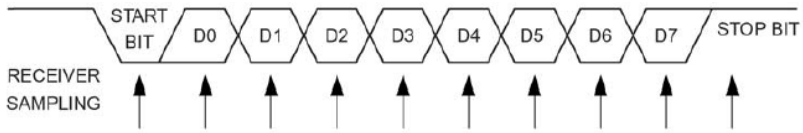
ارتباط سریال USART

ارسال و دریافت بیت به بیت اطلاعات را ارتباط سریال یا سری می گویند . اهمیت چنین ارتباطی در مکان های صنعتی و کاربرد آن در ارتباط بین یک میکرو با میکرو دیگر یا یک کامپیوتر می باشد که باعث کاهش تعداد سیم ها می گردد چون در نوع دیگر ارتباط (موازی) اطلاعات n بیتی را باید با n خط تبادل کرد. این ارتباط در دو مد UART و USART انجام می پذیرد :

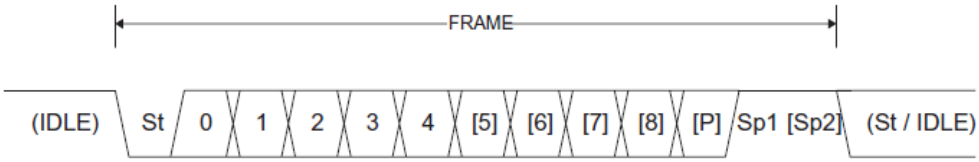
UART (Universal Asynchronous Reciver and Transmitter)

جهت ارسال و دریافت اطلاعات به صورت غیر همزمان یا آسنکرون می باشد . به طور کلی در ارتباط سریال هر بایت اطلاعات در یک قاب دیتا فرستاده و دریافت می شود این قاب دیتا شامل بیت شروع ارتباط ، بیت های دیتا ، بیت توازن (Parity) و بیت توقف ارتباط می باشد . بیت شروع ارتباط با یک لبه پایین رونده می باشد و توسط این بیت میکرو فرستنده به میکرو گیرنده می فهماند که قصد برقراری ارتباط را دارد پس از آن طبق تنظیمات انجام شده میکرو گیرنده هر چند لحظه یکبار اطلاعات را می خواند. به این فاصله زمانی برداشت اطلاعات Baud Rate می گویند که باید در هر دو میکرو یکسان باشد و در کدویزارد این تنظیمات را انجام می دهیم . بعد تبادل هشت بیت اطلاعات و یک بیت توازن(صفر یا یک) توسط فرستنده ارسال می شود ، این بیت نشان

می دهد که اطلاعات بدون نقص و درست دریافت شده است . مثلا میکرو دریافت کننده به گونه ای تنظیم شده که بعد از دریافت هشت بیت اطلاعات باید یک بیت صفر دریافت کند تا بفهمد که اطلاعات درست به دستش رسیده است و اگر بیت دریافتی یک باشد میکرو متوجه می شود که اطلاعات ناقص رسیده و اطلاعات نا صحیح می باشد. سپس بیت توقف که یک لبه بالا رونده (یک) است ، به نشانه اتمام ارتباط فرستاده میشود.



(بدون بیت توازن) شکل**8-1**



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.

شکل**8-2**

(Universal Synchronous Serial Receiver and Transmitter) USART

جهت ارسال و دریافت اطلاعات به صورت همزمان یا سنکرون است . در این نوع ارتباط نیز اطلاعات به صورت قاب دیتا ارسال می شوند . تفاوت USART با UART در بافرهای دریافت می باشد که در حالت سنکرون بهبود یافته می باشد . در این حالت فرستنده یک پالس سنکرون کننده (همزمان کننده) با یک خط کلاک ارسال می کند و گیرنده نیز می

تواند دیتا را بر روی یک خط توسط کلاک همزمان کننده که از طرف فرستنده ارسال می شود ، دریافت نماید .

رجیستر UDR (USART I/O data Register)

یک رجیستر 8 بیتی واسطه بین cpu و گیرنده _فرستنده UART است که هم جهت ارسال و هم جهت دریافت بیت به بیت اطلاعات می باشد این رجیستر 8 بیت اطلاعات را بیت به بیت دریافت یا ارسال کرده و سپس 8 بیت دیگر را ارسال یا دریافت می کند .

رجیستر UCSRA (USART Control and Status Register)

این رجیستر جهت کنترل پروتکل ارسال و دریافت اطلاعات می باشد و دارای 8 بیت می باشد و ما از 4 بیت ان جهت کنترل تبادل اطلاعات استفاده می کنیم .

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	

U2× : در صورت یک شدن این بیت تعداد بیت های ارسال یا دریافت در یک ثانیه (Baud Rate) دو برابر می شود .

UDRE : هرگاه به هنگام ارسال سریال، رجیستر UDR آماده گرفتن اطلاعات جدید شود این بیت یک می گردد.

TXc : هرگاه به هنگام ارسال سریال تمام بیت های رجیستر UDR ارسال گردد این بیت یک می شود .

RXc : اگر به هنگام دریافت سریال تمام 8 بیت رجیستر UDR پر شود این بیت یک می گردد.

چهار بیت دیگر که ما از آنها کمتر استفاده می کنیم :

MPCM(Multi Processor Communication Mode) : اگر بخواهیم با چند میکرو ارتباط سریال داشته باشیم باید این بیت را یک کنیم .

PE (Parity Error): اگر در هنگام دریافت خطای توازن رخ دهد (به هر دلیلی اطلاعات نادرست دریافت شوند) این بیت یک خواهد شد . توجه کنید که اگر می خواهید به رجیستر UCSRA مقدار بدهید باید این بیت را صفر کنید .

DOR(Data Over Run) : اگر بافر دریافت پر باشد و دیتای جدیدی به شیفت رجیستر وارد شود خطای Data Over Run رخ داده و این بیت یک می شود .

FE(Frame Error) : یک شدن این بیت به معنای خطا در دریافت قالب دیتای دریافت می باشد .

نحوه کنترل ارسال یا دریافت سریال بدون تابع :

برای کنترل سریال دریافت یا ارسال باید به ترتیب بیت های RXC و UDRE و را کنترل می کنیم بدینگونه که بعد از هر بار ارسال 8 بیت اطلاعات بیت UDRE ، و بعد از هر بار دریافت 8 بیت اطلاعات RXC را صفر می کنیم تا میکرو برای ارسال یا دریافت بعدی آماده شود . برای دریافت به صورت زیر عمل می کنیم :

```
While(!UCSR.7) ;                صبر کردن تا پر شدن رجیستر دریافت//
UCSRA.7=0 ;                     // صفر کردن بیت هفتم UCSRA و آماده شدن برای دریافت بعدی
X=UDR ;                         // ریختن اطلاعات دریافتی در متغیر
```

و برای ارسال اطلاعات :

```
While(!UCSR.5) ;                صبر کردن تا آماده شدن میکرو برای ارسال//
```

صفر کردن بیت پنجم رجیستر UCSR // UCSRA.5=0 ;

ریختن دیتا در رجیستر UDR جهت ارسال // UDR=x ;

ارسال و دریافت سریال به کمک توابع پورت سریال

توسط توابع کتابخانه stdio.h می توان به راحتی از طریق ارتباط سریال اطلاعات را تبادل کرد این توابع در زیر توضیح داده شده اند :

توابع ارسال

1- putchar

برای ارسال یک یک کاراکتر 8 بیتی می باشد . مثلا :

```
putchar ('a');
```

توجه کنید که زمانی که CPU به این دستور می رسد آنقدر آنجا می ماند تا کاراکتر ارسال شود .

2- putsf

از این تابع جهت ارسال رشته ذخیره شده در ROM استفاده می شود. مثلا :

```
putsf("ali") ;
```

3- puts

ارسال رشته ذخیره شده در RAM

4- printf

printf("متغییر آرایه ای مبدا ,"کارکترهای دلخواه به همراه کارکتر کنترلی و فرمت")

بکمک کاراکترهای فرمت می توان نوع کاراکتر ارسالی و یا دریافتی،و توسط کاراکترهای کنترلی می توان تابع ارسال را کنترل کرد. تعدادی از این توابع در زیر آمده است .

کاراکتر های فرمت

%c : کارکتر %d : اعداد صحیح مثبت و منفی %s : رشته %u : اعداد صحیح مثبت %x : مبنای هگز %b : عدد باینری و ...

کاراکتر های کنترلی

/n : رفتن به خط جدید /f : رفتن به صفحه جدید /t : هشت تا فاصله و ...

توابع دریافت

1- getchar : برای دریافت یک کاراکتر می باشد .

مثال : x=getchar ();

توجه کنید که زمانی که CPU به این دستور می رسد آنقدر آنجا می ماند تا یک کاراکتر دریافت شود .

2- gets : جهت دریافت یک رشته به کار می رود .

gets (متغیر آرایه ای ,تعداد کاراکترهای دریافتی)

مثال: دریافت کلمه ali

```
unsigned char x[3]
gets(x,3)
```

3- scanf :

scanf("کارکترهای فرمت" , متغییر مقصد)

مثال :

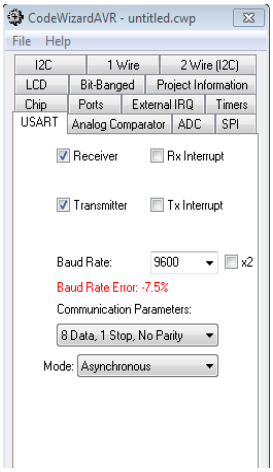
```
unsigned char  m[10] , y ;

Scanf("%s%d ", m, y);
```

علاوه بر توابع فوق که برای ارسال و دریافت استفاده می شوند تابعی بنام sprint وجود دارد که توسط آن می توان یک رشته ساخت و در وتغییر مقصد ریخت که می تواند هم در ارسال و هم در دریافت به ما کمک کند نحوه به کارگیری ان به صورت زیر می باشد .

sprint (متغییر مبدا , " کاراکتر دلخواه همراه با کاکتر فرمت , "متغییر ارایه مقصد)

تنظیمات CodeWizard

در CodeWizard با زدن گزینه USART می توان تنظیمات مورد نظر را انجام داد .



شکل3-8

Resiver : این گزینه میکرو درحالت دریافت قرار می گیرد .

Transmitter : میکرو را در حالت ارسال قرار می گیرد .

RX Interrupt : وقفه دریافت را فعال می کند .

Tx Interrupt : وقفه ارسال را فعال می کند .

Baud Rate : میزان سرعت انتقال اطلاعات را بر حسب بیت بر ثانیه تعیین می نماید.

تذکر: باید توجه کرد که افزایش سرعت باعث افزایش میزان خطا می شود .

مثال : بدون استفاده از توابع کتابخانه stdio.h می خواهیم توسط ارتباط سریال کلمه ای را از یک میکرو به میکرو دیگر بفرستیم :

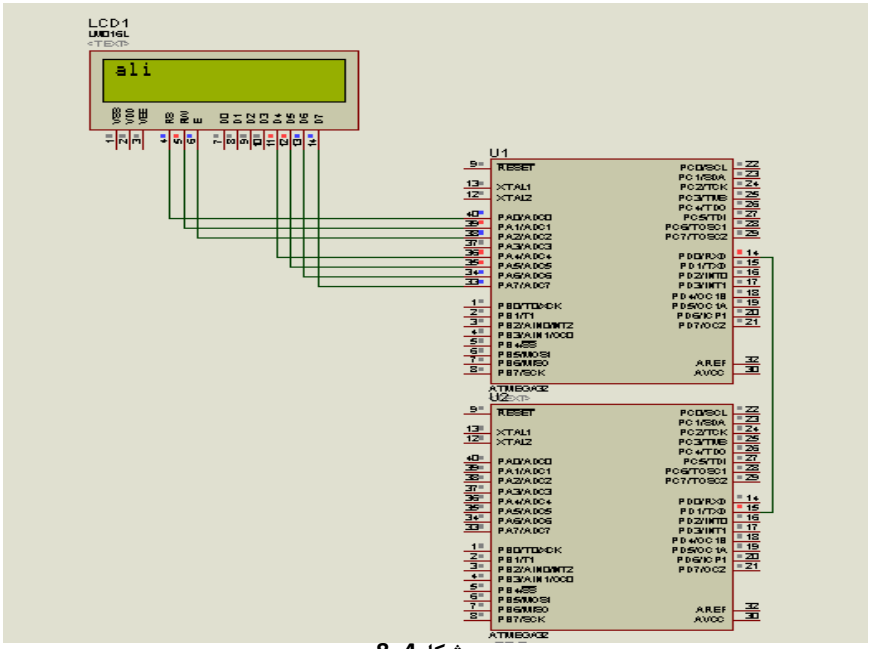
برنامه میکرو فرستنده

```
#include <mega32.h>                                معرفی میکرو مورد استفاده//
#include <delay.h>                                 فراخوانی کتابخانه تاخیر زمانی //
unsigned char a[]="ali",i;                         معرفی متغیرها//
void main(void)  {                                 برنامه اصلی //
while (1)  {                                       حلقه بینهایت//
   for(i=0;i<3;i++)  {                             ایجاد یک حلقه برای ارسال کلمه به کلمه//
   while(!UCSRA.5);                                صبر کردن تا ارسال اطلاعات قبلی//
   UCSRA.5=                                        صفر کردن بیت پرچم ارسال کامل اطلاعات//
   UDR=a[i];                                       قرار دادن دیتا در رجیستر UDR جهت ارسال//
   delay_ms(125);  } } }                           تاخیر زمانی //
```

برنامه میکرو دریافت کننده

```
#include <mega32.h>                                معرفی میکرو مورد استفاده//
#asm                                               شروع برنامه اسمبلی  //
.equ __lcd_port=0x1B ;POR                          معرفی پورت A برای اتصال به نمایشگر//
#endasm                                            پایان برنامه اسمبلی//
#include <lcd.h>                                   فراخوانی کتابخانه lcd//
unsigned char i=0,a[3],b,x;                        معرفی متغیرها //
interrupt [USART_RXC] void usart_rx_isr(void  {    وقفه دریافت //
```

```
a[i]=UDR;                                   ریختن دیتا دریافتی رجیستر UDR در متغیر//
i++;                                        افزایش متغیر کمکی جهت دریافت دیتای بعدی//
if(i==3)  {                                 ایجاد یک شرط برای اعمال مورد نظر بعد از دریافت کامل اطلاعات//
i=0;                                        صفر کردن متغیر کمکی //
lcd_gotoxy(0,0);                            رفتن به مکان دلخواه در نمایشگر//
lcd_puls(a);  }  }                          نمایش دیتای دریافتی //
void main(void)  {                          برنامه اصلی //
while (1) ;                                 حلقه بینهایت و بیکاری برنامه //
}
```



شکل4-8

مثال : می خواهیم یک عدد دو رقمی را از یک صفحه کلید دریافت کنیم و به کمک توابع کتابخانه stdio.h از طریق USART ، عدد دورقمی را به میکرو دیگر منتقل کرده و آن را نمایش دهیم :

برنامه میکرو فرستنده

```
#include <mega16.h>                                   //معرفی میکرو مورد استفاده
#include <delay.h>                                    // فراخوانی کتابخانه تاخیر زمانی
#include <stdio.h>                                    //فراخوانی کتابخانه stdio.h
unsigned char i,b,w ,x;                               // معرفی متغیرها
void main(void)  {                                    // برنامه اصلی
while (1)   {                                         //حلقه بینهایت
   PORTC=0xf0;          //یک کردن چهار بیت پر ارزشC وصفر کردن چهار بیت کم ارزش
                        //*************row1*********  خواندن سطر اول //
PORTC.4=0;                                            //صفر کردن بیت چهار پورت c
delay_ms(3);                                          // تاخیر زمانی
if(PINC.0==0){ w=1;b=1;while(!PINC.0);}               //اگر پین صفر برابر صفر شد یعنی عدد یک وارد شده است
if(PINC.1==0){ w=2;b=1;while(!PINC.1);}               //اگر پین یک برابر صفر شد یعنی عدد دو وارد شده است
if(PINC.2==0){ w=3;b=1;while(!PINC.2); }              //اگر پین دو برابر صفر شد یعنی عدد سه وارد شده است
PORTC.4=1;                                            // یک کردن پین چهار
                        //***********row2*********  خواندن سطر دوم //
PORTC.5=0;                                            // صفر کردن پین پنجم
delay_ms(3);                                          // تاخیر زمانی
if(PINC.0==0){ w=4;while(!PINC.0);b=1;}               //اگر پین صفر برابر صفر شد یعنی عدد چهار وارد شده است
```

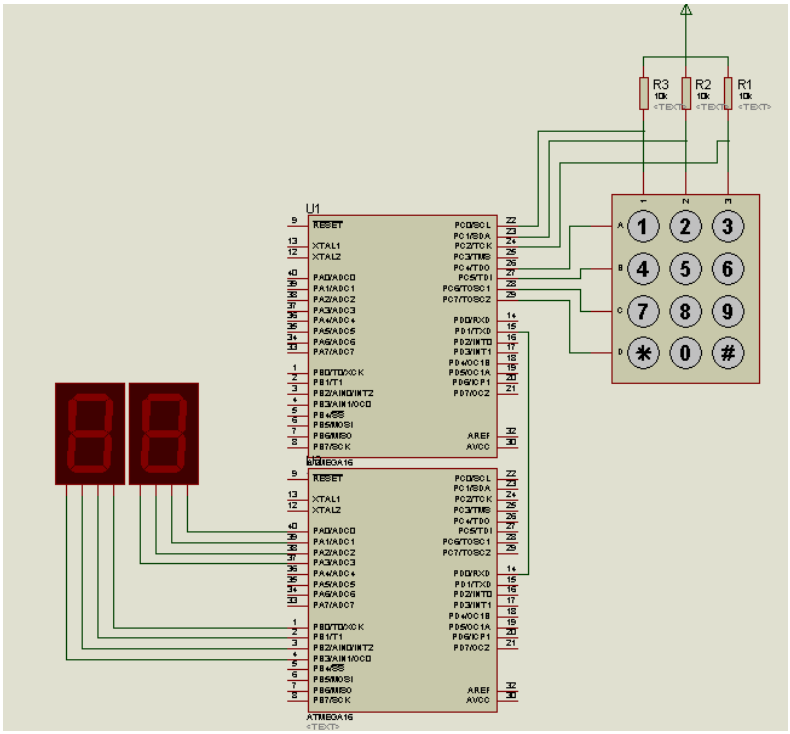
```
if(PINC.1==0){ w=5;while(!PINC.1);b=1; }          //اگر پین یک برابر صفر شد یعنی عدد پنج وارد شده است
if(PINC.2==0){w=6;while(!PINC.2);b=1;  }          //اگر پین دو برابر صفر شد یعنی عدد شش وارد شده است
PORTC.5=1;                                        //یک کردن پین پنجم
                                                  //**********row3*********  خواندن سطر سوم //
PORTC.6=0;                                        //صفر کردن پین شش
delay_ms(3);                                      //تاخیر زمانی
if(PINC.0==0) {w=7;while(!PINC.0);b=1;}           //اگر پین صفر برابر صفر شد یعنی عدد هفت وارد شده است
if(PINC.1==0) {w=8;while(!PINC.1);b=1; }          //اگر پین یک برابر صفر شد یعنی عدد هشت وارد شده است
if(PINC.2==0) {w=9;while(!PINC.2);b=1;  }         //اگر پین دور برابر صفر شد یعنی عدد نه وارد شده
PORTC.6=1;                                        //یک کردن پین شش
                                                  //**********row4*********  خواندن سطر چهار//
PORTC.7=0;                                        //صفر کردن پین هفت
delay_ms(3);                                      //تاخیر زمانی
if(PINC.1==0) {w=0;while(!PINC.1);b=1;  }         //اگر پین صفر برابر صفر شد یعنی عدد صفر وارد شده است
if(PINC.2==0){ x=3;while(!PINC.2);b=2;  }         //اگر پین دور برابر صفر شد یعنی کلید مربع وارد شده است
PORTC.7=1;                                        //یک کردن پین هفت
if(b==1&&x<2) {                                   //ایجاد یک شرط که بیشتر از دو عدد نتوان وارد کرد مگر اینکه کلید مربع زده شود و خروجی صفر شود
putchar(w);                                       //ارسال عدد وارد شده
b=0;                                              //آماده شدن برای دریافت عدد بعدی
 x++;       }                                     //استفاده از متغیر کمکی
```

```
if(x==3)  {                                   //وارد شدن به تابع در صورت زدن کلید مربع
for(i=0;i<2;i++)  {                           //ارسال دو بار عدد صفر تا خروجی صفر شود
putchar(0);     }x=0;   }                     //آماده شدن برای وارد کردن عدد جدید
```

برنامه میکرو گیرنده

```
#include <mega16.h>                           //معرفی میکرو مورد استفاده
#include <stdio.h>                            //فراخوانی کتابخانه stdio.h
unsigned char a[2],i;                         //معرفی متغیرها
void main(void)    {                          //برنامه اصلی
while (1)    {                                //حلقه بینهایت
    for(i=0;i<2;i++)  {                       //ایجاد یک حلقه برای دریافت دو عدد
    a[i]=getchar();                           //دریافت دیتا
    PORTB=a[0];                               //نمایش دیتای اول بر روی پورت B
    PORTA=a[1];                               //نمایش دیتای دوم بر روی پورت A
                        }
}
}
```

شکل در صفحه بعد


R3
10k
R2
10k
R1
10k
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA16
1
2
3
4
5
6
7
8
9
*
0
#

شکل5-8