Hardware Software Platforms Project Presentation

Measuring temperature on TMP100

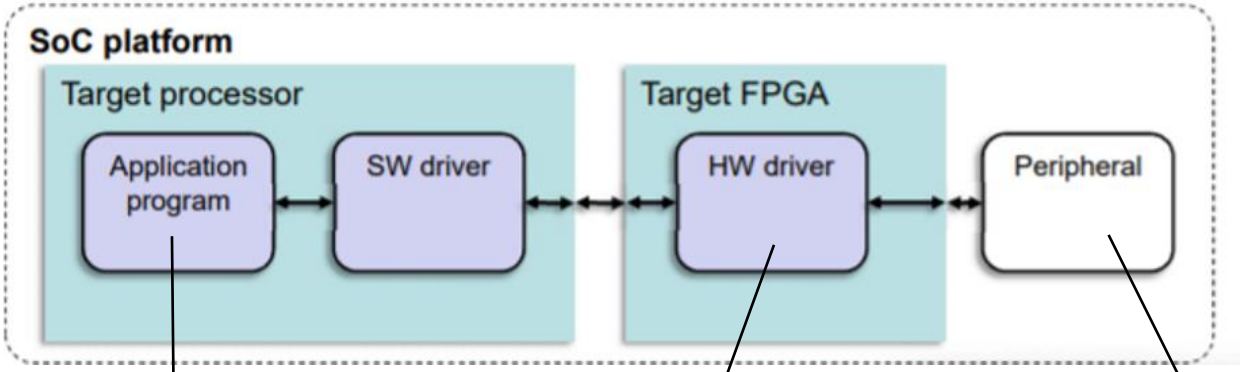
NAJI Doha
(doha.naji@student.umons.ac.be)
HOEDENAEKEN Clément
(clement.hoedenaeken@student.umons.ac.be)

Table of contents

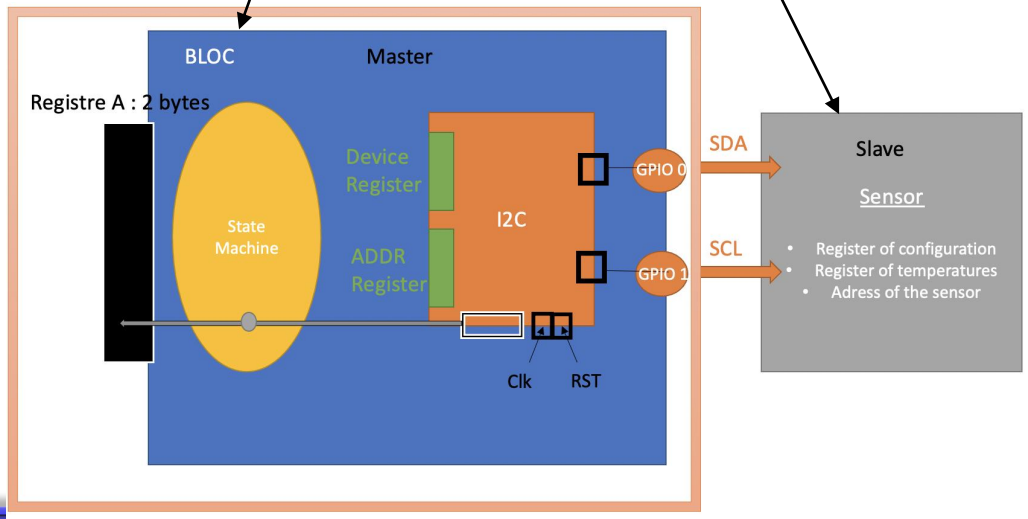
A) Introduction

- Objective : create a tutorial based on Quartus and DE0 kits

→ interaction with peripheral

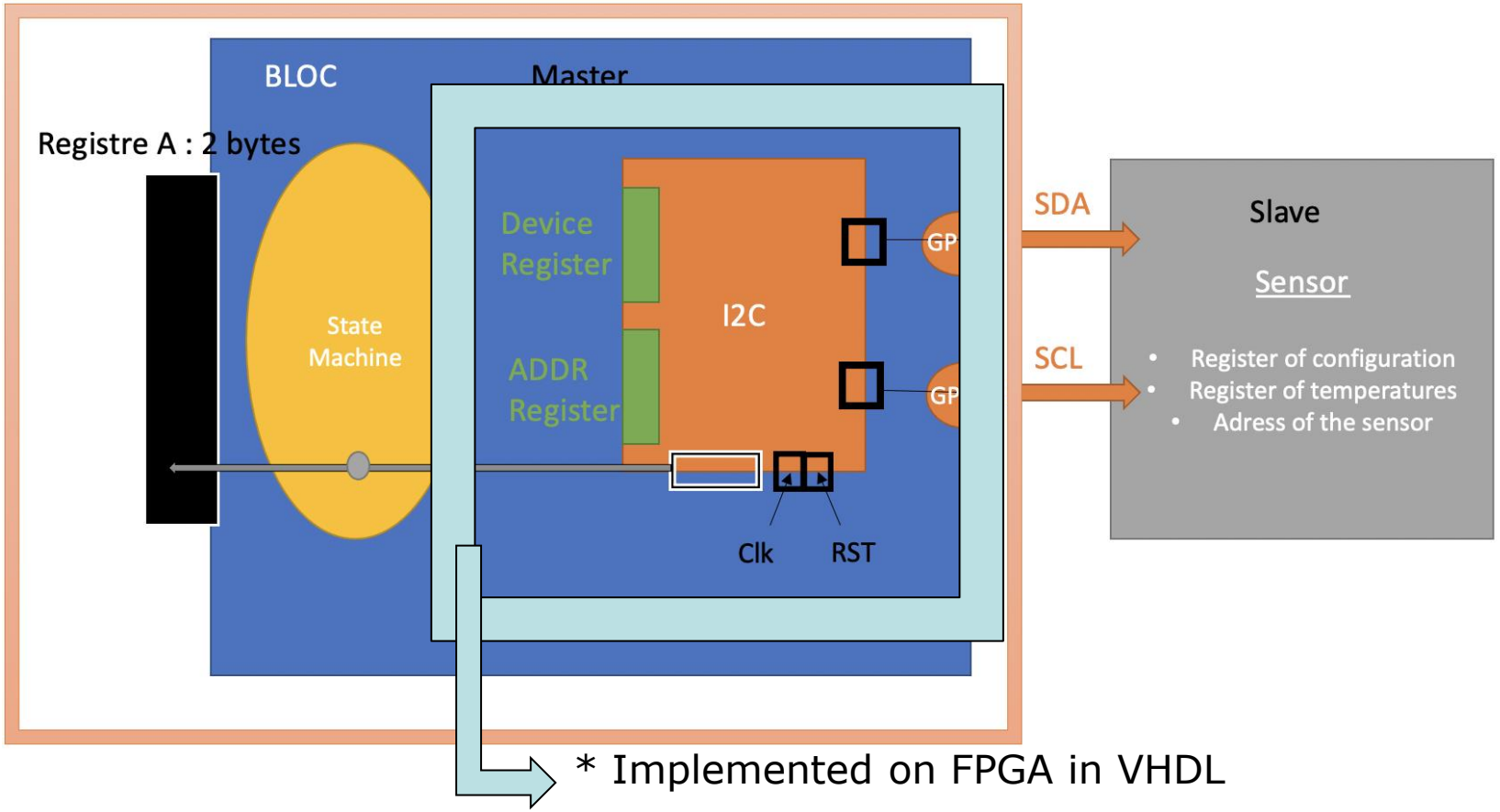


Control driver and interaction with the environment



B) Steps of the design

1. I2C driver : context

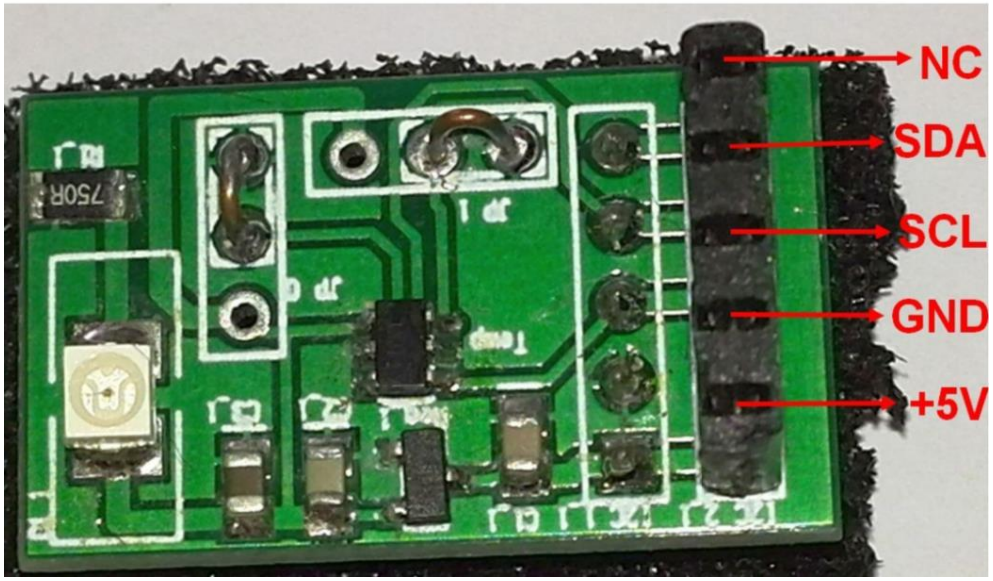


* Implemented on FPGA in VHDL
* Pilots the TMP100 Temperature Sensor
* Uses I2C protocol and bus

1. I2C driver : I2C protocol and bus

➡ Half-duplex bidirectional synchronous serial bus

- One element speaks at a time
- Communication possible in both direction
- Clock signal to synchronize the communication
- One bit sent at a time



2 wire-interface:

➡ 1 wire for the Data : SDA

➡ 1 wire for the Clock : SCL

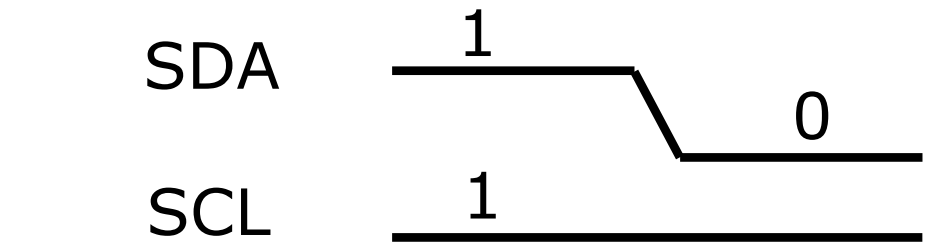
+ 2 for power supply

1. I2C driver : I2C protocol and bus

The first who takes the control of the bus becomes the master. It generates the start Condition. The other one becomes the slave.

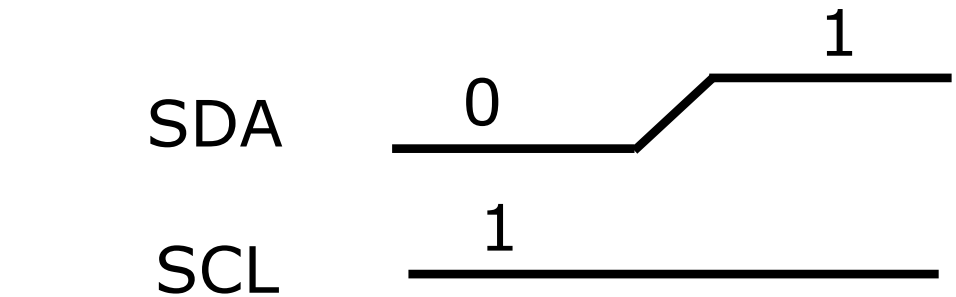
Star Condition

(Bus busy)



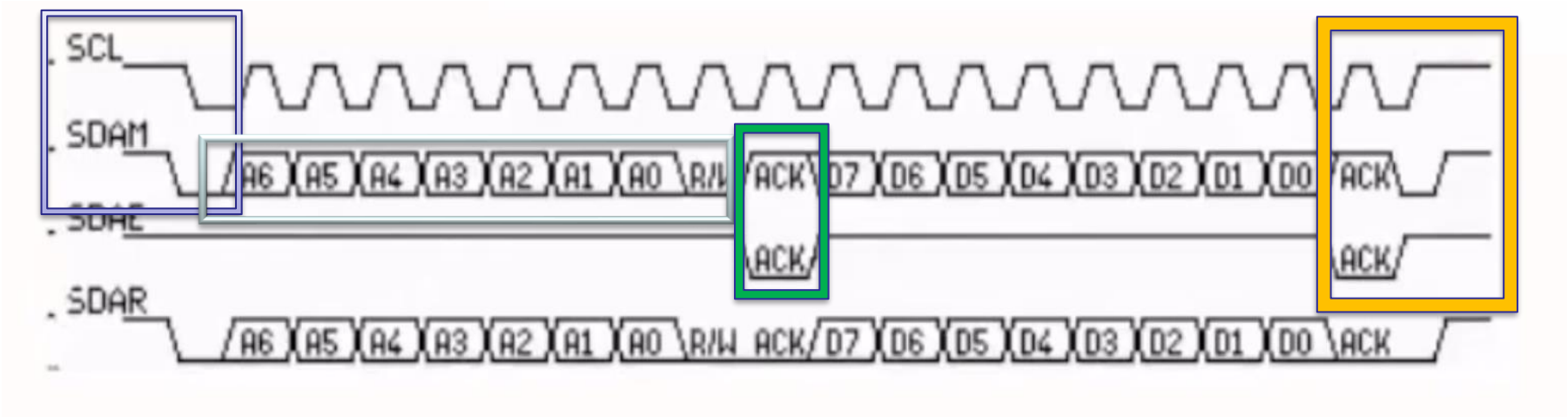
Stop Condition

(Bus free)



1. I2C driver : I2C protocol and bus

1) Master ⇨ Slave



⇨ Start Condition : The master puts the line to 0 + generation of the clock

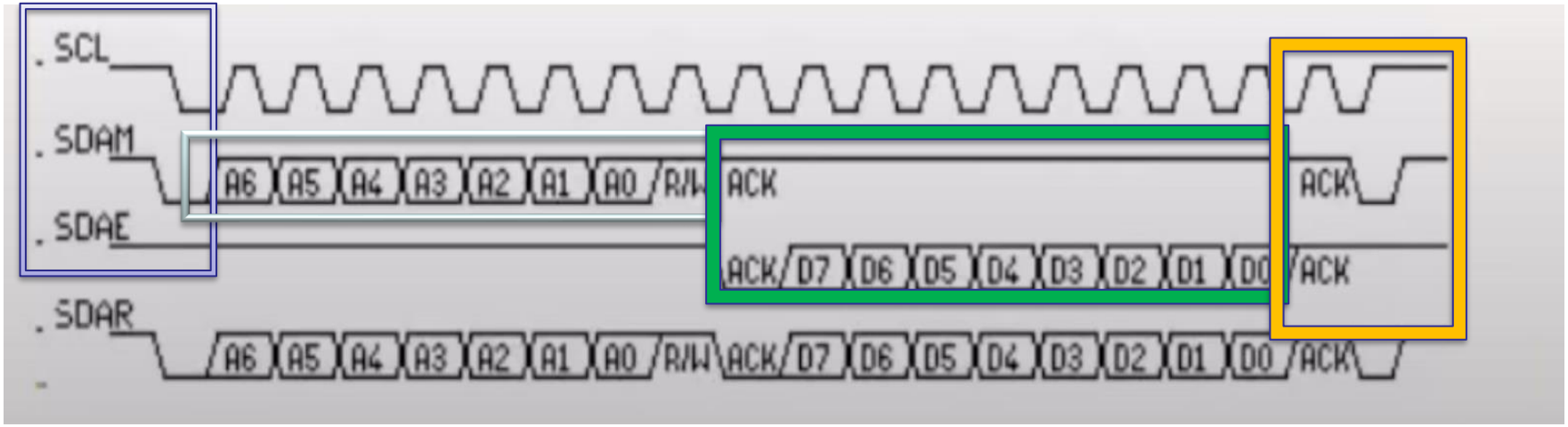
⇨ At each rising edge, sending of a bit of the slave address (8 bits : 8th bit = R/W)

⇨ The master puts the line to 1 untill the slave puts the line back to 0 (ACK) : the communication begins at the next rising edge.

⇨ The master puts the line to 1 untill the slave puts it back to 0: ACK --> Stop Condition

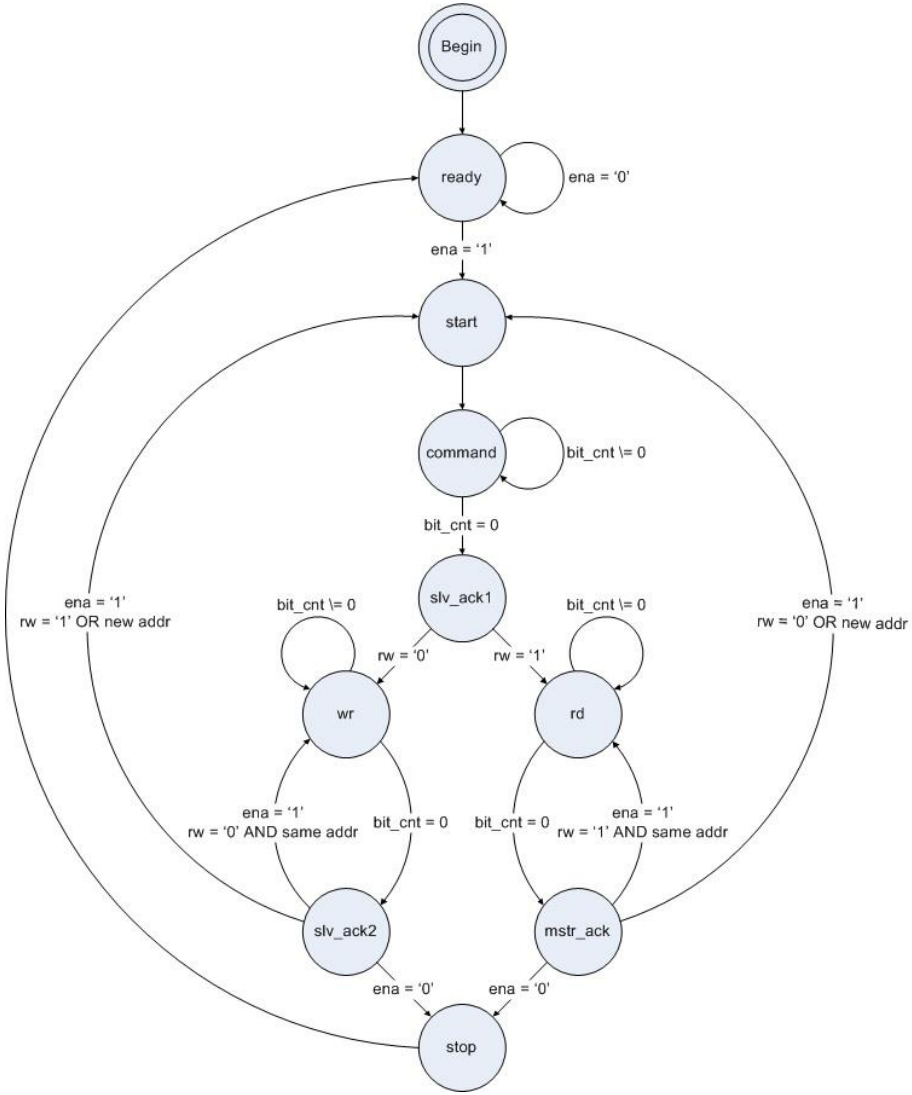
1. I2C driver : I2C protocol and bus

2) Slave ⇨ Master



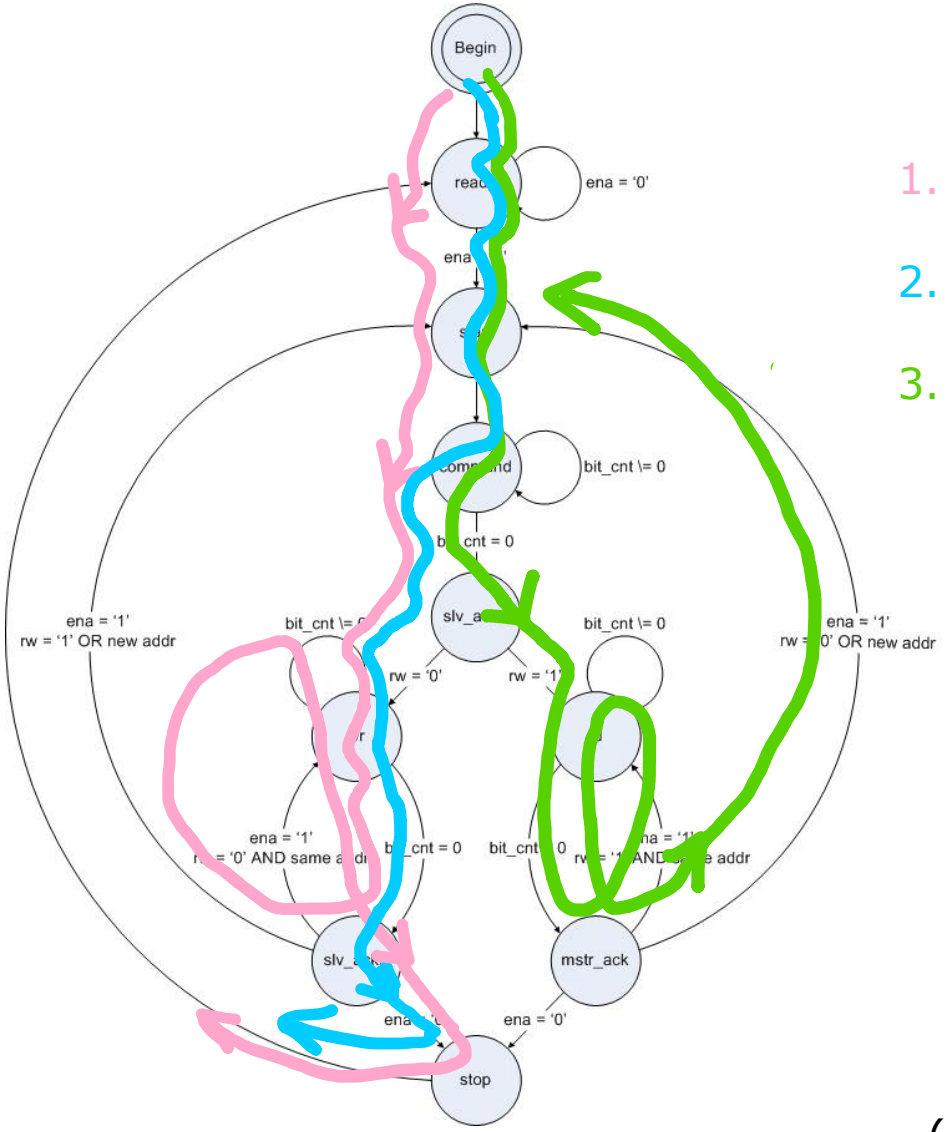
- Start Condition (similary to the master/slave communication)
- At each rising edge, sending of a bit of the address of the slave
- The Master puts the line into1 untill the slave puts back the line to zéro (ACK) + sending of the data
- The slave puts the line to 1 untill the master puts the line back to 0 : ACK --> Stop Condition

1. I2C driver : I2C protocol and bus



- Start condition
- Slave address writing
- Writing loop
- Reading loop
- End condition

1. I2C driver : I2C protocol and bus



1. Writing the configuration
2. Connecting the temperature register
3. Reading the temperature

(Given by the sensor datasheet)

1. I2C driver : configuration of the sensor

1. Slave address : 1001 000
2. Configuration of the sensor : 12 bits

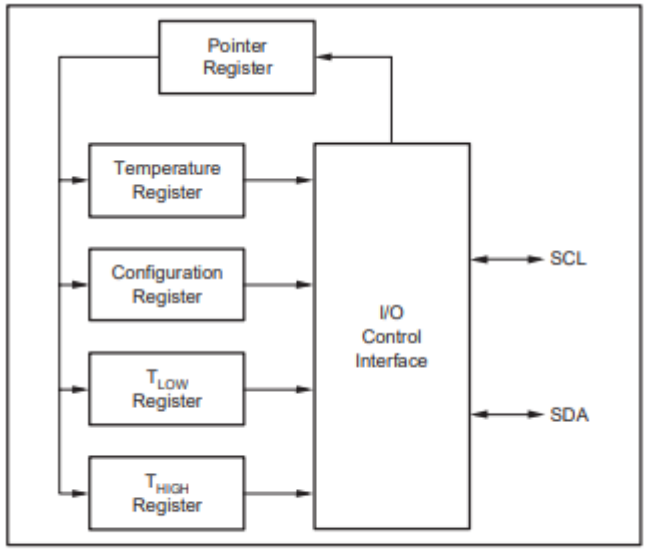


Figure 3. Internal Register Structure of the TMP100 and TMP101

- Write address of config register

Table 1. Pointer Register Type

P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	Register Bits	

Table 2. Pointer Addresses of the TMP100 and TMP101 Registers

P1	P0	REGISTER
0	0	Temperature Register (READ Only)
0	1	Configuration Register (READ/WRITE)
1	0	T_{LOW} Register (READ/WRITE)
1	1	T_{HIGH} Register (READ/WRITE)

- Write the config (01100000)

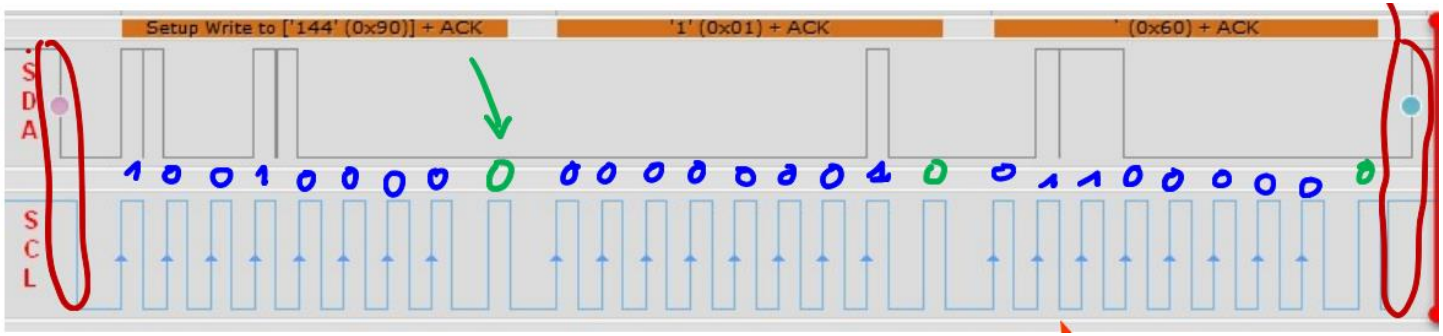
Table 6. Configuration Register Format

BYTE	D7	D6	D5	D4	D3	D2	D1	D0
1	OS/ALERT	R1	R0	F1	F0	POL	TM	SD

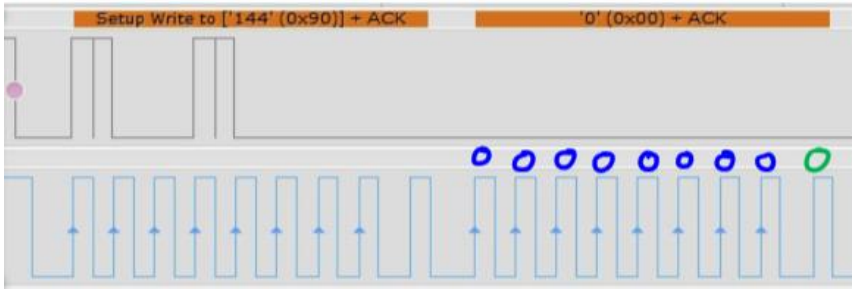
- Write the address of temp register
- Read temp register

1. I2C driver : the signals we want to produce

Writing the configuration

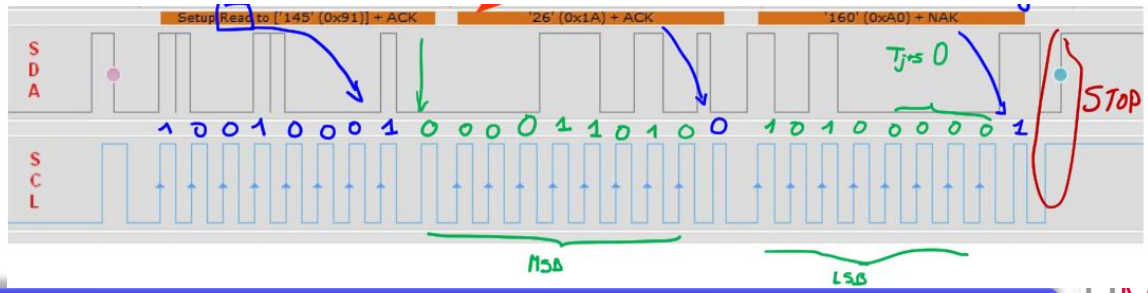


Writing connecting the temperature register

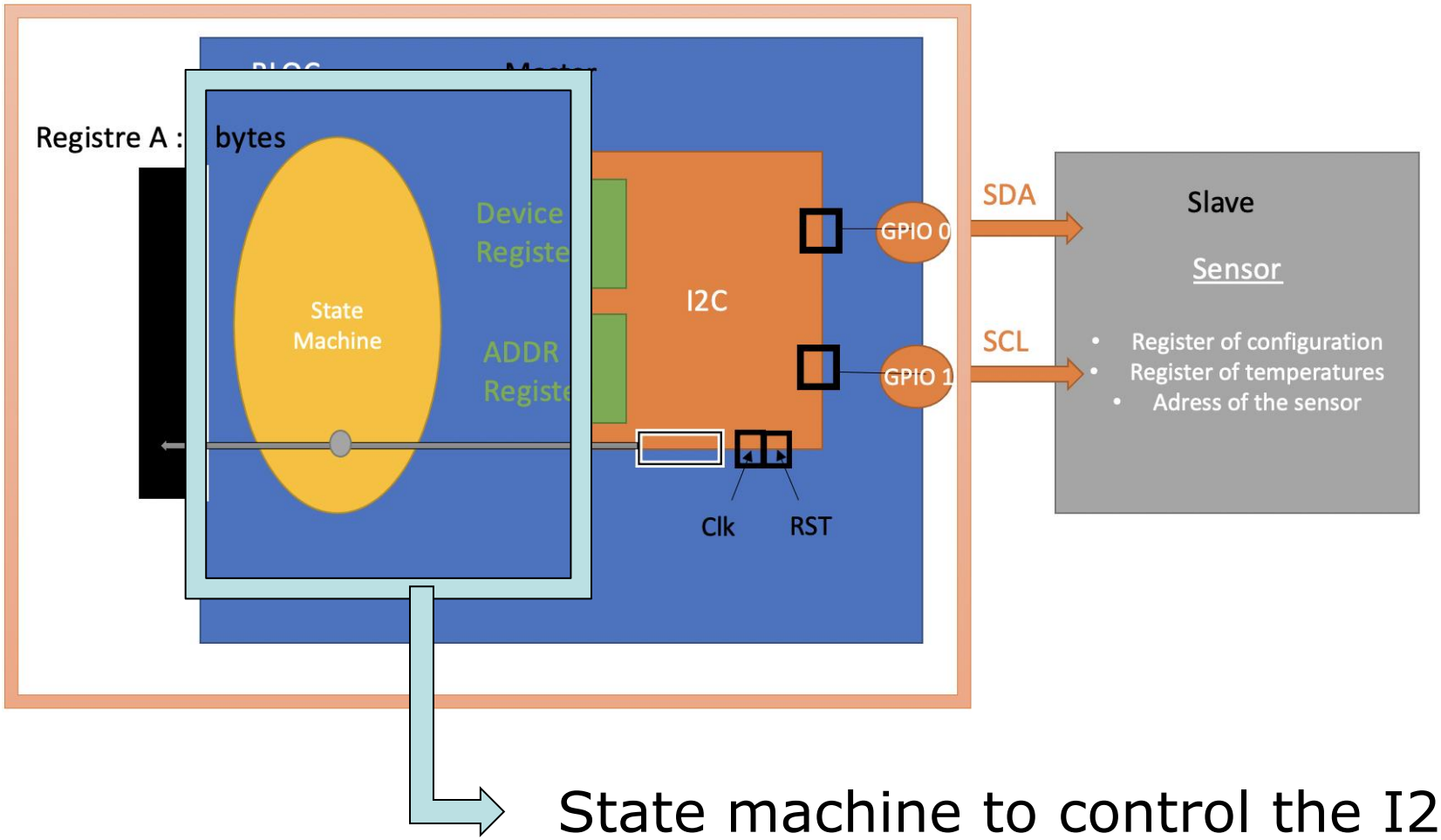


Delay : min 320 ms

Reading the temperature



2. Bloc (control state machine)



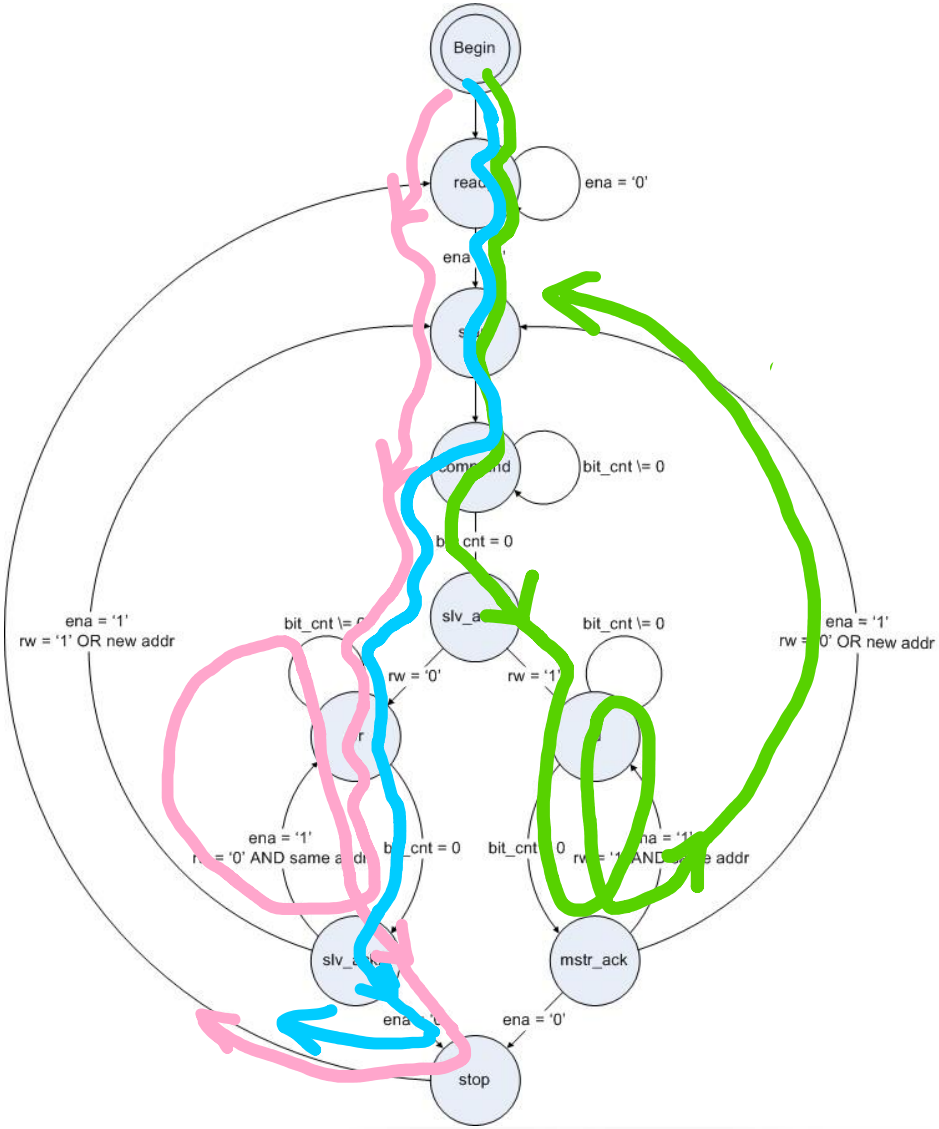
State machine to control the I2C driver and process the data

2. Bloc (control state machine)

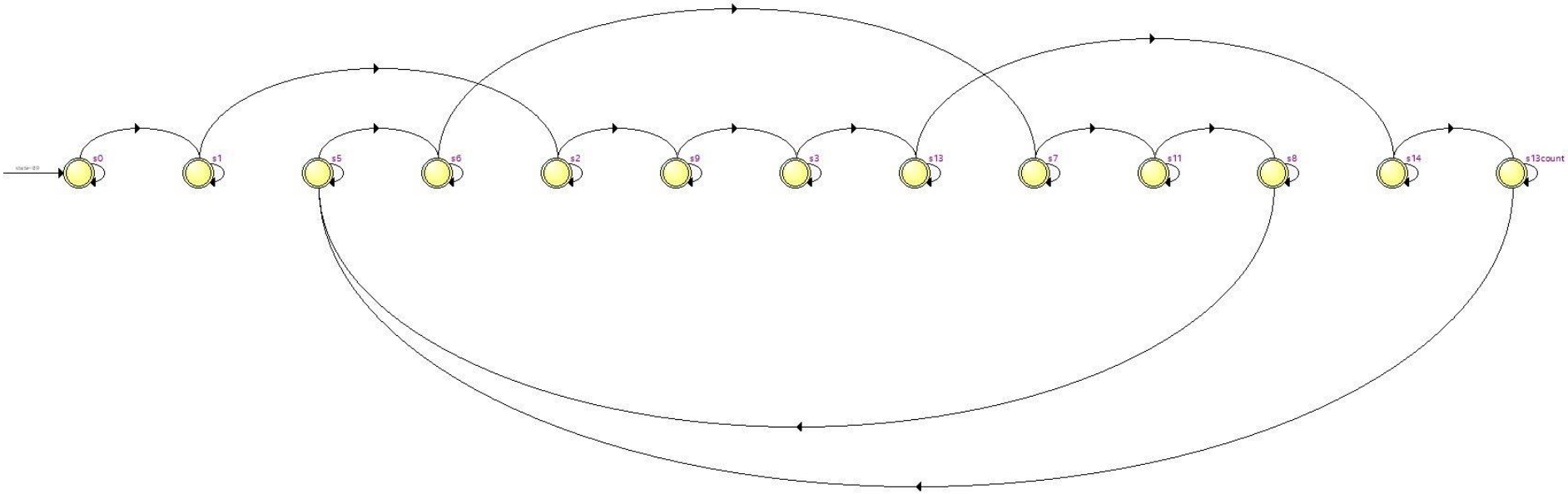
Objective : Drive this state machine (follow the colored paths)

Action : Place values in the registers and activate right signals at the right moment (e.g. : enable, rw,…)

Help : Flags raised by the machine (e.g. : busy, val_rdy, reg_rdy,…)

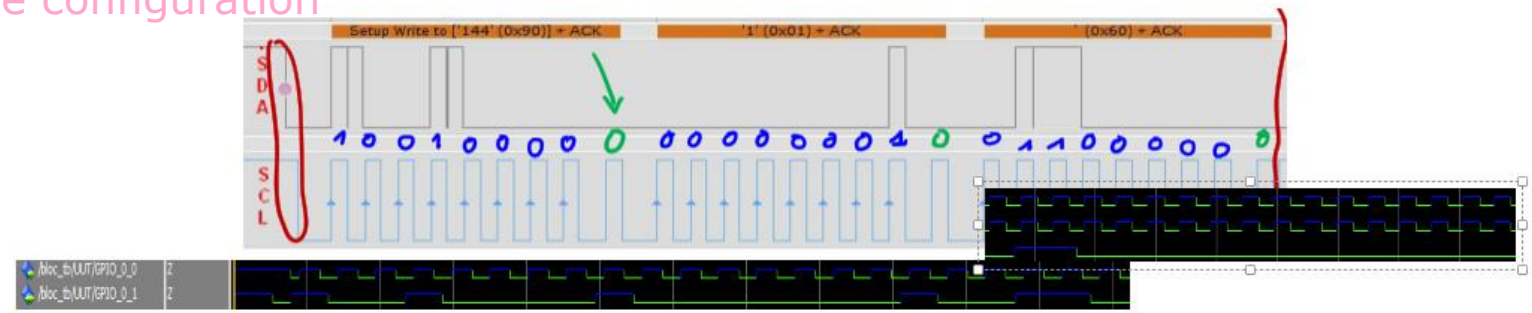


2. Bloc (control state machine)

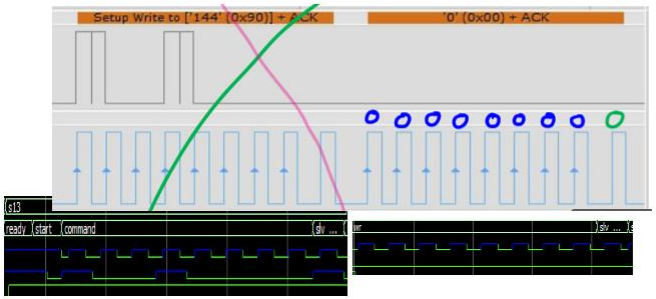


2. Bloc (control state machine) : testing it with a testbench

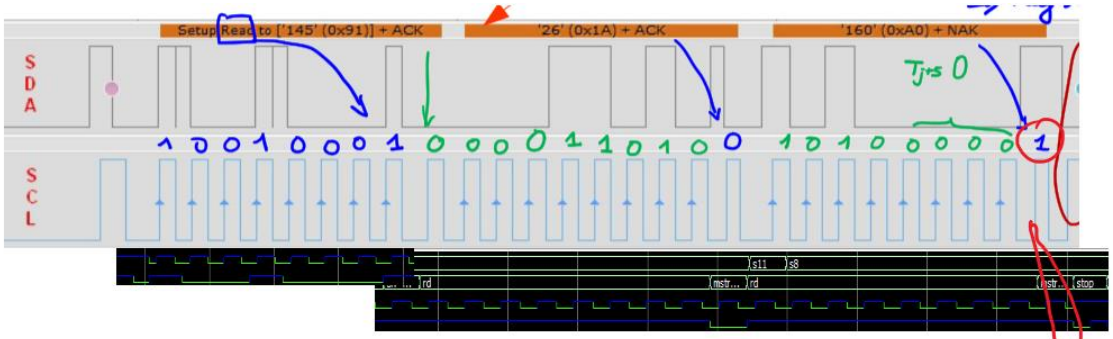
Writing the configuration



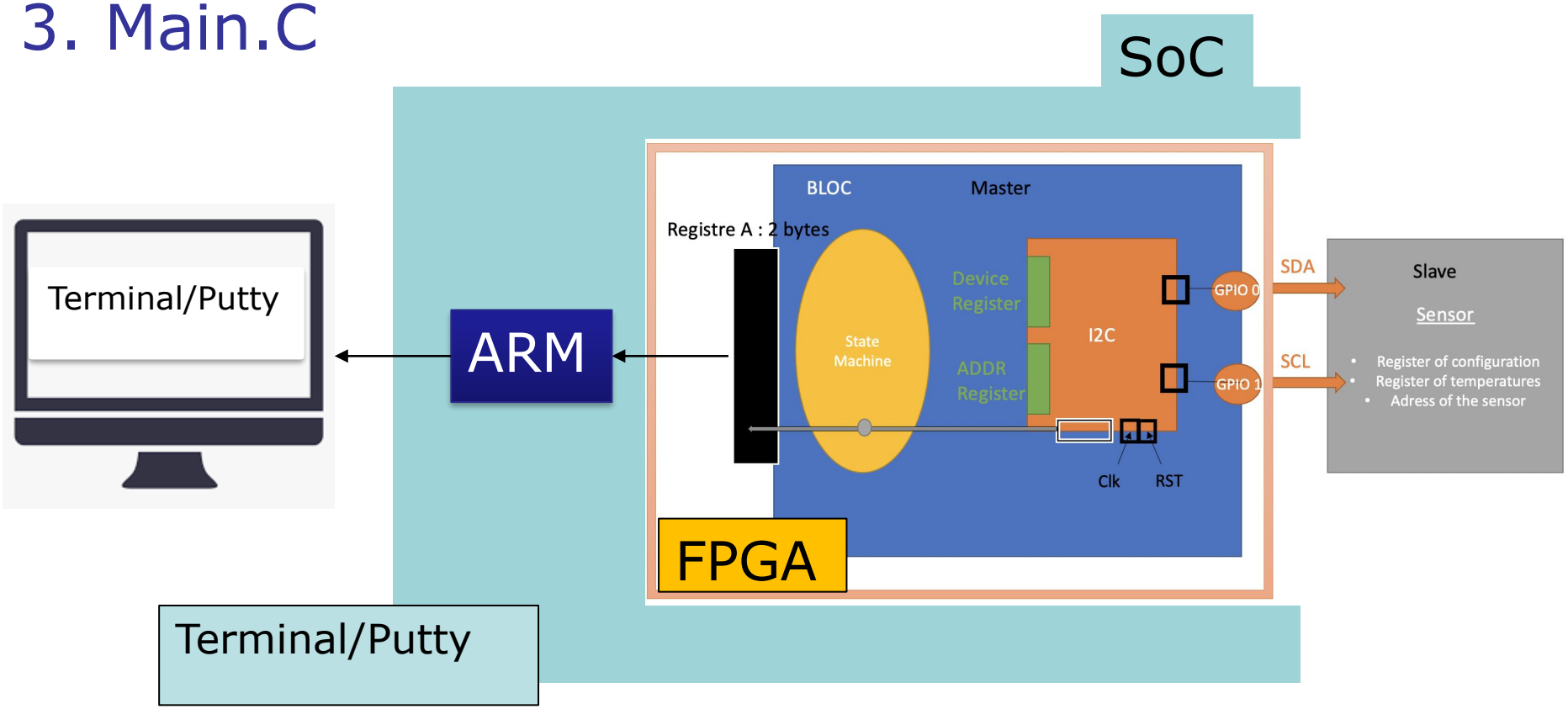
Writing connecting the temperature register



Reading the temperature



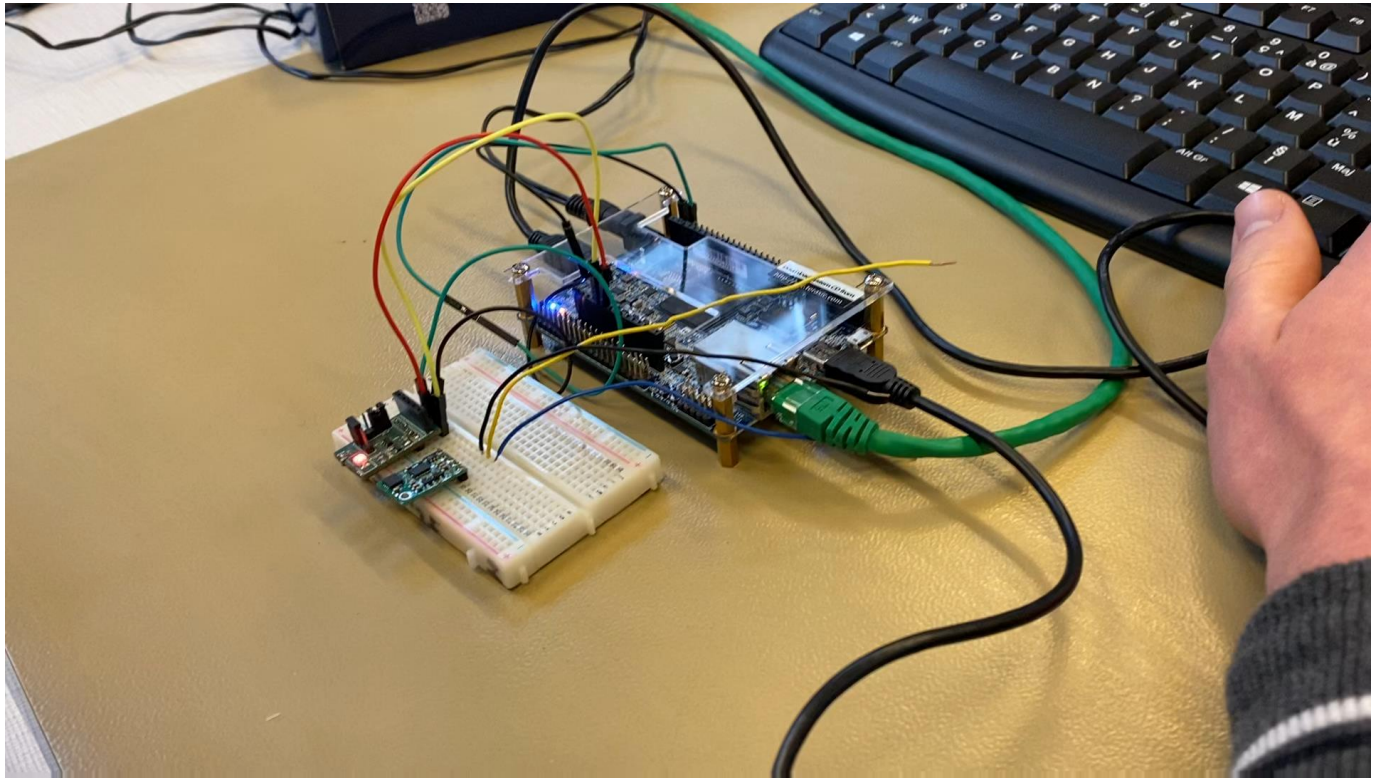
3. Main.C



Serial Protocol Communication

```
printf("Temperature (entier): %d\n", *((int8_t *)h2p_lw_reg_out_addr));
printf("\n");
printf("Temperature (précis) : %.4f \n", temperature);
printf("\n");
```

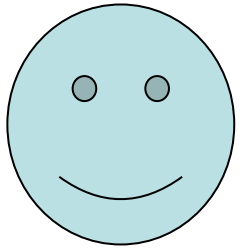
C) Demonstration video



D) Conclusion

	Working	Tested
Driver I²C	✓	Modelsim : ✓
Control State Machine (Bloc)	✗ (clock not working)	Modelsim : ✓ Oscilloscope : ✗
Code in C	✓ (reads « 0°C »)	✓

Thank you !



Questions?