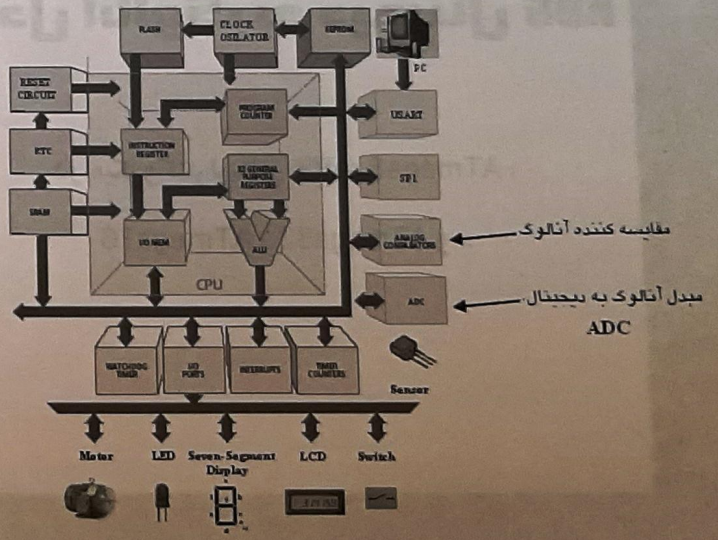
۸-۱: مقدمه

در مواقعی که میکروکنترلر به‌عنوان دستگاه کنترل و نظارت یک دستگاه صنعتی یا کنترلی ... به‌کار گرفته می‌شود، کمیات فیزیکی مانند درجه حرارت، فشار، ارتفاع، ... توسط سنسورها[1] به ولتاژ یا جریان نظیر تبدیل می‌شوند، ولی این سیگنال‌ها برای میکروکنترلر یا کامپیوتر قابل استفاده نیستند و باید به‌صورت دیجیتال یعنی یک سری 0 و 1 تبدیل گردند.

مبدل آنالوگ به دیجیتال ADC[2] یک دستگاه ورودی است که سیگنال‌های ولتاژ و جریان سنسورها و... را تبدیل به مقادیر دیجیتال قابل استفاده برای پروسسور یا CPU میکروکنترلرهای AVR می‌کند (شکل ۸-۱).

مبدل آنالوگ به دیجیتال ADC ، n بیتی دارای 2^n مقدار خروجی می‌باشد که ولتاژ نظیر کوچکترین بیت[3] LSB را یک پله یا Step نامند و مقدار آن برابر $Step = \frac{V}{2^n}$ که V حداکثر ولتاژ می‌باشد. بنابراین برای افزایش دقت مبدل ADC باید تعداد بیت خروجی را افزایش دهیم. به‌عنوان مثال مبدل آنالوگ به دیجیتال ADC ده بیتی، دارای یک سیگنال ورودی مداوم یا آنالوگ[4] و ده بیت خروجی دیجیتالی می‌باشد که مقدار خروجی از 0000000000 تا 1023=1111111111 $-1=2^{10}$ می‌تواند باشد که بزرگترین مقدار آن یعنی، 1111111111 نظیر حداکثر ولتاژ ورودی و کوچکترین مقدار خروجی، یعنی کوچکترین بیت LSB برابر 0000000001 نظیر کوچکترین ولتاژ ورودی است.

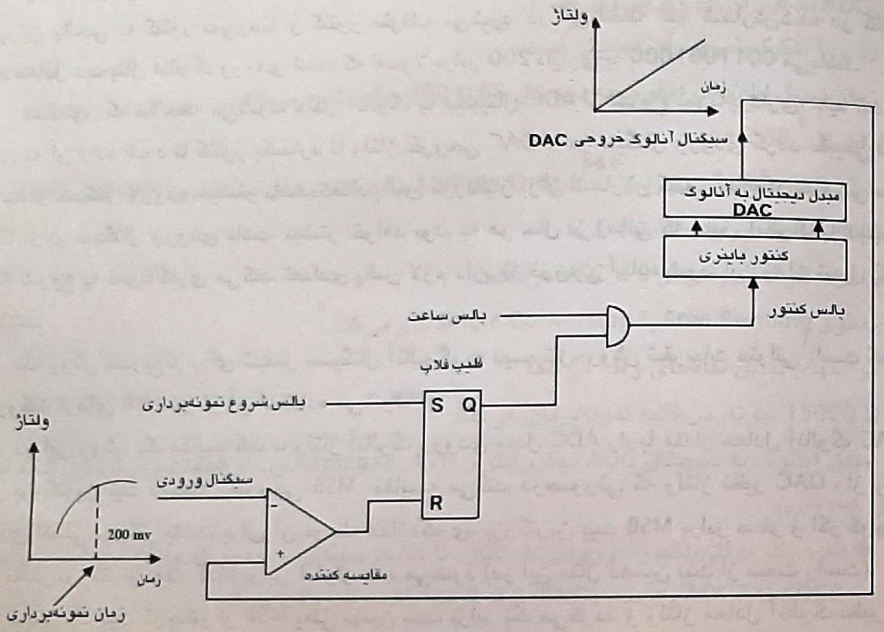


شکل (۸-۱) مبدل آنالوگ به دیجیتال ADC و مقایسه‌کنندهٔ آنالوگ و... در میکروکنترلرهای AVR

1- Sensors
2- Analog to Digital Converter (ADC)
3- Least Significant Bit (LSB)
4- Analog

به‌عنوان مثال اگر حداکثر ولتاژ آنالوگ ورودی برابر 1 ولت یا 1000 میلی‌وات باشد، این مبدل نظیر عدد دیجیتال ماکزیمم، یعنی 1111111111 است. لذا هر بیت خروجی نظیر $\frac{1000\ mv}{1023} \simeq 1mv$ می‌باشد. به‌عبارت دیگر اگر ورودی 1 میلی‌ولت باشد، خروجی برابر 0000000001 و اگر ورودی مساوی 2 میلی‌ولت باشد، خروجی مساوی 00000000010 و... می‌گردد.

برای تبدیل ولتاژ آنالوگ به دیجیتال می‌توان از مداری مطابق شکل (۸-۱- الف) استفاده نمود. برای این کار ولتاژ سنسور که به‌صورت آنالوگ یا مداوم می‌باشد، در یک لحظه نمونه‌برداری می‌گردد و عدد معادل دیجیتال آن ساخته می‌شود.



شکل (۸-۱- الف) ساختار اصولی مبدل آنالوگ به دیجیتال ADC

فرض می‌کنیم در یک لحظه سیگنال یا ولتاژ ورودی مدار، برابر 200 میلی‌ولت باشد. در این صورت مبدل آنالوگ به دیجیتال ADC عدد 200 را به‌صورت باینری در ده بیت مانند 0011001000 تولید می‌کند. برای این کار در شکل مذکور پالس شروع نمونه‌برداری را، فعال می‌کنیم تا خروجی Q فلیپ فلاپ برابر 1 شود، در این صورت پالس ساعت به خروجی AND می‌رود و کنتور باینری از 0 شروع به شمارش می‌کند. در خروجی کنتور باینری، یک مبدل دیجیتال به آنالوگ[1] DAC قرار گرفته که عدد دیجیتال را به معادل آنالوگ آن تبدیل می‌کند. به‌عنوان مثال اگر خروجی کنتور باینری عدد 96 یعنی 0001100000 باشد، خروجی مبدل، DAC ، ولتاژ 96 میلی‌ولت را تولید می‌کند. این ولتاژ به ورودی

1- Digital to Analog Converter (DAC)

کننده وارد می‌شود و ورودی دیگر مقایسه‌کنندهٔ، سیگنال ورودی است که در این لحظه 200 میلی‌ولت می‌باشد. در این مرحله چون سیگنال ورودی بیشتر از خروجی مبدل DAC است، لذا خروجی مقایسه‌کننده صفر می‌گردد. بنابراین خروجی فلیپ‌فلاپ مساوی 1 می‌ماند و پالس ساعت به کنتور می‌رسد و کنتور به شمارش ادامه می‌دهد و بالا می‌رود و در هر لحظه نیز عمل مقایسه انجام می‌شود. همین‌طور که شمارش کنتور ادامه می‌یابد و مقدار کنتور افزایش می‌یابد، در نتیجه ولتاژ خروجی مبدل DAC نیز افزایش می‌یابد و آنقدر این عمل ادامه می‌یابد تا ولتاژ خروجی مبدل DAC برابر ولتاژ سیگنال ورودی شود که در این لحظه خروجی مقایسه‌کننده، یک پالس 1 می‌فرستد و فلیپ‌فلاپ را Reset می‌کند، در نتیجه خروجی فلیپ‌فلاپ Q برابر 0 می‌شود، لذا خروجی گیت AND نیز مساوی صفر می‌گردد. بنابراین پالسی به کنتور نمی‌رسد و کنتور متوقف می‌شود. در این لحظه عدد شمارش‌شده در کنتور، عدد معادل دیجیتال آنالوگ ورودی است که اصولاً برابر 200 باینری یا 0011001000 می‌باشد.

همانطور که ملاحظه می‌شود، مبدل آنالوگ به دیجیتال ADC از شروع نمونه‌برداری، باید تعدادی پالس به آن داده شود تا کنتور بشمارد تا ولتاژ خروجی DAC برابر سیگنال ورودی گردد. طبیعی است هر چقدر سیگنال ورودی بیشتر باشد، تعداد پالس موردنیاز برای شمارش کنتور تا اینکه خروجی مبدل DAC برابر سیگنال ورودی باشد، بیشتر خواهد بود. به هر حال از زمانی که مبدل آنالوگ به دیجیتال ADC شروع به نمونه‌گیری می‌کند، تعدادی پالس لازم دارد تا خروجی آماده شود، این زمان تبدیل ADC می‌باشد.

یک روش سریع‌تر برای تبدیل سیگنال آنالوگ به دیجیتال، روش تقریبات متوالی[1] است که در میکروکنترلرهای AVR نیز از آن استفاده می‌شود.

در این روش یک مقایسه‌کننده، ولتاژ آنالوگ ورودی مبدل ADC را، با مقدار معادل آنالوگ DAC ، نظیر بزرگترین بیت دیجیتال خروجی MSB[2] مقایسه می‌کند. درصورتی که ولتاژ نظیر DAC ، از ولتاژ ورودی اصلی بزرگتر باشد، برای آن مرحله اندازه‌گیری، بزرگترین بیت MSB برابر صفر و اگر کوچکتر بود، مقدار بزرگترین بیت MSB برابر 1 قرار داده می‌شود (در این مثال دهمین بیت از سمت راست).

سپس یک بیت کوچکتر از MSB یعنی نُهمین بیت، برابر یک می‌گردد و ولتاژ معادل آنالوگ نظیر آن، با ولتاژ ورودی اصلی مقایسه می‌شود اگر ولتاژ نظیر آن از ورودی اصلی بزرگتر بود، این بیت (یعنی نُهمین بیت) برابر 0 می‌شود، در غیر این صورت این بیت مساوی یک می‌گردد و به همین ترتیب این روش تا کم‌ارزش‌ترین بیت یعنی LSB ادامه می‌یابد تا عدد معادل دیجیتال نظیر سیگنال آنالوگ ورودی به‌دست آید.

به‌عنوان مثال اگر مبدل ADC دارای ده بیت خروجی باشد، ده بار عمل مقایسه در ده پالس ساعت انجام می‌گیرد تا اطلاعات ده بیتی دیجیتال خروجی تولید گردد.

به این ترتیب از زمان شروع تبدیل سیگنال آنالوگ به دیجیتال، حداقل باید تعدادی پالس ساعت، مثلاً ده پالس ساعت منتظر شد تا عدد دیجیتال خروجی حاصل گردد. بدیهی است هر چه سرعت پالس ساعت ADC زیادتر باشد، عمل تبدیل سیگنال آنالوگ به دیجیتال سریع‌تر می‌شود.

1- Successive Approximation

2- Most Significant Bit (MSB)

◆ ۸-۲: مبدل‌های آنالوگ به دیجیتال ADC در میکروکنترلرهای AVR : ATmega8 ، ATmega16 ، ATmega32 و...

امروزه انواع، ICهای مبدل آنالوگ به دیجیتال ADC ساخته شده و در میکروکنترلرهای AVR نیز یک مبدل آنالوگ به دیجیتال ADC ده بیتی، داخل آن قرار داده شده که پالس ساعت ADC ، از پالس ساعت میکروکنترلر تأمین می‌گردد و به روش تقریبات متوالی، بعد از 13 پالس ساعت، سیگنال ورودی آنالوگ را به معادل دیجیتال آن تبدیل می‌کند.*

مبدل ADC مذکور دارای ویژگی زیر است:

- دقت یا تفکیک‌پذیری[1] آن ده بیتی است.**
- زمان تبدیل اطلاعات آنالوگ به دیجیتال 65 تا 260 میکروثانیه می‌باشد.
- دقت کل[2] 2LSB ± است.
- شش کانال آنالوگ ورودی Single Ended*** (میکروکنترلرهای بسته‌بندی TQFB و MLF هشت کانال آنالوگ ورودی دارند) دارد.
- ولتاژ آنالوگ ورودی از 0 تا VCC می‌باشد.
- به‌طور Free Running یا Single Conversion کار می‌کند.*
- در انتهای تبدیل تقاضای وقفه می‌کند.
- تا 15000 نمونه، در ثانیه نمونه‌برداری می‌کند.****

مبدل آنالوگ به دیجیتال ADC میکروکنترلر AVR: ATmega8 و... از قطعات زیر تشکیل شده است (شکل ۸-۱- ب).

◆ الف: یک مولتی‌پلکسر[3] ورودی ۸ کانال: تا هشت سیگنال ورودی که ولتاژ آن‌ها بین پایه‌های ADC0 تا ADC7 و زمین (GND) قرار گرفته باشد را دریافت می‌کند.

* بعداً جزئیات آن بیشتر بحث می‌شود.

** حداکثر عدد دیجیتال خروجی برابر $1023 = 1111111111 = 2^{10} - 1$ است که نظیر حداکثر ولتاژ ورودی 5 ولت می‌باشد، لذا دقت یا تفکیک‌پذیری آن برابر $\frac{5}{1023} \approx 5\ mv$ ، می‌شود. یعنی به ازای هر 5 میلی‌ولت، ADC یک بیت تغییرمی‌کند.

*** یعنی ولتاژ ورودی بین ADC و زمین می‌باشد در میکروکنترلرهای AVR: ATmega16 ، ATmega32 و برخی دیگر میکروکنترلرها، ورودی به‌صورت Differential نیز دارند، لذا زمین سیگنال ورودی می‌تواند با زمین میکروکنترلر متفاوت باشد و میکروکنترلر تفاضل دو پایه سیگنال ورودی را می‌گیرد.

**** تا 15 KSPS یا 15000 نمونه در ثانیه (Sample Per Seconds (SPS است. لذا حداکثر فرکانس ورودی کمتر از نصف آن یعنی حدود 6 کیلوهرتز می‌باشد.

1- Resolution

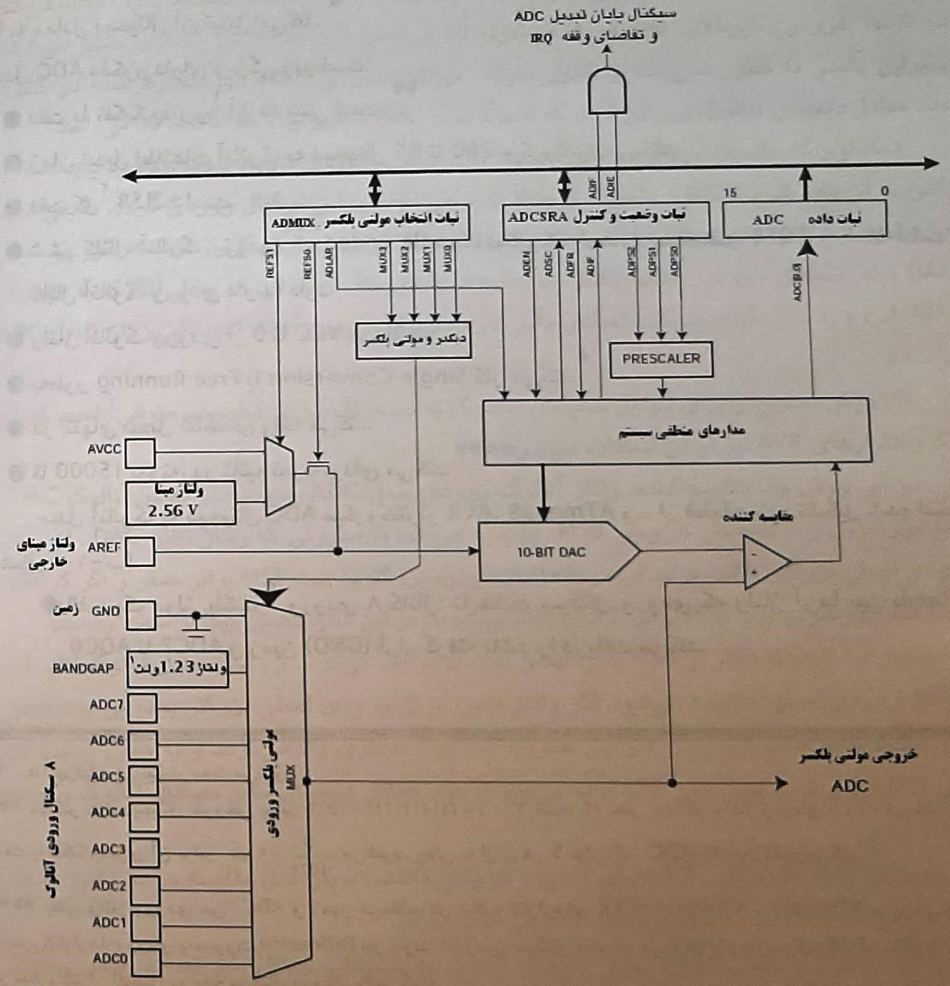
2- Absolute Accuracy

3- Multiplexer

◆ ب: ADC دارای یک ولتاژ مبنای آنالوگ مجزا AVCC است که این ولتاژ باید برابر VCC باشد (با تقریب 0.3± ولت).

◆ ج: ولتاژ مبنا 2.56 ولت نیز در ADC تولید می‌شود.

علاوه بر این ولتاژ مبنا را می‌توان از خارج به پایه AREF نیز متصل نمود. حداقل ولتاژ ورودی برابر 0 ولت و حداکثر آن برابر ولتاژ مبنا AREF است که نظیر عدد 0 تا $2^{10}-1=1023$ خروجی ADC با تقریب 1 بیت کم‌ارزش‌ترین بیت LSB± است.



شکل (۸-۱- ب) ساختار مبدل آنالوگ به دیجیتال میکروکنترلرهای AVR: ATmega8 و...

1- Bandgap Voltage (تابع درجه حرارت نیست و ثابت است)

◆ د: ولتاژ آنالوگ ورودی، تبدیل به ده بیت دیجیتال می‌شود و در ثبات داده ADC (یا مجموعه ثبات‌های ADCL و ADCH) قرار می‌گیرد که به‌طور پیش‌فرض اطلاعات در ده بیت طرف راست ثبات‌های داده ADCL و ADCH قرار می‌گیرد. در صورتی که هشت بیت در طرف چپ ثبات ADC قرار گیرد*، می‌توان فقط اطلاعات هشت بیت پرارزش‌تر ADCH را خواند.**

◆ هـ: به‌وسیلهٔ ثبات انتخاب ADMUX : می‌توان یکی از ورودی‌های آنالوگ ADC را انتخاب کرد و همچنین یکی از ولتاژهای مبنای ADC را انتخاب نمود.

◆ و: با ثبات کنترل و وضعیت ADCSRA : می‌توان مبدل ADC را فعال نمود، نمونه‌گیری از ولتاژ ورودی را شروع کرد، فرکانس پالس ساعت ADC و... را انتخاب نمود.

● **ADC در دو حالت کار می‌کند:**

◆ **الف: حالت یک بار تبدیل Single Conversion**

با 1 کردن بیت شروع ADSC در ثبات کنترل و وضعیت ADCSRA، مبدل ADC شروع به‌کار می‌کند و ولتاژ ورودی را تبدیل به ده بیت دیجیتال معادل آن کرده و در ثبات داده ADCL و ADCH قرار می‌دهد.**

> **نکته:** *بیت شروع ADSC در تمام مدت تبدیل سیگنال آنالوگ به دیجیتال برابر 1 می‌ماند و به محض اینکه عمل تبدیل تمام شد، توسط سخت‌افزار، این بیت برابر 0 می‌شود.*

◆ **ب: حالت نمونه‌گیری دائم یا Free Running**

با 1 کردن بیت شروع ADSC و بیت ADFR در ثبات کنترل و وضعیت ADCSRA*** مبدل ADC به‌طور مداوم سیگنال ورودی آنالوگ را تبدیل به دیجیتال می‌کند و در ثبات داده ADC قرار می‌دهد.

> **نکته:** *برای تبدیل یک سیگنال آنالوگ به دیجیتال، آسان‌تر است از Free Running استفاده کرد، ولی برای تبدیل چند سیگنال آنالوگ به دیجیتال باید Single Conversion به‌کار برد که بعد از پایان تبدیل، بتوان کانال ورودی را عوض کرد و سپس نمونه‌گیری نمود.*

* شکل (۸-۲)

** بعداً بیشتر بحث می‌شود.

*** در بخش (۸-۴) بحث می‌شود.

۸-۳: ثبات‌های داده[1] ADCL و ADCH

ثبات‌های داده ADCL و ADCH ثبات‌هایی هستند که نتیجهٔ تبدیل ولتاژ آنالوگ به دیجیتال ADC در آن‌ها قرار می‌گیرد. موقعی که ADC ، ولتاژ ورودی را تبدیل به ده بیت دیجیتال ADC0 تا ADC9 کرد، آن را در ثبات هشت بیتی داده ADCL و ADCH مطابق شکل (۸-۲) قرار می‌دهد.

Bit	15	14	13	12	11	10	9	8	شماره بیت
ADCH	–	–	–	–	–	–	ADC9	ADC8	ثبات‌ها
ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	
	7	6	5	4	3	2	1	0	شماره بیت
Read/Write	R	R	R	R	R	R	R	R*	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	مقدار اولیه

الف: در حالت بیت ADLAR=0**

Bit	15	14	13	12	11	10	9	8	شماره بیت
ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ثبات‌ها
ADCL	ADC1	ADC0	–	–	–	–	–	–	
	7	6	5	4	3	2	1	0	شماره بیت
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	مقدار اولیه

ب: در حالت بیت ADLAR=1**

شکل (۸-۲) ثبات داده ADC در حالتی که اطلاعات ده بیتی از طرف راست (بیت ADLAR=0) و یا چپ (بیت ADLAR=1) باشد در میکروکنترلرهای AVR: ATmega8 ، ATmega16 ، ATmega32 و...

نکته: اگر نتیجهٔ تبدیل سیگنال ورودی بزرگتر از هشت بیت باشد، می‌توان با ADLAR = 1 فقط اطلاعات بایت پرارزش‌تر ADCH را خواند در غیر این صورت باید ابتدا بایت کم‌ارزش‌تر ADCL و سپس بایت پرارزش‌تر ADCH ثبات داده خوانده شود.

نکته: اگر فقط بایت کم‌ارزش‌تر ADCL لازم باشد، باید ابتدا ADCL خوانده شود و سپس ADCH نیز خوانده شود، تا بتوان مجدداً اطلاعات جدید مبدل آنالوگ به دیجیتال (ADC) را خواند.***

* فقط قابل خواندن

** در بخش بعدی بحث می‌شود.

*** اگر ADCH لازم نیست می‌توان آن را خواند و در یک متغیر دلخواه قرار داد، ولی از آن استفاده نکرد تا عمل تبدیل ADC پایان یابد.

1- ADC Data Registers

مثال

دستوراتی بنویسید که از ثبات‌های داده ADCL و ADCH بخواند و اطلاعات ADC را در ثبات‌های R30 و R31 قرار دهد.

حل

برای این کار دستورات زیر را می‌نویسیم:

```
in R30, ADCL      ; (1)
in R31, ADCH      ; (2)
```

- دستور (۱): اطلاعات هشت بیت کم‌ارزش‌تر ثبات داده، یعنی ADCL را می‌خواند و در ثبات R30 قرار می‌دهد.
- دستور (۲): اطلاعات هشت بیت پُرارزش‌تر ثبات داده، یعنی ADCH را می‌خواند و در ثبات R31 قرار می‌دهد.

همانطور که ملاحظه می‌شود، ابتدا بایت کم‌ارزش‌تر ADCL خوانده شده و سپس بایت پُرارزش‌تر ADCH خوانده شده است.

۸-۴: ثبات کنترل و وضعیت[1] ADCSRA در میکروکنترلر AVR: ATmega8 و...

توسط ثبات کنترل و وضعیت می‌توان مبدل ADC را فعال کرد و شروع به نمونه‌گیری از ولتاژ ورودی آنالوگ نمود. علاوه بر این، بیت‌های این ثبات پایان عملیات ADC را اعلام می‌کنند، وقفه را فعال می‌نمایند و فرکانس پالس ساعت مبدل آنالوگ به دیجیتال ADC را انتخاب می‌کنند (شکل الف-۸-۳)

Bit	7	6	5	4	3	2	1	0	شماره بیت
ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ثبات
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	مقدار اولیه

شکل (۸-۳- الف) ثبات کنترل و وضعیت ADCSRA میکروکنترلرهای AVR: ATmega8 و...

بیت‌های ثبات کنترل و وضعیت ADCSRA در میکروکنترلر AVR: ATmega8** و... به شرح زیر می‌باشند:

بیت (۷): بیت فعال‌ساز مبدل ADEN[2]

اگر این بیت را در ثبات مذکور 1 کنیم، آن وقت مبدل ADC فعال می‌شود و اگر آن را 0 کنیم مبدل ADC غیرفعال می‌گردد.

* قابل خواندن و نوشتن

** ثبات کنترل و وضعیت میکروکنترلرهای AVR: ATmega16 ، ATmega32 و... در ادامه این بخش بحث می‌شود.

1- Control and Status Register A

2- ADC Enable (ADEN)

نکته: به محض اینکه مبدل ADC توسط بیت ADEN فعال شود، مبدل ADC شروع به کار می‌کند.

بیت (۶): بیت شروع نمونه‌گیری ADSC[1]

اگر بیت ADSC را 1 کنیم، در این صورت مبدل ADC شروع به نمونه‌گیری می‌کند و در حالت Free Running به‌طور دائم به نمونه‌گیری ادامه می‌دهد. در حالت یک بار تبدیل Single Conversion اگر این بیت را 1 کنیم، در پایان نمونه‌گیری بیت ADSC برابر 0 می‌شود. درصورتی که مجدداً لازم به نمونه‌گیری باشد، باید با دستورات AVR مجدداً بیت ADSC را 1 کنیم.*

بیت (۵): بیت انتخاب در حالت نمونه‌گیری دائم** ADFR[2]

موقعی که این بیت را 1 کنیم*** و درصورتی که بیت شروع به نمونه‌گیری ADSC را قبلاً 1 کرده باشیم، مبدل ADC مرتباً از سیگنال ورودی نمونه‌گیری می‌کند و نتیجه را در ثبات‌های داده ADCL و ADCH قرار می‌دهد، به‌عبارت دیگر ثبات داده مذکور را به روز می‌کند. اگر این بیت را 0 کنیم، حالت Free Running غیرفعال می‌شود و Single Conversion می‌شود.

بیت (۴): بیت پرچم وقفه ADIF[3]

موقعی که سیگنال آنالوگ ورودی، تبدیل به معادل دیجیتال ده بیتی شد و در ثبات‌های داده ADCL و ADCH قرار گرفت، آن وقت بیت پرچم وقفه ADIF برابر 1 می‌شود.

درصورتی که بیت فعال‌ساز وقفه ADIE (بیت شماره ۳) و بیت فعال‌ساز کلی وقفه I در ثبات SREG برابر 1 باشد، به شرطی که بیت پرچم وقفه ADIF برابر 1 شود، آن وقت روتین سرویس وقفه اجرا می‌گردد و بعد از اجرای روتین مذکور، بیت ADIF به‌وسیلهٔ سخت‌افزار به‌طور خودکار مساوی 0 می‌شود. البته با نوشتن 1 در این بیت، بیت ADIF ، Clear می‌گردد.

بیت (۳): بیت فعال‌ساز وقفه ADIE[4]

زمانی که بیت فعال‌ساز وقفهٔ ADIE را برابر 1 کنیم و بیت فعال‌ساز کلی وقفهٔ I در ثبات SREG نیز مساوی 1 باشد، آن وقت وقفه ADC فعال می‌شود.

بیت‌های (۲) تا (۰): بیت‌های مشخص‌کننده فرکانس پالس ساعت ADC ، با ADPS0 ، ADPS1 و ADPS2

بیت‌های ADPS0 ، ADPS1 و ADPS2 مطابق جدول (۸-۱) مشخص‌کننده عددی هستند که فرکانس پالس ساعت میکروکنترلر AVR بر آن تقسیم می‌شود تا فرکانس پالس ساعت ADC را مشخص کند.

---

* بعداً مثال خواهیم داشت.

** در مورد AVR: ATmega16 ، ATmega32 و... بیت ADATE می‌باشد که در ادامه بحث می‌شود.

*** این بیت تا پایان Free Running برابر 1 می‌ماند.

1- ADC Start Conversion (ADSC)
2- ADC Free Running Select (ADFR)
3- ADC Interrupt Flag (ADIF)
4- ADC Interrupt Enable (ADIE)

جدول (۸-۱) بیت‌های ADPS0 ، ADPS1 و ADPS2 برای تعیین فرکانس پالس ساعت مبدل ADC در میکروکنترلرهای AVR: ATmega8 ، ATmega16 ، ATmega32 و...

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

مثال

دستوراتی بنویسید که مبدل ADC را فعال کند، در حالت Single Conversion قرار گیرد، وقفه را نیز فعال کند و فرکانس پالس ساعت ADC را برابر 1/128 فرکانس پالس ساعت AVR قرار دهد. برای این کار دستورات زیر را می‌نویسیم:

```
ldi  R30, 0x8F          ; (1)
out  0x06, R30          ; (2)
        یا
ldi  R30, 0b10001111    ; (3)
out  ADCSRA, R30        ; (4)
```

◆ دستورات (۱) و (۲): عدد 10001111 را در ثبات کنترل و وضعیت ADCSRA قرار می‌دهند. در نتیجه چون:

سه بیت کم‌ارزش‌تر ثبات وضعیت، یعنی ADPS0 ، ADPS1 و ADPS2 برابر1 شده‌اند، لذا طبق جدول (۸-۱) فرکانس مبدل ADC برابر 1/128 فرکانس AVR می‌شود.

بیت شماره (۳): یعنی چهارمین بیت از سمت راست برابر 1 شده است، لذا بیت فعال‌ساز وقفه ADIE برابر 1 شده در نتیجه، وقفه می‌تواند فعال شود.

بیت شماره (۷): بین بیت فعال‌ساز مبدل ADEN مساوی 1 شده در نتیجه مبدل ADC فعال می‌گردد.

و چون بیت (۵) یعنی ADFR برابر 0 شده، لذا ADC در حالت Single Conversion قرار می‌گیرد.

نکته: در دستور (۲) آدرس ثبات وضعیت ADCSRA یعنی 06 قرار داده شده است.

◆ دستورات (۳) و (۴): معادل دستورات (۱) و (۲) می‌باشند که در دستور (۳)، عدد باینری استفاده شده و در دستور (۴) نام ثبات ADCSRA ذکر شده است.

۸-۴-۱: ثبات کنترل و وضعیت ADCSRA در میکروکنترلرهای AVR: ATmega16، ATmega32 و...

بیت‌های این ثبات، مشابه بیت‌های ثبات کنترل و وضعیت میکروکنترلر AVR: ATmega8 شکل (۸-۳- الف) می‌باشد، فقط بیت شمارهٔ (۵) آن اختلاف دارد که در ذیل شرح داده می‌شود.

شماره بیت	7	6	5	4	3	2	1	0	
Bit	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ثبات ADCSRA
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	مقدار اولیه

شکل (۸-۳- ب) ثبات کنترل و وضعیت ADCSRA در میکروکنترلرهای AVR: ATmega16، ATmega32 و...

بیت شماره (۵): بیت فعال کردن تریگر اتوماتیک ADATE

اگر این بیت را 1 کنیم، در این صورت مبدل ADC به‌صورت اتوماتیک توسط مقدار بیت‌های (۵) تا (۷)، ثبات SFIOR[1] یعنی بیت‌های ADTS0، ADTS1 و ADTS2 (شکل ۸-۳- ج) تریگر[2] می‌شود.

روش‌های مختلف تریگر کردن مبدل آنالوگ به دیجیتال ADC، با انتخاب مقدار بیت‌های مذکور در ثبات SFIOR مشخص می‌شود (مطابق جدول (۸-۱- الف)).

اگر سه بیت‌های مذکور در ثبات SFIOR برابر 000 قرار داده شوند، مبدل آنالوگ به دیجیتال ADC به‌صورت Free Running کار می‌کند. نظر به اینکه به‌طور پیش‌فرض این سه بیت مساوی 000 هستند، لذا اگر فقط بیت ADATE درثبات ADCSRA را برابر 1 کنیم، مبدل ADC به صورت Free Running کار می‌کند.

شماره بیت	7	6	5	4	3	2	1	0	
Bit	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	ثبات SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	مقدار اولیه

شکل (۸-۳- ج) ثبات SFIOR در میکروکنترلرهای AVR: ATmega16 و ATmega32 و...

جدول (۸-۱- الف) طرق مختلف تریگر کردن مبدل ADC در میکروکنترلر AVR: ATmega16، ATmega32 و...

ADTS2	ADTS1	ADTS0	Trigger Source	
0	0	0	Free Running mode	
0	0	1	Analog Comparator	توسط مقایسه کنند آنالوگ
0	1	0	External Interrupt Request 0	توسط وقفه خارجی 0
0	1	1	Timer/Counter0 Compare Match	توسط تایمر 0
1	0	0	Timer/Counter0 Overflow	توسط تایمر 0
1	0	1	Timer/Counter Compare Match B	توسط تایمر 1
1	1	0	Timer/Counter1 Overflow	توسط تایمر 1
1	1	1	Timer/Counter1 Capture Event	توسط تایمر 1

- اگر سه بیت مذکور در ثبات SFIOR برابر مقدار 001 قرار داده شوند، ADC توسط از 0 به 1 رفتن بیت پرچم وقفه مقایسه‌کنندهٔ آنالوگ، تریگر می‌گردد. و به همین ترتیب اگر در بیت‌های مذکور مقادیر دیگر قرار داده شوند (مطابق جدول (۸-۱- الف))، نمونه‌گیری مبدل ADC با تریگر کردن آن توسط از 0 به 1 رفتن بیت پرچم وقفه دستگاه‌های مذکور، یعنی وقفه خارجی 0، تایمر 0، تایمر 1 -- انجام می‌شود.

1- Special Function I/O Register (SFIOR)
2- Trigger

نکته: درصورتی که بیت ADATE در ثبات کنترل و وضعیت ADCSRA در میکروکنترلر AVR: ATmega16، ATmega32 و... برابر 0 قرار داده شود، بیت‌های ADTS0، ADTS1 و ADTS2 در ثبات SFIOR اثری ندارند و مبدل آنالوگ به دیجیتال ADC به‌صورت Single Conversion عمل می‌کند.

۸-۵: ثبات انتخاب کانال مولتی‌پلکسر ADMUX[1] در میکروکنترلرهای AVR: ATmega8 و...

ثبات ADMUX (شکل ۸-۴) یکی از ورودی‌های آنالوگ مبدل ADC را انتخاب می‌کند، نتیجهٔ تبدیل را به چپ یا راست منتقل می‌کند و ولتاژ مبنا را برای ADC انتخاب می‌کند.

شماره بیت	7	6	5	4	3	2	1	0	
Bit	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ثبات ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	مقدار اولیه

شکل (۸-۴) ثبات انتخاب کانال مولتی‌پلکسر ADMUX در میکروکنترلر AVR: ATmega8 و...

بیت‌های ثبات ADMUX به شرح زیر می‌باشند:

بیت‌های (۶) و (۷): بیت‌های انتخاب REFS0 و REFS1 ولتاژ مبنا

این بیت‌ها مطابق جدول (۸-۲) ولتاژ مبنای مبدل ADC را تعیین می‌کنند.

جدول (۸-۲) انتخاب ولتاژ مبنای مبدل ADC برای میکروکنترلرهای AVR: ATmega8، ATmega16، ATmega32 و...

REFS1	REFS0	انتخاب ولتاژ مبنا
0	0	ولتاژ مبنا از خارج به AREF متصل می‌شود.
0	1	ولتاژ مبنا AVCC است.
1	0	رزرو شده
1	1	ولتاژ مبنای داخلی* 2.56 ولت

در موقع مقدار اولیه دادن به ثبات‌ها است باید یکی از ولتاژهای مبنای فوق را برای ADC انتخاب نمود. ورودی AREF ولتاژ دلخواه مبنا است. به‌عنوان مثال اگر این ولتاژ را برابر VCC=5V بگیریم، در این صورت خروجی دیجیتال مبدل ADC بین 0 تا $2^{10}-1$ 1111111111=1023= برای ولتاژ ورودی بین 0 تا 5 ولت خواهد بود (با تقریب ±1 بیت).

* با خازن متصل به پایه AREF

1- ADC Multiplexer Selection Register (ADMUX)

اگر مقدار REFS0=1 و REFS1=0 قرار داده شود در این صورت ولتاژ مبنا V_{REF} برابر AVCC یا مساوی 5 ولت می‌باشد.

در این حالت ولتاژ مورد نیاز برای تغییر یک بیت ADC یا دقت آن برابر است با:

$$\frac{5}{1023^{\times}} \approx 5mv$$

یعنی به‌ازای هر 5 میلی‌ولت ورودی، 1 بیت ADC تغییر می‌کند.

اگر مقدار REFS0=1 و REFS1=1 قرار داده شود، ولتاژ مبنا V_{REF} برابر 2.56 ولت از داخل میکروکنترلر تأمین می‌گردد.

در این صورت ولتاژ موردنیاز برای تغییر یک بیت ADC یا دقت آن برابر است با:

$$\frac{2.56}{1023^{\times}} \approx 2.5mv$$

یعنی به‌ازای هر 2.5 میلی‌ولت، یک بیت ADC تغییر می‌کند.**

> **نکته:** *اگر از ولتاژهای AVCC یا 2.56 ولت داخلی به‌عنوان ولتاژ مبنا V_{REF} استفاده شود، نباید به پایه AREF میکروکنترلر ولتاژی متصل نمود.*

به‌طور کلی نتیجهٔ تبدیل سیگنال آنالوگ به دیجیتال در ثبات‌های ADCL و ADCH قرار می‌گیرد و برابر است با:

$$\text{عدد دیجیتال خروجی ADC} = \frac{V_{in} \times 1023}{V_{REF}}$$

که V_{in} ولتاژ ورودی و V_{REF} ولتاژ مبنا است و عدد خروجی مبدل ADC بین 0 تا 1111111111 یا بین 0 تا 1023 می‌باشد.

> **نکته:** *حداقل ولتاژ ورودی برابر زمین GND و حداکثر ولتاژ ورودی برابر ولتاژ V_{REF} است.*

بیت (۵): بیت[1] ADLAR برای قرار دادن نتیجهٔ تبدیل در طرف چپ ثبات داده

اگر بیت ADLAR را برابر 1 کنیم، نتیجهٔ تبدیل در طرف چپ ثبات‌های داده‌های ADCL و ADCH قرار می‌گیرد، در غیر این صورت نتیجه در طرف راست ثبات‌های مذکور قرار می‌گیرد (شکل ۸-۲).

بیت‌های (۳) تا (۰): بیت‌های انتخاب کانال سیگنال ورودی آنالوگ[2] در میکروکنترلرهای AVR:ATmega8

بیت‌های (۰) تا (۳)، یعنی بیت‌های MUX0 تا MUX3 ثبات مذکور مطابق جدول (۸-۳) ورودی‌های سیگنال آنالوگ ADC0 تا ADC7 و یا ولتاژ مبنای Bandgap (1.23 ولت) را برای مبدل ADC انتخاب می‌کند.***

* 1023 موقعی است که تمام بیت‌های ADC برابر 1111111111 باشد.
** در این حالت حداکثر ولتاژ ورودی 2.56V ولت است.
*** موقعی که مبدل ADC فعال شود، به‌طور اتوماتیک ورودی‌های ADC0 تا ADC7، ورودی می‌شوند و نیازی به تعریف این پایه‌ها به‌عنوان ورودی نیست.

1- ADC Left Adjust Result (ADLAR)
2- Analog Channel Selection Bits

جدول (۸-۳) بیت‌های انتخاب سیگنال آنالوگ ورودی در میکروکنترلر ATmega8 :AVR

بیت‌های MUX3..0	ورودی[1]
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	
1001	
1010	
1011	
1100	
1101	
1110	1.23V (V_{BG})
1111	0V (GND)

به‌عنوان مثال با انتخاب MUX0 = 1، MUX1 = 0، MUX2 = 1 و MUX3 = 0 کانال پنج سیگنال ورودی آنالوگ ADC5 برای مبدل ADC انتخاب می‌شود.

> **نکته:** *قبل از شروع نمونه‌برداری ADC یا فعال کردن بیت شروع ADSC ، باید کانال ورودی انتخاب گردد.*

◆ ۸-۵-۱: ثبات انتخاب کانال مولتی پلکسر ADMUX در میکروکنترلرهای AVR: ATmega16 ، ATmega32 و...

در ثبات ADMUX (شکل (۸-۴- الف)) با انتخاب مقادیری برای بیت‌های MUX0 تا MUX4 ، یکی از ورودی‌های آنالوگ مبدل ADC به‌عنوان Signal Ended (یعنی ولتاژ ورودی نسبت به زمین) و یا به شکل تفاضلی Differential (یعنی اختلاف ولتاژ دو ورودی) برای مبدل ADC انتخاب می‌شود.

علاوه بر این، در ثبات مذکور با مقدار ADLAR ، نتیجهٔ تبدیل ADC به چپ یا راست منتقل می‌شود.

همچنین با بیت‌های REFS0 و REFS1 می‌توان ولتاژ مبنا برای مبدل ADC را انتخاب نمود.

Bit	7	6	5	4	3	2	1	0	شماره بیت
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ثبات ADMUX
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	مقدار اولیه

شکل (۸-۴-الف) ثبات انتخاب کانال مولتی‌پلکسر ADMUX در میکروکنترلر AVR: ATmega16 ، ATmega32 و...

1- Single Ended Input

عملکرد بیت‌های این ثبات به شرح زیر می‌باشند:

بیت‌های (۵) تا (۷): عیناً مانند میکروکنترلر AVR: ATmega8 است که در بخش (۸-۵) به تفصیل بحث شده است.

بیت‌های (۰) تا (۴): MUX4:MUX0: بیت‌های انتخاب کانال سیگنال آنالوگ ADC در میکروکنترلر AVR: ATmega16 ، ATmega32 و...

بیت‌های (۰) تا (۴)، یعنی بیت‌های MUX0 تا MUX4 ثبات مذکور مطابق جدول (۸-۳- الف)، ورودی‌های سیگنال ADC0 تا ADC7 یا ولتاژ مبنا Bandgap (1.23ولت) را برای مبدل ADC انتخاب می‌کنند.

جدول (۸-۳- الف) بیت‌های انتخاب سیگنال آنالوگ ورودی در میکروکنترلر AVR: ATmega16 ، ATmega32 و...

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	N/A	ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010[۲۹]		ADC0	ADC0	200x
01011[۲۹]		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110[۲۹]		ADC2	ADC2	200x
01111[۲۹]		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x

همانطور که از جدول مذکور مشاهده می‌شود، اگر بیت‌های انتخاب کانال MUX0 تا MUX4 برابر 00000 تا 00111 انتخاب شوند. یکی از ورودی‌های ADC0 تا ADC7 به‌صورت Single Ended (یعنی ولتاژ ورودی نسبت به زمین) برای مبدل ADC انتخاب می‌شود.

اگر مقادیر MUX0 تا MUX4 برابر 10000 تا 10111 انتخاب شوند، هفت ورودی تفاضلی خواهیم داشت که ورودی مثبت ADC0 ، ADC2، ADC3 و... یا ADC7 می‌باشند و ورودی منفی ADC1 است و خروجی مبدل ADC عدد مثبت یا منفی در سیستم مکمل 2 می‌باشد.

دو ورودی تفاضلی نیز یکی ADC0- ADC1 و دیگری ADC2 – ADC3 باگین دارد. در صورتی که مقادیر MUX0 تا MUX4 برابر 01000 تا 01111 انتخاب شوند ورودی مثبت ADC1 یا ADC3 است و ورودی منفی ADC0 یا ADC2 می‌باشد. در این حالت ورودی‌های سیگنال دارای گین 10 یا 200 مطابق جدول مذکور می‌تواند داشته باشند، ولی تعداد بیت خروجی به ترتیب ۸ و ۷ بیت خواهد بود، یعنی دقت کم می‌شود.

به این ترتیب در میکروکنترلر AVR: ATmega16 ، ATmega32 و... ولتاژ ورودی نسبت به زمین (Single Ended) و یا تفاضل ولتاژ ورودی (Differential) را می‌توان، به مقدار دیجیتال نظیر تبدیل نمود.

به‌طور کلی نتیجه تبدیل سیگنال آنالوگ به دیجیتال در ثبات‌های داده ADCL و ADCH قرار می‌گیرد. اگر ولتاژ ورودی آنالوگ نسبت به زمین (Single Ended) باشد، در این صورت عدد دیجیتال خروجی ADC برابر است با:

$$\text{عدد دیجیتال خروجی ADC} = \frac{Vin \times 1023}{V_{REF}}$$

که Vin ولتاژ ورودی و V_{REF} ولتاژ مبنا است و عدد خروجی مبدل ADC بین 0 تا 1111111111 یا بین 0 تا 1023 می‌باشد.

اگر ولتاژ ورودی به صورت تفاضلی (Differential) استفاده شود، در این صورت نتیجه تبدیل برابر است با:

$$\text{عدد خروجی دیجیتال مبدل ADC} = \frac{512(V_p - V_N)Gain}{V_{REF}}$$

که V_P ولتاژ پایه مثبت، V_N ولتاژ پایه منفی، V_{REF} ولتاژ مبنا و Gain مقدار گین ADC است. در این صورت خروجی به صورت مکمل 2 بین 511+ تا 512- می‌باشد و بیت علامت، پُرارزش‌ترین بیت، یعنی ADC9 است. اگر این بیت 1 باشد یعنی عدد منفی است و اگر 0 باشد عدد مثبت می‌باشد.

مثال

دستورات زیر در میکروکنترلر AVR: ATmega8

```
ldi  R30, 0b01000000        ; (1)
out  ADMUX, R30             ; (2)
```

الف: باعث می‌شوند عدد 01000000 در ثبات ADMUX قرار گیرد در نتیجه مطابق جدول (۸-۲) ولتاژ مبنا AVCC=5v برای مبدل ADC انتخاب می‌شود، یعنی ورودی می‌تواند بین 0 تا 5 ولت تغییر کند.

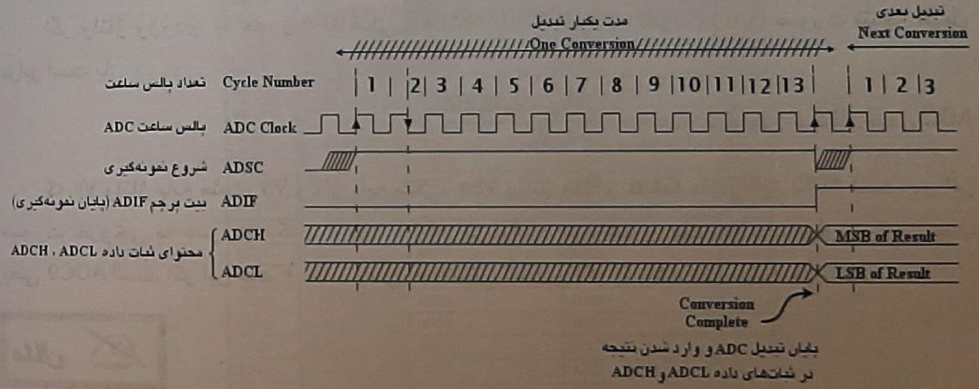
ب: چون بیت (۵) یعنی ADLAR برابر 0 است، لذا اطلاعات در طرف راست ثبات‌های ADCL و ADCH قرار می‌گیرد.

ج: چون بیت‌های (۰) تا (۲) برابر 0 هستند، لذا سیگنال ورودی ADC0 مطابق جدول (۸-۳) برای مبدل ADC انتخاب می‌شود.

نکته: قبل از شروع به کار ADC باید به ثبات‌های ADC مقدار اولیه داد. مقدار اولیه دادن ثبات‌ها معمولاً شامل انتخاب یکی از ۸ کانال ورودی سیگنال، انتخاب پالس ساعت ADC، فعال کردن ADC، نوع کار ADC، فعال کردن وقفه ... و بالاخره شروع به کار ADC می‌باشد.

۸-۶: زمان تبدیل اطلاعات ورودی آنالوگ به دیجیتال در ADC

مبدل ADC با پالس ساعتی فرکانس بین حدود 50KHz تا 200KHz با روش تقریبات متوالی کار می‌کند. به محض اینکه بیت شروع ADSC در ثبات کنترل ADCSRA برابر 1 شد، مبدل ADC شروع به نمونه‌گیری ولتاژ ورودی می‌کند و به‌طور معمول در 13 پالس ساعت اطلاعات ورودی را تبدیل به معادل دیجیتال آن می‌کند.* نتیجه را نیز در ثبات‌های داده ADCL و ADCH قرار می‌دهد که در این صورت بیت پرچم وقفه ADIF در ثبات کنترل و وضعیت ADCSRA برابر 1 می‌شود، که نشانهٔ پایان تبدیل سیگنال آنالوگ به دیجیتال می‌باشد (شکل ۸-۵).



شکل (۸-۵) دیاگرام زمانی مبدل ADC در میکروکنترلرهای AVR: ATmega8، ATmega16، ATmega32 و...

نکته: عوض کردن کانال ورودی باید موقعی که عمل تبدیل تمام شده، انجام شود. یعنی زمانی که بیت پرچم وقفهٔ ADIF برابر 1 شده است، در روتین سرویس وقفه، کانال را عوض نمود.

* البته اولین باری که مبدل ADC شروع به نمونه‌گیری می‌کند ADC نیاز به آماده شدن دارد لذا زمان کل تبدیل، 25 پالس ساعت طول می‌کشد.

۸-۷: مثال‌ها و پروژه‌هایی در میکروکنترلرهای AVR: ATmega8، ATmega16، ATmega32 و... مربوط به مبدل آنالوگ به دیجیتال ADC، با دستورات اسمبلی میکروکنترلرهای AVR و زبان C

مثال ۸-۱

برنامه‌ای برای میکروکنترلر AVR: ATmega8 با دستورات اسمبلی میکروکنترلر بنویسید که با استفاده از وقفه و روش Free Running سیگنال آنالوگ ورودی ADC0 را تبدیل به دیجیتال کند و در پورت D قرار دهد.

حل

برنامهٔ مذکور مطابق شکل (۸-۶- الف) می‌باشد. در این برنامه در:

- **بخش A:** نام میکروکنترلر AVR: ATmega8 و نام فایل ADC0.asm قرار داده شده است.
- **بخش B:** Header File و آدرس‌های برنامه اصلی و روتین سرویس وقفه تعیین می‌شوند.
  - عبارت (۱): فایل Header File به نام "m8def.inc" برای میکروکنترلر AVR: ATmega8 معرفی شده است.
  - عبارت (۲): آدرس برنامهٔ اصلی است.
  - عبارت (۳): پرش به بخش C است که با برچسب reset شروع می‌شود.
  - عبارت (۴): آدرس روتین سرویس وقفهٔ مبدل ADC است که برابر 00E هگزا دسیمال می‌باشد.
  - عبارت (۵): پرش به روتین سرویس وقفهٔ ADC است که در قسمت E می‌باشد.
- **بخش C:** به اشاره‌گر پُشته (SP)، پورت D، ثبات‌های ADC مقدار اولیه داده می‌شود و وقفه کلی I فعال می‌گردد.

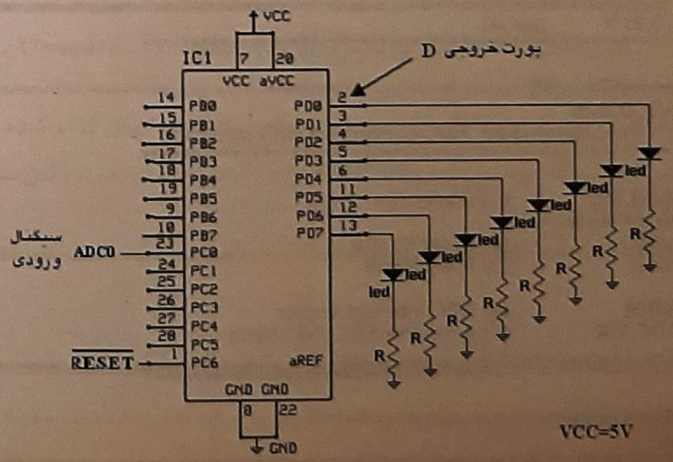
◆ دستورات (۶)، (۷)، (۸) و (۹): ثبات اشاره‌گر پُشته (SP) را مقدار اولیهٔ می‌دهد.

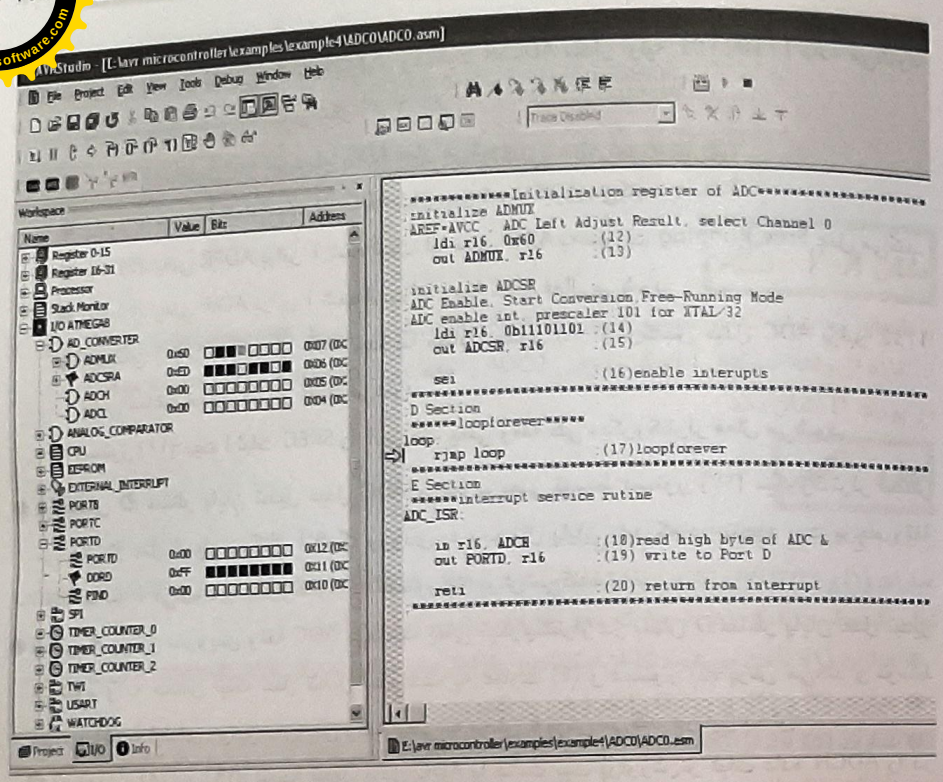
```
;          ADC0.asm          نام فایل
;*************************************************************
;A Section
;AVR number    ATmega8
;File name ADC0.asm
;Description    Free running for ADC conversion with interrupt
;*************************************************************
;B Section
.include "m8def.inc"          ;(1)header file
;interrupt vector table
      .org 0x000              ;(2) Reset Vector
      rjmp reset              ;(3) Rjmp Reset

      .org 0x00E              ;(4) ADC interupt vector
      rjmp ADC_ISR            ;(5) Jump the ADC_ISR routin
;*************************************************************
;C Section
;**********Main program entry point***********
;Initialization
```

```
;initialize stack pointer
reset:
      ldi r16, low(RAMEND)      ;(6)stack setup
      out SPL, r16               ;(7)SPL &
      ldi r16, high(RAMEND)     ;(8)SPH to
      out SPH, r16               ;(9)RAMEND
;initialize port D as output
      ldi r16, 0xFF             ;(10)
      out DDRD, r16             ;(11)

;************Initialization register of ADC*********************
;initialize ADMUX
;AREF=AVCC , ADC Left Adjust Result, select Channel 0
      ldi r16, 0x60             ;(12)
      out ADMUX, r16            ;(13)

;initialize ADCSRA
;ADC Enable, Start Conversion,Free-Running Mode
;ADC enable int, prescaler:101 for XTAL/32
      ldi r16, 0b11101101   ;(14)
      out ADCSRA, r16           ;(15)

      sei                       ;(16)enable interupts
;*****************************************************************
;D Section
;******loopforever*****
loop:
      rjmp loop                 ;(17)loopforever
;*****************************************************************
;E Section
;******interrupt service rutine
ADC_ISR:

      in r16, ADCH              ;(18)read high byte of ADC &
      out PORTD, r16            ;(19) write to Port D

      reti                      ;(20) return from interrupt
;*****************************************************************
```

الف: برنامه

ب: شکل میکروکنترلر





ج: نتیجه تست برنامه ADC0.asm در سیمولاتور AVR Studio

شکل (۸-۶) برنامهٔ تبدیل سیگنال آنالوگ ورودی ADC0 به دیجیتال

بـا روش Free Running در مـیکروکنتـرلـر AVR: ATmega8 و...

- دستورات (۱۰) و (۱۱): پورت D را خروجی می‌کنند.
- دستورات (۱۲) و (۱۳): به ثبات انتخاب کانال ADMUX مقدار 01100000 داده می‌شود، در نتیجه مطابق بخش (۸-۵) و شکل (۸-۴):
  - چون بیت‌های (۶) و (۷): برابر 01 قرار داده شده‌اند لذا ولتاژ مبنا AVCC است که در این حالت AVCC به VCC یا 5 ولت متصل شده است.
  - چون بیت (۵): یعنی ADLAR برابر 1 قرار داده شده است، پس مطابق شکل (۸-۲) ده بیت نتیجه تبدیل ADC در طرف چپ ثبات‌ها داده ADCL و ADCH قرار می‌گیرد.
  - چون بیت‌های (۰) تا (۳) برابر 0 قرار داده شده است، لذا سیگنال آنالوگ کانال 0 یعنی ADC0 برای مبدل ADC انتخاب می‌شود.

دستورات (۱۴) و (۱۵): به ثبات کنترل و وضعیت ADCSR مقدار اولیه 11101101 داده می‌شود، در نتیجه مطابق بخش (۸-۴) و شکل (۸-۳- الف):

- بیت (۷): برابر 1 شده است، لذا ADC فعال می‌شود.
- بیت (۶): برابر 1 شده است، لذا ADC شروع به نمونه‌گیری می‌کند.
- بیت (۵): یعنی ADFR برابر 1 شده است، لذا مبدل ADC به‌صورت Free Running عمل می‌کند.
- بیت (۳): یعنی ADIE برابر 1 شده است لذا وقفهٔ ADC فعال می‌شود.
- بیت‌های (۰) تا (۲): برابر 101 شده لذا مطابق جدول (۸-۱) فرکانس مبدل ADC برابر 1/32 پالس ساعت میکروکنترلر AVR می‌شود.

◆ دستور (۱۶): بیت I ثبات SREG را 1 می‌کند، یعنی وقفه کلی میکروکنترلر فعال می‌شود.

● **در بخش D**: منتظر پایان تبدیل مبدل ADC می‌شود، یعنی توسط دستور (۱۷) میکروکنترلر منتظر می‌ماند، تا عمل تبدیل سیگنال آنالوگ ورودی به دیجیتال پایان یابد که در نتیجه بیت پرچم وقفهٔ ADIF برابر 1 می‌شود و میکروکنترلر به دستور (۵) پرش می‌کند.

● **بخش E**: روتین سرویس وقفهٔ ADC می‌باشد، یعنی میکروکنترلر در بخش D منتظر پایان عمل تبدیل می‌شود و به محض اینکه عمل تبدیل تمام شد، به عبارت (۴) و دستور (۵) پرش می‌کند و توسط دستور (۵) به ابتدای روتین سرویس وقفه، یعنی به بخش E پرش می‌کند. در این بخش:

◆ دستورات (۱۸) و (۱۹): نتیجهٔ تبدیل مبدل ADC یا هشت بیت پُرارزش‌تر ثبات داده ADCH را در پورت D قرار می‌دهد.

◆ دستور (۲۰): برگشت از روتین سرویس وقفه است، یعنی به بخش D برمی‌گردد و منتظر عمل تبدیل جدید مبدل آنالوگ به دیجیتال ADC می‌شود و عملیات فوق تکرار می‌گردد.

به این ترتیب مبدل ADC میکروکنترلر، به‌طور مرتب سیگنال ورودی آنالوگ را به دیجیتال تبدیل می‌کند و نتیجه را در پورت D قرار می‌دهد.

باید توجه نمود که در این مسئله هشت بیت پرارزش‌تر عمل تبدیل به پورت D ارسال شده و از دو بیت کم‌ارزش‌تر، آن یعنی بیت‌های 0 و 1 صرف‌نظر شده است که این موضوع تقریبی حداکثر برابر 3/1023 ولتاژ ورودی است. در صورتی که این تقریب موردنظر نباشد، می‌توان کل ده بیت نتیجهٔ تبدیل را به دو پورت خروجی ارسال نمود که برنامهٔ آن کمی بزرگتر می‌شود.

این برنامه در سیمولاتور AVR Studio تست گردیده است و همانطور که در شکل (۸-۶- ج) ملاحظه می‌شود با اجرای دستورات میکروکنترلر به ثبات‌های ADC و پورت D مقدارهای موردنظر داده شده است *. علاوه بر این، فایل با پسوند HEX برنامه (شکل (۸-۷)) نیز توسط نرم‌افزار PonyProg در میکروکنترلر Program و تست شده است.

* در ثبات‌ها رنگ مشکی و سفید به ترتیب مقادیر 1 و 0 می‌باشد.

نام فایل adc0.HEX

```
:020000020000FC
:020000000EC030
:10001C000CC00FE50DBF04E00EBF0FEF01BB00E6F7
:10002C0007B90DEE06B97894FFCF05B102BB189550
:00000001FF
```

شکل (۸-۷) فایل HEX برنامهٔ اسمبلی شکل (۸-۶) برای تبدیل سیگنال آنالـــوگ ورودی ADC0 بـه دیجیتال بـا روش Free Running

مثال ۸-۲

برنامه‌ای به زبان C در میکروکنترلر AVR: ATmega8 در محیط CodeVisionAVR بنویسید که با استفاده از وقفه، سیگنال آنالوگ ورودی ADC0 را به‌صورت Free Running تبدیل به دیجیتال نماید و در پورت D نشان دهد.

حل

برنامه آن مطابق شکل (۸-۸- ب) می‌باشد.

عبارات (۱) و (۳) تا (۷) متناسب با انتخاب ما توسط ابزار CodeWizardAVR و نرم‌افزار CodeVisionAVR تولید شده است شکل (۸-۸- ج) که معنی آن‌ها به شرح زیر می‌باشند:

● عبارت (۱): باعث می‌شود که از امکانات و پارامترهای میکروکنترلر AVR: ATmega8 استفاده شود.

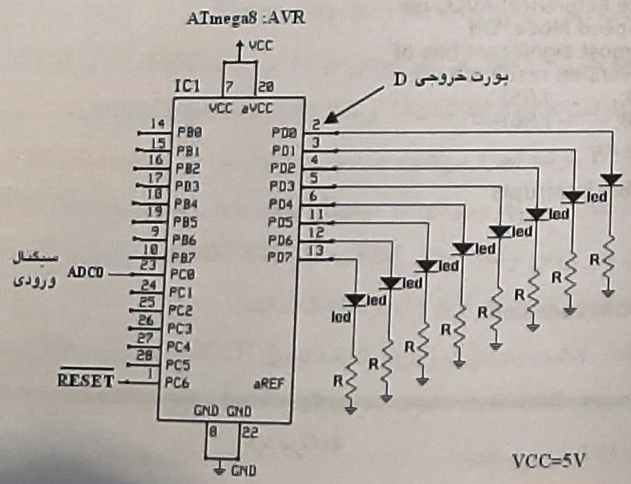
● عبارت (۳) و (۴): پورت D را خروجی می‌کنند.

● عبارت (۵): مقدار 01100000 را در ثبات ADMUX قرار می‌دهد. در این صورت مطابق شکل (۸-۴) باعث می‌شود:

◆ ولتاژ AVCC به‌عنوان ولتاژ مبنا انتخاب شود.

◆ هشت بیت پرارزش‌تر ADCH ، برای نتیجهٔ تبدیل سیگنال آنالوگ به دیجیتال انتخاب شود و کانال 0 سیگنال آنالوگ ورودی ADC انتخاب گردد.

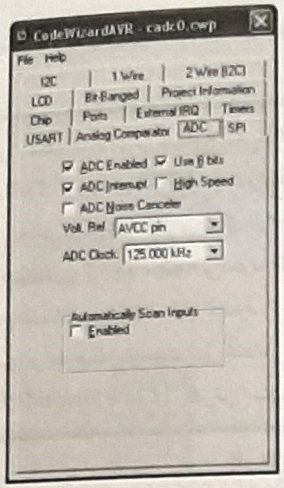
الف: شکل میکروکنترلر



نام فایل: cadc0.c

```
/*****************************************************
Date      : 2008/08/19
Author    : srazi
Company   :
Comments:


Chip type           : ATmega8
Program type        : Application
Clock frequency     : 1.000000 MHz
*****************************************************/

#include <mega8.h> //(1)

// ADC interrupt service routine
interrupt [ADC_INT] void adc_isr(void)
{

// Read the 8 most significant bits
// of the AD conversion result
// Place your code here
PORTD=ADCH;          //(2)
}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out
//Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0
//State2=0 State1=0 State0=0
PORTD=0x00;          //(3)
DDRD=0xFF;           //(4)

// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC High Speed Mode: Off
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=0x60;          //(5)
ADCSRA=0xEB;         //(6)
SFIOR&=0xEF;

// Global enable interrupts
#asm("sei")          //(7)

while (1)
    {
    // Place your code here

    };
}
```

ب: برنامه



ج: تنظیمات ابزار CodeWizardAVR برای ADC و برنامه cadc0.c

شکل (۸-۸) برنامه به زبان C در میکروکنترلر AVR: ATmega8 تبدیل سیگنال آنالوگ ورودی ADC0 به دیجیتال با روش Free Running

- عبارت (۶): مقدار 11101011 را به ثبات کنترل و وضعیت ADCSRA بار می‌کند، در این صورت مطابق شکل (۸-۳- الف) باعث می‌شود که:
- مبدل ADC فعال شود.
- مبدل ADC شروع به کار کند.
- مبدل ADC به‌صورت Free Running کار کند.
- وقفهٔ ADC فعال شود.
- و فرکانس پالس ساعت مبدل ADC مطابق جدول (۸-۱) برابر فرکانس پالس ساعت میکروکنترلر تقسیم بر 8 شود، یعنی برابر 125 KHz=1000 KHz/8 گردد.
- عبارت (۷): وقفه کلی I را فعال می‌کند (یعنی بیت I در ثبات SREG را 1 می‌کند).

به این ترتیب پارامترهای اولیهٔ ثبات‌ها توسط CodeVisionAVR داده شده است.

حلقهٔ تکرار While (1) مرتباً اجرا می‌شود و چون دستور در آن نیست، میکروکنترلر منتظر می‌ماند تا اینکه عمل تبدیل سیگنال آنالوگ ورودی به دیجیتال پایان یابد و بیت پرچم وقفهٔ ADIF برابر 1 گردد. در این صورت روتین سرویس وقفهٔ مبدل ADC به نام ADC_INT (در ابتدای برنامه، بعد از عبارت (۱))، اجرا می‌شود که در این روتین، عبارت (۲) نوشته شده است.

- عبارت (۲): مقدار هشت بیت پُرارزش‌تر نتیجه تبدیل ADCH را در پورت D قرار می‌دهد.

به این ترتیب مبدل ADC به‌طور مرتب از سیگنال آنالوگ ورودی کانال 0 یعنی ADC0 نمونه‌برداری می‌کند و در پورت D قرار می‌دهد.

فایل با پسوند HEX برنامه توسط نرم‌افزار PonyProg در میکروکنترلر Program شده و تست گردیده است.

نکته: *برنامهٔ اسمبلی شکل (۸-۶) و برنامهٔ C شکل (۸-۸) هر دو مربوط به یک مسئلهٔ تبدیل سیگنال آنالوگ ورودی ADC0 به دیجیتال می‌باشد، و شکل (۸-۷) فایل HEX برنامه به زبان اسمبلی شکل (۸-۶) و شکل (۸-۹) فایل HEX برنامه به زبان C شکل (۸-۸) می‌باشد. همانطور که ملاحظه می‌شود فایل HEX برنامهٔ C حدود ۳ تا ۴ برابر فایل HEX برنامه اسمبلی می‌باشد، لذا حدود ۳ تا ۴ برابر محل حافظهٔ بیشتری در میکروکنترلر می‌گیرد و به همان نسبت نیز کُندتر می‌شود* ولی نوشتن برنامه به زبان C به کمک ابزار CodeVisionAVR آسان‌تر است.

نام فایل cadc0

```
:0C00000013C0FECFFDCFFCCFFBCFFACF2A
:10000C00F9CFF8CFF7CFF6CFF5CFF4CFF3CFF2CFC0
:10001C0034C0F0CFEFCFEECFEDCF0000F894EE2749
:10002C00ECBBF1E0FBBFEBBFE5BFF8E1F1BDE1BD1F
:10003C008DE0A2E0BB27ED938A95E9F780E094E090
:10004C00A0E6ED930197E9F7E6E2F0E08591959152
:10005C00009761F0A591B59105901590BF01F00145
:10006C0005900D920197E1F7FB01F0CFEFE5EDBFA5
:10007C00E4E0EEBFC0E6D1E005C0EA93E5B1E2BB37
:10008C00E9911895E0E0E2BBEFEFE1BBE0E6E7B900
:10009C00EBEEE6B9E0B7EF7EE0BF7894FFCFFFCF91
:00000001FF
```

شکل (۸-۹) فایل HEX برنامه به زبان C شکل (۸-۸) برای تبدیل سیگنال آنالوگ ورودی ADC0 به دیجیتال با روش Free Running

مثال ۸-۳

برنامه‌ای به زبان C در محیط CodeVisionAVR با میکروکنترلر AVR: ATmega16 ، ATmega32 و... بنویسید که با استفاده از وقفه، سیگنال آنالوگ ورودی ADC0 را به‌صورت Free Running تبدیل به دیجیتال نماید و در پورت D نشان دهد.

حل

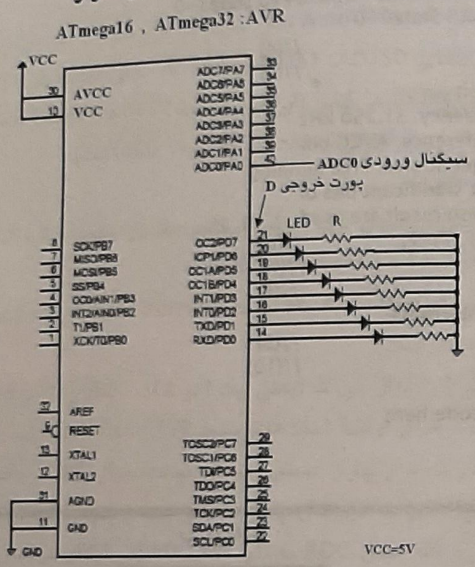
برنامه آن مطابق شکل (۸-۱۰-ب) می‌باشد.

- عبارات (۱) و (۲) و (۴) تا (۱۰): متناسب با انتخاب ما توسط ابزار CodeWizardAVR در نرم‌افزار CodeVisionAVR تولید شده‌اند (شکل (۸-۱۰-ج)) که معنی آن‌ها به شرح زیر می‌باشند.

* چون دارای دستور به زبان ماشین بیشتری است.

- عبارات (۱): باعث می‌شود که از امکانات و پارامترهای میکروکنترلر AVR: ATmega16 ، ATmega32* و... استفاده شود.
- عبارات (۲): متغیر ADC_VREF_TYPE را برابر 0x60 ، یعنی برابر 60 هگزادسیمال تعریف می‌کند که در عبارات (۶) از آن استفاده شده است.
- عبارات (۴) و (۵): پورت D را خروجی می‌کنند.
- عبارت (۶): مقدار 01100000 را در ثبات ADMUX قرار می‌دهد، در این صورت باعث می‌شود:

الف: شکل میکروکنترلر



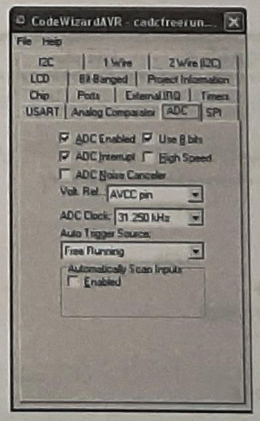
نام فایل cadcfreerun_ATmega16.c

```
*****************************************************
Chip type       : ATmega16
Program type    : Application
Clock frequency : 1.000000 MHz
*****************************************************/
#include <mega16.h>                    //(1)

#define ADC_VREF_TYPE 0x60             //(2)
// ADC interrupt service routine
interrupt [ADC_INT] void adc_isr(void)
{
unsigned char adc_data;
// Read the 8 most significant bits
// of the AD conversion result
adc_data=ADCH;
// Place your code here
```

* اگر میکروکنترلر AVR: ATmega32 ... باشد فایل HeaderFile آنها <mega32.h> ... می‌باشد.

```
PORTD=ADCH;                         //(3)
}
// Declare your global variables here

void main(void)
{
// Declare your local variables here
// Input/Output Ports initialization
// Port D initialization as output
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out
//Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0
//State2=0 State1=0 State0=0
PORTD=0x00;                         //(4)
DDRD=0xFF;                          //(5)
// ADC initialization
// ADC Clock frequency: 31.250 kHz
// ADC Voltage Reference: AVCC pin
// ADC Auto Trigger Source: Free Running
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE;                //(6)
ADCSRA=0xED;                        //(7)
SFIOR&=0x1F;                        //(8)
// Global enable interrupts
#asm("sei")                         //(9)
while (1)                           //(10)
    {
    // Place your code here

    };
}
```

ب: برنامه



ج: تنظیمات CodeWizardAVR برای ADC و برنامه cadcfreerun_ATmega16.c

شکل (۸-۱۰) برنامه به زبان C تبدیل سیگنال آنالوگ ورودی ADC0 به دیجیتال با روش Free Running در میکروکنترلر AVR: ATmega16 ، ATmega32 و...

- ولتاژ AVCC به‌عنوان ولتاژ مبنا انتخاب شود.
- هشت بیت پرارزش‌تر ADCH ، برای نتیجهٔ تبدیل سیگنال آنالوگ به دیجیتال انتخاب شود و کانال 0 سیگنال آنالوگ ورودی انتخاب شود.

● عبارت (۷) : مقدار 11101101 را به ثبات کنترل و وضعیت ADCSRA بار می‌کند، در این صورت باعث می‌شود* که:

- مبدل ADC فعال شود.
- مبدل ADC شروع به کار کند.**
- بیت ADATE یعنی Auto Trigger فعال شود، در این صورت اگر در سه بیت پُرارزش‌تر ثبات SFIOR یعنی بیت‌های ADTS0، ADTS1 و ADTS2 مقدار 000 قرار داده شود، مبدل ADC به شکل Free Running کار می‌کند.* (این عمل توسط CodeVisionAVR به‌صورت SFIOR&=0x1F عمل شده که سه بیت پُرارزش‌تر ثبات SFIOR برابر 000 است).
- وقفه ADC فعال شود.
- فرکانس پالس ساعت مبدل ADC برابر فرکانس پالس ساعت میکروکنترلر تقسیم بر 32 شود یعنی برابر 1000KHz/32 = 31.25KHz گردد.

● عبارت (۸): باعث می‌شود که ADC به‌صورت Free Running کار کند (چون سه بیت پرارزش‌تر ثبات SFIOR برابر 000 است).

● عبارت (۹): وقفهٔ کلی I را فعال می‌کند (یعنی بیت I در ثبات SREG را 1 می‌کند).

به این ترتیب پارامترهای اولیهٔ ثبات‌ها توسط CodeVisionAVR داده شده است. حلقهٔ تکرار while (1) مرتباً اجرا می‌شود و چون دستور در آن نیست، میکروکنترلر منتظر می‌ماند تا اینکه عمل تبدیل سیگنال آنالوگ ورودی به دیجیتال پایان یابد و بیت پرچم وقفهٔ ADIF برابر 1 گردد. در این صورت روتین سرویس وقفهٔ مبدل ADC به نام ADC_INT (در ابتدای برنامه بعد از عبارت (۲)) اجرا می‌شود که در این روتین عبارت (۳) نوشته شده است.

● عبارت (۳): مقدار هشت بیت پُرارزش‌تر نتیجهٔ تبدیل ADCH را در پورت D قرار می‌دهد.

به این ترتیب مبدل ADC به‌طور مرتب از سیگنال آنالوگ ورودی کانال 0 یعنی ADC0 نمونه‌برداری می‌کند، تبدیل به دیجیتال می‌کند و در پورت D قرار می‌دهد.

فایل با پسوند HEX برنامه توسط نرم‌افزار PonyProg در میکروکنترلر Program و تست شده است.

◆ دستورات شیفت Shift و چرخش Rotate به راست

با دستورات شیفت و چرخش می‌توان یک یا چند بیت را به طرف راست شیفت داد که این عمل باعث می‌شود با هر بار شیفت به راست، عدد داخل آن بر 2 تقسیم شود. علاوه بر این با مجموعهٔ شیفت و چرخش به راست می‌توان دو بایت، یعنی یک عدد 16 بیتی را به راست شیفت داد.

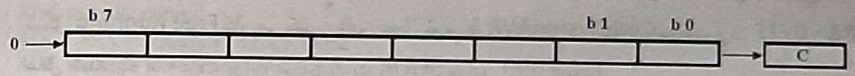
* شکل (۸-۳- ب) و شکل (۸-۳- ج).

** بیت ADSC برای شروع به‌کار مبدل به‌صورت دستی توسط ما برابر 1 گردید.

چون نتیجهٔ تبدیل سیگنال آنالوگ به دیجیتال ADC ، در ده بیت و در دو بایت ثبات‌های ADCL و ADCH قرار می‌گیرد، لذا می‌توان این دو بایت را با دستورات مذکور به راست شیفت داد که از این موضوع در مثال بعدی استفاده خواهیم کرد. بنابراین ابتدا این دستورات را در ذیل مورد بحث قرار می‌دهیم و سپس مثالی از ADC خواهیم داشت.

دستور شیفت منطقی به راست LSR[1]

این دستور بیت‌های ثبات هشت بیتی Rd را یک بیت به طرف راست شیفت می‌دهد. در این صورت کم‌ارزش‌ترین بیت LSB ، یعنی b0 به بیت پرچم نقلی C در ثبات SREG منتقل می‌شود و 0 به پُرارزش‌ترین بیت b7 وارد می‌گردد (شکل ذیل).



این عملیات، عدد داخل ثبات را، بر 2 تقسیم می‌کند. فرم کلی این دستور به‌صورت:

```
LSR Rd
```

می‌باشد که Rd بین R0 تا R31 است. این دستور بر بیت‌های پرچم C ، Z ، V و S اثر می‌گذارد و بیت N برابر 0 می‌شود.

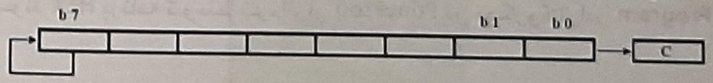
مثال

```
LSR R25
```

اگر داخل ثبات R25 عدد 00000110 باشد، بعد از اجرای دستور فوق عدد داخل ثبات مذکور برابر 00000011 می‌گردد.

دستور شیفت ریاضی به راست ASR[2]

این دستور بیت‌های ثبات هشت بیتی Rd را یک بیت به طرف راست شیفت می‌دهد. در این صورت کم‌ارزش‌ترین بیت LSB ، یعنی b0 به بیت پرچم نقلی C در ثبات SREG منتقل می‌شود و پُرارزش‌ترین بیت b7 تغییری نمی‌کند (شکل زیر):



این عملیات، عدد داخل ثبات را، بر 2 تقسیم می‌کند. فرم کلی این دستور به‌صورت:

```
ASR Rd
```

می‌باشد که Rd بین R0 تا R31 است. این دستور بر بیت‌های پرچم C ، Z ، N ، S اثر می‌گذارد و بیت پرچم N را برابر 0 می‌کند.

1- Logical Shift Right (LSR)

2- Arithmetic Shift Right (ASR)

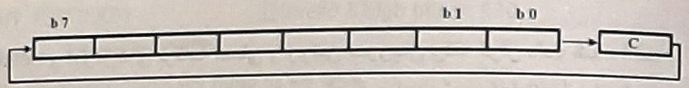
مثال

```
ASR R16
```

در این صورت محتوای ثبات R16 یک بیت به راست شیفت می‌کند و در ضمن عدد مذکور بر 2 تقسیم می‌شود.

دستور چرخش به راست از طریق بیت نقلی ROR[1]

این دستور بیت‌های ثبات Rd را یک بیت به راست شیفت می‌دهد و کم‌ارزش‌ترین بیت ثبات، یعنی b0 را به بیت نقلی C در ثبات SREG منتقل می‌کند و بیت نقلی C را به پُرارزش‌ترین بیت ثبات، یعنی به b7 نیز انتقال می‌دهد. (شکل زیر):



فرم کلی این دستور به‌صورت زیر:

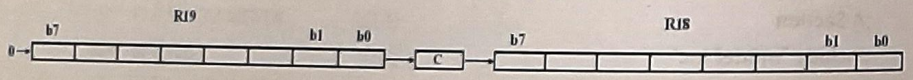
```
ROR Rd
```

می‌باشد که Rd بین R0 تا R31 است. این دستور بر بیت‌های پرچم C ، Z ، N ، V و S اثر دارد.

ترکیب این دستور با دستور شیفت راست LSR ، باعث می‌شود که عدد دوبایتی یک بیت به راست شفیف پیدا کند و بر 2 تقسیم گردد. به‌عنوان مثال:

```
LSR R19
ROR R18
```

مطابق شکل زیر عمل می‌کند که باعث می‌شود عدد دو بایتی داخل جفت ثبات R19:R18 ، یک بیت به راست شیفت داده شوند و بر 2 تقسیم گردد.

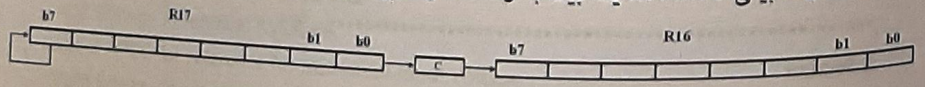


و ترکیب این دستور با شیفت ریاضی به راست ASR باعث می‌شود که عدد دو بایتی علامت‌دار یک بیت به راست شیفت پیدا کنید و بر 2 تقسیم گردد.

مثال

```
ASR R17
ROR R16
```

که عدد شانزده بیتی R17:R16 یک بیت به راست شیفت پیدا می‌کند و بر 2 تقسیم می‌شود (شکل زیر):



1- Rotate Right Through Carry (ROR)

مثال ۸-۴

برنامه‌ای در میکروکنترلر AVR: ATmega8 با دستورات اسمبلی میکروکنترلر و با استفاده از وقفه و روش Single Conversion بنویسید که سیگنال آنالوگ ورودی ADC0 را تبدیل به دیجیتال کند (با امکان تغییر کانال) و در پورت D قرار دهد.

حل

برنامهٔ آن مطابق شکل (۸-۱۱- الف) می‌باشد.

در این برنامه در:

● **بخش A:**

فایل Header File و آدرس‌های برنامهٔ اصلی و روتین سرویس وقفه قرار داده شده است.

◆ فایل Header File اولین عبارت مذکور است که با عبارت ".include "m8def.inc مشخص شده است.

◆ عبارت (۱): پرش به ابتدای برنامه، یعنی به بخش C برنامه است که با برچسب Reset شروع می‌شود.

◆ عبارت (۲): پرش به روتین سرویس وقفهٔ ADC، یعنی پرش به بخش B برنامه است که با برچسب ADC_isr شروع می‌شود.

● در این برنامه روتین سرویس وقفه ADC_isr قبل از بخش C است که به ثبات‌های میکروکنترلر مقدار اولیه داده است، لذا ابتدا بخش C مورد بحث قرار می‌گیرد.

```
;            ADC1.asm          نام فایل
;A Section
.include "m8def.inc"
;single conversion A to D with atmega8
; Interrupt Vector Table
.org  0x000            ;Reset address
      rjmp  Reset      ;(1)

.org  0x00E            ; ADC complete ISR vector
      rjmp  ADC_isr    ;(2)


;*****************************************************
;B Section
;interrupt service routine to read ADC result
ADC_isr:
      in r17,SREG                ;(3) save SREG in R17
```



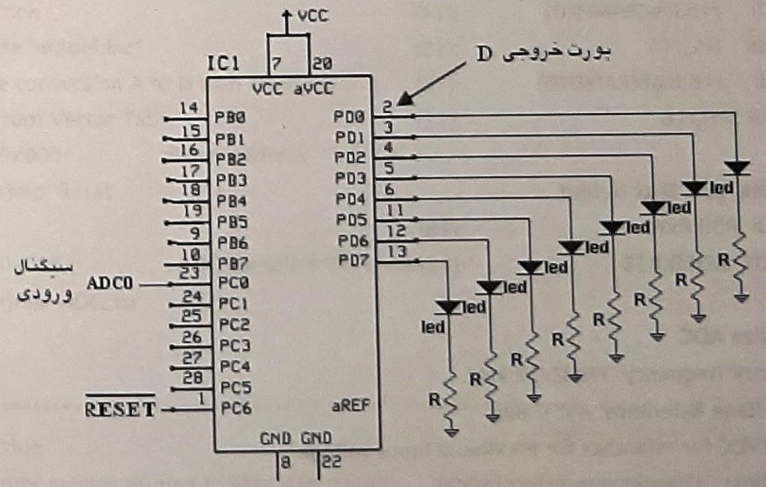
```
;read ADCL & ADCH & put in R31:R30
      IN  R30,ADCL               ;(4)
      IN  R31,ADCH               ;(5)
;shift R31:R30 2 bits to right to delete
; 2 bits of least significant bits of result
      LSR  R31                   ;(6)shift one bit
      ROR  R30                   ;(7)to right R31:R30
      LSR  R31                   ;(8)shift one bit
      ROR  R30                   ;(9)to right R31:R30

      OUT  PORTD,R30             ;(10)read 8 high bits in R30
;              & put in PORTD
;changing channel or doing any thing as we wish
; Start the next AD conversion
      SBI  ADCSRA,6              ;(11)
      out SREG,r17               ;(12)
      RETI                       ;(13)return interrupt
;*****************************************************
;C Section
; *** Begin of Program Execution ***
Reset:
;*****Initialization of stack pointer, port D & ADC*****
;**initialization of the Stack Pointer
      ldi   r16,low(RAMEND)      ;(14)
      out   SPL,r16              ;(15)
      ldi   r16,high(RAMEND)     ;(16)
      out SPH,r16                ;(17)

;**inialize  port D as output
      LDI  R30,0xFF              ;(18)
      OUT  DDRD,R30              ;(19)

;**initialize ADC
;ADC Clock frequency: 1MHZ/32 kHz
;ADC Voltage Reference: AVCC pin
;select AVCC for reference for maximum input voltage
;right adjust , channel zero select (ADC0)
      LDI  R30,0x40              ;(20)
```

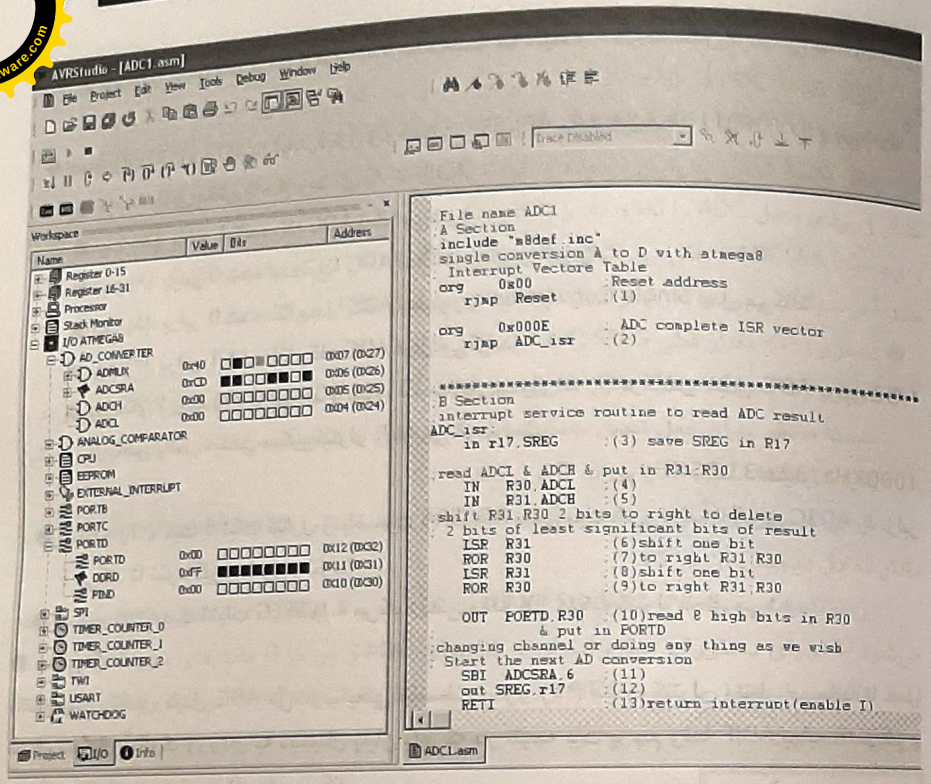
```
        OUT  ADMUX,R30              ;(21)

;Enable ADC,single conversion,ADC interrupt enable
;ADC clock = 1Mga HZ/32
        LDI  R30,0x8D               ;(22)
        OUT  0x06,R30               ;(23) 0x06 is address of ADCSRA

; Start the first AD conversion
        SBI  ADCSRA,6               ;(24)

;Enable global interrupt (I bit ) in SREG
        sei                         ;(25)
;*****************************************************
;D Section
;Endlessloop
loopforever:
;
;waite for interrupt of ADC conversion complete
;
        rjmp loopforever            ;(26)
```

الف : برنامه

ب : شکل میکروکنترلر

ATmega8 :AVR





ج: نتیجه تست برنامه ADC1 در سیمولاتور AVR Studio

شکل (۸-۱۱) برنامهٔ اسمبلی تبدیل سیگنال آنالوگ ورودی ADC0 به دیجیتال

با روش Single Conversion در میکروکنترلر AVR: ATmega8 و...

● بخش C:

به اشاره‌گر پشته (SP) ، پورت D ، ثبات‌های مبدل ADC ، مقدار اولیه داده می‌شود و وقفهٔ کلی I در ثبات SREG فعال می‌گردد.

◆ دستورات (۱۴) تا (۱۷): ثبات اشاره‌گر پُشته (SP) را مقدار اولیه می‌دهد.

◆ دستورات (۱۸) و (۱۹): پورت D را خروجی می‌کنند.

◆ دستوارت (۲۰) و (۲۱): به ثبات انتخاب کانال ADMUX مقدار 01000000 می‌دهد، در نتیجه مطابق بخش (۸-۵) و شکل (۸-۴):

■ بیت‌های (۶) و (۷): برابر 01 قرار داده شده‌اند، لذا ولتاژ مبنا AVCC است که در این حالت AVCC به VCC ، یعنی 5 ولت متصل شده است.

■ بیت (۵): یعنی ADLAR برابر 0 قرار داده شده است، پس مطابق شکل (۸-۲) ده بیت نتیجهٔ تبدیل ADC در طرف راست ثبات‌های داده ADCL و ADCH قرار می‌گیرد.

■ بیت‌های (۰) تا (۳): برابر 0 قرار داده شده است، لذا سیگنال آنالوگ کانال 0 ، یعنی ADC0 برای مبدل ADC انتخاب می‌شود.

◆ دستورات (۲۲) و (۲۳): به ثبات کنترل و وضعیت ADCSRA مقدار اولیهٔ 10001101 داده می‌شود*، در نتیجه مطابق بخش (۸-۴) و شکل (۸-۳- الف):

■ بیت (۷): برابر 1 شده، لذا ADC فعال می‌شود.

■ بیت (۶): برابر 0 شده است، لذا ADC فعلاً نمونه‌گیری نمی‌کند.

■ بیت (۵): برابر 0 شده، لذا مبدل ADC به‌صورت Single Conversion عمل می‌کند.

■ بیت (۳): برابر 1 شده، لذا وقفه ADC فعال می‌گردد.

■ بیت‌های (۰) تا (۲): برابر 101 شده، لذا مطابق جدول (۸-۱) فرکانس مبدل ADC برابر 1/32 فرکانس پالس ساعت میکروکنترلر AVR می‌شود، یعنی

$1000KHz/32 = 31.25KHz$

◆ دستور (۲۴): بیت 6 ثبات کنترل و وضعیت ADCSRA یعنی بیت شروع نمونه‌گیری ADSC را برابر 1 می‌کند تا نمونه‌گیری شروع شود.

◆ دستور (۲۵): بیت I ثبات SREG را 1 می‌کند، یعنی وقفهٔ کلی میکروکنترلر فعال می‌شود.

● **بخش D:**

منتظر پایان تبدیل مبدل ADC می‌شود، یعنی توسط دستور (۲۶) میکروکنترلر منتظر می‌ماند تا عمل تبدیل سیگنال آنالوگ ورودی به دیجیتال پایان یابد که در نتیجه بیت پرچم وقفهٔ ADIF برابر 1 می‌شود و میکروکنترلر به دستور (۲) پرش می‌کند.

● **بخش B:**

روتین سرویس وقفهٔ ADC می‌باشد، یعنی میکروکنترلر در بخش D منتظر پایان عمل تبدیل می‌شود و به محض اینکه عمل تبدیل تمام شد، به‌عبارت (۲) پرش می‌کند و با دستور rjmp ADC_isr به بخش B به روتین سرویس وقفهٔ ADC پرش می‌کند. در این بخش:

◆ دستور (۳): مقدار ثبات SREG** را موقتاً در ثبات r17 ذخیره می‌کند.

◆ دستورات (۴) و (۵): نتیجهٔ تبدیل مبدل ADC که در ثبات‌های ADCL و ADCH قرار دارند را، به ترتیب در ثبات‌های R30 و R31 قرار می‌دهند.

◆ دستوارت (۶) و (۷): مطابق بخش قبلی، محتوای ثبات‌های R31:R30 را یک بیت به طرف راست شیفت می‌دهند.

◆ دستورات (۸) و (۹): نیز مطابق فوق محتوای ثبات‌های R31:R30 را یک بیت به سمت راست شیفت می‌دهند.

* در این دستور به‌جای نام ثبات ADCSRA آدرس آن یعنی 0x06 استفاده شده است.

** برای اینکه بعداً مقدار آن را مجدداً بازیابی کنیم.

در نتیجه محتوای ثبات‌های R31:R30 دو بیت به طرف راست شیفت پیدا می‌کنند، یعنی هشت بیت پُرارزش‌تر نتیجهٔ ADC در ثبات R30 قرار می‌گیرد و دو بیت کم‌ارزش‌تر آن صرف‌نظر شده است.

◆ دستور (۱۰): هشت بیت پُرارزش‌تر تبدیل که در ثبات R30 قرار دارد را در پورت D می‌نویسد.

حال نتیجهٔ تبدیل به پایان رسیده و نتیجه در پورت D نوشته شده است. در این فاصله می‌توان کانال ورودی مبدل ADC را تغییر داد و یا کار دیگری انجام داد. و سپس دستور (۱۱) بیت ۶ ثبات کنترل و وضعیت ADCSRA ، یعنی بیت شروع کار مبدل ADC را فعال می‌کند و مبدل آنالوگ به دیجیتال ADC مجدداً از سیگنال ورودی نمونه‌برداری می‌نماید و عملیات را تکرار می‌کند.

◆ دستور (۱۲): مقدار قبلی SREG را به آن باز می‌گرداند.

◆ دستور (۱۳): برگشت از روتین سرویس وقفه است، یعنی به بخش D و دستور (۲۶) برمی‌گردد و مجدداً منتظر پایان عمل تبدیل سیگنال آنالوگ ورودی به دیجیتال می‌شود، تا نتیجه را در روتین سرویس وقفهٔ بخش B در پورت D قرار دهد.

به این ترتیب مرتباً این برنامه سیگنال آنالوگ ورودی را با روش Single Conversion تبدیل به مقدار معادل دیجیتال آن می‌کند.

این برنامه در سیمولاتور AVR Studio تست گردیده و همانطور که از شکل (۸-۱۱- ج) ملاحظه می‌شود با اجرای دستورات میکروکنترلر به ثبات‌های ADC و پورت D مقدارهای موردنظر داده شده است. علاوه بر این فایل با پسوند HEX برنامه نیز توسط نرم‌افزار PonyProg در میکروکنترلر Program و تست شده است.

مثال ۸-۵

برنامه‌ای با استفاده از وقفه و روش Single Conversion در میکروکنترلر AVR: ATmega16 ، ATmega32 و... با دستورات اسمبلی میکروکنترلرهای AVR بنویسید که سیگنال آنالوگ ورودی ADC0 را تبدیل به دیجیتال کند (با امکان تغییر کانال) و در پورت D قرار دهد.

حل

برنامهٔ آن مطابق شکل (۸-۱۲- الف) می‌باشد.

همانطور که ملاحظه می‌شود، برنامه مذکور همان برنامهٔ شکل (۸-۱۱- الف) می‌باشد فقط:

● نام فایل Header File آن در بخش A برای میکروکنترلر AVR: ATmega16 عبارت "m16def.inc" و جهت AVR: ATmega32 عبارت "m32def.inc" می‌باشد.

● آدرس روتین سرویس وقفه در بخش A برای میکروکنترلرهای AVR: ATmega16 و ATmega32 به ترتیب 0x01C و 0x020 هگزادسیمال است.

بقیه برنامه مشابه برنامه (۸-۱۱- الف) است، لذا برای اطلاعات بیشتر لطفاً برنامهٔ مذکور را مطالعه فرمایید.

```
;                single_convesion_atmega16.asm          نام فایل
;A Section
.include "m16def.inc"
;single conversion A to D with atmega16
; Interrupt Vectore Table
.org  0x000             ;Reset address
      rjmp  Reset       ;(1)

.org  0x01C             ; ADC complete ISR vector
      rjmp  ADC_isr     ;(2)
;*********************************************************
;B Section
;interrupt service routine to read ADC result
ADC_isr:
      in r17,SREG       ;(3) save SREG in R17

;read ADCL & ADCH & put in R31:R30
      IN   R30,ADCL     ;(4)
      IN   R31,ADCH     ;(5)
;shift R31:R30 2 bits to right to delete
; 2 bits of least significant bits of result
      LSR  R31          ;(6)shift one bit
      ROR  R30          ;(7)to right R31:R30
      LSR  R31          ;(8)shift one bit
      ROR  R30          ;(9)to right R31:R30

      OUT  PORTD,R30    ;(10)read 8 high bits in R30
;                & put in PORTD
;changing channel or doing any thing as we wish
; Start the next AD conversion
      SBI  ADCSRA,6     ;(11)
      out SREG,r17      ;(12)
      RETI              ;(13)return interrupt(enable I)
;*********************************************************
;C Section
; *** Begin of Program Execution ***
Reset:
;*****Initialization of stack pointer, port D & ADC*****
;**initialization of the Stack Pointer
      ldi   r16,low(RAMEND) ;(14)
      out   SPL,r16         ;(15)
      ldi   r16,high(RAMEND) ;(16)
      out SPH,r16           ;(17)

;**inialze  port D as output
      LDI  R30,0xFF         ;(18)
      OUT  DDRD,R30         ;(19)

;**initialize ADC
;ADC Clock frequency: 1MHZ/32 kHz
;ADC Voltage Reference: AVCC pin
;select AVCC for reference for maximum input voltage
;right adjust , channel zero select (ADC0)
      LDI  R30,0x40         ;(20)
      OUT  ADMUX,R30        ;(21)
```

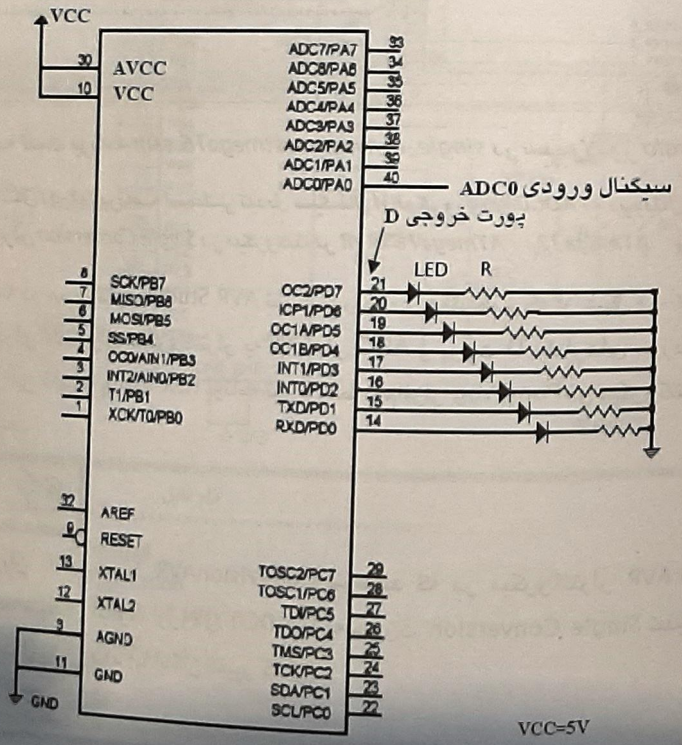


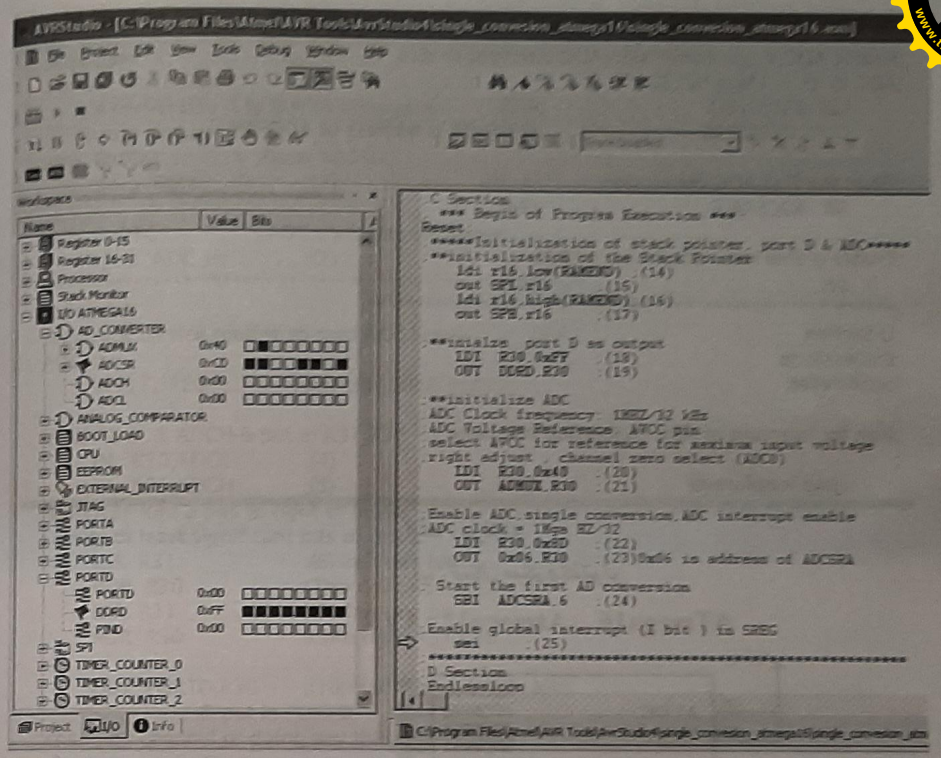
```
;Enable ADC,single conversion,ADC interrupt enable
;ADC clock = 1Mga HZ/32
      LDI  R30,0x8D         ;(22)
      OUT  0x06,R30         ;(23)0x06 is address of ADCSRA

; Start the first ADC conversion
      SBI  ADCSRA,6         ;(24)

;Enable global interrupt (I bit ) in SREG
      sei                   ;(25)
;*********************************************************
;D Section
;Endlessloop
loopforever:
;
;wait for interrupt of ADC conversion complete
;
      rjmp loopforever      ;(26)
```

الف : برنامه

ب : شکل میکروکنترلر

ATmega16 , Atmega32 :AVR





ج: نتیجه تست برنامه single_convesion_atmega16.asm در سیمولاتور AVR Studio

شکل (۸-۱۲) برنامهٔ اسمبلی تبدیل سیگنال آنالوگ ورودی ADC0 به دیجیتال با روش Single Conversion در میکروکنترلر AVR: ATmega16*، ATmega32* و...

این برنامه در سیمولاتور AVR Studio تست گردیده و همانطور که از شکل (۸-۱۲- ج) ملاحظه می‌شود، با اجرای دستورات میکروکنترلر به ثبات‌های ADC و پورت D مقدارهای موردنظر داده شده است. علاوه بر این فایل با پسوند HEX برنامه نیز توسط نرم‌افزار PonyProg در میکروکنترلر Program و تست شده است.

مثال ۸-۶

برنامه‌ای به زبان C در محیط CodeVisionAVR بنویسید که در میکروکنترلر ATmega8:AVR با استفاده از وقفه، سیگنال آنالوگ ورودی ADC0 را به‌صورت Single Conversion تبدیل به دیجیتال نماید و در پورت D نشان دهد (با امکان تغییر کانال).

* در این میکروکنترلرها فقط فایل Header File ، عبارت مربوط به میکروکنترلرهای مذکور می‌باشد که در توضیح برنامه شرح داده شده است.

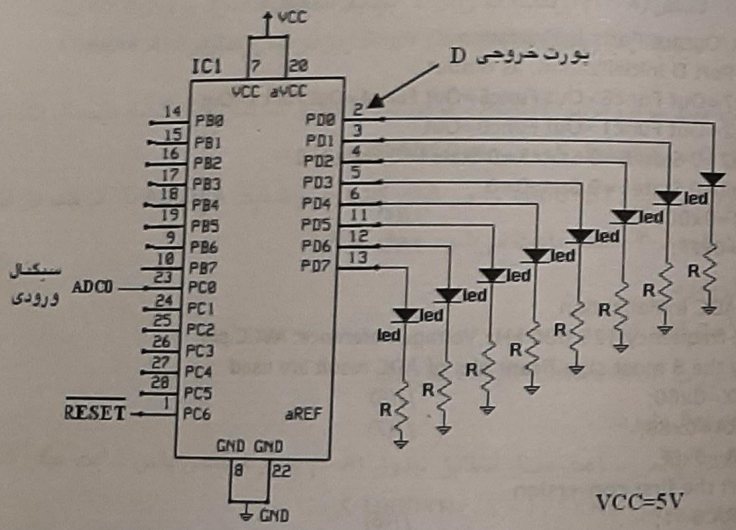
حل

برنامه آن مطابق شکل (۸-۱۳- ب) می‌باشد.

- عبارات (۱) و (۴) تا (۸): متناسب با انتخاب ما توسط ابزار CodeWizardAVR و نرم‌افزار CodeVisionAVR تولید شده‌اند (شکل (۸-۱۳- ج)) که معنی آن‌ها به شرح زیر می‌باشند:
- عبارت (۱): باعث می‌شود که از امکانات و پارامترهای میکروکنترلر AVR: ATmega8 استفاده شود.
- عبارات (۴) و (۵): پورت D را خروجی می‌کنند.
- عبارت (۶): مقدار 01100000 را در ثبات ADMUX قرار می‌دهد. در این صورت مطابق شکل (۸-۴) باعث می‌شود:

ولتاژ AVCC به‌عنوان ولتاژ مبنا انتخاب شود.

الف : شکل میکروکنترلر

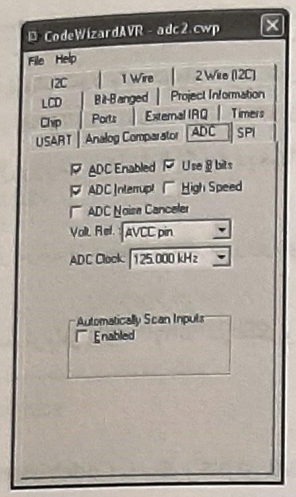


نام فایل adc2.c

```
/*****************************************************
Chip type           : ATmega8
Program type        : Application
Clock frequency     : 1.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/
```

```
#include <mega8.h>                    //(1)

// *************ADC interrupt service routine*************
interrupt [ADC_INT] void adc_isr(void)
{
// Place your code here
// Read the 8 most significant bits of the AD conversion
//result & send the result of ADC to PORTD
PORTD=ADCH;                           //(2)
//changing channel or doing any thing as we wish
// start the next conversion
ADCSRA.6=1;                           //(3)
}
//************************************************************
// Declare your global variables here

void main(void)
{
// Declare your local variables here

//Input/Output Ports initialization
//*****Port D initialization, as output
//Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out
//Func2=Out Func1=Out Func0=Out
//State7=0 State6=0 State5=0 State4=0 State3=0
//State2=0 State1=0 State0=0
PORTD=0x00;                           //(4)
DDRD=0xFF;                            //(5)

//****ADC initialization
//ADC frequency:125.000 kHz,Voltage Reference: AVCC pin
//Only the 8 most significant bits of ADC result are used
ADMUX=0x60;                           //(6)
ADCSRA=0x8B;                          //(7)
SFIOR&=0xEF;
// start the first conversion
ADCSRA.6=1;                           //(8)
// Global enable interrupts
#asm("sei")                           //(9)

while (1)
    {
    // Place your code here

    };
}
```

ب : برنامه



ج: تنظیمات ابزار CodeWizardAVR برای ADC و برنامه adc2.c

شکل (۸-۱۳) برنامه به زبان C ، تبدیل سیگنال آنالوگ ورودی ADC0 به دیجیتال با روش Single Conversion زیر در میکروکنترلر AVR: ATmega8

◆ هشت بیت پُرارزش‌تر ADCH ، برای نتیجهٔ تبدیل سیگنال آنالوگ به دیجیتال انتخاب شود و کانال 0 سیگنال آنالوگ ورودی ADC0 انتخاب شود.

● عبارت (۷): مقدار 10001011 را در ثبات کنترل و وضعیت ADCSRA بار می‌کند، در این صورت مطابق شکل (۸-۳- الف) باعث می‌شود مبدل ADC :

◆ فعال شود.

◆ فعلاً متوقف باشد.

◆ به‌صورت Single Conversion کار کند.

◆ وقفه فعال شود.

و فرکانس پالس ساعت مبدل مطابق جدول (۸-۱) برابر فرکانس پالس ساعت میکروکنترلر تقسیم بر ۸ شود، یعنی برابر $1000KHz/8 = 125KHz$ گردد.

● عبارت (۸): بیت ۶ ثبات کنترل و وضعیت ADCSRA یعنی بیت شروع نمونه‌برداری ADSC را 1 می‌کند تا مبدل ADC شروع به کار کند.

● عبارت (۹): وقفه کلی I را فعال می‌کند (یعنی بیت I در ثبات SREG را 1 می‌کند).

به این ترتیب پارامترهای اولیهٔ ثبات‌ها توسط CodeVisionAVR داده شده است.

حلقه تکرار While (1) مرتباً اجرا می‌شود و چون دستوری در آن نیست، میکروکنترلر منتظر می‌ماند تا اینکه عمل تبدیل سیگنال آنالوگ ورودی به دیجیتال پایان یابد و بیت پرچم وقفهٔ ADIF برابر 1 گردد. در این صورت روتین سرویس وقفهٔ مبدل ADC به نام ADC_INT (در ابتدای برنامه بعد از عبارت (۱)) اجرا می‌شود که در این روتین عبارات (۲) و (۳) نوشته شده‌اند.

● عبارت (۲): مقدار هشت بیت پُرارزش‌تر نتیجه تبدیل ADC را، در پورت D قرار می‌دهد.

برای اینکه نمونه‌گیری مجدداً انجام شود باید دوباره بیت شروع نمونه‌گیری ADSC یعنی بیت 6 ثبات کنترل و وضعیت برابر 1 گردد که این عمل توسط عبارت (۳) انجام می‌شود.*

قبل از عبارت (۳) می‌توان کانال ورودی مبدل آنالوگ به دیجیتال ADC را عوض نمود و یا کارهای دیگری را انجام داد.

به این ترتیب مبدل ADC به‌طور مرتب از سیگنال آنالوگ ورودی کانال 0 یعنی ADC0 نمونه‌برداری می‌کند و در پورت D قرار دهد.

فایل با پسوند HEX برنامه توسط نرم‌افزار PonyProg در میکروکنترلر Progam و تست شده است.

مثال ۸-۷

برنامه‌ای به زبان C در محیط CodeVisionAVR با میکروکنترلر AVR: ATmega16 ، ATmega32 و... بنویسید که با استفاده از وقفهٔ سیگنال آنالوگ ورودی ADC0 را به‌صورت Single-Conversion تبدیل به دیجیتال نماید و در پورت D قرار دهد.

حل

برنامه آن مطابق شکل (۸-۱۴- الف) می‌باشد.

همانطور که ملاحظه می‌شود برنامهٔ مذکور همان برنامهٔ شکل (۸-۱۳- ب) می‌باشد فقط:

● نام فایل Header File آن در میکروکنترلرهای AVR: ATmega16 و ATmega32 به ترتیب عبارت <mega16.h> و <mega32.h> است.

● متغیر ADC_VREF_TYPE در عبارت (۲) برابر 60 هگزادسیمال تعریف شده است که این کار از نظر زیبایی برنامه نوشته شده که در عبارت (۷) نیز استفاده شده است.

لذا برای اطلاعات بیشتر لطفاً برنامه (۸-۱۳- ب) را مطالعه فرمایید.

نام فایل csingle_conversion_atmega16.c

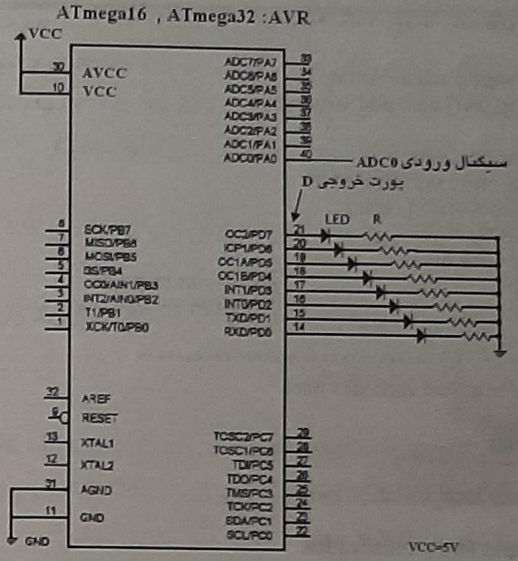
```
/*****************************************************
Chip type           : ATmega16
Program type        : Application
Clock frequency     : 1.000000 MHz
*****************************************************/
//A Section
#include <mega16.h>                    //(1)

#define ADC_VREF_TYPE 0x60             //(2)
```

* در روش Single Conversion بعد از عمل نمونه‌گیری، بیت شروع نمونه‌گیری، یعنی بیت 6 ثبات کنترل و وضعیت ADCSRA مساوی 0 می‌شود.

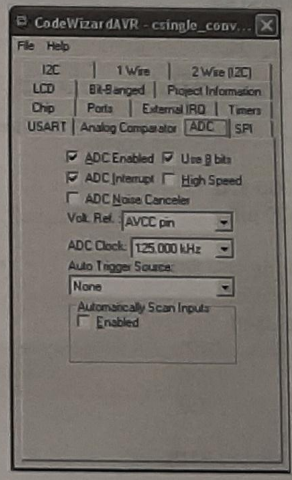

```
//*****************************************************
//B Section
// ADC interrupt service routine
interrupt [ADC_INT] void adc_isr(void)
{
unsigned char adc_data;
// Read the 8 most significant bits
// of the AD conversion result
adc_data=ADCH;
// Place your code here                //(3)result to portd
PORTD=ADCH;                            //(4)start conversion
ADCSRA.6=1;
}
//*****************************************************
// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port D initialization, as output
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out
// Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0
// State2=0 State1=0 State0=0
PORTD=0x00;                            //(5)
DDRD=0xFF;                             //(6)

// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE;                   //(7)
ADCSRA=0x8B;                           //(8)
SFIOR&=0xEF;                           //(9)
//start first conversion
ADCSRA.6=1;                            //(10)
// Global enable interrupts
#asm("sei")                            //(11)

while (1)                              //(12)
    {
    // Place your code here

    };
}
```

الف : برنامه

ب: شکل میکروکنترلر



فایل با پسوند HEX برنامه توسط نرم‌افزار PonyProg در میکروکنترلر Progam و تست شده است.



ج: تنظیمات ابزار CodeWizardAVR برای ADC و برنامه csingle_conversion_atmega16.c

شکل (۸-۱۴) برنامه به زبان C ، تبدیل سیگنال آنالوگ ورودی ADC0 به دیجیتال با روش Single Conversion در میکروکنترلرها AVR: ATmega16 و AVR: ATmega32 و...

* در این میکروکنترلرها فقط فایل Header File یعنی عبارت (۱) مربوط به میکروکنترلرهای مذکور می‌باشد که در توضیح برنامه شرح داده شد.

◆ ۸-۸: پروژه اندازه‌گیری درجه حرارت با سنسور LM35 در میکروکنترلرهای AVR

برای اندازه‌گیری درجهٔ حرارت از قطعات زیر می‌توان استفاده نمود:

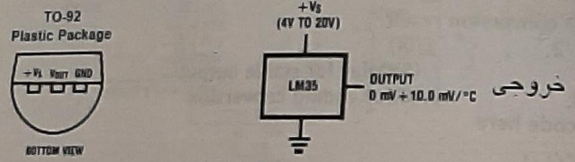
۱. سنسور حرارت LM35 برای تبدیل درجه حرارت به ولتاژ.

۲. میکروکنترلر AVR به‌عنوان مبدل آنالوگ به دیجیتال ADC ...

لذا ابتدا مشخصات سنسور حرارت LM35 موردبحث قرار می‌گیرد.

◆ الف) سنسور حرارت LM35

LM35 یک سنسور حرارت است که ولتاژ خروجی آن متناسب با درجهٔ حرارت می‌باشد و به ازای هر درجهٔ حرارت سانتیگراد، 10 میلی‌ولت، ولتاژ خروجی تولید می‌کند. این سنسور با دقت 1/4 درجه، از 55- تا 150+ درجه کار می‌کند و دارای امپدانس خروجی کم است و فقط به یک منبع ولتاژ نیاز دارد* (شکل ۸-۱۵).



شکل (۸-۱۵) سنسور حرارت LM35

خروجی سنسور حرارت LM35 را می‌توان مستقیماً به ورودی آنالوگ به دیجیتال ADC میکروکنترلرهای AVR متصل نمود (مثال ۸-۸) و شکل (۸-۱۶)) و درجهٔ حرارت را به‌صورت دیجیتال در یکی از پورت‌های AVR و یا در LCD متصل به میکروکنترلرها مشاهده نمود (مثال ۸-۱۰).

◆ ب) برنامه‌های میکروکنترلرها مطابق مثال‌های زیر می‌باشند:

مثال ۸-۸

برنامه‌ای به زبان C در محیط CodeVisionAVR برای میکروکنترلر AVR: ATmega8 بنویسید که درجهٔ حرارت محیط را، توسط سنسور حرارت LM35 گرفته و به‌صورت دیجیتال در پورت D نشان دهد.

حل

برنامهٔ آن مطابق شکل (۸-۱۶- الف) می‌باشد.

* برای اطلاعات بیشتر به Data Sheet آن در CD ضمیمه مراجعه فرمایید.

عبارات (۱) و (۷) تا (۱۳): متناسب با انتخاب ما توسط ابزار CodeWizardAVR و نرم‌افزار CodeVisionAVR تولید شده‌اند (شکل (۸-۱۶- ج)) که معنی آن‌ها به شرح زیر می‌باشند:

- عبارت (۱): باعث می‌شود که از امکانات و پارامترهای میکروکنترلر AVR: ATmega8 استفاده شود.
- عبارت (۲): تابع تأخیر delay.h را در اختیار برنامه قرار می‌دهد تا بتوان تأخیر مناسب را در برنامه تولید کرد.

نام فایل lm35_atmega8.c

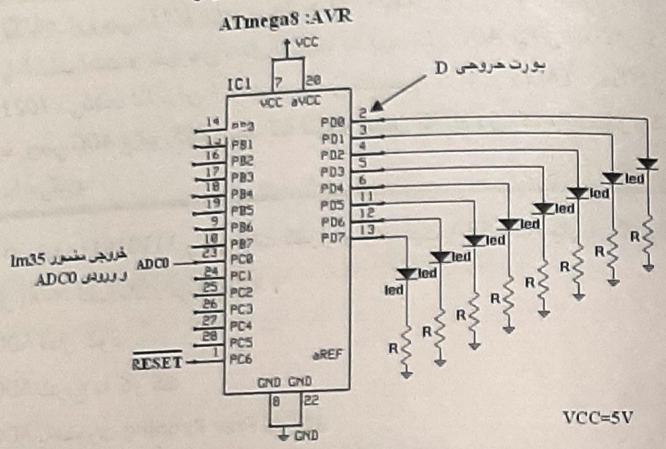
```
/*****************************************************
A Section
Chip type           : ATmega8
Program type        : Application
Clock frequency     : 1.000000 MHz
*****************************************************/
//B Section
#include <mega8.h>                  //(1)
#include <delay.h>                  //(2)
unsigned char value1;               //(3)
// ADC interrupt service routine
interrupt [ADC_INT] void adc_isr(void)
{
// Read the AD conversion result
PORTD=ADCL/2;                       //(4)
delay_ms(30);                       //(5)Delay for stable output
value1=ADCH;                        //(6)for ending conversion
// Place your code here

}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out
// Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0
// State2=0 State1=0 State0=0
PORTD=0x00;                         //(7)
DDRD=0xFF;                          //(8)

// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC High Speed Mode: Off
ADMUX=0x40;                         //(9)
ADCSRA=0xEB;                        //(10)
SFIOR&=0xEF;                        //(11)

// Global enable interrupts
```

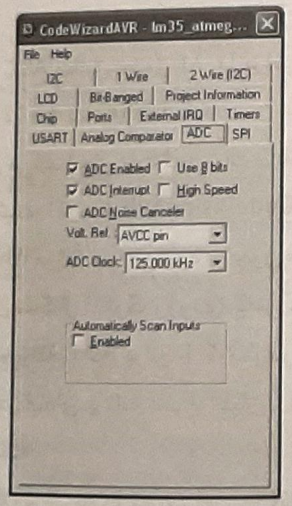


```
#asm("sei")                         //(12)

while (1)                           //(13)
    {
    // Place your code here

    };
}
```

الف: برنامه

ب: شکل میکروکنترلر





ج: تنظیمات ابزار CodeWizardAVR برای ADC و برنامه lm35_atmega8.c

شکل (۸-۱۶) برنامه به زبان C، اندازه‌گیری درجهٔ حرارت با سنسور lm35 و میکروکنترلر AVR : ATmega8 و...

- عبارت (۳): متغیر value1 را تعریف می‌کند تا در عبارت (۶) از آن استفاده شود.

- عبارات (۷) و (۸): پورت D را خروجی می‌کنند.
- عبارت (۹): مقدار 01000000 را در ثبات ADMUX قرار می‌دهد. در این صورت مطابق شکل (۸-۴) باعث می‌شود:
  - ◆ ولتاژ AVCC به‌عنوان ولتاژ مبنا انتخاب شود.
  - ◆ کانال 0 سیگنال آنالوگ ورودی ADC0 انتخاب گردد و هشت بیت کم ارزش‌تر ADCL برای نتیجهٔ تبدیل سیگنال آنالوگ به دیجیتال ADC انتخاب شود.

> **نکته:** خروجی LM35 برای درجهٔ حرارت حداکثر 100 درجهٔ حدود 1000 میلی‌ولت یا 1 ولت است و خروجی مبدل آنالوگ به دیجیتال ADC برای حداکثر 5 ولت برابر 1023 می‌باشد، لذا برای 1 ولت خروجی سنسور حرارت LM35 ، حداکثر عدد مقدار خروجی ADC برابر 205 است، لذا نتیجهٔ تبدیل ADC در یک بایت کوچکتر* ADCL جا می‌گیرد.

- عبارت (۱۰): مقدار 11101011 را در ثبات کنترل و وضعیت ADCSRA بار می‌کند. در این صورت مطابق شکل (۸-۳- الف) باعث می‌شود که:
  - ◆ مبدل ADC فعال شود.
  - ◆ مبدل ADC شروع به کار کند.
  - ◆ مبدل ADC به‌صورت Free Running کار کند.
  - ◆ وقفه ADC فعال شود.

  و فرکانس پالس ساعت مبدل ADC مطابق جدول (۸-۱) برابر فرکانس پالس ساعت میکروکنترلر تقسیم بر ۸ شود، یعنی برابر 1000KHz/8 = 125KHz گردد.
- عبارت (۱۲): وقفهٔ کلی I را فعال می‌کند (یعنی بیت I در ثبات SREG را 1 می‌کند).
- به این ترتیب پارامترهای اولیهٔ ثبات‌ها توسط CodeVisionAVR داده شده است.
- عبارت (۱۳): حلقه تکرار (1) While مرتباً اجرا می‌شود. و چون دستوری در آن نیست، میکروکنترلر منتظر می‌ماند تا اینکه عمل تبدیل سیگنال آنالوگ ورودی، یعنی خروجی سنسور حرارت LM35 ، به دیجیتال پایان یابد و بیت پرچم وقفهٔ ADIF برابر 1 گردد. در این صورت روتین سرویس وقفهٔ مبدل ADC به نام ADC_INT (در ابتدای برنامه بعد از عبارت (۳)) اجرا می‌شود که در این روتین عبارات (۴) تا (۶) نوشته شده است.
- عبارت (۴): مقدار هشت بیت کم‌ارزش‌تر نتیجهٔ تبدیل، یعنی ADCL را بر ۲ تقسیم می‌کند و نتیجه را در پورت D قرار می‌دهد.
- عبارت (۵): تأخیری برابر 30 میلی‌ثانیه تولید می‌کند تا تغییرات پورت خروجی با چشم دیده شود.

* حداکثر مقدار یک بایت مساوی 255 است.

- عبارت (۶): مقدار بایت پرارزش مبدل آنالوگ به دیجیتال ADCH را می‌خواند و در متغیر value1 قرار می‌دهد تا بتوان اطلاعات جدید را از مبدل ADC خواند (البته مقدار value1 استفاده نمی‌شود).

> **نکته:** چون خروجی ADC به ازای یک بیت 5 میلی‌ولت تغییر می‌کند و خروجی LM35 به‌ازای هر درجه، 10 میلی‌ولت تغییر می‌کند. لذا برای هر درجه حرارت، خروجی ADC دو بیت تغییر می‌نماید به‌عبارت دیگر ADC دو برابر مقدار را نشان می‌دهد پس اگر خروجی ADC را بر 2 تقسیم کنیم، مقدار واقعی درجهٔ حرارت در پورت D به‌صورت دیجیتالی قرار می‌گیرد.

به این ترتیب مبدل ADC به‌طور مرتب از سیگنال ورودی، یعنی خروجی سنسور حرارت LM35 نمونه‌برداری می‌کند و در پورت D نشان می‌دهد.

فایل با پسوند HEX برنامه توسط نرم‌افزار PonyProg در میکروکنترلر Program و تست شده است.

مثال ۸-۹

برنامه‌ای به زبان C در محیط CodeVisionAVR برای میکروکنترلر AVR: ATmega16 ، ATmega32 و... بنویسید که درجه حرارت محیط را، توسط سنسور حرارت LM35 گرفته و به‌صورت دیجیتال در پورت B نشان دهد.

حل

برنامهٔ آن مطابق شکل (۸-۱۷- الف) می‌باشد.

همانطور که ملاحظه می‌شود برنامهٔ مذکور همان برنامهٔ شکل (۸-۱۶- الف) می‌باشد.

فقط:

- فایل Header File آن برای میکروکنترلرهای AVR: ATmega16 و ATmega32 به ترتیب به صورت <mega16.h> و <mega32.h> است.
- در این برنامه چون پورت B به‌عنوان خروجی، استفاده شده لذا عبارات (۷) و (۸) برای پورت B نوشته شده‌اند.
- در عبارت (۱۰): چون بیت 5 ، یعنی ADATE برابر ۱ قرار داده شده، لذا این حالت، Auto Trigger است و باید به مقدار ثبات SFIOR مراجعه شود (بخش (۸-۴-۱)).

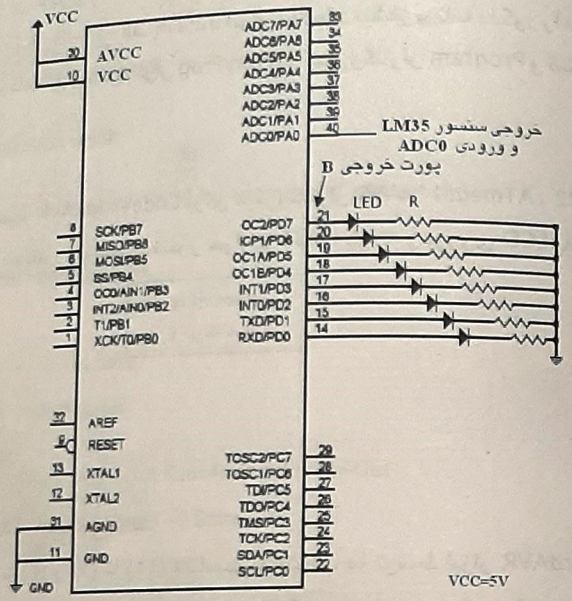
نام فایل: lm35_atmega16_led.c

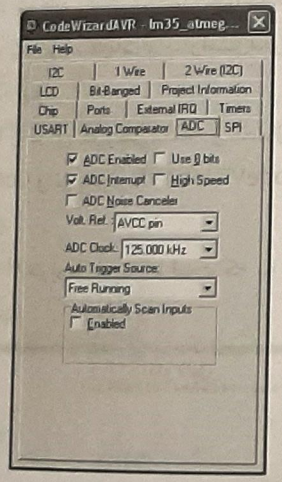
```
/*****************************************************
A section
Chip type           : ATmega16
Program type        : Application
Clock frequency     : 1.000000 MHz
*****************************************************/
//B section
```

```
#include <mega16.h>          //(1)
#include <delay.h>           //(2)
unsigned char value1;        //(3)
//*****************************************************
//C section
// ADC interrupt service routine
interrupt [ADC_INT] void adc_isr(void)
{
// Read the AD conversion result
// Place your code here
 PORTB=ADCL/2;               //(4)
delay_ms(30);                //(5)Delay for stable output
value1=ADCH;                 //(6)for ending c
}
//*****************************************************
// Declare your global variables here
//D section
void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization

// Port B initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out
//Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0
// State2=0 State1=0 State0=0
PORTB=0x00;                  //(7)
DDRB=0xFF;                   //(8)

// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: Free Running
ADMUX=0x40;                  //(9)
ADCSRA=0xEB;                 //(10)
SFIOR&=0x0F;                 //(11)

// Global enable interrupts
#asm("sei")                  //(12)

while (1)                    //(13)
    {
    // Place your code here

    };
}
```

الف : برنامه

ب: شکل میکروکنترلر
ATmega16 , ATmega32 :AVR





ج: تنظیمات ابزار CodeWizardAVR برای ADC و برنامه lm35_atmega16.led.c

شکل (۸-۱۷) برنامه به زبان C ، اندازه‌گیری درجهٔ حرارت با سنسور LM35 و میکروکنترلر AVR : ATmega16 ، ATmega32 و...

* در این میکروکنترلرها فقط فایل Header File یعنی عبارت (۱) مربوط به میکروکنترلرهای مذکور می‌باشد که در توضیح برنامه شرح داده شد.

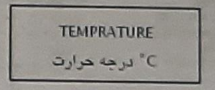
عبارت (۱۱): ثبات SFIOR=0x0F قرار داده شده، در این ثبات چون سه بیت پُرارزش‌تر آن برابر 000 می‌باشد، لذا مطابق جدول (۸-۱- الف) حالت ADC، به‌صورت Free Running می‌باشد.

بقیه برنامه عیناً برنامه شکل (۸-۱۶- الف) است، لذا برای اطلاعات بیشتر برنامه مذکور را مطالعه فرمایید.

فایل با پسوند HEX برنامه توسط نرم‌افزار PonyProg در میکروکنترلر Program و تست شده است.

مثال ۸-۱۰

برنامه‌ای به زبان C در محیط CodeVisionAVR برای میکروکنترلر AVR: ATmega16، ATmega32 و... بنویسید که درجه حرارت محیط را توسط سنسور حرارت LM35 گرفته و برروی LCD به‌صورت:



نشان دهد.

حل

برنامهٔ آن مطابق شکل (۸-۱۸- الف) می‌باشد.

در این برنامه، عبارات (۱) تا (۳) و (۹) تا (۱۲) متناسب با انتخاب ما توسط ابزار CodeWizardAVR و نرم‌افزار CodeVisionAVR تولید شده‌اند (شکل (۸-۱۸- ج)) که معنی آن‌ها به شرح زیر می‌باشند:

- عبارت (۱): باعث می‌شود که از امکانات و پارامترهای میکروکنترلر AVR: ATmega16* و AVR: ATmega32* استفاده شود.
- عبارت (۲): چون ما پورت C را برای خروجی ADC و اتصال به LCD انتخاب کردیم، لذا عبارت (۲) توسط ابزار CodeWizardAVR و نرم‌افزار CodeVisionAVR با آدرس پورت C برای LCD انتخاب شده است (شکل (۸-۱۸- ج)).
- عبارت (۳): فایل Header File مربوط به LCD است که به ما اجازه می‌دهد تا از امکانات و توابع LCD در زبان C استفاده کنیم.

نام فایل: Lm35_lcd_atmega16.c

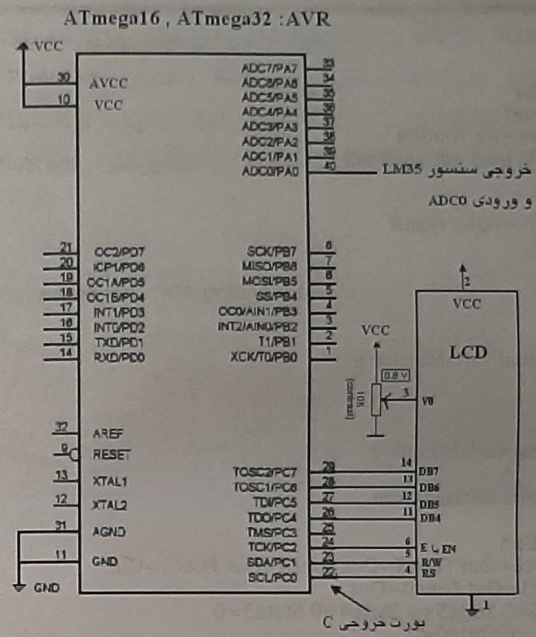
```
/*****************************************************
A Section
Chip type           : ATmega16
Program type        : Application
Clock frequency     : 1.000000 MHz
*****************************************************/
//B Section
#include <mega16.h>                    //(1)
// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC          //(2)
```

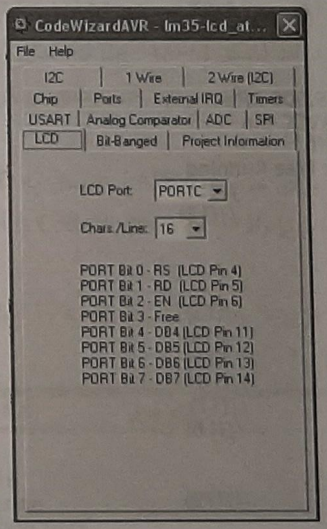
* فایل Header File برای میکروکنترلرهای AVR: ATmega16 و ATmega32 به ترتیب <mega16.h> و <mega32.h> است.


```
#endasm                                //(3)
#include <lcd.h>
                                       //(4)
unsigned char value;                   //(5)
unsigned char value1;
// ADC interrupt service routine
interrupt [ADC_INT] void adc_isr(void)
{
// Read the AD conversion result
value=ADCL;                            //(6)
value/=2;                              //(7)
value1=ADCH;                           //(8)for ending ADC conversion
}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization

// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out
//Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0
// State2=0 State1=0 State0=0
PORTC=0x00;                            //(9)
DDRC=0xFF;                             //(10)

// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: Free Running
ADMUX=0x40;                            //(11)
ADCSRA=0xEB;                           //(12)
SFIOR&=0x0F;

// LCD module initialization
lcd_init(16);                          //(13)
lcd_gotoxy(0,0);                       //(14)
lcd_putsf(" TEMPERATURE ");            //(15)
lcd_gotoxy(8,1);                       //(16)
lcd_putchar(0xDF);                     //(17)
lcd_putchar('C');                      //(18)

// Global enable interrupts
#asm("sei")
                                       //(19)
while (1)
    {                                  //(20)
    lcd_gotoxy(5,1);
    lcd_putchar(value/10+48);          //(21)
    lcd_putchar(value%10+48);          //(22)
    };                                 //(23)
}
```

الف: برنامه

ب: شکل میکروکنترلر





ج: تنظیمات ابزار CodeWizardAVR برای LCD و برنامه lm35_lcd_atmega16.c

شکل (۸-۱۸) برنامه به زبان C ، در محیط CodeVisionAVR برای نشان دادن درجهٔ حرارت توسط سنسور LM35 و LCD با میکروکنترلر AVR: ATmega16 ، ATmega32 و...*

* در این میکروکنترلرها فقط فایل Header File یعنی عبارت (۱) مربوط به میکروکنترلرهای مذکور می‌باشد که در توضیح برنامه شرح داده شد.

- عبارات (۹) و (۱۰): پورت C را خروجی می‌کنند.*
- عبارت (۱۱): مقدار 01000000 را در ثبات ADMUX قرار می‌دهند. در این صورت مطابق شکل (۸-۴-الف) باعث می‌شود:
  - ولتاژ AVCC به‌عنوان ولتاژ مبنا انتخاب شود.
  - هشت بیت کم ارزش‌تر ADCL برای نتیجهٔ تبدیل سیگنال آنالوگ به دیجیتال ADC انتخاب شود**
  - و کانال 0 سیگنال آنالوگ ورودی ADC0 انتخاب شود.
- عبارت (۱۲): مقدار 11101011 را در ثبات کنترل و وضعیت ADCSRA قرار می‌دهد. در این صورت مطابق شکل (ب-۸-۳) باعث می‌شود که:
  - مبدل ADC فعال شود.
  - مبدل ADC شروع به کار می‌کند.
  - مبدل ADC به‌صورت Free Running کار کند.
  - وقفه ADC فعال شود.

و فرکانس پالس ساعت مبدل ADC مطابق جدول (۸-۱) برابر فرکانس پالس ساعت میکروکنترلر تقسیم بر 8 شود، یعنی برابر 1000KHz/8 = 125KHz گردد.

- عبارات (۱۳) تا (۱۸) برای مقدار اولیه دادن به LCD است. در این صورت:
  - عبارت (۱۳): چون LCD که ما انتخاب کردیم 16 کاراکتری است، لذا توسط ابزار CodeVisionAVR عبارت (۱۳) قرار داده شده که نشانهٔ LCD ، 16 کاراکتری است.
  - عبارت (۱۴): توسط تابع ( )lcd_gotoxy ، مکان‌نما یا نشانهٔ LCD را، به ستون 0 و سطر 0 می‌برد.
  - عبارت (۱۵): توسط تابع lcd_Putsf(" "). کلمهٔ TEMPRATURE را برروی LCD می‌نویسد.
  - عبارت (۱۶): توسط تابع ( ) lcd_gotoxy مکان‌نمای LCD را به ستون 8 و سطر ۱ می‌برد.
  - عبارت (۱۷): کد اسکی علامت □ (یعنی 0xDF) برای درجه حرارت در تابع lcd_putchar قرار داده شده تا این علامت برروی LCD نوشته شود.
  - عبارت (۱۸): کد اسکی درجه حرارت سانتیگراد C در تابع lcd_putchar قرار داده شده تا حرف C برروی LCD نوشته شود.

به این ترتیب در LCD عبارت:

TEMPRATURE
°C درجه حرارت

نوشته می‌شود.

- عبارت (۱۹): بیت وقفهٔ کلی I را فعال می‌کند (یعنی بیت I در ثبات SREG را 1 می‌کند).

* بیت‌های پورت C توسط تابع lcd در موقع لزوم ورودی خروجی می‌شوند.

** در مثال (۸-۸) بحث شد.

● عبارت (۴): چون برای نمایش اطلاعات برروی LCD نیاز به یک متغیر داریم، لذا توسط عبارت (۴) متغیر value در ابتدای برنامه به‌صورت هشت بیتی (unsigned char value) تعریف کرده‌ایم.

● عبارات (۵) و (۸): چون باید ثبات ADCH را خواند، تا بتوان اطلاعات جدیدی را از ADC خواند، لذا توسط عبارت (۵) متغیر value1 تعریف شده و توسط عبارت (۸) مقدار ADCH خوانده و در آن قرار گرفته است (البته این مقدار در برنامه استفاده نشده). به این ترتیب پارامترهای اولیه ثبات‌ها توسط CodeVisionAVR داده شده است.

● عبارت (۲۰): حلقهٔ تکرار (1) While مرتباً اجرا می‌شود. در این حلقه:

● عبارت (۲۱): مکان‌نما را به ستون 5 و سطر 1 می‌برد تا مقدار درجهٔ حرارت در این محل نوشته شود.

● عبارت (۲۲): مقدار value که متغیر برای درجهٔ حرارت است را بر 10 تقسیم می‌کند تا دهگان درجهٔ حرارت حاصل شود و با عدد 48 جمع می‌کند تا کد اسکی دهگان درجه‌حرارت ایجاد شود. در این صورت با تابع lcd_Putchar مقدار دهگان درجه حرارت روی LCD نوشته می‌شود.

● عبارت (۲۳): مقدار value را بر 10 تقسیم می‌کند و باقیماندهٔ آن را حساب می‌کند که این باقیمانده، عدد یکان درجه حرارت است. عدد یکان را با 48 جمع می‌کند تا کد اسکی یکان تولید شود. در این صورت با تابع lcd_putchar مقدار یکان درجهٔ حرارت برروی LCD نوشته می‌شود.

● به این ترتیب عبارات (۲۱) تا (۲۳) مرتباً اجرا می‌شوند. به محض اینکه عمل تبدیل سیگنال آنالوگ ورودی یعنی خروجی سنسور LM35 ، به دیجیتال پایان یافت، بیت پرچم وقفه ADIF برابر ۱ می‌گردد. در این صورت روتین سرویس وقفهٔ ADC به نام ADC_INT در ابتدای برنامهٔ بعد از عبارت (۵) اجرا می‌شود که در این روتین عبارات (۶)، (۷) و (۸) نوشته شده‌اند.

● عبارت (۶): مقدار هشت بیت کم‌ارزش‌تر نتیجهٔ تبدیل، یعنی ADCL را در متغیر value قرار می‌دهد.

● عبارت (۷): مقدار value را بر 2 تقسیم می‌کند تا مقدار واقعی درجهٔ حرارت حاصل شود* و آن را توسط عبارات (۲۱) تا (۲۳) برروی LCD نمایش می‌دهد.

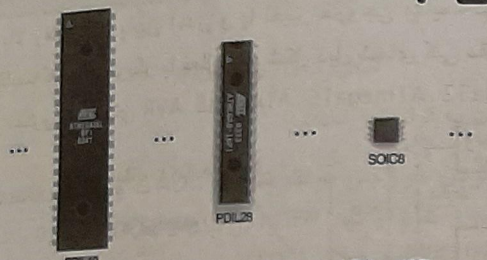
به این ترتیب مبدل ADC به‌طور مرتب از سیگنال ورودی، یعنی خروجی سنسور حرارت LM35 نمونه‌برداری می‌کند و برروی LCD نمایش می‌دهد.

فایل با پسوند HEX برنامه توسط نرم‌افزار PonyProg در میکروکنترلر Program و تست شده است.**

* در نکته مثال (۸-۸) بحث شد.

** برای اتصال پورت C به LCD باید تیک (✓) فیوز بیت JTAGEN را برداشت یا آن را UnProgrammed کرد (فصل ۴ بخش (۴-۱۶))

بخش ب



مقایسه‌کنندهٔ آنالوگ

در میکروکنترلرهای AVR: ATmega8 ،

ATmega16 ، ATmega32 و...