Why serial data communication

Less cost than parallel where cable with no of wires = no of bits to transfer or receive

In serial one line is sufficient for data tx and rx and so cost is less

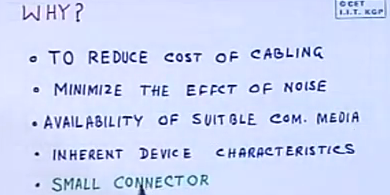
Less noise since only pair of wires

With bunch of parallel wires there will be cross talk and noise increases

So serial communiation reduces the effect of noise

Availability of suitable commn media

Main commn media are inherently serial in nature – telephone line / satellite and other private links are serial in nature



In context of battery operated devices – handheld – small connector is required than in parallel where we need a parallel port connector

Small connector with only two pins for tx and rx

In context of MCU – need for serial communication is more stringent and important

**Need for distributed computing in embedded environment**

**Sensor networking**

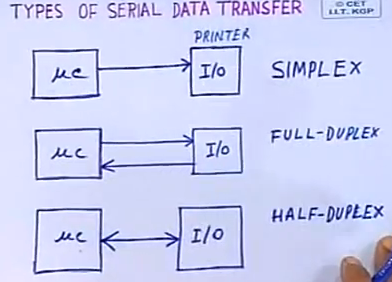
MCU are presently used in realizing the sensor networks – where large no of devices communicating with each other – we need serial communication

8051 has provided an inbuilt serial communication facility

**Key features of 8051 serial interface**

**It supports full duplex communication** –

Different types of serial data transfer



Simplex – only one way – MC to IO

Full duplex – simultaneous read and write – two way communication – separate line for tx and rx

Half duplex – same line for two way communication bw MC and IO

Use suitable protocol in half duplex so that communication can be done in both direction in one after another

**Receive buffered** – serial circuitry is provided with two buffers

One for communicating with microprocessor

One for serially collecting data from the serial data line

After 1 byte is buffered – we can receive the second byte but the first buffered byte data should be read before the second byte is buffered

Otherwise the data will be lost

So receiving data serially into buffer and reading data from the serial buffer can take place simultaneously upto one byte because of the serial buffer facility (receive buffered)

**SFR – SBUF**

Although single buffer is available for serial rx and tx

Internally there exists two buffers

When transmitting – data goes directly into transmit buffer

When receiving – data goes directly into receive buffer

So physically device has got two such buffers for transmit and receive which actually facilitates full duplex communication

**Four different modes of serial operation**

to satisfy the need of wide range of apps

different apps need different data rates

also no of bits – 8/9 to tx or rx

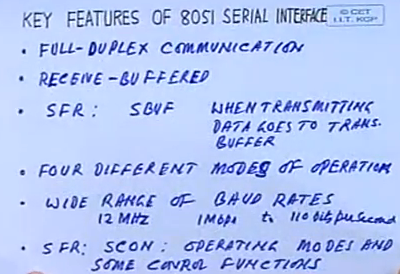
**in four different modes it supports wide range of baud rates**

ex with max clock freq for a typical 8051 12MHz – we can have baud rate ranging from 1Mbps to 110Mbps

such a wide range of baud rate is supported which can support high and low speed data communication depending on the apps need

**SFR to facilitate programming**

**SCON** – serial control – to define operating modes and control functions



We can operate the serial device different modes of operation and also can control some modes

**PCON – commonly used for power control**

But one MS bit is used for controlling the baud rate in serial communication

**How to program SCON SFR to configure the serial interface in different modes of operation**

SCON – serial port control register – SFR

Mode 0 – shift register – baud rate will be typically with 12Mhz gives 1Mbps – we can communicate at higher rate

Mode 1 - 8 b UART is the normal usage of serial interface and baud rate is variable and programmable

9b UART mode 2 and 3 are used typically for multiprocessor communication

Baud rate is at higher speed (divide by 32 or 64) or variable

SM2 – enable the use of serial device in multiprocessor communication mode

REN – enable or disable receiving of data

TB8

0-7 is the normal 8b data

So now the 8th bit (ie 9 bit ) can be decided by the programmer

It can be parity bit

Or it can used for multiprocessor communication – for communication in master slave mode

ie bit 8 which is stored in TB8 is transmitted as the 8th bit in 9b data communication in serial mode

RB8

8 th bit received in 9b serial mode that goes into this RB8 bit

TI – used for interrupting the processor after the completion of tx of 1 byte of data

Automatically set by hw after tx of a byte of data

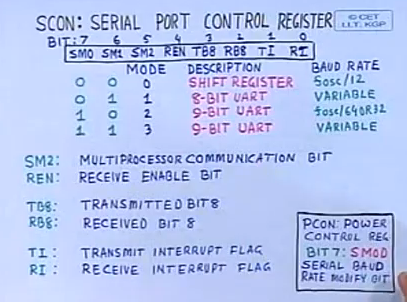
Can be reset by programming or whenever the processor jumpts to ISR of serial interrupt for TI

RI – for rx of 1 byte data

Used in the receive mode of data tx whenever we are receiving the data

After 1 byte of data is received or 9b of data in other modes of operation – so after 7th or 8th bit is received – hw sets this bit and it can interrupt the processor

It can be reset from programming or whenever the processor jumps to IRQ of serial interrupt then this RI bit is resetted automatically



PCON – bit 7 – Msb – serial baud rate modify bit

When 1 – baud rate is modified – doubled

When 0 – baud rate is as programmed

**Mode 0 operation – shift register mode of operation**

**Use** the device in half duplex mode – a single line for data communication bw MCU and IO

**So half** duplexed communication is done in mode 0 operation

**Here** rxd pin is used for both tx and rx

**So at a time** either we can transmit or receive using the rxd pin and cannot do both simultaneously

**So only one pin (rxd) for communication of data**

Txd pin is used for generating the shift signal which is also avbl outside for shifting data into the hw or discrete circuitry

This mode is not used for data communication in computers

Since baud rate in this mode will be fosc/12

So for 12MHz max frequency -> baud rate will 1Mbps which is not possible in rs232 standard and other serial standard modes

However we can tx data or shfit data into the hw devices in this mode of operation

So not used for data communication with other computers

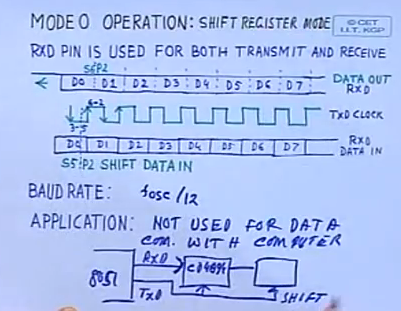
Used for sending data to other external devices

Ex: 8051 connected to shift registers

Rxd pin for data pin

Txd pin for shift signal

So we can shift data into the shift registers from 8051



Whenever tx data – in above case of data out via rxd pin

Each bit is shifted out at state6 (S6) phase 2 of the clock (12 clock cycles makes 6 states with each state having 2 phases)

So in each machine cycle – 1 bit of data is shifted out via rxd pin

In this case each bit shifted for every machine cycle and shifted at s6 p2 of the machine cycle

Data is shifted out serially through rxd pin

Txd pin acting as a clock for shifting of data to the outside world

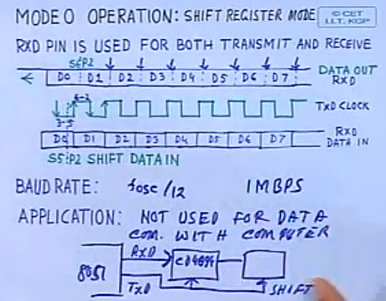
It is low for states 3-5 and high for states 6-2 and so acts as a clock for shfting data

Similarly on receiving data from rxd line – edge shown above is used for reading the bit data

Shifting data in (bit) is done at s5 p2 of the clock

P2 is the trailing edge of a clock period

So s5 p2 just one clock cycle before the txd pin goes high – during which data is shifted in



So this mode 0 is useful for communication with discrete devices at max baud rate of 1MBps

**Mode 1 operation**

**Most** widely used mode of operation

Standard UART

Supports 10b full duplex receiver/transmitter

It includes start bit – sync takes place at the trailing edge of the start bit from idle state

After start bit – it goes high and low as per the data bit for the bit duration

At the receiving end – shifting of the data takes place in the middle of the clock period

As a result of shifting data in the middle of the clock period – this allows the small difference in the frequency bw the clock frequency of the transmitter and the receiver

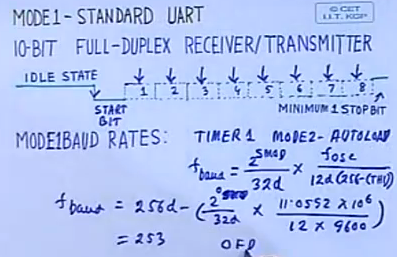
This avoids the noises or false data transmission (since data is read in the middle of the bit period)

In this case MP is provided with both rxd and txd lines so full duplex communication – transmit and receive simultaneously

Baud rate in this mode is programmable

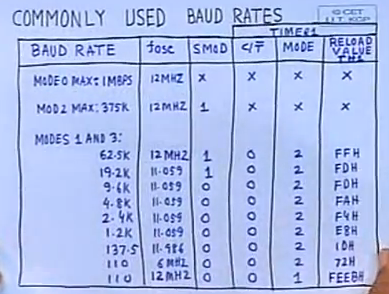
Timer 1 in mode 2 – 8b auto reload is usually used for baud rate generation

As a result a wide range of baud rate can be generated with the help of timer 1



Various baud rates can be generated with the help of the timer 1 at different clock frequency

It supports the RS232 standard baud rate ranges from 19.2K to 110 bps – and so can be very easily used for data communication with any standard computer using the standard rs232 baud rate



We will use it for data communication with standard interface RS232C – serial interface

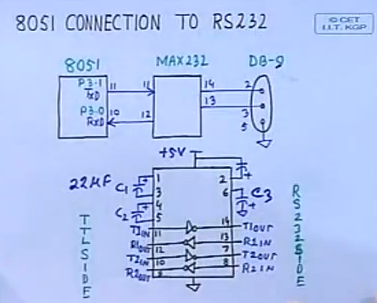
**How 8051 can be interfaced with RS232C interface**

8051 outputs are TTL compatible

And RS232C outputs are not compatible with TTL

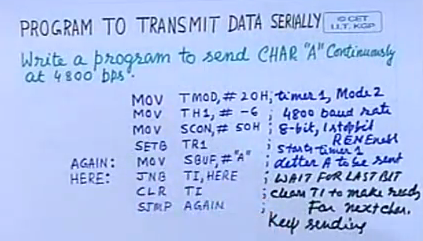
So level converter IC is used for level conversion bw TTL and RS232

MAX232 provides 2 pair of signal converters inside IC

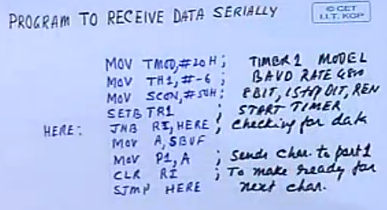


**Now how the device can be programmed for tx data and rx data serially**

Timer 1 actually generates the clock – which is supplying the baud rate



Either poll TI or ISR

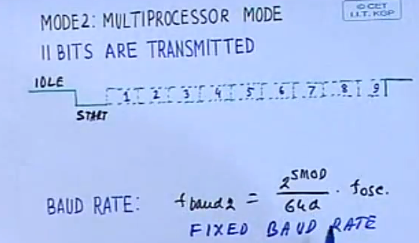


**Mode 2 operation**

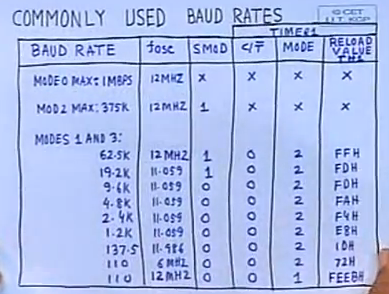
11 bit – start + 8b data + 9th bit + stop bit

9th bit is decided from program via TB8 and RB8 in SCON

Baud rate is fixed and not variable



Different baud rates can be



With 12MHz fsoc – and smod=1 the baud rate will be 375K bps which is high – can be used for communication

Again here data is sent in same fashion as in mode 1 but the no of bits is 11 instead of 10

**Mode 3 operation – standard UART – 11 bits with variable baud rate**

Mode 1 and 3 has variable baud rate from timer 1 in mode 2

Mode 0 and 2 has fixed baud rate

Main use of mode 2 and mode 3 is for multiprocessor communication (special provision)

SM2 bit decides the multiprocessor mode of communication

Ex: large no of sensors networked – where each sensor communicate…

And proactive computing from intel – auto action based on data from sensors without interacting with user

To facilitate proactive computing – must have large no of sensors communicating with each other

So if we have large no of computers or sensors for which 8051 is used as processor

Now here one can be master and others can be slaves

Now 9th bit in RB8 and SM2 bit – can be used to address the slaves with which master wants to communicate

Ex: RB8 = 1 and SM2=1 => all the slaves will be interacted – in this mode we are sending the address

8b data sent in first case will be essentially the address of slave of interest

All the slaves of interest will disable the SM2 bit and subsequently will be ready to receive the data from the master

On sending address – we make rb8=1

On sending data – we make rb8=0

So slaves which are relevant as per the address will keep sm2=0 and all other not relevant will keep sm2=1

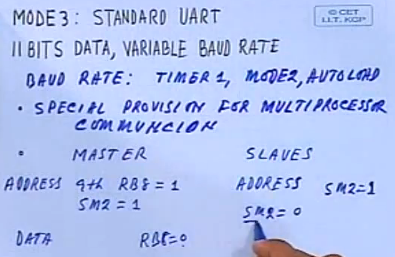
Those with sm2=1 will not be interrupted when data byte is sent

Those with sm2 =0 will be ready to receive data

In this way communication takes place bw large no of sensors

Mode 2 can be used for higher baud rate in multiprocessor mode

Mode3 for smaller baud rate in multiprocessor mode



With rb8 = 0 for sending data only designated slaves will be ready to receive data and others with sm2=1 are not interrupted

So we can used 8051 to implement intelligent nw for large no of sensor nw

Serial Communication

|  |
| --- |
| Serial Communication is a form of I/O in which the bits of a byte being transferred appear one after other in a timed sequence on a single wire. Serial Communication uses two methods, asynchronous and synchronous. The Synchronous method transfers a block of data at a time, while the asynchronous method transfers a single byte at a time. In Synchronous Communication the data get transferred based on a common clock signal. But in Asynchronous communication, in addition to the data bit, one start bit and one stop bit is added. These start and stop bits are the parity bits to identify the data present between the start and stop bits.  The 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 (Port-3.0 and Port-3.1). Pin 11 of the 8051 is assigned to TXD and pin 10 is designated as RXD. These pins are TTL compatible; therefore they require a line driver to make them RS232 compatible. The line driver chip is MAX232. The MAX232 uses +5v power source, which is same as the source voltage for 8051.  The 8051 transfers and receives data serially at different baud rates. The baud rate of the 8051 is programmed into the timers.   Generally Null Modem Connections are used for Data transfer between two device serially.  http://www.electroons.com/8051/electroons/images/rs232.jpg  SCON (Serial Control Register) is responsible for all serial communication related settings in 8051.  http://www.electroons.com/8051/electroons/images/SCON.gif  **Calculating Baud Rates**  As we know that 8051 microcontrollers takes 12 clock cycles to complete one machine cycle.  So our effective Instruction execution frequency is Fosc/12. If we are using a crystal of 11.0592MHz; our efeective frequency is somewhere around Feffective= 11.0592/12 MHz => 921.6 KHz.  8051 UART or serial communication block further divide this frequency (921.6 KHz) by 32 to generate its baud rate.  Therefore Effective frequency available to generate Baud rates is 921.6 KHz/32 = 28800 Hz.  So for different standard baud rates the values of TH1 will be   Baud Rate 9600 -- TH1=0xFD --- because 28800/9600 = 3   **Serial Buffer Register (SBUF)** SBUF is an 8-bit register used for serial communication specific programs. For a byte to to be transferred via TxD line, it must be placed in the SBUF register. Similarly SBUF holds the byte of data when it is received by 8051's receive line. SBUF can be accessed similar to any other register in 8051, but it is not bit addressable.   **Hardware Connections** As explained above We need a RS232-TTL level converter to enable 8051 communicate serially with other RS232 compatible devices. Here is the connection schematic...  http://www.electroons.com/8051/electroons/images/rs232_1.jpg  Final Schematic Diagram -  http://www.electroons.com/8051/electroons/images/rs232_final.jpg    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* >Example program to send some characters serially at 9600 bps. >Make all connection according to the schematic given above. >Serial port Mode 1 is used with 8bit data, 1 stop bit, 1 start bit >One **important** thing is that all calulations for baud rate generation using Timer1 are made for Timer1 8 bit auto reload mode \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   #include>at89x52.h<   void main(void) { TMOD=0x20; // Timer1 Mode2 8 bit auto reload TH1=0xFD; // 9600 bps SCON=0x50; // 8 Data bit, 1 start bit, 1 stop bit TR1=1; // Timer1 ON while(1==1) { SBUF='S'; while(TI==0); // Pole TI flag for complete transmission TI=0; SBUF='A'; while(TI==0); TI=0; SBUF='M'; while(TI==0); TI=0; } }      /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* >Example program to send a string serially at 9600 bps. >Make all connection according to the schematic given above. >Serial port Mode 1 is used with 8bit data, 1 stop bit, 1 start bit >One **important** thing is that all calulations for baud rate generation using Timer1 are made for Timer1 8 bit auto reload mode \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   #include>at89x52.h<   void serial(unsigned char x) { SBUF=x; while(TI==0); TI=0; }   void rs\_puts(char \*aaa) { unsigned int i; for(i=0;aaa[i]!=0;i++) { serial(aaa[i]);  } } void main(void) { TMOD=0x20; // Timer1 Mode2 8 bit auto reload TH1=0xFD; // 9600 bps SCON=0x50; // 8 Data bit, 1 start bit, 1 stop bit TR1=1; // Timer1 ON while(1==1) { rs\_puts("www.electrroons.com\n\r"); } }     Now its your turn to make a program to receive data from UART and show it on LCD. Poll RI (Receive Flag) and as RI goes 1; retrive your data from serial buffer, and call lcd\_data(c=SBUF);   Try it and mail me if you face any problem. |