به نام خدا

آزمایشگاه ریزپردازنده دانشکده مهندسی کامپیوتر

آزمایش شماره ده

آشنایی با اینتراپتهای خارجی و کار با آنها

در این آزمایش می‌خواهیم با مفهوم وقفه (interrupt) آشنا شویم و اینتراپتهای خارجی را به کار گرفته و آزمایشی با آن انجام دهیم. در ابتدا تفاوت بین وقفه و بررسی مداوم (polling) را بررسی کرده و سپس وقفه‌های مختلف AVR را توضیح دهیم.

وقفه‌ها در مقابل بررسی مداوم (polling)

یک میکروکنترلر می‌تواند به وسایل مختلفی سرویس بدهد. برای این منظور دو روش وجود دارد: یکی وقفه و دیگری روش بررسی مداوم.

در روش وقفه هر موقع که یک وسیله به سرویس میکروکنترلر نیاز داشت، با ارسال یک سیگنال وقفه، میکروکنترلر را خبر می‌کند. هنگام دریافت یک سیگنال وقفه، میکروکنترلر کارهای جاری خود را متوقف کرده و به وسیله مذکور سرویس لازم را ارائه می‌نماید. برنامه‌ای که با آمدن وقفه اجرا می‌شود، روال سرویس وقفه (ISR) یا سرویس دهنده وقفه (Interrupt Service Routine) نامیده می‌شود.

در روش بررسی مداوم یا polling، میکروکنترلر بطور پیوسته وضعیت یک وسیله مشخص را بررسی می‌کند، وقتی شرایط مورد نظر فراهم شد، سرویس لازم را انجام می‌دهد. پس از آن به بررسی وضعیت وسیله بعدی پرداخته، تا با همه وسایل سرویس دهد. اگر چه روش بررسی مداوم می‌تواند وضعیت چندین وسیله را بررسی کرده و بر اساس شرایط خاص هر کدام، به هر یک از آنها سرویس مورد نظر را ارائه دهد، ولی این روش از میکروکنترلر استفاده بهینه نمی‌کند.

مزیت وقفه این است که میکروکنترلر می‌تواند به تعداد زیادی وسیله (البته نه بصورت همزمان) سرویس دهد و هر وسیله به توجه به اولویتی که برای آن در نظر گرفته شده، وقت میکروکنترلر را به خود اختصاص می‌دهد. در روش بررسی مداوم امکان اختصاص اولویت وجود ندارد، زیرا همه وسایل به روش تخصیص دوره‌ای یا round-robin بررسی می‌شوند. از آن مهمتر در روش وقفه، میکروکنترلر می‌تواند

تقاضای سرویس یک وسیله را نادیده (mask) بگیرد. این موضوع در روش بررسی مداوم امکان‌پذیر نیست. در نتیجه وقفه‌ها برای اجتناب از کاهش کارایی میکروکنترلر، مورد استفاده قرار می‌گیرند.

روال سرویس وقفه (Interrupt Service Routine)

برای هر وقفه باید یک روال سرویس وقفه (ISR) یا سرویس دهنده وجود داشته باشد. وقتی یک وقفه صدا زده می‌شود، میکروکنترلر روال سرویس وقفه را اجرا می‌کند. بطور کلی، در اغلب میکروکنترلرها برای هر وقفه یک خانه ثابت در حافظه وجود دارد، که آدرس ISR آن وقفه را نگهداری می‌کند. گروهی از خانه‌های حافظه که برای نگهداری آدرس‌های ISR استفاده می‌شود، جدول بردار وقفه نامیده می‌شود.

وقفه	خانه حافظه
ریست (Reset)	0x0000
تقاضای وقفه خارجی صفر	0x0002
تقاضای وقفه خارجی یک	0x0004
تقاضای وقفه خارجی دو	0x0006
تطبیق مقایسه تایمر/کانتر دو	0x0008
سرریز تایمر/کانتر دو	0x000A
ضبط کردن تایمر/کانتر یک	0x000C
تطبیق مقایسگر A تایمر/کانتر یک	0x000E
تطبیق مقایسگر B تایمر/کانتر یک	0x0010
...	...

مراحل اجرای یک وقفه

هنگام فعال شدن یک وقفه، میکروکنترلر مراحل زیر را انجام می‌دهد.

۱- دستور در حال اجرا را تمام کرده و آدرس دستور بعدی (شمارنده برنامه) را بر روی پشته یا استک ذخیره می‌کند.

۲- به خانه ثابتی از حافظه که جدول بردار وقفه نامیده می‌شود، پرش می‌کند. جدول بردار وقفه، میکروکنترلر را به آدرس روال سرویس وقفه (ISR) هدایت می‌کند.

۳- میکروکنترلر شروع به اجرای روال سرویس وقفه کرده تا به دستور آخر روال، یعنی RETI (RETurn from Interrupt) برسد.

۴- پس از اجرای دستور RETI، میکروکنترلر به محلی که وقفه رخ داده، بر می‌گردد. ابتدا آدرس شمارنده برنامه (PC) را، با برداشتن دو بایت از بالای پشته و قرار دادن آن در PC، بدست می‌آورد. سپس شروع به اجرای دستورات از آن آدرس می‌کند.

با دقت در مرحله ۴ و در نقش کلید پشته متوجه می‌شوید که چرا باید در زیرروال وقفه نسبت به تغییر محتوای پشته دقت کنیم. در ISR همانند هر زیرروال CALL دیگری، باید تعداد push (قرار دادن بر روی پشته) و pop (برداشن از روی پشته) با هم برابر باشد.

منابع وقفه در AVR

منابع بسیاری، بسته به این که چه وسیله جانبی با تراشه در حال کار کردن است، برای وقفه‌ها در AVR وجود دارد. در ادامه برخی از منابع پر استفاده وقفه‌ها در AVR معرفی شده‌اند:

۱- حداقل دو وقفه برای هر تایمر کنار گذاشته شده‌اس، یکی برای سرریز و دیگری برای تطبیق مقایسه

۲- سه وقفه برای سخت‌افزارهای خارجی وجود دارد. پایه‌های PD2 (PORTD.2)، PD3 (PORTD.3) و PB2 (PORTB.2) به ترتیب به وقفه‌های سخت‌افزاری خارجی INT0، INT1 و INT2 اختصاص یافته‌اند.

۳- ارتباط سریال توسط USART سه وقفه دارد، یکی برای دریافت و دو تا برای ارسال.

۴- وقفه‌های مربوط به SPI

۵- وقفه مربوط به مبدل آنالوگ به دیجیتال یا ADC

۶- وقفه مربوط به EEPROM.

AVR وقفه‌های خیلی بیشتری نسبت به این لیست دارد. هر یک از وقفه‌ها را در مواقع لزوم بررسی خواهیم کرد. اگر به جدول صفحه قبل دقت کنید ملاحظه می‌نمایی که تعداد محدودی از بایت‌ها برای وقفه کنار گذاشه شده است. برای مثال، جمعا دو کلمه (یا چهار بایت)، از خانه 0x0006 تا 0x0008 برای تقاضای وقفه خارجی دو در نظر گرفته شده است. معمولا روال سرویس برای یک وقفه، طولانی‌تر از آن است

که در فضای حافظه اختصاص داده شده، گنجانده شود. به همین دلیل، یک دستور پرش در جدول بردار وقفه قرار گرفته است، تا به آدرس ISR اشاره کند.

هم چنین در این جدول توجه کنید که فقط دو کلمه از فضای ROM به پایه ریست اختصاص یافته است، که خانه‌های صفر و یک حافظه ROM می‌باشند. به همین دلیل، هنگام برنامه‌نویسی باید یک پرش به عنوان اولین دستور قرار داده، و پردازنده را از جدول بردار وقفه به محل دورتری هدایت کنیم.

```
        .ORG    0                   ;wake-up ROM    reset location
        JMP     MAIN                ;bypass interrupt vector table

        ;--------------------------- the wake-up program
        .ORG    $100
MAIN:   ....
        ....                        ;enable interrupt flags
```

فعال سازی و غیرفعال سازی یک وقفه

پس از ریست، همه وقفه‌ها غیرفعال می‌شوند، یعنی در صورت درخواست سرویس، میکروکنترلر به آنها پاسخ نخواهد داد. وقفه‌ها باید بوسیله نرم‌افزار فعال شوند تا میکروکنترلر به آنها پاسخ دهد. بیست D7 از SREG (ثبات وضعیت) مسئول فعال و غیرفعال کردن تمامی وقفه‌ها می‌باشد. شکل زیر ثبات وضعیت را نشان می‌دهد. بیت I کار غیرفعال کردن تمام وقفه‌ها را آسان می‌کند. با استفاده از فقط یک دستور CLI (پاک کردن وقفه یا Clear Interrupt)، می‌توانیم هنگام انجام وظایف حساس، یک را صفر کنیم.

Status Register

Bit	7	6	5	4	3	2	1	0
Flag	I	T	H	S	V	N	Z	C

The representation of the 8 bits in the status register are:

- Bit 0: Carry Flag
- Bit 1: Zero Flag
- Bit 2: Negative Flag
- Bit 3: Two's Complement Overflow Flag
- Bit 4: Sign Bit
- Bit 5: Half Carry Flag
- Bit 6: Bit Copy Storage
- Bit 7: Global Interrupt Enable

مراحل فعال سازی یک وقفه

برای فعال کردن هر یک از وقفه‌ها مراحل زیر را طی می‌کنیم:

۱- بیت D7 از ثبات وضعیت باید یک شود تا وقفه‌ها اتفاق بیافتند. این کار با استفاده از دستورالعمل SEI (یک کردن وقفه یا SEt Interrupt) انجام می‌شود.

۲- اگر I=1 شود، هر وقفه با یک کردن بیت پرچم فعال‌ساز وقفه (IE) مربوط به آن وقفه، فعال می‌شود. چند ثبات وجود دارند که بیت‌های فعال‌ساز وقفه را در خود نگه می‌دارند. شکل زیر نشان می‌دهد که ثبات TIMSK (Timer Interrupt MaSk) بیت‌های فعال‌ساز وقفه تایمرهای صفر و یک و دو را در خود نگه می‌دارد. اگر I=0 باشد به هیچ وقفه‌ای پاسخ داده نخواهد شد، حتی اگر بیت فعال‌ساز وقفه مربوط به آن یک باشد.

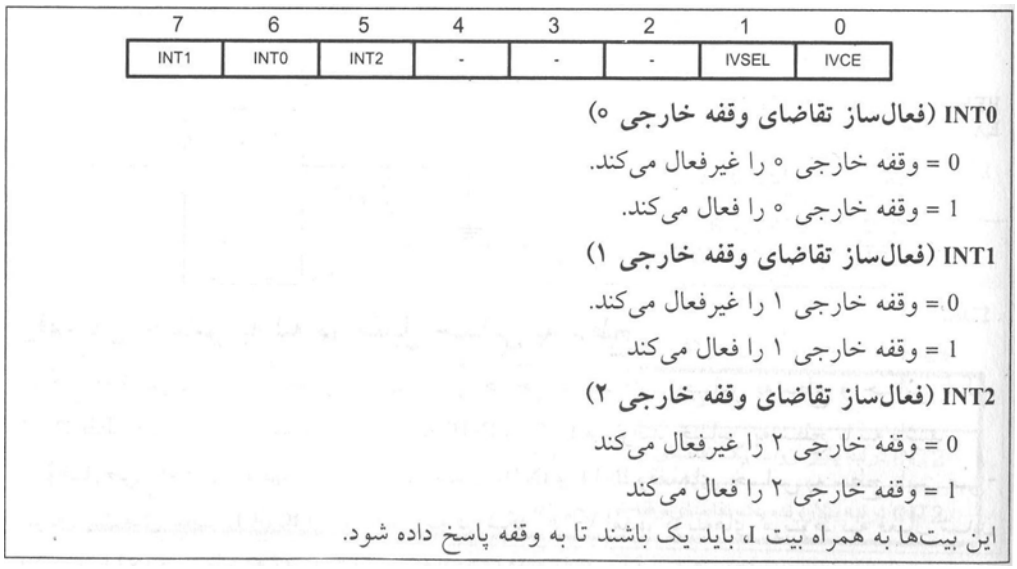
The Timer/Counter Interrupt Mask Register - TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	x	0	

برای اینکه به وقفه پاسخ داده شود، باید این بیت‌ها به همراه بیت I یک باشند. پس از فعال شدن وقفه، بیت I توسط AVR پاک می‌شود تا وقفه دیگری نتواند به میکروکنترلر، هنگام سرویس دادن به وقفه جاری، سیگنال ارسال کند. در انتهای ISR، دستورالعمل RETI دوباره I را یک کرده و به بقیه وقفه‌ها امکان کار کردن را می‌دهد.

وقفه‌های خارجی INT0، INT1 و INT2

سه وقفه سخت‌افزاری خارجی در میکروکنترلر ATmega32 وجود دارد: INT0، INT1 و INT2. این وقفه‌ها به ترتیب بر روی پایه‌های PD2، PD3 و PB2 قرار دارند. وقفه‌های سخت‌افزاری پیش از تحریک شدن، باید فعال شوند. این کار توسط بیت INTx که در ثبات GICR قرار گرفته، انجام می‌شود.



7	6	5	4	3	2	1	0
INT1	INT0	INT2	-	-	-	IVSEL	IVCE

INT0 (فعال‌ساز تقاضای وقفه خارجی ۰)

0 = وقفه خارجی ۰ را غیرفعال می‌کند.

1 = وقفه خارجی ۰ را فعال می‌کند.

INT1 (فعال‌ساز تقاضای وقفه خارجی ۱)

0 = وقفه خارجی ۱ را غیرفعال می‌کند.

1 = وقفه خارجی ۱ را فعال می‌کند

INT2 (فعال‌ساز تقاضای وقفه خارجی ۲)

0 = وقفه خارجی ۲ را غیرفعال می‌کند

1 = وقفه خارجی ۲ را فعال می‌کند

این بیت‌ها به همراه بیت I، باید یک باشند تا به وقفه پاسخ داده شود.

INT0 به صورت پیش فرض، یک وقفه حساس به سطح پایین است. یعنی وقتی یک سیگنال پایین (صفر) به پایه PD2 اعمال شود، کنترلر وقفه را دریافت کرده و به خانه 0x0002 جدول بردار وقفه، به سرویس ISR پرش می‌کند.

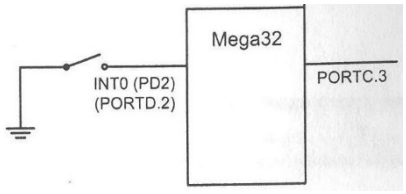
برای درک بهتر وقفه‌های سخت‌افزاری خارجی، مثال زیر را مورد بررسی قرار دهید. در این برنامه، میکروکنترلر پیوسته در حلقه HERE می‌چرخد. هر موقع سوئیچ موجود بر روی INT0 (پایه PD2) فعال شد، میکروکنترلر از حلقه بیرون رفته و به خانه 0x0002 جدول بردار وقفه پرش می‌کند. ISR مربوط به INT0، پایه PC0 را (از صفر به یک و بالعکس) تغییر وضعیت می‌دهد. اگر پس از اجرای دستورالعمل RETI، پایه INT0 هنوز پایین (صفر) باشد، میکروکنترلر وقفه را دوباره آغاز می‌کند. بنابراین، اگر می‌خواهیم که ISR یک بار اجرا شود، پایه INT0 باید قبل از اجرای RETI با بالا برگردانده شود، یا باید بر طبق مثال بعدی، وقفه را حساس به لبه کنیم.

مثال: فرض کنید INT0 به یک سوئیچ که در حالت عادی یک است، متصل می‌باشد. هر موقع صفر شد، باید PORTC.3 تغییر وضعیت دهد. برنامه‌ای بنویسید که این کار را انجام دهد.

```
.INCLUDE "M32DEF.INC"
.ORG    0                           ;location for reset
        JMP     MAIN
.ORG    0x0002                      ;vect. location for external int.
        JMP     EX0_ISR

MAIN:   LDI     R20,HIGH(RAMEND)
        OUT     SPH,R20
        LDI     R20,LOW(RAMEND)
        OUT     SPL,R20             ;initialize stack
        SBI     DDRC,3              ;PORTC.3 = output
        SBI     PORTD,2             ;pull-up activated
        LDI     R20,1<<INT0         ;Enable INT0
        OUT     GICR,R20
        SEI                         ;Enable Global Interrupts
HERE:   RJMP    HERE

EX0_ISR:
        IN      R21,PINC
        LDI     R22,0X08
        EOR     R21,R22
        OUT     PORTC,R21
        RETI
```



وقفه‌های حساس به لبه در مقابل حساس به سطح

دو نوع فعال شدن برای وقفه‌های سخت‌افزاری خارجی وجود دارد: حساس به سطح و حساس به لبه. INT2 فقط حساس به لبه است. در حالی که INT0 و INT1 می‌توانند حساس به سطح یا لبه باشند.

همانطور که قبلا گفته شد، پس از ریست شدن، INT0 و INT1 وقفه‌های حساس به سطح پایین می‌شوند. بیت‌های ثبات MCUCR، همانطور که در شکل آمده گزینه‌های مربوط به فعال شدن INT0 و INT1 در سطح یا لبه را نشان می‌دهد.

مثال: مثال قبلی را دوباره بنویسید، بطوری که وقتی سوئیچ صفر شد، portc.3 فقط یکبار تغییر وضعیت دهد.

```
.INCLUDE "M32DEF.INC"
.ORG    0                           ;location for reset
        JMP     MAIN
.ORG    0x0002                      ;vect. location for external int.
        JMP     EX0_ISR

MAIN:   LDI     R20,HIGH(RAMEND)
        OUT     SPH,R20
        LDI     R20,LOW(RAMEND)
        OUT     SPL,R20             ;initialize stack
        LDI     R20,0X02            ;make INT0 Falling edge triggered
        OUT     MCUCR,R20
        SBI     DDRC,3              ;PORTC.3 = output
        SBI     PORTD,2             ;pull-up activated
        LDI     R20,1<<INT0         ;Enable INT0
        OUT     GICR,R20
        SEI                         ;Enable Global Interrupts
HERE:   RJMP    HERE

EX0_ISR:
        IN      R21,PINC
        LDI     R22,0X08
        EOR     R21,R22
        OUT     PORTC,R21
        RETI
```

توجه کنید که تنها تفاوت این برنامه با برنامه قبلی در دستورات زیر می‌باشد:

```
        LDI     R20,0X02            ;make INT0 Falling edge triggered
        OUT     MCUCR,R20
```

که INT0 را به یک وقفه حساس به لبه تبدیل می‌کند. وقتی یک لبه پایین رونده سیگنال به پایه INT0 اعمال شود، PORTC.3 تغییر وضعیت می‌دهد. برای تغییر وضعیت دوباره در LED، باید یک سیگنال بالا به پایین دیگر به INT0 اعمال شود. که این موضوع بر خلاف مثال قبلی است. در مثال قبلی، وقفه بطور طبیعی حساس به سطح است و تا وقتی INT0 در سطح پایین نگه داشته شود، PORTC.3 تغییر وضعیت می‌دهد. ولی در این مثال برای روشن کردن تغییر وضعیت در PORTC.3، پالس INT0 باید دوباره بالا برده شده و سپس پایین بیاید تا برای فعال سازی وقفه، یک لبه پایین رونده ایجاد کند.

7	6	5	4	3	2	1	0
SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00

ISC00،ISC01 (بیت‌های کنترل احساس در وقفه)

این بیت‌ها سطح یا لبه‌ای را که پایه INT0 را فعال می‌کند، بر طبق جدول زیر، مشخص می‌نمایند.

ISC11	ISC10		توضیحات
0	0		INT1 درسطح پایین یک تقاضای وقفه تولید می کند
0	1		هر تغییر منطقی در INT1 یک تقاضای وقفه تولید می کند
1	0		INT1 درلبه پایین رونده یک تقاضای وقفه تولید می کند
1	1		INT1 درلبه بالارونده یک تقاضای وقفه تولید می کند

ISC10،ISC11

این بیت‌ها سطح یا لبه‌ای را که پایه INT1 را فعال می‌کند، بر طبق جدول زیر، مشخص می‌نمایند.

ISC11	ISC10		توضیحات
0	0		INT1 درسطح پایین یک تقاضای وقفه تولید می کند
0	1		هر تغییر منطقی در INT1 یک تقاضای وقفه تولید می کند
1	0		INT1 درلبه پایین رونده یک تقاضای وقفه تولید می کند
1	1		INT1 درلبه بالارونده یک تقاضای وقفه تولید می کند

شکل ۱۰-۷ ثبات MCUCR (ثبات کنترل MCU)

در شکل زیر بیت ISC2 از ثبات MCUCSR فعال شدن INT2 در لبه پایین رونده یا لبه بالا رونده، مشخص می‌کند. پس از ریست شدن، ISC2 صفر است، یعنی وقفه سخت‌افزاری خارجی در INT2 حساس به لبه پایین رونده است.

7	6	5	4	3	2	1	0
JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF

ISC2

این بیت سطح یا لبه‌ای را که پایه INT2 را فعال می‌کند، بر طبق جدول زیر، مشخص می‌نماید.

ISC2		توضیحات
0		INT2 درلبه پایین رونده یک تقاضای وقفه تولید می کند
1		INT2 درلبه بالارونده یک تقاضای وقفه تولید می کند

شکل ۱۰-۸ ثبات MCUCSR (ثبات وضعیت و کنترل MCU)

نمونه‌برداری وقفه‌های حساب به لبه و حساس به سطح

وقفه حساس به لبه (لبه پایین رونده، لبه بالا رونده، یا تغییر سطح) توسط میکروکنترلر لچ (نگهداری) شده و در بیت‌های INTFx از ثبات GIFR ذخیره می‌شود. یعنی وقتی یک وقفه خارجی در مُد حساس به لبه

(لبه پایین رونده، لبه بالا رونده، یا تغییر سطح) است، هنگام تحریک وقفه، فلگ INTFx مربوط به آن یک می‌شود. در این حالت اگر وقفه فعال باشد (بیت INTx و بیت I یک باشند)، میکروکنترلر به جدول بردار وقفه پرش کرده و فلگ INTFx بطور خودکار پاک می‌شود، در غیر اینصورت این فلگ فعال باقی می‌ماند. فلگ INTFx را می‌توان با نوشتن یک بر روی آن پاک کرد. برای مثال فلگ INTF1 را می‌تولان با استفاده از دستورات زیر پاک (صفر) کرد:

```
LDI     R20,(1<<INTF1)          ; R20 = 0x80
OUT     GIFR,R20                ; clear the INTF1 flag
```

7	6	5	4	3	2	1	0
INTF1	INTF0	INTF2	-	-	-	-	-

شکل ۱۰-۹ ثبات GIFR (ثبات پرچم وقفه عمومی)

توجه کنید که در وقفه‌های حساس به لبه (لبه پایین رونده، لبه بالا رونده، یا تغییر سطح)، هر پالس باید حداقل یک چرخه دستور طول بکشد، تا مطمئن شویم که این تغییر توسط میکروکنترلر دیده شده است. یعنی پالس‌های کوتاه‌تر از یک چرخه ماشین، تولید وقفه را تضمین نمی‌کنند.

وقتی یک وقفه خارجی در مُد حساس به سطح است، وقفه لچ (نگهداری) نمی‌شود. یعنی هنگام وقوع یک وقفه، فلگ INTFx بدون تغییر مانده، و وضعیت پایه مستقیم خوانده می‌شود. در نتیجه وقتی یک وقفه حساس به سطح است، باید پایه حداقل ۵ چرخه ماشین پایین نگه داشته شود تا بتوان آن را تشخیص داد.

الویت وقفه‌ها در میکروکنترلر AVR

موضوع دیگری که باید به آن بپردازیم این است که اگر دو وقفه همزمان تحریک شوند، چه اتفاقی خواهد افتاد؟ کدامیک از این وقفه‌ها ابتدا سرویس مورد نظر را دریافت خواهند کرد.

اگر دو وقفه در یک زمان تحریک شوند، وقفه‌ای که اولویت بیشتری دارد اول سرویس‌دهی می‌شود. اولویت هر وقفه بستگی به آدرس وقفه در بردار وقفه دارد. وقفه‌ای که آدرس پایین‌تری (کوچکتری) دارد، از اولویت بالاتری برخوردار می‌باشد. به عنوان مثال آدرس وقفه خارجی صفر، برابر 0x0002 است. در حالی که آدرس وقفه خارجی دو، برابر با 0x0006 است. بنابراین وقفه خارجی صفر از اولویت بالاتری برخوردار است و اگر هر دوی این وقفه‌ها با هم تحریک شوند، وقفه خارجی صفر، اول سرویس‌دهی می‌شود.

وقفه درون یک وقفه دیگر

وقتی میکروکنترلر در حال اجرای ISR مربوط به یک وقفه است و وقفه دیگری رخ دهد، چه اتفاقی خواهد افتاد؟ وقتی میکروکنترلر شروع به اجرای یک ISR می‌کند و بیت I از ثبات وضعیت را غیرفعال می‌نماید. اینکار باعث می‌شود تمام وقفه‌ها غیرفعال شوند و هیچ وقفه دیگری هنگام سرویس دادن به یک وقفه رخ ندهد. وقتی دستور RETI اجرا شد، میکروکنترلر بیت I را فعال می‌کند. اینکار موجب می‌شود تا دیگر وقفه‌ها سرویس‌دهی شوند. می‌توانید بیت I را با استفاده از دستورالعمل SEI، یک کنید. ولی اینکار را با احتیاط انجام دهید. برای مثال، در یک وقفه خارجی حساس به سطح، فعال کردن بیت I در حالی که پایه وقفه هنوز در حالت تحریک است، موجب می‌شود بطور بی‌نهایت وارد ISR شده و سرریز پشته و اتفاقات غیر قابل پیش‌بینی روی دهد.

ذخیره‌سازی فلگ‌های ثبات وضعیت

فلگ‌های ثبات وضعیت (SREG)، مخصوصا در جاهایی از برنامه که پرش‌های شرطی وجود دارد، بسیار مهم هستند. بنابراین، اگر فلگ‌ها در یک روتین اینتراپت تغییر پیدا می‌کنند، باید ثبات SREG را ذخیره نماییم.

```
1   Sample_ISR:
2           PUSH    r20
3           IN      R20,SREG
4           PUSH    R20
5           .
6           .
7           .
8           POP     R20
9           OUT     SREG,R20
10          POP     R20
11          RETI
```

تاخیر وقفه

مقدار زمان از لحظه‌ای که یک وقفه تحریک می‌شود تا لحظه‌ای که سرویس خود را دریافت کند، تاخیر وقفه (interrupt latency) نامیده می‌شود. این تاخیر چهار چرخه ماشین طول دارد. در طی این زمان، ثبات شمارنده برنامه (PC) بر روی پشته قرار می‌گیرد و بیت I از ثبات SREG پاک می‌شود تا تمام وقفه‌ها غیرفعال شوند. از آنجایی که پردازنده اجرای دستور جاری را قبل از سرویس دادن به وقفه تمام می‌کند، طول تاخیر وقفه می‌تواند تحت تاثیر نوع دستوری که پردازنده هنگام رخ دادن وقفه در حال اجرای آن بوده،

قرار بگیرد. این تاخیر، در حالت‌هایی که دستورالعمل در حال اجرا دو چرخه ماشین (یا بیشتر) طول می‌کشد (مانند دستورالعمل MUL)، نسبت به دستوراتی که فقط یک چرخه ماشین طول می‌کشند (مانند ADD)، کمی طولانی‌تر خواهد بود.

آزمایشی که باید انجام دهید:

برای این آزمایش سخت‌افزاری طراحی کرده‌ایم که دارای سه کلید و یک صفحه نمایش LCD و یک LED است. کلید INC به اینتراپت خارجی صفر، کلید DEC به اینتراپت خارجی یک و کلید REG به اینتراپت خارجی دو وصل شده است.

برنامه‌ای بنویسید که فشردن شدن کلید REG باعث روشن شدن LED شود و فشرده شدن مجدد آن، LED را خاموش نماید.

اگر LED خاموش بود. فشرده شدن کلید INC باعث افزایش یک واحدی روی عدد صفر موجود در روی LCD شود و با فشردن شدن کلید DEC این عدد یک واحد کاهش یابد. اگر LED روشن بود، این افزایش و کاهش ده واحدی باشد. همانطور که در برنامه اجرایی مشاهده می‌نمایید.