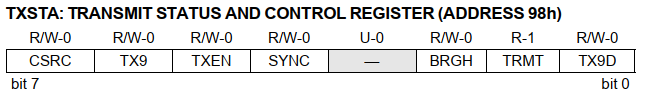
**Giao thức UART với PIC16F877A**

**A, Các lý thuyết cần nắm về UART trong PIC16F877A**

**I, Các thanh ghi quan trọng**

**1, TXSTA: TRANSMIT STATUS AND CONTROL REGISTER**

Đây là thanh ghi điều khiển quá trình truyền trong giao thức UART của PIC16F877A. Thanh ghi bao gồm 8bits và có 6bits đọc/ ghi được.



Bit 6: **TX9** – bit kích hoạt chế độ truyền 9bits

1 = chế độ truyền 9bits

0 = chế độ truyền 8bits

Bit 5: **TXEN** – bit kích hoạt quá trình truyền dữ liệu

1 = kích hoạt quá trình truyền

0 = vô hiệu hóa quá trình truyền

Bit 4: **SYNC** – bit lựa chọn chế độ USART hoặc UART

1 = chọn chế độ USART

0 = chọn chế độ UART

Bit 2: **BRGH** – bit lựa chọn Baud Rate ở tốc độ cao (không khả dụng trọng chế độ USART)

1 = High speed

0 = Low speed

Bit 1: **TRMT** – bit trạng thái của **Transmit Shift Register** (**TSR**, chúng ta sẽ tìm hiểu thanh ghi này ở phần sau)

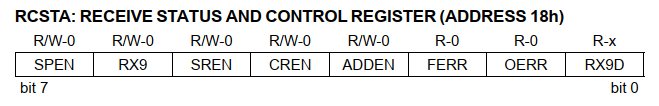
1 = TSR còn trống

0 = TSR đã đầy

Bit 0: **TX9D** – bit dành cho bit thứ 9 được truyền đi nếu chọn chế độ truyền 9bits

**2, RCSTA: RECEIVE STATUS AND CONTROL REGISTER**

Đây là thanh ghi điều khiển và hiển thị trạng thái của quá trình nhận dữ liệu trong giao thức UART của PIC16F877A. Thanh ghi 8bits và có 5bits có thể đọc/ ghi được.



Bit 7: **SPEN** – bit kích hoạt cổng nối tiếp

1 = kích hoạt cổng nối tiếp (chân RC7/RX và RC6/TX như 1 cổng nối tiếp)

0 = đóng cổng nối tiếp

Bit 6: **RX9** – kích hoạt chế độ nhận 9bits

1 = chọn chế độ nhận dữ liệu 9bits

0 = chọn chế độ nhận dữ liệu 8bits

Bit 4: **CREN** – bit cho phép nhận dữ liệu liên tục

1 = cho phép nhận dữ liệu liên tục

0 = không cho phép nhận dữ liệu liên tục

Bit 2: **FERR** – bit báo lỗi khung dữ liệu

1 = có lỗi khung dữ liệu

0 = không có lỗi

Bit 1: **OERR** – bit báo tràn dữ liệu

1 = có lỗi tràn dữ liệu (reset bit CREN sẽ gỡ lỗi này)

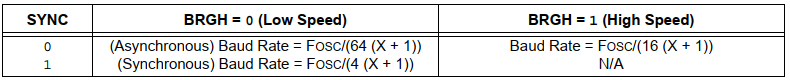
0 = không bị tràn

Bit 0: **RX9D** – bit dữ liệu thứ 9 nhận về trong chế độ nhận 9bits

**3, Thanh ghi SPBRG và khởi tạo Baud Rate**

* Thanh ghi **SPBRG** là thanh ghi khởi tạo Baud Rate cho PIC16F877A
* Bit **TXSTA<2> (BRGH)** là bit lựa chọn chế độ High Speed hoặc Low Speed cho Baud rate

Đối với giao thức UART trong PIC16F877A, chúng ta có công thức sau:



(**X** là giá trị của thanh ghi **SPBRG**, từ 0 đến 255)

Dựa vào công thức trên và bit **BRGH**, ta có thể dễ dàng khởi tạo tốc độ Baud mà mình mong muốn.

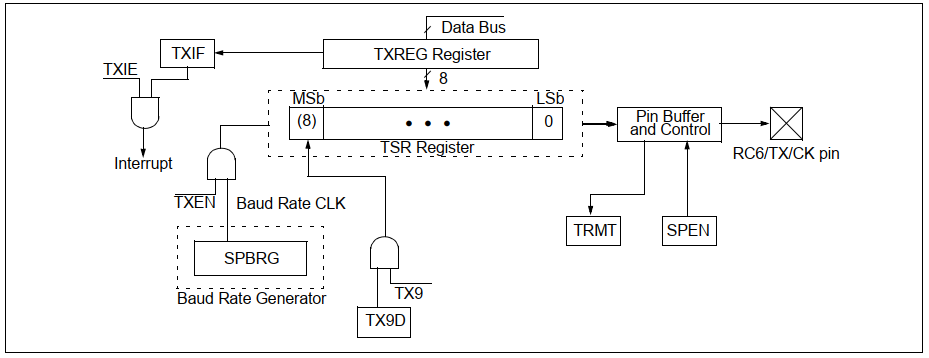
Trong bài này, chúng ta sẽ chọn chế độ High Speed. Vậy giá trị cần điền cho thanh ghi **SPBRG** là:

**SPBRG = ((Fosc/16) / Baud Rate) – 1**

**II, Tuyền và nhận dữ liệu với UART trong PIC16F877A**

**1, Truyền dữ liệu**

Sơ đồ khối của bộ truyền dữ liệu:

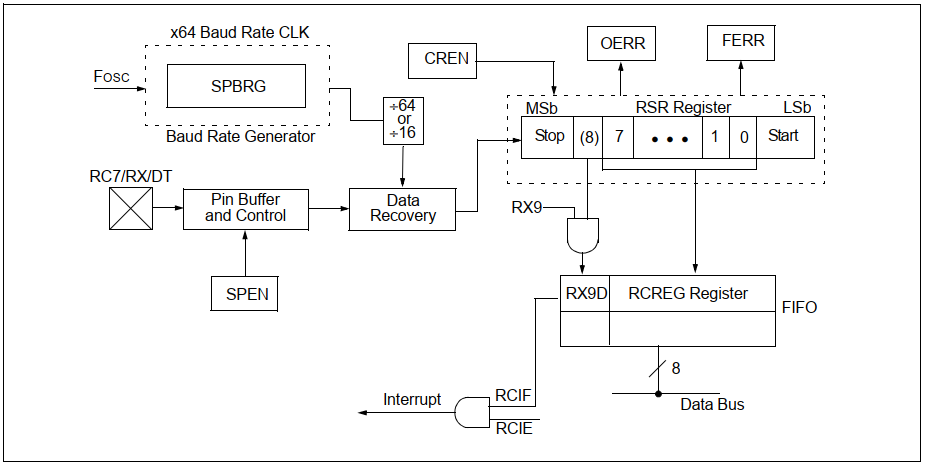


* Thanh ghi **TXREG** là 1 bộ đệm đọc/ ghi dữ liệu. Thanh ghi này sẽ truyền dữ liệu đến thanh ghi dịch (**TSR**).
* Thanh ghi **TSR** sẽ không tải xong dữ liệu, cho đến khi **Stop bit** của quá trình truyền được tải đến nó. Thanh ghi **TSR** không thể bị can thiệp bởi lập trình viên.
* Khi thanh ghi **TXREG** chuyển dữ liệu sang thanh ghi TSR (trong 1 chu kỳ), thanh ghi **TXREG** trống => flag bit **TXIF (PIR<4>)** được set lên mức 1.
* Flag bit **TXIF** sẽ được set bất kể trang thái nào của TXIE và không thể bị clear bằng code được. Flag bit **TXIF** chỉ bị reset khi data mới được load vào thanh ghi **TXREG**.
* Bit **TRMT (TXSTA<1>)** cho biết trạng thái thanh ghi **TSR**. Bit này là **read-only** bit, sẽ được set (1) khi thanh ghi **TSR** trống.
* Trong chế độ truyền 9bit, bit truyền **TX9 (TXSTA<6>)** phải được được set và bit thứ 9 phải được viết lên **TX9D** bit **(TXSTA<0>).** Bit thứ 9 phải được truyền trước khi truyền 8bit data vào **TXREG**. Bởi vì 1 data viết lên **TXREG** có thể dẫn đến bị truyền ngay sang **TSR** (nếu **TSR** trống).
* Như vậy, để truyền dữ liệu với giao thức UART trong PIC16F877A, chúng ta cần thực hiện các công việc sau:

1. Khởi tạo thanh ghi **SPBRG** để thiết lập Baud rate
2. Enable UART: clear bit **SYNC** và set bit **SPEN**
3. Nếu sử dụng ngắt, enable bit **TXIE**
4. Nếu sử dụng chế độ truyền 9bit, set (1) sbit TX9
5. Enable quá trình truyền bằng cách set bit **TXEN**, điều này cũng sẽ set bit **TXIF**
6. Nếu truyền 9bit thì bit thứ 9 phải được load vào bit **TX9D**
7. Load data vào thanh ghi **TXREG**
8. Nếu dùng ngắt thì phải enable bit **GIE** và **PEIE**

**2, Nhận dữ liệu**

Sơ đồ khối của bộ nhận dữ liệu:



* Trung tâm của khối nhận dữ liệu là thanh ghi dịch **RSR**. Sau khi nhận bit stop, dữ liệu nhận được trong **RSR** được chuyển đến thanh ghi **RCREG** (nếu nó trống).
* Nếu quá trình chuyền hoàn tất, cờ bit **RCIF(PIR<5>)** được set (1). Ngắt thực tế có thể disable/enable bởi cấu hình cho bit **RCIE(PIE1<5>)**.
* Flag bit **RCIF** là bit read-only, chỉ bị clear bởi hardware. (Bị clear khi thanh ghi **RCREG** trống).
* Khi phát hiện bit Stop của byte thứ 3 được truyền đến, nếu thanh ghi **RCREG** vẫn đầy, bit báo lỗi (**OERR-RCSTA<1>**) được set lên 1), dữ liệu trong **RSR** sẽ mất.
* Thanh ghi **RCREG** có thể được đọc 2 lần để lấy 2 byte, lỗi trang **OERR** phải được clear bằng code. Điều này được thực hiện bằng cách reset lại bit **CREN**. Nếu bit **OERR** được set, việc chuyển từ thanh ghi **RSR** sang thanh ghi **RCREG** bị nghẽn và không nhận thêm data nữa.

**B, Viết thư viện UART cho PIC16F877A**

**I, Hàm khởi tạo UART**

void UART\_Init**(**unsigned int baudRate**)**

**{**

//((Fosc/16) / baud rate) - 1

int value\_register **=** **(**500000 **/** baudRate**)** **-** 1**;**

//config pins for UART

TRISC**.**B6 **=** 0**;** //TX pin set as output

TRISC**.**B7 **=** 1**;** //RX pin set as input

//initialize SPBRB register for baudrate

BRGH\_bit **=** 1**;** //high baud rate

SPBRG **=** value\_register**;**

//enable UART mode

SYNC\_bit **=** 0**;** //Asynchronous mode

SPEN\_bit **=** 1**;** //serial port enable

//config for transmission and reception

TXEN\_bit **=** 1**;** //transmit enable

CREN\_bit **=** 1**;** //enable continous receive

TX9\_bit **=** 0**;** //8 bits transmission

RX9\_bit **=** 0**;** //8 bits reception

**}**

* Tham số truyền vào hàm này là tốc độ Baud cần khởi tạo.
* Trước hết chúng ta cần set cho 2 chân **PC6** và **PC7** lần lượt ở chế độ output (để truyền) và input (để nhận).
* Như đã trình bày ở phần khởi tạo Baud Rate, chúng ta sẽ cần set bit **BRGH** lên mức 1, và điền giá trị cho thanh ghi **SPBRG** theo tốc độ Baud ta mong muốn.
* Để kích hoạt giao thức UART, ta cần phải clear bit **SYNC** và set bit **SPEN**.
* Cấu hình các bit **TXEN**, **CREN** ở mức 1 để cho phép truyền dữ liệu và nhận dữ liệu liên tục.
* Clear bit **TX9** và **RX9** để chọn chế độ truyền, nhận 8bits.

**II, Hàm truyền dữ liệu**

**1, Truyền 1byte**

void UART\_SendByte**(**char myData**)**

**{**

**while(!**TXIF\_bit**);** //wait for transmission to finish

TXREG **=** myData**;** //load the data to TX buffer

**}**

Khi việc truyền dữ liệu được thực hiện, bất kỳ được giá trị nào được tải vào thanh ghi **TXREG** sẽ được truyền qua UART, nhưng việc truyền có thể bị lặp. Vì vậy chúng ta luôn phải kiểm tra cờ ngắt **TXIF**. Khi bit này thấp (0) thì chúng ta mới tiến hành truyền bit tiếp theo.

**2, Truyền 1 chuỗi**

void UART\_Write**(**char **\***str**)**

**{**

**while(\***str**)** //if character != NULL

**{**

UART\_SendByte**(\***str**++);**

**}**

**}**

Hàm này sẽ cho phép chúng ta truyền đi 1 chuỗi dữ liệu.

**III, Hàm nhận dữ liệu**

char UART\_ReceiveByte**(**void**)**

**{**

**if(**OERR\_bit**)** //if Error flag = 1

**{** //reset CREN bit

CREN\_bit **=** 0**;**

CREN\_bit **=** 1**;**

**}**

**while(!**RCIF\_bit**);** //wait complete receive

**return** RCREG**;**

**}**

Trong hàm này, trước hết chúng ta cần kiểm tra xem bit **OERR** có bị đẩy lên mức cao không, nếu có tức là đã có lỗi tràn dữ liệu. Để gỡ lỗi này, chúng ta phải reset bit **CREN**.

Trong quá trình dữ liệu, bộ nhận dữ liệu sẽ lấy dữ liệu và lưu trữ trong thanh ghi **RCREG**. Chún ta chỉ cần chuyển giá trị này sang bất kỳ biến nào để sử dụng, nhưng có thể bị lỗi trùng lặp, hoặc người dùng có thể gửi dữ liệu liên tục và chúng ta chưa chuyển chúng sang 1 biến nào đó kịp. Do đó, bit **RCIF** sẽ được sử dụng, bit này sẽ ở mức thấp bất cứ khi nào dữ liệu được nhận và chưa được xử lý.