Microcontroller (EEC421)

Lecture 4

Dr. Islam Mohamed

Electrical Engineering Department
Shoubra Faculty of Engineering, Benha University
Islam.ahmed@fen.bu.edu.eg

1- DC Machines

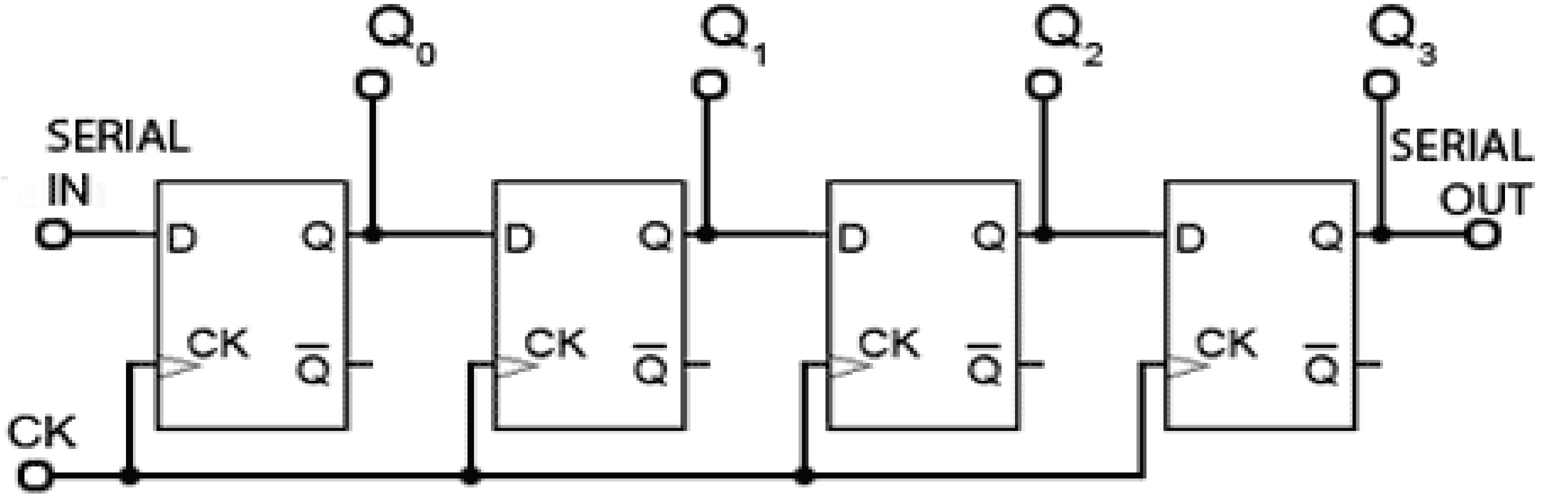
- ❑ **<u>Lecture 4: THE 8051 ARCHITECTURE</u>**
- ✓ 8051 Microcontroller Hardware
- ✓ Counters, Registers, Memory
- ✓ Input / Output Pins, Ports, and Circuits
- ✓ Timers, Serial data, interrupt

What is Shift Register?

- Shift Registers are constructed using latches or the Flip-Flops. These Sequential circuits are used to generate certain delays in the digital waveforms or signals.
- To send data bits into the microprocessors/microcontrollers and for reading data out these, shift registers are used.
- The ‘n’ number of stages in the register is proportional to the delay generated in the circuit by n times.
- These registers play a vital role in long distance communication between two processors to avoid the complexity of parallel wiring and its slow data transmission.

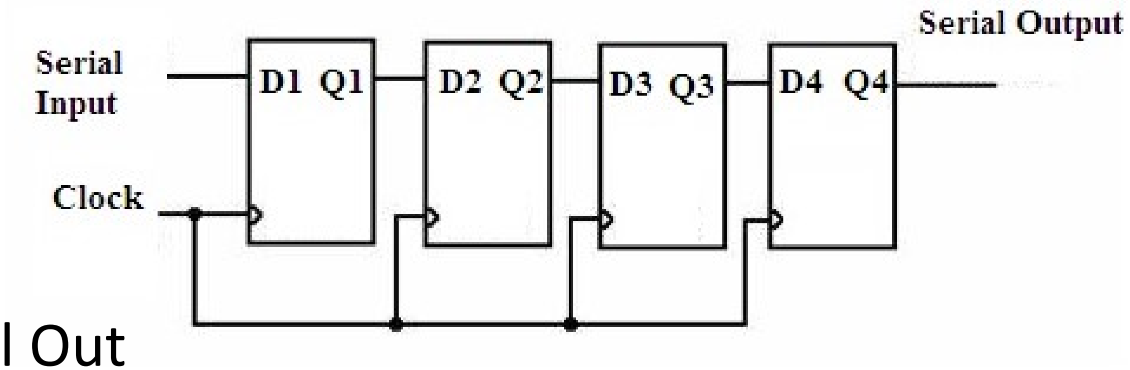
Shift Registers:

- Shift Registers are the logic circuits of sequential type. These circuits are influenced by both present stage inputs and Past values.
- The basic purpose of these circuits is to store and transfer the data. The data stands for binary information.
- These registers are comprised of latches which are of D-type. The number of bits is directly proportional to the number of latches used. The formation of shift registers is dependent on the latches used.
- The most frequently used one is the register with eight bits. The data transfer or the data taken for storage can follow the process of parallel or serial both.

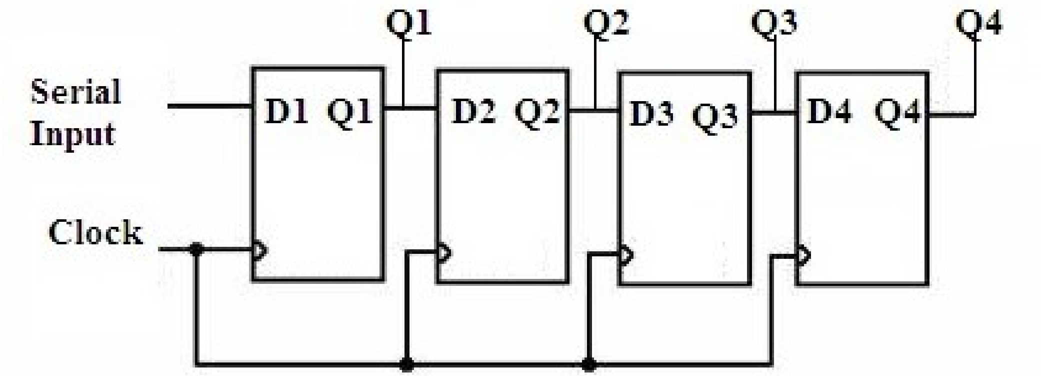


Types of Shift Registers,

- Serial in serial Out

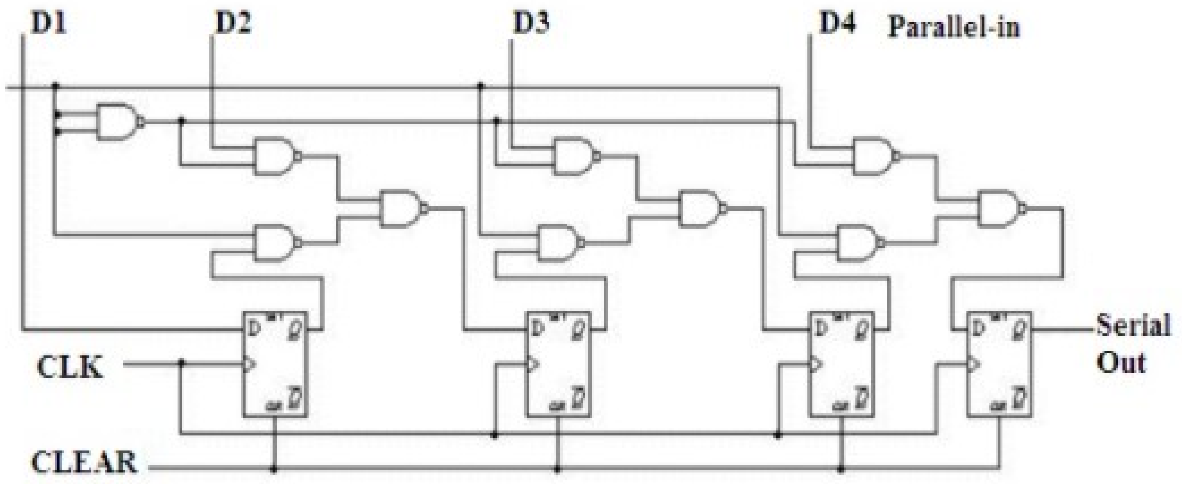


- Serial in parallel Out

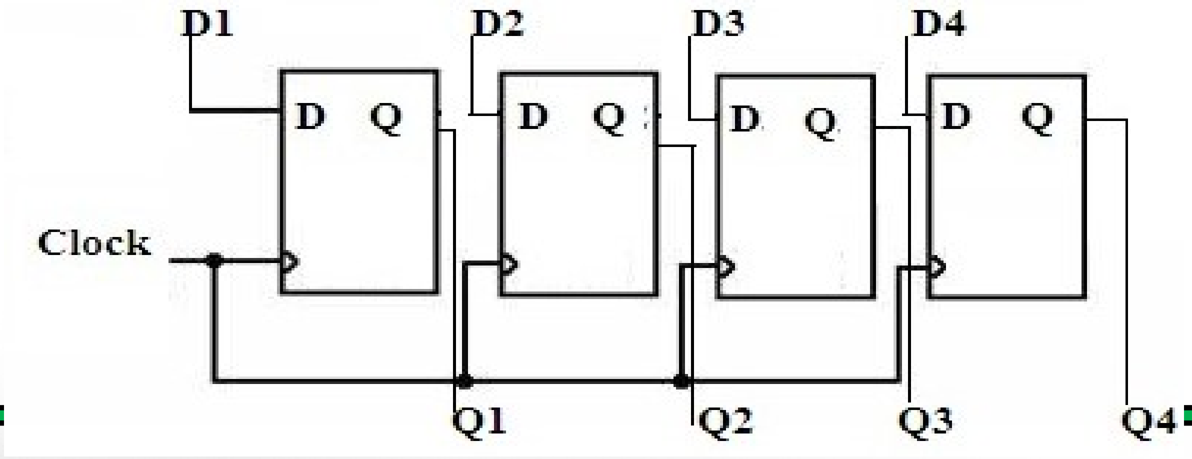


Types of Shift Registers,

- Parallel in Serial Out

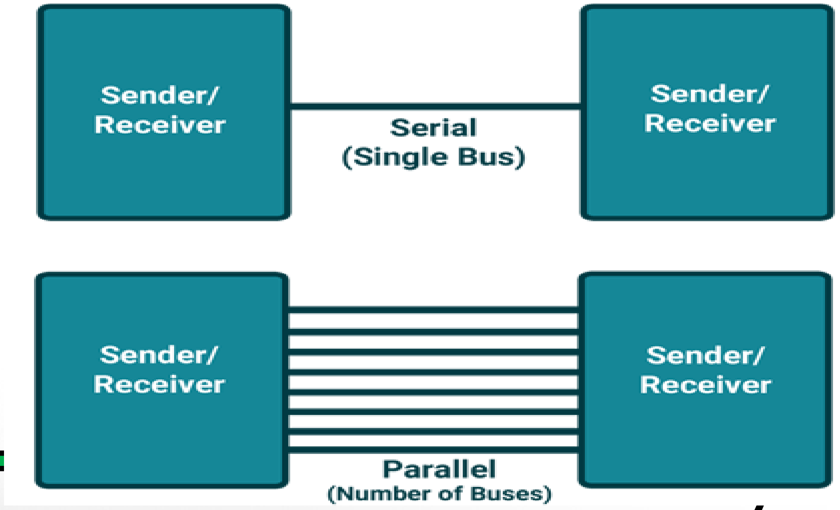


- Serial in parallel Out



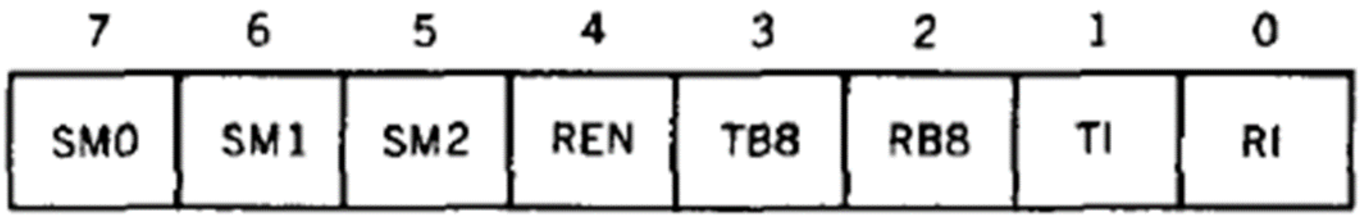
Serial Data Input / Output

- Computers must be able to communicate with other computers in modern multiprocessor.
- One cost-effective way to communicate is to send and receive data bits serially.
- The 8051 has a serial data communication circuit that uses register SBUF to hold data. Register SCON controls data communication, register PCON controls data rates, and pins RXD (P3.0) and TXD (P3. I) connect to the serial data network.



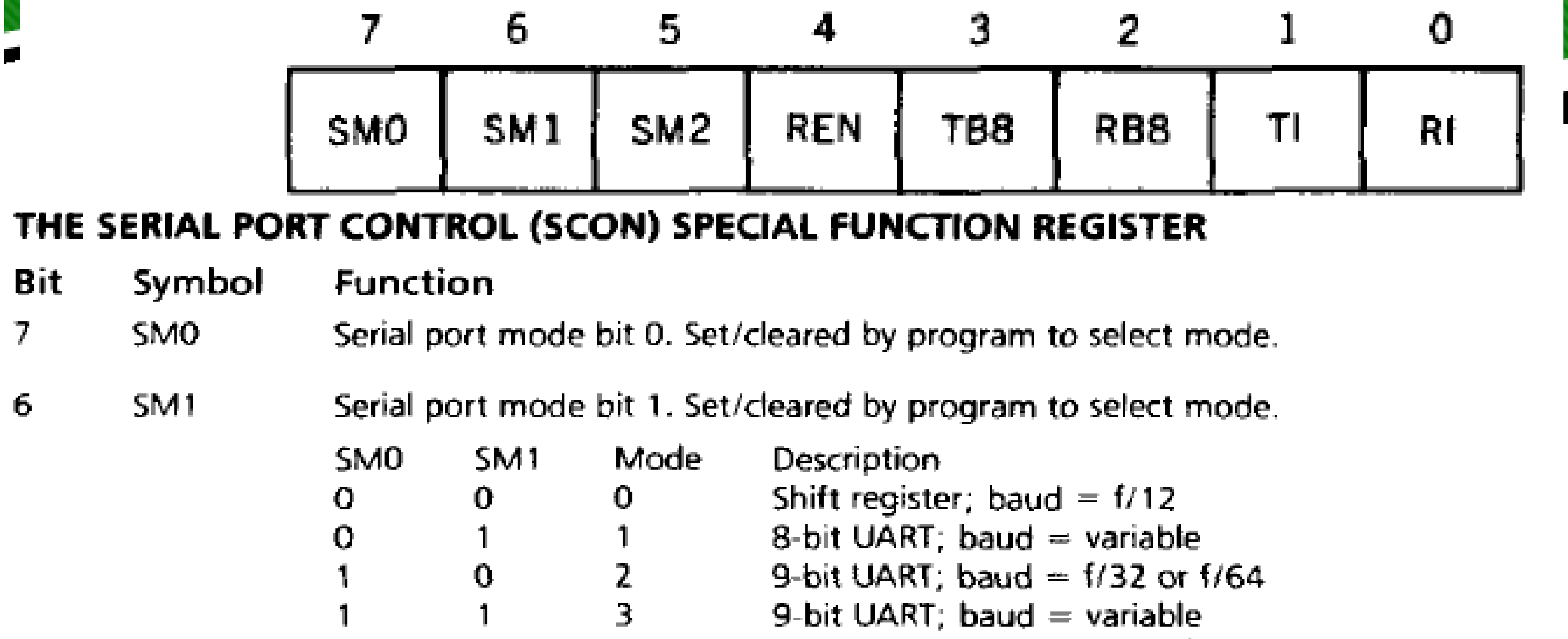
Serial Data Input / Output

- SBUF is physically two registers. One is write only and is used to hold data to be transmitted out of the 8051 via TXD. The other is read only and holds received data from external sources via RXD.
- There are four programmable modes for serial data communication that are chosen by setting the SMX bits in SCON.
- Baud rates are determined by the mode chosen. Figure shows the bit assignments for SCON and PCON.



7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Serial Data Input / Output



7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

THE SERIAL PORT CONTROL (SCON) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7	SM0	Serial port mode bit 0. Set/cleared by program to select mode.
6	SM1	Serial port mode bit 1. Set/cleared by program to select mode.

SM0	SM1	Mode	Description
0	0	0	Shift register; baud = f/12
0	1	1	8-bit UART; baud = variable
1	0	2	9-bit UART; baud = f/32 or f/64
1	1	3	9-bit UART; baud = variable

The baud rate is the rate at which information is transferred in a communication channel. Baud rate is commonly used when discussing electronics that use serial communication. In the serial port context, "9600 baud" means that the serial port is capable of transferring a maximum of 9600 bits per second.

UART: Universal asynchronous receiver-transmitter, is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. A UART is usually an individual (or part of an) integrated circuit (IC) used for serial communications over a computer or peripheral device serial port.

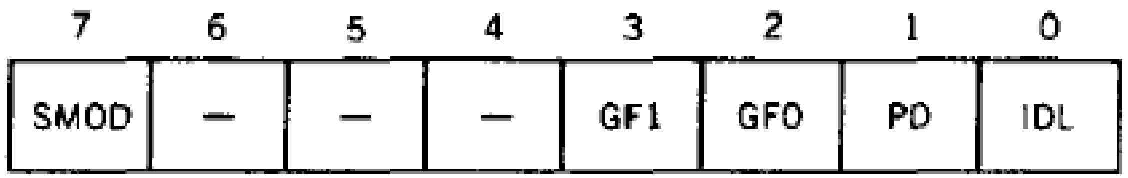
Serial Data Input / Output

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

5	SM2	Multiprocessor communications bit. Set/cleared by program to enable multiprocessor communications in modes 2 and 3. When set to 1 an interrupt is generated if bit 9 of the received data is a 1; no interrupt is generated if bit 9 is a 0. If set to 1 for mode 1, no interrupt will be generated unless a valid stop bit is received. Clear to 0 if mode 0 is in use.
4	REN	Receive enable bit. Set to 1 to enable reception; cleared to 0 to dissable reception.
3	TB8	Transmitted bit 8. Set/cleared by program in modes 2 and 3.
2	RB8	Received bit 8. Bit 8 of received data in modes 2 and 3; stop bit in mode 1. Not used in mode 0.
1	TI	Transmit interrupt flag. Set to one at the end of bit 7 time in mode 0, and at the beginning of the stop bit for other modes. Must be cleared by the program.
0	RI	Receive interrupt flag. Set to one at the end of bit 7 time in mode 0, and halfway through the stop bit for other modes. Must be cleared by the program.

Bit addressable as SCON.0 to SCON.7

THE POWER MODE CONTROL (PCON) SPECIAL FUNCTION REGISTER



7	6	5	4	3	2	1	0
SMOD	—	—	—	GF1	GF0	PD	IDL

Bit	Symbol	Function
7	SMOD	Serial baud rate modify bit. Set to 1 by program to double baud rate using timer 1 for modes 1, 2, and 3. Cleared to 0 by program to use timer 1 baud rate.
6-4	—	Not implemented.
3	GF1	General purpose user flag bit 1. Set/cleared by program.
2	GF0	General purpose user flag bit 0. Set/cleared by program.
1	PD	Power down bit. Set to 1 by program to enter power down configuration for CHMOS processors.
0	IDL	Idle mode bit. Set to 1 by program to enter idle mode configuration for CHMOS processors. PCON is not bit addressable.

CHMOS refers to one of a series of Intel CMOS processes developed from their HMOS process. (H stands for high-density). CHMOS was used in the Intel 80C51BH,

In **Power Down mode**, the oblillator clock provided to the system is OFF i.e. CPU and peripherals clock remains inactive in this mode.
In **Idle Mode,** only the clock provided to the CPU gets deactivated, whereas the peripherals clock will remain active in this mode. Hence power saved in power-down mode is more than in idle mode.

Serial Data Interrupts

- Serial data communication is a relatively slow process, occupying many milliseconds per data byte to accomplish.
- In order not to tie up valuable processor time, serial data flags are included in SCON to aid in efficient data transmission and reception.
- The serial data flags in SCON. TI and RI, are set whenever a data byte is transmitted (TI) or received (RI). These flags are ORed together to produce an interrupt to the program.
- The program must read these flags to determine which caused the interrupt and then clear the flag.

Data Transmission

- Transmission of serial data bits begins anytime data is written to SBUF. TI is set to a 1 when the data has been transmitted and signifies that SBUF is empty (for transmission purposes) and that another data byte can be sent.
- If the program fails to wait for the TI flag and overwrites SBUF while a previous data byte is in the process of being transmitted, the results will be unpredictable

Data Reception

- Reception of serial data will begin if the receive enable bit (REN) in SCON is set to 1 for all modes. In addition, for mode 0 only, R1 must be cleared to 0 also.
- Receiver interrupt flag RI is set after data has been received in all modes. Setting REN is the only direct program control that limits the reception of unexpected data; the requirement that RI also be 0 for mode 0 prevents the reception of new data until the program has dealt with the old data and reset RI.
- Reception can begin in modes 1, 2, and 3 if RI is set when the serial stream of bits begins. RI must have been reset by the program before the last bit is received or the incoming data will be lost.
- Incoming data is not transferred to SBUF until the last data bit has been received so that the previous transmission can be read from SBUF while new data is being received.

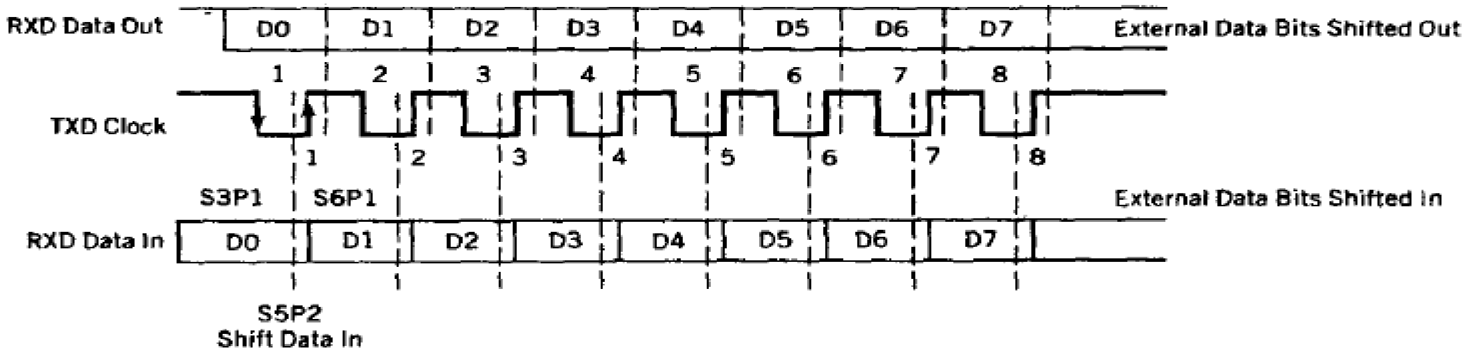
Serial Data Transmission Modes

- The 8051 designers have included four modes of serial data transmission that enable data communication to be done in a variety of ways and a multitude of baud rates.
- Modes are selected by the programmer by setting the mode bits SMO and SM 1 in SCON.
- Baud rates are fixed for mode 0 and variable, using timer 1 and the serial baud rate modify bit (SMOD) in PCON, for modes I. 2, and 3.

Serial Data Transmission Modes

❖ Serial Data Mode 0-Shift Register Mode

- Setting bits SMO and SM 1 in SCON to OOb configures SBUF to receive or transmit eight data bits using pin RXD for both functions. Pin TXD is connected to the internal shift frequency pulse source to supply shift pulses to external circuits.
- The shift frequency, or baud rate, is fixed at 1/12 of the oscillator frequency, the same rate used by the timers when in the timer configuration. The TXD shift clock is a square wave that is low for machine cycle states S3-S4-S5 and high for S6-S1-S2.

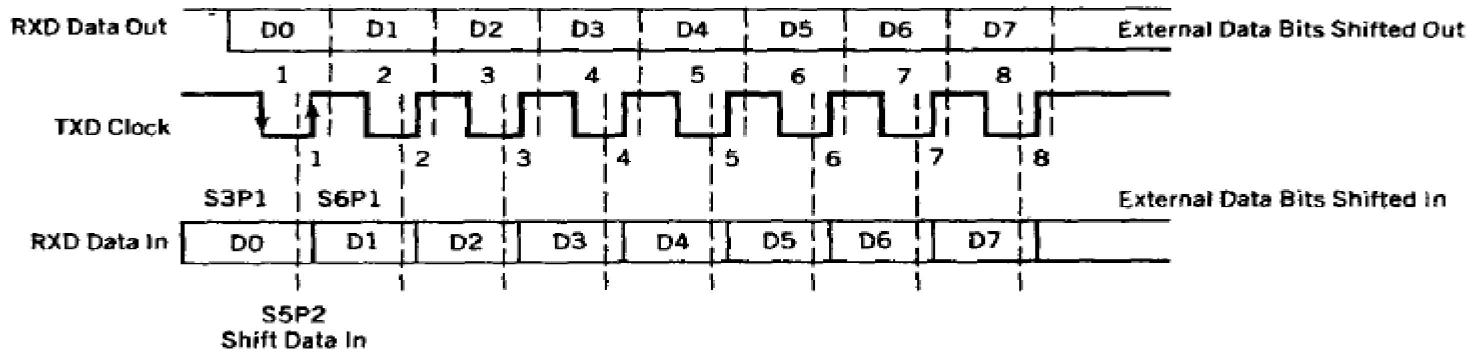


Serial Data Transmission Modes

❖ Serial Data Mode 0-Shift Register Mode

- Received data comes in on pin RXD and should be synchronized with the shift clock produced at TXD. Data is sampled on the falling edge of S5P2 and shifted in to SBUF on the rising edge of the shift clock.

- Mode 0 is intended not for data communication between computers, but as a high speed serial data-collection method using discrete logic to achieve high data rates. The baud rate used in mode 0 will be much higher than standard for any reasonable oscillator frequency; for a 6 MHz crystal, the shift rate will be 500 kHz.

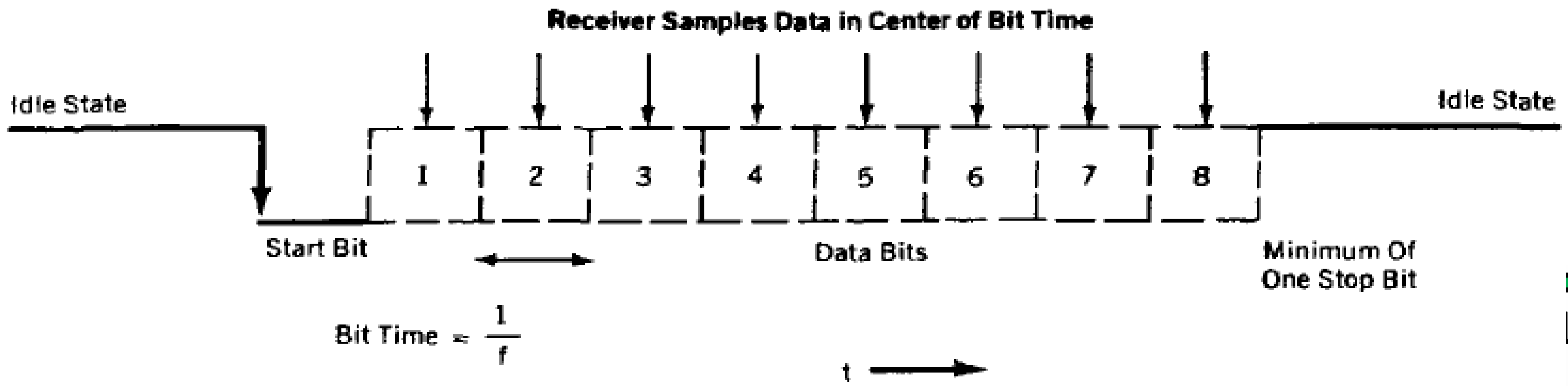


Serial Data Transmission Modes

❖**Serial Data Mode 1-**

1- Standard UART

- When SMO and SMI are set to O1b, SBUF 1 becomes a 10-bit full-duplex receiver/ transmitter that may receive and transmit data at the same time. Pin RXD receives all data, and pin TXD transmits all data. Figure shows the format of a data word.
- Transmitted data is sent as a start bit, eight data bits (Least Significant Bit, LSB,
- first), and a stop bit. Interrupt flag TI is set once all ten bits have been sent. Each bit interval is the inverse of the baud rate frequency, and each bit is maintained high or low over that interval.



Serial Data Transmission Modes

❖**Serial Data Mode 1-**

1- Standard UART

- Received data is obtained in the same order; reception is triggered by the falling edge of the start bit and continues if the start bit is true (0 level} halfway through the start bit interval. This is an anti-noise measure; if the reception circuit is triggered by noise on the transmission line, the check for a low after half a bit interval should limit false data reception.
- Data bits are shifted into the receiver at the programmed baud rate, and the data word will be loaded to SBUF if the following conditions are true:

1) RI must be 0, and mode bit SM2 is 0 or the stop bit is 1 (the normal state of stop bits).

Serial Data Transmission Modes

❖**Serial Data Mode 1-**

2- Baud Rates

- Timer 1 is used to generate the baud rate for mode 1 by using the overflow flag of the timer to determine the baud frequency. Typically. timer 1 is used in timer mode 2 as an auto load R-hit timer that generates the baud frequency:

$$f_{baud} = \frac{2^{SMOD}}{32d} \times \frac{\text{oscillator frequency}}{12d \times [256d - (TH1)]}$$

- SMOD is the control hit in PCON and can be 0 or 1. which raises the 2 in the equation to a value of I or 2.

If timer 1 is not run in timer mode 2, then the baud rate is

$$f_{baud} = \frac{2^{SMOD}}{32d} \times (\text{timer 1 overflow frequency})$$

Timer 1 can be run using the internal clock or as a counter that receives clock pulses from any external source via pin T1.

Example

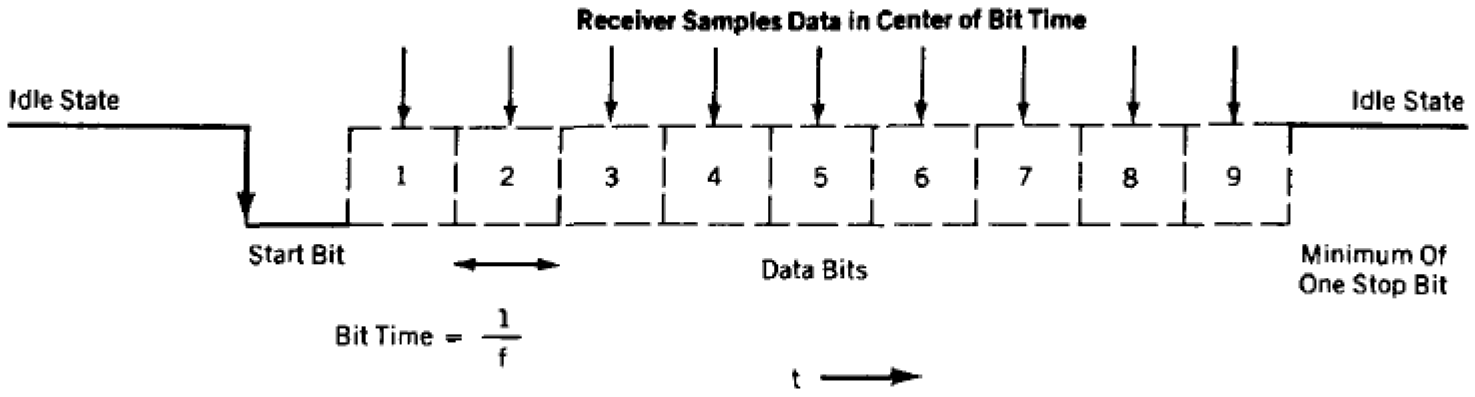
- If standard baud rates are desired to be 9600 hz , and the crystal frequency 11.0592 megahertz. What is the setting value of TH1

If SMOD is cleared to 0.

$$\mathrm{TH1} = 256d - \left(\frac{2^0}{32d} \times \frac{11.0592 \times 10^6}{12 \times 9600d} \right) = 253.0000d = 0FDh$$

Serial Data Transmission Modes

- **Serial Data Mode 2-Multiprocessor Mode**
  - Mode 2 is similar to mode 1 except 11 bits are transmitted: a start bit, nine data bits. and a stop bit, as shown in Figure 2. 16. The ninth data bit is gotten from bit TB8 in SCON during transmit and stored in bit RB8 of SCON when data is received. Both the start and stop bits are discarded.
  - The baud rate is programmed as follows: $f_{baud2} = \frac{2^{SMOD}}{64d} \times \text{oscillator frequency}$



Serial Data Transmission Modes

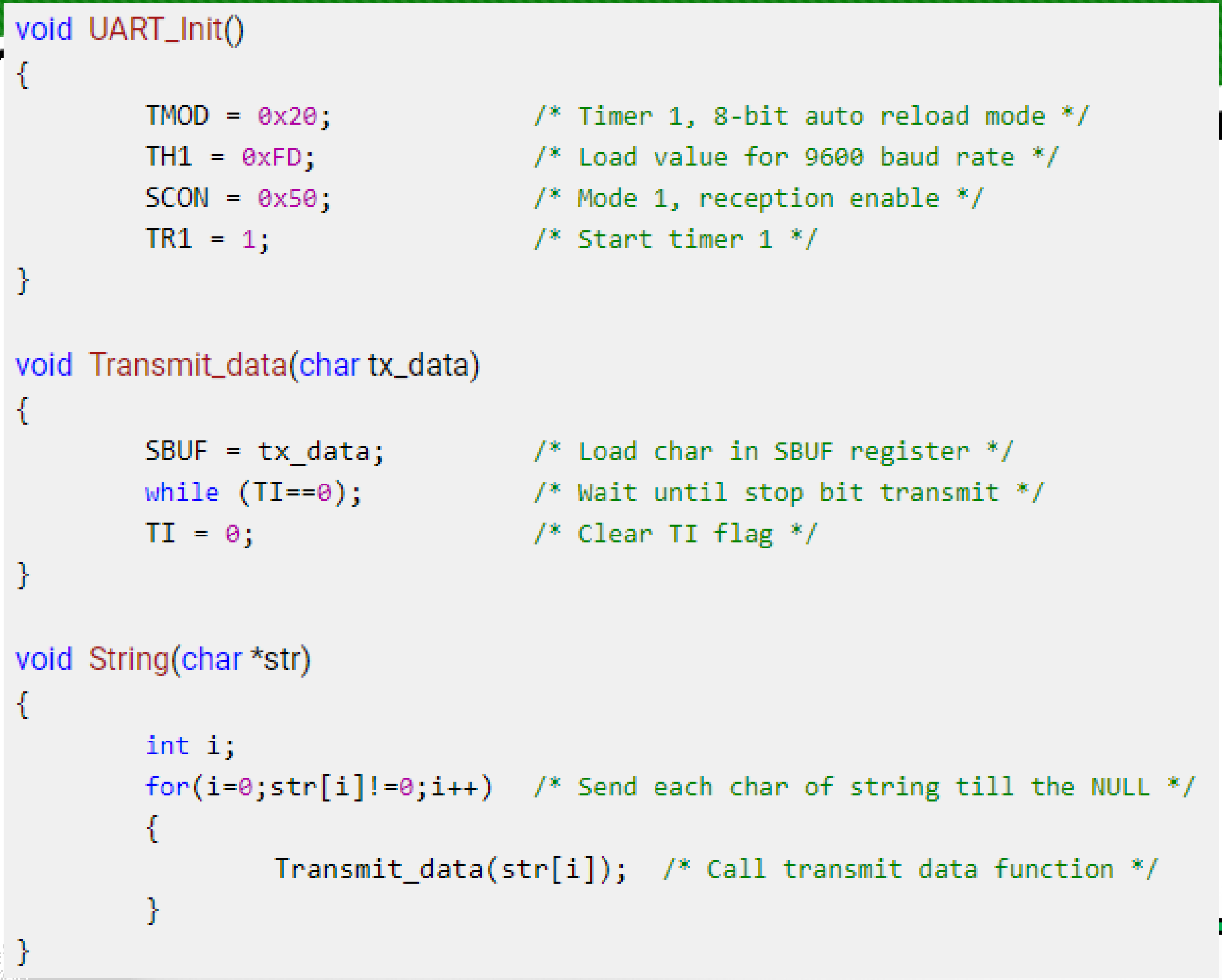
- **Serial Data Mode 2-Multiprocessor Mode**
  - As in the case for mode 0, the baud rate is much higher than standard communication rates. This high data rate is needed in many multi-processor applications. Data can be collected quickly from an extensive network of communicating microcontrollers if high baud rates are employed.
  - The conditions for setting RI for mode 2 are similar to mode 1
- **Serial Data Mode 3**

Mode 3 is identical to mode 2 except that the baud rate is determined exactly as in mode 1 , using Timer 1 to generate communication frequencies.

Programming steps

1. Configure Timer 1 in auto-reload mode.
2. Load TH1 with value as per required baud rate e.g. for 9600 baud rate load 0xFD. (-3 in decimal)
3. Load SCON with serial mode and control bits. e.g. for mode 1 and enable reception, load 0x50.
4. Start timer1 by setting TR1 bit to 1.
5. Load transmitting data in the SBUF register.
6. Wait until loaded data is completely transmitted by polling the T1 flag.
7. When the T1 flag is set, clear it, and repeat from step 5 to transmit more data.

Let's Program 8051 (here AT89C51) to send character data “test” serially at 9600 baud rate in mode 1



```c
void UART_Init()
{
        TMOD = 0x20;                /* Timer 1, 8-bit auto reload mode */
        TH1 = 0xFD;                 /* Load value for 9600 baud rate */
        SCON = 0x50;                /* Mode 1, reception enable */
        TR1 = 1;                    /* Start timer 1 */
}

void Transmit_data(char tx_data)
{
        SBUF = tx_data;             /* Load char in SBUF register */
        while (TI==0);              /* Wait until stop bit transmit */
        TI = 0;                     /* Clear TI flag */
}

void String(char *str)
{
        int i;
        for(i=0;str[i]!=0;i++)      /* Send each char of string till the NULL */
        {
                Transmit_data(str[i]);  /* Call transmit data function */
        }
}
```