTR	IDE	RXIDA	RXIDB[1]	RXIDC[2]	预留
A	FGM[31]	FGM[30]	FGM[29:1]	-	-	FGM[0]
B	FGM[31], FGM[15]	FGM[30] FGM[14]	-	FGM[29:16 FGM[13:0]]		-
C	-	-		-	FGM[31:24] FGM[23:16] FGM[15:8] FGM[7:0]]	

1.如果MCR[IDM]字段的值和格式B的相同，那么只有接收帧的的14个最重要的标识符比特和接收队列FIFO的过滤器进行比较。

2.如果MCR[IDAM]字段的值和格式C的相同，那么只有接收帧的8个最重要的标识符比特和接收队列FIFO的过滤器进行比较。

0 过滤器的相应比特位“不受影响”；1 检查过滤器的相应比特位

48.3.18 接收队列 FIFO 信息寄存器（CANx_RXFIR）

RXFIR 寄存器提供接收队列 FIFO 的有关信息。

通过该寄存器 CPU 可以访问位于 RAM 的 RXFIR 队列 FIFO 的输出结果。当有一个新的报文移到接收队列 FIFO 或者当接收队列 FIFO 的结果被新的消息更新时接收队列 FIFO 的结果也被

更新,，RXFIE 队列 FIFO 由 FlexCAN 写入。该寄存器的介绍参见“接收队列 FIFO”章节。

地址：CAN0_ RXFIR -4002_4000h（基址）+4Ch（偏移地址）=4002_404Ch

CAN1_ RXFIR -400A_4000h（基址）+4Ch（偏移地址）=400A_404Ch

Bit	31-9	8-0
读	0	IDHIT
写	无效	
复位	复位值没有定义。	

CANx_ RXFIR 字段描述

字段	说明
31-9（预留）	只读位，并且读出总为0.
8-0（IDHIT）	接收过滤器命中指示标识符。该字段指示哪个接收过滤器被接收队列FIFO中的接收报文命中，如果多个过滤器与接收到的报文ID相匹配，那么第一个匹配的IDAF（最小的数字）在匹配过程中所指示。只有当IFLAG[BUF5]字段置位时该字段是有效的。

48.3.19 接收私有掩码寄存器（CANx_RXIMRn）

该寄存器位于 RAM 区。

RXIMR 寄存器用作在接收 MBs 以及接收队列 FIFO 的 ID 过滤器的可接收掩码。如果没有使能接收队列 FIFO，那么会给每个可用的邮箱提供一个掩码寄存器，该掩码控制器为邮箱的每一位提供 ID 掩码功能。

当开启了接收队列 FIFO（MCR[RFEN]位置位），可以有高达 32 个接收私有掩码寄存器以一对一的方式应用于接收队列 FIFOID 过滤器表元素，这取决于 CTRL2[RFEN]位的设置。

RXIMR 只能在冻结模式下被 CPU 写入，不然在其他模式下该寄存器被锁定。

该寄存器不会由于复位而受到影响因此必须在任何接收之前对其进行初始化。

地址：4002_4000h（基址）+880h（4d×n），n=0 到 15d。

Bit	31-0
读	MI[31:0]
写	
复位	复位值没有定义。

CANx_ RXIMRn 字段描述

字段	说明
31-0（MI[31:0]）	私有掩码位。每一个私有掩码比特位以不同的方式为邮箱过滤器以及接收队列过滤表元素提供相应的掩码位。对于邮箱过滤器，参见RXMGMASK寄存器的介绍；对于接收队列过滤器表元素,可以参见RXFGMASK寄存器的描述。0 过滤器中相应比特位“不受影响”；1 检查过滤器中相应位。

48.3.56 报文缓冲区结构

FlexCAN 模块所使用的报文缓冲区结构如下图所示。用于 CAN 规范（版本 2.0B）的两种帧格式：扩展帧（29 位标识符）与标准帧（11 位标识符）在下表都有体现。每一个私有 MB 都由 16 个字节组成。

邮箱所有的内存空间地址范围为：0x80 到 0x47C。

31-28	27-24	23	22	21	20	19-16	15-0
无效	CODE	无效	SRR	IDE	RTR	DLC	TIME STAMP
PRIO	ID（标准/扩展）					ID（扩展）	
数据字节 0	数据字节 1			数据字节 2		数据字节 3	
数据字节 4	数据字节 5			数据字节 6		数据字节 7	

CODE——报文缓冲区码

这 4 位可以被 CPU 以及 FlexCAN 模块所访问（读和写），用来作为报文缓冲区匹配与仲裁过程的一部分。该码的解码如表 48-109 及表 48-110 所示。参见功能描述来了解附加的信息。

表 48-108 接收缓冲区报文缓冲区码字

CODE	接收到帧之前的接收码	SRV[1]	成功接收帧之后的接收码[2]	RRS[3]	注解
0b0000:非激活MB没有激活	INACTIVE	-	-	-	MB没有参与匹配过程
0b0100:空MB激活并且为空	EMPTY	-	FULL	-	如果一个帧被成功地接收（在移入过程之后，具体细节参见“移入”章节），CODE字段自动更新为FULL
0b0010:FULL-MB为满		是	FULL	-	在解锁MB（SRV）之后读C/S字不会使code返回到EMPTY状态。如果，在MB服务之后有一个新帧移到MB，该码仍然为FULL。
		NO	OVERRUN	-	如果在CPU为

					其服务之前，MB位空并且一个新帧被移入到MB，那么code自动更新为OVERRUN状态。
0b0110:OVERRUN-MB正被覆盖写入到一个满缓冲区	OVERRUN	是	FULL	-	如果CODE字段指示OVERRUN并且CPU已经处理了MB，当一个新帧移入到MB时，CODE变为FULL
		否	OVERRUN	-	如果CODE字段已经为OVERRUN，并且有另外一个新帧必须移入，那么MB将会被再次覆盖，并且code会仍然处于OVERRUN状态。至于OVERRUN状态的细节参见“匹配过程”章节。
0b1010:RANSWER[4]:一个新帧被配置为确认一个远程请求帧并且发送一个回复帧。	RANSWER	-	TANSWER（0b1110）	0	一个远程应答被配置用来识别接收到的远程请求帧，在那之后MB被设置为传输一个回复帧。Code自动改变为TANSWER（0b1110）。具体细节参见“匹配过程”章节。如果CTRL2

					寄存器的RRS位被置0，当收到同样ID的远程请求帧时发送一个应答帧。
		-	-	1	在匹配与仲裁阶段，code码将会被忽略。具体细节参见“匹配过程”章节。
CODE[0]=1b1;BUSY-FlexCAN正在更新MB的内容。CPU不允许访问MB。	BUSY[5]	-	FULL	-	表示正在更新MB，他会被自动清0并且不会干扰下一次的CODE
		-	OVERRUN	-	

1.SRV：处理 MB。MB 被读取，通过读 TIMER 或者其他的 MB 解锁。

2.如果一个帧被移入到 MB（Move-in 过程），那么该帧被认为是被成功地接收。参考“Move-in”章节。

3.来自于 CTRL2 寄存器的远程请求存储比特位。具体细节参见“控制寄存器 2”章节。

4.Code 0b1010 不被认为是发送，并且具有该 code 的 MB 不应该被丢弃。

5.注意：对于发送 MBs 来说，BUSY 应该在读的时候被忽略，除非 MCR 寄存器中 AEN 位被置位。如果该位被置位，那么相应的 MB 将不会参与到匹配过程。

表 48-110 用于发送缓冲区的报文缓冲区码字

CODE	接收帧之前的 Tx Code	MBRTR	成功发送之后的 Tx Code	描述
0b1000:INACTIVE MB 处于非激活状态	INACTIVE	-	-	MB 不参与到仲裁过程
0b1001:ABORT。MB 被丢弃	ABORT	-	-	MB 不参与到仲裁过程
0b1100:DATA-MB 为一个发送数据帧（MB 的 RTR 位必为 0）	DATA	0	INACTIVE	无条件发送一次数据帧。传输过之后，MB 自动转变为 INACTIVE 状态。
0b1100:REMOTE-MB 为一个发送远程请求帧（MB 的 RTR 位必须为 1）	REMOTE	1	EMPTY	无条件发送一次远程请求帧。在传输过之后，MB 自动以相同的 ID 转换为接收空 MB。
0b1110:TANSWER-MB 是远程请求帧的一个发送回应帧，	TANSWER	-	RANSWER	这个码为一个中间码，如果与远程请求帧相匹配的话会

				由 CHI 自动写入到 MB。远程回应帧会被无条件地发送一次,并且 code 码会自动转变为 RANSWER（0b1010）。CPU 也可以写该 code 并产生相同的影响。远程回应帧可以是一个数据帧也可以是其他远程请求帧，这取决于 RTR 的值。参见“匹配过程”和“仲裁过程”章节。

SRR——替代远程请求。固定隐形位只用于扩展帧格式。在传输时（发送缓冲区）该位必须设置为‘1，并且会将从 CAN 总线上的接收到的值存储于接收缓冲区。它可以为隐形或者显性位。如果 FlexCAN 以显性位接收，那么其解释为仲裁丢失。

1=隐形位强制用来以扩展帧模式来传输；

0=显性位对于传输扩展模式是不可用的。

IDE——ID 扩展位。该位用来表示该帧是标准帧还是扩展帧。

1=帧为扩展帧；0 帧为标准帧。

RTR——远程传输请求。该位影响远程帧的行为并且是接收过滤器的一部分。可以参考表 48-109、表 48-110 以及控制寄存器 2（CTRL2）的 RRS 比特的详细描述。

如果 FlexCAN 传输‘1’（隐性）接收‘0’（显性），这解释为仲裁丢失。如果该位传输‘0’（显性），接收‘1’（隐形），那么 FlexCAN 模块将会认为这是位错误。如果接收到的值与发送值相同，那么其将会被认为是一次成功的位传输。

1=如果 MB 是发送 MB，那么表示当前的 MB 可能有一个远程请求帧等待发送；如果 MB 是接收 MB，那么接收到的远程请求帧将会被存储起来；

0=表示当前的 MB 有一个数据帧等待传输。接收 MB 可能会被用于匹配过程。

DLC——数据的字节长度。该 4 位字段为发送/接收数据的长度，该字段位于 MB 空间的偏移地址为 0x8 到 0xf 的空间（参见表 48-108）.在接收阶段，该字段由 FlexCAN 模块写入，从接收帧的 DLC（Data Length Code）字段拷贝而得到。在传输阶段，该字段由 CPU 写入，并且与要传输帧的 DLC 字段相对应。当 RTR=1 时，被传输的帧为远程帧并且不管 DLC 字段是什么都不包括在数据字段。

TIME STAMP——自由运行计数器时间戳。该 16 位字段拷贝自自由运行计数器，在当标识符字段的开始出现在 CAN 总线时用于捕捉传输或者接收帧。

PRIO——本地优先级。这 3 位只有当 MCR 寄存器的 LPRION_EN 位置位时这三位才有用，并且只对传输邮箱有效。这些位不会被传输。他们经常被附加到 ID 来定义传输优先级。参见仲裁过程。

ID——帧标志符。在标准准格式内，只有 11 个最重要的比特（28 到 18）用来标识接收或者发送的帧。18 个最低位将会被忽略。在扩展帧格式内，所有的都用来标识传输或者接收的

数据帧。

DATA BYTE0-7——数据字段。总共 8 个字节可以被用作数据帧。对于接收数据帧，从总线上接收到的数据帧以它被接收时的格式进行存放。DATA BYTE（n）只有当 n 小于 DLC 的值时才有效。

对于传输数据帧，该字段用来表示用于发送的数据帧的长度。

表 48-111 可用的数据字节	
DLC	有效的数据字节
0	没有
1	DATA BYTE0
2	DATA BYTE0-1
3	DATA BYTE0-2
4	DATA BYTE0-3
5	DATA BYTE0-4
6	DATA BYTE0-5
7	DATA BYTE0-6
8	DATA BYTE0-7

48.3.57 接收队列 FIFO 结构

当 MCR[RFEN]位被置位，内存空间的 0x80 到 0xDC（通常被 MB0 到 MB5 占用）将会被用于接收队列 FIFO 引擎。

区间 0x80 到 0x8C 包含队列 FIFO 的输出，该队列必须被 CPU 当作报文缓冲区读取。该输出主要包含最旧的没有被读取的接收报文。区间 0x90 到 0xDC 主要为 FIFO 引擎的内部使用所预留。

另外一个内存区域，开始地址为 0xE0 并且可以扩展到最大 0x2DC（通常被 MBs6 到 MBs37 占用）这主要取决于 CTRL2[RFEN]字段的设置，包含 ID 过滤器表（可配置的从 8 到 128 个表元素）为接收到队列 FIFO 的帧制定过滤标准。

复位之后，ID 过滤器表的内存区域默认为 0xE0 并且仅仅只能扩展到 0xFC，这个区间主要包含 MBs6 到 MBs7（RFEN=0），为了向后兼容以前的 FlexCAN 版本。

下表为接收队列 FIFO 的数据结构图。

31-23	22	21	20	19-16	15-0
无效	SRR	IDE	RTR	DLC	TIME STAMP
	ID（标准/扩展）				ID（扩展）
数据字节 0		数据字节 1		数据字节 2	数据字节 3
数据字节 4		数据字节 5		数据字节 6	数据字节 7
0x90-0xDC 预留					
0xE0 ID 过滤器表元素 0					
0xE4 ID 过滤器表元素 1					
0xE8 到 0x2D4 ID 过滤器表元素 2 到 125					
0x2D8 ID 过滤器表元素 126					

0x2DC　　ID 过滤器表元素 127

每一个 ID 过滤器表元素占据一整个 32 位字并且可以被 1 个、2 个或者四个 IDAF（标记符接收过滤器）所复合，这取决于 MCR[IDAM]字段的设置。下图显示了 IDAF 索引。

下图显示了三种不同 ID 表元素的格式。注意表的所有元素必须具有相同的格式。参考接收队列 FIFO 了解更多的信息。

表 48-113 ID 表结构

31	30	29-1				0
RTR	IDE	RXIDA（标准帧=29-19，扩展帧=29-1）				无效
RTR	IDE	TXIDB_0（标准帧=29-19，扩展帧=29-1）	RTR	IDE	RXIDB_1（标准帧=13-3，扩展帧=13-0）	
RXIDC_0（标准/扩展 31-24）		RXIDC_1（标准/扩展 23-16）	RXIDC_2（标准/扩展 15-8）		RXIDC_3（标准/扩展 7-0）	

RTR——远程帧。该位表示如果远程帧匹配目标 ID 则被接收到 FIFO。

1=接收远程帧拒绝数据帧；

0=拒绝远程帧接收数据帧。

IDE——扩展帧。该位用来指示如果匹配目标 ID 扩展帧或者标准帧是否接收到队列 FIFO。

1——接受扩展帧但是拒绝标准帧；

0——拒绝扩展帧但是接收标准帧。

RXIDA——接收帧标识符（格式 A）。表示被用作队列 FIFO 接收标准的 ID。在标准帧格式中，只有 11 个最重要的比特（29-19）被用作帧标识符。在扩展帧格式内，用到了所有的比特。

RXIDB_0，RXIDB_1——接收帧标志符（格式 B）。指定一个 ID 用来作为队列 FIFO 的接受标准。在一个标准帧格式中，有 11 个最重要的比特位（一个完全标准 ID）（从第 29 到第 19 位以及从第 13 位到第 3 位）被用来作为帧的唯一标志符。在扩展帧格式中，字段的所有 14 比特都被用来和接收到的 ID 进行比较。

RXIDC_0， RXIDC_1，RXIDC_2，RXIDC_3——接收帧标识符（格式 C）。指定一个 ID 用来作为队列 FIFO 的接受标准。在标准与扩展帧格式中，该字段的所有 8 个比特用来和接受 ID 的 8 位比特进行比较。

48.4 功能描述

FlexCAN 模块具有 CAN 协议引擎，并且具有一个用于发送与接受 CAN 帧的非常灵活的邮箱系统。邮箱系统由高达 64 个报文缓冲区组成，这些报文缓冲区用于存储配置与控制数据，时间戳，报文 ID 以及数据（参见报文缓冲区结构）。和前 38 个 MBs 相对应的内存空间可以被配置成支持 FIFO 接受机制，该机制具有一个很强大的 ID 过滤机制，能够检测接受帧的 ID 表（最多可以是 128 个扩展帧 ID 或者 256 个标准帧 ID 或者 512 个 8 比特的 ID 分片），还具有高达 32 个 ID 表提供私有掩码寄存器。同时支持通过队列 FIFO 以及邮箱接收。对于邮箱接收，一个匹配算法可以使得将接受到的帧存储到 MBs 中，该 MB 的 ID 字段和接收到的帧具有相同的 ID 字段。掩码机制可以使得在 MB 中设置的 ID 号与一系列的接收到的 CAN 帧的 ID 进行匹配。对于传输，仲裁算法可以基于报文的 ID（通过本地 3 个比特的优先权字段来选择）或者 MB 的排序来决定将要发送的 MBs 的优先权。

在进行功能描述之前，必须先解释一个重要的概念。如果一个报文缓冲区即能参与匹配以又能参与仲裁过程，那么这个报文缓冲区就是激活的（“ACTIVE”）。一个具有 0b0000code 的接收 MB 是非激活的。类似的，具有 0b1000b 或者 0b1001code 的发送 MB 同样也是非激活的。

48.4.1 传输过程

为了传输一个 CAN 帧，CPU 必须通过以下步骤为传输准备一个报文缓冲区：

1.检查各自的中断标志位是否置位并将其清 0；

2.如果 MB 是激活的状态(等待传输),那么向控制与状态字的 CODE 字段写 ABORT 码(0b1001)以请求传输终止。如果开启了相应中断的屏蔽,可以通过轮询查询 IFLAG 寄存器或者中断请求，等待响应的 IFLAG 置位。然后再次读取 CODE 字段检查传输被终止了还是已经传输了。如果想实现向后兼容（MCR[AEN]位被清 0），只要想 CODE 字段写 INACTIVE 码（0b1000）就可以非激活 MB，但是没有被处理的帧可能在没有通知的情况下被传输（参见“报文缓冲区非激活”章节）。

3.写 ID 字；

4.写数据字节；

5.写控制与状态字的 DLC、控制及 CODE 字段以激活 MB。

一旦在第四步中 MB 被激活，那么其就会参与到仲裁过程并且最终会根据它的优先级被传输出去。在成功传输的最后，自由运行计数器的值会被写入到时间戳字段，控制与状态字的 CODE 字段也会被更新，CRC 寄存器被更新，中断标志寄存器的状态位被置位并且如果使能了相应的中断掩码位则会产生一个中断请求。在传输之后最新的 CODE 码会就是在第四步中被用来激活 MB 的码字。

当开启丢弃功能时(MCR[AEN]被置位)，在被配置为传输缓冲区的 MB 的中断标志位被置位时，该 MB 会被锁定，因此 CPU 不会更新它直到中断标志被 CPU 清零。这意味着 CPU 必须在开始为新的传输/接受准备 MB 之前清除相应的 IFLAG 标志。

48.4.2 仲裁过程

仲裁过程扫描整个邮箱以寻找用于下次发送的邮箱，该邮箱具有下一次将要发送的报文。这个邮箱被称之为“仲裁优胜者”。

扫描从最小邮箱号开始，然后扫描到最大邮箱号。仲裁过程由以下事件触发：

1.来自于 CAN 帧的 CRC 字段，开始点决定于 CTRL2[TASD]字段的值；

2.处于 CAN 帧的错误界定符字段；

3.处于 CAN 帧的过载定界符字段；

4.当胜利者为非激活状态并且 CAN 总线仍然没有达到 Intermission 字段的第一个比特；

5.当 CPU 写获得优先权的 MB 的控制与状态字并且 CAN 总线仍然没有达到 Intermission 字段的第一个比特；

6.如果 CHI 处于 Idle 状态并且 CPU 写任意 MB 的 C/S 字；

7.当 FlexCAN 模块退出总线关闭状态；

8.离开冻结模式或者低功耗模式。

如果在 CAN 总线达到 Intermission 字段的第一个比特之前仲裁过程没有能够完成对每一个邮箱的评估，那么临时仲裁胜利者将会无效，并且 FlexCAN 将不再在下一次竞争 CAN 总线。

仲裁过程将会根据 CTRL1[LBUF]以及 MCR[LPRIO_EN]位的设置。在扫描的结束时在所有激活的发送邮箱中选择胜利者。

48.4.2.1 最小邮箱号优先

如果 CTRL1[LBUF]为置位，那么第一个（最小号）激活的发送邮箱将获得仲裁优先权。当

CTRL1[LBUF]位置位时，MCR[LPRIO_EN]位将不会产生作用。

48.4.2.2 高优先级邮箱优先

如果 CTRL1[LBUF]位清空，那么仲裁过程将会根据最高优先权来选择激活的发送邮箱，这意味着该邮箱的帧具有比 CAN 总线上其他同时驱动每一个发送邮箱的帧的节点具有较高的优先权而赢得仲裁。

用于这次仲裁的比特序列将会被称为邮箱的仲裁值。具有最高优先权的发送邮箱是所有发送邮箱中具有最小仲裁邮箱值的。

如果两个或者两个以上的邮箱具有相同的仲裁值，那么具有最小号的邮箱将会获得仲裁优先权。

仲裁值的构成取决于 MCR[LPRIO_EN]位的设置。

48.4.2.2.1 禁止本地优先权

如果 MCR[LPRIO_EN]位被清 0，那么仲裁值的组成将会以确切的比特序列，正如他们以禁止本地优先权的方式在 CAN 帧中被传输一样。

表 48-114 当禁止本地优先权是仲裁值的构成

格式	邮箱仲裁值（32比特）				
标准（IDE=0）	标准ID（11比特）	RTR（1比特）	IDE（1比特）	-（18比特）	-（1比特）
扩展（IDE=1）	扩展ID（28：18）	SRR（1比特）	IDE（1比特）	扩展ID[17:0]	RTR（1比特）

48.4.2.2.2 开启本地优先权

如果想使用本地优先权那么 MCR[LPRIO_EN]位必须置位。在这种情况下，邮箱 PRIO 字段将会被包括在其左边的仲裁值之内（参看以下表）

表 48-115 当开启本地优先权时仲裁值的构成

格式	邮箱仲裁值（32比特）					
标准（IDE=0）	PRIO（3比特）	标准ID（11比特）	RTR（1比特）	IDE（1比特）	-（18比特）	-（1比特）
扩展（IDE=1）	PRIO（3比特）	扩展ID（28：18）	SRR（1比特）	IDE（1比特）	扩展ID[17:0]	RTR（1比特）

PRIO 字段作为仲裁值的最重要的组成部分，具有较低 PRIO 值的比具有高 PRIO 值的邮箱具有较高的优先权，不考虑他们仲裁值的其余部分。

注意：PRIO 字段不是 CAN 总线上帧的一部分。它的目的仅仅是影响内部仲裁过程。

48.4.2.3 仲裁过程（续）

一旦找到仲裁优胜者，他的内容将会被拷贝到一个称之为发送串行报文缓冲区（Tx SMB）的隐藏的辅助 MB，它和普通的 MB 具有相同的结构，但是用户不可访问。该操作称之为“Move-out”并且在此之后，相应的 MB 的写操作将会被锁定（如果 MCR 中的 AEN 为被置位）。如果发生以下

时间那么写访问操作将会被释放：

1.MB 被传输过后；

2.FlexCAN 模块进入到冻结模式或者总线关闭模式

3.FlexCAN 模块丢失总线仲裁，或者在传输时发生了错误。

在 CAN 总线上的第一个机会窗口，位于 Tx SMB 内的报文将会依据 CAN 协议规则进行传输。FlexCAN 模块最多传输 8 个数据字节，即使 DLC 字段（Data Length Code）的值远远大于该值。

下列情况将会触发仲裁过程：

1.在发送与接收帧的从 CAN CRC 字段到帧的最后，仲裁开始点取决于参数 NUMBER_OF_MB 以及 TASD 的初始化值。此外，可以改变 TASD 的值优化仲裁开始点。

2.在 CAN 总线关闭状态 TX_ERR_CNT 的值从 124 到 128。总裁开始点取决于 NUMBER_OF_MB 以及 TASD 参数的初始化值。另外，TASD 的值在仲裁开始点可能会变为最优。

3.在总线的 IDLE 状态时，C/S 的值被 CPU 写入。第一个 C/S 写操作将会启动仲裁过程并且在同一个仲裁中第二次写 C/S 将会重新开始该过程。如果执行了其他的 C/S 操作，那么仲裁过程将会被中止。如果在仲裁过程完成之后没有仲裁优胜者，那么发送仲裁机制将会开始一个全新的仲裁过程。

3.1 如果有一个没有处理的仲裁过程并且开始了 BusIdle 状态，那么将会触发仲裁过程。在这种情况下，在 BusIdle 状态下的第一次与第二次的 C/S 写将不会重新开始仲裁过程。可能是由于没有足够的时间在等待总线闲置状态来完成仲裁并且下一个状态为 Idle。在这种情况下，扫描将不会被中断，并且它将会在总线 Idle 状态下完成。在这次仲裁过程中，写 C/S 将不会导致重新开始仲裁。

4.在有效的仲裁窗口，仲裁胜利者将会失活。

5.一旦离开冻结模式（等待总线 Idle 状态的第一个比特）。如果在等待总线 Idle 状态下进行重同步，那么将会重启仲裁过程。

在下列情况下将会停止仲裁操作：

1.所有的邮箱都被扫描过；

2.在开启最小缓冲区功能的情况下找到了一个发送激活邮箱；

3.仲裁优胜者为非激活或者在任何仲裁过程中被丢弃；

4.没有足够的时间来完成发送仲裁过程。例如，在接受帧的末尾是执行失活操作。

在这种情况下，仲裁过程将会被中止。

5.总线上的错误或者过载标志；

6.在 Idl 状态下请求低功耗或者冻结模式。

仲裁将会在以下情况发生时被挂起：

1.没有及时完成仲裁过程；

2.在仲裁过程中写 C/S，如果在 MB 中执行写操作，其中 MB 的值比发送仲裁指针的值小；

3.如果没有任何发送仲裁过程而进行的 C/S 写操作；

4.发送匹配刚刚更新了一个接收 Code 或者发送 Code；

5.进入到总线停止状态。

在仲裁过程中的写 C/S 将会产生以下影响：

1.如果写 C/S 发生在仲裁优胜者中，那么将会立即重启一个新的过程；

2.如果写 C/S 发生在其值大于发送仲裁者指针的 MB 上，那么仲裁过程将会像平常一样扫描该 MB。

48.4.3 接收过程

为了能将接收到的CAN帧放入邮箱内，CPU必须执行以下步骤以准备接收：
1.如果邮箱为激活状态（发送或者接收），那么使该邮箱（参见章节“使消息缓冲区失活”）失活，最好选用一个安全的失活方法（参见“发送终止机制”）。
2.写ID字；
3.向控制与状态字的CODE字段写EMPTY码（0b0100），以激活该邮箱；
一旦MB被激活，那么它将能够接收匹配过滤器的帧。在成功接收帧的末尾，邮箱将会按照以下方式被“move-in”过程所更新：
1.接收到的数据字段（最多8个字节）被存储；
2.接收到的标识符字段被存储；
3.自由运行计时器的值在帧标识符字段的第二个比特时被写入到邮箱的时间戳字段；
4.接收到的SRR、IDE、RTR以及DLC字段被存储；
5.控制与状态字的CODE字段被更新(参见“消息缓冲区结构”章节的表48-109和表48-110)；
6.中断标志寄存器的状态为被置位,并且如果中断掩码寄存器的相应位允许那么会产生一个中断；
建议CPU接收帧到邮箱的过程如下：
1.读邮箱的控制与状态字；
2.检查BUSY是否被清零，表示邮箱已被锁定。重复步骤1直到该位被置位，参见“报文缓冲区锁定机制”；
3.读邮箱的内容,一旦邮箱被锁定那么它的内容将不会被FlexCAN模块的Move-in过程所修改，参见章节“Move-in”；
4.置IFLAG寄存器的相应位。
5.读自由运行计数器，这是可选的，但是建议尽快解锁邮箱并且使得其可以用于接收。
CPU应该通过特定邮箱的IFLAG寄存器的状态标识,与接受到的帧同步,而不是基于邮箱的CODE字段。轮询CODE字段并不能起到作用因为一旦接收到帧，那么CPU会立即处理该邮箱（在解锁邮箱之后读取C/S字),CODE字段并不会转变为EMPTY。它将仍然是FULL状态,正如表48-109所描述的那样。如果CPU想在没有安全失活读取邮箱之后，通过强制向C/S字写EMPTY码改变这种状况，一个最新接收到的并且匹配其邮箱过滤器的帧将会丢失。
注意：不要通过直接读邮箱的C/S字，而要读取IFLAG寄存器。

注意接收到的帧的标志符字段总是存储在匹配的邮箱中，因此如果匹配被屏蔽，在邮箱中的ID字段的内容可能会改变。同时也要注意如果存在一个匹配的接收邮箱，FlexCAN模块可以自己来传输帧，只要MCR[SRXDIS]位没有被置位。如果MCR[SRXDIS]为置位，那么FlexCAN将永不会在任何MB中自己存储被传输的帧，即使其包含一个匹配的MB，并且没有中断标志或者不会因为接受帧而产生中断信号。

为了能够通过接收队列FIFO接受CAN帧,CPU必须在冻结模块使能并且配置接受队列FIFO。只要接收到的帧在接收队列FIFO中断中可用（参考IFLAG[BUF5I]的表述，“在接收队列中可用的帧”帧），CPU可以以下面的过程来处理接收到的帧:
1.读控制与状态字（可选的——只有IDE及RTR位使用掩码时才是必须的）；
2.读ID字段（可选的——仅仅在使用掩码时才是必须的）；
3.读数据字段；
4.读RXFIR寄存器（可选）；

5.通过向 IFLAG[BUF5I]位来写 1 清除在接收队列中可用的帧(必须的——释放 MB 并且允许 CPU 读取下一个接收队列 FIFO 实体)。

48.4.4 匹配过程

匹配过程扫描 MB 内存以寻找具有和从 CAN 总线上接收到帧相同 ID 的接收 MBs。如果 FIFO 被使能，可以在邮箱和 FIFO 过滤器中选择扫描的优先级。在任何情况下，匹配将从具有最小号的报文缓冲区向最大号的报文缓冲区进行。如果在第一个结构中没有找到匹配的，那么随后将扫描下一个结构。如果 FIFO 队列满，那么匹配算法会在 FIFO 区域之外选择匹配的 MB。

如果正在接受帧，那么它将会被存储在一个隐藏的附加 MB，该 MB 称之为接收串行报文缓冲区（Rx SMB)。

匹配的开始点取决于以下条件：

1.如果接收到的帧为远程帧，那么开始点位帧的 CRC 字段；

2.如果接受到的帧为数据帧并且其 DLC 字段的值为 0，那么开始点为帧的 CRC 字段；

3.如果接受到的帧为数据帧并且其 DLC 字段的值不为 0，那么开始点为帧的 DATA 字段；

如果在队列 FIFO 表或者邮箱中找到一个匹配的 ID，那么 SMB 的内容会被传入到 FIFO 或者通过 move-in 过程来匹配邮箱。如果检测到任何 CAN 协议的错误，那么不会有任何的匹配结果被传入到 FIFO 或者在接受结束时传入匹配的邮箱。

匹配过程扫描所有的接收队列 FIFO（如果开启）以及激活的接收邮箱（CODE 字段为 EMPTY,FULL,OVERRUN 或者 RANSWER)来寻找从 CAN 总线上接收到的 SMB 的匹配元素的一次成功比较。SMB 和邮箱具有相同的结构。接受结构（接受队列 FIFO 或者邮箱）与具有一次成功比较的匹配元素称之为“匹配结构”。在扫描所有这些匹配结构的最后将会选择出匹配成功者并且这取决于前面所描述的条件。参考下表。

表 48-116 匹配架构

结构	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID]	MB[CODE]
邮箱	0	-	0	cmp	no-cmp	cmp_msk	EMPTY、FULL、OVERRUN
邮箱	0	-	1	cmp_msk	cmp-msk	cmp-msk	EMPTY、FULL、OVERRUN
邮箱	1	0	-	cmp	no_cmp	cmp	RANSWER
邮箱	1	1	0	cmp	no_cmp	cmp_msk	EMPYT、FULL、OVERRUN
邮箱	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY、FULL、OVERRUN
FIFO	-	-	-	cmp_msk	cmp_msk	cmp_msk	-

1.对于邮箱结构，如果 SMB[IDE]置位，ID 字段为 29 比特（ID 标准+ID 扩展)。如果 SMB[IDE]为清空，那么 ID 只有 11 位（ID 标准)。对于 FIFO 结构，ID 长度取决于 IDAM。

2.cmp:将 SMB 的内容与 MB 的内容进行比较而不管屏蔽位；
3.no_cmp:不将 SMB 的内容与 MB 的内容比较并且不考虑掩码；
4.cmp_msk:将 SMB 的内容与 MB 的内容进行比较并且考虑相应的掩码位；
5.当 IDAM 的类型为 C 时，不考虑 SMB[IDE]与 SMB[RTR]。
当满足以下条件中的任何一个时，接受结构为 free-to-receive：
1.邮箱的 CODE 字段为 EMPTY；
2.邮箱的 CODE 字段是 FULL 或者 OVERRUN，并且该邮箱已经被处理过（C/S 字已经被 CPU 读取过并且已经被解锁）；
3.邮箱的 CODE 字段为 FULL 或者 OVERRUN，并且该邮箱已经失活；
4.接受队列不满。

扫描邮箱及接收队列 FIFO 的顺序为从具有最小号的待匹配元素到最大号的待匹配元素。

对于邮箱匹配优胜者受 MCR[IRMQ]为的影响。如果该位为 0，那么匹配优胜者为第一个被匹配的邮箱而不管它是不是 free-to-receive。如果该位置位，那么匹配优胜者通过以下优先级来选择：

1.第一个匹配邮箱的 free-to-receive；

2.最后一个匹配邮箱的非 free-to-receive；

可以通过 CTRL2[MRP]位选择扫描的优先级为邮箱优先或者接收队列 FIFO 优先。如果选择的优先级是接收队列 FIFO 第一：

1.如果接收队列 FIFO 是匹配结构并且是 free-to-receive，那么接收队列 FIFO 就是匹配优胜者而不再扫描邮箱；

2.不然（接收队列 FIFO 不是匹配结构或者不是 free-to-receive），则匹配优胜者是在上面所描述的邮箱中找到的。

如果选择的优先级是邮箱优先：

1.如果匹配邮箱的 free-to-receive 被找到,那么它就是匹配优胜者而不再扫描接收队列 FIFO；

2.如果没有找到匹配的邮箱，那么匹配优胜者将会在接收队列 FIFO 中寻找。

如果上面的两个情况都不满足并且没有找到一个匹配邮箱的 free-to-receive，那么匹配优胜者将由 MCR[IRMQ]位来决定：

如果 MCR[IRMQ]位为 0，那么匹配优胜者为第一个匹配的邮箱；

如果 MCR[IRMQ]位为 1，那么如果有一个 free-to-receive 匹配结构，匹配优胜者为接收队列 FIFO，不然匹配胜利者为最后一个非 free-to-receive 匹配邮箱。

以下为所有匹配可能的一个总结表。

表 48-117 匹配的可能性及由此产生的接收结构						
RFEN	IRMQ	MRP	在 MB 中匹配	在 FIFO 中匹配	接收结构	描述
没有 FIFO，只有 MB，总是从 MB 开始匹配						
0	0	X	None	-	None	因为没有匹配而丢失帧
0	0	X	Free	-	FirstMB	
0	1	X	None	-	None	因为没有匹配而丢失帧
0	1	X	Free	-	FirstMB	
0	1	X	NotFree	-	LastMB	Overrun
开启 FIFO，在 FIFO 中不匹配正如 FIFO 不存在						
1	0	X	None	None	None	因为没有匹配而丢失帧
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	因为没有匹配而丢失帧
1	1	X	Free	None	FirstMB	
1	1	X	NotFree	None	LastMB	Overrun
开启 FIFO，但是禁止队列						
1	0	0	X	NotFull	FIFO	
1	0	0	None	Full	None	由于 FIFO 满而丢失帧(FIFO 溢出)
1	0	0	Free	Full	FirstMB	
1	0	0	NotFree	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	由于 FIFO 满而丢失帧(FIFO 溢出)
1	0	1	Free	X	FirstMB	
1	0	1	NotFree	X	FirstMB	Overrun
开启 FIFO，开启队列						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	由于 FIFO

						满而丢失帧(FIFO溢出)
1	1	0	Free	Full	FirstMB	
1	1	0	NotFree	Full	LastMB	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	NotFree	NotFull	FIFO	
1	1	1	NotFree	Full	LastMB	Overrun

1,。不是必备条件

2.在 MB 中的匹配“None”意味着帧没有被任何 MB 所匹配（free-to-receive 或者 non-free-to-receive)。

3.表示禁止条件;

4.在MB的匹配中“Free”意味着该帧与至少一个MBfree-to-receive匹配而不管其是否与非 free-to-receive 匹配。

5.在FIFO中的“None“表示该帧没有被FIFO中的任何过滤器所匹配，就好象FIFO不存在一样（CTRL2[RFEN]=0);

6.在FIFO中的“NotFull”意味着该帧已经和FIFO的过滤器相匹配并且有空余的间隙来接收它;

7.在FIFO中的“Full”意味着该帧已经和FIFO的过滤器相匹配，但是因为没有空余间隙接收而不能将其存储起来。

如果不安全的邮箱失活操作（参见“消息缓冲区失活“）发生在匹配过程中，失活的邮箱是暂时的匹配优胜者那么这个临时匹配优胜者将会变为非法的。匹配元素的扫描不会停止并且也不会重新启动，它将继续进行。结果是当前的匹配过程正常进行就像在去激活不存在之前的匹配元素比较一样，因此可能会丢失一个报文。

例如假设禁止了FIFO，开启了 IRMQ 并且有两个 MBs 具有相同的 ID，那么 FlexCAN 模块将会以该 ID 开始接收报文。这些 MBs 是数组中的第 2 个和第 5 个。当第一个报文到达时，匹配算法会找到第一个与值相匹配的 2 号 MB。该 MB 的 code 为 EMPTY，因此该报文会存放在那里。当第二个报文到达时，匹配算法会又找到 2 号 MB，但是它不为“free-to-receive”，因此它将继续寻找并且找到了5号MB然后将报文存储在那里。如果具有相同ID号的其他报文又到达，匹配算法会发现已经没有可匹配的的并且是“free-to-receive”MBs 了，因此它将决定覆盖最后一个匹配的MB也就是5号MB。这样做了之后，它将设置该MB的CODE字段为OVERRUN。

在多于一个的MB中匹配具有相同ID的能力可以被解释为实现了一个接收队列(除了全功能的FIFO）用来允许CPU可以处理多次MB。通过编程设置多于一个的具有相同 ID 的 MB，接收到的报文必须排队进入到MBs。CPU可以测试MBs的时间戳来决定接收到的报文的顺序。

通过使用 ID 接收掩码使得匹配一系列的 ID 成为可能。FlexCan 模块的每一个 MB 都支持私有掩码。可以参见接收队列私有掩码寄存器（RXIMRx）的描述。在匹配算法阶段，如果掩码置位，那么将比较相应的ID位。如果掩码为0，那么将不比较相应的ID位。请注意私有掩码寄存器位于RAM中，因此他们在复位之后不会被初始化。同时他们只有在模块处于冻结模式时被编程，不然会被硬件锁定。

FlexCAN 同时也支持一个可以替代的掩码机制，并且只需要四个掩码寄存器（RGXMASK, RX14MASK, RX15MASK 以及 RXFGMASK)主要是为了做到向后兼容。当MCR寄存器的IRMQ为被清

0 时，那么将开启替换掩码机制。

48.4.5 移动过程

一共有两种类型的移动过程：移入（move-in）与移出（move-out）。

48.4.5.1 移入（Move-in）

移入过程是将由 Rx SMB 接收到的报文拷贝到与之想匹配的接收邮箱或者接收 FIFO。如果移动的目的地是接收队列 FIFO，那么报文的属性也会被拷贝到 RXFIR FIFO 内。每一个 Rx SMB 都有自己的移入过程,但是如前面描述的那样在给定的时间内只能有一个执行自己的移入操作。只有当拥有报文的 Rx SMB 具有相应的匹配优胜者并且以下条件都为真时才可以开始移入操作：

1. CAN 总线已经达到或者已经让位于 Rx SMB 的携带报文的帧的 Intermission 字段的第 2 个比特；通过或者让位于 Rx SMB 中携带报文的帧之后的一个过载帧的第一个比特；

2. 没有正在执行的匹配过程；

3. 目标邮箱没有被 CPU 锁定；

4. 没有正在进行的从其他 Rx SMB 移入的过程。如果有多于一个移入过程准备在同一时间开始，两个都可以操作并且最新的替代最老的。

整个文档所出现的短语“pending move-in”代表一个没有全部满足上面所给定的条件的将要开始移入的操作。

移入过程将会被取消并且如果能够满足以下任意条件的 Rx SMB 都可以接收其他的报文：

1. 在 CAN 总线到携带报文帧的 Intermission 字段的第一个比特之后目的邮箱为非激活状态并且它的匹配过程已经完成；

2. 先前有一个没有完成的移入过程并且其具有相同的目的邮箱；

3. Rx SMB 正在接受由 FlexCAN 模块传输的帧，并且自我接受功能被禁止（MCR[SRXDIX]位置位）；

4. 检测到任何 CAN 协议错误。

注意如果模块进入到冻结或者低功耗模式那么将不会取消没有完成的移入过程。它将会等待退出冻结及低功耗模式并且等待被解锁。如果 MB 在冻结模式下被解锁，那么会立即开始移入过程。

移入过程将会由 FlexCAN 模块按照以下步骤执行：

1. 如果报文的目的地是接收 FIFO，将 IDHIT 移入到 RXFIR 队列 FIFO；

2. 从 Rx SMB 读取字 DATA0-3 及 DATA4-7；

3. 将其写入到接受邮箱的 DATA0-3 及 DATA4-7；

4. 从 Rx SMB 读取控制/状态及 ID 字；

5. 将其写入到接收邮箱的控制/状态字及 ID 字，并更新 CODE 字段。

移入过程并不是自动进行的，该过程会由于目的邮箱的失活而被立即取消，并且在这种情况下可能导致邮箱的部分内容被更新从而出现不一致性。例外的是当移入过程的目的地址为接收报文缓冲区时，该过程不能被取消。

当正在进行移入操作时，目标报文缓冲区的 BUSY 位（CODE 字段最重要的一个比特）置位，用来警告 CPU 报文缓冲区的内容会暂时的不一致。

48.4.5.2 移出（Move-out）

移出过程是当传输的报文有效时，将报文内容从发送邮箱拷贝到 Tx SMB 的过程（参见“仲裁过程”）。移出过程发生在以下条件中：

1. Intermission 字段的第一个比特；
2. 在总线关闭时当 Tx 错误计数器的值位于 124 到 128 之间；
3. 处于总线闲置的时候；
4. 处于等待总线闲置的时候。

移出过程不是自动完成的。只有当 CPU 具有访问内存的优先权同时退出了总线限制状态。在总线限制状态中，移出过程具有最低的内存访问优先权。

48.4.6 数据一致性

为了维持数据的一致性以及 FlexCAN 模块的正确操作，CPU 必须遵守“传输过程”与“接受过程”所描述的规则。CPU 以任何其他没有指明的方式访问 FlexCAN 模块内的 MB 结构时，都有可能导致 FlexCAN 模块处于一个不可预知的结果。

48.4.6.1 传输终止机制

终止机制提供了一种安全的方式请求终止一个不能完成的传输反馈机制用来通知 CPU 传输是否终止或者帧是否能被终止或者该帧被传输了。

为了终止传输必须满足两个主要的条件：

1. MCR 的 AEN 位置位；
2. CPU 必须向控制与状态字的 CODE 字段写入一个具体的终止码（0b1001）。

被配置用于传输的激活的 MB 第一次必须被终止然后更新它们。如果一个当前正在传输的邮箱被写入终止码，或者向一个已经装载到 SMB 用于传输的邮箱中写终止码，那么写操作将会被锁定并且 MB 继续保持激活，但是会捕获到终止请求并且保持未解决直到满足以下条件：

1. 模块失去总线仲裁；
2. 在传输过程中发生了一个错误；
3. 模块进入冻结模式；
4. 模块进入总线关闭状态；
5. 有一个过载帧。

如果上面所提到的条件都没有满足，那么 MB 会正确地传输，IFLAG 寄存器的中断标志位会置位并且如果使能了中断会向 CPU 产生一个中断。当中断标志为被置位时终止请求会被自动清除。另一方面，如果满足了以上条件中的任何一个，帧将不会被传输；因此，终止码会被写入到 CODE 字段，IFLAG 寄存器的中断标志置位并且向 CPU 产生一个中断。

如果在传输开始之前 CPU 写终止码，那么写操作将会被锁定；因此，MB 被更新并且中断标志位置位。在这种方式下 CPU 仅仅需要读取终止码以确定激活的 MB 被安全的失活。尽管 AEN 位被置位并且 CPU 写了终止码，在这种情况下 MB 处于失活状态并且不会终止，因为传输并没有开始。当捕获到一个终止请求时仅有一个邮箱会被终止，并且保持未解决状态直到满足了前面条件中的任何一个。

终止操作的过程总结如下：

1. CPU 检查相应的 IFLAG 位如果置位那么清除该位；
2. CPU 向 C/S 字的 CODE 字段写 0b1001；

3.CPU 等待读相应的 IFLAG 以了解帧是被传输了还是被终止了；

4.CPU 读取 CODE 字段以检查该帧是被传输了（CODE=0b1000）还是被终止了（CODE=0b1001）；

5.有必要清除相应的 IFLAG 标志，以允许 MB 可以被重新配置。

48.4.6.2 报文缓冲区失活

提供失活机制是为了保护邮箱不会被 FlexCAN 内部程序所更新，因此在更新过之后，即使在正常模式下，允许 CPU 依赖于邮箱数据的一致性。

当 MCR 寄存器的 AEN 位重新置位时传输邮箱必须失活。

如果一个邮箱失活，那么它既不会参与仲裁过程也不参与匹配过程直到它被重新激活。参见章节“传输过程”及“接收过程”来了解关于失活及重新激活一个邮箱的详细细节。

为了使邮箱失活，CPU 必须更新其 CODE 字段为 INACTIVE（0b0000 或者 0b1000）。

如果用户不能通过 FlexCAN 模块内部处理失活操作来同步 CODE 字段，可能导致一个不愿看到的结果：

1.在总线上匹配失活接收邮箱过滤器的帧可能会因为没有关注而丢失,即使有具有相同过滤器的其他邮箱存在；

2.位于失活邮箱中包含报文的帧可能会因为没有注意而被传输。

为了消除这样的风险并且执行安全的失活操作，CPU 必须使用下列机制来失活：

对于发送邮箱，传输终止（参看章节“传输终止机制”）；

失活会自动锁定邮箱（参见章节“报文缓冲区锁定机制”）。

注意：作为接收队列 FIFO 一部分的报文缓冲区不能被失活，在 FIFO 区域不会被 FlexCAN 模块写保护。当 RFEN 置位时，CPU 必须维持在 FIFO 区域的数据一致性。

48.4.6.3 报文缓冲区锁定机制

除了 MB 的失活，FlexCAN 模块还有另外一个用于维护接收操作数据一致性的机制。当 CPU 读 CODE 码为 FULL 或者 OVERRUN 的接收 MB 的控制与状态字时，FlexCAN 模块假定 CPU 想以自动的方式读取整个 MB，因此为那个 MB 设置内部锁定标志。当 CPU 读取自由运行计数器（全局解锁操作）时，或者当 CPU 不考虑其他的 MB 的 CODE 码而读取其控制与状态字时，或者当 CPU 向锁定的 MB 中写入 C/S 字时，会释放锁定。MB 的锁定主要是为了避免在 CPU 读取某个 MB 时一个新帧写入到 MB 中。

注意：锁定机制只用于接收 MB，该 MB 不是 FIFO 队列的一部分并且它的 CODE 码不同于 INACTIVE（0b0000）或者 EMPTY（0b0100）。并且发送 MB 不能被锁定。

例如，假定 FIFO 被禁止并且数组的第二个与第五个 MB 被编程设置为具有相同的 ID，并且 FlexCAN 已经接收并且将接收到的报文存储在这两个 MB 中。假设现在 CPU 决定读取 5 号 MB 与此同时具有相同 ID 的另外一个报文到达。当 CPU 读取 5 号 MB 的控制与状态字时，该 MB 被锁定。新报文到达并且匹配算法发现已经没有“free-to-receive”的 MB 了，因此，它将决定覆盖 5 号 MB。虽然此 MB 是锁定的，因此新的报文不会被写在那里。它会一直呆在 SMB 中直到 MB 被解锁，然后该报文会被写入到 MB 中。如果 MB 没有及时被解锁并且另一个具有相同 ID 的报文到达,那么新的报文会覆盖掉在 SMB 中的报文并且在 MB 的 CODE 字段及错误与状态寄存器中不会有报文丢失的提示。

当一个报文被从 SMB 移入到 MB 时，CODE 字段的 BUSY 为会置位。如果 CPU 读取控制与状态字并且发现 BUSY 位被置位，那么它应该延缓访问 MB 直到 BUSY 被清 0.

注意：如果 BUSY 被置位或者 MB 为空，那么读控制与状态字并不会锁定 MB。

失活优先于锁定。如果 CPU 失活一个被锁定的接收 MB，那么它的锁定状态将会被忽略并且 MB 会在本轮的匹配中被标记为无效。在 SMB 中任何未被处理的报文不会被传入到任何的 MB 中。当 CPU 读自由运行计数器或者另外一个 MB 的 C/S 字时一个 MB 不会被锁定。

锁定与解锁机制在正常与冻结模式下具有相同的功能。

在正常或冻结模式下的锁定会导致未处理报文的移入。如果锁定操作发生在任何的低功耗模式下，那么移入操作会被延后并且这种情况只会在模块恢复到正常模式或者冻结模式时才会发生。

48.4.7 接收队列 FIFO

通过将 MCR 寄存器的 RFEN 位置位可以使能只用于接收的队列 FIFO。复位值为 0，这是为了软件兼容以前的版本，因为以前的版本没有队列功能。FIFO 队列可以具有 6 个报文的深度，因此当使能 FIFO 时，那么内存空间会预留 6 个报文缓冲的区间以便使用 FIFO 引擎。通过在 FIFO 的输出接口重复地读取报文缓冲区结构，CPU 可以按照报文被接收到的顺序读取报文。

当 FIFO 队列中有至少一个可读的帧时，IFLAG[BUF5I]（在接收 FIFO 队列中帧可用）标志位被置位。通过使能相应的掩码会产生一个中断。一旦接收到中断，CPU 可以读取报文（以报文缓冲区的方式访问 FIFO 队列的输出）、RXFIR 寄存器然后清除该中断。如果在 FIFO 中有多于一个的报文，通过清中断操作将会以下一个报文来更新 FIFO 输出并且会根据报文的属性来更新 RXFIR 寄存器，再次 CPU 发出中断。不然，标志位会被清除。只有当 IFLAG[BUF5I]位被置位时 FIFO 输出才有效。

当接收队列中没有读取的报文数由于接收到一个新的报文而从 4 增加到 5 时，IFLAG[BUF6I]（接收队列警告）会被置位，这意味着接收队列 FIFO 将要满了。该标志位会一直置位直到 CPU 清除它。

当因为队列满而导致新来的报文丢失时 IFLAG[BUF7I]位会被置位。当接收队列满并且该报文被邮箱所捕获时该位不会被置位。该标志位会一直置位直到 CPU 清除它。

清除这三个标志位中的任何一个都不会对其他两个产生影响。

如果 IFLAG 位置位并且相应的掩码位也被置位时会产生一个中断。

提供了一个非常高效的过滤机制只为目标应用接收帧，因此将减少中断服务的工作量。过滤标准通过设置一个高达 128 个 32 位寄存器来制定，通过 CTRL2[RFEN]的设定可以设置为以下的格式：

1. 格式 A：128 个 IDAFs（包括 IDE 及 RTR 的标准或者扩展 ID）；
2. 格式 B：256 个 IDAFs（包括 IDE 及 RTR 的标准 ID 以及扩展的 14 位 ID）；
3. 格式 C：512 个 IDAFs（标准或者扩展 8 比特的 ID 分片）。

注意：一个可选的格式可以应用于过滤表的所有实体。不可能在表中出现混合格式。

在 FIFO 队列中每一个可用的帧都有相应的 IDHIT（标志符可接受过滤命中指示符），该 IDHIT 可以通过访问 RXFIR 寄存器来读取。RXFIR[IDHIT]字段指示了在 FIFO 队列输出的报文，并且当 IFLAG[BUF5I]位置位时该字段是有效的。必须在清除该位之前读取 RXFIR 寄存器，RXFIR 寄存器会给在 FIFO 队列中的帧提供相关信息。

过滤表中高达 32 个元素分别受私有掩码寄存器（RXIMRx）的影响，通过 CTRL2[RFEN]位的设置，允许定义非常高效的过滤标准。如果 IRMQ 位被清 0，那么 FIFO 过滤器表会受到 RXFGMASK 的影响。

48.4.8 CAN 协议相关特征

该节主要介绍 CAN 协议的相关功能。

48.4.8.1 远程帧

远程帧是一种很特别的帧。用户可以通过将邮箱传输时的 RTR 位设为“1”来编程一个邮箱为远程请求帧。远程请求帧发送成功之后，邮箱变为了具有相同 ID 的接收报文缓冲区。

当 FlexCAN 模块接收到一个远程请求帧时，可以对其采取三种处理方式，这取决于远程请求存储（CTRL2[RRS]）以及接收队列 FIFO 使能位（MCR[RFEN]）：

1.如果 RRS 为 0,那么帧的 ID 将与 CODE 字段为 0b1010 的传输报文缓冲区的 ID 进行比较。如果有一个匹配的 ID，那么该邮箱将会被传输。注意如果匹配的邮箱 RTR 位被置位，那么 FlexCAN 将会传输一个远程帧作为回应。接收到远程请求帧将不会存放在接收缓冲区。它仅仅用来触发回应帧的传输。掩码寄存器不会用于远程帧的匹配，并且接收到帧的所有 ID 位（除了 RTR）将会被匹配。在接收到远程请求帧并且匹配了一个邮箱的情况下，报文缓冲区立即进入到内部仲裁过程。但是它将会被认为是正常的没有高优先权的发送邮箱。该帧的数据长度与已初始化的传输过程远程帧的 DLC 字段无关。

2.如果 RRS 位置位，那么帧的 ID 将会与 CODE 字段为 0b0100, 0b0010 或 0b0110 的接收邮箱的 ID 进行比较。如果有匹配的 ID，那么该邮箱将会存储以与数据帧相同的格式来存储远程帧。不会自动产生远程回应帧。掩码寄存器用于匹配过程。

3.如果 RFEN 位置位，FlexCAN 将不会为匹配 FIFO 队列过滤标准的远程请求帧产生自动回复帧。如果远程帧匹配目标 ID 中的任何一个，那么它将会被存储在 FIFO 队列中并且提交给 CPU。注意，对于过滤格式 A 和 B，可以选择是否接受远程帧；对于格式 C，总是接收远程帧（如果匹配了 ID）。远程请求帧被认为是正常帧，并且当成功完成接收队列 FIFO 满时，会产生一个 FIFO 队列溢出。

48.4.8.2 过载帧

当在 CAN 总线上检测到以下条件时 FlexCAN 模块将会发送过载帧：

1.在间隙的第一个和第二个比特见到 1 个‘0‘位；

2.在帧字段（接收帧）的第 7 个比特（最后一位）检测到 1 个‘0‘位；

3.在错误定界符或者过载界定符的第 8 位（最后一位）检测到 1 个‘0‘位；

48.4.8.3 时间戳

自由运行计数器的值为 CAN 总线上标志符字段开始时的采样值,并且该值会在移入过程的最后存储在 TIME STAMP 字段，为网络行为提供时间参考。

注意,如果接收到一个特定帧,那么自由运行计数器可以被复位,从而使能网络时间同步。可以参考控制寄存器 1（CTRL1）TSYN 的描述。

48.4.8.4 协议时序

下图显示了时钟产生电路的结构图，产生的时钟供给 CAN 协议引擎（PE）子模块。CTRL1 寄存器的时钟源位 CLKSRC 用来定义内部时钟是连接到外部晶振（振荡器时钟）还是连接到外设时钟（通过 PLL 电路产生）。为了保证运行的可靠性，必须在禁止模式（模块配置寄存器的

MDIS 位）下来选择时钟源。

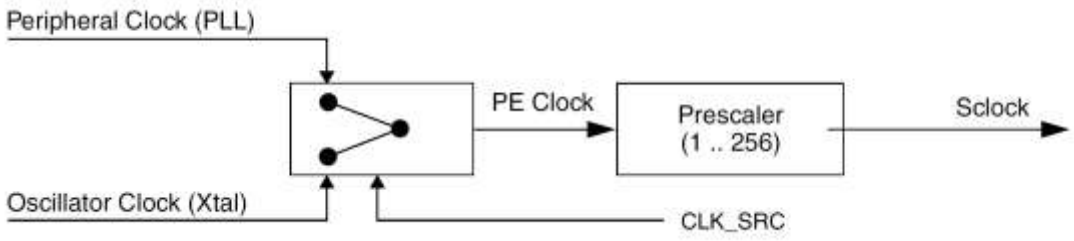

图 48-104 CAN 引擎时钟图

当 CAN 总线时序要求精度最高 0.1%时应该选择晶体振荡器时钟。晶体振荡器产生的时钟的抖动性能应该优于 PLL 所产生的时钟。

FlexCAN 模块支持不同的方式来建立 CAN 协议所要求的位时序参数。控制寄存器有不同的字段用来控制为时序参数：PRESDIV, PROPSEG, PSEG1, PSEG2 以及 RJW。可以参见控制寄存器 1（CTRL1）的描述。

PRESDIV 字段用来控制串行时钟（Sclock）的预分频。它的周期定义了“最小时间份额”（Time Quanta）用于组成 CAN 总线的波形。一个最小时间份额是可以被 CAN 引擎所能处理的最小时间单位。

$$f_{Tq} = \frac{f_{CANCLK}}{(\text{Prescaler Value})}$$

一个位时间可以被分为三个部分（参考图 48-105 及表 48-118）；

1.SYNC_SEG：同步段，该段有一个时间份额的固定长度。电平信号边缘出现在该段内；

2.时间段 1：该段包括 CAN 标准的传播段以及相位段 1。可以通过 CTRL1 寄存器的 PSEG1 字段以及 PROPSEG 字段来设置其所占时间份额，因此他们的时间总和为 4 到 16 个最小时间份额；

3.时间段 2:该段代表 CAN 标准的相位段 2.可以通过 CTRL1 寄存器的 PSEG2 字段来设置该段所占的时间份额，为 2 到 8 个不同的时间份额长。

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

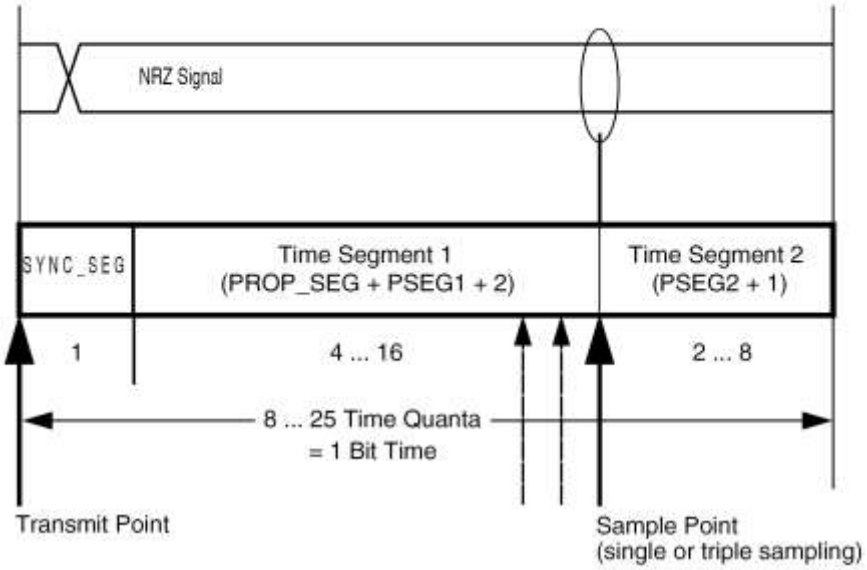

图 48-105 位时间内的段

每当用 CAN 位时间来作为持续时间的衡量标准时，CAN 比特位的外设时钟表示可以按照以下方式来计算：

$$NCCP = \frac{f_{SYS} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)] \times (PRESDIV + 1)}{f_{CANCLK}}$$

其中，

NCPP 是外设时钟的数值，以 CAN 位时间为单位；

f_{CANCLK} 为协议引擎时钟（PE），单位 Hz；

fSYS 为系统时钟的（CHI）频率，单位 Hz；

PSEG1 为 CTRL1[PSEG1]字段的值；

PSEG2 为 CTRL1[PSEG2]字段的值；

PROPSEG 为 CTRL1[PROPSEG]字段的值；

PRESDIV 为 CTRL1[PRESDIV]字段的值。

例如：180 个 CAN 比特时间=180×NCCP 外设时钟周期。

表 48-118 时间段语法

语法	描述
SYNC_SEG	在该周期内系统期望发生传输过程
传输点	在该点，在传输模式下一个节点传输一个新值到CAN总线上。
采样点	节点在该采样总线。如果每个比特选择使用三次采样，那么该点位于第三个采样点。

下表给出了一个 CAN 兼容的设置及相关参数值的总结。

表 48-119 CAN 标准兼容位时间段设置

时间段1	时间段2	重新同步跳跃宽度
5-10	2	1-2
4-11	3	1-3
5-12	4	1-4
6-13	5	1-4
7-14	6	1-4
8-15	7	1-4
9-16	8	1-4

注意：用户应该确保位时间的设定满足 CAN 标准。对于位时间的计算，可以使用二分之一的 IPT（信息处理时间），该值完全满足 FlexCAN 模块。

48.4.8.5 仲裁与匹配时序

在一个帧的正常传输与接收期间，匹配、仲裁、移入及移出过程将在 CAN 帧的某个时间窗口被执行，正如下图所示。

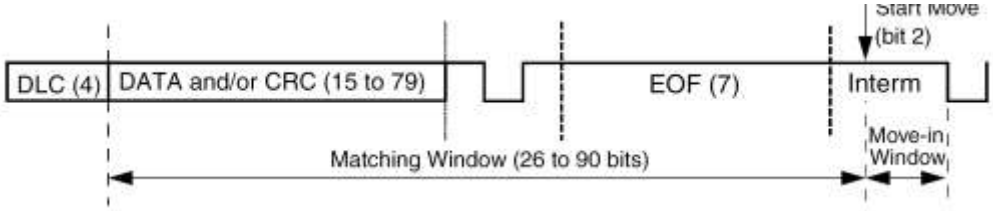

图 48-106 匹配及移入时间窗口

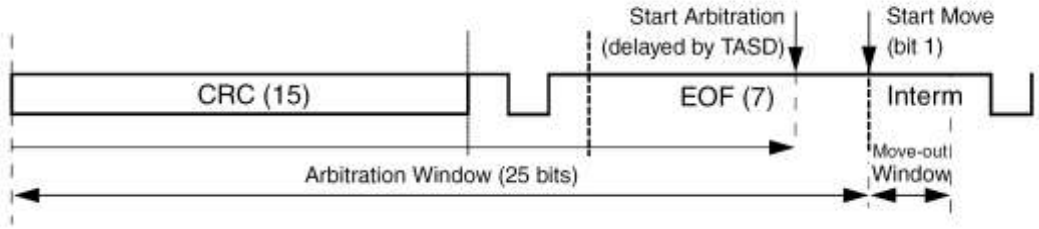

图 48-107 仲裁及移出时间窗口

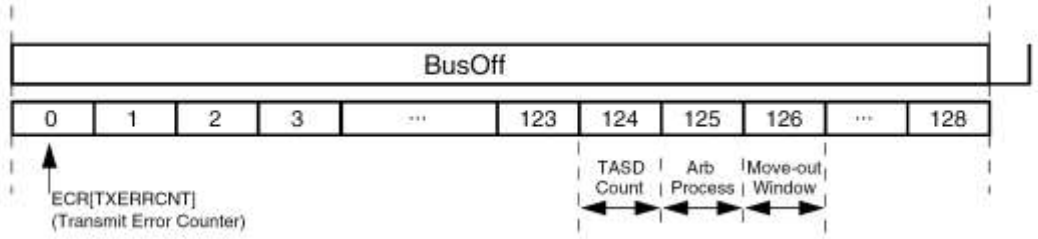

图 48-108 在总线关闭末的仲裁及移出时间窗口

注意:在上图中的匹配及仲裁时序不应该考虑 CPU 及其他内部外设同时访问当前内存引起的延迟。

当正在匹配和仲裁时，FlexCAN 应该在可用的时间片内扫描整个报文缓冲区内存。为了有充足的时间来做这个事情，必须遵守下列要求：

1. 一个有效的 CAN 位时间必须是可编程的，如表 48-119 所示；
2. 外设的时钟频率不应该小于晶振时钟频率，例如 PLL 不能设置为晶振时钟的 2 分频；
3. 在外设时钟频率与 CAN 比特率之间必须有一个最小的比率，如下表所示，

表 48-120 外设时钟频率与 CAN 比特率的最小比率

报文缓冲区个数	RFEN	每个CAN比特最小的外设时钟
16-32	0	16

64	0	25
16	1	16
32	1	17
64	1	30

第一个要求的直接后果是：每个 CAN 比特的最小时间份额为 8，因此晶振时钟频率应该至少为 CAN 比特率的 8 倍。当相比于晶振时钟频率或者通过调整一个或者更多个比特时间参数（PRESDIV, PROPSEG, PSEG1, PSEG2），上表中所提到的最小频率比率可以通过选择一个足够高的外设时钟来频率达到。

在进行同步操作的时候（当外设时钟频率等于晶振频率），为了达到上表中的需求，每个 CAN 位的外设时钟数可以通过 PRESDIV 的数值调整。在进行异步操作时（外设时钟频率大于晶振时钟频率），每个 CAN 位的外设时钟数可以通过 PRESDIV 的数值或者频率比率调整

例如，考虑有 64 个 MB 的情况，如果晶振与外设时钟频率相等，那么 CAN 位时间被设置为每个比特有 8 个最小时间份额，然后预分频系数（PRESDIV + 1）应该至少为 2。对于预分频系数等于 1 并且 CAN 比特时间为每个比特 8 个最小时间份额,外设与晶振时钟频率之间的比率至少要为 2。

48.4.9 详细的操作模式

FlexCAN 模块一共具有 4 个功能性的模式(正常模式、冻结模式、监听模式以及回环模式)，并且具有三个低功耗模式（禁止模式、睡眠模式、停止模式）。可以参见“操作模式”章节来查看各个模式的描述。下面的几个章节主要讲述冻结模式及低功耗模式下的功能细节。

注意：FlexCAN 模块不支持 CAN 总线上的“恒显性”错误。如果在“恒显性”阶段中请求低功耗或者冻结模式，那么将不会产生相应的确认。

48.4.9.1 冻结模式

CPU 可以通过设置 MCR 的 HALT 位或者当 MCU 进入到调试模式时请求冻结模式。在这两种情况下，MCR 寄存器的 FRZ 位必须置位并且模块不应处于以下任何一种低功耗模式（禁止、睡眠、停止模式）。FlexCAN 模块设置同一个寄存器的 FRZ_ACK 位来确认已经进入到该模式。当请求与确认的条件都满足时，CPU 认为 FlexCAN 模块处于冻结模式。

当在传输或者接收阶段请求冻结模式时，FlexCAN 将做以下事情：

1.等待处于间隙、被动错误、总线关闭或者闲置状态；

2.等待像仲裁、匹配、移入或者移出的内部激活完成。一个未完成的移入过程将不予考虑；

3.忽略接收输入引脚并驱动发送引脚为隐性；

4.停止预分频器，因此将停止所有 CAN 协议的活动；

5.允许写错误计数寄存器，该寄存器在其他模式下只读；

6.置位 MCR 寄存器的 NOT_RDY 及 FRZ_ACK 位。

请求冻结模式之后，必须等待 MCR 寄存器的 FRZ_ACK 位被置位，这样用户才能执行任何操作，不然 FlexCAN 模块可能操作在一个非预期的方式之下。在冻结模式下，任何映射的寄存器都是可访问的，除了 CTRL1[CLK_SRC]位只读但是不能写。退出冻结模式应该做以下：

1.CPU 清除 MCR 寄存器的 FRZ 位；

2.MCU 退出调试模式或者 HALT 位被清 0。

在协议引擎确认忽略冻结请求之后，FRZ_ACK 将会被清 0。退出冻结模式之后，FlexCAN 模块将会通过等待 11 个连续“隐性”位，试图与 CAN 总线重新同步。

48.4.9.2 模块禁止模式

低功耗模式一般用于临时禁止整个 FlexCAN 块，没有电源损耗。可以通过 CPU 置位 MCR 寄存器的 MDIS 位请求低功耗模式，通过置位同一寄存器的 LPM_ACK 位来确认该模式。当所有的请求与确认条件都满足时，CPU 认为 FlexCAN 模块处于禁止模式。

当模块在冻结模式时模块被禁止，那么它将请求禁止 PE 以及 CHI 子模块的时钟，置位 LPM_ACK 位清除 FRZ_ACK 位。如果在传输或者接收过程中禁止模块，那么 FlexCAN 模块必须做以下事情：

1.等待进入 Idle 或者总线关闭状态，或者也可以等待间隙的第三个比特并且检查该比特是不是隐性位；

2.等待所有内部活动比如：仲裁、匹配、移入以及移出过程完成。一个未完成的移入过程不考虑在内；

3.忽略接收输入引脚并且驱动发送引脚为隐性；

4.关掉 PE 以及 CHI 子模块的时钟；

5.置位 MCR 寄存器的 NOTRDY 和 LPMACK 位；

总线接口单元将会继续操作，使能 CPU 访问映射内存寄存器，除了接收邮箱的全局掩码寄存器，接收缓冲区 14 掩码寄存器、接收缓冲区 15 掩码寄存器，接收队列全局掩码寄存器。接收队列消息寄存器、报文缓冲区、接收私有掩码寄存器以及 RAM 中的预留字在模块处于禁止模式时不能够被访问。通过 CPU 清除 MDIS 位可以退出该模式，这将导致 FlexCAN 模块在 CAN 协议引擎确认被 CPU 请求的禁止模式之后请求恢复时钟并且忽略 LPM_ACK 位。

48.4.9.3 睡眠模式

这是一个系统低功耗模式,在该模式下 CPU 总线仍然保持激活并且一个全局睡眠模式请求会被发送给所有的外设以要求他们进入到低功耗模式。当睡眠模式被全局请求时，MCR 的 DOZE 位需要在触发睡眠模式之前被置位。通过置位同一个寄存器中的 FlexCAN 模块的 LPM_ACK 位来确认。当所有的请求与确认条件都满足时，CPU 认为 FlexCAN 模块处于睡眠模式。

如果在冻结模式下触发睡眠模式，那么 FlexCAN 模块将会请求关掉 PE 以及 CHI 子模块的时钟，设置 LPM_ACK 位并且清除 FRZ_ACK 位。如果在接收或者发送阶段触发睡眠模式，那么 FlexCAN 应该做以下：

1.等待进入 Idle 或者总线关闭状态，或者也可以等待间隙的第三个比特并且检查该比特是不是隐性位；

2.等待所有内部活动比如：仲裁、匹配、移入以及移出过程完成。一个未完成的移入过程不考虑在内；

3.忽略接收输入引脚并且驱动发送引脚为隐性；

4.关掉 PE 以及 CHI 子模块的时钟；

5.置位 MCR 寄存器的 NOTRDY 和 LPMACK 位.

总线接口单元将会继续操作，使能 CPU 访问映射内存寄存器，除了接收邮箱全局掩码寄存器，接收缓冲区 14 掩码寄存器、接收缓冲区 15 掩码寄存器，接收队列全局掩码寄存器。接收队列消息寄存器、报文缓冲区、接收私有掩码寄存器以及 RAM 中的预留字在模块处于睡眠模式时不能够被访问。

退出睡眠模式应该做以下：

1.CPU 移出睡眠模式请求；

2.CPU 忽略 MCR 寄存器的 DOZE 位；

3.自我唤醒机制。

在自我唤醒模式中，如果 MCR 寄存器的 SLF_WAK 位在 FlexCAN 模块进入到睡眠模式的同时被置位，那么一旦在 CAN 总线上检测到一个从隐性到显性的电平转换，FlexCAN 模块将会清除 DOZE 位，请求恢复自己的时钟并且在 CAN 协议引擎确认忽略睡眠模式请求时清 OLPM_ACK 位。同时它也会置位 ESR 寄存器的 WAK_INT 位，并且如果使能了 MCR 寄存器的 WAK_MSK，那么会向 CPU 产生一个唤醒中断。FlexCAN 模块会等待连续 11 个隐性位以和 CAN 总线同步。结果，它将不会接收到唤醒它的帧。下表显示了 SLF_WAK 以及 WAK_MAS 位影响从睡眠模式唤醒的影响细节。

表 48-121 从睡眠模式唤醒

SLF_WAK	WAK_INT	WAK_MSK	使能 FlexCAN 时钟	产生唤醒中断
0	–	–	no	no
0	–	–	no	no
1	0	0	no	no
1	0	1	no	no
1	1	0	yes	no
1	1	1	yes	yes

48.4.9.4 停止模式

这是一个系统低功耗模式，在该模式下，所有的 MCU 时钟会因为省电而被停止。停止模式会被 CPU 全局请求，并且可以通过将 FlexCAN 模块的停止确认位置位来获得确认。当满足所有的请求与确认条件时，CPU 认为 FlexCAN 模块处于停止模式。

如果 FlexCAN 在冻结模式接收到全局停止模式请求。FlexCAN 将会置位 LPM_ACK 位，忽略 FRZ_ACK 位然后发送停止确认信号给 CPU，以关闭全局时钟。如果在传输或者接收过程请求停止模式，FlexCAN 模块应该做以下：

1.等待进入 Idle 或者总线关闭状态，或者也可以等待间隙的第三个比特并且检查该比特是不是隐性位；

2.等待所有内部活动比如：仲裁、匹配、移入以及移出过程完成。一个未完成的移入过程不考虑在内；

3.忽略接收输入引脚并且驱动发送引脚为隐性；

4.置位 MCR 寄存器的 NOT_RDY 及 LPM_ACK 位；

5.向 CPU 发送停止确认信号，以至于 CPU 可以关闭全局时钟。

退出停止模式应该以以下方式之一：

1.CPU 恢复时钟并且移除停止模式请求；

2.CPU 恢复时钟并且停止模式请求自我唤醒机制。

在自我唤醒机制中，如果在 FlexCAN 模块进入到停止模式的同时 MCR 寄存器的 SLF_WAK 位置位，那么如果在 CAN 总线上检测到一个从隐形到显性的转变，FlexCAN 会置位 ESR 寄存器的 WAK_INT 位，并且如果开启了 MCR 寄存器的 WAK_MSK 位那么会向 CPU 产生一个唤醒中断。如果接收该中断，CPU 应该恢复时钟并且移除停止模式请求。FlexCAN 模块会等待 11 个隐性位来和 CAN 总线取得同步。结果，它将不会收到用于唤醒它的帧。下表详细介绍了 SLF_WAK 位与 WAK_MSK 位在停止模式唤醒的影响。注意从停止模式唤醒只有当这两位都置位时才能工作。

在 CAN 协议引擎确认了停止模式请求的忽略时，FlexCAN 模块会清除 LPM_ACK 位。

表 48-122 从停止模式唤醒

SLF_WAK	WAK_INT	WAK_MSK	使能 MCU 时钟	产生唤醒中断
0	-	-	no	no
0	-	-	no	no
1	0	0	no	no
1	0	1	no	no
1	1	0	no	no
1	1	1	yes	yes

48.4.10 中断

如果报文缓冲区的相应 IMASK 位置位，那么任何一个报文缓冲区都可以是中断源。对于一个特定的缓冲区，假设既可以被初始化为接收缓冲区也可以被初始化为发送缓冲区，那么接收与发送中断之间没有任何的区别。每个缓冲区在 IFLAG 寄存器都有相应的标志位。当相应的缓冲区成功地完成传输或者接收时该位被置位，并且当 CPU 向该位写‘1’时，清除该位（除非在同时又有另外一个中断产生）。

注意：必须保证 CPU 仅仅清除该位，导致当前的中断。对于这种原因，位操作指令（BSET）禁止用于清除中断标志。这些指令可能导致清除中断标志，在进入当前中断服务程序之后该位又会被置位。

如果开启了接收队列FIFO(MCR的RFEN位被置位),MBs0 到MBs7的中断会有不同的行为。IFLAG1寄存器的Bit7位为“队列FIFO溢出”标志；Bit6位为“队列FIFO警告”标志；Bit5位成了“队列FIFO帧可用”标志;Bit4-0没有用。更多信息参见“中断标志1寄存器(IFLAG1)”的描述。

一个用于所有MBs的混合中断来自于一个或者所有MBs中断源。当MBs或者FIFO的任何一个产生中断时都会产生该中断。在这种情况下CPU必须读取IFLAG1寄存器以决定是哪个MB或者FIFO导致了中断。

其他中断源（总线关闭、错误、发送警告、接收警告、唤醒）也会像MB一样产生中断。并且可以从错误与状态寄存器1和2读取。总线关闭、错误、发送警告、接收警告中断掩码位位于控制寄存器1，唤醒中断掩码位位于MCR。

48.4.11 总线接口

CPU访问FlexCAN寄存器遵守以下规则：

1.无限制读写管理员寄存器会导致访问错误；

2.读写访问预留地址空间会导致访问错误；

3.写只读位时会导致访问错误，如果至少有一位不是只读的话，那么不会产生访问错误，写一个寄存器或者一个寄存器的某些位会因为不同操作模式或者短暂状态而改变写时,应该具有相应的权限。具体细节参见寄存器的描述；

4.读写访问没有实现的地址空间会导致访问错误；

5.在低功耗模式下读写位于RAM区的位置会导致访问错误；

6.如果MAXMB设置的值小于当前可用的MBs的值,那么没有使用的地址空间可以当作通用RAM空间。注意在RAM区的预留字不能被使用。例如，假设FlexCAN被配置为具有16个MBs，RFEN=0x0，并且MAXMB被编程设置为0，那么在这种情况下，最大的MBs数变为1。RAM开始地址为0x0080，并且地址空间0x0080到0x008F都被MB所使用，地址空间0x0090到0x017F是可用的。0x0180到0x087F预留；0x0080到0x0883被一个私有地址掩码所用，位于掩码寄存器中可用的地址空间可能为0x0884到0x08BF。从地址0x08C0 到 0x09DF位内部使用预留字，这个地址区间不能被当作通用RAM。作为一般规则，用于通用目的的自由内存空间大小取决于MAXMB。

注意：当FlexCAN模块正在传输接收帧时，没有用到的MB空间禁止用于通用RAM。

48.5 初始化/应用信息

该节主要介绍初始化FlexCAN模块的指导。

48.5.1 FlexCAN模块初始化序列

FlexCAN模块可以使用以下3种方式重置：

1.MCU级硬件复位，复位所有的内存映射寄存器异步；

2.MCU级软件复位，该复位会同步复位一些内存映射寄存器；

3.MCR寄存器的SOFT_RST位，该位与MCU级的软件复位具有相同的效应。

软件复位是同步的并且通过内部时钟域必须遵守一个内部请求/确认过程。因此，可能需要一些时间来完全传播其影响。当软件复位没有被处理时SOFT_RST位会一直保持置位，因此软件可以轮询检查该位来知道复位何时完成。同时软件复位不能用于在任何低功耗模式时钟关

闭的情况下。应该退出低功耗模式并且在使用软件复位之前应该恢复时钟。

当模块处于禁止模式时，应该选择时钟源（CLK_SRC 位）。在选择好时钟源之后使能模块（MDIS 被清 0），FlexCAN 模块自动进入到冻结模式。在冻结模式中，FlexCAN 模块并没有与 CAN 总线取得同步，MCR 寄存器的 HALT 及 FRZ 位置位，内部状态机禁止并且 MCR 寄存器的 FRZ_ACK 及 NOT_RDY 位置位。发送引脚 Tx 为隐性状态并且 FlexCAN 没有初始化任何传输或者接收帧。注意:报文缓冲区与接收私有掩码寄存器不受复位影响,应此他们不会被自动初始化。

对于任何配置改变/初始化，要求 FlexCAN 模块处于冻结模式。下列位用于 FlexCAN 模块的一个常用初始化模块：

1.初始化模块配置寄存器:（1）通过设置 IRMQ 位使能每一个 MB 的私有过滤器以及接收队列功能;（2）通过设置 WRN_EN 位来开启警告中断;（3）如果需要，通过设置 SRX_DIS 位来禁止帧自接收;（4）通过设置 RFEN 位来开始接受队列 FIFO;（5）通过设置 AEN 位开启终止机制;（6）通过设置 LPRIO_EN 位来开启本地优先功能;

2.初始化控制寄存器:（1）确定位时间参数：PROPSEG, PSEG1, PSEG2, RJW;（2）通过编程 PRESDIV 字段来设置位率;（3）确定内部仲裁模式（LBUF 位);

3.初始化报文缓冲区:（1）所有报文缓冲区的控制与状态寄存器必须被初始化;（2）如果开启接收队列 FIFO，那么 ID 过滤器表必须被初始化;（3）报文缓冲区的其他实体应该按照要求被初始化;

4.初始化接收私有掩码寄存器;

5.置位 IMASK 寄存器中所要求的中断屏蔽位（对于所有的 MB 中断），CTRL 寄存器（对于总线关闭及错误中断），MCR 寄存器（唤醒中断);

6.清除 MCR 寄存器的 HALT 位。

从上一事件起，FlexCAN 模块将试图与 CAN 总线同步。