

دانشگاهصنعتیاصفهان

دانشکدهبرقوکامپیوتر

**دستورکار آزمایشگاه طراحی مدارهای واسط**

**آزمایش ششم**

تهیه کننده: مهران صفایانی

پاییز95

**آشنایی با مبدل قیاسی به رقمی و حسگر دما**

## اهداف

## آشنایی با نحوه تبدیل سیگنال قیاسی به رقمی

## استفاده از مبدل قیاسی به رقمی

* **استفاده از حسگر دما LM35**

## مقدمه:

در دنیای خارج از میکرو تمامی مقادیر و سیگنال ها و کمیت ها پیوسته هستند ، اما در دنیای دیجیتال کمیت ها و سیگنال ها شکل گسسته به خود می گیرند . بنابراین برای به کار بردن این مقادیر و کمیت ها در سیستم های دیجیتال ناگریز به استفاده از مبدلی برای تبدیل سیگنال های آنالوگ به دیجیتال می باشیم . در این آزمایش با نحوه تبدیل سیگنالهای آنالوگ به دیجیتال آشنا شده و سپس اطلاعات یک حسگر دما به درون میکروکنترلر خوانده می شود.

سنسور ها مقدار یک کمیت آنالوگ را به ولتاژ یا جریان تبدیل میکند، سپس این ولتاژآنالوگ به مبدل آنالوگ به دیجیتال میکرو داده میشود و مبدل آنالوگ به دیجیتال با استفاده از روش هایی مقدار ولتاژ را به کمیت دیجیتال متناظرتبدیل میکند، سـپـس این مقدار دیجیتـال با اعمال ریـاضی به مـقدارعددی متـناظر تبـدیل میشود و رویLCD و 7- segment نمایش داده می شود .

* ویژگی های ADC میکروکنترلر LPC1768 :
* دقت تبدیل ۱۲ بیت
* ۸ کانال ورودی
* قابلیت کاهش توان
* محدوده اندازه گیری بین VREFN وVREFP
* فرکانس نمونه برداری 200KHz
* دارای وضعیت Burst و نرم افزاری (معمولی)
* **بررسی رجیسترها و نحوه استفاده از ADC :**

همانطور که گفته شد از ADC میکرو LPC1768 در دو حالت نرم افزاری وBurst می توان استفاده نمود.در حالت نرم افزاری برای خواندن مقدار مورد نظر هربار میکرو تنظیماتی را انجام و سپس مقدار خوانده شده را به ما می دهد.۸ کانال ADC میکروکنترلر LPC1768 به پایه های زیر متصل هستند:



برای استفاده از ADC ابتدا با ید دو کار انجام داد :

1. فعال کردن رجیستر توان مربوط به ADC
2. تنظیمات رجیستر مربوط به PINSEL را در وضعیت ADC قرار دهیم.به عنوان مثال برای استفاده از کانال ADC0.0 تنظیمات زیر را انجام می دهیم:

LPC\_PINCON->PINSEL1 = 1<< 14; // p0.23 select as ADC0.0

LPC\_SC->PCONP |= (1 << 12); // Enable power to AD block

**بررسی رجیستر) ADCR رجیستر کنترل مبدل A/D ( :** بیت های صفر تا ۷ این رجیستر جهت انتخاب کانال ADC استفاده می شود و در مد نرم افزاری تنها یکی از این بیت ها می تواند یک باشد.به عنوان مثال جهت انتخاب کانال صفر یعنی AD0.0 از دستور زیر استفاده می نماییم :

LPC\_ADC->ADCR |= 0x01; //select AD0.0

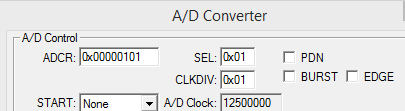
بیت های ۸ تا ۱۵ به نام CLKCIV جهت تنظیم کردن کلاک ADC استفاده می شود. براساس مقداری که به آن می دهیم فرکانس کلاک ADC تقسیم می شود و حداکثرکلاک ADC 13MHz است.رابطه این تقسیم در زیر آمده است :

CLKDIV=(PCLK/ADCLK)-1

ADCLK=PCLK/(CLKDIV+1)

به عنوان مثال با PCLK=25MHz جهت داشتن فرکانس کلاک ADC ،12.5MHzبیت های ۸ تا ۱۵ به صورت زیر مقدار دهی می شوند:

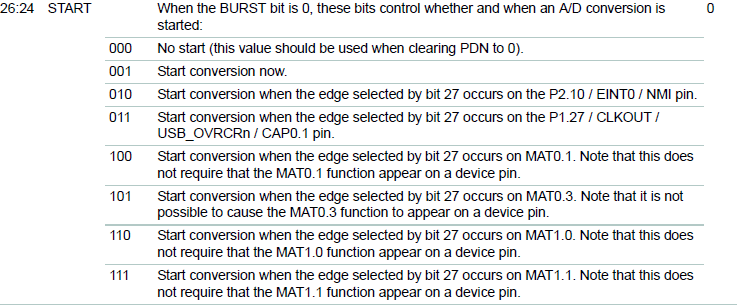
LPC\_ADC->ADCR |= 0x0100; //ADCLK is 12.5 MHz



قسمت مهم بعدی این رجیستر بیت شماره ۲۱ یعنی PDNاست که پاور ADC را فعال می نماید :

LPC\_ADC->ADCR |= 0x200000;

بیت های شماره ۲۴ تا ۲۶ این رجیستر نیز جهت START نمونه برداری استفاده می شوند ، وقتی مقدار 1 را به آن ها دهیم، عملیات تبدیل شروع به اجرا می شود. البته شروع نمونه برداری را توسط قسمت های دیگر مثل پایه P2.10 و یا تایمرها نیز می توان کنترل نمود ، که در تصویرهای زیر لیست قسمت هایی که می توانند عملیات کنترل شروع نمونه برداری را انجام دهند آمده است :



LPC\_ADC->ADCR |= 0x01000000; // Start A/D Conversion

**رجیستر ADGDR :** این رجیستر وضعیت مبدل آنالوگ به دیجیتال را مشخص می نماید و یکی از بیتهای مهم این رجیستر برای ما بیت شماره 31 می باشد که به وسیله این بیت متوجه می شویم که تبدیل انجام شده است یا نه و زمانی که تبدیل کامل شد این بیت یک می شود.

**رجیستر AD0DR0 تا AD0DR8 :** در این رجیستر مقدار دیجیتال سیگنال تبدیل شده در بیتهای ۴ تا ۱۵ قرار می گیرد و برای خواندن ADC از این رجیستر استفاده می نماییم.

i = LPC\_ADC->ADDR0;

**برنامه نمونه اول:** در این برنامه قصد داریم تا کانال AD0 را راه اندازی کرده و سپس مقدار تبدیل شده را برگرداند.

unsigned short ADC\_Read(void)

{

unsigned int i;

LPC\_PINCON -> PINSEL1 = 0x00004000; // p0.23 select as ADC0.0

LPC\_SC->PCONP |= (1 << 12); // Enable power to AD block

LPC\_ADC->ADCR |= 0x01; //select AD0.0

LPC\_ADC->ADCR |= 0x0100; //ADCLK is 12.5 MHz

LPC\_ADC->ADCR |= 0x200000; //Power up,1 << 21

LPC\_ADC->ADCR |= 0x01000000; // Start A/D Conversion

while ((LPC\_ADC->ADGDR & 0x80000000) == 0); // Wait for end of A/D Conversion

i = LPC\_ADC->ADDR0; // Read A/D Data Register

return (i >> 4) & 0x0FFF; // bit 4:15 is 12 bit AD value

}

**برنامه نمونه دوم:** برنامه ای بنوسید که به وسیله سنسور دما lm35 دما را خوانده و بر روی LCD نمایش دهد.

#include <lpc17xx.h>

#define LCD\_LPC1768

#define LCD\_PORT\_2

#define LCD\_RS 0

#define LCD\_RW 1

#define LCD\_E 2

#define LCD\_DB4 4

#define LCD\_DB5 5

#define LCD\_DB6 6

#define LCD\_DB7 7

#include "lcd.h"

void delay(int w){

while(w--);

}

int main(){

int i = 0;

LPC\_GPIO2 -> FIODIR = 0xffffffff;

LPC\_PINCON -> PINSEL1 = 1<<14; // Select P23.0 as ADC

LPC\_SC -> PCONP |= (1<<12); // Enable power to AD block

LPC\_ADC -> ADCR |= 0x01; // Select ADC0.0

LPC\_ADC -> ADCR |= 0x0400; // ADC Clock is 25MHz / 5

LPC\_ADC -> ADCR |= 0x200000; // Power up,1 << 21

while(1){

LPC\_ADC -> ADCR |= (0<<24); // Stop ADC

LPC\_ADC -> ADCR |= (1<<24); // Start ADC

while((LPC\_ADC -> ADGDR & 0x80000000) == 0); // Wait for end of A/D Conversion

i = (LPC\_ADC -> ADGDR & 0xfff0);

i = i / 12.41; //(3.3V / 2^12)=0.0806 mV 10mV / 0.0806 = 12.41 unit

lcd\_clear();

lcd\_gotoxy(1,1);

lcd\_puts(i);

delay(2000000);

}

}

* **دستورکار**

1. برنامه ای بنویسید که هر ده ثانیه یکبار اطلاعات دما را خوانده و بروی نمایشگر نشان دهد
2. برنامه ای بنویسید که در صورت فشردن یک کلید اطلاعات دما بروی نمایشگر بروز رسانی گردد