Politecnico di Torino

Master's Degree Course in Computer Engineering

Master's Degree Thesis

Development and test of an Iron Bird (Airplane wing simulator)

Supervisor:
Prof. Luca Ardito

Co-supervisor:
Prof. Maurizio Morisio

Candidate:
Chenguang Long

Academic year 2020/21
Torino

Abstract

***Context*:** Reducing the power consumption and thus the fuel burn is a major target for the next generation of aircraft. Two technological areas that can contribute to the power saving are wing load alleviation and electrical actuation. Load alleviation is a technique for redistributing aircraft loads encountered during flight with the purpose of reducing the wing root bending moment, hence allowing a lighter wing design with a resulting weight saving, reduction of the needed propulsive power. Electrical actuation can contribute to the reduction of the non-propulsive power because electromechanical actuators, when compared to the hydraulic actuators, rely on a power less subject to losses and lighter to distribute, besides presenting higher reliability and maintainability with a lower life-cycle cost.

***Goal*:** The aim of this thesis' work is simulating of the wing during a flight requires a number of computers: FMSC Flight Mechanics Simulation Computer to simulate the flight dynamics; FCC Flight Control Computer to control the airplane during the flight, HMS Health Management System for reactive and proactive maintenance.

Contents

List of Figures 4

1 SUMMARY 7

2 Architecture of Iron Bird 21

2.1 Introduction to real time simulation 22

Description of the Flight Mechanics Simulation Computer . 23

2.2 Flight Control Computer 31

2.3 Health Management System Module 33

2.4 Reflective Memory and Synchronization 37

3 Mapping Flight Mechanics Simulation Computer (FMSC) 39

3.1 Code generation 40

3.2 Deployment inside Veristand SystemExplore 41

3.3 Setting ports 43

3.4 Mapping set 45

For example 47

4 Conclusions 49

Bibliography 52

List of Figures

1.1 Model-based Software Engineering flow 13
1.2 V-shaped development flow of Model-based Software Design 15

2.1 Architecture of IB 22
2.2 Communication of architecture 23
2.3 Rigid model in Simulink. 24
2.4 Rigid and Flexible models. 25
2.5 Bending Right Wing. 26
2.6 Torsion Right Wing. 26
2.7 Accelerations at the wing root. 26
2.8 Hinge Moments Block. 27
2.9 Elevator Hinge Moment. 29
2.10 Rudder Hinge Moment. 29
2.11 Right Aileron Hinge Moment. 29
2.12 Right Wingtip Hinge Moment. 30
2.13 HMSM 33
2.14 Interface 34
2.15 Features Behaviour 35
2.16 Histograms (fault detection in progress) 35
2.17 Confidence (fault detection completed) 36

Chapter 1

SUMMARY

The thesis project develops an Iron Bird, a simulation of a wing of a regional airplane. The simulator is partially physical (half wing, sensors, actuators) and partially virtual (second half wing). The project is part of the AstIb Clean Sky2 (CS2) European project. The ASTIB (development of Advanced Systems Technologies and hardware/software for the flight simulator and Iron Bird demonstrators for regional aircraft) project brings together 7 European companies and academic partners led by LEONARDO. It aims at supporting the improvement of the Technological Readiness Level for a significant number of equipments that are being considered of critical importance for the future Green Regional Aircraft (GRA). The design and production of the Iron Bird is responsibility of CERTIA. The Iron Bird is the ground test bench allowing the integration of the different aircraft systems. This Iron Bird is equipped with new innovations (semi-virtual, innovative loading systems, health monitoring, etc.). The thesis work is about receiving the models (developed by aeronautical engineers), deploying them on the computers, testing and improving them until complete validation. The simulation of the wing during a flight requires several computers: FMSC (Flight Mechanics Simulation Computer) to simulate the flight dynamics; FCC (Flight Control Computer) to control the airplane during the flight, HMS (Health Management System for reactive and proactive maintenance. The computers rely on various MatLab Simulink models. The workflow follows the principles of the Model-based software design, that is based on the development of a model of the plant and the controller with focusing on the details that are useful to understand system's behaviour. A significant advantage of following that approach is the automated code generation, which allows to automatically translate the Simulink model into code that can be executed on a dedicated hardware. In a first phase models are tested in a simulation environment. Then they have to be executed in real time to verify the interactions between the models and the shared memory. Executing a model in real time requires the usage of high-performance platforms. The chosen one is NI VeriStand, a software designed also for Hardware-in-the-loop (HIL) and Software-in-the-loop (SIL) simulations. In order to import the model in NI VeriStand we need a dll (dynamic linking library) file. It is obtained as result of code generation. The models received from the partners are modified to make them suitable for code generation. This means substituting every source block with an Input Port and every sink block with an Output Port. Then code is generated and the model, translated into a dll, is ported to NI VeriStand. Next step is, exploiting VeriStand, to map inputs and outputs on the reflective memory. In this way the 3 models composing the project shall be able to interact with each other. Last step is, indeed, to verify their interactions.

Air transport contributes today about 3% to global greenhouse gas emissions, with traffic expected to triple by 2050. Although other sectors are more polluting (electricity and heating produces 32% of greenhouse gases), this expected growth makes it necessary to address aviation's environmental impact. Meeting the EU's climate and energy objectives will require a drastic reduction of the sector's environmental impact by reducing its emissions. Maximising fuel efficiency, using less to go farther, is also a key cost-cutting factor in a very competitive industry – and as air traffic increases, better noise reduction technologies are needed. But game-changing innovation in this sector is risky, complex and expensive, and requires long-term commitment. This is why all relevant European stakeholders must work together to develop proof-of-concept demonstrators. Reducing the power consumption and thus the fuel burn is a major target for the next generation of aircraft. Two technological areas on which can be possible work are wing load alleviation and electrical actuation. Load alleviation is a technique for redistributing aircraft loads encountered during flight with the purpose of reducing the wing root bending moment. Electrical actuation can contribute to the reduction of the non-propulsive power because electromechanical actuators rely on a power less subject to losses and lighter to distribute, besides presenting higher reliability and maintainability with a lower life-cycle cost. These two areas have been widely addressed in the past years. Over the last years, several industrial programmes initiated the concept of a More Electric Aircraft. In particular, the aero-equipment industry has started to develop a more electrical actuation with Electro Hydrostatic Actuators (EHA) and to introduce electromechanical actuators (EMA) for auxiliary equipment. This has provided incremental approaches to address hydraulic circuits issues (Power-by-Wire technologies, 2-hydrulic/2-electric power distribution architecture and use of EMA for some systems). Several collaborative research and development projects also started to develop an All-Electric Aircraft, showing the effectiveness of electrical actuation. EMA systems are so considered the best option for the aircraft of the future considering they are:

- Less complex
- Better suited to long term storage
- Energy saving with respect to hydraulic systems
- Installation and maintenance are easier

• Power distribution management easier.

Nevertheless, there still exist technological barriers for a wide adoption of EMA. For example, the sensitivity to certain single point of failures that can lead to mechanical jams, and so on. The promising perspectives of load alleviation / load control technologies as well as electrical actuation for flight control surfaces and landing gear need to be thoroughly investigated and verified in order to gain the necessary confidence and maturity level for moving to their implementation in a flying demonstrator.
This requires:

•Development of suitable prototype components integrating the innovative features capable of making electrical actuation an accepted proposition for future flight controls and landing gears

•Design and construction of an integration test rig (Iron Bird) allowing verification and validation of:

*o*Enhanced electrical power distribution and load management

*o*Electrical landing gear technologytechnology

*o*Flight control system technology

•Development of a health monitoring platform able to collect and process data provided by the sensors of the aircraft structure and systems such to assess the aircraft health status

ASTIB – Advanced Systems Technologies and hardware/software for the flight simulator and Iron Bird

The ASTIB (development of Advanced Systems Technologies and hardware/software for the flight simulator and Iron Bird demonstrators for regional aircraft) project brings together 7 European companies and academic partners led by LEONARDO. Its goal is to develop new technologies for the future regional aircraft.

ASTIB is specifically focused on contributing to generate technological advancements to be implemented in a future Green Regional Aircraft (GRA); its main objectives are to support the improvement of the Technological Readiness Level up to above TRL 5 for a number of significant electrical equipment that are being considered of critical importance for the future GRA. This will help supporting industrial application decisions for future deployments of GRA. In particular, ASTIB will develop:

- Electromechanical actuators (EMAs) with their associated electronic control units (ECUs) for selected flight control surfaces
- Electrically actuated one main and one nose landing gears
- Reliable prognostics and health management functions for the electromechanical actuators
- An advanced multi-functional integration, verification and validation test rig ("Iron Bird")
- Contribution to the development of a health monitoring validation platform
- Tools for evaluation of the benefits of an integrated health monitoring platform.

MODEL-BASED SOFTWARE DESIGN

The project has been developed following the principles of Model-Based Software Design. To understand what this means, let's start with an example. Let's think we want to design a controller for an industrial robot or a vehicle and put our attention on the code necessary. Software complexity is becoming one of the dominant cost items in several sectors, including automotive, avionic and industrial markets. It can be reduced using a structured process, a more effective way of developing and validating the code, a more effective way of reusing code. Following a traditional approach implies handwritten code and the only way to verify requirements fulfilment is through a "trial and error" procedure. Tests can be done only at the end of the process and on a physical prototype, increasing costs. If a single requirement changes, the whole system must be redesigned, increasing the already large time needed for production and delivery of the final product.

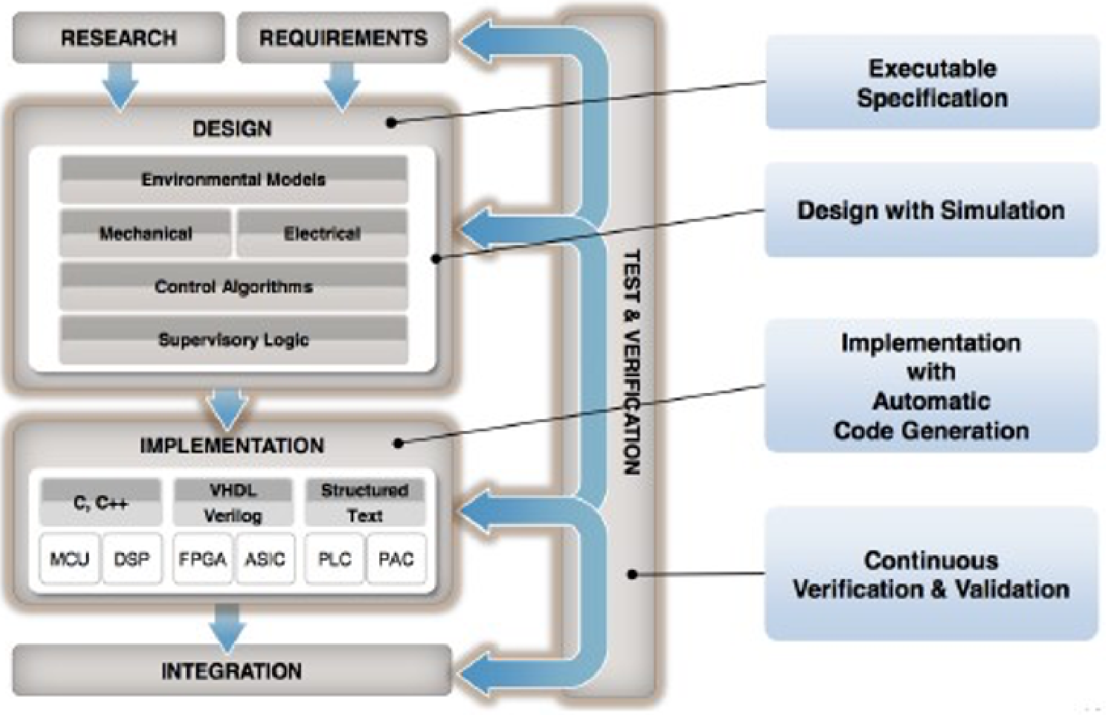


Figure 1.1. Model-based Software Engineering flow

Model-driven engineering (MDE) offers an alternative that allows to reduce time-to-market and development costs. Principles of MDE are:

•Abstraction from specific realization technologies

*o*A model should care about system's behaviour not its implementation

*o*Requires modelling languages that do not hold specific concepts of realization technologies

•Automated code generation from abstract models

•Separate development of application and infrastructure.

According to this approach, there will be a software tool to translate a project in software. Usually the project consists of one or more models. A model is a representation of the original system that includes only the key features useful to understand its behaviour. Modelling is the key of this approach. The functionality of the application we want to develop will be modelled using a language that is abstract enough to allow to work without specifying too much details (e.g. Simulink and Stateflow). A transformation tool will automatically generate the code relative to the model, applying to it some transformations defined through a specification language. The advantage of this approach is that the model can be continuously refined throughout the whole development process and it can be simulated at any time to see how the system behaves. Multiple scenarios can be tested without any risk and without using specific and expensive machinery. The typical design flow of model-based software design is the so-called "V-Shaped" development flow, shown in the picture below.

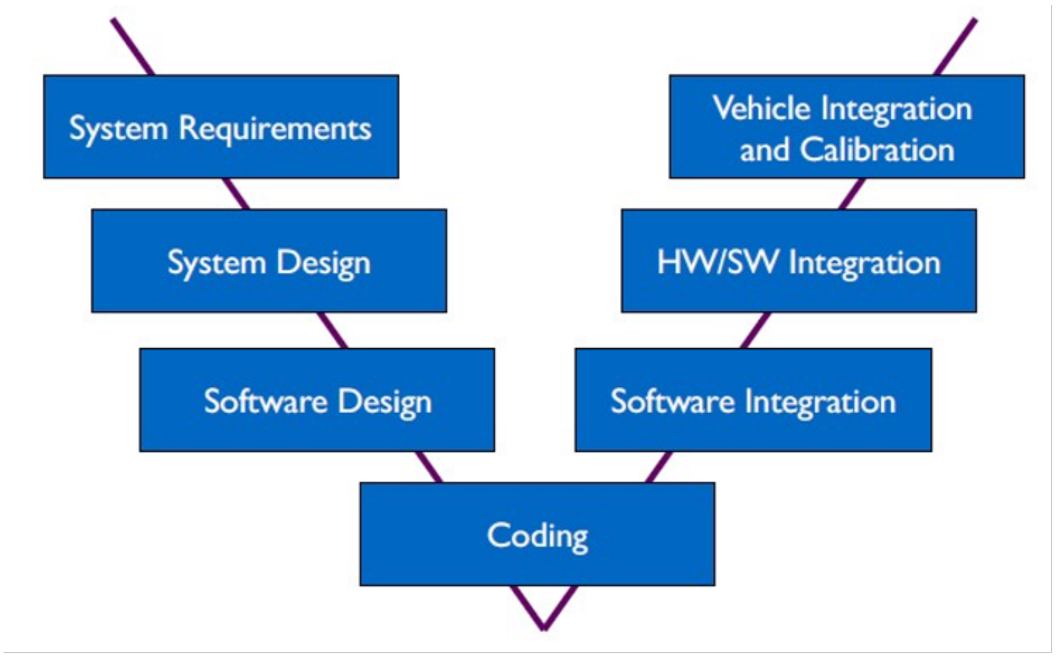
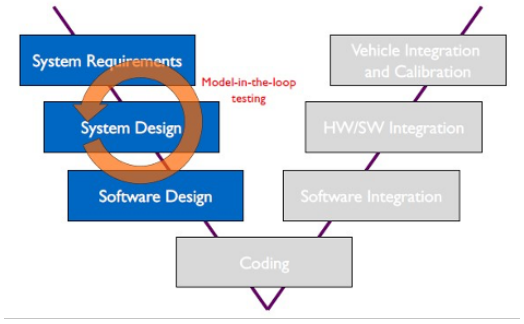


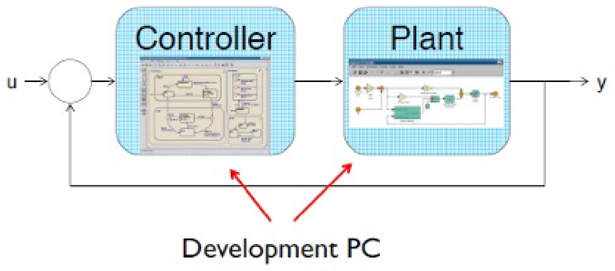
Figure 1.2. V-shaped development flow of Model-based Software Design

It is possible to perform simulation during the whole process in order to avoid unwanted behaviours at the end. The best thing is to perform simulations to verify each step to avoid going back of several steps if an error occurs. Different types of test can be performed: Model-in-the-loop (MIL), Software-in-the-loop (SIL), Processor-in-the-loop (PIL) and Hardware-in-the-loop (HIL).

Model-in-the-loop testing

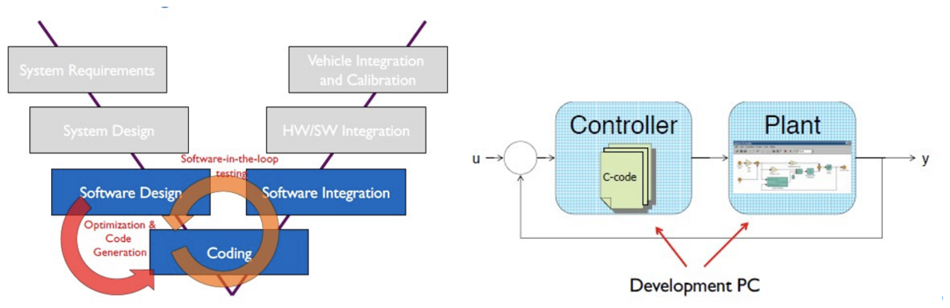
It can be performed in the first steps of the V-Shaped design flow when both the plant (system to be controlled) and controller (algorithm controlling the plant) are modelled. It is performed on a PC. Models are developed using a suitable domain-specific language (e.g. Simulink/Stateflow). The simulation is used to validate the correct behaviour of the controller looking only at the functional aspect. For example, we can test if the plant is able to follow a given input.



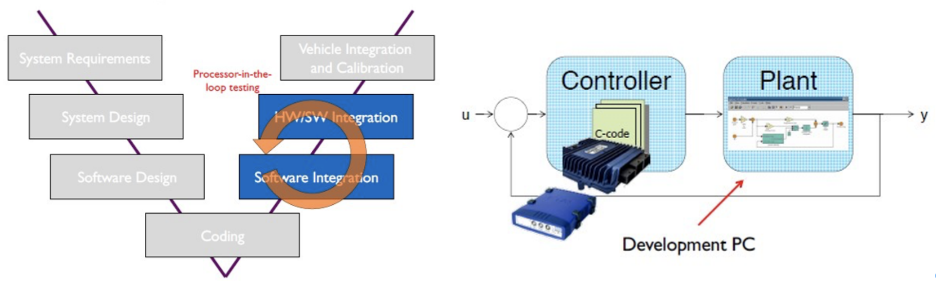


Code generation and Software-in-the-loop testing

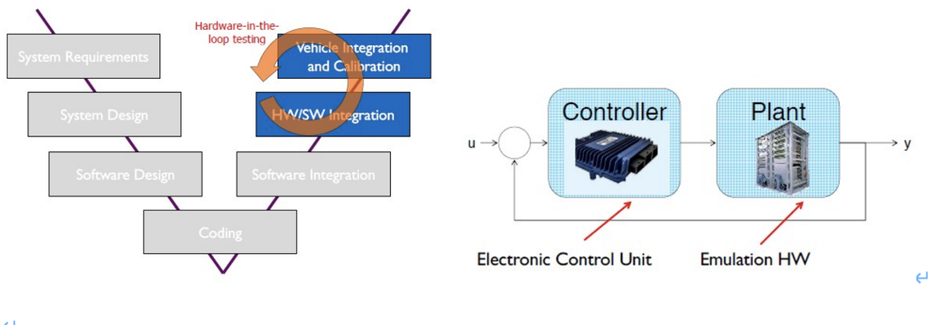
When MIL tests give acceptable results, the model of the controller is translated into code by means of the so-called code generation. It implies the application of some transformations defined by trough a specification language. For example, it allows to transform a Simulink model into C code. Software-in-the-loop is the step done after code generation to validate the correct behaviour of the software resulting from the controller model. The software is co-simulated with the plant model, by executing both on the development PC. The goal is to evaluate the behaviour of the code resulting from the controller model, looking again at its functional aspect.(Wikipedia, n.d.).



Processor-in-the-loop testing After validating the software, the successive step is the Processor-in-the-loop test. It is performed using different machines. The model of the control algorithm is translated into code and deployed on an embedded processor (e.g. an evaluation board or an ECU). It is then executed in combination with a model of the plant running on a simulation environment into the development PC. In this way, the designer can verify the correct operations of the code implementing the controller while running on the hardware that will be used in the final product or one close to it. In this kind of test we are interested in the functional aspect of the controller too, still neglecting the real time behaviour.



Processor-in-the-loop testing The last type of test is called Hardware-in-the-loop (HIL). It is performed using different machines. It consists in running a software implementation of a control algorithm in a microprocessor-based system, for example an evaluation board or the target hardware, in combination with a real-time plant model that is executed by a real-time computer. The input/output connections between the two elements are implemented using the very same connections that will be used in the real application. Some hardware components belonging to the physical plant may also be connected. In this stage, validation of the software is done in real-time, consider both the functional and timing properties of the software.



Goals and used tools

The thesis work aims to validate the models, received from and designed by aeronautical engineers, that are part of the Iron Bird. This implies deploying them on the development PC, testing and improving them until complete validation. In particular, it is about performing SIL tests using NI VeriStand.

MATLAB Simulink

According to MBSE the key concept is the model of the functionality that we want to design. A model reflects the relevant section of the original system properties, so focussing on important properties only.

EXPECTED RESULTS

The technologies developed under CS2 will reduce environmental pollution and noise levels and will therefore improve the quality of life. The close collaboration between the partners of CS2 will accelerate the pace of technological progress and create a mutual win-win situation. CS2 will help Europe's aeronautics sector remains competitive. Europe currently has a world market share of 40% and the global aviation sector is expected to grow by 4-5% per year. But faced with fierce competition Europe needs to develop new technologies to create new market opportunities and new highly specialised jobs. For stakeholders in the EU-13 countries the new, enlarged JTI offers more opportunities to participate in building the best technologies.

Chapter 2

Architecture of Iron Bird

2.1 Introduction to real time simulation

In this part all models are built and running by Matlab/Simulink in PC environment.The Figure represents the architecture of the IB. It demonstrates the various systems integrated into the IB and the communicative relation between each component of the test bench. The Simulation module should simulate in real time

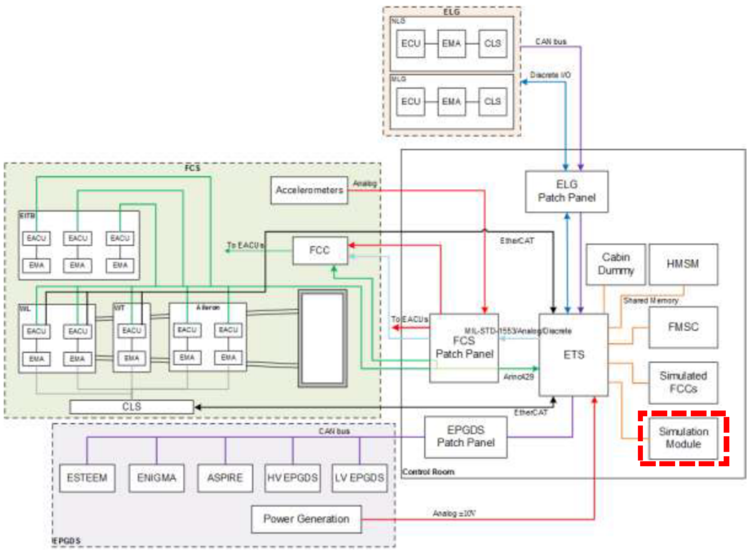


Figure 2.1. Architecture of IB

environment and get response from the modelled Iron Bird equipment.Since only the half side of the Iron bird is the real wing(left semi wing),so the right side will be simulated in the Simulation Module way in order to obtain full ground test results. The Simulation Module shall be designed and developed following a modular and parametric philosophy.

The FMSC FCC and HMSM are parts of the Iron Bird. They are designed and integrated and operated to:

2.1) *FMSC:* The primary purpose of the Flight Mechanics Simulation Computer (FMSC) is for the aircraft to be able to perform simulations(both longitudinal and lateral directional plane) under real-time conditions to verify the performance of various devices in a real-time environment.The model is based on the aircraft selected for the Iron Bird experiments, which will be a turbo-prop aircraft with 90 seats (TP90), it can be dividend into **Aeroelastic Model** ,

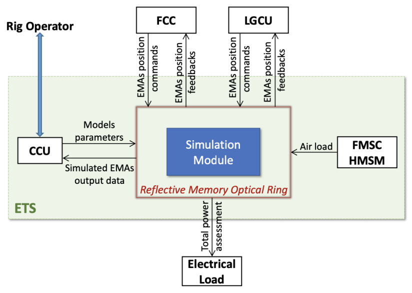


Figure 2.2. Communication of architecture

Rigid Body Mechanics , **Flexible Body Mechanics** , **Gust Simulation** , **Hing Moments**.

The FMSC accepts the following inputs: elevator deflection $\boldsymbol{\delta}$**e**, rudder deflection $\boldsymbol{\delta}$**r** , right aileron deflection $\boldsymbol{\delta}$**aR** , left aileron deflection $\boldsymbol{\delta}$**aL** , and winglet deflection $\boldsymbol{\delta}$**w** or wingtip deflection $\boldsymbol{\delta}$**wtip**. The whole systems are developed in Matlab/Simulink environment and every block,subsystem are validated using a fixed-step integrator(Runge-Kutta) with a sample time $T_s = 0.01s$.

During simulation we set mass equal to 20147 kg,mach speed equal to 0.3, without aerodynamic derivatives and altitude is 20000ft in Turin with innovative wing tip.

2.2) *FCS: The Flight Control System (FCS) is the part of the IB in order to control the airplane during air. It consists with Flight Control Computer (FCC), the Remote Interface Unit (RIU) and some other subsystems representing Winglet and Wingtip that refers to the Electronic Actuation Control Unit (EACU).*

2.3) *HMSM:TheHealth Management System Module(HMSM) wilL provide an advanced integrated reasoning system that integrates a set of tools that can automatically handle failures based on predictive information to enhance vehicle safety and decision support.A health monitoring verification platform and tools that can evaluate overall performance are also one of the goals of the ASTIB project.*

Description of the Flight Mechanics Simulation Computer

2.1.1) *Aeroelastic Model:* The aeroelastic model of the aircraft is developed under some assumptions,that are introduce this paragraph:

– the inertial tensor in body reference frame $\mathbf{I}_B$ is considered diagonal.

- the only deformable element for this 6 DoF aircraft model is the wing.
- the flexible components and deformations are modelled considering only the 1^{st} and 2^{nd} symmetric bending modes and the 1^{st} symmetric torsional mode.
- the A/C weight is constant in time.
- the sweep angle $= 0$.

In order to reduce the complexity of computation and reduce the computational load as much as possible, Strip theory modelled is adopted to simulate the aerodynamic force and moments. Under this theory, the load is only related to the flow direction of 'local flow'.The natural modes of the structure are used with the Galerkin method to express the deformation as truncated series expansions.

2.1.2) *Description of the Rigid Body Mechanics:* Besides the initial trim condition, the input of this model is a kind of step variation function. In addition, due to the lack of aerodynamic parameters (such as the corresponding derivative), the model does not include winglet deflections.

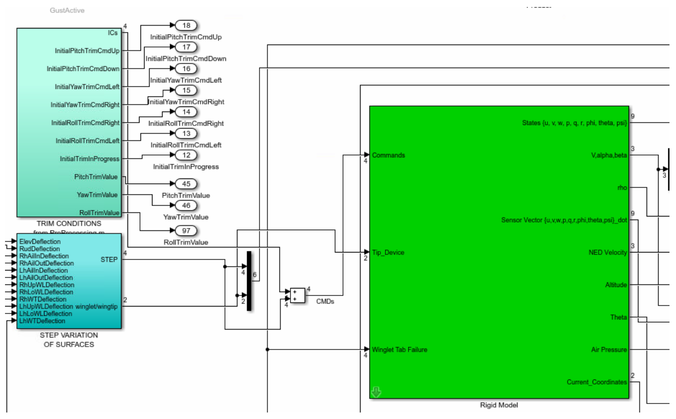


Figure 2.3. Rigid model in Simulink.

•As outputs, the system evaluates:

- The state vector x = u, v, w, p, q, r, ϕ, θ, ψ, where u, v, w are the linear velocity components of the A/C in body reference frame, p, q, r the angular rates and ϕ, θ, ψ the Euler angles.

- An auxiliary vector which collects V, α and β, where V is A/C speed, α the angle of attack, β angle of sideslip is also included.
- The air density ρ, calculated thanks to the ISA atmosphere model, and the altitude.
- The velocity vector in North-East-Down (NED) reference frame.
- Moreover, the accelerations of the state vector are included for the sensor model.

2.1.3) *Description of the Flexible Body Mechanics* The module connected with Rigid model is flexible body, which has 6 DoF. The flexible model is constructed

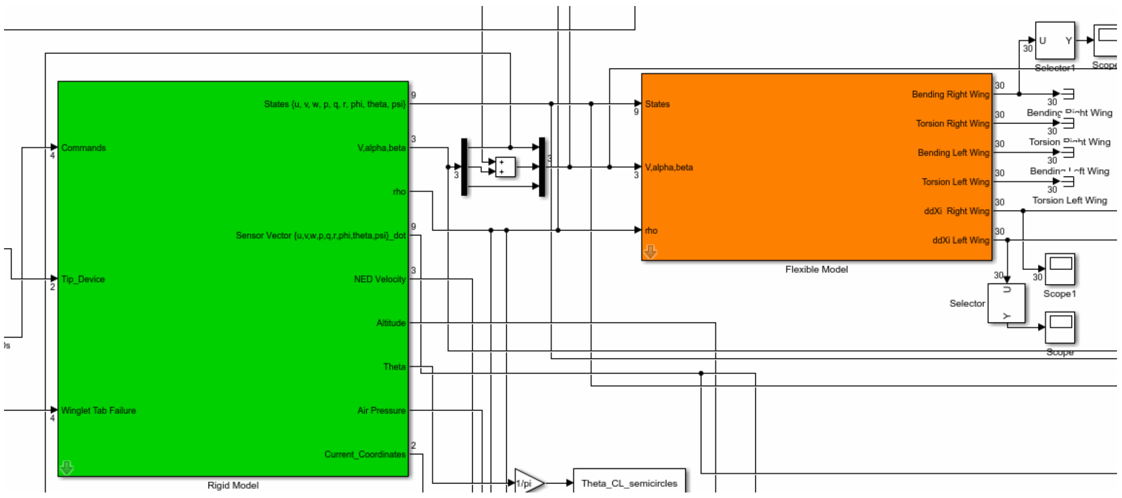


Figure 2.4. Rigid and Flexible models.

based on the Lagrangian approach, which considers the wing as a deformable component. While the tail and fuselage are regarded as rigid parts.The deformation state of the whole flexible part will be described based on the Galërkin method that is to evolution bending and torsion of wings.

– *Bending Right Wing:*

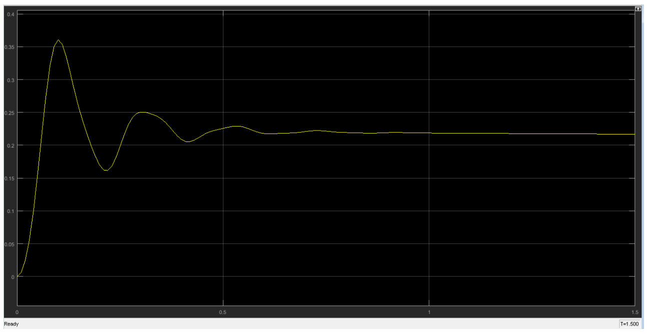


Figure 2.5. Bending Right Wing.

– *Torsion Right Wing:*

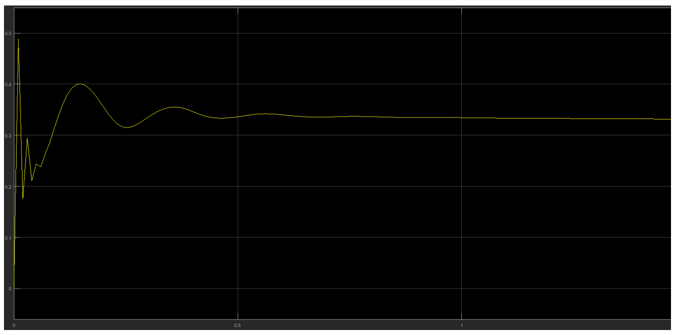


Figure 2.6. Torsion Right Wing.

– *Accelerations at the wing root.*

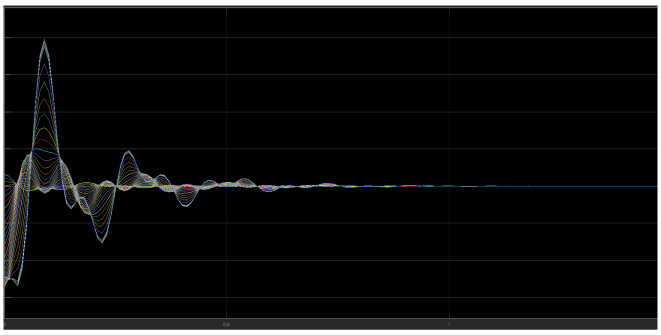


Figure 2.7. Accelerations at the wing root.

In the model both half wings are considered, but in this section only right half wing results will be represented to avoid redundancy.

2.1.4) *Description of Gust Simulation:* Gust input causes a variation of the system aerodynamics, with an induced incidence on the wing for every node of the aerodynamic model.

2.1.5 *Description of Hinge Moments:* This subsystem(pink one) connected with Rigid model works to calculate the hinge moments acting on the control surface. The input source of the module is the velocity of A/C , the deflection of the elevator, and the density of the air, while the output source is the hinge moments acting on the aileron, elevator, and rudder.

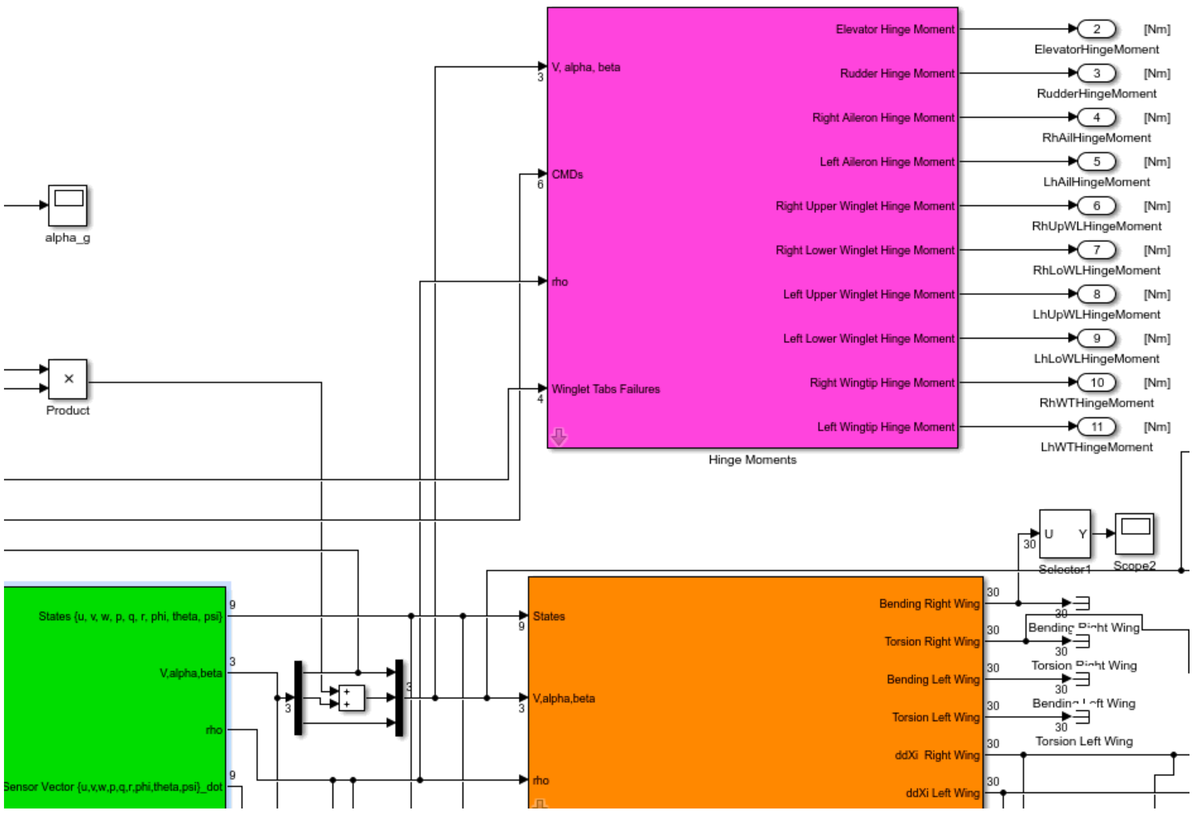


Figure 2.8. Hinge Moments Block.

Inside this block there are three different sections, one for each control surface. Both the aileron and the winglet surface are modelled starting from data received from LDO and CIRA. No data are provided for the elevator and rudder surfaces. (Garrison, 2018).

Followings are relative performance .

- *Elevator Hinge Moment:*

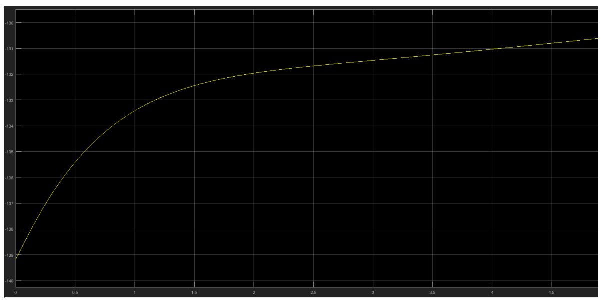


Figure 2.9. Elevator Hinge Moment.

- *Rudder Hinge Moment:*

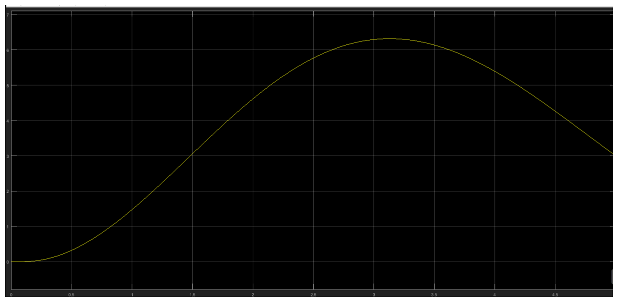


Figure 2.10. Rudder Hinge Moment.

- *Right Aileron Hinge Moment*

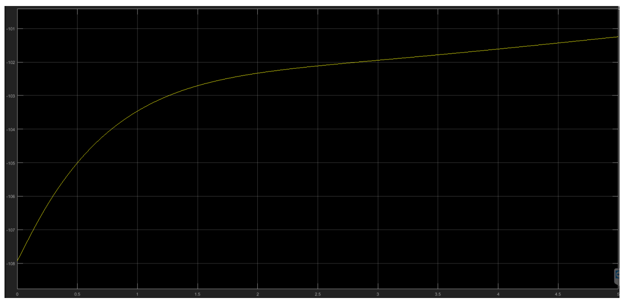


Figure 2.11. Right Aileron Hinge Moment.

- *Right Wingtip Hinge Moment:*

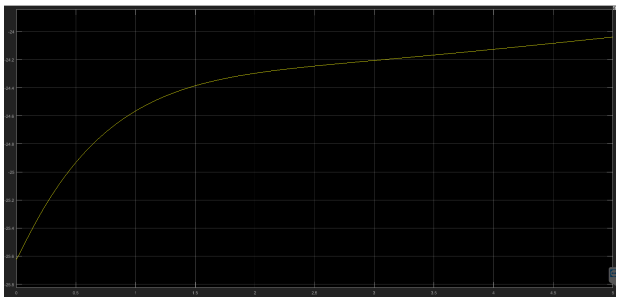


Figure 2.12. Right Wingtip Hinge Moment.

2.2 Flight Control Computer

2.2.1 The model consists of following functionalities:

- Interface management:
  The subsystems represented by FCCs , RIUs and Actuation are interconnected by A429 ports.In the simulation of Real time,the multiplexer/demultiplexer was used to connect each other in Simulink and the corresponding simulated fields are also grouped.
- Simulation :
  The main purpose of this function is to validate the model behavior of this real equipment.
- Stimulation – Type I
  In order to make the model be able to get the expected output,this type of stimulation will be performed inside the modeled FCC subsystems. The following commands are processed during the run:
  - The enumerative contents are modified acting on “Manual Switches”.
  - The data contents are modified acting on “Slider Gains”.
- Stimulation – Type II
  This type of stimulation is in order to active the actuation subsystem by introducing instructions for similar malfunction or failure. According to the final Iron Bird utilization, type 2 stimulation shall be commanded by the ETS, providing inputs to the “STIM inputs” ports of the AIL/WT/UWL/LWL.dll/.exe
- Stimulation – Type III
  This type of stimulation is performed by applying data from Cabin Dummy (trim commands and failure reset command) and acceleration from A/C to the FCS model. These inputs are connected to actuate subsystem of the FCC or WL/WT. According to the final ETS(Engineering Test Station) for stimulating these subsystems.

2.2.2 The model consists of the following subsystems:

- 3 FCC subsystems: FCC A,FCC B and FCC C

All outputs of the system are determined by the response of FCC Models(type I stimuli).The corresponding response and interface also contains both trim and Failure reset commands(type III stimuli).COM1 and COM2 lanes are equivalent to S1 and S2 sections. They are independent of each other.

- 3 RIU subsystems: RIU A,RIU B and RIU C
  The re-routing messages between the input ports and output ports are provided by RIU models.So the main role of these models is to manage the interface.
- 4 Aileron subsystems: 2 Wingtip subsystems and 4 Winglet subsystems are representative of the actuation systems on the A/C wings (flap and spoilers excluded)
  COM and MON lanes are modelled independently of each other, but they can interact with each other when needed.Both lanes are fed with Type II stimuli and, for WT and lower WL only, with the A/C accelerometer simulation (type III stimuli).

2.3 Health Management System Module

The Health Management System Module will perform the following key functions:

- Allow injection of simulated degradations and their progression to verify the merits of the PHM algorithms.

- Receive and store actuators measured data.

- Implement the prognostics and health management functions developed in the research project.

As shown in the figure below, HMSM can adjust the response parameters of the simulated electromechanical actuators. In general, HMSM will receive signals from reflective memory, including test and real EMAs, as well as some additional variables that help PHM analysis. The user will be able to enable

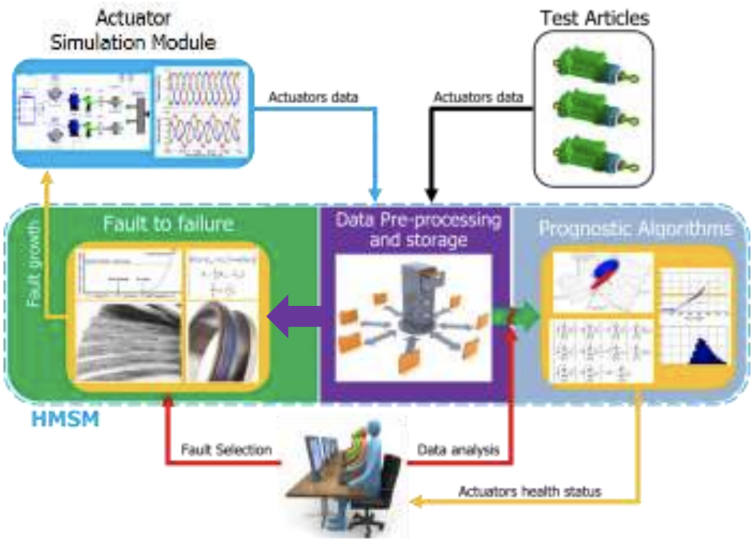


Figure 2.13. HMSM

the storing of data and will have access to the saved file for the unit management and for selecting the data to be analysed by the prognostics and health monitoring functions. The data will be stored after an eventual pre-processing (e.g. features extraction, fault detection) to limit the file dimension. Upon user request it will be possible to run the PHM algorithms that will retrieve the selected data and provide as feedback to the user the EMAs health status and all relevant information.Because the Prognostics Function will run on hardware in Iron Bird as well as on external computers, the algorithm does not need to run in real time. So HMSM has two sets of software that are composed to inject errors and store data in real time and PHM algorithm to work offline. HMSM interconnects with other devices in Iron Bird through reflective memory card.A TCP-IP connection is foreseen for loading reference parameters and files during the initialisation phase of the Iron Bird. The just presented functionalities allow to define following operating modes:

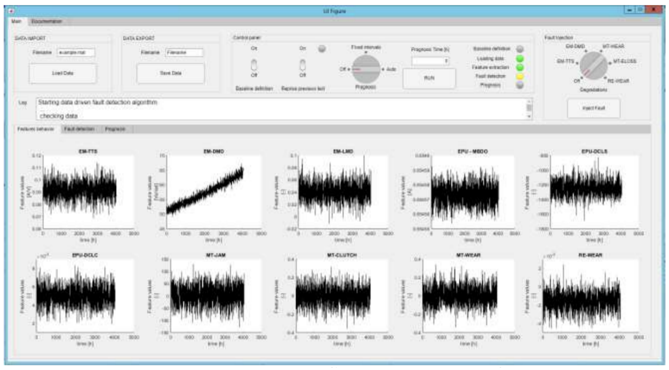
– HIL tests: user can select an actuator model, a specific fault and a fault-to-failure trend to create a data set to be analysed by the PHM algorithm.
– Passive Mode: data coming from Iron Bird tests can be periodically stored and analysed to assess the health status of the real equipment.
– Active Mode: inject a fault into the mathematical model of the EMA while user is performing normal Iron Bird tests or complete HIL tests.

The interface is divided in two main tab groups: “Main” and “Documentation”.The “Main” tab can be generally divided in two parts; one dedicated to the input through which the user can interact with the software and another one used by the software to provide visual feedbacks. The purpose of "DATA

Figure 3: Graphical User Interface

Figure 2.14. Interface

IMPORT" panel is to load data from user-defined files, and the algorithm will examine these file names and the data contained in the files. At the same time, during the loading process, the "Loading Data" indicator will be displayed in orange. Once the data is successfully loaded, it will turn green. If the light is red, it indicates that the load failed and the "Run" button function will be restricted.Under the green light, press the "Run" button, and the PHM algorithm will be executed.The algorithm's feedback results will be provided by a GUI with three graphical options: "Features Behaviour", "Fault Detection", and "Prognosis". The "Features Behaviour" TAB shows features related to the running state during the test.



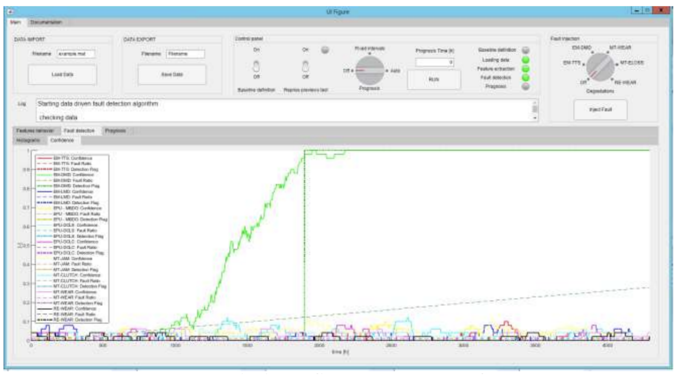
```
Starting data driven fault detection algorithm
checking data
```

Figure 2.15. Features Behaviour

Fault Detection consists of two tabs in which the characteristics of the system are represented by a histogram that effectively highlights exceptions when they are detected.

Figure 2.16. Histograms (fault detection in progress)

The confidence information reflected by "confidence" is related to the association level of the exception declaration, thus identifying the size of the injection defect.



Starting data driven fault detection algorithm

checking data

Figure 2.17. Confidence (fault detection completed)

The prognosis tab allows to depict the performed RUL estimates. By default, only the first prediction is shown, but users can change this by operating the dedicated slider.

2.4 Reflective Memory and Synchronization

Since "Iron Bird" is a system composed of multiple processors, many computers need to communicate with each other.Reflective memory is a very efficient way to address that need.There are two main types of systems based on how data and information are shared. The first type of system has multiple terminal processors and one global physical memory. In the latter case, each processor has its own local memory.
Reflective Memory is unique in its special shared Memory system. The system can make multiple independent computers realize data sharing. At the same time Reflective Memory keeps a independent copy of every connected system in the entire shared network. In addition, extremely high transmission speed and very low transmission latency are all crucial advantages when building real-time systems.(ABACO Systems, n.d.)

In fact, the key feature of "RMS" is that every day the computer has its own local physical memory, and different types of memory can be connected to each other for updates. If take RT-CRM for example.Real-Time and Channel-based Reflective Memory that is based on memory channel, i.e. hardware assisted, virtual connection-based memory to memory transfer of data (Chia Shen, 2001). It represents an approach similar to the one used in this project. Indeed, as we will see later, the reflective memory used is organized in channels and each of them will be reserved for each input or output signal involved in the project.
For the "Iron Bird" of this project, the simulation of the whole software part needs real-time synchronization. In order to ensure the integrity of the entire flight test, and all relevant data can be generated within a time period. A reflective memory-based algorithm is necessary, especially when dealing with the handshake between the main scheduler (ETS) and all software modules.

The master scheduler receives commands from the test rig operator and, exploiting the Iron Bird optical ring, it pilots the following states:

- *Initialization*: executed once when 'Start' command is received and is reserved for the modules initialization and configuration

- *Execution*: executed iteratively until 'Stop' command is received from the rig operator. It is composed of the following sub-states:
    - Write cycle: software modules copy their output data into the optical ring, with the purpose of sharing them for the next execution cycle. In this phase ETS takes the needed data from the physical interfaces and copy them into reflective memory area
    - RT cycle: the software modules execute their specific algorithm and tasks
    - Read cycle: the software modules retrieve from reflective memory area the input data necessary for their execution. ETS converts those data into the physical interfaces.
- *Stop*: executed once when 'Stop' command is received to stop the current test and software modules execution.

For handshake communication between scheduler and nodes, two reflective memory events are used:

- Interrupt#1, sent by the scheduler to all the nodes to command the current state
- Interrupt#2, sent by every node as acknowledge

An enumerative value is used as parameter of the previous events:

- 1000: Inizialization
- 1001: Read
- 1002: Write
- 1003: Stop
- 2000: Acknowledge

Before proceeding to the next stage, the master scheduler must receive all validation information to ensure that all modules have completed the corresponding computations.

Chapter 3

Mapping Flight Mechanics Simulation Computer (FMSC)

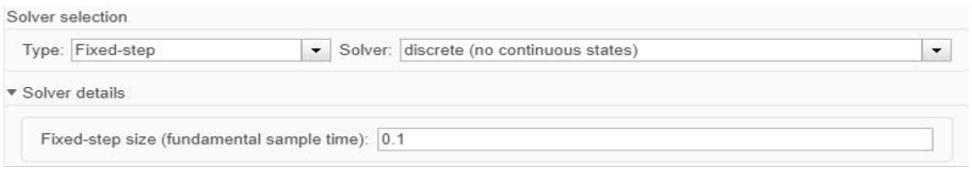
In order to carry out the model's input and output mapping.First, FMSC's Simulink model will be converted to a 'DLL' file through code generation by Matlab.Then the dll file will be deployed inside Veristand System Explorer and according to the number of inputs and outputs, set the corresponding number of ports.Finally, the ports are connected to each other according to the corresponding relation of RFM files.

3.1 Code generation

To import the model in NI VeriStand we need a dll (dynamic linking library) file. This file describes the model in a way VeriStand can understand. It is obtained as a result of code generation. To perform code generation we have to change some model settings. (National Instruments, n.d.).

In particular we set:

– In solver section type=fixed-step, solver=discrete and fixed-step size selected according to the time requirements



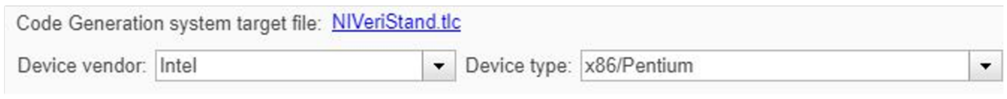
Solver selection

Type: Fixed-step Solver: discrete (no continuous states)

Solver details

Fixed-step size (fundamental sample time): 0.1

- In hardware implementation device vendor= Intel and device type=x86/Pentium

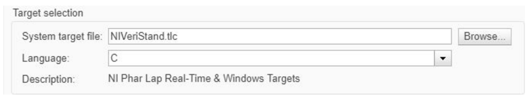


Code Generation system target file: NIVeriStand.tlc

Device vendor: Intel

Device type: x86/Pentium

- In code generation system target file=NIVeriStand.tlc



Target selection

System target file: NIVeriStand.tlc

Language: C

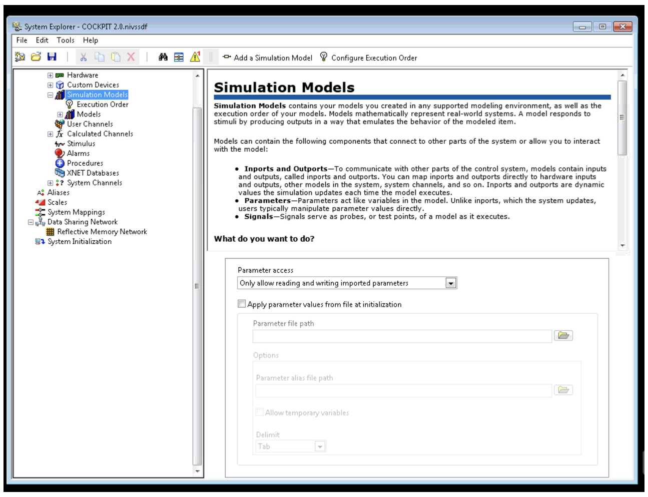
Description: NI Phar Lap Real-Time & Windows Targets

Now the model can be built and if there aren't errors,the dll will be obtained and which is needed to import the model in VeriStand.

3.2 Deployment inside Veristand SystemExplore

NI Veristand allow to manage several independent models and compile and run several models from different partners (this thesis is about FMSC) by using the custom device to manage reflective memory.

- Copy the dll file in the "Model" folder of the NIVS poject. Open the .nivssdf file and navigate in the tree.



System Explorer - COCKPIT 2.0.nivssdf

Simulation Models

Simulation Models contains your models you created in any supported modeling environment, as well as the execution order of your models. Models mathematically represent real-world systems. A model responds to stimuli by producing outputs in a way that emulates the behavior of the modeled item.

Models can contain the following components that connect to other parts of the system or allow you to interact with the model:

- **Inports and Outports**—To communicate with other parts of the control system, models contain inputs and outputs, called inports and outports. You can map inports and outports directly to hardware inputs and outputs, other models in the system, system channels, and so on. Inports and outports are dynamic values the simulation updates each time the model executes.
- **Parameters**—Parameters act like variables in the model. Unlike inports, which the system updates, users typically manipulate parameter values directly.
- **Signals**—Signals serve as probes, or test points, of a model as it executes.

What do you want to do?

Parameter access: Only allow reading and writing imported parameters

☐ Apply parameter values from file at initialization

Parameter file path

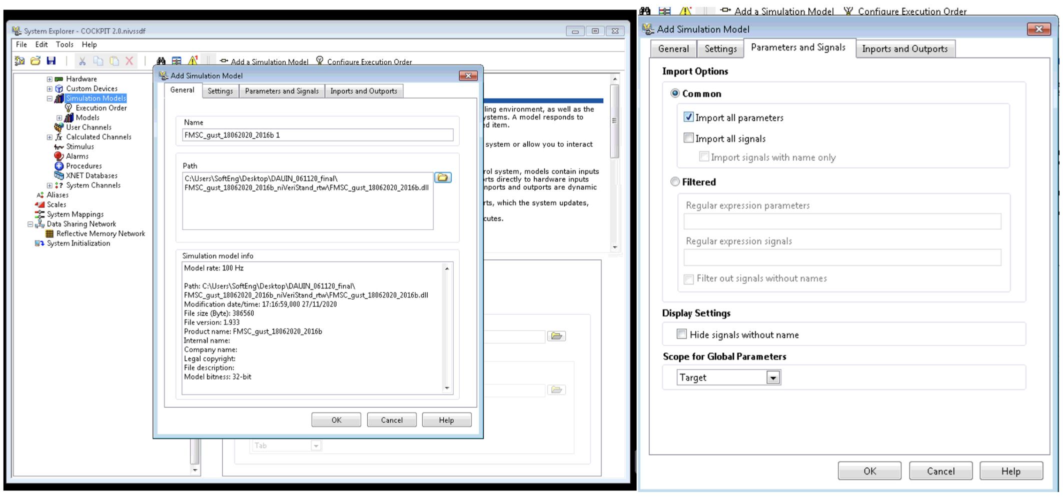
Options

Parameter alias file path

☐ Allow temporary variables

Delimit: Tab

– Add a simulation model import all parameters



Name: FMSC_gust_18062020_2016b 1

Path: C:\Users\SoftEng\Desktop\DAUIN_061120_final\FMSC_gust_18062020_2016b_niVeriStand_rtw\FMSC_gust_18062020_2016b.dll

Simulation model info
Model rate: 100 Hz

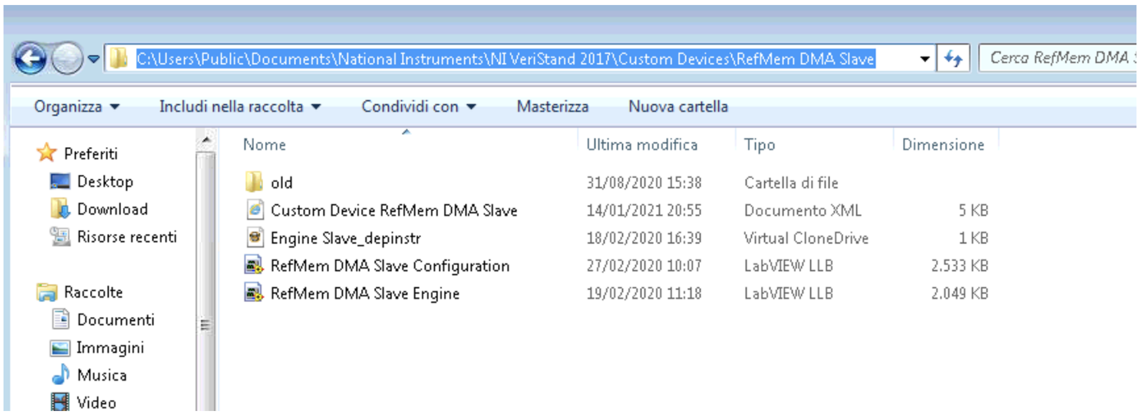
Path: C:\Users\SoftEng\Desktop\DAUIN_061120_final\FMSC_gust_18062020_2016b_niVeriStand_rtw\FMSC_gust_18062020_2016b.dll
Modification date/time: 17:16:59,000 27/11/2020
File size (Byte): 386560
File version: 1.933
Product name: FMSC_gust_18062020_2016b
Internal name:
Company name:
Legal copyright:
File description:
Model bitness: 32-bit

Import Options
[✓ Common]
Import all parameters [✓]
Import all signals [ ]
Import signals with name only [ ]
Filtered [ ]
Regular expression parameters
Regular expression signals
Filter out signals without names [ ]
Display Settings
Hide signals without name [ ]
Scope for Global Parameters: Target

Until now the project can be run and with a complete access to inputs / outputs / parameters of the models from NIVS interface.When the reflective memory custom device will be ready,Hardware / Custom Device can be added and then use System Mappings to connect the inputs / outputs of the models to reflective memory variables.

3.3 Setting ports

According an excel file named <RFM affectation> which is describing the reflective memory allocation .Followed this file there are 2116 input signals and 1443 output signals. In order to engage shared memory custom there are several files have to be extracted in the folder:

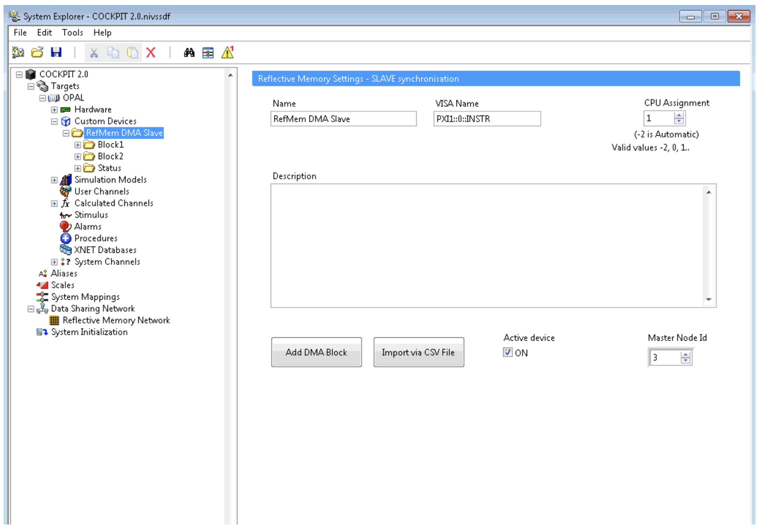


C:\Users\Public\Documents\National Instruments\NI VeriStand 2017\Custom Devices\RefMem DMA Slave

Nome	Ultima modifica	Tipo	Dimensione
old	31/08/2020 15:38	Cartella di file	
Custom Device RefMem DMA Slave	14/01/2021 20:55	Documento XML	5 KB
Engine Slave_depinstr	18/02/2020 16:39	Virtual CloneDrive	1 KB
RefMem DMA Slave Configuration	27/02/2020 10:07	LabVIEW LLB	2.533 KB
RefMem DMA Slave Engine	19/02/2020 11:18	LabVIEW LLB	2.049 KB

At same time those rules should be followed:

- Name the controller as "OPAL"
- Add the custom device "RefMem DMA Slave"
- Add the model
- In the custom device, add 2 blocks as shown below:

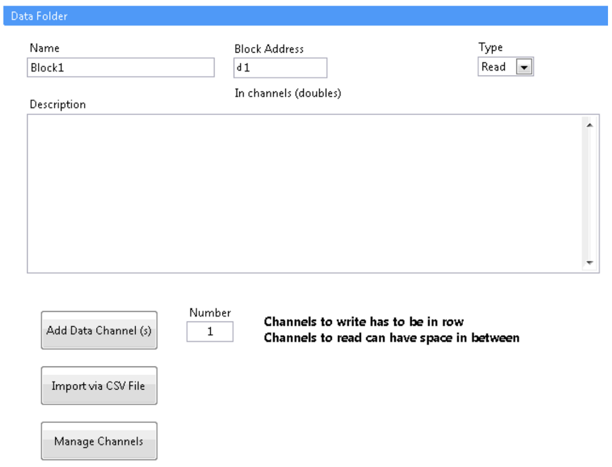


Reflective Memory Settings - SLAVE synchronisation

Name: RefMem DMA Slave
VISA Name: PXI1::0::INSTR
CPU Assignment: 1
(-2 is Automatic)
Valid values -2, 0, 1..
Description:
Active device: ☑ ON
Master Node Id: 3

Block1 is aim to 2116 input signals should have these properties :

- "Read" mode
- Number of data channels : 2116
- Block address = 1


Data Folder

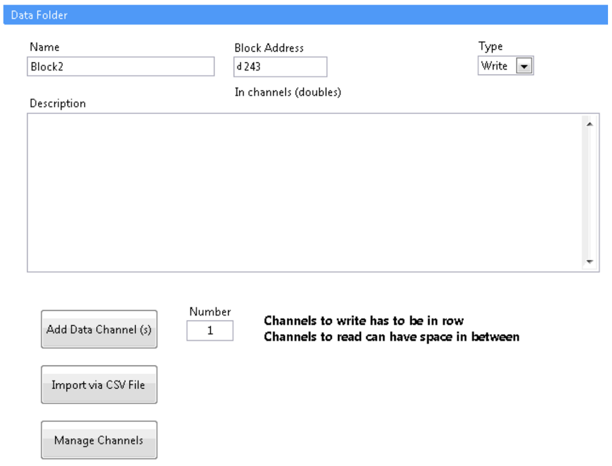
Name: Block1
Block Address: d 1
Type: Read
In channels (doubles)
Description

Number: 1

Channels to write has to be in row
Channels to read can have space in between

Block2 is aim to 1443 output signals should have these properties :

- "write" mode
- Number of data channels : 1443
- Block address = 243


Data Folder

Name: Block2
Block Address: d 243
Type: Write
In channels (doubles)
Description

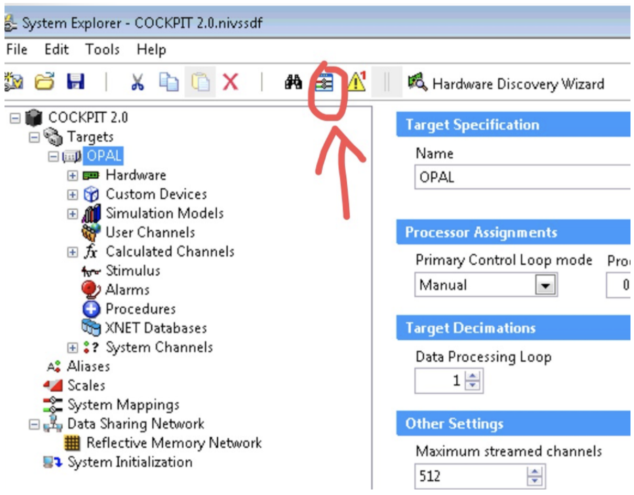
Number: 1

Channels to write has to be in row
Channels to read can have space in between

At this point, all the preparation work of mapping has been completed.

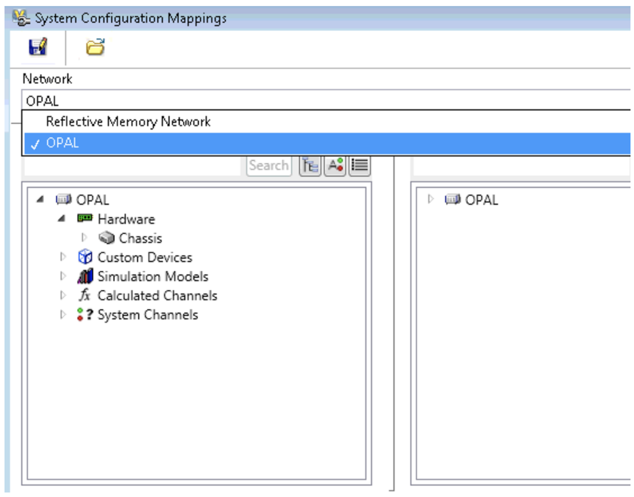
3.4 Mapping set

For the mapping step, first should open the mapping window by clicking the icon focused hereafter :

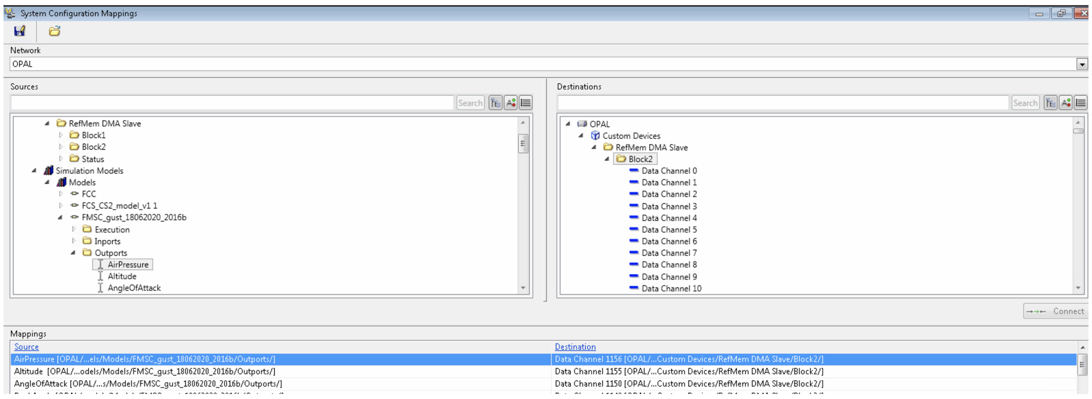


Name: OPAL
Primary Control Loop mode: Manual
Data Processing Loop: 1
Maximum streamed channels: 512

Select the controller in the list at the top to "OPAL"

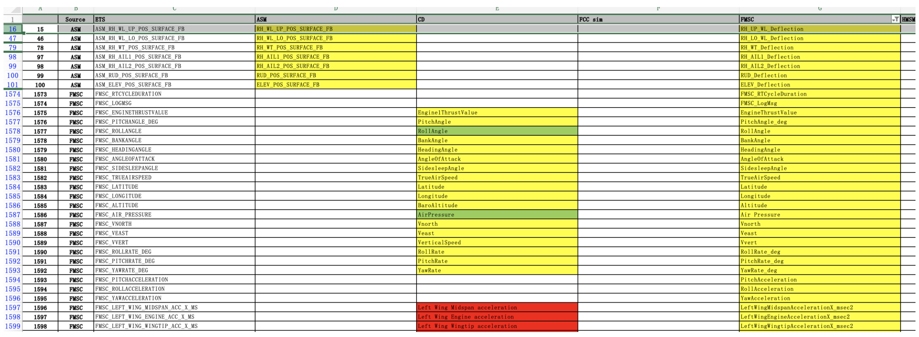


For each variable, select source and destination variables as shown, then click at "Connect" button :



Source	Destination
AirPressure [OPAL/...els/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1156 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
Altitude [OPAL/...odels/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1155 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
AngleOfAttack [OPAL/...s/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1150 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]

All outports of the model should be connected to block 2 of RFM custom device, and all block 1 of RFM custom device should be connected to inports of the model.Addresses of RFM variables, as declared in the RFM affectation file, are calculated with block address (block 1 address = 1, block 2 address = 430) + variable address in the block.

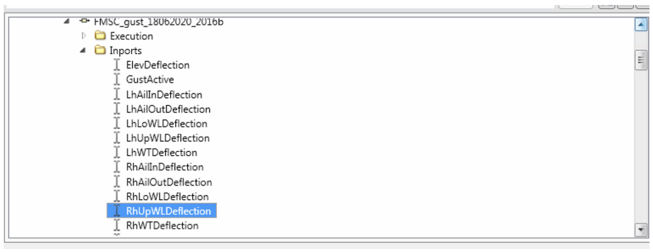


	A	B	C	D	E	F	G	
1		Source	ETS	ASM	CD	FCC sim	FMSC	HMSM
16	15	ASM	ASM_RH_WL_UP_POS_SURFACE_FB	RH_WL_UP_POS_SURFACE_FB			RH_UP_WL_Deflection	
47	46	ASM	ASM_RH_WL_LO_POS_SURFACE_FB	RH_WL_LO_POS_SURFACE_FB			RH_LO_WL_Deflection	
79	78	ASM	ASM_RH_WT_POS_SURFACE_FB	RH_WT_POS_SURFACE_FB			RH_WT_Deflection	
98	97	ASM	ASM_RH_AIL1_POS_SURFACE_FB	RH_AIL1_POS_SURFACE_FB			RH_AIL1_Deflection	
99	98	ASM	ASM_RH_AIL2_POS_SURFACE_FB	RH_AIL2_POS_SURFACE_FB			RH_AIL2_Deflection	
100	99	ASM	ASM_RUD_POS_SURFACE_FB	RUD_POS_SURFACE_FB			RUD_Deflection	
101	100	ASM	ASM_ELEV_POS_SURFACE_FB	ELEV_POS_SURFACE_FB			ELEV_Deflection	
1574	1573	FMSC	FMSC_RTCYCLEDURATION				FMSC_RTCycleDuration	
1575	1574	FMSC	FMSC_LOGMSG				FMSC_LogMsg	
1576	1575	FMSC	FMSC_ENGINETHRUSTVALUE		Engine1ThrustValue		EngineThrustValue	
1577	1576	FMSC	FMSC_PITCHANGLE_DEG		PitchAngle		PitchAngle_deg	
1578	1577	FMSC	FMSC_ROLLANGLE		RollAngle		RollAngle	
1579	1578	FMSC	FMSC_BANKANGLE		BankAngle		BankAngle	
1580	1579	FMSC	FMSC_HEADINGANGLE		HeadingAngle		HeadingAngle	
1581	1580	FMSC	FMSC_ANGLEOFATTACK		AngleOfAttack		AngleOfAttack	
1582	1581	FMSC	FMSC_SIDESLEEPANGLE		SidesleepAngle		SidesleepAngle	
1583	1582	FMSC	FMSC_TRUEAIRSPEED		TrueAirSpeed		TrueAirSpeed	
1584	1583	FMSC	FMSC_LATITUDE		Latitude		Latitude	
1585	1584	FMSC	FMSC_LONGITUDE		Longitude		Longitude	
1586	1585	FMSC	FMSC_ALTITUDE		BaroAltitude		Altitude	
1587	1586	FMSC	FMSC_AIR_PRESSURE		AirPressure		Air Pressure	
1588	1587	FMSC	FMSC_VNORTH		Vnorth		Vnorth	
1589	1588	FMSC	FMSC_VEAST		Veast		Veast	
1590	1589	FMSC	FMSC_VVERT		VerticalSpeed		Vvert	
1591	1590	FMSC	FMSC_ROLLRATE_DEG		RollRate		RollRate_deg	
1592	1591	FMSC	FMSC_PITCHRATE_DEG		PitchRate		PitchRate_deg	
1593	1592	FMSC	FMSC_YAWRATE_DEG		YawRate		YawRate_deg	
1594	1593	FMSC	FMSC_PITCHACCELERATION				PitchAcceleration	
1595	1594	FMSC	FMSC_ROLLACCELERATION				RollAcceleration	
1596	1595	FMSC	FMSC_YAWACCELERATION				YawAcceleration	
1597	1596	FMSC	FMSC_LEFT_WING_MIDSPAN_ACC_X_MS		Left Wing Midspan acceleration		LeftWingMidspanAccelerationX_msec2	
1598	1597	FMSC	FMSC_LEFT_WING_ENGINE_ACC_X_MS		Left Wing Engine acceleration		LeftWingEngineAccelerationX_msec2	
1599	1598	FMSC	FMSC_LEFT_WING_WINGTIP_ACC_X_MS		Left Wing Wingtip acceleration		LeftWingWingtipAccelerationX_msec2	

For example

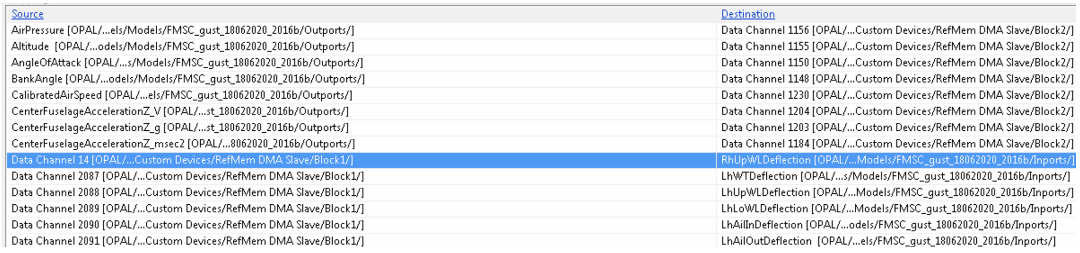
– Input

The address of the 'RH UP WL Deflection' is determined to be 15 based of the RFM document and the signal is determined from the Configuration as the input signal.



FMSC_gust_18062020_2016b
- Execution
- Inports
  - ElevDeflection
  - GustActive
  - LhAilInDeflection
  - LhAilOutDeflection
  - LhLoWLDeflection
  - LhUpWLDeflection
  - LhWTDeflection
  - RhAilInDeflection
  - RhAilOutDeflection
  - RhLoWLDeflection
  - RhUpWLDeflection
  - RhWTDeflection

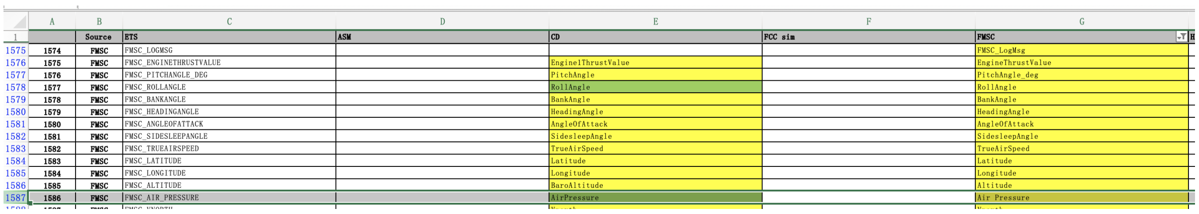
It is therefore certain that it should interconnect with channel 14 in BLOCK1 calculated by formula input signals should be minus 1 .



Source	Destination
AirPressure [OPAL/...els/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1156 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
Altitude [OPAL/...odels/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1155 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
AngleOfAttack [OPAL/...s/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1150 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
BankAngle [OPAL/...odels/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1148 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
CalibratedAirSpeed [OPAL/...els/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1230 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
CenterFuselageAccelerationZ_V [OPAL/...st_18062020_2016b/Outports/]	Data Channel 1204 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
CenterFuselageAccelerationZ_g [OPAL/...st_18062020_2016b/Outports/]	Data Channel 1203 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
CenterFuselageAccelerationZ_msec2 [OPAL/...8062020_2016b/Outports/]	Data Channel 1184 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
Data Channel 14 [OPAL/...Custom Devices/RefMem DMA Slave/Block1/]	RhUpWLDeflection [OPAL/...Models/FMSC_gust_18062020_2016b/Inports/]
Data Channel 2087 [OPAL/...Custom Devices/RefMem DMA Slave/Block1/]	LhWTDeflection [OPAL/...s/Models/FMSC_gust_18062020_2016b/Inports/]
Data Channel 2088 [OPAL/...Custom Devices/RefMem DMA Slave/Block1/]	LhUpWLDeflection [OPAL/...Models/FMSC_gust_18062020_2016b/Inports/]
Data Channel 2089 [OPAL/...Custom Devices/RefMem DMA Slave/Block1/]	LhLoWLDeflection [OPAL/...Models/FMSC_gust_18062020_2016b/Inports/]
Data Channel 2090 [OPAL/...Custom Devices/RefMem DMA Slave/Block1/]	LhAilInDeflection [OPAL/...odels/FMSC_gust_18062020_2016b/Inports/]
Data Channel 2091 [OPAL/...Custom Devices/RefMem DMA Slave/Block1/]	LhAilOutDeflection [OPAL/...els/FMSC_gust_18062020_2016b/Inports/]

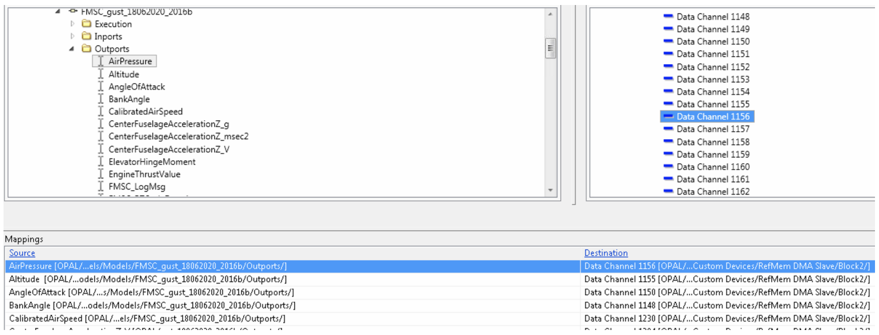
– Output:

The address of the 'AirPressure' is determined to be 1586 based of the RFM document and the signal is determined from the Configuration as the Output signal.



	A	B	C	D	E	F	G
1		Source	ETS	ASM	CD	FCC sim	FMSC
1575	1574	FMSC	FMSC_LOGMSG				FMSC_LogMsg
1576	1575	FMSC	FMSC_ENGINETHRUSTVALUE		Engine1ThrustValue		EngineThrustValue
1577	1576	FMSC	FMSC_PITCHANGLE_DEG		PitchAngle		PitchAngle_deg
1578	1577	FMSC	FMSC_ROLLANGLE		RollAngle		RollAngle
1579	1578	FMSC	FMSC_BANKANGLE		BankAngle		BankAngle
1580	1579	FMSC	FMSC_HEADINGANGLE		HeadingAngle		HeadingAngle
1581	1580	FMSC	FMSC_ANGLEOFATTACK		AngleOfAttack		AngleOfAttack
1582	1581	FMSC	FMSC_SIDESLEEPANGLE		SidesleepAngle		SidesleepAngle
1583	1582	FMSC	FMSC_TRUEAIRSPEED		TrueAirSpeed		TrueAirSpeed
1584	1583	FMSC	FMSC_LATITUDE		Latitude		Latitude
1585	1584	FMSC	FMSC_LONGITUDE		Longitude		Longitude
1586	1585	FMSC	FMSC_ALTITUDE		BaroAltitude		Altitude
1587	1586	FMSC	FMSC_AIR_PRESSURE		AirPressure		Air Pressure

It is therefore certain that it should interconnect with channel 1156 in BLOCK2 calculated by formula output signals should be minus 430.



Mappings

Source	Destination
AirPressure [OPAL/...els/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1156 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
Altitude [OPAL/...odels/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1155 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
AngleOfAttack [OPAL/...s/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1150 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
BankAngle [OPAL/...odels/Models/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1148 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]
CalibratedAirSpeed [OPAL/...els/FMSC_gust_18062020_2016b/Outports/]	Data Channel 1230 [OPAL/...Custom Devices/RefMem DMA Slave/Block2/]

Chapter 4

Conclusions

This thesis is helpful to show the advantages of following a model-based approach in developing applications like control systems or, as in our case, in aerospace or in automotive fields.

In particular, we surely think about automated code generation and rapid control prototyping, referring with this to model simulation inside NI VeriStand.

The automated code generation permitted the team involved in the project to almost forget about the code. We only cared about the correct file extension that we needed. In other words, we checked only if a dll file were correctly generated.

Rapid control prototyping allows to analyze the performances and the integration of the various parts considering also the real time aspect in the simulation.

Considering the real-time aspect allowed us to shorten the testing time, and gave us the possibility to understand how the code for the three models we dealt with, behaves in a situation very similar to the final one, without having a physical prototype of the whole Iron Bird.

Our tests showed that the modules composing CERTIA's test bench are able to correctly communicate with each other. It is, then, possible to assert that Politecnico carried out his task in a proper way.

Bibliography

ABACO Systems. (n.d.). *Reflective memory network.* Retrieved from `https://www.abaco.com/download/real-time-networking-reflective-memory`

Chia Shen, I. M. (2001). Rt-crm: Real-time channel-based reflective memory. *Mitsubishi Electric Research Laboratories*.

Garrison, P. (2018). Hinge moments explained. *Flying*.

National Instruments. (n.d.). *What is veristand?* Retrieved from `https://www.ni.com/it-it/shop/data-acquisition-and-control/application-software-for-data-acquisition-and-control-category/what-is-veristand.html`

Wikipedia. (n.d.). *Simulink — Wikipedia, the free encyclopedia.* Retrieved from `https://en.wikipedia.org/wiki/Simulink`