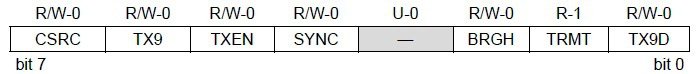
UART significa Universal Asynchronous Receiver/Transmitter. É uma interface de comunicação em série muito popular, que fornece comunicação *full* duplex entre dois dispositivos. UART utiliza duas linhas de dados para o envio (TX) e recepção de dados (RX). Terra/ Referência de ambos os dispositivos devem ser feitas comum. Como o nome indica, é uma interface de comunicação assíncrona, o que significa que ele não precisa de enviar CLOCK juntamente com os dados como em comunicação síncrona. UART é a interface de comunicação utilizado pela porta RS-232 do computador. Alguns dos Microchip PIC microcontrolador tem construído no módulo USART. USART significa Universal Synchronous Asynchronous Receiver Transmitter e pode ser configurado nos seguintes modos:

1. UART – Assincrono (Full Duplex)
2. USRT Master – Sincrono (Half Duplex)
3. USRT Slave – Sincrono (Half Duplex)

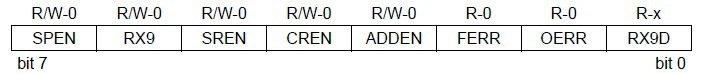
Modulo USART do PIC16F877A

TXSTA – Transmit Status and Control Register



* Bit 7 **CSRC**: Bit de seleção de clock, este bit não tem aplicação na operação em modo assíncrono de módulo USART. Ele é usado para selecionar o modo mestre ou escravo no modo de operação síncrona:
  + Modo assíncrono: Don’t care.
  + Modo Síncrono:
    - 1 = modo mestre (clock gerado internamente do BRG)
    - 0 = modo Slave (clock com fonte externa)
* Bit 6 **TX9**: Quando este bit é setado ele habilita a transmissão do bit 9 caso contrario a transmissão do bit 8 é usada. O 9º bit no modo de transmissão do bit 9 geralmente é o bit de paridade.
  + 1 = Seleciona a transmissão do bit 9
  + 0 = Seleciona a transmissão do bit 8
* Big 5 **TXEN**: Definir este bit permite a transmissão. No modo de operação síncrona os bits CREN e SREN do registrador RCSTA sobrepõem este bit.
  + 1 = Habilita Transmissão.
  + 0 = Desabilita a Transmissão.
* Bit 4 **SYNC**: Este é o bit de seleção do modo UART. Definir este bit seleciona o modo síncrono.
  + 1 = modo síncrono
  + 0 = modo assíncrono
* Bit 3 Não implementado: Este bit não é implementada e vai ler como 0.
* Bit 2 **BURGH**: Esta é a taxa de bits High Baud para operação em modo assíncrono e não é utilizada no modo síncrono. Definir este bit seleciona alta velocidade e limpando este bit seleciona taxas de transmissão de baixa velocidade.
  + Modo assíncrono:
    - 1 = Alta velocidade
    - 0 = Baixa velocidade
* Bit 1 **TRMT**: Este é o bit de status *Transmit Shift Register* (TSR). Isto pode ser utilizado para verificar se os dados escritos para transmissão são transmitidos ou não. Quando o TRS está vazio esse bit é setado como um e quando o TSR está cheio este bit será zero.
  + 1 = TSR vazio
  + 0 = TSR cheio
* Bit 0 **TX9D**: Este é o nono bit de dados no modo de transmissão de 9 bits, que é comumente utilizado como um bit de paridade.

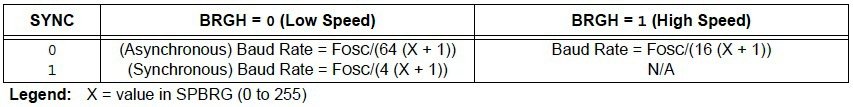
### RCSTA – Receive Status and Control Register



* Bit 7 **SPEN**: bit de habilitação da porta serial. Definir este bit permite habilita a porta serial e configura os pinos de porta serial RC7, RC6.
  + 1 = Habilita porta serial (configura pinos de porta serial RC7/RX/DT e RC6/TX/CK)
  + 0 = Desabilita porta serial.
* Bit 6 **RX9**: Definir este bit permite a recepção do bit 9 caso contrário, será no modo de recepção de 8 bits.
  + 1 = Seleciona o bit 9 para recepção
  + 0 = Seleciona o bit 8 para recepção
* Bit 5 **SREN**: bit que habilita recepção singular. Este bit não tem efeito sobre o modo assíncrono e modo síncrono escravo. A definição deste bit permite a recepção individual. Este bit será apagado após a recepção ser completada.
  + Modo assincrono: Don’t care.
  + Modo sincrono: – Master:
    - 1 = Habilita recebimento inidividual
    - 0 = Disabilita recebimento individual. Este bit é limpo depois da recepção ser completada.
    - Synchronous mode – Slave: Don’t care
* Bit 4 **CREN**: Habilita recebimento contínuo. Este bit permitirá recebimento contínuo. No modo síncrono CREN substitui SREN.
  + Modo assincrono:
    - * 1 = Habilita recebimento continuo.
      * 0 = Desabilita recebimento continuo.
    - Modo sincrono:
      * 1 = Habilita recebimento continuo
      * 0 = Desabilita recebimento continuo
* Bit 3 **ADDEN**: bit que habilita detecção de endereço. Este bit é aplicável somente no modo assíncrono com o 9º bit. Definir este bit permite detectar o endereço.
  + Modo assincrono 9-bit (RX9 = 1):
    - * 1 = Habilita detecção de endereço, habilita interrupção e carrega o buffer de recebimento quando RSR é um.
      * 0 = Desabilita detecção de endereço, todos os bytes são recebidos e o nono bit pode ser usado como paridade.
* Bit 2 **FERR**: bit de Erro de framing. 1 neste bit significa erro de Framing, enquanto 0 significa sem erro de Framing.
  + 1 = Erro de Framing (pode ser atualizado por um registrador de leitura RCREGe recebe o próximo byte valido.)
  + 0 = Sem erro de framing.
* Bit 1 **OERR**: bit de erro de despejo. 1 neste bit indica que o erro de despejo ocorreu.
  + 1 = Erro de despejo. (pode ser limpo limpando o bit CREN)
  + 0 = Sem Erro de despejo.
* Bit 0 **RX9D**: Este é o nono bit de dados recebidos e é comumente usado como bit de paridade.

## **USART Baud Rate Generator (BRG) Gerador da velocidade de transferência**

O Baud Generator fornece um clock necessário para a transmissão e recepção de dados. O módulo USART tem um gerador de taxa de transmissão de 8 bits dedicado que suporta os modos assíncronos e sincrono. O registrador de 8 bits SPBRG controla o período de tempo deste temporizador. No modo assíncrono o BRGH, o segundo bit do registrador TXSTA também controla a taxa de transmissão gerada, mas no modo síncrono é ignorado. A Taxa de transmissão pode ser calculada a partir das seguintes equações, onde FOSC é a frequência de clock do microcontrolador.



Inicializando a UART

char UART\_Init(const long int baudrate)

{

unsigned int x;

x = (\_XTAL\_FREQ - baudrate\*64)/(baudrate\*64); //SPBRG for Low Baud Rate

if(x>255) //If High Baud Rage Required

{

x = (\_XTAL\_FREQ - baudrate\*16)/(baudrate\*16); //SPBRG for High Baud Rate

BRGH = 1; //Setting High Baud Rate

}

if(x<256)

{

SPBRG = x; //Writing SPBRG Register

SYNC = 0; //Setting Asynchronous Mode, ie UART

SPEN = 1; //Enables Serial Port

TRISC7 = 1; //As Prescribed in Datasheet

TRISC6 = 1; //As Prescribed in Datasheet

CREN = 1; //Enables Continuous Reception

TXEN = 1; //Enables Transmission

return 1; //Returns 1 to indicate Successful Completion

}

return 0; //Returns 0 to indicate UART initialization failed

}

**Enviando dado através da UART**

Escrevendo um caracter

void UART\_Write(char data)

{

while(!TRMT);

TXREG = data;

}

Checando registrador de transmissão

char UART\_TX\_Empty()

{

return TRMT;

}

Escrevendo um texto

void UART\_Write\_Text(char \*text)

{

int i;

for(i=0;text[i]!='\0';i++)

UART\_Write(text[i]);

}

**Recebendo dado através da UART**

Dado recebido ou não

A função a seguir pode ser usada para verificar se os dados estão prontos para serem lidos do registrador de recebimento. Ele usa o bit RCIF que será um quando a recepção de dados está concluída.

char UART\_Data\_Ready()

{

return RCIF;

}

Recebendo um caracter

A função a seguir espera até que a recepção esteja completa e lê o dado de 8 bits do registrador de recebimento.

char UART\_Read()

{

while(!RCIF);

return RCREG;

}

Lendo texto

void UART\_Read\_Text(char \*Output, unsigned int length)

{

unsigned int i;

for(int i=0;i<length;i++)

Output[i] = UART\_Read();

}

Fonte : https://electrosome.com/uart-pic-microcontroller-mplab-xc8/