

CAN driver with prioritized transmit buffer

1. Introduction.....	2
2. Module Features	2
3. List of Component Modules	3
4. Using the Library Module in a Project.....	3
5. List of Shared Parameters.....	4
<i>Shared Data Bytes</i>	<i>4</i>
<i>Shared Functions</i>	<i>4</i>
<i>Shared Macros</i>	<i>5</i>
6. Functions	6
7. Macros	10
8. Error and Status Flags.....	16

1. Introduction

The primary objective of CAN general-purpose library module is to speed up programmer's job. With this programmer's user interface is shifted from one of implementation specifics, such as setting control bits and testing status bits to one of logical commands such as read, write and execute; In a very much like that of a high level language. This allows user to focus on the requirements of their applications. It provides interrupt-based operation and has data buffer, which provides maximum benefit of parallel processing. Module code is linkable and relocatable, which provides user facility to use it without modifications. Please refer AN853 for more details on the module. The AN853 uses different naming convention so use this document for details on functions, macros and variables.

2. Module Features

- It implements prioritized transmit buffer to avoid priority inversion.
- It supports user defined sized FIFO (First In First Out) buffer for Transmission.
- Incorporates interrupt-driven transmission, allowing user other tasks to execute in the foreground.
- Provides simple functions and macro wrappers to read from and write to the buffers.

3. List of Component Modules

CANPrTx.ex.txt	This is main test file developed to demonstrate use of the library functions.
CANPrTx.asm	This is prioritized CAN driver code implementation file. <u>One needs include this file in their project.</u>
CANPrTx.inc	This file contains definition of shared parameters for use in Assembly language. One needs to include this file in the Assembly file where they are calling library routines. This file is taking care of definition of all Extern Global parameter so one can directly call library routines in their program.
P18xxx.inc	General purpose processor definition file for PIC18xxx family
P18xxx8.inc	General purpose symbol definition file for PIC18xxx8 family

4. Using the Library Module in a Project

Please follow below steps to use this library module in your project.

1. Use Application Maestro to configure your code as required.
2. At the Generate Files step, save the output to the directory where your code project resides.
3. Launch MPLAB, and open the project's workspace.
4. Verify that the Microchip language tool suite is selected (*Project>Select Language Toolsuite*).
5. In the Workspace view, right-click on the "Source Files" node. Select the "Add Files" option. Select CANPrTx.asm and click **OK**.
6. Now right-click on the "Linker Scripts" node and select "Add Files". Add the appropriate linker file (.lkr) for the project's target microcontroller.
7. Add any other files that the project may require. Save and close the project.
8. In your main source (assembler) file, add include directive at the head of the code listing to include the file CANPrTx.inc. By doing so, all files required to make the generated code work in your project will be included by reference when you build the project.
9. To use the module in your application, invoke the functions or macros as needed.

5. List of Shared Parameters

Shared Data Bytes

vCANPrTxBufSize	It indicates transmit buffer write position offset. It can be used to get information of pending data in transmit buffer
-----------------	--

Shared Functions

CANPrTxInit	Initializes CAN library module
CANPrTxSetOpMode	Sets the operating mode.
CANPrTxSetBaudRate	It initializes the CAN baud rate register with supplied values
CANPrTxSetReg	Bit adjusts the CAN ID value and loads it into the CAN register.
CANPrTxReadReg	Reads and bit adjusts the CAN ID value and copies at desired location
CANPrTxSendMsg	Used to send the message over CAN bus.
CANPrTxReadMsg	Used to read the received CAN message.
CANPrTxISR	CAN Interrupt handler for transmission.
CANPrTxIsTxPassive	Checks if transmitter is in error passive mode.
CANPrTxIsRxPassive	Checks if receiver is in error passive mode.
CANPrTxIsBusOff	Checks if transmitter is in bus off state
CANPrTxIsRxReady	Checks if at least one receive buffer is free
CANPrTxIsTxReady	Checks if at least one transmit buffer is free

Shared Macros

mDisableCANTxInt	Disables CAN transmit interrupt.
mDisableCANTxInt	Enables CAN transmit interrupt.
mCANPrTxAbortAll	Aborts all pending transmission
mCANPrTxGetTxErrCnt	Gets transmit error count
mCANPrTxGetRxErrCnt	Gets receive error count
mCANPrTxInit	Initializes the CAN module
mCANPrTxSetBaud	Initializes the CAN baud rate register
mCANPrTxSetOpMode	Sets the operating mode (Blocking)
mCANPrTxSetOpModeNoWait	Sets the operating mode (Non-blocking)
mCANPrTxSetReg	Bit adjusts the CAN ID value and loads it into the CAN register
mCANPrTxSetReg_IF	Bit adjusts the CAN ID value and loads it into the CAN register. Expects address pointer for Flags info
mCANPrTxSetReg_IV	Bit adjusts the CAN ID value and loads it into the CAN register. Expects address pointer for ID info
mCANPrTxSetReg_IV_IF	Bit adjusts the CAN ID value and loads it into the CAN register. Expects address pointer for ID, Flags info and destination register address in FSR0
mCANPrTxSetReg_PREG_IV_IF	Bit adjusts the CAN ID value and loads it into the CAN register. Expects address pointer for Flags info and ID value, destination register address in FSR0
mCANPrTxSetReg_PREG_DV_IF	Bit adjusts the CAN ID value and loads it into the CAN register. Expects address pointer for Flags info, destination register address in FSR0 and ID value in _vReg1_O
mCANPrTxReadReg	Corresponding CAN id registers are read and value is bit adjusted and copied into 32-bit destination
mCANPrTxReadReg_PREG	Corresponding CAN id registers are read and value is bit adjusted and copied into 32-bit destination. Expects the source address in FSR0
mCANPrTxSendMsg	Copies the data in available transmit buffer according to priority
mCANPrTxSendMsg_IID_IDL_IF	Copies the data in available transmit buffer according to priority. Expects address pointer for ID, data length and Flags.
mCANPrTxReadMsg	Reads the received CAN data

6. Functions

Function	CANPrTxInit
Preconditions	None
Overview	Initializes CAN module
Input	vCANPrTxSJW_O (SJW value as defined in 18XXX8 datasheet (Must be between 1 thru 4)) vCANPrTxBRP_O (BRP value as defined in 18XXX8 datasheet (Must be between 1 thru 64)) vCANPrTxPHSEG1_O (PHSEG1 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)) vCANPrTxPHSEG2_O (PHSEG2 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)) vCANPrTxPROPSEG2_O (PROPSEG2 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)) vFlags1_O – Flag value of type CAN_CONFIG_CLASS
Output	CAN bit rate is set. All masks registers are set '0' to allow all messages. Filter registers are set according to flag value. If (config & CAN_CONFIG_VALID_XTD_MSG). Set all filters to XTD_MSG Else if (config & CONFIG_VALID_STD_MSG). Set all filters to STD_MSG else Set half of the filters to STD while rests to XTD_MSG
Side Effects	Aborts all pending transmission. W, STATUS, BSR, PRODL, FSR0 changed.
Stack Requirement	3 level deep

Function	CANPrTxSetOpMode
Preconditions	None
Overview	Sets the CAN operating mode by copying the given byte in CANCON. It ensures that given mode is set
Input	WREG – Operation mode code of type CAN_OP_MODE
Output	MCU is set to requested mode
Side Effects	This is a blocking call it will not return until the requested mode is set. W, STATUS changed
Stack Requirement	1 level deep

Function	CANPrTxSetBaudRate
Preconditions	MCU must be in configuration mode or else values will be ignored.
Overview	It initializes the CAN baud rate register with supplied values
Input	vCANPrTxSJW_O (SJW value as defined in 18XXX8 datasheet (Must be between 1 thru 4)) vCANPrTxBRP_O (BRP value as defined in 18XXX8 datasheet (Must be between 1 thru 64)) vCANPrTxPHSEG1_O (PHSEG1 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)) vCANPrTxPHSEG2_O (PHSEG2 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)) vCANPrTxPROPSEG2_O (PROPSEG2 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)) vFlags1_O – Flag value of type CAN_CONFIG_CLASS
Output	CAN bit rate is set.
Side Effects	W, STATUS, BSR and PRODL changed
Stack Requirement	1 level deep

Function	CANPrTxSetReg
Preconditions	None
Overview	If given id is of type standard identifier, only SIDH and SIDL are updated. If given id is of type extended identifier, bits val<17:0> is copied to EIDH, EIDL and SIDH<1:0> bits val<28:18> is copied to SIDH and SIDL
Input	FSR0 – Starting address of the 32-bit buffer to be updated. vReg1_O: _vReg1_O+3 - 32-bit value to be converted vReg1_O= LL, _vReg1_O+1 =LH, _vReg1_O+2 = HL and _vReg1_O+3 = HH byte) vFlags1_O - Type of message Flag. Either CAN_CONFIG_XTD_MSG or CAN_CONFIG_STD_MSG
Output	Given CAN id value 'val' is bit adjusted and copied into corresponding PIC18XXX8 CAN registers
Side Effects	W, STATUS, BSR, FSR0 changed
Stack Requirement	2 level deep
Function	CANPrTxReadReg
Preconditions	None
Overview	If given id is of type standard identifier, only SIDH and SIDL are read. If given id is of type extended identifier, EIDH, EIDL are read too.
Input	FSR0 – Starting address of the 32-bit buffer to be read. FSR1 – Starting address of the 32-bit buffer to store the read value.
Output	Corresponding PIC18xxx8 CAN id registers are read and value is bit adjusted and copied into 32-bit destination
Side Effects	W, STATUS, BSR, FSR0, FSR1 changed
Stack Requirement	2 level deep
Function	CANPrTxSendMsg
Preconditions	None
Overview	It copies the data in available hardware or software buffer. If present data is of higher priority then the data in hardware buffer then it aborts transmission of lowest priority data in HW buffer and copies it to SW buffer and copies present data to HW buffer for immediate transmission.
Input	_vReg1_O<3:0> - 32-bit Message ID FSR1 – Starting address of the 32-bit data buffer _DataLength – Total data bytes to be send m_Flags – CAN_TX_MSG_FLAGS type flag value
Output	W reg = 0, If buffer is full W reg = 1, If successful.
Side Effects	W, STATUS, BSR, FSR0, FSR1 changed
Stack Requirement	3 level deep

Function	CANPrTxReadMsg
Preconditions	None
Overview	It formats the data in Rx Buffer and returns it. It stores 32-bit Message ID, Recd. data, Data Length and Receiver Status Flags into fixed memory locations
Input	FSR0 – Starting address of the buffer to store received data
Output	W reg = 0, If no data is pending W reg = 1, If successful. _vTemp32Data – Received message ID _DataLength – Total data bytes received _RxFlags - Type of message Flag – either CAN_CONFIG_XTD_MSG or CAN_CONFIG_STD_MSG
Side Effects	W, STATUS, BSR, FSR0, FSR1 changed
Stack Requirement	3 level deep

Function	CANPrTxISR
Preconditions	None
Overview	If any data is pending for transmission then it copies highest priority data in empty buffer and requests Transmission.
Input	None
Output	None
Side Effects	W, STATUS, BSR changed
Stack Requirement	4 level deep

Function	CANPrTxIsTxPassive
Preconditions	None
Overview	Checks if transmit module is in error passive state.
Input	None
Output	C = 1, If CAN transmit module is in error passive state. C = 0, If CAN transmit module is in active state
Side Effects	STATUS changed
Stack Requirement	1 level deep

Function	CANPrTxIsRxPassive
Preconditions	None
Overview	Checks if receive module is in error passive state.
Input	None
Output	C = 1, If CAN receive module is in error passive state. C = 0, If CAN receive module is in active state
Side Effects	STATUS changed
Stack Requirement	1 level deep

Function	CANPrTxIsBusOff
Preconditions	None
Overview	Checks if CAN module is in bus off state.
Input	None
Output	C = 1, If CAN transmit module is in bus off state. C = 0, If CAN transmit module is in active state
Side Effects	STATUS changed
Stack Requirement	1 level deep

Function	CANPrTxIsBusOff
Preconditions	None
Overview	Checks if CAN module is in bus off state.
Input	None
Output	C = 1, If CAN transmit module is in bus off state. C = 0, If CAN transmit module is in active state
Side Effects	STATUS changed
Stack Requirement	1 level deep

Function	CANPrTxIsTxReady
Preconditions	None
Overview	Checks if transmit module is ready for new data
Input	None
Output	C = 1, If at least one CAN transmit buffer is empty. C = 0, If all CAN transmit buffer is full
Side Effects	W, STATUS and BSR changed
Stack Requirement	1 level deep

Function	CANPrTxIsRxReady
Preconditions	None
Overview	Checks if receive module has new data
Input	None
Output	C = 1, If at least one CAN receive buffer is empty. C = 0, If all CAN receive buffer is full
Side Effects	STATUS and BSR changed
Stack Requirement	1 level deep

7. Macros

Macro	<code>mDisableCANTxInt</code>
Overview	Disables CAN transmit interrupt.
Input	None
Output	None
Side Effects	W, STATUS changed
Stack Requirement	None

Macro	<code>mEnableCANTxInt</code>
Overview	Enables CAN transmit interrupt.
Input	None
Output	None
Side Effects	W, STATUS changed
Stack Requirement	None

Macro	<code>mCANPrTxAbortAll</code>
Overview	Aborts all pending transmission
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	<code>mCANPrTxGetTxErrCnt</code>
Overview	Gets transmit error count
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	<code>mCANPrTxGetTxErrCnt</code>
Overview	Gets receive error count
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	mCANPrTxInit
Overview	Initializes the CAN module
Input	<p>SJW - SJW value as defined in 18XXX8 datasheet (Must be between 1 thru 4)</p> <p>BRP - BRP value as defined in 18XXX8 datasheet (Must be between 1 thru 64)</p> <p>PHSEG1- PHSEG1 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)</p> <p>PHSEG2- PHSEG2 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)</p> <p>PROPSEG2 - PROPSEG2 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)</p> <p>Flags- Flag value of type CAN_CONFIG_CLASS</p>
Output	CAN bit rate is set. All masks registers are set '0' to allow all messages. Filter registers are set according to flag value. If (config & CAN_CONFIG_VALID_XTD_MSG). Set all filters to XTD_MSG Else if (config & CONFIG_VALID_STD_MSG). Set all filters to STD_MSG else Set half of the filters to STD while rests to XTD_MSG
Side Effects	Aborts all pending transmission. W, STATUS, BSR, PRODL, FSR0 changed.
Stack Requirement	3 level deep

Macro	mCANPrTxSetBaud
Overview	Initializes the CAN baud rate register
Input	<p>SJW - SJW value as defined in 18XXX8 datasheet (Must be between 1 thru 4)</p> <p>BRP - BRP value as defined in 18XXX8 datasheet (Must be between 1 thru 64)</p> <p>PHSEG1- PHSEG1 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)</p> <p>PHSEG2- PHSEG2 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)</p> <p>PROPSEG2 - PROPSEG2 value as defined in 18XXX8 datasheet (Must be between 1 thru 8)</p> <p>Flags- Flag value of type CAN_CONFIG_CLASS</p>
Output	CAN bit rate is set. All masks registers are set '0' to allow all messages. Filter registers are set according to flag value. If (config & CAN_CONFIG_VALID_XTD_MSG). Set all filters to XTD_MSG Else if (config & CONFIG_VALID_STD_MSG). Set all filters to STD_MSG else Set half of the filters to STD while rests to XTD_MSG
Side Effects	Aborts all pending transmission. W, STATUS, BSR, PRODL, FSR0 changed.
Stack Requirement	1 level deep

Macro	mCANPrTxSetOpMode
Overview	Sets the CAN operating mode by copying the given byte in CANCON. It ensures that given mode is set
Input	Operation mode code of type CAN_OP_MODE
Output	MCU is set to requested mode
Side Effects	This is a blocking call it will not return until the requested mode is set. W, STATUS changed
Stack Requirement	1 level deep
Macro	mCANPrTxSetOpModeNoWait
Overview	Sets the CAN operating mode by copying the given byte in CANCON. It ensures that given mode is set
Input	Operation mode code of type CAN_OP_MODE
Output	MCU is set to requested mode
Side Effects	This is a non-blocking call it will not verify if the requested mode is set. W changed
Stack Requirement	1 level deep
Macro	mCANPrTxSetReg
Overview	Bit adjusts the CAN ID value and loads it into the CAN register
Input	RegAddr - Starting address of a 32-bit buffer to be updated. Type REG_ADDR val - 32-bit value to be converted Flags – Type of message. Either CAN_CONFIG_XTD_MSG or CAN_CONFIG_STD_MSG
Output	Given CAN id value 'val' is bit adjusted and copied into corresponding PIC18CXX8 CAN registers
Side Effects	W, STATUS, BSR, FSR0 changed
Stack Requirement	2 level deep
Macro	mCANPrTxSetReg_IF
Overview	Bit adjusts the CAN ID value and loads it into the CAN register. Expects address pointer for Flags info
Input	Arguments: RegAddr - Starting address of a 32-bit buffer to be updated. Type REG_ADDR val - 32-bit value to be converted FlagsReg – Address of Register containing flags info. Either CAN_CONFIG_XTD_MSG or CAN_CONFIG_STD_MSG
Output	Given CAN id value 'val' is bit adjusted and copied into corresponding PIC18CXX8 CAN registers
Side Effects	W, STATUS, BSR, FSR0 changed
Stack Requirement	2 level deep

Macro	mCANPrTxSetReg_IV
Overview	Bit adjusts the CAN ID value and loads it into the CAN register. Expects address pointer for ID info
Input	<p>Arguments:</p> <p>RegAddr - Starting address of a 32-bit buffer to be updated. Type REG_ADDR</p> <p>var - Starting address of 32 bit value to be converted (LL:LH:HL:HH) format(Low -> High)</p> <p>Flags - Address of Register containing flags info. Either CAN_CONFIG_XTD_MSG or CAN_CONFIG_STD_MSG</p>
Output	Given CAN id value 'val' is bit adjusted and copied into corresponding PIC18CXX8 CAN registers
Side Effects	W, STATUS, BSR, FSR0 changed
Stack Requirement	2 level deep
Macro	mCANPrTxSetReg_IV_IF
Overview	Bit adjusts the CAN ID value and loads it into the CAN register. Expects address pointer for ID, Flags info and destination register address in FSR0
Input	<p>Arguments:</p> <p>RegAddr - Starting address of a 32-bit buffer to be updated. Type REG_ADDR</p> <p>var - Starting address of 32 bit value to be converted (LL:LH:HL:HH) format(Low -> High)</p> <p>FlagsReg - Type of message. Either CAN_CONFIG_XTD_MSG or CAN_CONFIG_STD_MSG</p>
Output	Given CAN id value 'val' is bit adjusted and copied into corresponding PIC18CXX8 CAN registers
Side Effects	W, STATUS, BSR, FSR0 changed
Stack Requirement	2 level deep
Macro	mCANPrTxSetReg_PREG_IV_IF
Overview	Bit adjusts the CAN ID value and loads it into the CAN register. Expects address pointer for Flags info and ID value, destination register address in FSR0
Input	<p>FSR0 - Starting address of a 32-bit buffer to be updated</p> <p>Arguments:</p> <p>var - Starting address of 32 bit value to be converted (LL:LH:HL:HH) format(Low -> High)</p> <p>FlagsReg - Type of message. Either CAN_CONFIG_XTD_MSG or CAN_CONFIG_STD_MSG</p>
Output	Given CAN id value 'val' is bit adjusted and copied into corresponding PIC18CXX8 CAN registers
Side Effects	W, STATUS, BSR, FSR0 changed
Stack Requirement	2 level deep

Macro	mCANPrTxSetReg_PREG_DV_IF
Overview	Bit adjusts the CAN ID value and loads it into the CAN register. Expects address pointer for Flags info, destination register address in FSR0 and ID value in _vReg1_O
Input	FSR0 – Starting address of a 32-bit buffer to be updated vReg1_O - Starting address of 32 bit value to be converted (LL:LH:HL:HH) format(Low -> High) Arguments: FlagsReg – Type of message. Either CAN_CONFIG_XTD_MSG or CAN_CONFIG_STD_MSG
Output	Given CAN id value 'val' is bit adjusted and copied into corresponding PIC18CXX8 CAN registers
Side Effects	W, STATUS, BSR, FSR0 changed
Stack Requirement	2 level deep
Macro	mCANPrTxReadReg
Overview	If given id is of type standard identifier, only SIDH and SIDL are read. If given id is of type extended identifier, EIDH, EIDL are read too.
Input	RegAddr – Starting address of the 32-bit buffer to be read. Type REG_ADDR Val – Starting address of the 32-bit buffer to store the read value.
Output	Corresponding PIC18xxx8 CAN id registers are read and value is bit adjusted and copied into 32-bit destination
Side Effects	W, STATUS, BSR, FSR0, FSR1 changed
Stack Requirement	2 level deep
Macro	mCANPrTxReadReg_PREG
Overview	Corresponding CAN id registers are read and value is bit adjusted and copied into 32-bit destination. Expects the source address in FSR0
Input	FSR0 – Starting address of the 32-bit buffer to be read. Arguments: Val – Starting address of the 32-bit buffer to store the read value.
Output	Corresponding PIC18xxx8 CAN id registers are read and value is bit adjusted and copied into 32-bit destination
Side Effects	W, STATUS, BSR, FSR0, FSR1 changed
Stack Requirement	2 level deep
Macro	mCANPrTxSendMsg
Overview	Copies the data in available transmit buffer according to priority
Input	msgID - 32-bit Message ID DataPtr - Starting address of data to be transmitted DataLngth - Data Length (Must be between 0 to 8) type - Type of message -CAN_TX_MSG_FLAGS type CAN_CONFIG_XTD_MSG or CAN_CONFIG_STD_MSG & CAN_TX_RTR_FRAME or CAN_TX_NO_RTR_FRAME
Output	W reg = 0, If buffer is full W reg = 1, If successful.
Side Effects	W, STATUS, BSR, FSR0, FSR1 changed
Stack Requirement	3 level deep

Macro	mCANPrTxSendMsg_IID_IDL_IF
Overview	Copies the data in available transmit buffer according to priority. Expects address pointer for ID, data length and Flags.
Input	msgID - 32-bit Message ID DataPtr - Starting address of data to be transmitted DataLngthPtr - Address of Register for Data Length storage FlagsReg - Address of Register containing flags info CAN_TX_MSG_FLAGS CAN_CONFIG_XTD_MSG or CAN_CONFIG_STD_MSG & CAN_TX_RTR_FRAME or CAN_TX_NO_RTR_FRAME
Output	W reg = 0, If buffer is full W reg = 1, If successful.
Side Effects	W, STATUS, BSR, FSR0, FSR1 changed
Stack Requirement	3 level deep

Macro	mCANPrTxReadMsg
Overview	Reads the received CAN data
Input	msgIDPtr - Starting address of 32-bit message ID storage DataPtr - Starting address for recd. data storage DataLngthPtr - Address of Register for Data Length storage Flags- Location to store flags info
Output	If any of the received data is available then ID, data, data length and flags information is returned at specified locations.
Side Effects	W, STATUS, BSR, FSR0, FSR1 changed
Stack Requirement	3 level deep

8. Error and Status Flags

Flag value of type `CAN_CONFIG_FLAGS`

This parameter can be any combination (ANDed together) of the following values:

Value	Meaning	Bit(s)	Position	Status ¹
<code>CAN_CONFIG_DEFAULT</code>	Specifies default flags			
<code>CAN_CONFIG_PHSEG2_PRG_ON</code>	Specifies to use supplied PHSEG2 value	1	<code>CAN_CONFIG_PHSEG2_PRG_BIT_NO</code>	Set
<code>CAN_CONFIG_PHSEG2_PRG_OFF</code>	Specifies to use maximum of PHSEG1 or Information Processing Time (IPT), whichever is greater	1	<code>CAN_CONFIG_PHSEG2_PRG_BIT_NO</code>	Clear
<code>CAN_CONFIG_LINE_FILTER_ON</code>	Specifies to use CAN bus line filter for wake-up	1	<code>CAN_CONFIG_LINE_FILTER_BIT_NO</code>	Set
<code>CAN_CONFIG_LINE_FILTER_OFF</code>	Specifies to not use CAN bus line filter for wake-up	1	<code>CAN_CONFIG_LINE_FILTER_BIT_NO</code>	Clear
<code>CAN_CONFIG_SAMPLE_ONCE</code>	Specifies to sample bus once at the sample point	1	<code>CAN_CONFIG_SAMPLE_BIT_NO</code>	Set
<code>CAN_CONFIG_SAMPLE_THRICE</code>	Specifies to sample bus three times prior to the sample point	1	<code>CAN_CONFIG_SAMPLE_BIT_NO</code>	Clear
<code>CAN_CONFIG_ALL_MSG</code>	Specifies to accept all messages including invalid ones	2	<code>CAN_CONFIG_MSG_BITS</code>	
<code>CAN_CONFIG_VALID_XTD_MSG</code>	Specifies to accept only valid Extended Identifier messages	2	<code>CAN_CONFIG_MSG_BITS</code>	
<code>CAN_CONFIG_VALID_STD_MSG</code>	Specifies to accept only valid Standard Identifier messages	2	<code>CAN_CONFIG_MSG_BITS</code>	
<code>CAN_CONFIG_ALL_VALID_MSG</code>	Specifies to accept all valid messages	2	<code>CAN_CONFIG_MSG_BITS</code>	

Flag value of type `CAN_OP_MODE`

This parameter can be any combination (ANDed together) of the following values:

Value	Meaning
<code>CAN_OP_MODE_NORMAL</code>	Specifies Normal mode of operation
<code>CAN_OP_MODE_SLEEP</code>	Specifies Sleep mode of operation
<code>CAN_OP_MODE_LOOP</code>	Specifies Loop-back mode of operation
<code>CAN_OP_MODE_LISTEN</code>	Specifies Listen Only mode of operation
<code>CAN_OP_MODE_CONFIG</code>	Specifies Configuration mode of operation

¹ If definition has more than one bit then Position symbol provides information for bit masking, ANDing it with the value will mask all the bits except required one. Status information is not provided as one needs to use ANDing and Oring to set/get value.

Type of message flag

This parameter can be only of the following values:

Value	Meaning	Bit(s)	Position/Status	Status
CAN_CONFIG_STD_MSG	Specifies Standard Identifier message	1	CAN_CONFIG_MSG_TYPE_BIT_NO	Set
CAN_CONFIG_XTD_MSG	Specifies Extended Identifier message	1	CAN_CONFIG_MSG_TYPE_BIT_NO	Clear

Type REG_ADDR

This parameter can be only of the following values:

Value	Meaning
CAN_MASK_B1	Specifies Receive Buffer 1 mask value
CAN_MASK_B2	Specifies Receive Buffer 2 mask value
CAN_FILTER_B1_F1	Specifies Receive Buffer 1, Filter 1 value
CAN_FILTER_B1_F2	Specifies Receive Buffer 1, Filter 2 value
CAN_FILTER_B2_F1	Specifies Receive Buffer 2, Filter 1 value
CAN_FILTER_B2_F2	Specifies Receive Buffer 2, Filter 2 value
CAN_FILTER_B2_F3	Specifies Receive Buffer 2, Filter 3 value
CAN_FILTER_B2_F4	Specifies Receive Buffer 2, Filter 4 value

Type CAN_TX_MSG_FLAGS

This parameter can be only of the following values:

Value	Meaning	Bit(s)	Position/Status	Status
CAN_TX_STD_FRAME	Specifies Standard Identifier message	1	CAN_TX_FRAME_BIT_NO	Set
CAN_TX_XTD_FRAME	Specifies Extended Identifier message	1	CAN_CONFIG_MSG_TYPE_BIT_NO	Clear
CAN_TX_NO_RTR_FRAME	Specifies regular message – not RTR	1	CAN_TX_RTR_BIT_NO	Set
CAN_TX_RTR_FRAME	Specifies RTR message	1	CAN_TX_RTR_BIT_NO	Clear

Type CAN_RX_MSG_FLAGS

This parameter can be only of the following values:

Value	Meaning	Bit(s)	Position/Status
CAN_RX_FILTER_1, CAN_RX_FILTER_2, CAN_RX_FILTER_3, CAN_RX_FILTER_4, CAN_RX_FILTER_5, CAN_RX_FILTER_6	Specifies receive buffer filter which caused this message to be accepted.	3	CAN_RX_FILTER_BITS
CAN_RX_OVERFLOW	Specifies receive buffer overflow condition	1	CAN_RX_OVERFLOW_BIT_NO
CAN_RX_INVALID_MSG	Specifies invalid message	1	CAN_RX_INVALID_MSG_BIT_NO
CAN_RX_XTD_FRAME	Specifies Extended message	1	CAN_RX_XTD_FRAME_BIT_NO
CAN_TX_RTR_FRAME	Specifies RTR message	1	CAN_RX_RTR_FRAME_BIT_NO
CAN_RX_DBL_BUFFERED	Specifies that this message was double buffered	1	CAN_RX_DBL_BUFFERED_BIT_NO