Principle and Interface Techniques of Microcontroller

--8051 Microcontroller and Embedded Systems Using Assembly and C

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

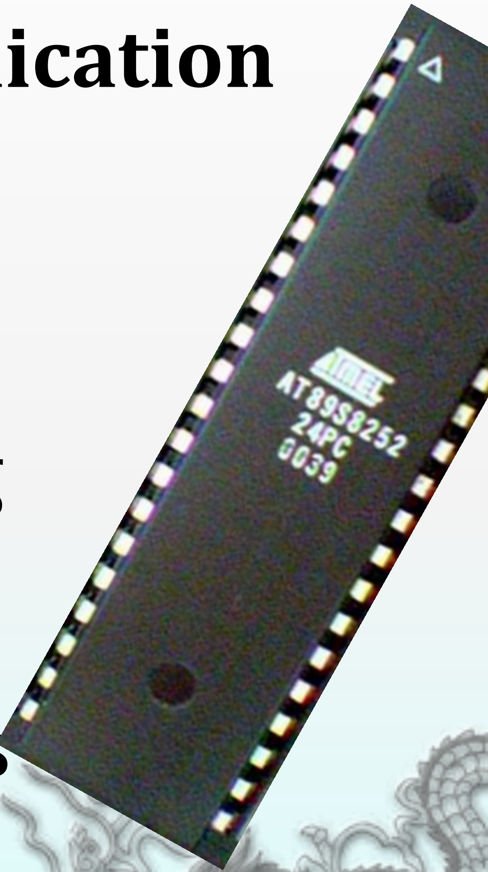
WANG, You (王酉) **PhD, MIET**

杭州 • 浙江大学 • 2022

Chapter 11
Serial Communication

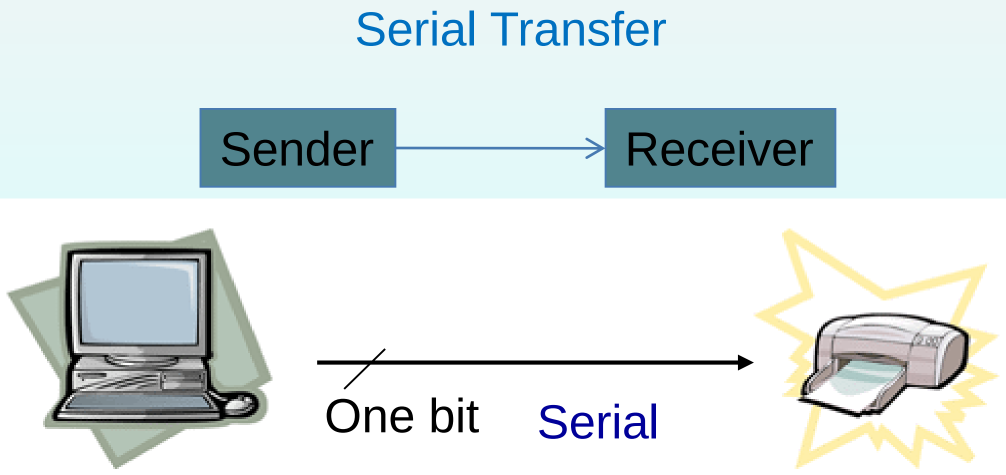
Outline

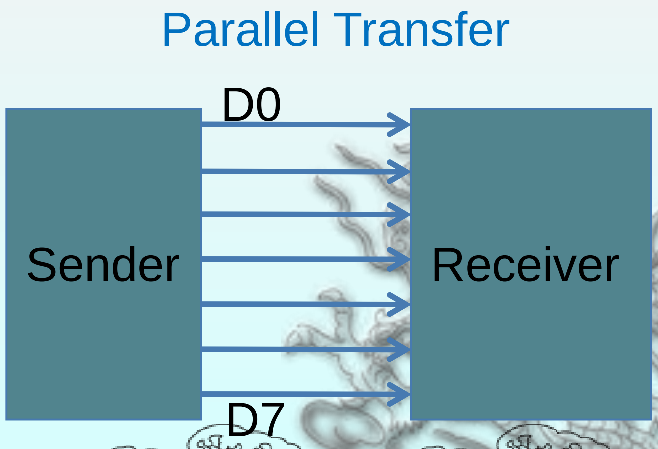
- § 11-1 Basics of Serial Communication
- § 11-2 Serial Communication Programming
- § 11-3 Serial Port Programming in C
- § 11-4 Communication methods



§ 11-1 Basics of Serial Communication

- Computers transfer data in two ways:
  - Parallel
    - ✓ Often 8 or more lines (wire conductors) are used to transfer data to a device that is only a few feet away
  - Serial
    - ✓ To transfer to a device located many meters away, the serial method is used
    - ✓ The data is sent one bit at a time

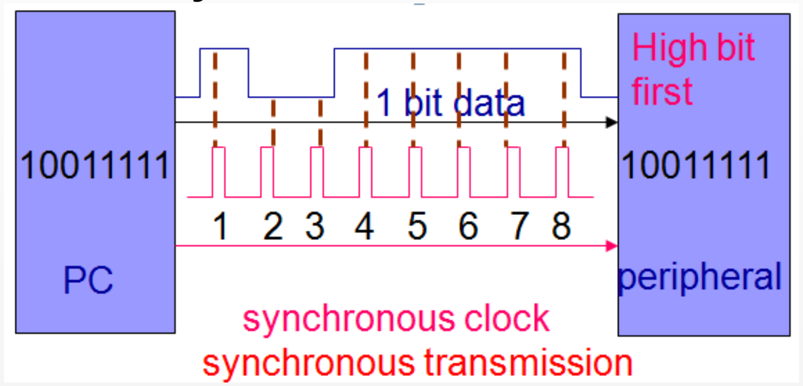


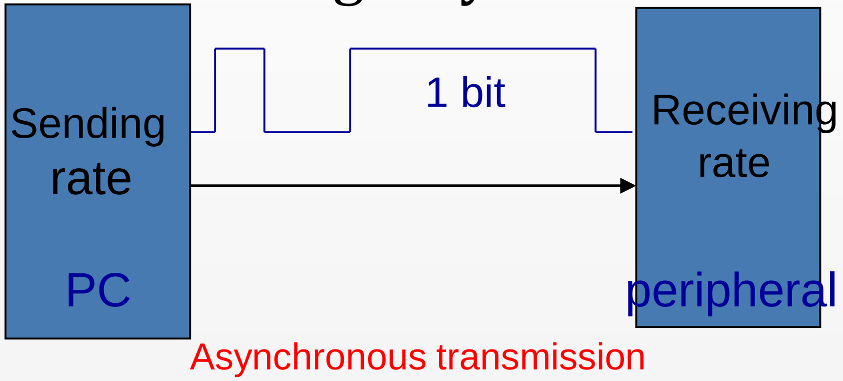


§ 11-1 Basics of Serial Communication

- 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

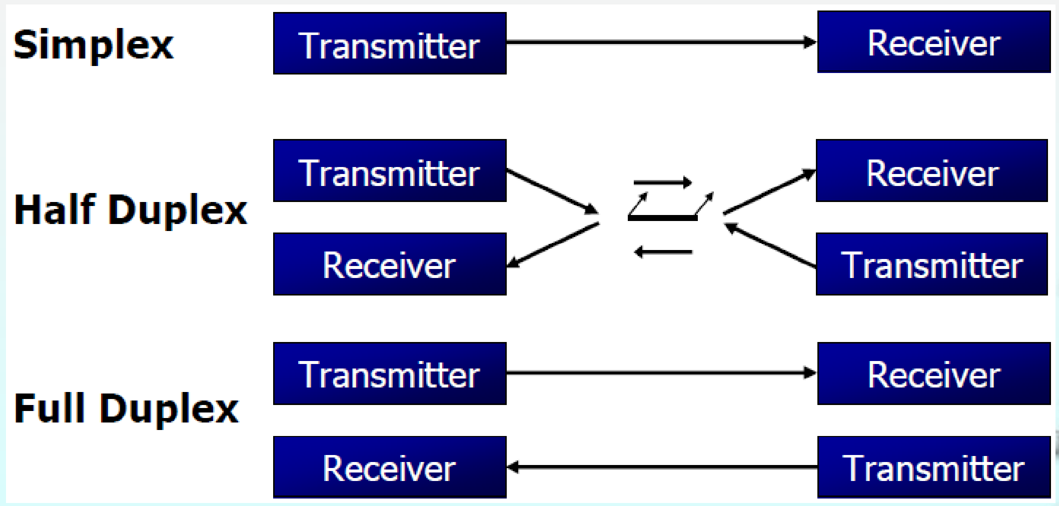




- It is possible to write software to use either of these methods, but the programs can be tedious and long
  - There are special IC chips made by many manufacturers for serial communications
  - UART (universal asynchronous Receiver-transmitter)
  - USART (universal synchronous-asynchronous Receiver-transmitter)

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



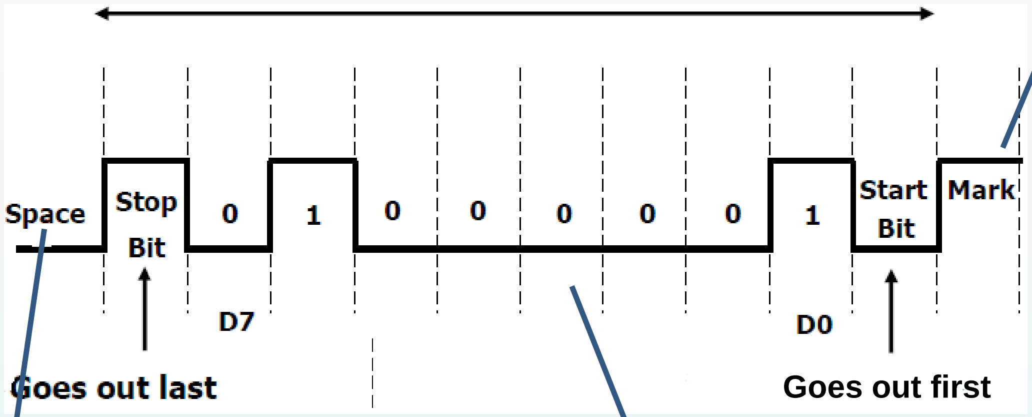
Start and Stop Bits

- 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

- 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

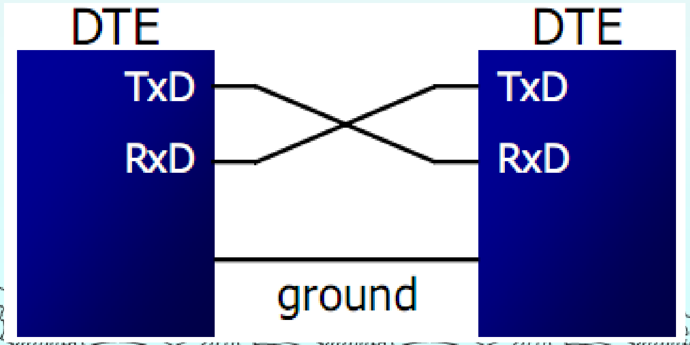
RS232 Standards

- 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 ~ -25 V, while a 0 bit is +3 ~ +25 V, making -3 to +3 undefined

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

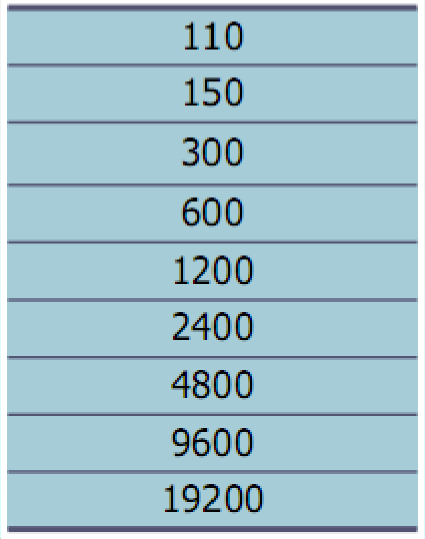
Null modem connection



§ 11-2 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



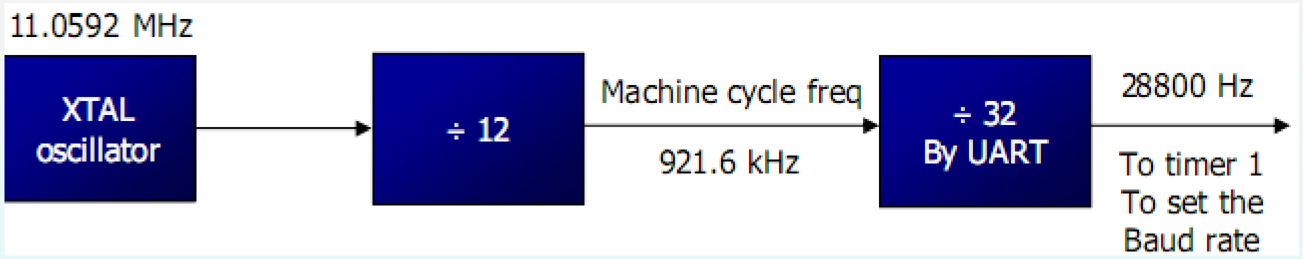
110
150
300
600
1200
2400
4800
9600
19200

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 kHz / 32 = 28,800 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
0	**1**	Serial Mode 1, 8-bit data, 1 stop bit, 1 start bit
1	0	Serial Mode 2, 11 bits, fosc × 2SMOD/64
1	1	Serial Mode 3, 11 bits

- SM2
  - This enables the multiprocessing capability of the 8051

- REN (receive enable)
  - It is a bit-adressable register
    - When it is high, it allows 8051 to receive data on RxD pin
    - If low, the receiver is disable
- 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
```

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

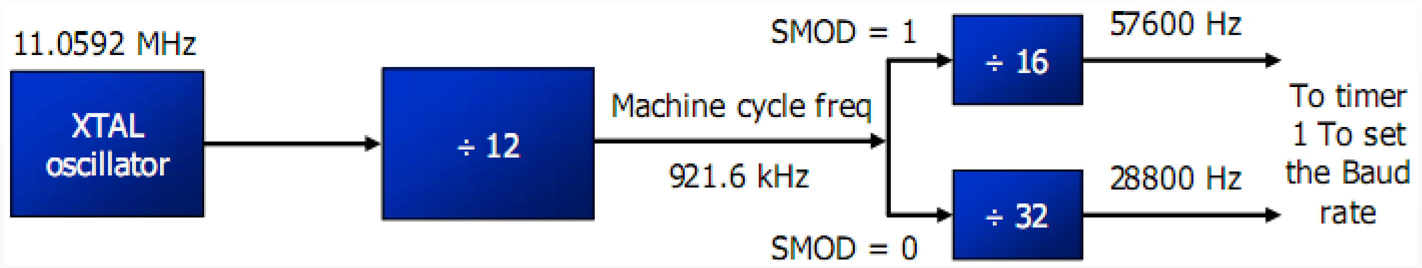
- There are two ways to increase the baud rate of data transfer
  - To use a higher frequency crystal
  - To change a bit in the PCON register
- PCON register is an 8-bit register
  - When 8051 is powered up, SMOD is zero
  - We can set it to high by software and thereby double the baud rate

The system crystal is fixed

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
          SETB  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.

§ 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 串口发送字符串

```
void InitUART (void)
{
 SCON = 0x50; // SCON: 模式 1, 8-bit UART, 使能接收
 TMOD |= 0x20; // TMOD: timer 1, mode 2, 8-bit 重装
 TH1=0xfd;
 TL1=0xfd; //11.0592MHZ晶振 9600波特率 对应应装初值
 TR1 = 1; // TR1: timer 1 打开
 EA = 1; //打开总中断
 ES = 1; //打开串口中断
}
void main (void)
{
 InitUART();
 SendStr("The UART test， 请在发送区输入信息");
 ES = 1; //打开串口中断
 while (1);
}

void SendStr(unsigned char *s)
{
 while(*s!='\0')// \0 表示字符串结束标志，通过检测是否字符串末尾
 {
  SendByte(*s);
  s++;
 }
```

```
void SendByte(unsigned char dat)
{
 SBUF = dat;
 while(!TI);
 TI = 0;
}
```

查询方式

```
void UART_SER (void) interrupt 4 //串行中断服务程序
{
  unsigned char Temp; //定义临时变量
  if(RI) //判断是接收中断产生
 {
    RI=0; //标志位清零
    Temp=SBUF; //读入缓冲区的值
    P1=Temp; //把值输出到P1口，用于观察
 }
  if(TI) //如果是发送标志位，清零
 {
     TI=0;
     if (*s!='\0’)
     {
         SBUF=*s;
          s++;
     }
}
```

中断方式

§ 11-5 Communication methods

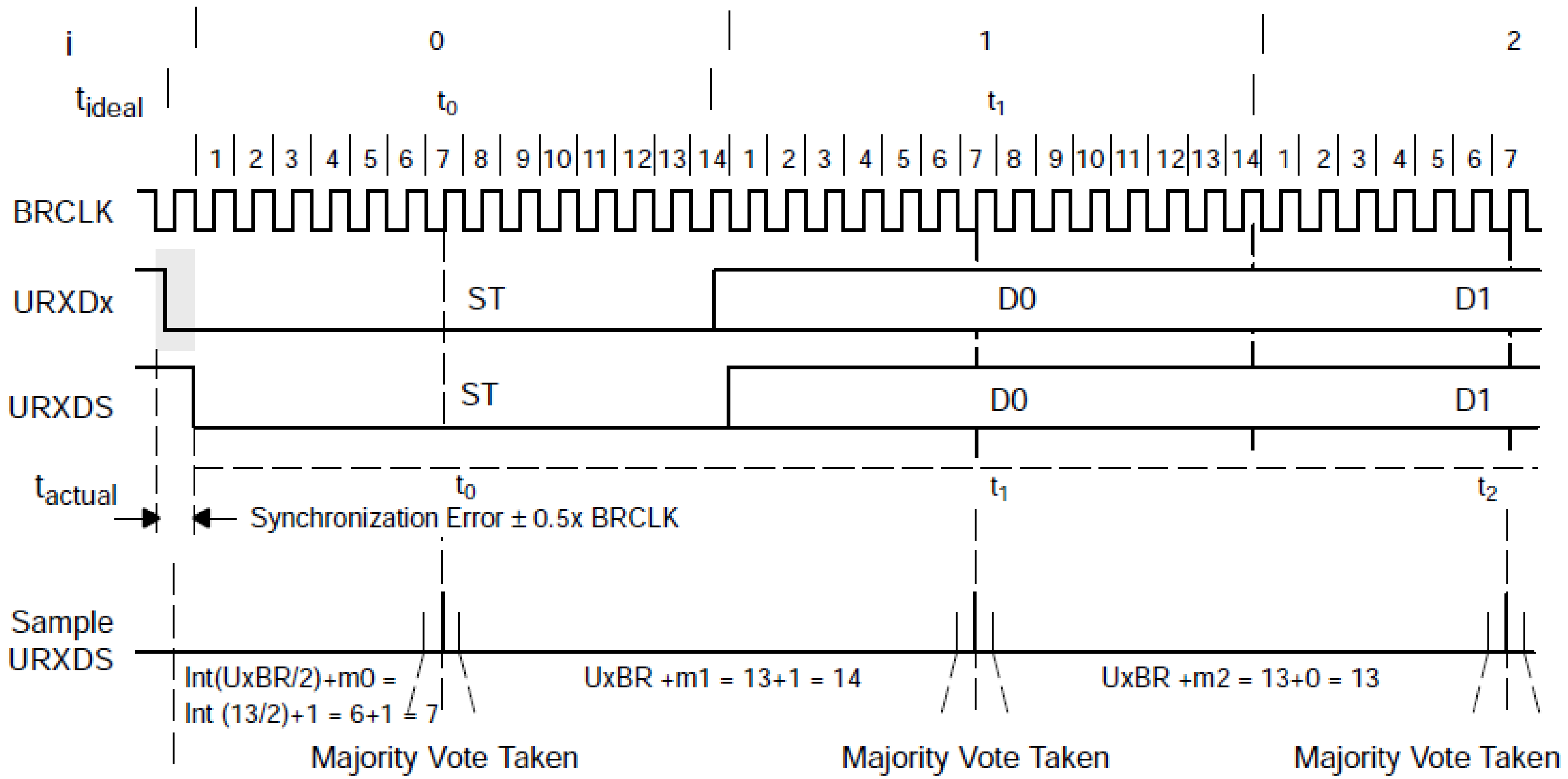
- UART
- SPI
- I2C
- USB
- WLAN
- Bluetooth
- WIFI
- 3G/4G/5G

Universal Asynchronous Receiver/Transmitter （UART）

- 7- or 8-bit data with odd, even, or non-parity
- Independent transmit and receive shift registers
- Separate transmit and receive buffer registers
- LSB-first data transmit and receive
- Receiver start-edge detection for auto-wake up
- Programmable baud
- Status flags for error detection and suppression and address detection
- Independent interrupt capability for receive and transmit

Receive Bit Timing

Figure 14–9. Receive Error

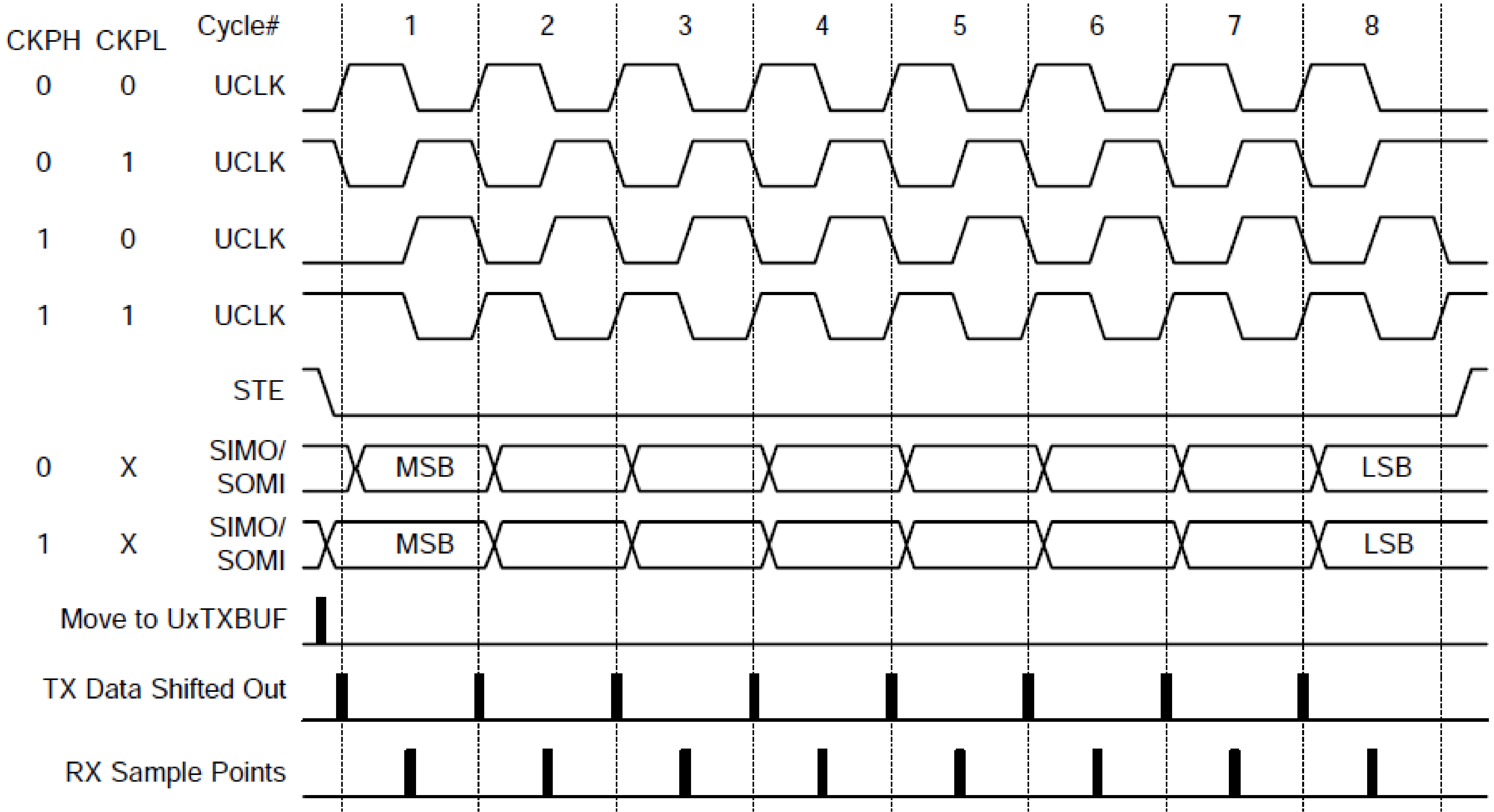


Serial Peripheral Interface

- 7- or 8-bit data length
- 3-pin and 4-pin SPI operation
- Master or slave modes
- Independent transmit and receive shift registers
- Separate transmit and receive buffer registers
- Selectable UCLK polarity and phase control
- Programmable UCLK frequency in master mode
- Independent interrupt capability for receive and transmit

SPI Timing

Figure 15−9. USART SPI Timing



I^2C（Inter – Integrated Circuit）

- 同步通信的一种特殊形式，具有接口线少，控制方式简单，器件封装形式小，通信速率较高等优点

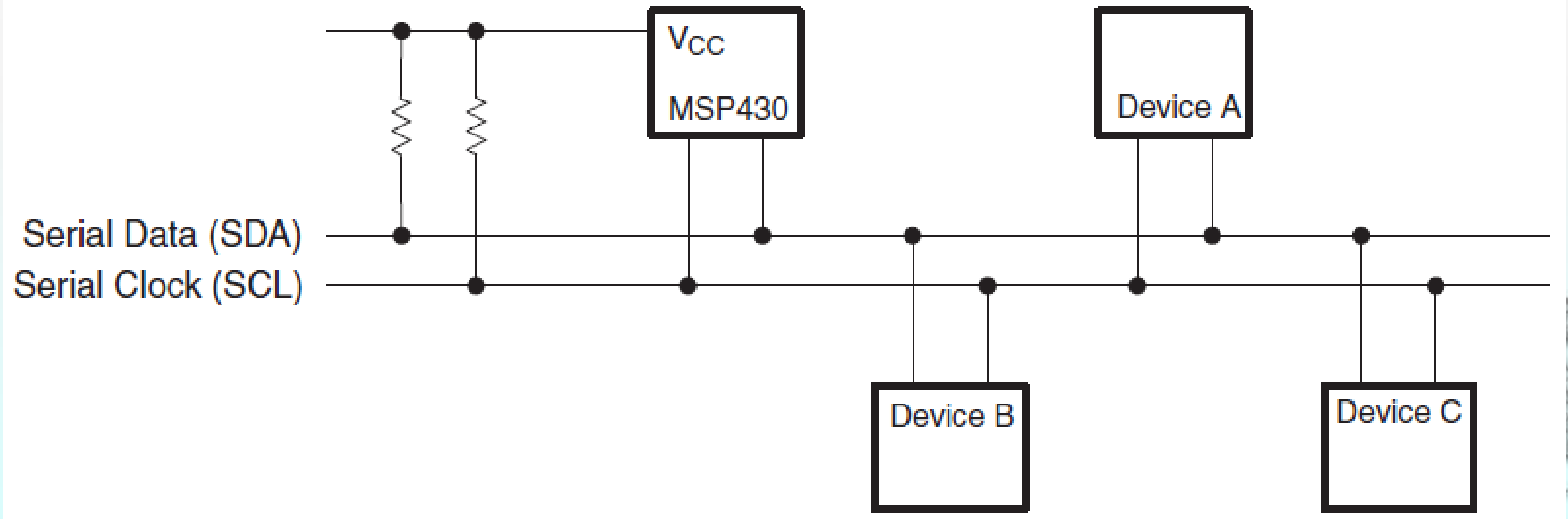


Figure 17-2. I^2C Bus Connection Diagram

I²C Serial Data

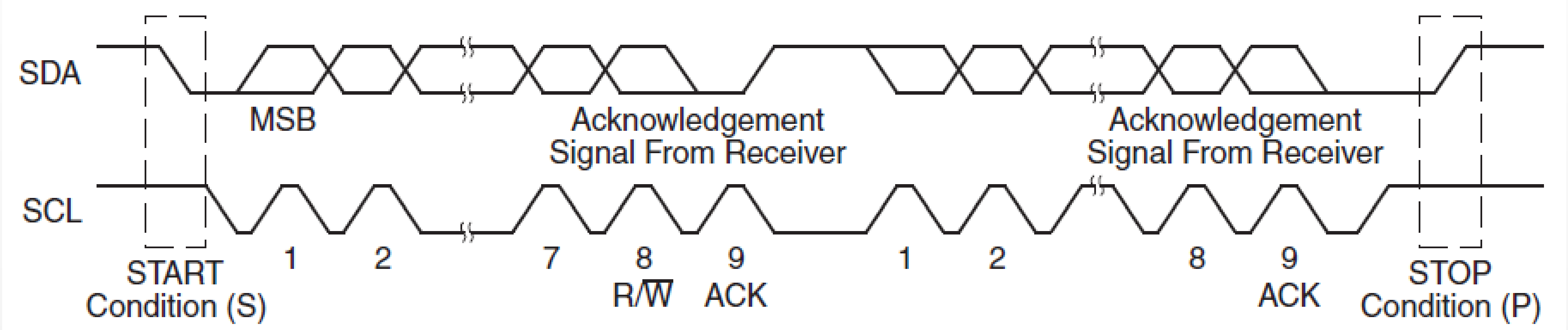


Figure 17-3. I²C Module Data Transfer

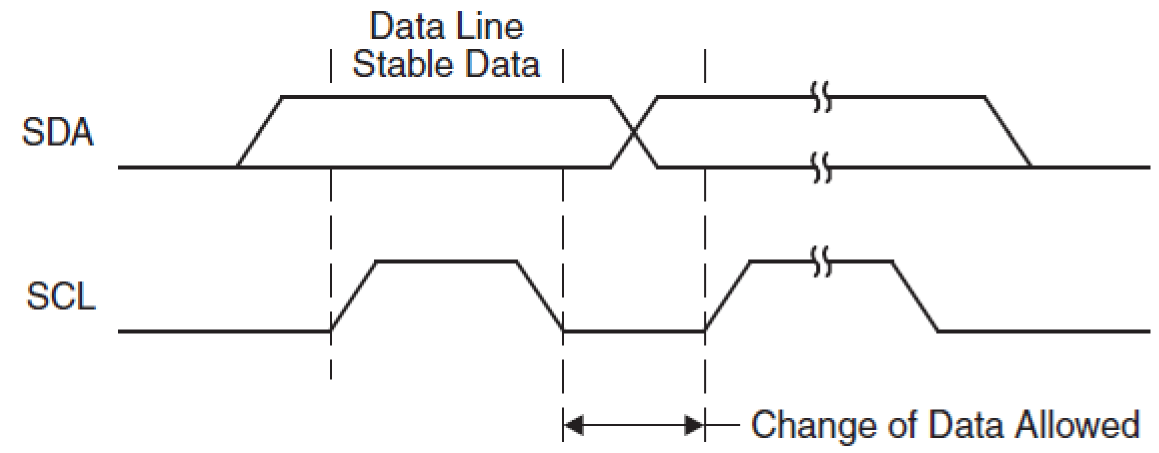


Figure 17-4. Bit Transfer on the I²C Bus

I^2C Addressing Modes

the first byte is the 7-bit slave address and the R/W bit. The ACK bit is sent from the receiver after each byte.

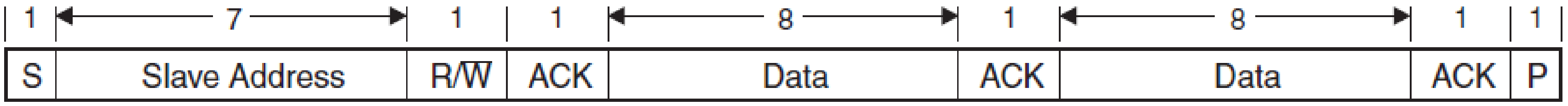


Figure 17-5. I²C Module 7-Bit Addressing Format

Repeated Start Conditions

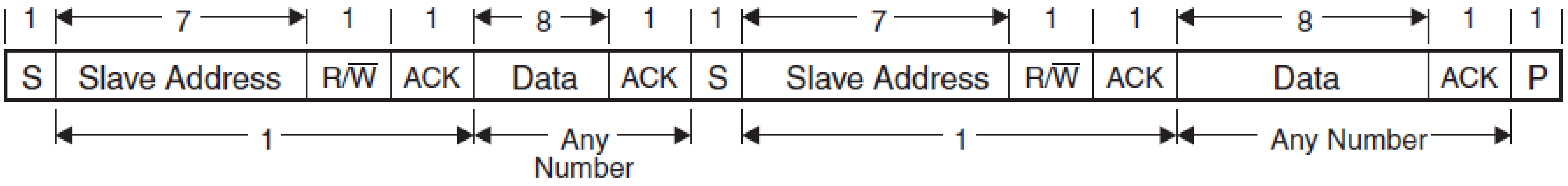
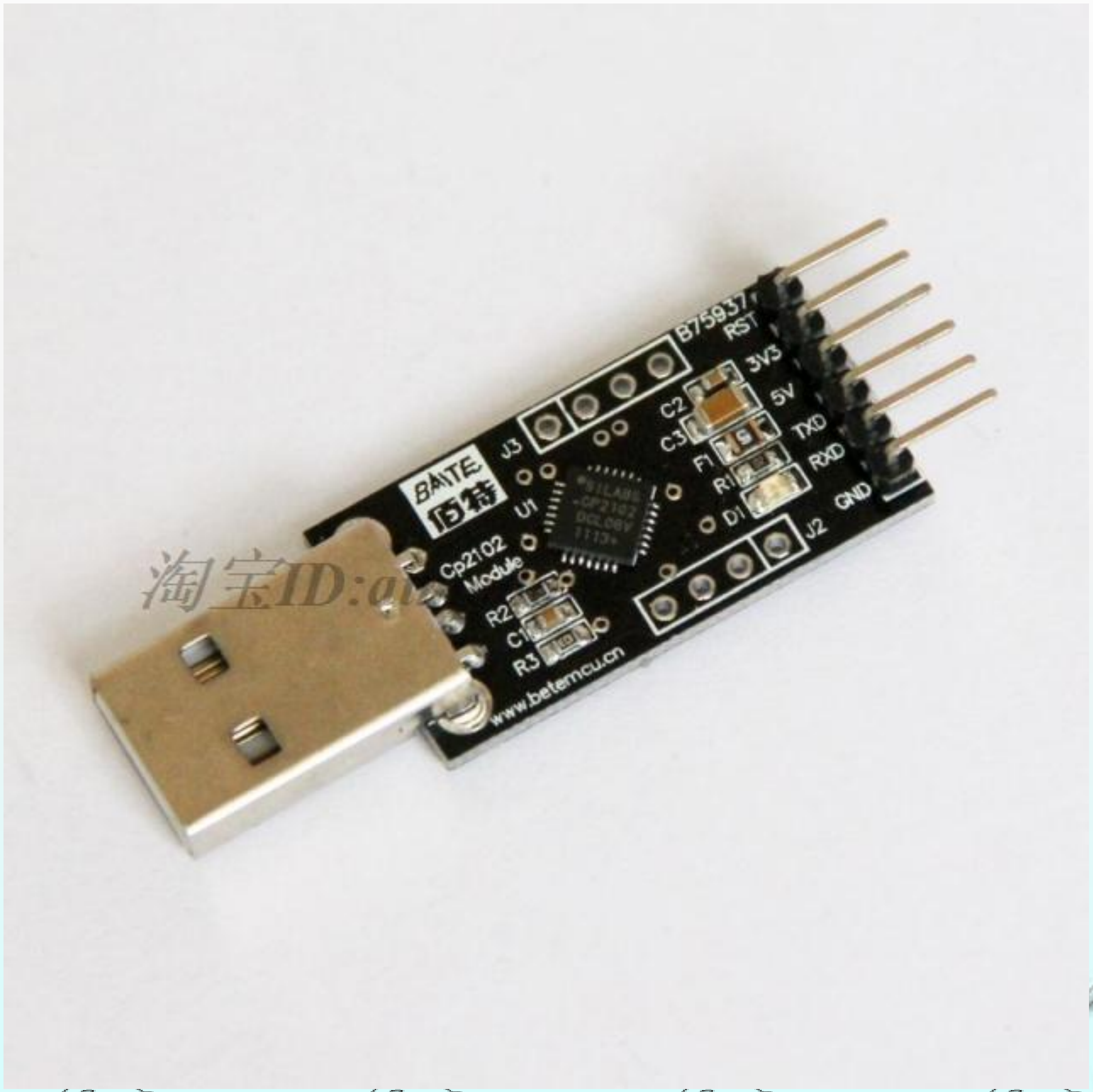
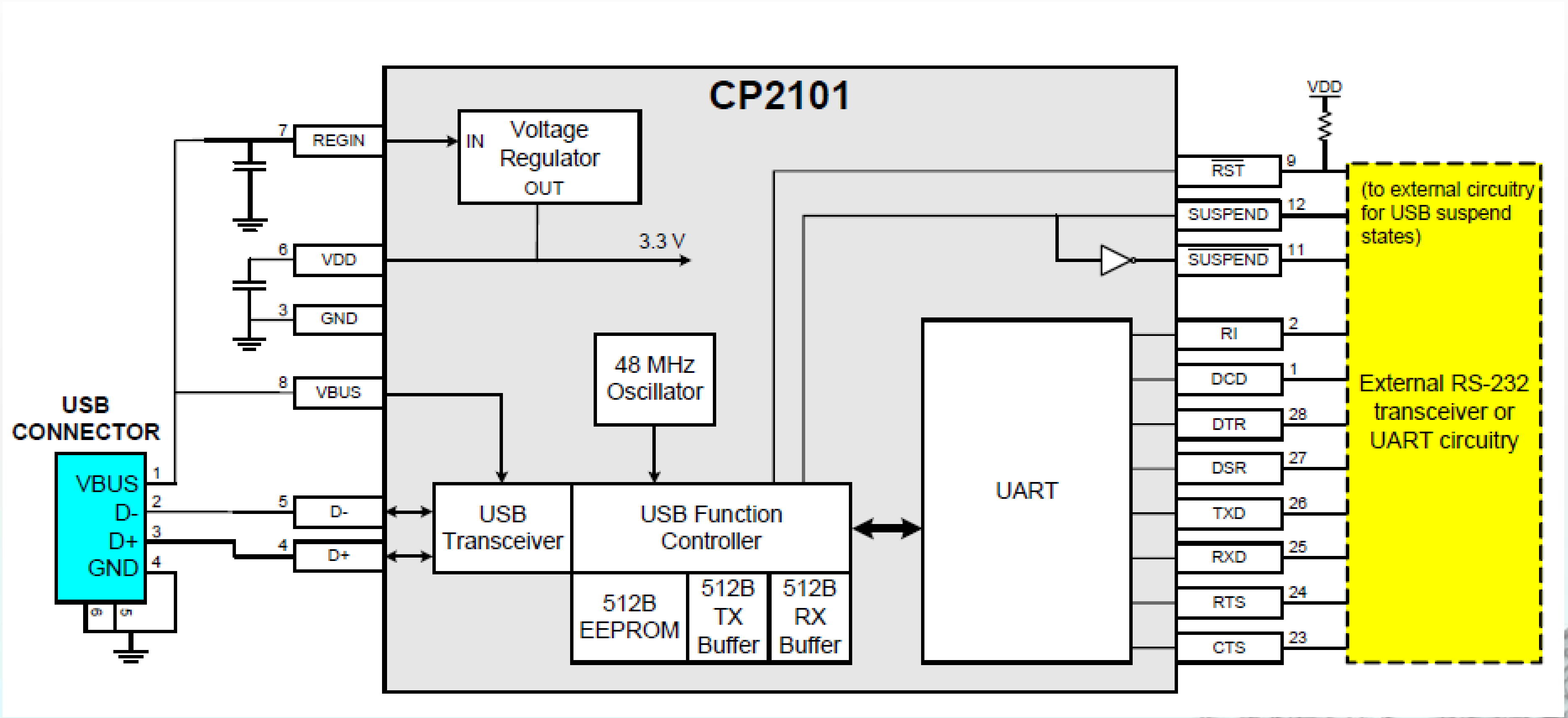


Figure 17-7. I²C Module Addressing Format with Repeated START Condition

USB转TTL转换板

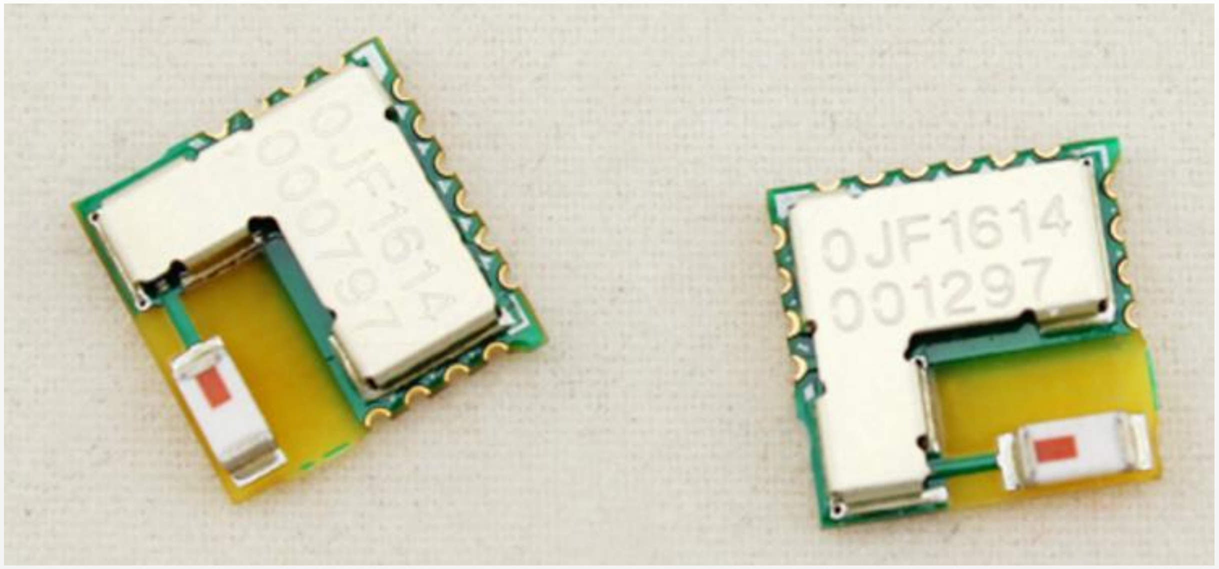

B75937
RST
3V3
5V
TXD
RXD
GND
J3
J2
C2
C3
F1
R1
D1
U1
BAITE 佰特
Cp2102 Module
R2
C1
R3
www.betemcu.cn
SILABS CP2102
淘宝ID:

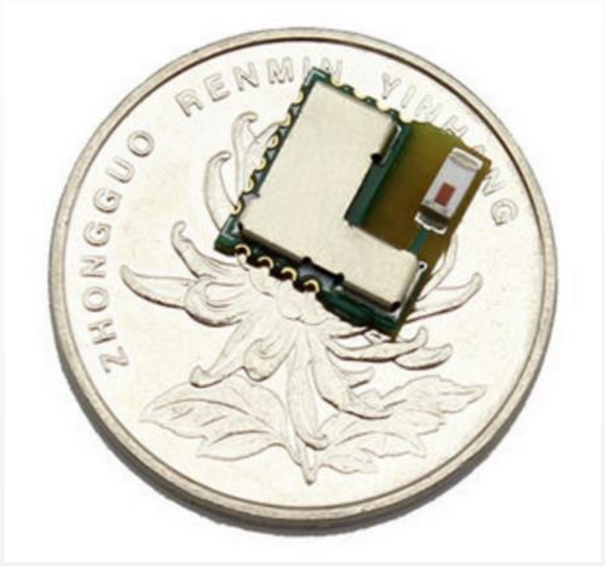
CP2101（USB 转 UART 的单芯片桥接器 ）

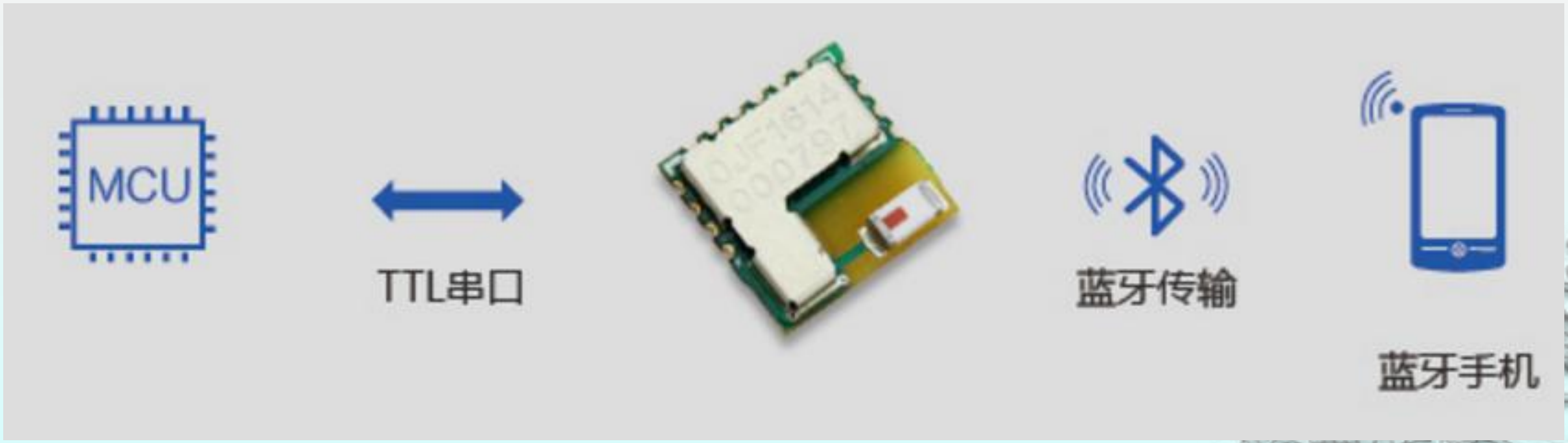


蓝牙转TTL转换板

钦源盛数码专营店
QINYUANSHENG
支持蓝牙 Class1和Class2模块
输入电压3V 内置PCB天线
工业级设计 标准BLE协议

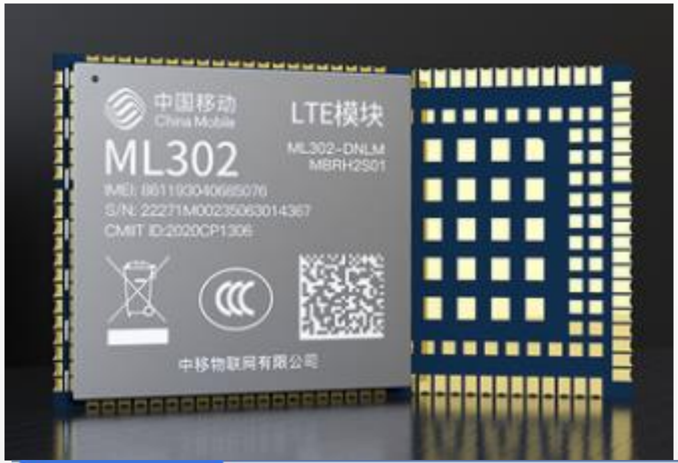

0JF1614
000797
0JF1614
001297

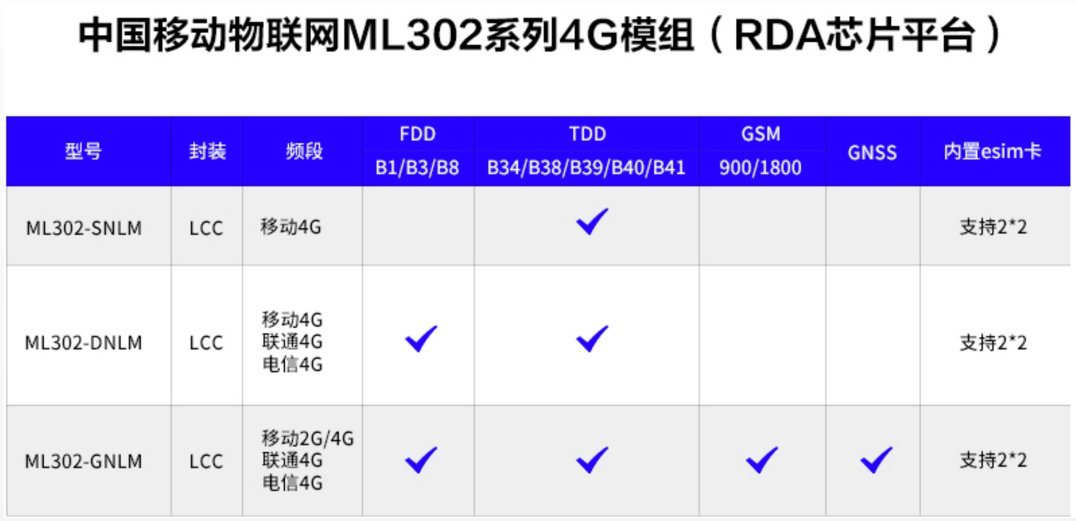

ZHONGGUO RENMIN YINHANG


MCU
TTL串口
蓝牙传输
蓝牙手机

WIFI转TTL转换板

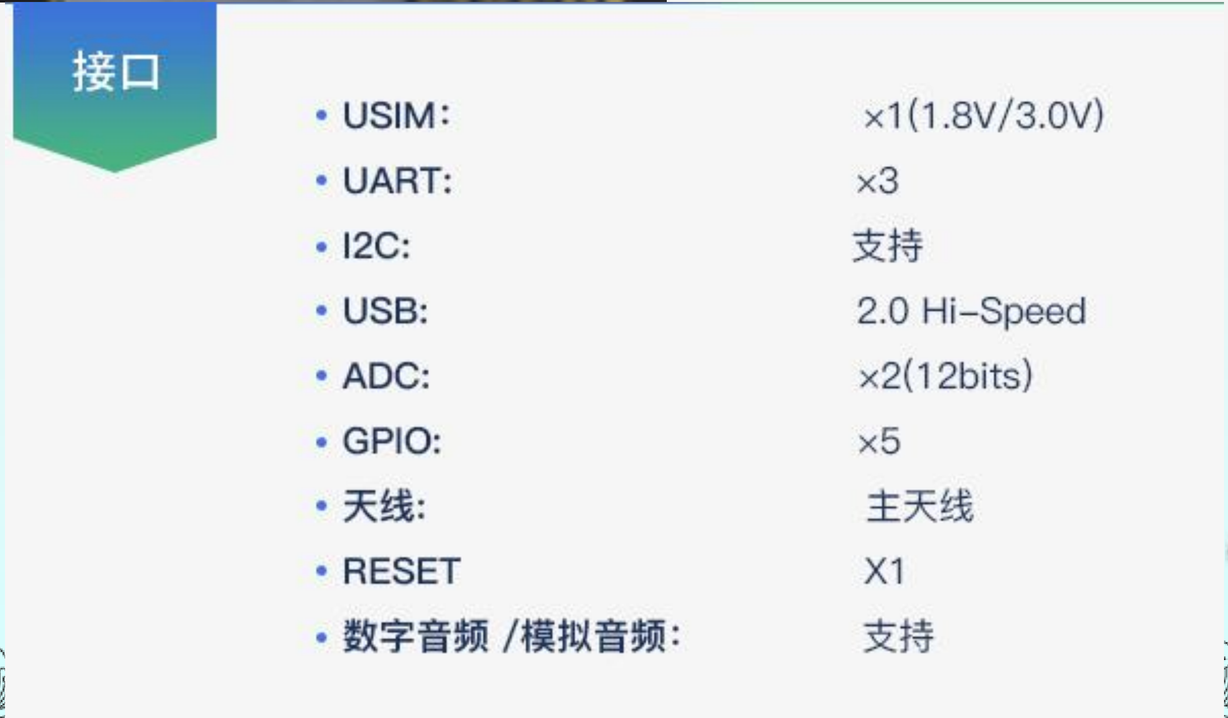
4G模块





中国移动物联网ML302系列4G模组（RDA芯片平台）

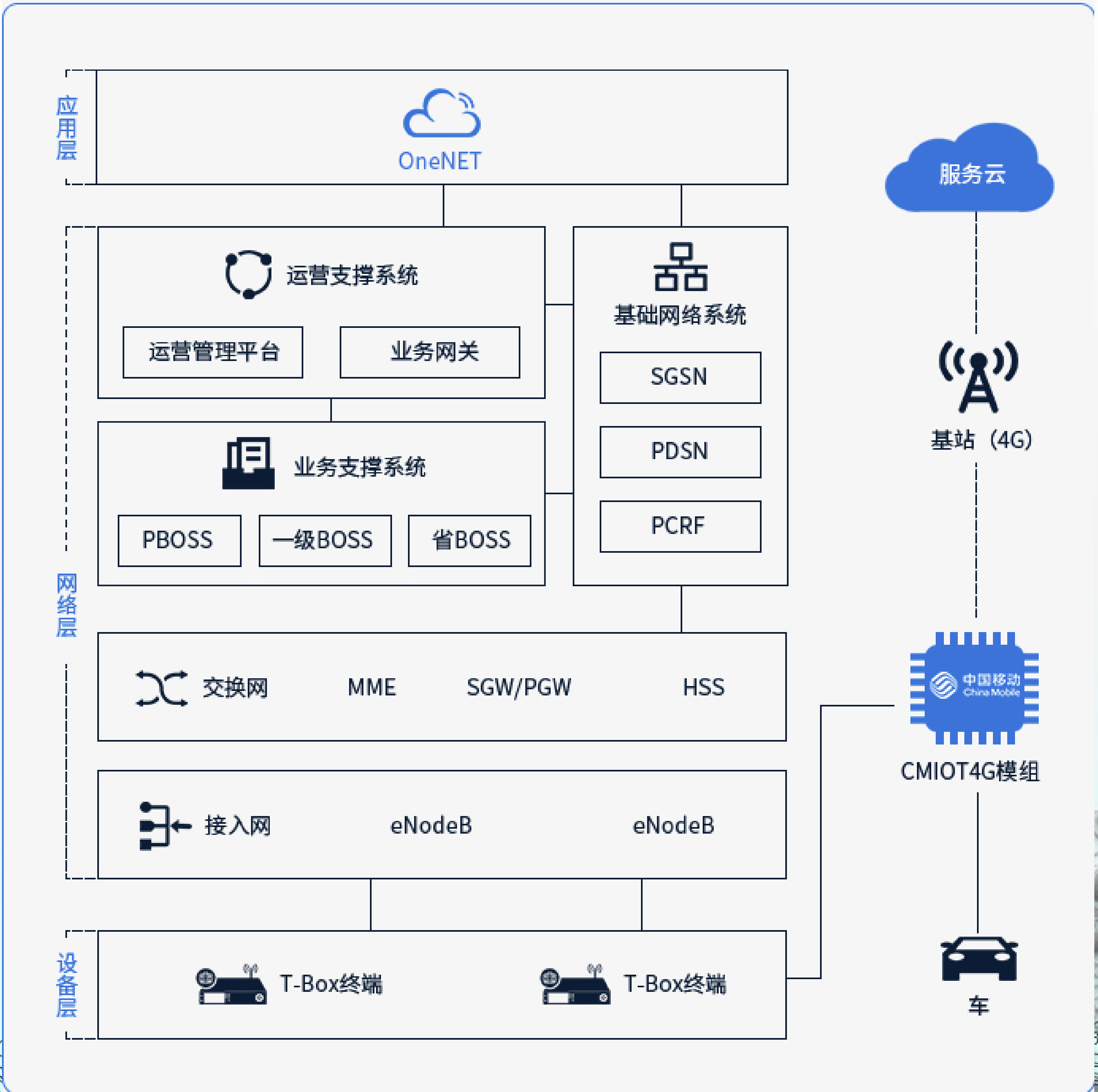
型号	封装	频段	FDD	TDD	GSM	GNSS	内置esim卡
			B1/B3/B8	B34/B38/B39/B40/B41	900/1800		
ML302-SNLM	LCC	移动4G		✓			支持2*2
ML302-DNLM	LCC	移动4G 联通4G 电信4G	✓	✓			支持2*2
ML302-GNLM	LCC	移动2G/4G 联通4G 电信4G	✓	✓	✓	✓	支持2*2



接口

• USIM:	×1(1.8V/3.0V)
• UART:	×3
• I2C:	支持
• USB:	2.0 Hi–Speed
• ADC:	×2(12bits)
• GPIO:	×5
• 天线:	主天线
• RESET	X1
• 数字音频 /模拟音频:	支持

4G模块应用图例


应用层
OneNET
网络层
运营支撑系统
运营管理平台
业务网关
业务支撑系统
PBOSS
一级BOSS
省BOSS
基础网络系统
SGSN
PDSN
PCRF
交换网
MME
SGW/PGW
HSS
接入网
eNodeB
eNodeB
设备层
T-Box终端
T-Box终端
服务云
基站（4G）
中国移动
China Mobile
CMIOT4G模组
车

THANK YOU!!