CAN RX Filter

Details for selected row:

Configuration 0

Name	can_pal1_Config0
Read only	☑
Number of buffers	32
Operation Mode	Normal
PE clock source	Oscillator clock
Enable FD	☑
Buffer payload size	CAN_PAYLOAD_SIZE_8
Module clock	48 MHz
PE clock	8 MHz
Bitrate to time segments	☑

Bitrate configuration

Item	Propagation segment	Phase segment 1	Phase segment 2	Prescaler Division Factor	Resync
Data Phase	11	1	1	0	1

☐ Enable Rx FIFO extension

RxFIFO ID filters number	8 Rx FIFO Filters
RxFIFO ID filter format	Format A

```c
int main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                 /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of Processor Expert internal initialization.                    ***/

    /* Do the initializations required for this application */
    BoardInit();
    GPIOInit();

    CAN_Init(&can_pal1_instance, &can_pal1_Config0);

    /* Set information about the data to be sent
     *  - Standard message ID
     *  - Bit rate switch enabled to use a different bitrate for the data segment
     *  - Flexible data rate enabled
     *  - Use zeros for FD padding
     */
    can_buff_config_t buffCfg =  {
        .enableFD = true,
        .enableBRS = true,
        .fdPadding = 0U,
        .idType = CAN_MSG_ID_STD,
        .isRemote = false
    };

    /* Configure RX buffer with index RX_MAILBOX */

    status_t status;
    uint8_t idx = 32;
    uint32_t i;

    for(i = 0; i < idx; i ++)
    {
        CAN_ConfigRxBuff(&can_pal1_instance, i, &buffCfg, i + 1);
    }
```

```
while(1)
 {
     /* Define receive buffer */
     can_message_t recvMsg;

     /* Start receiving data in RX_MAILBOX. */
     for(i = 0; i < idx; i ++)
     CAN_Receive(&can_pal1_instance, i, &recvMsg);

     status = 0xff;
     while(status != 0)
     {
     /* Wait until the previous FlexCAN receive is completed */
     for(i = 0; i < idx; i ++)
     {
             status = CAN_GetTransferStatus(&can_pal1_instance, i);
             if(status == 0)
                     break;
     }
     }



     /* Check the received message ID and payload */
     if((recvMsg.data[0] == LED0_CHANGE_REQUESTED) &&
             recvMsg.id == RX_MSG_ID)
     {
         /* Toggle output value LED1 */
         PINS_DRV_TogglePins(GPIO_PORT, (1 << LED0));
     }
     else if((recvMsg.data[0] == LED1_CHANGE_REQUESTED) &&
             recvMsg.id == RX_MSG_ID)
     {
         /* Toggle output value LED0 */
         PINS_DRV_TogglePins(GPIO_PORT, (1 << LED1));
     }
  }

/*** Don't write any code pass this line, or it will be deleted during code generation. ***/
/*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
#ifdef PEX_RTOS_START
  PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS component. */
#endif
/*** End of RTOS startup code.  ***/
/*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
```

```
  for(;;) {
     if(exit_code != 0) {
       break;
     }
   }
   return exit_code;
   /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```