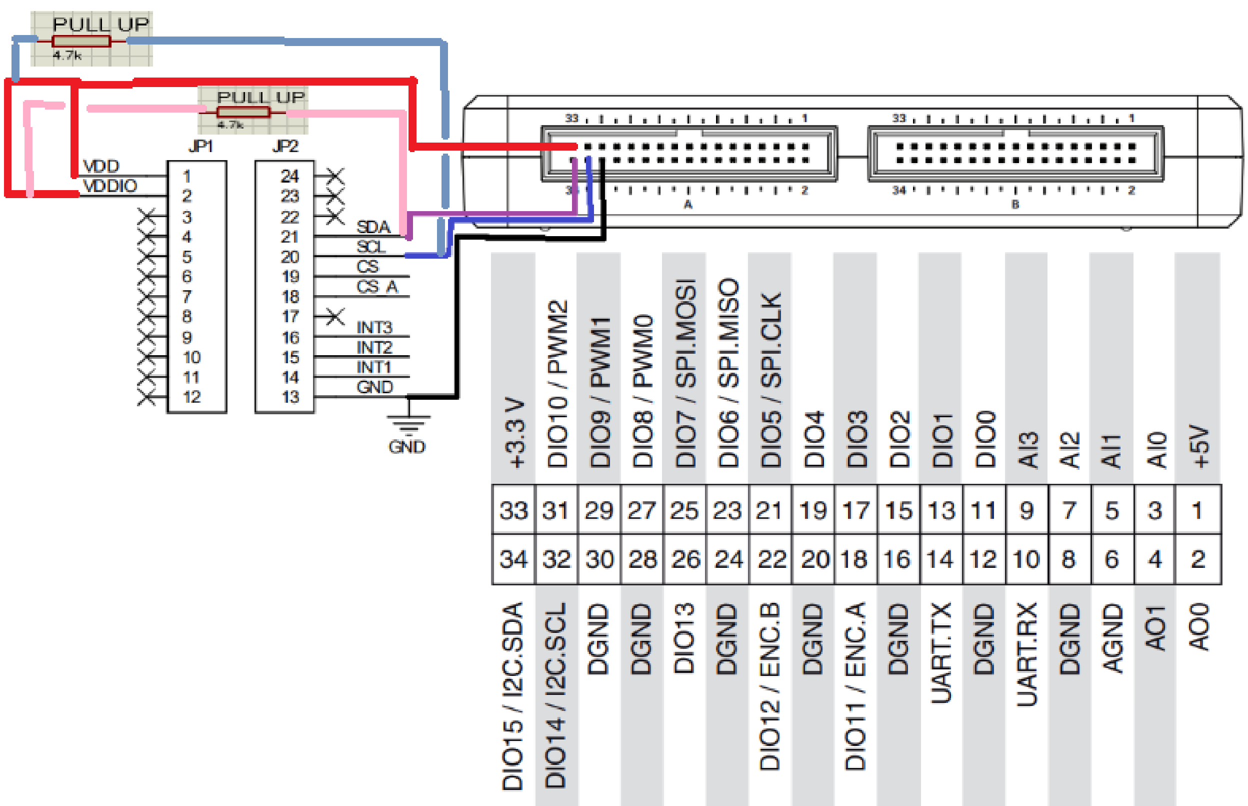
ABBES Ayoub M1 Embedded Systems

LABVIEW FPGA PROJECT REPORT

Wiring :

STEVAL-MKI179V1 circuit	myRIO-1900
PIN 33 (VDD + VDDIO)	Pin 33 (3.3V)
PIN 21 (SDA)	PIN 34 (I2C.SDA)
PIN 20 (SCL)	Pin 32 (I2C.SCL)
PIN 13 (GND)	PIN 30 (GND)

We use pull up resistors to ensure that the signal lines (SDA and SCL) are pulled to a known voltage level when they are not actively being driven by an I2C device.



Theorical Study

Reading accelerometer's dummy register :

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)

8.4 WHO_AM_I_A (0Fh)

Table 30. WHO_AM_I register

0	0	1	1	0	0	1	1

Device identification register.

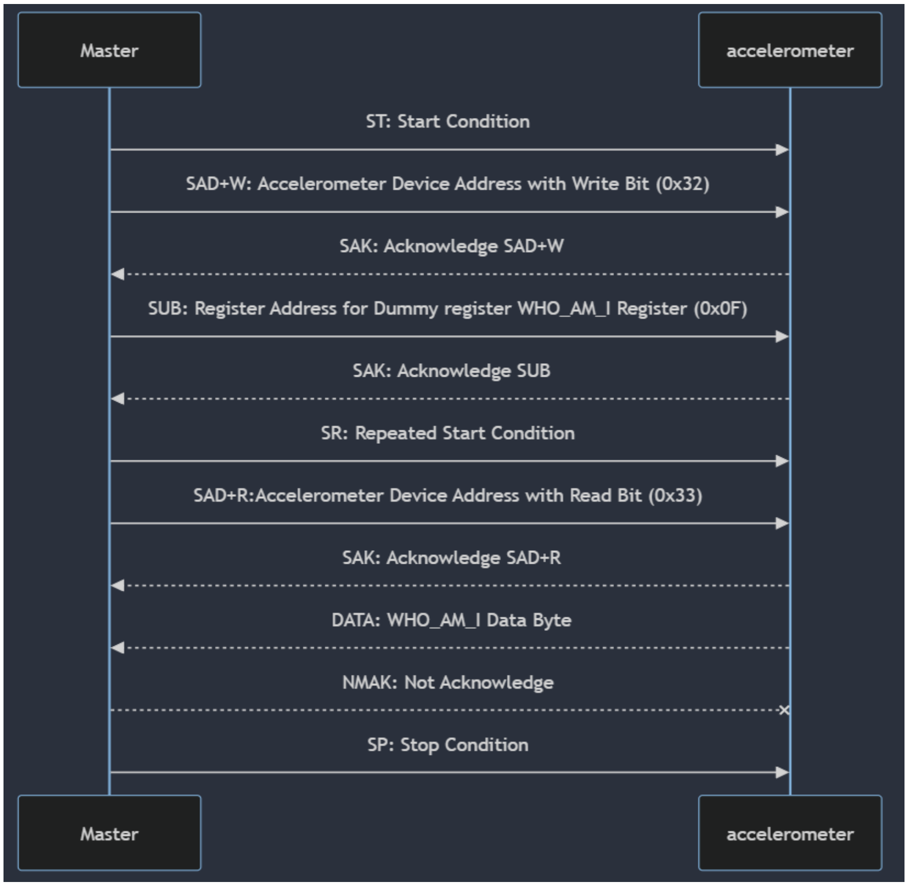
I2C Sequence between MyRIO1900 (MASTER) and Accelerometer (SLAVE) in order to read dummy register :

Since MyRIO1900 is the master and the accelerometer is the slave , we start by initializing the I2C communication involving setting the appropriate clock settings and other parameters.

- ➔ **Send Start conditions**
- ➔ **Send Slave address (SAD) with write bit (W)**
- ➔ **The Slave sends a acknowledgement bit (SAK) which recognizes the reception of data from the master**
- ➔ **Master uses a subaddress which allows multiple registers or functionalities within a single device to be individually addressed.**
- ➔ **The Slave sends a acknowledgement bit (SAK) which recognizes the reception of data from the master**
- ➔ **Master sends a Repeated Start bit (SR) which initializes a repeated start condition allowing the master to start a new data transfer without releasing control of the bus after a previous communication, INSTEAD of sending a stop condition to conclude a transaction, it sends an SR condition to signal that the next operation is part of the same logical sequence.**
- ➔ **"SAD + R" means the master is initiating a read operation with a specific slave device identified by its slave address , followed by SAK and DATA which means the slave acknowledged and read the data.**
- ➔ **NMAK refers to the master choosing NOT to acknowledge the reception of a byte from the slave followed by a Stop condition to indicate the end of the I2C communication.**
- ➔ **Send Stop condition**

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		



Reading (X,Y,Z) Accelerometer values:

First of all , the device address + r/w bit remains the same as the reading accelerometer dummy register part .

To read multiple bytes of data from an accelerometer that measures acceleration in the X,Y and Z directions, we will need to follow a different procedure as shown above

Table 23. Transfer when master is receiving (reading) multiple bytes of data from slave

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

Data are transmitted in byte format (DATA). Each data transfer contains 8 bits. The number of bytes transferred per transfer is unlimited. Data is transferred with the most significant bit (MSb) first. If a slave receiver doesn't acknowledge the slave address (that is, it is not able

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	

Explanation :

Here we're going to use the OUT_X_L_A as reference for the SUB, which is 0x28

But since we want to enable auto-incrementing register, we're going to set the MSB at 1 , which gives use now 0XA8 , so that we can read low high data for each Axis without the need of changing register each time and for sure , we want to follow the sequence .

In the coding part we'll use JOIN NUMBER in order to make a whole data byte of 16 bits for each Axis .

On the next page I2C Sequence between MyRIO1900 (MASTER) and Accelerometer (SLAVE) in order to read accelerometer values :

Reading magnetometer's dummy register :

We set the magnetometer's slave address which has a specific slave address for example 0x3C and it's important to ensure we have the correct slave address.

This sequence allows the myRIO-1900 to communicate with the magnetometer, specifically targeting the dummy register for reading.

Magnetic field sensor: the default (factory setting) 7-bit slave address is 0011110b.

Table 25. SAD + read/write patterns

Command	SAD[6:0]	R/W	SAD + R/W
Read	0011110	1	00111101 (3Dh)
Write	0011110	0	00111100 (3Ch)

8.4 WHO_AM_I_A (0Fh)

Table 30. WHO_AM_I register

0	0	1	1	0	0	1	1

Device identification register.

<u>I2C Sequence between MyRIO1900 (MASTER) and Magnetometer (SLAVE) in order to read dummy register :</u>

As for the dummy register of the magnetometer we use the same sequence as the dummy register of the accelerometer , except we change the device address.

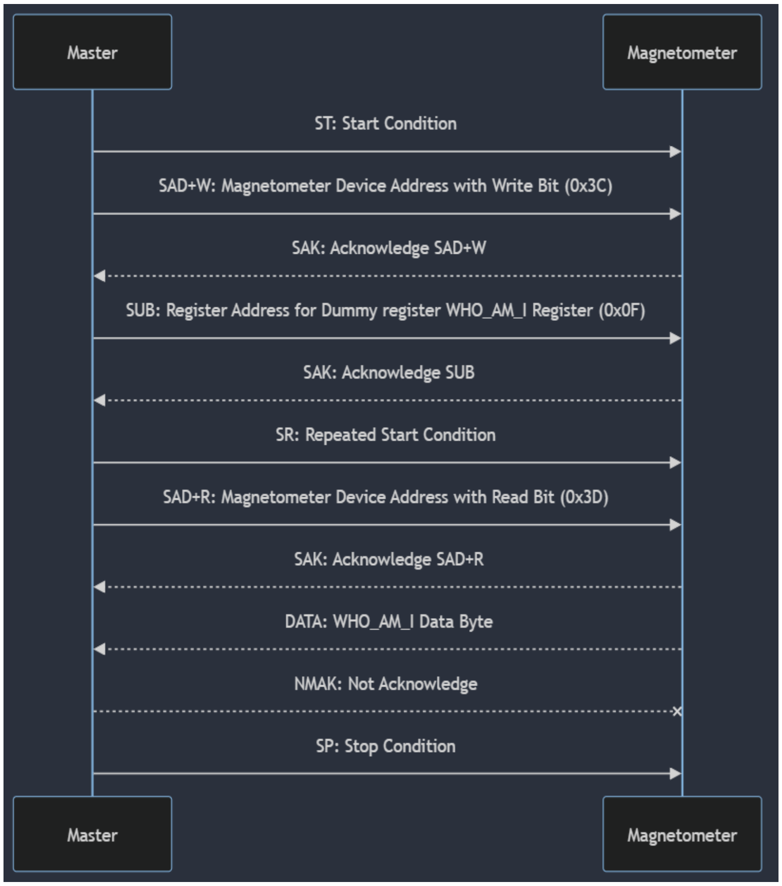


Table 22. Transfer when master is receiving (reading) one byte of data from slave

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

Reading magnetometer value :

Name	Type(1)	Register address		Default	Comment
		Hex	Binary		
STATUS_REG_M	R	67	01100111		
OUTX_L_REG_M	R	68	01101000	output	Magnetometer output registers
OUTX_H_REG_M	R	69	01101001	output	
OUTY_L_REG_M	R	6A	01101010	output	
OUTY_H_REG_M	R	6B	01101010	output	
OUTZ_L_REG_M	R	6C	01101100	output	
OUTZ_H_REG_M	R	6D	01101101	output	

Table 23. Transfer when master is receiving (reading) multiple bytes of data from slave

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

Data are transmitted in byte format (DATA). Each data transfer contains 8 bits. The number of bytes transferred per transfer is unlimited. Data is transferred with the most significant bit (MSb) first. If a slave receiver doesn't acknowledge the slave address (that is, it is not able

to read the magnetometer values , we do the same as reading the accelerometer values , same sequence but we of course change the device address and register.

In this case , we use OUTX_LREG_M as reference which 0x68 but since we need to enable auto-incrementing , we need to put the MSB at 1 which now gives us 0xE8 .

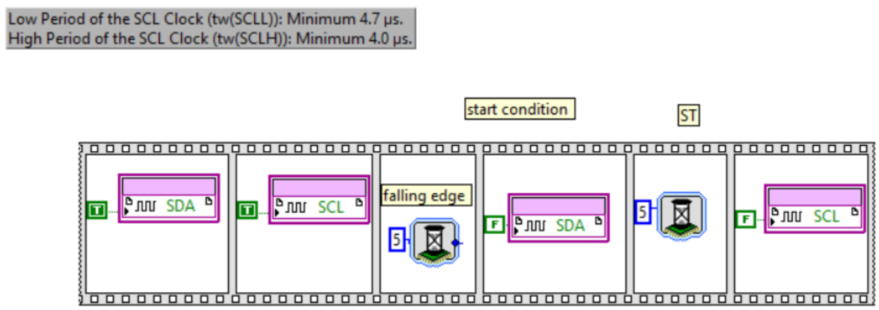
the devices address + r/w bit remains the same as the reading magnetometer dummy register part .


Master
Magnetometer
ST (Start Condition)
SAD+W (0x3C) Magnetometer Device Address with Write Bit
SAK (Acknowledge Write Command)
SUB Register Address for X-Axis Low Data (OUTX_L_REG_M) with MSB = 1 to enable auto-increment
SAK (Acknowledge Register Address)
SR (Repeated Start Condition)
SAD+R (0x3D) Magnetometer Device Address and READ Command
SAK (Acknowledge Read Command)
DATA: Sequential Read Data for X-Axis Low (OUTX_L_REG_M)
Acknowledge after X-Axis Low Data (auto-increment to X-Axis High Data)
DATA: Sequential Read Data for X-Axis High (OUTX_H_REG_M)
Acknowledge after X-Axis High Data (auto-increment to Y-Axis Low Data)
DATA: Sequential Read Data for Y-Axis Low (OUTY_L_REG_M)
MAK: Acknowledge after Y-Axis Low Data (auto-increment to Y-Axis High Data)
DATA: Sequential Read Data for Y-Axis High (OUTY_H_REG_M)
MAK: Acknowledge after Y-Axis High Data (auto-increment to Z-Axis Low Data)
DATA: Sequential Read Data for Z-Axis Low (OUTZ_L_REG_M)
MAK: Acknowledge after Z-Axis High Data (auto-increment to Z-Axis High Data)
DATA: Sequential Read Data for Z-Axis High (OUTZ_H_REG_M)
NMAK: Not Acknowledge after Z-Axis High Data (to signal the last byte)
SP (Stop Condition)
Master
Magnetometer

SUBVIS CREATED/USED FOR THIS PROJECT

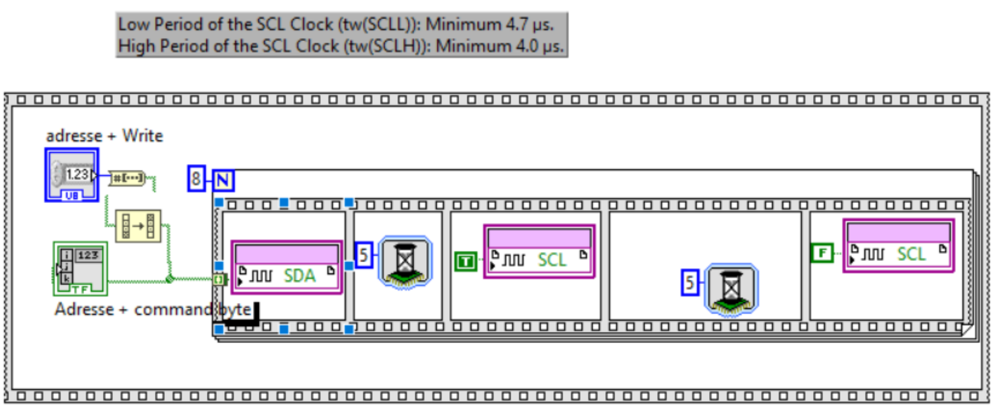
CODING PART :

Start condition :

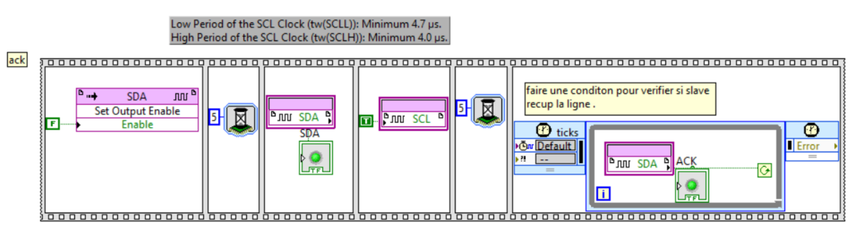


Adress + Write :

We need to use reversed array function in order to send the MSB first .

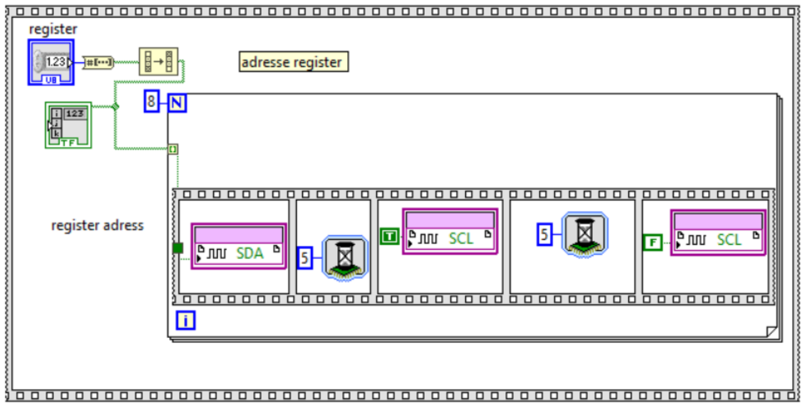


Acknowledge (ACK) :



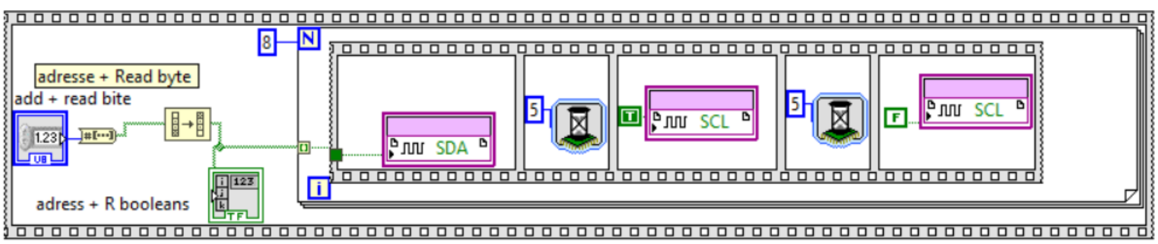
Adress Register :

We need to use reversed array function in order to send the MSB first .



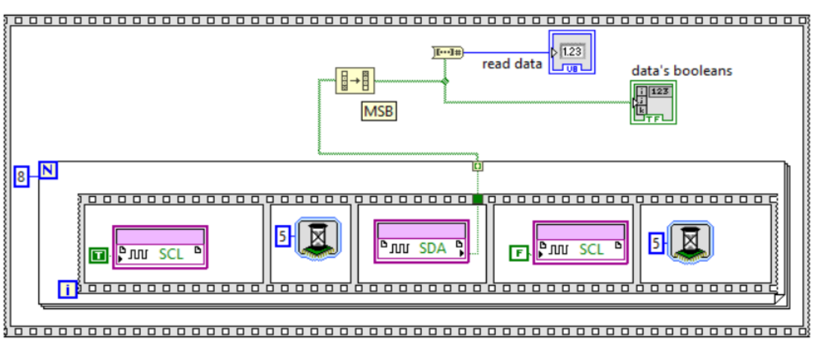
Adress + read bit :

We need to use reversed array function in order to send the MSB first .

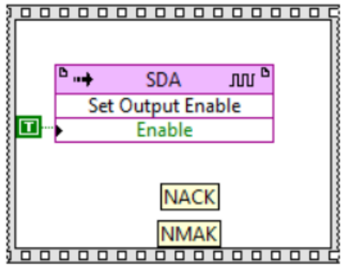


Reading Data :

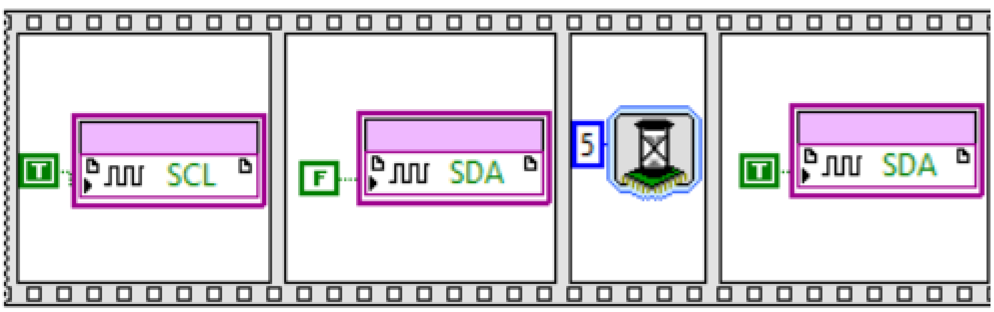
In I2C communication, data is transmitted and received in a specific bit order, usually with the most significant bit (MSB) first . We need to use the reversed array function .



Not Acknowledge (NACK) :



STOP condition :

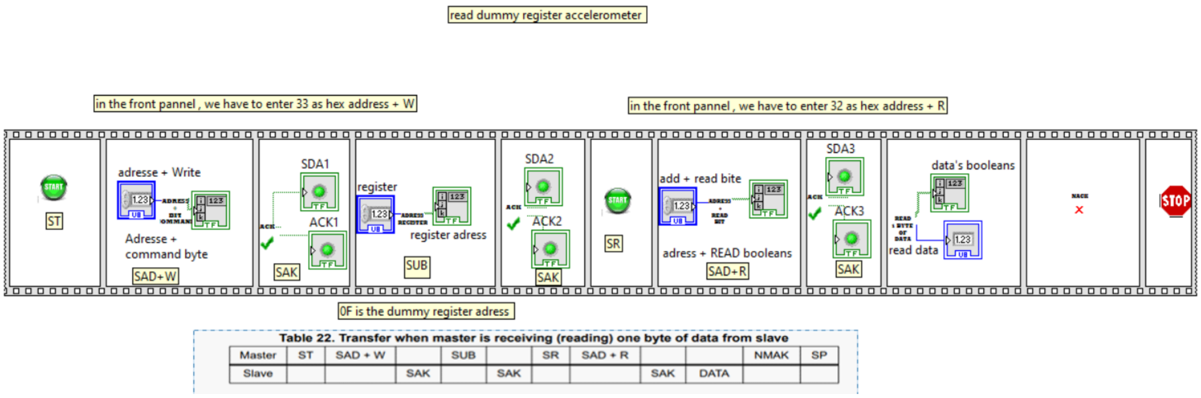


All these Subvis are used for this I2C frame all while respecting the lsm303agr datasheet.

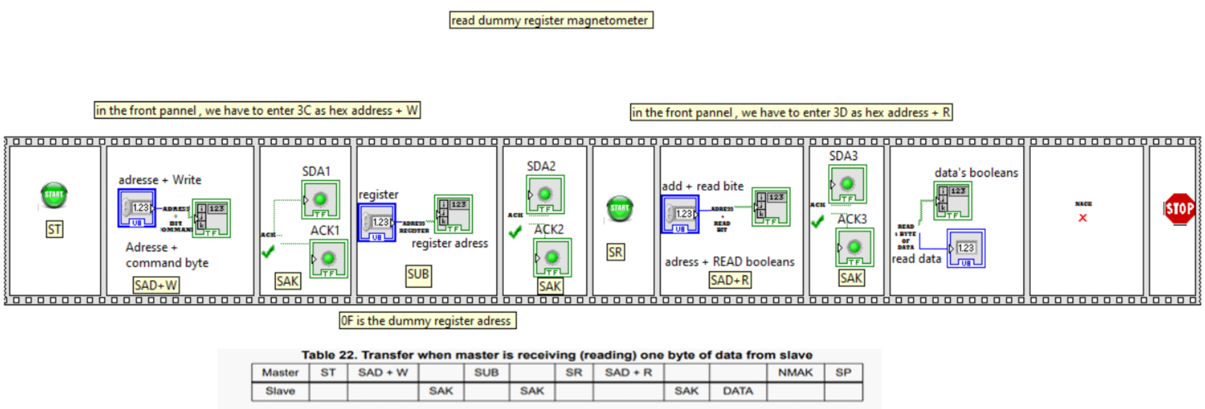
This allows us to obtain a certain data in the register from the accelerometer as well as the magnetometer.

As for the dummy register , we would be getting the same values every time because it's a dummy register, so we need to change the address to make sure we are communication with the RIGHT device , either it's accelerometer or magnetometer

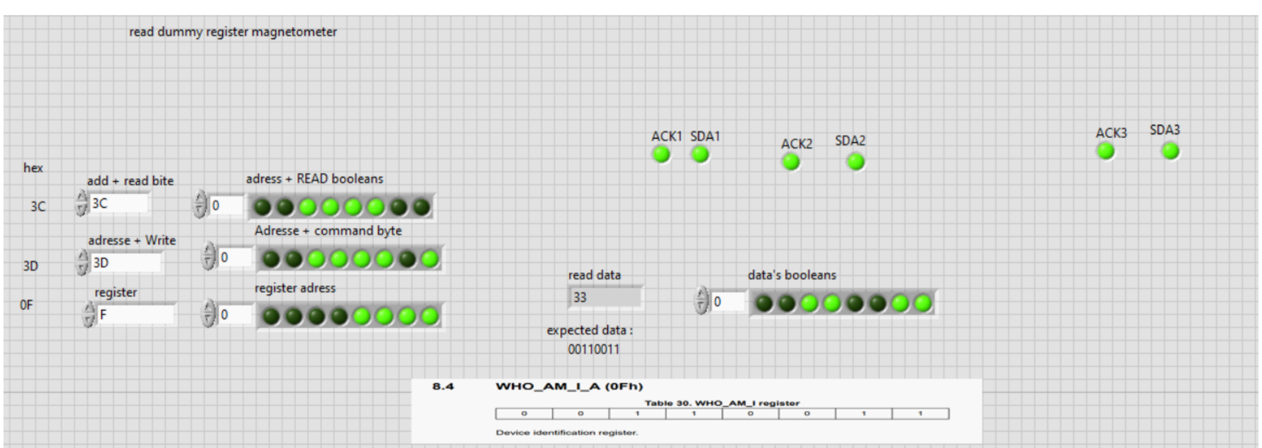
This part is for reading dummy register accelerometer :



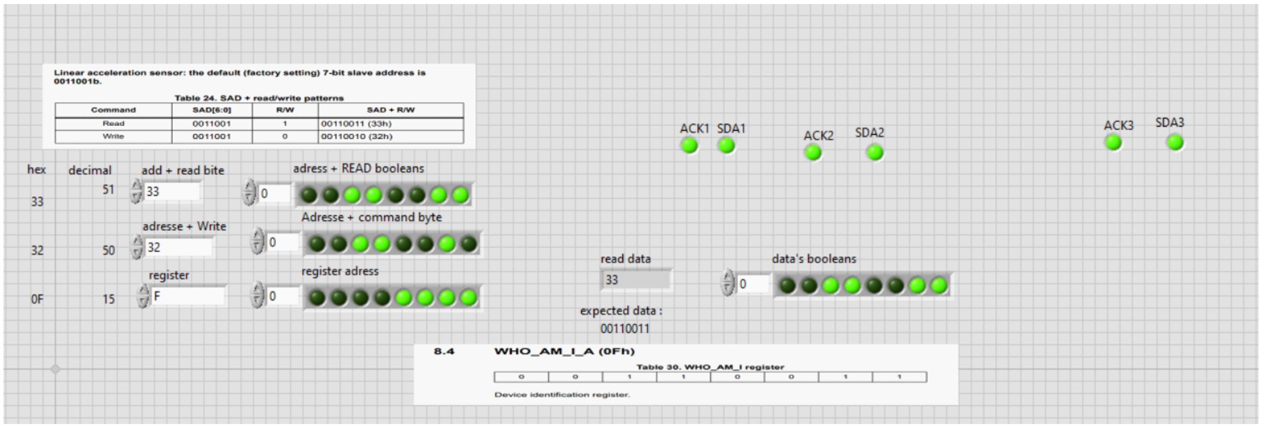
This part is for reading dummy register magnetometer:



Front panel and programming of the “read dummy register magnetometer” :



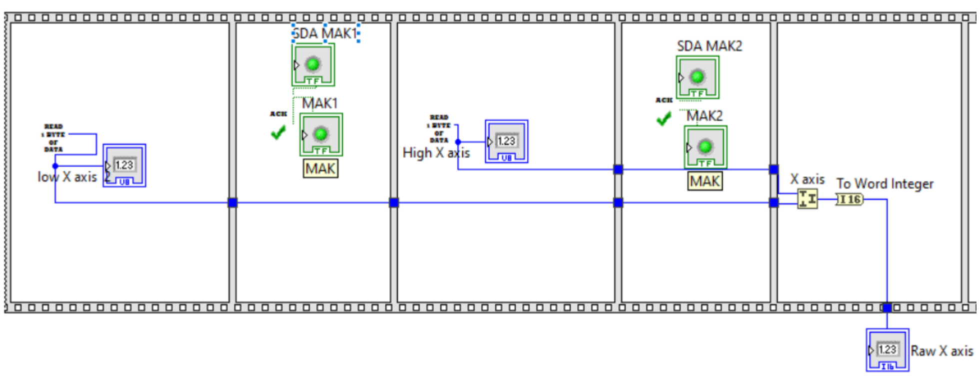
Front panel and programming of the “read dummy register accelerometer” :



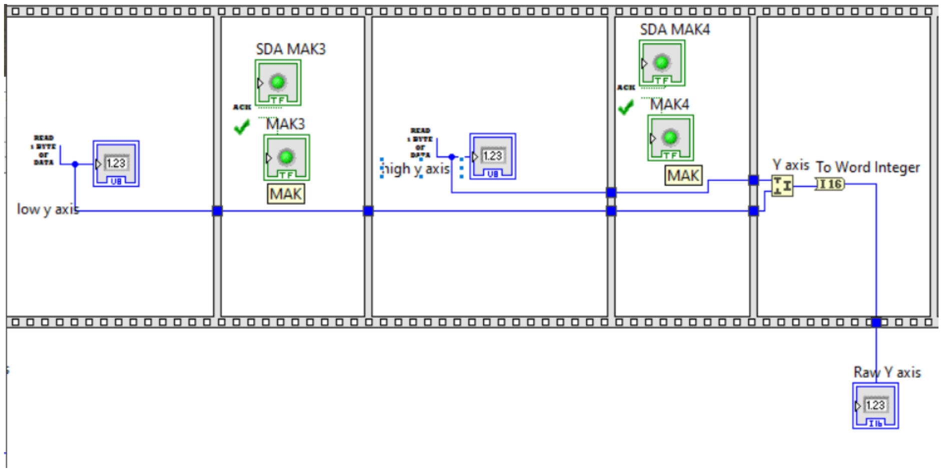
SUBVI for reading X ,Y,Z AXIS :

As mentioned before , we are using the join number function in order to make a whole byte from the low and high byte of each Axis .

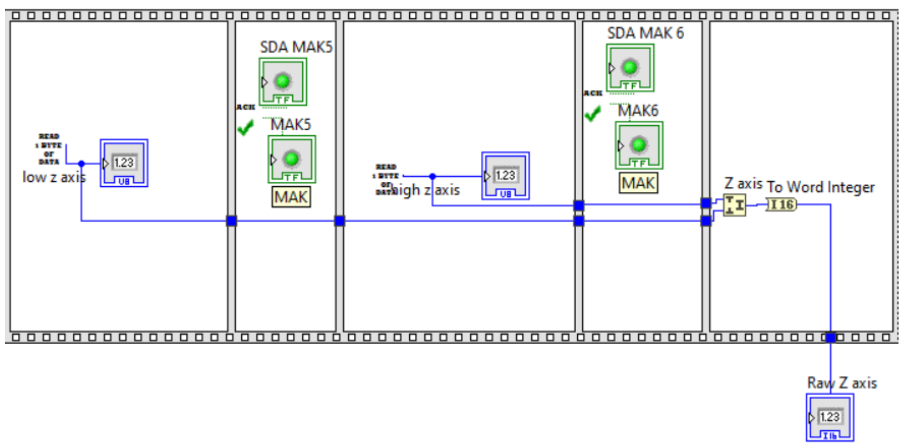
reading X AXIS



reading Y AXIS :



reading Z AXIS :

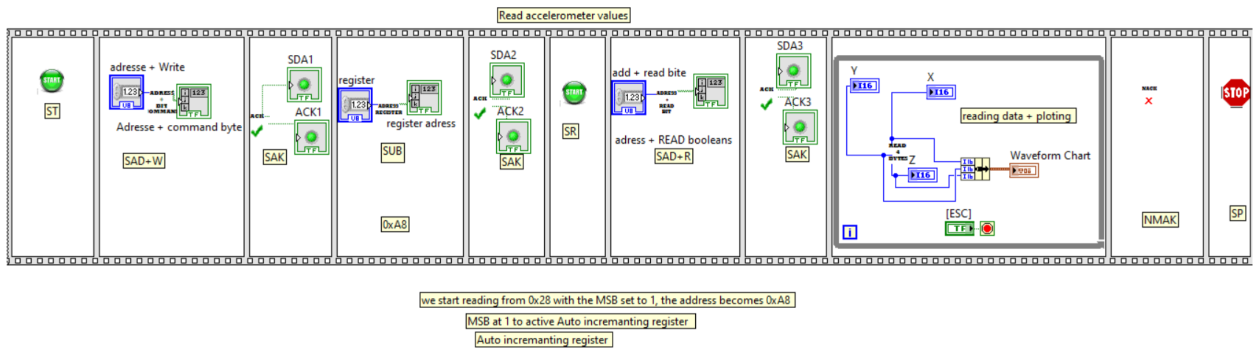


This SUBVI read 4 bytes, which uses the subvi that can read 1 byte (read data 1 subvi).

These are for reading X,Y and Z axis of the accelerometer and magnetometer.

The accelerometer coding part :

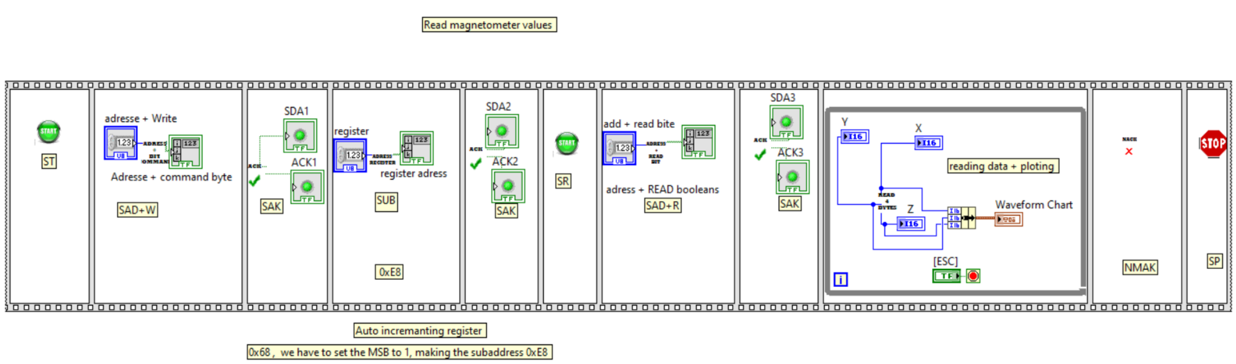
This part is specifically made for the accelerometer and to read the values.



Front panel view of the accelerometer :

The magnetometer coding part :

This part is specifically made for the magnetometer and to read the values.



<u>Front panel view of the magnetometer :</u>