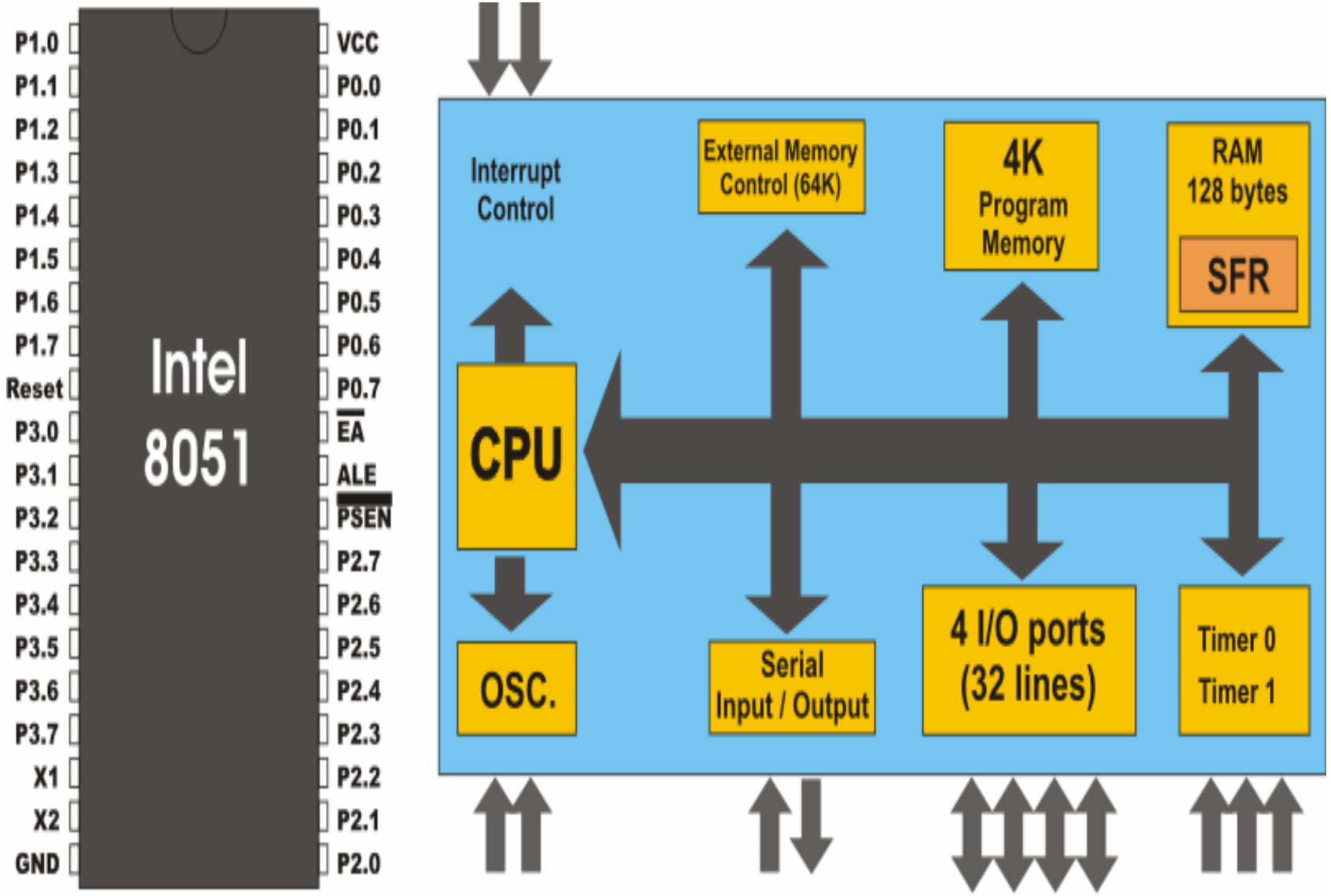
8051 Microcontroller Programming



ADDRESSING MODES: -

An "addressing mode" refers to how you are addressing a given memory location. In summary, the addressing modes are as follows, with an example of each:

Immediate Addressing

MOV A, #20H MOV R4, #64H MOV B, #40H

Register Addressing

MOV A, R0 MOV R2, A ADD A, R7

Direct Addressing

MOV A, 30H MOV 56H, A MOV B, 04H

Register Indirect Addressing

MOV A, @R0 MOV @R1, B

Indexed Addressing

Push 05 = Push R5 Pop 02 = Pop R2

Program status (PSW) word of 8051: -

Cy Carry Ac Auxiliary carry P Parity

Cy	Ac	x	RS1	RS0	OV	x	P

RS1	RS0	
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

The user may make use of 128 byte memory variables with commands such as SETB and CLR.

For example, to set bit number 24 (hex) to 1 you would execute the instruction:
SETB 24h

DPTR (Data pointer):

In order to access the external memory(usually upto 64k) data , a 16bit accessing register is needed for 8051.

Usually the address of the data in external memory is loaded or stored in DPTR which points or directs to that data. And this data can be accessed using indirect addressing (i.e @DPTR).

Ex: MOVX A, @DPTR

It means that the data stored in the address specified or loaded in DPTR is moved or loaded into A register or Accumulator.

Data Type and Directive: -

DB (Define Byte): -

To define 8 bit data. It can be decimal, hex, binary or ASCII. For decimal ‘D’ is optional at the end. But ‘B’ or ‘H’ is mandatory. DB is used to define ASCII string larger then two character.

Ex: -

```
        ORG   500H
Data1:  DB    28          in decimal
Data2:  DB    00101010B   in binary
Data3:  DB    39H         in hex
```

Data Type and Directive: -

ORG (origin): - Used to indicate the start of the address of the program. It can be either in hex or decimal.

```
ORG  518        In decimal
ORG  518H       In Hex
```

EQU (equate): -

Used to define a constant without occupying a memory location. By the use of EQU programmer can change it once and the assembler will change all its occurrence.

Ex: -

```
        COUNT           EQU  25
        ------          ----    --
        ------          ----    --
        MOV             R3, # COUNT
```

END: -

Used to indicate the end of the program.

Ex: Show the contents of RAM after the following program: -

```
MOV R0, # 99H
MOV R2, # 3FH
```

Ex: - Show the stack and stack pointer after the following. (initially SP =07)

```
MOV R6, #25H
MOV R1, #12H
PUSH    06
PUSH    01
```

Ex: - Show the stack and stack pointer after the following. (initially SP =0B)

```
POP  03
POP  02
POP  05
```

LOOP and JUMP Instruction in 8051: -

Loop instruction: -

DJNZregister label (DJNZ B, BACK)

This format is used to make the loop.

Ex: - Multiply 25 by 10 times.

```
                ORG 0500
                MOV A, #00H
                MOV R2, #10
AGAIN:          ADD A, # 25
                DJNZ R2, AGAIN
                MOV R5, A
                END
```

There are three types of jump instruction in 8051

LJMP, SJMP and AJMP

Syntax: LJMP address

The program execution directly jumps to the specified address. This instruction can be used for 64Kb of memory. It has a range of 65,536 addresses, and can jump over these many addresses. So named Long jump.

syntax: SJMP Raddress;

The range is -128 to 127. It means we can use jump only for ahead or behind 128 jumps.

syntax: AJMP address;

Whole 64Kb memory is divided into 32 parts each consisting of 2KB of memory. Each page consists of 2048 locations.

LOOP and JUMP Instruction in 8051: -

Ex: - WAP to load Accu. With 55H and complement the Accu. 700 times.

```
                MOV A, #55H
                MOV R3, #10
NEXT            MOV R2, #70
AGAIN           CPL A
                DJNZ R2, AGAIN
                DJNZ R3, Next
                END
```

Conditional & Unconditional jump in 8051: -

***Conditional jump:* -**

Ex: -	JNC	jump if no carry CY=0.
	JZ	jump if A=0.

***Un-Conditional jump:* -** in 8051 there are two unconditional jump LJMP and SJMP.

LJMP is 3 byte instruction. First byte is opcode second and third byte is the 16 bit address of the memory.

SJMP is two byte instruction. First byte is opcode and second byte is the address of the target (00H to FFH). It is divided into forward and backward jump. Forward means 127 bytes from the current PC. Backward means -128 bytes from the current PC.

CALL instruction in 8051: -

LCALL (Long call): - Three byte instruction. First byte opcode second and third byte is used for address of the target subroutine. (Any where in the 64 K Byte). In the last RET is used to return back to the main program.

Ex: - WAP to toggle all the bits of port 1by sending 55H and AAH. Put delay also.

```
        ORG 0000H
BACK:   MOV A, #55H
        MOV P1, A
        LCALL DELAY
        MOV A, #0AAH
        MOV P1, A
        LCALL DELAY
        SJMP  BACK

        ORG 0300H
DELAY:  MOV R5, #0FFH
AGAIN:  DJNZ R5, AGAIN
        RET
        END
```

ACALL (Absolute call): - ACALL is two byte instruction. Target address should be within 2K of the PC.

Ex: - WAP to toggle all the bits of port 1by sending 55H and AAH. Put delay also.

```
        ORG 0000H
BACK:   MOV A, #55H
        MOV P1, A
        ACALL DELAY
        MOV A, #0AAH
        MOV P1, A
        ACALL DELAY
        SJMP  BACK

        ORG 0030H
DELAY:  MOV R5, #0FFH
AGAIN:   DJNZ R5, AGAIN
        RET
        END
```

Time delay in 8051: -

CPU takes certain numbers of clock cycle to execute an instruction. These clock are referred as machine cycle. Length of machine cycle depends on the applied crystal. Crystal can vary from 4MHz to 30MHz.

Ex: - Find the period of the machine cycle in each case.

1. 11.0592MHz 2. 16 MHz 3. 20 MHz

1. 11.0592/12 = 921.6 KHz; i.e machine cycle 1/921.6 KHz = 1.085 μs.
2. 16/12 = 1.333 MHz; i.e machine cycle 1/1.333 MHz = 0.75 μs.
3. 20/12 = 1.66 MHz; i.e machine cycle 1/1.66 MHz = 0.60 μs.

Baud rate?

The term “baud” originates from the French engineer Emile Baudot, who invented the 5-bit teletype code. Baud rate refers to the number of signal or symbol changes that occur per second. A symbol is one of several voltage, frequency, or phase changes.

The Baud rate refers to the **total number of signal units transmitted in one second**. The Bit rate refers to the total Bits transmitted in one unit time. Baud rate indicates the total number of times the overall state of a given signal changes/ alters. Bit rate indicates the total bits that travel per second.

Why we use 11.0592 MHz value crystal?

110,592,000,000 clock cycles/12 =9,216,000,000 cycle frequency

{12 clock pulse for one cycle/instruction} .

UART works with the help of auto-reload timer(mode 2)

Which will result in standard baud rates used like **19200, 9600, 4800, 2400** etc.

As these baud rates is achieved exactly by dividing the cycle frequency by whole number. (don't ask why timer works like that, timer will be explained later)

Ex: - For 8051 with 11.0592 MHz how long it take to execute each instruction.

Instruction	M/C cycle	Time to execute
MOV R3, #55H	1	1x1.085 μs = 1.085 μs.
LJMP	2	2x1.085 μs = 2.17 μs.
MUL A B	4	4x1.085 μs = 4.34 μs.

Ex: - For 8051 with 11.0592 MHz find the time required to execute the program.

		Machine cycle
DELAY:	MOV R3, #250	1
HERE:	NOP	1
	NOP	1
	NOP	1
	DJNZ R3, HERE	2
	RET	2

So time required for loop [250x(1+1+1+2)]x1.085 μs = 1500x1.085 μs = 1627.5 μs.

For program 1627.5 μs + 3x1.085 μs = 1630.755 μs

Ex: - Write the following program: -

1. Create a square wave of 50% duty cycle on bit 0 of port 1

2. Create a square wave of 66% duty cycle on bit 3 of port 1

```
1.    HERE:   SETB P1.0
              LCALL DELAY
              CLR P1.0
               LCALL DELAY
              SJMP HERE

2.    BACK:   SETB P1.3
              LCALL DELAY
              LCALL DELAY
              CLR P1.3
              LCALL DELAY
              SJMP BACK
```

Jump Instruction

```
JB P1.0  HERE
JNB P1.0 HERE
```

Addition of two 8 bit numbers

```
        MOV     R0, #20H        ;set source address 20H to R0
        MOV     R1, #30H        ;set destination address 30H to R1
        MOV     A,@R0           ; take the value from source to register A
        MOV     R5,A            ; Move the value from A to R5
        MOV     R4,#00H         ; Clear register R4 to store carry
        INC     R0              ; Point to the next location
        MOV     A,@R0           ; take the value from source to register A
        ADD     A,R5            ;Add R5 with A and store to register A
        JNC     SAVE
        INC     R4              ; Increment R4 to get carry
        MOV     B,R4            ;Get carry to register B
        MOV     @R1,B           ; Store the carry first
        INC     R1              ; Increase R1 to point to the next
SAVE:   MOV@R1,A                ;Store the result
HALT:   SJMP HALT               ;Stop the program
```

Ex: - Write a program to perform the : - Keep monitor P0.1 until it become high. If high read the data from port 1 and send a low to high pulse on P0.2.

```
        SETB P0.1          // P0.1 as an input
        MOV P1, #0FFH
AGAIN:  JNB P0.1, AGAIN
        MOV A, P1
        CLR P0.2
        SETB P0.2
```

Ex: - A switch is connected to P1.0 and an LED to pin P2.7. WAP to get the status of the switch and send it to the LED.

```
        SETB P1.7          //P1.7 as an input
AGAIN:  MOV C, P1.0
        MOV P2.7, C
        SJMP AGAIN
```

Ex: - Write a program to clear 16 RAM location starting at RAM address 60H

```
        CLR A
        MOV R1, #60H
        MOV R7, #16
AGAIN:  MOV @R1, A
        INC R1
        DJNZ R7, AGAIN
```

Ex: - Assume that ROM at 250H contain "Malik", WAP to transfer the bytes into RAM starting at 40H

```
        MOV DPTR, "MALIK"
        MOV R0, #40H
BACK:   CLR A
        MOVC A, @A+DPTR
        JZ HERE
        MOV @R0, A
        INC DPTR
        INC R0
        SJMP BACK
```

Instruction in 8051: -

```
ADD A, B
ADDC A, B          // With Carry
DA                 // Decimal Adjustment
SUBB A, B
MUL A, B           // Result in A & B
DIV A, B           // Quotient is in A Reminder in B
CPL A              // Complement A
ANL A
OR A
XRL A
CJNE A, B, BACK          // Compare and jump if equal
SWAP                     // Swap Accumulator Nibbles
XCH A, R0
```

Timer programming in 8051: -

There are two timers T0 16 bit (TH0 and TL0) and T1 16 bit (TH1 and TL1).

TMOD register (8 bit) : -

GATE	C/T	M1	M0	GATE	C/T	M1	M0

M1	M0		
0	0	13 bit timer mode.	Mode 0
0	1	16 bit timer mode.	Mode 1
1	0	8 bit auto reload.	Mode 2
1	1	Split timer mode.	Mode 3

Timer Register: -

TR0 and TR1, TF 0 and TF 1

Ex: - Assume crystal 11.0592, WAP to generate a time delay of 5 ms using timer 0 mode 1.

As crystal = 11.0592 thus machine cycle is 1.085µs. Now to generate delay of 5ms. 5ms/1.085µs = 4608 clocks. Thus we have to load into TL0 and TH0 Value 65536-4608 = 60928. As 60928 (decimal) = EE00 H. Thus TH0 = EE H and TL0 = 00 H.

```
        MOV TMOD, # 01
BACK    CLR P2.3
HERE:   MOV TL0, # 00
        MOV TH0, #0EEH
        SETB P2.3
        SETB TR0
AGAIN:  JNB TF0, AGAIN
        CLR TR0
        CLR TF0
        SJMP BACK
```

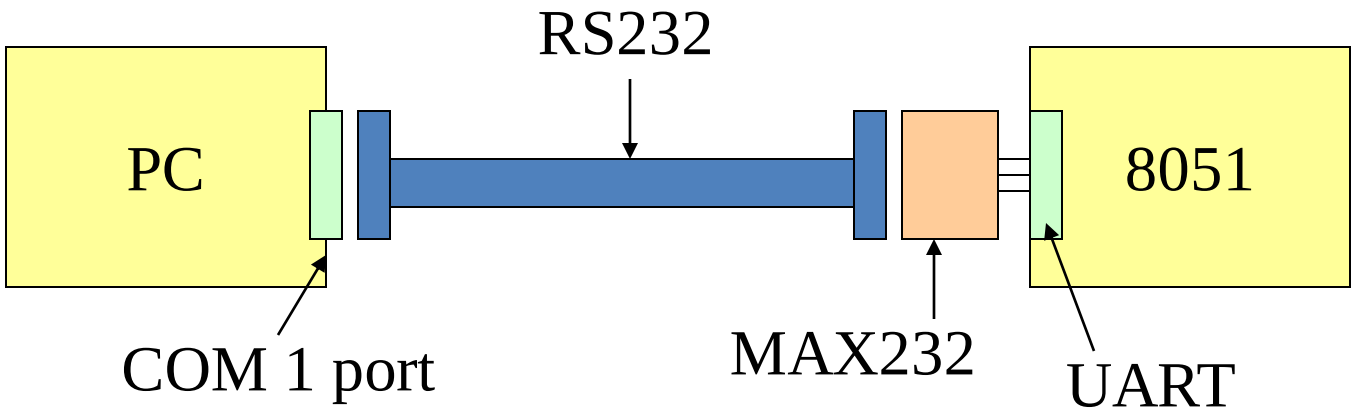
Ex: - Assume crystal 11.0592, find the frequency of a square wave generated on pin1.0 using timer 0 mode 2.

```
                MOV TMOD, # 02H
                MOV TH0, # 00H
AGAIN:          MOV R5, # 250          //count for multiply delay
                ACALL DELAY
                CPL P1.0
                SJMP AGAIN
DELAY:          SETB TR0               // start the timer 0
BACK:           JNB TF0, BACK
                CLR TR0
                CLR TF0
                DJNZ R5, DELAY
                RET
```

As crystal = 11.0592 thus machine cycle is 1.085µs.

T=2x(250x256x1.085µs) = 138.88 ns, and frequency = 72KHz

Serial Communication in 8051: -

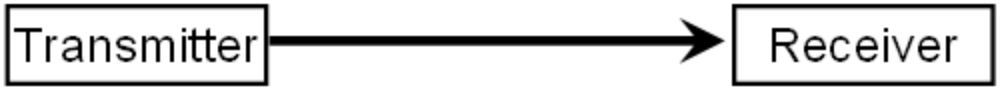


The 8051 module connects to PC by using RS232. RS232 is a protocol which supports half-duplex, synchronous/asynchronous, serial communication.

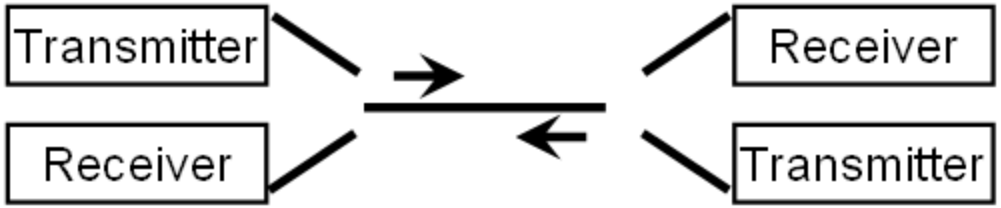
RS232 is a standard protocol used for serial communication, it is used for connecting computer and its peripheral devices to allow serial data exchange between them. As it obtains the voltage for the path used for the data exchange between the devices

Simples, Half Duplex and Full duplex: -

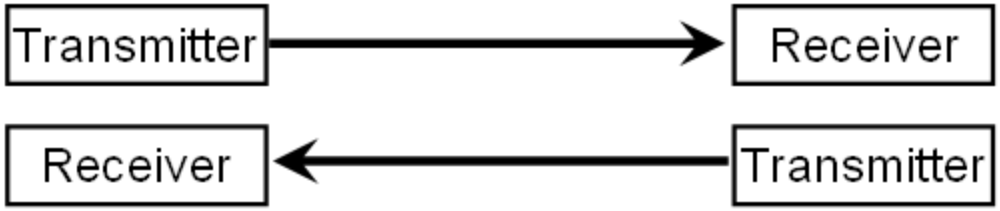
Simplex transmission



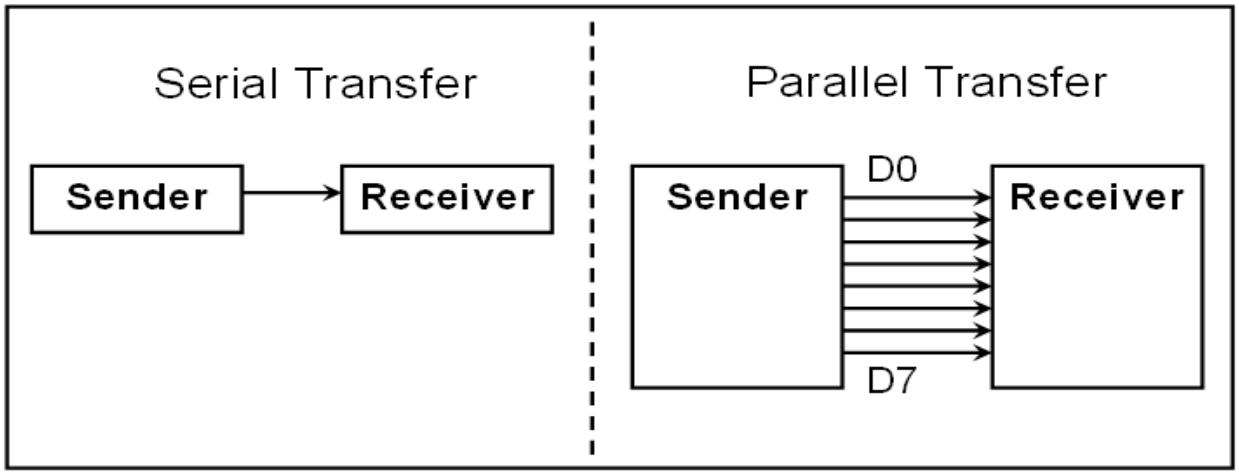
Half Duplex



Full Duplex

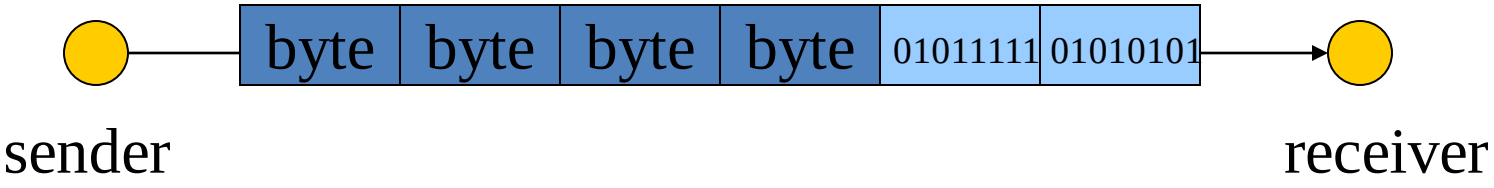


Serial and Parallel Transmission: -

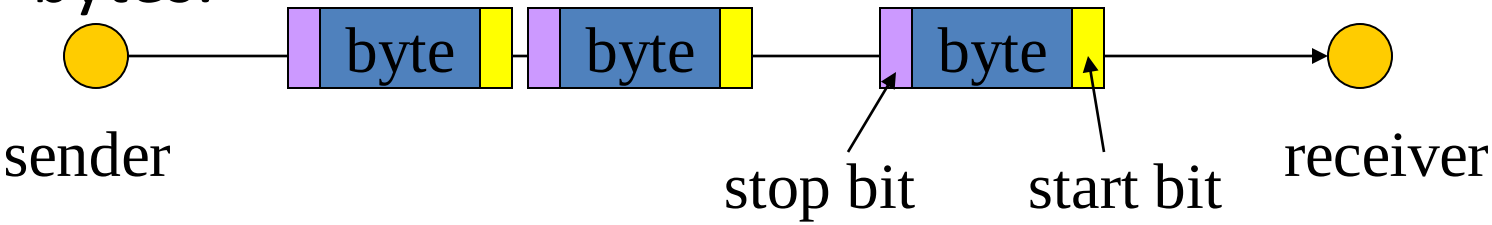


Asynchronous vs. Synchronous

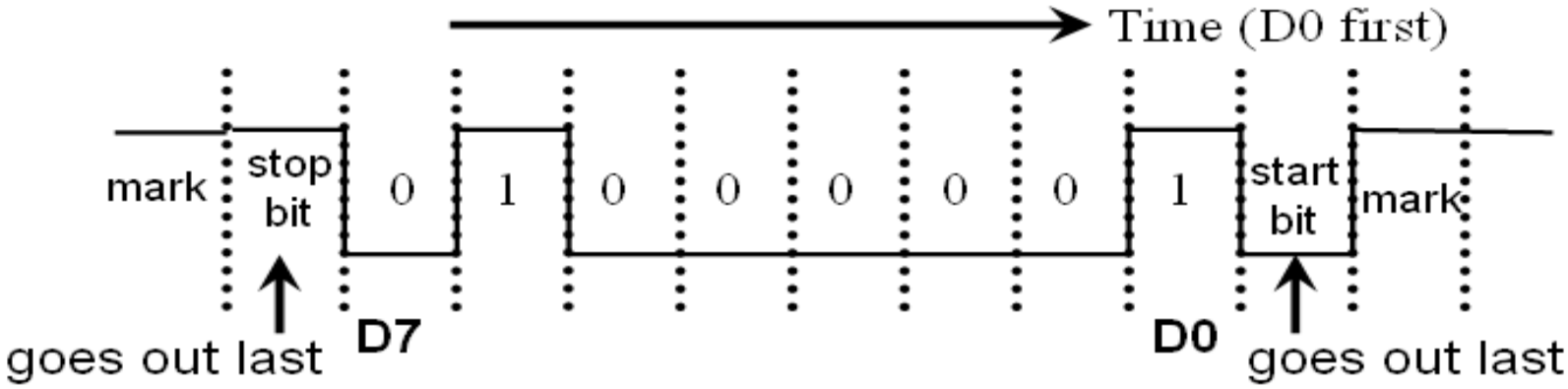
- Serial communication uses two methods:
  - In synchronous communication, data is sent in blocks of bytes.



  - In asynchronous communication, data is sent in bytes.



How data is transferred: -



- The LSB is sent out first.
- The start bit is 0 (low) and always one bit.
- The stop bits is 1 (high).
- The stop bit can be one (if 8 bits used in ASCII) or two bits (if 7 bits used in ASCII).
- When there is no transfer, the signal is 1 (high), which is referred to as mask.
- We have a total of 10 bits for each character:
  - 8-bits for the ASCII code
  - 2-bits for the start and stop bits

Data Transferred Rate: -

How fast is the data transferred?

Three methods to describe the speed:

Baud rate is defined as the number of signal changes per second.

The rate of data transfer is stated in *Hz* (used in modem).

Date rate is defined as the number of bits transferred per second.

Each signal has several voltage levels.

The rate of data transfer is stated in *bps* (bits per second).

Effective data rate is defined as the number of actual data bits transferred per second.

https://www.youtube.com/watch?v=fCRActJDR9U

https://www.youtube.com/watch?v=hPhMz9eWMC8

TxD and RxD pin in 8051: -

In 8051, the data is received from or transmitted to

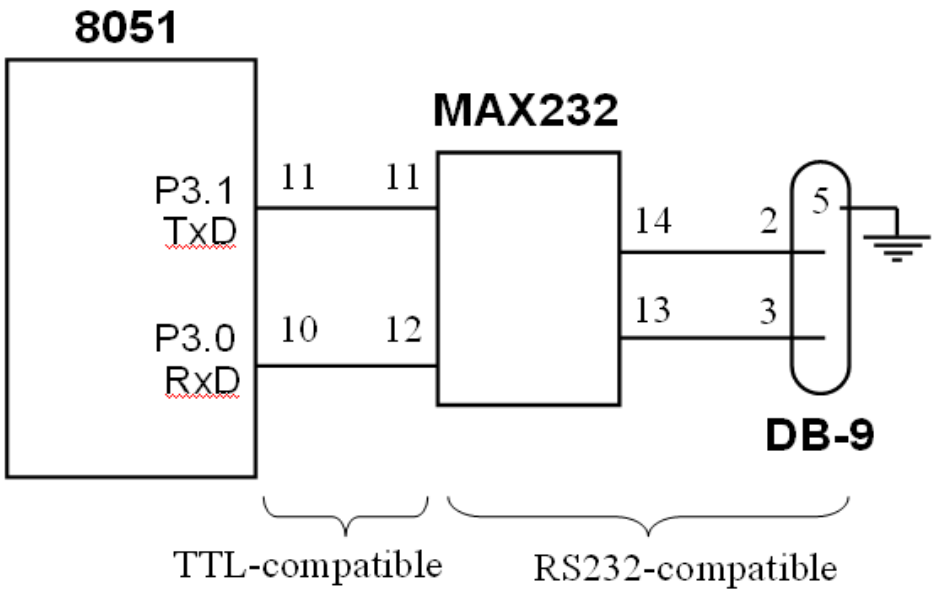
RxD: received data (Pin 10, P3.0)

TxD: transmitted data (Pin 11, P3.1)

TxD and RxD of the 8051 are TTL compatible.

The 8051 requires a line driver to make them RS232 compatible.

One such line driver is the MAX232 chip.



Serial Programming in 8051: - SCON Register

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. (Make it 0)
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 hardware at the beginning of the stop bit in mode 1. Must be cleared by software.
RI	SCON.0	Receive interrupt flag. Set by hardware halfway through the stop bit time in mode 1. Must be cleared by software.

Serial Programming in 8051: -

With XTAL = 11.0592 MHz, find the TH1 value needed to have the following baud rates. (a) 9600 (b) 2400 (c) 1200

Solution:

With XTAL = 11.0592 MHz, we have:
The frequency of system clock = 11.0592 MHz / 12 = 921.6 kHz
The frequency sent to timer 1 = 921.6 kHz/ 32 = 28,800 Hz
(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 1/12th 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

Note: XTAL = 11.0592 MHz.

Serial Programming in 8051: -

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

Note: XTAL = 11.0592 MHz.

The 9600 baud rate is a standard rate for serial communication. The configuration for 8051 microcontroller is divided as Timer 1 is used as Baud rate generator and set to Mode 2 (8-bit auto-reload) with the value of -3. This is because the value of the TH1 register is calculated using the formula TH1 = 256 - (crystal frequency/384/desired baud rate). So for 9600 baud rate, TH1 is calculated as 256 - (1.8432 MHz/192/9600) = 256 -3 = 253.

Serial Programming in 8051: -

Write a program for the 8051 to transfer letter "A" serially at 4800 baud, continuously.

Solution:

```
        MOV   TMOD,#20H    ;timer 1, mode 2
        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 be transferred
HERE:   JNB   TI,HERE      ;wait for the last bit
        CLR   TI           ;clear TI for next char
        SJMP  AGAIN        ;keep sending A
```

Disclaimer:

Proposed notes in the PPT are for your reference and for more detail description kindly refer the text books and reference books of the syllabus.

As per the new guidelines MTE/ETE question paper may be from the external agency. Therefore, it is requested to prepare yourself through the suggested reading materials in the syllabus also.