Allegro Microsystems Europe
74650 CHAVANOD
http://www.allegromicro.com

GOUILA Bastien
IUT Annecy
GEII Department
Year 2016

Internship Report

Acquisition system for magnetic sensors

Internship Responsible : MESSIER Loïc

Teacher : BERTRAND Stéphane

Table of contents

List of figures

Short company presentation : Allegro Microsystems Europe LTD

Allegro Microsystems is an American semiconductors company specialized in high-performance semiconductors such as motor drivers, regulators, magnetic sensors and current sensors. It is world leading in the magnetic sensors area. Its main market is the automotive industry, but is also highly present in the consumer electronics market and the industrial market. The french technical center based in Annecy where I made my internship is primary focusing on magnetic sensors and is responsible for supporting european customers. An interesting fact : in a classic car, there is an average of 60 Allegro sensors.
The Annecy technical center is involved in the development of new products and is working closely with customers to provide adequate solutions. Its activities include tests, characterizations and evaluations of magnetics sensors and also supports of failure analysis.

I. Introduction

I.1 Project Overview

The purpose of this project is to create a tests system for magnetic sensors in order to make all tests needed faster and easier. Tests could either be manual or automatic. This system is made of four parts: the sensor, a target, an FPGA with its external acquisition board and a computer program in C# language.

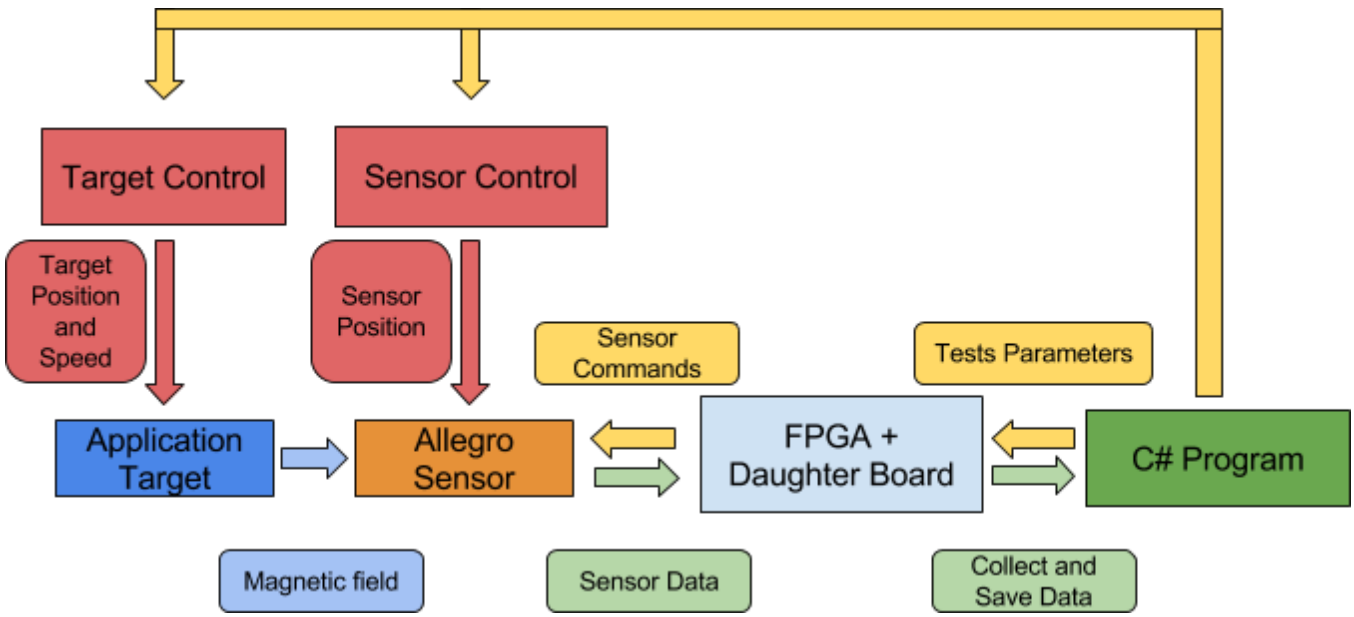


Figure 1 : Project functional diagram

The sensor responds to magnetic stimulus and transmits data to the FPGA with electronic signals. To capture this signals, Allegro has designed a daughter board that can be fixed on the FPGA and controlled by it. When the acquisition board (FPGA + daughterboard) has finished capturing the sensor data, it is saved in the FPGA internal memory, the C# program running on the computer collects the data and saves it into an Excel file. The C# program is also capable of controlling the positions of the target and of the sensor.

I.2 Specifications

Objectives :

The objective of this internship is to complete the C# program and make it interact with the rest of the system. At the begin of the project, the pre-existing C# project, is capable of doing simple manual tests with the FPGA and to control the motor and axes for the target and sensor positioning.

At the end of the project the C# program will be able to do different types of manual and automatic tests with the Acquisition Board, and save the tests results into an Excel file. This part of the program will be an interface of communication with the FPGA and completely independent from the rest of the C# program.

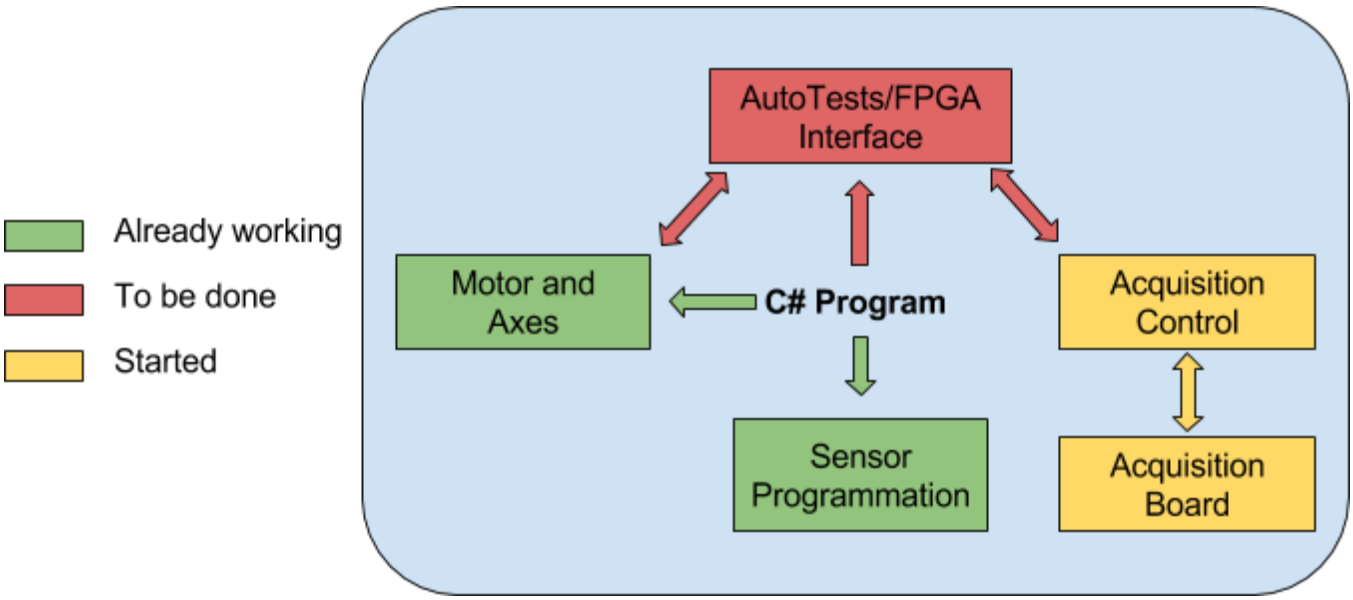


Figure 2 : C# program functional diagram

Project restrictions :

- The software must be written in C# language.
- All acquisition board functionalities must be debugged and verified.
- The project deadline is June, 17th 2016.

I.3 Sensor Description

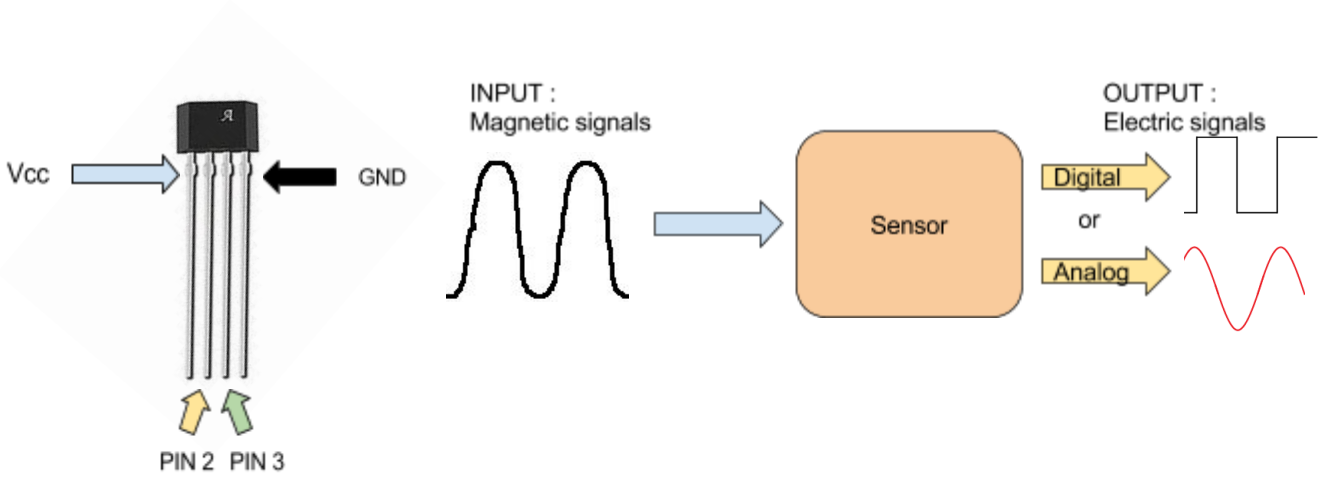


Figure 3 : Sensor description

All sensors are based on the same principle : they are converters of magnetic signals into electronic signals. Allegro sensors can be used on magnetic target but also on ferromagnetic target, see annexes. The sensor can have 2, 3 or 4 pins, the only two constant pins are Vcc and GND.

II. Acquisition Board

The Acquisition Board is composed of an FPGA and a daughter board designed by Allegro as mentioned earlier. This daughter board is connected to the FPGA (model Altera DE2-115 from Texas Instrument, see annexes) and allows it to acquire sensor data. It is the interface between the FPGA and the outside variables (sensor, encoder...).

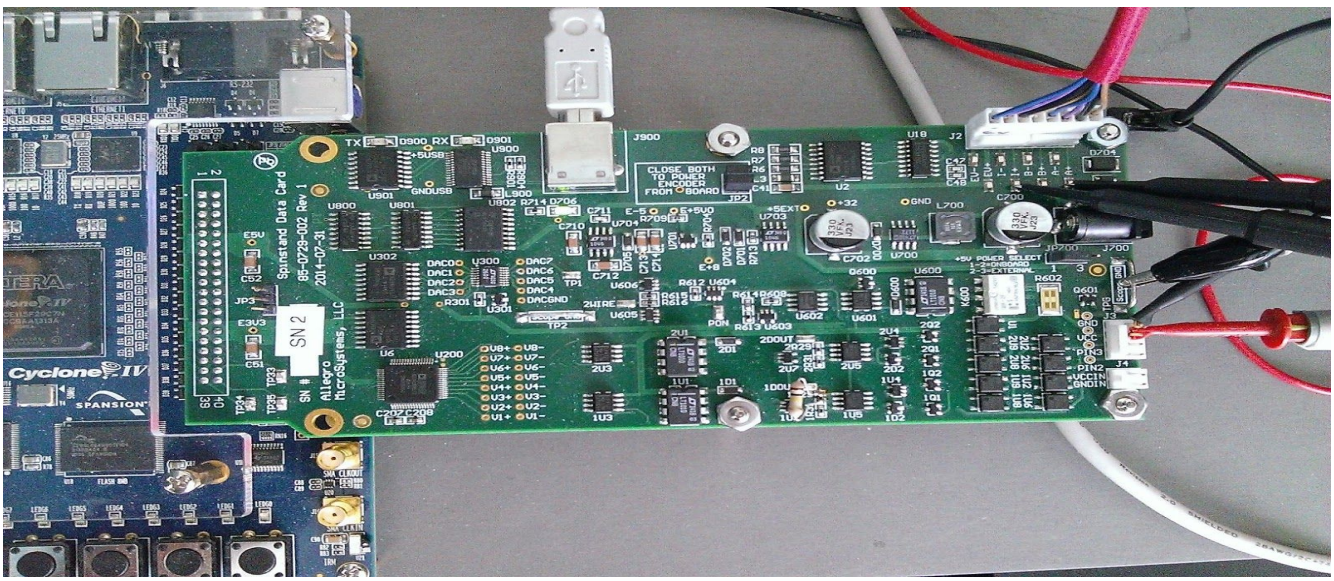


Figure 4 : Picture of the FPGA with the Daughter Board

II.1 Hardware

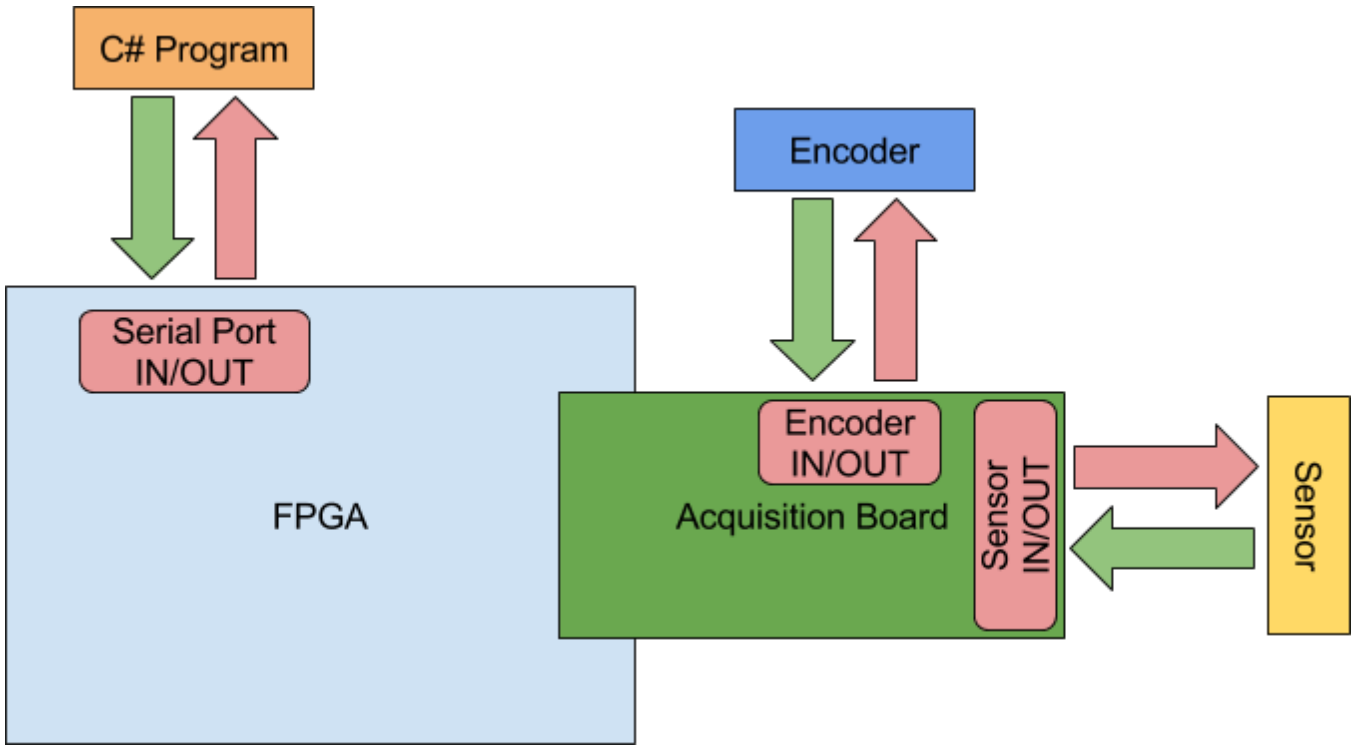


Figure 5 : FPGA and Daughter Board functional diagram

For the hardware part, the acquisition board is mainly composed of filters, comparators, regulators and of analog/numeric and numeric/analog converters. Its purpose is to acquired and emit signals.

There are three IN/OUT ports that are used during this project :

- The Serial port, which communicates with the C# program in order to set the FPGA and Board parameters and to collect the data when the tests are completed.
- The sensor port, which is used to supply and communicate with the sensor.
- The encoder port that is used to supply and communicate with an external optical encoder. This encoder is used as reference to know the absolute position and speed of the target during the test.

II.2 Firmware

The FPGA firmware has been developed by Allegro to interact with the daughter board and offers multiple tests capabilities:

- Changing the sensor supply voltage, applying different threshold values on Vcc (supply voltage) and Icc (supply current).

- Recording target speed and position using the external encoder.

- Selecting different input capacitors and resistors, threshold values and measure types (analog or digital) on Pin 2 and 3.

- Reading and Writing in the FPGA memory in order to modify and read tests parameters and to save the data.

- Possibility to do tests on four different channels in parallel.

- Choosing between multiple tests parameters such as acquisition clock, number of samples, start time and signal type to acquire (period, tension, current, angle…).

III. Acquisition Control

The Acquisition Control is the part of the C# program which directly communicate with the FPGA. In order to use this communication interface in other programs without starting over every time, this interface has been placed in a C# Control. Consequently, this interface is independent from the rest of the code and allows more flexible updates.

III.1 Pre-existing functions

At the beginning of the internship, the Control was able to communicate with the FPGA and to do simple and manual tests. But many function had not been tested yet.

Most low-level functions that directly write or read in the FPGA memory were already done. Some of these functions were available but not used in the Acquisition Control.

The functions already available were :

- Configuration of the Vcc
- Configuration of Pin2
- Configuration of Pin3
- Configuration of the four acquisition channels
- Start and Save manual tests

Below, a picture of the original interface.

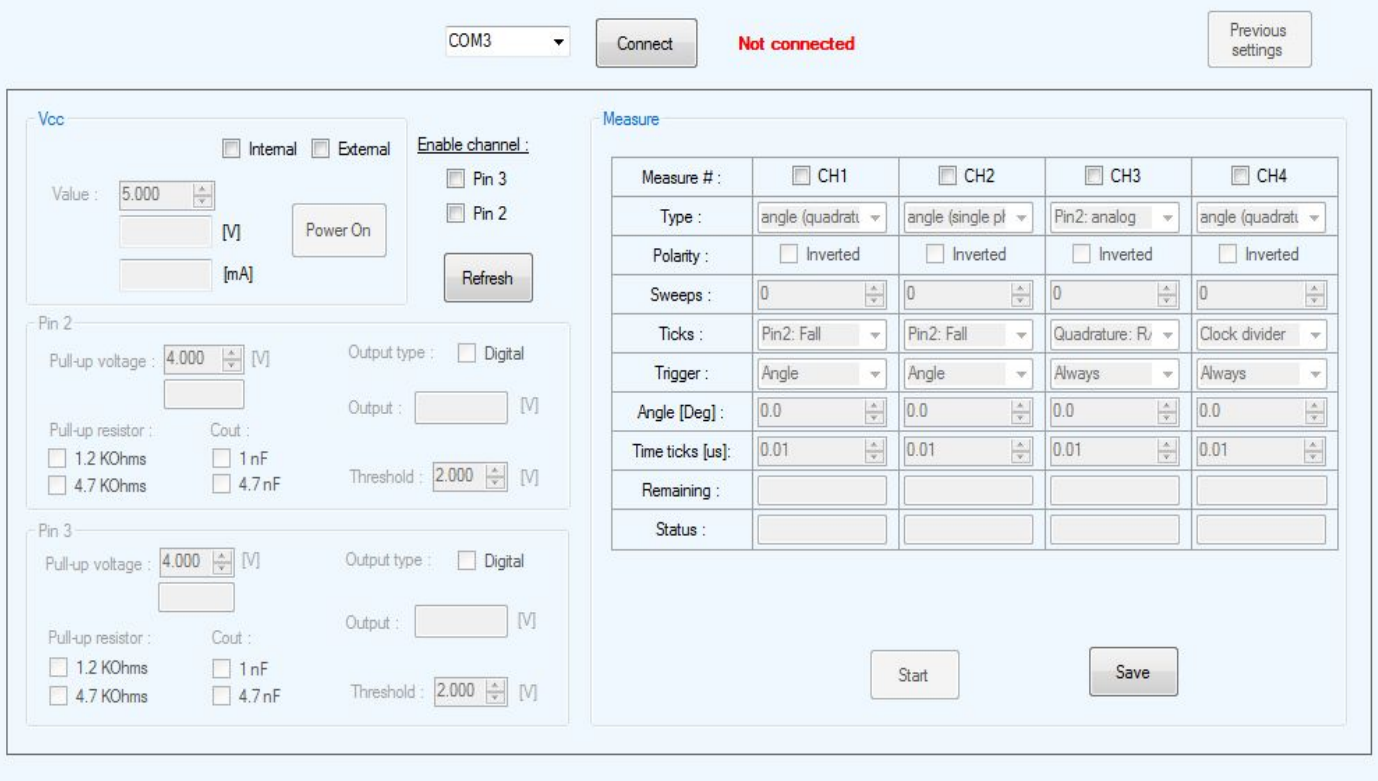


Figure 6 : Acquisition Control at the beginning of the project

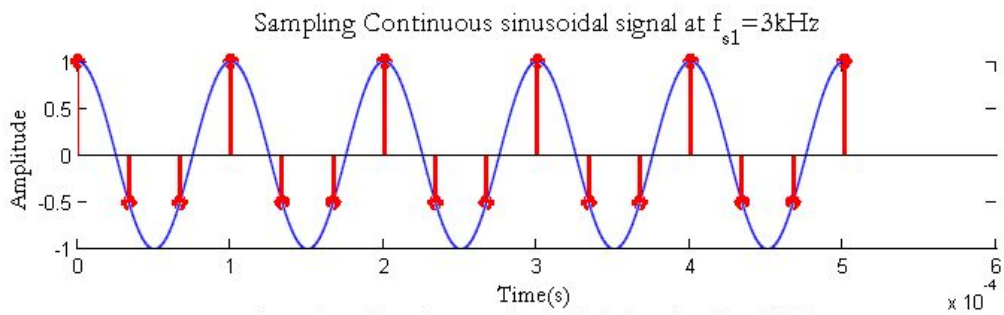
III.2 Improvements

The Control was able to establish a connection with the FPGA thanks to the Serial Port. But the disconnection wasn't possible and the program needed to be closed to disconnect the board. A disconnection is now possible when the user click again on the button used to connect. The disconnection function is also called when the form is closed in order to release the serial port.

The possibility to read and set the Icc threshold has been added to configure the current comparator. Some Allegro sensors use two-wire principle and communicate through current modulation.

The sample frequency has been limited and is automatically adjusted if we are using the FPGA internal clock. So that the maximum sampling rate is fixed by the ADCs conversion time. Indeed, the ADCs (Analog/Digital Converter) used have a maximum conversion frequency of 200kHz while the FPGA has a maximum internal clock at 150 MHz.
This conversion problem was observed when the sample frequency wasn't limited. The internal clock is too fast and the ADCs cannot convert fast enough resulting in identical values(figure 8) and missing samples. To avoid this problem, the maximum sample frequency for analog measures has been limited to 100 kHz to keep some margin and fulfil Nyquist-Shannon law.

See below an example of such case :



Data CH1 #:	Icc [mA]
0	8.97206
1	8.97206
2	8.97206
3	8.97206
4	8.97206
5	8.97206
6	8.97206
7	8.97206
8	8.97206
9	8.97206
10	8.97206
11	8.97206
12	8.97206
13	8.97206
14	8.97684
15	8.97684
16	8.97684
17	8.97684
18	8.97684
19	8.97684
20	8.97684
21	8.97684
22	8.97684
23	8.97684
24	8.97684
25	8.97684
26	8.97684

Figure 7 *Figure 8*

Figure 7 : *Example of too fast sampling rate* : red = samples, blue = original signal

Figure 8 : *First data from the FPGA* (sample frequency = 100ns)

The possibility to get the target speed seen by the FPGA thanks to the encoder and change the encoder reference (number of pulse per revolution or ppr).

Some corrections of channel measurement status, data collection and data conversion were made on every channel. Besides, the acquisition with multiple channel has been corrected (problems of synchronization and data exploitation).

Finally, all functions for automatic tests and typical tests has been added. As we will describe in the next pages.

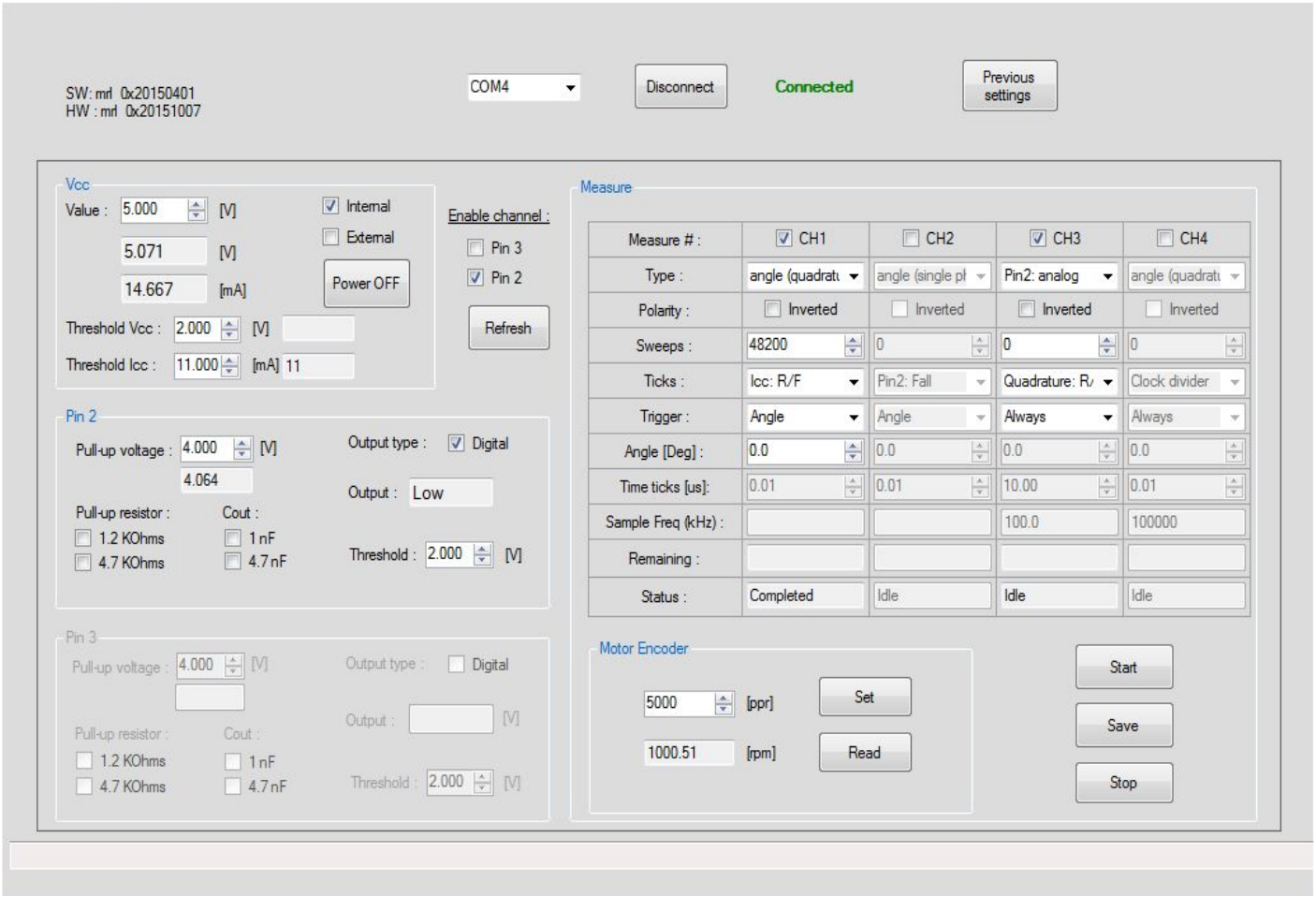


Figure 9 : Acquisition Control at the end of the project

III.3 Acquisition functions

The next acquisition functions have been designed in order to do some typical tests that Allegro needs to do on its sensors.

Main tests : Startup Accuracy test and the Jitter test

Startup Accuracy Test : "Startup Accuracy Test" is defined as the "time" it it takes for the sensor to initialize and give the first output pulses. This maximum time corresponds to the amount of magnetic periods the device needs to calibrate. It is specific to each device and the purpose of this test is to make sure the device is fully functional before this maximum delay time.

- **Air Gap effect :** Another purpose of the Startup Accuracy test is to determinate the maximum air gap (distance between the sensor to the target) capability of the device on the application target. Basically, we measure the position of the first edges on several PowerOn cycles at different distances and record from what distance the maximum authorize initialization time is exceeded.

- **Speed effect :** The Startup Accuracy Test can also be performed over speed to evaluate the impact of different speed on the sensor accuracy and response time.

Figure 10 shows a Startup Accuracy Test representation during two PowerOn cycles at the same air gap. The cursors delimit the time between the PowerOn and the first pulse specified on the device datasheet. The first cycle (green = sensor pulses) respects the datasheet but the second one (yellow = sensor pulses) doesn't. This shows that the current distance (air gap) doesn't fulfill the sensor specifications.

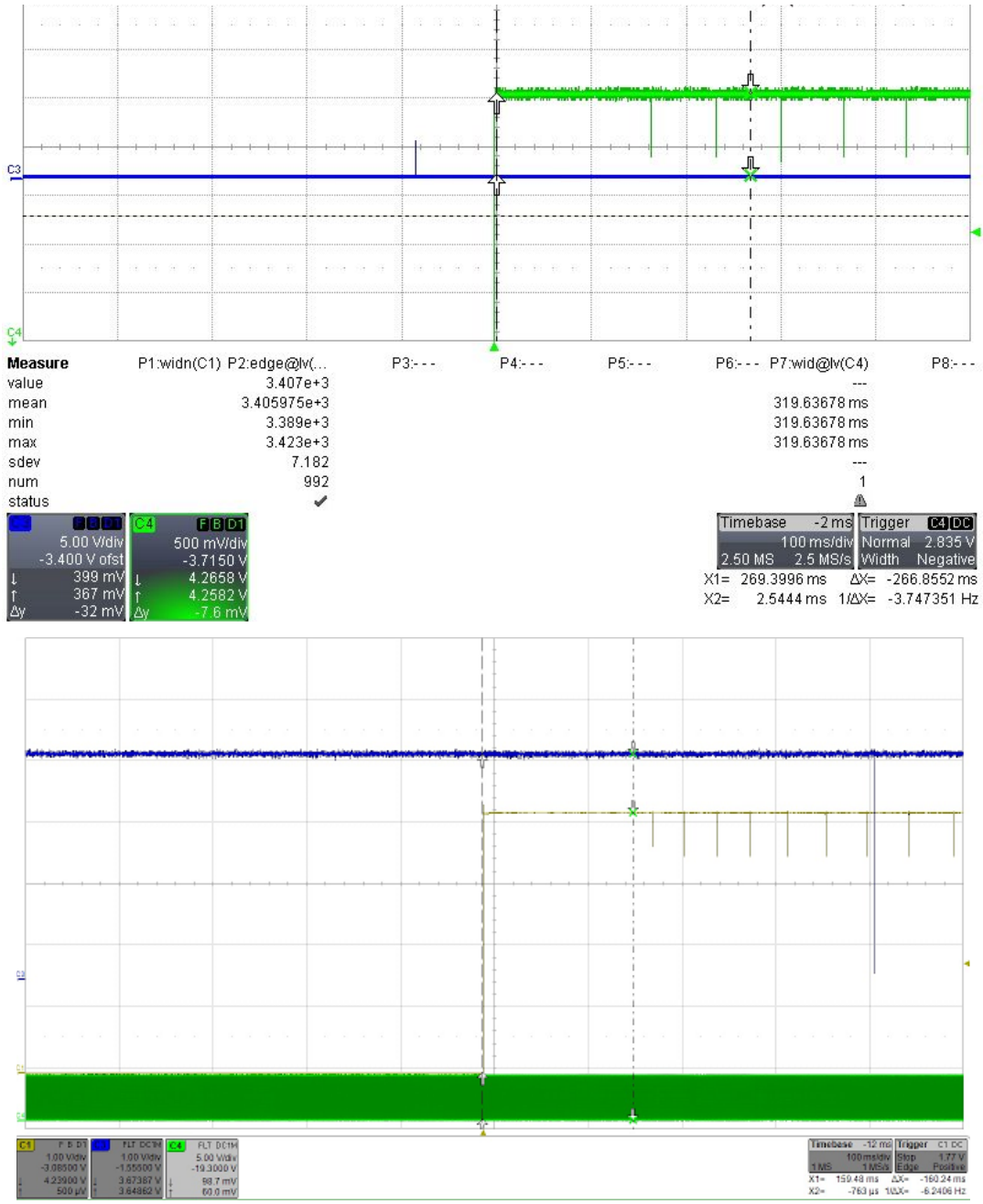


Figure 10 : Startup Accuracy representation on the scope

Jitter Test :The "Jitter" is a repeatability measurement defined as the standard deviation of the pulses sent by the sensor. It represents the statistical distribution of multiple switching location Basically, we record the position (in degree) of each edge over several target revolutions and calculate their distribution.

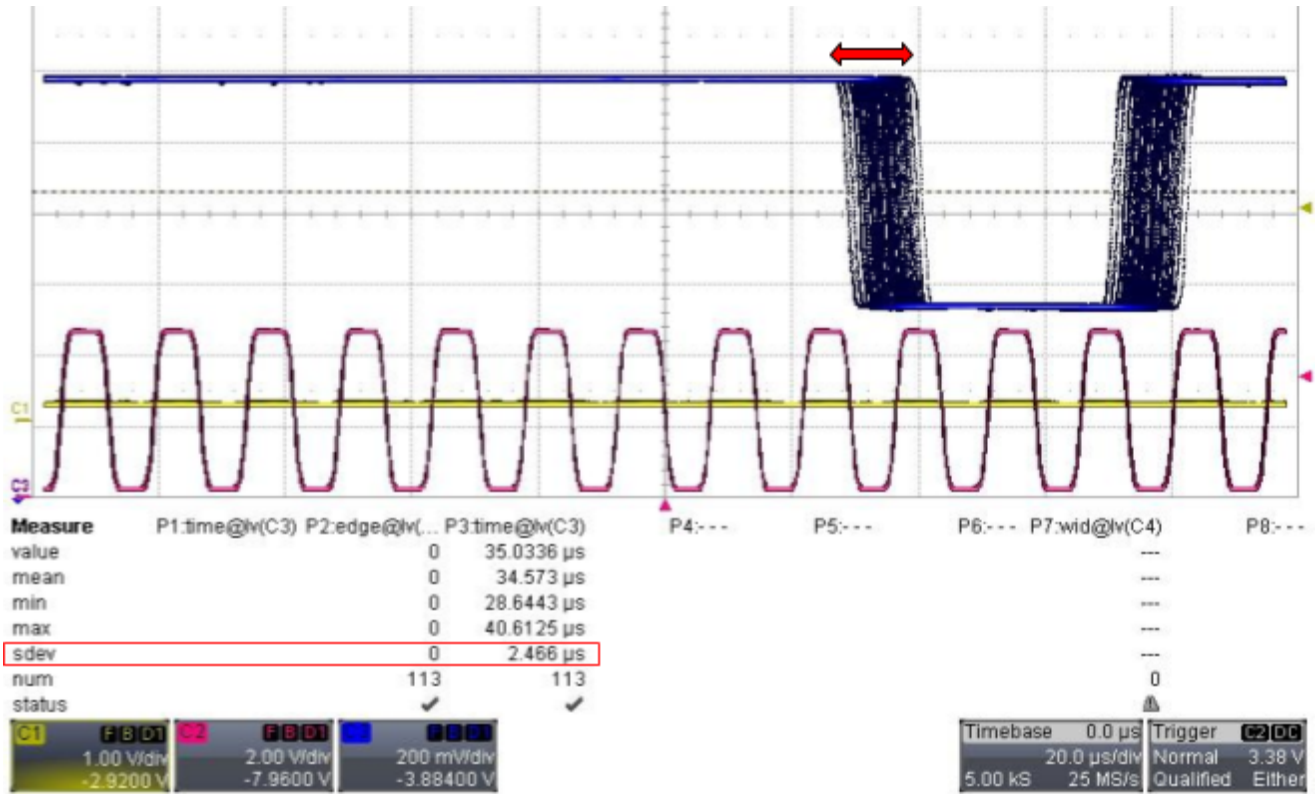


Figure 11 : Jitter representation on the scope (= blue)

Figure 11 shows a Jitter representation of two edges on several revolutions. Sensor output = blue; Reference encoder = red; One revolution index = yellow.

In this case, on figure 11 we see that the Jitter value for the first edge is 2.466us.

Description of the main acquisition functions :

public *void* **Multiple_Acqui_Measur_Start**(*uint* uiCurrentCycleNbr)

uiCurrentCycleNbr = the index number of the current measurement cycle.

This function starts one measurement cycle after setting the FPGA according to what has been selected in the measurement parameters (clock, number of samples, trigger...). "uiCurrentCycleNbr" is used to shift the start address where the data will be stored.

This function allows up to four channel to measure at the same time.

For example, if we want to do a measure composed of 3 cycles with 120 samples on the first channel, *data start address* is initialized at address 0 in the channel memory for the first cycle and then shift to address 120 for the second cycle and finally shift to address 240 for the third cycle.

public *double [,]* **DataCollection**(*uint* uiTotalNumberOfCycle*, uint* uiChannel)

uiTotalNumberOfCycle = the number of measurement cycles to be collected.

uiChannel = the channel memory that needs to be read.

This function reads the FPGA memory containing the tests results and returns the values in a two dimensional array of double [Samples,Cycle].

Internally, the function reads the FPGA memory 200 items at a time(a buffer of 200 items avoid serial port overflow and a loss of data) coded in hexadecimal format. Then, the function converts the items in decimal format before converting it into the appropriate format (a scale factor is applied depending on the measurement type : current, tension, angle...).

The function also deals with the negative values that are written in two's complement, consequently, another conversion is previously done if the function recognizes a negative value.

public *void* **StartupAccuracy**(*uint* iNbPowerON*, bool* bPowerONType*, int* iAngleStartup*, double* dAngleStepSize*, uint* uiChannel)

iNbPowerON = total number of PowerOn cycles wanted.

bPowerONType = *true* means a phased PowerOn (at particular angles) and *false* means a random PowerOn.

iAngleStartup = the angle where the PowerOn occurs with a phased PowerOn.

dAngleStepSize = the angle increment at each cycle. (Not coded yet)

uiChannel = the channel used.

The function does multiple cycles of PowerOn and captures the first edges according to the number that was specified by the user in the measurement parameters. This function is described in the next figure.

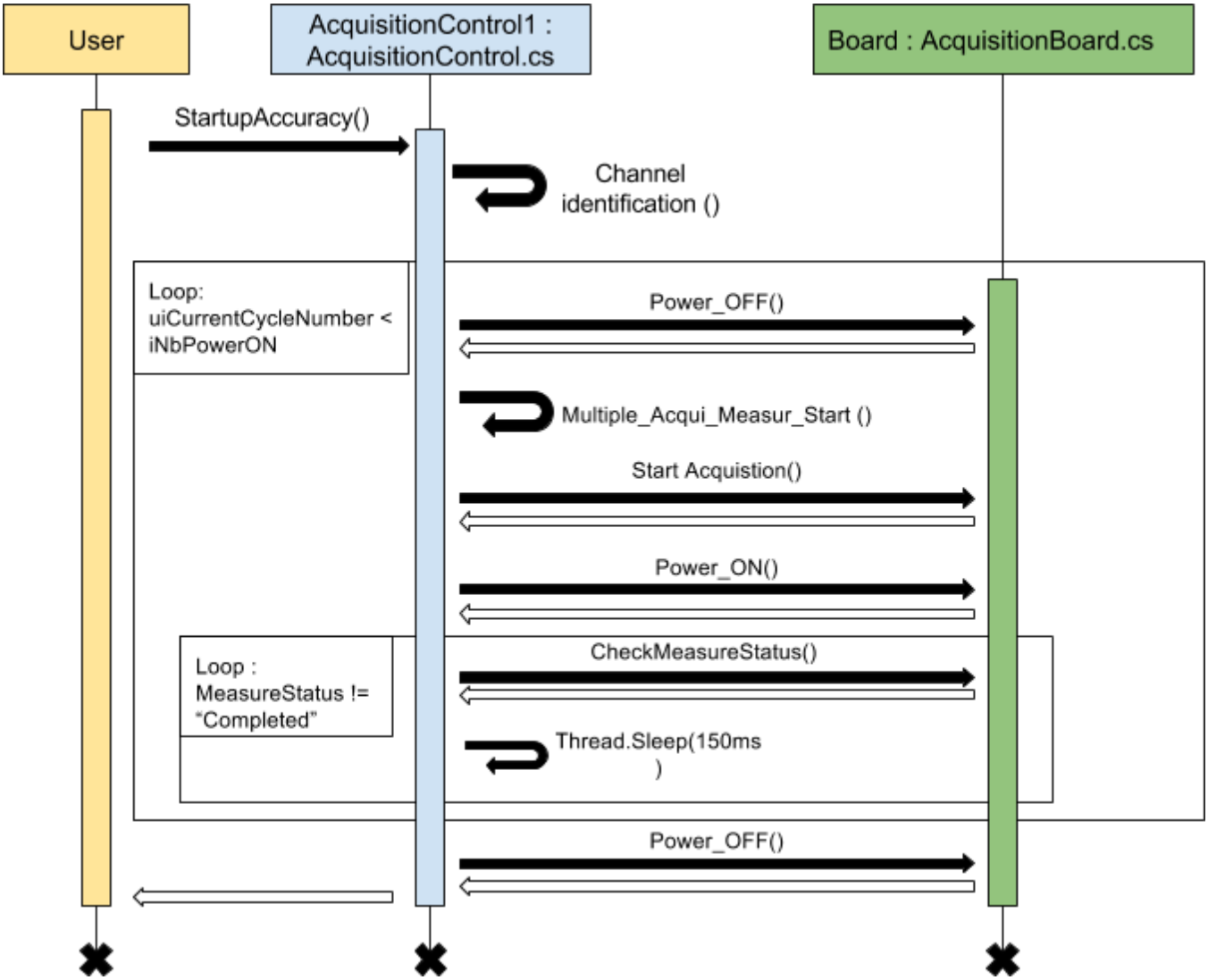


Figure 12: UML representation of the Startup Accuracy function

public *void* **Jitter(** *uint* uiChannel)

uiChannel = the channel used.

The Jitter function works in same way as the StartupAccuracy() function without the PowerOn() and PowerOff() cycle. See figure 12.

The Jitter function captures the number of specified edges on several revolutions.

public *uint* **SaveDataToExcelFile**(*Excel.Worksheet* xlWorkSheet, *int* iStartLine, *int* iStartColumn, *string* strComment, *double [,]* dTabAllData, *string* strTestType, *uint* uiChannel, *bool* bFirstsave, *string* strTestStartDateTime, *string* strTestEndDateTime, *string* strMotorSpeed, *string* strAirGap, *string* strMisalignment, *uint* uiEdgePerPeriod)

xlWorkSheet = the sheet of the Excel WorkBook that we are writing in.

iStartLine = index of the Excel line where we want to start to write.

iStartColumn = index of the Excel column where we want to start to write.

strComment = used to record various user comments (device ID....).

dTabAllData = data to be saved in two dimensional array of double [Sample, Cycle].

strTestType = test type (“Accuracy” or “Jitter”).

uiChannel = the channel to be saved.

bFirstsave = *true* means it is the first time that we are writing in this file.

strTestStartDateTime = the date and time at the beginning of the measure.

strTestEndDateTime = the date and time at the end of the measure.

strMotorSpeed = the current motor speed (target speed).

strAirGap = the current air gap (distance between the target and the sensor).

strMisalignment = the current misalignment (distance between center position and the sensor current position).

uiEdgePerPeriod = the number of edges that sends the sensor per magnetic period (depending on the sensor used).

The function returns the index of the last Excel line written.

It organizes the data depending on the test type (Jitter or Accuracy). For Jitter test, the data organization depends on the number of edges per magnetic period and the number of teeth on the target.

For Accuracy, the final data are formatted as : Cycles in line and Edges in column.

For Jitter, the data are formatted as : Revolutions in line and Edges in column. The number of edges per magnetic periods and the number of teeth on the target are used to find the number of revolutions and the number of edges per revolution.

The function also reports if an error has occurred during the test in the last Excel line.

public *void* **StopTest**(*void*)

In every tests functions a global variable *bool* bTestStop is tested to know if the test must be stopped. If bTestStop = *true* the tests is stopped, otherwise, it continues. This function sets this variable at *true* and stops the FPGA acquisition cycle.

IV. AutoTests/FPGA Interface

The AutoTests Interface is the user interface where the user can start automatic and/or manual tests, and interact with the motor and the axes. The Acquisition Control is one part of this user interface and is located in the Acquisition Tab. The other parts of this interface are simplified version of this Control designed to be user friendly and add some new parameters depending on the nature of the test. These parts are located in the Tests Tab. The class diagram can be found in the annexes.

IV.1 Automatic Tests Interface

The test interface (located in the Tests Tab) is divided in three parts :

- The control of the motor and of the axes which will be added later in order to move the target and the sensor.
- The test commands that allow us to start and stop automatic tests, to monitor the test status, to specified the target used and finally, to select the test type wanted (Accuracy or Jitter).
- The test parameters that are simplified version of the Acquisition Control and also add test-specific parameters.

The tests commands and the tests parameters are only available if the FPGA is connected.

The simplified test interface is linked with the Acquisition Control using Getter/Setter functions so that any modification on the test parameters is reproduced in the Control.

The test-specific parameters that have been added :

- The test type (Jitter or Accuracy).
- The number of teeth on the target.
- The number of cycles (Accuracy Test).
- The number of edges that the sensor sends per magnetic periods (Jitter Test).
- Tests commentaries.
- A user-friendly selection of the number of samples based on : number of teeth, number of edges per magnetic period and slope (Rising or Falling or Rising and Falling). An indication of the number of revolution corresponding is also indicated.(Jitter Test)

- The step size of the phased PowerOn (Accuracy Test).

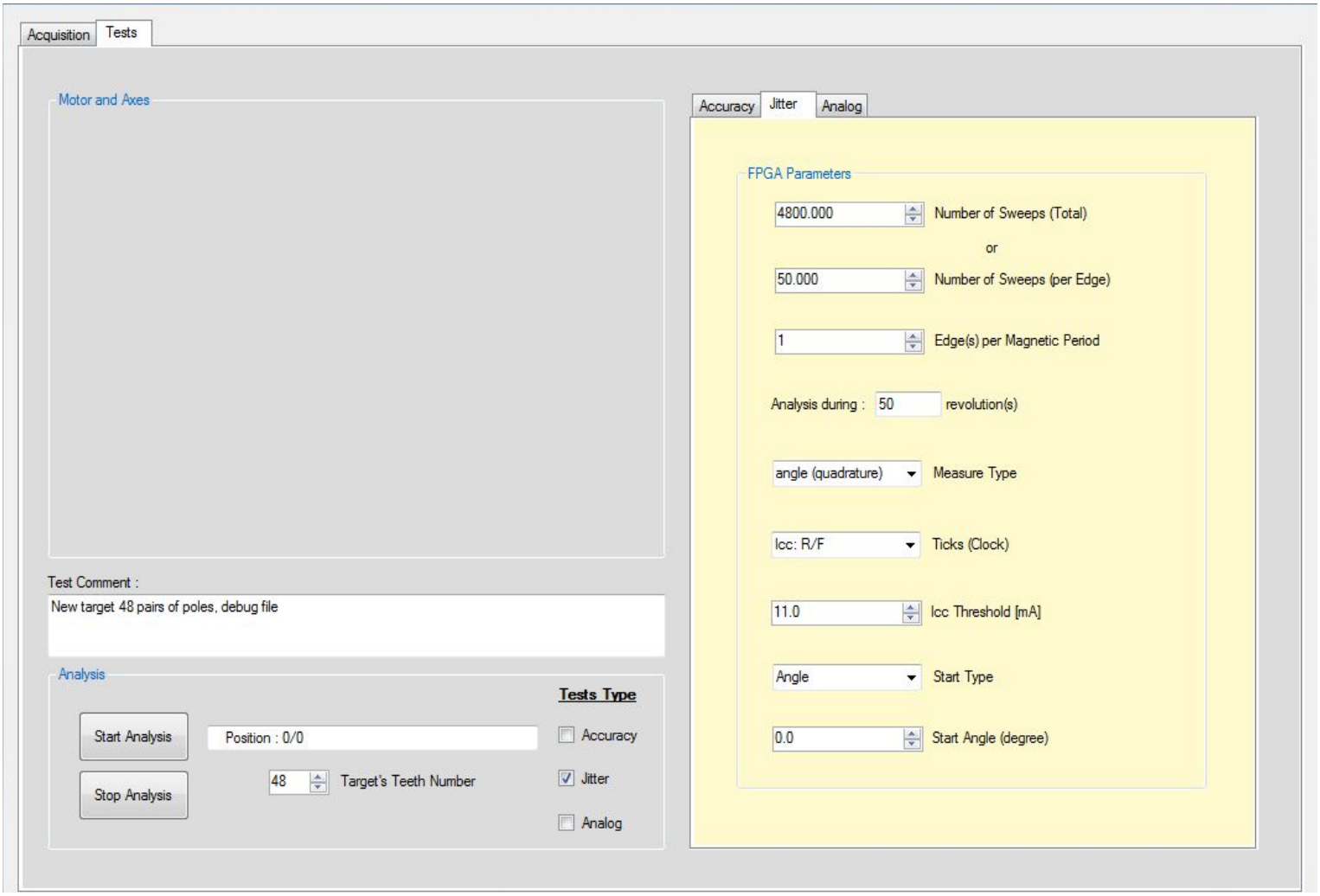


Figure 13 : AutoTests Interface

IV.2 Typical sensor tests

All automatic sensor tests with the user AutoTest interface use the same function called when the user clicks on the Start Analysis button. The test is identified and the corresponding setup is realized, then, function creates a new thread to avoid locking the form interface.

At the beginning of the new thread, all test parameters are gathered and an Excel application, an Excel workbook and an Excel WorkSheet are created (the Worksheet will be given as input parameter to the SaveDataToExcelFile() function of the Acquisition Control at the end of every measure). The file name is specified by the user thanks to a dialog box.

The Excel file will only be closed at the end of test to prevent the user from opening the file and making the test crash because the file is already opened.

In the same time, a timer checks the test status and refreshes the interface every second to inform the user of the test progression.

When the control of the motor and the axes will be added, the test will be repeated at every position.

Figure 14 explains what happen when a Jitter Test is started.

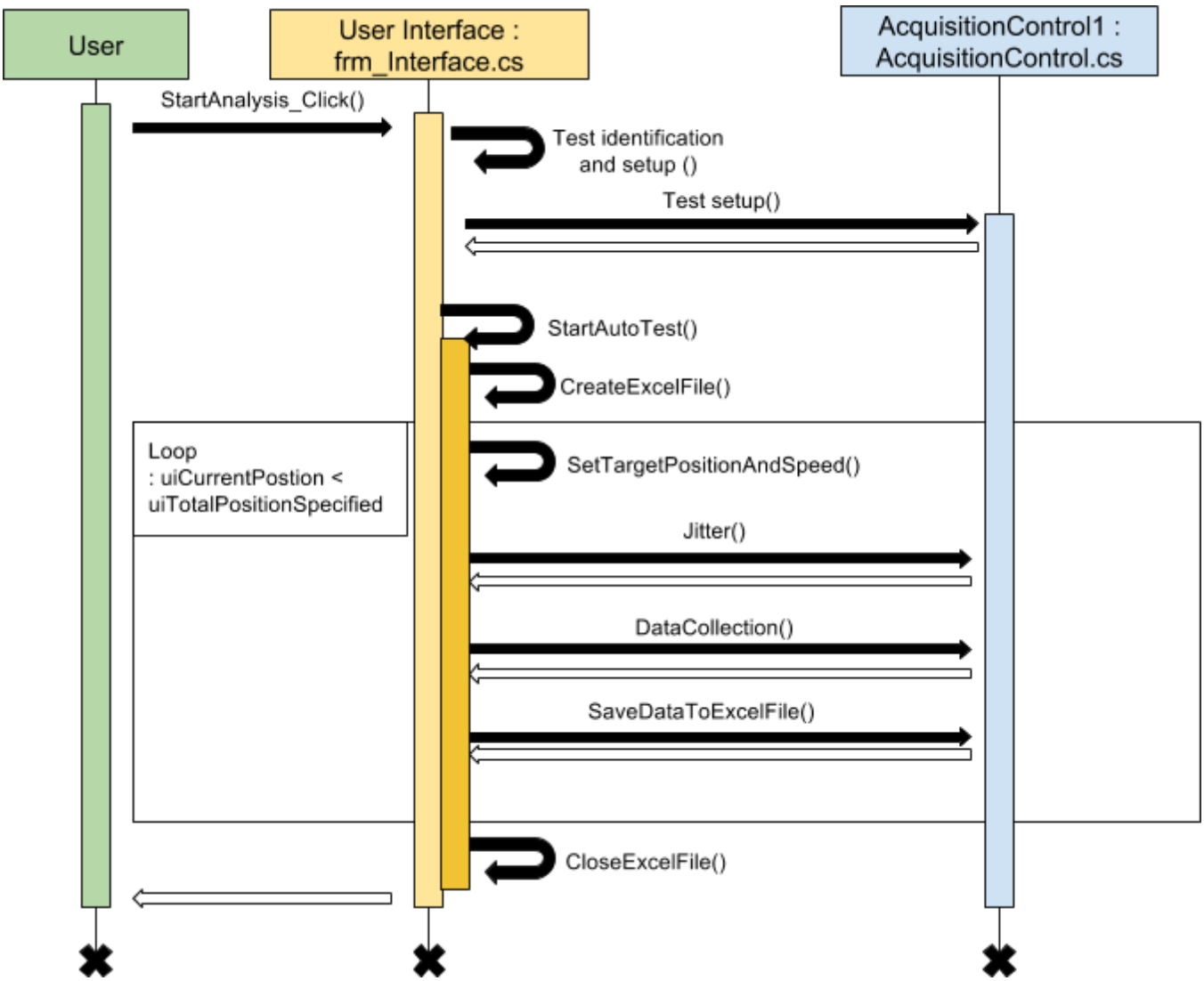


Figure 14 : UML representation of a Jitter Test with the AutoTest interface

The same thing happens when the user starts the Startup Accuracy Test but the function called is StartupAccuracy().

At the end of a test, the user can easily studied the data thanks to the data organization done in the SaveDataToExcelFile() function.

For example, the user has realized a Jitter Test, he will have an Excel file as in figure 15. In one column he will have the same edge during the specified revolutions. A simple Excel command (STDEV(array)), will return the corresponding jitter values for each edge.

Test started at : 5/31/2016 10:43:37 AM									
Test type: Jitter									
Channel : 1									
Measure type : Angle [°] (Quadrature)									
Signal acquired : Icc: Rise									
Comment : Jitter Debug on 10 rev									
Misalignment : -1									
Air Gap : 1.5									
Motor Speed : 100	Speed FPGA : 100,62								
Revolutio	0	1	2	3	4	5	6	7	8
1	13.08563	40.335	67.72125	95.08313	122.3288	18.0525	45.35625	72.71438	99.96
2	13.00688	40.35188	67.71188	95.07	122.3231	18.045	45.35625	72.68625	99.96188
3	13.00688	40.40813	67.71375	95.07375	122.3419	18.04313	45.32438	72.71438	99.96
4	13.00875	40.4025	67.70625	95.08125	122.3231	18.08625	45.35063	72.69188	99.95063
5	13.01813	40.38	67.69875	95.07938	122.3138	18.08063	45.34125	72.69375	99.9525
6	13.01438	40.39125	67.695	95.0775	122.3119	18.10125	45.34125	72.69563	99.96375
7	13.00688	40.37625	67.70625	95.0775	122.3213	18.11063	45.31875	72.6975	99.96375
8	13.01625	40.36875	67.695	95.08313	122.3044	18.04875	45.34688	72.69375	99.96188
9	13.0125	40.38	67.70063	95.07375	122.3006	18.09938	45.32625	72.6975	99.96375
10	12.975	40.41188	67.68563	95.07	122.3363	18.10875	45.3225	72.69938	99.96
Test ended at : 5/31/2016 10:43:45 AM									
Error :									

Figure 15 : Excel file after Jitter test

User can stop the test at any time by clicking on the Stop Analysis button (calling the StopTest() function of the Acquisition Control).

V. Conclusion

Project review :

At the end of the project, the initial objective has been reached. Manual and automatic tests can be performed thanks to the Acquisition Board and the C# program is ready to be integrated in other projects.

The Acquisition Control is independent and located in a library for more flexibility. The user interface is also ready to do two typical and automatic tests : "Startup Accuracy" and "Jitter" on channel 1 (but the intern functions in the Acquisition Control for these tests are also available for the other channels.).

The function MultipleAcquisitionCycle() and DataCollection() are independent of the test type and the selected channel. They can be reused for other custom tests and are made to be flexible.
In case other test types are implemented, the function SaveDataToExcelFile() must be updated, because it follows a different data treatment depending on the nature of test.

Most of the function were tested, but there are still some function that need to be verified.

Project cost :

The overall cost of the project is limited, most equipment have been developed and manufactured internally by Allegro. The external costs are :
FPGA cost : 595 €
11 weeks of internship : 1386€.
Total cost : 1981€

Possible upgrades :

- Create a ¨Script¨ where the user can enter keywords and simplified commands to run custom tests.
- Add an intelligent measurement timeout that stops the measure and alerts the user if the test is taking too long.
- Verify all functionalities of the interface.
- Complete the "Previous Settings" function in the Control to previous user setup.
- Add a "Mapping" Tab to use the FPGA as an oscilloscope.
- In the Accuracy function, implement a phase-controlled PowerOn.

Personal review :

This internship was a good experience for me because I had put in practice a lot of subject that I studied during my two years at the IUT. The hardest thing was to continue a pre-existing project, and in the same time, to understand how Allegro products work. Furthermore, the fact that Allegro is an international company helped me to improve my English skills both in technical language and in everyday language.

Thanks to:

- Allegro Microsystems Europe that allowed to make my internship in their company.

- The complete Allegro team for its warm welcome and support. I also want to especially thank my internship responsible, Mr.Loic Messier for having taking his time to help me to realize this project and complete my formation.

- The IUT d'Annecy-le-Vieux from the University Of Savoie Mont-Blanc, its GEII department and my teacher Mr.Stephane Bertrand that allow me to do this internship.

Allegro Microsystems Europe
74650 CHAVANOD
http://www.allegromicro.com

GOUILA Bastien
IUT Annecy
GEII Department
Year 2016

Summary :

The purpose of this project is to create a tests system for Allegro magnetic sensors in order to make all tests needed faster and easier. This system is made of four parts : the sensor, a magnetic target, an FPGA with an external daughter board and a computer program in C# language.

The sensor responds to magnetic stimulus and transmits data to the FPGA with electronic signals. To capture this signals, Allegro has designed a daughter board that can be fixed on the FPGA and controlled by it. When the acquisition board (FPGA + daughterboard) has finished capturing the sensor data, it is saved in the FPGA internal memory, the C# program running on the computer collects the data and saves it into an Excel file. The C# program is also capable of controlling the positions of the target and of the sensor.

My internship was focused on the C# program. The program is divided in three parts :

- The Acquisition Control which does all the communication with the FPGA.
- The sensor and target control. The C# program can modify the position of the target and of the sensor. It can also program the sensor if the right equipment is used. This part was already done before the internship started.
- The AutoTests interface which uses the two previous parts in order to do automatic tests.

I was charged to test and improve what was already existing and then to continue the Acquisition Control and the AutoTests interface in order to be able to make automatic and manual tests of Allegro sensors thanks to the FPGA.

Key-Words :

C# ; FPGA ; Magnetic sensors ; Encoder ; Serial Port ; Multi-threading