

# Principle and Interface Techniques of Microcontroller

--8051 Microcontroller and Embedded Systems  
Using Assembly and C

**LI, Guang (李光)** Prof. PhD, DIC, MIET

**WANG, You (王酉)** PhD, MIET

杭州 • 浙江大学 • 2015

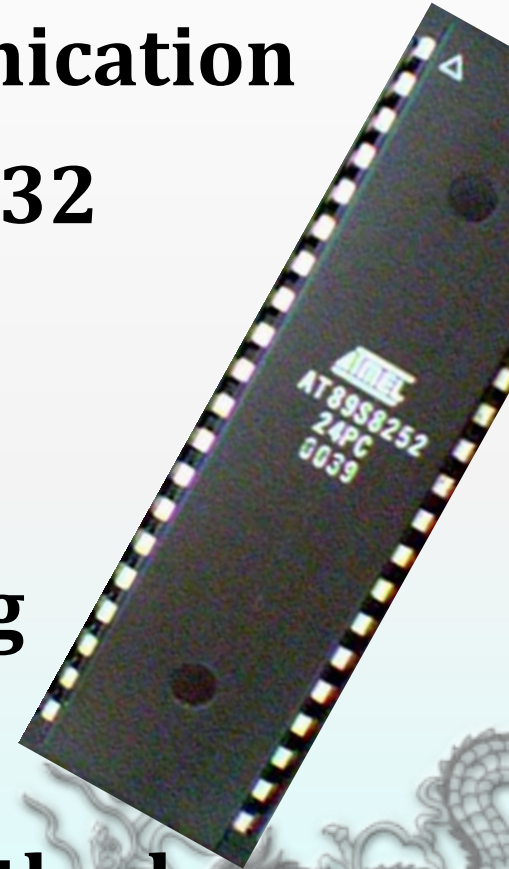
# Chapter 11

## Serial Communication



# Outline

- ◆ § 11-1 Basics of Serial Communication
- ◆ § 11-2 8051 Connection to RS232
- ◆ § 11-3 Serial Communication Programming
- ◆ § 11-4 Serial Port Programming in C
- ◆ § 11-5 New communication methods



# § 11-1 Basics of Serial Communication

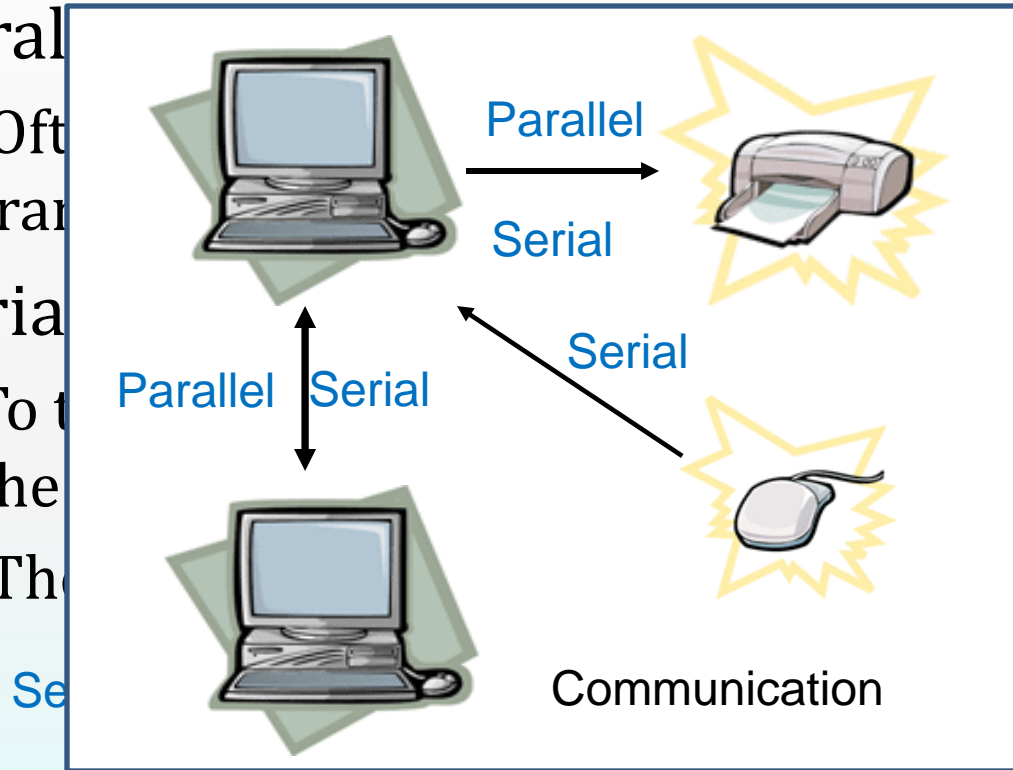
## ❖ Computers transfer data in two ways:

### ➤ Parallel

- ✓ Often used to transfer data

### ➤ Serial

- ✓ To transfer data over long distances
- ✓ The data is transferred one bit at a time



(s) are used to transfer data a few feet away

meters away,

el Transfer

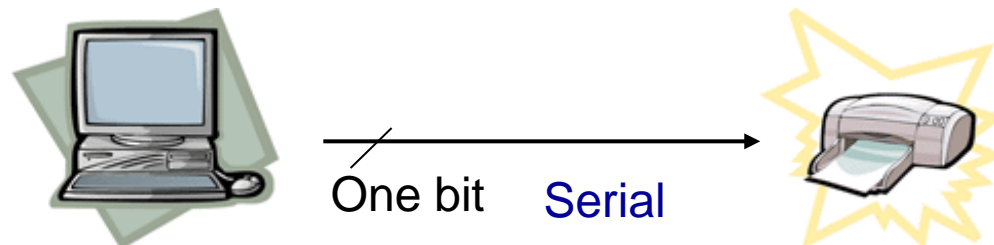
Sender

Receiver

Sender

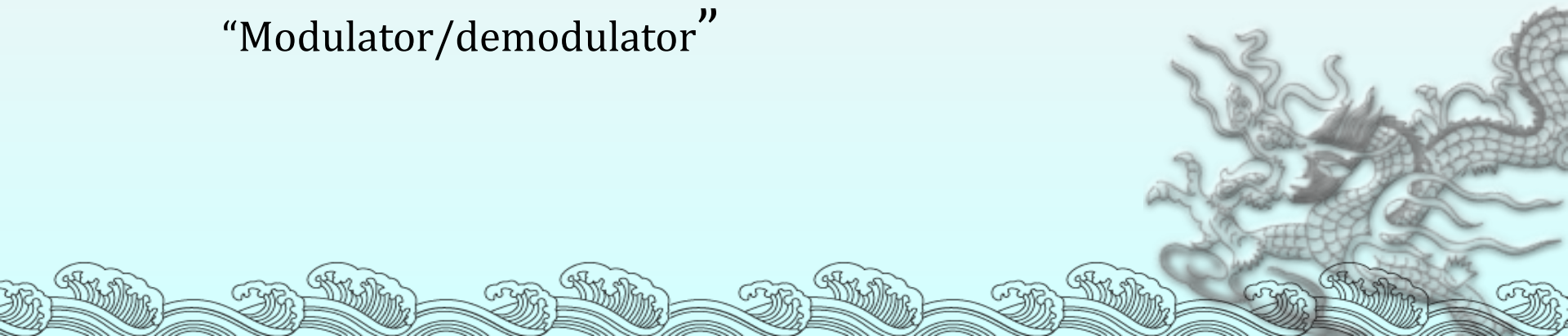
Receiver

D7



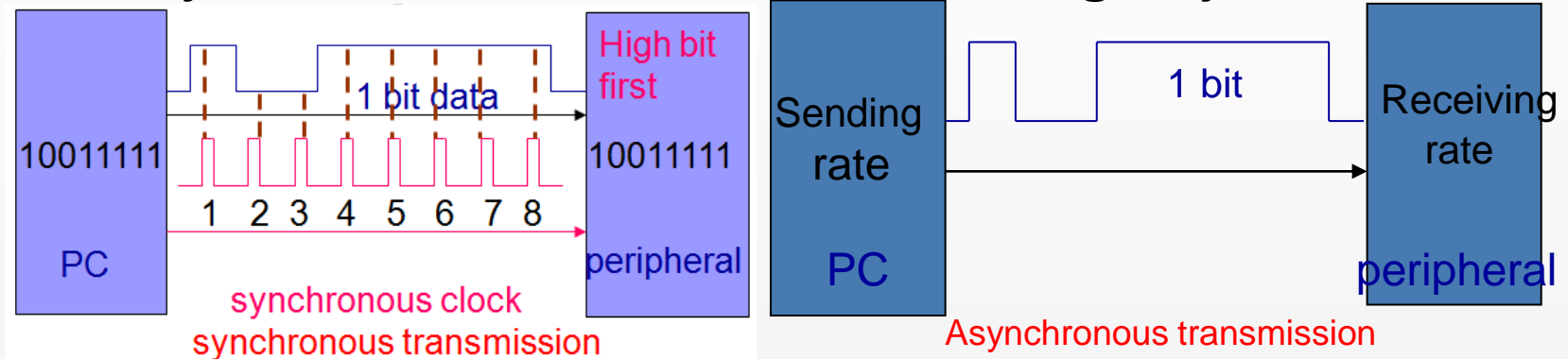
# § 11-1 Basics of Serial Communication

- ◆ At the transmitting end, the byte of data must be converted to serial bits using parallel-in-serial-out shift register
- ◆ At the receiving end, there is a serial in-parallel-out shift register to receive the serial data and pack them into byte
- ◆ When the distance is short, the digital signal can be transferred as it is on a simple wire and requires no modulation
- ◆ If data is to be transferred on the telephone line, it must be converted from 0s and 1s to audio tones
  - This conversion is performed by a device called a modem, “Modulator/demodulator”





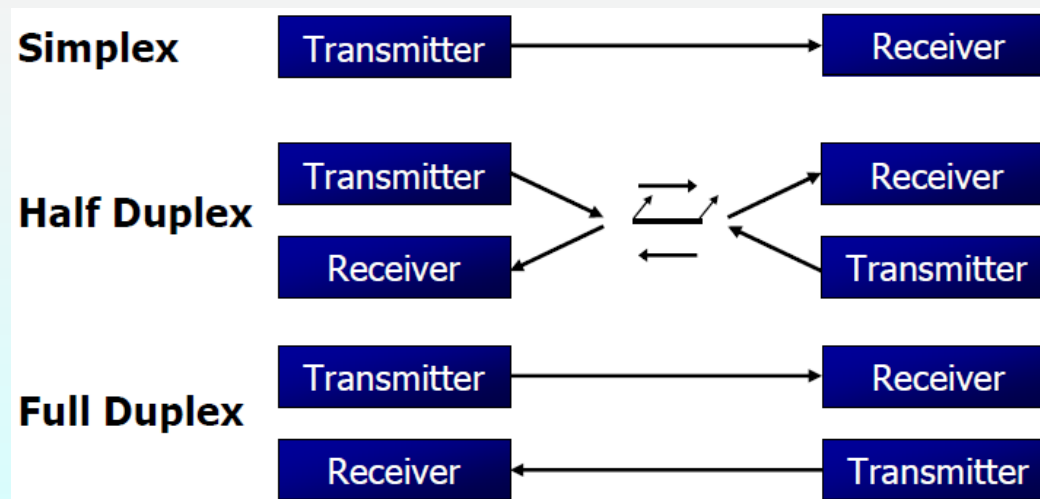
- Serial data communication uses two methods
  - Synchronous method transfers a block of data at a time
  - Asynchronous method transfers a single byte at a time



- It is possible to write software to use either of these methods, but the programs can be tedious and long
  - There are special IC chips made by many manufacturers for serial communications
  - UART (universal asynchronous Receiver-transmitter)
  - USART (universal synchronous-asynchronous Receiver-transmitter)

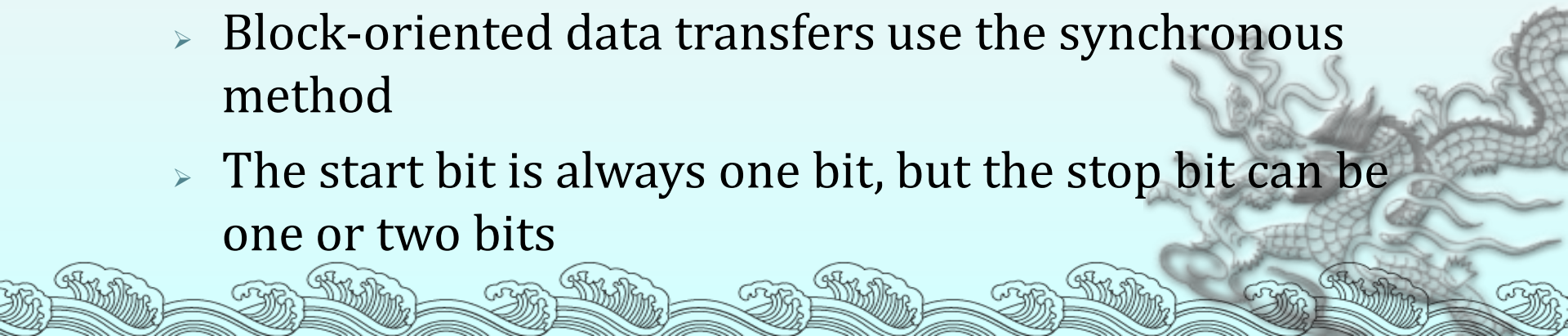
# Half- and Full-Duplex Transmission

- ◆ If data can be transmitted and received, it is a duplex transmission
  - If data transmitted one way a time, it is referred to as half duplex
  - If data can go both ways at a time, it is full-duplex
- ◆ This is contrast to simplex transmission



# Start and Stop Bits

- ◆ A protocol is a set of rules agreed by both the sender and receiver on
  - How the data is packed
  - How many bits constitute a character
  - When the data begins and ends
- ◆ Asynchronous serial data communication is widely used for character-oriented transmissions
  - Each character is placed in between start and stop bits, this is called framing
  - Block-oriented data transfers use the synchronous method
  - The start bit is always one bit, but the stop bit can be one or two bits

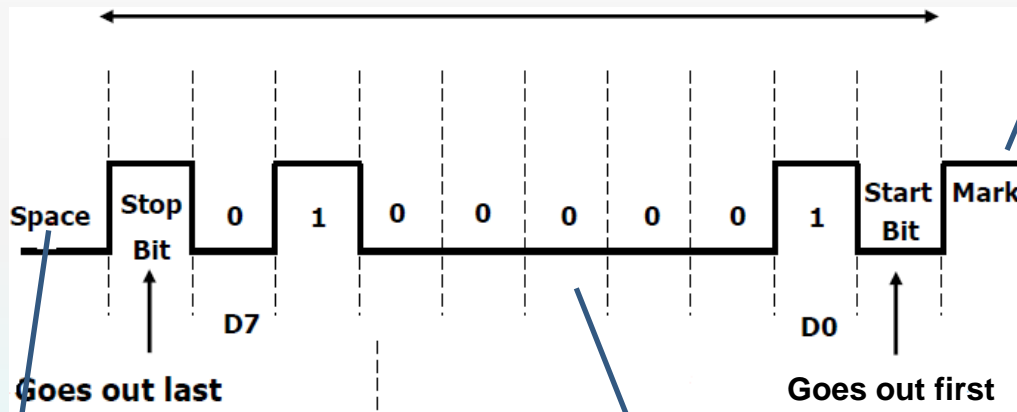




# Start and Stop Bits

- ◆ The start bit is always a 0 (low) and the stop bit(s) is 1 (high)

ASCII character "A" (8-bit binary 0100 0001)



When there is no transfer, the signal is 1 (high), which is referred to as *mark*

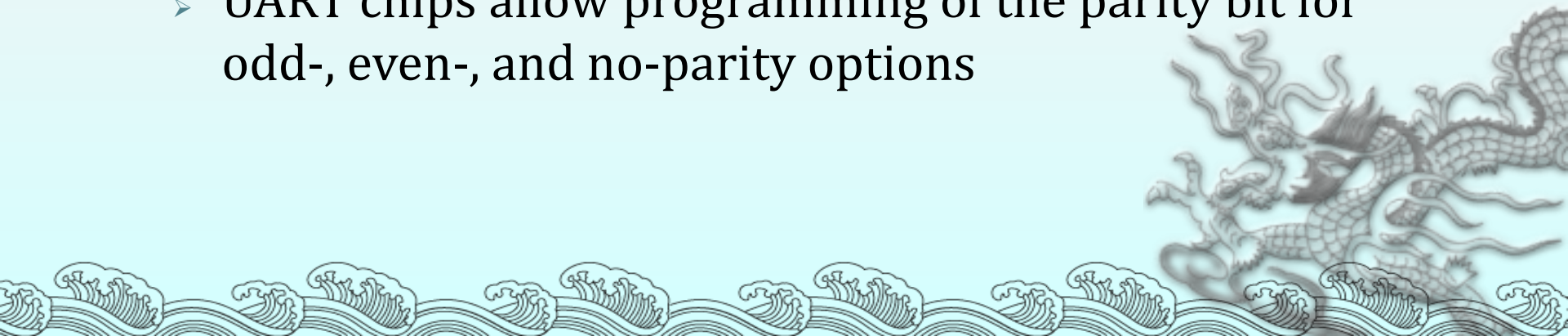
The 0 (low) is referred to as *space*

The transmission begins with a start bit followed by D0, the LSB, then the rest of the bits until MSB (D7), and finally, the one stop bit indicating the end of the character

- ◆ Due to the extended ASCII characters, 8-bit ASCII data is common
  - In older systems, ASCII characters were 7-bit
- ◆ In modern PCs the use of one stop bit is standard
  - In older systems, due to the slowness of the receiving mechanical device, two stop bits were used to give the device sufficient time to organize itself before transmission of the next byte

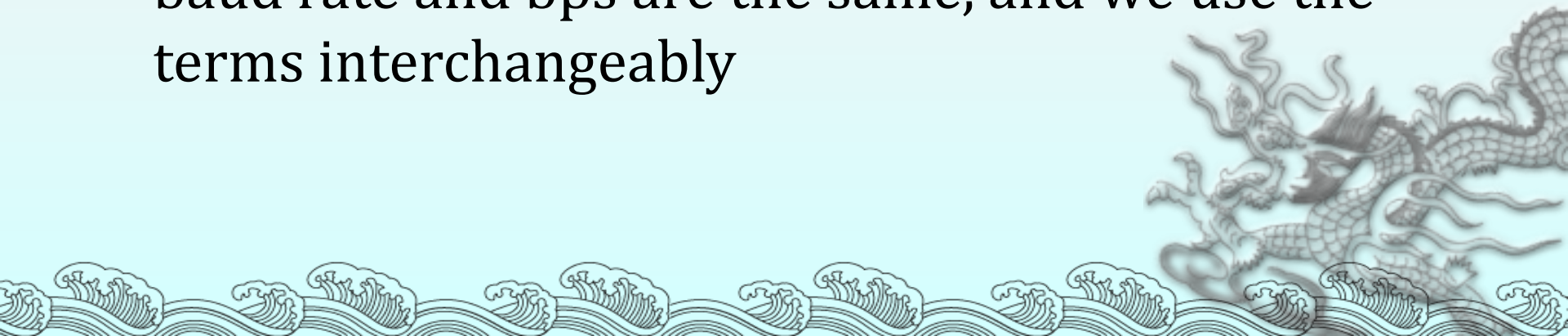


- ◆ Assuming that we are transferring a text file of ASCII characters using 1 stop bit, we have a total of 10 bits for each character
  - This gives 25% overhead, i.e. each 8-bit character with an extra 2 bits
- ◆ In some systems in order to maintain data integrity, the parity bit of the character byte is included in the data frame
  - UART chips allow programming of the parity bit for odd-, even-, and no-parity options



# Data Transfer Rate

- ◆ The rate of data transfer in serial data communication is stated in bps (bits per second)
- ◆ Another widely used terminology for bps is baud rate
  - It is modem terminology and is defined as the number of signal changes per second
  - In modems, there are occasions when a single change of signal transfers several bits of data
- ◆ As far as the conductor wire is concerned, the baud rate and bps are the same, and we use the terms interchangeably



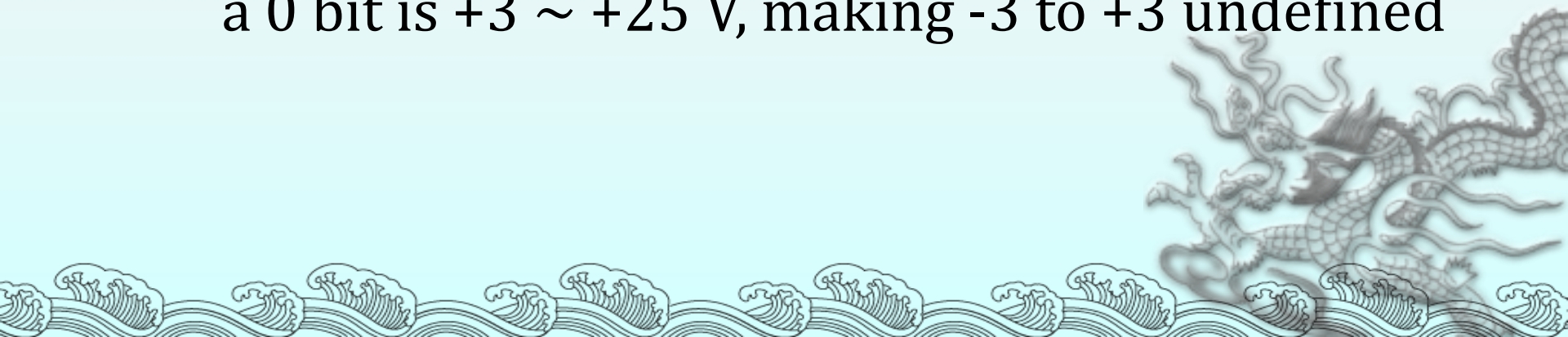
- ◆ The data transfer rate of given computer system depends on communication ports incorporated into that system
  - IBM PC/XT could transfer data at the rate of 100 to 9600 bps
  - Pentium-based PCs transfer data at rates as high as 56K bps
  - In asynchronous serial data communication, the baud rate is limited to 100K bps





# RS232 Standards

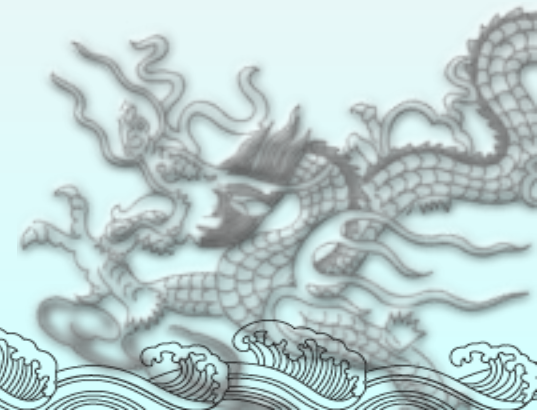
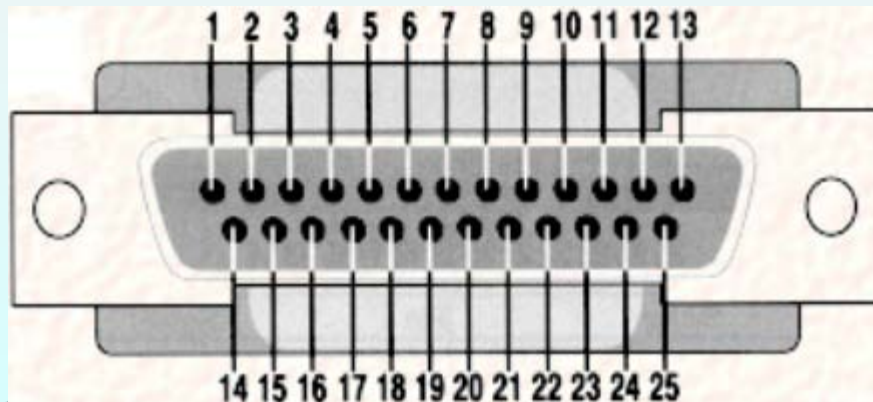
- ◆ An interfacing standard RS232 was set by the Electronics Industries Association (EIA) in 1960
- ◆ The standard was set long before the advent of the TTL logic family, its input and output voltage levels are not TTL compatible
  - In RS232, a 1 is represented by  $-3 \sim -25$  V, while a 0 bit is  $+3 \sim +25$  V, making  $-3$  to  $+3$  undefined



## RS232 DB-25 Pins

Pin	Description	Pin	Description
1	Protective ground	14	Secondary transmitted data
2	Transmitted data (TxD)	15	Transmitted signal element timing
3	Received data (RxD)	16	Secondary receive data
4	Request to send (-RTS)	17	Receive signal element timing
5	Clear to send (-CTS)	18	Unassigned
6	Data set ready (-DSR)	19	Secondary receive data
7	Signal ground (GND)	20	Data terminal ready (-DTR)
8	Data carrier detect (-DCD)	21	Signal quality detector
9/10	Reserved for data testing	22	Ring indicator (RI)
11	Unassigned	23	Data signal rate select
12	Secondary data carrier detect	24	Transmit signal element timing
13	Secondary clear to send	25	Unassigned

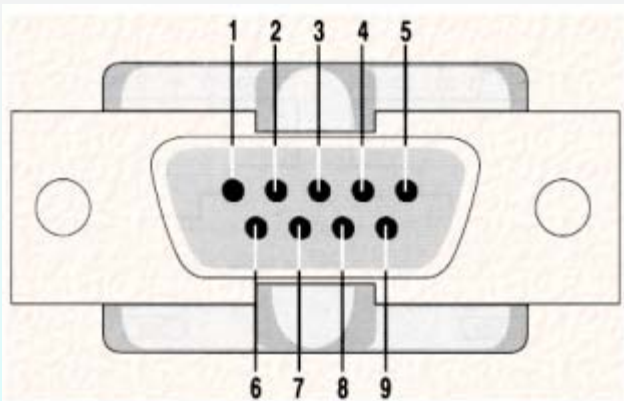
## RS232 Connector DB-25



# RS232 Standards

- ❖ Since not all pins are used in PC cables, IBM introduced the DB-9 version of the serial I/O standard

RS232 Connector DB-9



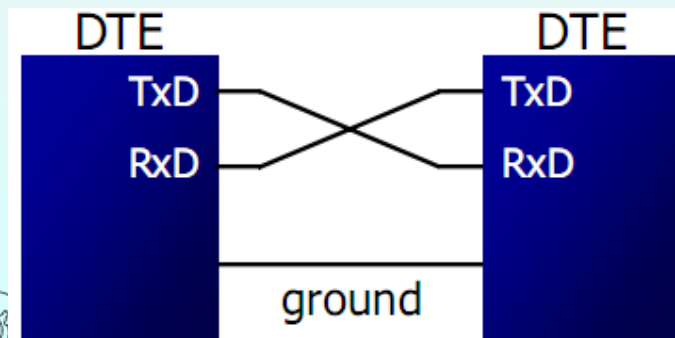
RS232 DB-9 Pins

Pin	Description
1	Data carrier detect (-DCD)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (-DSR)
7	Request to send (-RTS)
8	Clear to send (-CTS)
9	Ring indicator (RI)

# Data Communication Classification

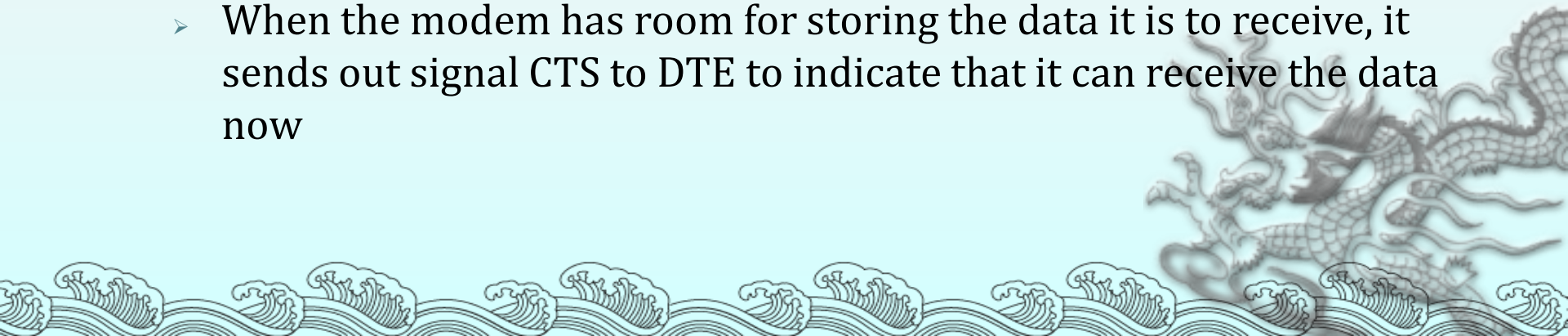
- ◆ Current terminology classifies data communication equipment as
  - DTE (data terminal equipment) refers to terminal and computers that send and receive data
  - DCE (data communication equipment) refers to communication equipment, such as modems
- ◆ The simplest connection between a PC and microcontroller requires a minimum of three pins, TxD, RxD, and ground

## Null modem connection



# RS232 Pins

- ◆ DTR (data terminal ready)
  - When terminal is turned on, it sends out signal DTR to indicate that it is ready for communication
- ◆ DSR (data set ready)
  - When DCE is turned on and has gone through the self-test, it assert DSR to indicate that it is ready to communicate
- ◆ RTS (request to send)
  - When the DTE device has byte to transmit, it assert RTS to signal the modem that it has a byte of data to transmit
- ◆ CTS (clear to send)
  - When the modem has room for storing the data it is to receive, it sends out signal CTS to DTE to indicate that it can receive the data now





# RS232 Pins

- ◆ DCD (data carrier detect)
  - The modem asserts signal DCD to inform the DTE that a valid carrier has been detected and that contact between it and the other modem is established
- ◆ RI (ring indicator)
  - An output from the modem and an input to a PC indicates that the telephone is ringing
  - It goes on and off in synchronous with the ringing sound



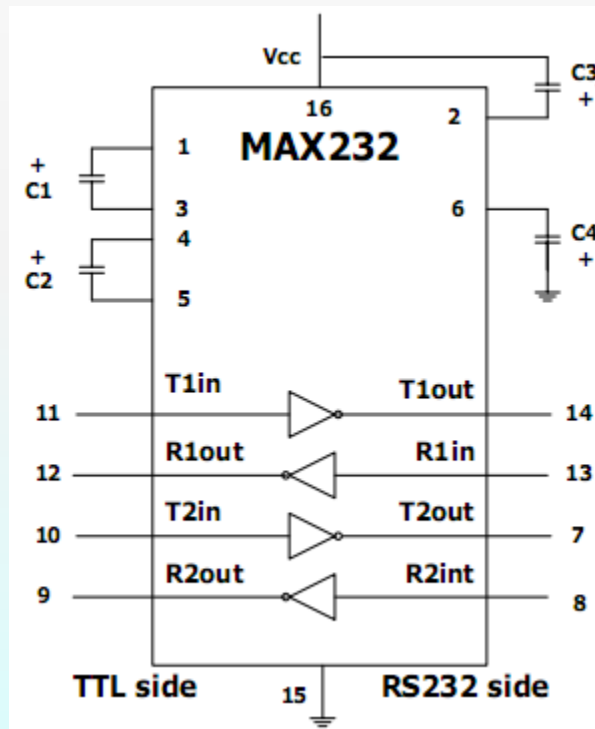
## § 11-2 8051 Connection to RS232

- ◆ A line driver such as the MAX232 chip is required to convert RS232 voltage levels to TTL levels, and vice versa
- ◆ 8051 has two pins that are used specifically for transferring and receiving data serially
  - These two pins are called TxD and RxD and are part of the port 3 group (P3.0 and P3.1)
  - These pins are TTL compatible; therefore, they require a line driver to make them RS232 compatible

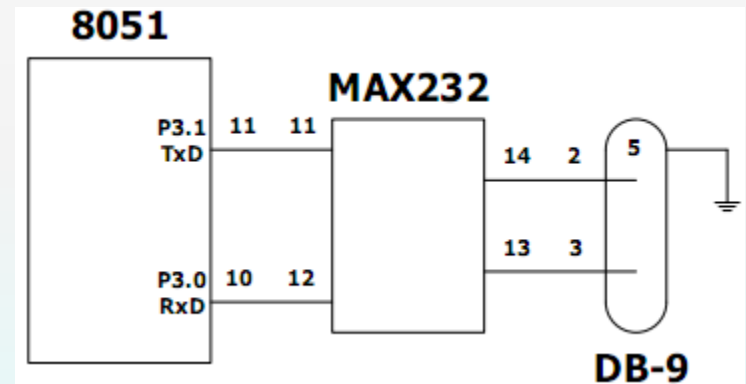


# MAX232

- ◆ We need a line driver (voltage converter) to convert the R232's signals to TTL voltage levels that will be acceptable to 8051's TxD and RxD pins



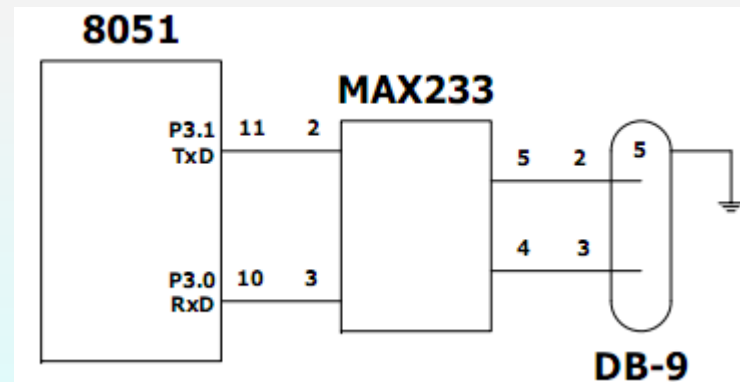
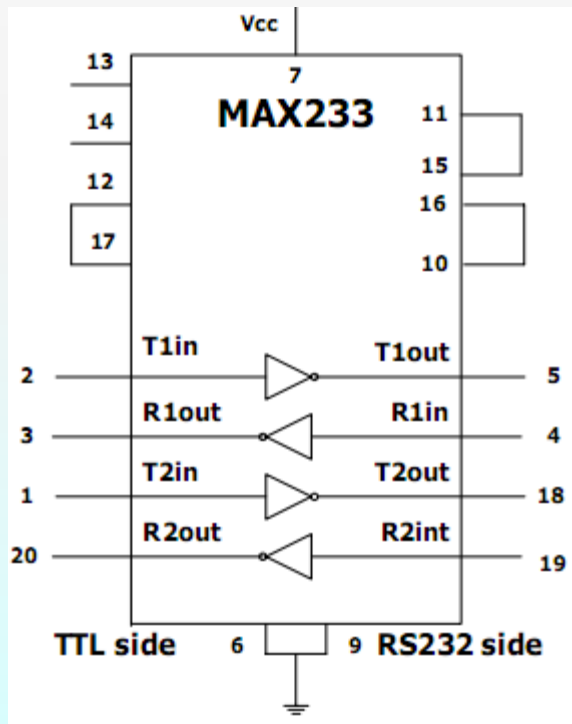
MAX232 requires four capacitors



MAX232 has two sets of line drivers

# MAX233

- ◆ To save board space, some designers use MAX233 chip from Maxim
  - MAX233 performs the same job as MAX232 but eliminates the need for capacitors
  - Notice that MAX233 and MAX232 are not pin compatible



# § 11-3 Serial Communication Programming

- ◆ To allow data transfer between the PC and an 8051 system without any error, we must make sure that the baud rate of 8051 system matches the baud rate of the PC's COM port
- ◆ Hyperterminal function supports baud rates much higher than listed below

PC Baud Rates

110
150
300
600
1200
2400
4800
9600
19200

Baud rates supported by  
486/Pentium IBM PC BIOS



With XTAL = 11.0592 MHz, find the TH1 value needed to have the following baud rates. (a) 9600 (b) 2400 (c) 1200

Solution:

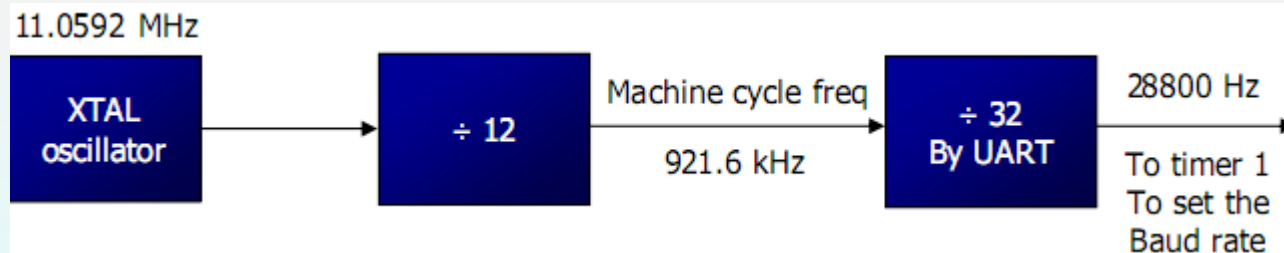
The machine cycle frequency of 8051 =  $11.0592 / 12 = 921.6$  kHz, and  $921.6 \text{ kHz} / 32 = 28,800 \text{ Hz}$  is frequency by UART to timer 1 to set baud rate.

(a)  $28,800 / 3 = 9600$  where -3 = FD (hex) is loaded into TH1

(b)  $28,800 / 12 = 2400$  where -12 = F4 (hex) is loaded into TH1

(c)  $28,800 / 24 = 1200$  where -24 = E8 (hex) is loaded into TH1

Notice that dividing 1/12 of the crystal frequency by 32 is the default value upon activation of the 8051 RESET pin.



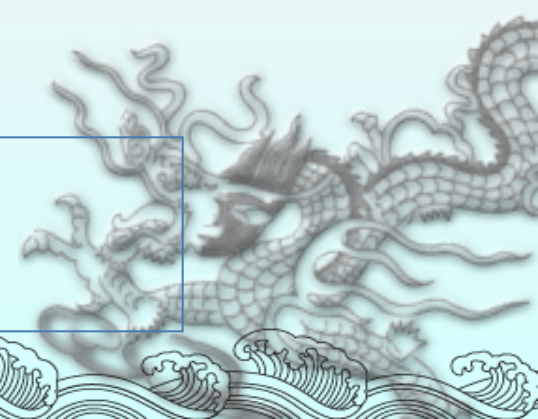
Baud Rate	TH1 (Decimal)	TH1 (Hex)
9600	-3	FD
4800	-6	FA
2400	-12	F4
1200	-24	E8

TF is set to 1 every 12 ticks, so it functions as a frequency divider

# SBUF Register

- ❖ SBUF is an 8-bit register used solely for serial communication
  - For a byte data to be transferred via the TxD line, it must be placed in the SBUF register
    - ✓ The moment a byte is written into SBUF, it is framed with the start and stop bits and transferred serially via the TxD line
  - SBUF holds the byte of data when it is received by 8051 RxD line
    - ✓ When the bits are received serially via RxD, the 8051 deframes it by eliminating the stop and start bits, making a byte out of the data received, and then placing it in SBUF

```
MOV SBUF,#'D' ;load SBUF=44h, ASCII for 'D'  
MOV SBUF,A    ;copy accumulator into SBUF  
MOV A,SBUF    ;copy SBUF into accumulator
```



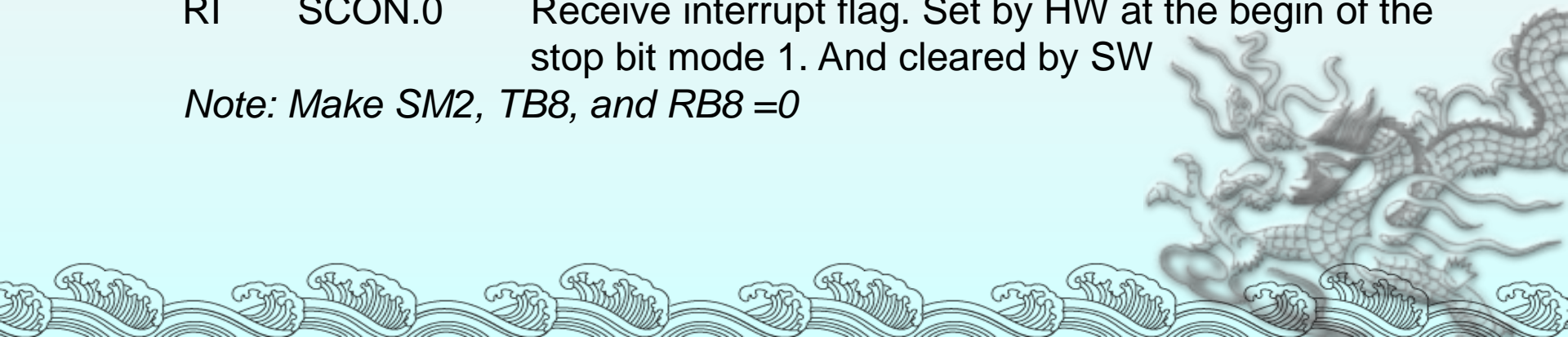
# SCON Register

- ◆ SCON is an 8-bit register used to program the start bit, stop bit, and data bits of data framing, among other things

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
-----	-----	-----	-----	-----	-----	----	----

SM0	SCON.7	Serial port mode specifier
SM1	SCON.6	Serial port mode specifier
SM2	SCON.5	Used for multiprocessor communication
REN	SCON.4	Set/cleared by software to enable/disable reception
TB8	SCON.3	Not widely used
RB8	SCON.2	Not widely used
TI	SCON.1	Transmit interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW
RI	SCON.0	Receive interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW

*Note: Make SM2, TB8, and RB8 =0*



# SCON Register

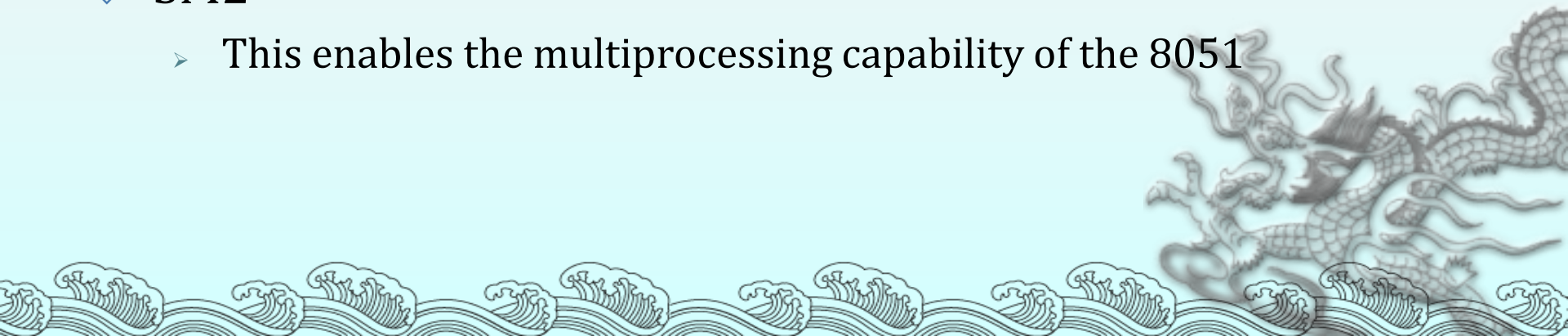
## ◆ SM0, SM1

- They determine the framing of data by specifying the number of bits per character, and the start and stop bits

SM0	SM1	
0	0	Serial Mode 0
<b>0</b>	<b>1</b>	Serial Mode 1, 8-bit data, 1 stop bit, 1 start bit
1	0	Serial Mode 2, 11 bits, $f_{osc} \times 2^{SMOD/64}$
1	1	Serial Mode 3, 11 bits

## ◆ SM2

- This enables the multiprocessing capability of the 8051



## ◆ REN (receive enable)

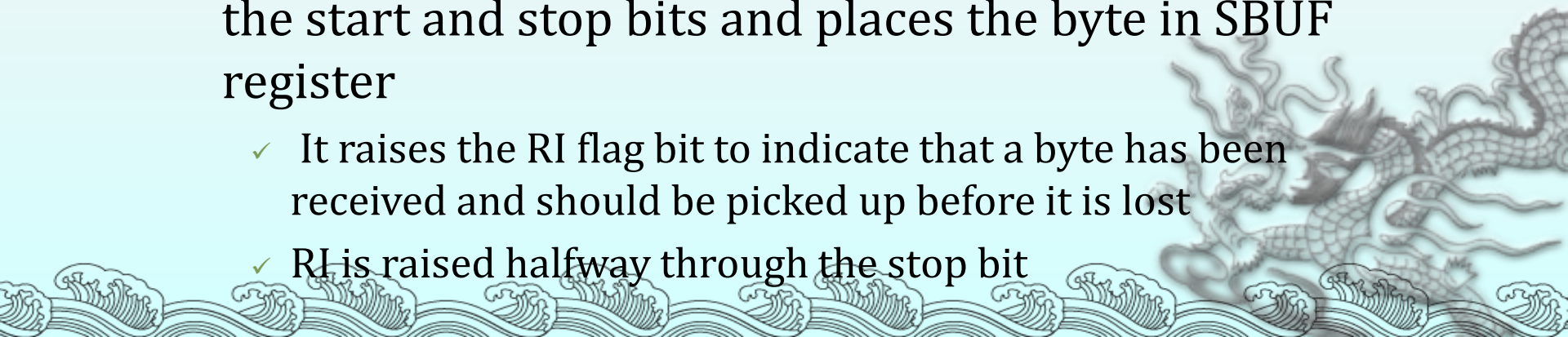
- It is a bit-addressable register
  - ✓ When it is high, it allows 8051 to receive data on RxD pin
  - ✓ If low, the receiver is disabled

## ◆ TI (transmit interrupt)

- When 8051 finishes the transfer of 8-bit character
  - ✓ It raises TI flag to indicate that it is ready to transfer another byte
  - ✓ TI bit is raised at the beginning of the stop bit

## ◆ RI (receive interrupt)

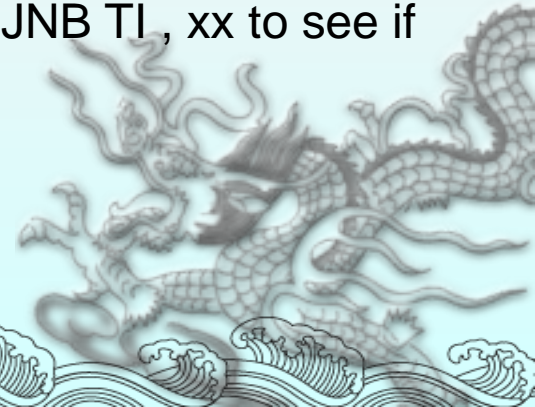
- When 8051 receives data serially via RxD, it gets rid of the start and stop bits and places the byte in SBUF register
  - ✓ It raises the RI flag bit to indicate that a byte has been received and should be picked up before it is lost
  - ✓ RI is raised halfway through the stop bit





# Programming Serial Data Transmitting

- In programming the 8051 to transfer character bytes serially
  1. TMOD register is loaded with the value 20H, indicating the use of timer1 in mode 2 (8-bit auto-reload) to set baud rate
  2. The TH1 is loaded with one of the values to set baud rate for serial data transfer
  3. The SCON register is loaded with the value 50H, indicating serial mode1, where an 8-bit data is framed with start and stop bits
  4. TR1 is set to 1 to start timer 1
  5. TI is cleared by CLR TI instruction
  6. The character byte to be transferred serially is written into SBUF register
  7. The TI flag bit is monitored with the use of instruction JNB TI , xx to see if the character has been transferred completely
  8. To transfer the next byte, go to step 5

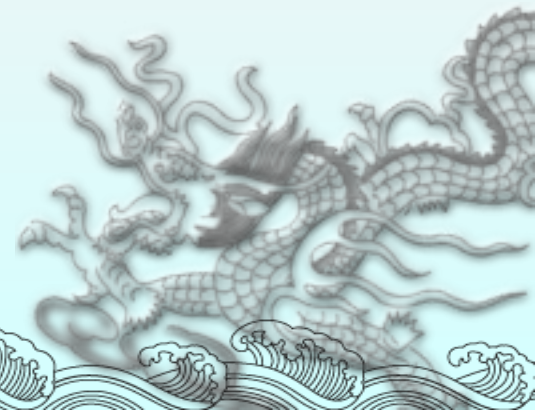


# Programming Serial Data Transmitting

Write a program for the 8051 to transfer letter “A” serially at 4800 baud, continuously.

Solution:

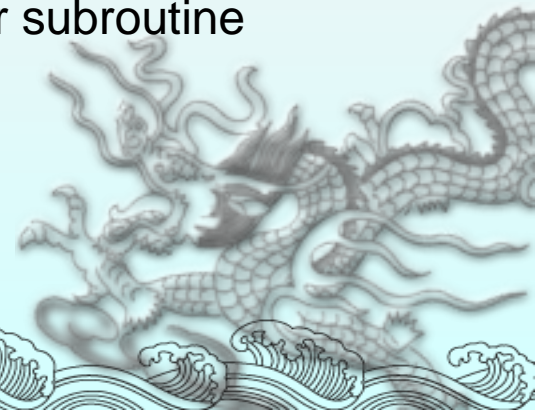
```
        MOV  TMOD,#20H    ;timer 1,mode 2(auto reload)
        MOV  TH1,#-6      ;4800 baud rate
        MOV  SCON,#50H    ;8-bit, 1 stop, REN enabled
        SETB TR1          ;start timer 1
AGAIN:   MOV  SBUF,#"A"    ;letter "A" to transfer
        HERE: JNB  TI,HERE ;wait for the last bit
        CLR  TI           ;clear TI for next char
        SJMP AGAIN        ;keep sending A
```



Write a program for the 8051 to transfer "YES" serially at 9600 baud, 8-bit data, 1 stop bit, do this continuously

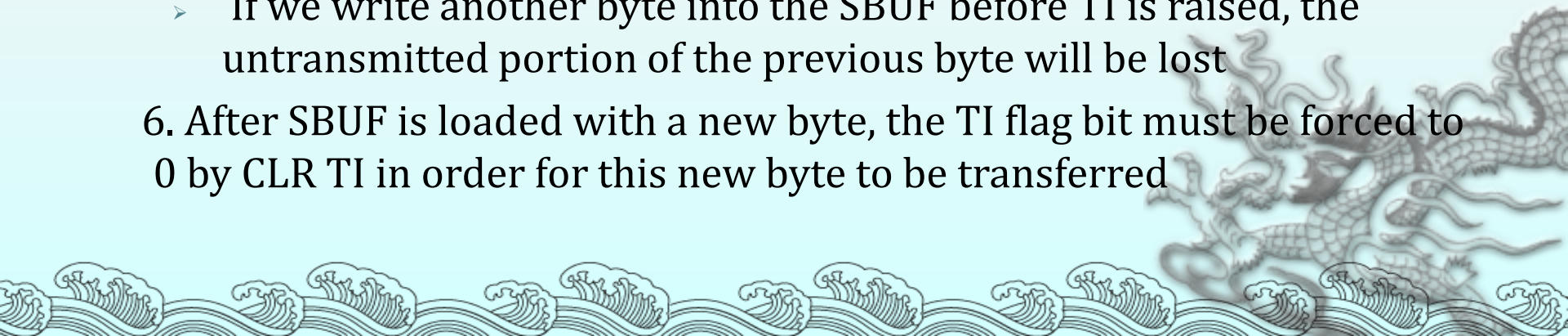
Solution:

```
        MOV  TMOD,#20H      ;timer 1,mode 2(auto reload)
        MOV  TH1,#-3        ;9600 baud rate
        MOV  SCON,#50H      ;8-bit, 1 stop, REN enabled
        SETB TR1            ;start timer 1
AGAIN:   MOV  A,#"Y"         ;transfer "Y"
        ACALL TRANS
        MOV  A,#"E"         ;transfer "E"
        ACALL TRANS
        MOV  A,#"S"         ;transfer "S"
        ACALL TRANS
        SJMP AGAIN          ;keep doing it serial data transfer subroutine
TRANS:   MOV  SBUF,A         ;load SBUF
HERE:    JNB  TI,HERE        ;wait for the last bit
        CLR  TI              ;get ready for next byte
        RET
```



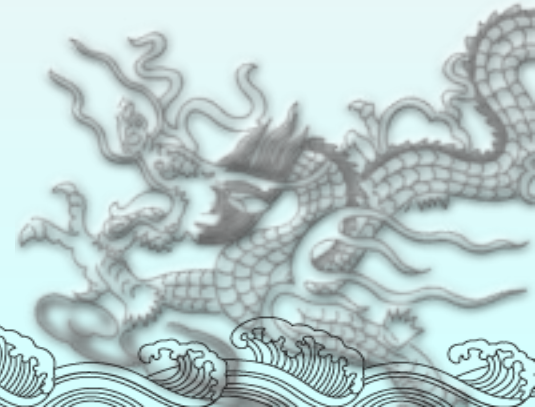
# Importance of TI Flag

- ◆ The steps that 8051 goes through in transmitting a character via TxD
  1. The byte character to be transmitted is written into the SBUF register
  2. The start bit is transferred
  3. The 8-bit character is transferred on bit at a time
  4. The stop bit is transferred
    - It is during the transfer of the stop bit that 8051 raises the TI flag, indicating that the last character was transmitted
  5. By monitoring the TI flag, we make sure that we are not overloading the SBUF
    - If we write another byte into the SBUF before TI is raised, the untransmitted portion of the previous byte will be lost
  6. After SBUF is loaded with a new byte, the TI flag bit must be forced to 0 by CLR TI in order for this new byte to be transferred



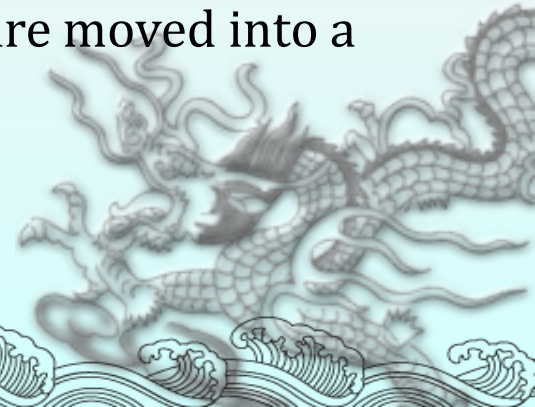
# Importance of TI Flag

- ◆ By checking the TI flag bit, we know whether or not the 8051 is ready to transfer another byte
  - It must be noted that TI flag bit is raised by 8051 itself when it finishes data transfer
  - It must be cleared by the programmer with instruction CLR TI
  - If we write a byte into SBUF before the TI flag bit is raised, we risk the loss of a portion of the byte being transferred
- ◆ The TI bit can be checked by
  - The instruction JNB TI , xx
  - Using an interrupt



# Programming Serial Data Receiving

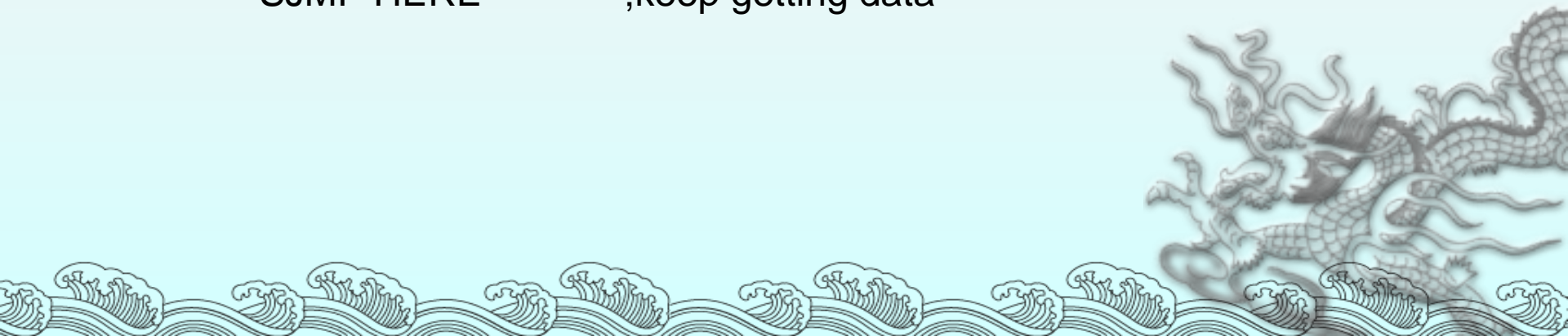
- ◆ In programming the 8051 to receive character bytes serially
  1. TMOD register is loaded with the value 20H, indicating the use of timer 1 in mode2 (8-bit auto-reload) to set baud rate
  2. TH1 is loaded to set baud rate
  3. The SCON register is loaded with the value 50H, indicating serial mode 1, where an 8-bit data is framed with start and stop bits
  4. TR1 is set to 1 to start timer 1
  5. RI is cleared by CLR RI instruction
  6. The RI flag bit is monitored with the use of instruction JNB RI,xx to see if an entire character has been received yet
  7. When RI is raised, SBUF has the byte, its contents are moved into a safe place
  8. To receive the next character, go to step 5





Write a program for the 8051 to receive bytes of data serially, and put them in P1, set the baud rate at 4800, 8-bit data, and 1 stop bit  
Solution:

```
        MOV  TMOD,#20H    ;timer 1,mode 2(auto reload)
        MOV  TH1,#-6      ;4800 baud rate
        MOV  SCON,#50H    ;8-bit, 1 stop, REN enabled
        SETB TR1          ;start timer 1
HERE:    JNB  RI,HERE      ;wait for char to come in
        MOV  A,SBUF        ;saving incoming byte in A
        MOV  P1,A         ;send to port 1
        CLR  RI           ;get ready to receive next byte
        SJMP HERE         ;keep getting data
```



## Example 11-1

Assume that the 8051 serial port is connected to the COM port of IBM PC, and on the PC, we are using the terminal.exe program to send and receive data serially. P1 and P2 of the 8051 are connected to LEDs and switches, respectively. Write an 8051 program to (a) send to PC the message “We Are Ready”, (b) receive any data send by PC and put it on LEDs connected to P1, and (c) get data on switches connected to P2 and send it to PC serially. The program should perform part (a) once, but parts (b) and (c) continuously, use 4800 baud rate.

Solution:

```
ORG 0
MOV P2,#0FFH    ;make P2 an input port
MOV TMOD,#20H   ;timer 1, mode 2
MOV TH1,#0FAH   ;4800 baud rate
MOV SCON,#50H   ;8-bit, 1 stop, REN enabled
SETB TR1        ;start timer 1
MOV DPTR,#MYDATA ;load pointer for message
H_1: CLR A
MOV A,@A+DPTR   ;get the character
JZ B_1          ;if last character get out
ACALL SEND      ;otherwise call transfer
INC DPTR        ;next one
SJMP H_1        ;stay in loop
```

```

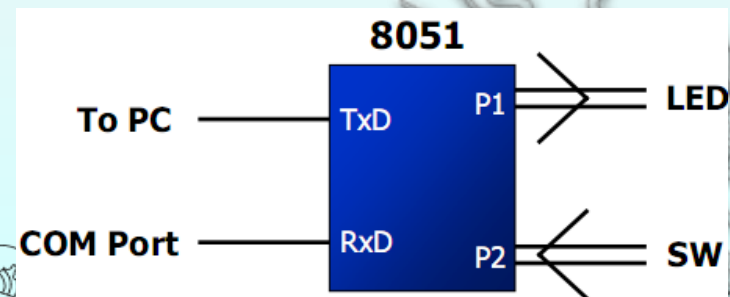
B_1:   MOV  a,P2      ;read data on P2
        ACALL SEND    ;transfer it serially
        ACALL RECV     ;get the serial data
        MOV  P1,A      ;display it on LEDs
        SJMP B_1       ;stay in loop indefinitely

;----serial data transfer. ACC has the data-----
SEND:   MOV  SBUF,A    ;load the data
H_2:    JNB  TI,H_2     ;stay here until last bit gone
        CLR  TI        ;get ready for next char
        RET           ;return to caller

;----Receive data serially in ACC-----
RECV:   JNB  RI,RECV    ;wait here for char
        MOV  A,SBUF     ;save it in ACC
        CLR  RI        ;get ready for next char
        RET           ;return to caller

;----The message-----
MYDATA: DB  "We Are Ready",0
        END

```



# Importance of RI Flag

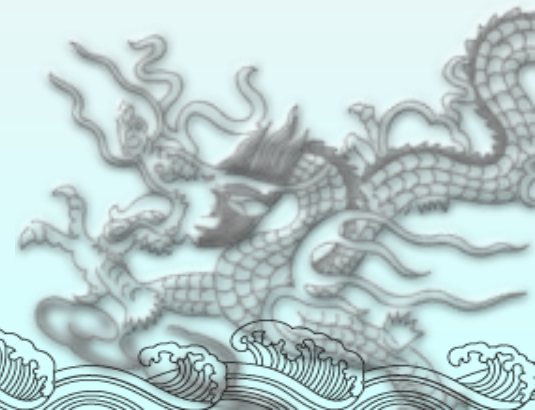
- ◆ In receiving bit via its RxD pin, 8051 goes through the following steps
  1. It receives the start bit
    - Indicating that the next bit is the first bit of the character byte it is about to receive
  2. The 8-bit character is received one bit at time
  3. The stop bit is received
    - When receiving the stop bit 8051 makes  $RI = 1$ , indicating that an entire character byte has been received and must be picked up before it gets overwritten by an incoming character



4. By checking the RI flag bit when it is raised, we know that a character has been received and is sitting in the SBUF register
  - We copy the SBUF contents to a safe place in some other register or memory before it is lost
5. After the SBUF contents are copied into a safe place, the RI flag bit must be forced to 0 by CLR RI in order to allow the next received character byte to be placed in SBUF
  - Failure to do this causes loss of the received character



- ◆ By checking the RI flag bit, we know whether or not the 8051 received a character byte
  - If we failed to copy SBUF into a safe place, we risk the loss of the received byte
  - It must be noted that RI flag bit is raised by 8051 when it finish receive data
  - It must be cleared by the programmer with instruction CLR RI
  - If we copy SBUF into a safe place before the RI flag bit is raised, we risk copying garbage
- ◆ The RI bit can be checked by
  - The instruction JNB RI,xx
  - Using an interrupt





# Doubling Baud Rate

- ◆ There are two ways to increase the baud rate of data transfer

- To use a higher frequency crystal
- To change a bit in the PCON register

The system crystal is fixed

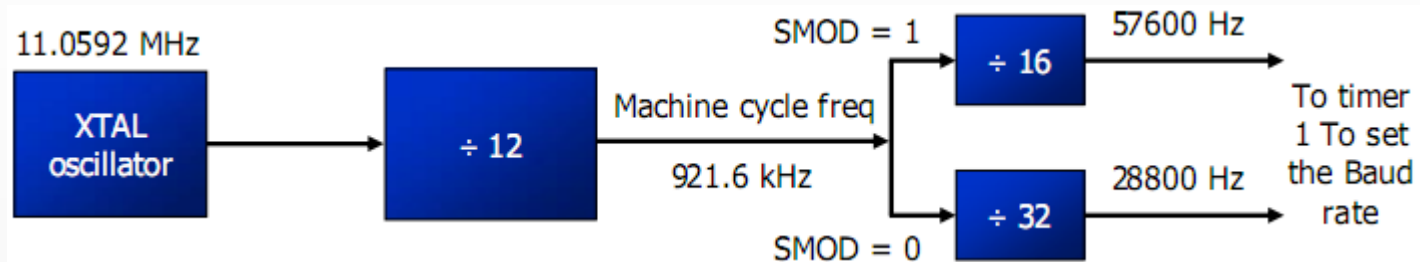
- ◆ PCON register is an 8-bit register

- When 8051 is powered up, SMOD is zero
- We can set it to high by software and thereby double the baud rate

SMOD	--	--	--	GF1	GF0	PD	IDL
------	----	----	----	-----	-----	----	-----

It is not a bit-addressable register

```
MOV A, PCON    ;place a copy of PCON in ACC
SETB ACC.7     ;make D7=1
MOV PCON, A    ;changing any other bits
```



## Baud Rate comparison for SMOD=0 and SMOD=1

TH1	(Decimal)	(Hex)	SMOD=0	SMOD=1
-3		FD	9600	19200
-6		FA	4800	9600
-12		F4	2400	4800
-24		E8	1200	2400



## Example 11-2

Assume that XTAL = 11.0592 MHz for the following program, state

(a) what this program does, (b) compute the frequency used by timer 1 to set the baud rate, and (c) find the baud rate of the data transfer.

```
MOV A,PCON      ;A=PCON
MOV ACC.7       ;make D7=1
MOV PCON,A      ;SMOD=1, double baud rate with same XTAL freq.
MOV TMOD,#20H   ;timer 1, mode 2
MOV TH1,-3      ;19200 (57600/3 =19200)
MOV SCON,#50H   ;8-bit data, 1 stop bit, RI enabled
SETB TR1        ;start timer 1
MOV A,#"B"      ;transfer letter B
A_1: CLR TI      ;make sure TI=0
      MOV SBUF,A  ;transfer it
H_1:  JNB TI,H_1  ;stay here until the last bit is gone
      SJMP A_1    ;keep sending "B" again
```

## Solution:

(a) This program transfers ASCII letter B (01000010 binary) continuously

(b) With XTAL = 11.0592 MHz and SMOD = 1 in the above program, we have:

$11.0592 / 12 = 921.6$  kHz machine cycle frequency.

$921.6 / 16 = 57,600$  Hz frequency used by timer 1 to set the baud rate.

$57600 / 3 = 19,200$ , the baud rate.

Find the TH1 value (in both decimal and hex ) to set the baud rate to each of the following. (a) 9600 (b) 4800 if SMOD=1. Assume that XTAL 11.0592 MHz

**Solution:**

With XTAL = 11.0592 and SMOD = 1, we have timer frequency = 57,600 Hz.

(a)  $57600 / 9600 = 6$ ; so TH1 = -6 or TH1 = FAH

(b)  $57600 / 4800 = 12$ ; so TH1 = -12 or TH1 = F4H

**Example 11-3**

Find the baud rate if TH1 = -2, SMOD = 1, and XTAL = 11.0592MHz. Is this baud rate supported by IBM compatible PCs?

**Solution:**

With XTAL = 11.0592 and SMOD = 1, we have timer frequency = 57,600 Hz. The baud rate is  $57,600/2 = 28,800$ . This baud rate is not supported by the BIOS of the PCs; however, the PC can be programmed to do data transfer at such a speed. Also, HyperTerminal in Windows supports this and other baud rates.

## Example 11-4

Write a program to send the message “The Earth is but One Country” to serial port. Assume a SW is connected to pin P1.2. Monitor its status and set the baud rate as follows:

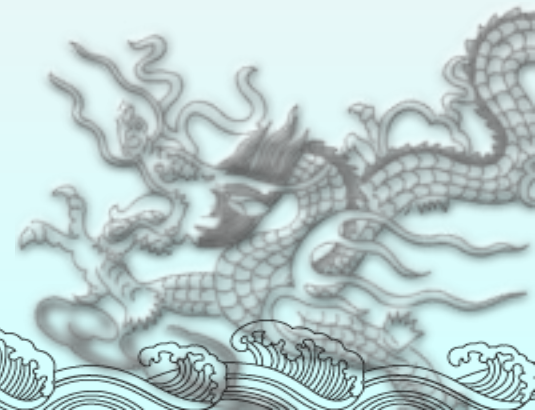
SW = 0, 4800 baud rate

SW = 1, 9600 baud rate

Assume XTAL = 11.0592 MHz, 8-bit data, and 1 stop bit.

### Solution:

```
        SW  BIT P1.2
        ORG 0H           ;starting position
MAIN:
        MOV TMOD,#20H
        MOV TH1,#-6      ;4800 baud rate (default)
        MOV SCON,#50H
        SETB TR1
        SETB SW           ;make SW an input
S1:     JNB SW,SLOWSP      ;check SW status
        MOV A,PCON        ;read PCON
        SETB ACC.7        ;set SMOD high for 9600
        MOV PCON,A        ;write PCON
        SJMP OVER         ;send message
```



SLOWSP:

```
MOV A,PCON    ;read PCON
SETB ACC.7    ;set SMOD low for 4800
MOV PCON,A    ;write PCON
```

OVER: MOV DPTR,#MESS1 ;load address to message

```
FN: CLR A
     MOVC A,@A+DPTR ;read value
     JZ  S1         ;check for end of line
     ACALL SENDCOM ;send value to serial port
     INC DPTR       ;move to next value
     SJMP FN        ;repeat
```

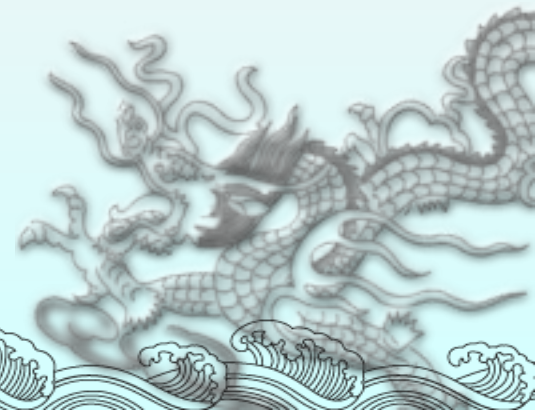
;-----

SENDCOM:

```
MOV SBUF,A    ;place value in buffer
HERE: JNB TI,HERE ;wait until transmitted
      CLR TI    ;clear
      RET      ;return
```

;-----

```
MESS1: DB "The Earth is but One Country",0
      END
```





# § 11-4 Serial Port Programming in C

## Transmitting and Receiving Data

### Example 11-7

Write a C program for 8051 to transfer the letter “A” serially at 4800 baud continuously. Use 8-bit data and 1 stop bit.

### Solution:

```
#include <reg51.h>
void main(void){
    TMOD=0x20; //use Timer 1, mode 2
    TH1=0xFA; //4800 baud rate
    SCON=0x50;
    TR1=1;
    while (1) {
        SBUF='A'; //place value in buffer
        while (TI==0);
        TI=0;
    }
}
```

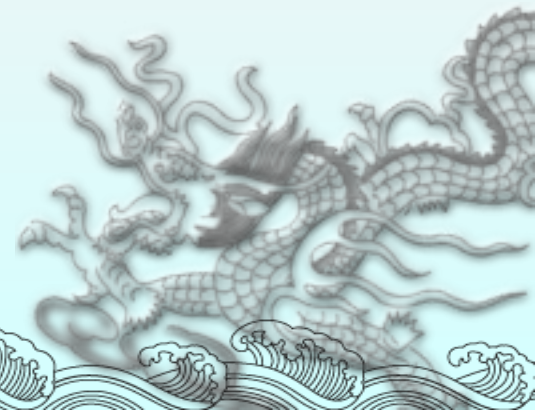
### Example 11-8

Write an 8051 C program to transfer the message “YES” serially at 9600 baud, 8-bit data, 1 stop bit. Do this continuously.

#### Solution:

```
#include <reg51.h>
void SerTx(unsigned char);
void main(void){
    TMOD=0x20; //use Timer 1, mode 2
    TH1=0xFD; //9600 baud rate
    SCON=0x50;
    TR1=1; //start timer
    while (1) {
        SerTx('Y');
        SerTx('E');
        SerTx('S');
    }
}

void SerTx(unsigned char x){
    SBUF=x;      //place value in buffer
    while (TI==0); //wait until transmitted
    TI=0;
}
```

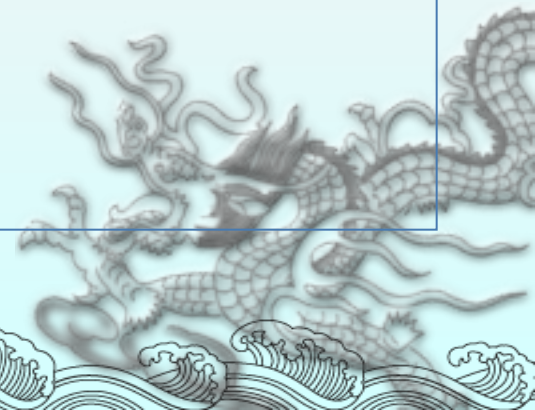


## Example 11-9

Program the 8051 in C to receive bytes of data serially and put them in P1. Set the baud rate at 4800, 8-bit data, and 1 stop bit.

### Solution:

```
#include <reg51.h>
void main(void){
    unsigned char mybyte;
    TMOD=0x20; //use Timer 1, mode 2
    TH1=0xFA; //4800 baud rate
    SCON=0x50;
    TR1=1; //start timer
    while (1) { //repeat forever
        while (RI==0); //wait to receive
        mybyte=SBUF; //save value
        P1=mybyte; //write value to port
        RI=0;
    }
}
```



### Example 11-10

Write an 8051 C Program to send the two messages “Normal Speed” and “High Speed” to the serial port. Assuming that SW is connected to pin P2.0, monitor its status and set the baud rate as follows:

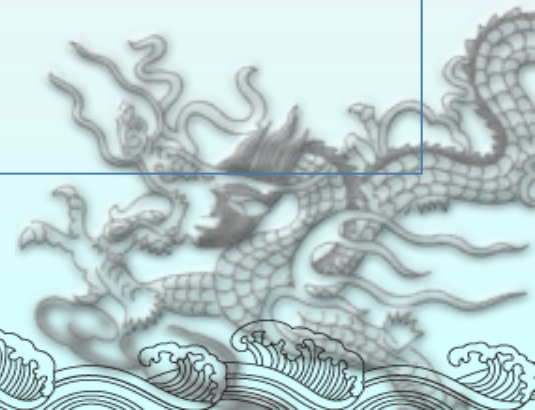
SW = 0, 28,800 baud rate

SW = 1, 56K baud rate

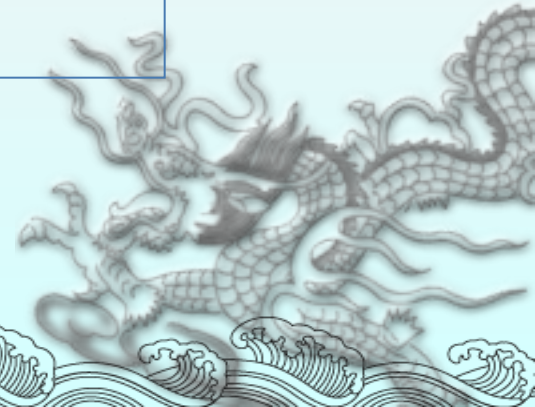
Assume that XTAL = 11.0592 MHz for both cases.

#### **Solution:**

```
#include <reg51.h>
sbit MYSW=P2^0; //input switch
void main(void){
    unsigned char z;
    unsigned char Mess1[]="Normal Speed";
    unsigned char Mess2[]="High Speed";
    TMOD=0x20;    //use Timer 1, mode 2
    TH1=0xFF;     //28800 for normal
    SCON=0x50;
    TR1=1;        //start timer
    .....
```

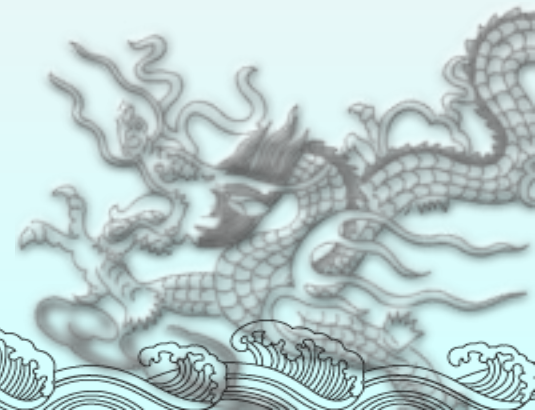


```
if(MYSW==0) {  
    for (z=0;z<12;z++) {  
        SBUF=Mess1[z];    //place value in buffer  
        while(TI==0);    //wait for transmit  
        TI=0;  
    }  
}  
else {  
    PCON=PCON|0x80; //for high speed of 56K  
    for (z=0;z<10;z++) {  
        SBUF=Mess2[z];    //place value in buffer  
        while(TI==0);    //wait for transmit  
        TI=0;  
    }  
}  
}
```



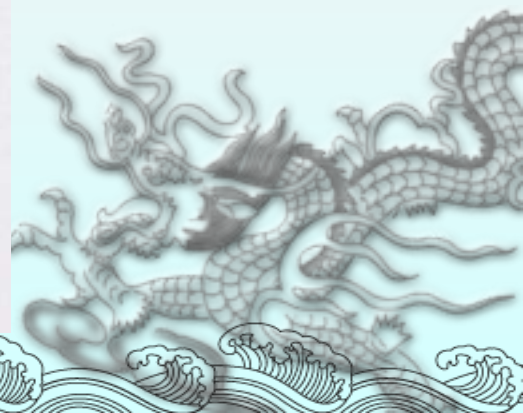
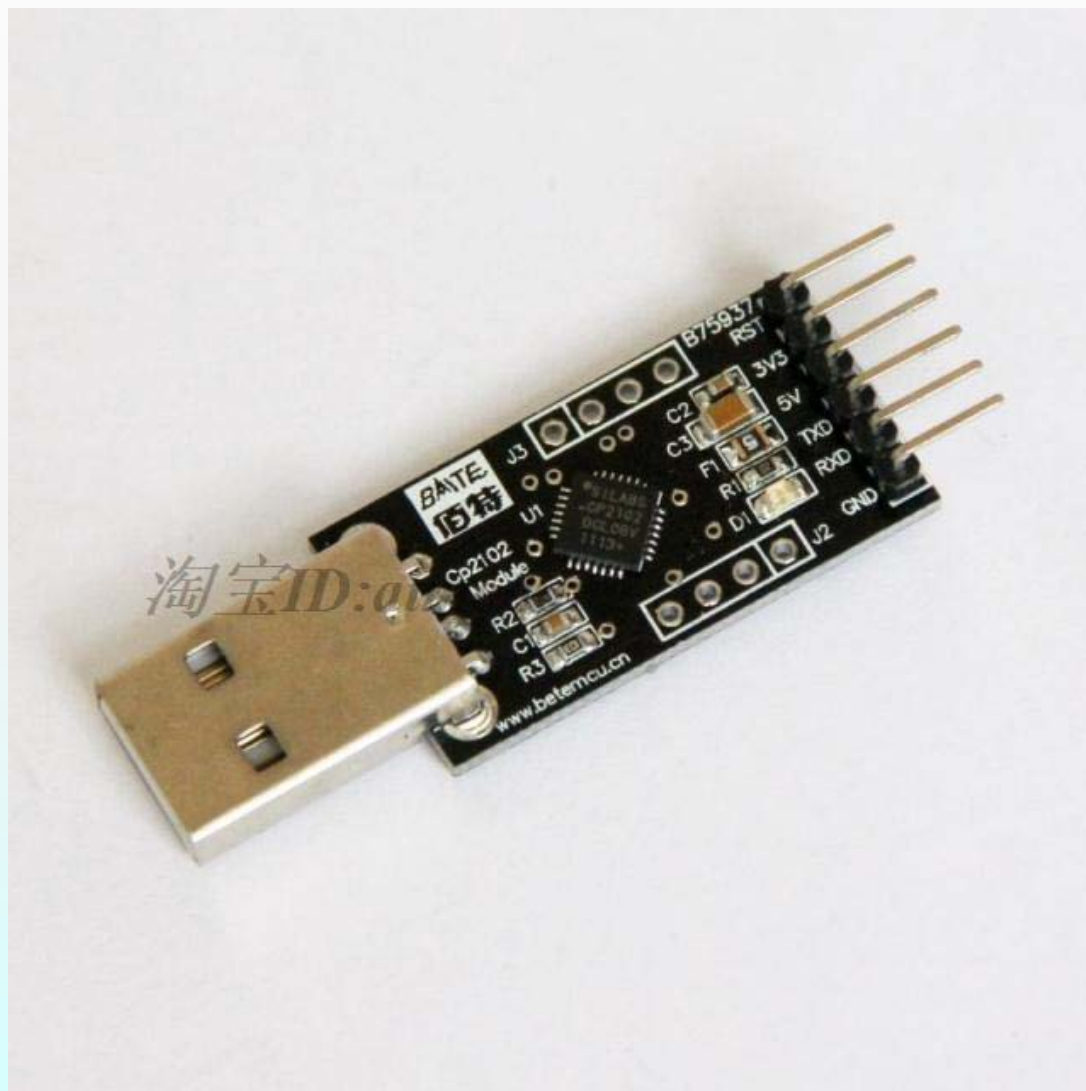
# § 11-5 New communication methods

- ◆ UART
- ◆ SPI
- ◆ I2C
- ◆ USB
- ◆ WLAN
- ◆ Bluetooth
- ◆ WIFI
- ◆ 3G

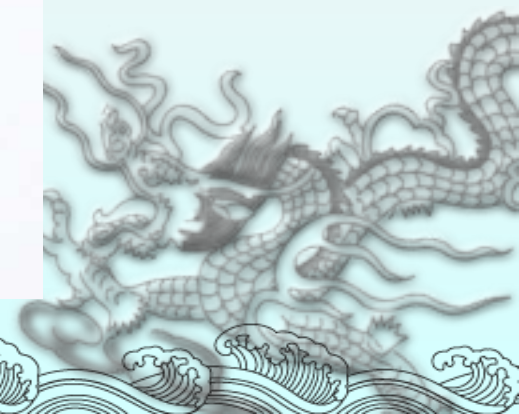




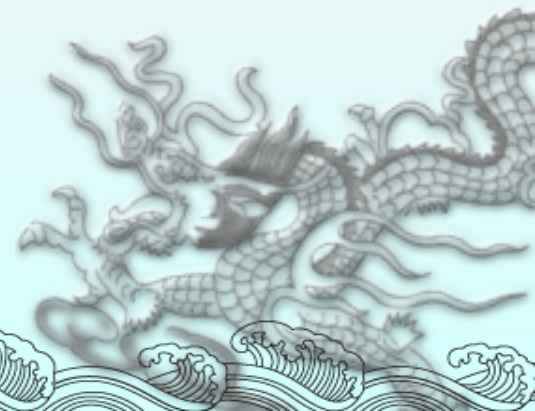
# USB转TTL转换板



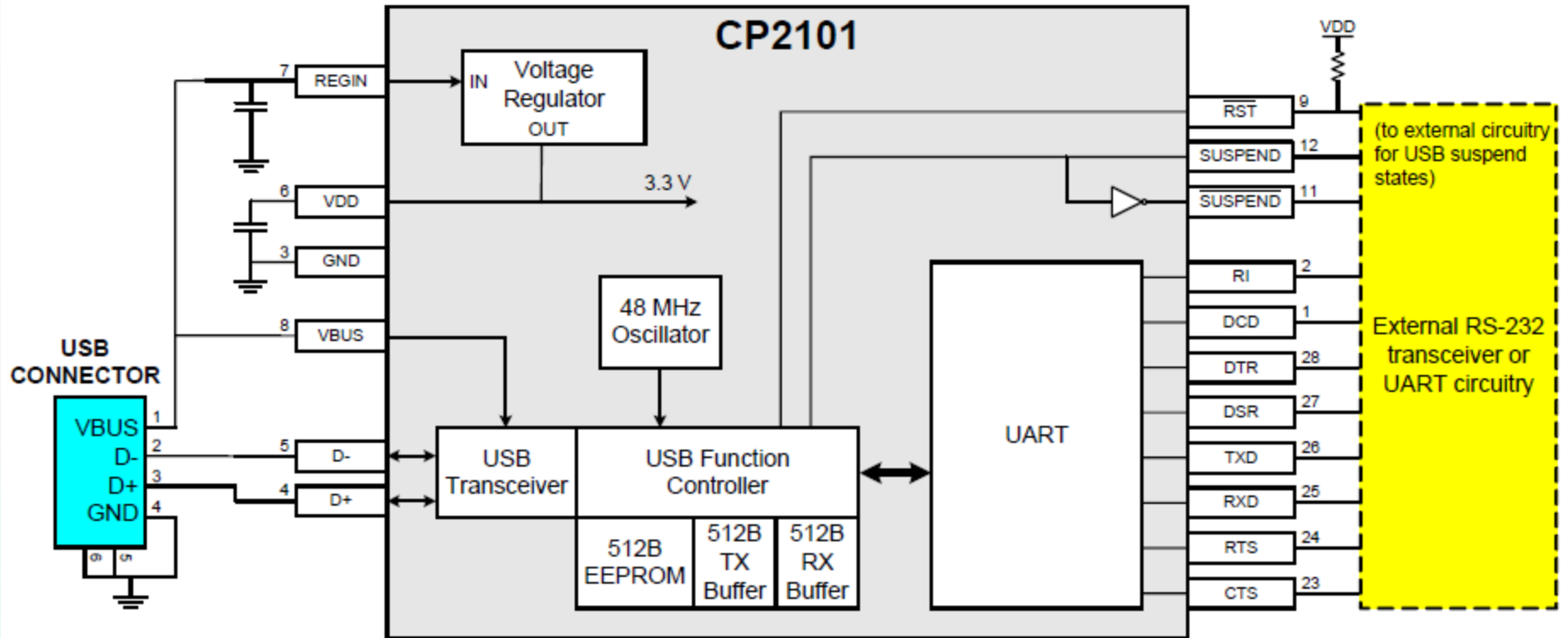
# 蓝牙转TTL转换板



# WIFI转TTL转换板

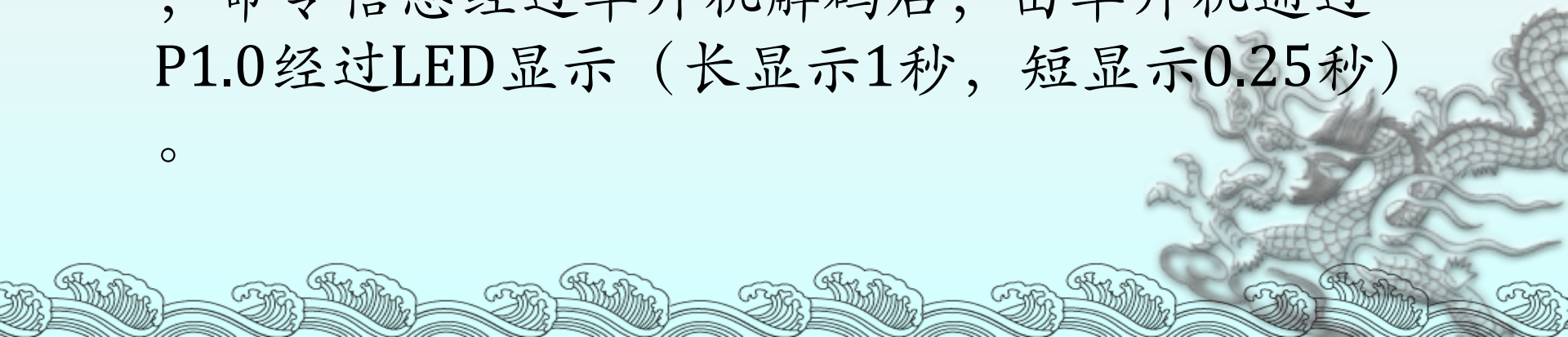


# CP2101 (USB 转 UART 的单芯片桥接器)



# Homework

- ◆ 问题：假设某种信号灯通讯，不同长短交替代表不同的信息。要求在单片机和PC间建立一个信号传输通道（串行通信）模拟这种信号灯通讯。假设8次不同的长短信号组合代表一个讯息（8位），通过长（ $>0.5$ 秒）短（ $<0.5$ 秒）的按键发送命令，按键命令由单片机检测到后通过串口发送给PC机。同时单片机有可能接收到PC机发来的命令，命令信息经过单片机解码后，由单片机通过P1.0经过LED显示（长显示1秒，短显示0.25秒）。

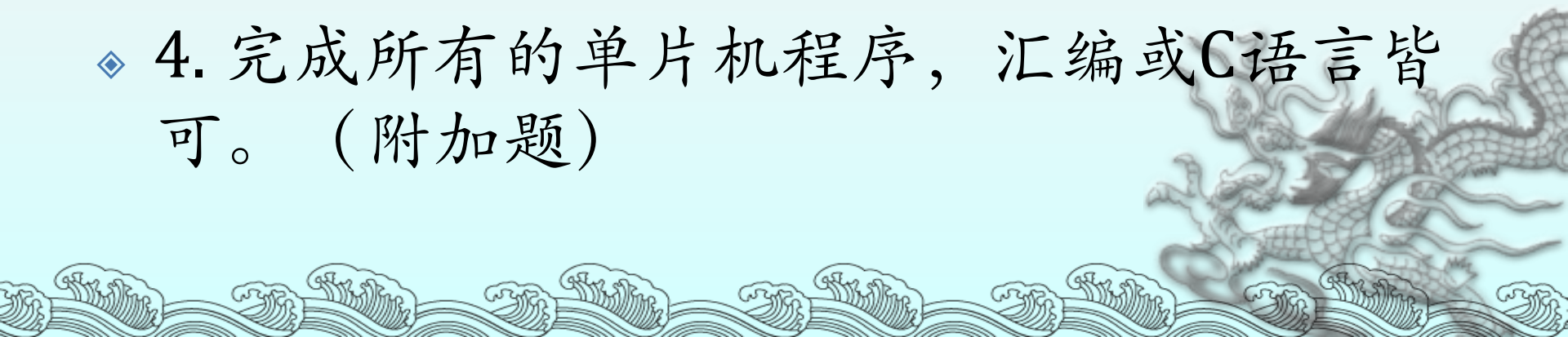




# Homework

## ◆ 要求：

- ◆ 1. 列出单片机需要哪些资源来完成这项工作；
- ◆ 2. 画出单片机完成本任务的流程图
- ◆ 3. 讨论在题目中，有哪些任务细节没有设置清楚，请补充你认为合理的设置；
- ◆ 4. 完成所有的单片机程序，汇编或C语言皆可。（附加题）





THANK YOU!!

