دانشگاه صنعتی اصفهان
دانشکده مهندسی برق و کامپیوتر

دستورکار آزمایشگاه ریزپردازنده
دستورکار آزمایشگاه طراحی سیستم‌های دیجیتال2

(مبتنی بر ریزپردازنده ATMEGA16/32)

تهیه کننده:
زهرا محمدزاده

بررسی کننده:
دکتر امیر خورسندی

شهریور 1401

3 جلسه سوم

آشنایی با LCD کاراکتری و صفحه کلید ماتریسی

3.1 هدف

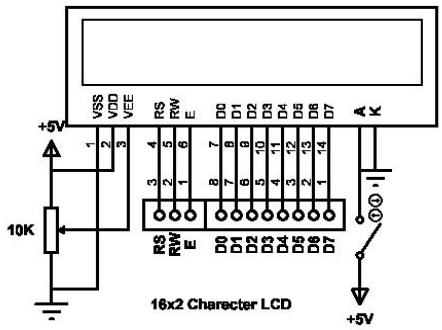
در این جلسه نمایشگر LCD و صفحه‌کلید ماتریسی به عنوان ادوات ورودی/خروجی متداول برای تعامل با کاربر بررسی می‌شوند. هم‌چنین کار با صفحه کلید به دو روش سرکشی و نیز استفاده از وقفه‌ها صورت خواهد پذیرفت.

3.2 معرفی LCD کاراکتری

به طور کلی LCD کاراکتری به منظور نمایش جملات و اطلاعات نوشتاری دلخواه به کاربر مورد استفاده قرار می‌گیرد.در پکیج آموزشی یک عدد LCD کاراکتری از نوع 16x2 (دارای 16 ستون و 2 ردیف) با نور پس زمینه‌ی به رنگ سبز یا آبی تحت بلوکی با عنوان 16x2 Character LCD تعبیه شده است.

3.2.1 آشنایی با مدار راه‌انداز، پایه‌ها و حالت‌های کاری

مدار راه‌انداز این LCD درشکل 3-1 مشاهده می‌شود. پایه‌های K و A برای تنظیم نور زمینه تعبیه شده‌اند و با استفاده از کلید کشویی تعبیه شده می‌توان این نور پس‌زمینه‌ی را وصل یا قطع نمود.



شکل 3-1: مدار راه‌انداز LCD کاراکتری

پایه‌های D0 تا D7 پایه‌های انتقال داده هستند. پایه‌ی E فعال‌ساز لچ داخلی است که برای ذخیره‌ی اطلاعات در LCD، لازم است یک سیگنال با لبه‌ی پایین رونده به آن اعمال شود. پایه‌ی R/W مشخص می‌کند که باید اطلاعات از LCD خوانده یا روی آن نوشته شود. پایه‌ی RS برای مشخص کردن این است که مقدار قرار گرفته روی D0 تا D7 دستور است یا داده: در صورتی که RS=0 باشد این مقدار به عنوان دستور و در صورتی که RS=1 باشد به عنوان داده (کد اسکی) در نظر گرفته می‌شود. پایه‌های VEE، VDD و VSS هم به ترتیب تنظیم کننده‌ی وضوح، ولتاژ تغذیه و زمین هستند. در این مدار، در سر راه VEE یک پتانسیومتر قرار دارد تا کاربر بتواند وضوح نمایشگر را در حد مطلوب تنظیم نماید.

ارتباط ریزپردازنده و LCD را می‌توان به دو صورت برقرار کرد:

ارتباط 4 سیمه : سریال

ارتباط 8 سیمه : موازی

ارتباط 4 سیمه از تعداد پایه کمتری از ریزپردازنده را اشغال می‌کند و از این جهت بهتر بوده و معمولاً از آن استفاده می‌شود. در برنامه Codevision نیز چنان‌چه از CodeWizard برای آماده‌سازی LCD استفاده شود، همین روش 4 سیمه به کار گرفته می‌شود.

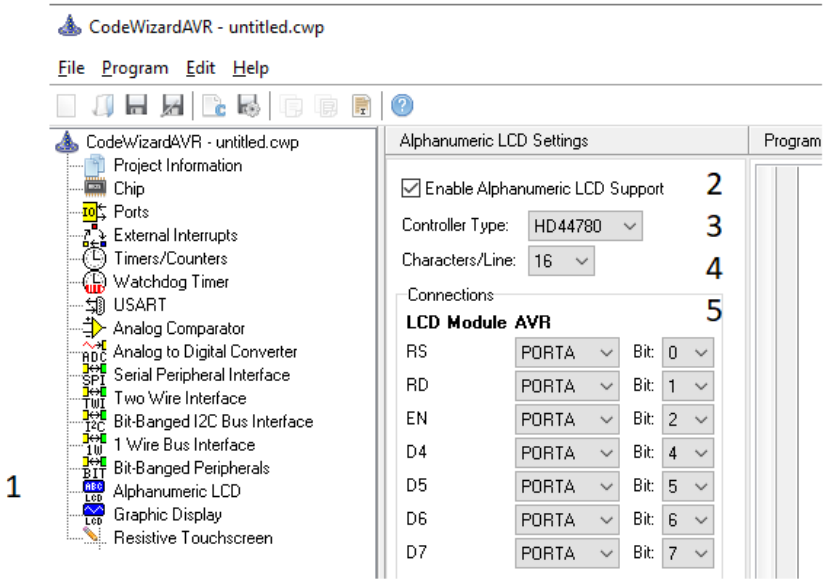
3.2.2 ارتباط 4 سیمه با ریزپردازنده

نحوه اتصال پایه های LCD به درگاه A ریزپردازنده بر روی پکیج آزمایشگاه در جدول 3-1 نشان داده شده است.

پایه‌ی ریزپردازنده	پایه‌ی LCD
PA.0	RS
PA.1	E
PA.3	(R/W)DB0*
PA.4-PA.7	DB4-DB7

جدول 3-1: نحوه اتصال پایه های LCD به درگاه A ریزپردازنده (*بر روی پکیج آموزشی پایه‌ی R/W از LCD به کانکتور با برچسب DB0 متصل شده است.)

بر اساس جدول بالا تنظیمات اولیه برای ارتباط 4 سیمه از طریق CodeWizard مانند شکل 3-2 انجام می‌شود.



1: انتخاب Alphanumeric LCD

2: فعال نمودن LCD

3: انتخاب کنترلر LCD

4: انتخاب تعداد کاراکتر در هر خط

5: تنظیمات اتصال درگاه میکرو به پایه‌های LCD (دقت شود که در تنظیمات کدویژن RD همان R/W است)

شکل 3-2: نمایی از تنظیمات کدویزار برای LCD

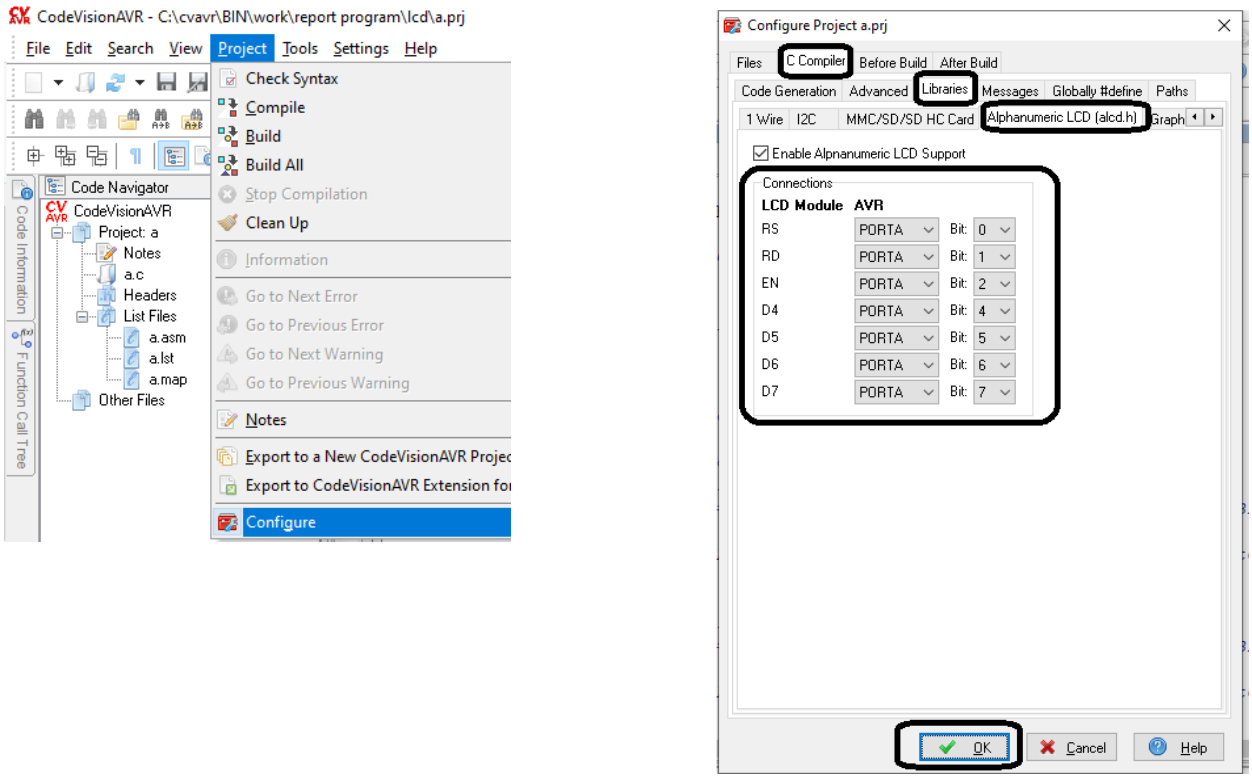
پس از انجام مراحل مذکور، کدهای ایجاد شده برای LCD مطابق برنامه 3-1 ایجاد می‌گردند.

برنامه 3-1

```
#include <mega16.h>
#include <alcd.h>
void main(void)
{
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) |
(0<<DDA1) | (0<<DDA0);
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) |
(0<<DDB1) | (0<<DDB0);
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) |
(0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);
```

```
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) |
(0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTA Bit 0
// RD - PORTA Bit 1
// EN - PORTA Bit 2
// D4 - PORTA Bit 4
// D5 - PORTA Bit 5
// D6 - PORTA Bit 6
// D7 - PORTA Bit 7
// Characters/line: 16
lcd_init;(16)

while (1)
   {
}
}
```

همچنین لازم به ذکر است که اگر در حین انجام پروژه سخت‌افزار تغییر نمود می‌توان مانندشکل 3-3 تغییرات را لحاظ نمود.



شکل 3-3: تغییر تنظیمات LCD

3.2.3 آشنایی با فایل سرآیند «alcd.h»

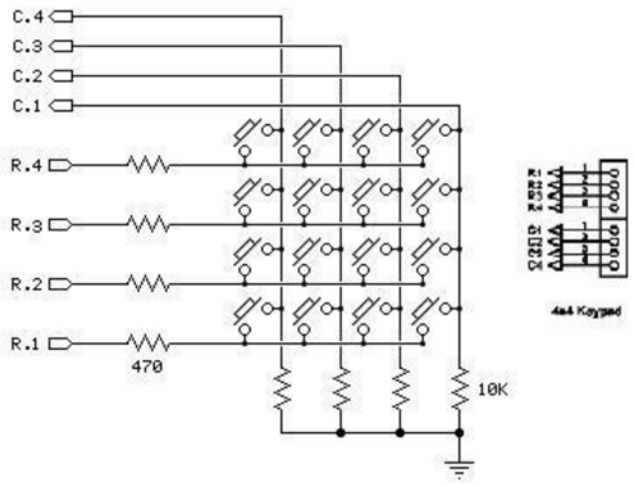
این فایل سرآیند حاوی دستورات آماده برای کار با LCD در محیط نرم افزار CodeVision است. بنا به نسخه مورد استفاده Codevision، ممکن است تفاوت‌های جزیی در دستورات و آرگومان‌های این فایل وجود داشته باشد. لذا برای اطلاع از این تغییرات به راهنمای کاربری نرم افزار مراجعه نمایید. تعدادی از دستورات متداول در جدول 3-2 شرح داده شده‌اند.

`void lcd_init(unsigned char lcd_columns)`	برای تنظیمات و پیکربندی اولیه LCD به کار می رود که در آن تعداد کاراکتر قابل نمایش در هر خط به عنوان ورودی دریافت می شود.
`void lcd_clear(void)`	محتوای نمایش داده شده بر روی LCD را پاک می‌کند.
`void _lcd_ready(void)`	اجرای ادامه کد را منتظر آماده شدن LCD نگه می‌دارد.
`void lcd_putchar(char c)`	کاراکتری که به عنوان ورودی دریافت شده را در موقعیت جاری مکان‌نما بر روی LCD نمایش می‌دهد.
`void lcd_puts(char *str)`	رشته ورودی را در موقعیت جاری مکان‌نما بر روی LCD نمایش می‌نماید.
`void lcd_putsf(char flash *str)`	این تابع نیز رشته ورودی را بر روی LCD نمایش می‌دهد. اما توجه کنید که وقتی از این تابع استفاده می‌کنیم که رشته مورد نظر درون flash قرار داشته باشد.
`void lcd_gotoxy(unsigned char x, unsigned char y)`	مکان‌نما را به مختصات تعیین شده با x و y منتقل می‌کند.

جدول 3-2: شرح برخی دستورات موجود در کتابخانه alcd.h

3.3 آشنایی با صفحه‌کلید ماتریسی

صفحه کلید یکی از متداول‌ترین ادوات ورودی برای دریافت داده‌ها از سوی کاربر می‌باشد. در پکیج آموزشی یک عدد صفحه‌کلید ماتریسی شامل 16 عدد کلید فشاری که چیدمان آن‌ها به صورت 4x4 (4 ردیف و 4 ستون) می‌باشد قرار داده شده است. شماتیک مربوط به مدار keypad در شکل 3-4 مشاهده می‌شود.



شکل 3-4: مدار صفحه کلید ماتریسی استفاده شده در بورد آموزشی

برای کار با keypad پایه‌های آن به یکی از درگاه‌های ریزپردازنده متصل می‌شود به این صورت که سطرهای R1 تا R4 به صورت خروجی و ستون‌های C1 تا C4 به صورت ورودی تعریف می‌گردند. سپس مقدار متناظر هر کلید به صورت یک آرایه در ریزپردازنده ذخیره می‌گردد. به عنوان مثال، در برنامه 3-2، ماتریس data_key به ازای هر یک از کلیدهای keypad یک مقدار متناظر را در برگرفته است.

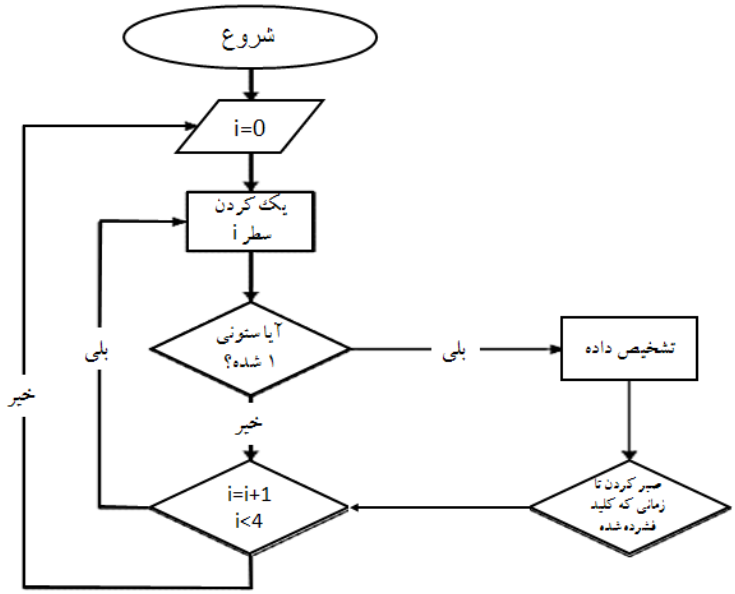
برنامه 3-2

```
char data_key[]={
'7','8','9','a',
'4','5','6','b',
'1','2','3','c',
'*','0','#','d'};
```

در ادامه مانند شکل 3-5 سطرها به ترتیب 1 گردیده و مقدار ستون‌ها خوانده می‌شود. هر جا که مقدار ستونی 1 شده باشد، یعنی کلید مربوط به آن سطر و ستون فشرده شده است. فرض کنید کلید 6 فشرده شده باشد، در این صورت با یک شدن پایه نظیر سطر دوم، مقدار پایه متناظر با ستون سوم نیز مقدار 1 و سایر ستون‌ها مقدار 0 را برمی‌گردانند. با پیدا شدن سطر و ستون، می‌توان از طریق ماتریس ذخیره شده در ریزپردازنده، به محتوای کلید دست پیدا کرد.

```
data_key[(2 - 1) * 4 + (3 - 1)] = '6'
```

این روش کار با صفحه کلید را روش سرکشی می‌نامند که یک نمونه کد برای تعیین کلید فشرده شده به این طریق در برنامه 3-3 نشان داده شده است.



شکل 3-5: الگوریتم کار با صفحه‌کلید ماتریسی به روش سرکشی

برنامه 3-3

```
char keypad2(void)
{
    char key=100;
    for (r=0;r<4;r++)
    {
    PORTB=row[r]; //row= 0x10,0x20,0x40,0x80

    c=20;
    delay_ms(10);
    if (PINB.0==1) c=0;
    if (PINB.1==1) c=1;
    if (PINB.2==1) c=2;
    if (PINB.3==1) c=3;


        if (!(c==20)){
            key=(r*4)+c;
            PORTB=0xf0;
            while (PINB.0==1) {}
            while (PINB.1==1) {}
            while (PINB.2==1) {}
            while (PINB.3==1) {}
        }
     PORTB=0xf0;
    }
    return key;

}
```

3.4 وقفه‌ها در ریزپردازنده

رسیدگی به رویدادهای مربوط به ادوات جانبی و نیز رویدادهای خارجی در ریزپردازنده‌ها به یکی از دو روش زیر قابل انجام است:

- روش سرکشی (Polling): در این روش که در بخش قبل نیز مورد استفاده قرار گرفت، برنامه‌نویس پردازنده را طوری برنامه‌ریزی می‌کند که وقوع رویداد مورد نظر با فواصل زمانی مشخص و پی‌درپی مورد بررسی قرار گیرد و در صورت وقوع رویداد، به آن پاسخ می‌دهد. در این روش زمان پردازنده در خیلی از موارد برای بررسی وقوع اتفاق احتمالی، به هدر می‌رود.
- روش استفاده از وقفه (Interrupt): وقفه امکانی در ریزپردازنده است که باعث می‌شود هسته پردازشی در قبال ایجاد یک رویداد لحظه‌ای (که معمولاً زمان آن قابل پیش‌بینی نیست) عملیات خاصی را در قالب یک زیربرنامه مشخص، انجام دهد. در روش استفاده از وقفه، برنامه اصلی در حالت عادی خود اجرا می‌شود، اما به محض وقوع رویداد مورد نظر، پردازنده دستور جاری برنامه را به انتها رسانده و اجرای بقیه‌ی برنامه را متوقف می‌نماید. سپس به صورت سخت‌افزاری به ابتدای زیربرنامه‌ی وقفه[1] پرش کرده و پس از اجرای آن، ادامه‌ی برنامه اصلی اجرا می‌گردد.

در استفاده از وقفه می‌توان درخواست وقفه از وسایل جانبی مختلف را اولویت بندی نمود و بر اساس این اواولیت‌ها و با حتی به صورت کلی وقوع برخی وقفه‌ها را نادیده گرفت. در ادامه به تعدادی از مزایای استفاده از وقفه اشاره شده است.

1. رسیدگی بی‌درنگ به درخواست وقفه (در صورت فعال نبودن یک درخواست وقفه با اولویت بالاتر)
2. عدم اتلاف زمان پردازنده در زمان‌هایی که درخواستی برای پردازش موجود نیست.

3.4.1 راه اندازی وقفه در AVR

برای فعال‌سازی وقفه‌ها در ریزپردازنده‌های AVR باید بیت هفتم از ثبات SREG (فعال ساز عمومی) را فعال کرد. پس از فعال کردن این بیت می‌توان هر کدام از وقفه‌های موجود در AVR را از طریق ثبات‌های مربوطه فعال کرد. در زبان C با نوشتن عبارت #asm(“sei”) این بیت فعال می‌شود.

جدول 3-3 وقفه های ریزپردازنده Atmega16/32 لیست گردیده است.

[1] Interrupt subroutine

جدول 3-3 : وقفه‌های ریزپردازنده Atmega16/32

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

همان‌طور که در جدول فوق مشاهده می‌شود سه مورد از وقفه‌ها، تحت عنوانExternal interrupt بوده (وقفه-های0INT، INT1، INT2) و به آن‌ها وقفه‌های خارجی گفته می‌شود. از وقفه‌های خارجی معمولاً برای تشخیص پالس بر روی پایه‌های تراشه استفاده می‌شود. این پالس می‌تواند حاوی اطلاعات خاصی مانند اطلاعات دریافتی از یک حسگر یا IC باشد که به یکی از پایه‌های PB2 ، PD2 و یا PD3 متصل است. غیر از این سه وقفه و البته وقفه Timer/Counter capture interrupt، سایر وقفه‌ها داخلی هستند. در این جلسه وقفه‌های خارجی مورد بررسی قرار می‌گیرند و سایر وقفه‌ها در جلسات بعدی و همزمان با سخت‌افزار جانبی مربوطه تشریح خواهند شد.

برای زیربرنامه‌ی هر وقفه، فضای مشخصی در نظر گرفته شده است که کدهای مورد نظر باید دراین فضا نوشته شوند. این فضاها تحت عنوان شماره بردار وقفه در جدول 3-3 مشخص شده‌اند. البته در فایل سرایند mega16.h، برای هر کدام از این شماره‌ها نام‌های معادلی در نظر گرفته شده که استفاده از وقفه‌ها را آسان‌تر می‌نماید. مثلاً برای استفاده

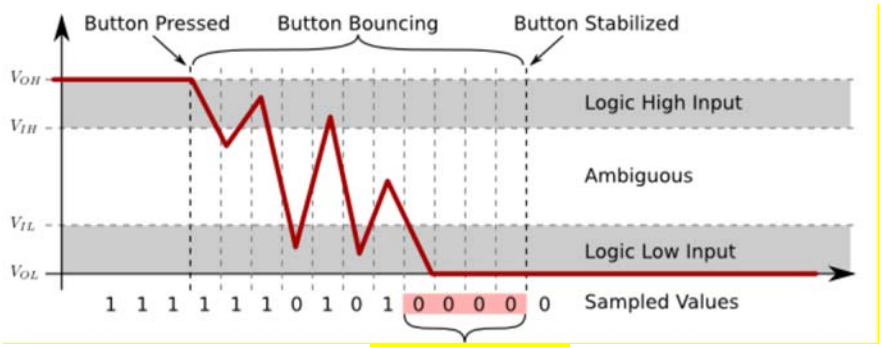
از وقفه خارجی 1 به جای نوشتن شماره ،3 می‌توان عبارت EXT_INT1 را به کار برد. این نحوه استفاده در برنامه 4-3 نشان داده شده است. در این مثال زیربرنامه‌ی ext_int1_isr در فضای مربوط به وقفه EXT_INT1 ذخیره می‌شود.

برنامه 4-3

```
Interrupt [شماره بردار وقفه] void نام تابع (void) {}
interrupt [EXT_INT1] void ext_int1_isr(void){…}
```

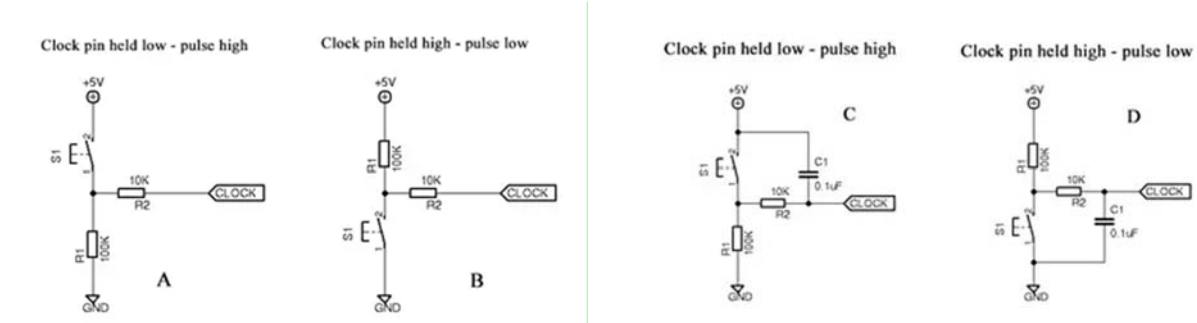
روی بورد موجود در آزمایشگاه تعدادی کلید Push-Botton وجود دارد که می توان از آن به عنوان منبع ایجاد وقفه خارجی استفاده نمود. Push-Button کلیدی است که در حالت عادی قطع است. مادامی که توسط کاربر فشار داده شود، وصل خواهد بود و به محض رها شدن، دوباره قطع خواهد شد.

یکی از مشکلات رایج در Push-Buttonها پدیده Bouncing است. این پدیده در اثر لرزش اتصالات مکانیکی کلیدها در هنگام قطع و وصل شدن، رخ می‌دهد. چنین لرزش‌هایی موجب می‌شود پیش از رسیدن کلید به حالت دائمی، چند بار صفر و یک شود. زمان رسیدن کلید به حالت دائمی معمولاً بین 10 تا 20 میلی‌ثانیه است. به عنوان مثال در شکل زیر بعد از فشرده شدن Push-Button تا زمانی که به حالت ثابت صفر برسد چند صفر و یک دیگر تولید شده و این باعث ایجاد خطا در خروجی می‌شود.



شکل 3-6: اثر bouncing

یکی از روش‌های حل این مشکل استفاده از مدارهای RC مانند شکل 3-7 است تا نویز به وجود آمده را فیلتر نماید.

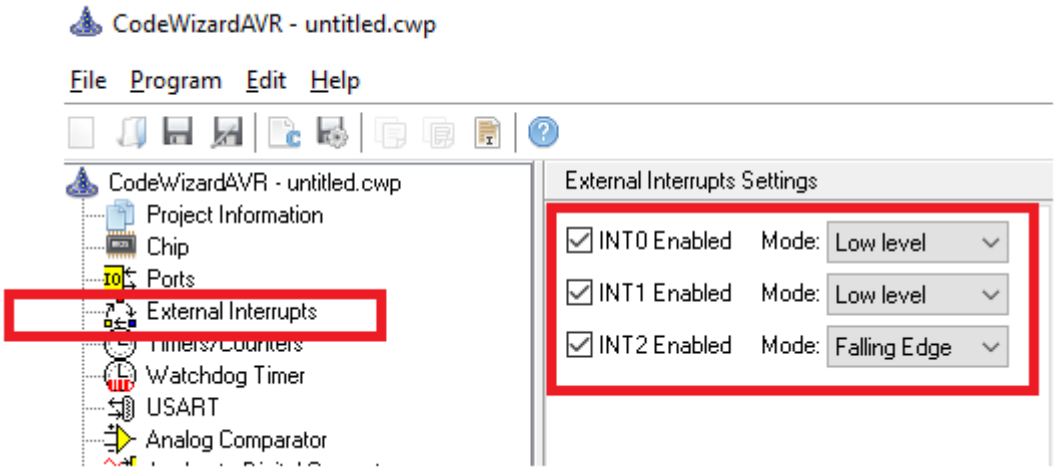


شکل 3-7: رفع پدیده bouncing با استفاده از مدار RC

3.5 پردازش صفحه کلید با وقفه

همان طور که در بخش قبلی به اهمیت وقفه‌ها در پیشگیری از اتلاف بیهوده وقت پردازنده اشاره شد، برای پردازش صفحه کلید نیز می‌توان از وقفه‌های خارجی استفاده کرد. بدین منظور لازم است از سخت‌افزار جانبی شامل دروازه‌های منطقی استفاده کرد تا با فشردن هر کلید، وقفه‌ای خارجی نیز رخ دهد و پروسه پردازش صفحه کلید در روتین وقفه‌ی مربوطه ادامه یابد. در این حالت لازم است مقدار مناسب در درگاه متصل به صفحه کلید نوشته شود و بعد از اتمام وقفه نیز به مقدار مورد نظر تنظیم گردد این مقدار بنا به سخت افزار طراحی شده برابر 0xF0 خواهد بود.

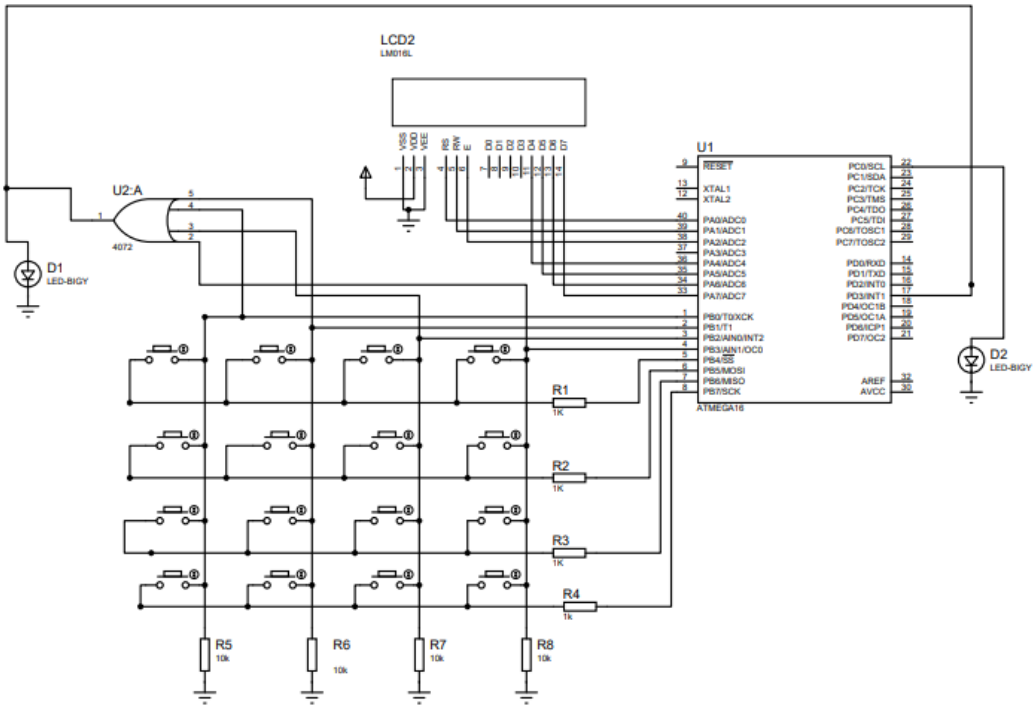
برای فعال کردن وقفه‌های خارجی می‌توان از قابلیت‌های CodeWizard در کد ویژن مانند شکل 3-8 استفاده کرد و حساسیت به لبه یا سطح را برای رخداد وقفه را تعیین نمود.



شکل 3-8: تنظیمات وقفه‌های خارجی در CodeWizard

3.6 برنامه‌های اجرایی مبحث LCD و صفحه کلید

برای سخت‌افزار نشان داده شده در شکل زیر، برنامه‌های خواسته شده را بنویسید.



1. نام خانوادگی در خط اول و شماره دانشجویی در خط دوم LCD کاراکتری نمایش داده شود.
2. عبارت زیر به صورت روان روی lcd نمایش داده شود. سرعت حرکت باید به گونه ای باشد که قابل دیدن و خواندن باشد.

"Welcome to the Microprocessor Laboratory at Isfahan University of Technology."

3. زیر برنامه ای بنویسید که صفحه کلید زیر را به روش سرکشی برای تشخیص کلید فشرده شده اسکن و مقدار نظیر کلید فشرده شده را روی lcd نمایش دهید.

0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

4. صفحه کلید فوق را با استفاده از وقفه خارجی اسکن و مقدار نظیر کلید فشرده شده را روی lcd نمایش دهید. (در این بند، زیربرنامه نوشته شده در بند 3 در زیر برنامه وقفه خارجی فراخوانی می‌گردد.)
5. داده‌های اولیه یک سیستم شامل سرعت، زمان، وزن و دما از طریق صفحه کلید دریافت می‌شود. بدین منظور ابتدا هر یک از پیام‌های ذیل روی LCD نمایش داده شده و مقدار اولیه متناظر از طریق صفحه

کلید دریافت و به جای عبارت ؟؟ نمایش داده می‌شود. اگر عدد دریافتی خارج از محدوده باشد، در این محل عبارت EE نمایش داده می‌شود و منتظر اصلاح عدد می‌ماند. اما چنانچه عدد در بازه مورد نظر باشد پیام بعدی نمایش داده خواهد شد و نهایتاً پیام پایان فراخوانی چاپ خواهد شد.

```
Speed:??(0-50r)
Time:??(0-99s)
W:??(0-99Kg)
Temp:??(20-80C)
End
```

6. آیا می‌توان عمل خواندن و نوشتن روی یک درگاه را بلافاصله پشت سر هم انجام داد؟ برای پاسخ دادن به این سوال به قسمت I/O Ports در دیتاشیت ریزپردازنده مراجعه نمایید.