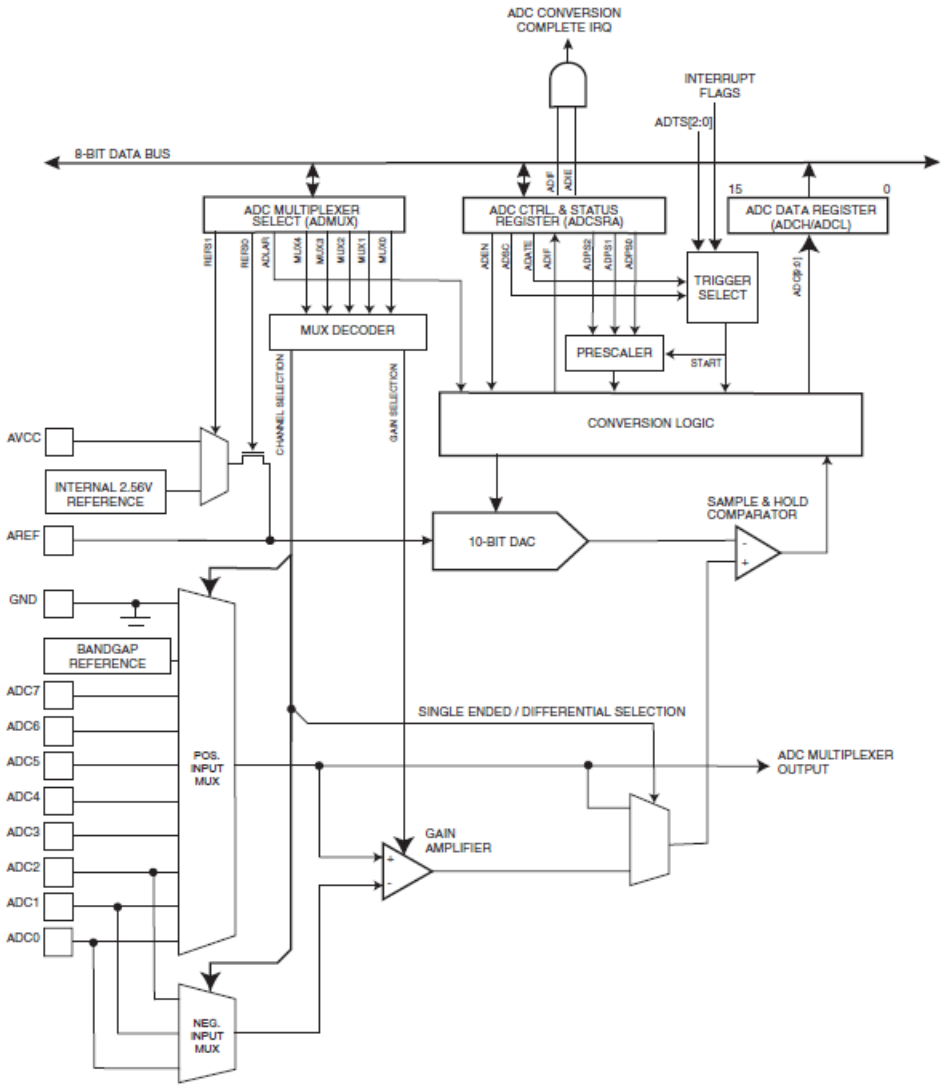
مبدل آنالوگ به دیجیتال (ADC)

میکروکنترلر های خانواده AVR دارای یک مبدل آنالوگ به دیجیتال می باشند که توسط آنها می توان مقدار یک ولتاژ آنالوگ را اندازه گیری کرده و به آن را به یک عدد دیجیتال تبدیل کرد که این یک مزیت بسیار بزرگ برای میکرو به حساب می آید زیرا تمام پارامترهای که می توان از آن در جاهای مختلف استفاده کرد آنالوگ هستند و ما نیازمند یک مبدل برای تبدیل این کمیت های آنالوگ به مقادیر دیجیتال هستیم . مثلا برای اندازه گیری نیرو وزن توسط لودسل، نیروی وزن را به یک ولتاژ آنالوگ تبدیل کرده و آن را از طریق مبدل آنالوگ به دیجیتال به یک دیتای دیجیتالی تبدیل کرده و مقدار ولتاژ را بدست می آوریم . این امکان توسط پین های پورت A که به عنوان ورودی های آنالوگ هم کار می کنند در اختیار ما قرار می گیرد .

تغذیه ADC

برای بالا بردن دقت ADC ولتاژ تغذیه آن را از بقیه سیستم جدا کرده اند تا کاربر بتواند با اعمال یک ولتاژ دقیق ، دقت اندازه گیری را بالا ببرد . پایه تغذیه مبدل در میکرو MEGA16 پایه 30 موسوم به AVCC می باشد و زمین (GND) نیز پایه 31 می باشد .

در زیر مدار داخلی ADC آمده است :



شکل1-9

ولتاژ Refrence در ADC

میکرو برای تبدیل ولتاژ آنالوگ به دیجیتال باید یک مرجع داشته باشد تا بتواند ولتاژهای ورودی را با آن مقایسه کند زیرا میکرو از روش تقریب متوالی برای تبدیل استفاده

می کند و برای این کار نیاز به یک ولتاژ مرجع دارد . توسط روش های زیر که تنظیمات مربوط به آن در code wizared موجود می باشد می توان یک ولتاژ مرجع برای مبدل آنالوگ به دیجیتال انتخاب کرد .

1-مرجع داخلی (int)

در این حالت ولتاژ 2.4 V داخلی میکرو به عنوان مرجع انتخاب می شود .

2- پایه Aref

در میکرو پینی با نام AREF وجود دارد که هر ولتاژی را به آن بدهیم به عنوان مرجع انتخاب می کند این پایه در میکرو های Mega 16 پایه32 میکرو می باشد . البته توجه کنید که ولتاژ اعمال شده به این پایه حداکثر به اندازه ولتاژ تغذیه (5 v) می باشد .

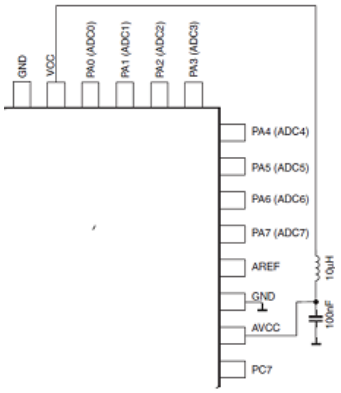
3- AVCC

در این حالت ولتاژ تغذیه میکرو (5 v) به عنوان مرجع انتخاب می شود .

کاهش نویز در ADC

از آنجا که نویز خارجی بر روی ADC بسیار تاثیر گذار است می توان توسط روش های زیر نویز ورودی را کاهش داد .

1- توسط یک فیلتر پایین گذر پایه AVCC را که تغذیه مبدل می باشد را به VCC وصل می کنیم .



شکل2-9

2- استفاده از مدهای sleep مانند idle در مدت زمان تبدیل ، که با استفاده از قابلیت Noise Canceller نویز ناشی از CPU میکروکنترلر را از یبن می برد محقق می شود .

3- کاهش طول سیم های آنالوگ

4- استفاده نکردن از سایر پایه های پورت A با فرکانس سوئیچ بالا در هنگام تبدیل

معرفی چند رجیستر مهم مرتبط با ADC

1-رجیستر 16 بیتی ADCW : شامل دو رجیستر 8 بیتی ADCL و ADCH است و نتیجه تبدیل در این دو رجیستر ریخته می شود . البته توجه کنید که کامپایلر Code Vision نتایج و مقادیر اندازه گیری را مستقیما و به صورت ده بیتی از رجیستر ADCW دریافت می کند .

ADCL (کم ارزش)	ADCH (با ارزش)

2- رجیستر ADCSRA

این رجیستر 8 بیتی به صورت زیر می باشد که در هر یک بیت های آن را توضیح خواهیم داد .

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	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	

ADEN(ADC Enable) :

در صورت یک شدن این بیت ADC روشن و اگر صفر شود ADC خاموش می شود .

ADSC (ADC start convertion) : اگر این بیت یک شود تبدیل شروع می شود .

نکته : در حالت Single running برای هر تبدیل این بیت را 1 می کنیم اما در حالت Free running فقط در تبدیل اول این اتفاق می افتد . باید توجه داشت که با کامل شدن تبدیل این بیت اتوماتیک صفر می شود .

ADATE (ADC auto terriger enable) :

اگر این بیت یک شود عمل تبدیل در حالت terriger قرار می گیرد و در لبه مثبت هر سیگنال تریگر انجام می شود .

ADIF(ADC interrupt flage) :

هر گاه تبدیل کامل شود این بیت یک می شود .

ADIE(ADC interrupt enable) : اگر این بیت یک شود وقفه ADC فعال می شود.

3-رجیستر SFIOR (special function i/o register)

سه بیت با ارزش این رجیستر (ADTS0-2) برای انتخاب یکی از منابع تحریک ADC مطابق جدول زیر می باشد (در صورتی که بیت ADATE از رجیستر ADCSRA یک باشد) .

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	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	

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

جدول 1-9

ضریب تفکیک

توسط این پارامتر دقت مبدل آنالوگ به دیجیتال به دست می آید و از طریق آن می توان عددی را که خروجی مبدل به ما می دهد را بدست آورد . به صورت زیر می توان ضریب تفکیک و عدد خروجی را بدست آورد :

$$\text{عدد خروجی} = \frac{Vin}{\text{ضریب تفکیک}} \qquad \text{ضریب تفکیک} = \frac{Vref}{2^n - 1}$$

در این رابطه n تعداد بیت های خروجی می باشد .

مثلا اگر ولتاژ ورودی یک ولت و ولتاژ مرجع ، داخلی (2.56 V) باشد ضریب تفکیک و عدد خروجی به صورت ده بیتی مانند زیر بدست می آید :

$$\text{ضریب تفکیک} = \frac{2560\,mv}{1024} = 2.5\ \text{mv} \qquad \text{عدد خروجی} = \frac{1v}{2.56\,mv} = 400$$

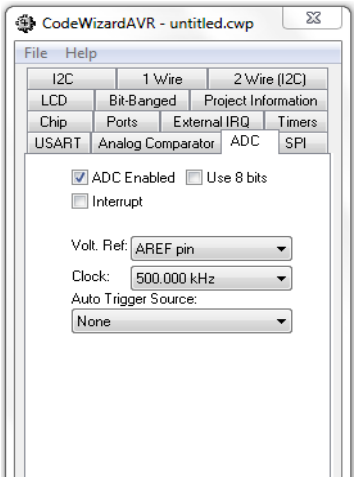
پس اگر ما یک ولتاژ یک ولت را به یکی از پایه های مبدل بدهیم عدد 400 در رجیسترهای ADCL و ADCH قرار می گیرد و اگر بخواهیم عدد اندازه گیری شده به درستی نشان داده شود باید آن را بر 400 تقسیم کرد .

نکته :

در حالت با وقفه می توان اولین و آخرین کانالی را که می خواهیم scan کنیم را انتخاب نماییم .

تنظیمات CodeWizard

در CodeWizard اگر بر روی برگه ADC کلید کنید و گزینه ADC Enable را تیک بزنید ADC شما فعال می شود و توسط گزینه های آن می توانید تنظیمات مورد نظر خود را



شکل3-9

اعمال کنید .

Interrupt : وقفه ADC را فعال می کند .

Use 8 bite : با فعال کردن آن ADC به صورت 8 بیتی کار می کند .

Volt Ref : نوع ولتاژ مرجع را مشخص می کند .

Clock : کلاک ADC را تعیین می کند .

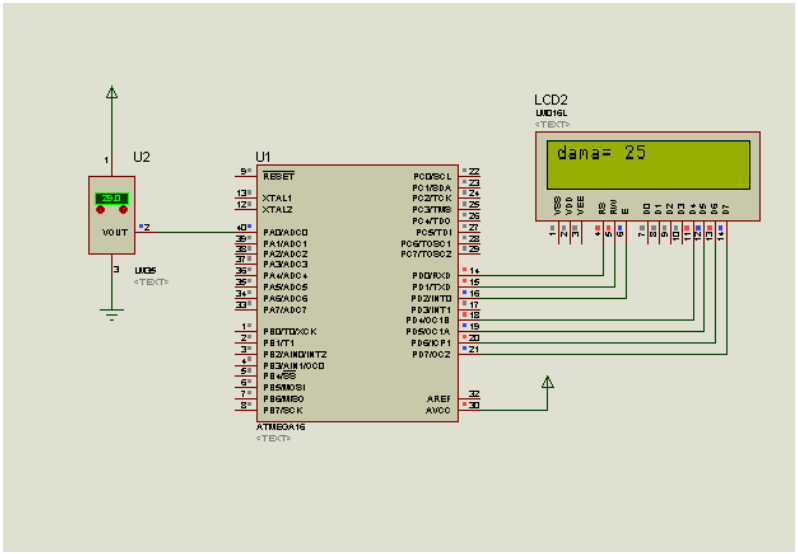
معرفی سنسور دما LM35

LM35 یک سنسور شناخته شده دما می باشد که کاربرد زیادی در صنعت و برای اندازه گیری دماهای پایین دارد . ولتاژ خروجی این سنسور به ازای هر درجه افزایش یا کاهش دما 10 mv تغییر می کند . مثلا ولتاژ خروجی این سنسور در دمای 10 درجه 100 میلی ولت می باشد . پس با اندازه گیری ولتاژ آنالوگی که به ازای دماهای مختلف از این سنسور بدست می آید می توان دما را اندازه گیری کرد .

مثال : به کمک تابع وقفه ADC و سنسور LM35 خروجی کانال آنالوگ را در هر لحظه ببینید .

```
#include <mega16.h>                          //معرفی میکرو مورد استفاده
#include <stdlib.h>                          //فراخوانی کتابخانه stdlib
#asm                                         //شروع برنامه اسمبلی
  .equ __lcd_port=0x1B ;PORTD                //معرفی پورت D به عنوان پورت نمایشگر
#endasm                                      //پایان برنامه اسمبلی
#include <lcd.h>                             //معرفی کتابخانه lcd
```

```
#include <delay.h>                                  //معرفی کتابخانه تاخیر زمانی
unsigned int a; char s[10];                         //معرفی یک متغیر از نوع int جهت ریختن مقدار مبدل در آن
float b;                                            //معرفی یک متغیر از نوع float برای تبدیل عدد خوانده شده به رشته جهت نمایش
void main(void)  {                                  //برنامه اصلی
while (1)    {                                      // حلقه بی نهایت
 a=read_adc(0);                                     //خواندن مقدار مبدل از کانال صفر
 b=(float)a/4;                                      // تقسیم کردن به چهار با توجه به محاسبات
 ftoa(b,2,s);                                       //تبدیل کردن به رشته جهت نمایش
 lcd_gotoxy(0,0);                                   // رفتن به سطر و ستون مورد نظر
 lcd_putsf( =" ");                                  //نمایش کاراکتر دلخواه
 lcd_puts(s);                                       //نمایش عدد خوانده شده
 lcd_putsf("     ") }}                              // پاک کردن حروف باقیمانده از دوره  قبل
```



شکل4-9