8051 Serial Communication
Basics of serial communication

**Parallel**: transfers eight bits of data simultaneously over eight data lines
expensive - short distance – fast

**Serial**: one bit at a time is transferred over a single line
cheaper – long (two different cities) - slow

Figure 10-1. Serial versus Parallel Data Transfer
Basics of serial communication

In simplex transmission such as with printers, computer only sends data.
If the data can be transmitted and received, it is a duplex transmission. Duplex can be half or full duplex, depending on whether or not the data transfer can be simultaneous.
Basics of serial communication

- Serial data communication uses two methods: Asynchronous and synchronous.

- The **synchronous method** transfers a block of data (characters) at a time – widely used for character-oriented transmission

- The **asynchronous method** transfers a single byte at a time – block oriented data transfers uses the method
Packaging Data
Start and stop bits

- In asynchronous transmission when there is no transfer the signal is high.
- Transmission begins with a start (low) bit LSB first.
- Finally 1 stop bit (high).
- Data transfer rate (baud rate) is stated in bps (the rate of data transfer in serial data communication is stated in bits per second).

Figure 10-3. Framing ASCII “A” (41H)
The 8051 has a single full duplex (means it can transmit and receive data at a time) serial port that can be used for either asynchronous or synchronous communication.

The port has 4 modes of operation:

- Mode 0: Synchronous at 1/12 oscillator frequency.
- Mode 1: 10 bit Asynchronous - variable baud rate.
- Mode 2: 11 bit Asynchronous - 1/32 OR 1/64 of the oscillator frequency.
- Mode 3: 11 bit asynchronous - variable rate.
In the memory map we have serial buffer special function register (SBUF) at location 99H.

Unlike any other register in the 8051, SBUF is in fact two distinct registers - the write-only register and the read-only register.

Transmitted data is sent out from the write-only register while received data is stored in the read-only register.

There are two separate data lines, one for transmission (TXD)(P3.1) and one for reception (RXD)(P3.0). Therefore, the serial port can be transmitting data down the TXD line while it is at the same time receiving data on the RXD line.
SBUF register

- Data to be transmitted via the TXD line must be placed in SBUF register.
- SBUF holds the byte of data when it is received by 8051 RXD line.
- MOV SBUF, #45H - this sends the byte 45H down the serial line
- MOV A, SBUF - this takes whatever data was received by the serial port and puts it in the accumulator
- Once byte is written in SBUF, it is framed with the start and stop bits and transferred serially via the TXD pin.
- When the bits are received serially via RXD, the 8051 deframes it by eliminating the stop and start bits and then placed it in SBUF register.
## Serial control (SCON) Register

8-bit register used to program the start bit, stop bit and data bits of data framing

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SM1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SM2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TB8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RB8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SM0**: mode specifier

**SM1**: mode specifier

<table>
<thead>
<tr>
<th>SM0</th>
<th>SM1</th>
<th>MODE</th>
<th>Operation</th>
<th>Transmit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>shift register</td>
<td>fixed (xtal/12)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8 bit UART</td>
<td>variable (timer1)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>9 bit UART</td>
<td>fixed (xtal/32 or xtal/64)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>9 bit UART</td>
<td>variable (timer1)</td>
</tr>
</tbody>
</table>
**SM2**: used for multi processor communication – make it 0 for mode 1

**REN**: receive enable (by software enable/disable)

- when high, it allows the 8051 to receive data on the RXD pin of the 8051
- 8051 in simultaneous transmit and receive mode REN must be set to 1
- REN = 0 means receiver is disabled

**TB8**: transmit bit 8 – not widely used – make it 0 – used for serial mode 2 and 3

**RB8**: receive bit 8– not widely used – make it 0 – in mode 1 this bit gets a copy of the stop bit, when 8 bit data is received. Also used for mode 2 and 3.

**TI**: transmit interrupt flag *set by H/W after send*, clear by SW

- when 8051 finishes the transfer of the 8–bit character, it raises the TI flag to indicate that it is ready to transfer another byte. TI is raised at the beginning of the stop bit.

**RI**: receive interrupt flag *set by HW after received*, clear by SW

- when 8051 receives data serially via RXD it gets rid of the start and stop bit and places the byte in SBUF register. Then it raises the RI bit to indicate that the byte has been received and should be picked up before it is lost. RI is raised halfway through the stop bit.
Mode 0 of operation

- Mode 0: simply a shift register
  - Serial data enters and exits through **RxD**
  - **TxD** outputs the shift clock.
  - 8 bits are transmitted/received (LSB first)
  - The baud rate is fixed at 1/12 the oscillator frequency.

Application
- Port expansion
As can be seen in the diagram the terms TXD and RXD are misleading in mode 0.

TXD serves as the clock while RXD is used for both receiving and transmitting data.

In mode 0, the serial port is only half duplex; it cannot transmit and receive data at the same time because the same line (RXD) is being used for both transmit and receive.

The serial port in mode 0 is an example of synchronous communication; the clock signal is sent with the data on the TXD line. TXD pulses at the same frequency as the machine cycle means TXD runs at 1/12th the frequency of the system clock.

If we are using a 12MHz system clock, then the frequency of TXD is 1MHz, which implies its cycle length is 1us. Therefore, each bit is active on the RXD line for 1us. To shift the entire 8-bit word along RXD takes 8us.
Mode 1 of operation

- Mode 1
  - Ten bits are transmitted (through TxD) or received (through RxD)
    (A start bit (0), 8 data bits (LSB first), and a stop bit (1))
  - On receive, the stop bit goes into RB8 in SCON
  - The baud rate is determined by the Timer 1 overflow rate.
  - Timer1 clock is 1/32 machine cycle (MC=1/12 XTAL)

- Timer clock can be programmed as 1/16 of machine cycle
- Transmission is initiated by any instruction that uses SBUF as a destination register.
Setting the Serial Port Baud Rate

- Serial ports baud rate setting is required for modes 1 and 3.
- The Baud Rate is determined based on the oscillator’s frequency when in mode 0 and 2.
- In mode 0, the baud rate is always the oscillator frequency divided by 12. This means if your crystal is 11.059Mhz, mode 0 baud rate will always be 921,583 baud.
- In mode 2 the baud rate is always the oscillator frequency divided by 64, so a 11.059Mhz crystal speed will yield a baud rate of 172,797.
- In modes 1 and 3, the baud rate is determined by how frequently timer 1 overflows.
- The more frequently timer 1 overflows, the higher the baud rate. There are many ways one can cause timer 1 to overflow at a rate that determines a baud rate, but the most common method is to put timer 1 in 8-bit auto-reload mode (timer mode 2) and set a reload value (TH1) that causes Timer 1 to overflow at a frequency appropriate to generate a baud rate.
Setting the Serial Port Baud Rate

- To determine the value that must be placed in TH1 to generate a given baud rate, we may use the following equation (assuming PCON.7 is clear).

- \[ TH1 = 256 - \left( \frac{\text{Crystal}}{384} \right) / \text{Baud} \]

- \[ TH1 = 256 - \left( \frac{11059000}{384} \right) / 9600 \]
  \[ TH1 = 256 - \left( \frac{28,799}{9600} \right) \]
  \[ TH1 = 256 - 2.9999 = 253.0001 = \text{FD H} \]
Programming steps to transfer data serially
( in mode 1 )

1. TMOD Reg is loaded with 20H (timer 1 in mode 2) to set the baud rate
2. TH1 is loaded with the count for specific baud rate
3. SCON with 50H if simultaneous transmit and receive operations are required else only for transmit load 40 H for mode 1.
4. Tr1 is set to start time -1
5. TI is cleared by CLR TI means make it 0 during the starting
6. The character byte to be transferred serially is written into SBUF.
7. TI flag is to be monitored to if the character has been transferred completely.
8. To transfer the next character go to step 5
Programming steps to receive data serially (in mode 1)

1. TMOD Reg is loaded with 20H (timer 1 in mode 2) to set the baud rate
2. TH1 is loaded with the count for specific baud rate
3. SCON with 50H in mode 1, receive enable must be turned on
4. Tr1 is set to start time -1
5. RI is cleared by CLR RI means make it 0 during the starting
6. The character byte to be transferred serially is written into SBUF.
7. RI flag is to be monitored to if the character has been transferred completely.
8. To transfer the next character go to step 5
Mode of operation

- **Mode 2**:  
  - Eleven bits are transmitted (through TxD), received (through RxD)  
  - A start bit (0)  
  - 8 data bits (LSB first)  
  - A programmable 9th data bit  
  - and a stop bit (1)  
  - On transmit, the 9th bit (TB8) can be assigned 0 or 1.  
  - On receive, the 9th data bit goes into RB8 in SCON.  
  - the 9th can be parity bit  
  - The baud rate is programmable to 1/32 or 1/64 the oscillator frequency in Mode 2 by SMOD bit in PCON register

- **Mode 3**:  
  - Same as mode 2  
  - But may have a variable baud rate generated from Timer 1.
Modes 1 and 3 can be used for standard asynchronous communication.

In this case, the RXD and TXD bits are normally routed through a level converter chip to produce and receive the specified RS232 levels.
Modes 2 and 3 can be used for multiprocessor communication. In this case, the 9th bit can be used as an address marker (bit 9 =1).

When the master sends out an address byte, all slaves will receive the byte and only one will be identified as the target for the message.

That processor will clear its SM2 bit and will be interrupted for all further message bytes.

Other slaves will leave their SM2 bit set and will not be interrupted by the data bytes that follow.
Mode of operation

(a) Mode 0

On-chip oscillator → +12 → Baud rate clock

(b) Mode 2

On-chip oscillator → +64 → SMOD = 0
On-chip oscillator → +32 → SMOD = 1 → Baud rate clock

(c) Modes 1 and 3

Timer 1 overflow → +32 → SMOD = 0
Timer 1 overflow → +16 → SMOD = 1 → Baud rate clock
What is SMOD

- Bit 7 of PCON register
- If SMOD=1 double baud rate
- PCON is not bit addressable
- How to set SMOD
  Mov a, pcon
  Setb acc.7
  Mov pcon,a
### Power control register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCON.7</td>
<td>SMOD</td>
<td>Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2, or 3.</td>
</tr>
<tr>
<td>PCON.6</td>
<td>—</td>
<td>Reserved.</td>
</tr>
<tr>
<td>PCON.5</td>
<td>—</td>
<td>Reserved.</td>
</tr>
<tr>
<td>PCON.4</td>
<td>—</td>
<td>Reserved.</td>
</tr>
<tr>
<td>PCON.3</td>
<td>GF1</td>
<td>General-purpose flag bit.</td>
</tr>
<tr>
<td>PCON.2</td>
<td>GF0</td>
<td>General-purpose flag bit.</td>
</tr>
<tr>
<td>PCON.1</td>
<td>PD</td>
<td>Power-Down bit. Setting this bit activates power-down operation.</td>
</tr>
<tr>
<td>PCON.0</td>
<td>IDL</td>
<td>Idle mode bit. Setting this bit activate idle mode operation.</td>
</tr>
</tbody>
</table>
Power-Down Mode

- An instruction that sets PCON.1 causes power down mode
- Last instruction executed before going into the power down mode
- The on-chip oscillator is stopped.
- All functions are stopped, the contents of the on-chip RAM and Special Function Registers are maintained.
- The ALE and PSEN output are held low
- The reset that terminates Power Down
Programming the serial communication interrupt

- TI – will be raised when the last bit of the framed data the stop bit is transferred indicating that SBUF register is ready to transfer the next byte.
- RI – is raised when the entire frame of data including stop bit is received.
- Implementation either by polling or an interrupt
- IE.4 in IE register is enabled, when RI or TI is raised the 8051 is interrupted and jumps to memory location 0023 H to execute the ISR.
- In the ISR we must examine the TI and RI flag to see which caused the interrupt and responds accordingly.
- the 8051 does not automatically clear the RI and TI flags you must clear these bits in your interrupt handler whereas in external and timer interrupt it is the job of the 8051 to clear the interrupt flags.
There are 3 commonly used error indications for UARTs.

These typically set a flag in a status register and may cause an interrupt.

Overrun Error: The received byte is not read by the processor before the next byte is received. (Data is lost.)

Framing Error: The receiver does not sense the stop bit at the appropriate time (usually due to baud rate or format difference).
Parity Error: The expected parity is incorrect. (This can be done with software.)

Most UARTs receive data based on a clock at 16 times the baud rate.

This allows a more precise determination of when the start bit actually appears.

Some UARTs have a fourth error condition based on whether several samples in what should be the middle of a bit time agree.

This error indication is typically called a “noise” flag.