

ECAN™ (Polling) Module

- 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
- 6. Functions 5
- 8. Error and Status Flags..... 17

1. Introduction

This document describes programming interface to ECAN (Enhanced Controller Area Network) module. At the time of writing this document, only PIC18F8680/6680 family of microcontroller contained ECAN module. This module provides access to ECAN module in polling fashion. This module is completely written in 'C' language. It provides many customization options that may result in significant code reduction. To utilize this module, one must understand all options offered by ECAN module. This module is also available in Microchip Application Note AN878.

2. Module Features

- Out-of-box support for Microchip C18 and HI-TECH PICC-18™ C compilers
- Offers simple abstract interface to ECAN module for most applications
- Additional functions/macros are available for advanced applications
- Supports all three functional modes
- Provides access to all ECAN features in Polling mode
 - Easily modifiable to Interrupt-driven mode
- Operates in two main modes:
 - Run-time Library Mode and Fixed Library Mode
- Various compile-time options to customization routines to a specific application
 - Also available as Microchip Application Note AN878

3. List of Component Modules

ECAN.ex.txt	This is main test file developed to demonstrate use of the library functions.
ECANPoll.c	This is ECAN code implementation file. <u>One needs include this file in their project.</u>
ECANPoll.h	This file contains prototypes of functions and macros. One needs to include this file in every source file where ECAN functions will be called.
ECANPoll.def	This file contains all compile-time options for ECAN module. If you are using Maestro, this file will be created as per your option selections. This file is automatically included by ECANPoll.h file.

4. Using the Library Module in a Project

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

1. Use the 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 C18 Toolsuite or HITECH-PICC18 Toolsuite is selected (Project>Select Language Toolsuite).
5. In the Workspace view, right-click on the "Source Files" node. Select the "Add Files" option. Select ECANPoll.c 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 your project may require. Save and close the project.
8. In your main source file, add include directive at the head of the code listing to include ECANPoll.h. By doing so, all definitions 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

None

6. Functions

Function	<code>void ECANAbortAll(void)</code>
Preconditions	None
Overview	This macro requests that all transmit buffers are aborted.
Input	None
Output	None
Side Effects	None

Function	<code>void ECANDisableCANTX2(void)</code>
Preconditions	None
Overview	This macro disables CANTX2 pin and makes it a digital I/O pin.
Input	None
Output	None
Side Effects	None

Function	<code>BYTE ECANGetFilterHitInfo(void)</code>
Preconditions	<code>ECANReceiveMessage()</code> was called and returned TRUE
Overview	This function retrieves filter hit info for last fetched message. This function simply copies filter hit information available in SFR into local variable for later use.
Input	None
Output	Filter Hit Info – 0 through 15: 0 means RXF0, 1 means RXF1 and so on.
Side Effects	None

Function	<code>BYTE ECANGetFunctionalMode(void)</code>
Preconditions	None
Overview	This macro gets current functional mode.
Input	None
Output	ECAN Functional mode. Returned value will be one of the following: <code>ECAN_MODE_0</code> – if Mode 0 <code>ECAN_MODE_1</code> – if Mode 1 <code>ECAN_MODE_2</code> – if Mode 2
Side Effects	None

Function	ECAN_OP_MODE ECANGetOperationMode(void)
Preconditions	None
Overview	This function gets current ECAN operation mode.
Input	None
Output	ECAN_OP_MODE_NORMAL – if Normal mode ECAN_OP_MODE_SLEEP – if SLEEP mode ECAN_OP_MODE_LOOP – if Loopback mode ECAN_OP_MODE_LISTEN – if Listen only mode ECAN_OP_MODE_CONFIG – if Configuration mode
Side Effects	None
Function	BYTE ECANGetRxErrorCount(void)
Preconditions	None
Overview	This macro returns current RXERRCNT as defined by CAN spec.
Input	None
Output	Current value of receive error count.
Side Effects	None
Function	BYTE ECANGetTxErrorCount(void)
Preconditions	None
Overview	This macro returns current TXERRCNT as defined by CAN spec.
Input	None
Output	Current value of transmit error count
Side Effects	None
Function	void ECANInitialize(void)
Preconditions	None
Overview	This function initializes ECAN module as per Maestro options.
Input	None
Output	None
Side Effects	None
Function	BOOL ECANIsAllAborted(void)
Preconditions	None
Overview	This macro checks previously issued All Abort request.
Input	None
Output	TRUE: if there is no pending transmission FALSE: if abort is till in progress
Side Effects	None
Function	BOOL ECANIsBusOff(void)
Preconditions	None
Overview	This macro checks current ECAN module status.
Input	None
Output	TRUE: if the ECAN module is in the Bus Off state FALSE: if the ECAN module is not n the Bus Off state
Side Effects	None

Function	BOOL ECANIsRxPassive(void)
Preconditions	None
Overview	This macro checks to see if ECAN Receive module is in passive state.
Input	None
Output	TRUE: If the ECAN receive module is in receive error passive state FALSE: If the ECAN receive module is not in receive error passive state
Side Effects	None

Function	BOOL EANIsTxPassive(void)
Preconditions	None
Overview	This macro checks to see if ECAN Transmit module is in passive state.
Input	None
Output	TRUE: If ECAN module is in transmit error passive state. FALSE: If ECAN module is not in transmit error passive state
Side Effects	None

Function	EACNLinkRXF0F1ToBuffer(RXF0Buffer, RXF1Buffer) EACNLinkRXF2F3ToBuffer(RXF2Buffer, RXF3Buffer) EACNLinkRXF4F5ToBuffer(RXF4Buffer, RXF5Buffer) EACNLinkRXF6F7ToBuffer(RXF6Buffer, RXF7Buffer) EACNLinkRXF8F9ToBuffer(RXF8Buffer, RXF9Buffer) EACNLinkRXF10F11ToBuffer(RXF10Buffer, RXF11Buffer) EACNLinkRXF12F13ToBuffer(RXF12Buffer, RXF13Buffer) EACNLinkRXF14F15ToBuffer(RXF14Buffer, RXF15Buffer)	
Preconditions	Run-time library mode is selected OR Mode 1 / 2 with Fixed library mode is selected	
Overview	These macros link filters to buffers. There are a total of eight macros. Each macro links two filters at a time. EACNLinkRXF0F1ToBuffer links RXF0 and RXF1 filters to buffers. These macros are available in Mode 1 and Mode 2 only.	
Input	RXFnBuffer RXFmBuffer	- Name of buffer that is to be linked to RXFn - Name of buffer that is to be linked to RXFm
	The only permitted values for both parameters are: RXB0 - Link to RXB0 buffer RXB1 - Link to RXB1 buffer B0 - Link to B0 buffer B1 - Link to B1 buffer B2 - Link to B2 buffer B3 - Link to B3 buffer B4 - Link to B4 buffer B5 - Link to B5 buffer	
Output	None	
Side Effects	None	
Note	Buffer value must be a constant of permitted values only. A variable parameter would cause a compile-time error. For example, EACNLinkRXF0F1ToBuffer(myRXF0Buffer, myRXF1Buffer) would not compile.	

Function	void ECANLinkRXF0Thru3ToMask(m0, m1, m2, m3) void ECANLinkRXF4Thru7ToMask(m4, m5, m6, m7) void ECANLinkRXF8Thru11ToMask(m8, m9, m10, m11) void ECANLinkRXF12Thru15ToMask(m12, m13, m14, m15)
Preconditions	Run-time library mode is selected OR Mode 1 or 2 with Fixed library

Overview	mode is selected These macros link filters to masks. There are total of four macros. Each macro links four filters at a time. <code>ECANLinkRXF0Thru3ToMask</code> links RXF0, RXF1, RXF2 and RXF3 filters to masks. These macros are available in Mode 1 and Mode 2 only.
Input	m - Name of masks that is to be linked to RXFn
Output	Permitted values are: EACN_RXM0 - Link to RXM0 mask ECAN_RXM1 - Link to RXM1 mask ECAN_RXMF15 - Link to RXF15 mask
Side Effects	None
Note	None These macros perform compile-time operations to reduce generated code. If possible, always supply a constant value of permitted type. A variable argument will result in larger code.
Function	<pre>void ECANLoadRTRBuffer(BYTE buffer, unsigned long id, BYTE *data, BYTE dataLen, BYTE type)</pre>
Preconditions	Run-time library mode is selected OR Mode 1 / 2 with Fixed library mode is selected AND Supplied buffer must be configured for Automatic RTR handling.
Overview	This function loads the given message to the specified buffer that is configured for automatic RTR handling. This function is available in Mode 1 and Mode 2 only.
Input	buffer Programmable buffer number that is to be loaded. The possible values are 0 through 5 inclusive. Use 0 for B1, 1 for B1 and so on. id 32-bit identifier value, which may correspond to right justified 11-bit Standard Identifier or 29-bit Extended Identifier. The exact number of bits to use depends on type parameter. data Pointer to zero or more data bytes to send dataLen Number of bytes send type Specified enumerated value of the message type. The possible values are ECAN_MSG_STD for Standard Message ECAN_MSG_XTD for Extended Message
Output	TRUE: If given message was loaded into the given buffer FALSE: If given buffer was not setup for automatic RTR handling or is in the middle of automatic transmission.
Side Effects	None
Function	<pre>BOOL ECANReceiveMessage(unsigned long *id, BYTE *data,</pre>

	<pre> BYTE dataLen, ECAN_RX_MSG_FLAGS *msgFlags) </pre>
Preconditions	The id, data, dataLen and msgFlags pointer must point to the desired and valid memory locations.
Overview	This function copies one of the full receive buffer messages into the given buffer, and marks the full receive buffer as empty.
Input	<p>Id</p> <p>32-bit Identifier value, which may correspond to right justified 11-bit Standard Identifier or 29-bit Extended Identifier. The exact number of bits to use depends on msgFlags parameter</p> <p>data</p> <p>Pointer to zero or more data bytes to send</p> <p>DataLen</p> <p>Number of bytes to send</p> <p>MsgFlags</p> <p>Specifies an enumerated value of the type ECAN_RX_MSG_FLAGS. This represents the logical OR of one or more flags. The possible values of all variables are:</p> <p>ECAN_RX_OVERFLOW – Specifies Receive Buffer overflow</p> <p>ECAN_RX_INVALID_MSG – Specifies invalid message</p> <p>ECAN_RX_XTD_FRAME – Specifies Extended Identifier message</p> <p>ECAN_RX_STD_FRAME – Specifies Standard Identifier message</p> <p>ECAN_RX_DBL_BUFFERED – Specifies that this message was double buffered</p> <p>If a flag bit is set, the corresponding meaning is TRUE, if cleared, the corresponding meaning is FALSE.</p>
Output	<p>TRUE: If new message was copied to the given buffer</p> <p>FALSE: If no new message was found.</p>
Side Effects	None
Function	<pre> void ECANSendMessage(unsigned long id, BYTE *data, BYTE dataLen, ECAN_TX_MSG_FLAGS msgFlags) </pre>
Preconditions	None
Overview	This function copies the given message to one of the empty transmit buffers and marks it as ready to be transmitted.
Input	<p>id:</p> <p>32-bit identifier value, which may correspond to right justified 11-bit Standard Identifier or 29-bit Extended Identifier. The exact number of bits to use depends on msgFlags parameter</p> <p>data:</p> <p>Pointer to zero or more data bytes to send</p> <p>dataLen:</p> <p>Number of bytes to send</p> <p>msgFlags:</p> <p>Specifies an enumerated value of the ECAN_TX_MSG_FLAGS. This</p>

represents the logical OR of a Priority value, an Identifier type value and a Message type value. The possible values of all variables are:

Priority Value:

ECAN_TX_PRIORITY_0 – Specifies Transmit Priority 0
 ECAN_TX_PRIORITY_1 – Specifies Transmit Priority 1
 ECAN_TX_PRIORITY_2 – Specifies Transmit Priority 2
 ECAN_TX_PRIORITY_3 – Specifies Transmit Priority 3

Identifier Type Value:

ECAN_TX_STD_FRAME – Specifies Standard Identifier Message
 ECAN_TX_XTD_FRAME – Specifies Extended Identifier Message

Message Value:

ECAN_TX_NO_RTR_FRAME – Specifies Regular message – no RTR
 ECAN_TX_RTR_FRAME – Specifies RTR message

Output	TRUE: If given message was successfully placed in one of the empty transmit buffers
	FALSE: If all transmit buffers were full
Side Effects	None

Function Preconditions Overview Input	<p>ECANSetBaudRate(sjw, brp, phseg1, phseg2, propseg)</p> <p>ECAN module must be in Configuration mode This macro sets baud rate values.</p> <p>sjw SJW value - must be between 1 through 4</p> <p>brp BRP value – must be between 1 through 64</p> <p>phseg1 PHSEG2 value – must be between 1 through 8</p> <p>phseg2 PHSEG2 value – must be between 1 through 8</p> <p>propseg PROPSEG value – must be between 1 through 8</p>
Output Side Effects Note	<p>None</p> <p>None</p> <p>These macros perform compile-time operations to reduce generated code. If possible, always supply a constant value of permitted type. A variable argument will result in larger code.</p>

Function Preconditions Overview	<p>void ECANSetBnAutoRTRMode(BYTE mode)</p> <p>Run-time library mode is selected OR Mode 1 or 2 with Fixed library mode is selected</p> <p>This macro enables/disables Automatic RTR handling capability of the specified programmable buffer. There are total of 6 macros, one for each programmable buffer. For example, for Buffer B0, use</p>
---------------------------------------	---

Input	<p>ECANSetB0AutoRTRMode, for B1 use ECANSetB1AutoRTRMode and so on.</p> <p>Mode</p> <p>The only permitted values are:</p> <p>ECAN_AUTORTR_MODE_DISABLE – Disable AutoRTR mode</p> <p>ECAN_AUTORTR_MODE_ENABLE – Enable AutoRTR mode</p>
Output	None
Side Effects	None
Note	The selected buffer must also be setup as a transmit buffer using Maestro option, or by using ECANSetBnTxRxMode macro at run-time.
Function	<code>void ECANSetBnRxMode(buffer, mode)</code>
Preconditions	Run-time library mode is selected OR Mode 1 or 2 with Fixed library mode is selected
Overview	This macro sets receive mode for the programmable receive buffer. This macro is available in Mode 1 and Mode 2 only.
Input	<p>Buffer</p> <p>Name of programmable receive buffer that needs to be setup. The only permitted values are:</p> <p>B0 – Setup B0 buffer</p> <p>B1 – Setup B1 buffer</p> <p>B2 – Setup B2 buffer</p> <p>B4 – Setup B4 buffer</p> <p>B5 – Setup B5 buffer</p> <p>Mode</p> <p>Mode to setup. The only permitted values are:</p> <p>ECAN_RECEIVE_ALL_VALID – Receive all valid messages</p> <p>ECAN_RECEIVE_ALL – Receive all including invalid messages</p>
Output	None
Side Effects	None
Note	<p>Programmable buffer value must be a constant of permitted values only. A variable parameter would cause a compile-time error. For example, <code>ECANSetBnRxMode(myBuffer, ECAN_RECEIVE_ALL)</code> would not compile.</p>
Function	<code>void EANSetBnTxRxMode(buffer, mode)</code>
Preconditions	Run-time library mode is selected OR Mode 1 or 2 with Fixed library mode is selected
Overview	This macro sets transmit or receive mode for a specified buffer. This macro is available in Mode 1 and Mode 2 only.
Input	<p>Buffer</p> <p>Name of the programmable buffer that needs to be setup. The only permitted values are:</p> <p>B0 – Setup B0 buffer</p> <p>B1 – Setup B1 buffer</p> <p>B2 – Setup B2 buffer</p> <p>B3 – Setup B3 buffer</p> <p>B4 – Setup B4 buffer</p> <p>B5 – Setup B5 buffer</p>

	<p>Mode</p> <p>ECAN_BUFFER_RX – Buffer will be configured as receiver</p> <p>ECAN_BUFFER_TX – Buffer will be configured as transmitter</p>
Output	None
Side Effects	None
Note	Parameter buffer must be a constant of permitted values only. A variable parameter would cause a compile-time error. For example, <code>ECANSetBnTxRxMode(myBuffer, ECAN_BUFFER_TX)</code> would not compile.
Function	<code>void ECANSetBusSampleMode(BYTE mode)</code>
Preconditions	ECAN module must be in Configuration mode
Overview	This macro sets the CAN bus sampling mode.
Input	<p>mode</p> <p>The only permitted values are:</p> <p>ECAN_BUS_SAMPLE_MODE_THRICE – Specifies that the CAN bus be sampled three times</p> <p>ECAN_BUS_SAMPLE_MODE_ONCE – Specifies that the CAN bus be sampled once</p>
Output	None
Side Effects	None
Function	<code>void ECANSetCANTX2Mode(BYTE mode)</code>
Preconditions	None
Overview	This macro sets the CANTX2 pin source mode
Input	<p>Mode</p> <p>The only permitted values are:</p> <p>ECAN_TX2_SOURCE_COMP – Specifies complement of CANTX1 as source</p> <p>ECAN_TX2_SOURCE_CLOCK – Specifies CAN clock as source</p>
Output	None
Side Effects	None
Note	This macro automatically enables the CANTX2 pin as CANTX pin. Use <code>ECANDisableCANTX2</code> to configure CANTX2 pin as a digital I/O.
Function	<code>void ECANSetCaptureMode(BYTE mode)</code>
Preconditions	None
Overview	This macro enables the CAN timestamp mode.
Input	<p>mode</p> <p>The only permitted values are:</p> <p>ECAN_CAPTURE_MODE_ENABLE – Enables timestamp mode. CCP1 must be configured separately</p> <p>ECAN_CAPTURE_MODE_DISABLE – Disables timestamp mode</p>
Output	None
Side Effects	None

Function	<code>void ECANSetFilterMode(BYTE mode)</code>
Preconditions	ECAN module must be in Configuration mode
Overview	This macro sets the CAN bus wake-up filter mode.
Input	<p><code>mode</code></p> <p>The only permitted values are:</p> <p><code>ECAN_FILTER_MODE_DISABLE</code> – Specifies that low-pass filter be disabled</p> <p><code>ECAN_FILTER_MODE_ENABLE</code> – Specifies that the low-pass filter be enabled</p>
Output	None
Side Effects	None

Function	<code>void ECANSetFunctionalMode(BYTE mode)</code>
Preconditions	Run-time library mode is selected AND ECAN module is in Configuration mode
Overview	This macro changes the ECAN module functional mode.
Input	<p><code>mode</code></p> <p>The only permitted values are:</p> <p><code>ECAN_MODE_0</code> – Specifies Mode 0</p> <p><code>ECAN_MODE_1</code> – Specifies Mode 1</p> <p><code>ECAN_MODE_2</code> – Specifies Mode 2</p>
Output	None
Side Effects	None

Function	<code>void ECANSetOperationMode(ECAN_OP_MODE mode)</code>
Preconditions	None
Overview	This function changes the ECAN module operation mode.
Input	<p>Mode</p> <p>Specifies an enumerated value of the type <code>ECAN_OP_MODE</code>. The only permitted values are:</p> <p><code>ECAN_OP_MODE_NORMAL</code> – Specifies Normal mode of operation</p> <p><code>ECAN_OP_MODE_SLEEP</code> – Specifies SLEEP mode of operation</p> <p><code>ECAN_OP_MODE_LOOP</code> – Specifies Loopback mode of operation</p> <p><code>ECAN_OP_MODE_LISTEN</code> – Specifies Listen only mode of operation</p> <p><code>ECAN_OP_MODE_CONFIG</code> – Specifies Configuration mode of operation</p>
Output	None
Side Effects	None
Note	This is a blocking function. It waits for a given mode to be accepted by the ECAN module and then returns the control. If a non-blocking call is required, see the <code>ECANSetOperationModeNoWait</code> macro.

Function	<code>void ECANSetOperationModeNoWait(ECAN_OP_MODE mode)</code>
Preconditions	None
Overview	This macro changes the ECAN module operation mode.

Input	<p>Mode</p> <p>Specifies an enumerated value of the type <code>ECAN_OP_MODE</code>. The only permitted values are:</p> <p><code>ECAN_OP_MODE_NORMAL</code> – Specifies Normal mode of operation</p> <p><code>ECAN_OP_MODE_SLEEP</code> – Specifies SLEEP mode of operation</p> <p><code>ECAN_OP_MODE_LOOP</code> – Specifies Loopback mode of operation</p> <p><code>ECAN_OP_MODE_LISTEN</code> – Specifies Listen Only mode of operation</p> <p><code>ECAN_OP_MODE_CONFIG</code> – Specifies Configuration mode of operation</p>
Output	None
Side Effects	None
Note	<p>This is a non-blocking macro. It requests the given mode of operation and immediately returns the control. Caller must ensure the desired mode of operation is set before performing any mode-specific operation. If a blocking call is required, see the <code>ECANSetOperationMode</code> function.</p>
Function	<code>void ECANSetPHSEG2Mode(BYTE mode)</code>
Preconditions	The ECAN module must be in Configuration mode
Overview	This macro sets Phase Segment2 programmability mode.
Input	<p><code>mode</code></p> <p>The only permitted values are:</p> <p><code>ECAN_PHSEG2_MODE_AUTOMATIC</code> – Phase Segment2 will be automatically programmed by ECAN module.</p> <p><code>ECAN_PHSEG2_MODE_PROGRAMMABLE</code> – Phase Segment2 will be manually programmable.</p>
Output	None
Side Effects	None
Function	<code>void ECANSetRXB0DblBuffer(mode)</code>
Preconditions	Run-time library mode is selected OR Mode 0 with Fixed library mode is selected
Overview	This macro enables the hardware double buffering option for the RXB0 buffer. This macro is available in Mode 1 only.
Input	<p><code>mode</code></p> <p>The only permitted values are:</p> <p><code>ECAN_DBL_BUFFER_MODE_DISABLE</code> – Disable double buffering</p> <p><code>ECAN_DBL_BUFFER_MODE_ENABLE</code> – Enable double buffering</p>
Output	None
Side Effects	None
Function	<code>void ECANSetRXBnMode(buffer, mode)</code>
Preconditions	None
Overview	This macro sets the receive mode for the dedicated receiver buffer.
Input	<p><code>buffer</code></p> <p>Name of dedicated receive buffer that needs to be setup. The only permitted values are:</p> <p><code>RXB0</code> – Setup RXB0 buffer</p> <p><code>RXB1</code> – Setup RXB1 buffer</p>

	<p>mode</p> <p>Mode to setup. The only permitted values are:</p> <p>ECAN_RECEIVE_ALL_VALID – Receive all valid messages</p> <p>ECAN_RECEIVE_STANDARD – Receive only standard messages</p> <p>ECAN_RECEIVE_EXTENDED – Receive only extended messages</p> <p>ECAN_RECEIVE_ALL – Receive all including invalid messages</p>
Output	None
Side Effects	None
Note	Dedicated buffer value must be a constant of permitted values only. A variable parameter would cause a compile-time error. For example, <code>ECANSetRxBnRxMode(myBuffer, ECAN_RECEIVE_ALL)</code> would not compile.
Function	<code>void ECANSetRXFnValue(value, type)</code>
Preconditions	The ECAN module must be in Configuration mode. To use <code>ECANSetRxF6Value</code> through <code>ECANSetRxF15Value</code> , run-time library mode must be selected OR Mode 1 or 2 with Fixed library mode is selected
Overview	This macro sets the value for a filter register. There are total of 15 macros, one for each register. For example, for filter <code>RXF0</code> , use <code>ECANSetRxF0Value</code> , for <code>RXF1</code> use <code>ECANSetRxF1Value</code> and so on.
Input	<p>value</p> <p>Value to be set. Range of value is dependent on type</p> <p>type</p> <p>Type of filter. The only permitted values are:</p> <p>ECAN_MSG_STD – Standard type. 11-bit of value will be used</p> <p>ECAN_MSG_XTD – Extended type. 29-bit of value will be used</p>
Output	None
Side Effects	None
Note	In Mode 0, only <code>ECANSetRxF0Value</code> through <code>ECANSetRxF5Value</code> are available. In Mode 1 and Mode 2, <code>ECANRxF0Value</code> through <code>ECANRxF15Value</code> are available.
Function	<code>void ECANSetRXMnValue(value, type)</code>
Preconditions	The ECAN module must be in Configuration mode
Overview	The macro sets the value for a mask register. There are total of two macros, one for each mask. For example, for mask <code>RXM0</code> , use <code>ECANSetRXM0Value</code> , for <code>RXM1</code> use <code>ECANSetRXM1Value</code> and so on.
Input	<p>value</p> <p>Value to be set. Range of value is dependent on type</p> <p>Type</p> <p>Type of mask. The only permitted values are:</p> <p>ECAN_MSG_STD – Standard type: 11-bit of value will be used</p> <p>ECAN_MSG_XTD – Extended type – 29-bit value will be used</p>
Output	None
Side Effects	None

Function	<code>void ECANSetTxDriveMode(BYTE mode)</code>
Preconditions	None
Overview	This macro sets the CANTX pin recessive state drive mode.
Input	<p><code>mode</code></p> <p>The only permitted values are</p> <p><code>ECAN_TXDRIVE_MODE_TRISTATE</code> – Specifies that CANTX pin be driven to tri-state in recessive state</p> <p><code>ECAN_TXDRIVE_MODE_VDD</code> – Specifies that CANTX pin be driven to Vdd in recessive state</p>
Output	None
Side Effects	None
Function	<code>void ECANSetWakeupMode(BYTE mode)</code>
Preconditions	ECAN module must be in Configuration mode
Overview	This macro sets the CAN bus activity wake-up mode.
Input	<p><code>mode</code></p> <p>The only permitted values are:</p> <p><code>ECAN_WAKEUP_MODE_ENABLE</code> – Specifies that CAN bus activity wake-up mode be enabled</p> <p><code>ECAN_WAKEU_MOE_DISABLE</code> – Specifies that CAN bus activity wake-up mode be disabled</p>
Output	None
Side Effects	None

8. Error and Status Flags

None