Truyền thông nối tiếp với 8051

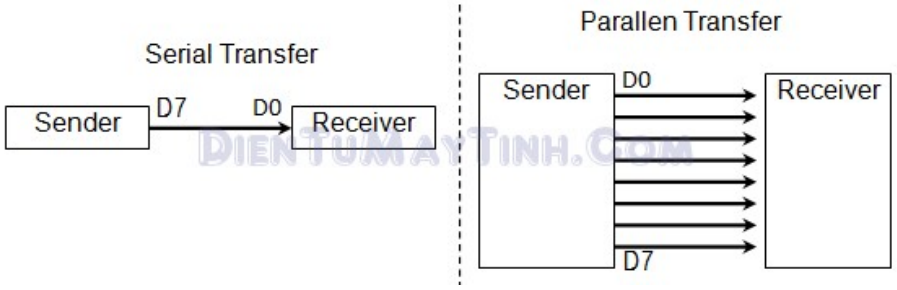
Truyền dữ liệu nối tiếp đồng bộ, không đồng bộ
Ø Đóng khung dữ liệu trong truyền thông không đồng bộ
Ø Chuẩn giao diện RS232
Ø Nối ghép 8051 với chuẩn RS232
Ø Các bước lập trình truyền thông nối tiếp cho 8051
- Cài đặt khung truyền
- Cài đặt tốc độ baud

Giới thiệu

Các máy tính truyền dữ liệu theo hai cách: Song song và nối tiếp. Trong truyền dữ liệu song song thường cần rất nhiều đường dây dẫn chỉ để truyền dữ liệu đến một thiết bị chỉ cách xa vài bước. Ví dụ của truyền dữ liệu song song là các máy in hoặc các ổ cứng, mỗi thiết bị sử dụng một đường cáp với nhiều dây dẫn. Mặc dù trong các trường hợp như vậy thì nhiều dữ liệu được truyền đi trong một khoảng thời gian ngắn bằng cách dùng nhiều dây dẫn song song, nhưng khoảng cách thì không thể lớn được. Vì các đường cáp dài làm suy giảm thậm chí làm méo tín hiệu. Ngoài ra, các đường cáp dài có giá thành cao. Vì những lý do này, để truyền dữ liệu đi xa thì phải sử dụng phương pháp truyền nối tiếp.

1. Các cơ sở của truyền thông nối tiếp

Trong truyền thông nối tiếp dữ liệu được gửi đi từng bit một, so với truyền song song thì là một hoặc nhiều byte được truyền đi cùng một lúc. **Hình 1** so sánh giữa việc truyền dữ liệu nối tiếp và song song.



Hình 1: Sơ đồ truyền dữ liệu nối tiếp so với sơ đồ truyền song song.

Trong truyền thông nối tiếp, một đường dữ liệu duy nhất được dùng thay cho nhiều đường dữ liệu của truyền thông song song không chỉ giúp giảm giá thành, giúp hệ thống đơn giản hơn nhiều mà nó còn mở ra khả năng để hai máy tính ở cách xa nhau có truyền thông qua đường thoại.

Truyền thông dữ liệu nối tiếp sử dụng hai phương pháp là **đồng bộ** và **không đồng bộ** (dị bộ):

Ø Trong **truyền đồng bộ**: thì bộ truyền và bộ thu được đồng bộ hóa qua một đường tín hiệu đồng hồ bên ngoài. Khái niệm "**đồng bộ**" để chỉ sự "báo trước" trong quá trình truyền. Lấy ví dụ: thiết bị 1 (tb1) kết nối với với thiết bị 2 (tb2) bởi 2 đường, một đường dữ liệu và 1 đường xung nhịp. Cứ mỗi lần tb1 muốn truyền 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 (dữ liệu và clock) cho 1 quá trình truyền hoặc nhận.

Sơ đồ trang web

Ø 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 đường truyền). Tuy nhiên, để quá trình truyền thành công thì việc tuân thủ các tiêu chuẩn truyền là hết sức quan trọng.

Trong 8051 có một bộ truyền dữ liệu không đồng bộ (UART - Universal Asynchronous serial Reveiver and Transmitter). Trước tiên chúng ta sẽ tìm hiểu các khái niệm quan trọng trong phương pháp truyền thông nối tiếp **không đồng bộ**:

1.1 Baud rate (tốc độ Baud)

Để việc truyền và nhận không đồng bộ xảy ra thành công thì các thiết bị tham gia phải “thống nhất” với nhau về khoảng thời gian 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.

1.2 Frame (khung truyền)

Dữ liệu đi vào ở đầu thu của đường dữ liệu trong truyền dữ liệu nối tiếp là một dãy các số 0 và 1, và rất khó để hiểu được ý nghĩa của các dữ liệu ấy nếu bên phát và bên thu không cùng thống nhất về một tập các luật, một thủ tục, về cách dữ liệu được đóng gói, bao nhiêu bit tạo nên một ký tự và khi nào dữ liệu bắt đầu và kết thúc. Bên cạnh **tốc độ baud**, **khung truyền** là một yếu tố 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.

Hình 2 là một ví dụ của một khung truyền của UART (truyền thông nối tiếp không đồng bộ): **khung truyền** này được bắt đầu bằng 01 **start bit**, tiếp theo là 08 **bit data**, sau đó là 01 **bit parity** dùng kiểm tra dữ liệu và cuối cùng là 02 **bits stop**. Công việc này được gọi là **đóng gói dữ liệu**. Chúng ta sẽ đi vào tìm hiểu các thành phần có trong một khung truyền:

Ø **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. **Start là bit bắt buộc**phải có trong khung truyền, và nó là một bit thấp (**0**).

Ø **Data (**dữ liệu**)**

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 8051 ta có thể quy định số lượng bit của data là 08 hoặc 09 bit. Trong truyền thông nối tiếp UART, bit có trọng số nhỏ nhất (**LSB** - Least Significant Bit, bit bên phải) của data sẽ được truyền trước và cuối cùng là bit có trọng số lớn nhất (**MSB** - Most Significant Bit, bit bên trái).

Ø **Parity bit**

Parity là bit dùng để kiểm tra dữ liệu truyền có 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.

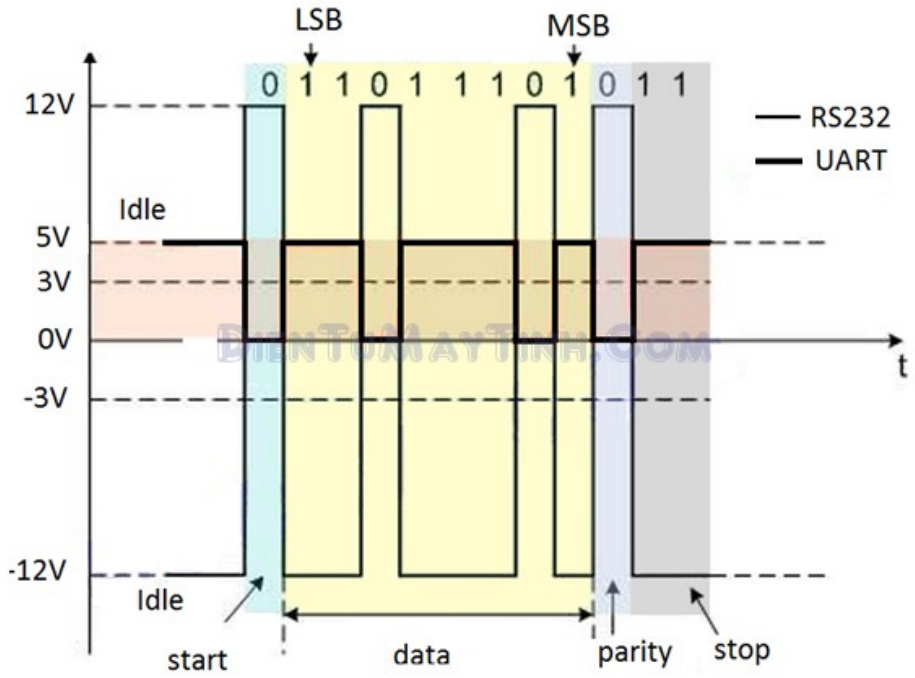
Ví dụ: nếu dữ liệu của bạn là 10111011 nhị phân, có tất cả 6 số "1" trong dữ liệu này, nếu quy định parity chẵn được dùng, bit parity sẽ mang giá trị 0 để đảm bảo tổng các số "1" là số chẵn (6 số 1). Nếu parity lẻ được yêu cầu thì giá trị của parity bit là 1. Sau khi truyền chuỗi dữ liệu kèm theo cả bit parity trên, bên nhận thu được và kiểm tra lại tổng số số "1" (bao gồm cả bit parity), nếu vi phạm quy định parity đã đặt trước thì ta khẳng định là dữ liệu nhận được là sai, còn nếu không vi phạm thì cũng không khẳng định được điều gì (mang tính tương đối). **Hình 2** mô tả một ví dụ với parity chẵn được sử dụng.

Parity bit không phải là bit bắt buộc và vì thế chúng ta có thể loại bit này khỏi khung truyền.

Ø **Stop bits**

Stop bits là 01 hoặc nhiều 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 bit bắt buộc** xuất hiện trong khung truyền, trong 8051 có thể là 01 hoặc 02 bit, và chúng là các bit cao (**1**).

Trong ví dụ ở **hình 2**: có 2 stop bits được dùng cho khung truyền.



Hình 2: Một khung truyền trong truyền thông nối tiếp không đồng bộ

2. Truyền thông nối tiếp trong 8051

2.1 Phần cứng

2.1.1 Các chân RxD và TxD trong 8051

Trong 8051 có hai chân được dùng cho truyền và nhận dữ liệu nối tiếp. Hai chân này được gọi là **TxD** và **RxD**, là một phần của cổng P3 (đó là P3.0-chân 10 và P3.1-chân 11). Các chân này hoạt động với **mức logic TTL** (mức logic cao "1" được gán cho **Vcc**và mức logic thấp được gán cho **0v**).

Vì các máy tính được sử dụng rất rộng rãi để truyền thông với các hệ thống vi điều khiển, do vậy ta chủ yếu tập trung vào truyền thông nối tiếp của 8051 với cổng COM – RS232 của PC.

2.1.2 Chuẩn giao diện RS232

Để cho phép tương thích giữa các thiết bị truyền thông dữ liệu được sản xuất bởi các hãng khác nhau thì một chuẩn giao diện được gọi là **RS232** đã được thiết lập bởi hiệp hội công nghiệp điện tử **EIA** vào năm 19960. Năm 1963 nó được sửa chỉnh và được gọi là RS232A và vào

các năm 1965 và 1969 thì được đổi thành RS232B và RS232C. ở đây chúng ta đơn giản chỉ hiểu là RS232. Ngày nay **RS232** là chuẩn giao diện I/O vào - ra nối tiếp được sử dụng rộng rãi nhất. Chuẩn này được sử dụng trong máy tính PC và hàng loạt các thiết bị khác nhau.

Ø **Các chân của cổng RS232**

Hình 3 là sơ đồ chân của cáp RS232 và chúng thường được gọi là đầu nối **DB - 25**. Trong lý hiệu thì đầu nối cắm vào (đầu đực) gọi là DB **- 25p** và đầu nối cái được gọi là **DB - 25s**.

Hình 3: Đầu nối **DB - 25** của RS232.

Vì không phải tất cả mọi chân của cổng RS232 đều được sử dụng trong cáp của máy tính PC, nên IBM đưa ra phiên bản của chuẩn vào/ra nối tiếp chỉ sử dụng có 9 chân gọi là **DB - 9** như trình bày ở **bảng 1** và **hình 4**.

Hình 4: Đầu nối **DB - 9** của RS232.

Số chân	Mô tả	
1	Data carrier detect (DCD)	Tránh tín hiệu mạng dữ liệu
2	Received data (RxD)	Dữ liệu được nhận
3	Transmitted data (TxD)	Dữ liệu được gửi
4	Data terminal ready (DTR)	Đầu dữ liệu sẵn sàng
5	Signal ground (GND)	Đất của tín hiệu
6	Data set ready (DSR)	Dữ liệu sẵn sàng
7	Request to send (RTS)	Yêu cầu gửi
8	Clear to send (CTS)	Xoá để gửi
9	Ring indicator (RI)	Báo chuông

Bảng 1: Các tín hiệu của các chân đầu nối DB - 9 trên máy tính.

2.1.4 Nối ghép 8051 tới RS232

Chuẩn RS232 được thiết lập trước **họ logic TTL** rất lâu do vậy điện áp đầu vào và đầu ra của nó **không tương thích với mức TTL**. Trong RS232 thì mức logic 1 được biểu diễn từ điện áp - 3v đến -25v trong khi đó mức 0 thì ứng với điện áp + 3v đến +25v làm cho điện áp - 3v đến + 3v là không xác định. Vì lý do này để kết nối một chuẩn RS232 bất kỳ đến một hệ vi điều khiển 8051 thì ta phải sử dụng các bộ biến đổi điện áp (như**MAX232**) để chuyển đổi các mức điện

áp **RS232** về các mức điện áp **TTL** sẽ được chấp nhận bởi các chân TxD và RxD của 8051 và ngược lại. Các **IC MAX232** nhìn chung được coi như các bộ điều khiển đường truyền.

Một điểm mạnh của **IC MAX232** là nó dùng điện áp nguồn +5v cùng với điện áp nguồn của 8051. Hay nói cách khác ta có thể nuôi 8051 và MAX232 với cùng một nguồn +5v, mà không phải dùng hai nguồn nuôi khác nhau.

IC MAX232 có hai bộ điều khiển đường truyền để nhận và truyền dữ liệu như trình bày trên **hình 5**. Các bộ điều khiển được dùng cho chân **TxD** được gọi là **T1** và **T2**, cho chân **RxD** gọi là **R1** và **R2**. Trong nhiều ứng dụng thì chỉ có 1 cặp được dùng. Ví dụ: ở hình dưới ta chỉ dùng đến **T2** và **R2** được dùng làm 1 cặp đối với **TxD** và **RxD** của 8051, còn cặp R1 và T1 thì không cần đến.

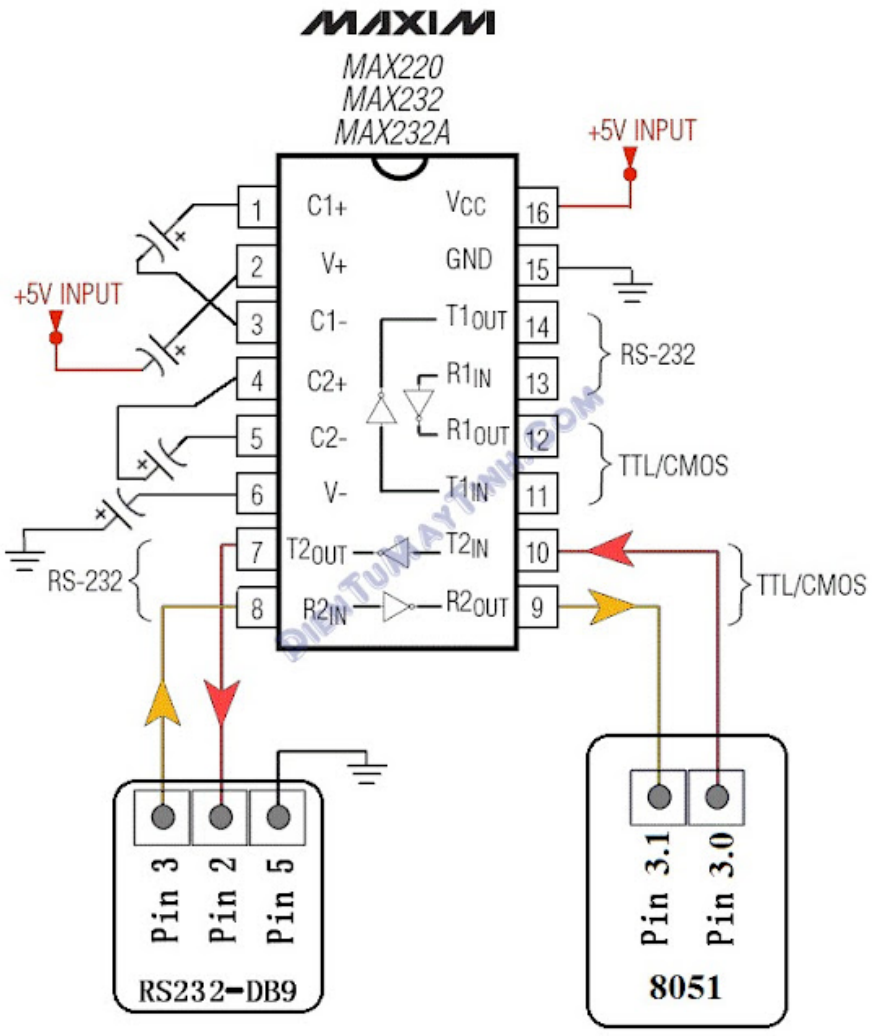
Để ý rằng trong **IC MAX232**, **T1** có gán $\mathbf{T1_{in}}$ (chân 11) và $\mathbf{T1_{out}}$ (chân 14):

- Chân $\mathbf{T1_{in}}$ là ở phía **TTL** và được nối tới chân **RxD** của bộ vi điều khiển.
- Chân $\mathbf{T1_{out}}$ là ở phía **RS232** được nối tới chân **RxD** của đầu nối **DB** của**RS232**.

Bộ điều khiển **R1** cũng có gán $\mathbf{R1_{in}}$ (chân 13) và $\mathbf{R1_{out}}$ (chân 12):

- Chân $\mathbf{R1_{in}}$ (chân số 13) là ở phía **RS232** được nối tới chân **TxD** ở đầu nối**DB** của **RS232**.
- Chân $\mathbf{R1_{out}}$ (chân số 12) là ở phía **TTL** được nối tới chân **RxD** của bộ vi điều khiển.

Tương tự cho **T2** và **R2.** Xem **hình 5**:



Hình 5: Sơ đồ bên trong của MAX232 và Sơ đồ nối ghép 8051 -Max232 - cổng COM DB-9.

Bộ MAX232 đòi hỏi 4 tụ hóa giá trị từ 1 đến 22mF. giá trị phổ biến nhất cho các tụ này là 22mF.

2.2 Lập trình phần mềm

Trong phần này chúng ta sẽ nghiên cứu về các thanh ghi truyền thông nối tiếp của 8051 và cách lập trình chúng để truyền và nhận dữ liệu nối tiếp.

2.2.1 Thanh ghi SBUF

SBUF là thanh ghi 8 bit được dùng riêng cho truyền thông nối tiếp trong 8051. Đối với một byte dữ liệu muốn truyền qua đường **TxD** thì nó phải được đặt trong thanh ghi **SBUF**. Tương tự, **SBUF** cũng giữ một byte dữ liệu khi nó được nhận từ đường **RxD**của 8051:

- Khi một byte được ghi vào thanh ghi **SBUF** nó sẽ được đóng khung với các bit **Start**, **Stop** và được truyền nối tiếp quan chân **TxD**.
- Khi các bit được nhận nối tiếp từ **RxD** thì 8051 mở khung đó để loại trừ các bit **Start**, **Sto**p để lấy ra một byte từ dữ liệu nhận được và đặt byte đó vào thanh ghi **SBUF**.

2.2.2 Thiết lập chế độ truyền bằng thanh ghi SCON

Điều đầu tiên chúng ta phải làm là gì khi sử dụng cổng nối tiếp tích hợp của 8051? Rõ ràng là cấu hình cho nó. Điều này cho phép chúng ta báo với 8051 biết: bao nhiêu bit dữ liệu chúng ta muốn truyền, tốc độ truyền. Vậy làm thế nào xác định các điều đó? Nhờ thanh ghi **SCON,** là thanh ghi 8 bit được dùng để lập trình việc đóng khung dữ liệu, xác định các chế độ làm việc của truyền thông nối tiếp. **SCON** là thanh ghi có thể đánh địa chỉ theo bit.

Dưới đây là mô tả các bit khác nhau của thanh ghi **SCON**:

Bit	Tên	Địa Chỉ	Chức Năng
7	SM0	9Fh	Xác định chế độ cổng nối tiếp (bit 0).
6	SM1	9Eh	Xác định chế độ cổng nối tiếp (bit 1).
5	SM2	9Dh	Cho phép truyền thông đa xử lý.
4	REN	9Ch	Bit cho phép nhận.
3	TB8	9Bh	Sử dụng trong chế độ 2 và 3.
2	RB8	9Ah	Sử dụng trong chế độ 2 và 3.
1	TI	99h	Cờ truyền: được bật sau khi truyền xong 1 byte.
0	RI	98h	Cờ nhận: được bật sau khi nhận đủ 1 byte.

Hình 6: Thanh ghi điều khiển cổng nối tiếp SCON.

Ø **Các bit SM0, SM1**

Đây là các bit **D7** và **D6** của thanh ghi **SCON**. Chúng được dùng để xác định các chế độ đóng khung dữ liệu, có 4 chế độ:

SM0	SM1	Chế Độ	Khung Dữ liệu	Tốc Độ Baud
0	0	0	8-bit Shift Register	Oscillator / 12
0	**1**	**1**	**8-bit UART**	**Cài đặt bởi Timer 1 (*)**
1	0	2	9-bit UART	Oscillator / 64 (*)
1	1	3	9-bit UART	Cài đặt bởi Timer 1 (*)

Hình 7: Các chế độ xác định bởi 2 bit SM0 và SM1

(*) **Lưu ý:** tốc độ truyền chỉ ra trong bảng này được tăng gấp đôi nếu bit **PCON.7** (bit **SMOD**) được thiết lập lên 1, mặc định của hệ thống là **PCON.7=0**.

Trong bốn chế độ trên ta chỉ quan tâm đến **chế độ 1**. Khi chế độ 1 được chọn thì dữ liệu được đóng khung thành **10 bit**: gồm **1 bit Start**, sau đó là **8 bit dữ liệu,** và cuối cùng là **1 bit Stop**. Quan trọng hơn là chế độ nối tiếp 1 cho phép **tốc độ baud thay đổi** và được thiết lập bởi **Timer1** của 8051.

Ø **Bit SM2**

Bit **SM2** là bit **D5** của thanh ghi **SCON**. Bit này cho phép khả năng đa xử lý của 8051. Đối với các ứng dụng của chúng ta, đặt **SM2 = 0** vì ta không sử dụng 8051 trong môi trường đa xử lý.

Ø **Bit REN**

REN (Receive Enable) là bit cho phép nhận (bit **D4** của thanh ghi **SCON**). Khi bit**REN** cao thì nó cho phép 8051 nhận dữ liệu trên chân **RxD** của nó. **Và kết quả là nếu ta muốn 8051 vừa truyền vừa nhận dữ liệu thì bit REN phải được đặt lên 1**. Bit này có thể được dùng để khống chế mọi việc nhận dữ liệu nối tiếp và nó là **bit cực kỳ quan trọng** trong thanh ghi **SCON**.

Ø **Bit TB8 và RB8**

Bit **TB8** và **RB8** được dùng trong chế độ nối tiếp **2** và **3**. Ta đặt **TB8=0** và **RB8=0**vì nó không được sử dụng trong các ứng dụng của mình.

Nói thêm, trong chế độ **2** và **3** thì có 9 bit dữ liệu được truyền đi hoặc nhận về. Bit**TB8** sẽ chứa bit dữ liệu thứ 9 khi truyền, còn bit **RB8** sẽ chứa bit dữ liệu thứ 9 khi nhận, trong chế độ nối tiếp **1** thì bit **RB8** này nhận một bản sao của bit **Stop** khi một dữ liệu 8 bit được nhận, và ta cũng không cần quan tâmJ.

Ø **Các bit TI và RI**

Các bit ngắt truyền **TI** và ngắt nhận **RI** là các bit **D1** và **D0** của thanh ghi **SCON**. Các bit này là **cực kỳ quan trọng** của thanh ghi **SCON**:

- Khi 8051 kết thúc truyền một ký tự 8 bit thì nó bật **TI** để báo rằng nó sẵn sàng truyền một byte khác. Bit **TI** được bật lên trước bit **Stop**.
- Khi 8051 nhận được dữ liệu nối tiếp qua chân **RxD** và nó tách các bit Start và Stop để lấy ra 8 bit dữ liệu để đặt vào SBUF, sau khi hoàn tất nó bật cờ **RI** để báo rằng nó đã nhận xong 1 byte và cần phải lấy đi kẻo dữ liệu bị mất. Cờ **RI**được bật khi đang tách bit Stop.

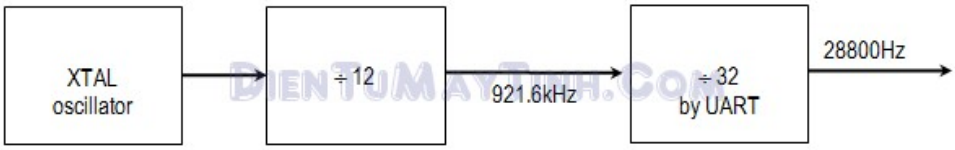
2.2.3 Thiết lập tốc độ baud trong 8051

Một khi các **chế độ cổng nối tiếp** đã được cấu hình, việc tiếp theo là chương trình cần phải cấu hình **tốc độ baud** cho các cổng nối tiếp. Điều này chỉ áp dụng cho chế độ Serial Port **1** và **3**. Còn ở chế độ **0** và **2**, tốc độ truyền được xác định dựa trên tần số dao động của thạch anh:

Ø **Trong chế độ 0:** tốc độ truyền luôn luôn là tần số dao động chia cho **12**. Điều này có nghĩa là nếu bạn đang sử dụng thạch anh tần số 11.059Mhz, tốc độ truyền của chế độ**0** sẽ luôn luôn là 921.583 baud. **Trong chế độ 2:** tốc độ truyền luôn luôn là tần số dao động chia cho **64**, do đó, với thạch anh tần số 11.059Mhz sẽ mang lại một tốc độ truyền 172.797 baud.

Ø **Trong chế độ 1 và 3**: tốc độ truyền được xác định bằng cách **cài đặt Timer1**. Phương pháp phổ biến nhất là cài đặt **Timer1** ở chế độ tự động nạp lại 8-bit (**chế độ 2**) và thiết lập một giá trị nạp lại (cho **TH1**) để tạo ra một tốc độ truyền.

Như ta đã biết ở trước đây, thì 8051 chia tần số thạch anh cho **12** để lấy tần số chu kỳ máy. Bộ UART truyền thông nối tiếp của 8051 lại chia tần số chu kỳ máy cho **32**một lần nữa trước khi nó được dùng bởi bộ định thời **Timer1** để tạo ra tốc độ **baud**:

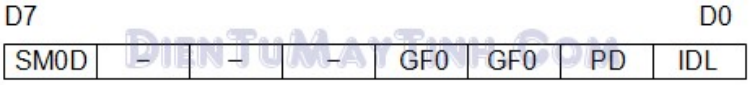


Hình 8: Tần số của bộ truyền thông nối tiếp UART

2.2.3.1 Nhân đôi tốc độ baud trong 8051

Có hai cách để tăng tốc độ baud truyền dữ liệu trong 8051:

1. Sử dụng tần số thạch anh cao hơn.
2. Thay đổi một bit trong thanh ghi điều khiển công suất **PCON** (Power Control) như chỉ ra dưới đây.



Hình 9: Thanh ghi PCON

Phương án 1 là không khả thi trong nhiều trường hợp vì tần số thạch anh của hệ thống là cố định. Do vậy, ta sẽ tập trung thăm dò phương án 2: **nhân đôi tốc độ baud bằng phần mềm** trong 8051 với tần số thạch anh không đổi. Điều này được thực hiện nhờ thanh ghi **PCON**, đây là thanh ghi 8 bit. Trong 8 bit này thì có một số bit không được dùng để điều khiển công suất của 8051. Bit dành cho truyền thông nối tiếp là bit **D7** (bit**SMOD)**. Khi 8051 được bật nguồn thì bit **SMOD** của thanh ghi **PCON** ở mức thấp (**0)**. Chúng ta có thể đặt nó lên 1 bằng phần mềm và do vậy nhân đôi được tốc độ **baud**. Tại sao có được điều đó? Ta hãy làm rõ tiếp:

Ø **Khi SMOD = 0**

Khi **SMOD = 0** thì 8051 chia **1/12** tần số thạch anh cho **32** và sử dụng nó cho bộ**Timer1** để thiết lập tốc độ **baud**. Đây là **giá trị mặc định** của **SMOD** khi 8051 bật nguồn.

Ø **Khi SMOD = 1**

Khi **SMOD = 1** thì 8051 chia **1/12** tần số thạch anh cho **16** (thay vì chia cho **32**như khi SMOD = 0) và đây là tần số được **Timer1** dùng để thiết lập tốc độ **baud**.

Để xác định giá trị cài đặt trong **TH1** để tạo ra một tốc độ **baud** nhất định, chúng ta có thể sử dụng các phương trình sau đây (giả sử bit **PCON.7=0**):

TH1 = 256 - ((Crystal / (12*32)) / Baud) = 256 - ((Crystal / 384) / Baud) **(1)**

Nếu **PCON.7=1** thì tốc độ truyền tăng gấp đôi, do đó phương trình trở thành:

TH1 = 256 - ((2*Crystal / (12*32)) / Baud) = 256 - ((Crystal / 192) / Baud) **(2)**

Ví dụ 1:

Nếu chúng ta có một tinh thể thạch anh tần số **11.059Mhz** và chúng ta muốn cấu hình cho cổng nối tiếp đạt tốc độ **19200 baud**, thì ta sử dụng **phương trình 1**:

TH1 = 256 - ((Crystal / 384) / Baud)
TH1 = 256 - ((11059000/384) / 19200)
TH1 = 256 - ((28799) / 19200)
TH1 = 256-1,5 = 254,5

Như bạn có thể thấy: để có được tốc độ **19200 baud** trên một tinh thể thạch anh 11.059Mhz ta phải cài đặt **TH1** một giá trị **254,5**. Nhưng giá trị trong các thanh ghi lại là 1 số nguyên. Nếu chúng ta thiết lập là **254**, chúng ta sẽ có tốc độ **14400 baud** và nếu chúng ta thiết lập là **255**,

chúng ta sẽ có tốc độ **28800 baud**. Như vậy dường như chúng ta không thể cài đặt chính xác tốc độ **baud** được ?!! L

Nhưng ta lại có một cách khác để cài đặt được tốc độ **19200 baud.**J Chúng ta đơn giản chỉ cần đặt bit **PCON.7=1** (bit **SMOD**). Khi đó ta đã tăng gấp đôi tốc độ **baud**và sử dụng **phương trình 2** được đề cập ở trên. Vì vậy chúng ta có:

TH1 = 256 - ((Crystal / 192) / Baud)
TH1 = 256 - ((11059000/192) / 19200)
TH1 = 256 - ((57.699) / 19.200)
TH1 = 256 - 3 = 253

Vậy: để có được tốc độ **19200 baud** với một tinh thể thạch anh tần số**11.059MHz** chúng ta phải:

1. Cấu hình **chế độ Serial Port 1** hoặc **3**.
2. Cấu hình **Timer 1** ở **chế độ 2** (8-bit tự động nạp lại).
3. Cài đặt **TH1** giá trị **253** (**FDH**).
4. Set bit **PCON.7=1** (**SMOD**) để tăng gấp đôi tốc độ truyền (19200 baud).

2.2.4 Lập trình 8051 để truyền dữ liệu nối tiếp

Để lập trình 8051 truyền các byte ký tự nối tiếp thì cần phải thực hiện các bước sau đây:

1. Nạp thanh ghi **TMOD** giá trị **20H**: báo rằng sử dụng Timer1 ở chế độ 2 để thiết lập chế độ baud.
2. Nạp thanh ghi **TH1** các giá trị phù hợp để thiết lập chế độ baud truyền dữ liệu nối tiếp.
3. Nạp thanh ghi **SCON** giá trị **50H** báo chế độ nối tiếp 1 để đóng khung 8 bit dữ liệu, 1 bit Start và 1 bit Stop.
4. Bật **TR1=1** để khởi động Timer1.
5. Xoá bit cờ truyền dữ liệu: **TI=0**.
6. Byte ký tự cần phải truyền được ghi vào **SBUF**.
7. Bit cờ truyền **TI** được kiểm tra bằng một vòng lặp để đợi đến lúc dữ liệu được truyền xong (cờ **TI=1**).
8. Để truyền ký tự tiếp theo quay trở về **bước 5**.

Các bạn hãy quan sát 2 ví dụ sau để thực hành:

Ví dụ 2:

Hãy viết chương trình cho 8051 để truyền dữ liệu nối tiếp một ký tự "**D**" với tốc độ **4800 baud** liên tục lên máy tính.

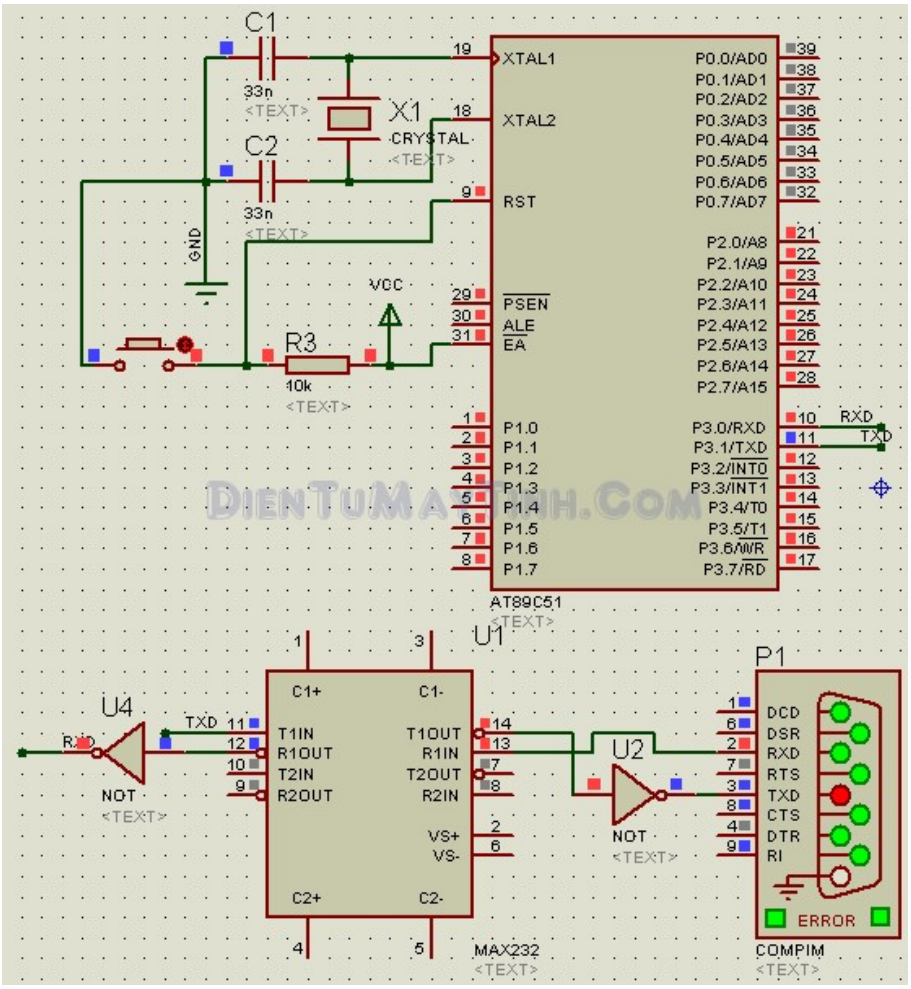
Lời giải:

Chương trình sử dụng **ngôn ngữ C** lập trình trên **Keil C uVision3**, mô phỏng trên**Proteus**, hiển thị lên máy tính qua giao diện **Hyper Terminal Hercules**.
(**Proteus** và **Hercules** sử dụng 2 **cổng COM ảo** được tạo ra và kết nối với nhau bởi chương trình **Configure Virtual Serial Port Driver**)

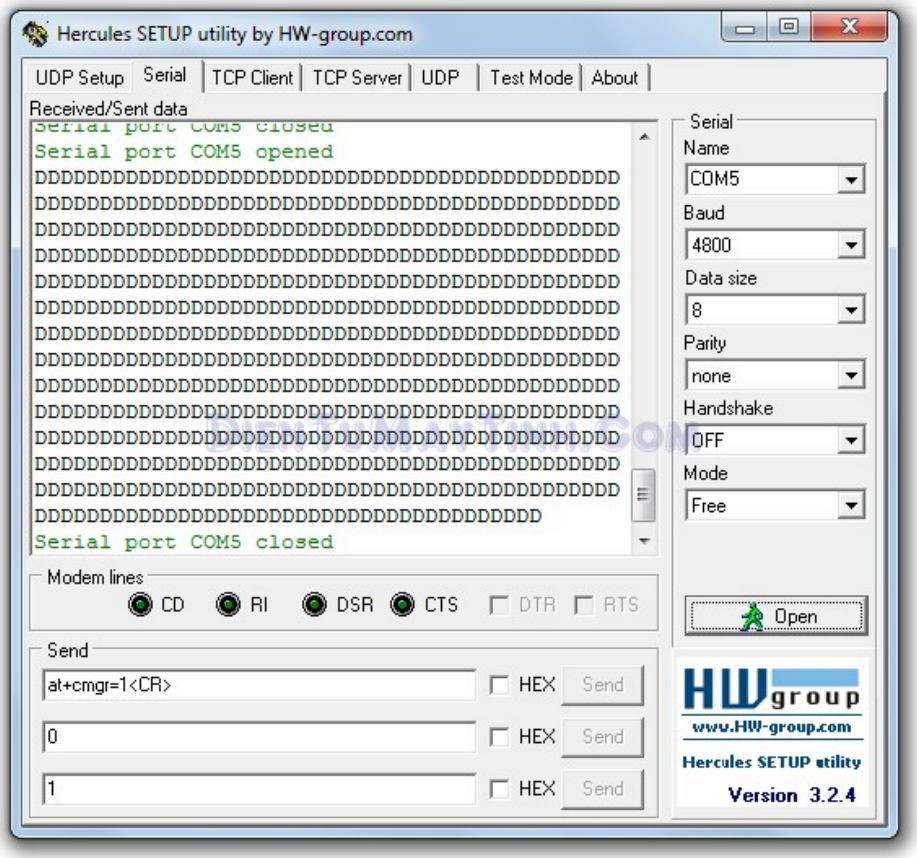
```
#include<at89x51.h>              //khai báo thư viện cho 89c51
void send(unsigned char a);      //khai báo nguyên mẫu hàm gửi 1 ký tự
main()                                   //Chương trình chính
{
        TMOD=0x20;               //Chọn Timer1, chế độ 2
        TH1=0xFA;                //Cài đặt tốc độ 4800 baud
        SCON=0x50;                       //0101 0000: Chọn chế độ 1, Cho phép nhận
        TR1=1;                   //Khởi động Timer1

        while(1)                 //Vòng lặp vô hạn
        {
                send('D');       //Gọi hàm gửi 1 ký tự lên máy tính
```

```
        }
}
void send(unsigned char a)              //Định nghĩa hàm gửi 1 ký tự
{
        SBUF=a;                         //Ghi 1 byte dữ liệu vào thanh ghi SBUF
        while(TI==0){}                  //vòng lặp để đợi cờ truyền TI lên 1
        TI=0;                           //Xóa cờ truyền TI sau khi truyền xong
}
```



Hình 10: Sơ đồ nguyên lý mạch mô phỏng **Ví Dụ 2** trên Proteus



Hình 11: Kết quả truyền lên máy tính **Ví Dụ 2** qua giao diện Hercules

Ví dụ 3:

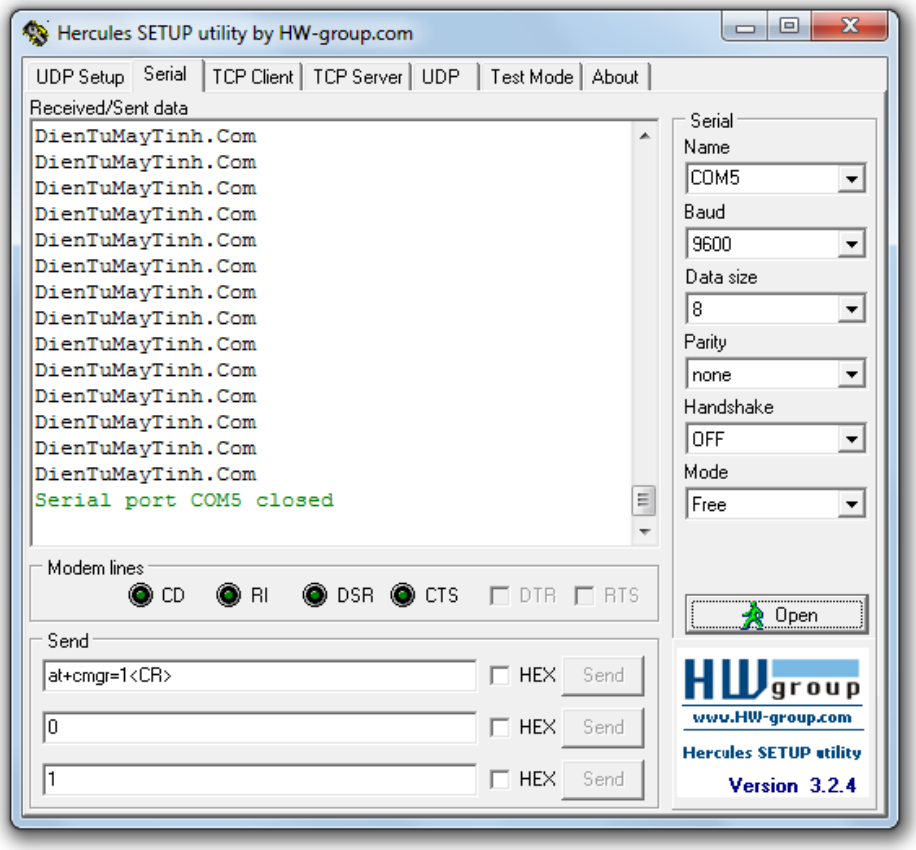
Hãy viết chương trình để 8051 truyền dòng chữ "**DienTuMayTinh.Com**" liên tục với tốc độ **9600 baud** (8 bit dữ liệu, 1 bit Stop) lên máy tính.

Lời giải:

```
#include<at89x51.h>                //Khai báo thư viện cho 89c51
#include<string.h>                 //Khai báo thư viện để sử dụng hàm strlen()
void send(unsigned char a);        //khai báo nguyên mẫu hàm gửi 1 ký tự
void sendchuoi(char *a);           //khai báo nguyên mẫu hàm gửi 1 chuỗi
main()                                    //Chương trình chính
{
        TMOD=0x20;                 //Chọn Timer1, chế độ 2
        TH1=0xFD;                  //Cài đặt tốc độ 9600 baud
        SCON=0x50;                        //0101 0000: Chọn chế độ 1, Cho phép nhận
        TR1=1;                     //Khởi động Timer1

        while(1)                   //Vòng lặp vô hạn
        {
                sendchuoi("DienTuMayTinh.Com");      //Gọi hàm gửi 1 chuỗi
                send(10);                                 //Gửi dấu xuống dòng
        }
}
void send(unsigned char a)                //Định nghĩa hàm gửi 1 ký tự
{
        SBUF=a;                    //Ghi 1 byte dữ liệu vào thanh ghi SBUF
        while(TI==0){}             //vòng lặp để đợi cờ truyền TI lên 1
        TI=0;                      //Xóa cờ truyền TI sau khi truyền xong
}
```

```
void sendchuoi(char *a)          //Định nghĩa hàm gửi 1 chuỗi ký tự
{
        int i,n;                  //Khai báo biến cục bộ số nguyên: i,n
        n=strlen(a);              //Tính độ dài của chuỗi *a, lưu vào biến n
        for(i=0;i<n;i++)          //Vòng lặp để gửi lần lượt từng ký tự lên,
        {                         //cho đến khi hết chuỗi *a (ký tự thứ n-1).
                send(a[i]);       //Gọi hàm gửi 1 ký tự.
        }
}
```



Hình 12: Kết quả truyền lên máy tính **Ví Dụ 3** qua giao diện Hercules

Ø Tầm quan trọng của cờ TI

Để hiểu tầm quan trọng của cờ ngắt TI ta hãy xét trình tự các bước dưới đây mà 8051 phải thực hiện khi truyền một ký tự qua đường TxD:

1. Byte ký tự cần phải truyền được ghi vào SBUF.
2. Truyền bit Start
3. Truyền ký tự 8 bit lần lượt từng bit một.
4. Bit Stop được truyền xong, trong quá trình truyền bit Stop thì cờ TI được bật (TI = 1) bởi 8051 để báo sẵn sàng để truyền ký tự kế tiếp.
5. Bằng việc kiểm tra cờ TI ta biết chắc rằng ta không nạp quá nhanh vào thanh ghi SBUF. Nếu ta nạp một byte vào SBUF trước ghi TI được bật thì phần dữ liệu của byte trước chưa truyền hết sẽ bị mất. Ta phải đợi 8051 bật cờ TI để báo đã truyền xong một byte và nó sẵn sàng truyền byte kế tiếp.
6. Trước khi SBUF được nạp một byte mới thì cờ TI phải được xóa để kiểm tra cho lần truyền dữ liệu tiếp theo.

Từ phần trình bày trên đây ta kết luận rằng: bằng việc kiểm tra bit cờ ngắt TI ta biết được 8051 có sẵn sàng để truyền một byte khác không. Quan trọng hơn cần phải nói ở đây là bit cờ TI

được bật bởi 8051 khi nó hoàn tất việc truyền một byte dữ liệu, còn việc xoá nó thì phải được lập trình viên thực hiện. Cũng cần lưu ý rằng, nếu ta ghi một byte vào thanh ghi SBUF trước khi cờ TI được bật thì sẽ có nguy cơ mất phần dữ liệu đang truyền. Bit cờ TI có thể kiểm tra bằng một vòng lặp hoặc có thể sử dụng ngắt, và ta sẽ bàn ở bài ngắt sau.

2.2.5 Lập trình 8051 để nhận dữ liệu nối tiếp

Để lập trình 8051 nhận các byte ký tự nối tiếp thì phải thực hiện các bước sau đây:

1. Nạp giá trị **20H** vào thanh ghi **TMOD**: báo sử dụng bộ Timer1, chế độ 2 (8 bit, tự động nạp lại) để thiết lập tốc độ baud.
2. Nạp **TH1** các giá trị phù hợp để thiết lập tốc độ baud.
3. Nạp giá trị **50H** vào thanh ghi **SCON** để báo sử dụng chế độ truyền nối tiếp 1: dữ liệu được đóng gói bởi 8 bit dữ liệu, 1 bit Start và 1 bit Stop.
4. Bật **TR1=1** để khởi động Timer1.
5. Xoá cờ nhận **RI**: RI=0.
6. Bit cờ nhận **RI** được kiểm tra bằng một vòng lặp để đảm bảo toàn bộ ký tự đã được nhận đủ (khi RI=1).
7. Khi **RI** được thiết lập thì trong **SBUF** đã có 1 byte. Các nội dung của nó cần được đọc ngay để tránh mất mát.
8. Để nhận một ký tự tiếp theo quay trở về **bước 5**.

Hãy quan sát ví dụ sau để thực hành:

Ví dụ 4:

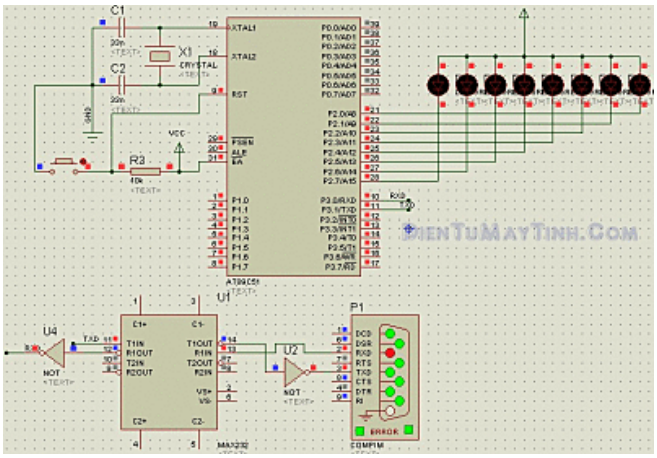
Hãy lập trình cho 8051 để nhận các byte dữ liệu nối tiếp tốc độ **9600 baud** và bật các **Led trên Port 2** tương ứng: Máy tính gửi xuống số 1: 1 Led sáng, số 2: 2 Led sáng, … , số 8: 8 Led sáng, nếu các ký tự khác thì tắt tất cả các Led.

Lời giải:

```
#include<at89x51.h>                //Khai báo thư viện cho 89c51
char c;
main()                                      //Chương trình chính
{
        TMOD=0x20;                 //Chọn Timer1, chế độ 2
        TH1=0xFD;                   //Cài đặt tốc độ 9600 baud
        SCON=0x50;                          //0101 0000: Chọn chế độ 1, Cho phép nhận
        TR1=1;                      //Khởi động Timer1

        while(1)                    //Vòng lặp vô hạn
        {
                while(RI==1)                //Vòng lặp kiểm tra cờ nhận RI
                {                    //Nếu RI=1 tức là đã nhận đủ 1 byte.
                        c=SBUF;     //lưu dữ liệu nhận được vào biến c
                        RI=0;       //Xóa cờ nhận RI.
                }
                switch(c)           //Kiểm tra ký tự vừa nhận được: tương ứng
                        {                   //trường hợp nào thì thực thi lệnh tươngứng.
                        case '1':
                                P2=0xFE;
                        break;
                        case '2':
                                P2=0xFC;
                        break;
                        case '3':
                                P2=0xF8;
                        break;
```

```
                        case '4':
                                P2=0xF0;
                        break;
                        case '5':
                                P2=0xE0;
                        break;
                        case '6':
                                P2=0xC0;
                        break;
                        case '7':
                                P2=0x80;
                        break;
                        case '8':
                                P2=0x00;
                        break;
                        default:                //mặc định là tắt tất cả Led.
                                P2=0xFF;
                        break;
                }
        }
}
```



Hình 13: Kết quả mô phỏng **Ví Dụ 4** (quá trình nhận lần lượt các ký tự 0,1,2,...,8 truyền xuống từ máy tính).

Ø **Tầm quan trọng của cờ RT**

Khi nhận các bit qua chân RxD của nó thì 8051 phải trải qua các bước sau:

1. Nó nhận bit Start báo rằng bit sau nó là bit dữ liệu đầu tiên cần phải nhận.
2. Ký tự 8 bit được nhận lần lượt từng bit một. Khi bit cuối cùng được nhận thì một byte được hình thành và đặt vào trong SBUF.
3. Khi bit Stop được nhận thì 8051 bật RT = 1 để báo rằng toàn bộ ký tự được nhận và phải lấy đi trước khi nó bị byte mới nhận về ghi đè lên.
4. Bằng việc kiểm tra bit cờ RI khi nó được bật lên chúng ta biết rằng một ký tự đã được nhận và đang nằm trong SBUF. Sao nội dung SBUF vào nơi an toàn trong một thanh ghi hay bộ nhớ khác trước khi nó bị mất.
5. Sau khi SBUF được ghi vào nơi an toàn thì cờ RI được xoá về 0 để chuẩn bị kiểm tra chu trình tiếp theo.

Từ mô tả trên đây ta rút ra kết luận rằng bằng việc kiểm tra cờ RI ta biết 8051 đã nhận được một byte ký tự chưa. Sai khi cờ RI=1, nếu ta không sao được nội dung của thanh ghi SBUF vào nơi an toàn thì có nguy cơ ta bị mất ký tự vừa nhận được. Quan trọng hơn là phải nhớ rằng cờ RI được 8051 bật lên nhưng lập trình viên phải xoá nó sau khi nhận được dữ liệu. Cũng nên

nhờ rằng, nếu ta sao nội dung SBUF vào nơi an toàn trước khi RI được bật thì ta đã mạo hiểm sao dữ liệu chưa đầy đủ. Bit cờ RI có thể được kiểm tra bởi một vòng lặp hoặc bằng ngắt mà ta sẽ bàn ở bài sau.

2.2.6 Nhận dữ liệu nối tiếp dựa trên các ngắt

Ta phải thấy rằng thật lãng phí thời gian để các bộ vi điều khiển phải luôn kiểm tra các cờ **TI** và **RI**. Do vậy, để tăng hiệu suất của 8051 ta có thể lập trình các cổng truyền thông nối tiếp của nó bằng các ngắt. Nội dung này sẽ được đề cập đến ở bài tiếp theo.

Ø Như vậy qua bài học này chúng ta đã biết truyền thông nối tiếp trong 8051 sử dụng phương pháp **không đồng bộ**, bằng cách **đóng khung dữ liệu** giữa các bit Start, bit Stop. Thanh ghi **SBUF** được sử dụng để vận chuyển dữ liệu, còn muốn thiết lập các **chế độ** truyền ta sử dụng thanh ghi **SCON**, để cài đặt **tốc độ baud** ta sử dụng **Timer1**, các cờ**TI** và **RI** là rất quan trọng vì nó báo cho ta biết lúc nào đã truyền hoặc nhận xong dữ liệu. Để truyền thông nối tiếp với máy tính qua cổng **COM** thì chúng ta phải chuyển đổi các mức điện áp cho phù hợp bằng cách sử dụng **IC Max232**.

Ở các ví dụ trên, chúng ta cũng đã biết cách gửi **1 ký tự** hoặc một **chuỗi ký tự** lên máy tính. Vậy còn muốn gửi lên **giá trị của biến**, chúng ta sẽ thực hiện như thế nào? Hoàn toàn tương tự như gửi các **chuỗi**, **ví dụ** sau đây sẽ thực hiện công việc này, bao gồm gửi các biến số nguyên, và số thực.

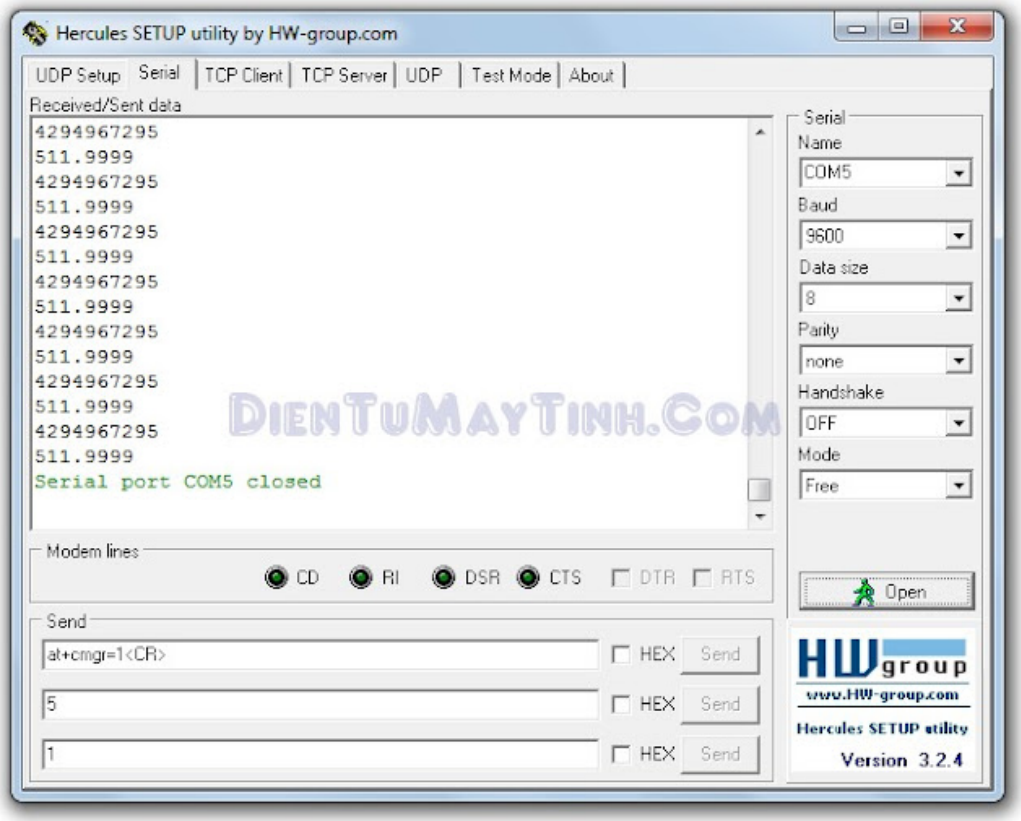
Ø **Gửi giá trị của biến**

```
#include <at89x51.h>                  //Khai báo thư viện 89x51
#include <string.h>                    //Khai báo thư viện xử lý chuỗi
unsigned long a=4294967295;      //Biến unsigned long, miền giá trị: 0->4294967295
float b=511.9999;                      //Biến float.
void send(unsigned char a);          //Khai báo nguyên mẫu hàm gửi 1 ký tự
void sendsonguyen(unsigned long n);      //hàm gửi 1 số nguyên
void sendsothuc(float n);                  //hàm gửi 1 số thực
main()                                    //Chương trình chính
{
        TMOD=0x20;                    //Chọn Timer1, chế độ 2
        TH1=0xFD;                     //Cài đặt tốc độ 9600 baud
        SCON=0x50;                    //0101 0000: Chọn chế độ 1, Cho phép nhận
        TR1=1;                         //Khởi động Timer1

        while(1)                       //Vòng lặp vô hạn
        {
                sendsonguyen(a);     //Gửi biến số nguyên
                send(10);             //Gửi dấu xuống dòng
                sendsothuc(b);       //Gửi biến số thực
                send(10);             //Gửi dấu xuống dòng
        }
}
void send(unsigned char a)          //Định nghĩa hàm gửi 1 ký tự
{
   SBUF=a;                              //Ghi 1 byte dữ liệu vào thanh ghi SBUF
   while(TI==0){}                     //vòng lặp đợi cờ truyền dữ liệu TI bật lên 1
   TI=0;                                 //Xóa cờ TI sau khi truyền dữ liệu xong
}
void sendsonguyen(unsigned long n)      // Định nghĩa hàm gửi 1 số nguyên
{
       if(n!=0)                              //Trường hợp số nguyên #0
        {
```

```
                unsigned char a[11];                    //mảng chứa các ký tự số sau khi tách số
                int i;                                  //biến chỉ số cho vòng for
                for(i=0;n>0;i++)                        //Vòng lặp tách các chữ số thành ký tự
                {
                        a[i]=(n%10)+48;                 //tách lấy chữ số hàng đơn vị,mã hóa ASCII
                        n=n/10;                         //loại bỏ chữ số hàng đơn vị
                }
                a[i]=NULL;                              //ký tự cuối cùng của chuỗi phải là NULL
           for(i=strlen(a);i>=0;i--)                    //Vòng lặp gửi lần lượt từng ký tự lên,
           {                                            //cho đến khi hết chuỗi a[].
                send(a[i]);                             //Gọi hàm gửi 1 ký tự.

           }
        }
        else send('0');                                 //Trường hợp số nguyên =0: chỉ cần gửi số 0
}
void sendsothuc(float n)                                // Định nghĩa hàm gửi 1 số thực
{
        unsigned long a=n/1;                            //Tách lấy phần nguyên của số thực
        unsigned long b=(n-a)*10000;                    //Tách lấy phần thập phân của số thực
        sendsonguyen(a);                                //Gọi hàm để gửi phần nguyên
        if(b!=0)                                        //Trường hợp tồn tại phần thập phân.
        {
                send('.');                              //Gửi ký tự '.'
                sendsonguyen(b);                        //Gọi hàm để gửi phần thập phân
        }                                               //Nếu không có phần thập phân thì không làm gì
}
```



Hình 14: Kết quả nhận được trên giao diện Hercules ở máy tính

Nhận xét

Bạn không có quyền thêm nhận xét.