Interrupt based UART library module

1. Introduction

The primary objective of UART 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.

2. Module Features

- Supports user-defined sized 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.
- Supports both PIC16 and PIC18 family devices.

3. List of Component Modules

`UARTInt.PIC16.ex.txt`	This is main test file developed to demonstrate the usage of the library functions for PIC16xxx family.
`UARTInt.PIC18.ex.txt`	This is main test file developed to demonstrate the usage of the library functions for PIC18xxx family.
`UARTInt.asm`	This is UART code implementation file. One needs include this file in their project.
`16UartI.asm`	This is USART code implementation file for PIC16xxx family. UARTInt.asm file selects this file depending on the processor family selection
`18UartI.asm`	This is USART code implementation file for PIC18xxx family. UARTInt.asm file selects this file depending on the processor family selection
`UARTInt.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.
`P16xxx.inc`	General purpose processor definition file for PIC16xxx family
`P18xxx.inc`	General purpose processor definition file for PIC18xxx 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 UARTInt.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 `UARTInt.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

`vUARTIntTxBuffer` It is starting location of Transmit Buffer. Data to be transmitted is stored here.

`vUARTIntTxBufDataCnt` It provides information regarding total pending data in `vUARTIntTxBuffer`.

`vUARTIntRxBuffer` It is starting location of Receive Buffer. The received data is stored here.

`vUARTIntRxBufDataCnt` It provides information regarding total available data in `vUARTIntRxBuffer`.

`vUARTIntStatus` It contents different Error/Status flag bits. Please refer section 8. Error and Status Flags for more information

Shared Functions

`UARTIntInit` It is used for UART Initialization It initializes the serial port according to Application Maestro selection and flushes the Rx and Tx buffer. It clears all UART errors

`UARTIntISR` This is an Interrupt service routine for Serial (Transmit & Receive) interrupt. It handles reception and transmission of data on interrupt. Call it from Interrupt service routine at proper Interrupt Vector (High or Low priority Vector)

`UARTIntPutCh` It writes Content of W reg. in Transmit Buffer. If vUARTIntTxBuffer is already empty then it immediately transmits the data. If Buffer is already full then it returns without any job. Otherwise it puts the data to be transmitted in vUARTIntTxBuffer and accordingly adjusts vUARTIntTxBufDataCnt. If buffer becomes full because of new data then it will set UARTIntTxBufFul bit.

`UARTIntGetCh` It reads data in Receive Buffer. If vUARTIntRxBuffer is empty then it will set UARTIntRxBufEmpty bit in vUARTIntStatus. Otherwise it returns received data in W reg, and accordingly adjusts vUARTIntRxBufDataCnt

Shared Macros

`mDisableUARTTxInt`	Disables Transmit interrupt.
`mEnableUARTTxInt`	Enables Transmit interrupt.
`mDisableUARTRxInt`	Disables receive interrupt.
`mEnableUARTRxInt`	Enables receive interrupt.
`mSetUARTRxIntHighPrior`	Sets high priority for receive interrupt (only for PIC18)
`mSetUARTRxIntLowPrior`	Sets low priority for receive interrupt (only for PIC18)
`mSetUARTTxIntHighPrior`	Sets high priority for transmit interrupt (only for PIC18)
`mSetUARTTxIntLowPrior`	Sets low priority for transmit interrupt (only for PIC18)
`mSetUART_BRGHHigh`	Sets BRGH bit
`mSetUART_BRGHLow`	Resets BRGH bit
`mSetUART_SPBRG`	Loads SPBRG register with argument
`msetUARTBaud`	Loads SPBRG with calculated value for required baud rate

6. Functions

Function UARTIntInit
Preconditions None
Overview It is used for UART Initialization It initializes the serial port according to Application Maestro selection and flushes the Rx and Tx buffer. It clears all UART errors
Input None
Output None
Side Effects Databank, W changed
Stack Requirement 1 level deep

Function UARTIntISR
Preconditions None
Overview This is an Interrupt service routine for Serial (Transmit & Receive) Interrupt. It handles Reception and Transmission of data on interrupt. Call it from Interrupt service routine at proper Interrupt Vector (High or Low priority Vector)
Input None
Output If data is received it puts it in vUARTIntRxBuffer and accordingly adjusts the vUARTIntRxBufDataCnt and clears UARTIntRxBufEmpty flag. If Receive Buffer becomes full then it will set UARTIntRxBufFul bit. If data is received when Receive buffer was full it will set UARTIntRxBufOF flag to indicate that transmitted data has been missed because of full vUARTIntRxBuffer. If any error is generated in reception it will set UARTIntRxError flag bit.

If last data is transmitted then it will transmit next pending data if any. It will accordingly adjust the vUARTIntTxBufDataCnt. It will clear the UARTIntTxBufFul bit to indicate space for data in vUARTIntTxBuffer.
Side Effects Databank changed
Stack Requirement 1 level deep

Function UARTIntPutCh
Preconditions None
Overview It writes Content of W reg. in Transmit Buffer. If vUARTIntTxBuffer is already empty then it immediately transmits the data. If Buffer is already full then it returns without any job. Otherwise it puts the data to be transmitted in vUARTIntTxBuffer and accordingly adjusts vUARTIntTxBufDataCnt. If buffer becomes full because of new data then it will set UARTIntTxBufFul bit.
Input W reg = data to be transmitted
Output If vUARTIntTxBuffer becomes full it sets UARTIntTxBufFul flag bit in UARTIntStatus
Side Effects Databank, FSR, W and STATUS changed
Stack Requirement 1 level deep

Function	`UARTIntGetCh`
Preconditions	None
Overview	It reads data in Receive Buffer. If vUARTIntRxBuffer is empty then it will set UARTIntRxBufEmpty bit in vUARTIntStatus. Otherwise it returns received data in W reg, and accordingly adjusts vUARTIntRxBufDataCnt
Input	None
Output	W reg = received data byte If receive buffer is empty then UARTIntRxBufEmpty=1. UARTIntRxBufEmpty is defined in vUARTIntStatus.
Side Effects	Databank, FSR, W and STATUS changed
Stack Requirement	1 level deep

7. Macros

Macro	`mDisableUARTTxInt`
Overview	Disables transmit interrupt
Input	None
Output	None
Side Effects	Data bank changed (for PIC16) None (for PIC18)
Stack Requirement	None

Macro	`mEnableUARTTxInt`
Overview	Enables transmit interrupt
Input	None
Output	None
Side Effects	Data bank changed (for PIC16) None (for PIC18)
Stack Requirement	None

Macro	`mDisableUARTRxInt`
Overview	Disables receive interrupt
Input	None
Output	None
Side Effects	Data bank changed (for PIC16) None (for PIC18)
Stack Requirement	None

Macro	`mEnableUARTRxInt`
Overview	Enables receive interrupt
Input	None
Output	None
Side Effects	Data bank changed (for PIC16) None (for PIC18)
Stack Requirement	None

Macro	`mSetUARTRxIntHighPrior`
Overview	Set higher priority for receive interrupt (PIC18 only)
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUARTRxIntLowPrior`
Overview	Set lower priority for receive interrupt (PIC18 only)
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUARTTxIntHighPrior`
Overview	Set higher priority for transmit interrupt (PIC18 only)
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUARTTxIntLowPrior`
Overview	Set lower priority for transmit interrupt (PIC18 only)
Input	None
Output	None
Side Effects	None
Stack Requirement	None

Macro	`mSetUART_BRGHHigh`
Overview	Sets BRGH bit in TXSTA register.
Input	None
Output	None
Side Effects	Data bank changed (for PIC16) None (for PIC18)
Stack Requirement	None

Macro	`mSetUART_BRGHLow`
Overview	Clears BRGH bit in TXSTA register.
Input	None
Output	None
Side Effects	Data bank changed (for PIC16) None (for PIC18)
Stack Requirement	None

Macro	`mSetUART_SPBRG`
Overview	Load SPBRG register with argument.
Input	Value to be loaded in SPBRG register
Output	None
Side Effects	Data bank, W changed (for PIC16) W changed (for PIC18)
Stack Requirement	None

Macro	`mSetUARTBaud`
Overview	Loads SPBRG with calculated value for required baud rate
Input	Baud rate value as argument
Output	None
Side Effects	Data bank, W changed (for PIC16) W (for PIC18)
Stack Requirement	None

8. Error and Status Flags

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

`UARTIntTxBufFul` For transmit buffer full indication. Bit is set when transmit buffer is full. It gets reset when data is transmitted to indicate space in buffer.

`UARTIntRxBufFul` For receive buffer full indication. Bit is set when receive buffer is full. It gets reset when data is read from buffer to indicate space in buffer.

`UARTIntRxBufOF` Data buffer over flow indication bit. It indicates that data received when vUARTIntRxBuffer was full. Therefore, it indicates that in between data is missing. It gets reset when data is read from buffer.

`UARTIntRxBufEmpty` For receive buffer empty indication. Bit is set when receive buffer is empty to indicate that no data is there to read. It gets reset when data is received to indicate presence of data

`UARTIntRxError` It is used to indicate some error in data reception. It reflects the error generated by FERR (framing error) and OERR (overrun error). User needs to clear this error-bit (UARTIntRxError) in their firmware