DEVELOPMENT OF ANGLE OF ATTACK (AOA) / STALL WARNING COMPUTER FUNCTIONS OF A COMBINED AIR DATA AND AOA COMPUTER

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF ÇANKAYA UNIVERSITY

BY

MEHMET MUSTAFA KARABULUT

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRONICS AND COMMUNICATIONS ENGINEERING

SEPTEMBER, 2011

Title of the Thesis: **Development of Angle of Attack (AOA) / Stall Warning Computer Functions of a Combined Air Data and AOA Computer**

Submitted By: **Mehmet Mustafa KARABULUT**

Approval of the Graduate School of Natural and Applied Sciences

[illegible]

Prof. Dr. Taner ALTUNOK

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Celal Zaim ÇİL

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Celal Zaim ÇİL

Supervisor

Examination Date : 16.09.2011

Examining Committee Members:

Prof. Dr. Celal Zaim ÇİL	(Çankaya Univ.)	[illegible]
Assistant Prof. Dr. Klaus SCHMIDT	(Çankaya Univ.)	[illegible]
Assistant Prof. Dr. Altan ÖZKİL	(Atılım Univ.)	[illegible]

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name	: Mehmet Mustafa KARABULUT
Signature	: [illegible]
Date	: 23.12.2011

ACKNOWLEDGEMENTS

The author wishes to express his deepest gratitude to his supervisor Prof. Dr. Celal Zaim ÇİL for his guidance, advice, criticism, encouragements and insight throughout the research.

The author would also like to thank Rafet ILĞIN for his suggestions and comments.

ABSTRACT

DEVELOPMENT OF ANGLE OF ATTACK (AOA) / STALL WARNING COMPUTER FUNCTIONS OF A COMBINED AIR DATA AND AOA COMPUTER

KARABULUT, Mehmet Mustafa
M.Sc., Department of Electronics and Communication Engineering
Supervisor: Prof. Dr. Celal Zaim ÇİL

September 2011, 36 Pages

In this thesis, the Angle of Attack (AOA part) of a combined air data system (CADS), and the CADS software are developed on a standard PC and without real interface. In its current form, a CADS system on an aircraft is composed of two different equipments, one is the ADC and the other is the AOA system. Therefore the developed CADS system combines both functionalities in an integral manner on a card. This way the volume and cost of the CADS system are reduced.

Keywords: Avionic, System Integration, Flight Simulation, Software, Air Data Calculation

ÖZ

TÜMLEŞİK HAVA VERİ VE HÜCUM AÇISI BİLGİSAYARI HÜCUM AÇISI VE PERDÖVİTES UYARISI FONKSİYONLARI HESAPLAMALARI GELİŞTİRMELERİ

KARABULUT, Mehmet Mustafa
Yüksek Lisans, Elektronik ve Haberleşme Mühendisliği Anabilim Dalı
Tez Yöneticisi: Prof. Dr. Celal Zaim ÇİL

Eylül 2011, 36 Sayfa

Bu tez çalışmasında, standart bir bilgisayar üzerinde gerçek arayüzler kullanılmadan Tümleşik Hava Veri Sistemi' nin Hücum Açısı Bilgisayarı tasarlanmış ve Tümleşik Hava Veri Sistemi Yazılımı geliştirilmiştir. Hava araçlarında Tümleşik Hava Veri Sisteminin görevini yerine getirebilmek için iki adet farklı donanım kullanılmaktadır. Bu ekipmanlardan bir tanesi Hava Veri Bilgisayarı, bir diğeri de Hücum Açısı Bilgisayarıdır. Bu nedenle, tasarlanmış tümleşik sistem gerekli fonksiyonları tek bir kartta gerçekleştirdiği gibi hacimden ve maliyetten de kazandırmaktadır.

Anahtar Kelimeler: Aviyonik, Sistem Entegrasyonu, Uçuş Simülasyonu, Yazılım, Hava Veri Hesaplamaları

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADC - air data computer

ANSI - american national standards institute

ARINC - aeronautical radio incorporated

AS - aerospace standard

A/C - aircraft

CADS - combined air data system

CD - compact disc

C_L - lift coefficient

CPU - central processing unit

CVI - c for virtual ınstrumentation

DSP – digital signal processor

EMI/EMC - electromagnetic interference/electromagnetic compatibility

FAA - federal aviation agency

FPGA - field programmable gate array

FSX - flight simulator x

GPS – global positioning system

IAS - indicated airspeed

IDE - integrated development environment

IFF - identification friend or foe

IMA - integrated modular avionics

INS - inertial navigation system

kbps – kilo bits per second

LRU - line replaceable unit

MFD – multi functional display

NATO - north atlantic treaty organization

OS - operatıng system

PC - personal computer

RAM – random access memory

RTCA - radio technical commission for aeronautics

RTOS - real time operating system

SAE - society of automotive engineers

SC - simulation computer

SDL - software development laboratory

SIL - system integration laboratory

SWCI - software configuration items

TACAN -	tactical air navigation
TAS -	true airspeed
TAT -	total air temperature
TSO -	technical standard order
TSV -	thread-safe variable
VHF -	very high frequency

CHAPTER 1

INTRODUCTION

Avionics is a combined word which is arisen from the integration of “aviation” and “electronics”. Although this term has got mature in the 20’th century, the history of “aviation” dates back to the 18th century. World War II and the years of Cold War years revealed the need for scientific research and technology development. These researches and developments have resulted in a huge growth in avionics. Nowadays, avionics systems become key components of aircraft.

The weather conditions were the main factor affecting the air travels in early times of the first quarter of the 20’th century. It was very difficult to travel in air in closed weather. Fog, darkness, rain or snow were nightmares for the pilots and they were the main candidates to cause fatal accidents.

The US Army, which is considered as the ancestor of modern avionics, was so concerned about its military operations which may be ceased because of harsh weather conditions. Therefore, it started to make an investigation about the basic information needed to make a safe flight under any weather conditions. The first parameter found in this investigation was altitude. An altimeter of sufficient accuracy was critical for a pilot to avoid unseen mainland and for a safer landing. The second parameter was the reference to the natural horizon. When fog, clouds or other precipitation obscure the earth’s horizon, a pilot cannot depend on his senses to differentiate up from down. If you stand on the earth, you can easily tell which way is “up”, even with eyes closed because you feel the force of gravity on your body. In an aircraft, there are a lot of several forces acting on the body. Only if the pilot sees the natural horizon can he avoid from being fooled by the force developed as the airplane turns. Therefore, the second requirement was the natural horizon to provide visual references to pilot to control the airplane. The final parameter was the radio for voice communications. It was very important to make a voice communication between pilots and air control stations.

After these primitive works to resolve the issues related to the parameters defined above, the avionics sector gained a big acceleration. In the first quarter of the 20’th century, the first electronic aids were introduced such as non-directional beacons, ground-based surveillance radar, and the single-axis autopilot. In the second quarter of the 20’th century, the development in this domain got further acceleration, and very high frequency (VHF) communications, identification friend or foe (IFF), gyro compass, attitude and heading reference systems, airborne intercept radar, early electronic warfare systems, military long-range precision radio navigation aids and the two-axis autopilot were introduced. The third quarter of the 20’th century was even more brilliant in terms of developments. Tactical Air Navigation (TACAN), Doppler radar, terrain-following radar, Mission Computer and Inertial Navigation System (INS), integrated electronic warfare systems were brought to the flight sector and integrated into the airborne platforms.

In the past, nearly all of the avionics architecture on the flying platforms was point-to-point. In point-to-point avionics architecture, all of the data communications between the sensor and the control unit or indicator in the cockpit were made between source equipment and the destination; there was no bus controller or bus master. This method had disadvantages in terms of mainly cabling, power consumption, space consumption and weight. Also it was very difficult to make any modification to the system if necessary. Approximately each system in this architecture had its dedicated subsystems, control panels and displays. The displays were electromechanical and prone to break down. Also the use of analog computing techniques did not provide the accuracy and stability offered by the digital systems introduced later.

After a while, when digital computing devices got mature enough and suitable for airborne use, distributed digital avionics had arisen. This led to the adoption of digital computers to avionics sector which resulted in greater speed of computation, accuracy and removal of bias and drift problems which arise in analog systems. In the distributed digital avionics architecture, major functional units contained their own digital computer and memory. Displays in the cockpit were dedicated to their function as for the analog architecture. The displays were still electromechanical devices used previously, with the known problems. In later implementations the displays become multifunctional and multicolor. Also the “data bus” term started to be used with distributed digital avionics. The well-known and already used data bus “Aeronautical Radio Incorporated (ARINC) 429” was introduced at these times. Data bus has offered a great deal of flexibility in the signal transmission and led to reduction in wiring. This, in turn, led to a reduction in weight, power consumption and cost.

Then, federated digital architectures, in which there is at least one bus controller/master came into the avionics world. With this new methodology, the avionics architecture became safer and upgradeable. In principle, federated architecture relied upon the availability of the extremely widely used military-standard MIL-STD-1553 data bus. The adoption of the 1553 data bus standard offered significant advantages and some drawbacks. One advantage was that this standard could be applied across all North Atlantic Treaty Organization (NATO) members, offering a data bus standard across a huge market. The federated architectures generally use dedicated 1553B-interfaced equipment (line replaceable units - LRUs) and subsystems, and are more robust and reliable than the preceding architectures. The disadvantage of federated avionics architecture is that the LRUs working with 1553B protocol is generally used in military platforms. Therefore federated architecture is not used widely in commercial aircrafts.

Nowadays, aircraft manufacturers are trying to make smaller and smaller aircrafts, or more functionality in less space, which means that there will be fewer places for the avionics equipment. Therefore, the Integrated Modular Avionics (IMA) concept was developed. With this concept, "equipment on a card" designs were arisen. By this way, instead of installing two or more devices for redundancy purposes, the integrator firms use two cards in a black box by sharing ruggedness of the hardware, cooling, data bus and electrical power infrastructure of “the box”.

In this thesis, it is aimed to make the AOA functionality of a Combined Air Data System (CADS) as preliminary work for this IMA concept. AOA Computer is very important equipment in avionics architecture. The basic and necessary parameters for a safe flight are calculated by the AOA Computer.

The AOA parameter has a crucial role especially in take-off and landing phases of the flight. It can be considered as the angle between the chord line of the wing and the vector representing the relative motion between the lifting body and the fluid through which it is moving [1].

The source of the AOA parameter is the AOA sensor. The sensed AOA value from the AOA sensor is sent to the AOA computer, in which the aircraft specific values/charts are stored. The stall is a reduction in the lift coefficient generated by an airfoil as angle of attack increases. This occurs when the critical angle of attack of the airfoil is exceeded. The critical angle of attack is typically about 15 degrees, but it may vary significantly depending on the airfoil [2].

The information calculated by the AOA system is essential for the pilot to fly the aircraft safely, and is also required by a number of key subsystems in the aircraft such as pilot control panel, audio warning panel, Multi Function Display (MFD) and etc. Since the AOA parameter is very critical for a safe flight, it should be timely calculated (real time) at a required rate. The real time concept refers that if latency or loss occurs in measuring, calculation or delivering etc. of certain information, it may bring a serious error in completing a task, a mission properly and on time and may even sometimes be so critical that the consequences can be fatal. Hence, the CADS software has to be considered as a real time system. Real-time systems are defined as those systems in which the overall correctness of the system depends on both the functional correctness and the timing correctness. The timing correctness is at least as important as the functional correctness.

In this thesis, AOA system field research with CADS software development is performed. CADS software is developed on a standard personal computer (PC) and without real interface. In its current form, the functionality provided by the CADS software is being provided by two different equipment, one is the Air Data Computer (ADC) and the other is the AOA system. Therefore CADS software combines both these functionalities in an integral manner on a card. This brings reduction in the volume and cost. The hardware is implemented on a commercially available PCI card that can be inserted in a motherboard of a PC.

CHAPTER 2

THE FUNCTION OF AOA IN THE COMBINED SYSTEM

The AOA Computer is very important equipment in avionics architecture. There are some mandatory standards for this equipment to be installed on aircrafts. These are basically:

- Federal Aviation Agency (FAA) Technical Specification Order (TSO) C54 for equipment's functionality

TSO-C54 is a minimum performance standard. It is established for stall warning instrument which specifically is required to be approved for use on civil aircraft of the United States. New models of stall warning instruments manufactured for installation on civil aircraft on or after October 15, 1961, shall meet the standards as set forth in Society of Automotive Engineers (SAE) Aeronautical Standard (AS) 403A, “Stall Warning Instrument”, revised July 15, 1958 [3].

The general avionics architecture around an AOA computer is given in Figure 1.

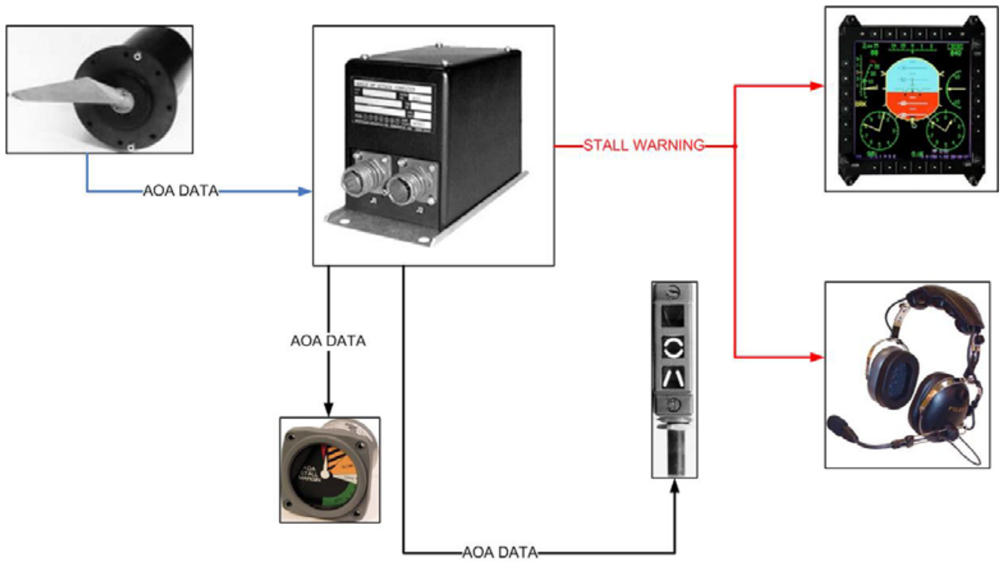


Figure 1 AOA in Typical Avionics Architecture

SAE 403A defines the performance of the designed equipment. It also specifies the must and optional parameters which will be given by the designed equipment and their accuracies.

- Radio Technical Commission For Aeronautics (RTCA)/Design Order (DO)-160 for equipment's environmental stamina

DO-160, "Environmental Conditions and Test Procedures for Airborne Equipment is a standard for environmental test of avionics hardware published by RTCA, Incorporated.

This document outlines a set of minimal standard environmental test conditions (categories) and corresponding test procedures for airborne equipment. The purpose of these tests is to provide a controlled (laboratory) means of assuring the performance characteristics of airborne equipment in environmental conditions similar of those which may be encountered in airborne operation of the equipment. The standard environmental test conditions and test procedures contained within the standard may be used in conjunction with applicable equipment performance standards, as a minimum specification under environmental conditions, which can ensure an adequate degree of confidence in performance during use aboard an air vehicle [4].

There are two (2) basic parameters calculated by the AOA Computer. These are:

- Angle of Attack
- Stall Warning.

To calculate these parameters the AOA computer needs information from the AOA vane. The AOA vane is installed by considering the position of the propeller of the aircraft (A/C) (if the A/C has a propeller). If the propeller is on the nose of the A/C the vane is installed on the wing of the A/C to exclude the propeller effect. If the propeller is on the wing of the A/C, then the vane is installed on the side surfaces of the A/C. If the A/C has its ignition power at its back like in fighter jets, then the probe is generally installed on the nose of the A/C. The number of the AOA vanes basically depends on the avionics architecture, and the redundancy of the AOA information. Generally, there is one AOA vane for one AOA computer in smaller A/Cs.

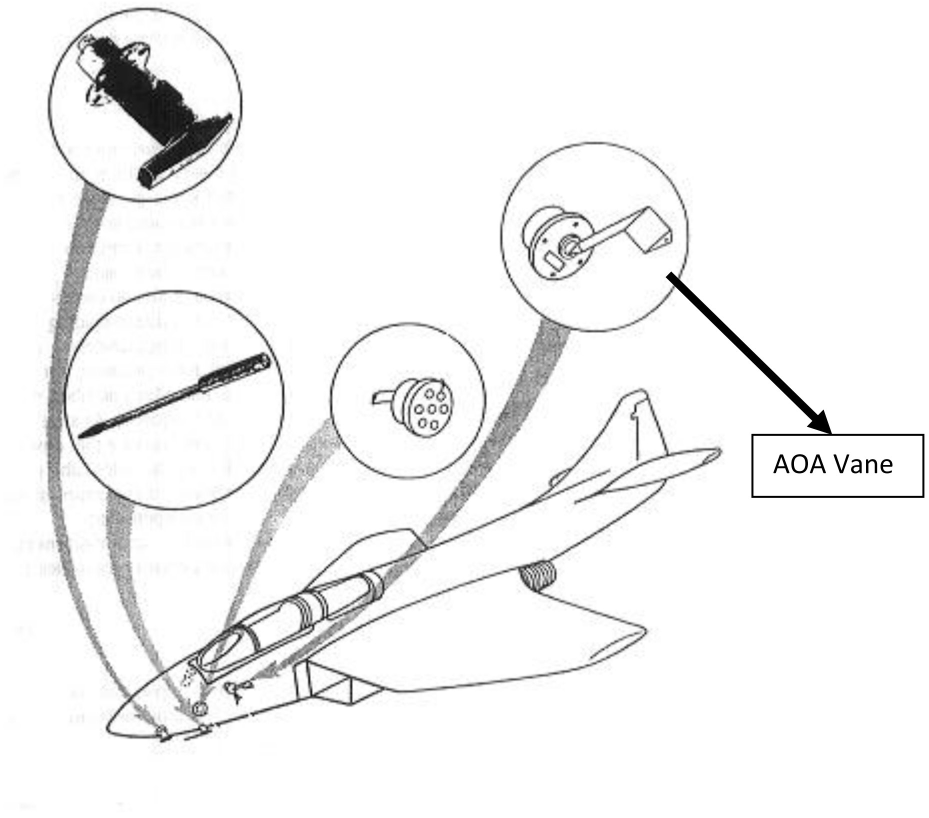


Figure 2 Typical Sensor Installation on a Jet Fighter [5]

The AOA is described as the angle between chord line of an airfoil and the vector representing the relative motion between the lifting body and the fluid (air) through which it is moving.

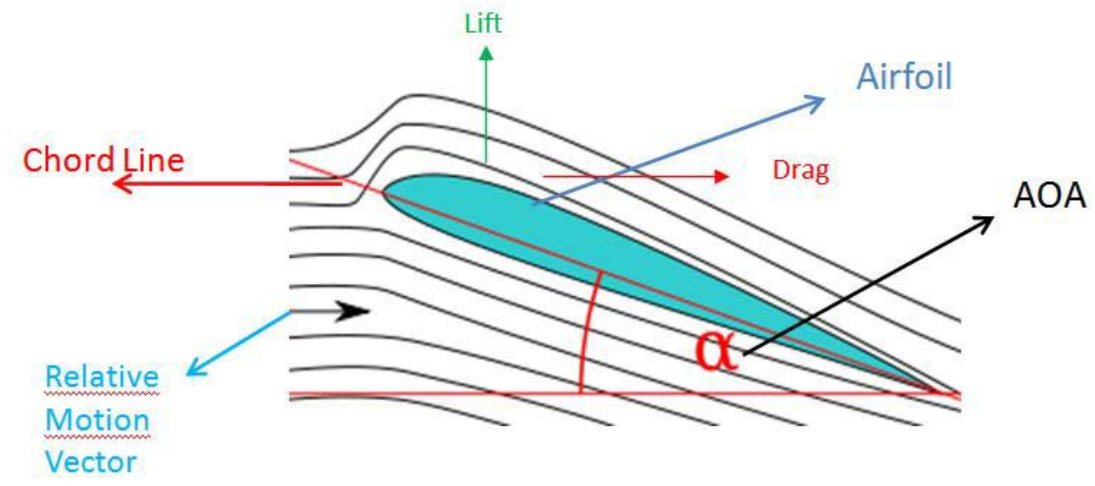


Figure 3 Forces on an Airfoil [1]

In Figure 3, an airfoil is the shape of a wing when looked at cross sectional view. The chord line is a straight line connecting the leading and trailing edges of the airfoil, at the ends of the mean camber line. Lift is defined to be the component of the force that is perpendicular to the relative motion vector. Drag is the total of the forces that oppose the relative motion of an object through a fluid.

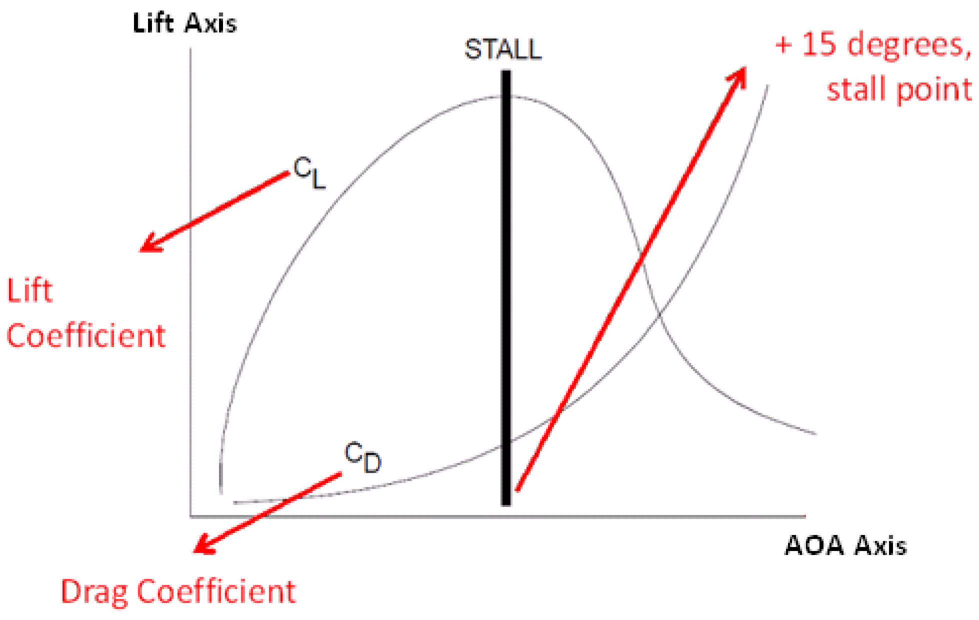


Figure 4 AOA versus Lift

As seen in Figure 4, Stall is a condition that the AOA exceeds a certain value after which lift starts to decrease. The corresponding AOA value from which the lift is decreasing is A/C-specific and is called the critical angle of attack (stall point in Figure 4). This critical angle is dependent upon the profile of the wing, its planform, its aspect ratio, and other factors, but is typically in the range of 8 to 20 degrees relative to the incoming wind for most subsonic airfoils. Stall is the peak point of the AOA in Figure 4. The maximum lift coefficient occurs at the critical angle of attack point [2].

C_L may be used to relate the total lift generated by an aircraft to the total area of the wing of the aircraft. In this application it is called the lift coefficient C_L [6].

$$C_L = \frac{L}{\frac{1}{2}\rho v^2 K} = \frac{2L}{\rho v^2 K} = \frac{L}{qK} \tag{2.1}$$

L is the lift force, *ρ* is the fluid density, *υ* is the true airspeed, *q* is the dynamic pressure, and *K* is the planform area. A planform is the shape and layout of a fixed-wing aircraft's fuselage and wing. Figure 5 shows the planform of Airbus A340-600.

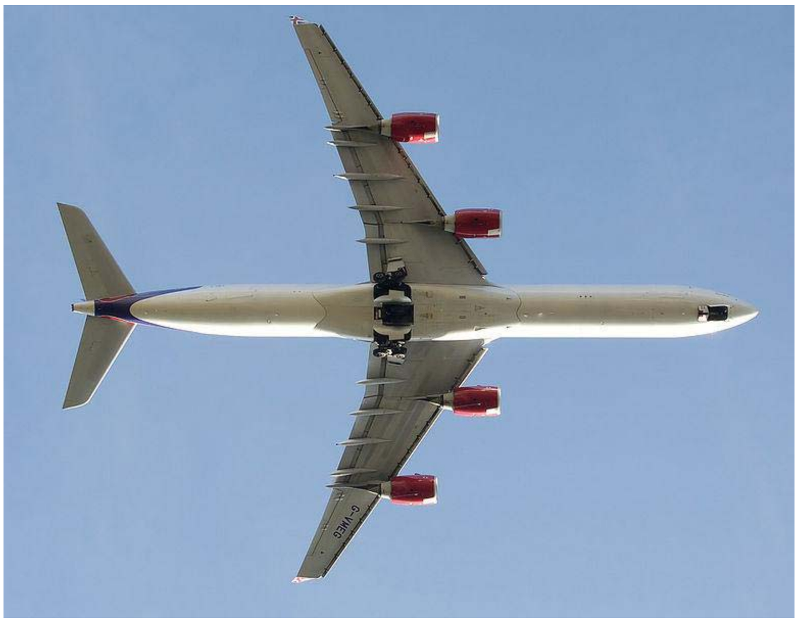


Figure 5 Planform of Airbus A340-600 [7]

Drag coefficient is the indication of the resistance to an object (drag) in air. Lower drag coefficient means the object will have less aerodynamic resistance to the flow through the air. The drag coefficient is associated with a particular surface area. The drag coefficient c_d is defined as

$$c_d = \frac{2F_d}{\rho v^2 K} \tag{2.2}$$

Fd is the drag force [8].

CHAPTER 3

DETAILED DESCRIPTION OF THE SYSTEM DEVELOPED

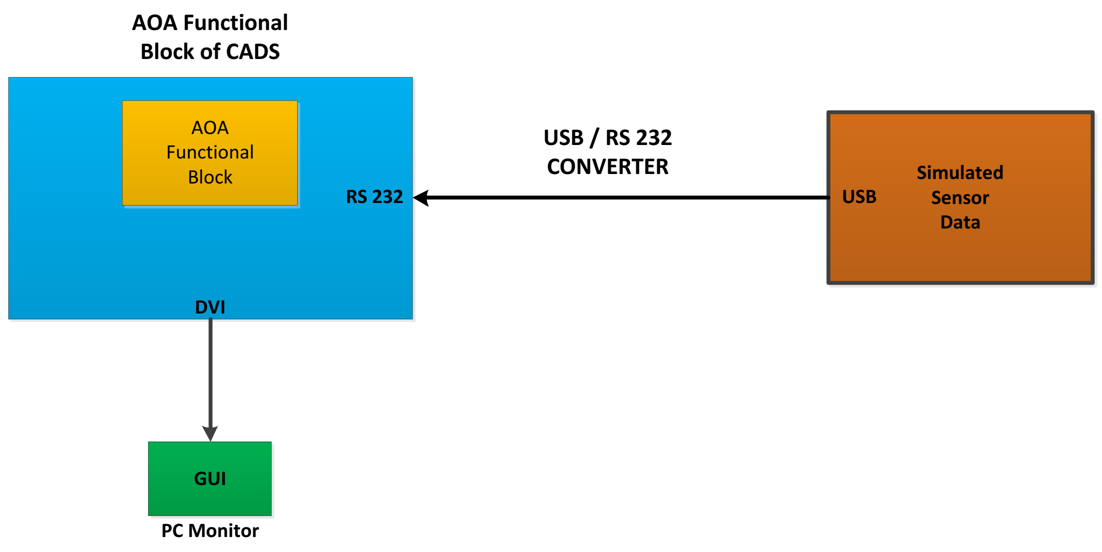


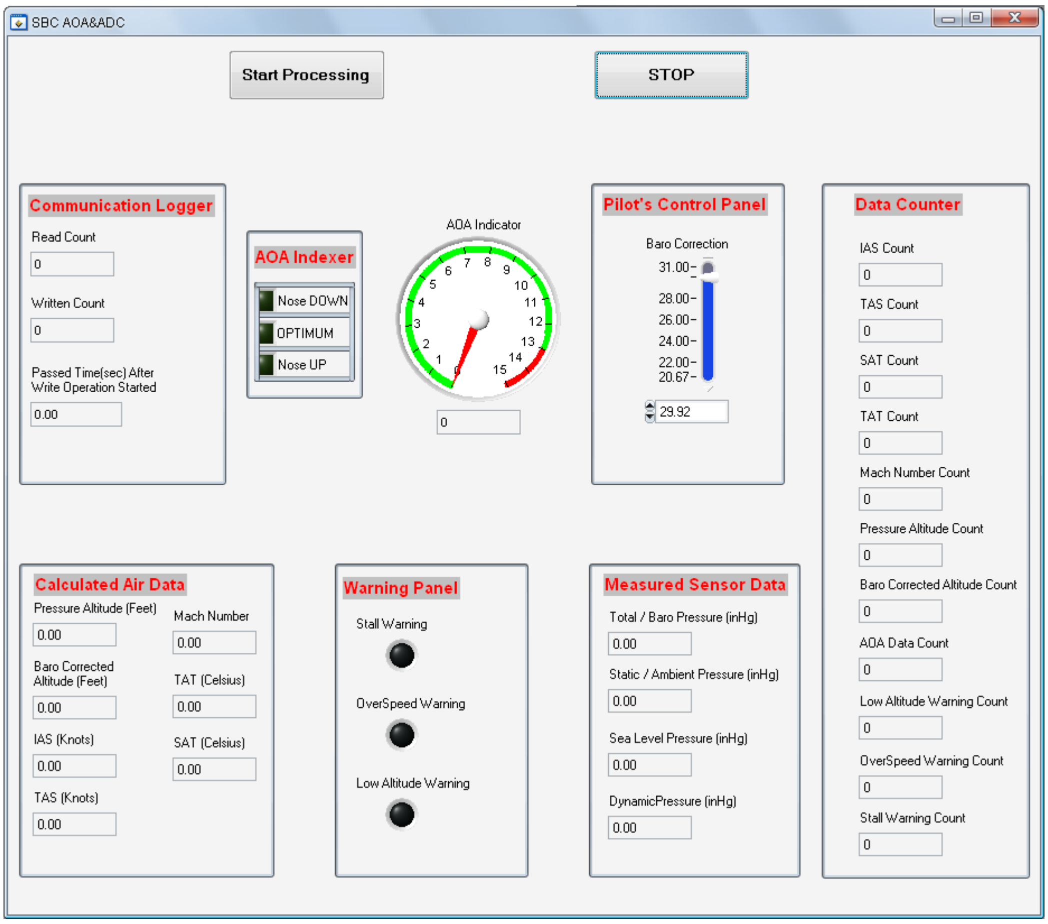
Figure 6 Demonstration of the System Architecture

In this thesis a CADS is developed on a card. This CADS combines the ADC and AOA computers on a card. This way of combining these two subsystems on a card is the improvement introduced. First of all, the CADS is developed on the card. Then, in order to test and verify the system developed, simulation computer (SC) is developed, which provides the necessary inputs, simulated in accordance with a flight scenario for a certain generic platform, to the CADS.

The CADS is the computer which gets the simulated sensor data over serial port, calculates the required variables and displays them by a refresh rate of at least 8 times in a second. In order to achieve this mission there is a formed infrastructure including the computer hardware, installed operating system, installed Integrated Development Environment (IDE) aligned with a program developed. Also in the program developed there are some special structures used for accomplishing the mission. In this chapter all of the contents under the infrastructure of this system will be defined.

The PC card which is used as CADS, has a central processing unit (CPU) of Intel Pentium 4. The CADS will communicate with the SC via RS 232 serial communication protocol. The operating system (OS) that is chosen for the CADS is Windows XP. The details of the OS could be reached in Appendix A1. The application software is developed with the C programming language.

As can be seen from Figure 7 if the Start Processing button is pressed, then the CADS starts functioning. It continuously listens to the serial port and when data arrives it reads the data from serial port buffer into the message structure mentioned before. After the reading threads fill in the message structure which is a thread safe variable, it does not reread from the serial port into the message structure until calculation threads get the related data from the message structure. After calculation threads receive their parameter calculation related variable from the message structure, reading thread rereads the new data from serial port and refreshes the message structure. Then calculation threads again get the related variables and this continues on until Stop button is clicked. For this process continue in synchronization, there is another controlling thread named ReadControlThread. This controlling function makes the value of another thread safe variable called tsvReadControl 0. The value of the thread safe variable becomes 0 when all the calculating parameter functions accessed the message structure and read the related simulation variable. Therefore it is the job of the synchronization function ReadControlThread. Reading thread checks the value of tsvReadControl, when its value is set to 0, new data from the serial port is read to the message structure and the tsvReadControl is set to 1. By this logic read synchronization is thread safely achieved.



Communication Logger
Read Count: 0
Written Count: 0
Passed Time(sec) After Write Operation Started: 0.00

AOA Indexer
Nose DOWN
OPTIMUM
Nose UP

AOA Indicator: 0

Pilot's Control Panel
Baro Correction: 29.92

Data Counter
IAS Count: 0
TAS Count: 0
SAT Count: 0
TAT Count: 0
Mach Number Count: 0
Pressure Altitude Count: 0
Baro Corrected Altitude Count: 0
AOA Data Count: 0
Low Altitude Warning Count: 0
OverSpeed Warning Count: 0
Stall Warning Count: 0

Calculated Air Data
Pressure Altitude (Feet): 0.00
Mach Number: 0.00
Baro Corrected Altitude (Feet): 0.00
TAT (Celsius): 0.00
IAS (Knots): 0.00
SAT (Celsius): 0.00
TAS (Knots): 0.00

Warning Panel
Stall Warning
OverSpeed Warning
Low Altitude Warning

Measured Sensor Data
Total / Baro Pressure (inHg): 0.00
Static / Ambient Pressure (inHg): 0.00
Sea Level Pressure (inHg): 0.00
DynamicPressure (inHg): 0.00

Figure 7 GUI and Working Principle of the CADS

Another place that uses synchronization is the display of calculated parameters. The calculated parameters are displayed only if all the calculating threads completed calculation and the calculated parameters (processed data) are ready to be filled into the structure which is also a thread safe variable named tsvReadControl. The thread type structure displaying the calculated parameters in the combined system is called asynchronous timer. Asynchronous timers are threads that are executed in given periods of time. In the combined system since there is a limit on refresh rate such that at least 8 times in a second, the asynchronous timer structure is used. Therefore it supplies the refresh criteria for a period 0.0625 seconds.

For detailed examination of the CADS's program architecture, see the code in Appendix A4.

CHAPTER 4

CADS VERSUS REAL EQUIPMENT

4.1. Real AOA System

As mentioned in the previous chapters, there are some standards to be complied with when designing the equipment. These standards can be grouped as:

- Functionality and Performance Standards (TSO C54, SAE AS 8014, SAE AS 403, SAE AS 8046)
- Environmental Standards (RTCA/DO-160).

4.1.1. Functionality and performance standards

Equipment manufacturers are trying to comply with TSOs and SAEs related with the ADC. Since TSOs are giving reference to SAE AS, it is important to understand what SAE AS 403 is.

SAE AS 403 specifies minimum requirements for stall warning instruments for use in aircraft. It defines the required performance of the equipment relating it with environmental conditions. It also states that the warning should be a continuous warning during conditions under which warning is required.

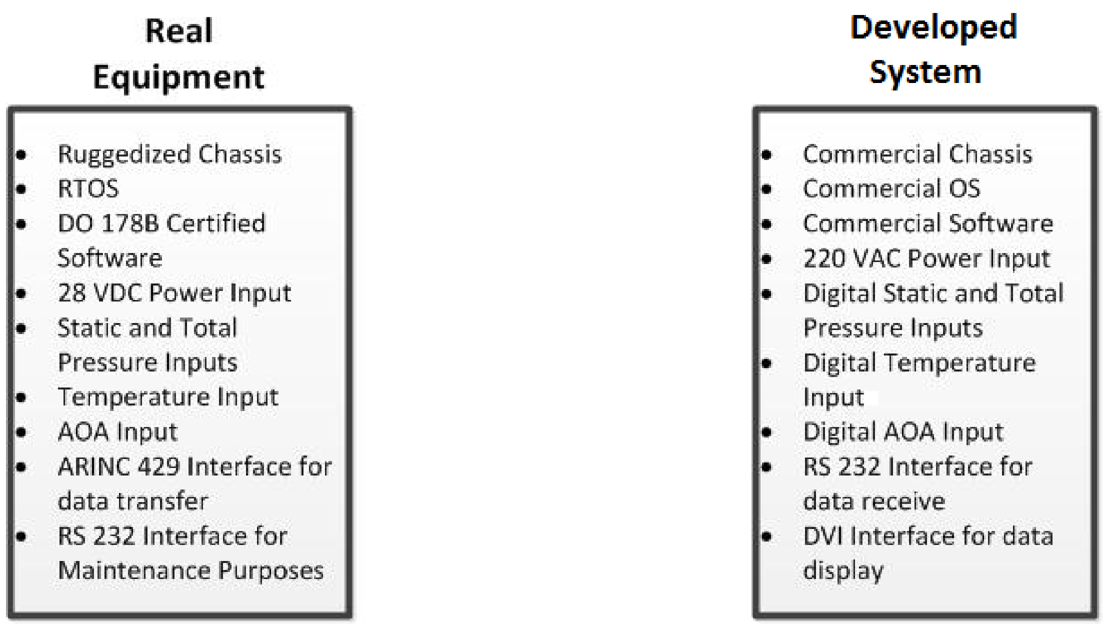
4.1.2. Environmental standards

Functionality and performance standards define the minimum performance under environmental conditions and when defining the performance it references RTCA – DO 160.

4.2. Differences between the Developed and the Real Equipment

- The system developed within the framework of this thesis provides all of the mandatory parameters given by the real equipment.
- The developed system provides whole parameters with a refresh rate of 8 in a second.
- The developed system works with 220 VAC power however the real equipment works with 28 VDC.
- The developed system works with commercial operating system software however real equipment needs to work with a real time operating system.
- Real equipment is qualified to work properly in certain environmental conditions however the developed system is established on a commercial desktop PC chassis which is not ruggedized.

All these comparisons are summarized in Figure 8.



Real Equipment

- Ruggedized Chassis
- RTOS
- DO 178B Certified Software
- 28 VDC Power Input
- Static and Total Pressure Inputs
- Temperature Input
- AOA Input
- ARINC 429 Interface for data transfer
- RS 232 Interface for Maintenance Purposes

Developed System

- Commercial Chassis
- Commercial OS
- Commercial Software
- 220 VAC Power Input
- Digital Static and Total Pressure Inputs
- Digital Temperature Input
- Digital AOA Input
- RS 232 Interface for data receive
- DVI Interface for data display

Figure 8 Real Equipment versus the CADS developed

CHAPTER 5

GENERAL VIEW OF THE CADS SYSTEM SOFTWARE DESIGN

This part contains the software architecture of the system. Firstly general software structure will be briefly explained. In accordance with the system design, the CADS system software gets simulated sensor data via a serial port.

The communication is composed of simulation variables sent to the CADS computer. One of the specifications is that the CADS system software should guarantee that each calculated parameter must be updated at least eight times in a second. In order to achieve this, serial port communication settings have a high importance. Another issue affecting the serial port communication settings is the data size to be transferred. The system's message structure determines the data size. The message structure composes only the simulation variables. For simulation variables used data types are double and int. In Figure 9 there are the structures of the messages which contain the transferred simulation variables and its data types with the reasons why these data types are selected. The name of the structure is "SensorData". After it is defined in the code, it becomes a data type as int, float, char and etc.

```
//Message Structure Data Type(Data type for read simulation
variables)

typedef struct SensorDatum

{       double TotalPressure;

        double StaticPressure;

        double SeaLevelPressure;

        double DynamicPressure;

        double TotalAirTemperature;

        double KohlsmanSetttingHg;

        int LandingGear;

        int AoaData;

}SensorData;
```

Figure 9 Message Structure Data Type for Reading Simulation Variables

As one can observe from Figure 9, data other than discrete signals (Landing Gear position and AOA Data) are defined as double data type. Discrete signals are decided to be implemented as int data types, which are 4-byte (32 bit) data types used for representing decimal numbers. Int data type can be examined in Appendix A5. The simulation variables defined in double data type could also be defined as float but this is not chosen.

Double is an 8-byte (64 bit) data type which is used to represent real numbers. Float is another data type used for representing real numbers, but float is a 4-byte (32 bit) data type. Therefore, by using double instead of float, larger numbers with much more precision can be expressed. Also as the number to be stored/displayed gets bigger in decimal part or floating part, float data type spoils earlier with respect to double data type. In order to prevent data loss and corruption double data type is used. As it can be observed that double data type allocates more region than float data type which causes the message structure to get bigger.

Since the CADS system software has a requirement to refresh calculated parameters eight times in a second, the simulated environment variables should be supplied at least nine times in a second. Because of this reason, serial port baud rate must be as fast as possible. But since data corruption via serial port cable is another performance concern and due to the fact that as the serial port's baud rate increases the probability of data corruption increases, it should be compensated. As it can be observed from program code in Appendix A10, serial port's baud rate is selected as 115200 kilo bits per second (kbps). Message structure size is calculated to be

(6x size of (double)) + (2x size of (int)) = 56 bytes.

By using a baud rate of 115200 kbps which is 14400 byte per seconds, it can be calculated that the system is capable of sending the message structure 257 times in a second. This rate makes the simulation variables seem to be coming continuously and the CADS system software can successfully fulfill the requirement of refreshing the calculated parameters in the required rate.

Besides detecting the baud rate of serial port, there also exist other settings for serial port. These can be listed as; parity bit, data bits and stop bit. In the system design odd parity, eight data bits and one stop bit are used. The setting for parity indicates that used parity checking mechanism is odd. Other than odd and even parity, there exists mark and space parity. In mark parity case there is parity which is not used and is always 1, whereas in space parity case there exists parity which is not used and is always 0. Since speed is the system's first priority, parity is used as one stage and only error detection mechanism.

As general system property, threads are used widely in software architecture for increasing the processor capability. The formed architecture is a multi-threaded architecture. Since calculating part is the CADS system software's most important feature, multi-threading structure is implemented in this software architecture.

Multi-threading is the operating system's ability to execute different parts (namely threads) of a program. In multi-threading, source allocations and thread source usage interferences should be carefully evaluated. Because wrong source allocations or source allocation interferences can cause deadlocks, data loss or evaluation of irrelevant data. In order to prevent these issues in the system design of the software, special data types are used. Because the designed CADS software is running on a single processor PC, the type of multi-threading performed on the system is performed by the time-division multiplexing like multitasking. In this type of multi-threading context switching is done very fast and frequent, therefore user perceives that the threads are executed simultaneously.

In summary, multi-threading advantages and disadvantages can be listed as follows.

The advantages of multithreading are:

- Improved performance and concurrency
- Simultaneous access to multiple applications
- Reduced number of required resources
- Increased throughput of data
- Faster applications.

The disadvantages of multithreading are:

- Difficulty of managing resources
- Difficulty of writing code
- Difficulty of testing and debugging

- Overhead creation for the processor
- Potential Deadlocks.

Operating system is the last general issue in the software architecture. Operating system serves as an intermediary between application programs and computer hardware. Operating systems have common tasks such as process management, interrupts, memory management, file system, device drivers, networking, security and I/O operations. Some of the best known operating systems can be listed as Microsoft Windows, Linux, Mac OS X and UNIX. Operating systems can be grouped mostly with respect to their determinism (timeliness), and multitasking capacity.

Main types of operating systems are:

- Real Time
- Multi/Single User
- Multi/Single Tasking
- Distributed
- Embedded.

In this project, software platform is based on Microsoft Windows operating system.

For program development in the C language an integrated development environment called LabWindows/ c for virtual ınstrumentation (CVI) is used. LabWindows/CVI is a proven American National Standard Institute (ANSI) C integrated development environment. It provides many tools for writing control applications. LabWindows/CVI uses ANSI C as basis. ANSI C is the standard published by ANSI for C programming language. Moreover, LabWindows/CVI combines the reusability and longevity of ANSI C with engineering-specific functionality of instrument control, analysis, data acquisition, and user interface development.

Below there is a short list for the general capabilities and properties of LabWindows/CVI:

- It is an ANSI C development environment.
- It is an intuitive environment optimized for test and control.
- It has data acquisition assistant and instrument I/O assistant.
- It is an interactive user interface editor. It enables easy creation of instrument drivers.
- It has built-in comprehensive debugging and remote debugging tools.
- It also has built-in measurement libraries acquisition, analysis and presentation.

Another issue that should be mentioned in the software architecture is thread-safe variables. The used threads are in general functions calculating the desired parameters from simulation variables. The other threads are for continuous reading from serial port and displaying the results. Before giving detailed information on thread safe variables thread functions and their missions are explained below.

5.1. Thread Functions and Their Missions

int CVICALLBACK ReadFunc(void *) : Serial port data read function.

int CVICALLBACK ReadControlThread(void *) : Serial port data read synchronization controlling function.

int CVICALLBACK IasThreadFunc(void *) : Indicated airspeed calculating function.

int CVICALLBACK TasThreadFunc(void *) : True airspeed calculating function.

int CVICALLBACK SatThreadFunc(void *) : Static air temperature calculating function.

int CVICALLBACK TatThreadFunc(void *) : Total air temperature calculating function.

int CVICALLBACK MachNumberThreadFunc(void *) : Mach number calculating function.

int CVICALLBACK PressureAltitudeThreadFunc(void *) : Pressure Altitude calculating function.

int CVICALLBACK BaroCorrectedAltitudeThreadFunc(void *) : Baro corrected altitude calculating function.

int CVICALLBACK AoaDataThreadFunc(void *) : Angle of attack data calculating function.

int CVICALLBACK LowAltitudeWarningThreadFunc(void *) : Low altitude warning indicator function.

int CVICALLBACK OverspeedWarningThreadFunc(void *) : Overspeed warning indicator function.

int CVICALLBACK StallWarningThreadFunc(void *): Stall warning indicator function.

int CVICALLBACK AsyncWrite(int panel, int , int , void *, int , int) : Function which is called by the periods of 16 times/second and this function displays the calculated results.

While thread functions are running in parallel, if there is a set of common variables read and/or changed by different threads, there is always high possibility of data corruption and data access collisions. Therefore in order to prevent data corruption and data access collisions, there is a special type variable called thread-safe variable used. Thread-safe variable is a variable type which is used for variables accessed from different threads in order to change or read. By the use of data type thread-safe variable (TSV), number of threads simultaneously accessing the variable can be limited. In the CADS's software design it is limited to 1. This means that in the software thread-safe variable typed variables can be accessed by only 1 thread at a time. Therefore there cannot be any chance such that while one thread is reading the variable, the other thread to access and change the variable, that is, an access to a thread safe variable locks it.

5.2. Thread Safe Variables and Their Missions

DefineThreadSafeScalarVar(int, tsvReadLoop, 0) : Thread safe variable that controls the continuity of running code.

DefineThreadSafeScalarVar(int, tsvReadControl, 0) : This variable determines the time that program reads the new serial port sensor data to the global data structure.

DefineThreadSafeScalarVar(int, tsvWriteControl, 0) : This variable determines the time that displaying function is permitted to display the calculated global data or not.

DefineThreadSafeScalarVar(SensorData, tsvSensorData, 0) : Global serial read data structure typed thread safe variable.

DefineThreadSafeScalarVar(ProcessedData, tsvProcessedData, 0) : Global calculated/displayed data structure typed thread safe variable.

DefineThreadSafeScalarVar(int, tsvIasFuncRead, 0) : This variable determines the indicated air speed function serial port data read status.

DefineThreadSafeScalarVar(int, tsvTasFuncRead, 0) : This variable determines the true air speed function serial port data read status.

DefineThreadSafeScalarVar(int, tsvSatFuncRead, 0) : This variable determines the static air temperature function serial port data read status.

DefineThreadSafeScalarVar(int, tsvTatFuncRead, 0) : This variable determines the total air temperature function serial port data read status.

DefineThreadSafeScalarVar(int, tsvMachFuncRead, 0) : This variable determines the mach number function serial port data read status.

DefineThreadSafeScalarVar(int, tsvPressureAltitudeFuncRead, 0) : This variable determines the pressure altitude function serial port data read status.

DefineThreadSafeScalarVar(int, tsvBaroCorrectedAltitudeFuncRead, 0) : This variable determines the baro corrected altitude function serial port data read status.

DefineThreadSafeScalarVar(int, tsvAoaDataFuncRead, 0) : This variable determines the angle of attack data function serial port data read status.

DefineThreadSafeScalarVar(int, tsvLowAltitudeWarningFuncRead, 0) : This variable determines the low altitude warning function serial port data read status.

DefineThreadSafeScalarVar(int, tsvOverspeedWarningFuncRead, 0) : This variable determines the overspeed warning function serial port data read status.

DefineThreadSafeScalarVar(int, tsvStallWarningFuncRead, 0) : This variable determines the stall warning function serial port data read status.

DefineThreadSafeScalarVar(int, tsvIasFuncWritten, 0) : This variable determines the indicated air speed function calculated data display status.

DefineThreadSafeScalarVar(int, tsvTasFuncWritten, 0) : This variable determines the true air speed function calculated data display status.

DefineThreadSafeScalarVar(int, tsvSatFuncWritten, 0) : This variable determines the static air temperature function calculated data display status.

DefineThreadSafeScalarVar(int, tsvTatFuncWritten, 0) : This variable determines the total air temperature function calculated data display status.

DefineThreadSafeScalarVar(int, tsvMachFuncWritten, 0) : This variable determines the mach number function calculated data display status.

DefineThreadSafeScalarVar(int, tsvPressureAltitudeFuncWritten, 0) : This variable determines the pressure altitude function calculated data display status.

DefineThreadSafeScalarVar(int, tsvBaroCorrectedAltitudeFuncWritten, 0) : This variable determines baro corrected altitude function calculated data display status.

DefineThreadSafeScalarVar(int, tsvAoaDataFuncWritten, 0) : This variable determines the angle of attack data function calculated data display status.

DefineThreadSafeScalarVar(int, tsvLowAltitudeWarningFuncWritten, 0) : This variable determines the low altitude warning function calculated data display status.

DefineThreadSafeScalarVar(int, tsvOverspeedWarningFuncWritten, 0) : This variable determines the overspeed warning function calculated data display status.

DefineThreadSafeScalarVar(int, tsvStallWarningFuncWritten, 0) : This variable determines the stall warning function calculated data display status.

CHAPTER 6

CONFIGURATION ITEMS

6.1. Hardware Configuration Items:

The CADS system performs the calculations for two systems namely the AOA and the ADC. Therefore, in listing hardware configuration items for the CADS it contains both of these systems. As before mentioned, CADS as a system takes the AOA data, static pressure, outside temperature and total pressure as input data. By using the data taken the CADS calculates AOA, indicated airspeed (IAS), true airspeed (TAS), SAT, total air temperature (TAT), Mach Number, Pressure Altitude and Baro Corrected Altitude. Along with the calculation, the CADS also provides Over Speed Warning, Low Altitude Warning and Stall Warning. In this design, the CADS including both AOA and ADC functionality is embedded in a PC of the hardware specifications below:

- Intel Pentium 4 Processor 1.3 gigahertz
- Serial Communication(RS232) Port
- Motherboard
- Hard disk
- 1 GB Random Access Memory (RAM)
- Compact Disc (CD) Player.

6.2. The Software Configuration Items:

As one can observe from Figure 10 the CADS software configuration items (SWCI) contains Platform, Development Environment, and Software items.

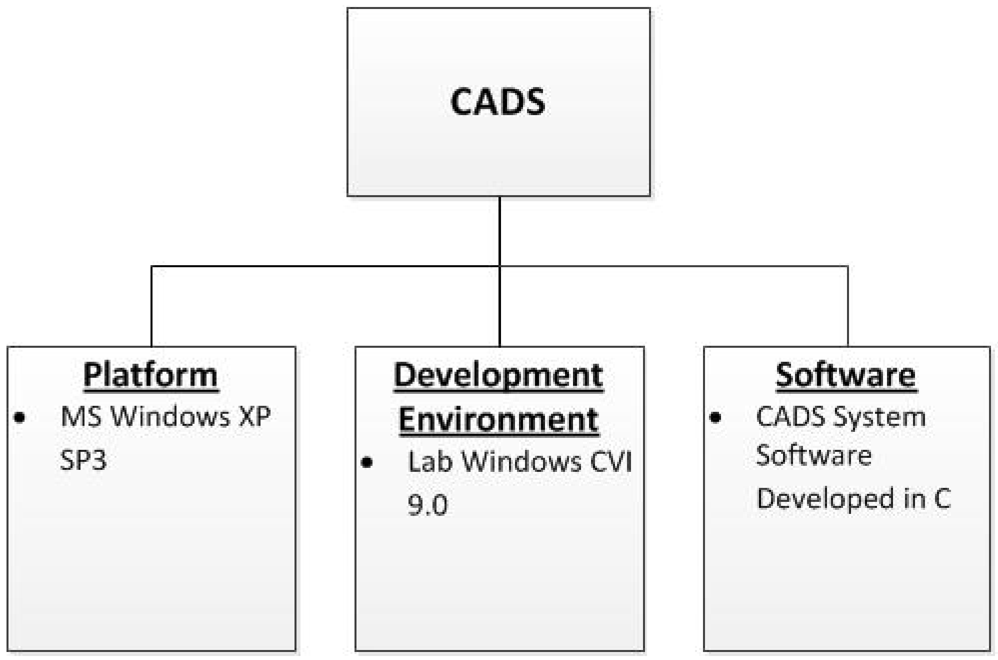


Figure 10 The SWCI of the Developed System

The CADS Software Configuration Item parts are listed below:

- Microsoft Windows XP SP3 (Platform)
- Lab Windows CVI 9.0 (ANSI C IDE)
- CADS System Software Developed in C (Software).

CHAPTER 7

TEST AND VERIFICATION OF THE SYSTEM DEVELOPED

The AOA sensors are potentiometers actually. Therefore, there should be a mechanism to measure the voltage drop on the AOA sensor’s resistance. After the voltage measurement, only thing that should be done is to make a look-up-table which shows the relation between voltage and corresponding angle. After collecting all of these inputs, required calculations should be made and the calculated parameters should be sent to the MFD in the flying platform.

As can be seen from Figure 11, the verification of the real system should be made via real sensors. AOA sensor should be put in motion and the verification should be made via controlling the output of the CADS. Stall warning should also be controlled by this manner.

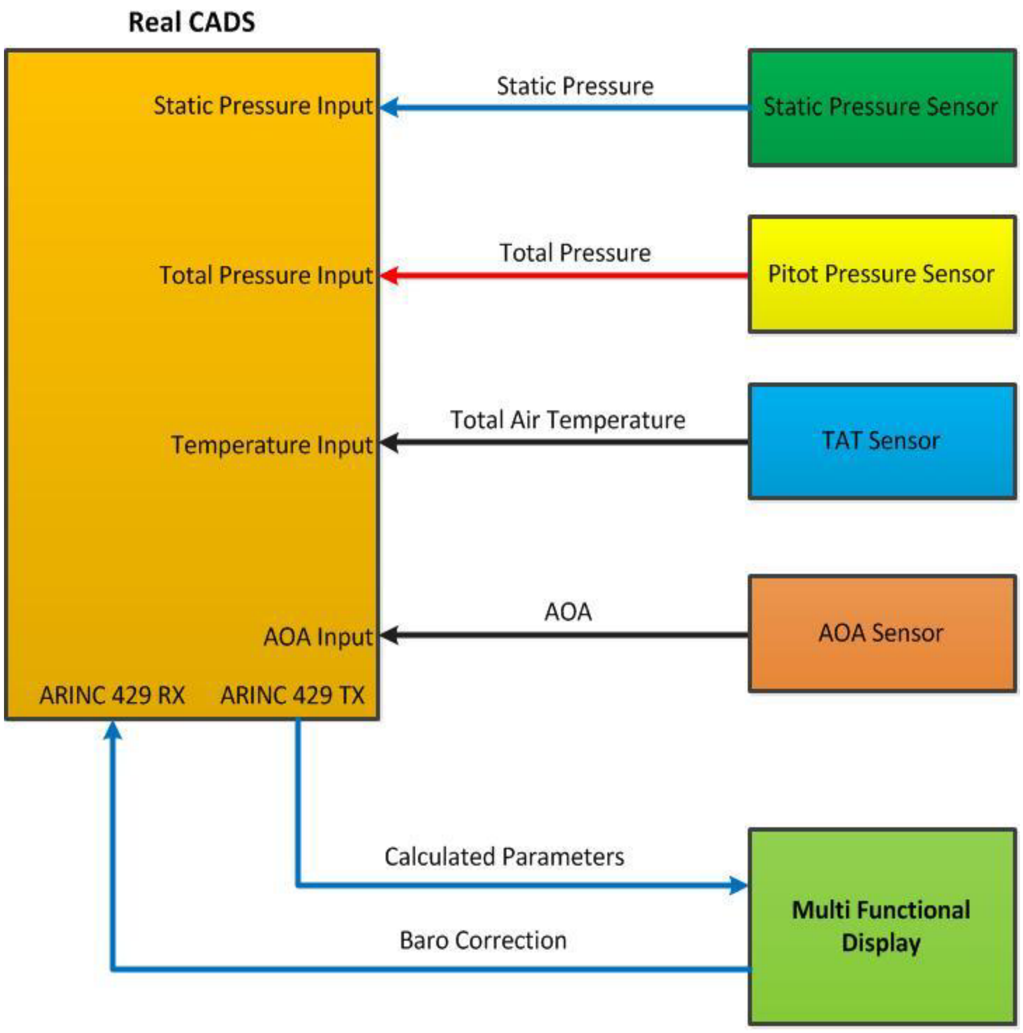


Figure 11 Real CADS System Verification

Since real sensors could not be generated, simulated sensor data is used in this project. Therefore in the system, the verification is made by comparing the CADS calculations with Microsoft flight simulator X (FSX)'s calculations. The full list and ranges of the inputs of the system are given in Table 1.

Table 1 List of the Inputs of the CADS

No	Parameter Name	Range
1	Angle of Attack	- 15 to + 50 degrees
2	Weight on Wheels	Discrete

The full list and ranges of the outputs of the system are given in Table 2:

Table 2 List of the Outputs of the CADS software

No	Parameter Name	Range	Refresh Rate (Hertz)
1	Pressure Altitude	-1871 to +36,089 feet	8
2	Baro Corrected Altitude	-1871 to +36,089 feet	8
3	IAS	0 to +510 knots	8
4	TAS	0 to +510 knots	8
5	Mach Number	0.200 to 0.999	8
7	TAT	- 60 to + 99 degrees	8
8	SAT	- 99 to + 60 degrees	8
9	Overspeed Warning	Discrete	-
10	Low Altitude Warning	Discrete	-
11	AOA	0 to + 20 degrees	8
12	Stall Warning	Discrete	-

CHAPTER 8

CONCLUSION

In this thesis, software of the CADS is developed and demonstrated on a PC card, which is not on-the-shelf right now. The CADS software designed combines the ADC functionality and the AOA functionality on a single box, actually on a card. Normally functions of these two systems are available as two different instruments in the market and on the flying platforms.

The functionalities fulfilled by the CADS software are among the most important issues for a flying system. They provide very critical flight parameters to the users of the platforms (pilots and ground control stations etc.). Since the CADS software in this thesis is developed for demonstration purposes, the criteria needed for a real system is not followed. The complicated and strict environmental standarts and safety specifications have not been applied. Therefore, for the CADS software to be applicable for real systems, a real time operating system (RTOS) in the software that satisfies all the timing requirements of the operation must be used. The application software to generate the required functions for these systems should be developed under a very tightly controlled environment and tested and verified rigorously. Beside, the laboratory tests for the software, integration tests with hardware must be performed in the laboratory with a software development laboratory (SDL) first using some simulated environment and emulators, and then in the system integration laboratory (SIL) utilizing mostly the real equipment and wirings, power and cooling and minimum number of simulators and emulators. The developed system is then tested in real flight testings and when passed such tests it is certified by the certain authorities as flyable.

Not only the software but the hardware on which the CADS software runs should meet very strict and detailed environmental tests and standards. They should operate under extreme ranges of pressure, temperature, vibration, humidity and the like. The integrated system, hardware and software together, is subject to very rigorous and detailed tests before certified as flyable.

The CADS software in this thesis is developed and built on a commercial hardware and tested by commercial flight simulator software for a generic platform. Therefore, it is not safe for flight, or in other words it cannot fly. Furthermore it even can not be run on a real platform. Aim in this thesis is to demonstrate that such very important instruments can be combined and integrated in one instrument and nearly all of the functions can be obtained by software.

In this thesis a commercial flight simulator is used for a generic flying platform. Flight data is obtained based on a flight scenario of this flying platform. The calculated output information from the CADS software developed is compared with the data in the flight simulator. It is observed that there is a very good agreement between them. For a better verification of the system, of course, a real flight data of a certain platform should be used. However, this type of data is generally restricted, very expensive, and in not the detail required [10].

Although, working on a commercial hardware and built on a commercial operating system software, nevertheless, the CADS developed here calculates all of the necessary parameters mentioned by related standards. Of course, the system has no environmental or electromagnetic interference/electromagnetic compatibility (EMI/EMC) qualifications.

Running the CADS software on an RTOS could be a one step further in the project. It was impossible to use an RTOS as their licenses and development tools are very expensive. However, a future work may be to develop the system on a specific hardware, not a generic PC card as used in this project, and utilize an RTOS on this hardware. The application software could also be developed based on real time restriction by using the required software languages such as Ada, C, C++ etc., and also in accordance with a system and software development standards.

A further improvement in the future could be to develop the hardware on a ruggedized system to satisfy the flight environmental conditions and tests as required. The developed system with RTOS and software standards could be tested first on a real SDL and SIL for a certain specific flying platform and also certified by the flight tests.

Another development in the future could be adding some new functionality such as INS/Global Positioning System (GPS). The GPS position, pitch, roll and yaw data could be get from a peripheral system and after making required calibration to these data they could be shown to the user also.

Actually, not all of the equipment producers use an RTOS. Some of them use field programmable gate array (FPGAs), digital signal processors (DSPs) and microprocessors to fulfill the real time compability of the system. Therefore, developing the CADS software on such kind of boards without any OS could also be implemented.

REFERENCES

[1] http://en.wikipedia.org/wiki/Airfoil

[2] http://en.wikipedia.org/wiki/Stall_(flight)

[3] http://rgl.faa.gov/Regulatory_and_Guidance_Library%5CrgTSO.nsf/0/E74EF22C8BCEF3B386256DC1006E015D?OpenDocument

[4] http://en.wikipedia.org/wiki/DO-160

[5] Myron Kayton , Walter R. Fried, John Wiley & Sons, Inc. (1997), Avionics Navigation Systems Second Edition.

[6] http://en.wikipedia.org/wiki/Lift_coefficient

[7] http://en.wikipedia.org/wiki/File:Virgin_a340-600_g-vmeg_planform_arp.jpg

[8] http://en.wikipedia.org/wiki/Drag_coefficient

[9] http://www.engineersgarage.com/contribution/porting-of-microc-os-2-kernel-in-arm-powered-microcontroller

[10] ILĞIN, R., 2011, Development of Air Data Computation Function of a Combined Air Data and AOA Computer, Thesis (MSc.), Çankaya University