STM32F4xx UART Library

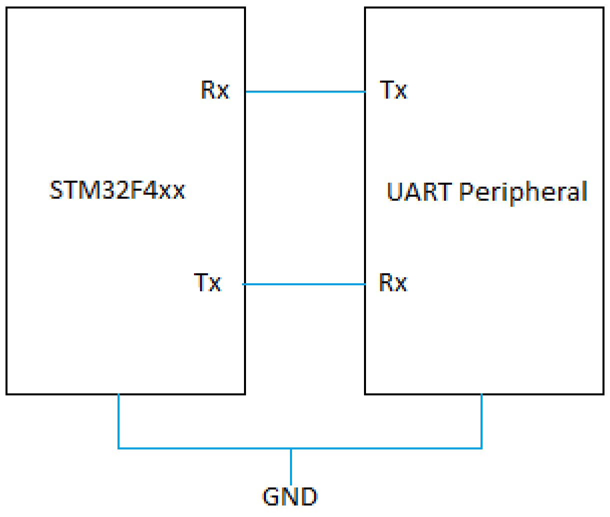
Valid for STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs

Introduction

UART (Universal Asynchronous Receiver Transmitter) is a *serial* communication protocol. This protocol is implemented on hardware through RS-232, USB, etc.

UART has two pins, Tx and Rx. There are two more optional pins, RTS and CTS, which are used for hardware flow control.

Connection Diagram



Pin Configuration

Peripheral	Tx Pin	Rx Pin
UART_1	PA9	PA10
UART_2	PA2	PA3
UART_3	PB10	PB11
UART_4	PA0	PA1
UART_5	PC12	PD2
UART_6	PC6	PC7

Function Documentation

Function	Return Type	Parameters	Description
`UARTx_Init(UARTx n, uint8_t data_bits, double STOP, _Bool HDSEL, uint32_t baud_rate)`	*void*	*n* = UART_1, UART_2, UART_3, UART_4, UART_5 or UART_6	Initializes the selected UART peripheral, with the given configurations. Also initializes the respective GPIO pins.

		data_bits = 8 or 9 *STOP* = 1, 0.5, 2 or 1.5 *baud_rate* = 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 or 230400	
`UARTx_En(UARTx n)`	*void*	...	Enables the selected UART peripheral.
`UARTx_Dis(UARTx n)`	*void*	...	Disables the selected UART peripheral.
`UARTx_Tx_En(UARTx n)`	*void*	...	Enables the selected UART transmission block.
`UARTx_Tx_Dis(UARTx n)`	*void*	...	Disables the selected UART transmission block.
`UARTx_Rx_En(UARTx n)`	*void*	...	Enables the selected UART receiver block.
`UARTx_Rx_Dis(UARTx n)`	*void*	...	Disables the selected UART receiver block.
`UARTx_Tx_char(UARTx n , char ch)`	*void*	... *ch* is the character to be sent	Sends a single character over the selected UART peripheral.
`UARTx_Tx_int(UARTx , int i)`	*void*	... *i* is the integer to be sent	Sends an integer over the selected UART peripheral.
`UARTx_Tx_string(UARTx  n, char *s)`	*void*	... *s* is the string to be sent	Sends a string over the selected UART peripheral.
`UARTx_Rx_char(UARTx n)`	*char*	...	Receives a single character through the selected UART peripheral.
`UARTx_Rx_int(UARTx n)`	*int*	...	Receives an integer through the selected UART peripheral.
`UARTx_Rx_char_memwrite(UARTx n, char * s, int data_items)`	*void*	...	Writes a char/string to the specified memory location.
`UARTx_Interrupt_Event_En(UARTx n, USART_Interrupt_Event ui)`	*void*	*ui* = TXE, CTS, TC, RXNE, ORE, IDLE,	Enables the selected UART interrupt.

		PE, LBD, NF_ORE_FE	*For details, see UART interrupts.*

UART Interrupts

Interrupt Event	Event flag	Event control bit
Transmit Data Register Empty	TXE	TXEIE
CTS flag	CTS	CTSIE
Transmission Complete	TC	TCIE
Received Data Ready to be Read	RXNE	RXNEIE
Overrun Error Detected	ORE	
Idle Line Detected	IDLE	IDLEIE
Parity Error	PE	PEIE
Break Flag	LBD	LBDIE
Noise Flag, Overrun error and Framing Error in multi-buffer communication	NF or ORE or FE	EIE

Source: STMicroelectronics RM0090 Reference manual

Code Examples

- Code to send a single character over UART.

```
#include "stm32f4xx.h"
#include "stm32f4xx_uart.h"

int main(void) {

	/* Initilize UART_5 with word length = 8-bits, STOP bits = 1, HDSEL = 0
	   (full-duplex) and baud rate = 9600 */
	UARTx_Init(UART_5, 8, 1, 0, 9600);

	/* Enable UART_5 */
	UARTx_En(UART_5);

	/* Enable UART_5 transmitter */
	UARTx_Tx_En(UART_5);

	/* Enable UART_5 transmit data register empty interrupt */
	UARTx_Interrupt_Event_En(UART_5, TXE);

	/* Send a character over UART_5 */
	UARTx_Tx_char(UART_5, 'A');

	/* Disable UART_5 transmitter, after transmission complete (saves power) */
	UARTx_Tx_Dis(UART_5);
	return 0;
}

void USART5_IRQHandler() {
	/* This section will be executed when a USART interrupt occurs */
}
```

- Code to receive a single character over UART, and store in a variable.

```
#include "stm32f4xx.h"
#include "stm32f4xx_uart.h"

int main(void) {

       /* Initilize UART_5 with word length = 8-bits, STOP bits = 1, HDSEL = 0
          (full-duplex) and baud rate = 9600 */
       UARTx_Init(UART_5, 8, 1, 0, 9600);

       /* Enable UART_5 */
       UARTx_En(UART_5);

       /* Enable UART_5 receiver */
       UARTx_Rx_En(UART_5);

       /* Enable UART_5 idle line detection interrupt */
       UARTx_Interrupt_Event_En(UART_5, IDLE);

       /* Receive and store a character, from UART_5 */
       char ch = UARTx_Rx_char(UART_5);

       /* Disable UART_5 receiver, after transmission complete (saves power) */
       UARTx_Rx_Dis(UART_5);
       return 0;
}

void USART5_IRQHandler() {
       /* This section will be executed when a USART interrupt occurs */
}
```