Процесна информация и обработка

Лекция №6

Designer	Philips Semiconductor, known today as NXP Semiconductors
Designed	1982; 35 years ago

Inter-IC (I^2C) е серийна шина за комуникация на данни с помощта на две двупосочни комуникационна линия (**SDA** и **SCL**). Използва се за свързване на нискоскоростни периферните компоненти, вградени системи и мобилни телефони. Името е съкращение на думите ***Inter-Integrated Circuit***.

Протокола е разработен от Philips в началото на 1980 г., като обикновенна шина за вътрешна комуникация за създаване на управляваща електроника.

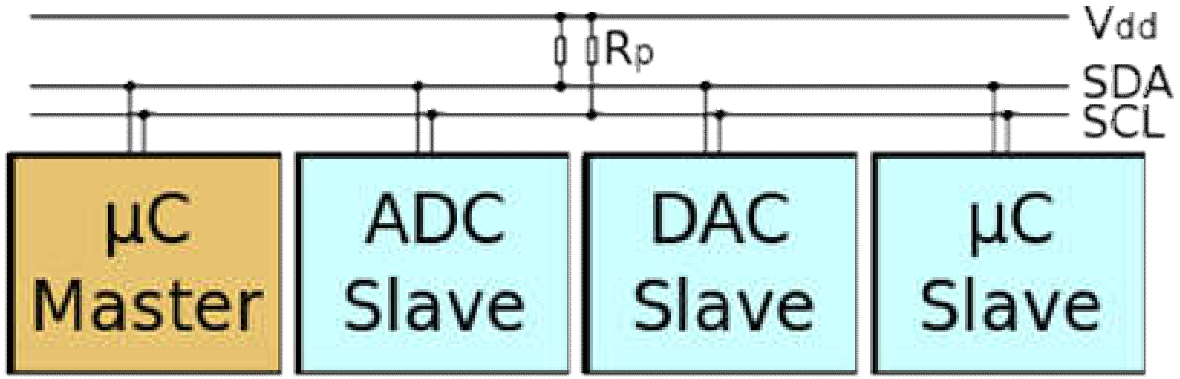
След преработка през 1992 г., става да се свържат повече устройства на шината (поради възможността за 10-битово адресиране), и увеличаване на скоростта до 400 ***Kbit/s*** в режим на висока скорост. Съответно, броя на наличните свободни възли се увеличава до 1008. Максималният брой устройства, които могат да се свържат към една и съща шина, се ограничава от максималния капацитет от 400 ***PF***.

Във версия 2.0 на стандарта, издаден през 1998 г., е въведен режим на висока скорост до 3.4 ***Mbit/s*** с ниска консумация на енергия.

Данните се предават по два проводника - проводник за данни и проводник за тактови импулси. Има водеща (**MASTER**) и водима (**SLAVE**) шина. Тактовете се генерират от водещата шина, а приемащата само потвърждава.

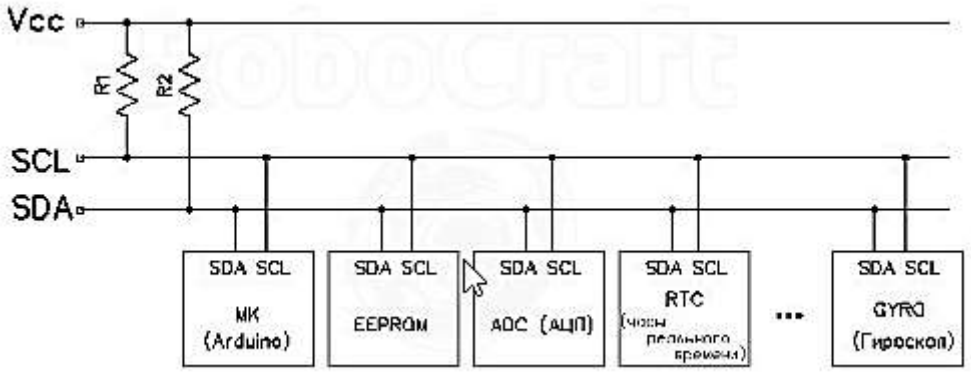
I²C използва две линии с по две направления— последователна линия за данни (**SDA**, англ. **Serial DAta**) и — последователна линия за тактуване (**SCL**, англ. **Serial CLock**). Стандартни са напреженията +5V или +3,3V, но се допускат и други.

Един пример за схема с един основен микроконтролер (µC Master) и три подчинени (slave) устройства (ADC - аналогово-цифров преобразувател, DAC - цифрово-аналогов преобразувател и втори микроконтролер (µC Slave), издърпващите резистори Rp е показан:



I²C намира приложение в устройства, изискващи простота на разработката и ниска себестоиност при относително висока скорост на работа. Възможни приложения:

- •достъп до модулите на паметта NVRAM;
- •достъп до нискоскоростни ЦАП/АЦП;
- •регулиране на контрастноста, наситеността и цветния баланс при мониторите;
- •регулиране на звука във високоговорителите;
- •управление на светодиододи, в това число в мобилни телефони;
- •четене на информация от датчици за наблюдение и диагностика на оборудоване, например, термостат на централния процесор или скорост на въртене на вентилатора за охлаждане;
- •четене на информация от часовници за реално време (кварцови генератори);
- •управление на включването и изключването на захранването на системни компоненти;
- •информационен обмен между микроконтролери;



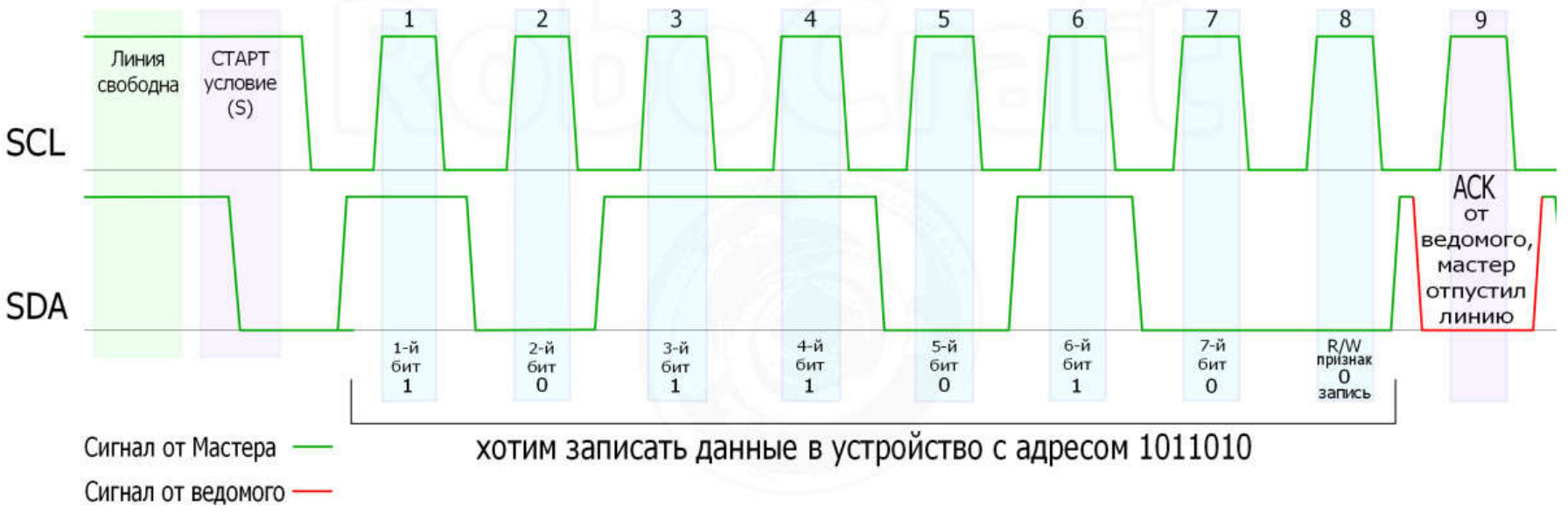
Физически, шината I2C представлява два проводника (без да броим земята и захранването), Свързани с резистори към плюса 1-10к.
Единия проводник е шина за данните(**SDA — Serial DAta**), втория — тактуването(**SCS — Serial CLock**).

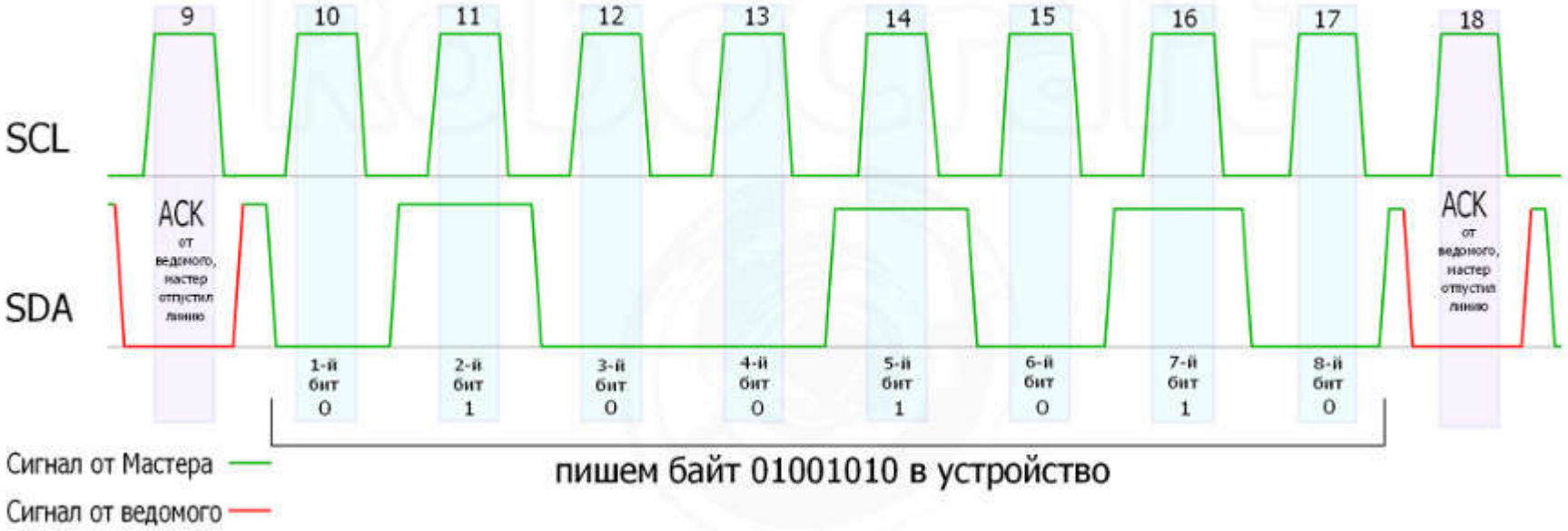
Работи просто:
На линии обикнавено има един (**MASTER**) — МК и някаква количество (**SLAVE**) — периферни устройства (възможна е и многомастерная “архитектура”).
Тъй като линиите са свързани с плюса, то на устройствата им остава да ги притискат към масата когато искат да предадат нула и да отпуснат за единица.
Оттук и важен извод за съвместната работа на такива устройства (нарича се — **монтажно “И”**) — ако някой е установил нула, то на останалите им остава само да се примирят.
И така, тактуването (повдигане на **SCL**) винаги се прави от **MASTER**, предаването започва пак той, след **предварителна проверка дали линията е свободна** (единици на SDA и SCL),
формира СТАРТ-условие (S) — притиска линията SDA (1->0), при единица на SCL,

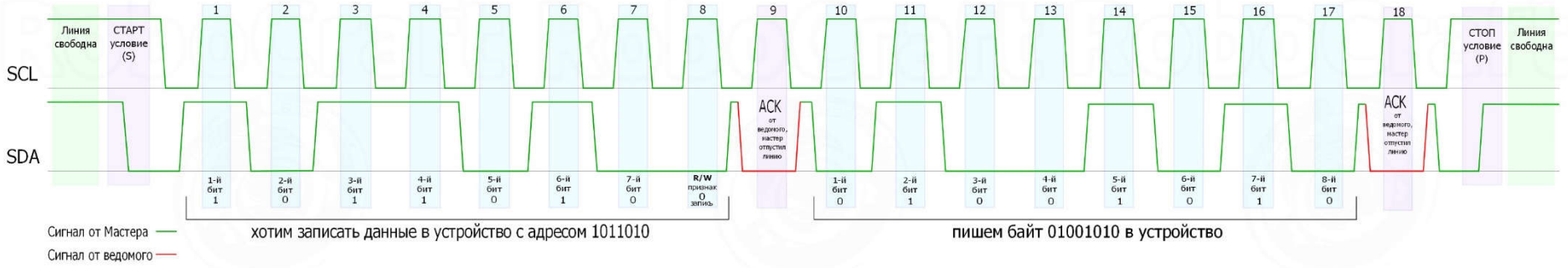
След това се предава адреса на устройството с което ще се комуникира.
При передване по I2C има две правила:
1. Данните се четат при единично състояние на SCL, а могат да се променят само при нулево състояние на SCL.
2. Данните вървят с главата напред — започват със старшия бит(MSB)

- 7 бита са за адреса, осмият е признак R/W — за четене 1, за писане 0.
- След приемане на осмия бит водимата схема трябва да изпрати сигнал за потвърждение (ack, acknowledge) — давайки логическа нула притискайки линията SDA (отпусната от водещия) по време на 9 такт на SCL. **MASTER** чака сигнала до момента в който водимата схема отпуска SDA)
  Ако липсва **ack** (нарича се **nack**) — значи на водимата схема нещо не и е ясно и трябва да се формира СТОП и да се повтори предаването.
- След това , **MASTER** или изпраща байт с данни на **SLAVE**, и отново чака потвърждение или, приема от него байт и вече сам издава подтвърждение.
- Байтовете на данните может да бъдат няколко, но все пак някога ще свършат и мастера сформира СТОП-условие(P). За тази цел трябва да отпусне линиятя SDA (0->1), без да пипа SCL.

комуникация

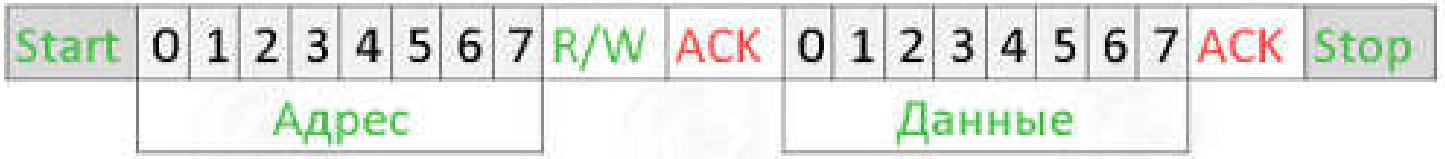






Пълен текст на комуникацията

Поредицата е:
СТАРТ-адрес(запис/четене)-потвърждение-данни-потвърждение-СТОП

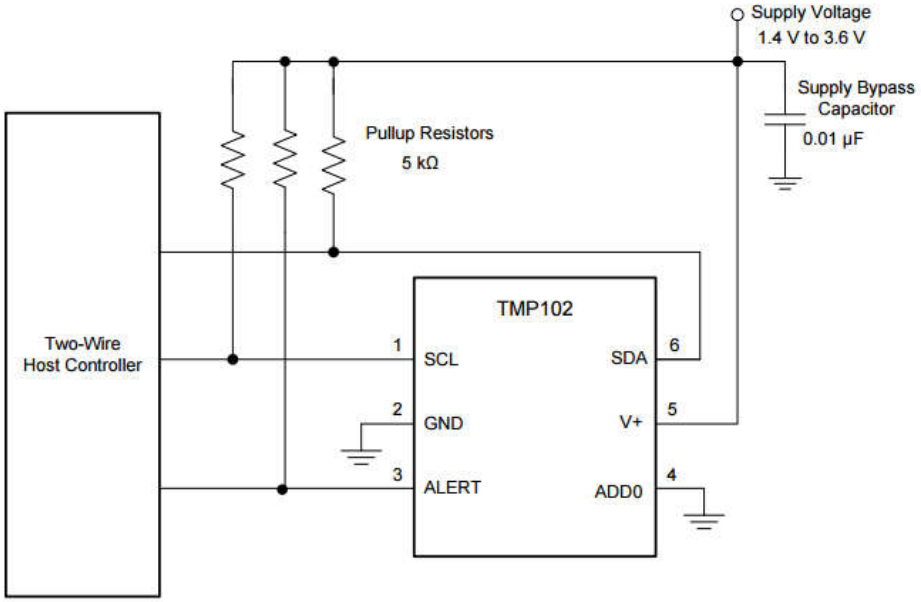


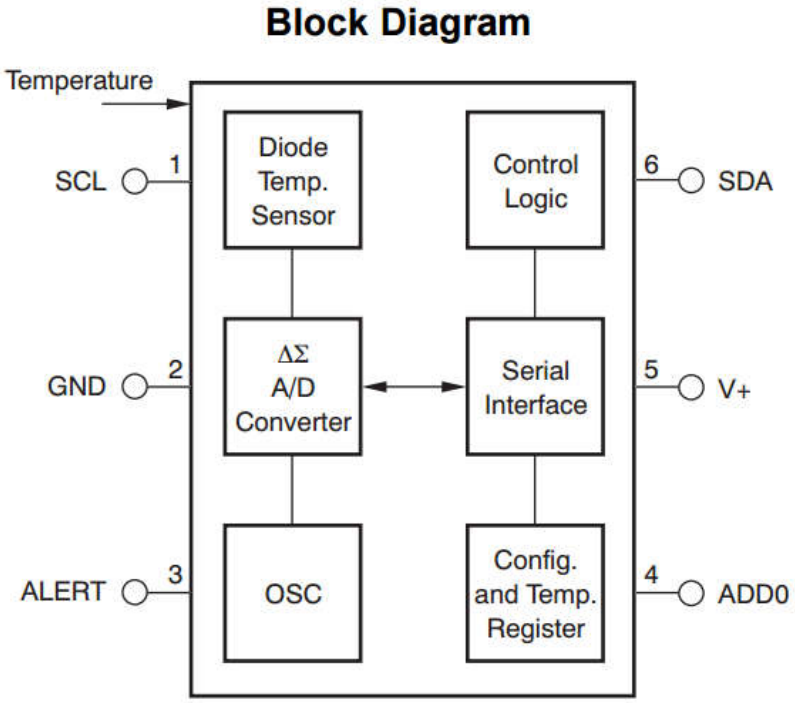
I²C on the mbed

The mbed I²C library functions are shown in the table below:

I2C	An I2C Master, used for communicating with I2C slave devices
Functions	**Usage**
I2C	Create an I²C Master interface, connected to the specified pins
frequency	Set the frequency of the I²C interface
read	Read from an I²C slave
read	Read a single byte from the I²C bus
write	Write to an I²C slave
write	Write single byte out on the I²CC bus
start	Creates a start condition on the I²C bus
stop	Creates a stop condition on the I²C bus

TMP102 Low-Power Digital Temperature Sensor With SMBus and Two-Wire Serial Interface in SOT563





Evaluating the TMP102 I²C temperature sensor

- Configuration and data register details are given in the TMP102 data sheet

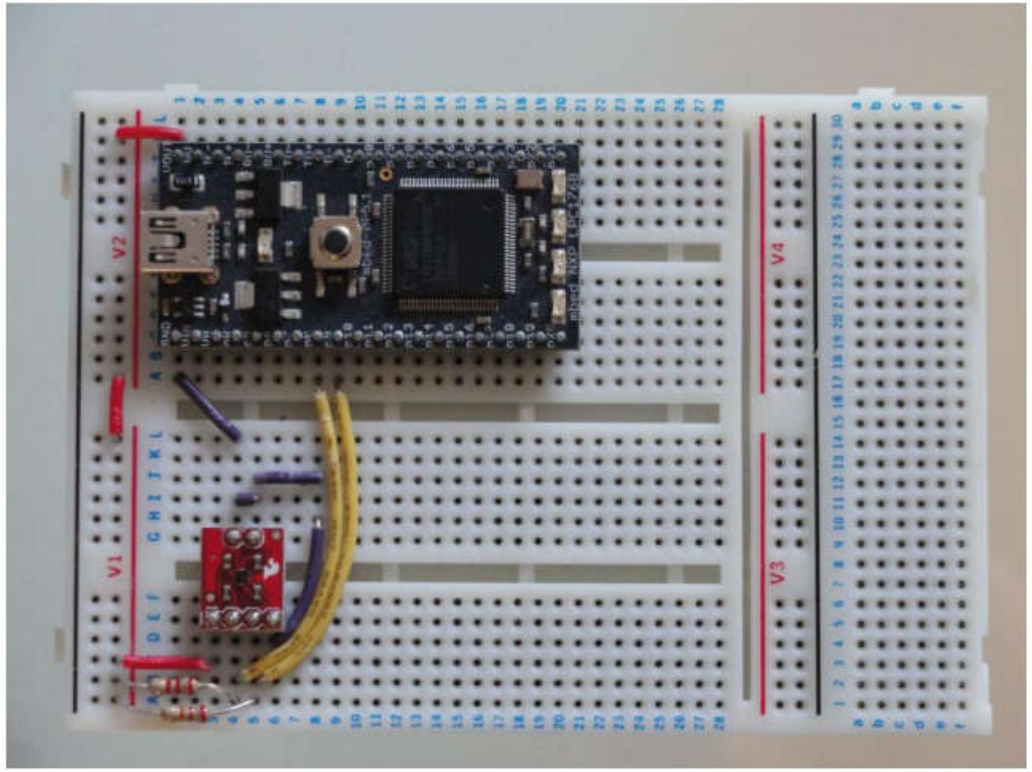
  http://focus.ti.com/lit/ds/sbos397b/sbos397b.pdf

- To configure the temperature sensor we need to:
  - Use arrays of 8-bit values for the data variables, because the I²C bus can only communicate data in one bytes.
  - Set the configuration register; we first need to send a data byte of 0x01 to specify that the Pointer Register is set to 'Configuration Register'.
  - Send two bytes of data to perform the configuration. A simple configuration value to initialise the TMP102 to normal mode operation is 0x60A0.
- To read the data register we need to:
  - To read the data register we need to set the pointer value to 0x00.
  - To print the data we need to convert the data from a 16-bit data reading to an actual temperature value. The conversion required is to shift the data right by 4 bits (its actually only 12-bit data held in two 8-bit registers) and to multiply by the 1-bit resolution which is 0.0625 degrees C per LSB.

Interfacing the TMP102 with the mbed

- The TMP102 can be connected to the mbed as shown:

Signal	TMP102 Pin	Mbed Pin	Notes
Vcc (3.3V)	1	40	
Gnd (0V)	4	1	
SDA	2	9	2.2kΩ pull-up to 3.3V
SCL	3	10	2.2kΩ pull-up to 3.3V



Working with the TMP102 I2C temperature sensor

The following program will configure the TMP102 sensor, read data, convert data to degrees Celsius and then display values to the screen every second:

```
#include "mbed.h"
I2C tempsensor(p9, p10); //sda, scl
Serial pc(USBTX, USBRX); //tx, rx
const int addr = 0x90;
char config_t[3];
char temp_read[2];
float temp;
int main() {
    config_t[0] = 0x01;                     //set pointer reg to 'config register'
    config_t[1] = 0x60;                     // config data byte1
    config_t[2] = 0xA0;                     // config data byte2
    tempsensor.write(addr, config_t, 3);
    config_t[0] = 0x00;                     //set pointer reg to 'data register'
    tempsensor.write(addr, config_t, 1);    //send to pointer 'read temp'
    while(1) {
        wait(1);
        tempsensor.read(addr, temp_read, 2);        //read the two-byte temp data
        temp = 0.0625 * (((temp_read[0] << 8) + temp_read[1]) >> 4);  //convert data
        pc.printf("Temp = %.2f degC\n\r", temp);
    }
}
```

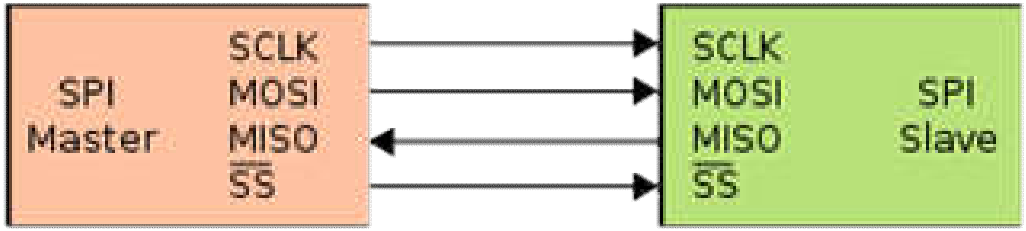
SPI
(Serial Peripheral Interface)

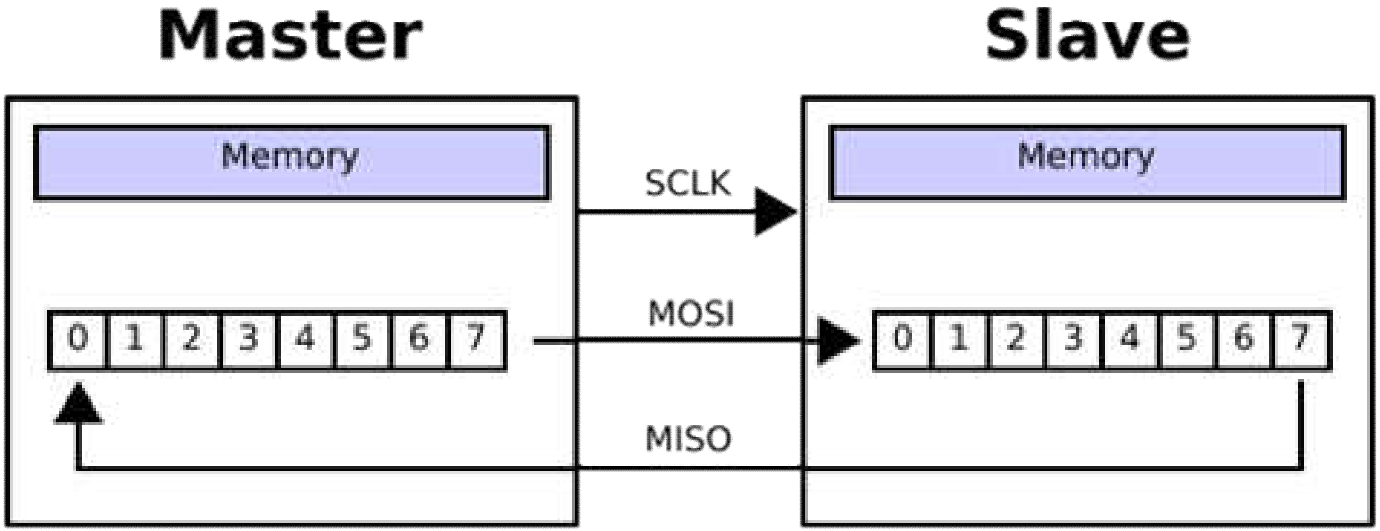
- Developed by Motorola
- Also known as MicroWire (National Semiconductor), QSPI (Queued),MicrowirePlus
- Synchronous Serial Communication

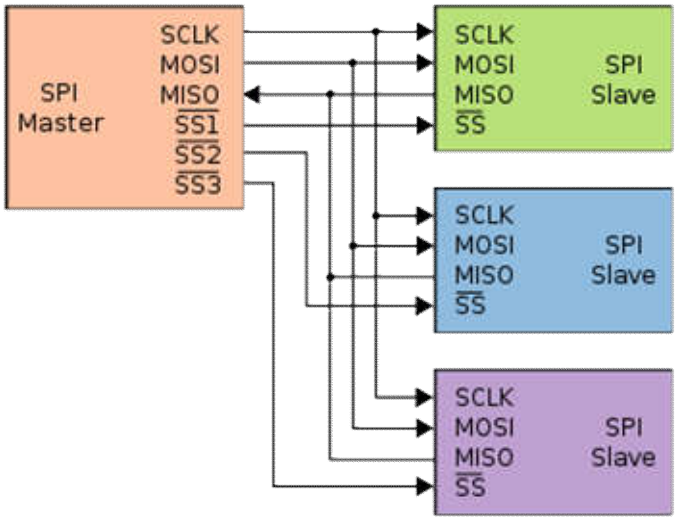
SPI (англ. Serial Peripheral Interface, SPI bus) - синхронен сериен стандарт за пренос на данни в пълен дуплекс режим, предназначен да осигури проста и евтина интерфейс периферия. SPI е също така понякога се нарича четирипроводен протокол(**four-wire**).
За разлика от стандартния сериен порт, SPI е синхронен интерфейс, където всяко прехвърляне се синхронизира с общ тактов сигнал, генериран от водещо устройство (CPU) За един сериен периферен интерфейс на мастер чип може да се свържат няколко устройства. Водещото устройство избира водимото устройство със сигнал "чип изберете" (англ. chip select). Периферииите неизбрани от процесора не участват в предаване на SPI.
В **SPI** се използват четири цифрови сигнала:

- **MOSI**— изход на водещия вход на водимия (Master Out Slave In). Служи за предаване на данни от водещото към водимото устройство.
- **MISO**— вход на водещия изход на водимия (Master In Slave Out). Служи за предаване на данни от водимото към водещото устройство.
- **SCLK**— последователен тактов сигнал (Serial Clock). Служи за предаване на тактовия сигнал за водимите устройства.
- **CS** или **SS** —избор на водимото устройство (Chip Select, Slave Select).

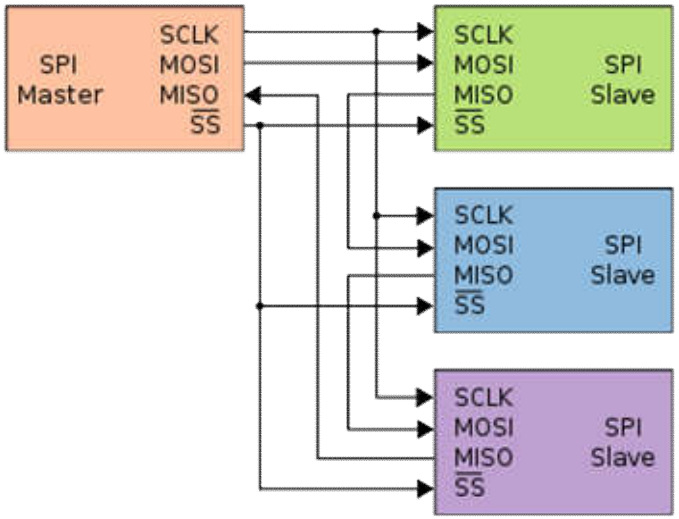
Прехвърлянето се извършва на пакети. Дължината на пакета, обикновено е 1 байт (8 бита), известно е прилагането на SPI пакети с различна дължина, например 4 бита. Водещото устройство инициира комуникационния цикъл чрез избор (SS) на устройството, към което да се свърже.


SPI
Master
SCLK
MOSI
MISO
SS
SCLK
MOSI
MISO
SS
SPI
Slave


Master
Slave
Memory
Memory
SCLK
MOSI
MISO
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7

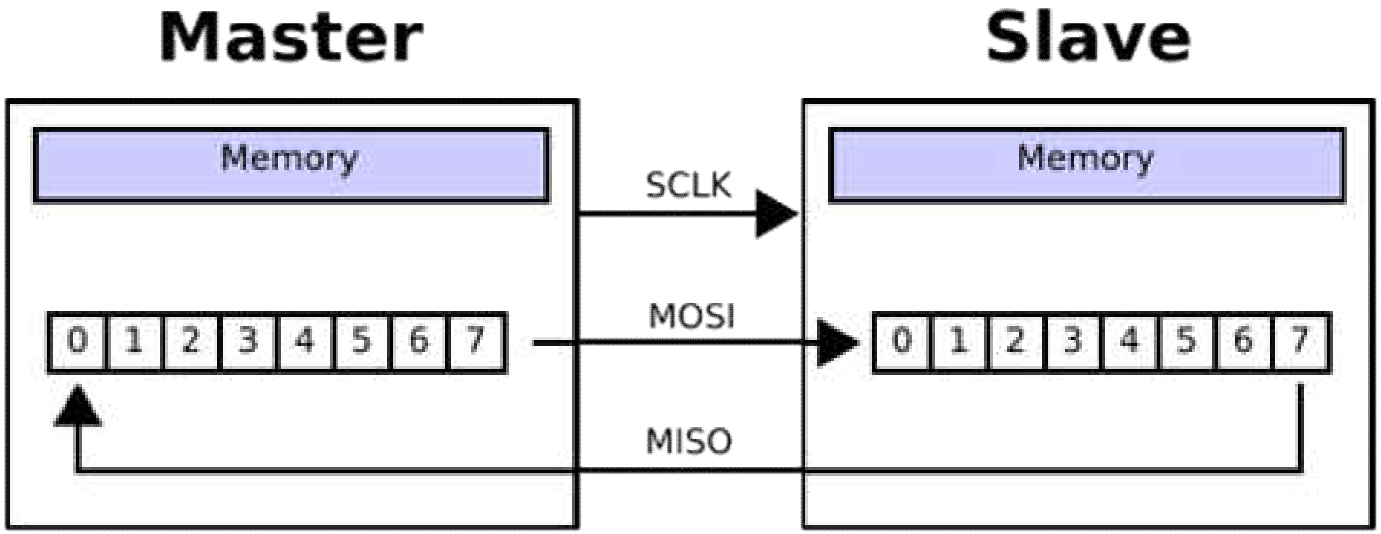


Радиална структура на връзка с няколко водими устройства чрез SPI



Кръгова структура на връзка с няколко водими устройства чрез SPI

Данните които трябва да предават се поставят в регистрите за смяна. След това водещото устройството започва да генерира импулси за синхронизиране на линии SCLK, което води до взаимен обмен на данни. Данните се прехвърлят бит по бит от MOSI на MISO. Предаването става, като правило, започвайки от старшите битове.



Стандартния алгоритъм за работа SPI е такъв:

1.Водещия установява ниско ниво на тази линия SS, към котято е включен необходимия водим.
2.Водещия задава такт, «повдигайки» нивото на SCLK, и едновременно с всяко повдигане на SCLK установява нужното ниво на MOSI, передавайки на водимото по бит за такт.
3.Водимия на всеки такт «повдигане» на SCLK установява нужното ниво на MISO, передавайки на водимото по бит за такт на водещия по бит за такт.
4.За завършване на предаването водещия установява високо ниво на SS.

SPI е пълнодуплексна шина — данните се передаватя едновременно в двете страни. Типиа скорост на работа е 1-50 МГц.
Благодарение на простия алгоритъм SPI е получил широко разпостранение в сензори, чипови за памет, радиомодули, и т.н.

SPI има четири режима за предаване, основани на комбинациите от «полярност» на тактовия сигнал (**clock polarity, CPOL**) и фазата на синхронизация (**clock phase, CPHA**). Посто казано, **CPOL** — е нивата на тактовата линия до началото и след края на предаването : нисък (0) или висок (1). А фазата определя, на фронта или спада на тактовия сигнал се предават битовете:

•Режим 0: CPOL=0, CPHA=0

Четенето на бита става на фронта на тактовия сигнал (преход 0 1), а записа — на спада (1 0).

•Режим 1: CPOL=0, CPHA=1

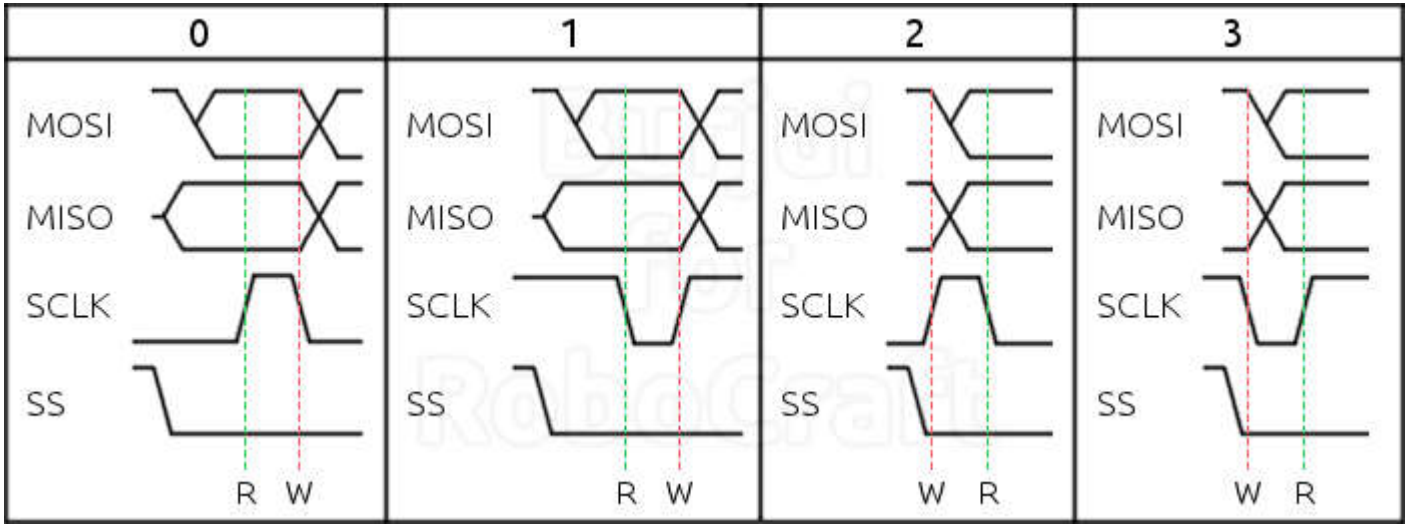
Четене — на спада, запис — на фронта.

•Режим 2: CPOL=1, CPHA=0

Четене — на спада, запис — на фронта.

•Режим 3: CPOL=1, CPHA=1

Четене — на фронта, запис — на спада.



Данните по SPI може да се передават или със старшия бит напред вперёд, или с младшия.

Host adapters

Manufacturer	SPI host adapter	Host bus	Bus protocol support	Max frequency
Byte Paradigm	SPI Storm	USB	SPI, dual/quad, custom	100 MHz
Corelis	BusPro-S	USB	SPI, dual/quad	60 MHz
HydraBus	HydraBus-HydraFW	USB	I²C, 2-Wire, 3-Wire, JTAG, CAN, UART, Parallel, Scriptable binary bitbang, SPI	42 MHz/varies
Microchip	MCP2210 Kit	USB	SPI	12 MHz
National Instruments	USB-8452	USB	I²C, SPI	50 MHz
Total Phase	Cheetah SPI Host Adapter	USB	SPI	40 MHz
Total Phase	Promira Serial Platform	USB, Ethernet	I²C, SPI, single/dual/quad, and eSPI	80 MHz
Dangerous Prototypes	Bus Pirate	USB	1-Wire, I2C, SPI, JTAG*, Asynchronous serial, Scriptable binary bitbang, UART	varies

Преимущества

•Пълнодуплексно предаване на данни по подразбиране.
•По висока пропускателна способност в сравнение с I²C или SMBus.
•Възможност за произолен избор на дължината на пакета.
•Проста апаратна реализация:
- по ниски изисквания към енергопотреблението в сравнение с I²C и SMBus;
- возможно е използване в системи с нискостабилна тактова частота;
- на водимата устройства не е необходим уникален адрес, за разлика от I²C, GPIB или SCSI.

•Използват се само четири извода, което е много по малко от паралелните интерфейси.
•Еднопосочността на сигнала позволява лесно да се организира галваническо разделение между водещото и водимото устройство.
•Максималната тактова честота е ограничена само от бързодействието на устройствата, участващи в обмена на данни.

Недостатъци

•Необходими са повече изводи от I²C.
•Водимото устройство не може да управлява потока на данни.
•Нямо потвърждение за получените данни.
•Няма определен стандарт за намиране на грешките.
•Отсътствие на официален стандарт.
•По далечина на предаваните данни SPI отстъпва на UART и CAN.
•Наличие на множество варианти на реализация на интерфейса.
•Отсътствие подръжка на горещо включване на устройства.

SPI Digital POT (EE 583)

- We used a Microchip digital POT in EE 583 with SPI interface.
- Motorola 68HC12 had SPI built in hardware, very easy to use.
- Data we sent to POT via SPI consisted of 2 bytes.
- Command Byte- XXC1C0XXP1P0
- C1C0 determines type of command,
- eg. 01 = Write Data
- 10 =Shutdown
- P1P0 Determines which potentiometer is affected by
- the command
- 00 = Nothing affected
- 01 = Command executed on POT 0
- 10 = Command executed on POT 1
- 11 = Command executed on both POT's
- Data byte indicates value of the wiper

SPI Potentiometer Transaction

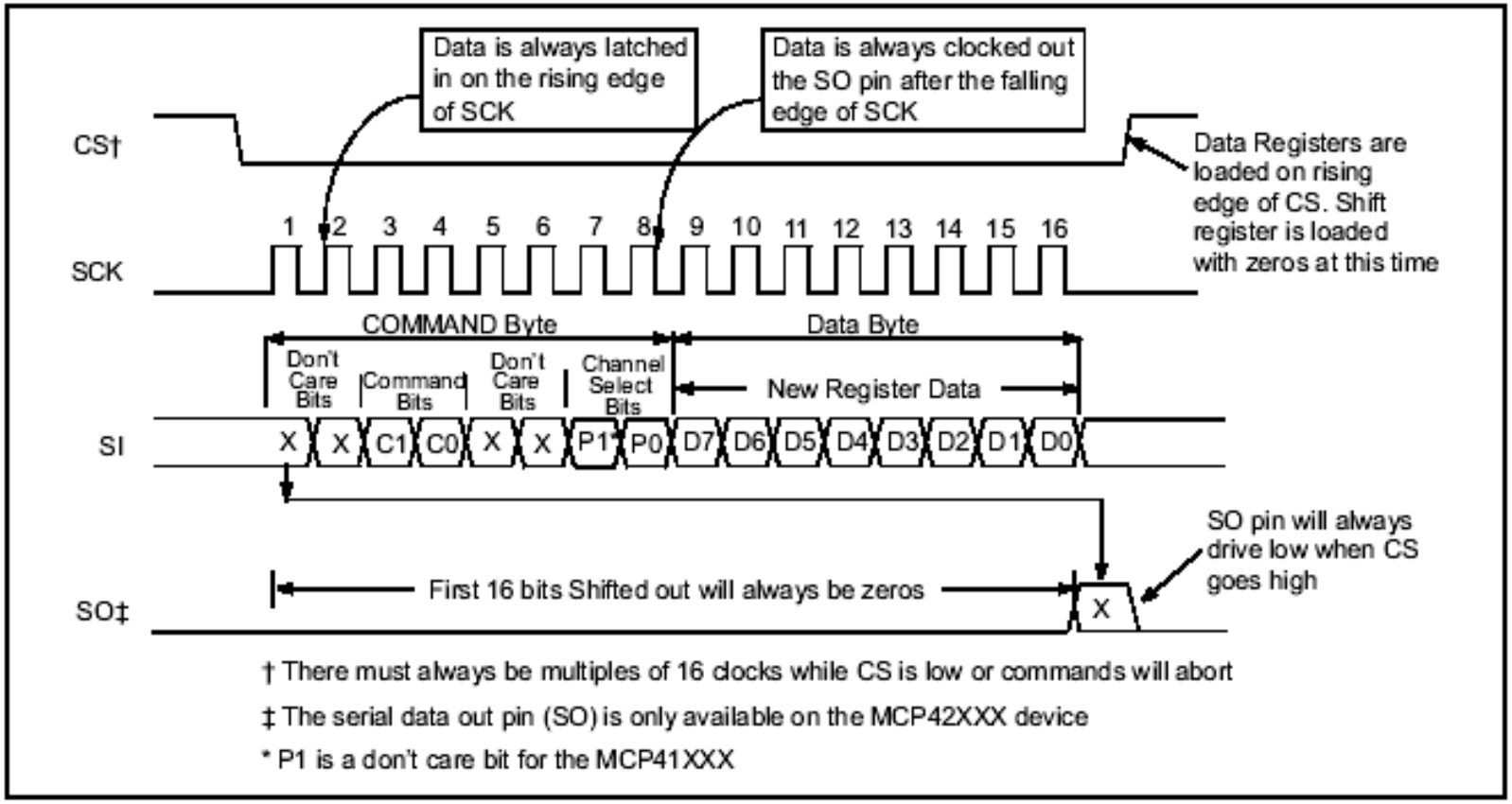


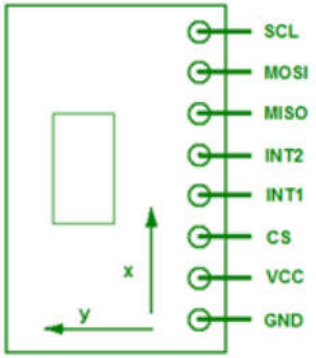
Figure 5-1: *Timing Diagram for Writing Instructions or Data to a Digital Potentiometer*

Evaluating the ADXL345 SPI accelerometer

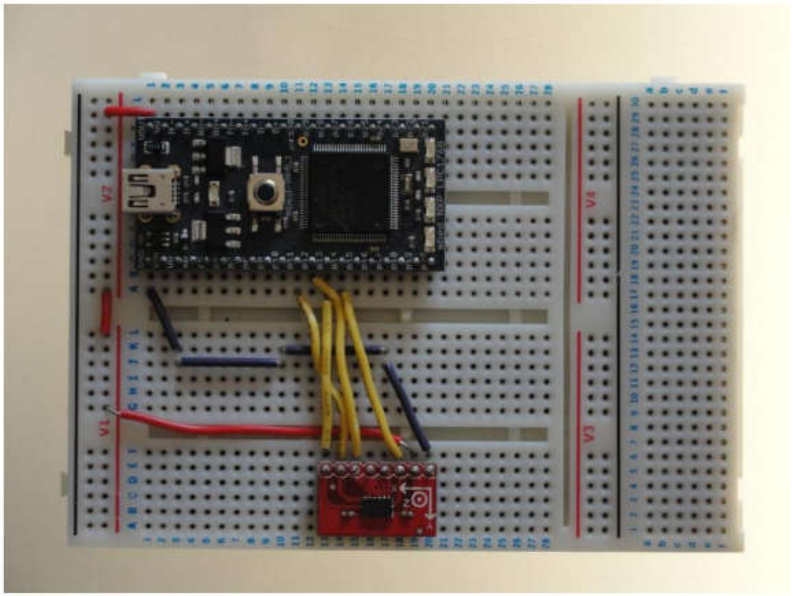
- To read 3-axis data back from the ADXL345 we must:
  - Set the read/write bit high (bit 7 of the address byte)
  - Set the multibyte-data flag high (bit 6 of the address byte)
  - Set CS low
  - Send the configured address byte for register 0x32
  - Read 6 bytes of data back from the ADXL345 by writing 6 dummy bytes of 0x00
  - Set CS high
- The 6 returned data bytes contain the most significant and least significant bytes for each of the three measurement axes (x, y and z).
- We therefore need to convert the data to floating point ‘g’ values by combining the relevant data bytes and multiplying by the configured data resolution.

Interfacing the ADXL345 SPI accelerometer with the mbed

- The ADXL345 can be connected to the mbed as shown:



ADXL345 signal name	mbed pin
Vcc	40
Gnd	1
SCL	13
MOSI	11
MISO	12
CS	14



Esbus

- Sensors bus
- Being developed by Esensors Inc, (see www.eesensors.com for more info)
- 6 wire sensor bus with modular connector based on modified SPI
- Byte of data is exchanged between the master and slave
- Optical isolators provide ground isolation for safety and noise reduction
- Data is transmitted from master along EDI lines
- The signal is connected to the data input to SPI serial bus on microcontroller
- Sensor information from slave are transmitted on EDO line to output of remote sensor
- Data line is connected to SDO in sensor end.
- Isolated DC to DC supply is used to retain ground isolation (optional)