"کتابخانه ی پروتکل I2C منطبق بر Codevision و Atmel studio"

استاد مربوطه:
آقای سعیدی امین آبادی
گردآورنده:
آرش خواجویی نژاد

تابع تنظیمات اولیه :i2c

```
void TWI_init_BitRate(char PrescalerValue) // Function to initialize
frequency and prescalervalue
{
      TWBR = ((F_CPU/SCL_CLOCK)-16)/2;  /* must be > 10 for stable
operation */
      if (PrescalerValue == 1) TWSR=(0<<TWPS1)|(0<<TWPS0);
// Setting prescalar bits (TWPS) to 1
      if (PrescalerValue == 4) TWSR=(0<<TWPS1)|(1<<TWPS0);
// Setting prescalar bits (TWPS) to 4
      if (PrescalerValue == 16) TWSR=(1<<TWPS1)|(0<<TWPS0);
// Setting prescalar bits (TWPS) to 16
      if (PrescalerValue == 64) TWSR=(1<<TWPS1)|(1<<TWPS0);
// Setting prescalar bits (TWPS) to 64
}
```

_این تابع با گرفتن آرگومان مقسم فرکانسی برای فرکانس کاری twi را مشخص می کند ! همچنین شامل فرمول twbr میشود و همانطور که مشاهده میکنید برای ارتباط پایدار باید این مقدار بالای 10 باشد.پس فرکانس را به گونه ای انتخاب می کنیم که twbr بزرگتر از 10 شود.

تابع استارت در مد :master transmitter

```
unsigned char TWI_start(void)
{
      //Clear TWI interrupt flag (TWINT)by writing a logic 1 to it,Put
start condition on SDA I/O PORTS, Enable TWI (TWEN)
      TWCR= (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
      while(!(TWCR & (1<<TWINT))); // Wait till start condition is
transmitted
      if (((TWSR & 0xF8)!= 0x08) && ((TWSR & 0xF8)!= 0x10)) return 1;
// Check for the acknowledgement the 5 higher bit of TWSR register
checked (0xF8)---0x08 for start in Master transmitter mod
      return 0;
}
```

_با پاک کردن پرچم twi و نوشتن یک منطقی در رجیستر twint همچنین نوشتن 1 منطقی در TWSTA میتوان ارتباط را با این تابع شروع کرد! آرگومان تابع از نوع VOID است و میتواند بازگشتی نباشد.رجیستر TWEN ارتباط I2C را فعال میکند.
حلقه بعدی منتظر میماند تا فرمان START روی پورت ارسال شود و پرچم TWINT مجددا پاک شود ! رجیستر TWSR استاتوس بازگشتی از خط ارتباطی را چک میکند ! ما در این کتابخانه همواره این رجیستر را با 0XF8 ANDمنطقی میکنیم چون به 5 بیت بالای این رجیستر نیاز نداریم و باید طبق جدول دیتشیت همواره 1 باشند. در صورت بازگشت 0x08 در رجیستر TWSR به این

معناست که در مد MASTER به عنوان فرستده دستور AKN توسط SLAVE های خط ارسال فرستاده شده و نشان دهنده این است که SLAVE ها دستور START از سمت MASTERرا دریافت کرده اند.

استارت تکراری در مد :master

```
void TWI_repetitive_start(void)
{
	//Clear TWI interrupt flag (TWINT)by writing a logic 1 to it,Put start condition on SDA I/O PORTS, Enable TWI (TWEN)
	TWCR= (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
	while(!(TWCR & (1<<TWINT))); // Wait till start condition is transmitted
	while((TWSR & 0xF8)!= 0x10); // Check for the acknowledgement the 5 higher bit of TWSR register checked (0xF8)---0x10 for repetitive start in Master transmitter mod
}
```

_همانطور که از اسم تابع مشخص است یک استارت تکراری ارسال میکند. این تابع برای مواقعی استفاده میشود که خط مشغول بوده و master به هر دلیلی akn را دریافت نمیکند اقدام به ارسال دوباره startمیکند. تفاوت این تابع با تابع استارت در akn دریافتی در رجیستر TWSR است که برابر با 0x10 میباشد.و در صورت دریافت AKN میتوان داده ها را ارسال کرد.

```
unsigned char TWI_write_address(unsigned char Address)// address must have 8 bit,7bit for address and 0 bit for data direction bit (write)
{
	TWDR=Address;	// Address and write instruction
	TWCR=(1<<TWINT)|(1<<TWEN);	// Clear TWI interrupt flag,Enable TWI
	while (!(TWCR & (1<<TWINT))); // Wait till complete TWDR byte sended
	if (((TWSR & 0xF8)!= 0x18) && ((TWSR & 0xF8)!= 0x40)) return 1; // Check for the acknoledgement when 0x18 occured the SLAVE address transmited with write bit and waiting for ack
	return 0;
}
```

_تابع نوشتن آدرس :
با کمک این تابع که 1 آرگومان دریافت میکند میتوان آدرس SLAVE هارا روی BUS فرستاد یا اینکه در مود Broadcasting همه slave ها را صدا زد ! همانند توابع قبلی اینجاهم رجیستر 0x18 برای دریاف akn چک میشود.در صورت ارسال آدرس slaveمورد نظر master منتظر دریافت akn از slave صدا زده شده میشود.

```
void TWI_write_data(unsigned char data)
{
      TWDR=data;    // put data in TWDR 8 bits = 7 bit slave address +
Data direction bit (write = 0)
      TWCR=(1<<TWINT)|(1<<TWEN);    // Clear TWI interrupt flag,Enable
TWI
      while (!(TWCR & (1<<TWINT))); // Wait till complete TWDR byte
transmitted
      while((TWSR & 0xF8) != 0x28); // Check for the acknoledgement---
0x28 it means data has been transmitted and akn received to master
}
```

_نوشتن دیتا روی خط :

در تابع نوشتن data که یک آرگومان دریافت میکند میتوان دیتا را بعد از دریافت slave akn مورد نظر ارسال کرد! در حلقه اول master تا ارسال کامل اطلاعات صبر میکند! در حلقه بعد master منتظر دریافت akn میشود که نشانگر این است که slave مورد نظر data را با موفقیت دریافت کرده است.

```
void TWI_stop(void)
{
      // Clear TWI interrupt flag, Put stop condition on SDA (TWSTO),
Enable TWI
      TWCR= (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
      while(!(TWCR & (1<<TWSTO)));  // Wait till stop condition is
transmitted
      //there is no need to check the TWSR value
}
```

_تابع stop رابط i2c:

بعد از هربار ارسال اطلاعات رابط i2c نیاز به متوقف شدن دارد حتی برای ارسال دوباره باید یکبار این تابع اجرا شود و این بسیار مهم است.شرایط توقف با یک کردن رجیستر TWSTO فعال میشود و با حلقه پایانی چک میکنیم که TWSTO بصورت کامل ارسال شود.

تابع سرکشی یا polling:
با استفاده از این تابع میتوان احتمال خطا را به صفر رساند.این تابع با برسی باز بودن خط اقدام به شروع ارسال آدرس میکند و منتظر akn میشود ! حالتهای اتفاقی اعم از busy بودن خط عدم دریافت akn به هر دلیل از جمله نویز یا مشکل در آخر یا ابتدای خط توسط این تابع دریافت شده و همواره برای ایجاد ارتباط صحیح به روش polling تلاش میکند! در مد master transmitter استفاده از این تابع نثبت به start توصیه میشود.

```
void TWI_start_polling(unsigned char Address)
{
      while ( 1 )
      {
            // send START condition
            TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
            // wait until transmission completed
            while(!(TWCR & (1<<TWINT)));
            // check value of TWI Status Register. Mask prescaler bits.
            if (((TWSR & 0xF8) != 0x08) && ((TWSR & 0xF8) != 0x10))
continue;
            // send device address
            TWDR = Address;
            TWCR = (1<<TWINT) | (1<<TWEN);
            // wail until transmission completed
            while(!(TWCR & (1<<TWINT)));

            // check value of TWI Status Register for notack .
            if ( ((TWSR & 0xF8) == 0x20)||((TWSR & 0xF8) == 0x58))
            {
                  /* device busy, send stop condition to terminate write
operation */
                  TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
                  // wait until stop condition is executed and bus
released
                  while(TWCR & (1<<TWSTO));
                  continue;
            }
            //if( twst != TW_MT_SLA_ACK) return 1;
            break;
      }
}
////////////////////////////////////////////////////////////
////////////////////////////////MASTER becomes a receiver mode
/////////////////////////////////////////
////////////////////////////////////////////////////////////
```

مد masterبه عنوان receiver :
در این مد master نقش دریافت کننده را دارد.اطلاعات توسط slave ارسال شده و master باید akn بفرستد.در این مد هم ابتدا Master آدرس slave را میفرستد و akn دریافت میکند سپس slave اطلاعات را برای Masterمیفرستد.

```
void TWI_read_address(unsigned char Address)// address must have 8
bit,7bit for address and 1 bit for data direction bit (read)
{
      TWDR=Address;     // Address and read instruction
      TWCR=(1<<TWINT)|(1<<TWEN);     // Clear TWI interrupt flag,Enable
TWI
      while (!(TWCR & (1<<TWINT))); // Wait till complete TWDR byte
received
      while((TWSR & 0xF8)!= 0x40);  // Check for the acknoledgement
,0x40 it means when the Slave address transmitted in read mode
}
```

_تابع ارسال آدرس برای slave:
بوسیله این تابع masterآدرس slave که قصد دریافت اطلاعات را در مد receiver از آن دارد ارسال میکند! و منتظر میماند تا akn را دریافت کند ! و سرانجام رجیستر TWSR را چک میکند.

```
unsigned char TWI_read_data(void)
{
       TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
        while(!(TWCR & (1<<TWINT)));
       return TWDR;
}
```

بعد از دریافت AKN توسط SLAVE مورد نظر با این تابع دیتای مورد نظر را میخواند و در رجیستر TWDR ذخیره میکند ! این تابع از نوع CHAR است و توانایی برگرداندن 1 بایت در خرار دریافت اطلاعات را دارد (8بیت).