Sofiane Rachdaoui – Tom Bollard

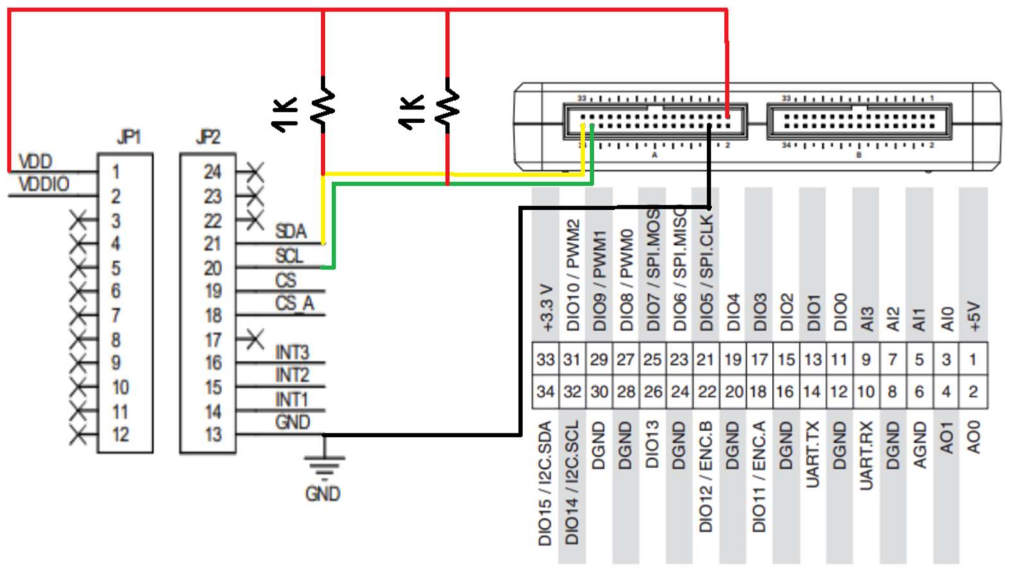
Project Report

DATASHEET AND EXPLANATION:

Wiring between NI myRio(master) and steval-mki172v1 (slave):

Here is the wiring explanation:

- Red wire is connected to the power supply of myRio +5V, the steval-mki172v1 needs at least 1,8V to work. It is also connected to SDA and SCL to have the acknowledge.
- Black wire is connected to the analog ground of myRio.
- Yellow wire is connected to the SDA (Serial Data Line) pin.
- Green wire is connected to the SCL (Serial Clock Line) pin.



Here is the default address for the accelerometer (LSM303AGR) is 0011001b (b means R/W bit):

Linear acceleration sensor: the default (factory setting) 7-bit slave address is 0011001b.

Table 24. SAD + read/write patterns

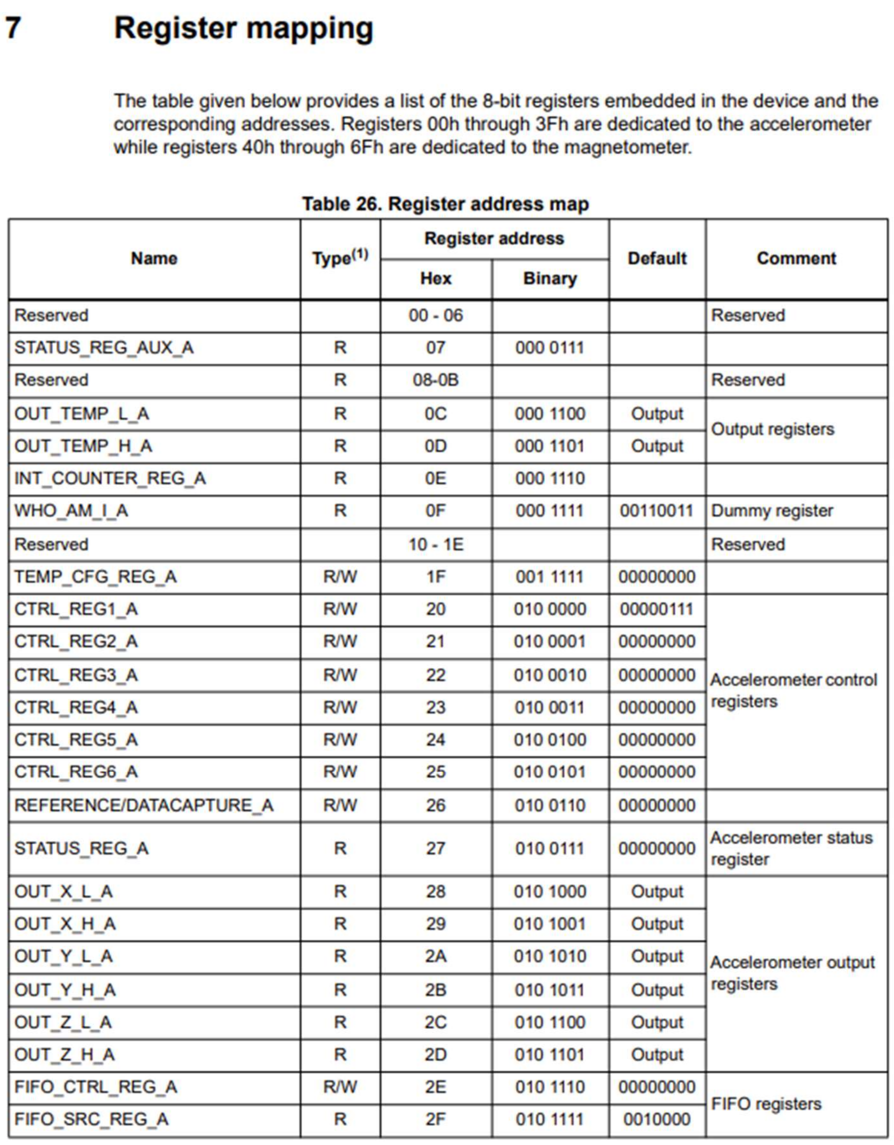
Command	SAD[6:0]	R/W	SAD + R/W
Read	0011001	1	00110011 (33h)
Write	0011001	0	00110010 (32h)

Table 22. Transfer when master is receiving (reading) one byte of data from slave

Master	ST	SAD + W		SUB		SR	SAD + R			NMAK	SP
Slave			SAK		SAK			SAK	DATA		

33h=51d

32h=50d



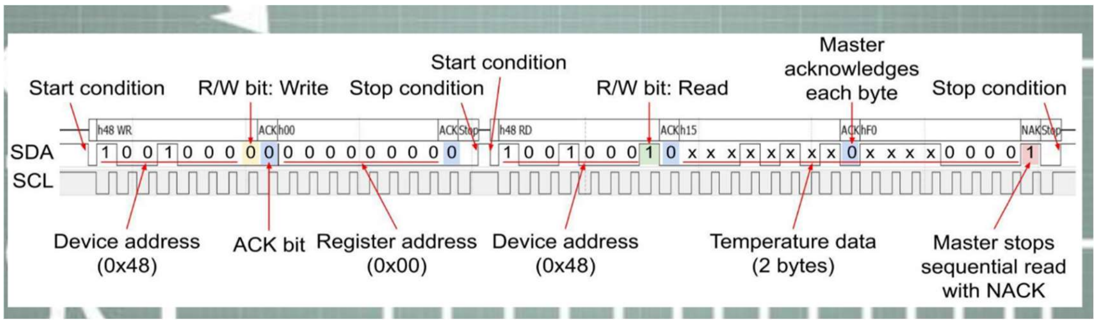
7 Register mapping

The table given below provides a list of the 8-bit registers embedded in the device and the corresponding addresses. Registers 00h through 3Fh are dedicated to the accelerometer while registers 40h through 6Fh are dedicated to the magnetometer.

Table 26. Register address map

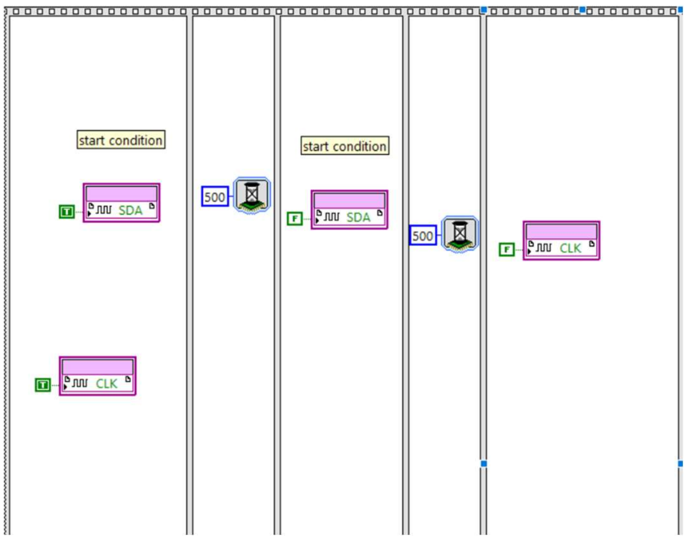
Name	Type[1]	Register address		Default	Comment
		Hex	Binary		
Reserved		00 - 06			Reserved
STATUS_REG_AUX_A	R	07	000 0111		
Reserved	R	08-0B			Reserved
OUT_TEMP_L_A	R	0C	000 1100	Output	Output registers
OUT_TEMP_H_A	R	0D	000 1101	Output	
INT_COUNTER_REG_A	R	0E	000 1110		
WHO_AM_I_A	R	0F	000 1111	00110011	Dummy register
Reserved		10 - 1E			Reserved
TEMP_CFG_REG_A	R/W	1F	001 1111	00000000	
CTRL_REG1_A	R/W	20	010 0000	00000111	Accelerometer control registers
CTRL_REG2_A	R/W	21	010 0001	00000000	
CTRL_REG3_A	R/W	22	010 0010	00000000	
CTRL_REG4_A	R/W	23	010 0011	00000000	
CTRL_REG5_A	R/W	24	010 0100	00000000	
CTRL_REG6_A	R/W	25	010 0101	00000000	
REFERENCE/DATACAPTURE_A	R/W	26	010 0110	00000000	
STATUS_REG_A	R	27	010 0111	00000000	Accelerometer status register
OUT_X_L_A	R	28	010 1000	Output	Accelerometer output registers
OUT_X_H_A	R	29	010 1001	Output	
OUT_Y_L_A	R	2A	010 1010	Output	
OUT_Y_H_A	R	2B	010 1011	Output	
OUT_Z_L_A	R	2C	010 1100	Output	
OUT_Z_H_A	R	2D	010 1101	Output	
FIFO_CTRL_REG_A	R/W	2E	010 1110	00000000	FIFO registers
FIFO_SRC_REG_A	R	2F	010 1111	0010000	

This is an example of an I2C bus. Of course, it isn't the same register and device addresses and not the same data.

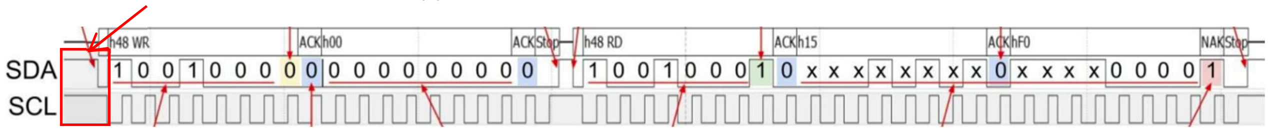


Labview VI :

The screen below is the start condition.

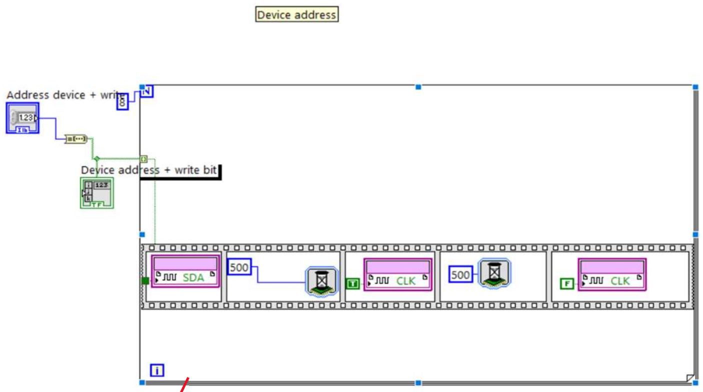


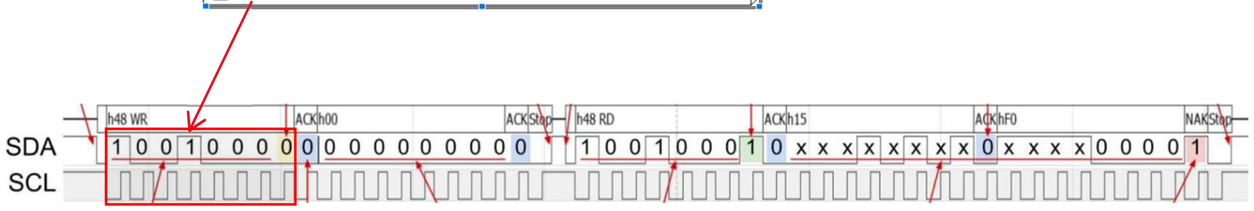
I framed in red the translation of the upper screen.



At the start the SDA and CLK are set to True, and on the second state we had a clock timer of 0,5s who set the SDA to false, who means that the frame is starting and after the second 0,5s clock timer the CLK is also set to false.

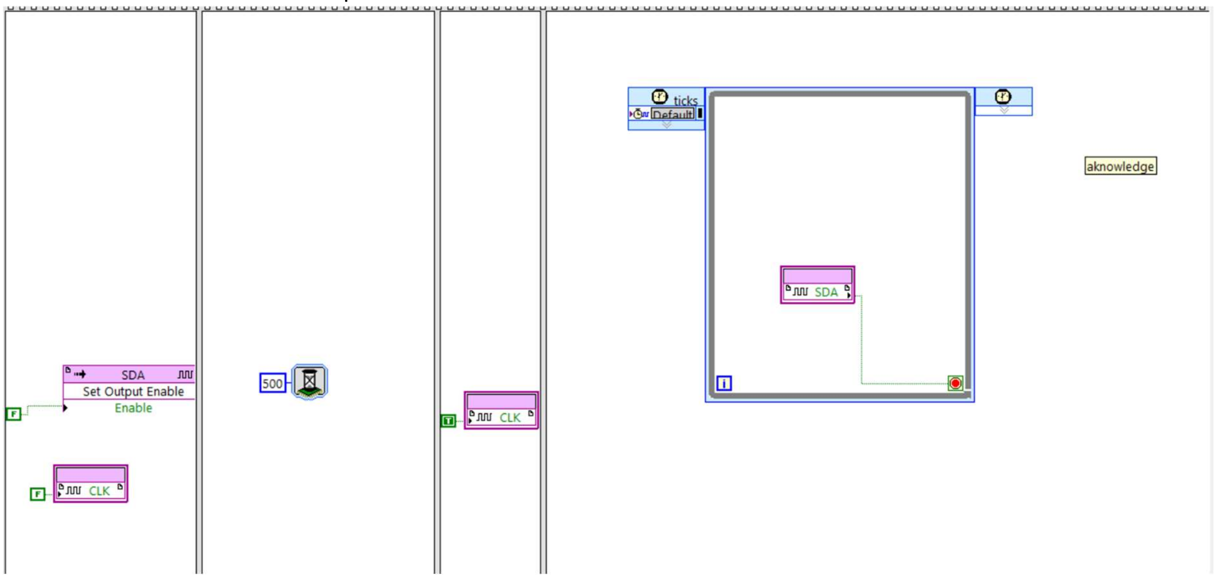
Device address + write

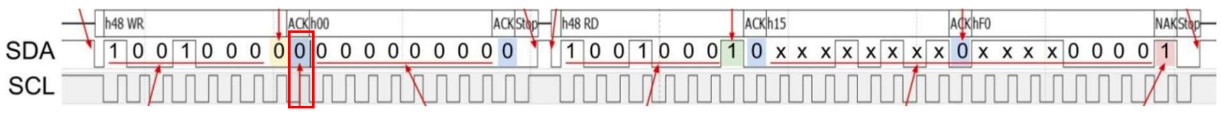




The address device is set to (50)10 = (00110010)2. On the other part of the VI screen we have the CLK function who altern between HIGH and LOW state.

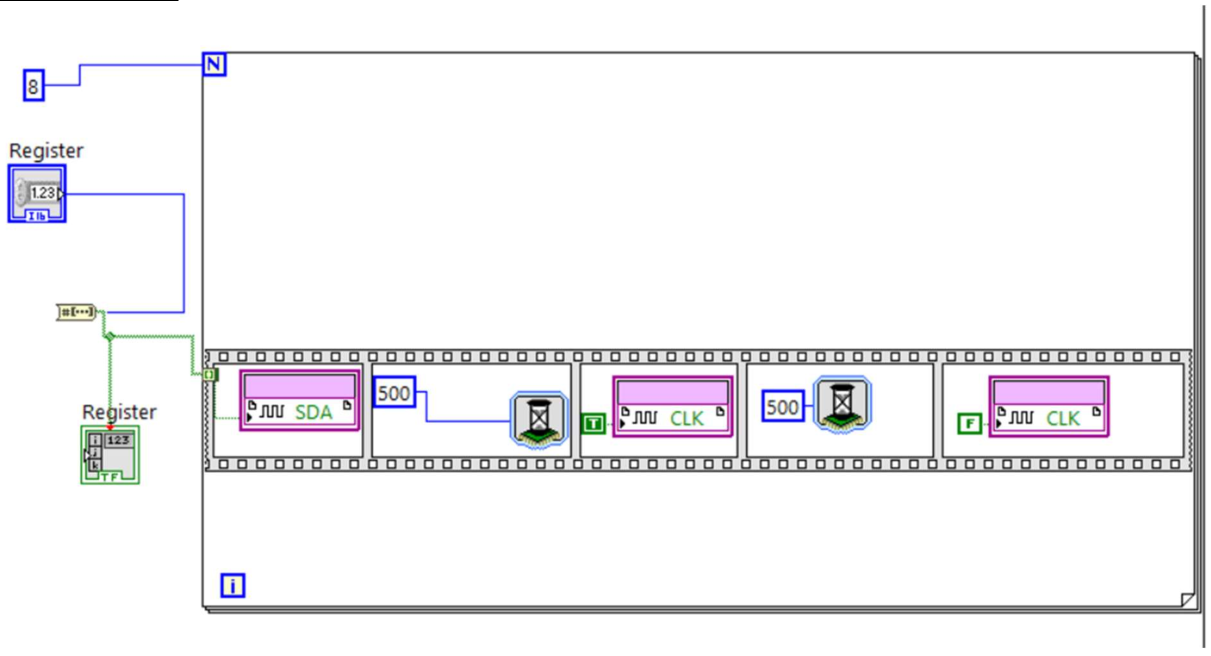
This is the VI screen who correspond to the ACK bit

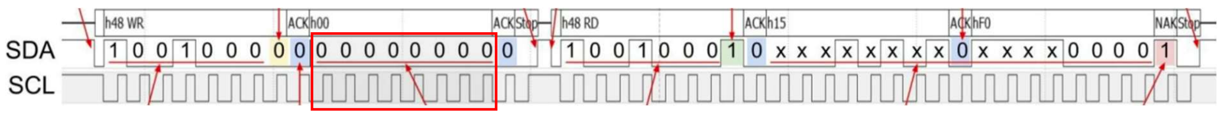




We change the state of SDA to enable so that we wait for an answer from the slave if the acknowledge is 0. If so, we carry on the next step.

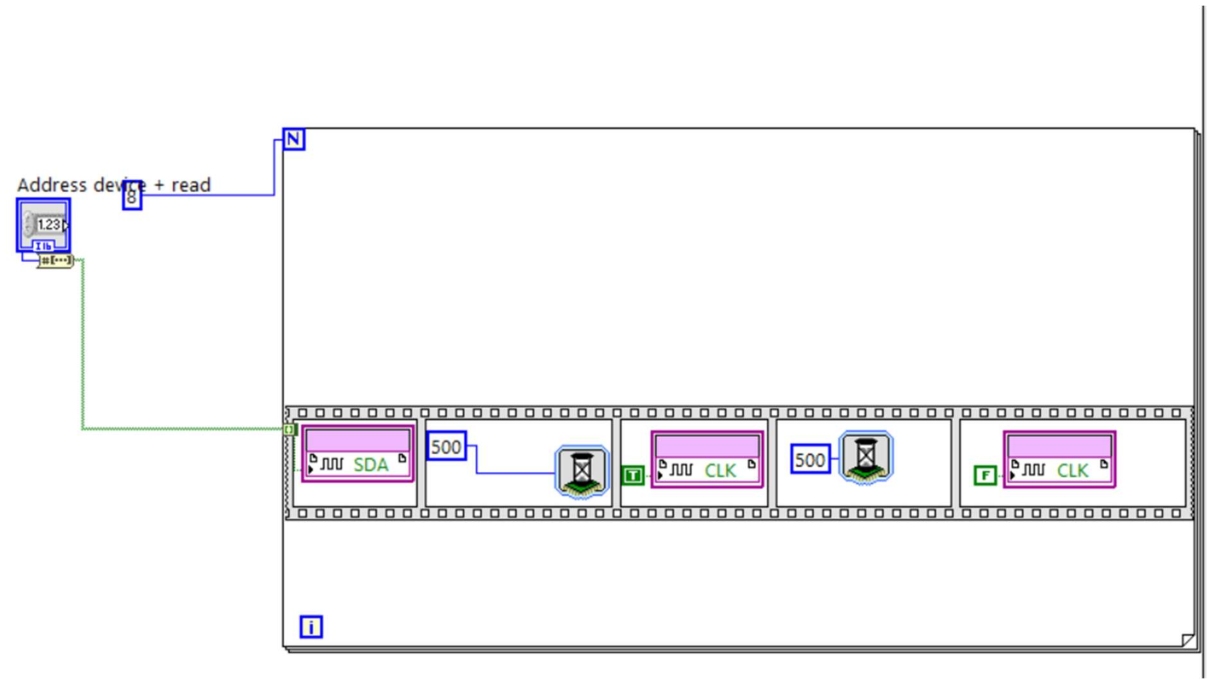
Register address

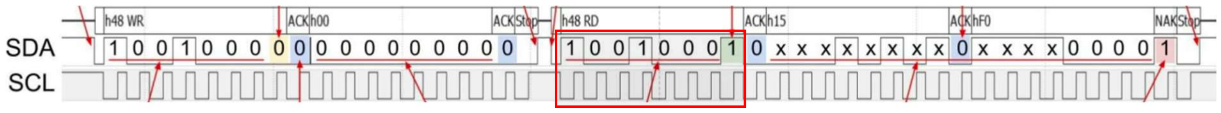




The default address for the register is (15)10 and once again we had the functionment of the CLK.

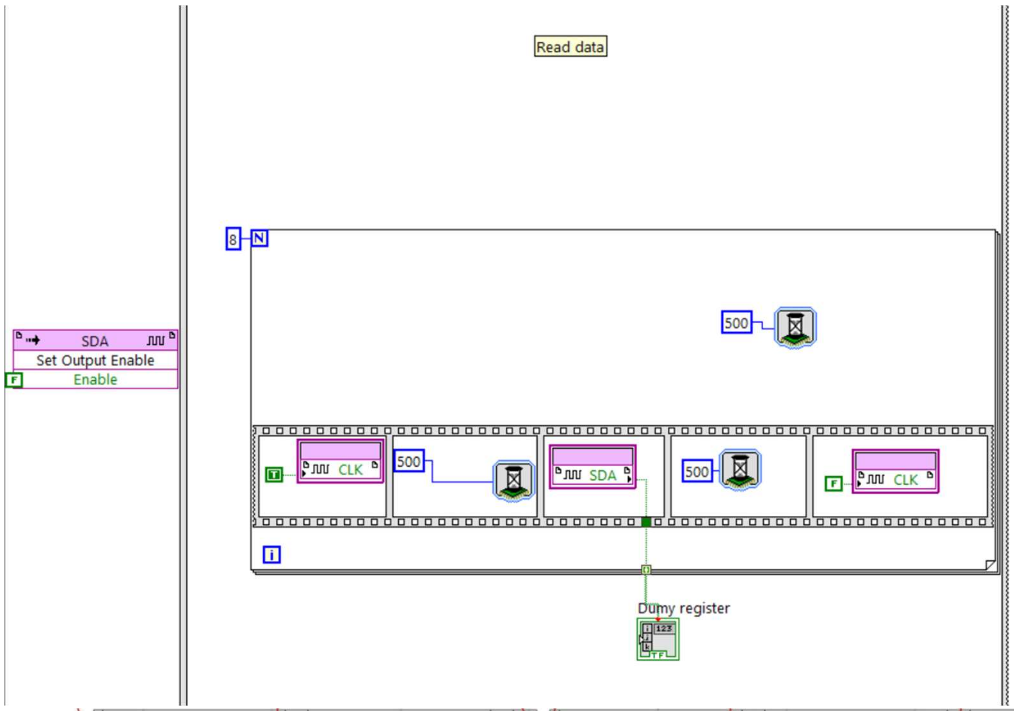
Address device + read

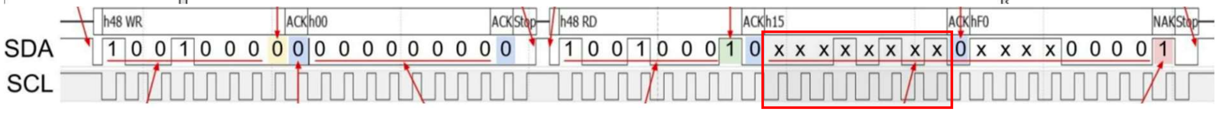




We had again the address device but we change the last bit here who is set to 1 which means it is a READ bit.

Data read





Thanks to that sequence we want to read the dummy register that is 51(10) and before that we need to change the state of SDA to enable again so that we wait the data that we need to read.

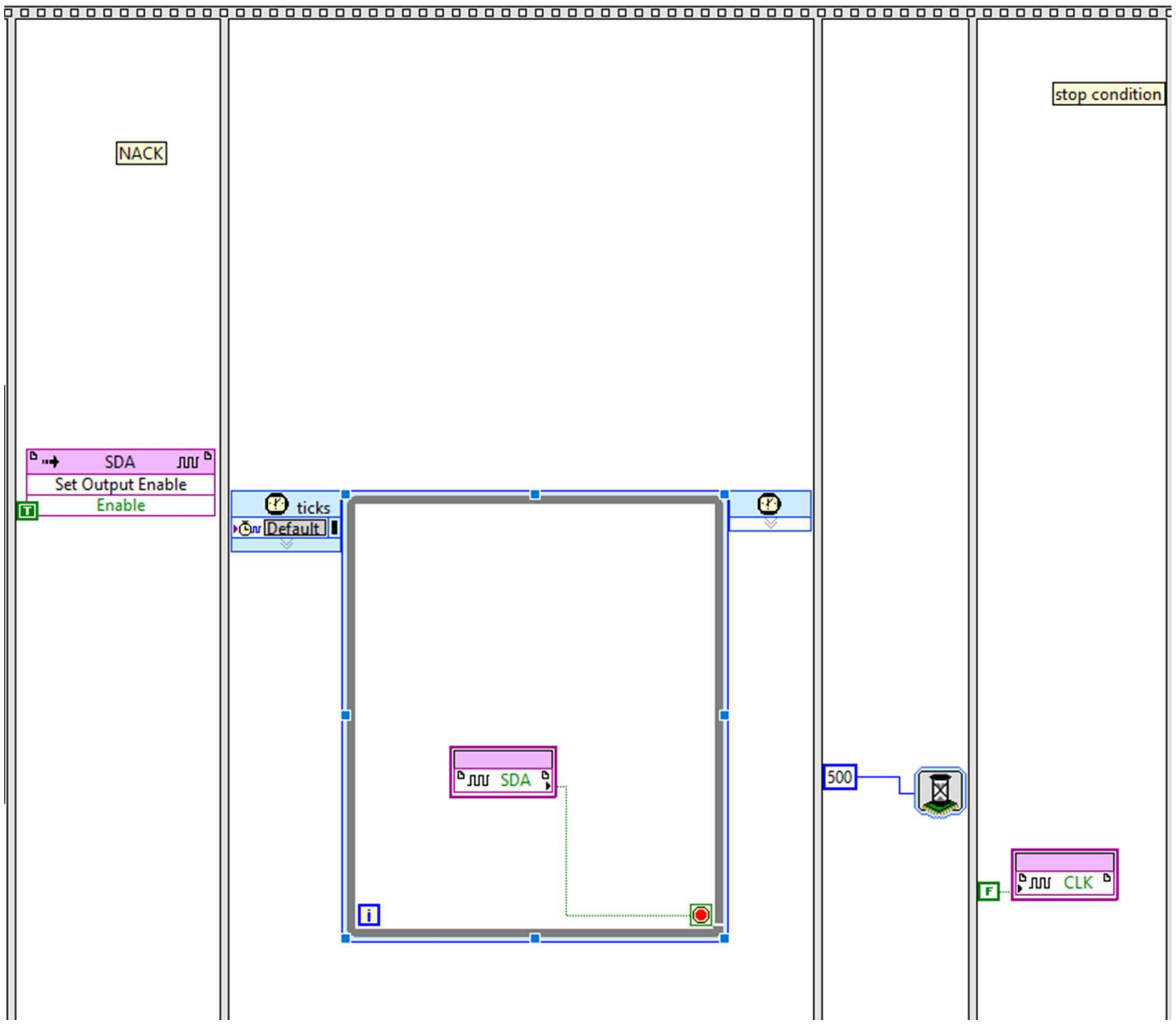
Table 22. Transfer when master is receiving (reading) one byte of data from slave

Master	ST	SAD + W		SUB		SR	SAD + R			NMAK	SP
Slave			SAK		SAK			SAK	DATA		

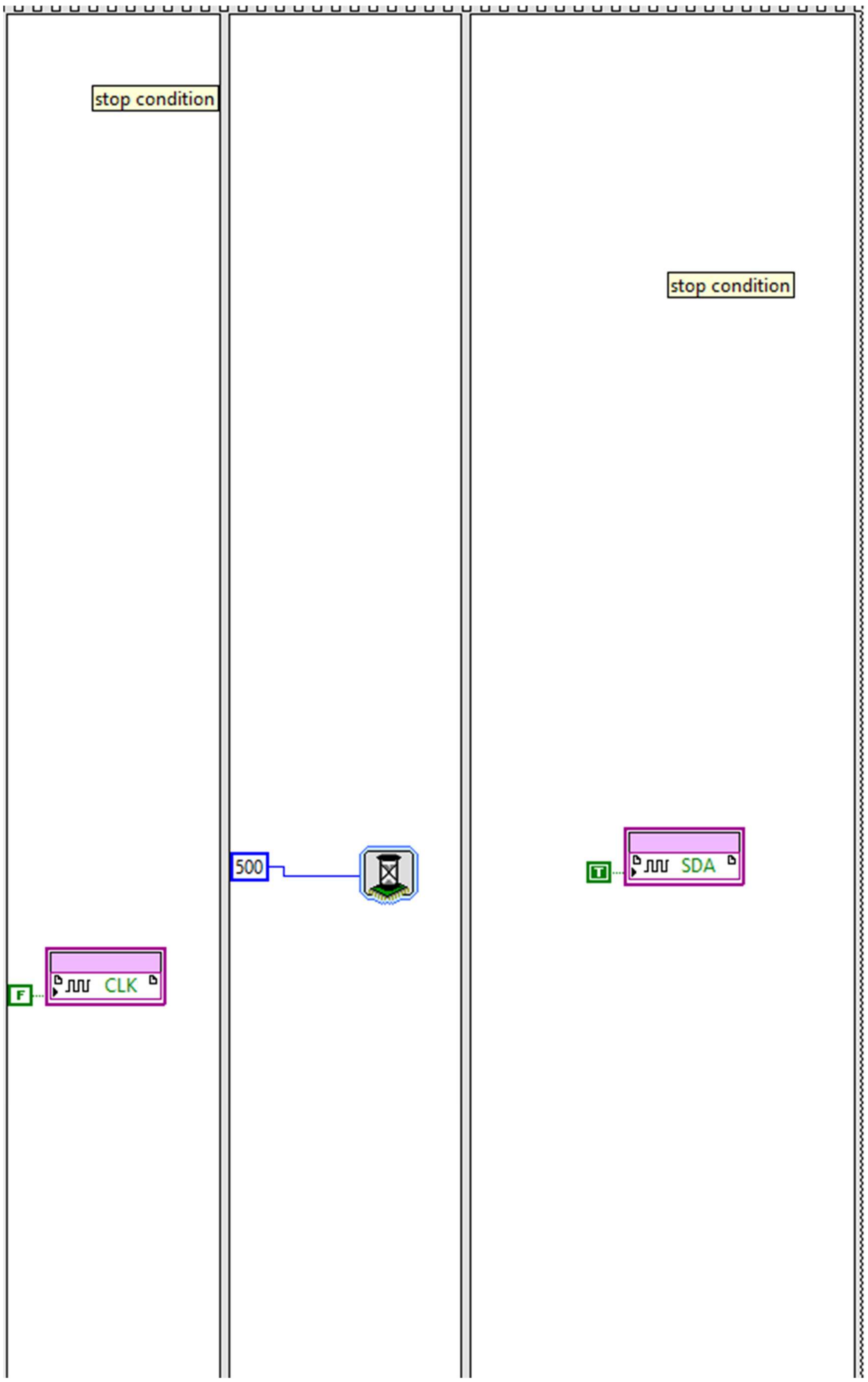
Table 22. Transfer when master is receiving (reading) one byte of data from slave

Master	ST	SAD + W		SUB		SR	SAD + R			NMAK	SP
Slave			SAK		SAK			SAK	DATA		

Now we need to put an NACK. Here it is. We enable the SDA and we wait an answer from the slave

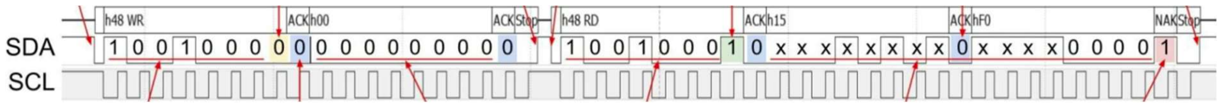


And at the end we need to have a stop bit.


stop condition
CLK
F
500
stop condition
SDA
T

Sofiane Rachdaoui – Tom Bollard

Now It's finished we realised an I2C driver.



END