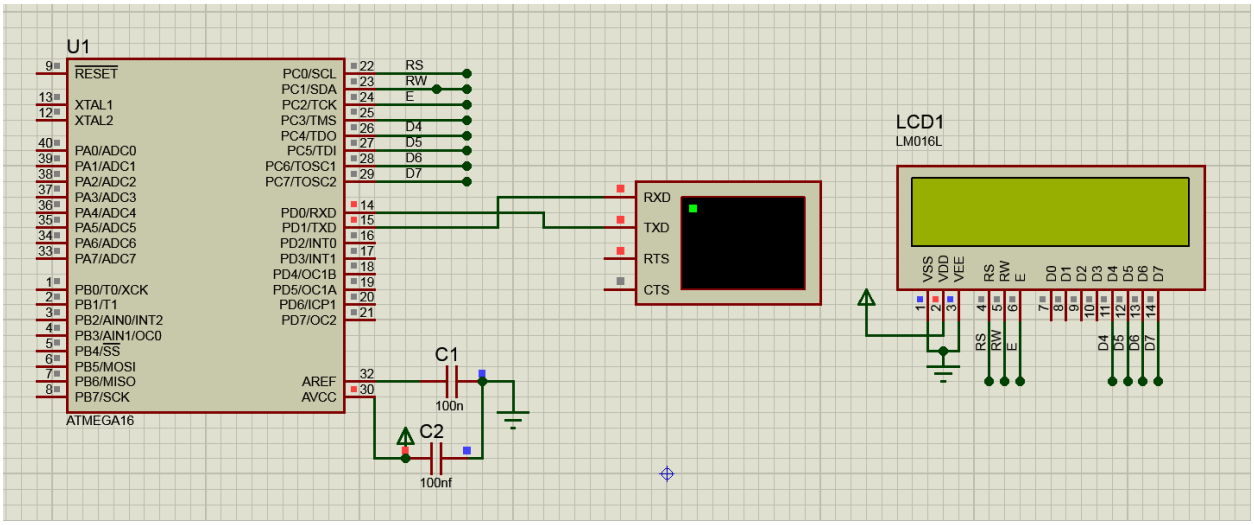
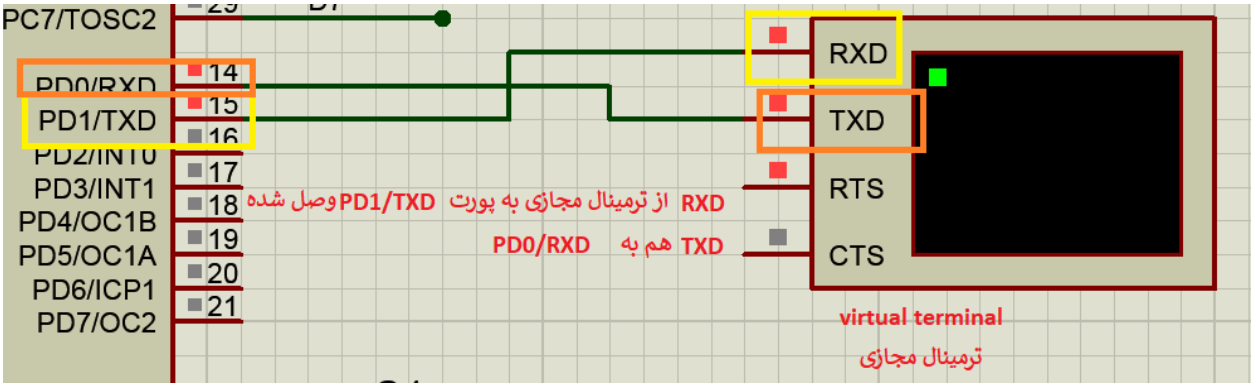
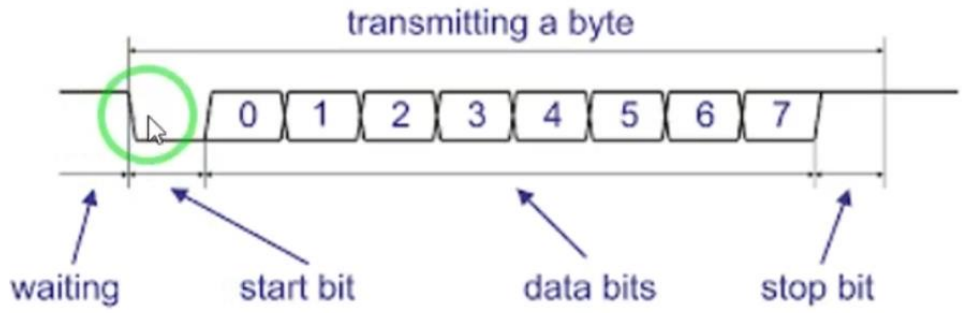
Session7 UART



نکته مهم:



فرمت ارسال اطلاعات



باس RXTx در وضعیت high قرار داره با دریافت بیت صفر که به عنوان start bit فرستاده میشه، منتظر دریافت اطلاعات میشه و ۸بیت داده را برداشت میکنه و درنهایت با بیت استاپ که یک منطقی است، میتوانه یه بایت داده را ارسال کنه.

UART دوتا بخش داره

۱. transmitter

۲. receiver

که انتقال اطلاعات به دو روش میتوانه انجام بشه

۱. pulling

۲. interrupt

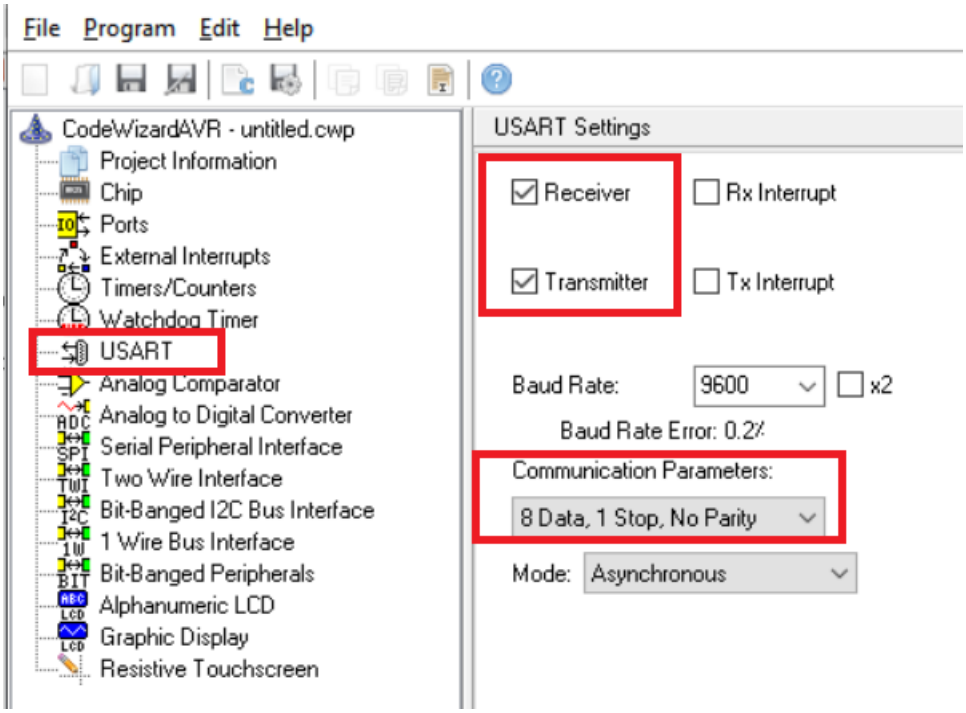
3 تا سناریو درباره ی UART خواهیم داشت:

۱. فرستنده و گیرنده هردو به صورت پولینگ باشند

۲. فرستنده در حالت پولینگ باشه و هروقت اطلاعاتی داره بفرسته و گیرنده در حالت اینتراپت باشه

۳. هم فرستنده و هم گیرنده در حالت اینتراپت باشند.

سناریو ۱




```
// Standard Input/Output functions
#include <stdio.h>
```

```
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=(0<<RXC) | (0<<TXC) | (0<<UDRE) | (0<<FE) | (0<<DOR) | (0<<UPE) | (0<<U2X) | (0<<MPCM);
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (1<<RXEN) | (1<<TXEN) | (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);
UCSRC=(1<<URSEL) | (0<<UMSEL) | (0<<UPM1) | (0<<UPM0) | (0<<USBS) | (1<<UCSZ1) | (1<<UCSZ0) | (0<<UCPOL);
UBRRH=0x00;
UBRRL=0x33;
```

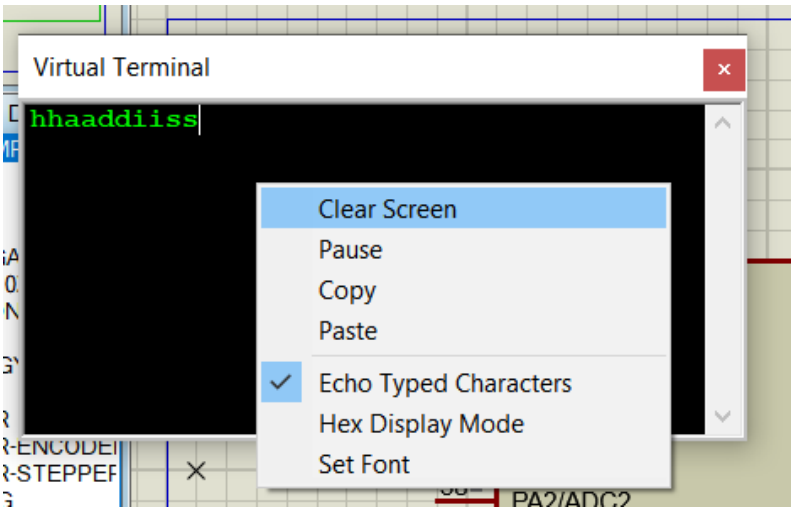
```
void main(void)
{
char a;

while (1)
      {
      // Place your code here
      a = getchar();
      putchar(a);

      }
}
```

تا زمانی که یک کاراکتر دریافت کنه در خط اول منتظر ورودی میمونه.

پس از دریافت کاراکتر توسط پورت سریال نمایشش میده

ارسال و دریافت اطلاعات به روش pulling

خروجی

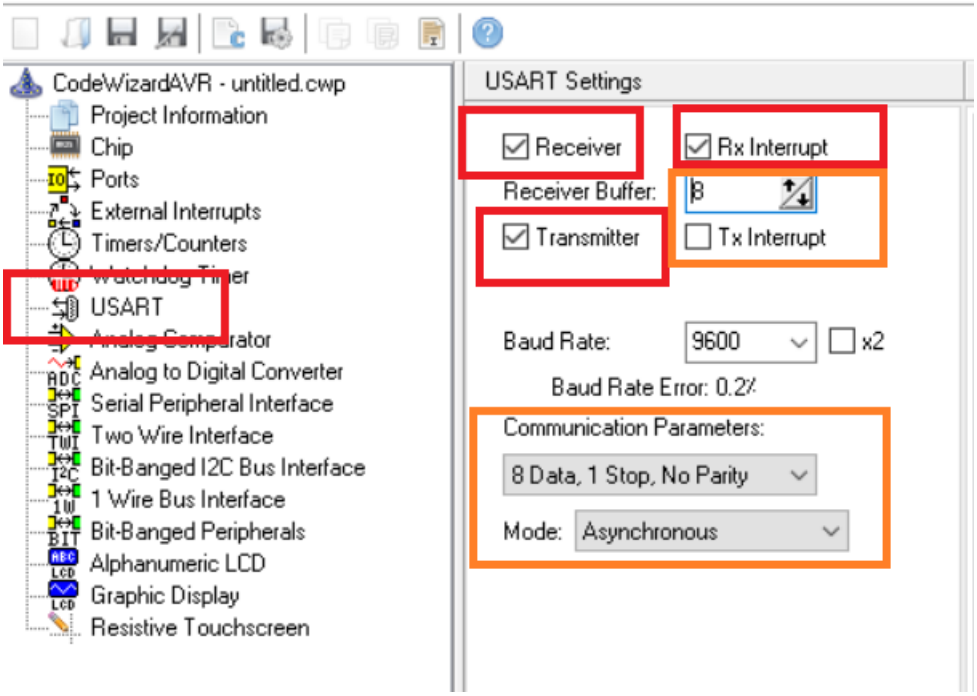


سناریو۲

فعال کردن اینتراپت برای گیرنده: RX interrupt

یک بافر براش درنظر گرفته میشه

فرستنده به روش پولینگ: TX pulling



تنظیمات گیرنده یا reciever

```
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE <= 256
unsigned char rx_wr_index=0,rx_rd_index=0;
#else
unsigned int rx_wr_index=0,rx_rd_index=0;
#endif

#if RX_BUFFER_SIZE < 256
unsigned char rx_counter=0;
#else
unsigned int rx_counter=0;
#endif
```

transmiter(TXD) => pulling
reciever (RXD) => interrupt

1.buffer size:8bits
2.rx_buffer[8]
3.rx_wr_index
4.rx_rd_index
5.rx_counter

1.buffer size:8bits
2.rx_buffer[8]
3.rx_wr_index
4.rx_rd_index
5.rx_counter

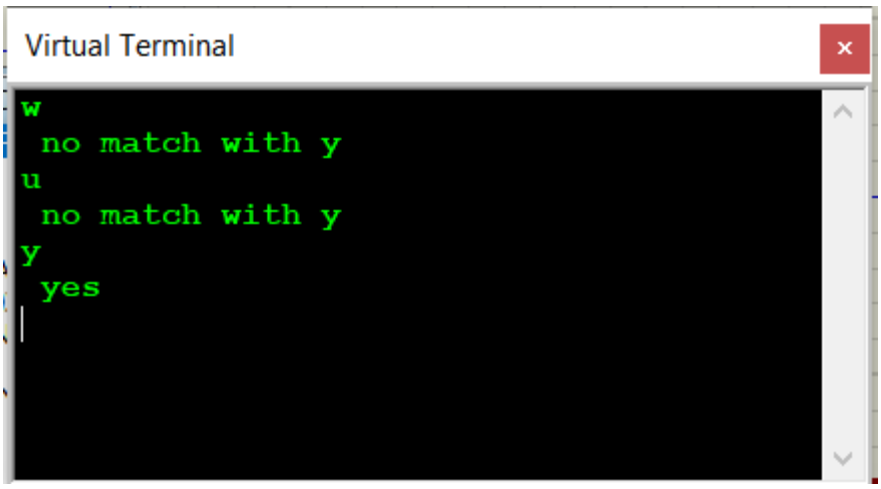
```
// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer[rx_wr_index++]=data;
#if RX_BUFFER_SIZE == 256
   // special case for receiver buffer size=256
   if (++rx_counter == 0) rx_buffer_overflow=1;
#else
   if (rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
      {
      rx_counter=0;
      rx_buffer_overflow=1;
      }
#endif
   }
}
```

به کمک دستور
getchar()
میشه محتویات بافر گیرنده را خواند

```
#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index++];
#if RX_BUFFER_SIZE != 256
if (rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#endif
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif
```

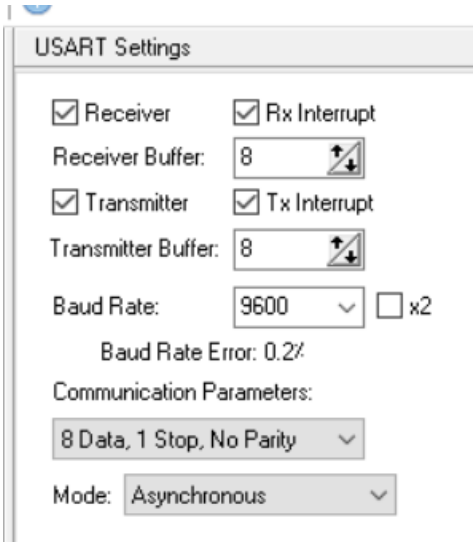
تغییر روتین اینتراپت

```
// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if(data == 'y') printf("\r\n yes \r\n");
else printf("\r\n no match with y \r\n");

}
```



سناریو ۳

فرستنده و گیرنده هردو از اینتراپت استفاده کنند.



کدهای گیرنده که مثل سناریو۲ است.

ولی کدهای فرستنده یا TRX به شکل زیر میشه

```
// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];

#if TX_BUFFER_SIZE <= 256
unsigned char tx_wr_index=0,tx_rd_index=0;
#else
unsigned int tx_wr_index=0,tx_rd_index=0;
#endif

#if TX_BUFFER_SIZE < 256
unsigned char tx_counter=0;
#else
unsigned int tx_counter=0;
#endif
```

```
// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
if (tx_counter)
   {
   --tx_counter;
   UDR=tx_buffer[tx_rd_index++];
#if TX_BUFFER_SIZE != 256
   if (tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
#endif
   }
}
```

اطلاعات برای ارسال روی رجیستر UDR قرار میگیرند تا به ترتیب ارسال شوند.

در UART اینطوری است که به محض اینکه اطلاعات روی رجیستر UDR قرار گرفت، فرایند ارسال شروع میشود و در حالت RX به محض اینکه وارد رجیسترUDR شد، بلافاصله اون داده را میخونه

```
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
   {
   tx_buffer[tx_wr_index++]=c;
#if TX_BUFFER_SIZE != 256
   if (tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
#endif
   ++tx_counter;
   }
else
   UDR=c;
#asm("sei")
}
#pragma used-
#endif
```

ظاهر استفاده از وقفه و استفاده نکردن از وقفه هیچی نیست و در ظاهر مثل هم هستند ولی اگه در حین اجرای برنامه به دستور getchar() در حالت pulling برسیم، تمام پروسه های میکرو متوقف میشه تا یه دیتایی دریافت بشه که باعث هدررفتن منابع میکرو میشه.

به کمک وقفه ها، هرجا که لازم باشه از بافرها داده ها را میخوانیم یا داخلشون مینویسیم.

فرایند ارسال و دریافت داده به طور مستقل از فرایند های دیگه میکرو داره انجام میشه و میکرو میتوانه به کارهای دیگرش برسه. => هیچ تاخیری در روند اجرای برنامه ها ایجاد نمیشه.

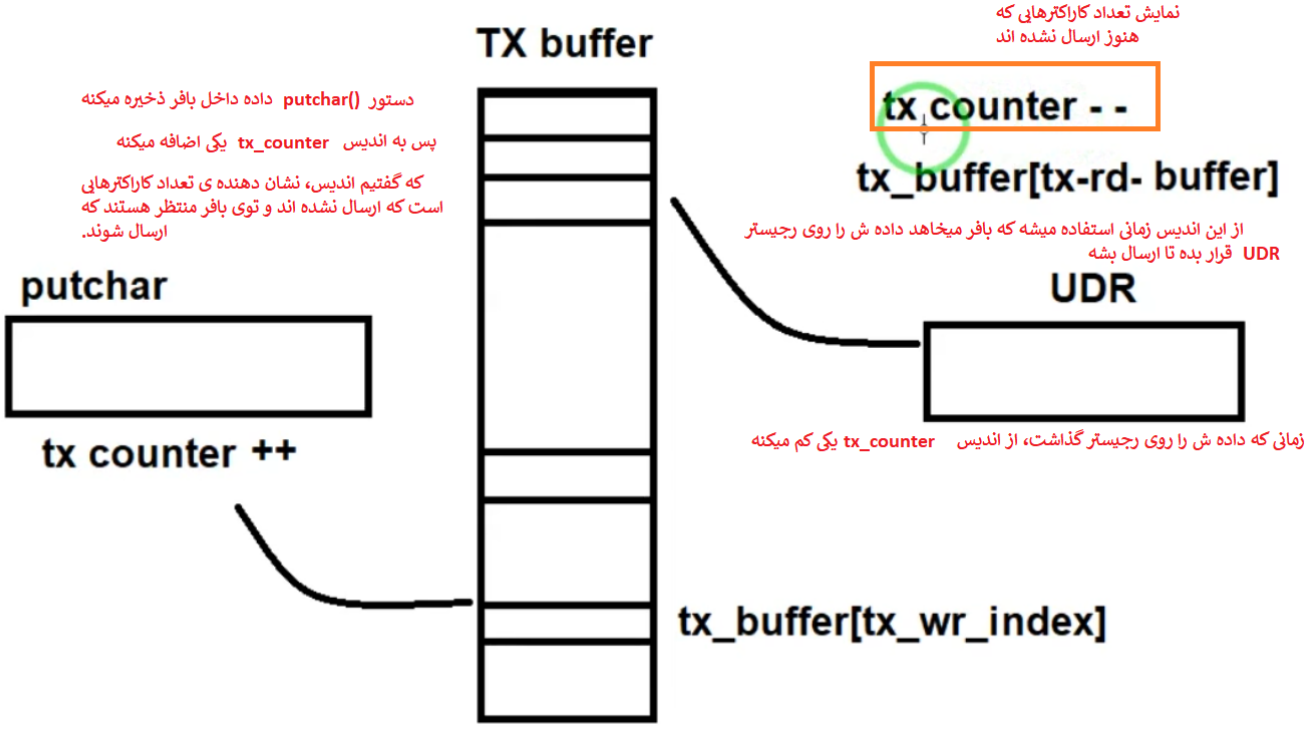
دستورات کاربردی برای uart

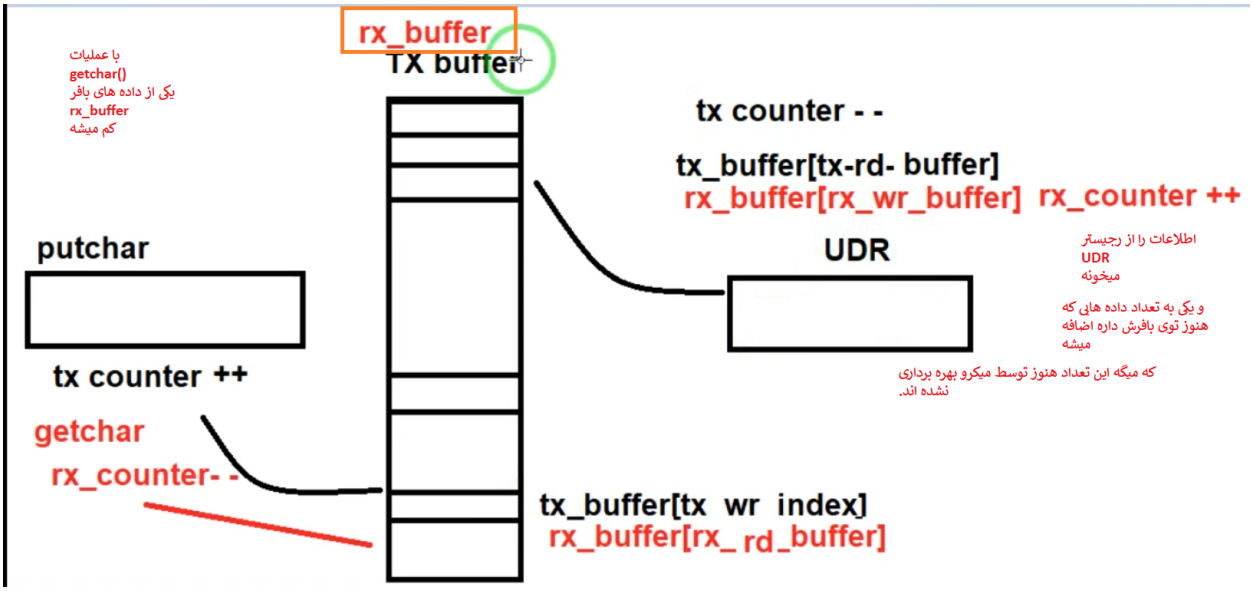
- The serial communication is realized using the Standard Input/Output Functions **getchar**, **gets**, **scanf**, **putchar**, **puts** and **printf**.
- #include <stdio.h>

- For interrupt driven serial communication, CodeWizardAVR automatically redefines the basic **getchar** and **putchar** functions.

اندیس tx_counter :

که گفتیم اندیس، نشان دهنده ی تعداد کاراکترهایی است که ارسال نشده اند و توی بافر منتظر هستند که ارسال شوند.




با عملیات
getchar()
یکی از داده های بافر
rx_buffer
کم میشه
rx_buffer
TX buffer
tx counter - -
tx_buffer[tx-rd- buffer]
rx_buffer[rx_wr_buffer] rx_counter ++
putchar
UDR
اطلاعات را از رجیستر
UDR
میخونه
و یکی به تعداد داده هایی که
هنوز توی بافرش داره اضافه
میشه
که میگه این تعداد هنوز توسط میکرو بهره برداری
نشده اند.
tx counter ++
getchar
rx_counter- -
tx_buffer[tx wr index]
rx_buffer[rx_ rd _buffer]