دستور کار ۵: مبدل آنالوگ به دیجیتال (ADC)

۱-۱ مقدمه

در این آزمایش هدف راه‌اندازی مبدل آنالوگ به دیجیتال میکروکنترلر AVR می‌باشد. میکروکنترلر ATMEGA32 دارای ۸ کانال ورودی آنالوگ برای مبدل آنالوگ به دیجیتال می‌باشد که برروی پورت A قرار دارند. مبدل موجود در این تراشه دارای رزولوشن ۱۰ بیت بوده و لذا داده خروجی مبدل عددی بین ۰ تا ۱۰۲۳ خواهد بود. برای محاسبه ولتاژ نمونه‌برداری شده می‌توانید از معادله (۰-۱) استفاده نمایید:

معادله (۰-۱)

$$V_{in} / V_{ref} = \frac{Data}{2^n - 1}$$

$$Data = (2^n - 1) * V_{in} / V_{ref}$$

: **V_{in}** همون ولتاژی هست که کانال دریافت می کند و قرار است تبدیل به سیگنال دیجیتال بشود.

: **V_{ref}** ولتاژ مرجع

: **n** دقت یا درجه تفکیک

: **Data** مقدار باینری خروجی مبدل آنالوگ به دیجیتال می‌باشد

توجه داشته باشید که در میکروکنترلر AVR تنها یک مبدل آنالوگ به دیجیتال وجود دارد و برای هر کانال از یک مبدل جدا استفاده نشده است. یعنی اگر ۸ کانال ADC در میکرو ATmega32 هست، به این معنی نیست که ۸ مبدل داریم بلکه این ۸ کانال با یکدیگر مالتی پلکس شدند و در نهایت اون که مورد نیازه به مبدل وصل میشود.

واحد ADC برای انجام عملیات تبدیل نیاز به یه ولتاژ مرجع دارد تا ولتاژ وارد شده رو با آن مقایسه کند. در واقع اگر ولتاژ مرجع رو مثلا ۳ ولت انتخاب کردیم، بازه ۰ تا ۳ ولت به ۱۰۲۴ قسمت تبدیل میشود.

بررسی رجیستر کنترلی : **ADMUX** – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	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	

بیت های **6**و **7**؛ : **REFS1:0 – Reference Selection Bits**

طبق جدول زیر میتوانیم ولتاژ مرجع را انتخاب کنیم:

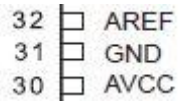
REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

-حالت اول : ولتاژ روی پایه **AREF**به عنوان مرجع انتخاب میشود.

-حالت دوم : ولتاژ روی پایه **AVCC**به عنوان مرجع انتخاب میشود.

-حالت سوم : -?- (رزرو شده)

-حالت چهارم : ولتاژ مرجع داخلی **2.56** ولت تثبیت شده به عنوان مرجع انتخاب میشود. پایه هایAREF و AVCC در شکل زیر نشان داده شده است.



بیت های **0**و **1**و **2**و **3**و **4**؛ : **MUX4:0 – Analog Channel and Gain Selection Bits**

از این بیت ها برای پیکربندی کانال های ADC میکروکنترلر AVR استفاده میشود.

برای اینکه مشخص کنیم که میخواهیم از کدام کانال (PA0-PA7) استفاده کنیم،طبق جدول زیر مقادیر رو انتخاب میکنیم:

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	
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	
10010		ADC2	ADC1	
10011		ADC3	ADC1	
10100		ADC4	ADC1	
10101		ADC5	ADC1	
10110		ADC6	ADC1	
10111		ADC7	ADC1	
11000		ADC0	ADC2	
11001		ADC1	ADC2	
11010		ADC2	ADC2	
11011		ADC3	ADC2	
11100		ADC4	ADC2	

بیت **5**: **ADLAR – ADC Left Adjust Result**

بررسی رجیستر : **ADCSRA** – ADC Control and Status Register A

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	

بیت **7**؛ : **ADEN – ADC Enable**

همونطور که از اسمش پیداست،وظیفه ی فعال/غیر فعال کردن واحد ADC را دارد.

بیت **6**؛ : **ADSC – ADC Start Conversion**

- قبل از اینکه هر عمل تبدیلی انجام بدهید،اول مقدار این بیت رو ۱ کنید.
- بعد از اینکه عمل تبدیل به پایان رسید مقدار این بیت ۰ میشود.

توجه: ADC میکروکنترلر AVR دو مد (Mode) تبدیل داره به نام های Single و Free.

در مد Single بعد از هر عمل تبدیل،ADC منتظر می شود تا دوباره تحریک بشود.که این تحریک شدن توسط برنامه ای که مینویسیم مشخص میشود.

در مد Free واحد ADC شروع به نمونه برداری می کند و بدون توجه به برنامه ای که برنامه نویس نوشته است،دائم ADC رو میخواند و تبدیلاتش رو انجام میدهد و در رجیستر ADCH,ADCL اطلاعات رو ذخیره میکنه.

حالا اگر مد تبدیل Single مدنظر باشد،کافیست در برنامه این بیت را ۱ مقداردهی میکنیم و بعد از اینکه عملیات تبدیل انجام شد خودش اتوماتیک صفر میشه.

ولی اگر مد تبدیل Free را بخواهیم، باید برای شروع حتما این بیت را ۱ کنیم.

برای انتخاب مد هم باید از طریق بیت های ADTS0:2 رجیستر SFIOR اقدام کرد.

بیت ۵ ؛ ADATE – ADC Auto Trigger Enable :

با ۱ شدن این flag،خاصیت تحریک اتوماتیک فعال میشود.

در اینصورت با هرلبه بالارونده پالس اعمال شده به CPU،واحد ADC تحریک میشود.

بیت ۴ ؛ ADIF – ADC Interrupt :

این flag هم معروف به flag وقفه ADC هست و هر وقت که عملیات تبدیل ADC تمام شد این بیت بصورت اتوماتیک ۱ میشود و ما رو مطلع میکند که آیا عملیات تبدیل انجام شده یا نه !

بیت ۳ ؛ ADIE – ADC Interrupt Enable :

۱ بودن این بیت به این معنی است که وقفه ADC فعال است و ۰ بودنش هم حاکی از غیرفعال بودنشه !

بیت های ۰ و ۱ و ۲ ؛ ADPS2:0 – ADC Prescaler Select Bits :

این بیت ها هم برای انتخاب ضریب تقسیم فرکانس واحد ADC هستندکه در زیر توضیح داده شده است.

میکروکنترلرهای AVR (ATmega32) با فرکانس های مختلفی میتونند کار کنند.

فرکانس ADC، قابلیت تنظیم بصورت نرم افزاری وجود دارد.

فرض کنید که فرکانس کاری روی ۱۶ مگاهرتز تنظیم کردیم.این ۱۶ مگاهرتز،فرکانس کاری کلی میکرو هست.

بخش ADC هم برای فعالیت نیاز به فرکانسی دارد که فرکانس خودش را از همین ۱۶ مگاهرتز تامین میکند.

برای اینکه میزان تفکیک پذیری و دقت بالا رود، سازنده میکروکنترلر AVR ناحیه فرکانسی برای فعالیت واحد ADC مشخص کرده که طبق اعلام سازنده باید مقداری بین ۵۰ کیلوهرتز تا ۲۰۰ کیلوهرتز باشد.

سوال این است که چطور ۱۶ مگاهرتز را به حداکثر ۲۰۰ کیلوهرتز برسانیم؟

راهی که خود ATMEL پیش پای ما گذاشته این است که فرکانس کلی را که ما ۱۶ مگاهرتز فرض کردیم به یه عددی تقسیم کنیم که یاین عدد در محدوده ۵۰ تا ۲۰۰ کیلوهرتز باشد.

توجه کنید که اگه بخواهید حداکثر دقت رو تو تبدیل ADC داشته باشید باید فرکانس کاریتون در همان محدوده ۵۰ کیلوهرتز تا ۲۰۰ کیلوهرتز باشد،چون اگه حاصل تقسیم بزرگتر از ۲۰۰ کیلوهرتز باشه باعث میشود که دیگه دقت۱۰ بیت نباشد !

مثلا با فرکانس کاری ۱۶ مگاهرتز و ضریب تقسیم ۱۲۸ داریم :

F_ADC = 16M/128 = 125kHz

پس اینجا ضریب تقسیم ۱۲۸،مقدار مناسبی است،چون عدد حاصل در رنجی که گفتم قرار دارد!

پس میتوانیم مقدار بیتهای این بخش را طبق جدول زیر بصورت ۱۱۱ در نظر بگیریم.

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

رجیسترهای ADCL و ADCH

این رجیسترها حاوی اطلاعات خروجی ADC هستند. یعنی پس از این که تبدیل کامل شد، خروجی دیجیتال در این رجیسترها قرار داده می شود. اما چون در AVR از ADC با رزولوشن ۱۰ بیت استفاده می شود؛ طبیعی است که ۶ بیت بدون استفاده باقی خواهد ماند. اکنون لازم است کاربرد بیت ADLAR را توضیح دهیم. به کمک این بیت می توان نحوه قرار گرفتن داده خروجی در این دو رجیستر را تعیین کرد. اگر این بیت یک باشد خروجی به صورت شکل زیر است:

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	

و اگر صفر باشد به صورت تصویر زیر خواهد بود.

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	

اگر ADLAR=1 باشه نتیجه تبدیل بصورت تنظیم از چپ و اگر ADLAR=0 باشه بصورت تنظیم از راست اطلاعات درون رجیسترها قرار میگیرد !

رجیستر SFIOR - Special Function IO Register

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	

در کل ما زیاد کاری به این رجیستر در مبحث ADC نداریم.ولی اگر بیت ADATE از رجیستر ADSRA رو ۱ کردید،تبدیل ها با یک لبه صعودی تحریک کننده انجام میشود!

بیت های ۵ و ۶ و ۷ ؛ ADTS0:2 - ADC Auto Trigger Source :

از طریق این سه بیت و با توجه به جدول زیر مشخص میکنیم که واحد ADC چطور تحریک بشود !

گاهی لازم است که براساس یه رخداد خاصی،ADC شروع به کار کند. مثلا : هرگاه وقفه خارجی اتفاق افتاد،ADC تبدیل رو شروع کند.

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/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

مراحل اجرای آزمایش

۱-۱-۱ نمونه‌برداری سیگنال آنالوگ با ADC

یک پروژه جدید در نرم‌افزار ایجاد نموده و برنامه زیر را به آن اضافه نمایید. توجه شود که از پورت B برای LCD استفاده شده و فرکانس کاری میکرو نیز 1MHz می‌باشد.

```
#include <mega32.h>
#include <delay.h>
#include <alcd.h>
#include <stdio.h>
#include <stdlib.h>

#define ADC_VREF_TYPE 0x00
int Data=0,Data1=0;
char Buf[16];
float Din;
unsigned int read_adc(unsigned char adc_input) {
   ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
   delay_us(10);
   ADCSRA|=0x40;
   while ((ADCSRA & 0x10)==0);
   ADCSRA|=0x10;
   return ADCW;}
void main(void) {
   PORTB=0x00;
   DDRB=0xFF;
   ADMUX=ADC_VREF_TYPE & 0xff;
   ADCSRA=0x83;
   lcd_init(16);
   lcd_clear();
   lcd_puts("No Data");
   delay_ms(2000);
   while (1){
```

```
    Data = read_adc(0);
    if (Data != Data1) {
        Data1 = Data;
        lcd_clear();
        sprintf(Buf,"Data = %d",Data);
        lcd_puts(Buf);
        lcd_gotoxy(0,1);
        Din = Data;
        Din = (Din*5)/1023;
  ftoa(Din,2,Buf);lcd_putsf("volage=");lcd_puts(Buf);
        lcd_puts(Buf);}
    delay_ms(1000);}}
```

در تنظیما فوق ولتاژ مرجع ADC از پایه Vref دریافت شده و چون برابر 5v می‌باشد لذا در محاسبات از عدد 5 برای محاسبه ولتاژ آنالوگ ورودی استفاده شده است.

تمرین. با استفاده از ADC یک ولتمتر، اهم متر و آمپرمتر بسازید و خروجی هریک را در lcd نمایش دهید.