Subject:
Year. Month. Date. ( )

- برای تولید موج PWM باید پایه PB3 به عنوان خروجی تعریف شود.
- برای محاسبه فرکانس موج در حالت P.C. PWM از رابطه زیر استفاده می‌کنیم.

$$f_{PWM} = \frac{f_{ClkI/O}}{Prescale \times 510} \quad (f_{ClkI/O} = f_{xtal}) \longleftrightarrow \text{Phase Correct PWM}$$

($510 = 2 \times 255$)

$$f_{PWM} = \frac{f_{clk\ I/O}}{Prescale\ (256 - TCNT0)} \longleftrightarrow \text{Fast PWM}$$

مثال: برنامه‌ای بنویسید که موج PWM را در حالت های P.C و Fast با استفاده از مقدار OCR0 = 0x80 و $f_{xtal} = 1\ MHz$ ، Pre. = 1 ، Non_Inverting تولید کند.

Phase Correct

```
#include <mega16.h>
Void main (Void){
PORTB = 0x00;
DDRB = 0x08;        → Pb.3 = out
TCNT0 = 0x00;
TCCR0 = 0x29;       → Fast → TCCR0 = 0x49
OCR0 = 0x80;
while (1){
    }
}
```

	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	
TCCR0	0	0	1	0	1	0	0	1	Ph. C.
TCCR0	0	1	0	0	1	0	0	1	Fast

$$f_{PWM} = \frac{1\ MHz}{1 \times 510} \cong 2\ kHz$$

$$f_{PWM} = \frac{1\ MHz}{1 \times (256 - 0)} \cong 4\ kHz$$

Subject:

Year. Month. Date. ( )

f_{xtal} $f_{xtal1} = \cdots$

مثال: برنامه‌ای بنویسید که با فرکانس 1KHz و D.C = 20% با روش Fast تولید کند.

$$\begin{cases} f_{xtal} = 16\ MHz \\ prc = 64 \end{cases}$$

$$f_{Pwm} = 1KHz = \frac{16^{MHz}}{64(256 - TCNT\emptyset)} \longrightarrow TCNT\emptyset = 6$$

$TCNT\emptyset$ ↓ 6

$$D.C = \frac{OCR\emptyset}{256 - TCNT\emptyset} \times 100\%$$

$TCNT\emptyset$

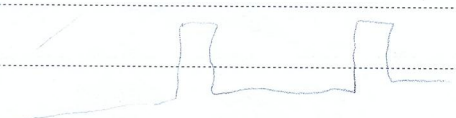
$$20\% = \frac{OCR\emptyset}{256} \times 100\% \longrightarrow OCR\emptyset = 50 \longrightarrow \text{Non-Inverting}$$

```
# include <mega16.h>
Interrupt [tim0_ovf] void timer0_ovf_isr(void){
    TCNT0 = 0x06;
}
void main(void){
PORTB = 0x00;
DDRB = 0x08;        Pb.3
TCNT0 = 0x06;
TCCR0 = 0x6B;
OCR0 = 0x32;        50 Decimal
TIMSK = 0x01;
# asm ("sei")
while (1){
    }
}
```

WGM00 Com01 Com00 WGM01

0	1	1	0	1	0	1	1	TCCR0

6 B



Subject:
Year. 1391 Month. 3 Date. 9 ( )

وقفه در حالت Compare :

اگر تایمر از مقدار اولیه شروع به شمارش کند، با رسیدن به مقدار مشخص OCR0، مقدار تایمر سرریز شده و دوباره از مقدار min (صفر) شروع به شمارش می کند.

با فعال کردن وقفه مربوط به تایمر (در حالت CTC) هر بار سرریز شدن زیر برنامه وقفه اجرا می شود.

مثال: برنامه ای بنویسید که با 200 بار شمارش کلاک تایمر، تایمر صفر را سرریز کند. (با وقفه)

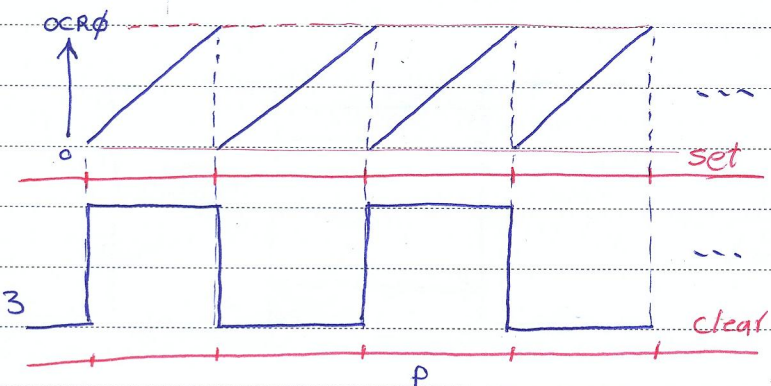
```
#include <mega16.h>
Interrupt [tim0_Comp] Void timer0_isr (Void)
{

// Place Your Code Here.

}
Void main(Void) {
TCCR0 = 0x01;
TCNT0 = 0x00;
OCR0 = 0xC8;
TIMSk = 0x02;
# asm ("sei")
while (1) {
}
}
```

PreScale = 1

$f_{xtal} = 1\ MHz$

هر بار 200 µs

- استفاده از حالت CTC برای تولید موج مربعی:



تایمر صفر از مقدار اولیه شروع به شمارش کرده و با رسیدن به مقدار OCR0 دوباره صفر می شود. و در این مدت زمان پایه PB.3 را Not می کند.

فرکانس موج ساخته شده در پایه Pb.3

$$f_{OCR} = \frac{f_{CLK\ IO}}{2 \times PreScale\ (1 + OCR0)}$$

Subject:
Year. Month. Date. ( )

- با مشخص کردن بیت های COM00 و COM01 می توان مشخص کرد که در فواصل شمارش ما پایه Pb.3 ، Not یا Clear یا Set شود.

COM01	COM00	
0	0	عدم عمل
0	1	NOT ، CTC
1	0	CTC و صفر Clear
1	1	CTC و یک Set

مثال: برنامه ای بنویسید که با استفاده از حالت CTC با فرکانس 1MHz بر روی پایه Pb.3 تولید کند.

$$\begin{cases} f_{xtal} = 16\ MHz \\ PreScale = 1 \end{cases}$$

$$1\ MHz = \frac{16\ MHz}{2 \times 1 \times (1 + OCR0)} \rightarrow OCR0 = 7$$

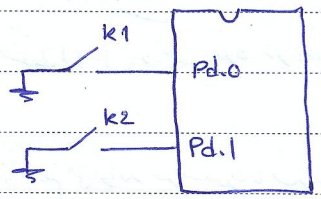
	0	0	1	1	0	0	1	TCCR0
	COM01	COM00	WGM01	CS02	CS01	CS00		
WGM00								

```
Void main(void){
    DDRB = 0x08;
    TCNT0 = 0x001
    TCCR0 = 0x19;
    OCR0 = 0x07;
    while(1){
        }
}
```

Subject:
Year. Month. Date. ( )

مثال: با فشردن کلید متصل به Pd.0 فرکانس افزایش
" " Pd.1 " " " " " کاهش

```
Void main (void) {
    DDRB = 0x08;
    DDRD = 0x00;
    TCNT0 = 0xC0;
    TCR0 = 0x19;
    OCR0 = 0x07;
    while (1) {
        if (Pind.0 == 0)   OCR0 = OCR0 - 100
        if (Pind.1 == 0)   OCR0 = OCR0 + 100
        delay_ms (500)
        }
    }
```

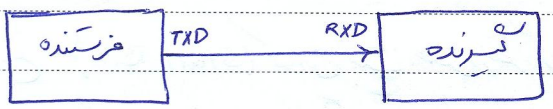


ارتباط سریال:

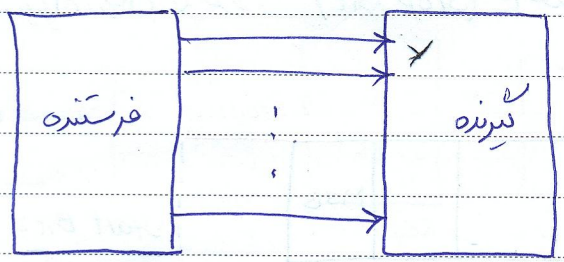
- در ارتباط سریال، اطلاعات به صورت بیت به بیت ارسال می‌شود یعنی در ارسال 8 بیت اطلاعات، هر بیت با یک کلاک پالس ارسال می‌شود. امّا در ارتباط موازی، اطلاعات در یک کلاک پالس ارسال می‌شود. پس سرعت ارتباط موازی از ارتباط سریال بیشتر است.

- در ارتباط سریال، برای ارسال به دو سیم احتیاج داریم { 1- دریافت Receive (RXD) ، 2- ارسال Transmint (TXD) }

- در ارتباط موازی به ازای هر بیت، یک سیم استفاده می‌شود، لذا در ارتباط موازی حجم و هزینه افزایش پیدا می‌کند.



ارتباط سریال
(Serial)



ارتباط موازی
(Parallel)

Subject:
Year. Month. Date. ( )

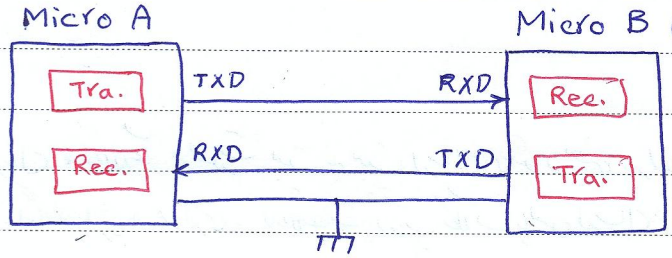
انواع ارتباط سریال:

۱- ارتباط Simplex (یک‌طرفه): فرستنده فقط ارسال و گیرنده دریافت می‌کند.

2- ارتباط Duplex (دوطرفه): هر دستگاه هم می‌تواند فرستنده باشد و هم گیرنده.

Duplex:
- Half_Duplex: ارسال و دریافت همزمان صورت نمی‌گیرد.
- Full_Duplex: ارسال و دریافت به صورت همزمان می‌تواند صورت گیرد.

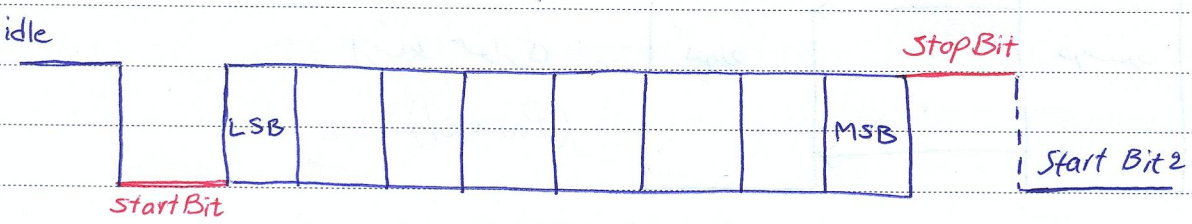
- میکروکنترلرهای AVR دارای ارتباط سریال از نوع Full_Duplex می‌باشد.



1391.3.16

- ارتباط سریال به دو صورت سنکرون و آسنکرون انجام می‌شود.
- در روش سنکرون هر بلوک می‌توان تعدادی داده را با کاراکتر ارسال کرد. امّا در روش آسنکرون در هر ارسال یک داده یا کاراکتر ارسال می‌شود.
- برای ارسال هر داده یا کاراکتر، یک بلاک Block (یا Frame) داده ارسال می‌شود که به صورت زیر است:

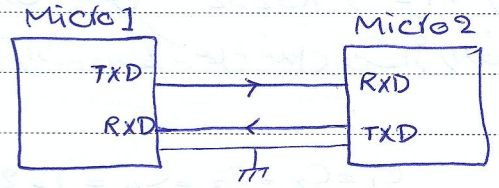
۱- بیت شروع (Start Bit) که این بیت صفر در نظر گرفته می‌شود.
2- بعد از بیت شروع داده ارسال می‌شود. از LSB تا MSB.
3- بیت پایان ارسال می‌شود. (Stop Bit) که تعداد بیت های پایان ۱ یا ۲ می‌باشد.



Subject:
Year. 1391 Month. 03 Date. 16 ( )

اتصال دو میکرو:

- برای ارسال اطلاعات، از معیار Baud Rate یا نرخ ارسال جهت تعیین سرعت استفاده می شود که بر حسب bps بیان می شود که بیانگر تعداد بیت های ارسالی در هر ثانیه است.
- بایستی نرخ ارسال در فرستنده و گیرنده به صورت مساوی تنظیم شود، در غیر این صورت اطلاعات به درستی منتقل نمی شود.



- ارتباط سریال در میکروکنترلرهای AVR از نوع USART می باشد:

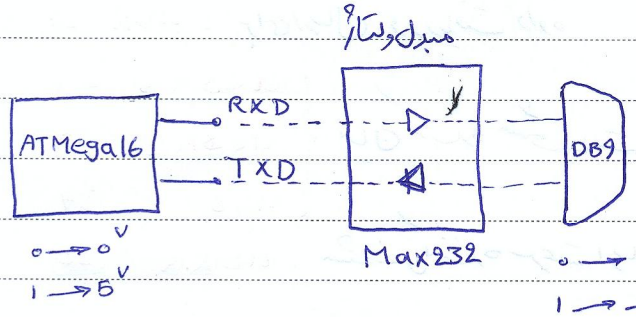
8051 → UART

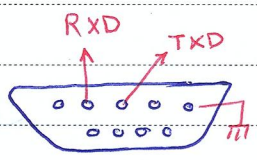
(Universal Synchronouse Asyn. Recieve Transmit)

- این پروتکل برای ارسال از میکرو به کامپیوتر و بالعکس تعریف شده است و دارای خواص زیر می باشد:

1- ارتباط دوطرفه Full Duplex
2- ارتباط سنکرون و آسنکرون
3- ارسال داده به صورت 5، 6، 7، 8 و یا 9 بیتی.
4- تولید و ارسال بیت توازن (Parity) به صورت زوج یا فرد.
5- ارسال بیت پایان (یک یا دو بیت)
6- ارسال وقفه به میکرو در هنگام ارسال یا دریافت

- برای برقراری ارتباط سریال میان میکرو و کامپیوتر از استاندارد RS232 استفاده می شود.
- در کامپیوتر، کانکتوری 9 پین (DB9) وجود دارد که با منطق RS232 کار می کند.
- در استاندارد RS232 با صفر منطقی معادل $3 \rightarrow 25^V$ و یک منطقی معادل $-3 \rightarrow -25^V$ است.



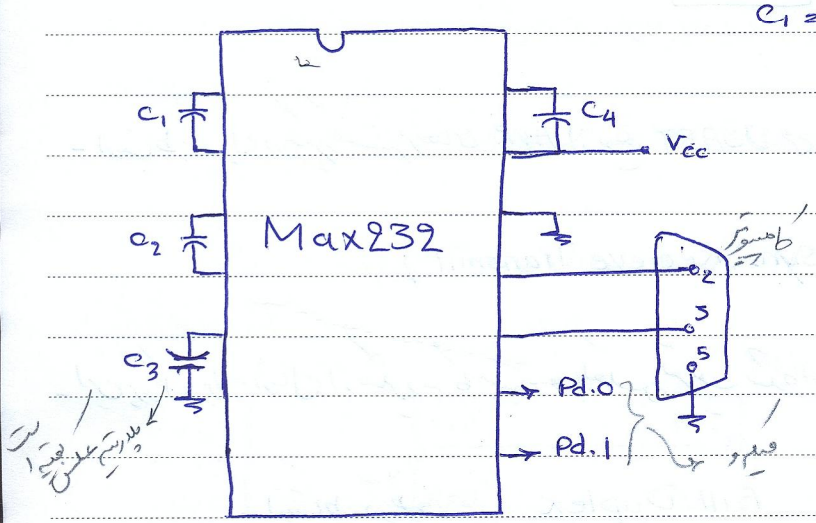


Subject:
Year. Month. Date. ( )

- در کانکتور DB9 پایه 2 پایه RXD و پایه 3، TXD می باشد.
- در میکرو پایه Pd.0 مربوط به دریافت (RXD) و پایه Pd.1 مربوط به ارسال (TXD) است.

- علت یک بودن منطق RS232 و TTL (متفاوت بودن سطح ولتاژ) ، برای اتصال میکرو به کامپیوتر باید از ترانسه واسط که عمل تبدیل ولتاژ را انجام می دهد، استفاده کرد.

$C_1 = C_2 = C_3 = C_4 = 1-22\mu F$



- بلاک دیاگرام ارتباط سریال دارای 3 بخش اصلی می باشد:

1- پالس ساعت: مشخص کننده سرعت ارسال و دریافت است.
2- قسمت گیرنده (RXD)
3- قسمت فرستنده (TXD)

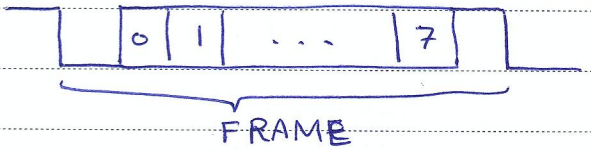
- رجیسترهای پورت سریال: پورت سریال دارای رجیسترهای زیر می باشد:

1- UDR : برای ارسال و دریافت داده
2- UCSRA
3- UCSRB
4- UCSRC
بیان کننده مشخصات ارتباط سریال
5- UBRR : مشخص کننده سرعت ارسال

Subject:
Year. Month. Date. ( )

- چگونگی ارسال در ارتباط سریال:

- برای ارسال داده 8 بیتی، ابتدا داده در رجیستر UDR نوشته می‌شود، سپس به رجیستر شیفت دهنده انتقال پیدا می‌کند و داده به صورت بیت به بیت تبدیل می‌شود و از طریق پایه TXD ارسال می‌شود.



- نحوه دریافت در ارتباط سریال:

- در گیرنده، اطلاعات به صورت بیت به بیت در رجیستر شیفت دهنده دریافت شده و پس از دریافت کامل فریم داده، اطلاعات در رجیستر UDR ذخیره می‌شود.

۱- رجیستر ورودی-خروجی داده: (UDR)

Universal I/O Data Register

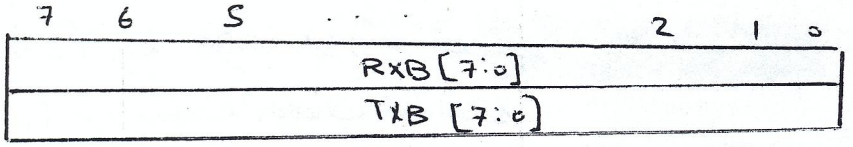
این رجیستر برای ارسال و دریافت داده استفاده می‌شود. این رجیستر دارای دو بافر RXB و TXB است.

در RXB اطلاعات دریافتی ذخیره می‌شود.
در TXB اطلاعات ارسالی نوشته می‌شود.

ثبات های پورت سری USART در میکرو کنترلرهای AVR :

۱- ثبات داده ورودی خروجی UDR (USART I/O Data Register)

- این ثبات، برای دریافت و ارسال اطلاعات است:
  - دارای یک بافر ارسال TxB است که اطلاعات ارسالی در آن نوشته می‌شود.
  - " " " دریافت RxB " " " ذخیره می‌گردد.



* به محض اینکه اطلاعات در ثبات UDR نوشته شده، و در صورتیکه قسمت فرستنده USART فعال باشد، اطلاعات مذکور به ثبات شیفت دهنده ارسال، منتقل می‌شود و از طریق پایه TXD به صورت سری، (یک بیت یک بیت) ارسال می‌گردد.

۲- ثبات های کنترل و وضعیت UCSRA، UCSRB و UCSRC :

این ثبات ها برای کنترل و مشخص نمودن طرز کار پورت سری به کار برده می‌شود. مثلاً با این ثبات ها می‌توان USART را در حالت سنکرون، آسنکرون قرار داد، تعداد بیت های تبادل اطلاعات را مشخص کرد، وقفه را فعال یا غیرفعال کرد و ...

۲-۱- ثبات کنترل و وضعیت UCSRA (USART Control and Status Register A)

این ثبات شامل بیت های پرچم دریافت و ارسال و بیت های پرچم وقوع خطا است.

7	6	5	4	3	2	1	0	
RxC	TXC	UDRE	FE	DOR	PE	U2x	MPCM	UCSRA

* بیت ۷ - بیت پرچم دریافت کامل کاراکتر (RxC) (USART Recieve Complet)

زمانی که اطلاعات سری به بافر دریافت ثبات UDR رسید، و در آن قرار گرفت، آن وقت بیت پرچم دریافت اطلاعات کامل یک کاراکتر برابر 1 می‌شود. یک شدن RxC به معنای حاضر بودن اطلاعات دریافتی در UDR است.

* اگر اطلاعاتی در ثبات UDR نباشد، بیت پرچم RxC برابر صفر است.

* بیت (۶) : بیت پرچم پایان ارسال کاراکتر TXC (USART Transmit Complet)

هنگامی که یک کاراکتر درثبات بافر UDR ، از طریق پایه TXD ارسال شده، آنگاه بیت پرچم ارسال TXC = ۱ می شود که نمایانگر پایان ارسال کاراکتر است.

* بیت (۵) : بیت پرچم خالی بودن ثبات داده : UDRE
USART Data Register Empty

زمانی که بیت UDRE برابر یک شده، نشانه این است که ثبات بافر ارسال UDR خالی است، وآماده دریافت اطلاعات جدید است.

* بیت (۴) : بیت پرچم خطای FE : (Frame Error)

موقعی که بیت پایان در انتهای کاراکتر برابر ۰ باشد، یعنی اطلاعات یک بایت تکمیل نشده

لذا بیت خطای FE = ۱ می شود.

** زمانی که اطلاعاتی درثبات UCSRA نوشته می شود، این بیت صفر نوشته می شود.

* بیت (۳) : بیت پرچم خطای DOR : (Data over Run)

زمانی که بافر دریافت پر باشد و کاراکتر دیگر کجا هد ارسال شود، یعنی بیت شروع صفر شود،

دراین صورت DOR = ۱ می شود.

** موقعی که اطلاعاتی درثبات UCSRA نوشته می شود، این بیت صفر نوشته می شود.

* بیت (۲) بیت پرچم توازن PE (Parity Error)

اگر بیت توازن دریافتی اشتباه باشد، PE برابر یک می شود

درموقع نوشتن در UCSRA، دراین بیت باید صفر نوشت.

* بیت (۱) : بیت دوبرابر کردن سرعت ارسال : Double the USART transmission Speed

با یک کردن این بیت سرعت ارسال، ۲ برابر می گردد.

* در حالت سنکرون این بیت صفر است.

* بیت (۰) : حالت مولتی پروسسور : Multi - Processor Communication Mode

وقتی این بیت یک شد، یعنی حالت مولتی پروسسور استفاده می شود.

۲-۲) ثبات کنترل وضعیت UCSRB :

این ثبات برای فعال کردن وقفه و فعال کردن فرستنده و گیرنده است :

7	6	5	4	3	2	1	0
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RxB8	TxB8

* بیت (۷) : بیت فعال کردن وقفه دریافت RXCIE

Rx Complete Interrupt Enable.

با یک کردن این بیت، درصورتیکه بیت پرچم دریافت RXC در ثبات UCSRA برابر یک باشد، و وقفه کلی فعال شده باشد، آن وقت وقفه دریافت اطلاعات فعال می‌شود.

* بیت (۶) : بیت فعال کردن وقفه ارسال TXCIE

با یک کردن این بیت، درصورتیکه بیت پرچم TXC در ثبات UCSRA برابر یک باشد، و وقفه کلی فعال شده باشد، آن وقت وقفه ارسال اطلاعات فعال می‌شود.

* بیت (۵) : بیت فعال کردن وقفه در حالت خالی بودن ثبات داده UDRIE :

با یک کردن این بیت، اگر بیت پرچم خالی بودن ثبات داده (UDRE) در ثبات UCSRA برابر یک باشد و وقفه کلی فعال شده باشد آنگاه وقفه خالی بودن ثبات داده فعال می‌باشد.

* بیت (۴) : بیت فعال کردن گیرنده RXEN (Rx Enable)

برای فعال کردن قسمت گیرنده پورت سری USART باید RXEN را برابر یک کرد.

در این صورت پایه RxD، آماده دریافت اطلاعات می‌شود.

* بیت (۳) : بیت فعال کردن فرستنده TXEN

با بیت فعال کردن قسمت فرستنده باید TXEN را برابر یک کرد.

در این حالت پایه TxD، آماده ارسال می‌شود.

* بیت (۲) : تعداد بیت‌های یک کاراکتر UCSZ2 (Character Size)

بیت UCSZ2 و UCSZ1 و UCSZ0 در ثبات UCSRC تعداد بیت‌های کاراکتر را مشخص می‌کند.

* بیت (۱) : نهمین بیت دریافت RXB8

هنگامی که تبادل اطلاعات ۹ بیتی انتخاب شود، در این صورت بیت RXB8 نهمین بیت کاراکتر دریافتی خواهد بود.

* بیت (۰) : نهمین بیت ارسال TXB8

اگر تبادل اطلاعات ۹ بیتی باشد، TXB8، نهمین بیت کاراکتر ارسالی خواهد بود.

۲-۳) ثبات کنترل و وضعیت UCSRC :

با این ثبات، سنکرون یا آسنکرون بودن، تعداد بیت ارسالی، فعال کردن بیت توازن، را مشخص می‌کنیم:

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL

(USART Mode Select)

* بیت (۷) : انتخاب ثبات :

زمانی که با ثبات UCSRC کار می‌کنیم، این بیت را برابر یک قرار می‌دهیم.

* بیت (۶) : انتخاب حالت سنکرون یا آسنکرون:

UMSEL ← ۰ ← آسنکرون
۱ ← سنکرون.

* بیت (۴ و ۵) : انتخاب حالت بیت توازن :

UPM1	UPM0	حالت بیت توازن
۰	۰	غیرفعال
۰	۱	زوج شده
۱	۰	بیت توازن فرد فعال شود
۱	۱	

اگر این بیت فعال شود، فرستنده، صورت اتوماتیک، این بیت را همراه با کاراکترهای ارسال می‌کند، گیرنده بررسی کرده و در صورت ارسال اشتباه، پرچم FE در رجیستر UCSRA را یک می‌کند.

* بیت (۳) : انتخاب تعداد بیت های پایان. (USART Stop bits Select)

USBS ← ۱ ← دو بیت پایان
۰ ← یک ″ ″

* بیت (۱ و ۲) . تعیین کننده تعداد بیت های کاراکتر :

UCSZ2	UCSZ1	UCSZ0	تعداد بیت ها
۰	۰	۰	۵
۰	۰	۱	۶
۰	۱	۰	۷
۰	۱	۱	۸
۱	۱	۱	۹

* بیت (۰) : انتخاب پلاریته پالس در حالت سنکرون Clock Polarity

این بیت در حالت سنکرون استفاده می‌شود . در حالت آسنکرون باید صفر باشد .

UCPOL	لحظه تغییر اطلاعات ارسالی روی پین TXD	لحظه نمونه برداری اطلاعات ورودی گیرنده
۰	در لبه بالا رونده پالس ساعت XCK	در لبه پایین رونده پالس
۱	" " پایین رونده "	" بالا " "

- فرستنده پورت سری USART :

* فرستنده USART با فعال کردن بیت TXEN در ثبات UCSRB فعال می‌شود، و این صورت پایه TXD میکرو برای پورت سری خروجی می‌گردد.

* قبل از ارسال اطلاعات باید به ثبات های پورت سری مقدار اولیه داده شود.

- دریافت اطلاعات توسط گیرنده :

* با یک کردن بیت فعال ساز RXEN در ثبات کنترل UCSRB، گیرنده پورت سری USART فعال می‌شود، و این صورت پایه به صورت اتوماتیک پایه RXD میکروکنترلر، برای ورودی اطلاعات سری USART تعریف می‌شود.

* قبل از دریافت یا ارسال اطلاعات، باید با مقدار اولیه دادن به ثبات ها، سرعت ارسال اطلاعات (B.R)، تعداد بیت ها و ... را تعیین کرد.

---

* تولید کننده پالس داخلی : Internal Clock Generation : The Band Rate Generator

پالس ساعت میکروکنترلر OSC، به قسمت تولید کننده پالس تبادل اطلاعات یا Band Rate داده می‌شود متناسب با مقداری که در ثبات کنترل UBRR قرار می‌دهیم، پالس ساعت تبادل اطلاعات مطابق فرمول زیر تولید می‌شود.

$$\text{Band Rate} = \frac{f_{osc}}{16(UBRR+1)}$$

f_{osc} : فرکانس اسیلاتور

UBRR : مقداری که در ثبات UBRR قرار می‌دهیم.

مثلاً اگر $f_{osc}=1^{MHz}$ باشد و UBRR = 25 باشد جواب چه است

$$\text{Band Rate} = \frac{1^{M}}{16(25+1)} \simeq 2400\ Hz$$

یعنی سرعت تبادل اطلاعات، 2400 bit/s است.

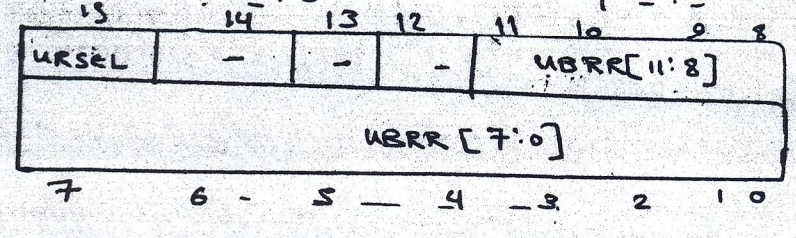
* با یک کردن بیت U2X در ثبات کنترل UCSRA، سرعت ارسال و دریافت ۲ برابر می‌شود یعنی:

$$\text{Band Rate} = \frac{f_{osc}}{8(UBRR+1)}$$

* با استفاده از جدول مربوط، با داشتن سرعت های مختلف پورت سری (Band Rate های متداول مانند 2400، 4800، 9600 و ...) و با داشتن فرکانس پالس ساعت ۱ و 2، 4، 8 ^{MHz} مقدار ثبات UBRR تعیین می‌شود و در آن قرار داده می‌شود.

۳- ثبات تعیین سرعت پورت سری UBRR :

ثبات ۱۶ بیتی UBRR ، دارای یک بایت کم ارزش UBRRL و یک بایت پر ارزش UBRRH است :



مطابق با جدول مربوطه با مقداردهی ثبات UBRR ، B.R. را سال مشخص می شود .

* بیت (۱۵): بیت انتخاب ثبات (Register Select)

با این بیت می توان ثبات UBRRH یا UCSRC را انتخاب نمود :

وقتی که اطلاعات از ثبات UBRRH خوانده می شود ، این بیت برابر صفر است .

در ثبات " نوشتن " " ۱ "

* بیت (۱۱) تا (۰) : ثبات سرعت پورت سری :

بیت های ۰ - ۱۱ یک ثبات ۱۲ بیتی است که مشخص کننده Baud Rate است .

ثبات UBRRH شامل ۴ بیت پر ارزش و ثبات UBRRL شامل ۸ بیت کم ارزش است .

شلاً اگر $f_{OCS}=8^{MHz}$ و Baud Rate=%۰۰ باشد ، مطابق جدول بالایی ، UBRR=51 یا 0x33

انتخاب میشود که در این حالت UBRRH=0x00 و UBRRL=0x33 است .

* برنامه‌ای بنویسید که در میکروکنترلر AVR ،

• پورت سری USART اعداد 0x77 ، 0x88 را در هر 500ms از پایه TXD میکرو ارسال کند.

• اگر عدد 0x55 در پایه RxD میکرو رسید، بیت 6 پورت D را یک کند و اگر عدد 0x66 رسید بیت 2 پورت D را صفر کند. (8 بیتی آسنکرون- توازن نداریم و یک S.B داریم)

$f_{XTAL} = 8M$
$B.R = 56000$

مقداردهی رجیسترها

1) UCSRA = 0x00

2)

7				4			
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	~~TXB8~~
1	0	0	1	1	0	0	0

RXCIE (7) = 1 ← اگر بیت پرچم RxC در ثبات UCSRA برابر یک شود وقفه سریال دریافت و زیربرنامه وقفه اجرا می‌شود (اگر 0x55 دریافت شده، پورت D را یک کرده و اگر 0x66 رسید، صفر می‌کند)

3و4 ← فعال کردن فرستنده و گیرنده.

UCSRB = 0x98

3) UCSRB = 0x86

URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
1	0	0	0	0	1	1	0

URSEL ← 1 ← کار با ثبات UCSRC
UMSEL ← 0 ← آسنکرون
UPM1 ، UPM0 ← 0 ، 0 ← بیت توازن نداریم
UCSZ1 و UCSZ0 = 1 ، UCSZ2 = 0 ← 8 بیتی
USBS = 0 ← یک S.B داریم.

4) $56000 = \frac{8 * 10^6}{16\,(UBRR+1)} \longrightarrow UBRR \simeq 8$

⇒ UBRRH = 0x00
UBRRL = 0x08

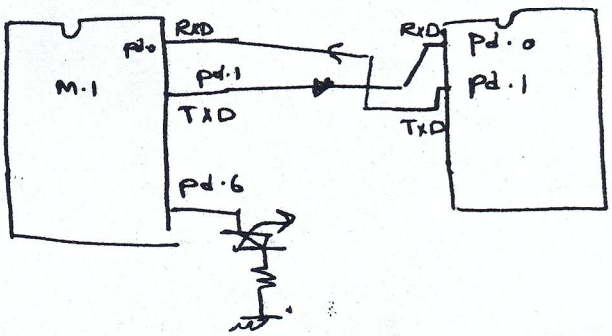
```
#include <mega8.h>
#include <delay.h>
#include <stdio.h>

interrupt [USART_RXC] void USART_rx_isr(void)
{
  char data;
  data = UDR;
  if (data =0x55)
        portd.6 =1;
  if (data = 0x66)
        portd.6 =0;
}

void main, void {

DDRD = 0x40 ;
UCSRA = 0x00;
UCSRB = 0x98;
UCSRC = 0x86;
UBRRH = 0x00;
UBRRL = 0x07;

#asm ("sei")
  while (1) {
    UDR = 0x77;
    delay_ms (500);
    UDR = 0x88
  } delay_ms (500)

}
```

A

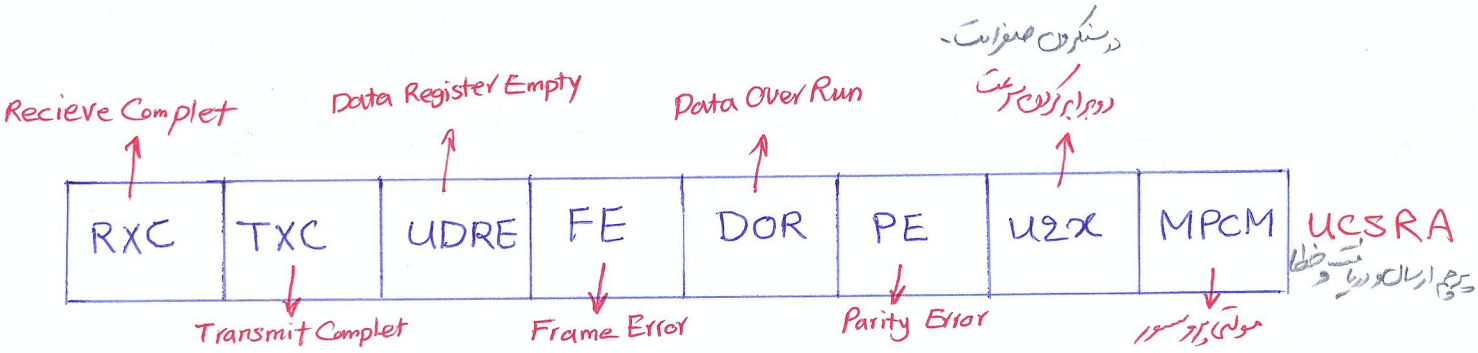
B

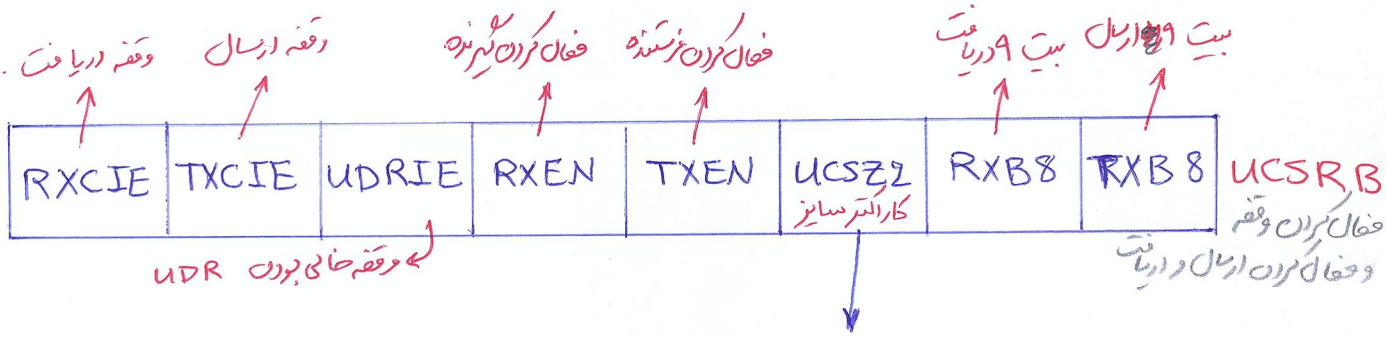
C

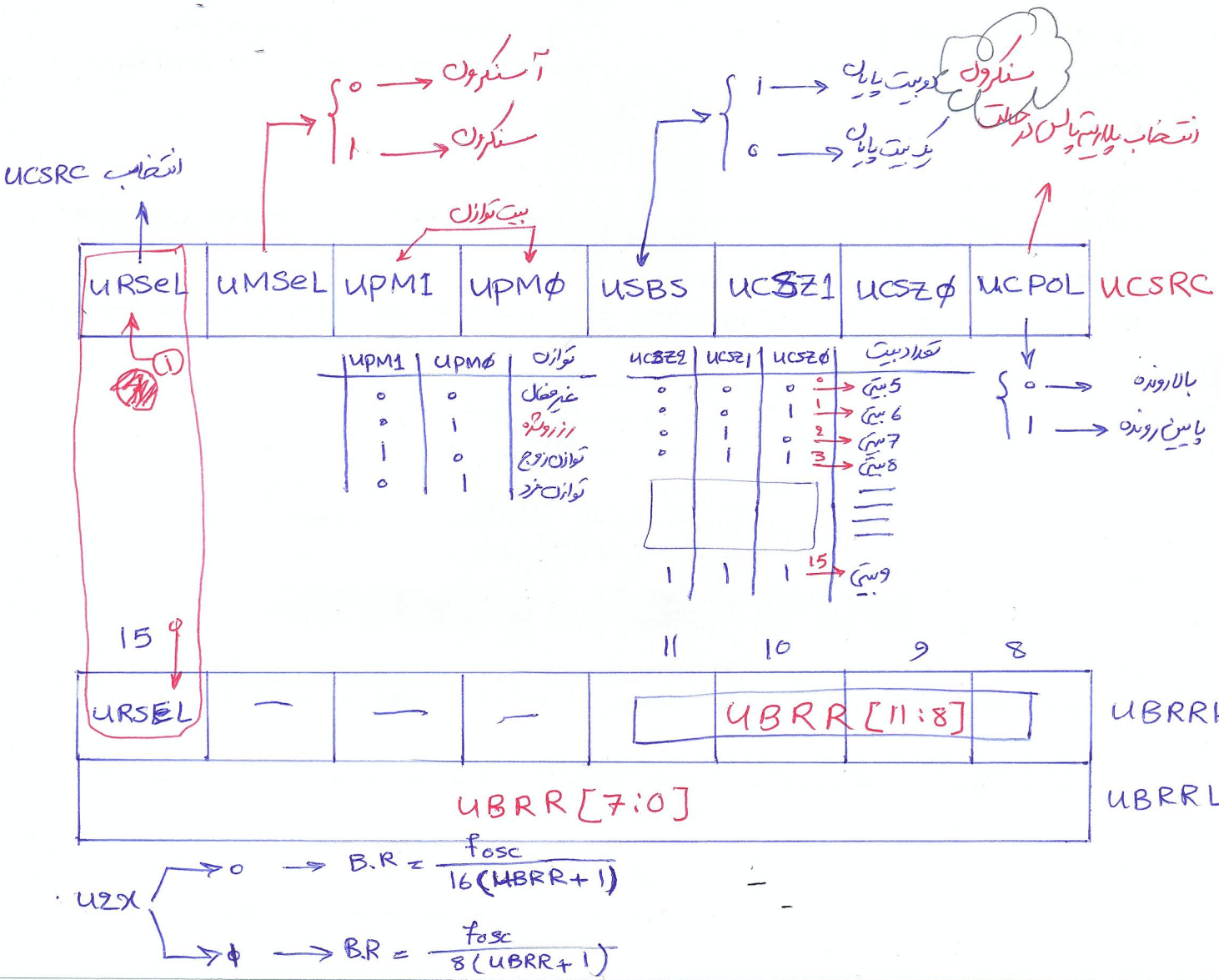
* حلقه while مرتباً تکرار می‌شود، تا وقفه رخ دهد.
در قسمت C، مرتباً 0x77 و 0x88 ارسال می‌گردد.
* اگر در این مدت اطلاعاتی از میکرو دیگر به پایه ورودی دریافت RxD برسد، روتین سرویس وقفه یعنی بخش A اجرا می‌شود.



ثبات داده ورودی - خروجی RXB / TXB } ← UDR







$f_{CTC} = \frac{f_{xtal}}{2 \times Pre\,(1+OCR\emptyset)}$

CTC فرکانس

```
#asm ("sei")
```

FOC0	WGM01	COM01	COM00	WGM01	CS02	CS01	CS00

TCCR∅

CS02	CS01	CS00	
0	0	0	0 → timer Stopped
0	0	1	1 → Pre = 1
0	1	0	2 → Pre = 8
0	1	1	3 → Pre = 64
1	0	0	4 → Pre = 256
1	0	1	5 → Pre = 1024
1	1	0	6 → کانتر ↑
1	1	1	7 → کانتر ↓

PWM

COM01	COM00	
0	0	تایمر در حالت Normal
0	1	CTC (toggle)
1	0	Non-Inverting
1	1	Inverting

WGM01	WGM00	حالت تایمر	Max
0	0	Normal	0xFF
1	0	ph.Cor. PWM	0xFF
0	0	CTC	OCR∅
1	1	Fast PWM	0xFF

$f = \frac{f_{xtal}}{Pre \times 510}$

$= \frac{f_{xtal}}{Pre(256 - TCNT\emptyset)}$

COM01	COM00	
0	0	Normal
0	1	CTC (toggle) not
1	0	CTC (Clear)
1	1	CTC (Set)

Pb.3 = out

```
Interrupt [tim∅_Comp] Void timer∅_Comp_isr(Void)
{
//Codes
}
```

COIE2	TOIE2	IOC1	COIE1B	COIE1A	TOIE1	COIE∅	TOIE∅

TIMSK وقفه

تایمر 2 — تایمر صفر — تایمر یک

سرریز CTC — سرریز تایمر صفر (Normal)

```
Interrupt [tim∅_ovF] Void timer∅_OVF_isr(Void)
{
// Codes
}
```

OCF2	TOV2	TCF1	OCF1B	OCF1A	TOV1	OCF∅	TOV∅

TIFR

پرچم سرریز CTC — پرچم سرریز Normal

پرچم

$$f_{clk\,timer} = \frac{f_{xtal}}{Pre\,Scale}$$

تایمر صفر ⟶ $\frac{0xFF - TCNT0}{t_{clk\,Timer}} =$ تعداد شمارش

زمان یک بار سرریز شدن ⟶ تعداد شمارش $\times T_{clk\,timer}$

تعداد دفعات سرریز OVF $=$ $\frac{\text{زمان مطلوب}}{\text{زمان شمارش}}$

مثال: ساخت یک تاخیر با تایمر صفر

$f_{clk\,timer} = \frac{16\,MHz}{1} = 16\,MHz$

$T = 0.0625\ \mu s$

$1^s = 4 \times 250^{ms}$ TCNT0 = 56

تعداد شمارش = FF − 56 = 200

زمان یک بار سرریز = $200 \times 0.0625 = 12.5\,\mu s$

تعداد دفعات سرریز OVF = $\frac{1^s}{12.5\,\mu s} = 80,000$