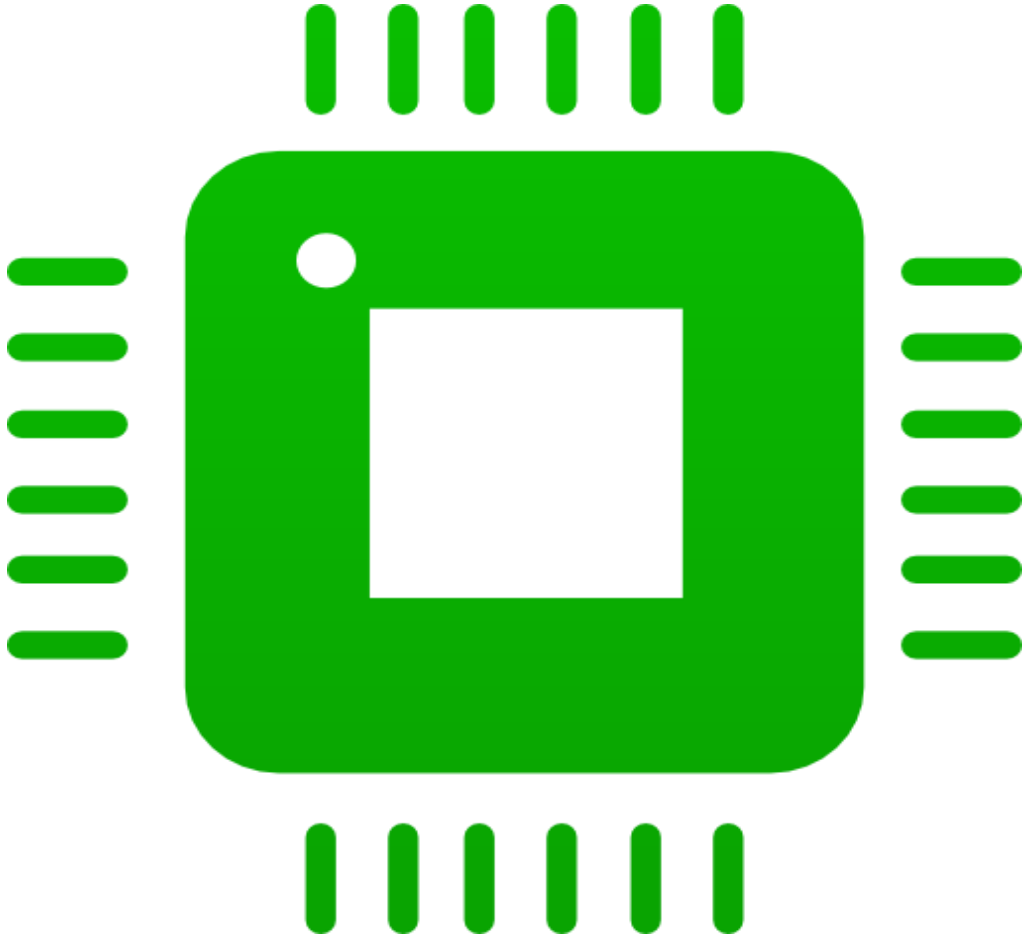
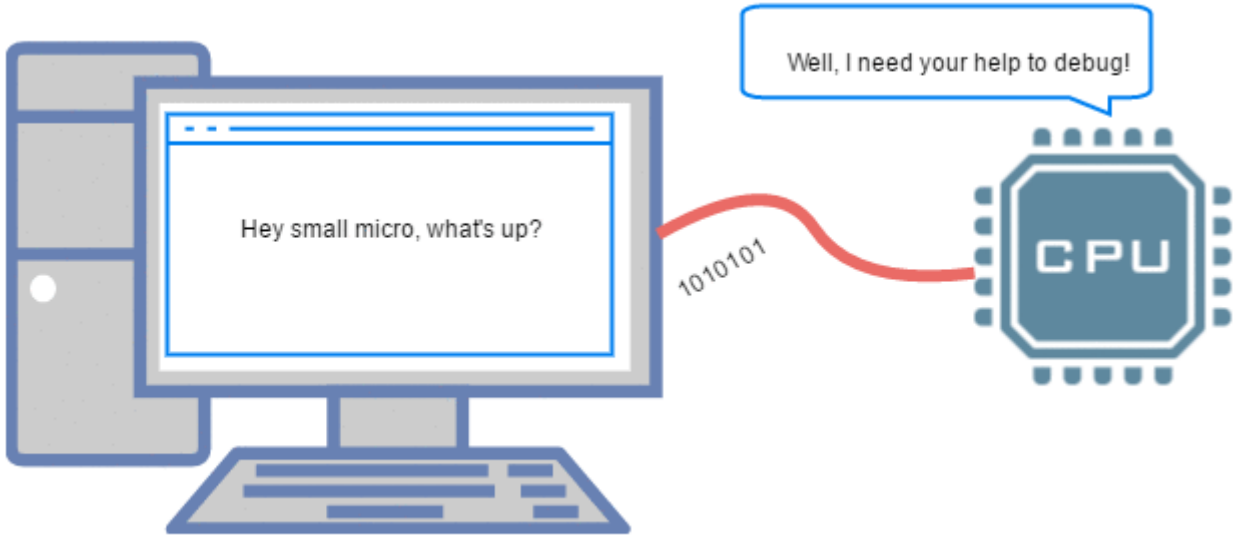
Fab Lab Ismailia represent :
Embedded Systems Workshop
by : Mohammed hemed



4- UART

Serial communication

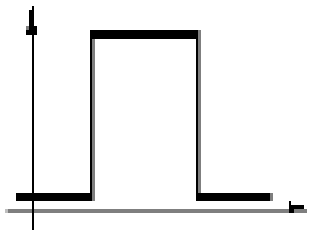
Universal Asynchronous Receiver Transmitter Protocol

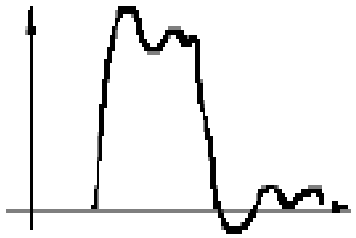


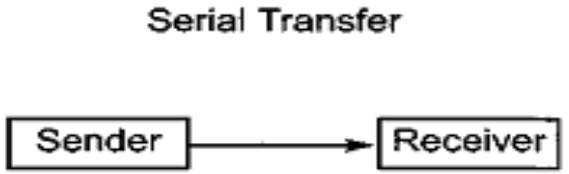
Serial vs Parallel communication

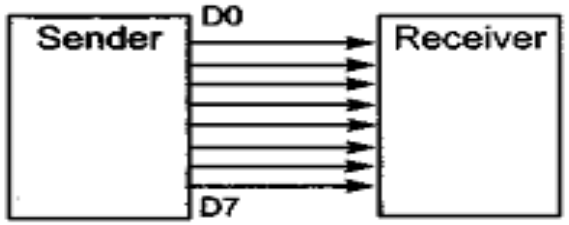
When uC communicate with the world , receiving and transmitting data looks like packet of 8-bits (1-byte) .

- Old printers communicate with computers via (8-bit data bus) , the disadvantage of this way of connection:
- You are limited with a short distance between the two devices
- The wires used to connect 8-bits at the same time are expensive .
- Although parallel is faster than serial , but it doesn't work for a long distances as long cables diminish and distort signals .
- So Serial communication is used for transferring data between two systems located at distances of hundreds of feets to million of miles .
- Transferring 8-bit data is not only cheaper but also enables two computers located in two different cities to communicate



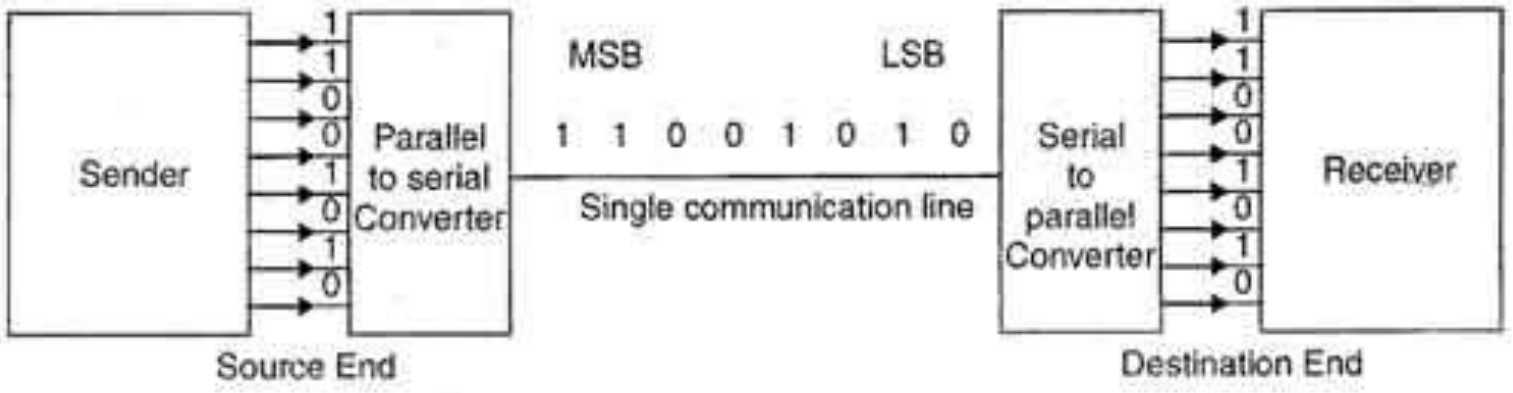






Serial communication

- uCs has many ways enables you to send - receive data to and from uC :
- UART – SPI – I2C
- Serial communication used just one wire to transfer data from a device to another device , instead of 8 wires in parallel communication , to do that first we convert 8-bit parallel to 8-bit serial data using an embedded chip inside uC called Parallel-in-Serial-out-shift Register : register its input 8-bit parallel and its output 8-bit serially .
- On the other side (the receiver) must have another chip Serial-in-Parallel-out shift register to convert the data again to 8-bit in parallel



Serial transmission

types of serial communication

- Simplex : one Sender – one receiver

- Such as : printers , computer send the data , printer receive

- duplex : can be Half duplex - full duplex depending on the two directions work at the same time or not

- Half duplex :

one sender – one receiver at a time

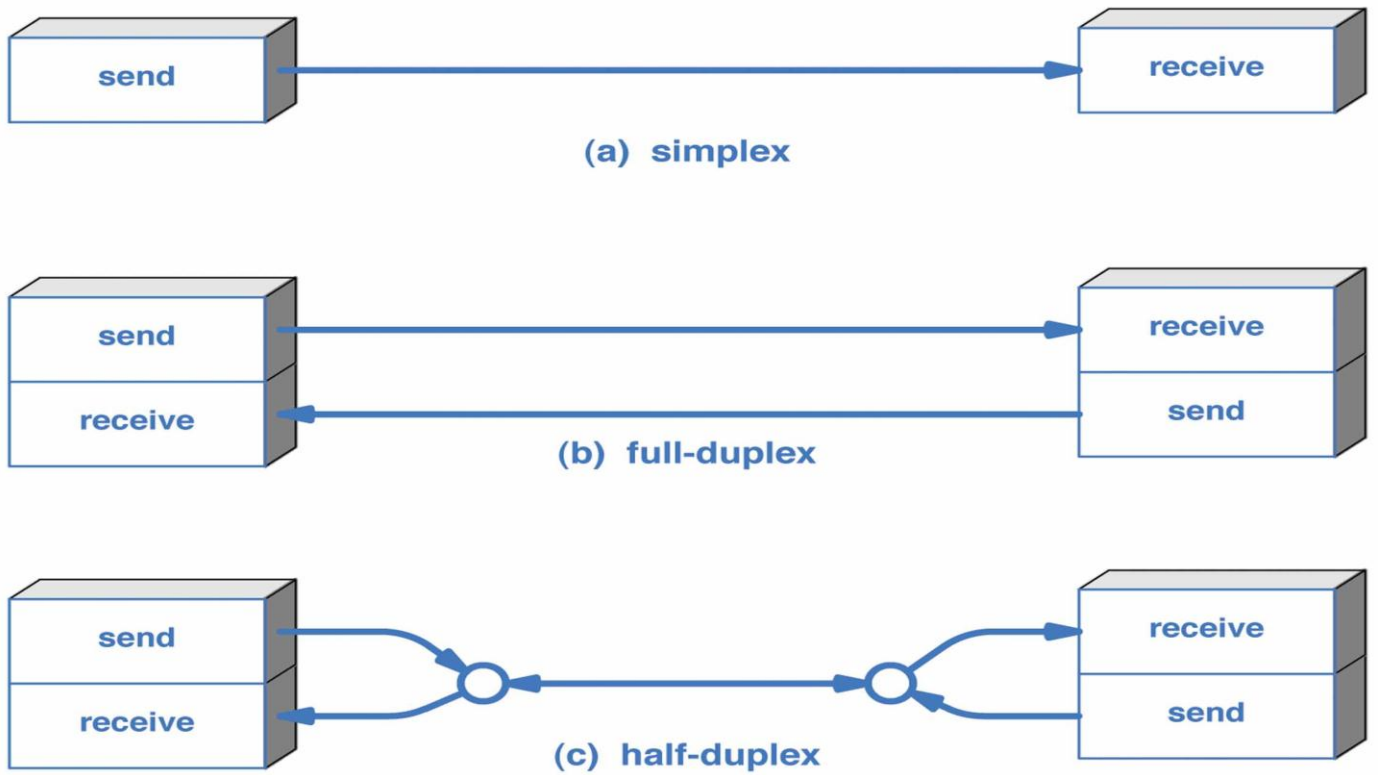
ex : policeman device

- Full duplex :

two senders and two receiver at a time

ex : mobile phones , you can talk

and listen at the same time



Synchronous vs Asynchronous

- There are two ways to transfer data serially with UART protocol :

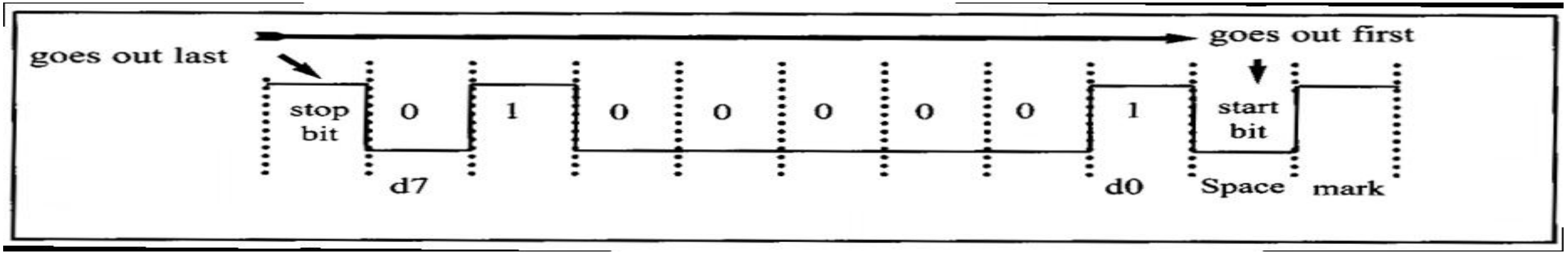
- Synchronous - Asyncronous

- Syncronous communication is used to transfer a block of data at a time , but asynchronous is used to transfer one byte at a time .

- We can program uC to work in one of them but the code will be long , so the designers of uCs made an embedded ICs inside the uCs specific for serial communication and can work parallel with uC

Asynchronous serial communication

- data coming to the receiver in a serial transfer is all 0s and 1s , it's difficult to make sense of the data unless the sender and receiver agree on a set of rules
- Protocol : mean a way of organizing receive and transmit data , data transfer speed , how data is packed , pins used like Rx - Tx - Gnd , when data begins and ends
- Start – end receiving bits : one byte is sent between one start and one stop bit this is called frame start bit is one LOW pulse , but end bit may be one or two bits and always be HIGH



Baud Rate

- The rate of transferring data over serial communication is measured : bps

or bit per second , it depend on the system used in communication :

ex : old IBM devices send data with speed in between 100 : 9600 bps , by the time and development modem devices reach 56Kbps

- Most of uCs including AVR support serial communication speed up to 115200 bps (about 100k per second)

- Some standard baud rates are

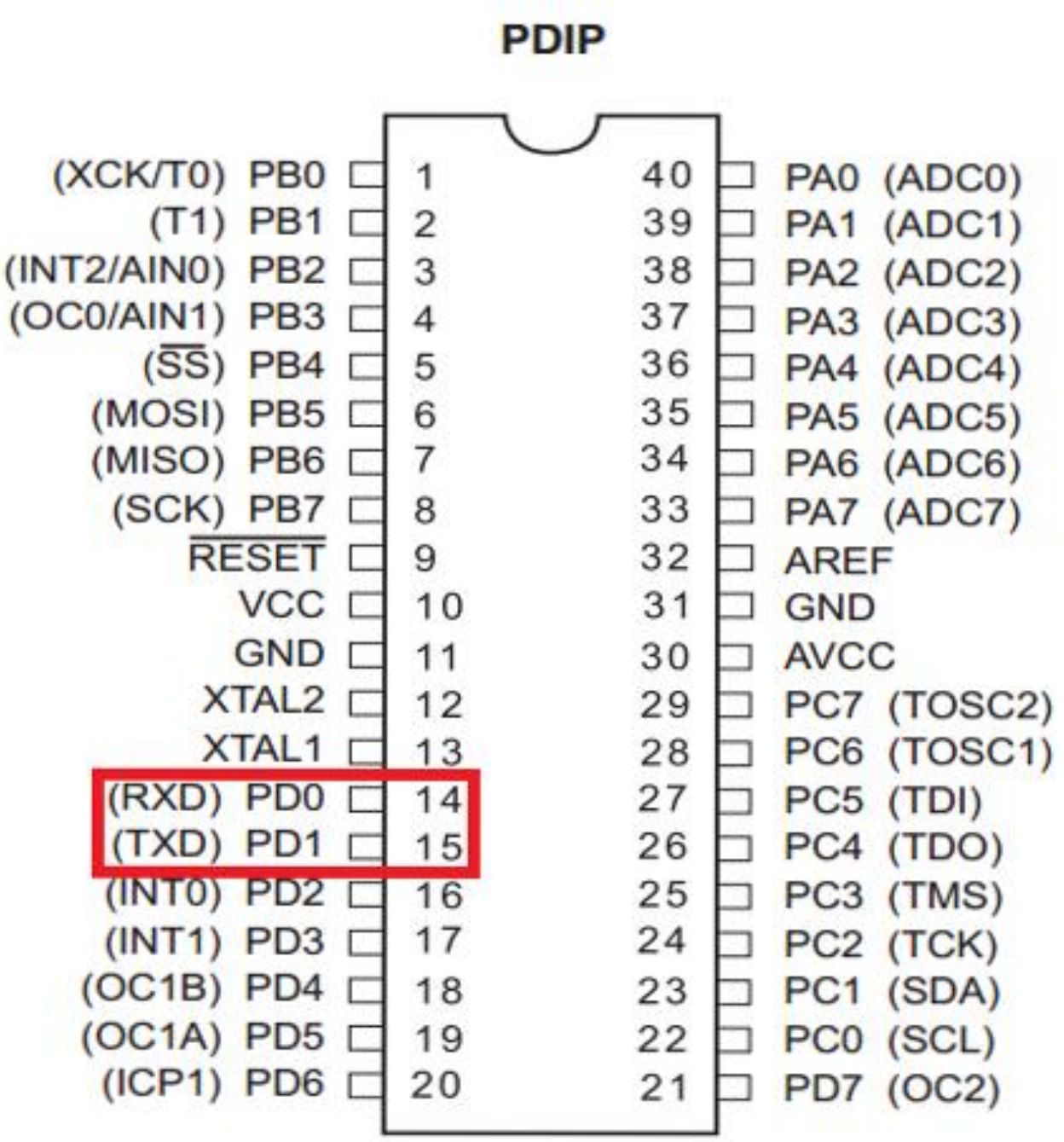
- 1200
- 2400
- 4800
- 9600
- 19200
- 38400
- 57600
- 115200
- ... etc

AVR UART pins

RXD : is used to receive data
connect it to TXD of sender

TXD : is used to send data
Connect it to RXD of receiver .

Gnd : Receiver and tranmiter
must have the same gnd pin .



AVR uart configuration

- we configure UART via set the baud rate - fram size – number of stop bits – error detection - the way uC use to serve UART via polling or interrupt we do this through five registers :

- UBRR : USART Baud Rate Register
- UCSRA – UCSRB – UCSRC :

UART Control & Status Registers (A , B , C)

- UDR : USART I/O Data Register

UBRR [H: L]

15	14	13	12	11	10	9	8	
URSEL	–	–	–	UBRR[11:8]				UBRRH
UBRR[7:0]								UBRRL
7	6	5	4	3	2	1	0	

UDR

7	6	5	4	3	2	1	0
TXB / RXB[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

UCSRA

7	6	5	4	3	2	1	0	
RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
R	R/W	R	R	R	R	R/W	R/W	

UCSRB

7	6	5	4	3	2	1	0	
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	

UCSRC

7	6	5	4	3	2	1	0	
URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

UART Registers

UBRR : UART Baud Rate Register [H : L] : consist of two registers the first

LOW 8-bits loads into **UBRRL** , and the HIGH value loads into **UBRRH** ,

the whole value load in bits (0:11)

```
#define baudRate 9600
uint16 myBaudRate = ((F_CPU / (8L * baudRate)) -1 ) ;
```

/* **Type Casting :**

type casting is a way to convert a variable from one data type to another data type. For example, if you want to store a 'long' value into a simple integer then you can type cast 'long' to 'int' */

```
/* casting uint16 to uint8 */
UBRRH = (uint8) (myBaudRate>>8) ;
UBRRL =  (uint8) myBaudRate ;
```

UBRR [H: L]

15	14	13	12	11	10	9	8	
URSEL	–	–	–	UBRR[11:8]				UBRRH
UBRR[7:0]								UBRRL
7	6	5	4	3	2	1	0	

UCSRA : UART control & status register A :

This register contain the flag used in UART .

UCSRA

7	6	5	4	3	2	1	0	
RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
R	R/W	R	R	R	R	R/W	R/W	

As we mentioned earlier that uC serve several peripherals in two ways

- Interrupt : when an event occur uC it trigger an interrupt signal then serve it (ISR)
- Polling : uC monitor the flag of the peripheral until it raised
- RXC : set bit = 1 , if byte receiving complete , remains = 0 during receiving .
- TXC : this bit = 1 , if byte sending complete , remains = 0 during sending .
- UDRE : this bit = 0 if uC busy , if the uC becomes = 1 when uC ready to receive another data
- U2X :if set this bit = 1 , it double transfer rate .

So if we used polling method we monitor the flags from this register :

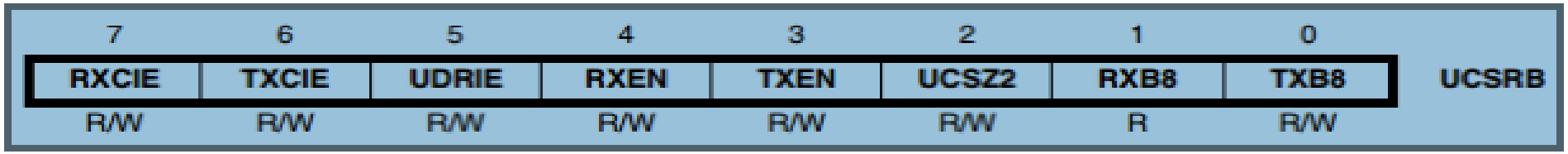
```
while (!(UCSRA &(1<<RXC))) ;    // do nothing until uC receive data
uint8 myReceivedData = UDR ;            // save the received data
```

UCSRB : UART control & status register B :

This register enable Rx - Tx - (interrupt for Tx - Rx)

- **RXCIE** : if set this bit = 1 , you enable the interrupt of receiving data
- **TXCIE** : if set this bit = 1 , you enable the interrupt of sending data
- **UDRIE** : if set this bit = 1 , you enable the interrupt of whether uC ready to send or receive data
- **RXEN** : if set this bit = 1 , you enable uC to receive data
- **TXEN** : if set this bit = 1 , you enable uC to send data
- **UCSZ2** : one of the bits to set the size of sending character

UCSRB




```
UCSRB |= (1<<RXEN) | (1<<TXEN) ;      // enable Tx- Rx
UCSRB |=(1<<RXCIE)|(1<<TXCIE) ;      // enable interrupt
```

When using interrupt :

```
ISR (USART_RXC_VECTOR){
 uint8 receivedData = UDR ;          // save the data into a variable
}
```

UCSRC : UART control & status register C :

This register contain two important bits to determine the character size

UCSZ0 : 1 we use 8-bit in

UCSRC |= (1<<UCSZ0) | (1<<UCSZ1) ; // use 8-bit character size

UCSRC

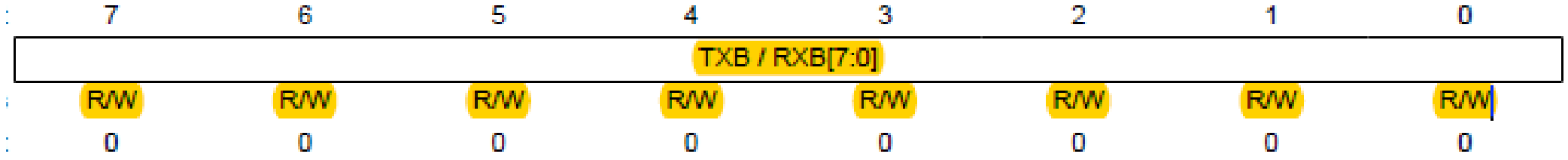
7	6	5	4	3	2	1	0	
URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

UDR : UART data Register :

This register is used to store the received data , or to load with the wanted data to be sent .

```
/* load UDR with the wanted data to be sent */
 uint8 txData = 'a' ;
 UDR = txData ;
 /* store the received data */
 uint8 rxData ;
 rxData = UDR ;
```



UDR UART Data Register

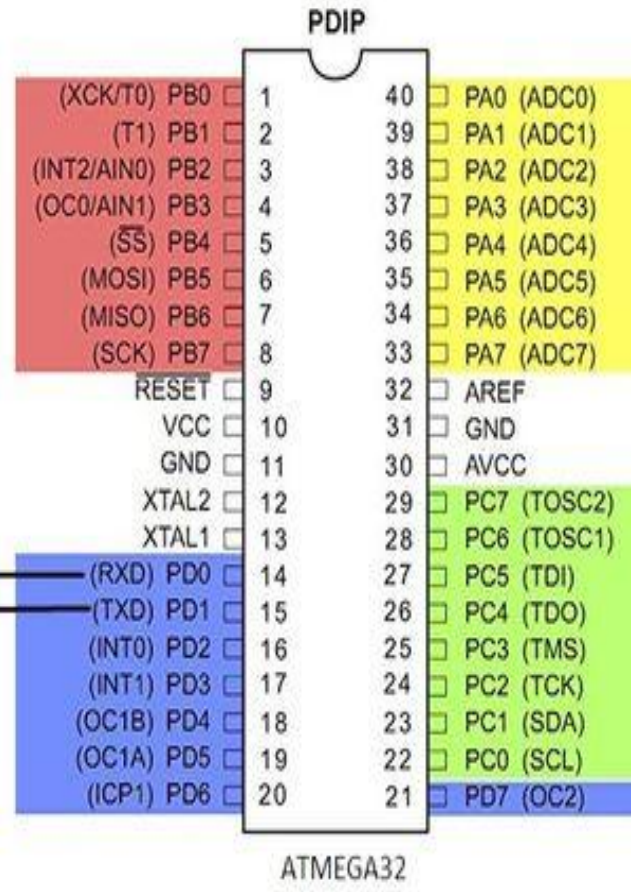
usb to UART converter

- One of the most common and easiest chip in the world of of robotics like

ROVs - minesweepers - etc ....,

- It's a great tool for embedded systems that require a serial connection to a computer. The board attaches to the USB bus and appears as a standard COM port

usb-uart	- >	uC
Tx	- >	Rx
Rx	- >	Tx
Gnd	- >	Gnd



References :

- Books :
- Simply AVR - > Abdallah Ali
- The AVR microcontroller & Embedded Systems using Assembly & c -- > Mazidi
- ATMEGA 32A Datasheet
- PIC microcontroller -- > Milan Verle
- Websites :
- https://www.sparkfun.com
- http://maxembedded.com
- https://www.tutorialspoint.com/cprogramming
- https://stackoverflow.com
- https://www.quora.com
- https://www.lucidchart.com

Any questions ?

- Instructor : Mohammed Hemed

- Embedded Systems developer at fab lab Ismailia

Repository link of Embedded workshop Material:

https://github.com/FabLab-Ismailia/Embedded-Systems-Workshop

Contact me :

- Gmail :

mohammedhemed23@gmail.com

- Linkedin :

https://www.linkedin.com/in/mohammedhemed