Serial Communication

- In serial communication we are transmitting the data/information bit after bit (only one bit goes through in a particular moment).

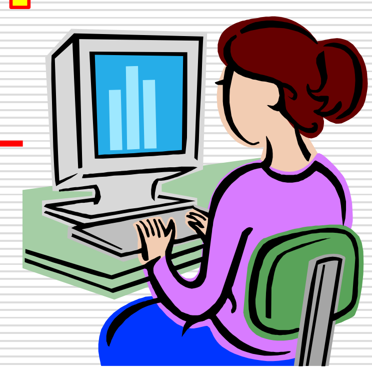
- In parallel communication we are transmitting a number of bits at once from one computer to the second computer.

<u>An example of serial communication</u>

For example: We want to transmit a word 10011101 using serial communication between two computers.

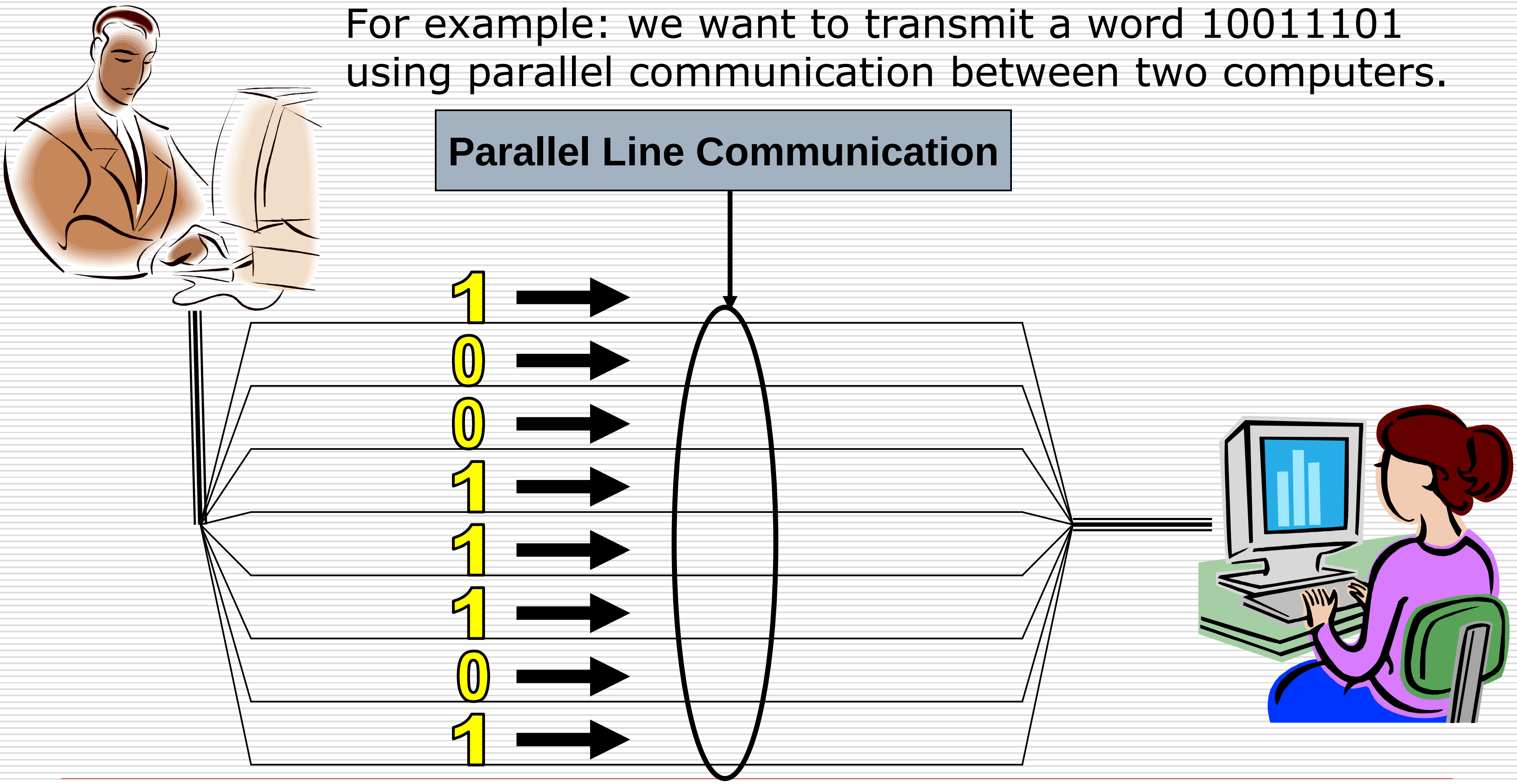
1 → 0 → 0 → 1 → 1 → 1 → 0 → 1

Serial line communication



An example of parallel communication

For example: we want to transmit a word 10011101 using parallel communication between two computers.



Synchronous/Asynchronous Communication

Synchronous communication:

The information is transmitted from the transmitter in sequence, bit after bit, with fixed baud rate, when the clock frequency along with the bits are transmitted to the receiver. This means that the transmitter and the receiver are synchronized between them by the same clock frequency.

Asynchronous communication:

The transmitter and the receiver refraining to broadcast long sequences of bits because there isn't a full synchronization between the transmitter that sends the data and the receiver that receives the data. In this case, the information is divided into frames, in the size of byte. Each one of the frame has a "Start" bit and a "Stop" bit. "Start" bit marks the beginning of a new frame, "Stop" bit marks the end. Frames of information must not necessarily be transmitted at equal time space, since they are independent of the clock.

REGISTER SETTINGS

- To allow serial communication with the PIC, we need to set different parameters within 2 registers:

TXSTA

Settings used for transmitting the data

RCSTA

Settings used for receiving the data

TXSTA REGISTER DEFINITION

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

ONLY in Half-Duplex
"1" = Master mode
"0" = Slave mode

Transmitted frame size:
"1" = Selects 9-bit transmission
"0" = Selects 8-bit transmission

"1" = Transmit enabled
"0" = Transmit disabled

"1" = Synchronous mode
"0" = Asynchronous mode

ONLY for asynchronous mode:
"1" = High speed
"0" = Low speed

"1" = TSR register is empty
"0" = TSR register is full

The place for 9th bit in case the transmitted data is 9 bits (can be parity bit)

RCSTA REGISTER DEFINITION

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Serial Port Enable bit
"1" = Serial port enabled
"0" = Serial port disabled

9-bit Receive Enable bit
"1" = Selects 9-bit reception
"0" = Selects 8-bit reception

ONLY Synchronous mode
and ONLY when PIC is Master:
"1" = Enables single receive
"0" = Disables single receive

ONLY in asynchronous mode:
"1" = Enables continuous receive
"0" = Disables continuous receive

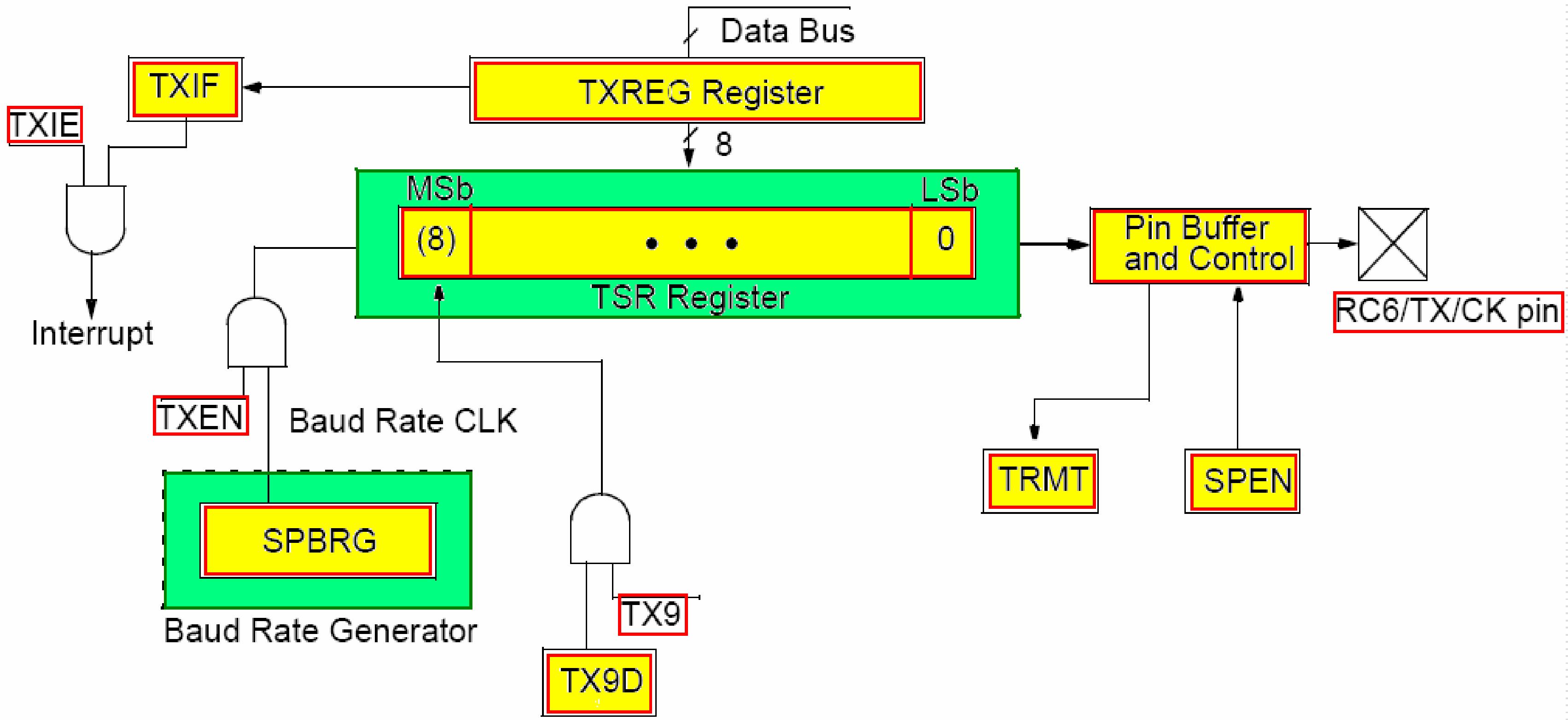
ONLY in asynchronous mode 9-bit:
"1" = Enables interrupt
"0" = Normal transmission

"1" = Framing error
"0" = No framing error

"1" = Overrun error
"0" = No overrun error

The place for the 9th
bit of received data

USART TRANSMIT BLOCK DIAGRAM



Registers and Control lines of the transmitter

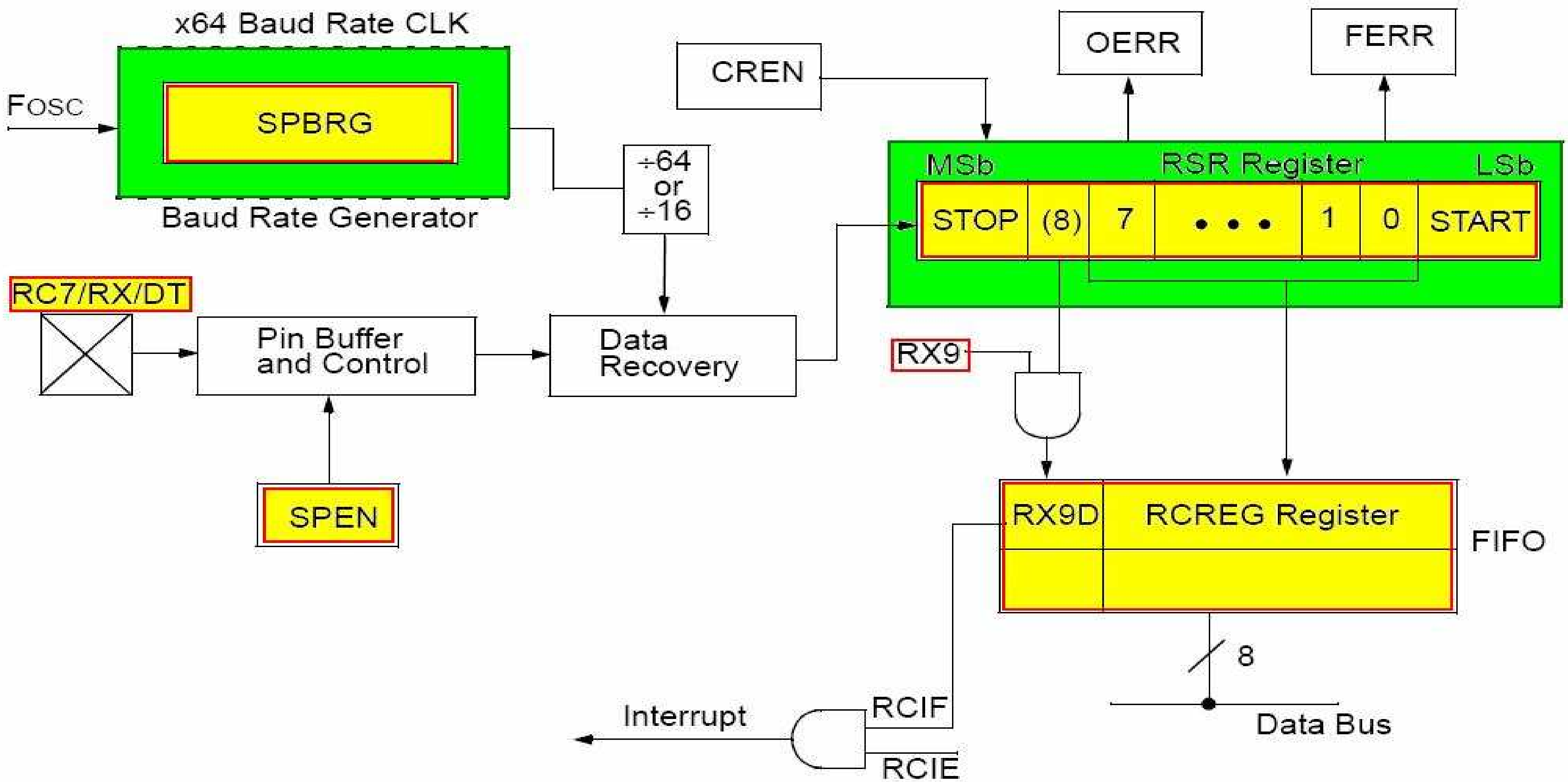
The information we want to transmit is loaded into register TXREG (8 bits size). In case the transmitted data is 9 bits long, the 9th bit is placed TX9D.

At the same time, the information above is being loaded into the register TSR, which is used as a temporary buffer before that information is transmitted.

TXIF – is set then TXREG is empty/free and ready to be loaded with a new information.
TXIE – is enabling the interrupt in the case TXREG is loaded/filled and TXIF = 1.

SPBRG – sets the desired baud rate in the system.
TXEN – is enabling the SPBRG.

USART RECEIVE BLOCK DIAGRAM



Registers and Control lines of the receiver

The received information is stored in the register RSR.

↓

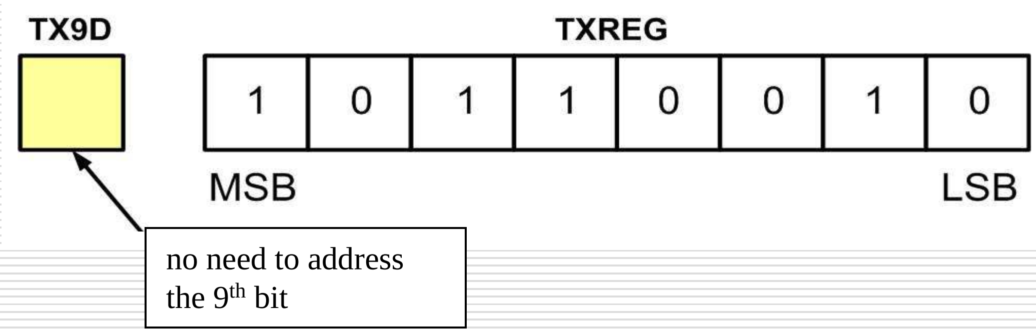
After receiving the data in the register RSR, the information is loaded at the same time into the register RCREG (8-bit size). In case the received data is 9-bit long, the 9th bit goes into RX9D.

CREN - continuous receive enable bit

Example: Transmission of 8-bit information

We wish to transmit the following information: 10110010.

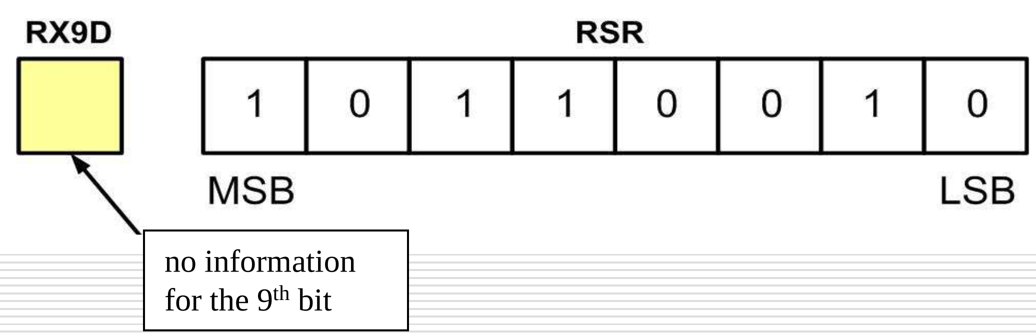
This information will be stored before transmission inside the register TXREG. The TX9 = 0.



We wish to receive the following information: 10110010.

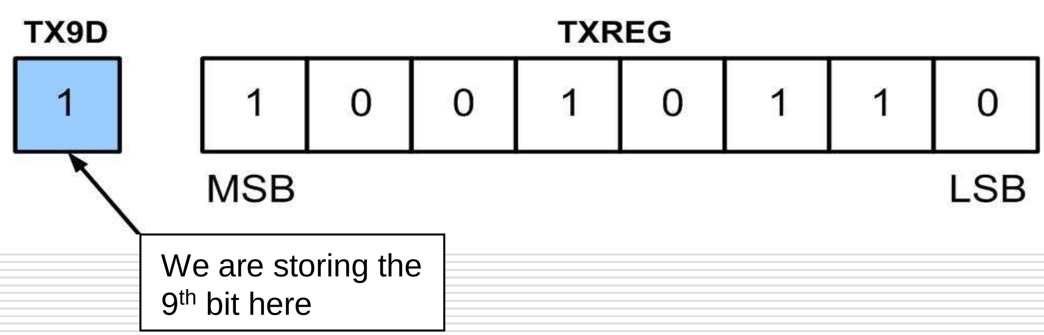
RX9 will be set to zero (RX9=0)

The information will be store temporary inside the RSR buffer.

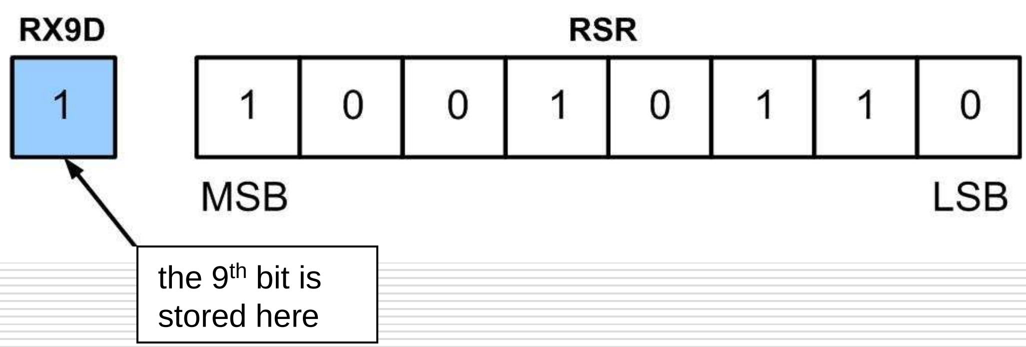


Example – transmission of 9-bit information

We wish to transmit the following information:110010110. The 8 bits (LSB) will be stored inside the buffer TXREG, and 9^{th} bit (MSB) will be stored inside temporary register TX9D. The TX9 will be set to 1 (TX9=1).



We wish to receive the following information: 110010110.
The RX9 will be set to 1 (RX9=1).



Baud rate

BAUD - baud rate
bps - units in which we are measuring pace of transmission

- To set desirable baud rate (for example 1200 bps), it is necessary to determine a new value of a clock system. The value of the clock will be determined by the hexadecimal number inserted into register SPBRG.
- The PIC can transmit at a high rate: BRGH=1 or at a low rate: BRGH=0. The calculation of the hexadecimal number inserted into register SPBRG done using the following formulas:

SPBRG = (Fosc/(16 x Baud rate)) - 1, BRGH=1 – High Speed

SPBRG = (Fosc/(64 x Baud rate)) - 1, BRGH=0 – Low Speed

Example - calculating the value of the register SPBRG

Let’s calculate the hexadecimal value to be insert into the register SPBRG, to get a transmission baud rate of 1200 bps at a lower rate.

We need to use the formula for a LOW SPEED:

SPBRG = (Fosc/(64 x Baud rate)) - 1,   BRGH=0 – Low Speed

SPBRG = (4MHz/(64x1200bps))-1= 51.08

The value need to be as a whole number (no decimal point), thus the value of SPBRG = 51.

Tables for values of SPBRG register

BRGH = 0

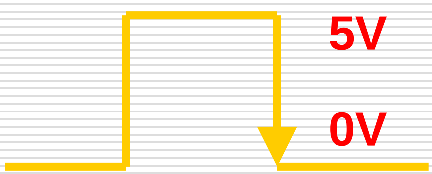
BAUD RATE (K)	Fosc = 4 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.300	0	207
1.2	1.202	0.17	51
2.4	2.404	0.17	25
9.6	8.929	6.99	6
19.2	20.833	8.51	2
28.8	31.250	8.51	1
33.6	-	-	-
57.6	62.500	8.51	0
HIGH	0.244	-	255
LOW	62.500	-	0

BRGH = 1

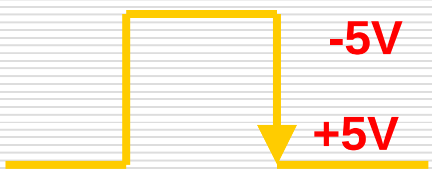
BAUD RATE (K)	Fosc = 4 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-
1.2	1.202	0.17	207
2.4	2.404	0.17	103
9.6	9.615	0.16	25
19.2	19.231	0.16	12
28.8	27.798	3.55	8
33.6	35.714	6.29	6
57.6	62.500	8.51	3
HIGH	0.977	-	255
LOW	250.000	-	0

Characteristics of USART and RSR232

- USART – is used for transmitting or receiving information. USART uses (0v) and (5v) signals to represent logical level.



- RSR232 - is a transmission protocol. RSR232 uses voltages lower than (-5v) and higher than (+5v) levels for the logical representation. The logical representation is opposite of voltage levels.



Level converter – MAX232 Driver / Receiver

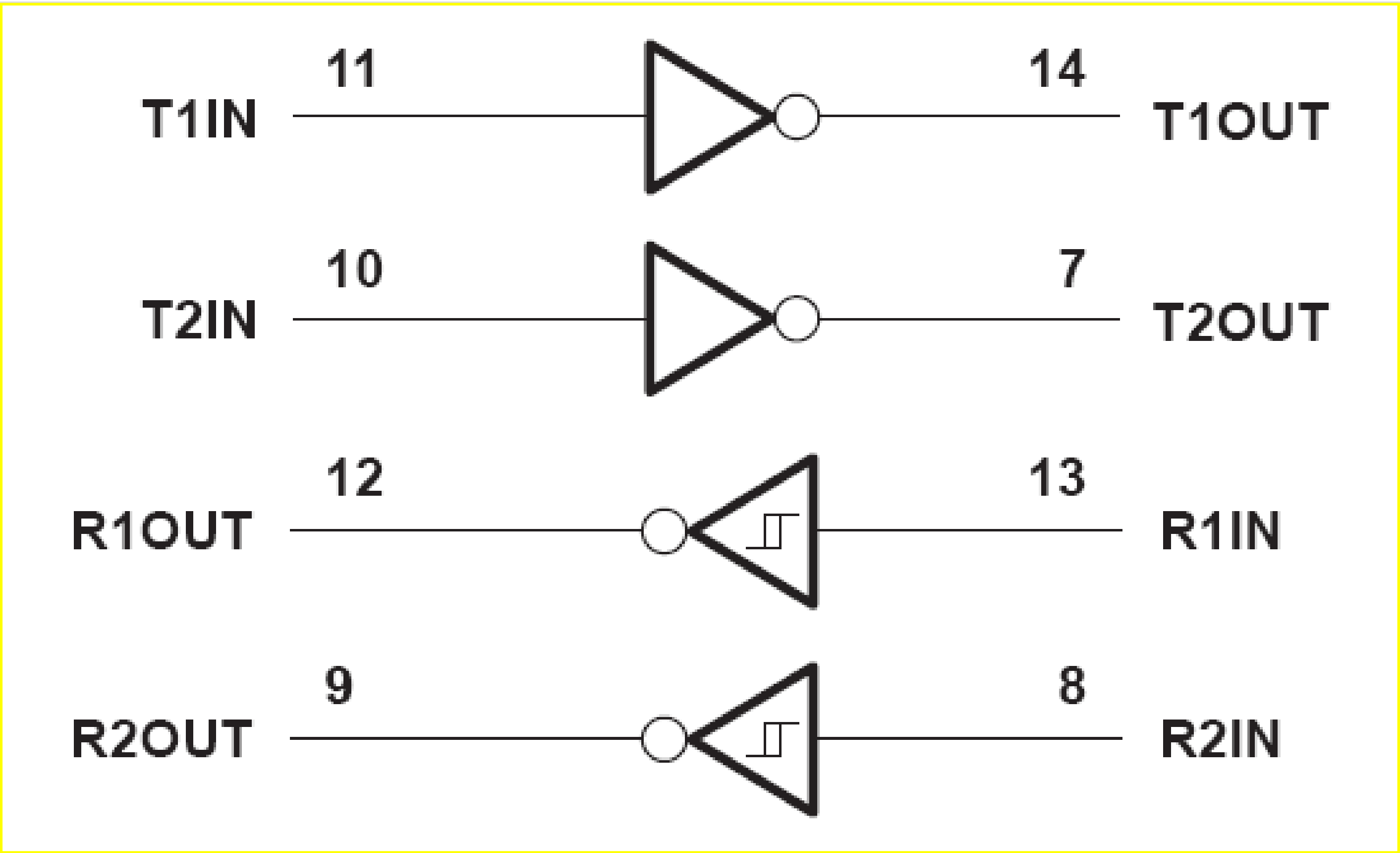
In order to transmit from the PIC to the computer we need to add another component, which allows to perform transmission in the RS232 protocol, and will be enable to convert voltage levels between the PIC and the PC (as we have seen in the previous slide, the two components use different levels of voltage).

↓

Level-voltage conversion is done with level converter MAX232.

MAX232 - is a simple component, which operates on a single 5v input, and has a set of 2 converters inside single chip.

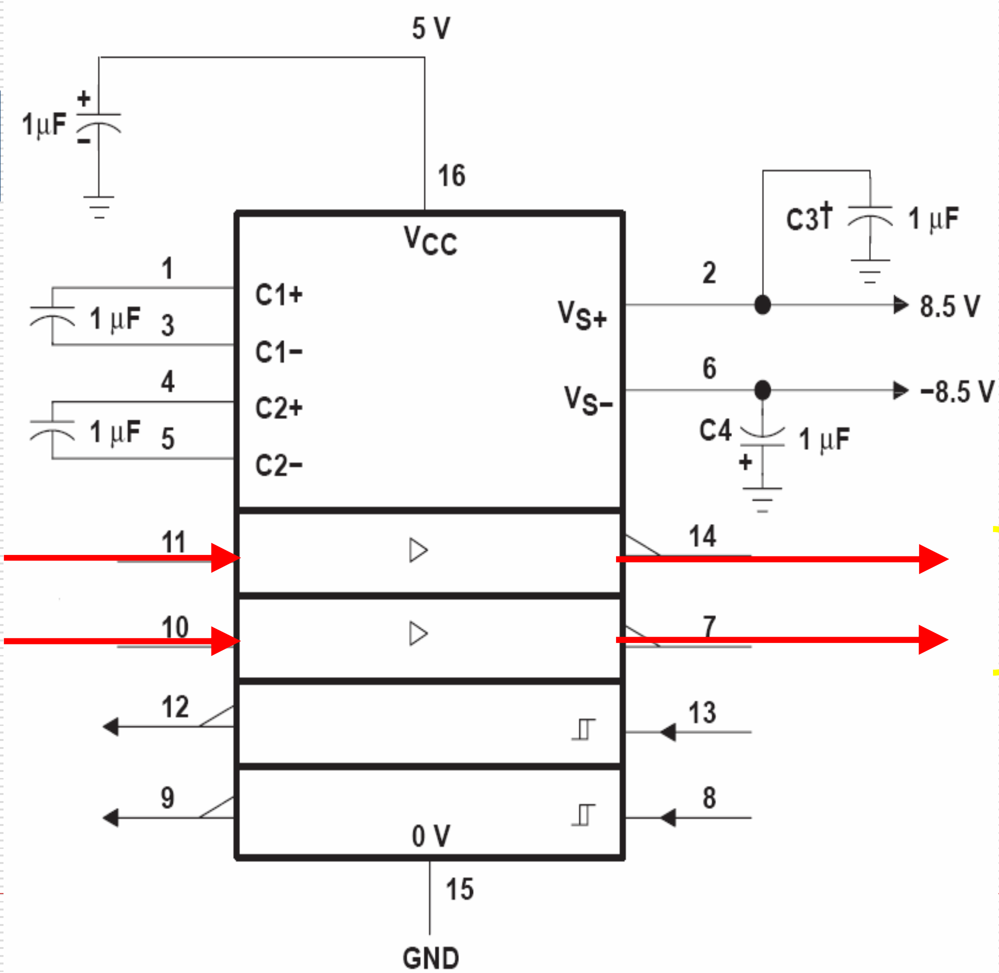
MAX232 – schematic structure



MAX232 connection form - transmitting from USART to the PC

The information comes from USART

Connect the output of the USART (transmitted information)

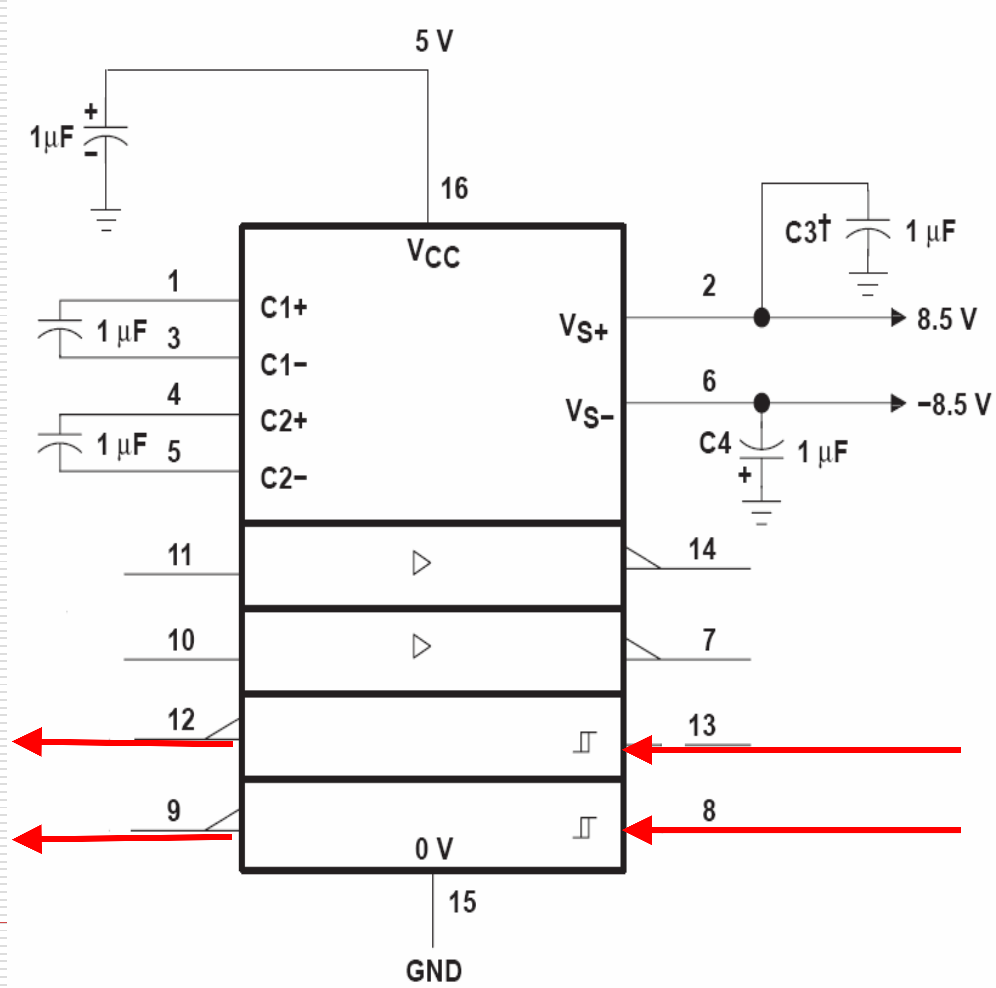


Levels of information are converted to the new voltage values to meet the RS232 protocol

The information is transmitted to the PC

MAX232 connection form - transmitting from the PC to USART

The information is transmitted to USART



The information that comes from the PC

Levels of information are converted to new voltage values that are suitable for USART

The information transmitted from the PC

Program Example

- The program will transmit information from USART to USART and will turn on appropriate LEDs according to the information received. The transmitted information will be a binary number, then the initial value is set to zero. The binary number will be incremented by one each time.

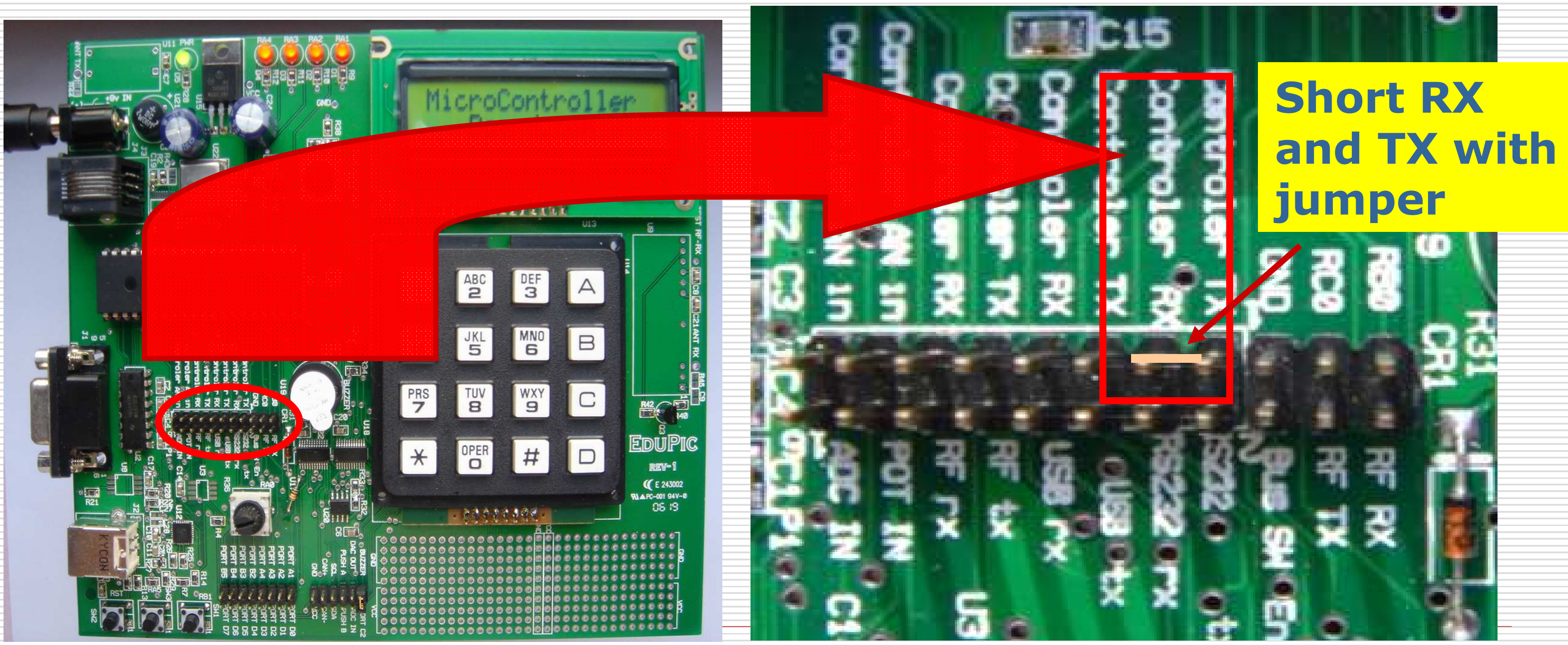
- To build and to implement our program we will us PIC microcontroller development board - EduPIC.

- By looking at the schematics of the transmitter and receiver, we see that the information is being transmitted through pin RC6 and being received through pin RC7.

  Thus in order to transmit the information and to receive the information using the SAME EduPIC, we need to short the two pins.

Program Example - continued

The following photo shows the EduPIC development board, where we've shorten between the pins RC6-RC7 using a jumper.



Program Example - continued

```
void main (void)
{
    TRISB = 0;                  // initializing PORT B as an output
    PORTB = 0;                  // zeroing out PORT B
    SPBRG = 51;                 // the hex value selected from the table
    TXSTA = 0b00100010;         // determining the settings for the transmitter
    RCSTA = 0b10010000;         // determining the settings for the receiver
    TXREG = 0x0;                // initializing the binary value of the transmitted information

    do                          // beginning of the endless loos from "do" to "while(1)"
    {
        TXREG++;                // increasing the TXREG by one
        while (!TRMT);          // waiting for a whole data frame to be ready for a transmission
        while (!RCIF); // waiting for a whole data frame to be received
        PORTB=RCREG;            // the received data is sent to PORT B
        for (i=0;  i< 300;  i++);  // delay in order to identify the change by looking at the LEDs
     }
    while(1);
}
```

For more information please visit:

MicrocontrollerBoard.com