Proceedings of the EuroGNC 2013, 2nd CEAS Specialist Conference
on Guidance, Navigation & Control, Delft University of Technology,
Delft, The Netherlands, April 10-12, 2013

Hardware-in-the-loop Flight Simulator - An Essential Part in the Development Process for the Automatic Flight Control System of a Utility Aircraft

André Kaden, Bernd Boche and Robert Luckner

Abstract. An automatic flight control system (AFCS) is designed for the utility aircraft STEMME S15, a high-performance motor glider. The AFCS shall automatically control the aircraft with high precision during surveillance, reconnaissance and measurement flights. To test the AFCS a ground test facility in form of a hardware-in-the-loop (HIL) simulator was built. The correct integration of the AFCS and its flight control functions into the aircraft are verified by this test system. HIL simulation is part of a cost-effective development process for safety-critical systems that will be established as part of this project. This paper gives an overview of the development process and describes the concept, the functional principle and the design of the HIL simulator. A comparison of flight test and simulation results of the first automatic landing of the S15 is shown, as example for the HIL simulator validation.

1 Introduction

The automatic flight control system (AFCS) is an integral part of the utility aircraft STEMME S15. The system is complex, highly-integrated and it has safety critical functions. It consists of software and hardware. The development process of such systems is subject to strict rules that are described in SAE ARP 4754 [1]. The potential of making mistakes during the development of complex systems is high. In order to assure the correctness of such a highly-integrated system, a structured system development process has to be established. That includes rigorous, systematic, and repeatable testing. The objective of *Aerospace Systems Engineering* is to develop and optimize the appropriate design methods in order to realize an airworthy product that meets all requirements and can be certified by the authorities. In order to facilitate the approval process, policies and standards for sys-

André Kaden
Berlin Institute of Technology, 10587 Berlin, Germany, e-mail: andre.kaden@ilr.tu-berlin.de

Bernd Boche
Berlin Institute of Technology, 10587 Berlin, Germany, e-mail: bernd.boche@ilr.tu-berlin.de

Robert Luckner
Berlin Institute of Technology, 10587 Berlin, Germany, e-mail: robert.luckner@tu-berlin.de

tem development in aviation exist. Guidelines for the development of software are described in RTCA DO-178B [2].

An essential element of the development process is the functional test. It is carried out on various test rigs on component, system and aircraft level. It is important to use efficient development methods to achieve the goal of the overall system validation and certification that is affordable. This is crucial especially for small and midsize enterprises. One possibility is the choice of appropriate test benches. It is important to ensure that tests can be performed cost and time efficient without reducing their quality. This report describes a ground test facility, called hardware-in-the-loop (HIL) flight simulator that replaces an *Iron Bird* by using the prototype aircraft in conjunction with a flight simulator.

2 Research Project LAPAZ

The acronym LAPAZ stands for Air Utility Platform for General Civil Aviation, *Luft-Arbeits-Plattform für die Allgemeine Zivilluftfahrt* in German. The LAPAZ project is funded by the Federal Ministry of Economics and Technology (BMWi) in the National Aerospace Research Program LUFO IV. Project partners are the STEMME AG as the coordinator, University of Stuttgart's Institute of Aircraft Systems (ILS), and TU Berlin's Department of Flight Mechanics, Flight Control and Aeroelasticity (FMRA). The objective of the research project is to develop and demonstrate a reliable and fault-tolerant automatic flight control system for a utility aircraft. ILS is responsible for the development of the fault-tolerant platform for the flight control system, including all redundancy mechanisms. FMRA develops the flight control laws (FCL), the flight mechanical simulation model, the human-machine interface as well as the FCL development process. For the planned EASA CS23 certification as utility aircraft, a specially designed development process for complex, safety-critical systems will be established. The software development is based on it. The following Section gives an overview of the development process with focus on the part that is related to develop the flight control functions of the AFCS.

2.1 Development Process

The overall LAPAZ development process follows the V-Model that is shown in Fig. 1. Initially all functional and non-functional requirements for the automatic flight control system are defined in the top-level specification. They are gradually refined top-down from aircraft level via system and assembly level to the hardware and software requirements on component level. After encoding and production, the gradual integration and verification (bottom-up) follows. Each process step ends with verification tests. The final step includes validation tests. The ulti-

mate validation is the acceptance by the market and the customer.

It should be noted that the V-model is an idealization. In reality, several iterations occur that are not shown here. For example, if an error in the system specification is detected during validation tests on aircraft level, a new iteration of the V-process will be initialized starting with updating the system specification.

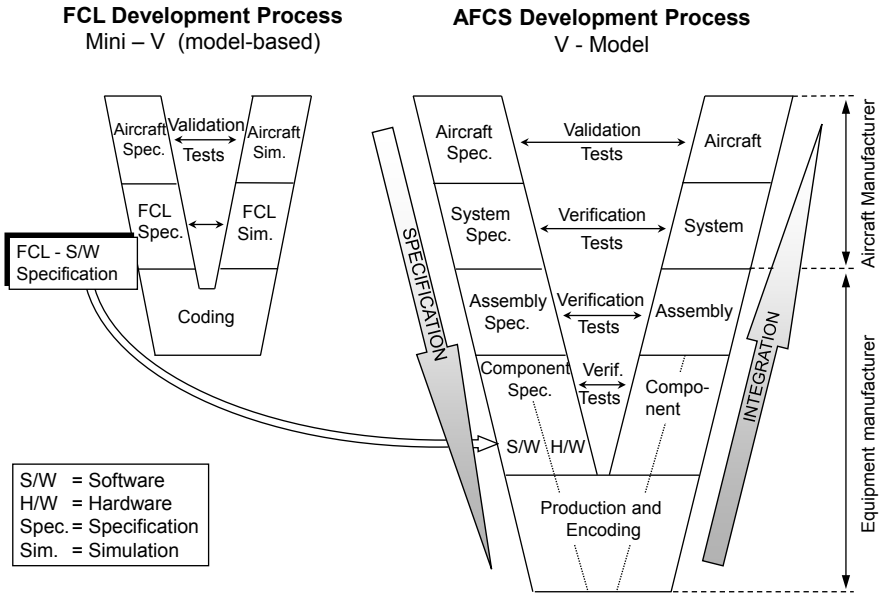

Fig. 1 System development process according to the V-model [3].

As LAPAZ is a research project, the AFCS development proceeds incrementally. That means not all functions are created simultaneously. Instead, starting from a basic functionality, new features are added gradually. Each of them is developed in accordance to the V-model including the validation by flight tests. At the beginning, the sequence, in which the partial functions shall be incrementally developed, has to be defined. This is achieved by continuously considering the interdependencies between the functions, the implementation risk, and the importance for the final product. The advantage of this procedure in comparison to the development of the entire functionality in one shot is the early attainment of feedback from flight tests to the development process. Thereby gradual improvements are possible, so that the existing functionalities are subject to a maturation process. The incremental approach leads to continuously growing software architecture. In order to avoid sub-optimal structures, the basic system architecture has to be fully defined in the very beginning. Any restructuring would be extremely costly and has to be avoided.

The development of flight control laws (FCL) is based on a flight mechanical simulation model of the aircraft. The software is specified by means of Simulink® and Stateflow® that are toolboxes of the software package MATLAB®. Source

code of the FCL is generated through the use of the Real-Time Workshop® Embedded Coder™ (RTW-EC). After compilation, an executable application is created that can be integrated on the target system. The integration work is done gradually from subsystem to the overall system.

The entire integration process is accompanied by verification and validation testing to demonstrate the functionality according to the specifications (verification), as well as the correctness of the requirements (validation). First model-based tests are carried out during FCL design using linear and non linear flight simulations, see Fig. 2.

Fig. 2 Work station and Automatic Flight Control Panel (AFCP) for flight control law development.

When the design is frozen, extensive automatic offline tests are performed to check all functional requirements (software-in-the-loop (SIL) simulations), followed by real-time simulator tests with pilot and SIL. The real-time tests are performed on TU Berlin's research simulator SEPHIR (*Simulator for Educational Projects and Highly Innovative Research*). After C code generation using RTW-EC and compilation, it is verified that the code will correctly run on the target system. This is achieved by using the so-called host simulator that is an emulation of the target system. The host simulator has been provided by the project partner ILS who implements the FCL code on the target system. Test vectors are specified and used to compare the correct implementation of the software. They comprise the input vector that is generated and used as input for the FCLs, and the corresponding output vector that is computed by the FCL Simulink code in the development environment. If the output test vector can be identically reproduced on the host simulator, the FCL software is delivered to ILS for implementation on the target system. ILS proves the correct implementation of the FCL software on the target system, by verifying that the embedded FCL software behaves as in the development environment. The testing of the software on the target system is a fundamental requirement of the Directive RTCA DO-178B [2] that requires a validated and

representative test environment. In Section 6.4.1 *Test Environment*, this is described as follows: *„An excellent test environment includes the software loaded into the target computer and tested in a high fidelity simulation of the target computer environment."* [2]

The HIL simulation is the next important step in the development process. It is performed at STEMME and is the first step to verify the correct integration of the AFCS into the aircraft. This verification step is particularly important because not all aircraft requirements can be tested model-based and on system level. With the HIL simulator, the following aspects can be tested:

1. *Hardware/Software Integration Testing* to ensure that the application software (FCL) in the target computer will satisfy the high-level requirements (Subsection 6.4.3 a [2]) in the fully integrated environment,
2. *Software Integration Testing*, to ensure that the application software (FCL) interacts correctly with the AFCS middleware and satisfies the software requirements and software architecture (Subsection 6.4.3 b, [2]) in the fully integrated environment,
3. Interaction of the pilot with the AFCS via the Automatic Flight Control Panel AFCP (human-machine-interface).

2.2 STEMME S15 Prototype

The utility aircraft S15 is a variant of the motor glider S6. It is designed for commercial applications and shall be certified according to EASA CS-23 [4]. In the LAPAZ project an AFCS was integrated into the S15 prototype [11]. The AFCS signal flow is schematically shown in Fig. 3. A detailed description can be found in [5], [6], [7], and [11]. Details for controller development are given in [8].

6

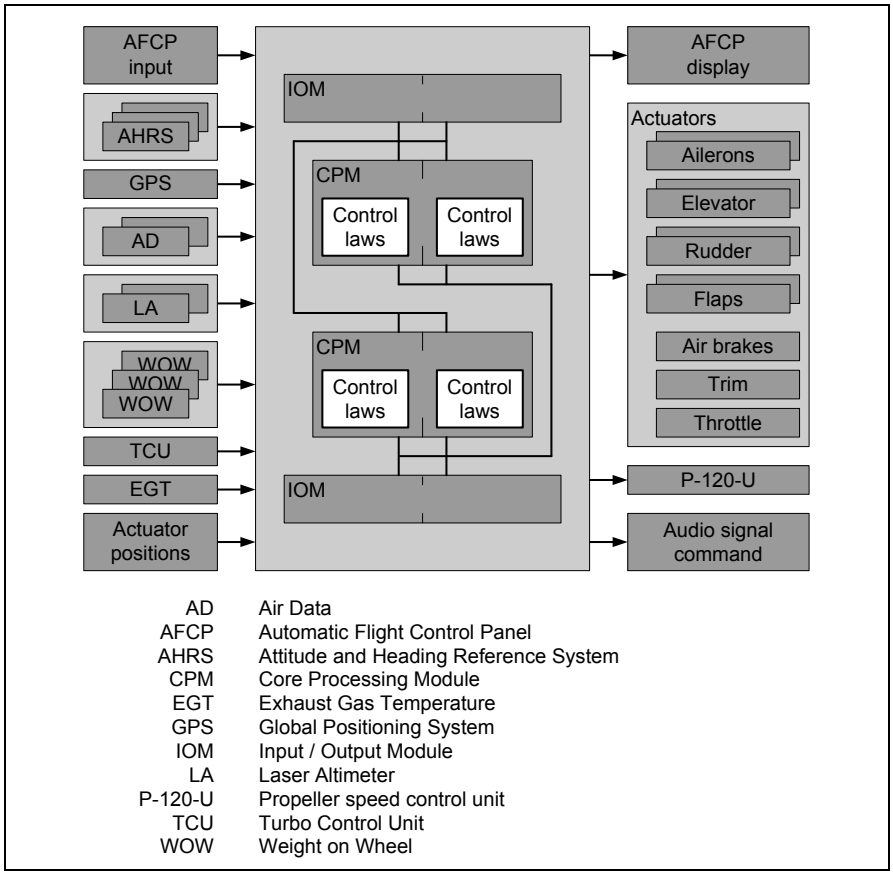

Fig. 3 Overview of automatic flight control system AFCS

3 Hardware-in-the-loop Simulator

This Section describes the concept, the design and the functionality of the HIL simulator. According to the definition of the LAPAZ development process, the HIL simulator shall be used in different applications. For those applications, requirements have been formulated, that are the basis for the design concept. One of the most important requirements is that the HIL simulator shall replace the *Iron Bird* for ground tests on aircraft level. This approach meets demands for time and cost efficiency, it also minimizes possible sources of error as it uses original hardware instead of simulations. The concept is to replace the *Iron Bird* by the prototype of the STEMME S15 as part of the HIL simulator.

3.1 Design Concept

The main task of the HIL simulator is the functional ground test of the AFCS hardware and software, in order to verify the correct function of the flight control laws in the integrated state. Additionally, there are further applications. The flight test program can be safely examined before flight. Its feasibility can be assessed by the test pilot. Abnormal situations can be simulated to test and practice procedures where the test pilot has to take over manual control and to prove that they are safe. The replay of recorded flight test data allows post-flight analyses. The reproducibility of experiments allows the comparison of different controller versions. Furthermore, the HIL simulator serves as a demonstrator for the automatic flight control system at air shows or in front of the customers. The following requirement list is derived from these applications:

- inclusion of as many as possible original aircraft and AFCS components,
- generation of the required sensor and engine data on ground by real-time simulation,
- measuring the control surface deflections to drive the S15 flight simulation,
- processing and transmitting simulated sensor data to the AFCS,
- mobility of the test system for transport,
- robustness of the test facility for use on rugged test sites,
- presentation of the visual scenery on a projection system and the cockpit instruments on LCD screens.

This requirement list is the basis for the design concept of the ground test system. Figure 4 schematically shows the HIL simulator. It consists of the aircraft (STEMME S15 prototype) and a simulation unit.

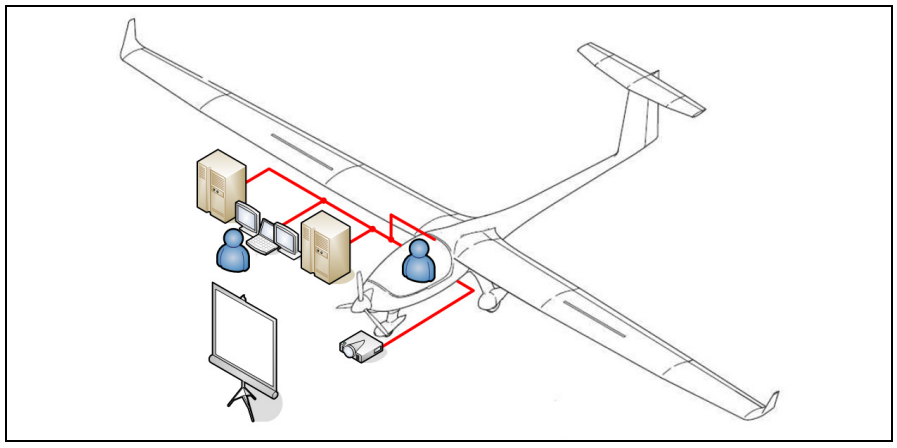

Fig. 4 Concept for establishing the HIL simulator

The simulation unit is mobile and ruggedized. All hardware is integrated into two wheeled transport cases that are connected to the S15 for testing. The workplace of the test pilot is in the cockpit. From there, he controls the autopilot with

the Automatic Flight Control Panel (AFCP) that is installed in the front panel. He monitors the aircraft motion on simulated instruments. Even manual flight control by using the control elements of the S15 is possible. A visual system that is installed in front of the aircraft, projects the outside view on a screen.

3.2 Functional Principle

The HIL simulator simulates the STEMME S15 flight dynamics throughout the entire flight envelope, i.e. in air and ground operations and allows flying in manual mode and with the AFCS. In the following the functional principle of the simulator is described, see. Fig. 5.

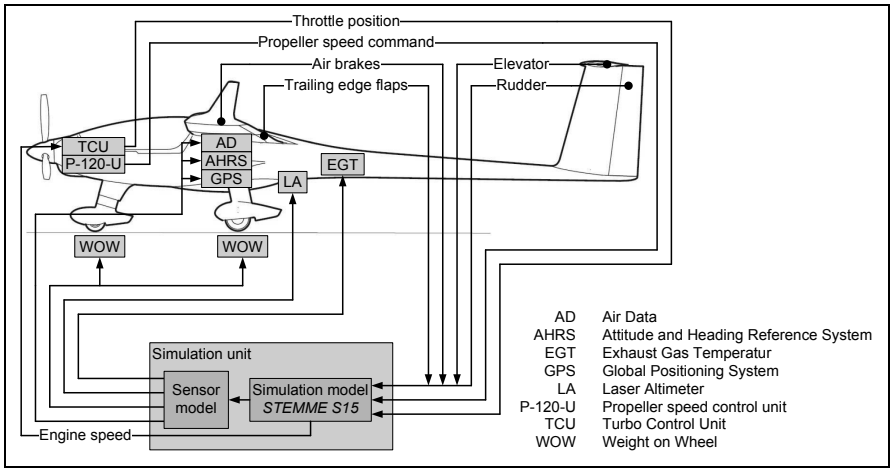

Fig. 5 Functional principle of the HIL simulator

Due to missing aircraft motion all AFCS sensors are inoperative. This applies to the air data systems (AD), the inertial platforms (AHRS), the satellite navigation system (GPS), the laser altimeter (LA), and the ground contact sensors (WOW). As the aerodynamic loads on the linkage and the actuators are missing, the deformation of the mechanical linkage between actuator and control surface is not exactly the same as in flight. For safety, environmental and economic reasons, the engine is not running during HIL simulator tests. Therefore sensor values for the propeller speed and exhaust gas temperature (EGT) are missing as well.

A high-fidelity flight simulation generates the missing sensor signals that are required as input variables for a correct operation of the AFCS. The simulation runs on the simulation unit, see Fig. 5. It includes a non-linear flight mechanical model of the aircraft STEMME S15. The model consists of a six degree of freedom simulation of the rigid aircraft and the first sixteen structural modes. The modelling of structural vibrations is important for testing of highly dynamic AFCS functions such as gust load alleviation. In addition, it includes sensor models, that consist of a dynamic model (dead time, etc.) and an error model (noise, bias, drift,

etc.). The simulated sensor and engine data are sent to the flight control computers as input signals.

Input signals for the flight simulation are the deflections of the control elements. They are measured by potentiometers. The use of the deflections instead of the AFCS actuator command eliminates the necessity to model and simulate the overall transfer function from actuator to control surface (linkage kinematics, elasticity, friction, etc.). Currently, the effect of aerodynamic loads on the linkage and actuators is not simulated, as it would require additional hardware to apply forces and moments to the control surfaces. The effect of aerodynamic loads on the control linkage can be analytically considered in the simulation model. The engine model uses the throttle position (from the Turbo Control Unit, TCU) and the propeller speed (from the propeller speed control unit, P-120-U) as input command.

3.3 Configuration

The HIL simulator has a similar architecture as FMRA's research flight simulator SEPHIR. It uses the same hardware components that are used in other simulators of the department and that have been proven in practice. Thus, existing software modules can be adapted to the HIL simulator by minimal programming effort. The architecture of the simulator is modular and the system is expandable. Mostly standard components are used. Procurement and maintenance costs are low. Figure 6 shows the main hardware components of the simulation unit and their interaction.

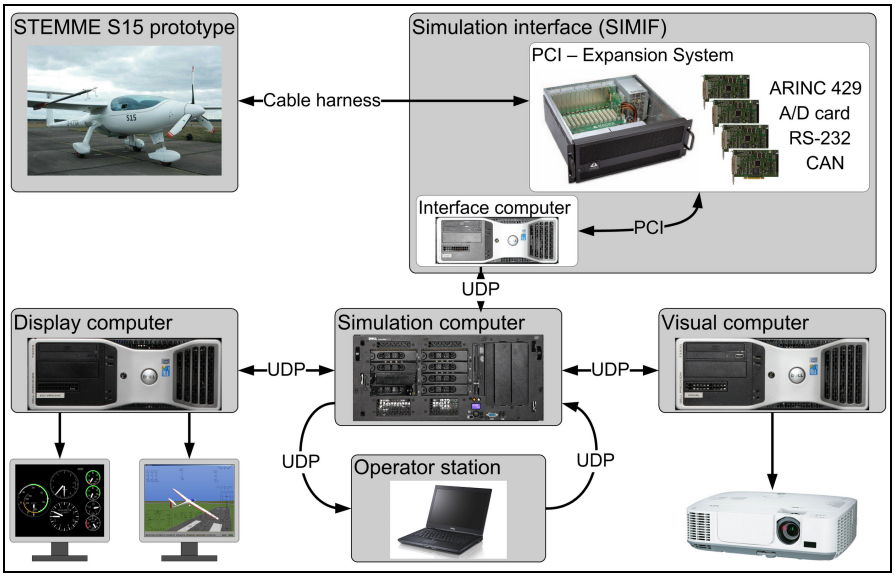

Fig. 6 Overview of the hardware components

The actual simulation process runs on a high performance real-time computer with a UNIX operating system that is optimized for time-critical applications. The simulation runs at 125 Hz.

The simulation interface (SIMIF) enables data transfer between the S15 and the simulation computer. It consists of a conventional personal computer (PC) and a PCI[4] expansion system for increasing the number of PCI cards. The interface cards include the transmission standards ARINC 429, RS-232 and CAN and allow analog/digital (A/D) conversion. The outputs of the cards are connected via a cable harness to the input/output modules of the computers of the AFCS. The process sequence of the data transfer is performed in the following order:

- receive data from the S15 prototype (e.g. control surface deflections),
- send data via User Datagram Protocol (UDP) to the simulation,
- receive data via UDP from the simulation, and
- send data to the S15 prototype.

These four steps form a cycle process with a frequency of 125 Hz. Latency resulting from this simulation process is discussed in Section 4.1. All communication between the client computers and the simulation unit is realized via UDP. The simulation is controlled by a notebook. Various simulation parameters can be modified in real time, for example atmospheric conditions or sensor offsets. Simulated aircraft position and attitude are processed by a visual system to present the outside view to the pilot. The image generator software PHILOSIM is a product of the company Philotech. Aircraft instruments and a view of an outside observer on the aircraft are generated by an additional PC and are indicated on LCD monitors. Figure 7 shows the HIL simulator connected to the STEMME S15 during a functional test of the AFCS.

Fig. 7 STEMME S15 connected to the HIL simulator

[4] PCI (Peripheral Component Interconnect) is a computer bus for integrating hardware devices into a computer.

3.4 Simulation Environment

To develop the FCL and to perform the subsequent tests (PC, SIL and HIL simulation) a highly accurate flight mechanical simulation model of the S15 has been developed. It is implemented in MATLAB and Simulink, see [9]. Using the model in the HIL simulator, the sub-models for actuators, linkages and AFCS mechanisms (consolidation, degradation) are not required as they are replaced by original hardware. The following effects are simulated:

- six degrees of freedom rigid body motion coupled with aeroelastic degrees of freedom,
- two-point aerodynamic (wing and tailplane) with stall and ground effect, that has been identified and modelled using flight test data, see [9],
- propulsion system consisting of engine and propeller,
- wing-mounted elastic boom for the angle of attack vane,
- landing gear,
- terrain,
- position in WGS84 coordinates,
- standard atmosphere and atmospheric turbulence,
- sensor dynamics.

Fig. 8 shows the integration of the S15 flight simulation model into the HIL simulator. It is embedded into a Simulink framework model and a real-time simulation environment that is generated with RTW-EC. The Simulink framework model includes components for data transfer between the simulation and the client computers of the simulation unit. Corresponding UDP ports are responsible for the transmission of data packages. For this reason, additional Simulink S-function blocks are created in the C programming language. The received data is processed and assigned to the inputs of the simulation model. Necessary initial values of the simulation are read from an input text file. Also, the binary model output data are converted for each sensor according to the respective communication protocol (for example ARINC 429) before sending. A block for data recording to an external file is also part of the framework model. The user can start and stop the data recording as required. The data are stored in binary format.

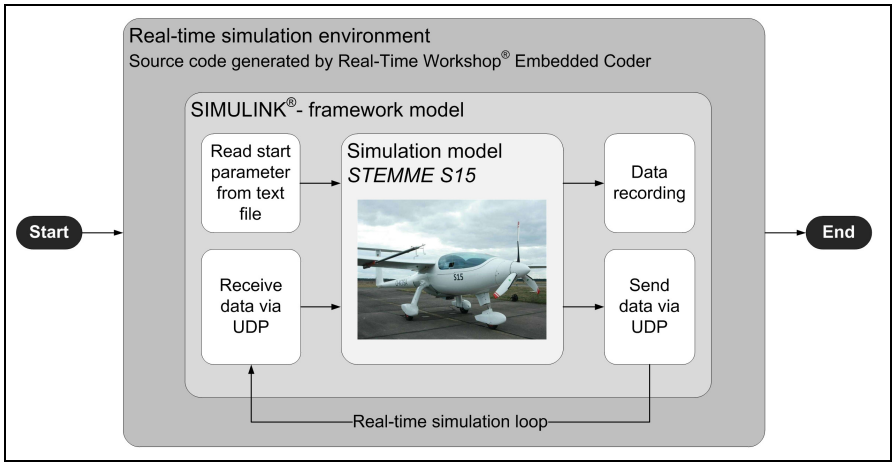

Fig. 8 Architecture of the simulation environment

Both models (framework and S15 flight simulation) are programmed in form of Simulink block diagrams, which has to be converted into C source code by means of RTW-EC in order to accelerate computation. The generated source code has to be supplemented with real-time functionalities that attach the entire simulation process to the real-time clock of the simulation computer using the POSIX application interface [10].

4 Validation

The validation of the HIL simulator has to prove that the simulator correctly represents the behaviour of the S15 prototype throughout the whole flight envelope. Several analyses were done for validation:

1. The flight simulation model was validated during the identification of the parameters for the aerodynamic and the thrust model, see [9]. The flight test data were partly used for model identification and partly for verification of the flight mechanical behaviour.
2. During the integration phase of the HIL simulator the correct implementation of the transmission protocols for the simulated sensor signals was verified.
3. A plausibility check of the simulated sensor values was done using the recorded data.

The next two Subsections deal with the analysis of the HIL simulation latency and the comparison of flight tests and simulation results.

4.1 Latency of the Simulation Unit

The HIL simulator consists of two main components, the S15 aircraft and the simulation unit. The HIL simulation consists of three principal asynchronous processes (P1, P2, and P3):

P1: automatic flight control system and S15 components,
P2: data transfer and
P3: flight simulation.

Process P1 represents the AFCS and all S15 components that belong to it. It comprises sensor signal acquisition, computation of redundancy management and flight control functions, actuator control loops as well as mechanical linkage to control surfaces and measurement of their positions. Process P1 is carried out on the S15 prototype. Process 2 and 3 run on the simulation unit. Part of processes P2 and P3 are the following data handling tasks:

- data handling by interface cards (e.g. A/D conversion),
- data transfer between PCI expansion system and SIMIF computer via PCI,
- data transfer between SIMIF and the simulation computer.

These tasks are handled within microseconds and their latency can be neglected here. Figure 9 shows the time differences due to sensor data generation[5]. The diagram shows all three main processes, without the clock rate of process 1. The time between reception of an input signal and the generation of corresponding output signal is defined as latency. Process P2 runs on the simulation interface computer (SIMIF) and P3 runs on the simulation computer. Both use the same clock rate. As the processes run asynchronously, latencies can vary. For the favourable case a) the latency is at least 8 ms. The cycle shift in the unfavourable case b) leads to a latency of at most 17 ms. As the maximum sensor update rate is 50 Hz (20 ms), the latencies are tolerable and do not significantly affect the behaviour of the controlled system.

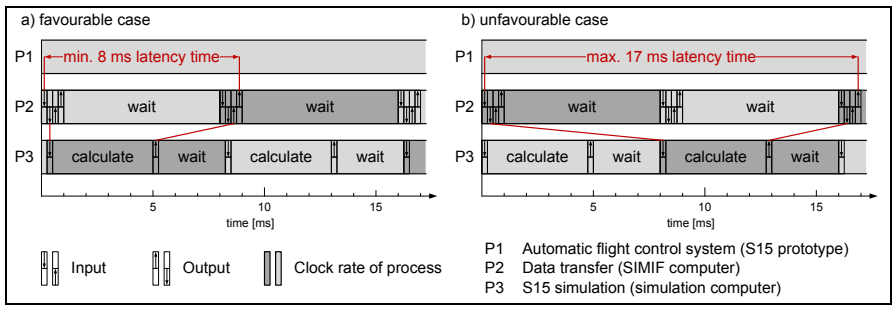

Fig. 9 Latencies of simulation a) favourable case; b) unfavourable case

[5] The simulated time lags as well as the sensor update rates are neglected.

4.2 Validation of HIL Simulation with Flight Test

In this Section the results of the HIL simulation are compared to recorded data from flight tests. The main focus lies on the flight dynamical behaviour of the aircraft that is controlled by the automatic flight control system. For the validation, a manoeuvre has to be defined; the flight conditions (initial state, atmosphere, mass, centre of gravity, etc.) have to be identical (or at least similar). The results are stored with the same data recording system in flight and in HIL tests.

The S15 simulation model has been validated with flight test data from specific identification flight. As a validation example for the closed loop system, the first automatic landing of the S15 prototype on March 22nd 2012 in Neuhardenberg is compared to results from the HIL simulation. During the landing process the aircraft follows a defined three dimensional trajectory with a predefined speed. The results are stored in both cases within the AFCS by the so-called SPY functionality. Amongst others, all sensor values are recorded there. The requirement for identical flight conditions is only partly met for the regarded case. During flight test, the turbulence level was low and the wind velocity was small; in the HIL simulation both were zero.

The observations are restricted to the symmetrical plane. The crosswind component plays a minor role in longitudinal motion. Mass and centre of gravity are similar for simulation and flight test.

Important parameters of the aircraft longitudinal motion over the distance to the runway threshold[6] are shown in Fig. 10 and Fig. 11. In Fig. 10, sensor values for pitch angle Θ, flight path angle γ, height above MSL (*Mean Sea Level*) H_{MSL} and calibrated airspeed (CAS) V_{CAS} are displayed. Figure 11 compares the height above MSL with the height above ground (GND - *Ground*) H_{GND}. Not surprisingly, the sensor values of the HIL simulation are relatively smooth, whereas the flight test data are noisy. This is attributed to atmospheric disturbances. During flight test and simulation, the gust load alleviation function was not active.

During the final approach, the graphs of simulation and flight test match well for all depicted sensor values. Particularly this applies to the vertical flight profile shown in the third diagram of Fig. 10. Comparing flight test and HIL simulation it can be stated that in both cases the system shows basically the same control behaviour.

[6] The runway threshold is located in the origin of the abscissa.

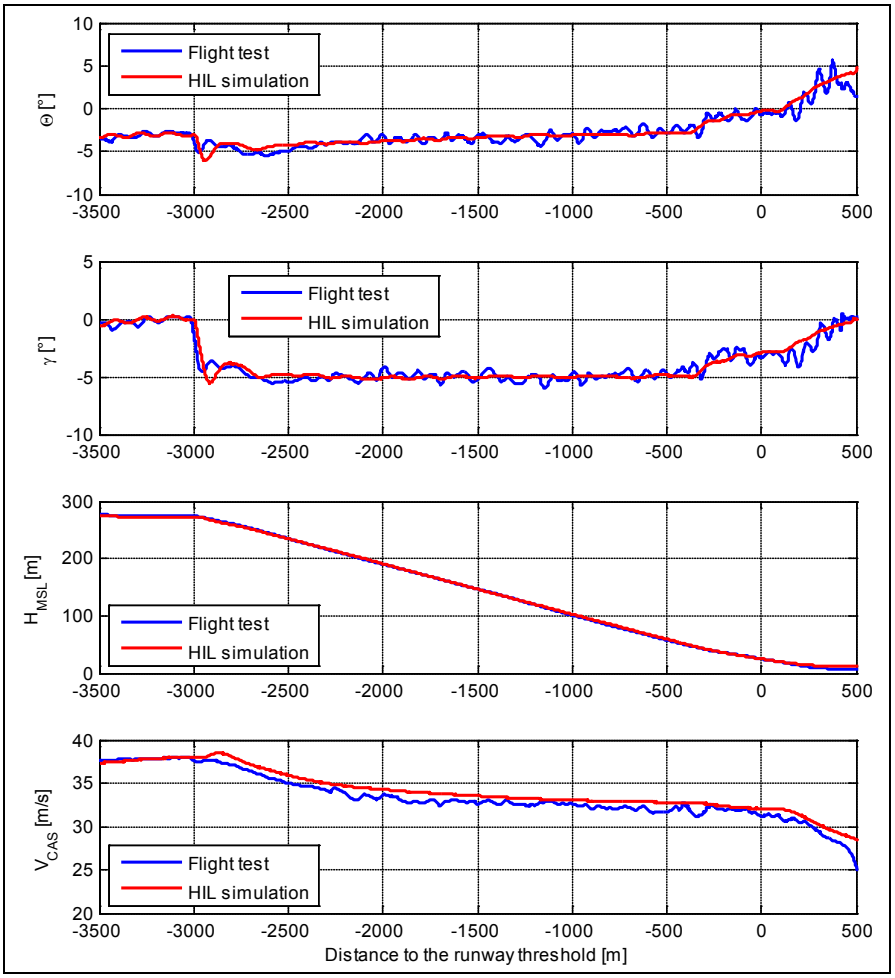

Fig. 10 Pitch angle Θ, flight path angle γ, height H_{MSL} und air speed V_{CAS} over the distance to the runway threshold

The third graph of Fig. 10 shows that after touchdown of the S15 on the runway (about 400 m after the threshold) the simulated height above MSL differs from the actually measured height. Figure 11 illustrates this in detail. The first diagram shows that the deviation is about 3 m. The height above MSL is measured by GPS. According to the AIP (*Aeronautical Information Publication*), the published landing elevation of the runway 08 in Neuhardenberg is 10 m above MSL. Correspondingly, the runway for the simulation is placed at this height. In reality, the GPS sensor of the AFCS measures a runway height of about 6-7 m MSL. This difference is attributed to the measurement accuracy. The height deviation is within the specified accuracy of the GPS that is augmented by EGNOS (*European Geostationary Navigation Overlay Service*).

It is noteworthy that the first automatic landing was accomplished successfully. This is due to the robust design of the controller. Until the S15 reaches the runway threshold, the altitude above MSL is used as control variable. The vertical flight profile specifies that the aircraft crosses the threshold at 15 m above ground. The controller uses the runway elevation that is entered by the pilot (for HIL simulation and flight test the runway elevation was set to 10 m MSL). Due to the error in the measured MSL during the flight test, the S15 crosses the threshold 3 m too high, about 18 m above ground (second diagram Fig. 11). This value is within the specified tolerance, so that the landing could be continued. Having passed the threshold the aircraft is over the runway and the height above ground measured by the laser altimeters is used for control. This assures the adaptation of the flare to the actual runway. From 300 m after the threshold to touch down the trajectory relative to ground from simulation and flight test match again.

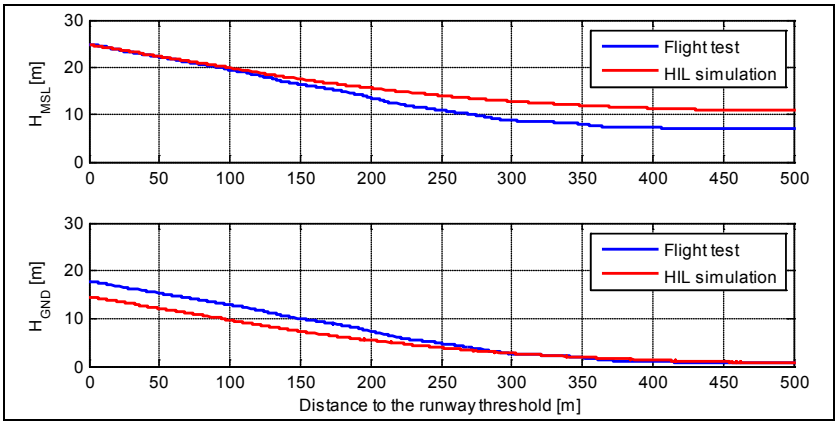

Fig. 11 Height above MSL and GND over the distance to the runway threshold

5 Conclusion and Outlook

As part of the research project LAPAZ, a hardware-in-the-loop (HIL) simulator was build. It has been validated and it is successfully used for functional tests of the STEMME S15 automatic flight control system. The HIL simulator is an integral part of the development process of this safety-critical system and contributes significantly to the success of the research project. Previous to the integration tests on the HIL simulator, the computer platform, its redundancy management and the correct integration of the FCL software are rigorously tested on a test system that is specifically designed for this purpose by the project partner ILS. During the HIL simulator tests the test pilot can sit in the S15 cockpit in his familiar working environment. The tests that are performed before the actual flight tests improve the efficiency of the flight tests and reduce the risk for test pilot and aircraft. Standard and abnormal AFCS behaviour are realistically represented and the corresponding reaction of the pilot can be analysed or trained. So, this concept has proven to be highly valuable. In comparison with an *Iron Bird* both time and money can be

saved, as well as floor space for storage. The HIL simulator can be linked to the aircraft in less than 15 minutes. The modular structure of the simulator facilitates the extension of the system if needed. The comparison of simulation and flight test shows that the HIL simulator realistically reproduces the behaviour of the STEMME S15. Therefore, findings from the HIL simulations can be used in the development of the AFCS.

References

[1] SAE: *Certification Considerations for Highly-Integrated or Complex Aircraft Systems*, ARP 4754, Systems Integration Requirements Task Group, AS-1C, ASD, SAE, Warrendale 1996

[2] RTCA: *Software Considerations in Airborne Systems and Equipment Certification*, DO-178B, RTCA, Inc., Washington, D.C. 1992

[3] Brockhaus, R.; Alles, W.; Luckner, R.: *Flugregelung*, 3. Auflage, ISBN 978-3-642-01442-0, Springer-Verlag, Berlin Heidelberg 2011

[4] EASA: *Certification Specification for Normal, Utility, Aerobatic and Commuter Category Aeroplanes*, CS-23, EASA 2003

[5] Hesse, S.; Reichel, R.; Görke, S.; Dalldorff, L.: *Eine skalierbare Plattform für sicherheitskritische, automatische Flugsteuerungssysteme der allgemeinen Zivilluftfahrt*, Deutscher Luft- und Raumfahrt Kongress 2009

[6] Hesse, S.; Görke, S.: *An Affordable, Fault-Tolerant Automatic Flight Control System for the Utility Aircraft Stemme S15*, CEAS EuroGNC 2011, Conference in Guidance, Navigation and Control in Aerospace, München 2011

[7] Polenz, S.; Cake, F.; Görke, S.; Reichel, R.: *SAFAR eine Fly-by-Wire Steuerung für ein Flugzeug der General Aviation (DIAMOND DA42)*, Deutscher Luft- und Raumfahrt Kongress 2011

[8] Lamp, M; Luckner, R.: *Flight Control Law Development for the Automatic Flight Control System LAPAZ*, CEAS EuroGNC 2011, Conference in Guidance, Navigation and Control in Aerospace, München 2011

[9] Meyer-Brügel, W.; Luckner, R.: *Flight Mechanical Simulation Models for Design and Test of Automatic Flight Control Functions*, CEAS EuroGNC 2011, Conference in Guidance, Navigation and Control in Aerospace, München 2011

[10] Burns, A.; Wellings, A.: *Real-Time Systems and Programming Languages: Ada 95, Real-Time Java and Real-Time C / Posix (Third Edition)*, ISBN: 0-201-72988-1, Addison Wesley Langmain, March 2001

[11] Dalldorff, L.; Luckner, R.; Reichel, R.: *A Full-Authority Automatic Flight Control System for the civil airborne utility platform S15-LAPAZ*, CEAS EuroGNC 2013, Conference in Guidance, Navigation and Control in Aerospace, Delft 2013