Example of Configuration of CAN Filters on LPCOpen

LPCopen CAN Api lets you set up filters using two parameters, a Filter ID and a Filter MASK:

11 Bit FILTER ID

MSG.ID = 0x3A = 58

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0

FILTER MASK = 0x26 = 38

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0

This configuration will filter OUT any message whose ID:

- BIT[1] is not = 1
- BIT[2] is not= 0
- Bit[5] is not= 1

Which means messages to be accepted must have their B[5] = 1 and have their BIT[1] = 1 BIT[2] = 0

So an incoming message with ID 0xFB would be accepted, another with ID 0xFF would be discarded

Why? How does it work?

Because the message ID acts as a base for the filter and the Message Mask selects which parts of this base interests us.

-IF a MASK bit equals one, then the Filter will take into account the precise value contained in the chosen ID, which means, the received message id, to be accepted, must have at that one bit the same value as the Filter ID.

-ELSE, when a mask bit equals zero, the FILTER ID value at that bit is ignored and the incoming message id can be, at that specific bit 0 or 1 without being excluded because the chosen FILTER ID corresponding bit is different.

→ 29 Bit identifiers, the mechanism is the same, but to set them up you need either to write a 29 bit word or put `| CAN_MSGOBJ_EXT`.

Here the code to set up the same filter illustrated before in a 29 bit format

The code also adds a second object intended to filter out messages whose ID is greater than 32 (decimal)!

```
msg_obj.msgobj = 1;//1st Filter exclude ID whose bit[5] and bit[1] is not 1 and whose bit[2] is not 0i
msg_obj.mode_id = 0x3A| CAN_MSGOBJ_EXT; // Filter ID, set for a 29BIT identifier, same as msg_obj.mode_id =0x0000003A;
msg_obj.mask = 0x26| CAN_MSGOBJ_EXT; // Filter MASK, set for a 29BIT identifier, same as msg_obj.mode_id =0x00000026;
LPC_CCAN_API->config_rxmsgobj(&msg_obj); // Tell the System to insert this Filter

msg_obj.msgobj = 2; //2nd Filter Exclude any ID higher than 64
msg_obj.mode_id = 0x00| CAN_MSGOBJ_EXT;
msg_obj.mask = 0x2FFFFFE0| CAN_MSGOBJ_EXT;
LPC_CCAN_API->config_rxmsgobj(&msg_obj);
```

What is the deal with multiple message objects?

Full CAN specifies controllers having 32 message object buffers, each one able to store a Filter, Data ecc..

Upon receiving a CAN message, the microcontroller scans each message object set by the user and tries to compare with the filter the received ID, the first configured message object compatible with the filter will store the received data (and consequently will be assigned to that message object slot)!

How do I allow all messages to be received?

Simple: set up both mask and filter id equal to 0 (for this apply the rule of adding |CAN_MSGOBJ_EXT in case we need 29bit frames).

Matteo Ricciutelli 21/1/2019