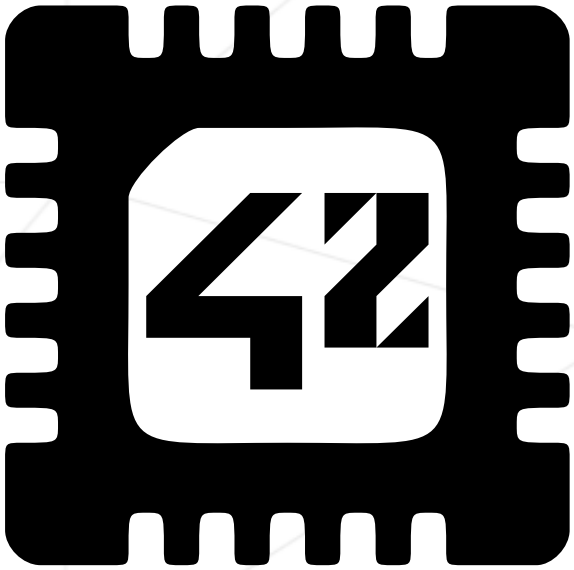

Electronics Project

Day 04 : I2C Protocol

contact@42chips.fr

Summary: Two wires to rule them all.

Chapter I

Introduction

The I2C protocol (Inter-Integrated Circuit) or TWI (Two Wire Interface) is a communication bus used to connect devices to a microcontroller or computer. It was developed by Philips Semiconductors in the 1980s and has become an industry standard widely used in various fields, such as consumer electronics, automotive, aerospace, and communications.

The I2C protocol allows multiple devices to communicate on a single bus using two wires: a data wire (SDA) and a clock wire (SCL).
Only one device is allowed to send data on the bus at a time, but multiple devices can be connected to the bus and addressed individually.
This reduces costs by using fewer wires and simplifying printed circuit boards.

The I2C protocol uses a master-slave scheme for communication.
The device that initiates communication is the master, while the devices that receive commands are the slaves.
The master sends a series of address bits to each slave to identify the target device, and then sends data to or reads data from the target device.

The I2C protocol also allows for point-to-point data communication, meaning the master can communicate directly with a single slave at a time.
This can be useful in applications where there are many devices on the bus and communication latency needs to be minimized.

In addition to data communication, the I2C protocol supports several other features, such as data collision detection, power management, and device self-identification. All of these features make the I2C protocol a popular choice for many data communication applications.

The I2C protocol is particularly useful in applications with multiple devices and limited space, as it only requires two wires for communication. It is also easy to implement and use, making it a popular choice for many projects.

Chapter II

General instructions

Unless explicitly stated otherwise, the following instructions will be valid for all assignments.

- The language used for this project is C.
- It is not necessary to code according to the 42 norm.
- The exercises are ordered very precisely from the simplest to the most complex. Under no circumstances will we consider or evaluate a complex exercise if a simpler one is not perfectly successful.
- You must not leave any files other than those explicitly specified by the exercise instructions in your directory during peer evaluation.
- All technical answers to your questions can be found in the `datasheets` or on the Internet. It is up to you to use and abuse these resources to understand how to complete your exercise.
- You must use the datasheet of the microcontroller provided to you and comment on the important parts of your program by indicating where you found the clues in the document, and if necessary, explaining your approach. Don't write long blocks of text, keep it clear.
- Do you have a question? Ask your neighbor to the right or left. You can ask in the dedicated channel on the Piscine's Discord, or as a last resort, ask a staff member.

Chapter III

Je vous ai compris!

	Exercise 00
Two Wire Interface	
Turn-in directory : *ex*00/	
Files to turn in : `Makefile, main.c`	
Allowed functions : `avr/io.h, util/twi.h`	
Notes : `n/a`	

- The AVR ATmega328P microcontroller has 1 I2C peripheral that you must use in this exercise to communicate with an AHT20 temperature sensor (U3).
- You must write a function `i2c_init` that initializes the I2C on the microcontroller.
- The MCU's I2C must be configured so that the communication frequency is 100kHz.
- You must write a function `i2c_start` that starts an I2C transmission between the microcontroller and the sensor and prepares it in a write mode.
- The program must return status values to your computer after each data transmission.
- You must write a function `i2c_stop` that interrupts communication between the microcontroller and the sensor.

```
void i2c_init(void)
void i2c_start(void)
void i2c_stop(void)
```

	Exercise 01
	Brute Data
Turn-in directory : *ex*01/	
Files to turn in : `Makefile, main.c`	
Allowed functions : `avr/io.h, util/twi.h, util/delay.h`	
Notes : `n/a`	

- Complete the previous program by adding a function `i2c_write` which will write the contents of the TWDR register of the microcontroller and send it to the temperature sensor.
- You must write a function `i2c_read` which will display the contents of the TWDR register after the sensor measurement.
- You must write a function `print_hex_value` which will write the contents of the 7 bytes of an AHT20 sensor measurement without modification to the standard output of your PC.
- You must display the return values in a loop that repeats with a frequency that meets the manufacturer's recommendations.

```
void i2c_write(unsigned char data)
void i2c_read(void)
void print_hex_value(char c)
```

```
0C 79 9A A6 4E 3C F2
0C 79 83 76 4E 29 78
0C 79 8D 36 4E 19 5A
0C 79 7E 06 4E 82 91
```

Be careful with the delay required between the end of the write command and the data read.

	Exercise 02
	It's heating up!
Turn-in directory :	*ex*02/
Files to turn in :	`Makefile, main.c`
Allowed functions :	`avr/io.h, util/twi.h, util/delay.h, dtostrf()`
Notes :	`n/a`

- Complete the previous program to display the temperature and humidity in place of the raw values obtained earlier.
- The temperature and humidity values should be displayed according to the following specifications:
  - the precision displayed on the terminal must respect the recommendations given in the datasheet, rounded to the nearest ten.
    (e.g.: 0.0031 -> 0.01 or 0.1 -> 0.1)
  - the temperature will be in degrees Celsius (.C), the humidity in percent (%).
  - the displayed measurement must be the average of the last 3 measurements taken by the program. The behavior for the first two measurements should be consistent with this rule.

```
Temperature: 17.C, Humidity: 43,4%
Temperature: 18.C, Humidity: 43,6%
Temperature: 18.C, Humidity: 43,5%
Temperature: 17.C, Humidity: 43,8%
```

`Note that in the above example, the resolution and precision are not those requested and are only provided to visualize the result display as requested in the exercise.`