Software Considerations for Subscale Flight Testing of Experimental Control Laws

Austin M. Murch[1], David E. Cox[2], and Kevin Cunningham[3]
NASA Langley Research Center, Hampton, VA 23681

The NASA AirSTAR system has been designed to address the challenges associated with safe and efficient subscale flight testing of research control laws in adverse flight conditions. In this paper, software elements of this system are described, with an emphasis on components which allow for rapid prototyping and deployment of aircraft control laws. Through model-based design and automatic coding a common code-base is used for desktop analysis, piloted simulation and real-time flight control. The flight control system provides the ability to rapidly integrate and test multiple research control laws and to emulate component or sensor failures. Integrated integrity monitoring systems provide aircraft structural load protection, isolate the system from control algorithm failures, and monitor the health of telemetry streams. Finally, issues associated with software configuration management and code modularity are briefly discussed.

Nomenclature

α	=	angle of attack, degrees
δ	=	control surface deflection, degrees
δ_e	=	elevator deflection, degrees
ϕ	=	bank angle, degrees
V_C	=	calibrated airspeed, knots
N_Z	=	normal load factor, positive up, g
$\bar{q}$	=	dynamic pressure, pounds per square foot
ADC	=	Analog to Digital Converter
AirSTAR	=	Airborne Subscale Transport Aircraft Research
CONOPS	=	Concept of Operations
COTS	=	Commercial Off The Shelf
FCL	=	Flight Control Law
FCS	=	Flight Control System
FCU	=	Flight Control Unit
HIL	=	Hardware In the Loop
INS	=	Inertial Navigation System
I/O	=	Input/Output
IRAC	=	Integrated Resilient Aircraft Controls
IVHM	=	Integrated Vehicle Health Management
LPS	=	Load Protection System
MOS	=	Mobile Operations Station
NASA	=	National Aeronautics and Space Administration
PWM	=	Pulse-Width Modulation
UDP	=	User Datagram Protocol

[1] Research Engineer, Flight Dynamics Branch, Mail Stop 308, AIAA Member.
[2] Senior Research Engineer, Dynamic Systems and Control Branch, Mail Stop 308.
[3] Senior Research Engineer, Flight Dynamics Branch, Mail Stop 308, AIAA Senior Member.

I. Introduction

RESEARCH is being conducted under the NASA Aviation Safety Program (AvSP) to advance the state of the art in adaptive control technologies. One goal of this research is to help reduce the fatal accident rate of transport airplanes due to loss of control. This research emphasizes the use of adaptive control technologies for recovery from extremely adverse conditions, including those resulting from flight control failures, sensor failures, and airframe damage. The AvSP is using subscale flight testing as a tool in the evaluation of experimental adaptive control laws. This is particularly beneficial for the test and evaluation of control law performance beyond the edge of the normal flight envelope, where the risk of vehicle loss is high due to limited knowledge of nonlinear aerodynamics beyond stall and the potential for high structural loads. Numerous examples of subscale unmanned aerial vehicles (UAVs) being used for research exist in the literature,[1,2,3,4,5,6,7] The primary focus of many of these programs is on guidance and navigation algorithms or fully-autonomous systems. The Airborne Subscale Transport Aircraft Research[8,9,10,11,12] (AirSTAR) system at the NASA Langley Research Center has been designed to provide a flexible research environment with the ability to conduct pilot-in-the-loop testing of control algorithms in adverse flight conditions. The implementation of this system will be described in this paper, with particular emphasis on the software components designed to handle the challenges of testing experimental control algorithms. This includes integration of multiple research control laws, emulation of failures, safeguarding the test aircraft against damage, and integrity monitoring.

II. AirSTAR Overview

AirSTAR is an integrated flight test infrastructure which utilizes remotely piloted, turbine-powered subscale models for flight testing. One particular use of AirSTAR is flight testing research control laws in adverse flight conditions. AirSTAR consists of a remotely piloted subscale test article, the Mobile Operations Station (MOS) (an integrated ground station and control room), and a test range. Under the current AirSTAR Concept of Operations (CONOPS) (Fig. 1) a safety pilot, using a commercial 2.4 GHz radio control transmitter, performs the takeoff and climbs to a specified altitude, where control of the aircraft is transferred to a research pilot through a handoff maneuver. The research pilot executes the flight test plan from a research cockpit located in the MOS, using synthetic vision displays driven with aircraft sensor data. The research pilot uses a ground-based flight control system (FCS) that is connected to the aircraft through an L-band telemetry uplink and S-band telemetry downlink. Once the flight test maneuvers are complete, the safety pilot resumes control of the aircraft and performs the landing. The safety pilot is the pilot-in-command of the aircraft and permits the research pilot to control the aircraft via a switch on the safety pilot's transmitter. When the onboard flight control unit (FCU) receives the appropriate handoff command from the safety pilot, the FCU responds to the research pilot's commands received through the L-band telemetry uplink. The command state of the FCU (*i.e.*, who actually has control of the aircraft) is part of the data on the S-band telemetry downlink.

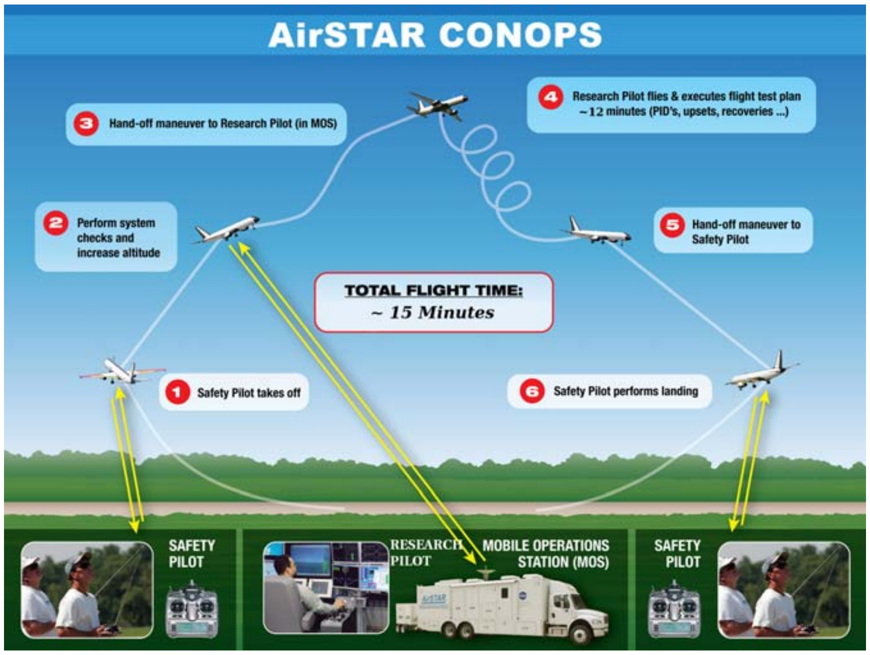
AirSTAR CONOPS
3 Hand-off maneuver to Research Pilot (in MOS)
4 Research Pilot flies & executes flight test plan ~12 minutes (PID's, upsets, recoveries ...)
2 Perform system checks and increase altitude
5 Hand-off maneuver to Safety Pilot
TOTAL FLIGHT TIME: ~ 15 Minutes
1 Safety Pilot takes off
6 Safety Pilot performs landing
SAFETY PILOT
RESEARCH PILOT
MOBILE OPERATIONS STATION (MOS)
SAFETY PILOT

Figure 1. AirSTAR Concept of Operations.

III. Avionics Architecture

The avionics architecture for AirSTAR consists of a custom-built FCU onboard the aircraft and a ground-based dSPACE® real-time computer. The onboard FCU handles sensor data collection, telemetry serial stream parsing and creation, command switching, and actuator command generation. Analog sensor data is collected via ADC boards while digital data from the INS and engine control units is collected via serial inputs. These data are compiled into a serial stream which is telemetered to the MOS, where it is received into the dSPACE real-time computer. In addition to the serial I/O, the dSPACE computer receives analog and discrete inputs from the research cockpit interface, PWM inputs from the safety pilot's commands, network connection, and a fiber-optic link to the a host PC. Data are stored at 200 Hz using a stream-to-disk method between the dSPACE real-time computer and the host PC's hard drive. These data can be converted to MATLAB® format within minutes after a flight. An Apogee® data recorder is also used to synchronously record serial telemetry data, intercom audio, and video streams. The Apogee system supports a playback capability that can be used to recreate the flight environment within the MOS.

IV. Software Architecture

The software architecture for the AirSTAR MOS consists of three main elements: simulation, real-time flight code, and display software. The displays provide the pilot with a synthetic scene view and instrument overlay customized for flight dynamics research. They also provide real-time status and health monitoring information to operations personnel within the MOS. The software for both simulation and the flight code is primarily developed in The MathWorks MATLAB/Simulink® environment. The real-time flight code is generated from Simulink using Real-Time Workshop® and implemented on the dSPACE real-time computer. MathWorks Mex-functions written in C are used for packing and unpacking the serial telemetry stream, while built-in dSPACE libraries are used for the analog/discrete/PWM inputs and network outputs.

A. Simulation

The simulation software runs in Simulink on a PC and has three different environments: design, emulation, and hardware-in-the-loop (HIL). The design environment is intended for design and initial checkout of control laws and consists primarily of the aircraft model (*i.e.*, aircraft dynamics and subsystems such as engines, actuators, and sensors). The emulation environment includes the aircraft model along with the flight control system, an avionics model, a telemetry system model, and all of the components in the real-time flight code, except for the hardware drivers. This emulation environment enables new algorithms and code changes to be debugged and tested in Simulink with a high degree of fidelity. The HIL environment is essentially the same as the emulation environment without the real-time flight code components. In this environment, the simulation runs in soft real-time on a PC with a serial I/O card emulating the aircraft, avionics, and telemetry system. In this mode the MOS systems and the flight control computer can be used for performance testing, flight profile planning and full mission rehearsals.

B. Libraries

As with any software development effort, configuration management is extremely important and becomes challenging as the size and complexity of the code grows and the number of developers increases. In a traditional Simulink model, all of the code (with the exception of built-in blocks) is contained within a single diagram and is stored as a single model file. This setup requires any and all changes to take place in a single file and makes configuration management, validation, and verification especially difficult in a project with multiple developers.

One approach is to use Model Referencing, which divides the diagram into multiple model files, each of which can be compiled separately and linked later into a final executable code. However, model referencing imposes additional constraints (*e.g.*, BusObjects to declare model interfaces) and is not compatible with some dSPACE configurations. Another method is to put major subsystem components into separate library files. Library files are generally used for blocks that have multiple instances throughout a diagram, but also serve the purpose of dividing a single model into multiple files. Libraries also allow the use of configurable subsystems, which enables the software to easily switch between different implementations for the same basic function.

Libraries are used extensively in the AirSTAR software, so that each major subsystem (*e.g.*, engines, aerodynamics, telemetry processing, etc) is maintained in a separate library file. With this arrangement all three simulation environments use the same library blocks for the aircraft models and subsystems. The emulation environment and real-time code also draw from the same libraries. This software reuse speeds development and testing and decreases the likelihood of errors being introduced through recoding and reimplementation.

Each library file is kept under configuration management using Subversion (SVN), an open source version control tool. Although Simulink models are ASCII text files, a contextual merge of updates to a model file is not desired. As with source code, a contextual merge on a model file may not produce syntactically correct code. However, with a model file these errors can not be easily corrected as the broken code won't load properly into the simulink diagram editor. The solution is to treat these files as binaries within Subversion. This allows updates to files only if the working copy is unmodified, otherwise it produces a conflict. To allow concurrent development the diagram should be broken into as many libraries as possible. Since libraries are loaded during initialization (pre-compile) there is no performance penalty with this method and version control information and change logs can be tracked with a high degree of granularity[13].

C. Software Block Diagram

A block diagram of the software in the flight control computer is shown in Fig. 2. The commands from both pilots are input to the FCS, in addition to the aircraft sensor data, the output from the Calculated Parameters subsystem, and the Caution & Warning subsystem. The Calculated Parameters subsystem calculates unmeasured quantities such as airspeed and altitude (from dynamic and static pressure) and applies center of gravity offset corrections to appropriate sensor data. The Caution & Warning subsystem provides alerts and advisories to the pilot based on sensor data, as well as performing integrity monitoring of the telemetry link and instrumentation. The outputs of the FCS are control surface and throttle commands in engineering units, which are calibrated to actuator commands and sent to the aircraft via the L-band telemetry uplink. Various data parameters are broadcast in real-time via user datagram protocol (UDP) messages over the MOS Ethernet network. Two UDP packets are used, one containing data needed to drive displays, and another specifically for research needs. Both packets are currently broadcast at 60 Hz.

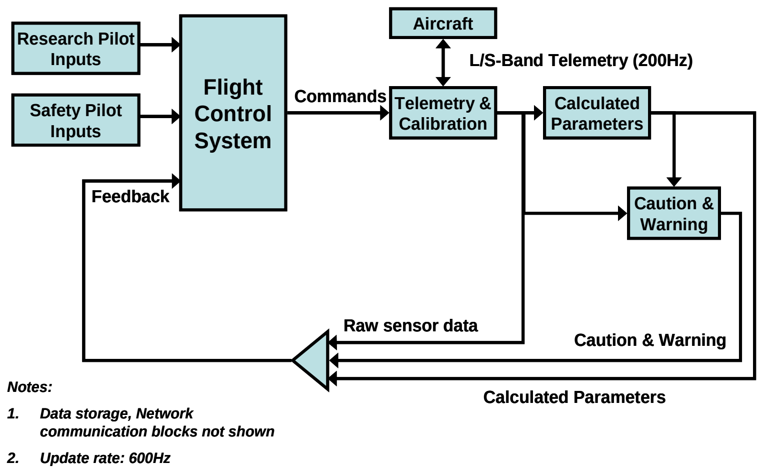

Figure 2. Block diagram of flight control computer software.

D. Flight Control System

The AirSTAR FCS[9] uses a reversionary build-up approach to mitigate the risks associated with flight testing complex research control laws. Complexity is added in distinct stages that can be quickly transitioned using a two-switch "arm" and "engage" process. The FCS is separated into three flight control law (FCL) modes, shown in Fig. 3. These three modes are mutually exclusive; only one can be active at a time. Mode 1 is a direct, stick-to-surface control law composed of stick shaping only; no sensor feedback is used. This mode is the reversionary control law and is simple by design. Mode 2 is the baseline FCL, containing a conventional (non-adaptive) closed-loop controller. Mode 3 is reserved for the research control laws, and can contain numerous FCLs, although only one can be operational at any given time.

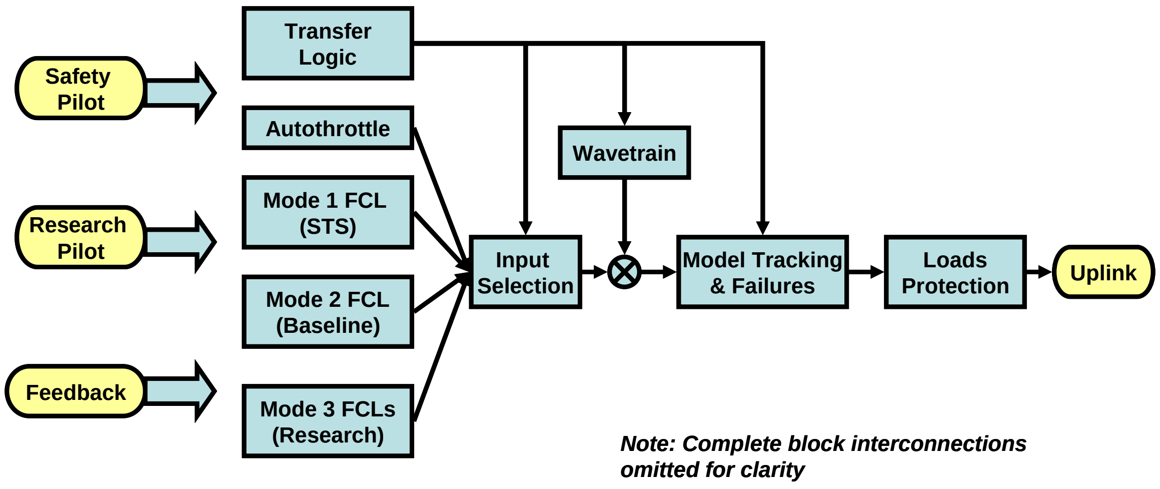

Figure 3. AirSTAR FCS top-level block diagram.

The FCS also contains three auxiliary modules that can be used in conjunction with Modes 1, 2, or 3: an Autothrottle, a Wavetrain module, and a Model Tracking & Failures module. The final component in the FCS is a Load Protection System (LPS). The purpose of the LPS is to prevent the FCS from commanding control deflections that would result in exceeding the structural limits of the test aircraft. The LPS, Transfer Logic, and Input Selection blocks are always active.

E. Simulink Pros and Cons

The use of Simulink for development and Real-Time Workshop to generate real-time code allows researchers and control law designers to develop and test algorithms in the familiar Simulink environment while avoiding the lower-level details of real-time code[2,3,4]. This has enabled a rapid transition from algorithm development to compiled real-time flight code and significantly improves the debugging and testing process. Such efficiency comes at the cost of reduced understanding of what is happening in the lower-level software. Simulink automatically handles many of the programming details, such as signal dimensions, sample times, data types, and buffers between multi-rate systems. However, as diagrams grow in size and complexity, the automatic determination of these properties sometimes fails. These initialization failures are presented as errors, although often there are no code errors in the diagram itself, just an under-specification of properties. The solution to these problems is to explicitly specify port and bus properties through the use of signal specification blocks, input /output property dialogs and bus objects. However, doing so makes the code less flexible and removes some of the benefit of using model based design as a high level language. To date these errors have been mitigated by specifying signal properties at selected interfaces within the diagram and taking greater care to explicitly handle transitions between multi-rate systems. The level of specification required is managed as a practical balance between the ease of code modifications and the consequences of the fragile initialization process described above.

V. Research Control Law Implementation

Two common themes in many flight test programs are the limited quantity and significant cost of flight time available. This often creates a conflict between a conservative build-up approach to safely evaluating new control algorithms and testing numerous control laws over a wide flight envelope. In particular, for control algorithms designed to enhance safety in upset flight or damage conditions, the desired research interest is not in optimizing nominal performance, but in finding out under what circumstance and why a control law technology will fail. Testing up to the boundaries of stability adds risk to the flight experiment, but provides valuable understanding and is a risk that may be deemed acceptable in an unmanned vehicle. To addresses these challenges, the AirSTAR software is designed to maximize productivity of the test time available by enabling multiple research controllers to be tested during a single flight and providing a rapid reversion from research to baseline control algorithms.

In the AirSTAR FCS, research control laws are hosted in the Mode 3 FCL subsystem. This module utilizes a selectable, Enabled Subsystem architecture that allows a large number of research control laws to be implemented concurrently and tested individually during a single flight. Each research control law can be a different type and have a different structure and implementation. From a software standpoint, there is no limitation on the type of control law (*i.e.*, inner-loop or outer-loop), but the current technical plans are focused on pilot-in-the-loop control laws rather than full outer-loop autopilots.

Research FCLs are disabled (*i.e.*, the code is not executed) until the particular FCL is selected and armed. The selected research FCL outputs are not active until Mode 3 is engaged by the research pilot. Sensor failures can be emulated in the Mode 3 FCL by modifying any or all of the sensor data from the aircraft. Any modification to the sensor data is local to the Mode 3 FCL and does not affect the remainder of the FCS. Finally, 30 channels of user-definable data can be sent out over the MOS network via UDP and observed in real-time during a test. Each FCL implemented in Mode 3 can define different parameters for the 30 user data channels, but only the armed/engaged FCL user data is sent out over the MOS network.

The control laws in the Mode 3 FCL subsystem are research products that are frequently updated and may include configurations that fail in ways that would adversely affect the rest of the software. To mitigate this risk, a development effort to partition the Mode 3 FCS subsystem to run on a separate processor in the dSPACE real-time computer is underway. Any software faults that may occur in Mode 3 (*e.g.*, frame overrun, segmentation fault, *etc.*) will have no impact on the remaining software, which will continue running normally.

VI. Failure Emulation

The purpose of the failure emulation capabilities in the AirSTAR software is to support validation of technologies developed under the NASA Integrated Vehicle Health Management (IVHM) project and the NASA

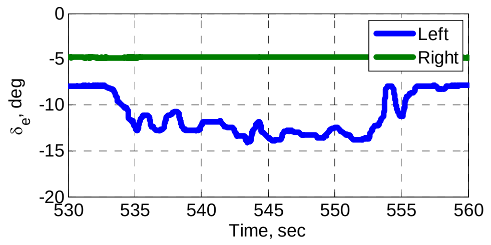

Figure 4. Time history of left and right elevator positions showing the right elevator is failed.

Table 1. Test conditions and performance criteria for jammed elevator evaluation task.

	Target Condition	Desired Criteria	Adequate Criteria
Airspeed	80 kts.	±4 kts.	±8 kts.
Altitude	950 ft.	±50 ft.	±100 ft.
Bank Angle	45°	±5°	±10°

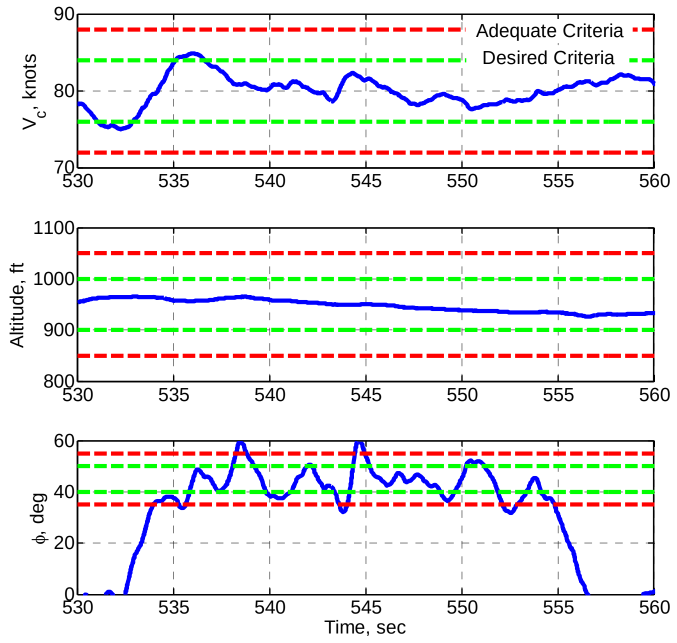

Figure 4. Velocity, altitude, and bank angle during capture task with right elevator fixed.

Integrated Resilient Aircraft Control Project (IRAC) project. The technical goals of the IVHM project include the development of validated tools, technologies, and techniques for the automated detection and diagnosis of various faults and failures. The technical goals of the IRAC project include the development of a set of validated flight control design tools and techniques for enabling safe flight in the presence of adverse conditions, which includes failures, upsets, damage, and icing. By designing this capability into the FCS architecture, the goals of both projects can be quickly and easily supported. This software-based approach also allows the emulated failure to be immediately reversed with the flip of a switch should the need arise during flight.

The AirSTAR FCS emulates failures in the Model Tracking & Failures module (Fig. 3). This provides the capability to emulate control surface failures by modifying any of the control surface commands. A large number of user-defined failure profiles can be implemented, which can then be easily selected and activated while in flight. The present implementation is focused on simple failures, *e.g.*, a jammed surface, bias, reduced effectiveness, or any combination of the three, which can be to be applied to any command (Eq. 1). To minimize transients when activated, the failures are applied relative to the initial control surface commands (δ_0).

$$\delta_{out} = \delta_0 + Gain * \left(\delta_{in} - \delta_0\right) + Bias \quad (1)$$

An example of one failure that was emulated during a flight test deployment is a jammed right elevator. During this scenario, the right elevator remained in a fixed position while pitch commands actuated only the left elevator (Fig. 4). While this failure was being emulated, the pilot performed an evaluation maneuver to capture (within 3 seconds) and maintain a precise bank angle while precisely maintaining airspeed and altitude. The primary purpose of this maneuver was to evaluate the suitability of this task for control law evaluation during failure emulation. Target conditions as well as desired and adequate performance criteria are shown in Table 1.

Figure 5 shows the time histories of calibrated airspeed, altitude, and bank angle that were recorded during this evaluation task. Although desired performance was obtained for altitude, and adequate performance was obtained for airspeed, adequate performance was not obtained for bank angle control. Pilot comments indicated the inability to adequately maintain bank angle was due to roll coupling produced by the asymmetric elevator deflections. The pilot commented that the task was satisfactory for future control law evaluation.

Planned extension of capabilities in this module include a model tracking controller which will enable in-flight simulation of a range of effects, such as reduced static and dynamic stability and the aerodynamic effects of structural damage or icing.

VII. Aircraft Structural Load Protection

Unintentional failures of experimental control algorithms and concepts can range from the benign (*e.g.*, lack of desired controller performance) to extreme (*e.g.*, control hardovers). While rigorous testing and analysis of the control algorithm under evaluation can prevent most failures of this nature, guaranteeing stability can be a challenge in some cases (*e.g.*, adaptive control algorithms, controllers in the presence of failure conditions, *etc.*). In keeping with AirSTAR's goal of a rapid prototyping development and test environment for control algorithms, the AirSTAR FCS utilizes software protections (in addition to planning test procedures so as to limit dynamic pressure) to prevent structural damage to the test aircraft in the event of a control algorithm failure.

The AirSTAR FCS has a Load Protection System (LPS), which is designed to prevent the FCS from exceeding the structural limits of the test aircraft. The primary focus of the LPS is normal load factor (Nz), with secondary focus on side force loads (Ny). The LPS uses a two-step approach to prevent the FCS from exceeding normal load factor limits: the first approach (Plan A) is proactive and limits elevator authority as a function of dynamic pressure. Limiting elevator authority proactively was found to the most effective way to prevent excessive load factor. A simulation study showed that elevator hardovers resulted in very high g-onset rates which are difficult to reverse quickly enough to prevent excessive load factors. The second approach (Plan B) is reactive and sets the controls to a neutral position if the specified load factor thresholds are exceeded. This approach is intended as a backup in case Plan A fails to limit load factor as expected.

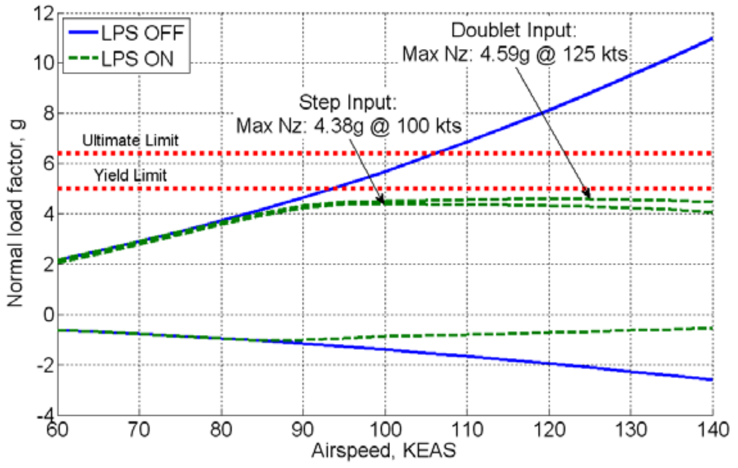

Figure 5. Simulated peak load factor for the S2 aircraft.

The LPS Plan A elevator limits were set by first creating a database of peak load factor as function of dynamic pressure and elevator inputs by simulating elevator hardovers (step inputs and doublet inputs) from a range of trimmed flight conditions and recording the peak transient load factor. Given a maximum and minimum load factor limit, this database can be used to set the maximum allowable elevator authority as a function of dynamic pressure.

Figure 6 is a plot of peak transient load factor versus equivalent airspeed for elevator hardover inputs (both steps and doublets). The solid blue line is the peak load factor without the LPS activated; these data are used to set the Plan A elevator limits. The dashed green lines show that the peak load factors with the LPS (Plan A & B) engaged are below the yield limit (5.0g) and ultimate limit (6.4g) of the S2 aircraft[9] (shown as dotted red lines).

In a manner similar to Plan A for normal force loads, excessive side force loads are prevented by limiting the rudder authority as a function of dynamic pressure. The rudder limits are set so a full rudder reversal at maximum attainable sideslip angle will not exceed the structural limits of the vertical tail at a given dynamic pressure. If the structural limits of the vertical tail are not known (as is the case for the S2 aircraft, because it is a COTS kit), the rudder limits are set to the maximum deflections expected to be needed for the maneuvers in the flight test plan. If the structural limits of the aircraft are known, the LPS can effectively mitigate the risk of structural damage due to excessive control inputs. As a result, research control laws can run through development and test iterations in a timely fashion with minimal risk to the test aircraft.

VIII. Integrity Monitoring

Knowledge of system health is crucial during high-risk flight test maneuvers, even more so when control of the aircraft is subject to this status. Several software elements are aimed at monitoring the health and integrity of the telemetry link and the research control laws being studied.

A. Telemetry Link Monitor

The telemetry link is a critical component for conveying the research pilot inputs and for the experimental control system implemented in the ground based flight computer. At a basic level the telemetry stream employs Fletcher checksums at the end of each message to verify the message was not corrupted in transmission. These checksums are verified both in the on-board computer for uplink messages and in the ground based computer for downlink messages. If a checksum test fails, the system simply retains the previous value for any data expected from that message. Checksums provide confidence in the integrity of the received message, however, they cannot provide a good sense of the link state because status of the uplink checksums are not part of any downlink message, and the checksums will not fail if the link is fully out and no message header is received.

To ensure integrity of the telemetry link, a monitor was implemented to interrogate the round trip data link from the ground system to the vehicle and back, and signal any failures. The design for the link monitor uses a spare set of uplink messages which are transmitted to the airborne digital I/O ports. These digital input and output channels are physically wired together in a loopback configuration. A 20 Hz square wave is sent to the digital output port from the ground computer via the telemetry uplink and received back down on the downlink. Since both the onboard and ground systems will “hold-last-value” on a transmission error, a failure to transition at the 20 Hz rate on this signal indicates a dropout condition on the link. When a sustained dropout occurs for 0.75 seconds, the link is declared inoperative.

Different actions are taken for dropouts versus an inoperative link. The FCS uses the dropout flag to freeze all integrators associated with the control laws to prevent integrator wind-up. However, the controllers continue to run and there is no automatic mode change or pilot notification. The FCS uses the link inoperative flag to automatically revert the FCS to the lowest reversionary state (Mode 1). The link inoperative flag also triggers a large yellow “X” overlay on all displays in the ground station, and procedurally this calls for the safety pilot to take control of the vehicle via his independent link.

The initial design of this system had a subtle but significant flaw. The uplink message which contains the 16 bits of digital I/O data is very short, with only a 2 byte payload. The two uplink messages which carry the 32 channels of surface commands to the vehicle are much longer with a combined 64 byte payload. If the link is experiencing random bit-loss type errors the probability of the surface command messages failing its checksum is larger than that of the digital I/O message. A condition can occur where the downlink is operative (so the cockpit displays remain “live”) and the short digital I/O message is getting through, but the uplink commands messages are failing to reach the aircraft. This situation occurred during a checkout flight test deployment. While the downlink remained functional, a period of 3.5 seconds passed where uplink surface commands were failing to reach the aircraft, but the pilot had no indication of the telemetry link being inoperative. Within this timeframe dropouts occurred, but none sustained long enough to trigger the link inoperative condition.

An upgrade to the link monitor has been implemented using a spare channel in both of the uplink surface command messages to transmit the link monitor signal. The PWM output of these channels is physically wired to the digital input ports and received back on the downlink. The link monitoring function on the ground computer ensures both returning signals are transitioning properly. The transition rate of the signal was also increased to 50 Hz to decrease the delay in detecting a dropout condition. Higher transition rates could not be used because spurious dropout indications would occur, due to the multiple asynchronous clocks involved in the full path of the link monitor signal.

B. Inf and NaN Check

In testing of any experimental algorithm, the possibility exists of having an infinite-valued signal (Inf). While infinite values can be handled with normal saturation and rate limiter blocks, an infinite-valued signal can easily become Not-a-Number (NaN). This can occur through the following operations which produce NaN[14]:

1. Any arithmetic operation on a NaN, such as sqrt(NaN)
2. Addition or subtraction, such as magnitude subtraction of infinities as (+Inf)+(-Inf)
3. Multiplication, such as 0*Inf
4. Division, such as 0/0 and Inf/Inf
5. Remainder, such as rem(x,y) where y is zero or x is infinity

A NaN signal will pass through saturation and rate limiter blocks (including the LPS) uninhibited and could cause control surface hardovers. A solution was found using an Embedded MATLAB function block. The “isnan” and “isinf” functions are supported in Embedded MATLAB, enabling a simple function to be written which detects Inf or NaN values and replaces them with the last non-Inf/NaN value. If any Inf or NaN values are detected, an audio and visual warnings are activated in the cockpit.

C. FCS Mode Inhibits

If a research control law is initialized properly, initial control surface commands will match the current commands and no transient will occur once it is engaged. However, if this is not the case, the FCS has inhibit logic which prevents control laws from being engaged if the initial command would cause a significant normal load transient. This system uses the same methodology and database as the LPS Plan A system described above, but has significantly lower load factor limits. In addition, the FCS automatically reverts to the lowest reversionary state (Mode 1) if the telemetry link fails or the safety pilot takes control. Once control is returned to the research pilot and the telemetry link is established, all FCS modes or functions must be rearmed before they can be engaged again.

D. Mode 3 UserData Outputs

Each control law implemented in Mode 3 can output 30 channels of user selected data. These data are stored at 200Hz and are broadcast in real-time over the MOS network via UDP at 60Hz, allowing real-time monitoring of internal control law parameters. Having access to the data in real-time allows researchers in the MOS to implement and evaluate near real-time algorithms such as parameter identification and fault detection logic without being tightly integrated to the real-time flight software. This also allows quick evaluations of research results that can impact the flight test sequence during flight if repeats or modifications to maneuvers are necessary.

IX. Concluding Remarks

Some of the risks associated with flight testing of experimental control laws can be reduced or eliminated by using subscale unmanned aircraft. Some challenges remain, however, including an increased reliance on the flight software and need for efficient software development.

The AirSTAR facility uses Simulink and Real-Time Workshop to reduce the time required for algorithm development and testing. Simulation is used throughout the development cycle, from algorithm design to full mission rehearsals. Simulink libraries are used to maximize code reuse, increasing the fidelity of developmental testing and improving configuration management. The AirSTAR FCS uses a reversionary, build-up approach to add control law complexity. Multiple research control laws, each of a different type, can be hosted in the FCS via use of Enabled Subsystems. Faults and failures, for the purpose of testing control law robustness, are emulated in software. A Load Protection System has been implemented to safeguard the aircraft against excessive structural loads resulting from unexpected control law failure. Finally, integrity monitoring is performed throughout the software, monitoring the telemetry link status, checking for numerical errors, inhibiting activation of out-of-range control laws, and providing numerous channels of real-time internal control law data to researchers.

References

[1]Dobrokhodov, V. N., et al, "New Generation of Rapid Flight Test Prototyping System for Small Unmanned Air Vehicles," AIAA 2007-6567, *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Hilton Head, SC, 2007.

[2]Jang, J. S., Tomlin, C. J., "Design and Implementation of a Low Cost, Hierarchical and Modular Avionics Architecture for the DragonFly UAVs," AIAA 2002-4465, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, CA, 2002.

[3]Christophersen, H. B., Pickell, W. J., Koller, A. A., Kannan, S. K., Johnson, E. N., "Small Adaptive Flight Control Systems for UAVs using FPGA/DSP Technology,", AIAA 2004-6556, *AIAA 3rd "Unmanned Unlimited" Technical Conference*, Workshop and Exhibit, Chicago, IL, 2004.

[4]Smith, J. I., Valente, E. G., Eubank, R. D., Atkins, E. M., "A Low-Cost Research Autopilot for System Identification," AIAA 2005-6963, *AIAA Infotech@Aerospace*, Arlington, VA, 2005.

[5]Gu, Y., Seanor, G., Gururajan, S., Napolitano, M. R., "Integrated Avionics System for Research UAVs," AIAA 2008-7490, *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, HI, 2008.

[6]Ma, L., Stepanyan, V., Cao, C., Faruque, I., Woolsey, C., Hovakimyan, N., "Flight Test Bed for Visual Tracking of Small UAVs," AIAA 2006-6609, AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO, 2006.

[7]Risch, T., Cosentino, G., Regan, C. D., Kisska, M., Princen, N., "X-48B Flight-Test Progress Overview," AIAA 2009-934, *47th AIAA Aerospace Sciences Meeting*, Orlando, FL, 2009.

[8]Cunningham K., Foster J. V., Morelli E. A., and Murch A. M., "Practical Application of a Subscale Transport Aircraft for Flight Research in Control Upset and Failure Conditions," AIAA 2008-6200, *AIAA Atmospheric Flight Mechanics Conference*, Honolulu, HI, 2008.

[9]Murch, A. M., "A Flight Control System Architecture for the NASA AirSTAR Flight Test Infrastructure," AIAA 2008-6990, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Honolulu, HI, 2008.

[10]Jordan, T. L., Foster, J. V., Bailey, R. M., and Belcastro, C. M., "AirSTAR: A UAV Platform for Flight Dynamics and Control System Testing,", AIAA 2006-3307, *25th AIAA Aerodynamic Measurement Technology and Ground Testing Conference*, San Francisco, CA, 2006.

[11]Jordan, T. L., Langford, W. M., and Hill, J. S., "Airborne Subscale Transport Aircraft Research Testbed – Aircraft Model Development," AIAA 2005-6432, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, 2005.

[12]Bailey, R.M., Hostetler, R.W., Barnes, K.N., Belcastro, Christine M., and Belcastro, Celeste M., "Experimental Validation:

Subscale Aircraft Ground Facilities and Integrated Test Capability," AIAA 2005-6433, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, 2005.

[13] Walker, G., Friedman, J., Aberg, R., "Configuration Management of the Model-Based Design Process," SAE-2007-01-1775, *SAE World Congress & Exhibition*, Detroit, MI, 2007.

[14]MATLAB R2007a Help, The MathWorks, Inc., Nantick, MA, 2007.