Interrupt based CAN library module

1. Introduction

The purpose of the CAN library module is to get the user up to speed with CAN applications and not worry too much about the low-level CAN routines that are required for the CAN module to function and focus more on the target application. The CAN routines are interrupt driven which allows the processor to do parallel processing.

2. Module Features

- Supports user defined FIFO buffering on both transmission and reception
- Interrupt driven transmission and reception that allow for other tasks to operate in the foreground
- Supports both the MPLAB C18 and the HI-TECH PICC18 compiler
- Easy to use user functions

3. List of Component Modules

`CANIntC.C18.ex.txt`	This is a main test file that demonstrates how the functions could be used with the MPLAB C18 compiler
`CANInt.PICC18.ex.txt`	This is a main test file that demonstrates how the functions could be use with the HI-TECH PICC18 compiler.
`CAN.C`	This is CAN code implementation.
`CAN.H`	This is the header file for CAN.C
`CANDef.H`	This file contains the configurable parameters set-up by Microchip Application Maestro

4. Using the Library Module in a Project

To use the CAN Library Module then please follow the steps below

1. Use Application Maestro to configure the module as required
2. At the generate files step, save the out put files where your project resides
3. Launch MPLAB IDE and open the your project
4. Verify that either the MPLAB C18 or HI-TECH PICC18 toolsuite is selected
5. Add at least CAN.C into your project. Optionally you can add CAN.H and CANDef.H as well
6. If you are using the MPLAB C18 toolsuite then make sure that you have added an appropriate linker script
7. In your source file make sure that you include CAN.H
8. Use the functions or macros provided in the module as needed
   (Please note that if you have enabled CAN error handling you will need to provide the function void CANErrorHandler(void); which will perform the error handling for your specific application.)

5. List of Shared Parameters

Shared Data Bytes

RXBUF	An array of received CAN messages each message occupies 14 bytes of RAM data. User code must not modify this data in any way. User defines buffer length in `CANDef.h`.
TXBUF	An array of CAN messages that is pending in the message queue each message occupies 14 bytes of RAM data. User code must not modify this data in any way. User defines buffer length in `CANDef.h`.

Shared Functions

char CANOpen(unsigned char CONFIG1, unsigned char CONFIG2, unsigned char CONFIG3)
Configures the CAN module and sets up the masks and filters associated with the CAN module. Transmits "My Identifier" to notify that the module is on bus. The inparameters CONFIG1, CONFIG2 and CONFIG3 specifies what is to be put in BRGCON1, BRGCON2 and BRGCON3 respectively.

void CANISR(void)
CAN interrupt sub routine, transmits/receives data on the CAN bus. This function needs to be resided in the ISR.

char CANPut(struct CANMessage Message)
Puts a message in the FIFO buffer queue. The input is a structure that is defined in CAN.H.

char CANRXMessageIsPending(void)
Function used determine whether there is a message in the receive FIFO buffer queue that haven't been read.
struct CANMessage CANGet(void)
Pulls a message off the receive FIFO buffer

void CANSetMode(unsigned char Mode)
Sets the CAN module in the mode specified by the input parameter. Valid inputs are defined in CAN.H

Shared Macros

CANInit()
Calls OpenCAN with the default input parameters as specified by Application Maestro

Shared structures

```
struct CANMessage {
        unsigned long Address;
        unsigned char Data[8];
        unsigned char NoOfBytes;
        unsigned char Priority;
        unsigned Ext:1;
        unsigned Remote:1;
};
```

This is the format of the input parameter of CANPut() and return value of CANGet(). Please note that Priority does not hold any valid data when the structure is used as a return value from CANGet().

6. Functions

Function	char CANOpen(void)
Preconditions	None
Overview	Sets up the appropriate register for the device to act as a CAN node
Input	Values to be written into BRGCON1 → BRGCON3
Output	0 → Initialization succeeded
Side Effects	None

Function	void CANISR(void)
Preconditions	None
Overview	Checks if a CAN reception/transmission was complete and if so write/read to the CAN RX/TX FIFO buffers
Input	None
Output	None
Side Effects	Will modify the RX/TX interrupt flags and interrupt enable bits

Function	void CANGetMessage(void)
Preconditions	<WIN2:WIN0> in the CANCON register has to set to reflect the desired RXB registers
Overview	Gets the registers for a RXB and puts them in the CAN Receive buffer
Input	None
Output	None
Side Effects	Will modify the RX FIFO Write pointer (RXWPtr)

Function	char CANPutMessage(void)
Preconditions	<WIN2:WIN0> in the CANCON register has to set to reflect the desired TXB registers
Overview	Checks if there is any messages to transmit and if so place it in the registers reflected by <WIN2:WIN0>
Input	None
Output	0 → A new message has been put in the transmit queue 1 → There was no messages in the TX buffer to send
Side Effects	Will modify the TX buffer´s Read pointer (TXRPtr)

Function	char CANPut(struct CANMessage Message)
Preconditions	None
Overview	Initially checks if at least one buffer slot is available and if so push the requested message in the buffer. Checks if the TX modules are idle and if they are, reactivate one.
Input	A CAN message
Output	1 → Failed to put a CAN on the buffer, buffer is full 0 → The CAN message is put on the buffer
Side Effects	Will modify the TX Buffer register's Write pointer

Function	char CANRXMessageIsPending(void)
Preconditions	None
Overview	Checks if the RX Write pointer is equal to RX Read pointer and if so returns 0, else returns 1
Input	None
Output	1 → At least one received message is pending in the RX buffer 0 → No received messages are pending
Side Effects	None

Function	struct CANMessage CANGet(void)
Preconditions	An unread message has to be in the buffer use CANRXMessageIsPending(void) prior to calling this function in order to determine if an unread message is pending.
Overview	Pops the first message of the RX buffer
Input	None
Output	The received message
Side Effects	Will modify the RX Buffer register's Read pointer

Function	void CANSetMode(unsigned char Mode)
Preconditions	None
Overview	Requests to set the desired mode and waits until the mode has been set.
Input	Desired CAN Mode, could one of the following (CAN_LISTEN_MODE, CAN_LOOPBACK_MODE CAN_DISABLE_MODE, CAN_NORMAL_MODE)
Output	None
Side Effects	None

7. Macros

Macro	CANInit()
Overview	Calls OpenCAN() with the default inarguments
Input	None
Output	None
Side Effects	None