FAQ0103

Frequently Asked Questions

How to set CAN filters?

Questions: How to set CAN filters of AT32F4xx?

Answer:

Note: The code in this FAQ is developed based on Artery V2.x.x BSP. When in use, users need to pay attention to the differences in use due to different versions of BSP.

CAN filter acts like a checkpoint. Each piece of message received must go through CAN filter. Those which pass the filtering are valid messages and they are stored in relevant FIFO (FIFO0 or FIFO1), while those which failed filtering would be discarded as invalid messages. AT32F4xx series supports both CAN2.0A and CAN2.0B protocols, which means that a filtering bank supports 11-bit standard CAN and 29-bit extended CAN identifiers.

Each CAN controller offers 14 bit-width-scalable and configurable filter banks (numbered from 0 to 13) to filter the received frames.

Each filter bank contains two 32-bit registers: CAN_FiFB1 and CAN_FiFB2. Filter bit width can be 2x 16 bits or one 32 bit by setting the CAN_FBWCFG registers.

A filter bank has two operation modes: identifier list mode and identifier mask mode. The FMSELx bit in the CAN_FMCFG register can be used to select one of the two modes.

Identifier mask mode is used to specify which bits match the pre-configured identifiers, and which bits do not.

Identifier list mode means that ID must be the same as pre-configured identifiers.

By using the two modes in conjunction with filter bit width settings, it is possible to achieve four filtering modes as shown below:

Figure 21-8 32-bit identifier mask mode

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	

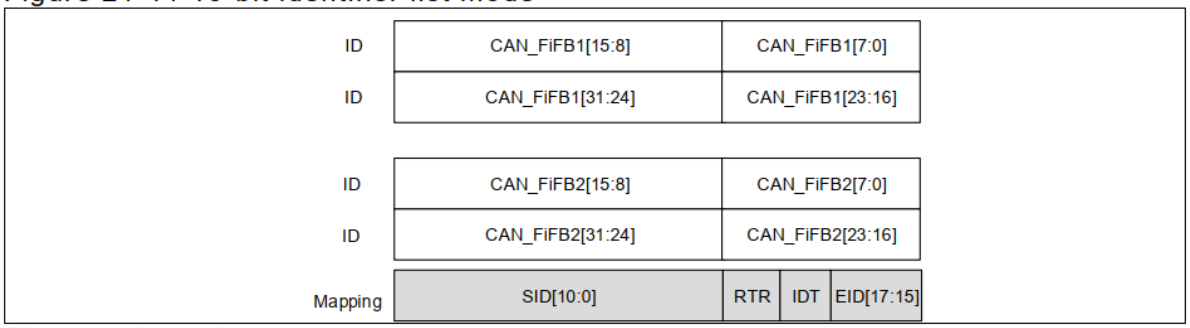
Figure 21-9 32-bit identifier list mode

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	

Figure 21-10 16-bit identifier mask mode

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]	

Figure 21-11 16-bit identifier list mode



Identifier mask mode:

This mode is able to filter a set of identifiers. For 32-bit wide filter, the CAN_FiFB1 holds identifier match value ID, and the CAN_FiFB2 stores a mask code. If one of the bits in CAN_FiFB2 is 1, the corresponding bits in CAN_FiFB1 must match the corresponding bits of the received frame in order to pass through a filter.

The bit of 1 in the CAN_FiFB2 indicates that the corresponding bit in the CAN_FiFB1 does not have to match the received frame.

Taking a standard identifier as an example, there are four identifiers which can pass a filter.

CAN_FiFB1 (identifier match value ID)	0b 00000000001
CAN_FiFB2 (identifier mask)	0b 11111111001
Valid message identifier	0b 00000000001 0b 00000000011 0b 00000000101 0b 00000000111

For 16-bit wide filter, CAN_FiFB1 and CAN_FiFB2 stores a set of identifier mask filtered data respectively.

The low 16 bits in CAN_FiFB1 hold identifier match value ID, and its high 16 bits store identifiers.

CAN_FiFB2 is similar to CAN_FiFB1 in terms of the setting method: a filter bank can be configured to 2x 16-bit mask mode filters. The filtering rules of identifier mask are the same to both 16-bit and 32-bit wide filters.

Identifier list mode:

For a 32-bit filter, both CAN_FiFB1 and CAN_FiFB2 hold “must match” identifiers. The incoming identifiers must match one of the two in order to pass a filtering. In other words, two identifiers are required to pass through filtering.

The identifiers of a frame received can pass filtering only when they match one of them.

The table below takes a standard identifier as an example. Here two identifiers are required to be able to pass filtering.

CAN_FiFB1 (identifier match value ID)	0b 00000000001
CAN_FiFB2 (identifier match value ID)	0b 00000000011
Valid message identifier	0b 00000000001 0b 00000000011

For a 16-bit filter, each of CAN_FiFB1 and CAN_FiFB2 saves two identifier list filter data. For CAN_FiFB1, its low 16 bits and high 16 bits hold two “must match” identifiers respectively. In terms of setting method, CAN_FiFB2 is the same as CAN_FiFB1. In other words, a filter bank can be set as 4x 16-bit list mode filters. A 16-bit filter share the same identifier list filtering rules as a 32-bit filter.

All the filter banks are connected in parallel – a message is regarded as valid only if it passes one of these filters.

Depending on operating mode and width, a filter bank can be set as one of the following:

- 1x 32-bit mask bit mode filter
- 2x 32-bit list mode filter
- 2x 16-bit mask bit mode filter
- 4x 16-bit list mode filter

Filter match index:

The 14 filter banks can provide different filtering effects according to the bit width and operating mode. For example, in a 32-bit identifier mask mode, it contains filters numbered “n”; in a 16-bit identifier list mode, it contains filters numbered “n”, “n+1”, “n+2” and “n+3”. When a frame of message successfully passes a filter numbered “N”, this number will be stored in the RFFMN [7:0] bit of the receive FIFO mailbox data length and time stamp register (CAN_RFCx). The distribution of filter number does not care about whether the corresponding filter bank is in active state or not.

Filter priority rules:

An incoming message may be able to pass successfully through several filters. In this case the filter match value stored in the receive mailbox is chosen according to the following priority rules:

- A 32-bit filter takes priority over a 16-bit filter
- For filters of equal scale, priority is given to the identifier list mode over the identifier mask mode
- For filters of equal scale and mode, priority is given by the filter number (the lower the number, the higher the priority)

Filter configuration:

- Enable the configuration of CAN filter by setting FCS=1 in the CAN_FCTRL register
- Select identifier mask mode or identifier list mode through the FMSELx bit in the CAN_FMCFG register
- Select 2x 16-bit filter width or a single 32-bit filter width by setting the FBWSELx bit in the CAN_FBWCFG register
- Associate filter “x” to FIFO0 or FIFO1 through the FRFSELx bit in the CAN_FRF register
- Activate a corresponding filter bank “x” by setting FAENx=1 in the CAN_FACFG register
- Configure 0~13 filter banks through the CAN_FiFBx register (i=0…13; x=1,2)
- Complete the filter configuration by setting FCS=0 in the CAN_FCTRL register

BSP demo of filter initialization:

This code is designed according to AT32F403A’s BSP DEMO\project\at_start_f403a\examples\can\filter

```
/*Filter bank 0: extended identifier list mode, two filters*/
/*Enable filter*/
can_filter_init_struct.filter_activate_enable = TRUE;
/*Set idenfier list mode, other optional parameter mask mode: CAN_FILTER_MODE_ID_MASK*/
can_filter_init_struct.filter_mode = CAN_FILTER_MODE_ID_LIST;
/*Select FIFO0 as Receive FIFO, other optional parameter FIFO1: CAN_FILTER_FIFO1*/
can_filter_init_struct.filter_fifo = CAN_FILTER_FIFO0;
/*Set filter bank 0, other optional parameters 1-13 */
can_filter_init_struct.filter_number = 0;
/*Set 32-bit filter, other optional parameter 16 bit: CAN_FILTER_16BIT */
can_filter_init_struct.filter_bit = CAN_FILTER_32BIT;
/* CAN_FiFB1 holds identifier match value 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 holds identifier match value 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);
```

Type: MCU application

Applicable products: AT32F403, AT32F403A, AT32F407, AT32F413, AT32F415, AT32A403A, AT32F435, AT32F437, AT32F423, AT32F402, AT32F405

Main function: CAN

Other function: None

Document revision history

Date	Revision	Changes
2022.2.25	2.0.0	Initial release