Synchronous and Asynchronous serial ports

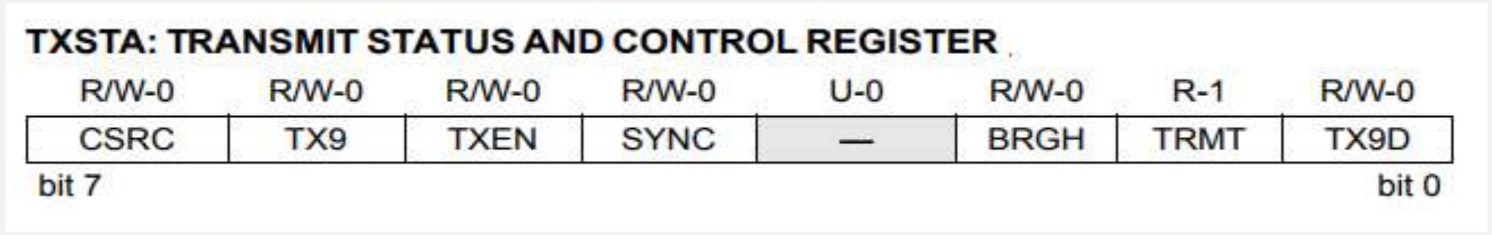
- **PIC16F877A Serial Communication Tutorial**
- PIC16F877A comes with inbuilt USART which can be used for Synchronous/Asynchronous communication. USART is a two-wire communication system in which the data flow serially. USART is also a full-duplex communication, which means you can send and receive data at the same time which can be used to communicate with peripheral devices, such as CRT terminals and personal computers.
- The **USART** can be configured in the following modes:
- Asynchronous (full-duplex)
- Synchronous – Master (half-duplex)
- Synchronous – Slave (half-duplex)
- But here We will be discussing only the UART (Asynchronous).

Registers used for Serial Communication

- •TXSTA (Transmit Status And Control Register)
- •RCSTA (Receive Status And Control Register)
- •SPBRG (USART Baud Rate Generator)
- •TXREG (USART Transmit Register)
- •RCREG (USART Receiver Register)

TXSTA (Transmit Status And Control Register)

This register is used to configure the Serial communication for TX.



TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

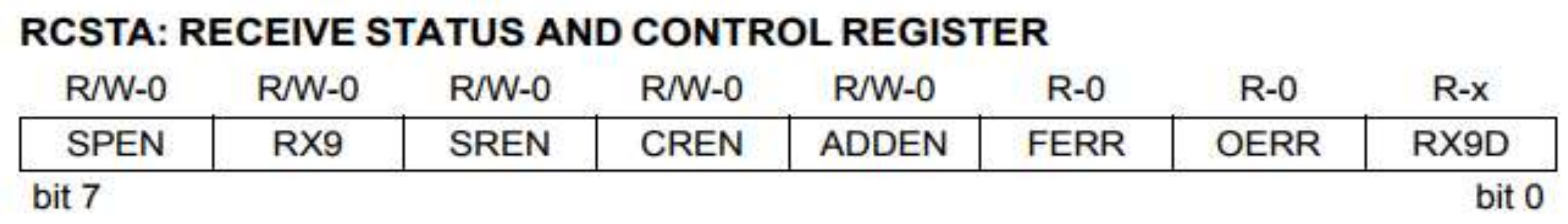
Synchronous and Asynchronous serial ports

CSRC: Clock Source Select bit (Asynchronous mode: Don’t care).
TX9: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission
TXEN: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled
SYNC: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode
BRGH: High Baud Rate Select bit
1 = High speed
0 = Low speed
TRMT: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

Synchronous and Asynchronous serial ports

RCSTA (Receive Status And Control Register)

This register is used to configure the Serial communication for RX.



RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Synchronous and Asynchronous serial ports

SPEN: Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled
RX9: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception
SREN: Single Receive Enable bit (Asynchronous mode: Don't care)
CREN: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
ADDEN: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR is set
0 = Disables address detection, all bytes are received and the ninth bit can be used as a parity bit
FERR: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

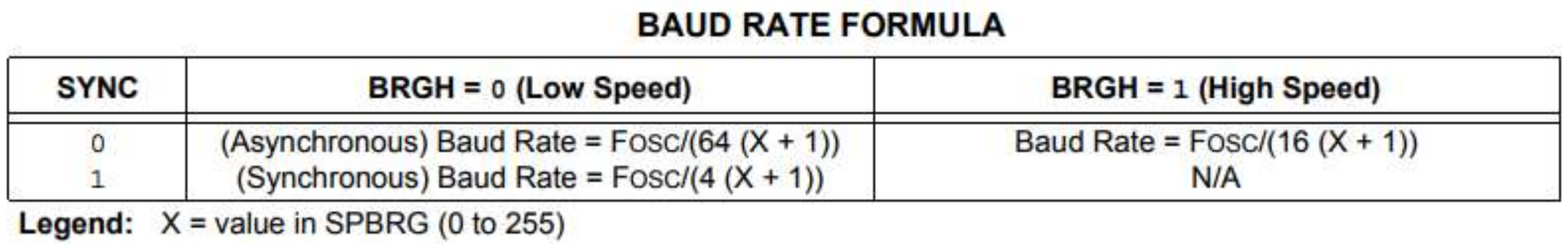
Synchronous and Asynchronous serial ports

SPBRG (USART Baud Rate Generator)

The main criteria for UART communication are its baud rate. Both the devices Rx/Tx should be set to the same baud rate for successful communication. This can be achieved by the SPBRG register. SPBRG is an 8-bit register that controls the baud rate generation. The SPBRG register controls the period of a free-running 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In Synchronous mode, bit BRGH is ignored.

Given the desired baud rate and FOSC, the nearest integer value for the SPBRG register can be calculated using the below formula.

Synchronous and Asynchronous serial ports



BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = FOSC/(64 (X + 1))	Baud Rate = FOSC/(16 (X + 1))
1	(Synchronous) Baud Rate = FOSC/(4 (X + 1))	N/A

Legend: X = value in SPBRG (0 to 255)

Calculation:

My Fosc = 11.0592MHz (You can put your board Fosc)
Baud Rate = 9600
9600 = 11059200 / (64X + 64)
64X+64 = 1152
X = 17.

If we want to generate 9600 Baudrate (Fosc = 11.0592MHz) you have to set 17 to SPBRG Register.

Synchronous and Asynchronous serial ports

TXREG (USART Transmit Register)

This is like a transmit buffer. But it is only an 8-bit register. So we have to place the character whatever we want to transmit via USART.

RCREG (USART Receiver Register)

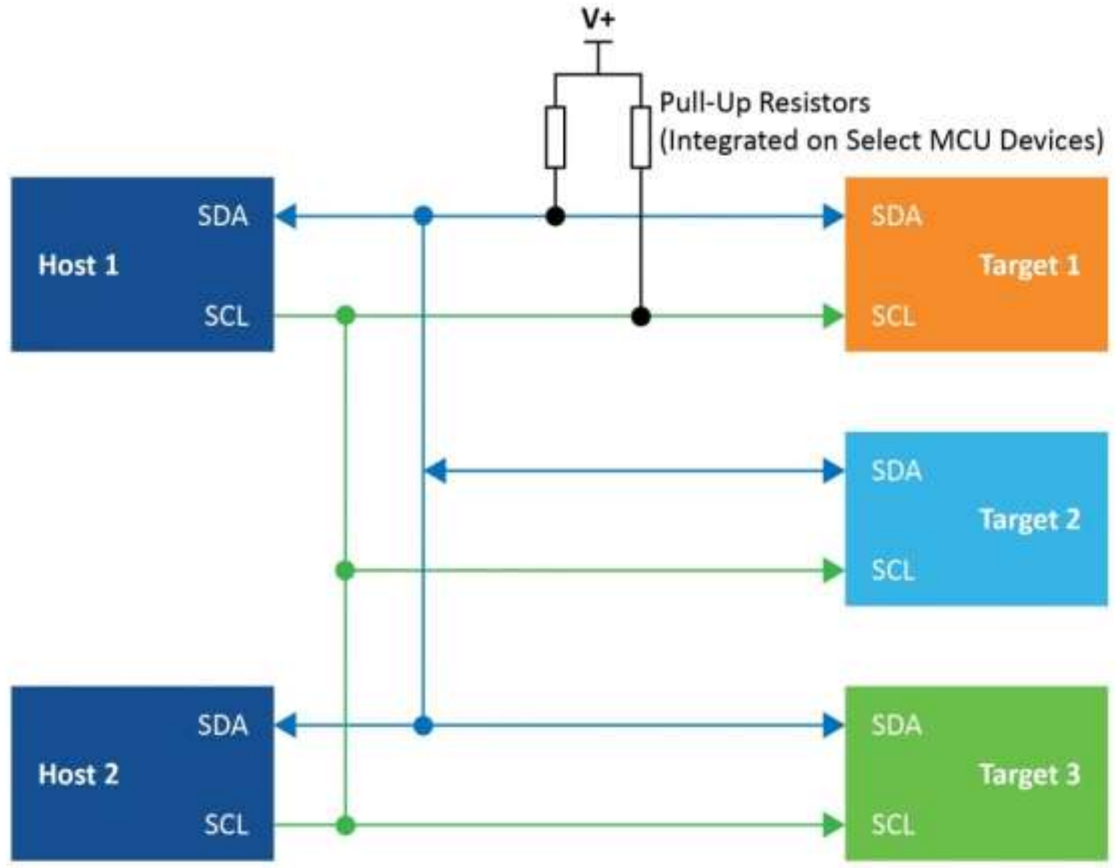
This is like a receiver buffer. But it is only an 8-bit register. So we can take the character whatever we microcontroller received via USART.

Inter-Integrated Circuit (I2C)

Inter-Integrated Circuit (I2C)? Inter-Integrated Circuit, also referred to as I^2C, is a two-wire, open-drain synchronous serial bus that supports multi-host and multi-client communication with acknowledgment. One of the biggest benefits of I^2C is its low Input/Output (I/O) usage. I^2C only requires two lines to communicate: Serial Clock (SCL) and Serial Data (SDA). While not required, I^2C is commonly used with an additional open-drain line called INT (Interrupt) to signal when an event occurs on the bus.

Inter-Integrated Circuit (I2C)

Basic I²C Network With Multiple Hosts


V+
Pull-Up Resistors
(Integrated on Select MCU Devices)
Host 1
SDA
SCL
Host 2
SDA
SCL
Target 1
SDA
SCL
Target 2
SDA
SCL
Target 3
SDA
SCL