به نام خدا

(جلسه هشتم آموزش AVR)

(Assembly 3)

مقدمه:

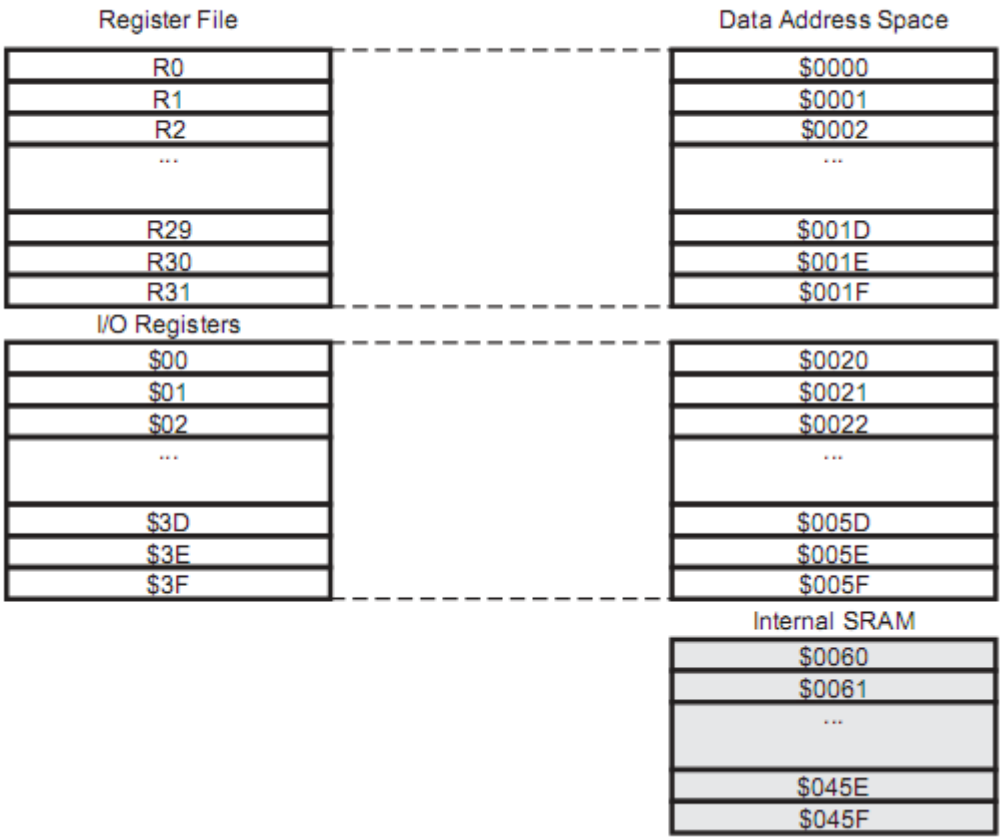
در جلسه قبل با نحوه برنامه نویسی LCD و Keypad و نیز دستیابی به متغیرها آشنا شدیم. در این جلسه در مورد بردارهای وقفه و دستورات ORG. ابتدای برنامه صحبت می کنیم. جلسه نهم و دهم را به تایمرها، پورت سریال، مبدل آنالوگ به دیجیتال، E^2PROM و سایر مباحث مهم اسمبلی که پیشنیاز C هستند اختصاص می دهیم و در جلسه یازدهم به طور مختصر با CodeVision آشنا شده و از آنجا که این برنامه یک کامپایلر تجاری می باشد از جلسه دوازدهم به بعد با کامپایلر قدرتمند WinAVR (زبان C) شروع به کار میکنیم و مباحث C و توابع مختلف را تا **جلسه بیستم** (پایان دوره رایگان آموزش AVR) پی گیری می نماییم.

قبل از نصب AVR Studio 5 بایستی یکی از کامپایلرهای Express و رایگان Microsoft مثل Visual Basic Express 2010 را دانلود و نصب نمایید تا برخی از امکانات از قبیل .NET 4 روی سیستم نصب شوند.

آدرس های حافظه:

در این مقاله از میکروکنترلر ATmega16 استفاده می کنیم و آدرس های حافظه از دیتا شیت همین میکرو برداشت شده اند و برای سایر میکرو ها متفاوت می باشد. در میکروکنترلر M16 درحالت پیشفرض آدرس شروع حافظه Flash یا همان ROM آدرس ۰ می باشد. آدرس شروع SRAM نیز 0x60 بوده و آدرس EEPROM هم ۰ است. در واقع آدرس هر سه بخش بایستی از ۰ شروع شود ولی ۹۶ آدرس ابتدای رم (از ۰ تا ۹۵) با سایر رجیستر ها و پورت ها همپوشانی داشته و ۳۲ آدرس اول برای رجیستر های همه منظوره R0 تا R31 و ۶۴ آدرس بعدی نیز برای پورت های ورودی خروجی و سایر رجیسترهای کنترلی استفاده می شوند. در تصویر زیر بخش بالای حافظه SRAM مشاهده می شود.

Figure 9. Data Memory Map

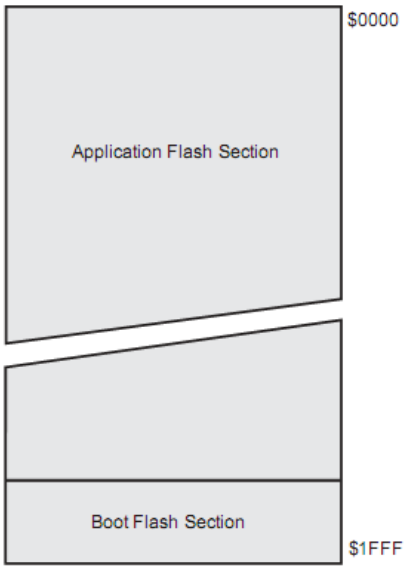


حافظه SRAM

برای تعریف متغیرها بایستی کدهای مربوطه را برای این بخش تعریف کنیم. (کدهای تعریف متغیر در این بخش نوشته نمی شوند بلکه به ازای مقادیر تعریف شده در این کدها از حافظه SRAM فضا گرفته می شود). برای این منظور از رهنمود پیش پردازنده .dseg استفاده کرده و بقیه کدها را در زیر آن می نویسیم. در حالت پیش فرض اسمبلر شروع تعریف متغیرها را از آدرس 0x60 آغاز می کند ولی با نوشتن رهنمود .org 0x65 می توانیم برای شروع رزرو متغیرها از آدرس 0x65 یا هر آدرس دیگری که بزرگتر از 0x60 و کوچکتر از پایان رم باشد استفاده نماییم.

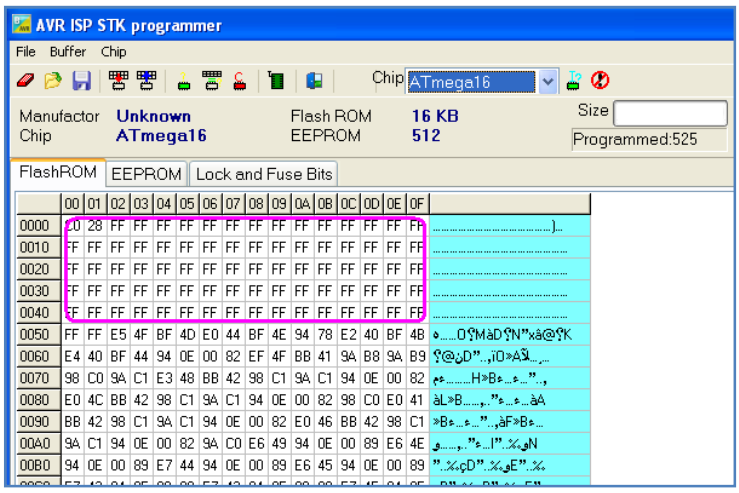
اما قسمتی که در این جلسه با آن سر و کار داریم بخش FLASH یا .cseg می باشد. این قسمت همان بخشی از حافظه غیر فرار میکرو است که کدهای هگز در آنجا قرار می گیرند. این بخش از آدرس ۰ شروع شده و در حالتی که bootloader غیر فعال باشد با روشن شدن میکرو و یا اعمال پالس Reset آدرس ۰ در شمارنده CPU قرار می گیرد. در میکروکنترلر mega16 آدرس های ۲ تا ۴۰ یعنی 0x2 تا 0x28 مربوط به بردارهای وقفه های میکرو (IRQ) می باشد.

Figure 8. Program Memory Map



حافظه Flash

به ازای هر وقفه داخلی با خارجی دو بایت فضا در نظر گرفته شده است که فقط برای نوشتن یک دستور پرش بلند مثل rjmp label مناسب است. معمولاً در آغاز کد نویسی این بخش، بلافاصله پس از دستور .cseg دستور .org 0x00 و بعد از آن rjmp main نوشته می شود. در صورتی که از هیچ وقفه ای در برنامه خود استفاده نکرده باشیم بایستی بعد از دستورات بالا .org 0x29 را نوشته و پس از آن لیبل main: را تایپ کنیم و ادامه برنامه خود را پی گیری کنیم. در این صورت تمام فضای ۴۰ بایتی ابتدای فلش نانوشته باقی می ماند و این موضوع به روشنی به هنگام پروگرم کردن میکرو دربافر پروگرمر به صورت فضای خالی ابتدای فلش مشاهده می شود.



نمایش فضای خالی بردارهای وقفه در ابتدای فلش

اما در این مقاله تصمیم داریم با وکتورهای موجود در این بخش آشنا شویم و از آنها استفاده کنیم. به همین منظور از اینتراپت خارجی شماره ۲ که روی پایه PB.2 میکروی نمونهٔ M16 قرار گرفته استفاده می کنیم و با یک سیم آنرا تحریک می نماییم. قبل از هر چیز بایستی با مراجعه به دیتا شیت میکروی مگا۱۶ از آدرس وکتور اینتراپت مربوطه اطلاع حاصل کنیم. در تصویر زیر بردارهای تمام وقفه های داخلی و خارجی این میکرو نشان داده شده اند. با بررسی این جدول مشاهده می کنیم که وکتور INT2 روی آدرس 0x24 قرار گرفته است. (با شماره وکتور ۱۶).

Table 18. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

آدرس بردارهای وقفه

پس بایستی در بخش بردارها و در آدرس 0x24 به لیبل حاوی کد مورد نظر پرش کنیم. در قطعه برنامه زیر پس از نوشتن .org 0x24 و به دنبال آن rjmp INT_2 به محض رویدادن وقفه خارجی به لیبل INT_2 پرش کرده و در آن لیبل علامت ستاره را روی LCD نشان می دهیم.

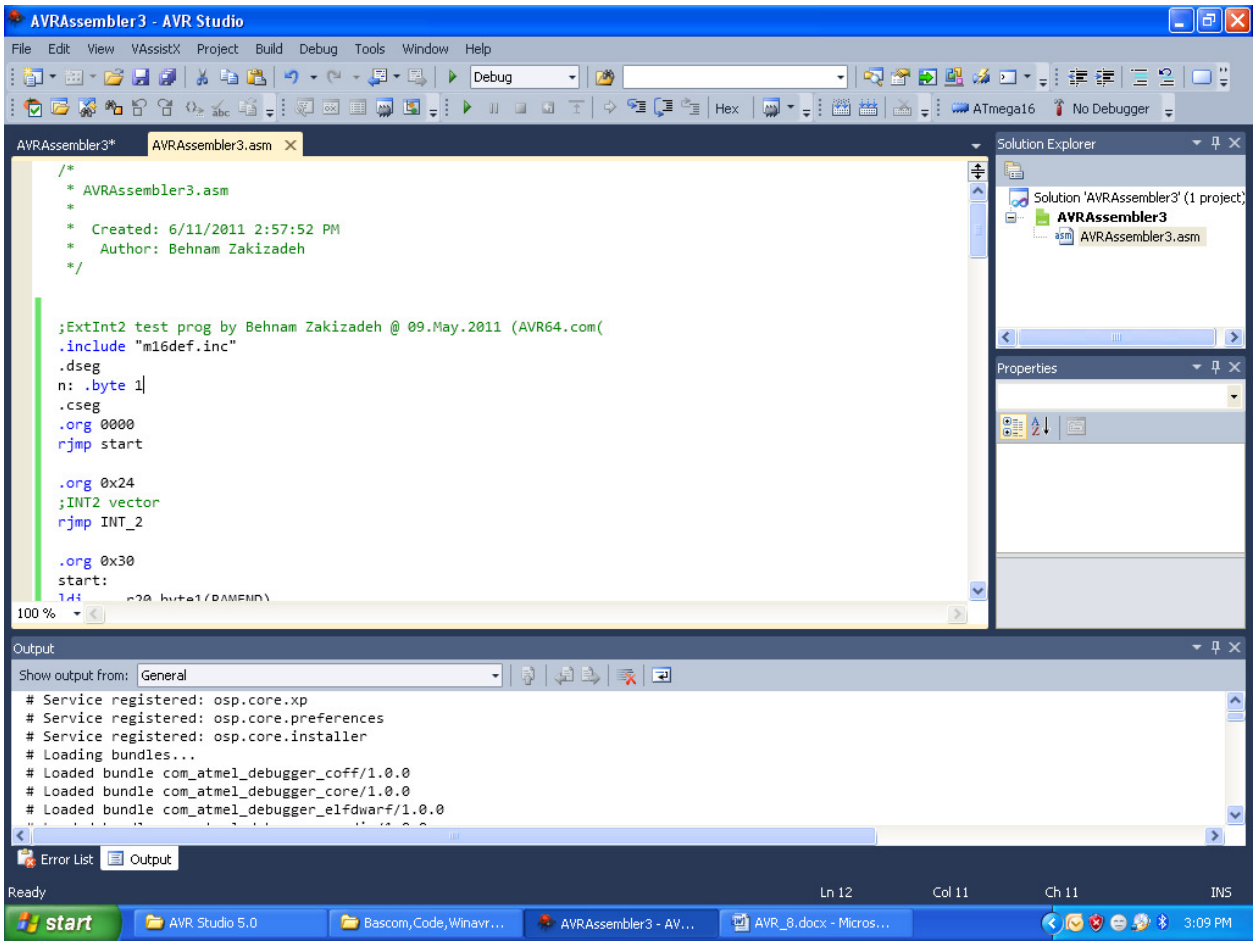
تا اینجا فقط قسمت آدرس دهی و روند انجام کارها مشخص شده است. ولی نکته اصلی پیکره بندی وقفه و فعال سازی آن است. همانطوریکه در سری مقالات گذشته در رابطه با Bascom توضیح داده شده بود برای فعال سازی وقفه ها ابتدا بایستی وقفه های سراسری را فعال کنیم و نیز وقفه مورد نظر را به طور جدا گانه توانا سازیم. برای فعال کردن وقفه های سراسری از دستور sei استفاده می شود. این دستور بیت هفتم رجیستر وضعیت میکرو (SREG) به نام i را یک می کند. دستور cli نیز وقفه های سراسری را غیر فعال می نماید. مرحله بعد فعال کردن وقفه INT2 میکرو می باشد. برای این منظور بایستی بیت شماره ۵ از رجیستر GICR به معنی جنرال اینتراپت کنترل رجیستر را ۱ نماییم. این کار را با لود عدد 0b00100000 در این رجیستر انجام می دهیم. مرحله آخر تنظیم حالت تحریک پایه این وقفه می باید. تصمیم داریم پایه مربوطه با یک لبه بالارونده (تغییر پایه از ۰ به ۱) تحریک شود. برای این منظور بایستی بیت ۶ از رجیستر MCUCSR را یک کنیم (و برای خلاف این حالت آنرا 0 نماییم). در قطعه برنامه زیر تمام این عملیات به صورت کد نوشته شده اند و پس از اسمبل کردن و پروگرم کردن در داخل میکرو و بستن شماتیک جلسه قبل در ابتدا عبارت interrupt Test: بر روی نمایشگر مشاهده می شود. (یک قطعه سیم از پایه PB.2 میکرو به زمین وصل شده تا در حال شروع، وقفه اتفاق نیفتد). در صورتی که سیم متصل به پایه INT2 را از زمین جدا کرده و به VCC متصل کنیم برنامه از حلقه end خارج شده و به بردار وقفه اشاره می کند که در آنجا نیز با یک دستور پرش به لیبل INT_2 رسیده و علامت ستاره * در جلوی عبارت بالا در LCD نشان داده می شود و بلافاصله با دستور cli وقفه های سراسری غیر فعال می شوند تا از تکرار تایپ این علامت درنمایشگر جلوگیری شود.

```
;ExtInt2 test prog by Behnam Zakizadeh @ 09.May.2011 (AVR64.com)
.include “m16def.inc"
.dseg
n: .byte 1
.cseg
.org 0000
rjmp start

.org 0x24
;INT2 vector
rjmp INT_2
```

```
.org 0x30
start:
ldi   r20,byte1(RAMEND(
out   SPL,r20
ldi   r20,byte2(RAMEND(
out   SPH,r20

sei ;enable interrupts

ldi r20, 0b00100000
out gicr, r20 ;enable INT2

ldi r20, 0b01000000
out mcucsr, r20 ;INT2 rising

call delay ;wait for lcd init

ldi r20,0xff
out ddrd, r20
sbi ddrb,0
sbi ddrb,1
cbi portb,0
sbi portb,1
ldi r20, 0x38
out portd, r20
cbi portb,1
sbi portb,1
call delay

ldi r20, 0x0c
out portd, r20
cbi portb,1
sbi portb,1
call delay

cbi portb,0

ldi r20, 0x01
out portd, r20
cbi portb,1
sbi portb,1
call delay

ldi r20, 0x06
out portd, r20
cbi portb,1
sbi portb,1
call delay

sbi portb,0

ldi r20,'i'
call show
ldi r20,'n'
call show
ldi r20,'t'
```

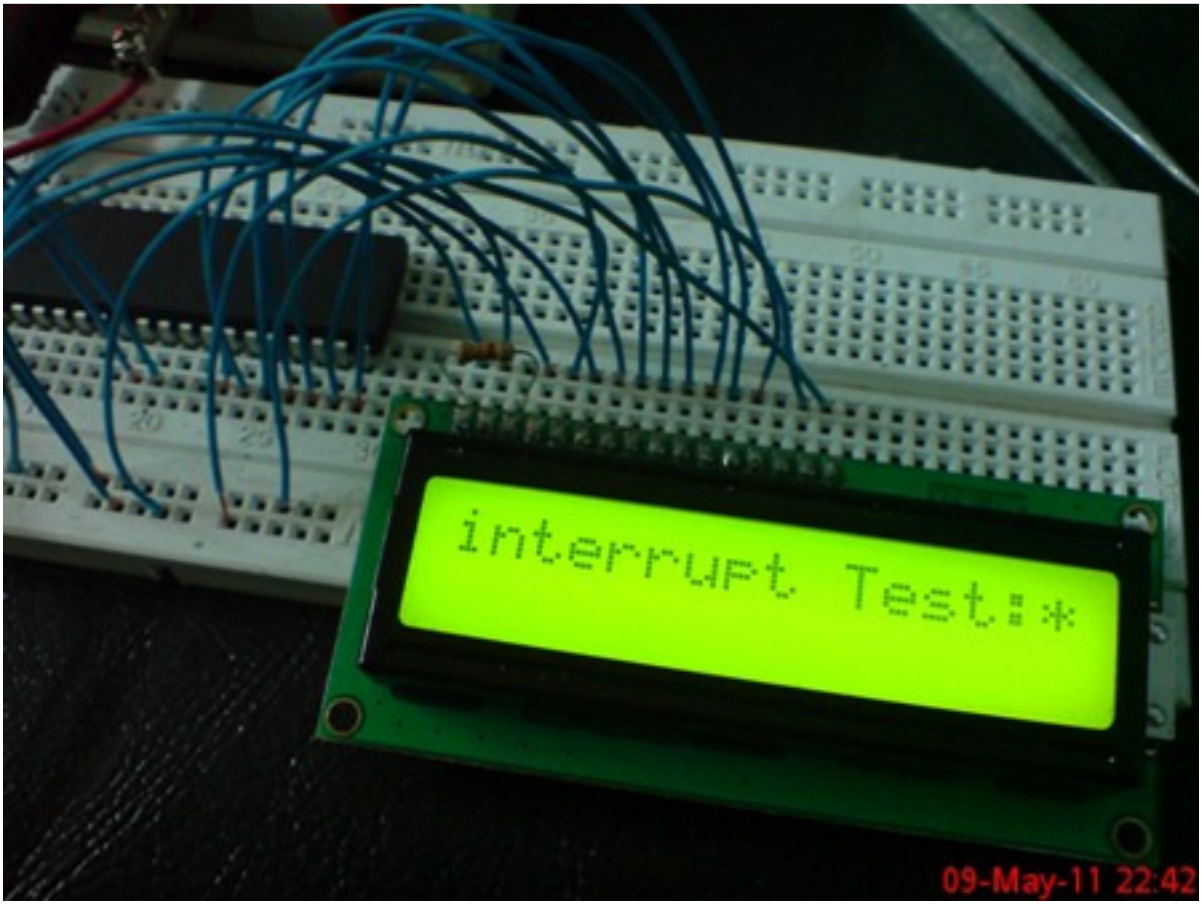
```
call show
ldi r20,’e'
call show
ldi r20,’r'
call show
ldi r20,’r'
call show
ldi r20,’u'
call show
ldi r20,’p'
call show
ldi r20,’t'
call show
ldi r20' ',
call show
ldi r20,’T'
call show
ldi r20,’e'
call show
ldi r20,’s'
call show
ldi r20,’t'
call show
ldi r20':',
call show



end:
rjmp end

delay:
ldi r20,0x40
loop2: ldi r21,0xff
loop3: dec r21
     brne loop3
     dec r20
     brne loop2
ret

show:
out portd, r20
cbi portb,1
sbi portb,1
call delay
ret

INT_2:
ldi r20'*',
call show
cli
reti
```

همانطوریکه می دانید در اثنای نوشتن این سری مقالات، ATMEL کامپایلر و اسمبلر قدرتمند AVR Studio 5 را معرفی کرد. این نرم افزار با استفاده از تکنولوژی ویژال استودیو ۲۰۱۰ نوشته شده و کار کردن با آن برای دوستانی که با محیط ویژوال استدیو (برنامه نویسی تحت ویندوز) آشنایی داشته باشند واقعاً لذت بخش می باشد. البته بخش اسمبلر این محیط قدرت نرم افزار را نشان نمی دهد و در مباحث بعدی که به آموزش C خواهیم پرداخت سایر ویژگیهای این محیط جدید بهتر نمایان می گردند. به عنوان مثال یکی از ویژگیهای جالب این محیط که دقیقاً مانند ویژوال استودیو 2010 می باشد نمایش سایر زیر مجموعه ها و توابع یک فایل و حدس زدن خودکار بقیه کد می باشد و نیز بخش Solution آن که دقیقاً محیط را مانند ویژوال استدیو کرده است. در ضمن این محیط از تمام میکروهای ۸ و ۳۲ بیتی و Xmega و... پشتیبانی می نماید و با ۴۰۰ مثال و پروژه کاربردی و نیز هدرفایل های فراوان با توجه به رایگان بودن آن به سوژه خبری خوبی برای ATMEL و سایر سایت های وابسته در سال ۲۰۱۱ تبدیل شده است.



تصویر کامپایلر و اسمبلر AVR Studio 5 محصول سال ۲۰۱۱

تصویر پروژه: (با اتصال پایه وقفه شماره ۲ به VCC علامت * در جلوی عبارت خط بالا ظاهر می شود).



تصویر تست برنامه وقفه (اجرا شده بر روی سخت افزار جلسه قبل)

نکته ای که در رابطه با وقفه های میکروکنترلرهای خانواده Mega AVR شایان ذکر است عدم ساپورت وقفه های تو در تو می باشد. در معماری این میکروکنترلرها اینطور که گفته می شود به محض وقوع یک وقفه بیت i به صورت خودکار توسط میکرو ۰ می شود تا وقفه های سراسری غیر فعال شوند و پس از عبور از دستور reti که در پایان روتین وقفه نوشته می شود این بیت به صورت خودکار ۱ می شود. در این شرایط در صورتی که در حال اجرای عمل داخل روتین وقفه باشیم وقفه ثانوی با شکست مواجه می شود. این قضیه وقفه های تو در تو در سیستم عامل با سمافورها و راه های پیچیده دیگری به صورت نرم افزاری حل می شود (در سیستم عامل Task ها به این صورت کار میکنند که می توان آنها را نوعی وقفه داخلی به حساب آورد) ولی در AVR این مساله نادیده گرفته شده است. البته با استفاده از راه حل های نرم افزاری میتوان با خواندن مقدار یک پایه خاص میکرو و یا شبکه کردن دو میکرو با یکدیگر در شرایط بحرانی این نقیصه را جبران نمود و محیطی با ویژگیهای Real time ایجاد کرد. نکته دوم اولویت وقفه هاست که به ترتیب شماره می باشد. یعنی وقفه شماره ۰ اولویت بیشتری نسبت به شماره ۱ داشته و در صورتی که هر دو وقفه با هم اتفاق بیفتند وقفه شماره ۰ اجرا می شود. البته این مسائل بایستی قبل از طراحی سیستم به صورت عملی تست شوند چرا که توسعه دهندگان تکنولوژی های سخت افزاری معمولاً با سرعت بالایی ویژگیهای جدیدی را به محصولات خود اضافه می کنند.

در این جلسه با اینتراپت ها آشنا شدیم. در جلسه بعدی در مورد تایمرها و پورت سریال بحث می کنیم و جلسه پس از آن نیز با مباحثی در خصوص A2D و EEPROM به پایان می رسد. دو جلسه بعد یعنی جلسات نهم و دهم پایان آموزش اسمبلی می باشند. ده جلسه بعد از آن به طور کامل به زبان C می پردازیم و برنامه نویسی حرفه ای با زبان استاندارد و قابل توسعه C را بررسی می کنیم تا با ویژگیهای این زبان قدرتمند و رایگان آشنا شویم. مباحث آموزش AVR به طور کامل در ۲۰ جلسه تکمیل می شوند ولی پس آن نیز با نوشتن مقالاتی پراکنده در خصوص برخی از دستورات و توابع پرکاربرد کامپایلر محبوب BASCOM و سایر کامپایلرها این جریان را به طور مستمر ادامه خواهیم داد. البته این سری مقالات در بخش مقالات عمومی وبسایت قرار خواهند گرفت و قسمت مقالات دنباله دار آموزش AVR پس از ۲۰ جلسه واقعاً به اوج خود رسیده و پایان می یابد و به احتمال زیاد به صورت یک کتاب کامل نیز منتشر شود. (البته به صورت PDF و رایگان و فقط به منظور دانلود راحت تر و قابلیت حمل بهتر).

منابع مورد استفاده در تدوین این مقاله:

[1] دیتا شیت میکروکنترلر ATmega16

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

این مقاله با نسخه قانونی ویندوزXP OEM و Word 2007 تولید شده و برای تبدیل آن به فرمت PDF از نسخه رایگان Cute PDF بهره گرفته شده است. برای تولید و ویرایش کدها از نسخه رایگان AVR Studio 5 استفاده شده و برای پروگرم کردن میکرو نیز از نسخه خریداری شده BASCOM 2.0.5.0 و سخت افزار پروگرمر STK200/300 که اجازه ساخت آن از Kanda.com گرفته شده بهره گرفته شده است. AVR64 به شدت تابع قوانین کپی رایت بین المللی بوده و همگان را به تبعیت از این قانون دعوت می نماید. انتشار این مقاله آزاد می باشد.

ادامه دارد...

((ای وی آر استدیو ۲۰۱۱ از سایت ATMEL قایل دانلود بوده و حجم آن در حدود ۵۰۰ مگابایت می باشد. به دلیل محدودیت های خاص برای دانلود این نرم افزار بایستی پرسشنامه ای را پر کنید تا لینک دانلود را دریافت نمایید))

پایان جلسه سوم اسمبلی (جلسه هشتم آموزش AVR)

مولف: بهنام زکی زاده

www.avr64.com

۱۹ اردیبهشت ۱۳۹۰

آخرین ویرایش: ۱۴ مهر ۱۳۹۲✓