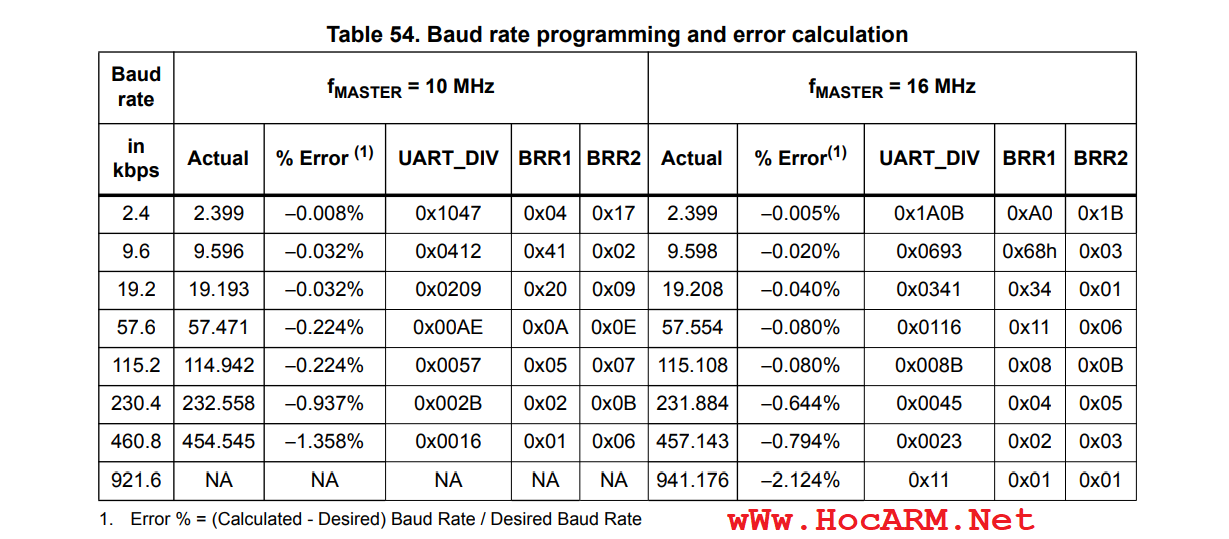
Truyền thông nối tiếp không đồng bộ. Thuật ngữ USART trong tiếng anh là viết tắt của cụm từ: Universal Synchronous & Asynchronous serial Reveiver and Transmitter, nghĩa là bộ truyền nhận nối tiếp đồng bộ và không đồng bộ. Cần chú ý rằng khái niệm USART (hay UART nếu chỉ nói đến bộ truyền nhận không đồng bộ) thường để chỉ thiết bị phần cứng (device,  
hardware), không phải chỉ một chuẩn giao tiếp. USART hay UART cần phải kết hợp với một thiết bị chuyển đổi mức điện áp để tạo ra một chuẩn giao tiếp nào đó. Ví dụ, chuẩn RS232 (hay COM) trên các máy tính cá nhân là sự kết hợp của chip UART và chip chuyển đổi mức điện áp.

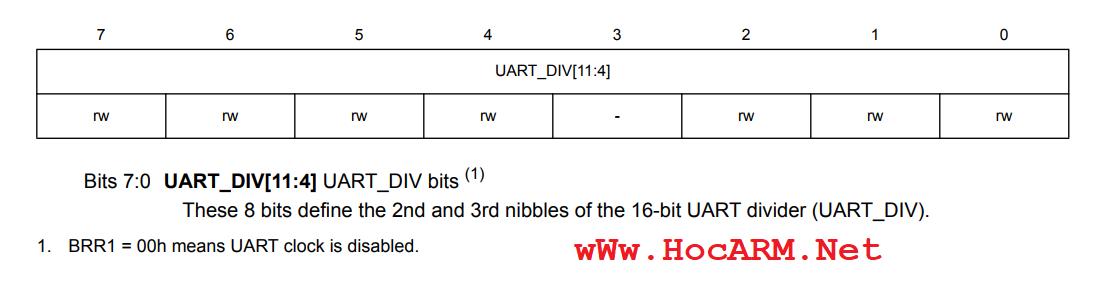
1. Các khái niệm cơ bản  
Truyền thông nối tiếp: giả sử bạn đang xây dựng một ứng dụng phức tạp cần sử dụng nhiều vi điều khiển (hoặc vi điều khiển và máy tính) kết nối với nhau.  
Trong quá trình làm việc các vi điều khiển cần trao đổi dữ liệu cho nhau, ví dụ tình huống Master truyền lệnh cho Slaver hoặc Slaver gởi tín hiệu thu thập được về Master xử lí…Giả sử dữ liệu cần trao đổi là các mã có chiều dài 8 bits, bạn có thể sẽ nghĩ đến cách kết nối đơn giản nhất là kết nối 1 PORT (8 bit) của mỗi vi điều khiển với nhau, mỗi line trên PORT sẽ chịu trách nhiệm truyền/nhận 1 bit dữ liệu. Đây gọi là cách giao tiếp song song, cách này là cách đơn giản nhất vì dữ liệu được xuất và nhận trực tiếp không thông qua bất kỳ một giải thuật biến đổi nào và vì thế tốc độ truyền cũng rất nhanh. Tuy nhiên, như bạn thấy, nhược điểm của cách truyền này là số đường truyền quá nhiều, bạn hãy tưởng tượng nếu dữ liệu của bạn có giá trị càng lớn thì số đường truyền cũng sẽ nhiều thêm. Hệ thống truyền thông song song thường rất cồng kềnh và vì thế kém hiệu quả. Truyền thông nối tiếp sẽ giải quyết vần đề này, trong tuyền thông nối tiếp dữ liệu được truyền từng bit trên 1 (hoặc một ít) đường truyền. Vì lý do này, cho dù dữ liệu của bạn có lớn đến đâu bạn cũng chỉ dùng rất ít đường truyền.   
Một hạn chế rất dễ nhận thấy khi truyền nối tiếp so với song song là tốc độ truyền và độ chính xác của dữ liệu khi truyền và nhận. Vì dữ liệu cần được “chia nhỏ” thành từng bit khi truyền/nhận, tốc độ truyền sẽ bị giảm. Mặt khác, để đảm bảo tính chính xác của dữ liệu, bộ truyền và bộ nhận cần có những “thỏa hiệp” hay những tiêu chuẩn nhất định. Phần tiếp theo trong chương này giới thiệu các tiêu chuẩn trong truyền thông nối tiếp không đồng bộ.  
Khái niệm “đồng bộ” để chỉ sự “báo trước” trong quá trình truyền. Lấy ví dụ thiết bị 1 (tb1) kết với với thiết bị 2 (tb2) bởi 2 đường, một đường dữ liệu và 1 đường xung nhịp. Cứ mỗi lần tb1 muốn send 1 bit dữ liệu, tb1 điều khiển đường xung nhịp chuyển từ mức thấp lên mức cao báo cho tb2 sẵn sàng nhận một bit. Bằng cách “báo trước” này tất cả các bit dữ liệu có thể truyền/nhận dễ dàng với ít “rủi ro” trong quá trình truyền. Tuy nhiên, cách truyền này đòi hỏi ít nhất 2 đường truyền cho 1 quá trình (send or receive). Giao tiếp giữa máy tính và các bàn phím (trừ bàn phím kết nối theo chuẩn USB) là một ví dụ của cách truyền thông nối tiếp  
đồng bộ.  
Khác với cách truyền đồng bộ, truyền thông “không đồng bộ” chỉ cần một đường truyền cho một quá trình. “Khung dữ liệu” đã được chuẩn hóa bởi các thiết bị nên không cần đường xung nhịp báo trước dữ liệu đến. Ví dụ 2 thiết bị đang giao tiếp với nhau theo phương pháp này, chúng đã được thỏa thuận với nhau rằng cứ 1ms thì sẽ có 1 bit dữ liệu truyền đến, như thế thiết bị nhận chỉ cần kiểm tra và đọc đường truyền mỗi mili-giây để đọc các bit dữ liệu và sau đó kết hợp chúng lại thành dữ liệu có ý nghĩa. Truyền thông nối tiếp không đồng bộ vì thế hiệu quả hơn truyền thông đồng bộ (không cần nhiều lines truyền). Tuy nhiên, để quá trình truyền thành công thì việc tuân thủ các tiêu chuẩn truyền là hết sức quan trọng.  
Chúng ta sẽ bắt đầu tìm hiểu các khái niệm quan trọng trong phương pháp truyền thông này.  
Baud rate (tốc độ Baud): như trong ví dụ trên về việc truyền 1 bit trong 1ms, bạn thấy rằng để việc truyền và nhận không đồng bộ xảy ra thành công thì các thiết bị tham gia phải “thống nhất” nhau về khoảng thời dành cho 1 bit truyền, hay nói cách khác tốc độ truyền phải được cài đặt như nhau trước, tốc độ này gọi là tốc độ Baud. Theo định nghĩa, tốc độ baud là số bit truyền trong 1 giây. Ví dụ nếu tốc độ baud được đặt là 19200 thì thời gian dành cho 1 bit truyền là 1/19200 ~ 52.083us.  
Frame (khung truyền): do truyền thông nối tiếp mà nhất là nối tiếp không đồng bộ rất dễ mất hoặc sai lệch dữ liệu, quá trình truyền thông theo kiểu này phải tuân theo một số quy cách nhất định. Bên cạnh tốc độ baud, khung truyền là một yếu tốc quan trọng tạo nên sự thành công khi truyền và nhận. Khung truyền bao gồm các quy định về số bit trong mỗi lần truyền, các bit “báo” như bit Start và bit Stop, các bit kiểm tra như Parity, ngoài ra số lượng các bit trong một data cũng được quy định bởi khung truyền.  
Start bit: start là bit đầu tiên được truyền trong một frame truyền, bit này có chức năng báo cho thiết bị nhận biết rằng có một gói dữ liệu sắp được truyền tới.   
Data: data hay dữ liệu cần truyền là thông tin chính mà chúng ta cần gởi và nhận. Data không nhất thiết phải là gói 8 bit, với STM8 bạn có thể quy định số lượng bit của data là 5, 6, 7, 8 hoặc 9 (tương tự cho hầu hết các thiết bị hỗ trợ UART khác). Trong truyền thông nối tiếp UART, bit có ảnh hưởng nhỏ nhất (LSB – Least Significant Bit, bit bên phải) của data sẽ được truyền trước và cuối cùng là bit có ảnh hưởng lớn nhất (MSB – Most Significant Bit, bit bên trái).  
Parity bit: parity là bit dùng kiểm tra dữ liệu truyền đúng không (một cách tương đối). Có 2 loại parity là parity chẵn (even parity) và parity lẻ (odd parity). Parity chẵn nghĩa là số lượng số 1 trong dữ liệu bao gồm bit parity luôn là số chẵn. Ngược lại tổng số lượng các số 1 trong parity lẻ luôn là số lẻ. Parity bit không phải là bit bắt buộc và vì thế chúng  
ta có thể loại bit này khỏi khung truyền (các ví dụ trong bài này tôi không dùng bit  
parity).  
Stop bits: stop bits là một hoặc các bit báo cho thiết bị nhận rằng một gói dữ liệu đã được gởi xong. Sau khi nhận được stop bits, thiết bị nhận sẽ tiến hành kiểm tra khung truyền để đảm bảo tính chính xác của dữ liệu. Stop bits là các bits bắt buộc xuất hiện trong khung truyền, trong STM8 USART có thể là 1 hoặc 2 bits  
(Trong các thiết bị khác Stop bits có thể là 2.5 bits).   
Gợi ý: khung truyền phổ biến nhất là : start bit+ 8 bit data+1 stop bit

< Admin tham khảo khái niệm tại trang hocavr.com >

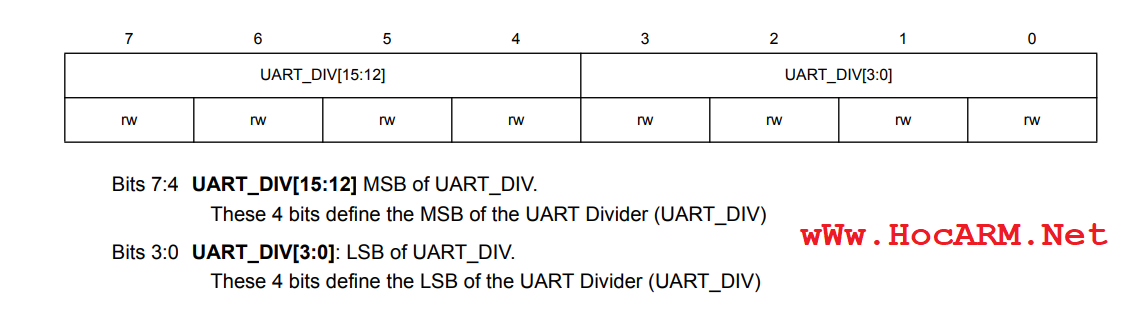
2. UART trong STM8  
Có 2 thanh ghi để chọn hệ số là BRR1 và BRR2 thiết lập BaudRate của UART  
Công thức tính như sau: Baud rate = Fmaster / UART\_DIV  
Dưới đây là 1 bảng tính sẵn các giá trị cho Baudrate khi Fmaster ở 10MHz và 16MHz. Chúng ta có thể dùng luôn các giá trị trong bảng được, nó bao gồm cả tỷ lệ sai



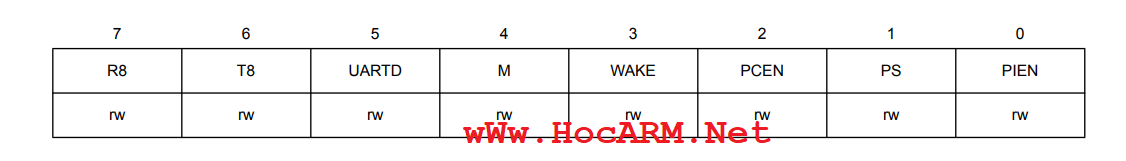
Không như Timer 1 muốn hệ số chia bao nhiêu chỉ việc lấy số đấy trừ đi 1 rồi tách 8 bit cao và 8 bit thấp để gán vào PSCRH và PSCRL  
Ở cái BRR này nó lại không phải là 8 bit cao và 8 bit thấp  
Thanh ghi UART\_BRR1 có ghi 11:4, tức là nó lại là 8 bit ở giữa của 16 bit, bắt đầu từ bit 4 đến bit 11



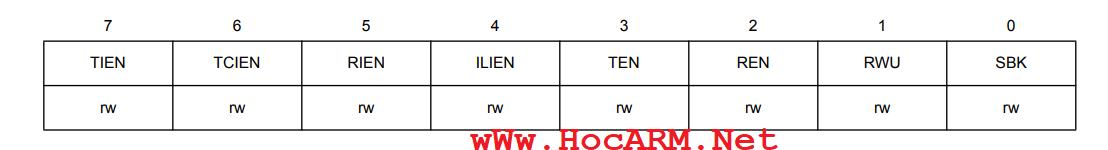
Thanh ghi UART\_BRR2 thì nó là 8 bit của 2 dầu 16 bit, tức là 4 bit thấp nhất và 4 bit cáo nhất ghép lại thành 16 bit.



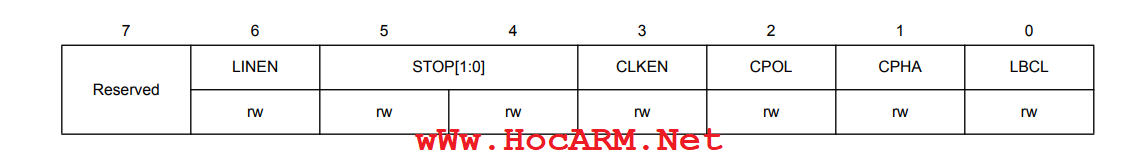
Ví dụ với  Fmaster là 16MHz, muốn Baudrate là 9600, thì hệ số chia phải là 1667  
Chỉ cần lấy 16MHz / 9600 = 1666.666666  
Vì mình chỉ lấy đc số nguyên lên làm tròn kết quả dẫn đến Baudrate thực ko chuẩn bằng 9600 nữa mà nó gần bằng thôi  
Baudrate thực = 16MHz / 1667. Ta chuyển sang Hex = 0x0683  
Lấy 8 bit giữa gán vào BRR1 đc 0x68  
Còn lại 4 bit đầu với bit cuối ghép lại với nhau đc 0x03 gán vào BRR2  
Ở thanh ghi UART\_CR1 cũng chỉ cần quan tâm tới 1 bit là bit cho phép UART Bit 0: PIEN



Thanh ghi UART\_CR2 để quản lý việc cho phép truyền, nhận và cho phép các ngắt



Trong UART không phải lúc nào mình cũng cần truyển nhận cả 2 chiều nên nó thêm cái này. giúp mình tiết kiệm đc IO vào việc khác  
Ở thanh ghi UART\_CR3 nó có chọn STOP bit và 1 bài bit cho chế độ đồng bộ, cái STOP bit thì không quan trọng, chỉ cần bên gửi và bên nhận giống nhau là được.



Thanh ghi này có thể gán giá trị 0 vào cũng đc (1 bit stop)  
  
Đây là đoạn init UART mẫu

void init\_uart(){

  UART1\_BRR1 = 0x68; *//*

  UART1\_BRR2 = 0x03; *// Baudrate ~9600*

  UART1\_CR2 = 0x2C;  *// Tx Enable, Rx Enable, Rx Interrupt Enable*

  UART1\_CR3 = 0x00;  *// 1 Stop bits*

}

Đây là gửi UART

void write\_byte\_uart(unsigned char byte){

  while(!(UART1\_SR & 0x80));   *// wait TXE*

  UART1\_DR = byte;

}

Hàm gửi một chuỗi

void print\_uart(unsigned char \*string){

  while(\*string){

    write\_byte\_uart(\*string);

    string++;

  }

}

Đây là trình phục vụ ngắt nhận data.   
Thanh ghi UART\_DR là thanh ghi Data của UART

#pragma vector = UART1\_R\_RXNE\_vector

\_\_interrupt void UART\_RXEN\_INTERRUPT(){

*// lay gia tri từ UART\_DR*

}