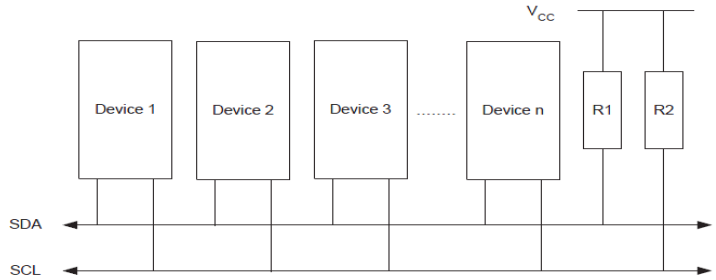
ارتباط دوسیمه i2c

در این بخش قصد داریم که یک نوع از ارتباط پرکاربرد که از آن برای ارتباط بین میکرو با میکروهای دیگر و قطعات دیگر استفاده می شود بپردازیم . این نوع ارتباط برای اولین بار توسط شرکت Philips ارائه گردید .

ارتباط دوسیمه همان طور که از نامش پیداست فقط با دو سیم ایجاد می شود که این مزیتی بزرگ از جهت کاهش سیم های ارتباطی و نویز پذیری می باشد . چنین امکانی سبب سهولت در ارتباط مدارات مجتمع و بسیاری از کارهای صنعتی است که با توجه به کاهش سیم ها و فضای اشغال شده باعث کاهش قیمت محصولات شده و همچنین عیب یابی سیستم نیز راحتر انجام می شود . توسط این ارتباط یک میکرو Master می تواند حداکثر تا 127 وسیله یا Slave را کنترل کند .

نحوه عملکرد i2c

این ارتباط توسط دو پایه SDA و SCL که پایه 23 و 22 میکرو می باشند انجام می شود البته به طوری که در ادامه گفته خواهد شد شما می توانید توسط code vision هر پایه را که مایل هستید به عنوان SDA و SCL انتخاب کنید . شکل این ارتباط به صورت زیر می باشد :

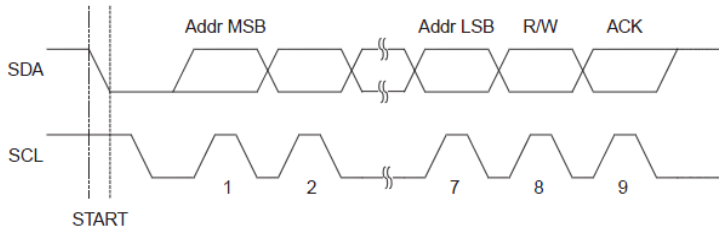


شکل1-12

خط SCL جهت ایجاد ارسال کلاک برای تبادل اطلاعات می باشد که این امر یعنی ایجاد کلاک، توسط میکرو Master انجام می پذیرد در واقع این ارتباط یک ارتباط سنکرون می باشد. خط SDA خط دیتا می باشد توسط این خط Slave فراخوانی شده و اطلاعات بین میکروها رد و بدل می شود. توجه شود که با هر کلاک یک بیت انتقال داده می شود.
حال فرض کنید که میکرو Master می خواهد با یکی از Slave ها ارتباط برقرار کند . ابتدا میکرو Master با ایجاد شرایط شروع ارتباط ، همه Slave ها را از قصد خود برای برقراری ارتباط مطلع می سازد سپس با قرار دادن هفت بیت بر روی باس اعلام می دارد که قصد دارد با کدام Slave تبادل اطلاعات کند.و در بیت هشتم می گوید که قصد دارد در Slave بنویسد یا از آن چیزی بخواند به این صورت که اگر بیت هشتم صفر بود به معنای نوشتن در و اگر یک بود به معنای خواندن است . اگر Slave مورد نظر این پیام را گرفته باشد و از قصد Master آگاه شود ، در کلاک نهم به نشانه تصدیق پیام Master با زمین کردن پایه SDA بیت تصدیق یا Acknowledge به Master ارسال می کند . اگر Slave مورد نظر این عمل را انجام ندهد به معنای آن است که در حال ارتباط با میکرو دیگری است و Master درخواست خود را تا زمانی که به آن پاسخ داده شود مجددا اعلام می دارد . پس از دریافت بیت تصدیق تبادل اطلاعات صورت گرفته و هرگاه کار به اتمام برسد Master شرایط Stop یا قطع ارتباط را ایجاد می کند.

نکته: توجه کنید که اگر Master بخواهد با همه میکرو ها ارتباط بر قرار کند آدرس 0000000 را ارسال خواهد کرد

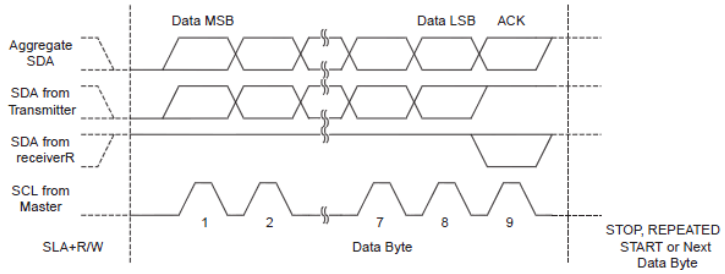
شکل فرمت دیتا در شکل زیر آمده است :



شکل2-12

بسته های دیتا

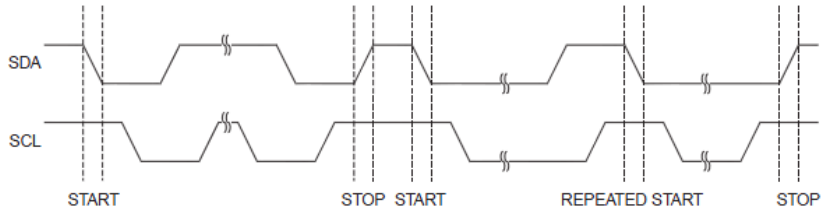
بسته های دیتا به صورت 9 بیتی بوده که 8 بیت آن دیتا و یک بیت هم به عنوان بیت تصدیق می باشد که توسط Slave و با صفر کردن پایه SDA ایجاد می شود .



شکل3-12

اگر میکرو Master بخواهد بدون اینکه خط را از دست بدهد اطلاعات جدیدی را با Slave قبلی یا Slave دیگری تبادل کند باید پس از بیت Stop شرایط شروع را سریعا ایجاد کند .

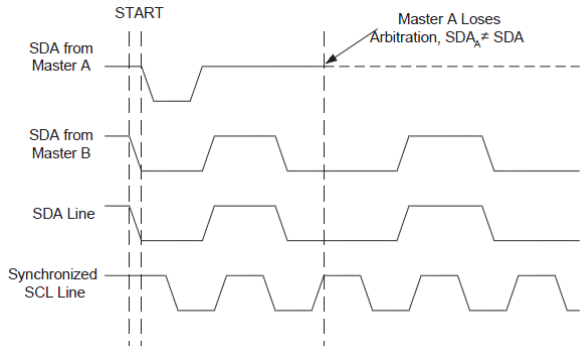
شرایط قطع و وصل مکرر به صورت زیر است :



شکل4-12

نکته : همانطور که در شکل (12_10) آمده باید خط SDA و SCL را توسط دو مقاومت (10 kΩ) به VCC وصل کنیم . دلیل این امر آن است که BUS این پرتکل به صورت Open Drain است و باید توسط دو مقاومت Pull Up شوند .

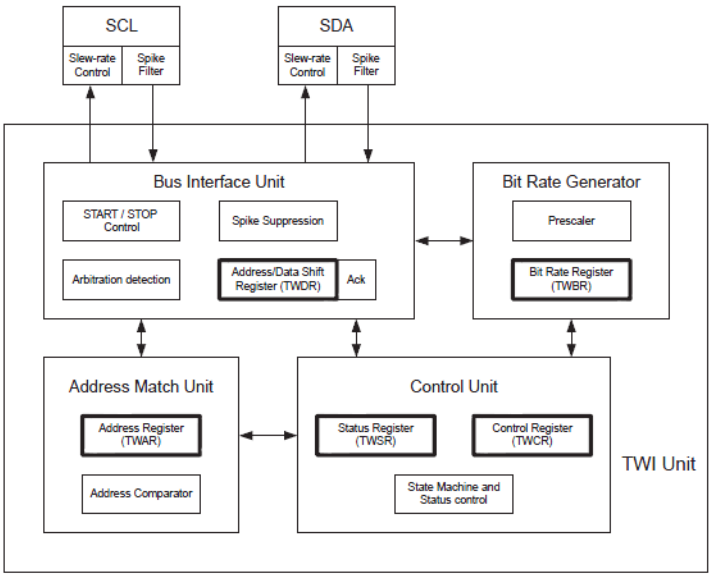
حال فرض کنید که در یک زمان دو میکرو می خواهند به عنوان Master عمل کنند و BUS را در اختیار بگیرند در این حالت میکرویی برنده خواهد بود که زودتر از میکرو دیگر شرایط شروع را ایجاد کند . در این صورت میکرو بازنده باید به حالت Slave رفته و منتظر بماند که آیا توسط Master فراخوانده می شود یا نه و پس از آن منتظر شرایط ایجاد Stop از طرف Master و تلاش دوباره برای برای بدست آوردن BUS باشد . شکل زیر نحوه تلاش دو میکرو برای بدست آوردن خط را نشان می دهد :



شکل12-5

ساختار داخلی i2c

ساختار این قسمت از میکرو در شکل زیر آمده است :



شکل12-6

معرفی کتابخانه i2c.h

توسط این کتابخانه می توان به راحتی از طریق ارتباط دوسیمه با وسایل دیگر همچون حافظه ها و ... ارتباط برقرار کرد . توابع این کتابخانه در زیر آمده است :

void i2c_init(void)

این تابع جهت تنظیمات اولیه i2c می باشد و باید قبل آغاز کار این تابع را نوشت .

unsigned char i2c_start(void)

برای ایجاد حالت شروع می باشد . اگر باس آزاد باشد عدد یک و در غیر اینصورت عدد صفر را بر می گرداند .

void i2c_stop(void)

برای ایجاد شرایط توقف ارتباط می باشد .

unsigned char i2c_read(unsigned char ack)

جهت خواندن یک بایت از طریق باس از این تابع استفاده می شود .

unsigned char i2c_write(unsigned char data)

جهت ارسال یک بایت اطلاعات بر روی باس از این تابع استفاده می شود .

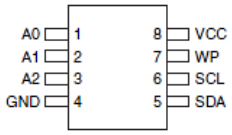
نکته : اگر در دستور i2c_read() در داخل پرانتز عدد صفر قرار دهیم دیگر بیت تصدیق ایجاد نمی شود .

معرفی حافظه های EEPROM سری AT24Cxx

گاهی اوقات در پروژه ها لازم است که اطلاعات خود را بر روی یک حافظه جانبی ذخیره کرد، از همین رو می توان از حافظه AT24Cxx که ارائه شده توسط شرکت ATMEL

می باشد استفاده کرد . این حافظه ها از طریق ارتباط TWI کار می کنند . عددی که به xx می نشیند معرف میزان حافظه قطعه می باشد مثلا 02 به معنای 2kb حافظه است.

Pin Name	Function
A0–A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No-connect



شکل7-12

پایه های A0..2 جهت تعیین آدرس قطعه می باشد . آدرس گذاری در این حافظه به صورت 1010A2A1A0 , R/W می باشد . مثلا 0xa5 یعنی Slave چهارمی که می خواهیم در آن بنویسیم . از تعداد بیت های آدرس معلوم است که ما می توانیم فقط هشت عدد از این حافظه ها را به عنوان Slave استفاده کنیم .

Wp (write protect) : این پایه جهت حفاظت از نوشتن می باشد یعنی اگر آن را یک کنیم فقط عمل خواندن از حافظه صورت خواهد گرفت .

نحوه نوشتن در حافظه

1 – ایجاد حالت شروع به دستور i2c_start()

2 – آدرس Slave با دستور i2c_write(آدرس Slave مورد نظر)

3 – تعیین آدرس خانه حافظه i2c_write(آدرس خانه حافظه)

4 – ارسال داده با دستور i2c_write(دیتای مورد نظر)

5 – پایان ارتباط i2c_stop()

توجه داشته باشید که مرحله چهارم برای EEPROM های یک و دو کیلو بایت می تواند تا هشت مرتبه و بدون تکرار مراحل یک تا سه تکرار شود . این مرحله برای حافظه های 4 ، 8 و 16 کیلو بایت تا شانزده بار قابل تکرار می باشد .

مراحل خواندن از حافظه

1 – ایجاد حالت شروع به دستور i2c_start()

2 – آدرس Slave با دستور (آدرس Slave مورد نظر)i2c_write

3 – تعیین آدرس خانه حافظه (آدرس خانه حافظه)i2c_write

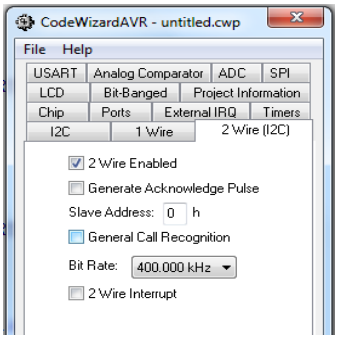
4 – خواندن دیتای مورد نظر با دستور i2c_read()

5 – پایان ارتباط i2c_stop()

توجه داشته باشید که در مرحله دو همانطور که قبلا گفته شد باید بیت آخر ارسالی در حالت خواندن صفر و در حالت نوشتن یک باشد . همچنین تکرار مرحله چهارم در هنگام خواندن همانند حالت نوشتن می باشد .

تنظیمات CodeWizard

در کدویزارد با انتخاب گزینه 2 wire می توانید تنظیمات مورد نظر را انجام دهید . با زدن تیک گزینه 2 Wire Enable ارتباط دو سیمه را فعال می کنید و پنجره ای همانند زیر را مشاهده خواهید کرد .



شکل8-12

Generate Aknowledge Puls :

زدن تیک این گزینه باعث تولید پالس تصدیق خواهد شد .

Slave Address :

اگر میکرو به عنوان Slave کار کند در این قسمت می توانید آدرس او را مشخص کنید در واقع برای میکرو یک نام انتخاب می کنید تا با آن فراخوانده شود.

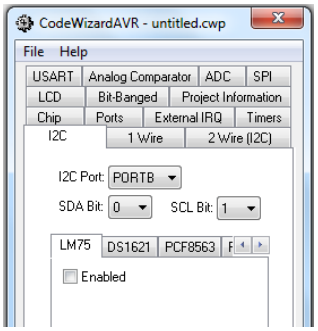
General Call Recognition : جهت فراخوانی عمومی می باشد .

Bit Rate : فرکانس کاری را مشخص می کند .

2 Wire Interrupt : فعال ساز وقفه i2c می باشد .

نکته :

در کد ویزارد برگه دیگری با عنوان i2c موجود می باشد که در آن کتابخانه هایی جهت استفاده راحت از برخی قطعات همچون DS1307 و LM75 موجود می باشد که در صورت تمایل می توانید قطعه مورد نظر را انتخاب کرده ، آدرس و سایر مشخصات آنها را به دلخواه تنظیم نمایید . همچنین در این قسمت این امکان وجود دارد که پرت و پینی را که می خواهید به عنوان SCL و SDA کار کنند را انتخاب کنید .



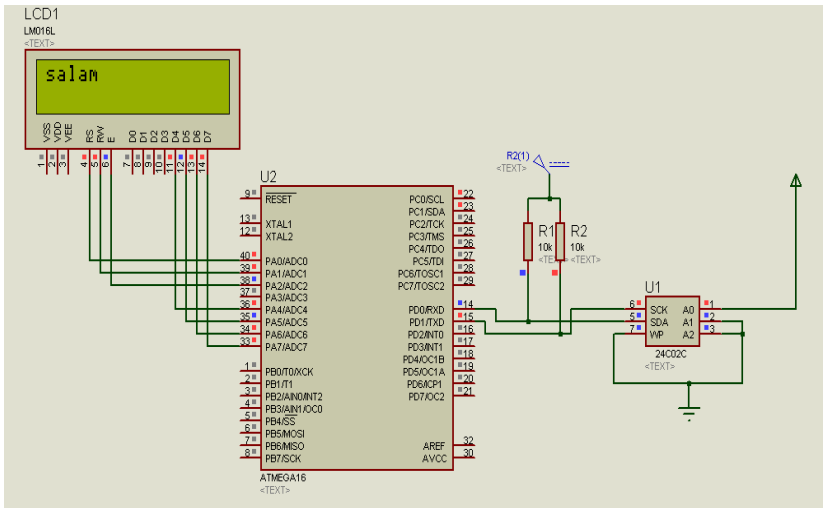
شکل9-12

مثال:کلمه salam را در حافظه 24c02 ذخیره کنید سپس مجددا آنرا خوانده و روی LCD نمایش دهید.

```
#include <mega16.h>                                  معرفی میکرو مورد استفاده//
#include <delay.h>                                   معرفی کتابخانه تاخیر زمانی//
#asm                                                 شروع برنامه اسمبلی//
  .equ __i2c_port=0x12 ;PORTD                        معرفی پورتD به عنوان پرت i2c//
  .equ __sda_bit=0                                   معرفی پین صفر پورتD به عنوان  پایه SDA//
  .equ __scl_bit=1                                   معرفی پین صفر پورتD به عنوان  پایه SCL//
#endasm                                              پایان برنامه اسمبلی //
#include <i2c.h>                                     معرفی کتابخانه ارتباط دوسیمه//
#asm                                                 شروع برنامه اسمبلی//
  .equ __lcd_port=0x1B ;PORT                         معرفی پورتA به عنوان پورت نمایشگر//
#endasm                                              پایان برنامه اسمبلی//
#include <lcd.h>                                     معرفی کتابخانه lcd//
unsigned char a[6],w[6]="salam",i;                   معرفی متغیر ها //
#define  slave_address  0xa2 معرفی آدرس حافظه ای که به عنوان Slave شناخته می شود//
unsigned char eeprom_read(unsigned char address) {   ایجاد یک تابع جهت نوشتن در حافظه //
i2c_start();                                         ایجاد حالت شروع ارتباط//
i2c_write(slave_address);                            فراخوانی Slave//
i2c_write(address);                                  تعیین خانه حافظه که می خواهیم از آن بخوانیم//
```

```
i2c_start();                                                    شروع مجدد//
i2c_write(slave_address | 1);      فهماندن به حافظه که می خواهیم اطلاعاتش را بخوانیم//
for(i=0;i<5;i++)    {
a[i]=i2c_read(0);                                  خواندن از حافظه بدون ایجاد بیت تصدیق//
i2c_stop();                                                     توقف ارتباط//
i2c_start();                                                    شروع مجدد//
i2c_write(slave_address);                                       فراخوانی Slave//
address++;                                                      رفتن به خانه بعدی حافظه//
i2c_write(address)                                 تعیین خانه حافظه که می خواهیم از آن بخوانیم//
i2c_start();                                                    شروع مجدد//
i2c_write(slave_address | 1); }  فهماندن به حافظه که می خواهیم اطلاعاتش را بخوانیم//
i2c_stop();          }                                          توقف ارتباط//
void eeprom_write(unsigned char address)  {                     تابع نوشتن در حافظه//
i2c_start();                                                    شروع ارتباط//
i2c_write(slave_address);                                       فراخوانی حافظه مورد نظر//
i2c_write(address);                                رفتن خانه حافظه مورد نظر جهت نوشتن //
for(i=0;i<5;i++)  {
i2c_write(w[i]);                                                نوشتن در حافظه//
delay_ms(10);     }                                             تاخیر زمانی//
```

```
i2c_stop();                                  // پایان ارتباط
delay_ms(10);     }                          //تاخیر زمانی
void main(void)   {                          //برنامه اصلی
i2c_init();                                  //آماده سازی ارتباط دوسیمه
lcd_init(16);                                //آماده سازی نمایشگر
eeprom_write(0xaa);                          // رفتن به آدرس aa و شروع نوشتن در حافظه
delay_ms(500);                               // تاخیر زمانی
eeprom_read(0xaa);                           // رفتن به آدرسaa و شروع خواندن از حافظه
lcd_clear();                                 // پاک کردن نمایشگر
lcd_gotoxy(0,0);                             //رفتن به آدرس مورد نظر در نمایشگر
lcd_puts(a);                                 //نمایش اطلاعات خوانده شده از حافظه
while (1);        }                          // رفتن در یک حلقه بینهایت
```



شکل10-12