Interrupt based UART library module

(for C language)

1. Introduction

The UART general purpose library module provides ‘C’ functions, which helps the user to transmit and receive character data through the predifined FIFO (First-in-First-out) buffer . This module uses the interrupt based transmission and reception of the data. By using these functions user can concentrate on high-level application development rather than worrying about setting and clearing the bits of registers . It allows user to do other processing , rather than waiting a lot of time for the chunk of data to be transmitted or received.

2. Module Features

- Supports user-defined First-in, First-out (FIFO) buffers for both transmission and reception.
- Incorporates interrupt-driven transmission and reception, allowing user other tasks to execute in the foreground.
- Provides simple functions to read from and write to the buffers.
- It supports PIC18 family devices.

3. List of Component Modules

`UARTIntC.PIC18.ex.txt`	This is main test file developed to demonstrate use of the library functions.
`UARTIntC.c`	This is USART code implementation file. One needs to include this file in their project.
`UARTIntC.h`	This fine contains definition of shared parameters for user. One needs to include this file in their project.

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 language tool suite is selected (*Project>Select Language Tool-suite> Microchip C18Toolsuit*e).
5. In the Workspace view, right-click on the "Source Files" node. Select the "Add Files" option. Select UARTIntC.C and click **OK.**
6. Right-click on the "Header Files" node and select "Add Files". Select UARTIntC.h and click **OK**.
7. Now right-click on the "Linker Scripts" node and select "Add Files". Add the appropriate linker file (`.lkr`) for the project's target microcontroller.
8. Add any other files that the project may require. Save and close the project.
9. To use the module in your application, invoke the functions as needed.

5. List of Shared Parameters

Shared Data Bytes

`unsigned char vUARTIntTxBuffer[TX_BUFFER_SIZE]`	It represents user-defined transmit buffer. Data to be transmitted is stored here. User defines buffer length in `MPAM options.`
`unsigned char vUARTIntTxBufDataCnt`	It indicates the number of bytes transmit buffer containing.
`unsigned char vUARTIntTxBufWrPtr`	It indicates the writing position in to the transmit buffer.
`unsigned char vUARTIntTxBufRdPtr`	It indicates reading position pointer, which is used for actually placing data from buffer in to TXREG (USART peripheral)
`unsigned char vUARTIntRxBuffer[RX_BUFFER_SIZE]`	It represents user-defined receive buffer. Data received is stored here.
`unsigned char vUARTIntRxBufDataCnt`	It indicates the number of bytes receive buffer containing.
`unsigned char vUARTIntRxBufWrPtr`	It indicates the writing position in to the receive buffer.
`unsigned char vUARTIntRxBufRdPtr`	It is reading pointer in receive buffer.It keeps track of data which is being read from receive buffer in to user application.
`struct status vUARTIntStatus`	This structure contains error and status flags.

Shared Functions

`void UARTIntInit(void)`	It is used to initialise the serial port according to Application Maestro selection. It flushes the user defined FIFO buffers. It initialises error and status flags of `vUARTIntStatus` structure variable in to appropriate values.
`void UARTIntISR(void)`	This is Interrupt Service Routine function for serial(transmit & receive) interrupt. This need to be called from interrupt servie routine in user's main application at proper interrupt vector(low or high priority vector).
`unsigned char UARTIntPutChar(unsigned char)`	This function places the data to be transmitted in to user defined buffer. If the transmit buffer is full it returns with out doing any thing. Otherwise it places argument data in transmit buffer and and updates `vUARTIntRxBufDataCnt and vUARTIntTxBufWrPtr variables. If buffer becomes full it sets UARTIntTxBufferFull bit in vUARTIntStatus structure.`
`unsigned char UARTIntGetChar(unsigned char*)`	This function places received data from USART in to the user defined receive buffer and updates `vUARTIntRxBufWrPtr`.If the receive buffer is empty it sets `UARTIntbRxBufferFull bit in vUARTIntStatus. Other wise it returns the data in argument and updates vUARTIntRxBufDataCnt and vUARTIntRxBufRdPtr accordingly.`
`unsigned char UARTIntGetTxBufferEmptySpace(void)`	This function returns the number of bytes of free space left out in transmit buffer at the calling time of this function. It helps the user to further write data in to transmit buffer at once, rather than checking transmit buffer is full or not with every addition of data in to the transmit buffer.
`unsigned char UARTIntGetRxBufferDataSize(void)`	This function returns the number of bytes of data available in receive buffer at the calling time of this function. It helps the user to read data from receive buffer at once, rather than checking receive buffer is empty or not with every read of data from receive buffer.

Shared Macros

`mDisableUARTTxInt()`	Disables transmit interrupt.
`mEnableUARTTxInt()`	Enables transmit interrupt.
`mDisableUARTRxInt()`	Disables receive interrupt.
`mEnableUARTRxInt()`	Enables receive interrupt.
`mSetUARTRxIntHighPrior()`	Sets receive interrupt priority to high.
`mSetUARTRxIntLowPrior()`	Sets receive interrupt priority to low.
`mSetUARTTxIntHighPrior()`	Sets transmit interrupt priority to high.
`mSetUARTTxIntLowPrior()`	Sets transmit interrupt priority to low.
`mSetUART_BRGHHigh()`	Sets BRGH bit.
`mSetUART_BRGHLow()`	Resets BRGH bit.
`mSetUART_SPBRG(iBRGValue)`	Sets SPBRG register value as the argument passed.
`mSetUARTBaud(iBaudRate)`	Sets the baudrate of USART to the argument passed.

6. Functions

Function	`void UARTIntInit(void)`
Preconditions	None
Overview	This function initialises USART peripheral.This function need to be called before using UARTIntPutChar and UARTIntGetChar functions to send and receive the characters. This function clears the user defined buffers and initialises error and status flags to appropriate values.
Input	None
Output	None
Side Effects	None
Stack Requirement	1 level deep

Function	`void UARTIntISR(void)`
Preconditions	UARTIntInit() function should have been called.
Overview	This is the Interrupt service routine which is called in the user application's ISR portion.This function actually sends the data from transmit buffer to USART and updates the data count and read pointer variables of transmit buffer. For the receive portion, it reads the data from USART and places the data in to receive buffer (if no errors occured) and updates data count and write pointer variables of receive buffer.
Input	None
Output	None
Side Effects	None
Stack Requirement	2 level deep

Function	`unsigned char UARTIntGetChar(unsigned char*)`
Preconditions	UARTIntInit() function should have been called.
Overview	This function reads the data from the receive buffer. It places the data in to argument and updates the data count and read pointer variables of receive buffer.
Input	unsigned char*
Output	`unsigned char` 0 - receive buffer is empty and the character could not be read from the receive buffer. 1 - single character is successfully read from receive buffer.
Side Effects	None
Stack Requirement	1 level deep

Function	`unsigned char UARTIntPutChar(unsigned char)`
Preconditions	UARTIntInit() function should have been called.
Overview	This function puts the data in to transmit buffer. Internal implementation wise , it places the argument data in transmit buffer and updates the data count and write pointer variables.
Input	unsigned char
Output	`unsigned char` 0 - single character is successfully added to transmit buffer 1 - transmit buffer is full and the character could not be added to transmit buffer.
Side Effects	None
Stack Requirement	1 level deep

Function	`unsigned char UARTIntGetTxBufferEmptySpace(void)`
Preconditions	UARTIntInit() function should have been called.
Overview	This function returns the number of bytes of free space left out in transmit buffer at the calling time of this function. It helps the user to further write data in to transmit buffer at once, rather than checking transmit buffer is full or not with every addition of data in to the transmit buffer.
Input	None
Output	`unsigned char` 0 - There is no empty space in transmit buffer. number - the number of bytes of empty space in transmit buffer.
Side Effects	None
Stack Requirement	1 level deep

Function	`unsigned char UARTIntGetRxBufferDataSize(void)`
Preconditions	UARTIntInit() function should have been called.
Overview	This function returns the number of bytes of data available in receive buffer at the calling time of this function. It helps the user to read data from receive buffer at once, rather than checking receive buffer is empty or not with every read of data from receive buffer.
Input	None
Output	`unsigned char` number - the number of bytes of data in receive buffer.
Side Effects	None
Stack Requirement	1 level deep

7. Macros

Macro	`mDisableUARTTxInt()`
Overview	Disables transmit interrupt
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mEnableUARTTxInt()`
Overview	Enables transmit interrupt
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mDisableUARTRxInt()`
Overview	Disables receive interrupt
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mEnableUARTRxInt()`
Overview	Enables receive interrupt
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUARTRxIntHighPrior()`
Overview	Sets receive interrupt priority to high
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUARTRxIntLowPrior()`
Overview	Sets receive interrupt priority to low
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUARTTxIntHighPrior()`
Overview	Sets transmit priority to high
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUARTTxIntLowPrior()`
Overview	Sets transmit priority to low
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUART_BRGHHigh()`
Overview	Sets BRGH bit in TXSTA
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUART_BRGHLow()`
Overview	Clears BRGH bit in TXSTA
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUART_SPBRG(iBRGValue)`
Overview	Sets the SPBRG register value to the argument value passed.
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUARTBaud(iBaudRate)`
Overview	Sets the baud rate of UART to the argument value passed.
Input	None
Output	None
Side Effects	None
Stack Requirement	None

8. Error and Status Flags

All errors/status are set as a bit flag in structure variable named `UARTIntStatus`. Individual bit flag indicates different errors. Please refer below list for the information.

`UARTIntTxBufferFull`	This bit is set when transmit buffer is full. When data gets tranmitted it is cleared.
`UARTIntTxBufferEmpty`	It is set when transmit buffer is empty. If any item is present in the buffer this bit is cleared
`UARTIntRxBufferFull`	It indicated receive buffer is full of data. User application starts reading from receive buffer with the help of shared function, it gets cleared.
`UARTIntRxBufferEmpty`	It is set when receive buffer is empty. If any item is received its cleared.
`UARTIntRxOverFlow`	It indicates receive buffer is full and UART is still receiving the data. It indicates in between data is missing. It gets cleared once the receive buffer is being read.
`UARTIntRxError`	It indicates either OERR or FERR occurred. User needs to clear this error-bit (UARTIntRxError) in their firmware.