به نام خدا

(جلسه نهم آموزش AVR)

(Assembly 4)

مقدمه:

در جلسه قبل با وقفه ها و نحوه رسیدگی به رویداد وقفه خارجی آشنا شدیم. در این جلسه در مورد تایمرها و کاربردهای آنها و نیز پورت سریال برای ارتباط با دنیای خارج بحث خواهیم کرد. همانطوریکه در مباحث Bascom اشاره شد تایمرها کاربردهای بسیاری دارند؛ از ایجاد تاخیر گرفته تا شمارش رویدادها و اندازه گیری زمان. در این مقاله در حد مختصری با نحوه راه اندازی تایمرها با زبان اسمبلی آشنا می شویم. بخش دیگری که در این مقاله مورد توجه است پورت سریال میکرو می باشد. پورت سریال در مد آسنکرون کاربردهای بسیار زیادی برای ارتباط با قطعات جانبی داشته و به میکرو این امکان را می دهد تا با کامپیوتر و سایر ماژول های بیسیم و قطعات جانبی سازگار با پروتکل RS232 ارتباط برقرار کند.

تایمرها:

در هر میکروکنترلر چندین تایمر مختلف با عرض ۸ یا ۱۶ بیت وجود دارد. در میکروی نمونه ATmega32 که در این مقاله از آن استفاده می نماییم ۲ تایمر ۸ بیتی و یک تایمر ۱۶ بیتی قرار دارد. تایمر شماره صفر و دو ۸ بیتی بوده و تایمر شماره یک ۱۶ بیتی می باشد. در این مقاله با تایمر شماره صفر میکرو کار می کنیم. در ابتدا برنامه ای برای ایجاد تاخیر یک ثانیه ای با استفاده از تایمر صفر می نویسیم.

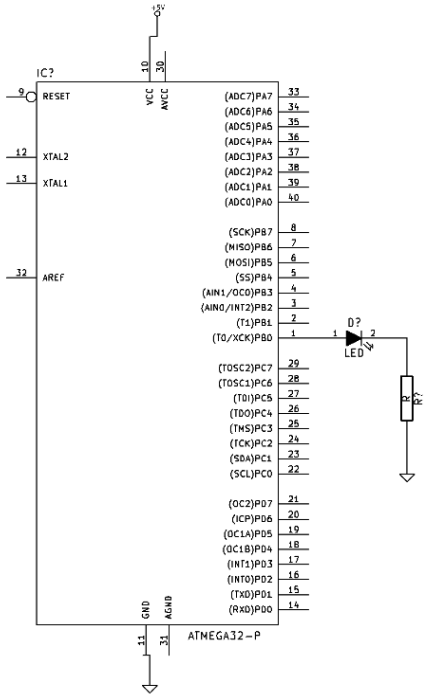
برای ایجاد تاخیر یک ثانیه ای بایستی فرکانس CPU را روی ۱ مگاهرتز بسته و تقسیم کننده تایمر را بر روی ۶۴ قرار دهیم. در این صورت تایمر با فرکانس ۶۴ ÷ ۱۰۰۰۰۰۰ یعنی ۱۵۶۲۵ هرتز تغذیه می شود و با توجه به ۸ بیتی بودن تایمر این عدد تقسیم بر ۲۵۶ می شود و عدد ۶۱ با تقریب یک رقم اعشار به دست می آید. یعنی تایمر صفر در هر ثانیه ۶۱ بار سرریز می شود و برای ایجاد فرکانس یک هرتز بایستی در هر بار سرریزی، رجیستری را افزایش دهیم و هر وقت به ۶۱ رسید LED را روشن و خاموش کنیم تا فرکانس ۱ هرتز را در خروجی ببینیم.

برای این کارها نیاز به تنظیم تایمر صفر داریم. یکی از رجیسترهای تایمر صفر TCCR0 نام دارد که تنظیمات بخش تقسیم فرکانس به این رجیستر مربوط می شود. برای تقسیم فرکانس میکرو بر ۶۴ مطابق با دیتا شیت میکروی مگا ۳۲ بایستی بیتهای ۰ و ۱ این رجیستر را با ۱ تنظیم کرده و بقیه را ۰ کنیم. رجیستر بعدی TIMSK بوده که بیت ۰ آن مربوط به فعال کردن وقفه سرریزی تایمر است. با تنظیم این بیت ها و نیز نوشتن قطعه کد فلش زدن LED و آدرس دهی آن به وکتور وقفه سرریزی تایمر ۰ که در این میکرو در آدرس ۱۶ هگز ابتدای رم قرار دارد مشاهده می کنیم که با هر سرریزی تایمر روتین وقفه فراخوانی شده و از آنجا که این فراخوانی هر ثانیه ۶۱ بار انجام می پذیرد این عمل با یک عبارت شرطی در روتین وقفه بر ۶۱ تقسیم می شود و شاهد فلش زدن LED در هر ثانیه یک بار خواهیم بود.

تایمرها برای تولید تاخیرهای دقیق (البته با کریستال خارجی)، شمارش رویدادها (با اعمال پالس به پایه ورودی تایمر)، تولید موج مربعی، تولید پالس PWM و غیره به کار می روند که ذکر تمامی آنها در این مقاله نمی گنجد. همانطوریکه می دانید هدف ما از این سری مقالات فقط آشنایی مقدماتی با نحوه راه اندازی امکانات داخلی میکرو می باشد و علاقمندان بایستی در صورت تمایل به مطالعه دیتا شیت و مراجع ذکر شده در انتهای هر مقاله بپردازند. شایان ذکر است این آشنایی مختصر پیشنیاز زبان های سطح بالا می باشد.

برنامه کامل راه اندازی تایمر صفر را در تصویر زیر مشاهده می فرمایید:

```
; Timer0 program by Behnam Zakizadeh
; www.avr64.com @ 17.july.2011
.include "Appnotes\m32def.inc"

.CSEG
.ORG 0x00
rjmp start

.org 0x16
;Ov0 vector
rjmp Ov0_int

.org 0x30
start:
; set stack pointer
ldi r20, low(RAMEND)
out SPL, r20
ldi r20, high(RAMEND)
out SPH, r20

sei ;enable Global interrupts
ldi r20, 0b00000001 ;enable overflow int timer0
out TIMSK, r20 ;enable overflow int timer0
ldi r20, 0b00000011 ;prescaler = 64
out TCCR0, r20 ;prescaler = 64

; set as output (LED)
sbi ddrb,0

ldi r16, 61

end:
rjmp end

Ov0_int:
dec r16
brne skip
ldi r16, 61
sbi portb,0
nop
nop
nop
cbi portb,0
skip:
reti
```



شماتیک پروژه راه اندازی تایمر صفر میکرو با زبان اسمبلی

پورت سریال:

پورت سریال یکی از امکانات ارتباطی میکرو برای اتصال ادوات جانبی، کنترل دستگاه های خارجی، ارسال اطلاعات به کامپیوترهای استاندارد، دریافت اطلاعات از کامپیوتر، ارتباط با ماژول های بیسیم و هزاران کاربرد دیگر می باشد. در میکرو پروتکل های ارتباطی مختلفی مانند SPI و I²C نیز وجود دارد که برای ارتباط با حافظه های SD/MMC و E²PROM های سریال به کار میرود. برای تمام این پروتکل ها هدرفایل های آماده و رایگان به زبان C موجود می باشد و در هر کدام از این مقالات فقط با یکی از پرکاربرد ترین بخشهای میکرو و راه اندازی آن با زبان سطح پایین آشنا می شویم تا از بیان مطالب اضافه خودداری کرده باشیم. در این قسمت پورت سریال را در مد آسنکرون فعال کرده و کاراکتر A را به سمت کامپیوتر ارسال می کنیم. قبل از هر چیز بایستی رجیسترهای پورت سریال را مورد بررسی قرار دهیم.

پورت سریال از چندین رجیستر مختلف تشکیل شده است. رجیسترهای UBRRH و UBRRL برای تنظیم Baud ارتباط به کار میروند. عددی که بر حسب فرکانس های مختلف بایستی در رجیسترهای بالا نوشته شود با استفاده از فرمول زیر محاسبه میگردد:

$$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$$

به طور مثال برای فرکانس ۱ مگاهرتز میکرو در صورتی که بخواهیم باود ارتباطی ما 2400bps باشد داریم: $\frac{۱۰۰۰۰۰۰}{۱۶(x+۱)} = ۲۴۰۰$ برای حل این معادله یک مجهولی به ۲۴۰۰ مخرج ۱ می دهیم و ۱۰۰۰۰۰۰ را در ۱ ضرب میکنیم. ۲۴۰۰ را نیز در ۱۶ ضرب میکنیم تا معادله را کوچکتر کنیم:

$$۱۰۰۰۰۰۰ = ۳۸۴۰۰(x+۱) \Rightarrow (x+۱) = ۱۰۰۰۰۰۰ / ۳۸۴۰۰$$

$$(x+۱) = ۲۶/۰ \quad \Rightarrow \quad x = ۲۶ - ۱ \quad \Rightarrow \quad x = ۲۵$$

پس از حل معادله بالا عدد ۲۵ را به دست می آوریم که بایستی آن را در دو رجیستر بالا قرار دهیم. نظر به اینکه ۲۵ از ۲۵۵ کوچکتر بوده و از ۸ بیت تجاوز نمی کند آن را منحصراً در رجیستر کم ارزشتر یعنی UBRRL قرار می دهیم و پر ارزش تر را برابر با صفر قرار می دهیم. (در صورتی که جواب معادله از ۲۵۵

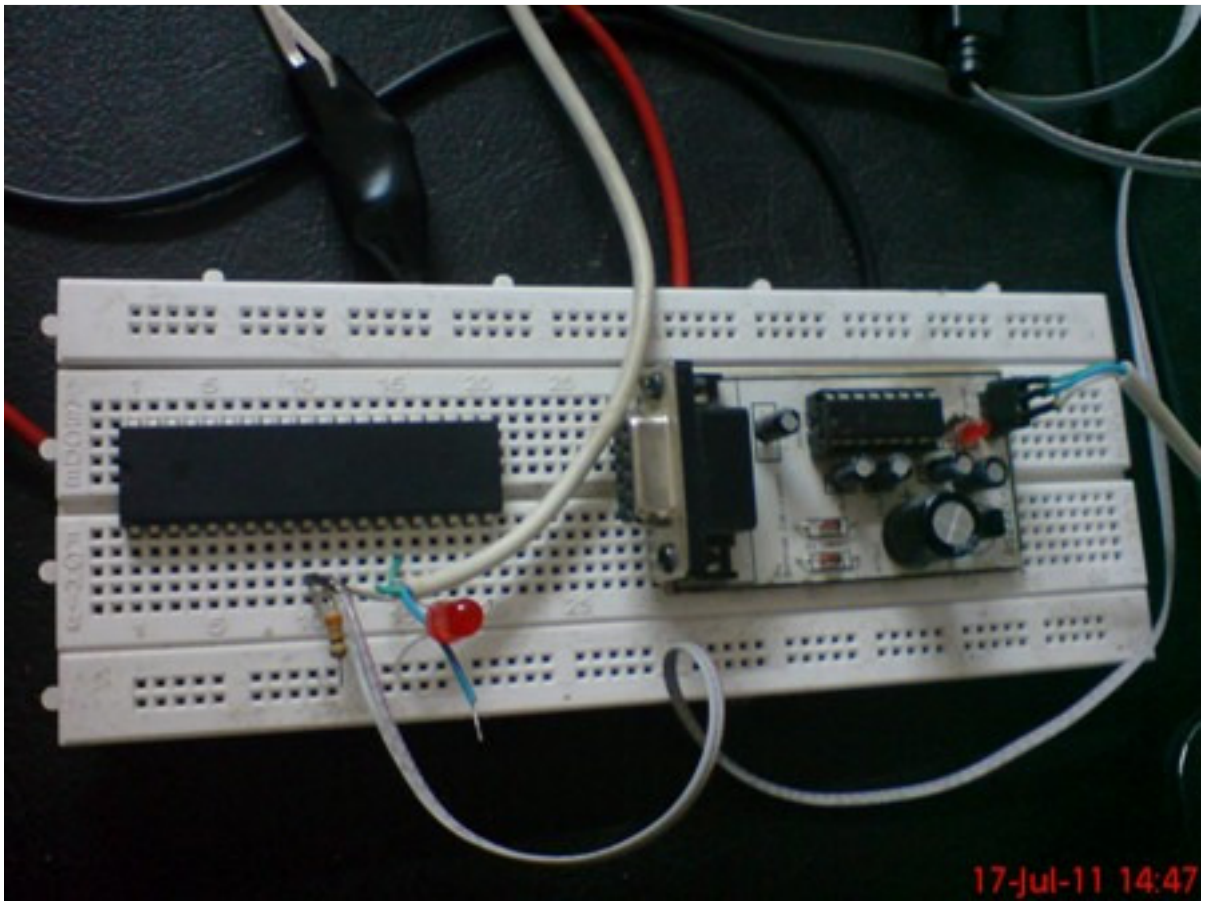
بیشتر باشد بایستی همانند نحوه تنظیم سایز پشته و دستورات Low و high با آن رفتار کنیم. (در کامپایلرهای سطح بالا تمام این کارها به صورت خودکار انجام می شود).

مرحله بعدی فعال کردن فرستنده (و گیرنده) است. برای این منظور بایسستی بیت های ۳ و۴ رجیستر UCSRB را با ۱ تنظیم کنیم. تنظیمات رجیسترهای دیگر به صورت پیشفرض مناسب است: دیتای ۸ بیتی و ۱ بیت توقف و بدون بیت توازن. مرحله بعد ارسال کاراکتر می باشد. برای ارسال ابتدا بایستی بیت شماره ۵ رجیستر UCSRA بررسی شود. در صورتی که این بیت ۱ باشد به معنای خالی بودن بافر فرستنده و آمادگی ارسال می باشد. در این صورت کاراکتر مورد نظر را داخل یک رجیستر قرار داده و آن را به رجیستر UDR ارسال می کنیم.

```
; USART program by Behnam Zakizadeh
; www.avr64.com @ 17.july.2011
.include "Appnotes\m32def.inc"

.CSEG
.ORG 0x00
rjmp start
.org 0x30
start:
; set stack pointer
ldi r20, low(RAMEND)
out SPL, r20
ldi r20, high(RAMEND)
out SPH, r20

;set baud rate 2400
ldi r20, 25
out UBRRL, r20
ldi r20, 0
out UBRRH, r20

;enable TXD & RXD
ldi r20, 0b00011000
out UCSRB, r20

ldi r19, 'A'

USART_Transmit:
;Wait for empty transmit buffer
sbis UCSRA,UDRE
rjmp USART_Transmit
;Put data (r19) into buffer, sends the data
out UDR,r19
call delay
rjmp USART_Transmit

delay:
ldi r16, 10
loop1: ldi r17, 255
loop2: ldi r18, 255
loop3: dec r18
```

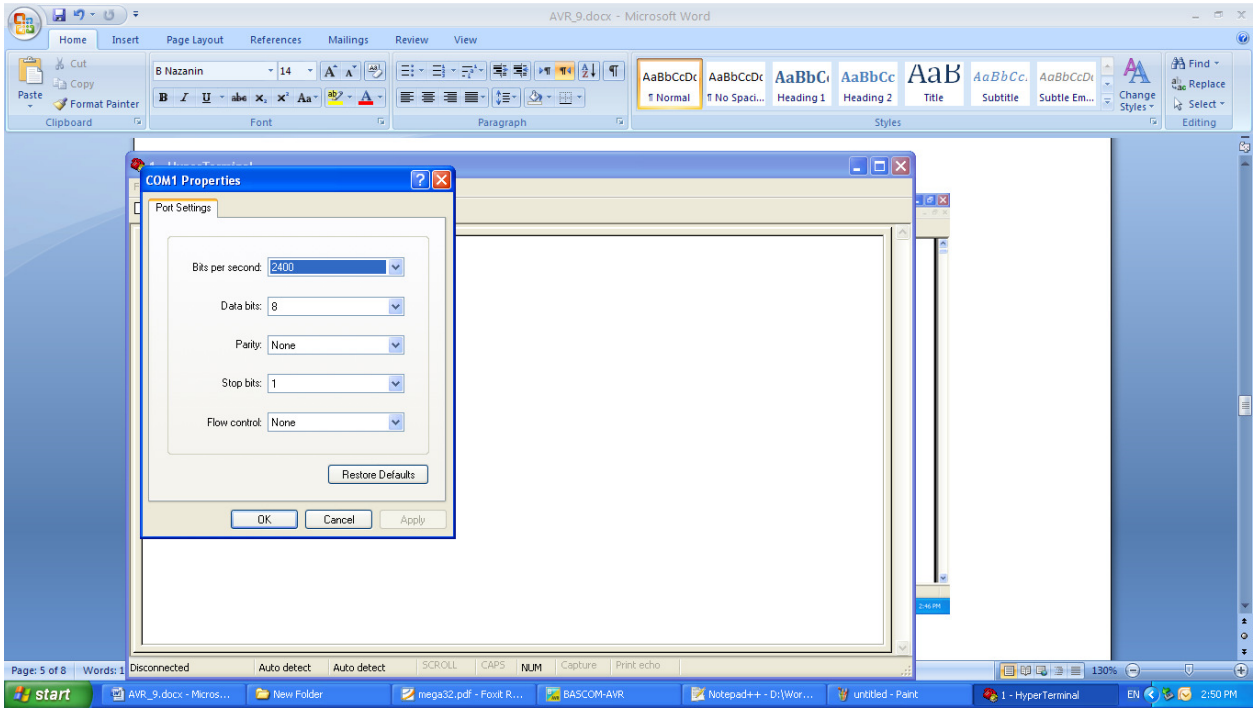
```
    brne loop3
    dec r17
    brne loop2
    dec r16
    brne loop1
ret
```

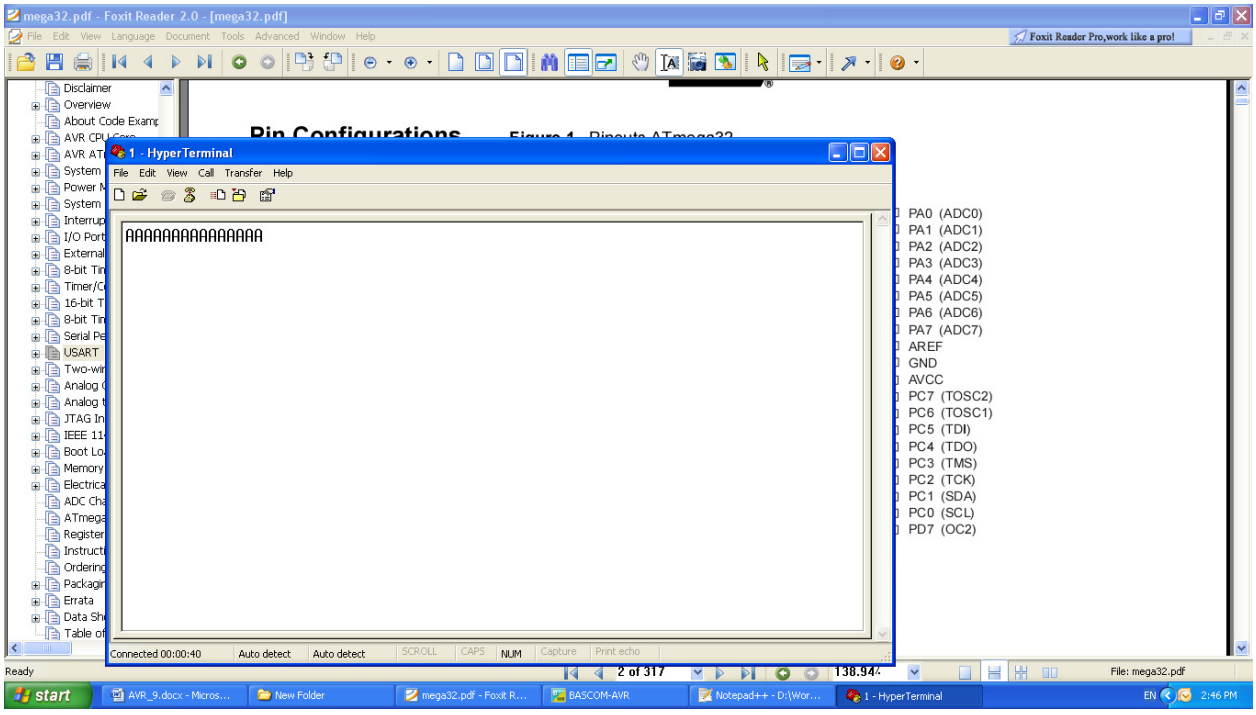
اطلاعات روی پایه شماره ۱۵ یعنی TXD ارسال می شود که بایستی با یک آی سی مبدل سطح مثل MAX232 به پورت سریال PC ارسال گردد. بخش گیرنده نیز به همین ترتیب است و در آنجا بایستی با دستور sbis UCSRA, RXC بیت شماره ۷ رجیستر UCSRA را چک کرد و در صورتی که این بیت ۱ شده بود با دستور in r16, UDR مقدار UDR را از پورت ورودی خواند.



تصویر پروژه ارتباط با پورت سریال و مبدل MAX232

برای مشاهده اطلاعات ارسال شده در محیط ویندوز XP روی Start و Run کلیک کرده و عبارت Hypertrm را تایپ و OK نمایید. نام دلخواهی به پنجره باز شده داده و پورت Com1 را انتخاب و تنظیمات را مطابق شکل زیر انجام دهید تا اطلاعات دریافتی از پورت را مشاهده کنید.





در این مقاله با تایمر و پورت سریال میکرو آشنا شدیم. جلسه بعد یعنی جلسه دهم، آخرین جلسه مبحث اسمبلی است و با آموزش راه اندازی مبدل آنالوگ به دیجیتال و نیز نحوه استفاده از متغیرهای دائمی EEPROM به پایان می رسد. جلسه یازدهم یک جلسه طولانی بوده و به CodeVision اختصاص دارد. در آنجا می بینیم که زبان C چقدر کارهار را ساده تر کرده است. ۹ جلسه بعد از آن که پایان دوره آموزش AVR می باشد به طور کامل به WinAVR می پردازد. تصمیم بر آن بود که از کامپایلر جدید AVR Studio 5 نیز استفاده کنیم ولی متاسفانه این کامپایلر وابسته به Visual Studio 2010 مایکروسافت بوده و به طور مرتب از کار می افتد و برای هر بار تعمیر آن بایستی ویندوز را عوض کرده و ویژال استودیو با حجم ۳ گیگ و سپس AVR استودیو با حجم ۵۰۰ مگ را نصب کنید که امری منطقی نیست. (البته این مشکلات در زمان نوشتن این سری مقالات وجود دارد و به دلیل نوپا بودن هر دو نرم افزار مذبور است و امید است در آینده Bug های این نرم افزارها حل شود).

با این حال WinAVR با توجه به قابل اجرا بودن بر روی ویندوز و لینوکس و استفاده از کامپایلر رایگان و خط فرمانی AVR-GCC گزینه خوبی به شمار میرود. ضمناً با توجه به محبوبیت زبان C سورس ها و کتابخانه های رایگان این کامپایلر نیز به وفور در اینترنت یافت می شود.

منابع مورد استفاده در تدوین این مقاله:

[1] دیتا شیت میکروکنترلر ATmega32

[2] شهریاری شیرزاد، مرجع علمی کاربردی میکروکنترلرهای AVR, PIC, MCS-51 انتشارات پرتونگار، ۱۳۸۵

ادامه دارد...

پایان جلسه چهارم اسمبلی (جلسه نهم آموزش AVR)

مولف: بهنام زکی زاده

www.avr64.com

۲۶ تیر ۱۳۹۰

آخرین ویرایش ۱۴ مهر ۱۳۹۲ ✓