



دانشگاه صنعتی اصفهان
دانشکده مهندسی برق و کامپیوتر

دستورکار آزمایشگاه ریزپردازنده دستورکار آزمایشگاه طراحی سیستم‌های دیجیتال 2

(مبتنی بر ریزپردازنده ATMEGA16/32)

تهیه کننده:

زهره محمدزاده

بررسی کننده:

دکتر امیر خورسندی

شهریور 1401

1 جلسه اول

درگاه‌های ورودی و خروجی

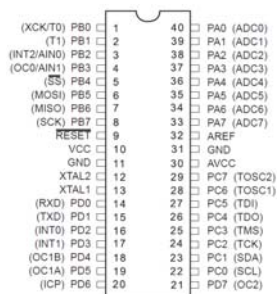
1.1 هدف

در این جلسه موارد ذیل بررسی می‌گردند.

- آشنایی با درگاه‌های ریزپردازنده و ثبات‌های مربوط به آن
- آشنایی با فایل‌های سرآیند
- آشنایی با واحدهای ورودی/خروجی اولیه: کلیدهای فشاری و کشویی، LED و نمایشگر 7-Segment
- ایجاد پروژه و پروگرام کردن ریزپردازنده
- زیربرنامه نویسی

1.2 معرفی درگاه‌های ورودی و خروجی Atmega16/32

نمای ریزپردازنده Atmega16/32 با بسته‌بندی PDIP در شکل 1-1 نشان داده شده است. این ریزپردازنده دارای 40 پایه است که 32 پایه‌ی آن مربوط به 4 درگاه 8 بیتی ورودی و خروجی می‌باشد.



شکل 1-1: پایه‌های ریزپردازنده Atmega16/32 با بسته‌بندی PDIP

به ازای هر درگاه سه ثبات^۱ به شرح زیر وجود دارد:

ثبات داده خروجی PORTx^۲

^۱ Register

^۲ PORTX data register

ثبات جهت داده DDR_x ¹

ثبات داده ورودی PIN_x ²

در شکل 1-2 ساختار این ثبات ها برای درگاه A نشان داده شده است و سایر درگاه ها نیز ساختاری مشابه دارند.

PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
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	

PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

شکل 1-2: ثبات های درگاه A

در مورد ثبات های درگاه های ورودی و خروجی نکات زیر حایز اهمیت است:

- 1- ثبات PIN_x تنها قابل خواندن است ولی ثبات های DDR_x و $PORT_x$ هم قابل خواندن و هم قابل نوشتن هستند.
- 2- اگر بیت n ام از ثبات DDR_x یک درگاه برابر صفر باشد پایه n ام از آن درگاه به صورت ورودی و اگر آن بیت برابر 1 باشد، پایه متناظر به صورت خروجی خواهد بود.
- 3- برای خواندن وضعیت پایه ای که به صورت ورودی تعریف شده است، بیت متناظر از ثبات PIN_x خوانده می شود و برای تعیین وضعیت پایه هایی که به صورت خروجی تعریف شده اند، مقدار آن ها در بیت متناظر از ثبات $PORT_x$ نوشته می شود.

¹ PORTX data direction register

² PORTX input pins address

به عنوان نمونه در برنامه 1-1، چهار بیت کم ارزش درگاه B به عنوان خروجی و چهار بیت پر ارزش آن به عنوان ورودی تعریف شده است.

برنامه 1-1 `DDRB = 0x0F;`

همان گونه که در برنامه 2-1 دیده می شود، دسترسی به بیت های هر کدام از ثبات ها به تنهایی نیز امکان پذیر است.

برنامه 2-1 `DDRA.3 = 1; //set bit 3 of port A as output
PORTA.3 = 1; //make bit 3 of port A high`

ذکر این نکته لازم است که با توجه به محدودیت تعداد پایه های ریزپردازنده و به منظور ایجاد امکان استفاده از سایر قابلیت های Atmega16/32، برخی از پایه های درگاه ها به کمک یک مالتی پلکسر به سایر واحدهای عملکردی ریزپردازنده نیز متصل شده است. لذا این پایه ها علاوه بر عملکرد یک ورودی/خروجی ساده، کاربردهای دیگری هم مانند مبدل آنالوگ به دیجیتال، تایمر، ارتباط سریال و ... دارند که در سایر جلسات بررسی خواهند شد.

1.3 معرفی فایل سرآیند (Header File)

فایل های سرآیند، شامل برخی زیربرنامه ها یا تعاریف اولیه هستند که جهت سهولت در برنامه نویسی به کار می روند و در پوشه INC در محل نصب CodeVision قابل دسترسی هستند. برای فراخوانی هر یک از این فایل ها، از دستور `#include` در ابتدای برنامه و قبل از تابع `main` استفاده می شود. در برنامه 3-1 دو روش برای این منظور معرفی شده است.

برنامه 3-1 `#include <file_name.h>
#include "file_name.h"`

در روش اول، کامپایلر فایل سرآیند را در زیر شاخه `inc` در محل نصب برنامه جستجو می کند. اما در حالت دوم، ابتدا پوشه جاری برنامه را جستجو می کند و سپس به مسیر `/inc` رجوع می کند.

1.3.1 فایل سرآیند mega16.h

ریزپردازنده‌های AVR دارای 32 ثبات هستند که ارسال و دریافت داده، پیکربندی ریزپردازنده و تنظیم امکانات داخلی آن از طریق این ثبات‌ها انجام می‌شود. کار با این ثبات‌ها از طریق دسترسی به حافظه ریزپردازنده میسر خواهد بود که آدرس‌ها و اشاره‌گرهای مربوطه را می‌توان با مطالعه Datasheet مربوطه به دست آورد. در برنامه 4-1 تنظیم درگاه D به صورت خروجی با استفاده از آدرس انجام شده است.

```
برنامه 4-1      void main(void)
                {
                *(unsigned char *) 0x11=0b11111111;
                while(1) { };
                }
```

اما برای سهولت کار، هر ریزپردازنده فایل سرآیند مخصوص به خود را دارد که در آن برای تمامی ثبات‌های ریزپردازنده، یک اشاره‌گر براساس نام و آدرس آن‌ها در حافظه تعریف شده است. با include کردن این فایل سرآیند می‌توان به جای آدرس ثبات، از اسم آن ثبات استفاده کرد.

در برنامه 5-1 تنظیم درگاه D به صورت خروجی با استفاده از نام DDRD که در سرآیند فایل ریزپردازنده تعریف شده، انجام شده است.

```
برنامه 5-1      #include <mega16.h>
                void main(void)
                {
                DDRD=0xFF;
                while(1) { };
                }
```

1.3.2 معرفی فایل سرآیند delay.h

این فایل سرآیند شامل دو تابع void delay_us(unsigned int n) و void delay_ms(unsigned int n) است که به کمک آن‌ها می‌توان در برنامه تأخیر ایجاد کرد. لازم به ذکر است در هنگام استفاده از این توابع اگر وقفه‌ای فعال باشد ممکن است تأخیر بیش از اندازه طولانی گردد. لذا در مواردی که میزان تأخیر اهمیت داشته باشد استفاده از این توابع توصیه نمی‌گردد.

1.4 معرفی مقاومت داخلی درگاه‌ها

هنگامی که یک پایه را به صورت ورودی تنظیم می‌کنیم، وظیفه تعیین مقدار آن بر عهده یک مدار راه‌انداز خارجی خواهد بود. به جهت ساده شدن این مدار و نیز کاهش هزینه و انرژی مصرفی عموماً این مدار صرفاً در زمانی که پایه باید مقدار صفر بگیرد آن را به ولتاژ زمین متصل می‌نماید و در غیر این صورت مقداری را به پایه اعمال نمی‌کند. در این حالت ممکن است در اثر تغییرات ولتاژ، تغییرات جریان، نویز و... حالت آن پایه به صورت ناخواسته تغییر کند که با توجه به سرعت بالای ریزپردازنده این تغییر ممکن است به اشتباه برای ریزپردازنده به معنی یک یا صفر شدن پایه تلقی شود و در اجرای برنامه اخلال ایجاد کند. لذا برای از بین بردن این تأثیرات ناخواسته باید با استفاده از مقاومت‌های بالاکش و یا پایین کش، پایه‌ها از حالت شناور خارج شده و به یکی از خطوط تغذیه متصل شوند. اگرچه برای این کار می‌توان از مقاومت خارجی استفاده نمود، اما در داخل تراشه‌های AVR نیز یک سری مقاومت‌های بالاکش جهت استفاده در درگاه‌های ورودی تعبیه شده‌اند.

برای فعال نمودن مقاومت‌های داخلی کفایت در رجیستر PORTx بیت مورد نظر برابر یک گردد.

1.5 معرفی واحدهای ورودی/خروجی اولیه پکیج آموزشی

1.5.1 معرفی کلیدهای کشویی

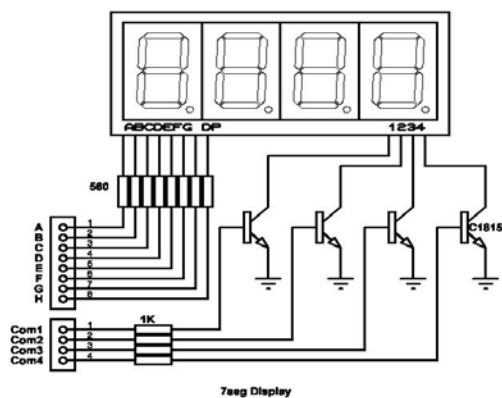
در پکیج آموزشی 8 عدد کلید کشویی به منظور تولید صفر و یک دائمی در بلوکی تحت عنوان Data Latch Switch قرار داده شده است. این کلیدها وظیفه تولید صفر و یک منطقی را بر عهده دارند. در ساختار این برد آموزشی، از مقاومت‌های بالاکش و پایین کش تعبیه شده روی برد برای جلوگیری از به وجود آمدن حالت شناور بر روی پایه‌های ریزپردازنده استفاده شده است.

1.5.2 معرفی Push Button

از کلیدهای Push Button به منظور تولید صفر و یک لحظه‌ای استفاده می‌شود. در برد آموزشی، 8 عدد Push Button قرار دارد که از آن‌ها برای تولید صفر و یک منطقی استفاده می‌شود. با فشار دادن این کلیدها، یک سیگنال صفر یا یک در پایه متناظر ایجاد می‌کند و بلافاصله پس از رها کردن کلید، خروجی به سطح یک یا صفر باز می‌گردد.

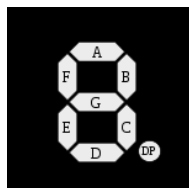
1.5.3 معرفی نمایشگر 7-Segment

این نمایشگر، برای نمایش اعداد و بعضی از حروف لاتین به کار می‌رود. مطابق شکل 1-3 بر روی برد آموزشی یک نمایشگر چهار رقمی وجود دارد که در آن برای هر 7-Segment یک پایه‌ی Enable تعبیه شده است. علاوه بر آن 8 پایه‌ی داده به نام‌های A تا H نیز وجود دارد که برای هر چهار 7-Segment مشترک هستند.



شکل 1-3 نحوه اتصال پایه‌های 7-Segment

یک کردن هر کدام از پایه‌های داده، موجب روشن شدن Segment متناظر با آن خواهد شد. بنابراین برای نمایش یک عدد خاص، بر روی هر 7-Segment باید داده‌ی مناسب را بر روی پایه‌های داده ارسال کرده و پایه Enable آن را یک کرد. چنانچه این کار به صورت متناوب و با فرکانس مناسب انجام شود، می‌توان به صورت همزمان اعداد چهار رقمی دلخواه را روی چهار 7-Segment رؤیت کرد. با توجه به شکل 1-4 برای نمایش عدد '1' باید به Segment‌های B و C مقدار 1 منطقی اعمال کرد.

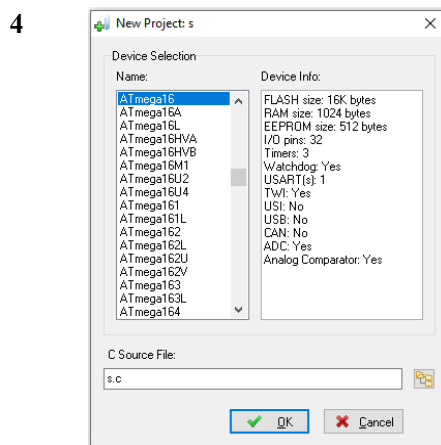
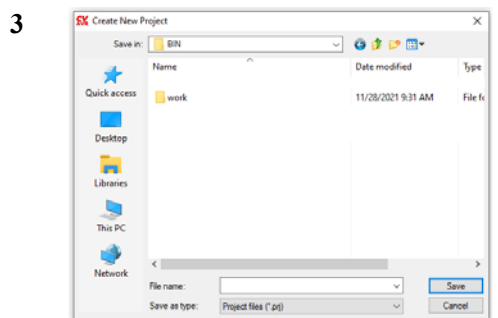
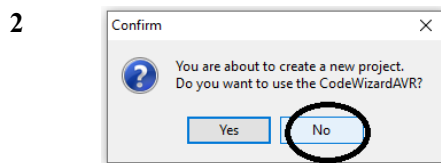
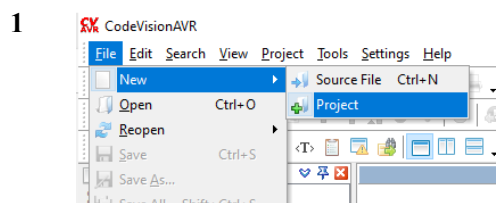


شکل 1-4 نامگذاری بخش‌های 7-Segment

1.6 ایجاد پروژه در محیط codevision

1.6.1 ایجاد پروژه بدون استفاده از CodeWizard

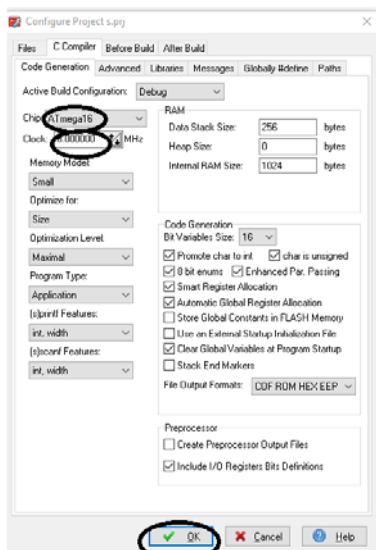
برای ایجاد پروژه بدون استفاده از CodeWizard باید مراحل نشان داده شده در شکل 1-5 اجرا گردد.



فایل‌ها در مسیر نصب در پوشه BIN قابل ذخیره است. بهتر است برای هر پروژه یک پوشه جدید ایجاد نموده و تمام فایل‌ها را به یک نام ذخیره نمایید.

ریزپردازنده مورد نظر را از لیست انتخاب نمایید.

5

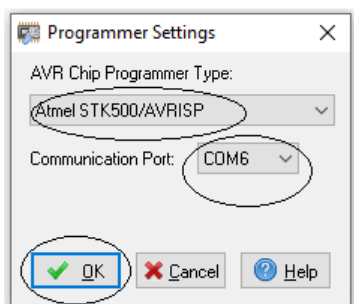


فرکانس کاری ریزپردازنده را تعیین نمایید.

7 IeVisionAVR - C:\cvavr\BIN\work14001\az2\s.prj

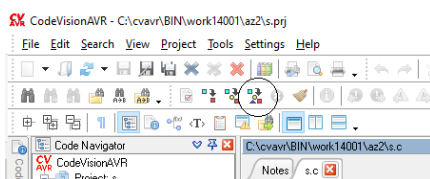


8



درگاهی که پروگرامر به آن متصل شده است را انتخاب نمایید.

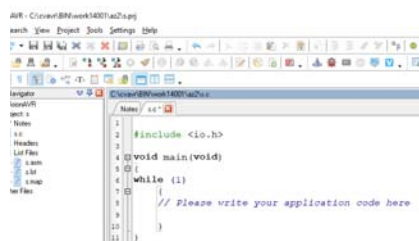
9



پروژه را کامپایل نمایید. بدین وسیله فایل اجرایی با قالب hex که قابل بارگذاری در ریزپردازنده است ایجاد می‌گردد.

شکل 1-5: مراحل ایجاد پروژه بدون استفاده از coddewizard

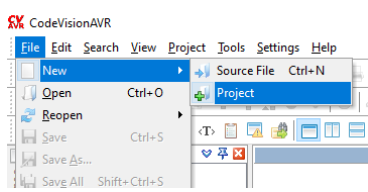
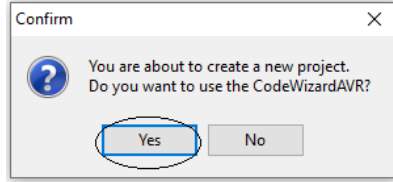
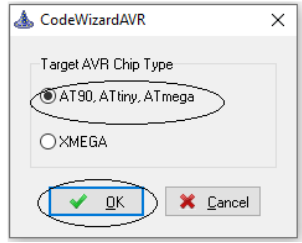
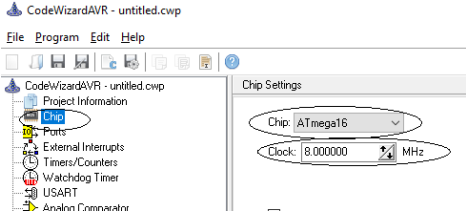
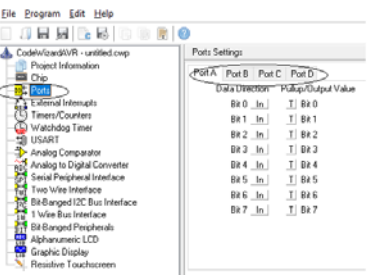
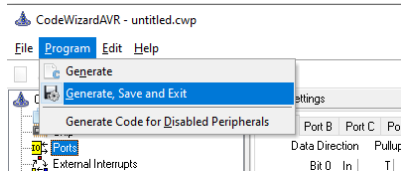
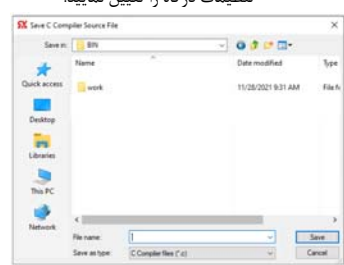
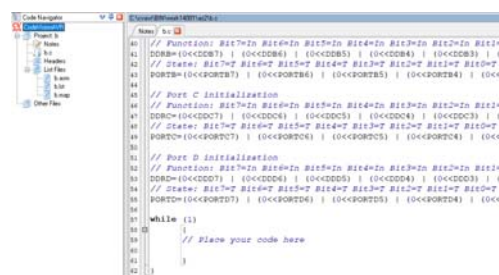
6



کدهای فوق به صورت پیش‌فرض تولید می‌شوند. فایل io.h حاوی فایل‌های سرآیند مربوط به تمام ریزپردازنده‌های معرفی شده در Codevision است. می‌توان آن را با mega16.h جایگزین نمود. در ادامه کدهای مورد نظر به برنامه اضافه می‌شود.

1.6.2 ایجاد پروژه با استفاده از CodeWizard

مراحل ایجاد پروژه با استفاده از قابلیت‌های codwizard در شکل 1-6 نشان داده شده است.

1. CodeVisionAVR

2. Confirm dialog box: "You are about to create a new project. Do you want to use the CodeWizardAVR?" with 'Yes' button highlighted.

3. CodeWizardAVR dialog box: "Target AVR Chip Type" with "AT90, ATtiny, ATmega" selected and "OK" button highlighted.

4. CodeWizardAVR - untitled.cwp window: "Chip Settings" with "Chip: ATmega16" and "Clock: 8.000000 MHz" highlighted.

5. CodeWizardAVR - untitled.cwp window: "Ports Settings" with "Port A", "Port B", "Port C", and "Port D" highlighted.

6. CodeWizardAVR - untitled.cwp window: "Generate" button highlighted.

7. Save C Compiler Source File dialog box: "File name" field highlighted.

8. Code Navigator window: "Generated Source File" showing the generated C code.


فایل‌های هر پروژه را داخل یک پوشه و با نام یکسان برای تمام فایل‌ها ذخیره نمایید.

کدهای فوق به صورت پیش‌فرض تهیه می‌شود. در ادامه کدهای مورد نظر به برنامه اضافه می‌شود.

شکل 1-6: مراحل ایجاد پروژه با استفاده از CodeWizard

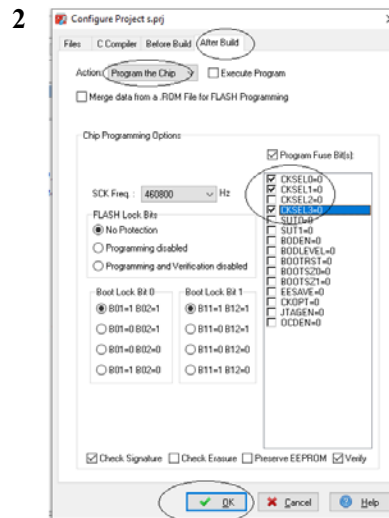
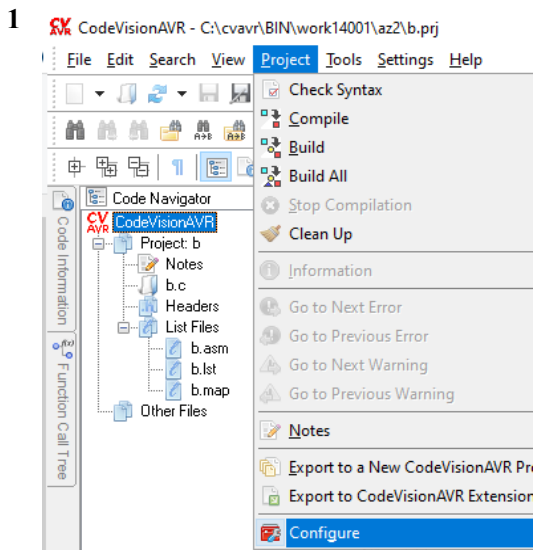
1.7 نحوه پروگرام نمودن ریزپردازنده

برای پروگرام ریزپردازنده ابتدا مطابق مراحل قبل، مدل پروگرامر انتخاب می‌شود و یکی از روش‌های زیر اجرا می‌گردد.

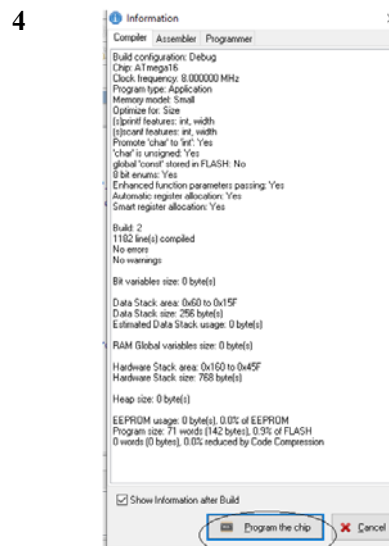
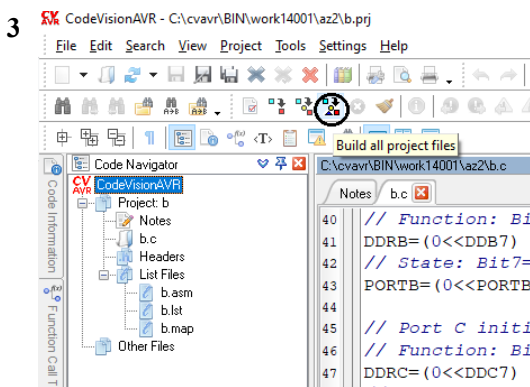
در روش نخست که مراحل آن در شکل 1-7 نشان داده شده، پس از کامپایل پروژه گزینه‌ی پروگرام تراشه فعال می‌شود و بدین وسیله روند پروگرام نمودن ساده‌تر می‌گردد.

در پردازنده Atmega16/32 تعدادی فیوز بیت (Fuse bits) وجود دارد که قسمتی از حافظه دائمی (رزرو شده) هستند که تنظیمات خاصی را در خود ذخیره می‌کنند. فیوز بیت‌ها تنها در زمان پروگرام کردن ریزپردازنده قابل تنظیم هستند و پس از آن امکان تغییر آن توسط CPU وجود نخواهد داشت. تنظیمات اعمالی بر روی فیوز بیت‌ها تا زمانی که عمل پروگرام کردن دوباره انجام شود، حفظ خواهد شد و قطع تغذیه نیز بر روی آن تأثیری نخواهد گذاشت. تنظیماتی که توسط مقداردهی فیوز بیت‌ها قابل اجراست تماماً از نوع سخت‌افزاری هستند.

به عنوان مثال فعال کردن قفل برنامه که از خوانده شدن اطلاعات ریزپردازنده به صورت خارجی جلوگیری می‌کند، انتخاب نوع اسیلاتور ریزپردازنده با توجه به محدوده فرکانس تولید شده‌ی آن و تعیین نوع پروگرام کردن ریزپردازنده از جمله تنظیماتی است که توسط فیوز بیت‌ها قابل انجام است و از طریق نرم‌افزار پروگرام کردن ریزپردازنده انجام می‌شود.

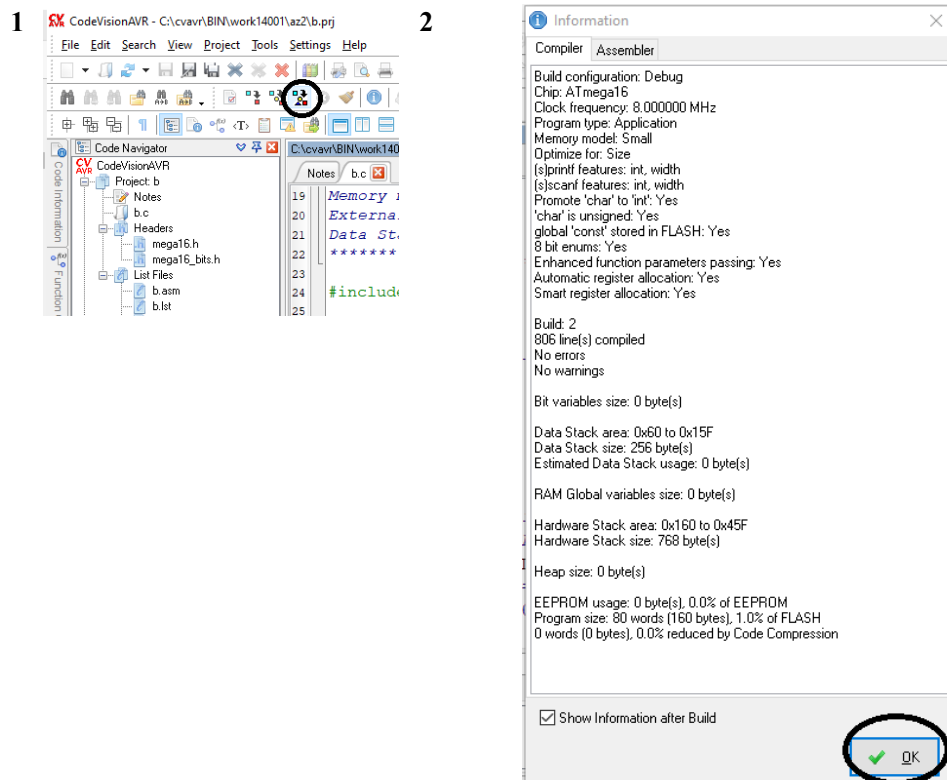


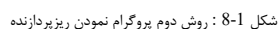
فیوز بیت‌ها با توجه به فرکانس کاری ریزپردازنده تعیین می‌گردند. تنظیم نامناسب آن گاهی سبب عملکرد نادرست ریزپردازنده می‌گردد.



شکل 7-1: روش اول پروگرام کردن تراشه

مراحل روش دوم در شکل 1-8 نشان داده شده است. با استفاده از این روش می‌توان هنگامی که کدهای ایجاد شده در Codevision موجود نیست و فقط کد Hex وجود دارد، محتوای این فایل را بر روی ریزپردازنده بارگذاری نمود. بدین منظور از منوی file می‌توان فایل hex مورد نظر را انتخاب نمود.





در ادامه چندین برنامه کاربردی برای درگاه‌های ورودی و خروجی قابل مشاهده است. در برنامه 1-6 یکی از درگاه‌ها بدون استفاده از Code Wizard مقدار دهی شده است.

```
برنامه 6-1                                     #include <mega16.h>
void main(void)
{
    DDRE=0xFF;
    PORTB=0b01010101;
    PORTB.5=1;
    while(1);
}
```

در برنامه 7-1 داده از کلیدهای کشویی پکیج که به درگاه B متصل خوانده شده و بر روی هشت LED که به درگاه D متصل هستند نمایش داده می‌شود.

```
برنامه 7-1
#include <mega16.h>
char number;
void main(void)
{
    DDRB = 0x00;
    DDRD = 0xFF;
    while(1)
    {
        number = PINB;
        PORTD = number;
    }
}
```

در برنامه 8-1 یک شمارنده طراحی شده است که در صورت یک بودن PINB.0 که به یک کلید کشویی متصل است به مقدار آن اضافه می‌شود. با استفاده از دستور `char count=0x00` یک متغیر 8 بیتی تعریف شده است که مقدار اولیه آن صفر بوده و در ادامه مقدار شمارنده را در هر لحظه در خود نگه می‌دارد. هم چنین مقدار آن همواره بر روی LEDهای متصل به درگاه D نمایش داده می‌شود.

```
برنامه 8-1
#include <mega16.h>
void main(void)
{
    char count = 0x00;
    DDRD = 0xFF;
    PORTD = 0x00;
    DDRB.0 = 0;
    while(1)
    {
        if (PINB.0)
        {
            count = count + 1;
            PORTD = count;
        }
    }
}
```

با ریختن این برنامه بر روی ریزپردازنده به نظر می‌رسد LEDها همگی با نور کمی روشن هستند و شمارنده به درستی عمل نمی‌کند. علت آن این است که سرعت کار ریزپردازنده بسیار زیاد است و شمارنده با سرعت بالایی LEDها را روشن و خاموش می‌کند. برای حل این مشکل باید شمارش با فاصله زمانی مناسبی انجام شود. برای ایجاد تأخیر می‌توان از توابع وجود در `delay.h` استفاده کرد.

در برنامه 9-1، مقدار متغیر `number` روی 7-Segmentهای متصل به درگاه D که Enable آنها از طریق تغذیه تامین گردیده نمایش داده می‌شود.

```
برنامه 9-1
#include <mega16.h>
#include <delay.h>
```

```

flash unsigned char digit[]={0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D,
0x07, 0x7F, 0x6F};
int number = 0;
void main(void)
{
    DDRD = 0xFF;
    number = 5;
    while(1)
    {
        PORTD = digit[number];
        delay_ms(1000);
    }
}

```

1.9 زیربرنامه نویسی و فراخوانی آن در تابع اصلی

در برنامه 10-1 نمونه‌ای از زیربرنامه نویسی نشان داده شده است.

برنامه 10-1

```

#include <mega16.h>
Void sub_routine(void)
{
    DDRB=0xFF;
    PORTB=0b01010101;
    PORTB.5=1;
}
void main(void)
{
    sub_routine();
    while(1);
}

```


1.10 سوالات تشریحی درگاه‌های ورودی و خروجی

(1) به سوالات زیر پاسخ دهید:

الف- ساختار مقاومت بالاکش (pull-up) و کاربرد آن را مختصراً شرح دهید.

ب - انواع روش‌های اتصال کلید به ریزپردازنده را نام ببرید.

ج - انواع روش‌های برنامه‌ریزی ریزپردازنده‌های AVR را نام ببرید.

د - ریزپردازنده AVR چند نوع حافظه دارد؟ نحوه استفاده از آن‌ها چگونه است؟ تفاوت آن‌ها در چیست؟

(2) نحوه‌ی عملکرد ثبات‌های $PORTx$ ، $DDRx$ و $PINx$ را توضیح دهید و پیکربندی لازم برای موارد زیر را

بنویسید:

الف) درگاه A را طوری تعریف کنید که بیت‌های آن به صورت یک در میان ورودی و خروجی باشند.

ب) درگاه B را به صورت خروجی تعریف کنید به طوری که مقاومت بالاکش آن حذف نشود.

3) برای نمایش کاراکترهای زیر، چه مقادیری را باید به ورودی خط داده‌ی 7-Segment اعمال کرد؟

d
A
H
F

- (1) همه‌ی ledaها 4 مرتبه خاموش و روشن شوند (در فاصله زمانی 0.5 ثانیه).
- (2) موقعیت LED روشن روی درگاه B، به مدت 3 ثانیه جابه جا گردد (یکی از LED ها را با قرار دادن عدد 1 روی پورت روشن نموده و در فاصله زمانی مشخصی عدد یک را شیفต์ دهید).
- (3) تغییرات درگاه A روی ledaها نمایش داده شود.
- (4) اعداد 0 تا 9 به صورت شمارش معکوس روی هر یک از 7segment نشان داده شود.
- (5) عدد سه رقمی خوانده شده از درگاه A روی 7Segment نشان داده شود و با دقت 0.2 و با فاصله زمانی 200 ms کاهش یابد.

6) فرض کنید که هر یک از کلیدهای فشاری برای ریست نمودن رقم‌های 7segment در نظر گرفته شده است. با استفاده از بند 5، برنامه‌ای بنویسید که ضمن کاهش عدد به اندازه‌ی 0.2، در صورت فشردن کلیدها رقم مربوطه صفر گردد و روند کاهشی همچنان ادامه یابد تا به صفر برسد.

7) زیربرنامه‌های بندهای 1 تا 6 را در یک پروژه تلفیق نمایید به صورتی که همه بندها به ترتیب اجرا گردند.

پروژه کامل کد ویژن (شامل تمام فایل‌ها) برای هریک از بندها را بنویسید و از فایل‌های کمکی نیز استفاده نمایید. سپس در محیط پروتئوس برنامه را شبیه سازی نموده و ارسال نمایید.