دانشگاه صنعتی اصفهان
دانشکده برق و کامپیوتر

دستورکار آزمایشگاه طراحی مدارهای واسط

آزمایش سیزدهم

تهیه کننده :مهران صفایانی

پاییز۹۵

آشنایی با پروتکل I2C

❖اهداف:

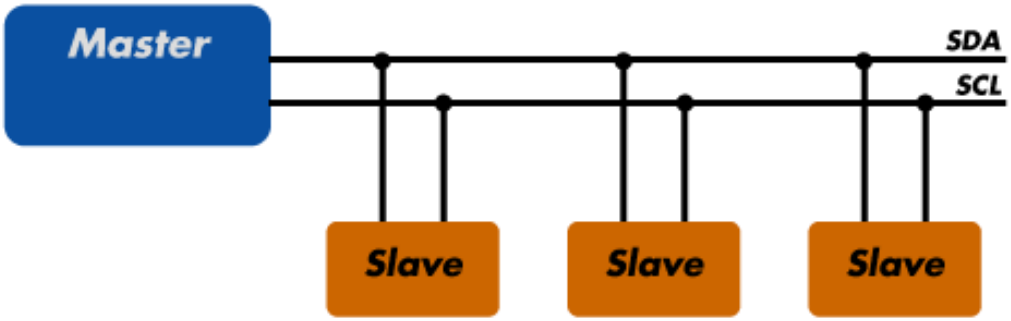
- **آشنایی با پروتکل ارتباطی I2C و امکانات آن**
- **راه اندازی بلوک I2C میکروکنترلر LPC1768**

❖مقدمه:

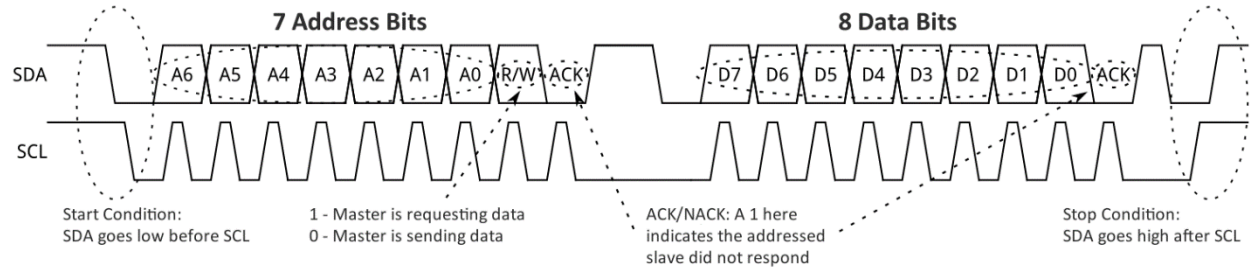
پروتکل ارتباطی I2C که مخفف Inter Integrated Circuit است و نام دیگر آن TWI می باشد ،یک پروتکل ارتباطی جهت ارتباط با وسایل جانبی ، حافظه ها ، سنسورها و... می باشد .این پروتکل توسط شرکت فیلپس معرفی شده است. از قابلیت های این پروتکل برقراری ارتباط از طریق تنها دو سیم می باشد.

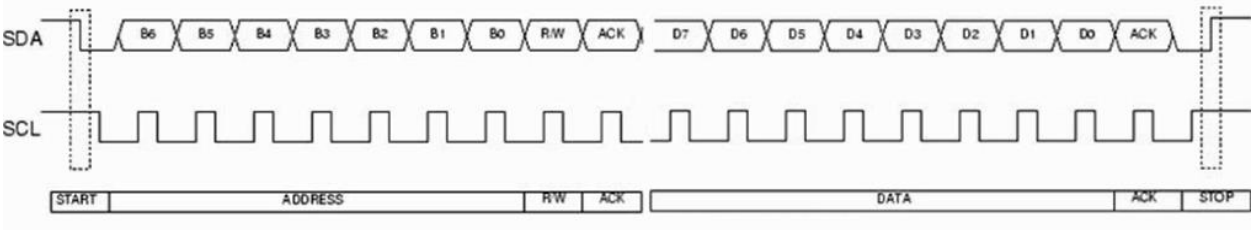
❖واحدI2C میکروکنترلر LPC1768 :

همانطور که در مقدمه ذکر شد از طریق این پروتکل تنها از طریق دو سیم می توان ارتباط سنکرون برقرار نمود که این سیمها به نامهای SDA (Serial Data) وSCL (Serial clock) می باشد.



با توجه به شکل به وسیله این پروتکل یک MASTER می تواند با چند SLAVE ارتباط برقرار نماید.ارسال اطلاعات در پروتکل I2C در ۵ مرحله صورت می پذیرد که ابتدا ۱-شرایط اولیه شروع ۲-ارسال آدرس ۳- تصدیق صحت آدرس ۴-ارسال دیتا ۵-توقف که در زیر شکل مربوط به این مراحل قرار داده شده است.





نکته مهم در رابطه با این پروتکل آدرس می باشد که توسط بیت مورد استفاده قرار می گیرد. ۴ بیت پرارزش برای مشخص کردن وسیله جانبی و ۳ بیت کم ارزش جهت آدرس دستگاه است . پس به وسیله این پروتکل تا ۱۲۸ دستگاه جانبی مختلف و ۸ وسیله یکسان را به باس متصل نمود.

میکروکنترلر LPC1768 دارای ۳ کانال I2C است که باید توسط رجیستر PINSEL در مد SDA وSCL قرار گیرند.

❖رجیسترهای I2C:

رجیسترهای I2C در ۵ گروه اصلی کنترلی ، وضعیت ، کلاک ،آدرس و داده تقسیم می شوند که در زیر شمای کلی آن ها را مشاهده می کنید. به دلیل فراوانی و پیچیدگی های رجیستر های I2C از کتابخانه آماده این واسط برای برقراری ارتباط استفاده می نماییم. در ادامه به بررسی توابع مهم این کتابخانه می پردازیم:

Table 384. I²C register map

Generic Name	Description	Access	Reset value[1]	I²Cn Name & Address
I2CONSET	**I2C Control Set Register.** When a one is written to a bit of this register, the corresponding bit in the I²C control register is set. Writing a zero has no effect on the corresponding bit in the I²C control register.	R/W	0x00	I2C0CONSET - 0x4001 C000 I2C1CONSET - 0x4005 C000 I2C2CONSET - 0x400A 0000
I2STAT	**I2C Status Register.** During I²C operation, this register provides detailed status codes that allow software to determine the next action needed.	RO	0xF8	I2C0STAT - 0x4001 C004 I2C1STAT - 0x4005 C004 I2C2STAT - 0x400A 0004
I2DAT	**I2C Data Register.** During master or slave transmit mode, data to be transmitted is written to this register. During master or slave receive mode, data that has been received may be read from this register.	R/W	0x00	I2C0DAT - 0x4001 C008 I2C1DAT - 0x4005 C008 I2C2DAT - 0x400A 0008
I2ADR0	**I2C Slave Address Register 0.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR0 - 0x4001 C00C I2C1ADR0 - 0x4005 C00C I2C2ADR0 - 0x400A 000C
I2SCLH	**SCH Duty Cycle Register High Half Word.** Determines the high time of the I²C clock.	R/W	0x04	I2C0SCLH - 0x4001 C010 I2C1SCLH - 0x4005 C010 I2C2SCLH - 0x400A 0010
I2SCLL	**SCL Duty Cycle Register Low Half Word.** Determines the low time of the I²C clock. I2nSCLL and I2nSCLH together determine the clock frequency generated by an I²C master and certain times used in slave mode.	R/W	0x04	I2C0SCLL - 0x4001 C014 I2C1SCLL - 0x4005 C014 I2C2SCLL - 0x400A 0014
I2CONCLR	**I2C Control Clear Register.** When a one is written to a bit of this register, the corresponding bit in the I²C control register is cleared. Writing a zero has no effect on the corresponding bit in the I²C control register.	WO	NA	I2C0CONCLR - 0x4001 C018 I2C1CONCLR - 0x4005 C018 I2C2CONCLR - 0x400A 0018
MMCTRL	**Monitor mode control register.**	R/W	0x00	I2C0MMCTRL - 0x4001 C01C I2C1MMCTRL - 0x4005 C01C I2C2MMCTRL - 0x400A 001C
I2ADR1	**I2C Slave Address Register 1.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR1 - 0x4001 C020 I2C1ADR1 - 0x4005 C020 I2C2ADR1 - 0x400A 0020

Table 384. I²C register map

Generic Name	Description	Access	Reset value[1]	I²Cn Name & Address
I2ADR2	**I2C Slave Address Register 2.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR2 - 0x4001 C024 I2C1ADR2 - 0x4005 C024 I2C2ADR2 - 0x400A 0024
I2ADR3	**I2C Slave Address Register 3.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR3 - 0x4001 C028 I2C1ADR3 - 0x4005 C028 I2C2ADR3 - 0x400A 0028
I2DATA_ BUFFER	**Data buffer register.** The contents of the 8 MSBs of the I2DAT shift register will be transferred to the I2DATA_BUFFER automatically after every 9 bits (8 bits of data plus ACK or NACK) has been received on the bus.	RO	0x00	I2C0DATA_ BUFFER - 0x4001 C02C I2C1DATA_ BUFFER - 0x4005 C02C I2C2DATA_ BUFFER - 0x400A 002C
I2MASK0	**I2C Slave address mask register 0**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK0 - 0x4001 C030 I2C1MASK0 - 0x4005 C030 I2C2MASK0 - 0x400A 0030
I2MASK1	**I2C Slave address mask register 1**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK1 - 0x4001 C034 I2C1MASK1 - 0x4005 C034 I2C2MASK1 - 0x400A 0034
I2MASK2	**I2C Slave address mask register 2**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK2 - 0x4001 C038 I2C1MASK2 - 0x4005 C038 I2C2MASK2 - 0x400A 0038
I2MASK3	**I2C Slave address mask register 3**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK3 - 0x4001 C03C I2C1MASK3 - 0x4005 C03C I2C2MASK3 - 0x400A 003C

❖ بررسی توابع مهم کتابخانه I2C

بررسی تابع I2C_Init : این تابع برای مقدار دهی اولیه رجیسترهای واسط I2C استفاده می شود وMaster یا Slave بودن را به عنوان پارامتر دریافت می کند.در زیرکدهای موجود در این تابع را مشاهده می نمایید:

```
uint32_t I2CInit( uint32_t I2cMode )
{
  LPC_SC->PCONP |= (1 << 19);

  /* set PIO0.27 and PIO0.28 to I2C0 SDA and SCK */
  /* function to 01 on both SDA and SCK. */
  LPC_PINCON->PINSEL1 &= ~~0x03C00000;
  LPC_PINCON->PINSEL1 |= 0x01400000;

  /*--- Reset registers ---*/
  LPC_I2C0->I2SCLL   = I2SCLL_SCLL;
  LPC_I2C0->I2SCLH   = I2SCLH_SCLH;
  if ( I2cMode == I2CSLAVE )
  {
     LPC_I2C0->I2ADR0 = 0xA0;         //Set Slave Address
  }

  /* Install interrupt handler */
  NVIC_EnableIRQ(I2C0_IRQn);

  LPC_I2C0->I2CONSET = I2CONSET_I2EN;  //Enable I2C
  return( 1 );
}
```

بررسی تابع I2C_WriteNByte : با استفاده از این تابع می توان مقادیر موجود در یک بافر را بر روی واسط I2C انتقال داد. در زیر امضای تابع ذکر شده را مشاهده می نمایید:

```
uint8_t I2C_WriteNByte(uint8_t sla, uint8_t suba_type, uint32_t suba,
uint8_t *s, uint32_t num);
```

در پارامتر های دریافتی sla همان slave address، suba_type میزان بایت suba ، sub address همان subaddress ، s اشاره گر به بافر و num میزان داده برای نوشتن را مشخص نماید.

بررسی تابع I2C_ReadNByte : با استفاده از این تابع می توان مقادیر موجود در یک بافر را از روی واسط I2C خواند . در زیر امضای تابع ذکر شده را مشاهده می نمایید:

```
uint8_t I2C_ReadNByte (uint8_t sla, uint8_t suba_type, uint32_t suba,
uint8_t *s, uint32_t num);
```

توضیح پارامتر های دریافتی این تابع مانند تابع قبلی می باشد.

برنامه نمونه: در برنامه ی زیر نحوه نوشتن خواندن برروی حافظه EEPROM با استفاده از پروتکلI2C آمده است.

```
#include "lpc17xx.h"
#include "i2c.h"
extern  uint8_t buf[32];
void Delay(uint32_t delaydata){
    uint32_t i,j,k;
    for(i=0;i<delaydata;i++)
        for(j=0;j<1000;j++)
            for(k=0;k<100;k++);
}
int main (void){
    uint32_t i;
    SystemInit();
    LPC_GPIO2->FIODIR    = 0x000000ff;      //LEDs PORT2 are Output
    LPC_GPIO0->FIODIR   |= 0x03f80000;
    LPC_GPIO0->FIOSET    = 0x03f80000;
    if ( I2CInit( (uint32_t)I2CMASTER ) == 0 ){  //initialize I2c
        while ( 1 );                             //Fatal error
    }
    for ( i = 0; i < 8; i++ ){              //clear buffer
        buf[i]=i+1;
    }
    I2C_WriteNByte(0xa0, 1, 0x00, buf, 8);//write buf array ito
    Delay(50);                                EEPROM
    for ( i = 0; i < 8; i++ ){          //clear buffer
        buf[i] =0;
    }
    I2C_ReadNByte (0xa0, 1, 0x00, buf, 8); //read from EEPROM & save
    Delay(50);                             in buff array
    while(1){
        for(i=0;i<8;i++){
            LPC_GPIO2->FIOPIN = buf[i];
            Delay(300);
        }
    }
}
```

❖دستور کار:

I. برنامه بالا یه صورت عملی بروی میکرو ببندید و تست کنید.

II. به وسیله ارتباط I2C دو میکرو را به هم متصل نمایید و میکروی اول پیامی را به میکرو دوم می فرستد و میکرو دوم پیام دریافتی را بروی LCD چاپ و پیام دریافت شده را برای میکرو اول ارسال می کند.