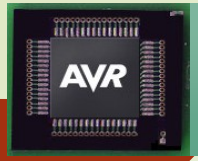
فصل چهارم

وقفه ها

(interrupts)

اهداف

1. آشنایی با روش های polling و interrupt
2. معرفی منابع وقفه و آشنایی با مفهوم بردار وقفه
3. آشنایی با نوشتن برنامه به روش وقفه در زبان C



وقفه ها (interrupts)

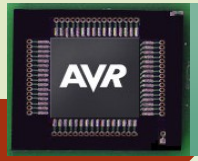
وقفه سخت افزاری ازمیکروکنترلر است که که میکروکنترلر را برای پاسخ گویی به رویدادهای لحظه ای مجهز می کند.

تعریف وقفه : به معنی تاخیر زمانی نیست ،بلکه به معنی قطع موقت برنامه جاری و سرویس دادن به زیر روال وقفه است.

روش های بررسی یک رویداد از جانب CPU

به طور کلی CPU برای تشخیص رویداد های داخلی (مانند تست پرچم سرریز تایمر یا کانتر)و رویداد های خارجی (مانند تشخیص لبه پایین رونده یک پالس و یا وصل شدن یک کلید) می تواند از دو روش زیر استفاده کند:

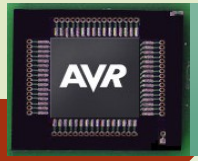
۱.روش سرکشی (polling) : در این روش کاربر توسط برنامه نویسی با فواصل زمانی مشخص و دائما رویداد مورد نظر را بررسی می کند تا به آن پاسخ دهد. به طور مثال شما در حال درس خواندن و منتظر یک شخص هستید اگر شما هر یک دقیقه یکبار بروید و از پنجره بیرون را نگاه کنید که ببینید آن شخص آمده یا نه؟به این روش سر کشی می گوییم. به طور نمونه اگر در برنامه نویسی ، ما برای بررسی فشردن یک کلید و یا فعال شدن یک پرچم خاص چند لحظه یک بار بخواهیم بررسی را انجام بدهیم ،در واقع بع روش polling عمل کرده ایم.
عیب این روش تلف کردن وقت cpuاست.



وقفه ها (interrupts)

۲.روش وقفه : در مثالی که بیان کردیم یک راه ساده تر آن است که شما به درس خواندن ادامه بدهید و منتظر زنگ شخص مورد نظر بمانید و هر موقع که زنگ منزل را زدند بروید و به آن پاسخ دهید و بعد از اتمام کار برگردید و به درس خواندن خود ادامه دهید به این روش وقفه می گویند. می بینیم که وقت ما بدون تلف شدن تا آمدن آن شخص صرف کار مفید دیگری شده است.

CPU نیز بدون در نظر گرفتن رویداد به انجام سایر اعمال مشغول می شود و با وقوع اتفاق مورد نظر، CPU انجام خط جاری برنامه را متوقف کرده و به بردار وقفه مربوطه پرش می کندو زیر روال سرویس وقفه ISR (Intrrupt Service Routin) را اجرا می کند و پس از اجرای این زیر روال به خطی از برنامه که آن را قطع کرده بود بر می گردد و ادامه برنامه را اجرا می کند.



وقفه ها (interrupts)

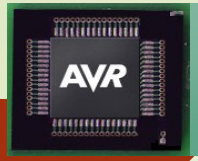
بردارهای وقفه و Reset در ATmega16

ابتدا بهتر است منظور از بردار را مشخص کنیم . شما موقعیکه برنامه را می نویسید برنامه تبدیل به یک سری کد ماشین (HEX) می شود که مکان صفر حافظه Flash توسط پروگرامر نوشته می شود.
حال اگر بیایم یک مکان مشخص از حافظه ثابت را به وقفه خاصی اختصاص دهیم و سایر قسمت های برنامه را در مکان دیگری ذخیره نمایم به طوری که با تحریک آن وقفه برنامه به آن مکانی که به طور ثابت و مشخص به وقفه اختصاص داده بودیم پرش کند که به آن مکان بردار وقفه گفته می شود.

جدول ۵-۱ آدرس و شماره بردار تمام منابع وقفه در میکروکنترلر ATmega16آورده شده است.

نحوه ی تعریف تابع وقفه در نر م افزار CodeVisionAVR

```
Interrupt [شماره بردار یا نام معادل] void  نام تابع( void) {
برنامه یا زیر روال سرویس وقفه;
}
```



وقفه ها (interrupts)

از هروقفه ای که بخواهیم استفاده کنیم باید ابتدا رجیستر های مربوز به آن وقفه را تنظیم کنیم و در برنامه ،تابع وقفه را به فرم فوق بنویسیم . کلمه کلیدی interrupt تعیین کننده تابع از نوع وقفه می باشد و شماره برداار وقفه نیز طبق جدول ۵-۱ تعیین می گردد در نرم افزار code vision VR برای این شماره ها نام معادلی تعریف شده است که می توان از آنها استفاده کرد.

مثال :

```
Interrupt  [2]void  ext_into_isr (void) { دستورالعملها}
```

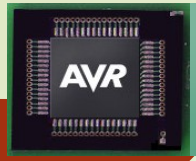
و یا می توان از نام معادل برای وقفه خارجی صفر استفاده کرد.

```
Interrupt  [EXT_INTO] void ext_into_isr(void) {دستورالعملها}
```

وقفه ها (interrupts)

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

جدول ۵-۱ بردارهای وقفه و Reset



وقفه ها (interrupts)

مراحل اجرای یک وقفه

میکروکنترلر زمانی که وقفه داخلی یا خارجی را دریافت می کند مراحل زیر را انجام می دهد:

۱.دستوری که در حال اجرای آن می باشد را به پایان رسانده و آدرس دستور العمل بعدی را در حافظه پشته ذخیره می کند.

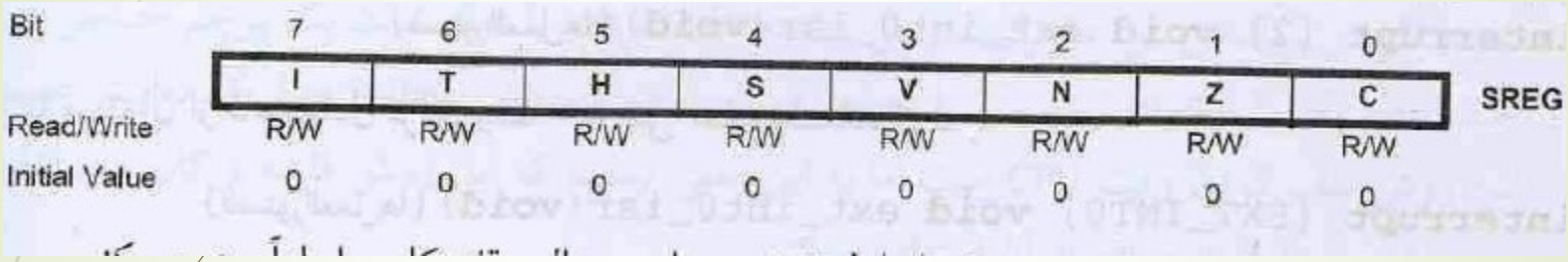
۲.به بردار وقفه مربوطه پرش می کند و برنامه یا زیر روال سرویس وقفه (ISR) را انجام می دهد.

۳.پس از اتمام زیر روال وقفه ،توسط دستور اسمبلی RETI از وقفه بر می گردد. البته در زبان C ما دستور RETI را بکار نمی گیریم اما کامپایلر بعد از تفسیر برنامه از آن استفاده می کند.

۴.آدرس دستورالعملی که در مرحله یک در حافظه پشته ذخیره کرده بود را بر می دارد و با قرار دادن آن آدرس ،در شمارنده برنامه (PC)به ادامه برنامه بر میگردد.

وقفه ها (interrupts)

رجیستر وضعیت **SREG** (State Register)



Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیت 7 (global interrupt enable)

اگر این بیت را یک کنیم وقفه کلی (همگانی یا سراسری) فعال می گردد و در این حالت هست که دیگر وقفه ها در صورت فعال بودن و تحریک شدن می توانند رخ دهند. در صورتی که این بیت صفر باشد و دیگر وقفه ها فعال باشند و حتی تحریک هم شوند قابل اجرا توسط CPU نخواهند بود پس این بیت ،کنترلی برای رخ دادن دیگر وقفه ها ست. در زبان برنامه نویسی C این بیت صورت زیر فعال و غیر فعال می شود.

```
#asm (“sei”)        // فعال کردن وقفه کلی یا همگانی
#asm (“cli”)        // غیر فعال کردن وقفه کلی یا همگانی
```

وقفه ها های خارجی (External Interrupts)

وقفه های خارجی

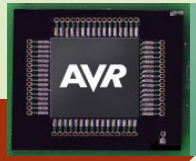
از وقفه های خارجی معمولا برای تشخیص پالساستفاده می شود حال این پالس می تواند حاوی اطلاعات گوناگونی باشد به طور مثال پالس هایی که از طرف یک سنسور خاص مانند SMT160 یا سنسور اپتوکانتر ارسال می گردد یا پالسی که در اثر وصل شدن یک میکروسوئیچ ایجاد می گردد یا پالسی که از طرف یک مبدل آنالوگ به دیجیتال بیرونی ارسال می شود و یا پالسی که از هر تراشه و رویداد دیگری ارسال می گردد که ما بخواهیم توسط میکروکنترلر خیلی سریع به آن پاسخ دهیم از وقفه خارجی بهر ه می گیریم.

رجیستر **GICR** (GENERAL INTERRUPT CONTROL REGISTER)

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **بیت INT2-5** (external interrupt request 2 enable)

با یک کردن این بیت وقفه خارجی دو فعال می گردد و اگر وقفه کلی نیز فعال باشد می تواند وقفه خارجی دو در صورت رخ دادن اجرا می گردد. وقتی که این بیت را یک کنیم عملکرد عادی PB2 به عنوان I/O قطع شده و به عنوان INT2 عمل می کند.



وقفه ها های خارجی (External Interrupts)

- بیت **INT0-6** (external interrupt request 0 enable)

با یک کردن این بیت وقفه خارجی صفر فعال می گردد و اگر وقفه کلی نیز فعال باشد می تواند وقفه خارجی صفر در صورت رخ دادن اجرا می گردد. وقتی که این بیت را یک کنیم عملکرد عادی PD2 به عنوان I/O قطع شده و به عنوان INT0 عمل می کند.

- بیت **INT1-7** (external interrupt request 1 enable)

با یک کردن این بیت وقفه خارجی یک فعال می گردد و اگر وقفه کلی نیز فعال باشد می تواند وقفه خارجی یک در صورت رخ دادن اجرا می گردد. وقتی که این بیت را یک کنیم عملکرد عادی PD3 به عنوان I/O قطع شده و به عنوان INT1 عمل می کند.

وقفه ها های خارجی (External Interrupts)

رجیستر **GIFR** (GENERAL INTERRUPT FLAG REGISTER)

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	INTF2	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- بیت **INTF2-5** (external interrupt flag 2)

اگر وقفه خارجی ۲ و وقفه کلی فعال باشند، در صورت تحریک شدن وقفه خارجی دو از طریق پایه بیرونی INT2، این پرچم فعال می گردد و درخواست اجرای زیر روال وقفه را می دهد و برنامه به تابع وقفه دو پرش خواهد کرد و بعد از اجرای آن ، به طور اتوماتیک این پرچم پاک می شود.

- بیت **INTF0-6** (external interrupt flag 2)

اگر وقفه خارجی صفر و وقفه کلی فعال باشند، در صورت تحریک شدن وقفه خارجی صفر از طریق پایه بیرونی INT0، این پرچم فعال می گردد و درخواست اجرای زیر روال وقفه را می دهد و برنامه به تابع وقفه صفر پرش خواهد کرد و بعد از اجرای آن ، به طور اتوماتیک این پرچم پاک می شود.

وقفه ها های خارجی (External Interrupts)

- بیت **INTF1-7** (external interrupt flag 1)

اگر وقفه خارجی ِیک و وقفه کلی فعال باشند، در صورت تحریک شدن وقفه خارجی یک از طریق پایه بیرونی INT1، این پرچم فعال می گردد و درخواست اجرای زیر روال وقفه را می دهد و برنامه به تابع وقفه یک پرش خواهد کرد و بعد از اجرای آن ، به طور اتوماتیک این پرچم پاک می شود.

رجیستر MCUCR (MCU Control Register)

Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

وقفه ها های خارجی (External Interrupts)

- بیت **ISC11,ISC10-3,2** (Iterrupt Sense Control 1 Bit 1 and Bit 0)

توسط این دو بیت می توان نحوه ی تحریک شدن وقفه خارجی یک را تعیین نمود . تحریک می تواند در سطح صفر یا هر تغییر منطقی یا لبه پایین رونده و یا بالا رونده صورت گیرد.

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

جدول ۵-۲ تعیین نحوهٔ تحریک شدن وقفه خارجی یک

وقفه ها های خارجی (External Interrupts)

- بیت **ISC01,ISC00-1,0** (Iterrupt Sense Control 1 Bit 1 and Bit 0)

توسط این دو بیت می توان نحوه ی تحریک شدن وقفه خارجی صفر را تعیین نمود . تحریک می تواند در سطح صفر یا هر تغییر منطقی یا لبه پایین رونده و یا بالا رونده صورت گیرد.

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

جدول ۵-۳ تعیین نحوهٔ تحریک شدن وقفه خارجی صفر

وقفه ها های خارجی (External Interrupts)

رجیستر MCUCSR (MCU Control and Status Register)

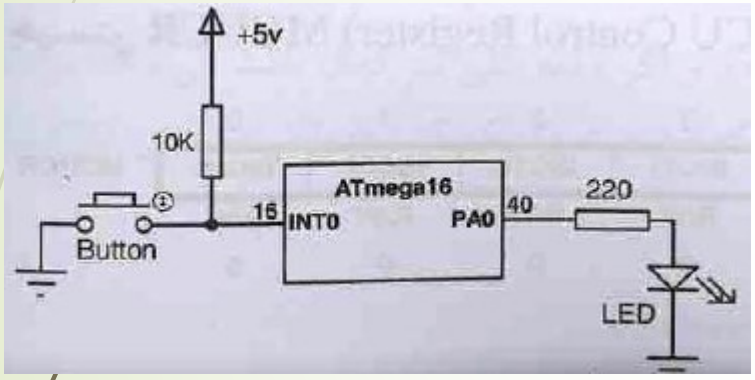
Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

- بیت **ISC2-6** (Interrupt Sense Control 2)

اگر این بست صفر باشد وقفه خارجی دو در لبه پایین رونده و اگر یک باشد در لبه بالا رونده پالس تحریک کننده اجرا خواهد شد.

وقفه ها های خارجی (External Interrupts)

مثال ۵-۱: برنامه ای بنویسید که توسط وقفه خارجی صفر ،مطابق شکل داده شده فشردن یک کلید فشاری را تشخیص دهد و یک LED را که به PA0 وصل شده است را معکوس گرداند.



شکل ۵-۱ استفاده از وقفه خارجی صفر

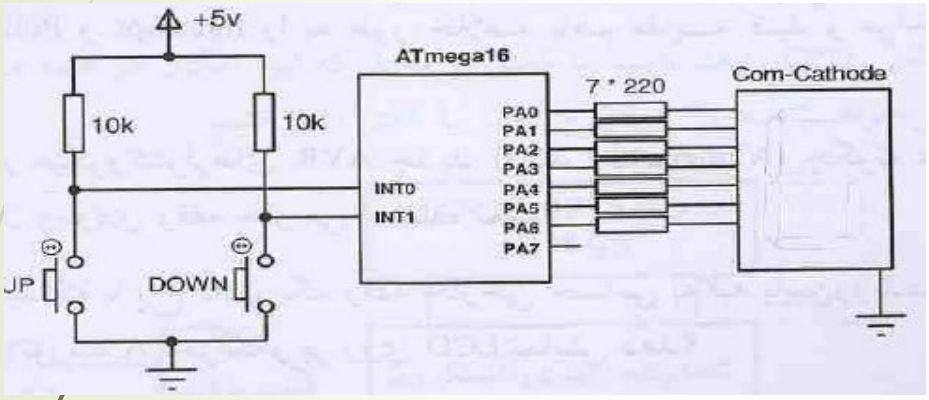
در مثال ۵-۱ پایه وقفه خارجی صفر را توسط یک مقاومت pull-up به 5v+ وصل کرد ه ایم . این بدان معنی است که در حالت عادی پایه INT0 در وضعیت یک منطقی می باشد و به محض فشردن کلید یک لبه پایین رونده بوجود می آید و موجب وقفه خوردن میکرو شده و برنامه در این لحظه به تابع وقفه خارجی صفر پرش می کند و LED را معکوس می کند و بر می گردد.

وقفه ها های خارجی (External Interrupts)

```
#include  <mega16.h>
Interrupt  [EXT_INT0]  void ext_int0_isr (void) {      //تابع وقفه خارجی صفر
While (PIND.2==0) ;                                    //تست برای رها شدن کلید
 PORTA.0=!                                             // معکوس کردن LED متصل به پایه PA0
                                                                          (PORTA.0);
  void main () {
PORTA.0=0;                                             // وضعیت اولیه ال ای دی خاموش است
DDRA.0=1   //پایه  PA0  به  عنوان  خروجی  تعیین  می  شود.
                                                                          ;
GICR=0x40 ;                                            //فعال کردن وقفع خارجی صفر.
MCUCR=0x02;                                            // تنظیم حساسیت وقفه خارجی صفر به لبه پایین رونده
GIFR=0x40;                                             //مقدار دهی اولیه پرچم خارجی صفر
#asm ("sei")                                           // فعال کردن وقفه کلی یعنی کردن بیت I در رجیستر وضعیت
While (1) ;                                            // حلقه ی بی نهایت و بیکار
}
```

وقفه ها های خارجی (External Interrupts)

مثال ۵-۲: مطابق شکل ۵-۲ ، دو کلید فشاری به وقفه های خارجی صفر و یک متصل نمو ده ایم.
می خواهیم برنامه ای بنویسیم که با زدن کلید UP یک واحد به نمایشگر تک رقمی کاتد مشترک اضافه گردد و با زدن کلید Down یک واحد کاهش یابد.



شکل ۵-۲ استفاده از وقفه های خارجی صفر و یک

در شکل ۵-۲ پایه PA7 را به این دلیل آزاد گذاشتیم که سگمنت تک رقمی ما فاقد نقطه یا Dot می باشد . هر بار کلید UP زده می شود تابع وقفه خارجی صفر اجرا شده و یک واحد به شمارشگر اضافه می شود و حداکثر عدد ممکن ۹ خواهد بود و با زدن کلید DOWN ،تابع وقفه خارجی یک اجرا شده و یک واحد از شمارشگر کم می شود که حداقل برابر صفر می باشد.

وقفه ها های خارجی (External Interrupts)

```
#include <mega16.h>
Flash unsigned char number []={ کد های سون سگمنت کاتد مشترک در حافظه ثابت
      //
0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f
Unsigned  char i=0; معرفی یک متغیر برای اندیس آرایه حاوی کد های سون سگمنت
          //
Interrupt  [EXT_INT0] void ext_int0_isr (void) { تابع وقفه خارجی
           //صفر
If (i<9)  PORTA=number [++i]; اگر عدد کوچکتر از ۹ بود یک واحد شمارشگر افزایش
      //یابد
While          //UP کلید شدن رها برای تست
                  (PIND.2==0);
}
Interrupt [EXT_INT1] void ext_int1_isr (void) {          //تابع وقفه خارجی یک
If (i > 0) PORTA=number[--1]; اگر عدد یزرگتر از صفر بود یک واحد شمارشگر کاهش یابد
      //
While          //DOWN کلید شدن رها برای تست
```

وقفه ها های خارجی (External Interrupts)

```
Void main () {
PORTA=number [0] ;                    //نمایش اولیه را عدد صفر قرار می دهیم
//          خروجی          عنوان          به          A          پورت          تعریف
                                                    DDRA=0xff;

GICR=0xC0 ;                          // فعال کردن وقفه ها خارجی صفر و یک
MCUCR=0x0 A;                         // تنظیم حساسیت وقفه خارجی صفر و یک  به لبه پایین رونده
GIFR=0xC0;                           //مقدار دهی اولیه پرچم های خارجی صفر و یک
#asm (“sei”)                         // فعال کردن وقفه کلی
While (1) ;                          // حلقه ی بی نهایت و بیکار
}
```