FAQ0103

常见问题解答

AT32F4系列CAN过滤器配置方法

Questions：如何设置 AT32F4xx 系列 CAN 过滤器？

Answer：

注：本 FAQ 对应的代码是基于雅特力提供的 V2.x.x 板级支持包（BSP）而开发，对于其他版本 BSP，需要注意使用上的区别。

CAN 过滤器相当于关卡，每当收到一条报文时，CAN 要先将收到的报文从这些过滤器上"过滤"一下，能通过的报文是有效报文，收进相关联 FIFO（FIFO0 或 FIFO1），不能通过的是无效报文直接丢弃。AT32F4 同时支持 CAN2.0A 和 CAN2.0B 协议，即过滤器组支持 11 位标准格式 CAN 和 29 位扩展格式 CAN 标识符。

每个 CAN 控制器提供 14 个位宽可变、可配置的过滤器组（0~13），用以对接收到的帧进行过滤。

每组过滤器包括了 2 个 32 位寄存器：CAN_FiFB1 和 CAN_FiFB2，通过配置 CAN 过滤器位宽配置寄存器（CAN_FBWCFG）的对应位，设置过滤器位宽为 2 个 16 位或者单个 32 位。

每组过滤器组有两种工作模式：标识符列表模式和标识符掩码模式。通过设置 CAN 过滤器模式配置寄存器（CAN_FMCFG） 的 FMSELx 位可以设置过滤器寄存器工作在标识符掩码模式或者标识符列表模式，掩码模式用来指定哪些位与预设标识符相同，哪些位无需比较，列表模式表示标识符（ID 号）必须与预设标识符一致。两种模式与过滤器位宽配合使用，可以有以下四种过滤方式：

图 21-8 32位宽标识符掩码模式

ID	CAN_FiFB1[31:21]	CAN_FiFB1[20:3]	CAN_FiFB1[2:0]
Mask	CAN_FiFB2[31:21]	CAN_FiFB2[20:3]	CAN_FiFB2[2:0]
Mapping	SID[10:0]	EID[17:0]	IDT / RTR / 0

图 21-9 32位宽标识符列表模式

ID	CAN_FiFB1[31:21]	CAN_FiFB1[20:3]	CAN_FiFB1[2:0]
ID	CAN_FiFB2[31:21]	CAN_FiFB2[20:3]	CAN_FiFB2[2:0]
Mapping	SID[10:0]	EID[17:0]	IDT / RTR / 0

图 21-10 16位宽标识符掩码模式

ID	CAN_FiFB1[15:5]	CAN_FiFB1[4:0]		
Mask	CAN_FiFB1[31:21]	CAN_FiFB1[20:16]		
ID	CAN_FiFB2[15:5]	CAN_FiFB2[4:0]		
Mask	CAN_FiFB2[31:21]	CAN_FiFB2[20:16]		
Mapping	SID[10:0]	RTR	IDT	EID[17:15]

图 21-11 16位宽标识符列表模式

ID	CAN_FiFB1[15:8]	CAN_FiFB1[7:0]		
ID	CAN_FiFB1[31:24]	CAN_FiFB1[23:16]		
ID	CAN_FiFB2[15:8]	CAN_FiFB2[7:0]		
ID	CAN_FiFB2[31:24]	CAN_FiFB2[23:16]		
Mapping	SID[10:0]	RTR	IDT	EID[17:15]

标识符掩码模式：

可过滤出一组标识符。当过滤器位宽为 32 位时，CAN_FiFB1 中保存的是标识符匹配值 ID，CAN_FiFB2 中保存的是屏蔽掩码，CAN_FiFB2 中如果某一位为 1，则 CAN_FiFB1 中相应的位必须与收到的帧的标志符中的相应位吻合才能通过过滤器；CAN_FiFB2 中为 0 的位表示 CAN_FiFB1 中的相应位可不必与收到的帧进行匹配。

以标准标识符为例，如下表设置，有 4 个标识符可以通过过滤器。

CAN_FiFB1（标识符匹配值 ID）	0b 00000000001
CAN_FiFB2（标识符掩码）	0b 11111111001
有效报文标识符	0b 00000000001 0b 00000000011 0b 00000000101 0b 00000000111

当过滤器位宽为 16 位时，CAN_FiFB1 和 CAN_FiFB2 各保存 1 组标识符掩码过滤器数据，CAN_FiFB1 的低 16 位中保存的是标识符匹配值 ID，CAN_FiFB1 的高 16 位中保存的是屏蔽码；CAN_FiFB2 的设置方法与 CAN_FiFB1 一样，即一组过滤器组可以设置为 2 个 16 位的屏蔽位模式的过滤器。过滤器位宽为 16 位的标识符掩码过滤规则与过滤器位宽为 32 位一样。

标识符列表模式：

当过滤器位宽为 32 位时，CAN_FiFB1 和 CAN_FiFB2 中值的都是要匹配的标识符，收到的帧的标识符必须与其中的一个吻合才能通过过滤，即有 2 个标识符的可以通过过滤器。

以标准标识符为例，如下表设置，有 2 个标识符可以通过过滤器。

CAN_FiFB1（标识符匹配值 ID）	0b 00000000001
CAN_FiFB2（标识符匹配值 ID）	0b 00000000011
有效报文标识符	0b 00000000001 0b 00000000011

当过滤器位宽为 16 位时，CAN_FiFB1 和 CAN_FiFB2 各保存 2 个标识符列表过滤器数据，CAN_FiFB1 的低 16 位和高 16 位中各保存 2 个要匹配的标识符；CAN_FiFB2 的设置方法与 CAN_FiFB1 一样，即一组过滤器组可以设置 4 个 16 位的列表模式的过滤器。过滤器位宽为 16 位的标识符列表过滤规则与过滤器位宽为 32 位一样。

所有的过滤器组是并联的，即一个报文只要通过了一个过滤器，就算是有效的。

按工作模式和宽度，一个过滤器组可以变成以下几中形式之一：

- 1 个 32 位的屏蔽位模式的过滤器
- 2 个 32 位的列表模式的过滤器
- 2 个 16 位的屏蔽位模式的过滤器
- 4 个 16 位的列表模式的过滤器

过滤器匹配序号：

14 组过滤器组根据位宽、模式的不同，具有不同的过滤效果，例如 32 位宽标识符掩码模式包含序号为 n 的过滤器， 而 16 位宽标识符列表模式包含序号为 n、n+1、n+2 以及 n+3 的过滤器。一帧报文通过了某个序号（Filter Nnumber）N 的过滤器，则该帧的接收 FIFO 邮箱数据长度和时间戳寄存器（CAN_RFCx）RFFMN[7：0]位存储该序号 N，过滤器序号的分配不关心对应的过滤器组是否处于激活状态。

优先级匹配规则：

CAN 控制器接收一帧报文，有可能能够通过多个过滤器的过滤，在这种情况下，存放在接收邮箱中的过

滤器匹配序号，根据以下优先级规则确定。

- 位宽为 32 位的过滤器，优先级高于位宽为 16 位的过滤器。
- 在相同位宽的情况下，标识符列表模式的优先级高于标识符掩码模式。
- 在位宽和标识符模式都相同的情况下，标号越小的过滤器具有更高的优先级。

过滤器配置：

- 将 CAN 过滤器控制寄存器（CAN_FCTRL）FCS 位置‘1’，允许配置 CAN 过滤器。
- 写 CAN 过滤器模式配置寄存器（CAN_FMCFG）FMSELx 位，控制过滤器工作模式为标识符掩码模式或者列表模式。
- 写 CAN 过滤器位宽配置寄存器（CAN_FBWCFG）FBWSELx 位，控制过滤器位宽为 2 个 16 位或者单个 32 位。
- 写 CAN 过滤器 FIFO 关联寄存器（CAN_FRF）FRFSELx 位，关联过滤器 x 到 FIFO0 或者 FIFO1。
- 将 CAN 过滤器激活控制寄存器（CAN_FACFG）FAENx 位置‘1’，激活对应的过滤器组 x。
- 写 CAN_FiFBx（其中 i=0…13；x=1,2）,配置 0~13 组过滤器组。
- 将 CAN 过滤器控制寄存器（CAN_FCTRL）FCS 位置‘0’，完成 CAN 过滤器配置过程。

BSP DEMO 初始化过滤器组说明：

此代码基于 AT32F403A 的 BSP DEMO 程序\project\at_start_f403a\examples\can\filter。

```
/*过滤器组 0：扩展标识符列表模式，2 个过滤器*/
/*使能过滤器*/
can_filter_init_struct.filter_activate_enable = TRUE;
/*配置为标识符列表模式，其他可选参数掩码模式：CAN_FILTER_MODE_ID_MASK*/
can_filter_init_struct.filter_mode = CAN_FILTER_MODE_ID_LIST;
/*接收 FIFO 配置为 FIFO0，其他可选参数 FIFO1：CAN_FILTER_FIFO1*/
can_filter_init_struct.filter_fifo = CAN_FILTER_FIFO0;
/*配置过滤器组 0，其他可选参数 1-13 */
can_filter_init_struct.filter_number = 0;
/*配置过滤器位宽 32 位，其他可选参数 16 位：CAN_FILTER_16BIT */
can_filter_init_struct.filter_bit = CAN_FILTER_32BIT;
/* CAN_FiFB1 中保存的是标识符匹配值 ID */
can_filter_init_struct.filter_id_high = (((FILTER_EXT_ID1 << 3) >> 16) & 0xFFFF);     /* extended identifier is 29 bit */
can_filter_init_struct.filter_id_low = ((FILTER_EXT_ID1 << 3) & 0xFFFF) | 0x04;
/* CAN_FiFB2 中保存的是标识符匹配值 ID */
can_filter_init_struct.filter_mask_high = ((FILTER_EXT_ID2 << 3) >> 16) & 0xFFFF; /* extended identifier is 29 bit */
can_filter_init_struct.filter_mask_low = ((FILTER_EXT_ID2 << 3) & 0xFFFF) | 0x04;
can_filter_init(CAN1, &can_filter_init_struct);

/* can filter 1 config */
can_filter_init_struct.filter_activate_enable = TRUE;
can_filter_init_struct.filter_mode = CAN_FILTER_MODE_ID_LIST;
can_filter_init_struct.filter_fifo = CAN_FILTER_FIFO0;
```

```
can_filter_init_struct.filter_number = 1;
can_filter_init_struct.filter_bit = CAN_FILTER_32BIT;
can_filter_init_struct.filter_id_high = FILTER_STD_ID1 << 5;    /* standard identifier is 11 bit */
can_filter_init_struct.filter_id_low = 0;
can_filter_init_struct.filter_mask_high = FILTER_STD_ID2 << 5;/* standard identifier is 11 bit */
can_filter_init_struct.filter_mask_low = 0;
can_filter_init(CAN1, &can_filter_init_struct);
```

类型： MCU 应用

适用型号： AT32F403, AT32F403A, AT32F407, AT32F413, AT32F415, AT32A403A, AT32F435, AT32F437, AT32F423, AT32F402, AT32F405

主功能： CAN

次功能： 无

文档版本历史

日期	版本	变更
2022.2.25	2.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：（A）对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）航天应用或航天环境；（D）武器，且/或（E）其他可能导致人身伤害、死亡及财产损害的应用。如果采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险及法律责任仍将由采购商单独承担，且采购商应独力负责在前述应用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。