COMMUNICATION INTEGRITY FOR FUTURE HELICOPTERS FLIGHT CONTROL SYSTEMS

Amira Zammali, Agnan de Bonneval, Yves Crouzet, Pascal Izzo, Jean-Maxime Massimi

COMMUNICATION INTEGRITY FOR FUTURE HELICOPTERS FLIGHT CONTROL SYSTEMS

Amira Zammali, Agnan de Bonneval and Yves Crouzet, LAAS-CNRS, Université de Toulouse, UPS, F-31400 Toulouse, France

Pascal Izzo and Jean-Maxime Massimi, Airbus Helicopters, Aéroport Marseille-Provence, 13725, Marignane, France

Abstract

The evolution from mechanical to Fly-By-Wire (FBW) designs of Flight Control Systems (FCS, the system that controls the aircraft trajectory) in both airplanes and helicopters has been a crucial step offering a variety of benefits such as easing the pilot mission and reducing the mechanical complexity of the aircraft. Yet, all these advantages have limited improvement unless the required safety level is met. In fact, for such systems, a very high safety level is imposed by both the safety-critical property of the system and certification standards (e.g., ARP4754A and ARP4761 standard). Now, industrials such as Airbus Helicopters aim at installing fully digital FBW architectures on future helicopters. This step raises new challenges particularly to comply with certification standards requirements. We present, in this paper, the architecture of future fully digital Airbus Helicopters FCS considered at the end of feasibility study. We focus particularly on the communication integrity issue of future digital architectures. In such systems, the non-detection of corrupted messages could lead to catastrophic consequences. To enhance communication integrity, we propose an end-to-end communication integrity approach based on the *black channel concept*, it is to be implemented in the application layer. This approach uses error detection codes. Given the constraints of targeted systems namely "embedded" and "safety-critical" features, the selection strategy of error detection codes consists in a trade-off between the computational cost and the error detection capability.

Problematic

The FBW evolution and the required high safety level imposed by certification standards such as ARP4754A [1] and ARP4761 [2] are the main drivers to design a safety-sensitive architecture for future Helicopters FCS. In such systems, the catastrophic failure (leading to casualties) rate should be extremely low; it must be lower than 10^{-9} failures per hour in FCS [3]. To meet this strict requirement, added fault prevention, fault removal and fault forecasting, one commonly used concept is the fault tolerance. Fault tolerance is carried out via error detection and system recovery. Fault tolerance consists in using redundancy in order to make the system works correctly despite the presence of failures. For example, to ensure system availability, some components are duplicated such that a backup component could replace a primary component in the case of failure. Adopting the fault tolerance concept raises two challenges. The first challenge is the fact that for the sake of reducing the system cost and weight, over-redundancy is to be avoided. The second challenge is the fact that symmetric redundancy does not cope with the problem of common mode failures. In fact, symmetric redundancy consists in duplicating some system components such that the copies of these components have exactly the same properties as those of primary components; they have therefore the same pitfalls. So, different diversification strategies (software, hardware, etc.) should be cooperatively adopted in order to reduce the risk of common mode failures.

One of the goals of this paper is to describe a fully digital architecture meeting stringent safety requirements while taking into account different system design constraints. The case study is based on future Airbus Helicopters FCS architecture. Since this future FCS is a distributed system based on digital networks, a second goal is to describe how to enhance communication integrity. We aim at coping with the problem of non-detection of erroneous messages.

Data corruption is caused by several factors such as thermal noise, electromagnetic interferences, radiation, memories failures, etc. A Helicopter FCS is a system with a very severe environment (e.g.,

vibration, temperature) which increases the likelihood of components failures and data corruption. To enhance communication integrity, we propose in this paper an end-to-end integrity approach based on error detection codes. The challenge we seek to address is the selection of efficient codes with good error detection capabilities and lightweight computational costs given the system constraints, particularly the fact that targeted systems are time-critical and limited in terms of processing and memory resources.

In this paper, we first present Helicopters FCS with a focus on the FBW evolution and future challenges. We present then the architectures of future Airbus Helicopters with a focus on the communication integrity approach to deal with the problem of data corruption, we assess the performances of different error detection codes and we give some guidelines and best practices to better use error detection codes in safety-critical embedded systems.

Helicopters Flight Control Systems: description, FBW history and future challenges

We describe in this section the targeted systems which are Helicopters FCS with a focus on key properties, the FBW evolution and future challenges.

Helicopters FCS description

The Flight Control System is the system that controls the trajectory of the aircraft (airplane, helicopter or other). It enables the control of the aircraft flight path all along take-off, flight and landing. Commands are sent to actuators of control devices either manually by the pilot or automatically by the autopilot. These commands are calculated using both the pilot orders and information provided by the sensors. To fly a helicopter, the pilot should use different flight controls in order to tune the controllable forces. In fact, a Helicopter is subjected to four forces (Figure 1):

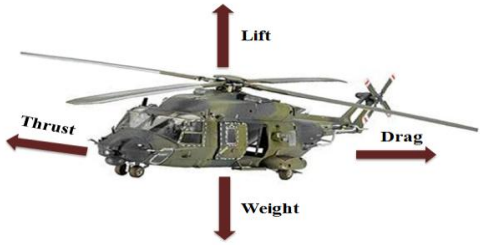

Figure 1. The Four Forces Applied to a Helicopter

- Lift: it is a force that opposes the weight of the aircraft permitting it to fly.
- Drag: it is the force that opposes the movement of the aircraft.
- Weight: the gravitational force acting on the total mass of the aircraft
- Thrust: it is the force generated to oppose the drag.

The three main flight controls to be used by the pilot are as following [4]:

- The collective stick, which controls of the pitch of the main rotor, source of helicopter's lift and thrust.
- The cyclic stick, which controls the tilt of the main rotor, in pitch and roll.
- The rudder pedals, which perform the yaw control and control of the anti-torque control.

Key properties and constraints of Helicopters FCS

Aircraft FCS have particular properties resulting in strict requirements and constraints to be taken into account in order to design a relevant architecture. These key properties are:

- *Safety-critical property*: this means strict safety requirements given the hazardous consequences that could be caused by potential failures. Failures in a FCS could lead to fatalities and material damages. Thus, the catastrophic failures rate must be extremely low ($<10^{-9}$/h for FCS of civil aircrafts [3]). To meet imposed strict safety requirements, FCS designs are usually based on fault tolerant architectures with different types of

redundancy: temporal, spatial, software, hardware, etc.

- *Embedded property*: this means limited resources (memories, processing, and power supply) which generally induce short communication messages. In such systems, architectures should not be over-redundant in order to reduce the weight and the volume of the system.
- *Real-time property*: this means that the system is time-critical. Thus, the output is not judged to be correct unless it is correct in both result and time. And the outputs (the commands) must be regularly refreshed (short refresh cycles) despite the fact that processing resources are limited.
- *Severe environment property*: this means that the system is subjected to different causes of failures such as radiation, vibration, etc.
- *Communication system complexity*: communications in fully digital FBW FCS are based on complex networks including a large number of nodes (e.g., calculators, actuators and sensors). This characteristic increases the occurrence likelihood of erroneous messages. Therefore, enhancing communication is crucial.

The selected Helicopter FBW FCS case study is only stressing more the design constraints due to rotorcraft application (weight issue, space issue, commercial competition, vibrations…).

Helicopters FCS FBW history

Unlike the helicopter industry, the airplane industry has adopted the FBW designs since many years [5]. Restricted initially to military airplanes, FBW technology appeared later in commercial airplanes. The first FBW-based FCS was designed by Aerospatiale and installed on Concorde [6].

Before the evolution to FBW designs, mechanical designs of FCS were conventionally deployed in aircrafts. It consists in transmitting the pilot orders to the actuators via mechanical components (rods, cables, etc.). With the increasing size of aircraft, mechanical designs of FCS have presented many pitfalls namely the deployment complexity and the high pilot workload. These drawbacks were the main drivers to the evolution to FBW designs. In FBW-based FCS, the conventional manual controls through mechanical linkages between the pilot and rotor control actuators are replaced by "electronics". FBW architectures are based on computers (hardware and software layers) and electrical sensors, actuators and links.

Compared to mechanical FCS designs, the FBW technology offers a variety of benefits. It makes it possible to introduce advanced flight control laws which increase and improve the aircraft manoeuvrability (especially for helicopters) and provide envelope protections. So, it eases the pilot mission and reduces its workload. It decreases the mechanical complexity and cost and helps to reduce the whole system weight too [7].

Following in the footsteps of airplanes development, Helicopters have moved to FBW designs of FCS some years later. This slow evolution is due to the fact that flight mechanics of a helicopter is more complex than the airplane one. In fact, a helicopter is a highly coupled aircraft, with several interactions and its system operates in a severe environment. The FBW technology was firstly deployed on military helicopters. The military helicopter NH 90 [8] of Airbus Helicopters, whose first electrical mode flight was on March 1997, belongs to the first FBW-based helicopters generation in service. In FBW-based Helicopters FCS, instead of moving the rotor actuators by the pilot through conventional mechanical controls (sticks and pedals), the pilot has just to move electrical sticks or side sticks (Figure 2) which generate electrical signals to be transmitted by wires to Flight Control Computers (FCC). FCCs generate electrical signals to be transmitted by wires in order to control the helicopter actuators.

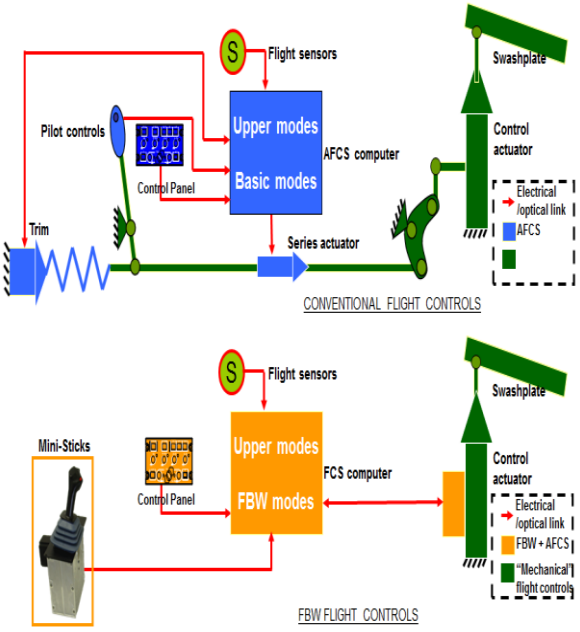

Figure 2. Conventional VS FBW Helicopters Flight Controls

A second great step in FCS designs was the evolution from analog FBW designs to digital FBW. In the start of the 1980's, several airplanes manufacturers launched their first airplanes with digital FBW-based FCS elements such as the Airbus A310 (spoilers system). A crucial step has been crossed when the Airbus A320, implementing a digital FBW-based FCS, was certified and entered into service in 1988 [6].

Helicopters FCS future challenges

Now, the next step, as was the case of airplanes, is to extend FBW, particularly digital FBW architectures, to civilian helicopters. The future Helicopters FCS and particularly the inter-communication architecture should take into account several safety and security (e.g., confidentiality) requirements, we consider particularly the following ones:

- *Availability*: it means the ability to deliver a service continuously. Particularly, in the context of communications, the availability means that the data exchange is ensured despite the presence of failures (e.g., links that are down or data corruption).
- *Integrity*: this means the non-occurrence of unauthorized data modification. In the context of communications, it is the ability of the system to detect and/or recover corrupted exchanged data.

To ensure these two major safety requirements in communications, a redundant architecture with an integrity approach is needed. In the following section, we present the architecture of future fully digital Airbus Helicopters FCS with a focus on the inter-communication architecture.

The architecture of future fully digital Airbus Helicopters FCS

Airbus Helicopters FCS: From current to future architecture

Two different FCS architecture philosophies are adopted by the two major avionics industrials Airbus and Boeing. For Boeing, the FCS architecture is based on the Triplex concept also called *Triple Modular Redundancy* (TMR) with an active/active controlled simplex [9] [10]. US helicopters manufacturers follow quite the same architectural approach without providing a Minimum Equipment List (MEL) (capability to dispatch helicopter despite a first failure). Airbus Group architectures are based rather on *Quadruplex or Sextuplex Active/Standby architecture* with a MEL for both Airplanes and Helicopters. The advantage of Quadruplex architectures for helicopters is the fact that even in the case of a first failure, the requirement of 10^{-9}/h failures is ensured, the pilot is not alerted and the aircraft is able to continue safely its flight. For Airbus Group, every secondary digital FCC (Flight Control Computer) is an asymmetric backup with a different design and running simpler flight control laws. To avoid common mode failures, every two FCC implement different software [5]. Communications are based mainly on ARINC 429. In Boeing architectures, redundancy consists in having three instantiations of every component including computers, communication buses and other hardware components. The Boeing FCS includes three digital computers PFC (Primary Flight Computer and three analog computers ACE (Actuator Control Electronics). Communications are based on ARINC 629

Adopting the same FCS architecture philosophy of Airbus airplanes, Airbus Helicopters particularly the NH 90 [11] is functionally composed (Figure 3) of:

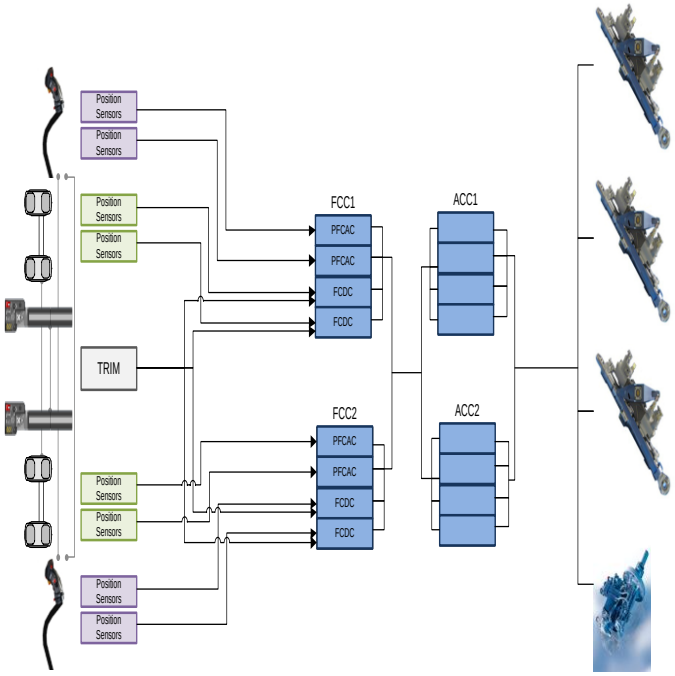

Figure 3. NH 90 FCS Architecture

- Two Flight Control Computer (FCC) boxes including two computers: one Flight Control Digital Computer dual channel (FCDC) and one Primary Flight Control Analog Computer dual channel (PFCAC).
- Two Actuators Control Computer (ACC) boxes including two analog dual channels.
- Conventional Pilots Controls.

The communication between FCC and FCC/ACC is based on ARINC 429 links, which provide digital duplex transmission. By against, the communication link between ACC is based on analog link..

We describe now the architecture proposed by Airbus Helicopters for future FCS. This future architecture aims at overcoming the limitations of existing FCS. These limitations are mainly concentrated on: i) the weight of wiring, ii) system packaging (integration of one Flight Control redundancy and one Actuator Control redundancy in a same box, as shown in Figure 3), and iii) the "cliff effect" resulting from the difference between the levels of Handling Qualities (HQ) of FCDC (HQ=1) and PFCAC (HQ=3) (HQ means great ease of piloting on FCDC operation and return to basic helicopter control on PFCAC). The Flight Control System for future programs enables the pilots to always control the helicopter with a great ease, on its four axes, either by hands-on control or by hands-off control. The FCS can be broken down into three main blocks:

- The *Pilot interface* includes pilot inceptors (cyclic active sidearm controller), collective stick with active force feel, yaw control device, pilot grips controls, displays and mode selection devices as well as mode reference datum tuning device.
- The *Flight Control Processing* (FCP) provides actuator positions requests, reflecting the pilot intended trajectory control. Processing of the required flight parameters is also addressed under this item.
- The *servo actuation* is ensuring that the FCP processed actuators position demands are actually achieved by the actuators. The servo actuation consists in two items, the Actuator Control Processing (ACP) and the actuators. The ACP is housed in the FCC for the Primary Actuation and in the actuator itself for secondary actuation.

The FCS implements also two main sub-systems, which provide helicopter control to the pilot on the four axes (pitch, roll, yaw, collective):

- The Primary Flight Control System (PFCS) which provides helicopter control to the pilot mainly in hands-on mode.
- The Automatic Flight Control System (AFCS) which provides helicopter control to the pilot mainly in hands-off mode, thanks to autopilot upper modes.

We detail now, the Flight Control Processing, which enables the basic control of the helicopter on its four axes and performs:

- The Primary Flight Control Piloting laws (PFCP), which provides the basic control of the four axis in terms of actuator

position requests (primary and when relevant secondary actuation). The PFCP is achieved thanks to a dedicated APSW.

- The Automatic Flight Control Processing (AFCP) enables the automatic control of the helicopter on its four axes, by providing the upper modes required for the helicopter missions. The AFCP is activated through the PFCP, interfaced at relevant PFCP Control law level. The AFCP is achieved thanks to a dedicated APSW, in order to take benefit of upper modes experience and to permit an easier upper mode upgrade.

The Flight Control Processing is organized along the primary control function contributing to helicopter basic control, and gathering the contribution of the automatic control function implementing upper modes, and possible limitation processing. This functional block is relying on flight data parameters in order to elaborate the actuator position demands.

The Actuator Control Processing of the primary actuators implemented inside the FCCs achieves MRAs and TRA loop closure.

The Flight Control processing operation is driven by ground/flight logic mainly based on load on wheel sensors parameters and engine & rotor data.

The limitations of theses current architectures are basically part of primary Flight Control, but can also intervene in upper modes operation (Figure 4).

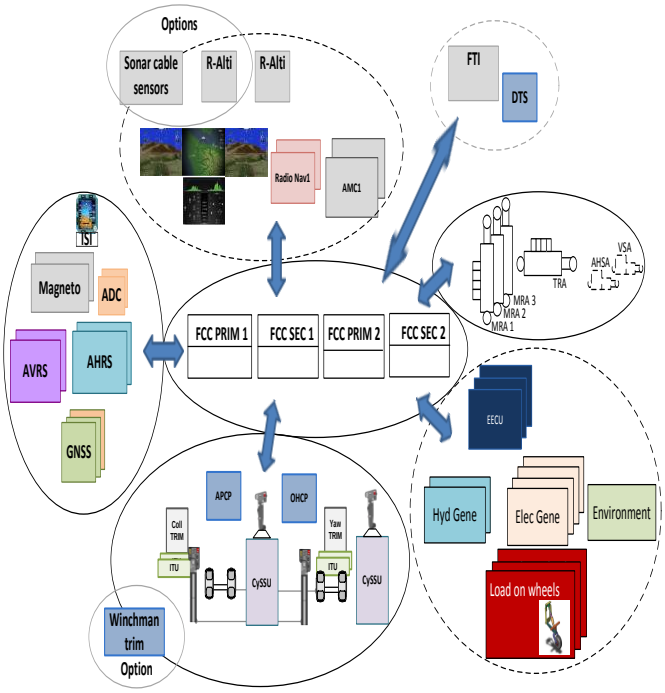

Figure 4. FCS Architecture Overview

The Flight Control System reference architecture considered at the end of feasibility study is the following.

The Quadruplex architecture is composed of:

- 2 Passive Cyclic Side Stick Units (CySSU)
- 1 Collective stick, with Collective Trim Unit and position sensors (ITU)
- Rudder pedals with effort Unit and position sensors (ITU)
- 2 Air Data Computers (ADC)
- 2 Attitude and Heading Reference Systems (AHRS)
- 2 Attitude and Velocity Reference Systems (AVRS)
- 3 GNSS
- 1 Backup Instrument
- 4 Flight Control Computers (FCC)
- Primary actuators: 3 Main Rotor Actuators and 1 Tail Rotor Actuator
- Secondary Actuation: 1 Vertical Stabilization Actuator, 1 Active Horizontal Stabilization Actuator

This FCS reference architecture is interfaced with external: 4 Display systems, 2 Avionic Management Computers, 2 Radio altimeters, 2 Radio Navigation chipsets, 3 Electronic Engine Control Units (EECU), 2 Hydraulic generation channels, 4 Electrical bus bares, 3 landing gears and one sonar sensor.

FBW computers architecture overview

The Fly-by-Wire Flight Control System architecture uses two types of digital computers:

- the primary computer also called PRIM computer.
- the secondary computer also called SEC computer.

For both FCC (Flight Control Computers) types, the computer architecture is based on dual lane architecture (denoted COM/MON), one lane acting as a COMMAND lane and the opposite lane acting as a MONITOR lane; for the sake of optimization (e.g. in order to balance the amount of IO between both lanes) it could be that COMMAND and MONITOR features are mixed within the lanes (e.g. one dedicated lane embodies fore and left main actuator command features and right main rotor actuator and tail rotor actuator monitor features whereas the opposite lane embodies fore and left main actuator monitor features and right main rotor actuator and tail rotor actuator command features). Therefore lane identification will refer to Lane A (denoted #a) and Lane B (denoted #b) to avoid confusion. Suitable mitigations with respect to common mode failures are also implemented.

The FCS operation principles are based on two functional items of Flight Control Computers (FCC): a Flight Control Processing Function (FCPF) providing actuator positions requests to Actuator Control Processing Function (ACPF), which positions the actuators.

The Flight Control Processing (FCP) operation principle is based on active/standby concept, COM/MON FCP processing flight data and pilot inputs as independently as feasible, and are either active alone, or in standby.

The Actuator Control Processing (ACP) operation principle is based on active/active concept, every dual COM/MON ACP redundancy controlling one actuator torque motor. The ACPF performs the selection of the active FCP on the basis of a selector function which will ensure that all redundant FCP select the same correspondent FCP.

FBW intercommunication architecture overview

The FBW system intercommunication principle adapted on the Flight Control System is based on a simplex communication links topology. Each link assumes a safe and secure, one way, 1-to-N transmission. The hardware interfaces are issued from the EIA-485 standard. The protocol used is issued from the HDLC standard**.** The interconnection basis is shown on Figure 5 and on **Figure 6**.

The Figure 5 illustrates the communication links issued from the FCP functions, to the others FCP functions and all ACP functions.

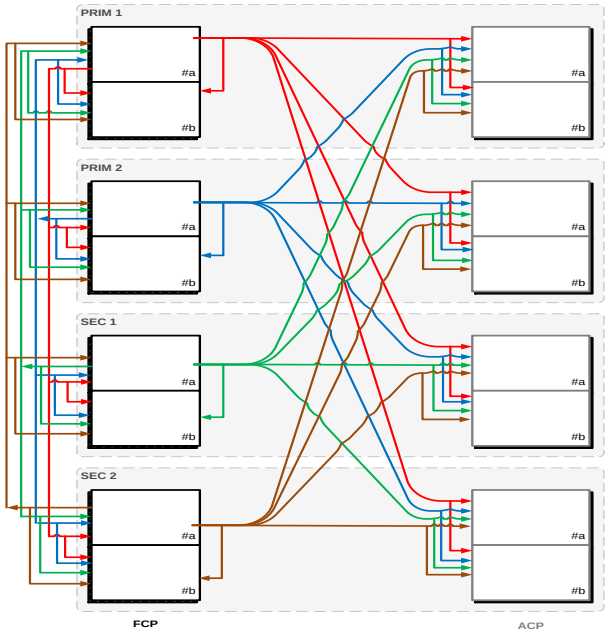

Figure 5. FCC Communication Links FCP-FCP and FCP-ACP

Figure 6 illustrates the communication links issued from the ACP functions, to the others ACP functions and all FCP functions.

Each FCP or ACP is able to transmit to all other FCP/ACP. For this reason**,** the global communication is based on a full-mesh topology. Moreover, this choice of topology is reinforced by a need of system availability increase. Thus, when any FCP or ACP transmitter or receiver fails, it does not impact the communication capability between all other valid

FCP and ACP. It can be possible to distinguish two categories of data links: intra-lane data links and inter-channel data links.

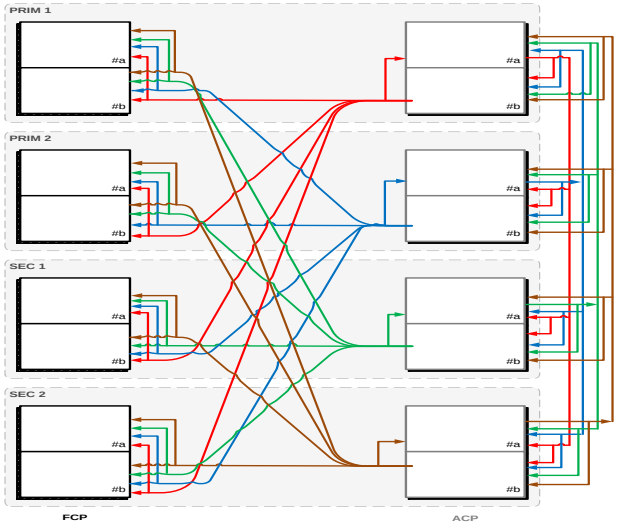

Figure 6. Communication Links ACP-ACP and ACP-FCP

The choice of the particular EIA-485 link is the result of a trade-off concerning simplicity of implementation, highest data-rate, shortest lag time and ease of certification. The bit rate of the links is fixed for all communication at a unique value comprised between 15 and 20 Mbps. The total frame size is 36 bytes. The *frame format* is presented in Figure 7:

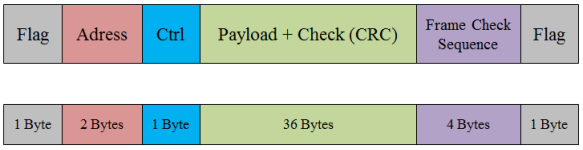

Figure 7. EIA-485 Frame Format

- FLAG: It defines the start or end frame delimiter, and is always equal to 0x7E for HDLC standard.
- Address: The addressing field is used as a message identifier on 2 bytes (with similitude to the ARINC-429 label), and is conforming to the HDLC standard. The message identifier is composed of a Frame ID and a Transmitter ID.
- Control: The control field is not used in our application, but is present to conform to the HDLC standard. It is sized on 1 byte.
- Payload+ Check field (For illustration, we take the example of CRC): The data field contains the message to transmit. It is based on a fixed number of bytes.
- FCS: The frame check sequence field is a CRC of 4 bytes (CRC-32-IEEE). It will be denoted FCRC.

It is noteworthy that communication links described above are redundant which enhance data availability. To enhance data integrity, communications of future Airbus Helicopters FCS will be protected by two error detection codes (Figure. 7); a data-link CRC (4 byte) which is the HDLC CRC (such that $G(x) = x^0 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$) and an end-to-end error detection code (4 bytes whose selection is one of the main objectives of this paper). We describe in the next section the error detection process in internal and external communications and the end-to-end integrity approach, and the selection strategy of the end-to end error detection code.

Internal Communication mechanisms

Internal communications correspond to communications within an FCC channel

At the step 1, data computed by the processing core of each lane are ready. The processing cores build corresponding payload and calculate the associated payload CRC. This step is shown on Figure 8, where the P1 blue & P1 red boxes represent the *payload* respectively of the lane #a and of the lane #b and E1 blue & E1 red boxes represent the CRC payload respectively of the lane #a and of the lane #b.

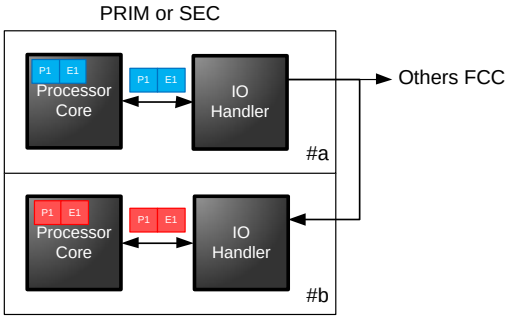

Figure 8. Step 1 of CRC Check in Internal Communication

At the step 2 (Figure 9), each frame containing the payload and associated CRC (Pi and Ei) is encapsulated by both IO handler in a frame with a frame check (FCRC), and then is sent on dedicated cross-lane links.

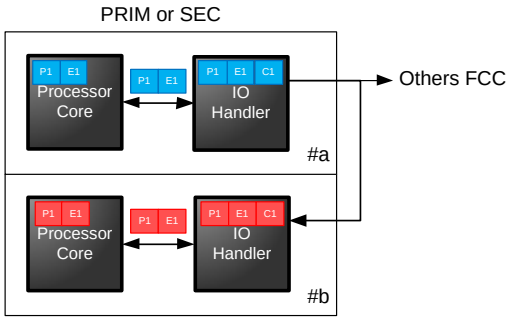

Figure 9. Step 2 of CRC Check in Internal Communication

In step 3 (Figure 10), the frame of the lane #a is received on the opposite lane, which can then unpack part by checking the FCRC.

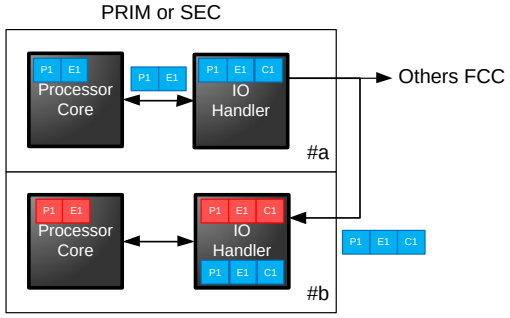

Figure 10. Step 3 of CRC Check in Internal Communication

In step 4 (Figure 11), the IO handler lane #b transmit unpacked payload and CRC associated (sent by the lane #a), after having compared the FCRC C1 blue and red to verify the integrity of the frame.

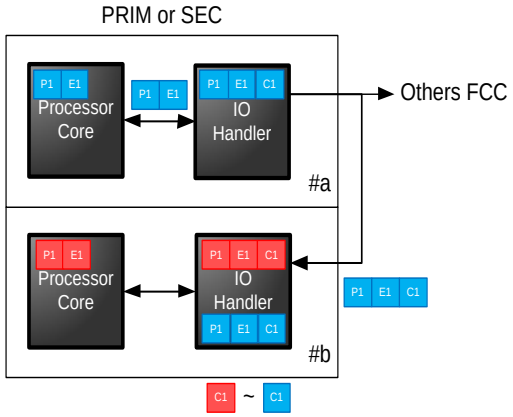

Figure 11. Step 4 of CRC Check in Internal Communication

At the step 5 (Figure 12), the processing core on lane #b receives the payload and associated CRC of the opposite lane, which can then unpack the frame by checking the payload CRC and discarding it.

The mechanism detailed above is the necessary and sufficient mean to assure a safe transmission of the calculated data from a lane to the other, and then to assure the integrity of calculation function by a dual comparison.

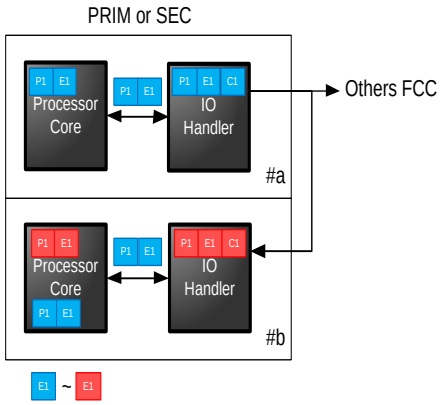

Figure 12. Step 5 of CRC Check in Internal Communication

External Communication mechanisms

External communications correspond to communication between one FCC channel and FCC other communications channels (Figure 13).

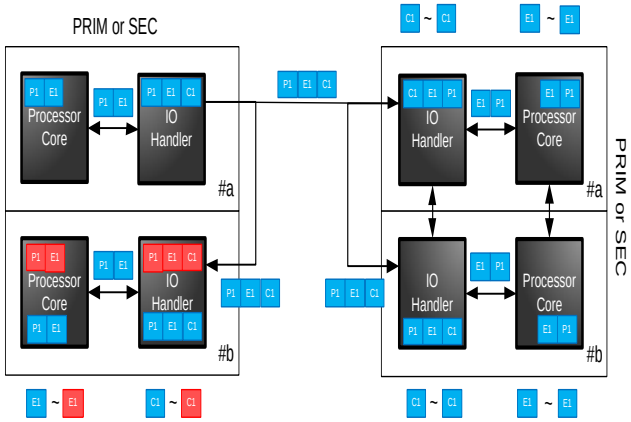

Figure 13. CRC Check in External Communication: The FCP Point of View

From the FCP point of view, the principle for transmitting chain channels remains the same as previously (Figure 13). For the receiver channel point of view, each channel receives the frame and compares to a first level (IO handler) FCRC C1 between lanes and a second level (Processor Core) CRC E1between lanes to check the consistency of the message that was sent (Figure 14).

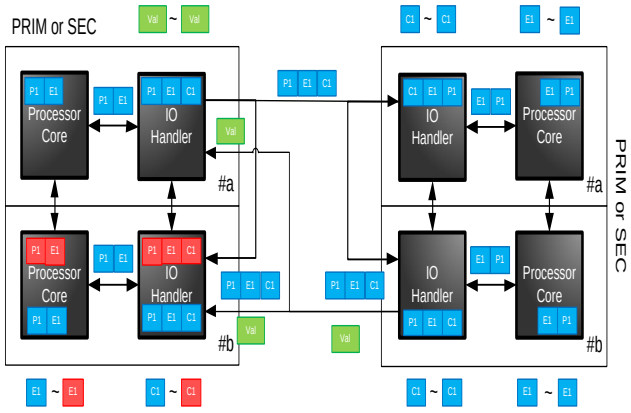

Figure 14. CRC Check in External Communication: the Receiver Channel Point of View

From the ACP point of view (Figure 15), the principle for the transmitting chain remains the same one as previously. On the other hand in the opposite direction (of the right-hand side towards the left), a simple validity (denoted Val) will be transmitted in return. A comparison will be made between ways in order to check the integrity of the transmission.

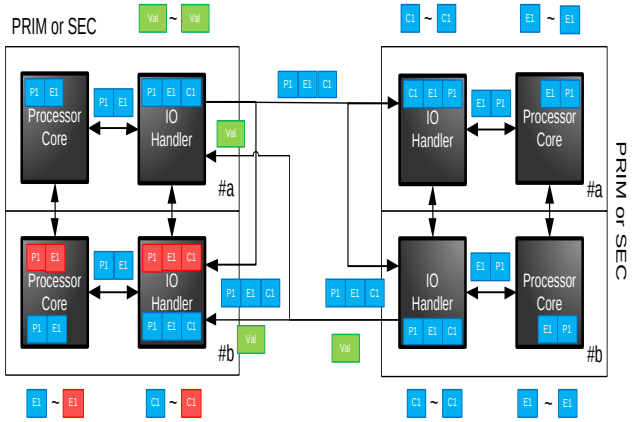

Figure 15. CRC Check in External Communication: the ACP Point of View

Up to now, we have described the architecture of future fully digital helicopters FCS and presented the use of check field (generated by error detection code like CRC) to enhance communication safety. As listed above, the frame includes two checks, the first one is to protect the whole frame (it is the CRC imposed by EIA-485) and the other check field is to protect exclusively the payload. In this section, we describe our communication integrity approach considering the *payload protection check*, the selection strategy of relevant codes we estimate to be relevant to our targeted system and some experiments to assess the performances of these codes.

End to end integrity approach

The conventional communication integrity approach is to deploy a redundant architecture by over-multiplying links or/and by eventually using oversized error detection codes. Yet, such architectures are prohibitively expensive in terms of computational cost and not amendable to embedded systems. Instead, we propose a lightweight approach using relevant error detection codes to be implemented in the application layer (end-to-end-integrity). We call this way to ensure safety “black channel safety” where the safety is ensured in the application layer (Figure 16) and is independent of the lower layers. This concept has several advantages as revealed in the cyber protection literature. In fact, for a such safety concept, certification is easier. Finally, end-to-end integrity permits to cover residual errors of under-layers.

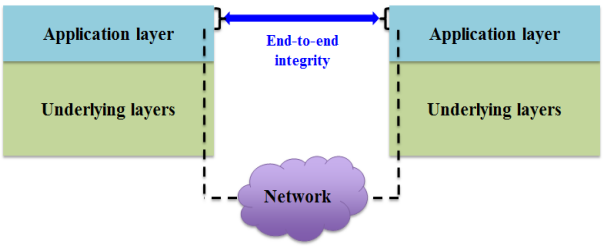

Figure 16. End-to-End Integrity

Our integrity approach is based on a single error detection code. This single code is used in all transmissions to generate the check bits to be added to the message to be sent. Thus, the effectiveness of our integrity approach in terms of error detection capability depends exclusively on this error detection code. So, the code selection is extremely important.

For targeted system, the selection strategy of codes is a trade-off between its error detection capability (which must be good enough to comply with certification requirements) and its computational cost (which must be as low as possible to comply with embedded and real-time constraints).

The approach consisting in using a single error detection code is commonly used in standards and protocols (e.g., CRC-CAN used in the CAN bus). According to [12], many of used codes in protocols and standards are sub-optimal. If not well selected, an error detection code could provide insufficient performances in terms of error detection capability and computational cost.

Unlike conventional use of error detection codes, our alternative communication integrity approach consists in using lightweight codes in terms of computational cost. Therefore, we consider *arithmetic checksums*. For cyclic redundancy check (CRC), we target implementations that fasten the computational time using for example lookup tables.

Relevant errors detection codes

According to related work on communication integrity in safety-critical systems [12] [13], CRC, *Adler and Fletcher codes* are relevant to enhance communication integrity which are common in digital networks. We focus on these codes in order to assess their performances. It is noteworthy that for an error detection code with a size of *n* check bits, the upper bound of the non-detection rate is equal to 2^{-n} for random errors [14]. Since for our targeted system, the non-detection rate should be lower than 10^{-9}, we consider error detection codes with 32 check bits whose the upper bound of non-detection rate is equal to $2.32\ 10^{-10}$. A 32 check bits error detection code is able to meet the required integrity level for our targeted systems.

Related work [13] assessing the performances of Adler, Fletcher and CRC codes proves via simulations that Adler and Fletcher codes have roughly the same computational cost and CRC has the highest computational cost compared to Adler and Fletcher. Given the fact that CRC has a better error detection capability for particular error patterns (e.g., burst errors), we run experiments to assess the computational cost of different implementations of Adler, Fletcher and CRC in order to select relevant ones.

Experiments methodology, environment and scenarios

In order to assess the performances of codes we estimate to be "good candidates" to our targeted systems, we carried out experiments in order to evaluate the computational costs of these codes. We evaluate the WCET (Worst Case Execution Time) of different codes and different implementations. We consider these following codes: i) Fletcher-32; ii) Adler-32 and iii) CRC-32, 32 is the size of check bits.

We consider different implementations, for example, for Fletcher and Adler we consider two different implementations, one is an optimized implementation proposed by [15] for Fletcher that we adopted for Adler too since these two codes are closely similar.

For CRC, there are two implementation philosophies; i) either following a bit by bit computation; ii) or following a byte by byte computation using logical operations or lookup

tables. We are based on two reference platforms: PyCRC [16] and Universal CRC [17].

PyCRC proposes 4 implementations:

- Bit-by-bit (CRC32_bbb): it is the straightforward implementation of the basic polynomial division.
- Bit-by-bit-fast (CRC32_bbf): like the bit-by-bit algorithm, this algorithm still iterates over every bit of the message.
- Table-driven (CRC32_tbl): unlike bit by bit implementation, this algorithm operates on one byte at a time using a look-up table (usually of 256 elements).
- Bitwise-expression (CRC32_bwe): This algorithm uses the same approach as the table-driven variant, but uses logical operations instead of a look-up table.

Universal CRC proposes 7 implementations:

- Bit (CRC32_bit): the basic bit-by-bit algorithm.
- Tab16 (CRC32_tab16): table-driven algorithm with 16 table entries.
- Tab16i (CRC32_tab16i): table-driven using two independent lookups tables with 32 table entries.
- Tab (CRC32_tab): standard table-driven algorithm with 256 table entries.
- Tabw (CRC32_tabw): standard table-driven algorithm, word-at-a-time same as CRC32_tab but reads 32 bits at a time from memory.
- Tabi (CRC32_tabi): table-driven algorithm, four independent lookups (1024 entries) inspired by crc32 algorithm in zlib.
- Tabiw (CRC32_tabiw): table-driven algorithm, four independent lookups, word-at-a-time same as CRC32_tabi but reads 32 bits at a time from memory.

To run experiments, we consider two different evaluation platforms that are commonly used in embedded systems which are:

- P2020 [18]: it has dual high-performance CPU cores, up to 1.33 GHZ with 32 KB L1 and 512KB L2 caches.
- TMS320C6657 [19]: it has a 32 KB L1P cache, a 32 KB L1D data cache, a 1 MB L2SRAM which could serve as a cache and a 1 MB L3SRAM shared memory

We consider two different data generation strategies: the first strategy is random where the error detection code is applied to different data sequences that are generated randomly; the second strategy is rather static, it consists in generating a fixed data sequence, the error detection code is applied only to this data sequence. The objective of considering both random and static generations is to assess the cache phenomenon when using a fixed data sequence. It consists in the hypothesis that lookup tables based algorithms have better performances when the data generation is static.

Results and discussion

As descried in Figure 17, for a static data generation, Adler and Fletcher have lower computational costs than all CRC implementations; this result validates what is said in related work. The optimized implementation of both Adler and Fletcher reduce slightly the computational cost compared to the basic implementations which is a predictable result that validates our implementations. CRC implementations using look-up tables (e.g., CRC32_tab16) have lower computational costs than basic bit-by-bit implementations (e.g., CRC32_bbb), this confirms the fact that lookup tables fasten the computational time of CRC.

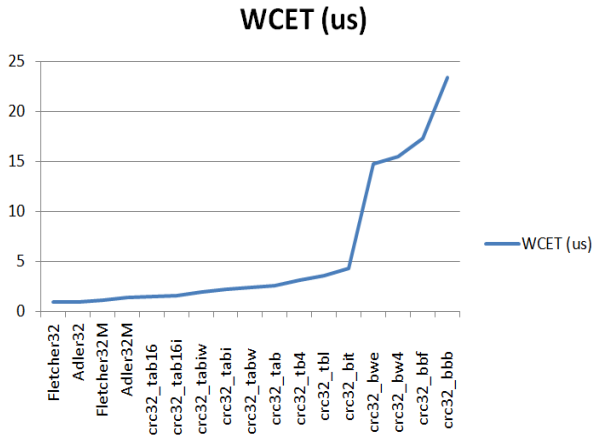

Figure 17. WCET of Different Algorithms in a P2020 Platform with a Static Data Generation

For a random data generation (Figure 18), we notice some results variations compared to the scenario of random data generation (e.g., CRC32_tbl has a lower computational cost) but the main results remain the same. For example, optimized versions of arithmetic checksums have the best performances and bit-by-bit implementations (e.g., CRC32_bbf and CRC32_bbb) have the worst performances.

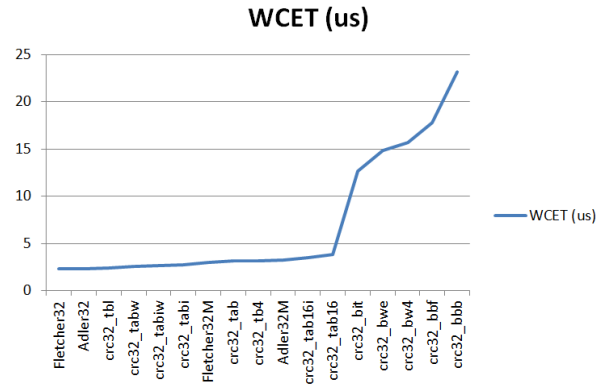

Figure 18. WCET of Different Algorithms in a P2020 Platform with a Random Data Generation

As described in Figure 19, for the TMS evaluation platform and in the context of static data generation, only bit-by-bit implementations of CRC provide higher computational costs compared to Adler and Fletcher. All other CRC implementations provide lower computational costs.

The optimized implementation of both Adler and Fletcher reduce slightly the computational cost compared to the basic implementations as in the two previous scenarios.

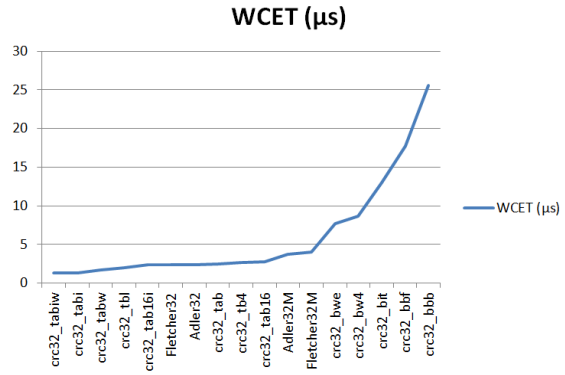

Figure 19. WCET of Different Algorithms in a TMS320C6657 Platform with a Static Data Generation

As described in Figure 20, we notice slight results variations in the context of random data generation. However, main results remain the same. For example, bit-by-bit implementations provide the worst performances as noticed in previous scenarios and both lookup table based CRC algorithms and arithmetic checksums provide interesting performances.

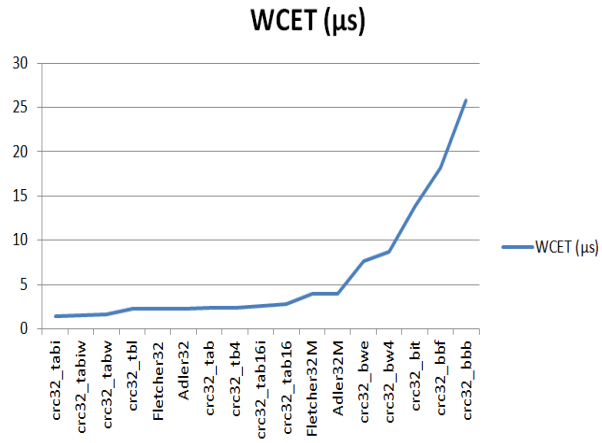

Figure 20. WCET of Different Algorithms in a TMS320C6657 Platform with a Random Data Generation

Conclusive remarks and guidelines

We present here some conclusive remarks and guidelines we propose in order to use efficiently error detection codes in safety-critical embedded systems:

- The computational cost depends on the considered platform
- The bit-by-bit CRC algorithms is not relevant to time-critical systems since they are heavy in terms of computational costs
- Using look-up tables reduce the computational cost of CRC and make it relevant to embedded systems
- Some look-up tables based CRC algorithmes provide more interesting computational costs compared to Adler or Fletcher codes
- For systems with high constraints that do not permit to use look-up tables, using Fletcher or Adler is a good trade-off
- As described in [20], different diversification strategies could be used in order to avoid some particular errors

particularly premanent errors. To deal with such errors, we can either use a set of complementary error detection codes (in the same frame or one code per a frame) in order to increase the overall error detection capability or diversify the data to be sent using re-expression functions.

Conclusion

The FBW evolution and particularly the goal to move to fully digital FBW for future helicopters FCS raise new challenges. Future designs should take into account different safety requirements imposed by certification standards. Being a distributed system relying on digital networks, communication integrity in FCS is a main issue. Therefore, this paper describes the safety-sensitive architecture of future fully digital Airbus Helicopters FCS considered at the end of feasibility study. It proposes an end-to-end integrity approach based on error detection codes assessing the performances of CRC, Adler and Fletcher codes in realistic environment using two evaluation platforms. Moreover, this paper proposes some guidelines for better using error detection codes in safety-critical embedded systems.

References

[1] ARP4754A, 2010, Guidelines for Development of Civil Aircraft and Systems, Aerospace Recommended Practice published by SAE International.

[2] ARP4761, 1996, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, Aerospace Recommended Practice published by SAE International.

[3] Federal Aviation Administration, System Safety Handbook – chapter 3: Principles of System Safety, 19 pages, Dec. 2000. Available at: http://www.faa.gov/regulations_policies/handbooks_manuals/aviation/risk_management/ss_handbook/media/Chap3_1200.pdf.

[4] Federal Aviation Administration, Helicopter Flying Handbook – chapter 3: Helicopter Flight Controls, 10 pages. Available at: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/helicopter_flying_handbook/media/hfh_ch03.pdf.

[5] Traverse P., I. Lacaze and J. Souyris, 2004, Airbus Fly-By-Wire: A Total Approach to Dependability, Proceedings of the 18th IFIP World Computer Congress (WCC 2004), Building the Information Society, Toulouse, France, pp. 191-212.

[6] Traverse P., I. Lacaze and J. Souyris, 2006, Airbus Fly-By-Wire: A Process Toward Total Dependability, Proceedings of the 25th International Congress of the Aeronautical Sciences (ICAS 2006), Hamburg, Germany.

[7] Boczar, B. and B. J. Hull, 2004, S-92 Fly-by-Wire Advanced Flight Control System, Proceedings of the 60th American Helicopter Society Annual Forum, Baltimore, Maryland, USA, pp. 404-424.

[8] Vidal, P-A., E.G.J. Woirin, J-M. Massimi, P.L. Ressent, 2000, Flight Control Device for an Aircraft, in particular a Helicopter, U.S. Patent 6 059 225.

[9] Yeh, Y. C., 1998, Design Considerations in Boeing 777 Fly-By-Wire Computers, Proceedings of the 3rd IEEE International High-Assurance Systems Engineering Symposium (HASE'98), Washington, D.C, USA, pp. 64-72.

[10] Yeh, Y.C., 2001, Safety Critical Avionics for the 777 Primary Flight Control System, Proceedings of the 20th IEEE Conference on Digital Avionics Systems (DASC 2001), Daytona Beach, Florida, USA, pp. 1.C.2.1-1.C.2.11.

[11] Vidal, P. A., 1997, NH90 Helicopter Fly-By-Wire Flight Control System, Proceedings of the 53th American Helicopter Society Annual Forum,Virginia Beach, Virgiana, USA, pp. 915–923.

[12] Koopman, P. and T. Chakravarty, 2004, Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks, Proceedings of the International Conference on Dependable Systems and Networks (DSN 2004), Florence, Italy, pp. 145-154.

[13] Maxino, T., and P. Koopman, 2009, The Effectiveness of Checksums for Embedded Control Networks, IEEE Transactions on Dependable and Secure Computing, vol. 6, no. 1, pp. 59-72.

[14] Paulitsch, M., J. Morris, B. Hall, K. Driscoll, E. Latronico and P. Koopman, 2005, Coverage and the use of Cyclic Redundancy Codes in Ultra-Dependable Systems, Proceedings of the International Conference on Dependable Systems and Networks (DSN 2005), Yokohama, Japan, pp. 346-355.

[15] Kodis, J., 1992, Fletcher's Checksum, Dr. Dobb's Journal, Volume 17, Issue 5, pp. 32-38. Available at:

http://www.drdobbs.com/database/fletchers-checksum/184408761.

[16] Pircher, T., 2014, pycrc Enviroment, version 0.84. Available under MIT licence at: http://www.tty1.net/pycrc/index_en.html

[17] McGougan, D., 2011, Universal CRC Environment, version 1.3a. Available under GPL licence at: http://www.mcgougan.se/universal_crc/

[18] Freescale, 2009, P2020 Communication Processor, description available at: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=P2020

[19] Texas Instruments, 2012, Digital Signal Processor, description available at: http://www.ti.com/product/TMS320C6657/description

[20] Zammali, A., A. de Bonneval and Y. Crouzet, 2015, A Diversity-Based Approach for Communication Integrity in Critical Embedded Systems, Proceedings of the 16th IEEE International High-Assurance Systems Engineering Symposium (HASE 2015), Daytona Beach, Florida, USA, pp. 215-222.