ELEG3923 Microprocessor
Ch.10 Serial Port Programming

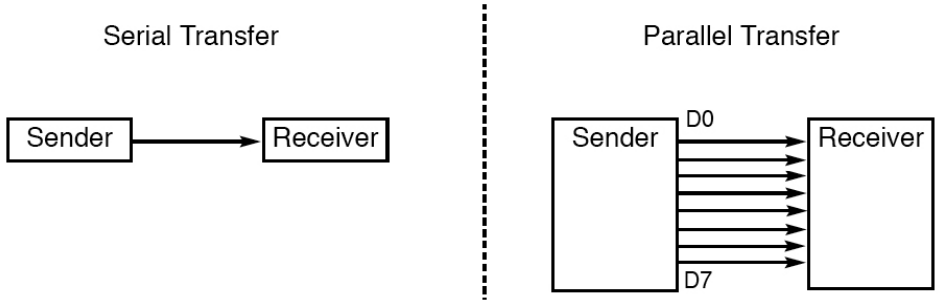
Dr. Jingxian Wu
wuj@uark.edu

OUTLINE

- **Basics of Serial Communication**
- **Serial port programming in Assembly**
- **Programming the second serial port**
- **Serial port programming in C**

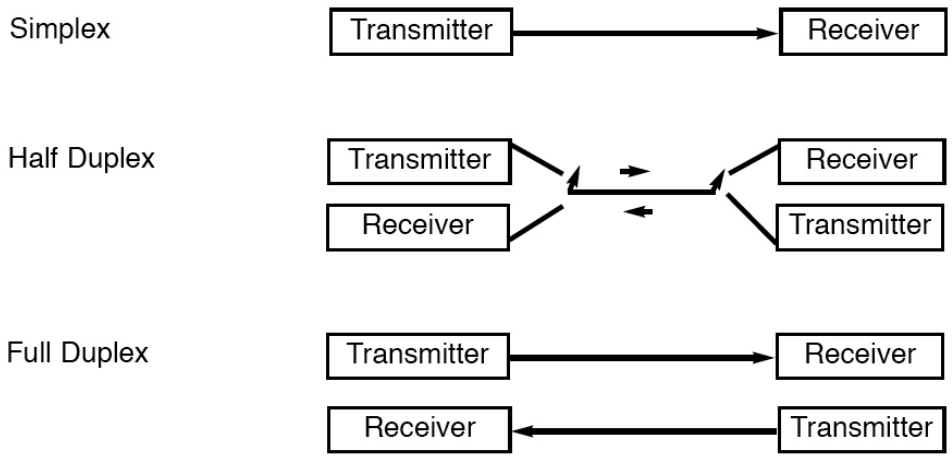
SERIAL COMMUNICATION

- **Serial communication v.s. parallel communication**
  - Computer transfer data in two ways: parallel, serial
  - Parallel communication
    - 8 or more parallel lines are used to transfer data
      - E.g. connecting 8051 to LCD, data bus, connecting to hard drive
    - More data can be transferred in unit period of time
    - Usually used for short distance data transfer
      - Long parallel wires function like antenna, and will leak signal during Tx
      - The leaked signals will cause mutual interference for signals in wire (cross-talk)
  - Serial communication
    - Use 1 data line to transfer data
      - Data is transmitted 1 bit a time
    - Usually used for data transfer over longer distance



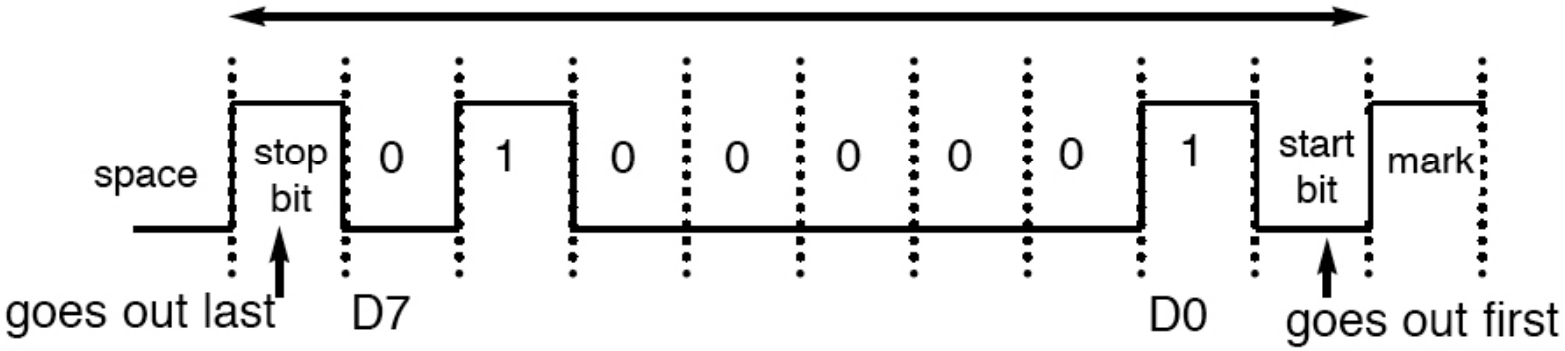
SERIAL COMMUNICATION: DUPLEX

- **Simplex, Half-duplex, and full-duplex**
  - Simplex: communication can occur in only one direction (A → B)
    - E.g. pager, broadcast radio
  - Half-duplex: communication can happen in both directions, but only one at a time (A → B, or B → A, but not simultaneously)
    - E.g. Police radio (walki-talki)
    - Only 1 channel (data line) is enough
  - Full-duplex: communication can happen in both directions simultaneously
    - E.g. telephone
    - Two channels (data lines) are required.
    - Full-duplex = two simplex



SERIAL COMMUNICATION: ASYNCHRONOUS

- **Asynchronous serial communication**
  - Data is transmitted in bursts without following a specific clock
    - Data can be transmitted at any time.
    - Synchronous transmission: data can only be transmitted at special instants
  - When there is no data, channel remains constant to indicate “idle” (no information).
    - e.g. some system use “high” voltage to indicate idle
    - How does the receiver tell the difference between “idle” and “11111111”?
  - Framing
    - Data characters are placed between start and stop bits
    - Start bit: 1 bit (e.g. low)
    - Stop bit: 1 bit (e.g. high), or, 2 bits (e.g. high, low)
    - E.g. 8-bit ASCII + 1 start bit + 1 stop bit = 10 bits/frame



SERIAL COMMUNICATION: TRANSFER RATE

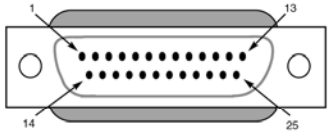
- **Transfer rate**
  - Bit rate
    - Number of bits that can be transferred in unit time (1 second)
    - Unit: bps (bit per second), Kbps (kilo-bit per second), Mbps (mega-bit per second), Gbps (Giga-bit per second), Tbps (Terra-bits per second)
  - Terminology conventions
    - For storage space (RAM size, ROM size, disk size)
      - 1 kilo = 2^10, 1 mega = 1^20, 1 giga = 1^30
    - For data rate
      - 1 kilo = 1,000, 1 mega = 1,000,000, 1 giga = 1,000,000,000
  - Baud rate
    - The number of symbols that can be transferred in unit time (1 second)
    - For some systems, 1 symbol = 1 bit, for some other systems, 1 symbol can be used to represent multiple bits (e.g. 1 symbol = 8 bits)

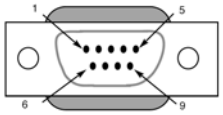
SERIAL COMMUNICATION: STANDARD

- **Communication standard**
  - A set of rules that must be followed by communication devices
    - To ensure that communication devices from different manufactures can interoperate with each other
  - Example rules:
    - Which voltage used to represent ‘0’, which voltage used to represent ‘1’
    - How many start bits, how many stop bits
    - Which voltage(s) used for start bits, which voltage(s) used for stop bits
    - How many bits in one frame (7 bits, 8 bits, 10 bits, …)
    - The format of control signals, how many control pins
  - Example standards
    - RS232, IEEE 802.11 (WiFi), IEEE 802.16 (WiMax), WCDMA, ……
- **RS232**
  - The most popular serial communication standard
  - Developed by Electronics Industries Association (EIA) in the 1960s’
  - Still widely used today

SERIAL COMMUNICATION: RS232

- **DTE v.s. DCE**
  - DTE: data terminal equipment (e.g. PC)
  - DCE: data communication equipment (e.g. modem, switch, router, and other communication device)
  - DTE and DCE have different pin definitions
- **RS232 connectors**
  - DB-25 25-pin connector
  - DB-9 9-pin connector
  - Pin definition for DTE
    - We can either use all 9 pins, or just use 3 pins: TxD, RxD, GND
    - TxD: transmit data
    - RxD: receive data
    - GND: signal ground
    - The remaining pins are for more sophisticated conrols

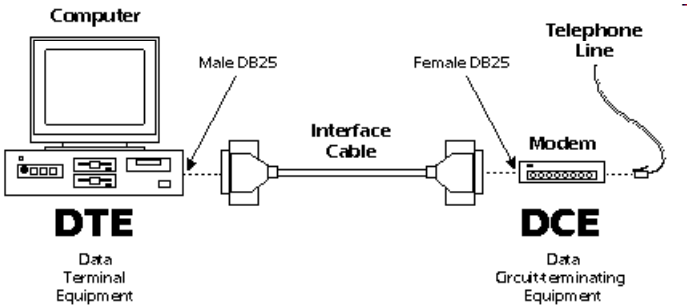


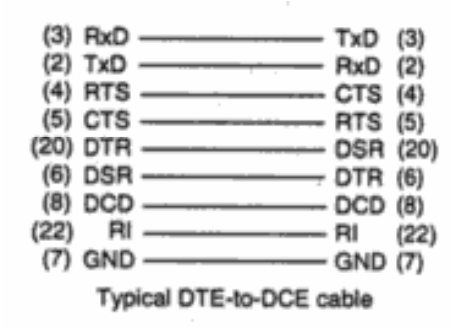


Pin	Description
1	Data carrier detect ($\overline{\text{DCD}}$)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready ($\overline{\text{DSR}}$)
7	Request to send ($\overline{\text{RTS}}$)
8	Clear to send ($\overline{\text{CTS}}$)
9	Ring indicator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

SERIAL COMMUNICATION: RS232

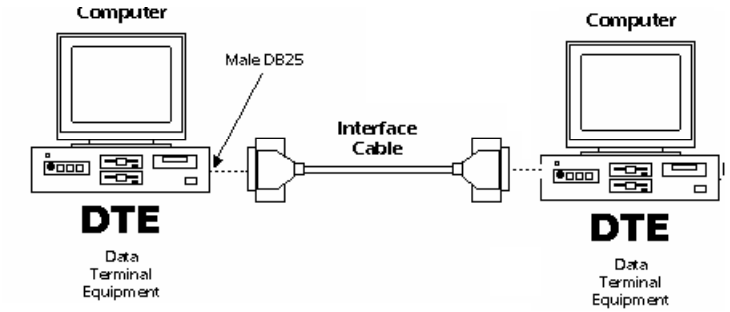
- **Different RS232 cables**
  - DTE ←→ DCE

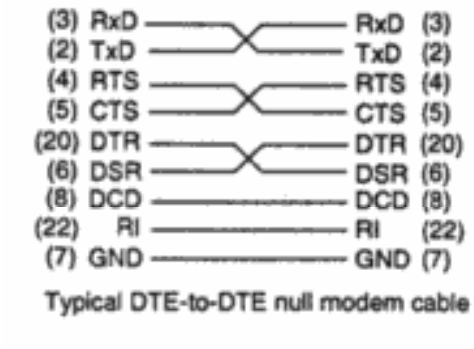




straight-through cable

  - DCE ←→ DCE, or DTE ←→ DTE

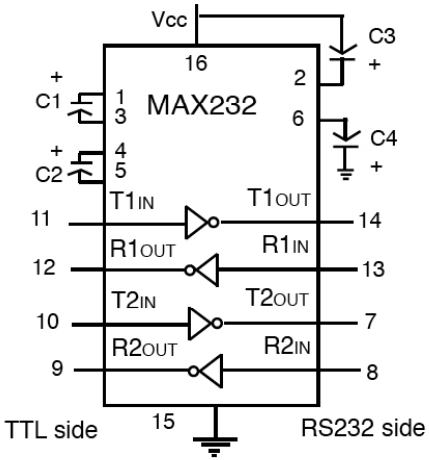


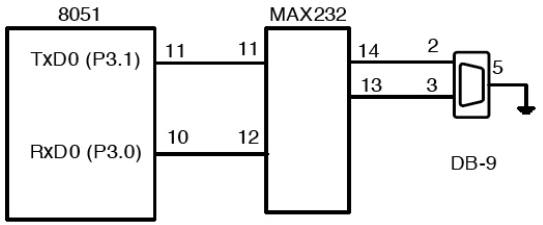


Null-modem cable

SERIAL COMMUNICATION: RS232

- **Signal level of RS232**
  - '1' is represented by -3 to -25 V
  - '0' is represented by +3 to +25 V
  - They are not TTL (transistor-transistor logic) compatible. In TTL
    - '1': 2.2 to 5 V
    - '0': 0 to 0.8 V
  - 8051 is TTL compatible:
    - 8051 and RS232 devices cannot be directly connected together!
- **MAX232**
  - A voltage converter (line driver) that can convert RS232 signal to TTL voltage level
  - 1 MAX232 chip can be used to drive 2 RS232 ports

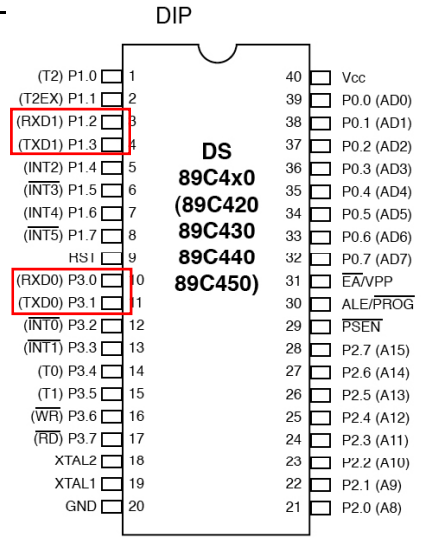




OUTLINE

- Basics of Serial Communication
- **Serial port programming in Assembly**
- Programming the second serial port
- Serial port programming in C

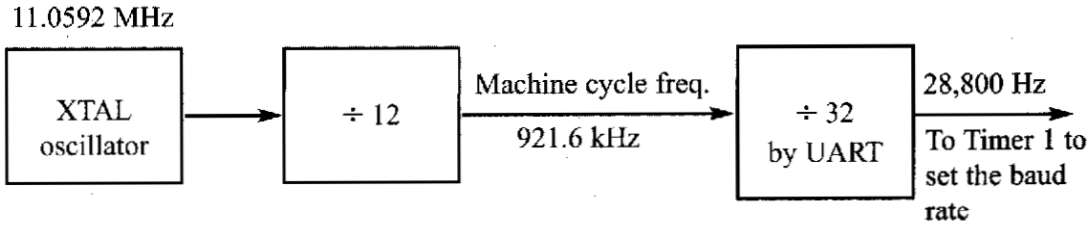
ASSEMBLY: 8051 AND UART



- **8051**
  - Most 8051 has 1 serial port
    - P3.0 (RXD0), P3.1 (TXD0)
  - DS89C4x0 has 2 serial ports
    - P1.2 (RXD1), P1.3 (TXD1)
    - P3.0 (RXD0), P3.1 (TXD0)
- **UART**
  - Universal asynchronous receiver transmitter
  - An integrated circuit commonly used in conjunction with RS232
    - It's built inside 8051
    - The circuit can interpret communication command
      - When an ASCII code is sent to UART, it will automatically add start and stop bits before transmit it through serial port
      - When receiving data from serial port, UART will automatically detect start bit and stop bit, remove the start and stop bits from the received data, and send the pure data to the CPU
      - UART saves us from the details of communication standards.

ASSEMBLY: BAUD RATE AND TIMER

- **Baud rate in 8051 and timer**
  - The baud rates supported by 8051 (unit: bps): 9600, 4800, 2400, 1200
  - How to set the baud rate?
    - Baud rate in 8051 can be set via timer 1 in mode 2 (8-bit auto-reload)
    - When used for serial port, the frequency of timer tick is determined by (XTAL/12)/32



    - 1 bit is transmitted for each timer period (the time duration from timer start to timer expires)

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

Note: XTAL = 11.0592 MHz.

ASSEMBLY: BAUD RATE AND TIMER

- **Calculation of baud rate**
  - With XTAL = 11.0592, find the TH1 value needed to have the baud rate 9600
    - Clock frequency of timer clock: f = (11.0592 MHz / 12)/32 = 28,800Hz
    - Time period of each clock tick: T0 = 1/f = 1/28800
    - Duration of timer (1 timer cycle): (# of clock ticks in timer)*T0
    - 9600 baud → duration of 1 symbol: 1/9600→1 timer cycle: 1/9600
    - 1/9600 = 1/f * (# of clock ticks in timer)
    - # of clock ticks in timer = f/9600 = 28800/9600 = 3 ➔ TH1 =-3
    - Similarly, for baud 2400
      - # of clock ticks = f/2400 = 12 ➔ TH1 = -12
    - Example: set baud rate at 9600

      ```
      MOV TMOD, #20H
      MOV TH1, #-3
      SETB TR1
      ……
      ```
  - When connecting two devices through serial port, both devices must have the same baud

ASSEMBLY: BAUD RATE AND TIMER

- **Example**
  - If the value in TH1 is B8, find the baud rate of the serial port (XTAL = 11.0592MHz)

ASSEMBLY: REGISTERS

- **SBUF register (Serial buffer)**
  - An 8-bit register used for serial communication
  - It holds the data to be transferred or received from serial port.
  - E.g. to send 'D' to serial port: MOV SBUF, #'D'
    - The data in SBUF will be automatically processed by UART, then sent to serial port (e.g. pin TXD0)
  - E.g. to receive data from serial port: MOV A, SBUF
    - Once UART receives data from serial port (e.g. pin RXD0), it will strip the start and stop bits and then put the data in SBUF
  - It serves as a buffer between CPU and serial ports
- **SCON register (Serial control register)**
  - An 8-bit register used to program the start bit, stop bit, and data bits of data framing, and some other serial related processing

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

ASSEMBLY: REGISTERS

- **SCON register**

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

  - SM0, SM1 (serial port mode)
    - Specify framing format, how to calculate baud
    - (SM0, SM1) = (0,1), mode 1: 8-bit data, 1 start bit, 1 stop bit, variable baud set by timer. Most commonly used
    - The other three modes are rarely used (not required for this course)
      - (SM0,SM1) = (0,0), mode 0: fixed baud = XTAL/12
      - (SM0, SM1) = (1,0), mode 2: 9-bit data, fixed baud
      - (SM0, SM1) = (1, 1), mode 3: 9-bit data, variable baud

  - SM2
    - SM2 = 0: single processor
    - SM2 = 1: multiprocessor communication (not required for this course)

ASSEMBLY: REGISTERS

- **SCON register**

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

  - REN (Receive Enable)
    - Enable/disable reception
    - REN = 1: the 8051 will accept incoming data from serial port
    - REN = 0: the receiver is disabled
    - E.g. SETB REN, CLR REN, SETB SCON.4, CLR SCON.4
  - TB8
    - Used by modes 2 and 3 for the transmission bit 8 (the 9th data bit)
    - CLR TB8 when using mode 1
  - RB8
    - Used by modes 2 and 3 for the reception of bit 8 (the 9th data bit)
    - Used by mode 1 to store the stop bit

ASSEMBLY: REGISTERS

- **SCON register**

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

  - TI (transmit interrupt)
    - When 8051 finishes the transfer of the 8-bit character, it set TI to ‘1’ to indicate that it is ready to transfer the next character
    - The TI is raised at the beginning of the stop bit
  - RI (receive interrupt)
    - When 8051 receives a character
      - 1. The UART removes start bit and stop bit
      - 2. The UART puts the 8-bit character in SBUF
      - 3. RI is set to ‘1’ to indicate that a new byte is ready to be picked up in SBUF
    - RI is raised halfway through the stop bit

ASSEMBLY: TRANSMISSION PROGRAM

- **Example**
  - Write a program to transfer letter "A" serially at 4800 baud, continuously

```
                MOV TMOD, #20H      ; timer 1, mode 2 (8-bit auto-reload)
                MOV TH1, #-6        ; 4800 baud
                MOV SCON, #50H      ; 0101 0000 (mode 1, single
                                    ; processor,REN=1)
                SETB TR1            ; start timer
AGAIN:          MOV SBUF, #'A'      ; store 'A' in SBUF
HERE:           JNB TI, HERE        ; wait for TI = 1 (transmission over)
                CLR TI              ; clear TI for next transmission
                SJMP AGAIN          ; repeat
```

  - The data in SBUF is transmitted serially, one bit at a time
    - If we write a new character to SBUF before TI is raised (the transmission of the previous character is not over yet), part of the original data will be lost

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

ASSEMBLY: TRANSMISSION PROGRAM

- **Example**
  - Write a program to transfer letter "YES" serially at 9600 baud, 8-bit data, 1 stop bit, continuously

```
                MOV TMOD, #20H          ; timer 1, mode 2 (8-bit auto-reload)
                MOV TH1, #-3            ; 9600 baud
                MOV SCON, #50H          ; 0101 0000 (mode 1, single processor, REN=1)
                SETB TR1                ; start timer
AGAIN:          MOV A, #'Y'             ; store 'A' in SBUF
                ACALL TRANSFER
                MOV A, #'E'
                ACALL TRANSFER
                MOV A, #'S'
                ACALL TRANSFER
                SJMP AGAIN
                CLR TI                  ;
                SJMP AGAIN              ;
;-----------------------------------------------------
TRANSFER:       MOV SBUF, A
HERE:           JNB TI, HERE
                CLR TI
                RET
```

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

ASSEMBLY: RECEPTION PROGRAM

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

```
        MOV TMOD, #20H
        MOV TH1, #-6          ; 4800 baud
        MOV SCON, #50H        ; mode 1
        SETB TR1
HERE:   JNB RI, HERE          ; wait for char to come in (RI=1)
        MOV A, SBUF           ; save incoming byte in A
        MOV P1, A             ; send to port 1
        CLR RI                ; clear, get ready for next byte
        SJMP HERE
```

  - RI = 1 indicates a new byte is copied in SBUF
  - We need to copy the data in SBUF to another place immediately after RI = 1
    - Otherwise the contents in SBUF will be overwritten by the next character

ASSEMBLY: TRANSMISSION AND RECEPTION

- **Example**
  - Write a program to (1) send to PC “We are ready”; (2) receive data from PC and send it to P1; (3) read data from P2 and send it to PC. (2) and (3) should be performed continuously.

```
        ORG     0
        MOV     P2,#0FFH          ;make P2 an input port
        MOV     TMOD,#20H         ;Timer 1, mode 2(auto-reload)
        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
        MOVC    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 indefinitly
```

ASSEMBLY: TRANSMISSION AND RECEPTION

- **Example (Cont'd)**

```
;---------------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
```

ASSEMBLY: DOUBLE BAUD RATE

- **The baud rate can be doubled by using the PCON register**
  - PCON: (Power control)

SMOD	--	--	--	GF1	GF0	PD	IDL

  - SMOD = 1: double the baud generated by crystal and/or timer
  - SMOD = 0: the baud is determined by crystal and/or timer
  - NOT bit addressable!
  - How to set SMOD?
    - Use register A as an intermediate register

```
MOV A, PCON
SETB ACC.7          ; A.7 is invalid!
MOV PCON, A
```

TH1 (Decimal)	(Hex)	SMOD = 0	SMOD = 1
–3	FD	9,600	19,200
–6	FA	4,800	9,600
–12	F4	2,400	4,800
–24	E8	1,200	2,400

Note: XTAL = 11.0592 MHz.

ASSEMBLY: EXAMPLES

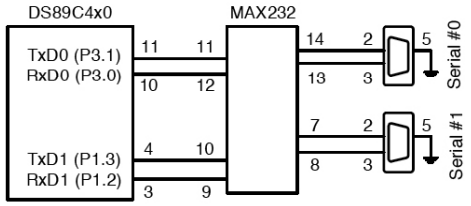
- **Example**
  - Find the baud rate if TH1 = -2, SMOD = 1, XTAL = 11.0592 MHz
  - Write a program to use serial port with baud 19200. (1) read a byte from serial port. (2) if the byte is ‘A’, write ‘1’ back to serial port (3) if the byte is not ‘A’, write ‘0’ back to serial port

OUTLINE

- Basics of Serial Communication
- Serial port programming in Assembly
- **Programming the second serial port**
- Serial port programming in C

SECOND PORT

- **Second serial port**
  - So far, all of our discussions are about the first serial port (serial port 0)
    - TXD P3.1, RXD P3.0
    - SFR addresses:
      - SCON = 98H, SBUF = 99H
      - TL1 = 8BH, TH1 = 8DH, TCON = 88H, PCON = 87H



  - Some 8051/8052 chips have a 2nd serial port (serial port 1)
    - TXD P1.3, RXD P1.2
    - SCON (98H) and SBUF (99H) can only be used for serial port 0
    - TL1, TH1, TCON, PCON can be used for both serial ports
    - We need two new registers SCON and SBUF for serial port 1. In DS89C4x0, for serial port
      - SCON1 = C0H, SBUF1 = C1H
      - TL1 = 8BH, TH1 = 8DH, TCON = 88H, PCON = 87H

SFR	First Serial Port	Second Serial Port
SCON (byte address)	SCON0 = 98H	SCON1 = C0H
SBUF (byte address)	SBUF0 = 99H	SBUF1 = C1H
TL (byte address)	TL1 = 8BH	TL1 = 8BH
TH (byte address)	TH1 = 8DH	TH1 = 8DH
TCON (byte address)	TCON0 = 88H	TCON0 = 88H
PCON (byte address)	PCON = 87H	PCON = 87H

SECOND PORT: EXAMPLE

- **Example**
  - Write a program for the second serial port to continuously transfer “A” at 4800 baud

```
                SBUF1 EQU 0C1H          ; address for 2nd SBUF
                SCON1 EQU 0C0H          ; address for 2nd SCON
                TI1 BIT 0C1H            ; bit address of 2nd TI1
                RI1 BIT 0C0H            ; bit address of 2nd RI1

                MOV TMOD, #20H      ; use timer 1
                MOV TH1, #-6            ; 4800
                MOV SCON1, #50H
                SETB TR1
AGAIN:          MOV A, #’A’
                ACALL SENDCOM2
                SJMP AGAIN
;----------------------
SENDCOM2:
                MOV SBUF1, A            ; SBUF1
      HERE:     JNB TI1, HERE           ; TI1
                CLR TI1
                RET
```

SECOND PORT: EXAMPLE

- **Example**
  - A switch is connected to P2.0. (1) if SW=0, send “Hello” to serial port 0. (2) if SW = 1, send “Goodbye” to serial port 1. 9600 baud. Requirements
    - A. write subroutines: SENDCOM0, SENDCOM1
    - B. Store “Hello” and “Goodbye” in ROM. Both strings are terminated by 0
    - C. Use directives.

```
SBUF1 EQU 0C1H          ; address for 2nd SBUF
SCON1 EQU 0C0H          ; address for 2nd SCON
TI1 BIT 0C1H            ; bit address of 2nd TI1
RI1 BIT 0C0H            ; bit address of 2nd RI1
```

OUTLINE

- Basics of Serial Communication
- Serial port programming in Assembly
- Programming the second serial port
- **Serial port programming in C**

C PROGRAMMING

- **Example**
  - Write an 8051 C program to receive a byte of data from serial port 0, then send it back to serial port 0. Do this continuously.

```
#include <reg51.h>
void SerTx(unsigned char);
void SerRx(unsigned char *);
void main(void)
{
        char byteBuf;
        TMOD = 0x20;        // timer 1, 8-bit auto-reload
        TH1 = 0XFD;         // or: TH1 = -3, 9600 baud
        SCON = 0x50;
        TR1 = 1;            // start timer
        while(1)
        {
                SerRx(&byteBuf);    // read byte from serial port
                SerTx(byteBuf);     // send byte back to serial port
        }
}
```

C PROGRAMMING

- **Example**
  - (Cont'd)

```
void SerTx(unsigned char x)
{
    SBUF = x;                   // put the char in SBUF register
    while(TI = =0);             // wait until transmitted
    TI = 0;
}

void SerRx(unsigned char * pX)
{
    while(RI = =0);             // wait until received
    RI = 0;
    *pX = SBUF;  // copy the data in SBUF to (pX)
}
```

C PRGRAMMING

- **Example**
  - Write a C program to transmit a letter ‘A’ serially at 4800 baud continuously. Use the 2nd serial port with 8-bit data and 1 stop bit.

```
#include <reg51.h>
sfr SBUF1 = 0xC1;
sfr SCON1 = 0xC0;
sbit TI1 = 0xC1;
void main(void)
{
  TMOD = 0x20;                  // timer 1, mode 2
  TH1 = 0XFA;                   // timer 1
  SCON1 = 0x50;                 // SCON1 for 2nd serial port
  TR1 = 1;                      // start timer 1
  while (1)
  {
        SBUF1 = ‘A’;            // SBUF1 for 2nd serial port
        while(TI1 == 0);        //  TI1 for 2nd serial port
        TI1 = 0;
  }
}
```