Embedded System Course 2021-22

C4µC

C/C++ Programming for Microcontrollers

Lecturer: Ing. Marco Santic

marco.santic@univaq.it

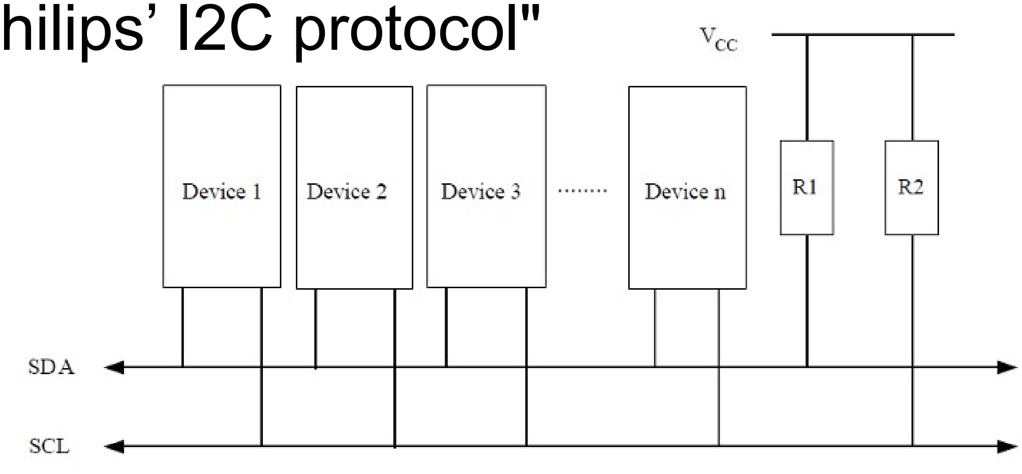
C4μC

- Introduction
  - Atmega328P
  - Toolchain
  - IDE
  - Examples
  - C language recap
- Basic functionalities
  - GPIO
  - Timers
  - Interrupt
  - Examples
- **Comm. Interfaces**
  - UART
  - **I2C**
  - **SPI**
  - **Examples**
- Other peripherals
  - ADC
  - PWM
  - GPIO bitbanging
  - Examples

C4μC - I2C

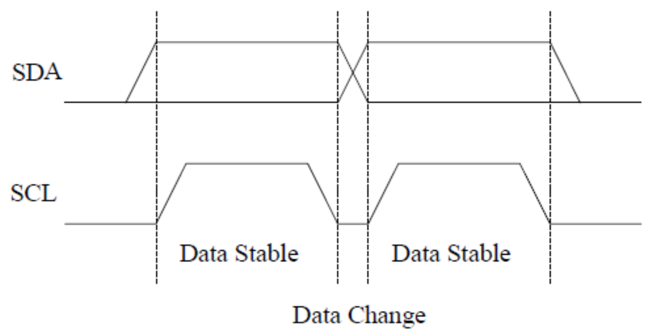
- Serial communication interfaces:
  - I2C:
    - On Atmega328 is called properly *Two wire interface*
      - Master or slave role supported
      - Hw device can transmit and receive
      - 7 bit adress (not 10 here)
      - Up to 400KHz (Fast Mode)
      - "Compatible with Philips' I2C protocol"
      - Two lines:
        - SDA (data)
        - SCL (clock)

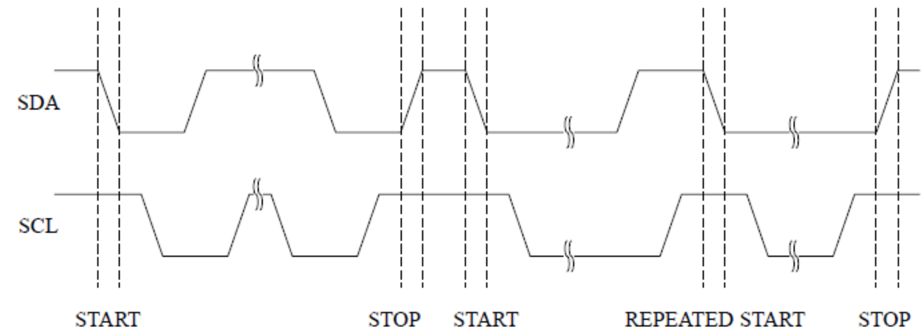
No matter, for us, if we call it I2C or TWI



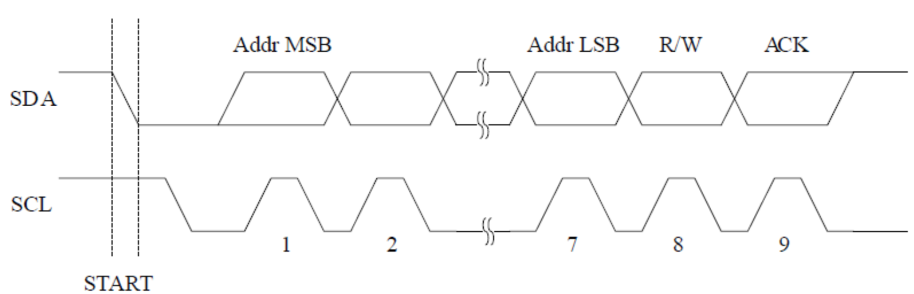
C4µC - I2C

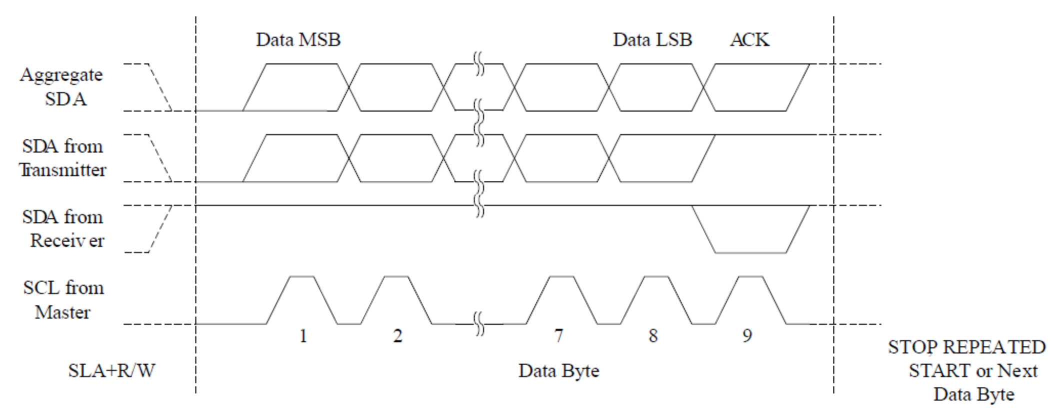
- Twi on the bus...





  - Data is valid on clock HIGH
  - Start, Stop and Repeted Start delimit frames

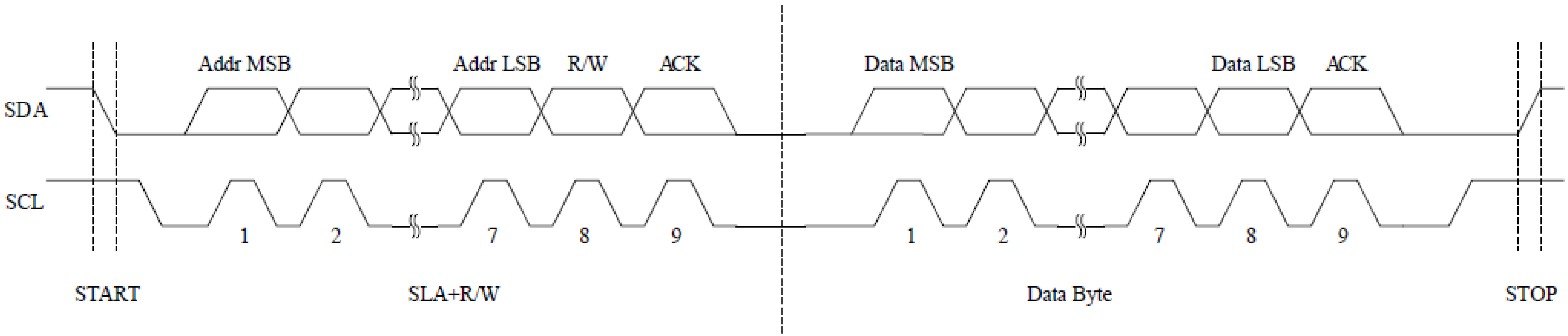




  - Address frame (first one)
  - Data frame, with ACK from the receiver

C4µC - I2C

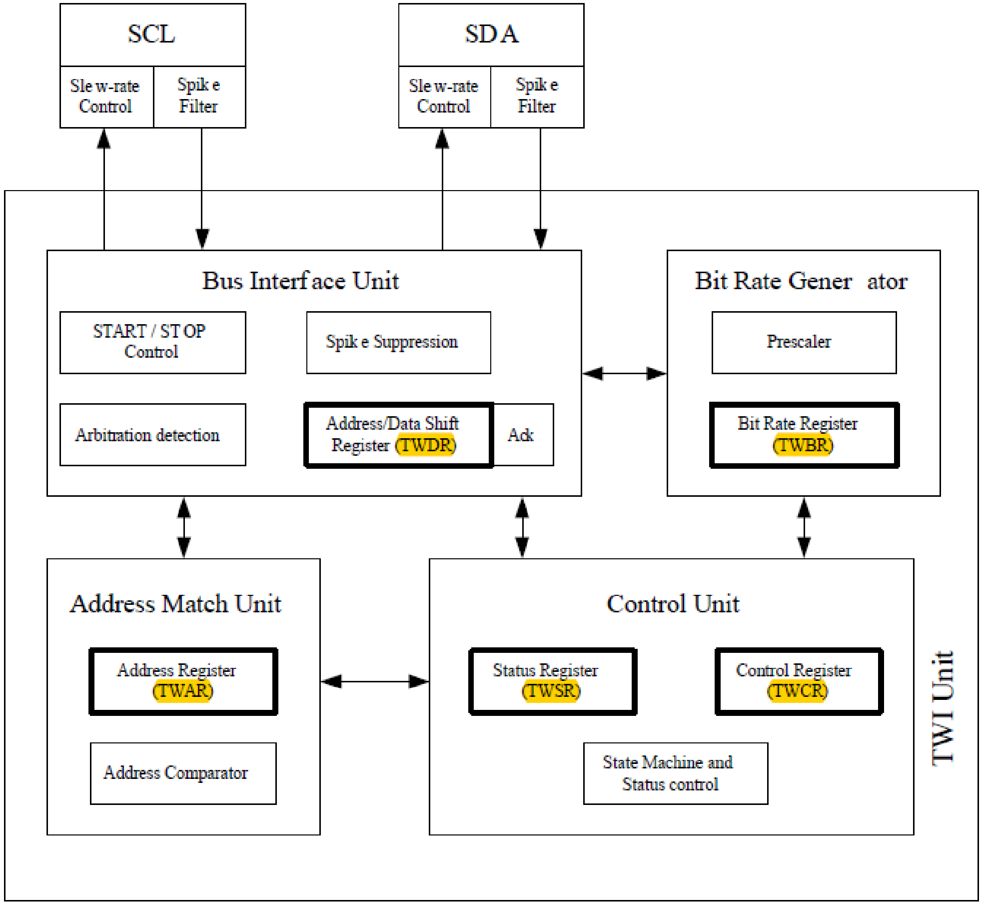
- Twi on the bus...



  - Address frame (first one) is always followed by some data...
    - Except if we are "scanning the bus"
      - Scanning the bus for a range of addresses and see on which ones we have an ACK
      - This procedure can be used if we don't know the address of some device

C4µC - I2C

- Block diagram
  - Regs:
    - TWBR (bitrate)
    - TWDR (addr/data)
    - TWAR (hw address)
    - TWSR (status)
    - TWCR (control)

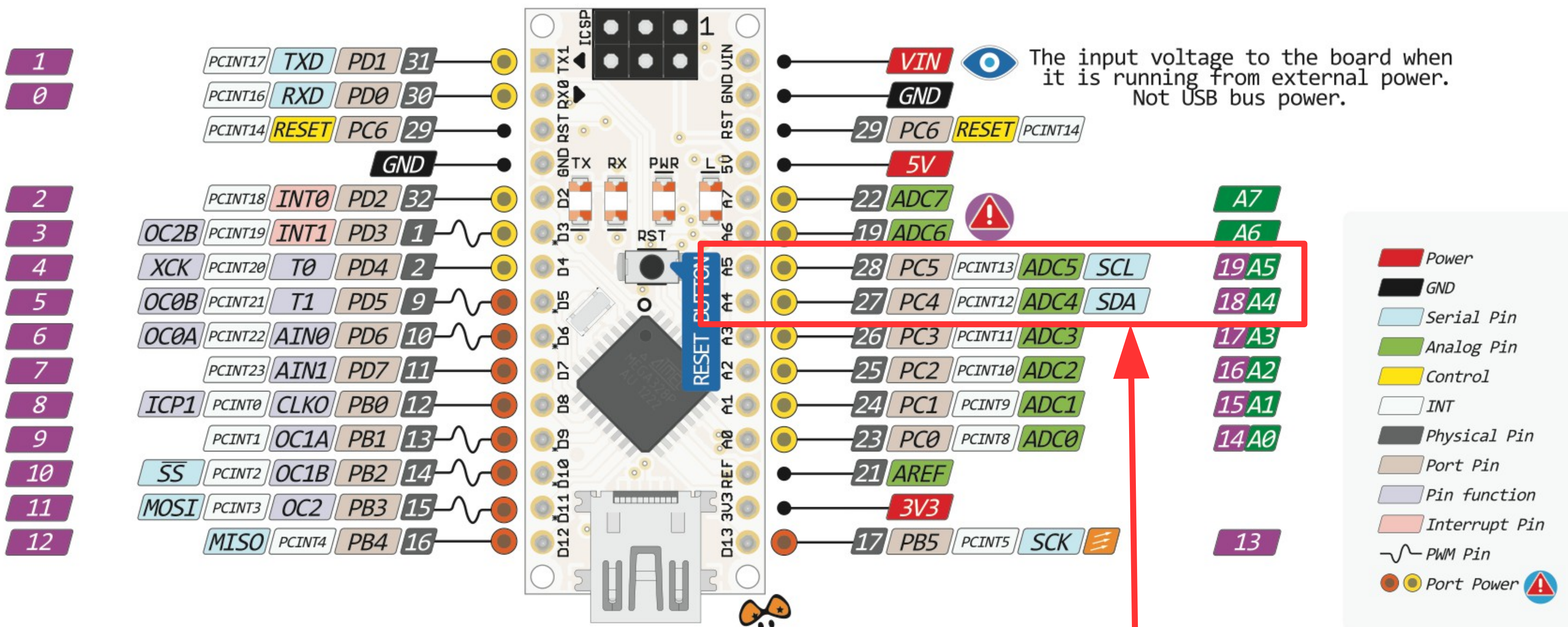


NOTE:
a good figure on the appl. - hw sequence is in datasheet p.268

Where is also reported "The AVR TWI is byte-oriented and interrupt based...
… the application software is free to carry on other operations during a TWI byte transfer"

C4µC - I2C

- Arduino Nano v3 pinout



SDA: serial data (on A4)
SCL: serial clock (on A5)

C4µC - I2C

- sketch_c4uc_7.1_scanTWIbus_AAPI

```
// I2C Scanner, Written by Nick Gammon, Date: 20th April 2011
```

← Credits!

```
#include <Wire.h>

void setup() {
  Serial.begin (115200);
  while (!Serial){}
  Serial.println (); Serial.println ("I2C scanner. Scanning ...");

  byte count = 0;
  Wire.begin();

  for (byte i = 8; i < 120; i++){
    Wire.beginTransmission (i);
    if (Wire.endTransmission () == 0){
      Serial.print ("Found address: "); Serial.print (i, DEC);
      Serial.print (" (0x"); Serial.print (i, HEX); Serial.println (")");
      count++;
      delay (1);  // maybe unneeded?
      } // end of good response
  } // end of for loop
  Serial.println ("Done."); Serial.print ("Found ");
  Serial.print (count, DEC); Serial.println (" device(s).");
}  // end of setup

void loop() {}
```

Let's analise briefly the code...

C4µC - I2C

- sketch_c4uc_7.2_scanTWIbus **AVR style, a little more complicated!**

```
// I2C Scanner, Written by us @ Embedded Systems course

#include <util/twi.h>
#include <util/delay.h>
#include "avr_uart.h"       //our AVR uart example inside a .h file (+some util print functs)

#define FOSC 16000000L // Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1  //See table 24-1 datasheet p.227

#define MYTWI_FREQ 100000UL

int main( void ){

  DDRB |= _BV(DDB5); // Led
  USART_Init(MYUBRR);
  unsigned char msg[] = "Device at ";
  uint8_t msg_len = 10;

  // activate internal pullups for twi.
  PORTC = _BV(DDC4) | _BV(DDC5);

  // initialize twi prescaler and bit rate
  TWSR &= ~_BV(TWPS0);
  // SCL Frequency = CPU Clock Frequency / (16 + (2 * TWBR))
  // note: TWBR should be 10 or higher for master mode
  // It is 72 for a 16mhz Wiring board with 100kHz TWI
  TWBR = ((FOSC / MYTWI_FREQ) - 16) / 2;
```

Look at procedure in datasheet (pp.270-271)

C4µC - I2C

- sketch_c4uc_7.2_scanTWIbus ... continue main()

```
  // enable twi module, acks
  TWCR = _BV(TWEN) | _BV(TWEA);
  uint8_t sla;
  for (uint8_t i = 8; i < 127; i++){
    TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN); // Send START condition
    while (!(TWCR &(1<<TWINT)));            // Wait for TWINT Flag set. This indicates that the..
                                            //   ..START condition has been transmitted.
    if ((TWSR & 0xF8) != TW_START)          // Check value of TWI Status Register. Mask prescaler bits.
      PORTB |= _BV(PORTB5);                 // If status different from START turn on the LED
    sla = ((i & 0x7F) << 1) | TW_WRITE;     // Shift sleave address one left (for the frame)..
                                            //   ..and set TW_WRITE flag

    TWDR = sla;                             // Load slave address into TWDR Register.
    TWCR = (1<<TWINT) | (1<<TWEN);          // Clear TWINT bit in TWCR to start tx of address.
    while (!(TWCR & (1<<TWINT)));           // Wait for TWINT Flag set. This indicates that..
                                            //   ..the SLA+W has been transmitted,..
                                            //   ..and ACK/NACK has been received.

    if ((TWSR & 0xF8) == TW_MT_SLA_ACK){    // ACK, a device is on the bus..
      USART_print(msg, msg_len);            //   ..print the message..
      USART_println_uHex(i);                //   ..and the device address
    }
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO); // Transmit STOP condition

    _delay_ms(1);
  }
}
```

We will LIMIT to the use of Wire.h API functions

C4µC - I2C

- Which are I2C devices we can use?
  - Memories
  - Sensors
  - External ADCs
  - Clock modules
  - Port expander
  - ...

Let's see some examples

C4µC - I2C examples

- DS3231: sketch_c4uc_7.3_clock_module

```
//... code from the write function...
void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte dayOfMonth,
byte month, byte year){
  // sets time and date data to DS3231
  Wire.beginTransmission(DS3231_I2C_ADDRESS);
  Wire.write(0); // set next input to start at the seconds register
  Wire.write(decToBcd(second)); // set seconds
  Wire.write(decToBcd(minute)); // set minutes
  Wire.write(decToBcd(hour)); // set hours
  Wire.write(decToBcd(dayOfWeek)); // set day of week (1=Sunday, 7=Saturday)
  Wire.write(decToBcd(dayOfMonth)); // set date (1 to 31)
  Wire.write(decToBcd(month)); // set month
  Wire.write(decToBcd(year)); // set year (0 to 99)
  Wire.endTransmission();
}
```


```
//... code from the read function...
{
  Wire.beginTransmission(DS3231_I2C_ADDRESS);
  Wire.write(0); // set DS3231 register pointer to 00h
  Wire.endTransmission();
  Wire.requestFrom(DS3231_I2C_ADDRESS, 7);
  // request seven bytes of data from DS3231 starting from register 00h
  *second = bcdToDec(Wire.read() & 0x7f);
  *minute = bcdToDec(Wire.read());
  *hour = bcdToDec(Wire.read() & 0x3f);
  *dayOfWeek = bcdToDec(Wire.read());
  *dayOfMonth = bcdToDec(Wire.read());
  *month = bcdToDec(Wire.read());
  *year = bcdToDec(Wire.read());
}
```

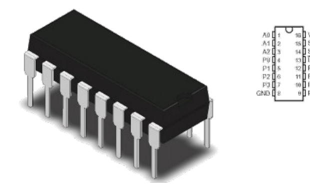
NOTE: use 3.3V for V+

Demo now!

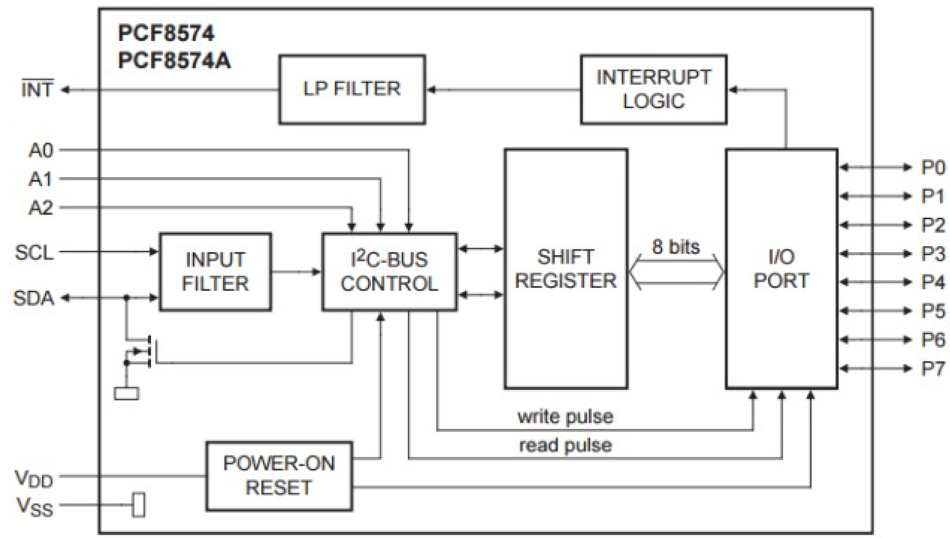
C4µC - I2C examples

- PCF8574: sketch_c4uc_7.4_port_expander

pcf8574
8bit I2C I/O expander



```
#include "Wire.h"
#define PCF8574_ADDRESS 0x20
#define PCF8574_LED 0      // bit 0
#define PCF8574_MOTOR 1    // bit 1

uint8_t remote_port_status = 0x00;

void setup(){
  Wire.begin();
  Serial.begin(115200);
  setPorts();
}

void setPorts(){
  Wire.beginTransmission(PCF8574_ADDRESS);
  Wire.write(remote_port_status);
  Wire.endTransmission();
}

void port_ON(uint8_t led_motor){
  remote_port_status |= _BV(led_motor);
  setPorts();
}

void port_OFF(uint8_t led_motor){
  remote_port_status &= ~_BV(led_motor);
  setPorts();
}
```



NOTE: LEDs (3W total) and motor (1.5W) are driven by an ULN2003A...

C4µC - I2C examples

- PCF8574: sketch_c4uc_7.4_port_expander

```
void loop(){
  uint8_t command = 0;
  while(!Serial.available());
  command = Serial.read();
  switch(command){
    case '1':
      port_ON(PCF8574_LED); break;
    case '0':
      port_OFF(PCF8574_LED); break;
    case 'm':
      port_ON(PCF8574_MOTOR); break;
    case 's':
      port_OFF(PCF8574_MOTOR); break;
  }
  delay(500);
}
```

PCF8574 and PCF8574A I2C-Bus Slave Address Map

INPUTS			PCF8574 I2C-Bus Slave Address	PCF8574A I2C-Bus Slave Address
A2	A1	A0		
L	L	L	20 (hexadecimal)	38 (hexadecimal)
L	L	H	21 (hexadecimal)	39 (hexadecimal)
L	H	L	22 (hexadecimal)	3A (hexadecimal)
L	H	H	23 (hexadecimal)	3B (hexadecimal)
H	L	L	24 (hexadecimal)	3C (hexadecimal)
H	L	H	25 (hexadecimal)	3D (hexadecimal)
H	H	L	26 (hexadecimal)	3E (hexadecimal)
H	H	H	27 (hexadecimal)	3F (hexadecimal)

- Send from *Serial Monitor* commands: "1", "0", "m", "s", ... (also together)
- Port expander's pin can also be used as Input pins, even with INT function
- Let's try to modify the address using pins A0, A1, A2 (see diff. with ScanI2C)

This last point as exercise now

C4μC - I2C examples

Questions on I2C and related examples??

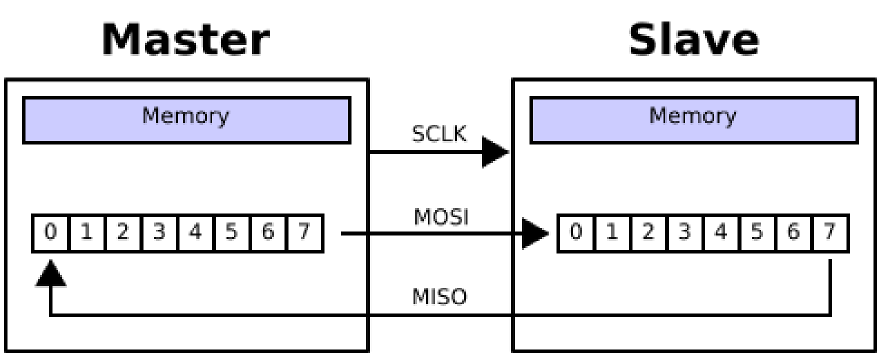
C4μC - I2C examples

Questions on I2C and related examples??

SPI next lesson

C4µC - SPI

- SPI stands for Serial Peripheral Interface and is a synchronous serial communication interface
  - Full duplex mode, master/slave architecture with a single master
  - The communication frame should be generated by the master
- It uses four lines:
  - SCLK: Serial Clock (output from master).
  - MOSI: Master Output Slave Input (data output from master).
  - MISO: Master Input Slave Output (data output from slave).
  - SS (or CS): Slave (or Chip) Select (often active low, output from master)

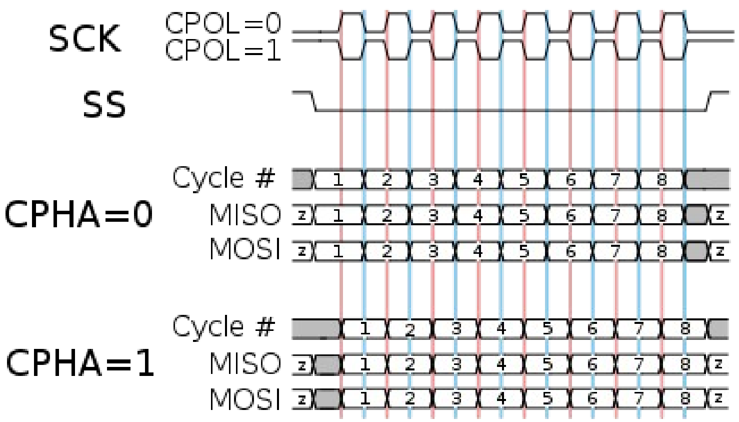


Circular transfer

C4µC - SPI

- Master and slave(s) must agree with some comm protocol parameters, such as:
  - Speed (clock frequency, up to hundreds of MHz)
  - Clock polarity
  - Clock phase
  - CS value (often active low)

Note different reading on rising or falling edge of SCLK

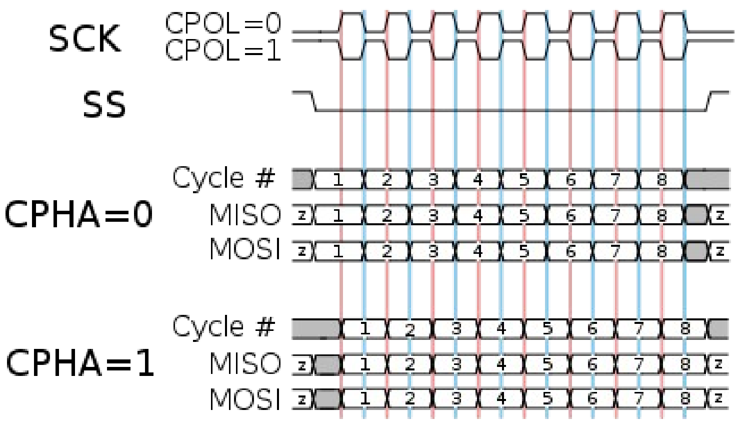


- Often libraries implement "transfers"
  - Since for each byte written there is a byte read, sometimes "dummy bytes" are sent in transfers
  - E.g.: Send a command (expecting an immediate answer), so send a dummy byte in which the answer will be written

C4µC - SPI

- Master and slave(s) must agree with some comm protocol parameters, such as:
  - Speed (clock frequency, up to hundreds of MHz)
  - Clock polarity
  - Clock phase
  - CS value (often active low)

Note different reading on rising or falling edge of SCLK



- Often libraries implement "transfers"
  - Since for each byte written there is a byte read, sometimes "dummy bytes" are sent in transfers
  - E.g.: Send a command (expecting an immediate answer), so send a dummy byte in which the answer will be written

C4µC - SPI

- To have an idea of the SPI, let's look in Arduino IDE:
  - File --> Examples --> SPI --> BarometricPressureSensor

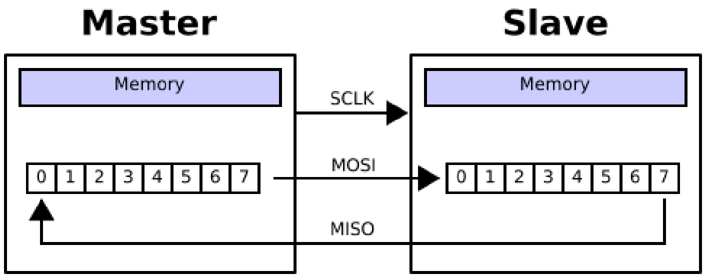
```
//Sends a write command to SCP1000
void writeRegister(byte thisRegister, byte thisValue) {

  // SCP1000 expects the register address in the upper 6 bits
  // of the byte. So shift the bits left by two bits:
  thisRegister = thisRegister << 2;
  // now combine the register address and the command into one byte:
  byte dataToSend = thisRegister | WRITE;

  // take the chip select low to select the device:
  digitalWrite(chipSelectPin, LOW);

  SPI.transfer(dataToSend); //Send register location
  SPI.transfer(thisValue);  //Send value to record into register

  // take the chip select high to de-select:
  digitalWrite(chipSelectPin, HIGH);
}
```

We don't have this sensor, but
try to have a look at SCP1000 datasheet to see how to read and write in registers

C4µC - SPI

- To have an idea of the SPI, let's look in Arduino IDE:
  - File --> Examples --> SPI --> BarometricPressureSensor

```
//Read from or write to register from the SCP1000:
unsigned int readRegister(byte thisRegister, int bytesToRead ) {
  byte inByte = 0;           // incoming byte from the SPI
  unsigned int result = 0;   // result to return
  //...
  byte dataToSend = thisRegister & READ;
  // take the chip select low to select the device:
  digitalWrite(chipSelectPin, LOW);
  // send the device the register you want to read:
  SPI.transfer(dataToSend);
  // send a value of 0 to read the first byte returned:
  result = SPI.transfer(0x00);
  // decrement the number of bytes left to read:
  bytesToRead--;
  if (bytesToRead > 0) { // if you still have another byte to read:
    // shift the first byte left, then get the second byte:
    result = result << 8;
    inByte = SPI.transfer(0x00);
    // combine the byte you just got with the previous one:
    result = result | inByte;
    // decrement the number of bytes left to read:
    bytesToRead--;
  }
  // take the chip select high to de-select:
  digitalWrite(chipSelectPin, HIGH);
  // return the result:
  return(result);
}
```



C4µC - SPI

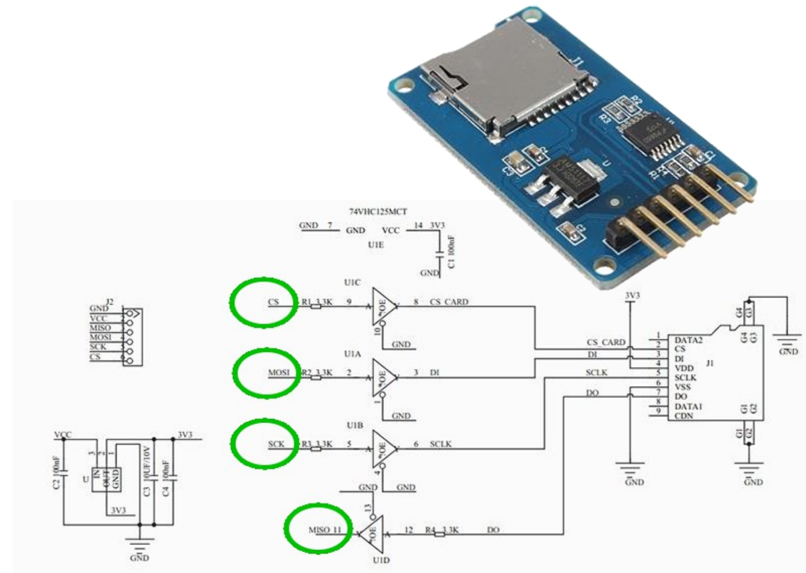
- On Atmega328 we have a hardware device for SPI (Just have a look at datasheet)
- An alternative software implementation could be:

```
// Simultaneously transmit and receive a byte on the SPI.Polarity and phase are assumed
// to be both 0, i.e.: - input data is captured on rising edge of SCLK.
//                     - output data is propagated on falling edge of SCLK.
// Returns the received byte.

uint8_t SPI_transfer_byte(uint8_t byte_out){
    uint8_t byte_in = 0;
    uint8_t bit;

    for (bit = 0x80; bit; bit >>= 1) {
        /* Shift-out a bit to the MOSI line */
        write_MOSI((byte_out & bit) ? HIGH : LOW);
        /* Delay for at least the peer's setup time */
        delay(SPI_SCLK_LOW_TIME);
        /* Pull the clock line high */
        write_SCLK(HIGH);
        /* Shift-in a bit from the MISO line */
        if (read_MISO() == HIGH)
            byte_in |= bit;
        /* Delay for at least the peer's hold time */
        delay(SPI_SCLK_HIGH_TIME);
        /* Pull the clock line low */
        write_SCLK(LOW);
    }
    return byte_in;
}
```

This is a bitbanging technique

C4µC - SPI examples

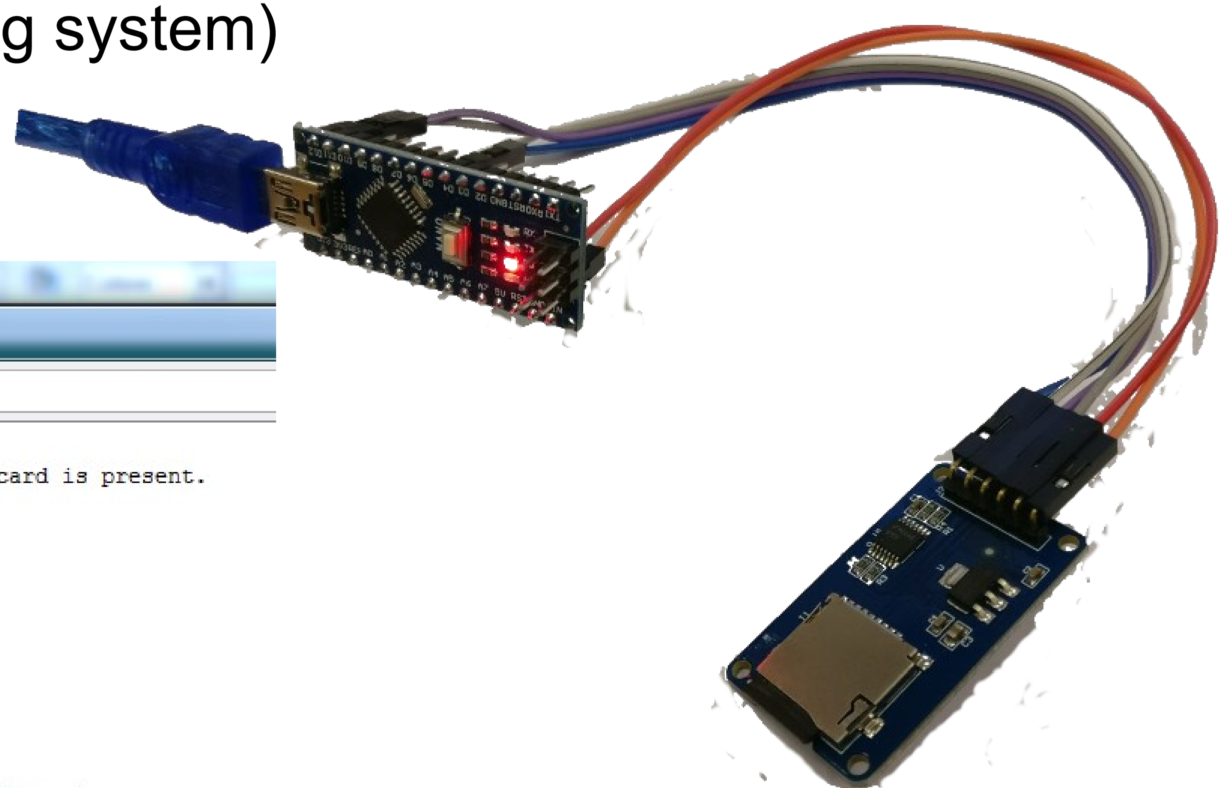
- SPI can be used to interface with an SD card:
  - Using a card adapter like...
  - We have access to the lines CS, MOSI, MISO, CLK
  - The "interfacing procedure" requires a longer initialization with different commands
  - Useful the library already in API

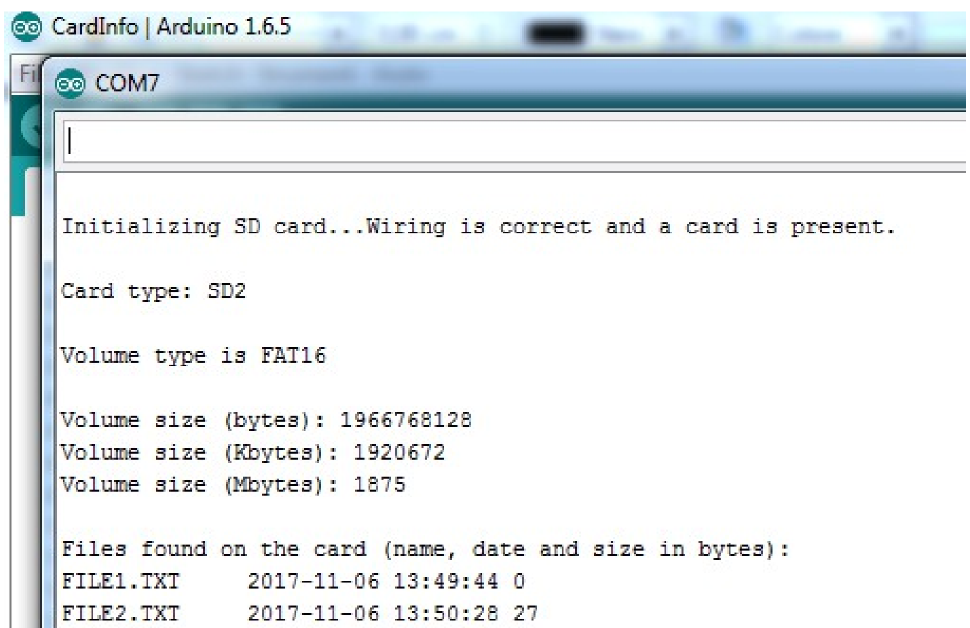


  - File --> Examples --> SD --> CardInfo
  - File --> Examples --> SD --> FileDump

C4µC - SPI examples

- SPI can be used to interface with an SD card:
  - The library is more complex, but offers functionalities at "filesystem level", so it's easy to create, read, write files (to use e.g for a log system)





C4µC - SPI examples

Questions on SPI and related example??