Modifier Version Date Changes

Skylar Graika v0.0 2/15/11 Created

Skylar Graika v0.1 2/24/11 Revised Functions

Skylar Graika v0.2 3/01/11 Revised Functions

UART Library Reference Manual

**Introduction:**

This document describes how the universal asynchronous receiver/transmitter library shall be used. This document describes the capabilities of the UART Library and the API functions used to access its features. The UART library allows the user to effectively configure UART ports on various pic32 boards (i.e. MX4 & MX7) with little effort. This reference manual will also break down the functions available to the UARTs and how they shall be used.

**UART Overview:**

The universal asynchronous receiver/transmitter are used in embedded hardware to transmit and receive serial data. This is done 1 bit at a time through serial communication. An example use for a UART is with a PmodCLS, bits of data can be transmitted to this device and as well received from. The baud rate generator is a component within a UART that produces the internal clock for which its receiving cycle relies on.

**UART Availability & Proper Usage:**

The number or UART ports available to the programmer are completely dependent on the piece of hardware being used. For example, the PIC32MX460F12L has 2 UART ports (UART1 & UART2). UARTs are allowed to transmit and receive information at the same time hence is a full-duplex communication system. Additionally, a UART is considered to be configurable since the speeds at which is transmits and receives may be adjusted.

**UART Library Goals:**

Since PIC32 microcontrollers are constantly being developed it is imperative for this library to have the capability of configuring the UARTs of any device (within its realm of PIC32 devices), and as well adjust the function names available to each UART port.

**UART Library Functions:**

**Description**

The standard usage functions for UARTs such as initializing, sending and receiving data shall all be found here. For each UART port available a buffer will be assigned to it. Therefore, if there are N UARTs there will also be N buffers. Each function ends with ‘n’, which refers to the UART port the function is referring to. This is a necessity to the UART library because macros are used to generate the proper amount of UART functions, and because each board can have a different amount of UARTs an N amount is required.

**Legacy UART wrappers (located in legacy\_uartX.c)**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void fnUARTStandardSetup( unsigned long sysClock,**

**UARTCHANNEL uartChannel )**

Parameters: sysClock -System clock speed in Hz

uartChannel -Channel that UART is on

This function initializes the PB Clock, and opens the specified UART passed in uartChannel for standard configurations. (Rx/Tx) This function will configure UARTs on both the pic32mx4 and pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void fnUARTEnableDigitalPinIO( UARTCHANNEL uartChannel )**

Parameters: uartChannel -Channel that UART is on

This function enables the corresponding pins to the UART channel provided. This function will configure to UARTs on both the pic32mx4 and pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void fnUARTputs( const char\* buffer,**

**UARTCHANNEL uartChannel )**

Parameters: buffer -A Null terminated string of characters

uartChannel -Channel that UART is on

This function is a wrapper for the legacy\_uart library that allows one function to adapt to both the pic32mx4 and pic32mx7 and transmit a string of characters. The parametres uartChannel specifies which UART is to transmit the string.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void fnUARTgets( unsigned int length,**

**char\* buffer,**

**unsigned int uart\_data\_wait,**

**UARTCHANNEL uartChannel )**

Parameters: length -The length of characters to receive

buffer -A pointer to a buffer where data will be received

uart\_data\_wait -Time-out count, if surpassed then UART stops receiving

uartChannel -Channel that UART is on

This function allows a UART to receive data from a specified buffer. The amount to read

is determined by the *length* parameter. If the attempt to read surpasses the

*uart\_data\_wait* parameter then the UART terminates receiving from the buffer. The

parameter uartChannel specifies which UART is to transmit the string. This function is

configured to support UARTs on both the pic32mx4 and pic32mx7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**int fnUARTbusy( UARTCHANNEL uartChannel )**

Parameters: uartChannel -Channel that UART is on

This function checks to see if the UARTs transmitter is busy. The parameter uartChannel is passed in to specify which UART to check. This function returns a ‘0’ if the UARTs transmitter is not busy, any non-zero value returned indicates it’s busy. This function is configured to support UARTs on both the pic32mx4 and pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**int fuUARTdataRdy( UARTCHANNEL uartChannel )**

Parameters: uartChannel -Channel that UART is on

This function checks the UART if there is new data to be read from the receive buffer. The parameter uartChannel is passed in to specify which UART to check. This function returns a ‘0’ if there is no new data in the receive buffer, any non-zero value returned indicates that there is new data waiting to be read. This function is configured to support UARTs on both the pic32mx4 and pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void fnUARTclose( UARTCHANNEL uartChannel )**

Parameters: uartChannel -Channel that UART is on

This function closes the UART and disables its interrupt. The parameter uartChannel is

passed in to specify which UART to close. This function is configured to work with both

the pic32mx4 and pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void fnUARTwrite( unsigned int data,**

**UARTCHANNEL uartChannel )**

Parameters: data -A character to be transmitted

uartChannel -Channel that UART is on

This function transmits a character over the UART. The parameter data is the character

in which to be transmitted. The parameter uartChannel is passed in to specify which

UART to transmit over. This function is configured to work with both the pic32mx4 and

pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**char fnUARTread( UARTCHANNEL uartChannel )**

Parameters: uartChannel -Channel that UART is on

This function reads a character from the receive buffer associated with the UART. The parameter uartChannel is passed in to specify which UART to read a character from. After reading a character from the receive buffer, the character is returned. This function is configured to work with both the pic32mx4 and pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void fnUARTenableAutoAddr( int address,**

**UARTCHANNEL uartChannel )**

Parameters: address -The 9-bit address associated with this UART

uartChannel -Channel that UART is on

This function the auto address matching mode on the UART. The parameter address is passed in to specify which address the UART is to match. The parameter uartChannel is passed in to specify which UART to enable. This function is configured to work with both the pic32mx4 and pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void fnUARTsendBreak( UARTCHANNEL uartChannel )**

Parameters: uartChannel -Channel that UART is on

This function initiates a break sequence on the UART. The parameter uartChannel is passed in to specify which UART to initiate the break sequence on. This function is configured to work with both the pic32mx4 and pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**int fnUARTgetErrors( UARTCHANNEL uartChannel )**

Parameters: uartChannel -Channel that UART is on

This function retrieves the errors on the UART and returns them. The parameter uartChannel is passed in to specify which UART to retrieve the errors from. There are three different types of errors that can be retrieved:

Overflow error - \_UxSTA\_OERR\_MASK

Frame error - \_UxSTA\_FERR\_MASK

Parity error - \_UxSTA\_PERR\_MASK

This function is configured to work with both the pic32mx4 and pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**int fnUARTclearError( int error,**

**UARTCHANNEL uartChannel )**

Parameters: error -Bits associated with the error

uartChannel -Channel that UART is on

This function clears an error on the UART. The error in which to be removed is passed in as one of the following defined constants:

Overflow error - \_UxSTA\_OERR\_MASK

Frame error - \_UxSTA\_FERR\_MASK

Parity error - \_UxSTA\_PERR\_MASK

The parameter uartChannel is passed in to specify which UART to remove the error from. This function is configured to work with both the pic32mx4 and pic32mx7.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**int fnUARTclearAllErrors( UARTCHANNEL uartChannel )**

Parameters: uartChannel -Channel that UART is on

This function clears all errors associated with the UART. The errors in which to be removed are of the type:

Overflow error - \_UxSTA\_OERR\_MASK

Frame error - \_UxSTA\_FERR\_MASK

Parity error - \_UxSTA\_PERR\_MASK

The parameter uartChannel is passed in to specify which UART to remove the errors from. This function is configured to work with both the pic32mx4 and pic32mx7.

**Pic32MX UART Library wrapper functions (located in 32MX\_uartX.c):**

**int fnUARTOpen\_basicRxTx(UART\_Module id)**

Parameters: id -Id associated with the UART

This function is used to open the UART and enable it for basic read and transfer capabilities. The parameter *id* is passed in to specify which uart to open. A 1 is returned on sucess, and -1 is returned otherwise.

**int fnUARTOpen\_basicRx(UART\_MODULE id)**

Parameters: id -Id associated with the UART

This function is used to open the UART and enable it for basic read capabilities. The parameter *id* is passed in to specify which uart to open. A 1 is returned on sucess, and -1 is returned otherwise.

**int fnUARTOpen\_basicTx(UART\_MODULE id)**

Parameters: id -Id associated with the UART

This function is used to open the UART and enable it for basic transfer capabilities. The parameter *id* is passed in to specify which uart to open. A 1 is returned on sucess, and -1 is returned otherwise.

**UART PMODs Beginning libraries**

**Pmod\_CLS ( located in Pmod\_CLS.c )**

Description of use:

This helper file provides the user with the basis for setting up the Pmod CLS. The user specifies which set of pins the UART is connected to ( for PIC32MX4 either JH or JE). The Pmod CLS is associated with UART2 and is setup to receive input. This helper file always provides the basis for setting up another UART to act as a transmitter. To adapt the file to allow another UART as a transmitter, uncomment the specified lines in the file. A basic usage scenario when two UARTs are being used, one as the receive and one as the transmitter is also provided. It too can be uncommented to provide functionality.

Issues of importance:

The default baud rate for this file is set to 9600. Be aware that the baud rate for the Pmod CLS should also be configured to 9600. The baud rate associated with this basis code can be altered to comply with the setup on the Pmod. To specify the correct pins the defined variables *\_UART\_JE\_* and  *\_UART\_JH\_* must be giving either a 1 or 0 meaning in use or not in use, repsectively.

Additional comments:

The code in Pmod\_CLS.c uses functions from the legacy uart libraries.