کتاب PDF

میکرو کنترلرهای AVR با پروژه های ۱۰۰٪ عملی

تألیف جابر الوندی

منبع درس ریزپردازنده۱ دانشگاه پیام نور

فایل PDF موجود نسخه RF، در بین سایر فایل های PDF ریزپردازنده موجود در سایت های دانلود کتاب، کامل ترین فایل PDF محسوب میشود. چرا که در سایر کتاب های PDF مشابه، علاوه بر ناقص بودن صفحات کتاب، نظم و ترتیب خاصی بین صفحات وجود ندارد که این نقص، باعث ایجاد مشکل و سردرگمی برای دانشجویان در هنگام مطالعه کتاب PDF می شود.

از همین رو ویرایش RF کتاب پی دی اف ریزپردازنده۱ را که نواقص فایل های قبلی را برطرف کرده، برای استفاده دانشجویان عزیزی که به کتاب اصلی این درس دسترسی ندارند، آماده کرده ایم.

Edited by Aref.b *4123ph@gmail.com*

فصل ۵

وقفه‌ها (Interrupts)

اهداف

۱. آشنایی با روش‌های Polling و Interrupt
۲. معرفی منابع وقفه و آشنایی با مفهوم بردار وقفه
۳. آشنایی با نوشتن برنامه به روش وقفه در زبان C

۵-۱ وقفه‌ها (Interrupts)

وقفه سخت‌افزاری از میکروکنترلر است که میکروکنترلر را برای پاسخ گویی به رویدادهای لحظه‌ای مجهز می‌کند. می‌خواهیم ابتدا به مفهوم وقفه بپردازیم و در ادامه مطلب به اهمیت آن اشاره کنیم.

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

روش‌های بررسی یک رویداد از جانب CPU

به طور کلی CPU برای تشخیص رویدادهای داخلی (مانند تست پرچم سرریز تایمر یا کانتر) و رویدادهای خارجی (مانند تشخیص لبهٔ پایین‌رونده یک پالس و یا وصل شدن یک کلید) می‌تواند از دو روش زیر استفاده کند:

۱. **روش سرکشی (Polling):** در این روش کاربر توسط برنامه‌نویسی با فواصل زمانی مشخص و دائماً رویداد موردنظر را بررسی می‌کند 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 است یا خیر، به این روش سر کشی یا Polling می‌گوییم. به طور نمونه اگر در برنامه‌نویسی، ما برای بررسی فشردن یک کلید و یا فعال شدن یک پرچم خاص هر چند لحظه یک بار بخواهیم بررسی را انجام دهیم در واقع به روش Polling عمل کرده‌ایم. عیب این روش تلف کردن وقت CPU است اما به این معنی نیست که این روش فایده و کاربردی ندارد چه بسا در بسیاری از مواقع از آن استفاده می‌کنیم.

۲. **روش وقفه (Interrupt)**: در مثالی که طرح کردیم، یک راه ساده‌تر آن است که شما به درس خواندن خود ادامه بدهید و منتظر زنگ شخص مورد نظر بمانید و هر موقع زنگ منزل را زدند بروید به آن پاسخ دهید و بعد از اتمام کارتان، برگردید و به درس خواندن خود ادامه دهید به این روش وقفه یا Interrupt می‌گوییم. می‌بینید که در این حالت وقت شما صرف سرکشی برای آمدن شخص مورد نظر نشده است و تا آمدن آن شخص شما توانسته‌اید کار مفید دیگری را انجام دهید. در این روش CPU بدون در نظر گرفتن رویداد، به انجام سایر اعمال مشغول می‌شود و با وقوع اتفاق مورد نظر، CPU انجام خط جاری برنامه را متوقف کرده و به بردار وقفه مربوطه پرش می‌کند و زیر روال سرویس وقفه ISR (Interrupt Service Routin) را اجرا می‌کند و پس از اجرای این زیر روال به خطی از برنامه که آن را قطع کرده بود بر می‌گردد و ادامه برنامه را اجرا می‌کند. در بسیاری از کاربردها، ما از وقفه‌های داخلی و بیرونی استفاده می‌کنیم. مثلاً می‌توان وقفه سرریز تایمر یا کانتر، وقفه مقایسه‌گر آنالوگ، وقفه مبدل آنالوگ به دیجیتال، وقفه تبادل سریال، وقفه ارتباط SPI و وقفه‌های بیرونی و ... را نام برد.

بُردارهای وقفه و Reset در ATmega16

ابتدا بهتر است منظورمان از بُردار را مشخص نمائیم. شما موقعیکه برنامه را می‌نویسید، برنامه تبدیل به یک سری کد ماشین (HEX) می‌شود که از مکان صفر حافظه Flash توسط پروگرمر نوشته می‌شود. حال اگر بیاییم یک مکان مشخص از حافظه ثابت را به وقفه خاصی اختصاص دهیم و سایر قسمت‌های برنامه را در مکان دیگری ذخیره نمائیم به طوری که با تحریک آن وقفه، برنامه به آن مکانی که به طور ثابت و مشخص به وقفه اختصاص داده بودیم پرش کند به آن مکان بردار وقفه می‌گوییم. در جدول ۵-۱ آدرس و شمارهٔ بردار تمام منابع وقفه در میکروکنترلر ATmega16 آورده شده است.

نحوه‌ی تعریف تابع وقفه در نرم‌افزار CodeVisionAVR

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

از هر وقفه‌ای که بخواهیم استفاده کنیم باید ابتدا رجیسترهای مربوط به آن وقفه را تنظیم کنیم و در برنامه، تابع وقفه را به فُرم فوق بنویسیم. کلمه کلیدی interrupt تعیین کننده تابع از نوع وقفه می‌باشد و

شماره بردار وقفه نیز طبق جدول ۵-۱ تعیین می‌گردد در نرم‌افزار CodeVisionAVR برای این شماره‌ها نام معادلی تعریف شده است که می‌توان از آنها نیز استفاده کرد. مثال:

```
interrupt [2] void ext_int0_isr(void) {دستورالعمل‌ها}
```

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

```
interrupt [EXT_INT0] void ext_int0_isr(void) {دستورالعمل‌ها}
```

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

مراحل اجرای یک وقفه

میکروکنترلر زمانی که وقفه داخلی یا خارجی را دریافت می‌کند مراحل زیر را انجام می‌دهد:

۱. دستوری که در حال اجرای آن می‌باشد را به پایان رسانده و آدرس دستورالعمل بعدی را در حافظه پشته ذخیره می‌کند.
۲. به بردار وقفه مربوطه پرش می‌کند و برنامه یا زیر روال سرویس وقفه (ISR) را انجام می‌دهد.
۳. پس از اتمام زیر روال وقفه، توسط دستور اسمبلی RETI از وقفه بر می‌گردد. البته در زبان C ما دستور RETI را بکار نمی‌گیریم اما کامپایلر بعد از تفسیر برنامه از آن استفاده می‌کند.
۴. آدرس دستورالعملی که در مرحله یک در حافظه پشته ذخیره کرده بود را بر می‌دارد و با قرار

دادن آن آدرس، در شمارنده برنامه (PC)، به ادامه برنامه بر می‌گردد.

رجیستر وضعیت SREG (Status 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	

در مورد بیت‌های رجیستر وضعیت در فصل اول توضیح دادیم حال وقفه کلی را یادآوری می‌کنیم.

- **بیت 7 - I** (Global Interrupt Enable)

اگر این بیت را یک کنیم وقفه کلی (همگانی یا سراسری) فعال می‌گردد و در این حالت هست که دیگر وقفه‌ها در صورت فعال بودن و تحریک شدن می‌توانند رخ دهند. در صورتی که این بیت صفر باشد و دیگر وقفه‌ها فعال باشند و حتی تحریک هم شوند قابل اجرا توسط CPU نخواهند بود پس این بیت، کنترلی برای رخ دادن دیگر وقفه‌هاست. در زبان برنامه‌نویسی C این بیت به صورت زیر فعال و غیر فعال می‌شود.

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

۲-۵ وقفه‌های خارجی (External Interrupts)

از وقفه‌های خارجی معمولاً برای تشخیص پالس استفاده می‌شود حال این پالس می‌تواند حاوی اطلاعات گوناگونی باشد به طور مثال پالس‌هایی که از طرف یک سنسور خاص مانند SMT160 یا سنسور اُپتوکانتر ارسال می‌گردد یا پالسی که در اثر وصل شدن یک میکروسوئیچ ایجاد می‌گردد یا پالسی که از طرف یک مبدل آنالوگ به دیجیتال بیرونی ارسال می‌شود و یا پالسی که از هر تراشه و رویداد دیگری ارسال می‌گردد که ما بخواهیم توسط میکروکنترلر خیلی سریع به آن پاسخ دهیم از وقفه خارجی بهره می‌گیریم.

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 5 - INT2** (External Interrupt Request 2 Enable)

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

- **بیت 6 - INT0** (External Interrupt Request 0 Enable)

با یک کردن این بیت وقفه خارجی صفر فعال می‌گردد و اگر وقفه کلی نیز فعال باشد می‌تواند وقفه خارجی صفر در صورت رخ دادن اجرا گردد. اگر این بیت یک شود عملکرد عادی PD2 به عنوان I/O قطع شده و به عنوان INT0 عمل می‌کند.

➢ **بیت 7 - INT1** (External Interrupt Request 1 Enable)

با یک کردن این بیت وقفه خارجی یک فعال می‌گردد و اگر وقفه کلی نیز فعال باشد می‌تواند وقفه خارجی یک در صورت رخ دادن اجرا گردد. اگر این بیت یک شود عملکرد عادی PD3 به عنوان I/O قطع شده و به عنوان INT1 عمل می‌کند.

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 5 - INTF2** (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 می‌گردد و در خواست اجرای زیر روال وقفه را می‌دهد و برنامه به تابع وقفه دو پرش خواهد کرد و بعد از اجرای آن، به طور اتوماتیک این پرچم پاک می‌گردد.

➢ **بیت 6 - INTF0** (External Interrupt Flag 0)

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

➢ **بیت 7 - INTF1** (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 می‌گردد و در خواست اجرای زیر روال وقفه را می‌دهد و برنامه به تابع وقفه یک پرش خواهد کرد و بعد از اجرای آن، به طور اتوماتیک این پرچم پاک می‌گردد.

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	

➢ **بیت‌های ISC11, ISC10 – 3 , 2** (Interrupt Sense Control 1 Bit 1 and Bit 0)

توسط این دو بیت می‌توان نحوه‌ی تحریک شدن وقفه خارجی یک را تعیین نمود. تحریک می‌تواند در سطح صفر یا هر تغییر منطقی یا لبهٔ پایین رونده و یا لبهٔ 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

➢ **بیت‌های ISC01, ISC00 – 1 , 0** (Interrupt Sense Control 0 Bit 1 and Bit 0)

توسط این دو بیت می‌توان نحوه‌ی تحریک شدن وقفه خارجی صفر را تعیین نمود. تحریک می‌تواند در سطح صفر یا هر تغییر منطقی یا لبهٔ پایین رونده و یا لبهٔ 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

رجیستر MCUCSR (MCU Control and Status Register)

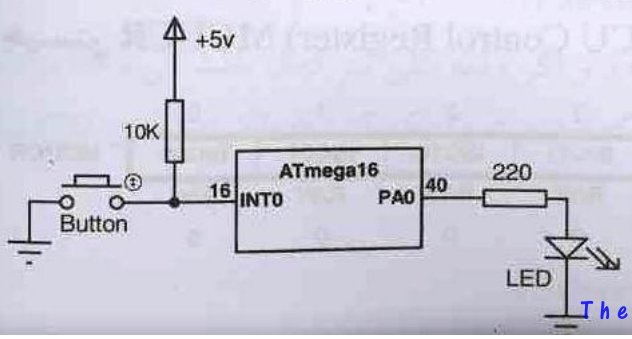
Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

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

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

مثال ۵-۱: برنامه‌ای 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؟

حل :



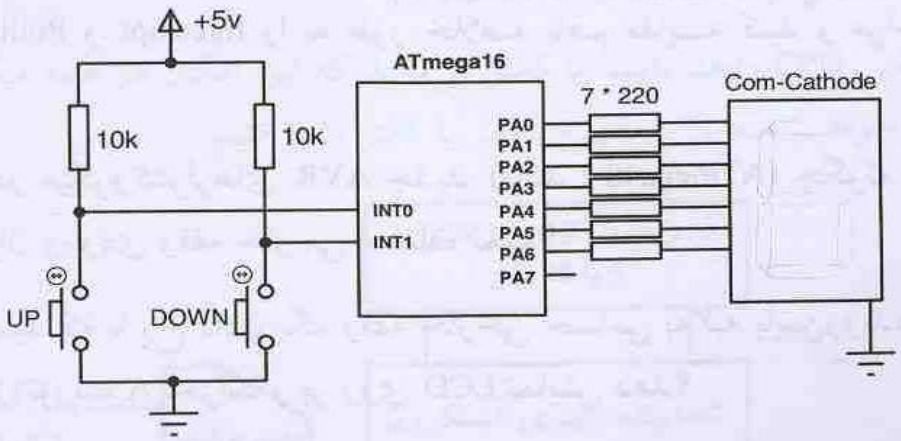
شکل ۵-۱ استفاده از وقفه خارجی صفر [مثال ۵-۱]

در مثال ۵-۱ پایه وقفه خارجی صفر را توسط یک مقاومت Pull-up به 5v+ وصل کرده‌ایم. این بدان معنی است که در حالت عادی پایه INT0 در وضعیت یک منطقی می‌باشد و به محض فشردن کلید یک لبهٔ پایین رونده بوجود می‌آید و موجب وقفه خوردن میکرو شده و برنامه در این لحظه به تابع وقفه خارجی صفر پرش می‌کند و LED را معکوس می‌کند و بر می‌گردد.

```
#include <mega16.h>                          //معرفی میکروکنترلر استفاده شده
interrupt [EXT_INT0] void ext_int0_isr(void){ //تابع وقفه خارجی صفر
while(PIND.2==0);                            //تست برای رها شدن کلید
PORTA.0=!(PORTA.0);                          //معکوس کردن LED متصل به پایه PA0
}
void main(){                                 //تابع اصلی برنامه
PORTA.0=0;                                   //وضعیت اولیه LED خاموش است
DDRA.0=1;                                    //پایه PA0 به عنوان خروجی تعیین می‌شود
GICR|=0x40;                                  //فعال کردن وقفه خارجی صفر
MCUCR=0x02;                                  //تنظیم حساسیت وقفه خارجی صفر به لبهٔ پایین رونده
GIFR=0x40;                                   //مقدار دهی اولیه پرچم وقفه خارجی صفر
#asm("sei")                                  //فعال کردن وقفه کلی یعنی یک کردن بیت I در رجیستر وضعیت
while (1);                                   //حلقه بی نهایت و بیکار برنامه
}
```

مثال ۵-۲ : مطابق شکل ۵-۲، دو کلید فشاری به وقفه‌های خارجی صفر و یک متصل نموده‌ایم. می‌خواهیم برنامه‌ای 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؟

حل :



شکل ۵-۲ استفاده از وقفه‌های خارجی صفر و یک [مثال ۵-۲]

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

```
#include <mega16.h>                    //معرفی میکروکنترلر استفاده شده
flash unsigned char number[]={         //کدهای 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 کدهای 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(PIND.2==0);                      //تست برای رها شدن کلید UP//
}
interrupt [EXT_INT1] void ext_int1_isr(void){ //تابع وقفه خارجی یک
if(i>0) PORTA=number[--i];             //اگر عدد بزرگتر از صفر بود یک واحد شمارشگر کاهش یابد
while(PIND.3==0);                      //تست برای رها شدن کلید DOWN//
}
void main(){                           //تابع اصلی برنامه
PORTA=number[0];                       //نمایش اولیه را عدد صفر قرار می‌دهیم
DDRA=0xFF;                             //تعریف پورت A به عنوان خروجی
GICR|=0xC0;                            //فعال کردن وقفه‌های خارجی صفر و یک
MCUCR=0x0A;                            //تنظیم حساسیت وقفه‌های خارجی صفر و یک به لبهٔ پایین رونده
GIFR=0xC0;                             //مقدار دهی اولیه پرچم‌های وقفه‌های خارجی صفر و یک
#asm("sei")                            //فعال کردن وقفه کلی
while (1);                             //حلقه بی نهایت و بیکار برنامه
}
```

تمرین‌های فصل ۵

۵-۱ روش‌های Polling و Interrupt را به طور خلاصه باهم مقایسه کنید و مراحل اجرای وقفه را بنویسید؟

۵-۲ تحقیق کنید در میکروکنترلرهای AVR جدید (مانند : ATmega48) چگونه می‌توان از هر پایه دلخواه به عنوان ورودی وقفه خارجی استفاده نمود؟

۵-۳ برنامه‌ای بنویسید که با رخ دادن یک وقفه خارجی حساس به لبه پایین‌رونده، یک دیتای ۸ بیتی علامتدار را از پورت A بخواند و بر روی LCD نمایش دهد؟

۵-۴ یک دیتای سریال به پایه PB0 اعمال کرده‌ایم به طوری که قالب سریال شامل یک بیت شروع کننده، ۸ بیت دیتا و یک بیت توقف می‌باشد و همراه با هر بیت یک کلاک ارسال می‌شود یعنی ۱۰ کلاک برای یک قالب خواهیم داشت. کلاک را به INT0 متصل نمائید و در لبهٔ پایین رونده کلاک، دیتای ۸ بیتی سریال را از پایه PD1 بخوانید و بر روی LCD نمایش دهید؟