12-2011

Automating Real-Time Fault Detection for the University of Tennessee Space Institute, Aviation Systems' Flight Testing and Airborne Science Applications

Samuel Vivek Williams
swilli90@utk.edu

Recommended Citation

Williams, Samuel Vivek, "Automating Real-Time Fault Detection for the University of Tennessee Space Institute, Aviation Systems' Flight Testing and Airborne Science Applications. " Master's Thesis, University of Tennessee, 2011.
https://trace.tennessee.edu/utk_gradthes/1107

To the Graduate Council:

I am submitting herewith a thesis written by Samuel Vivek Williams entitled "Automating Real-Time Fault Detection for the University of Tennessee Space Institute, Aviation Systems' Flight Testing and Airborne Science Applications." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Aviation Systems.

John Muratore, Major Professor

We have read this thesis and recommend its acceptance:

Uwe Peter Solies, Borja Martos

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Automating Real-Time Fault Detection for The University of Tennessee Space Institute, Aviation Systems' Flight Testing and Airborne Science Applications

A Thesis Presented for the

Master of Science Degree

The University of Tennessee, Knoxville

Samuel Vivek Williams

December 2011

ACKNOWLEDGMENTS

I thank everyone who has helped me get to this point. I would like to express my sincere gratitude to my advisor Professor Muratore for his guidance, encouragement and support throughout my time at UTSI. I wish to thank Dr. Corda for providing me with the opportunity to express myself. I thank Professor Martos and Dr. Solies for their guidance through my course and for serving on my committee. I also thank Mr. Simmons, Mr. Heatherly and Mr. Porter for their help during my thesis and GRA work.

I would like to thank Jonathan Kolwyck, Philip Appiah-Kubi and Coral Franklin for standing by me and for all the time I have spent in their company.

Finally, I would like to thank all my friends and family, both here and in India, for their continued support and encouragement.

ABSTRACT

The UTSI Aviation Systems program has conducted many successful airborne science campaigns in collaboration with premier research organizations including NASA and NOAA. Each airborne science mission requires dedicated FTEs to monitor the various instruments onboard the aircraft. A typical mission requires aircraft to be instrumented with a wide range of sensors (with approximately 145 data parameters). Monitoring the instruments requires highly skilled personnel who have a thorough understanding of the system.

With the advent of UTSI Aviation Systems program increasing capabilities to conduct multiple missions, using multiple airborne platforms, the requirement of a skilled FTE for each mission could effectively impede mission readiness. Conversely, the customers have also expressed interest in increased involvement in the airborne science missions and hence have to be accommodated within the limited confines of the aircraft. As a result of these requirements, a real-time expert system has been developed (using LabVIEW) to monitor mission-critical instrumentation. The program will provide the user with a tool to monitor the performance of the airborne sensors without requiring extensive knowledge of the system and rigorous training. The overall effect would be an increase in flexibility while simultaneously enhancing quality of operation wherein a mission would not be flown with a defective sensor onboard.

The following work describes the algorithms, system architecture and coding techniques used to develop the “go no-go” program. As the program is under constant refinement, the descriptions presented reflect the current state of the software.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

UTSI	The University of Tennessee Space Institute
AvSys	Aviation Systems
NASA	National Aeronautics and Space Administration
NOAA	National Oceanic and Atmospheric Administration
FTE	Flight Test Engineer
UMPC	Ultra Mobile PC
COTS	Commercially Off The Shelf
DAS	Data Acquisition System
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
SHARP	Spacecraft Health Automated Reasoning Program
STOL	Short Take-Off and Landing
NI PXI	National Instruments PCI eXtensions for Instrumentation
LAN	Local Area Network
UDP	Universal Datagram Protocol
GUI	Graphical User Interface
AWOS	Automated Weather Observation System

1. Introduction

1.1 Problem Statement

The University of Tennessee Space Institute (UTSI) Aviation Systems (AvSys) program has recently conducted multiple successful airborne science missions in collaboration with premier research institutions including NASA and NOAA. The program also simultaneously conducts a graduate level course as well as other short courses to educate future Flight Test Engineers (FTEs) and Test Pilots in the nuances of flight testing. To accomplish these missions, UTSI has been operating instrumented general aviation aircraft (such as the PA-31, Piper Navajo) equipped with Ultra Mobile PC (UMPC) based “electronic kneeboard” to display and store data acquired digitally (Muratore 2010).

To ensure efficient usage of the limited resources at its disposal, the UTSI AvSys program has been using commercially off the shelf (COTS) instruments as sensors onboard the aircraft with a National Instruments LabVIEW based data acquisition system (DAS). This system acquires, processes and displays data. While for the most part the instruments are found to function satisfactorily, they are still susceptible to malfunction. In such cases, it is imperative for the user to detect an error and take the necessary corrective action timely. A further requirement to maintain quality of operations also demands that no mission is flown with faulty instruments as it could compromise the mission. With approximately 145 parameters to be monitored on a typical mission, it is necessary to have a highly trained FTE to monitor the various instruments that are critical for the particular mission.

With UTSI augmenting its flight testing fleet, conducting multiple missions simultaneously may be hindered due to the want of sufficient skilled FTEs. Simultaneous increase in customer interest in mission involvement (and hence the need to accommodate them within the limited confines of the aircraft) requires fresh FTEs be trained on a per mission basis. In order to enable new FTEs to support a particular mission, a real time fault detection system has been developed to monitor mission critical instruments.

The system known as the “go no-go” program comprises a set of algorithms that have been developed using LabVIEW. These algorithms acquire the real time data output which is transmitted via UDP. The algorithms detect erroneous data which indicate a faulty instrument. To maintain mission specificity, a list of parameters critical to the particular mission is detailed in a “go no-go” checklist with associated limits. Through the use of limit-checking algorithms, data is analyzed. This gives rise to a violation signature for each data packet. Subsequently, noise filtering is performed before the violations are displayed to the user and logged for post-flight review.

1.2 Previous Efforts

An expert system is as an intelligent computer program that simulates human reasoning in problem solving (Angeli & Chatzinikolaou, 2004). In general, expert systems can be split into two categories namely:

1) Rule-based systems which utilize structured description of domain expertise to detect faults

2) Model based systems which are able to predict normal behavior of a system from first principles, thus being able to detect deviations.

While the former is a quick technique to verify against known fault situations, the latter is a method which is more applicable to fault detection systems where unexpected cases cannot be covered by purely heuristic rules (Prabu, 1991).

During the early years of expert-system development, NASA took the lead in introducing real-time systems in the aerospace domain. Investigations conducted by NASA during this period, reveal important guidelines for expert-system design and development. The space shuttle telemetry analysis using a real time expert system details a model which effectively combines traditional monitoring systems with algorithmic and expert system software techniques to provide real time assistance to shuttle mission controllers (Muratore, 1987). The Real Time Data System (RTDS) further expanded the capabilities of the expert system to enhance applicability and reduce training time (Muratore, 1989). The Spacecraft Health Automated Reasoning Program (SHARP) implemented domain expertise and artificial intelligence techniques to diagnose anomalies in the telecommunication link for deep space missions (Quan et al., 1994). The developments made at NASA were also applied for flight testing of F-15 STOL aircraft at Edwards Air force Base (Flanders et.al., 1992). While the above research initiatives discuss development of expert systems, integration of expert systems into an operational environment can become a critical roadblock, Hughes (1989) also discusses general guidelines for expert-system integration.

1.3 Statement of Requirements

To effectively perform the necessary tasks, the “go no-go” instrumentation fault detection program was designed to meet the following set of requirements:

1. The “go no-go” program shall perform checks based on:
    - Static data
    - Analog limits violation
    - Inconsistencies with data from other instruments and parameters
2. The program shall display all “go no-go” parameters and display violations in a time ordered manner
3. Noise filtering shall be applied to determine violation activity to ensure decrease in display jitter
4. The program shall enable the user to ignore/acknowledge violations during operation
5. The program shall change go no-go parameter limits depending on operation phase both with and without user input
6. The program shall enable the user to change limits of existing parameters during operation
7. Each “go no-go” file shall specify the different checks applicable to parameters for a given mission
8. Changes in “go no-go” files shall be made through the program and saved as a new file.

9. The program shall display engineering units of violations wherever applicable and these units will be sourced from the "DAS config” file
10. All violations shall be logged at first instance of detection and when return to normal performance is observed
11. The program shall perform search to find all “go no-go” files which will contain “go no-go.txt” in file name

2. “GO NO-GO” System Architectures

2.1 Design of Expert System

In real-time expert systems that operate at high throughput, as in an aerospace system, decrease in latency of fault detection is critical and hence more emphasis is placed on real-time demands of the system than the more computationally expensive diagnostic functions (Quan, et.al., 1994). Also, rule-based systems are better applied where experience and expertise are readily available but an in-depth knowledge of the system, capable enough to predict its behavior, is unavailable or too costly to obtain (Angeli, 2010). Considering these factors, a decision was made to follow a rule-based approach for the “go no-go” fault detection program.

To design the expert system, a three layer system model, as discussed by Flanders et.al. (1992), was considered. The layers can be detailed as:

Layer 1: This is the basic data acquisition layer. Data is acquired in real-time, but no processing is done.

Layer 2: Basic processing for filtering, unit conversion and calibration is conducted to output engineering data.

Layer 3: At this stage, the knowledge base is used to determine the status of the system. The heuristic rules that form the database are not programmed into the code but are stored in the form of rules (Muratore, 1987).

A further, higher, layer is the flight test engineer who is required to use the output provided by the expert system to make decisions.

The implementation of the “go no-go” program has been conducted entirely using LabVIEW software. This decision was taken so as to utilize available expertise within UTSI while also recognizing that the widely popular software is used by most customers (such as NOAA and NASA). The use of LabVIEW also establishes commonality with other UTSI airborne systems which are already in use.

2.2 System Architecture

The necessity to perform real-time fault detection requires that the “go no-go” program be accommodated along with the existing DAS architecture already in use aboard UTSI AvSys platforms. In the current setup, data from sensors is acquired by a NI PXI based processor via analog to digital (A to D) converters, serial ports and directly through the network. Using a LabVIEW based software; the raw data from the instruments is converted into engineering information and combined into a single time-tagged data packet. The data packet is then transmitted over Ethernet using a Multicast Universal Datagram Protocol (UDP). UMPC systems that are connected to the Ethernet LAN are then able to “listen” to the UDP transmissions. Then data monitoring devices designed in LabVIEW use this data to generate comprehensive real-time displays to enable the user to view the acquired parameters as tangible engineering data (Muratore et.al., 2010).

The “go no-go” program is designed to be an additional UMPC based system. By acquiring and processing data independently, it does not encroach on the other processes conducted by the existing system. The combined system architecture is as shown below in Figure 1.

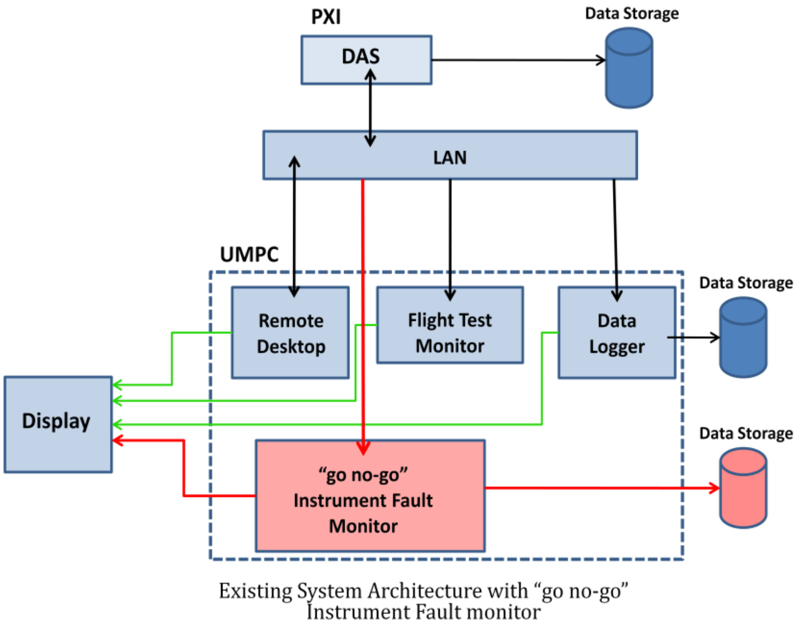

PXI
Data Storage
DAS
LAN
UMPC
Data Storage
Remote Desktop
Flight Test Monitor
Data Logger
Display
Data Storage
“go no-go” Instrument Fault Monitor
Existing System Architecture with “go no-go” Instrument Fault monitor

Figure-1: Combined System Architecture for UTSI Airborne Platforms

2.3 “GO NO-GO” Logical Architecture

The “go no-go” program is designed to cater to airborne science missions where an in-depth understanding of the domain is generally being investigated. An expert, who is either an experienced FTE or scientist, is tasked to provide the knowledge base which is then represented in the form of production rules. The rules are used to describe the action which must be taken if a symptom is observed, using an “IF-THEN” structure. Supporting architecture such as a graphic user interface (GUI), noise reduction mechanisms and explanation systems are domain independent systems provided to enhance user experience (Angeli, 2010). The basic structure of a classical rule-based expert system is illustrated in Figure 2.

The “go no-go” program achieves the basic rule-based expert system structure by dividing the program development functionally by use of “sub-VIs” provided by the LabVIEW software. A sub-VI is a standalone LabVIEW program which can be used as a sub-routine in a more complex program. Sub-VIs enable modular programming and allows for easier debugging of the software system (Young, 2010). The inference engine used by the program has been hard coded in LabVIEW while the rules themselves are sourced from a text document. This measure has removed the need to use a generalized inference engine while simultaneously building on the existing system software.

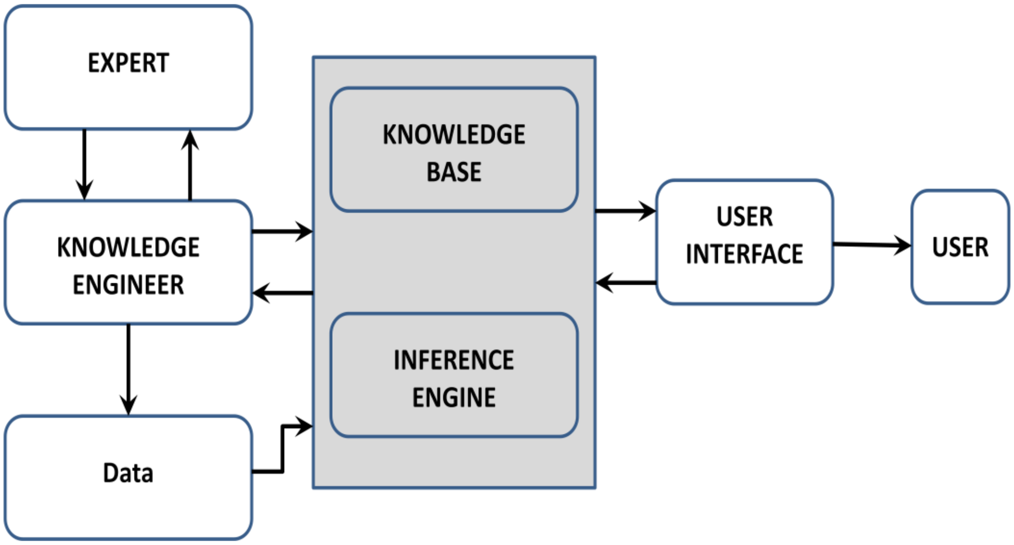


Figure-2: Basic structure of rule-based expert system (Angeli, 2010)

2.4 Data Flow

The data flow in the "go no-go" program is defined by three main data sources: the configuration files or "Inst List", the knowledge base or "go no-go list" and the real time data stream.

The configuration files used for UTSI AvSys missions describe the entire list of parameters along with their UDP indexes, calibration coefficients and units. Two separate lists are normally used per mission: one for baseline flight test instrumentation and the other for airborne science instrumentation. A sample from the "Navajo inst list" is shown in Table 1. These lists, though aircraft specific, are defined globally and are used to co-ordinate real-time data transfer between the DAS and other data monitoring (such as UMPC) and data storage (solid state disk) devices. The real-time data is transmitted via Ethernet as a tab-delimited string indexed according to the UDP numbers (detailed in the "UDP packet word" column) for each parameter.

A knowledge base can be defined as a set of "relatively small chunks of knowledge, represented in a highly modular way" (Feigenbaum, 1991).The "go no-go list" is the knowledge base used to determine instrument faults and consists of a list of parameters that are critical for the mission. To ensure proper data communication, the parameter names detailed in the "go no-go list" have to exactly match the names defined in the configuration lists. This also ensures that the parameter locations do not need to be hard coded into the software hence ensuring a more flexible structure. As configurations of the aircraft sensors vary with each mission, two separate "go no-go" lists are used. A "flight test" list is used for the baseline configuration and an "experiment" list is used for the additional mission specific airborne science instrumentation. Table 2 shows a sample of the "go no-go list" for NOAA Mercury airborne science missions.

To present data in a modular fashion, the data in the "go no-go" lists is organized under specific column headers. The columns are accessed by the program depending on the instrument and flight condition.

- The "parameter" column indicates the parameter whose data has to be checked.

- If a parameter is found violating, the “rational and recovery” section provides the user with procedures to follow to correct the given condition.
- The “activity counter” fixes the number of cycles before which an instrument has to update output data.
- For ground checks, “upper limit preflight” and “lower limit preflight” denote the maximum deviation allowed for a parameter data to diverge from the “nominal value preflight”. In cases where another data parameter can be compared to the given parameter, its parameter name is stored in the “compare to preflight” and is used to substitute the “nominal value preflight”. This provides dynamic comparison of data.
- During flight, the cruise checks are applied. The columns “upper limit cruise”, “lower limit cruise”, “nominal value cruise” and “compare to cruise” provide the same functionality as the preflight checks but the data fields are modified to cater to the cruise stage of flight (where the actual data acquisition takes place).

The data flow of the “go no-go” program is designed to compare real-time data with the limits detailed in the “go no-go” list so that checks can be performed in real-time. To achieve this, the parameter names from the “go no-go” list are used to obtain UDP indexes from the configuration file, which are then used to sort through the real time packets and isolate the required parameter data. Redundant data used for comparison checks are also obtained at this stage. Once the parameter data is obtained, further processing is done using the timed loop mechanism provided by LabVIEW. The timed loop cycles at the data acquisition rate (20Hz or every 50ms). Prior to processing parameter data using limit checking algorithms, AWOS and other relevant data are used to define nominal values which then drive the limits. Post limit checks, a violation signature is produced for each data packet which is then passed through a noise filter before it is displayed to the user and logged to a storage device. To enable the user to respond to dynamic flight conditions, mechanisms for modifying and storing limit checking parameters are also provided.

Table-1: sample of “Navajo inst list”

parameter name	UDP packet word	Source	DAS channel	a0	a1	a2	a3	a4	a5	notes
aircraft id	-1	computer	-1	0	0	0	0	0	0	
l aileron position	-1	das analog	-1	0	0	0	0	0	0	deg += TEU
aileron stick yoke position	-1	das analog	-1	0	0	0	0	0	0	deg
elevator stick yoke position	-1	das analog	-1	0	0	0	0	0	0	deg
rudder pedal position	-1	das analog	-1	0	0	0	0	0	0	deg
pxi gps enabled	-1		-1	0	1	0	0	0	0	
AHRS checksum	-1		-1	0	1	0	0	0	0	
time	0	computer	-1	0	0	0	0	0	0	
elapsed time	1	computer	-1	0	0	0	0	0	0	
maneuver counter	2	computer	-1	0	0	0	0	0	0	
sequential id	3	computer	-1	0	0	0	0	0		
rudder pedal force	4	das analog	0	1.2043	2.2939	0	0	0	0	lbs
aileron force	5	das analog	1	-2.833	0.6327	0	0	0	0	lbs
elevator force	6	das analog	2	0.0294	1.2938	0	0	0	0	lbs
angle of attack	7	das analog	3	16.363	0.1601	0	0	0	0	deg
angle of sideslip	8	das analog	4	-0.697	-0.3872	0	0	0	0	deg

Table-2: sample of “experiment go-no go – noaa hg”

parameter	rationale and recovery action	activity counter	upper limit preflight	lower limit preflight	nominal value preflight	compare to preflight	upper limit cruise	lower limit cruise	nominal value cruise	compare to cruise
time	TO RECOVER: restart airborne science das	20								
particle counter counts	Expect < 50 if pump off , Expected VIOLATION prior to takeoff. TO RECOVER function: check pump sw in proper position, check SEPS LED on. Cycle Exp power if SEPS off. If data problem, restart airborne science das		50	-50	50		5000000	-100	1000	
Particle Ctr Flowmeter	Expect < 1 if pump off, Expected VIOLATION prior to takeoff. TO RECOVER function: check pump sw in proper position, check SEPS LED on. Cycle Exp power if SEPS off. If data problem, restart airborne science das		0.5	-1	0		10	-0.5	1	

3. Graphical User Interface (GUI)

As the primary application of the “go no-go” program is in a flight environment, due consideration has been taken to ensure user friendliness and practicality while designing the GUI. To this end, NASA experiences with expert systems (Muratore, 1987 and Quan et al., 1994) have been utilized to determine salient features of the GUI design.

One of the primary considerations of the GUI design was to accommodate the entire window within the confines of the UMPC display area (7 inches across). To conserve screen space, the primary window is limited to the summary page, while the other pages can be accessed by activating tab controls provided by using the touch screen on the UMPC.

The summary page is provided with a time indicator, an iteration counter, a violation status indicator; ignore/enable controls and a parameter display array. As a parameter violates the predefined conditions, it is displayed in the array along with relevant information including parameter data, nominal value and rational and recovery action. Subsequent parameters detected are added to the top of the array, which can be scrolled to display all violating parameters. The violation status indicator provides a count of total parameters that are violating to alert the user to the existence of more parameters which are not immediately visible in the display area. The ignore control provided allows the user to acknowledge violations which then makes it unnecessary to display them further. Conversely, the enable control restores the visibility of the violating parameter. The time and iteration indicators on the summary page indicate the operational status of the “go no-go” program.

While the summary page shows only violating parameters, the subsequent “Flight Test” and “Experiment” pages display all parameters which are present in the respective “go no-go” lists. This separation enables the user to concentrate only on the baseline “Flight Test” instruments during educational missions. Each page has a violation indicator and parameter array. The parameter array displays the parameter name and value along with a status indicator which glows red or green to indicate if the parameter is violating. Unlike the summary page, the violations displayed in the flight test and experiment pages are not passed through the noise filter routines thus allowing the user to access a lower level of data processing.

Although the knowledge base can be determined preflight, the dynamic conditions experienced during flight and the unpredictability of scientific experiments can affect the nature of data output. Utilities provided within the program enable the user to tweak parameters to suit a specific flight condition which may not have been apparent preflight. These utilities include input of AWOS information (“AWOS input” page); selection of preflight/cruise, modification of existing parameter limits and addition of new parameters (“Mod GNG Parameters” page). The user may also save any modifications made to a new “go no-go” list (“Files & Networks” page) and access it using the “User Controls (GNG select)” page.

4. Fault Detection Algorithms

Domain dependant fault detection is generally based on limit checking or activity checking algorithms, each used separately to cater to two different kinds of onboard instruments. The limit checks are basically used to validate data from “receptive” sensors which have limited internal processing capabilities. The activity checks are applied to instruments with data processing capabilities and output data at a constant rate.

4.1 Limit Checks

Limit checking of parameter data is conducted with reference to a nominal value. The data is considered to be faulty if it does not occur within predefined tolerance limits of the given nominal value. The nominal value for each parameter can be derived from one of three different sources: the “go no-go” list which clearly defines nominal values for all limit checking parameters, associated environment data (e.g. AWOS information) which must be entered by the user and finally real-time data sourced from redundant instruments onboard the aircraft. As the assignment of nominal values is critical to the validity of these checks, expert input is accorded supremacy over the user, who cannot influence choice of nominal value (an exception to this case would occur only if a new parameter is added).

Using the limit checks, a parameter is found violating if:
(lower limit + nominal value) >= parameter data or
(upper limit + nominal value) <= parameter data

4.2 Activity Checks

Activity checks are mainly conducted to verify if an instrument is transmitting data at the desired rate. These checks assume importance for instruments which have an internal clock that is preset to transmit data at a given rate. If there is no change in data within a given time period (activity counter) the data and hence the instrument is be determined as non functional. This check is applicable especially if the device under consideration is an instrument which is being used to collect scientific data. Instruments in present use such as the “Hygrometer” and “SO_2 Detector” provide data output that cannot be instantly verifiable and are therefore subject to this check.

4.3 Noise Filtering and Time Ordering of Violations

To ensure that every marginal divergence of instrument data is not displayed to the user, noise filtering is applied. This reduces the probability of nuisance alarms which can distract the user from the actual faults. By this method, a violation is not passed to the summary page unless it is continuously detected for a given time period (e.g. 5000 ms). Similarly, a violation must remain undetected for a given period (5000 ms) before it can be “cleared” or removed from summary display. To execute the noise filter, a domain independent structure comprising of two bit change algorithms is applied. Each time a violation is detected or cleared, a time-stamp (millisecond counter provided by LabVIEW) is appended to it. The total violations and time-stamps for each packet is then stored as an array. If the subsequent packet contains the same violating parameter, the time-stamp of the parameter is replaced with that of the original detection. Comparing the time-stamp with the current time, the total time a violation has been continuously detected/cleared is determined. While displaying violations, importance has been given to the most recent detection. Therefore, violations displayed on the summary page are arranged in the descending order of detection times. To achieve this, time-stamp of violations which pass through the noise filter has been utilized.

5.4 Logging Violations

Logging of detected and cleared violations has been conducted to enable post flight review. In this case only violations which have passed the noise filtering are considered. User manipulation of the violation signature (by using the ignore/enable command) has however been ignored while logging violations. To reduce file size, logging is done only when a violation has been detected or cleared. As a separate violation signature is generated for each cycle, it is stored by the logging algorithm. It is then compared to the previous packet to determine if any new parameters are present. Conversely, it also checks if any of the previous parameters are not present in the new packet. These divergences are noted and then designated as “Detected” or “Cleared” respectively. The violating parameter along with the value, unit and time stamp is then appended to a tab-delimited text file which is saved to a storage medium.

5. Conclusions

The “go no-go” program has been developed to perform fault detection for UTSI flight test and airborne science missions. To achieve this, a rule-based expert system was developed using a knowledge base provided by experienced FTEs and scientists. The system was developed as an addition to the existing UTSI DAS which provides real-time sensor data from multiple airborne sensors.

While the present system provides basic limit checking algorithms for fault checking, it is capable of catering to the real-time needs of an airborne application. The GUI has been designed to enhance user experience, which will then enable quicker reaction times post fault detection. Program utilities enable the user to accommodate the dynamic conditions of airborne testing within the fault detections scheme.

The present “Go No-go” program, by conception, is a first order expert system (Feigenbaum, 1992). Faults detected by this program are limited to data errors, which leaves the user to determine the defective sensor. While the current version is an effective fault detection mechanism, improvements can be made to enable error diagnosis. The first step in this direction would be to include instrument mapping within the knowledge base which can be compared to the violation signature to isolate the faulty sensor(s).

However, to fully cater to the needs of airborne science, the program would have to improve on the artificial intelligence quotient by generating predictive models of sensor behavior. This model would integrate the heuristic knowledge base with a mathematical model generated for each sensor. Thus, the system would have a knowledge base architecture that permits the interaction of sensor information, modeling information and experimental knowledge representation (Angeli, 2010).

LIST OF REFERENCES

1. Angeli, C. and Chatzinikolaou, A., "On-Line Fault Detection Techniques for Technical Systems: A Survey", International Journal of Computer Science & Applications, Vol. I, No. 1, pp. 12-30.

2., Angeli, C., " Advanced Knowledge Based Systems: Model, Applications & Research, Chapter 4" TMRF open access e-book, Vol. 1. pp. 50-73, (published 2010).

3. Feigenbaum, E., "A Personal View of Expert Systems: Looking Back and Looking Ahead", Knowledge Systems Laboratory, Department of Computer Science, Stanford University, Stanford, California 94305, Report No. KSL 92-41, April 1992.

4. Flanders, J. B., Madison, R. M. and Jones, C H., " Expert System for Real-Time Aircraft Monitoring", Journal of Aircraft1 9920021-8669 vol. 29 no.1 (79-84) doi: 10.2514/3.46128.

5. Hughes, P. M., "Integrating Expert Systems into an Operating Environment", Automations Technology Section / code: 522.3, NASA, Goddard Space Flight Center.

6. Muratore, J. F., "Space Shuttle Telemetry Analysis by a Real Time Expert System" AIAA Computers in Aerospace Conference, 6th, Wakefield, MA, Oct7-9, 1987, Technical Papers (A88-12526 02-61). Washington, DC, American Institute of Aeronautics and Astronautics, 1987, p. 165-171.

7. Muratore, J. F. and Heindel T.A., "Real Time Data Systems: Incorporating New Technology in Mission Critical Environments" NASA, Mission Operations Directorate, Lyndon B. Johnson Space Center. Publication Year: 1990, Document ID: 19940005446, Accession Number: 94N72201, Updated/Added to NTRS: Aug 19, 2011.

8. Muratore, J., Moonan, W. and Young, J., "From kneeboards to Mobile Computers- Achieving Higher Quality Flight Test Engineering Education at Lower Cost", Society of Flight Test Engineers 2010 Annual Symposium, Patuxent River, Maryland, 13-16 September 2010.

9. Prabu, D., "A Generic Sensor Value Validation Procedure for Knowledge Based Fault Diagnosis Systems", Masters Thesis, Computer Science Dept., The University of Tennessee Space Institute, Tullahoma, TN, published 1991.

10. Quan A.G., Schwuttke U. M., Herstein J. S. and Atkinson D. J., " Automated Consultation for Diagnosis of Interplanetary Telecommunications" submitted to AAAI conference on Innovative Applications of Artificial Intelligence (IAAI-94).

11. Young, J., "Automating the Data Reduction Process for More Efficient Flight Tests and Reducing the Time from Days to Hours", Masters Thesis, Aviation Systems Dept., The University of Tennessee Space Institute, Tullahoma, TN, published 2010.

APPENDICES

Appendix 1: “GO NO-GO” PROGRAM USER MANUAL

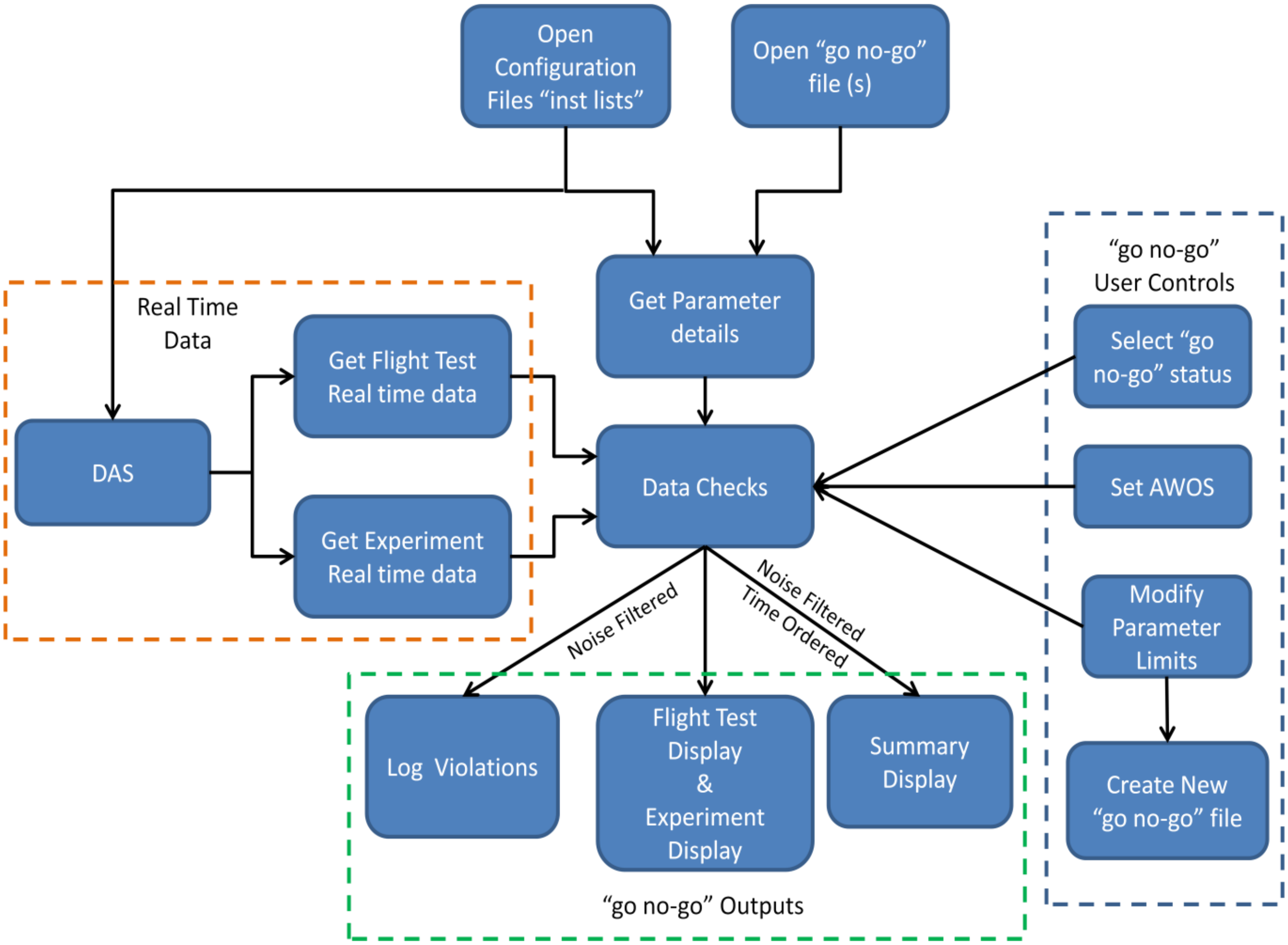
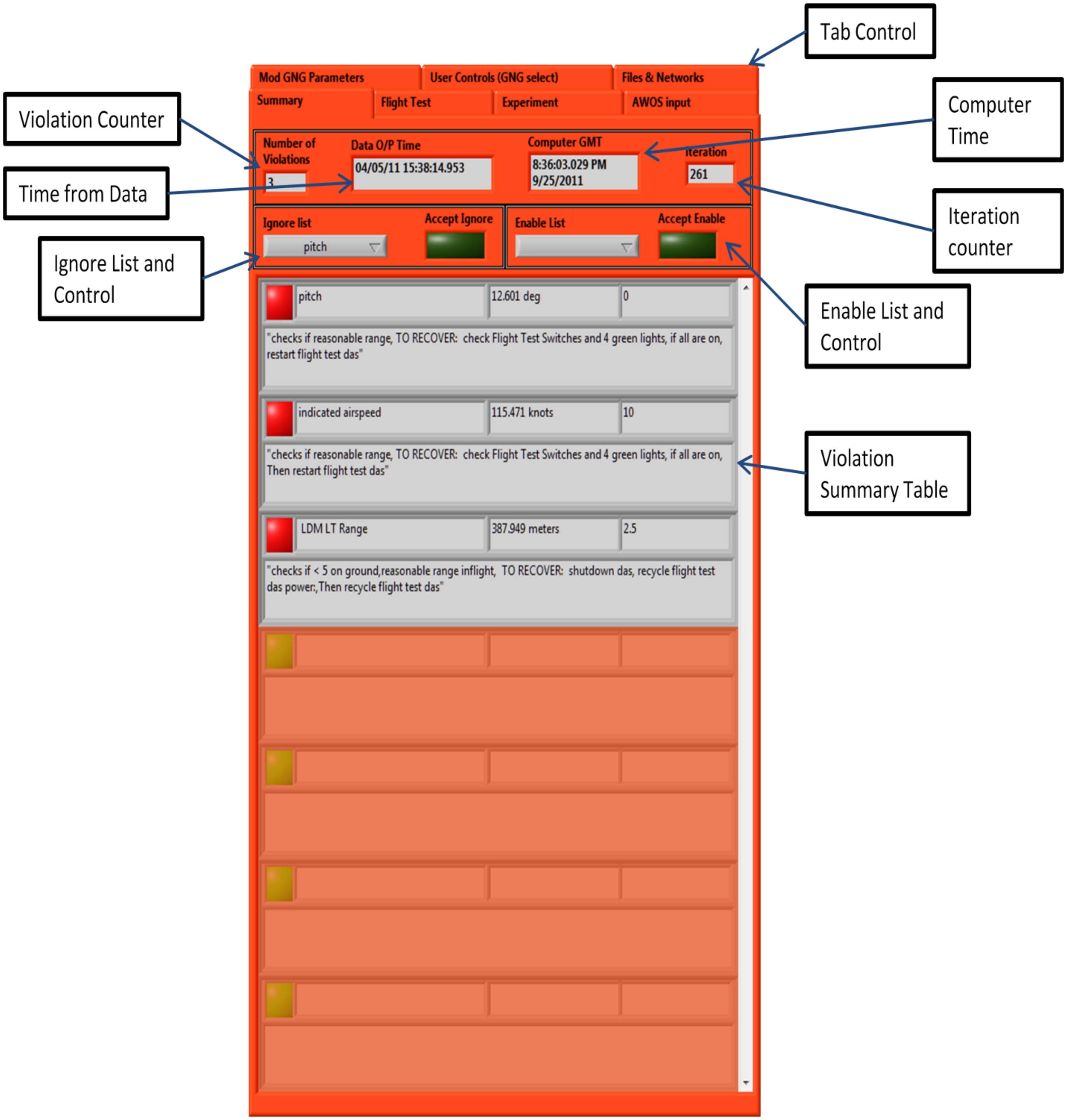


Figure A1.1 Schematic of the “go no-go” program and peripheries

Page 1: Summary Page



Number of Violations: 3

Data O/P Time: 04/05/11 15:38:14.953

Computer GMT: 8:36:03.029 PM 9/25/2011

Iteration: 261

Ignore list: pitch

Enable List:

🟥	pitch	12.601 deg	0
	"checks if reasonable range, TO RECOVER: check Flight Test Switches and 4 green lights, if all are on, restart flight test das"		
🟥	indicated airspeed	115.471 knots	10
	"checks if reasonable range, TO RECOVER: check Flight Test Switches and 4 green lights, if all are on, Then restart flight test das"		
🟥	LDM LT Range	387.949 meters	2.5
	"checks if < 5 on ground,reasonable range inflight, TO RECOVER: shutdown das, recycle flight test das power;,Then recycle flight test das"		

Figure A1.2: “Summary” page

This page provides the user with top level information which is sufficient for operations.

Description of functions:

- Tab Control: Provides user access to other pages, achieved by clicking on respective tabs
- Violation Counter: Indicates total number of violations at any given time
- Computer time: Time-stamp sourced directly from system clock
- Time from Data: Useful especially if viewing data post-flight. During real-time operations Data time-stamp and Computer time will match.
- Iteration Counter: Indicates the number of cycles already processed by the program during the current run.

  To verify if program is functioning normally, user has to check if Computer time and Iteration Counter are updating at a constant rate.
- Ignore List and Enable List with associated controls: The Ignore control is designed to remove from display a detected violation as and when it is activated. The Enable control allows the user to display any parameter that may have been previously acknowledged (removed).

  Method of operation: To acknowledge a violating parameter, the user has to select the appropriate parameter name from the "Ignore List" and then accept the selection by clicking on the "Accept Ignore" button. Similarly, the user can recover a display of a violating parameter by selecting the name from the "Enable List" and clicking on the "Accept Enable" button.
- Violation Summary Table: This table lists all violations as and when the program processes the detections. Whenever a new violation is detected it is added to the topmost row of the table. The violations may also be "cleared" or removed by the program when the parameter returns to normal state of operations. Violations (which have been previously detected) may also be cleared or displayed by the user by using the "Ignore" or "Enable" controls as described above.

  Each row in the Summary Table displays the following set of parameter data:

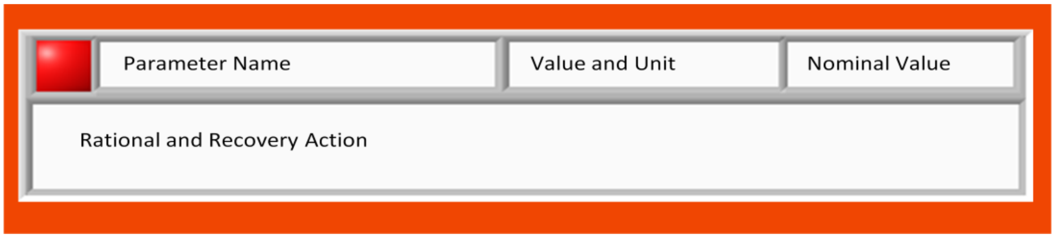
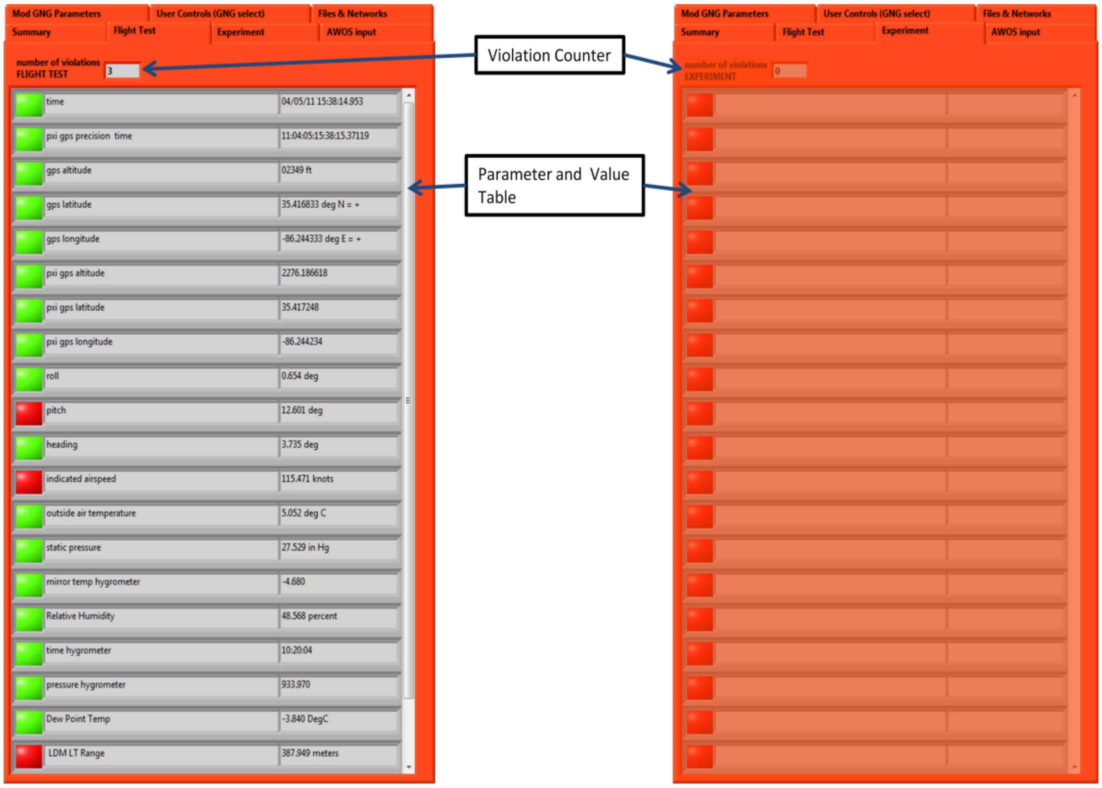


Figure A1.3: "Summary Table" description of fields

- If the program is reset, all functions on this page will be cleared of previous data and a new list will be started.

Page 2 & 3: Flight Test and Experiment Tables



number of violations FLIGHT TEST 3

🟩	time	04/05/11 15:38:14.953
🟩	pxi gps precision time	11:04:05:15:38:15.37119
🟩	gps altitude	02349 ft
🟩	gps latitude	35.416833 deg N = +
🟩	gps longitude	-86.244333 deg E = +
🟩	pxi gps altitude	2276.186618
🟩	pxi gps latitude	35.417248
🟩	pxi gps longitude	-86.244234
🟩	roll	0.654 deg
🟥	pitch	12.601 deg
🟩	heading	3.735 deg
🟥	indicated airspeed	115.471 knots
🟩	outside air temperature	5.052 deg C
🟩	static pressure	27.529 in Hg
🟩	mirror temp hygrometer	-4.680
🟩	Relative Humidity	48.568 percent
🟩	time hygrometer	10:20:04
🟩	pressure hygrometer	933.970
🟩	Dew Point Temp	-3.840 DegC
🟥	LDM LT Range	387.949 meters

number of violations EXPERIMENT 0

Page 2: Flight Test Table

Page 3: Experiment Table

Figure A1.4:"Flight Test and Experiment" pages

Pages 2 and 3 have similar displays which display either the parameter data under the "Flight Test go no-go list" or the "Experiment go no-go list" respectively.

Discussion of Functions:

- Violation counter: These displays are similar to the function on the Summary page and provide the total count of violations within that page.
- Parameter and Value Table: These tables display the total parameters within each respective list and their associated values as shown below:

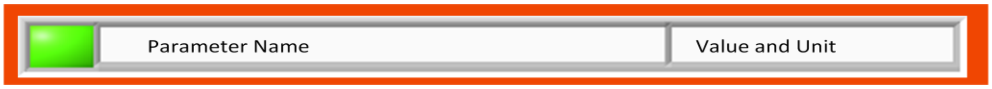
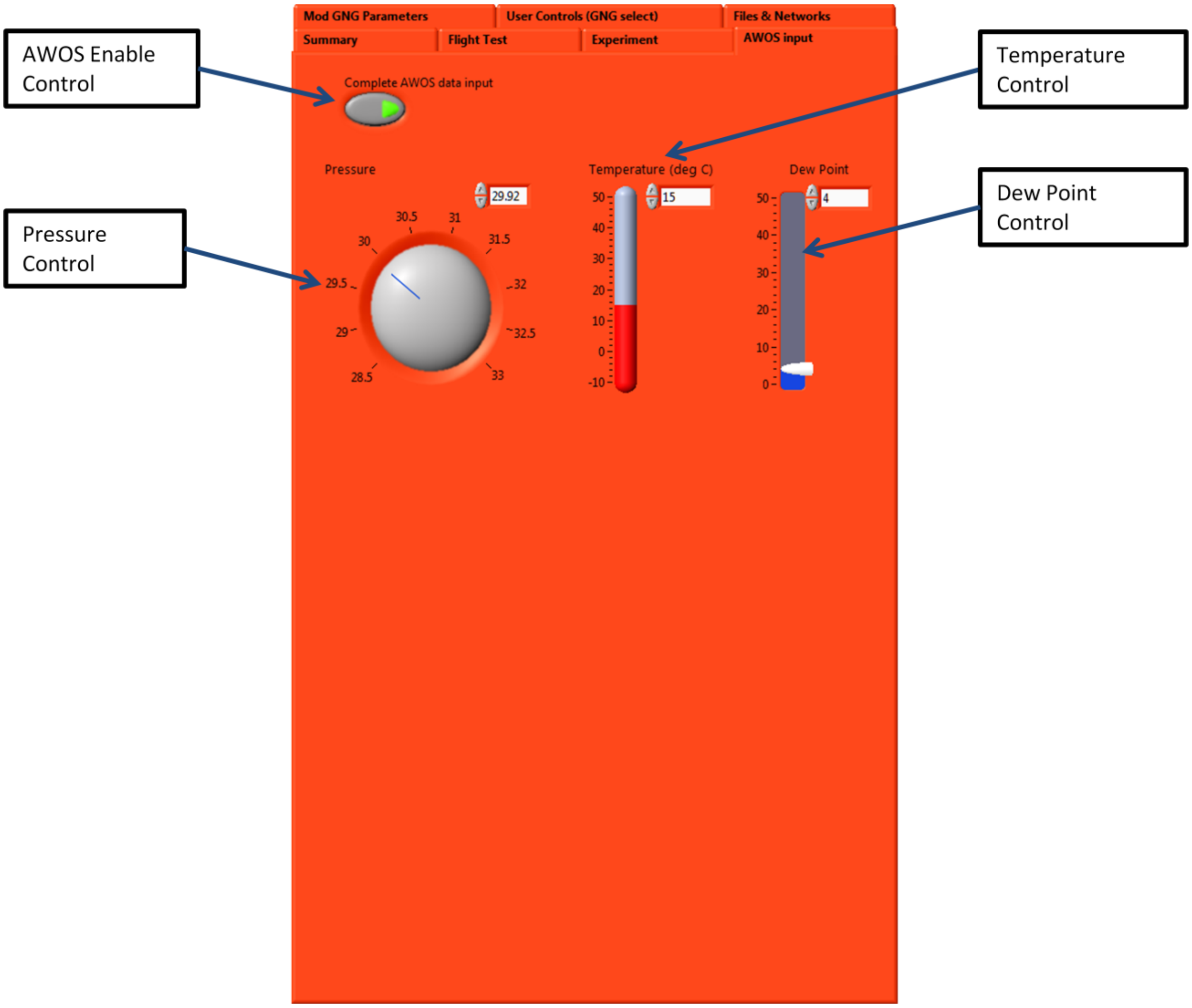


Figure A1.5: "Flight Test and Experiment" tables, description of fields

In case a certain data packet (e.g. "Experiment" data packet) is not received by the program, the display will blank out (as shown by the Experiment Table)

Page 4: AWOS Input



Pressure 29.92

Temperature (deg C) 15

Dew Point 4

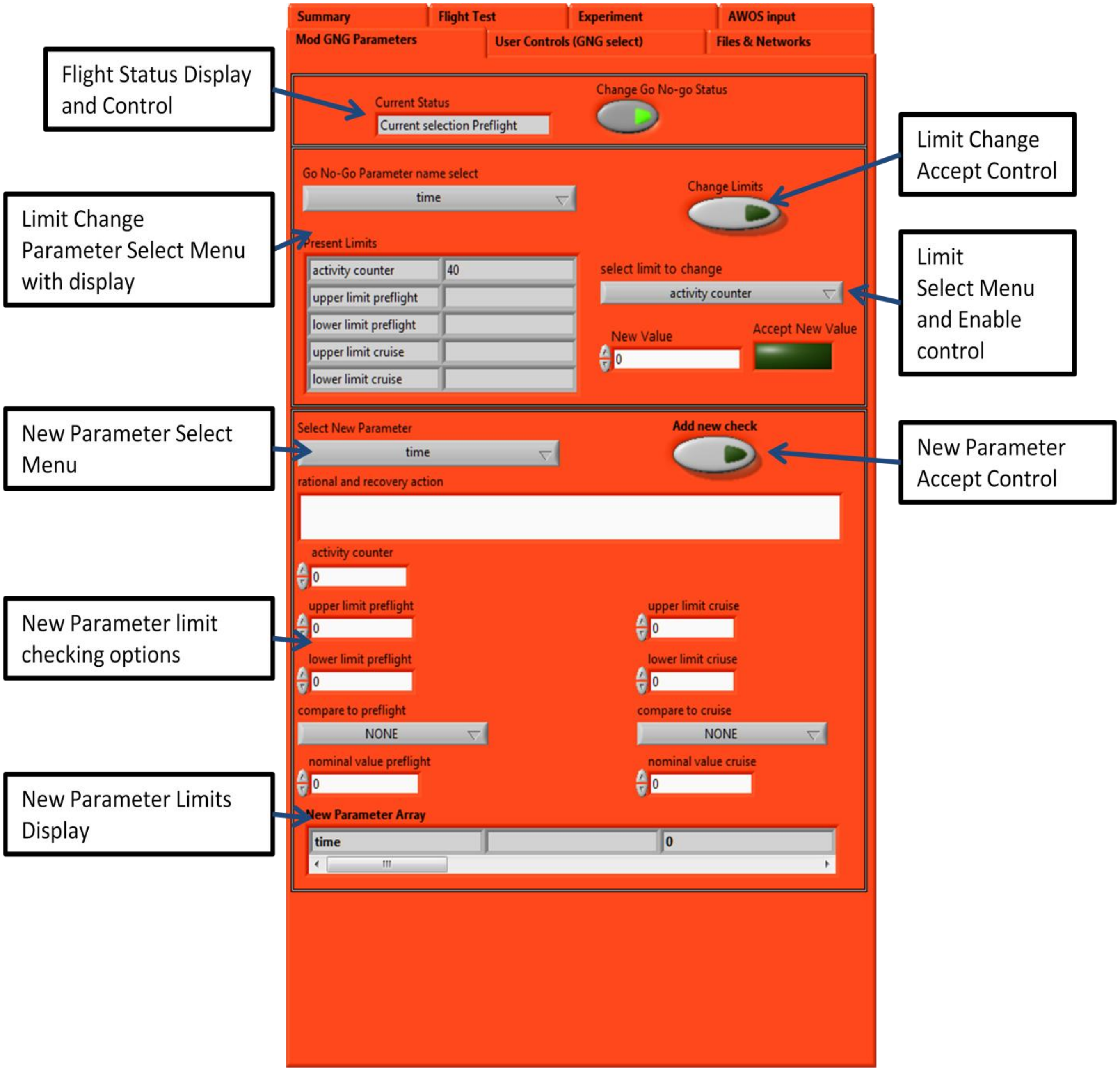
Figure A1.6: “AWOS Input” page

This page allows the user to enter AWOS data at any stage during operations. This data is then used to calculate expected temperatures and pressures which will be used to check parameter data.

Discussion of Functions:

- AWOS Enable Control: This control permits data on the AWOS page to be used by the program for calculations. If not green, the AWOS data is ignored.
- Pressure Control: It is to be used to enter AWOS pressure in inches of mercury.
- Temperature Control: It is to be used to enter temperature in degrees Celsius.
- Dew Point Control: It is to be used to enter Dew point as provided by AWOS.

Page 5: Modify Go No-go Parameters


Summary
Flight Test
Experiment
AWOS input
Mod GNG Parameters
User Controls (GNG select)
Files & Networks
Flight Status Display and Control
Change Go No-go Status
Limit Change Accept Control
Change Limits
Limit Change Parameter Select Menu with display
Limit Select Menu and Enable control
Accept New Value
New Parameter Select Menu
Add new check
New Parameter Accept Control
rational and recovery action
New Parameter limit checking options
New Parameter Limits Display

Current Status: Current selection Preflight

Go No-Go Parameter name select: time

Present Limits

activity counter	40
upper limit preflight	
lower limit preflight	
upper limit cruise	
lower limit cruise	

select limit to change: activity counter

New Value: 0

Select New Parameter: time

activity counter: 0

upper limit preflight: 0

upper limit cruise: 0

lower limit preflight: 0

lower limit criuse: 0

compare to preflight: NONE

compare to cruise: NONE

nominal value preflight: 0

nominal value cruise: 0

New Parameter Array

time		0

Figure A1.7: "Modify Go No-Go Parameters" page

This page allows the user to manipulate Program settings and Parameter limits during operations.

Discussion of Functions:

- Flight Status Display and Control: Allows the user to switch between “preflight” and “cruise” modes of operation by clicking on the provided Boolean.

  Whenever this option is used, the program will reset which will lead to loss of previous violation information on the Summary Page and creation of new Violation log file.
- Limit Change Parameter Select Menu with Display: This provides the user with a pull-down menu to select any parameter and displays the various limits associated with the parameter.
- Limit Select Menu and Enable Control: The user can select the desired limit which has to be changed for the parameter selected in the previous step. The user then has to enter the new value of the check and click on the accept button.
- Limit Change Accept Control: This Boolean Is just a safety measure which the user has to activate before the new checking value is accepted by the program for data checks.
- New Parameter Select Menu: This pull-down menu can be used by the user to select any parameter available in the configuration lists (which contain overall parameter information).
- New Parameter Limit Checking options: Once a new parameter has been selected, the use has to enter the necessary checking limits and nominal values.

  Nominal values must be added at this stage. If left blank, nominal value is assumed to be zero. Nominal values can be added as a number or by denoting another parameter (with similar output). The pull-down menus under heading “compare to preflight” and “compare to cruise” provide list of all available parameters to be chosen from.

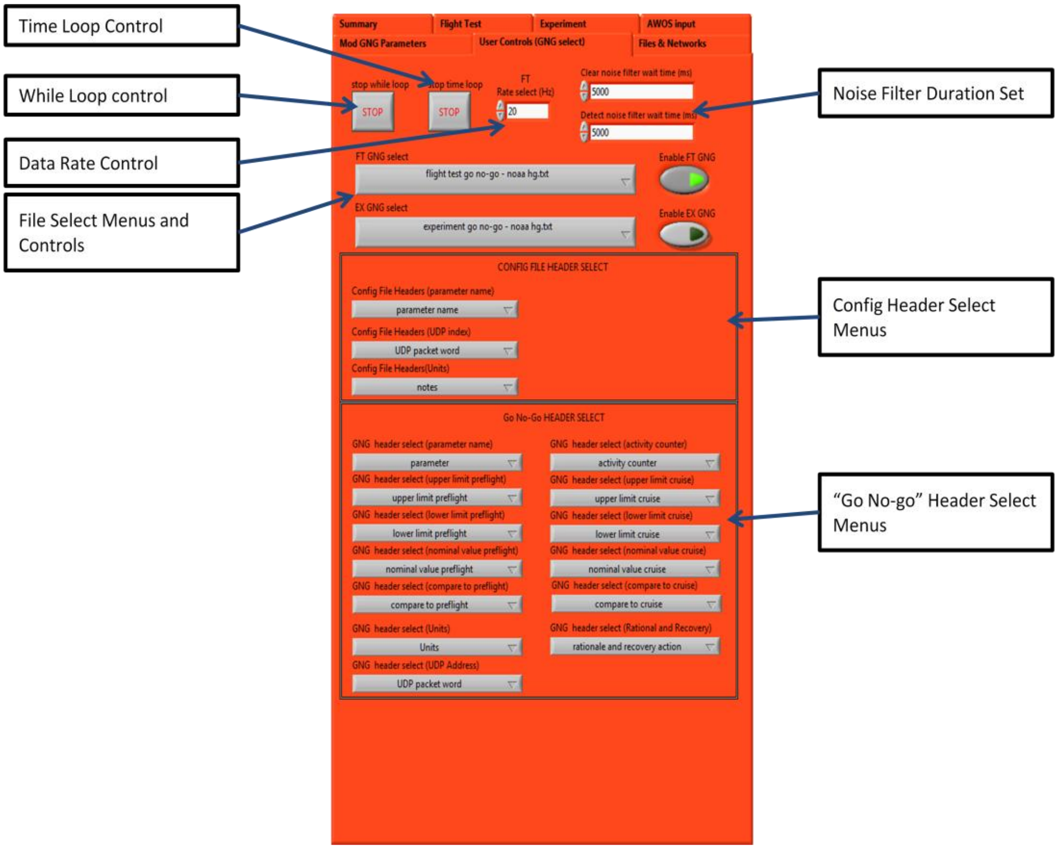
  Remaining checking functions may be added at this stage or later using the “Limit Change” functions described above.

  The “Rational and Recovery Action” description may only be added at this stage. However it may be ignored as it is not critical for data checks.
- New Parameter Accept Control: Once the required data is added to the “limit checking options”, the changes must be accepted by clicking on the Boolean provided.

  When this operation is conducted, the program will reset to accept the new parameter for checking.
- New Parameter Limits Display: This enables the user to review the new parameter that has just been added. In case of error, however, no method exists where the same parameter may be re-entered. Only by restarting the program, can the user add the same parameter again.

  Also, parameters already undergoing checks cannot be modified using this function.

Page 6: User Controls



FT Rate select (Hz): 20

Clear noise filter wait time (ms): 5000

Detect noise filter wait time (ms): 5000

FT GNG select: flight test go no-go - noaa hg.txt

EX GNG select: experiment go no-go - noaa hg.txt

Config File Headers (parameter name): parameter name

Config File Headers (UDP index): UDP packet word

Config File Headers(Units): notes

GNG header select (parameter name): parameter

GNG header select (upper limit preflight): upper limit preflight

GNG header select (lower limit preflight): lower limit preflight

GNG header select (nominal value preflight): nominal value preflight

GNG header select (compare to preflight): compare to preflight

GNG header select (Units): Units

GNG header select (UDP Address): UDP packet word

GNG header select (activity counter): activity counter

GNG header select (upper limit cruise): upper limit cruise

GNG header select (lower limit cruise): lower limit cruise

GNG header select (nominal value cruise): nominal value cruise

GNG header select (compare to cruise): compare to cruise

GNG header select (Rational and Recovery): rationale and recovery action

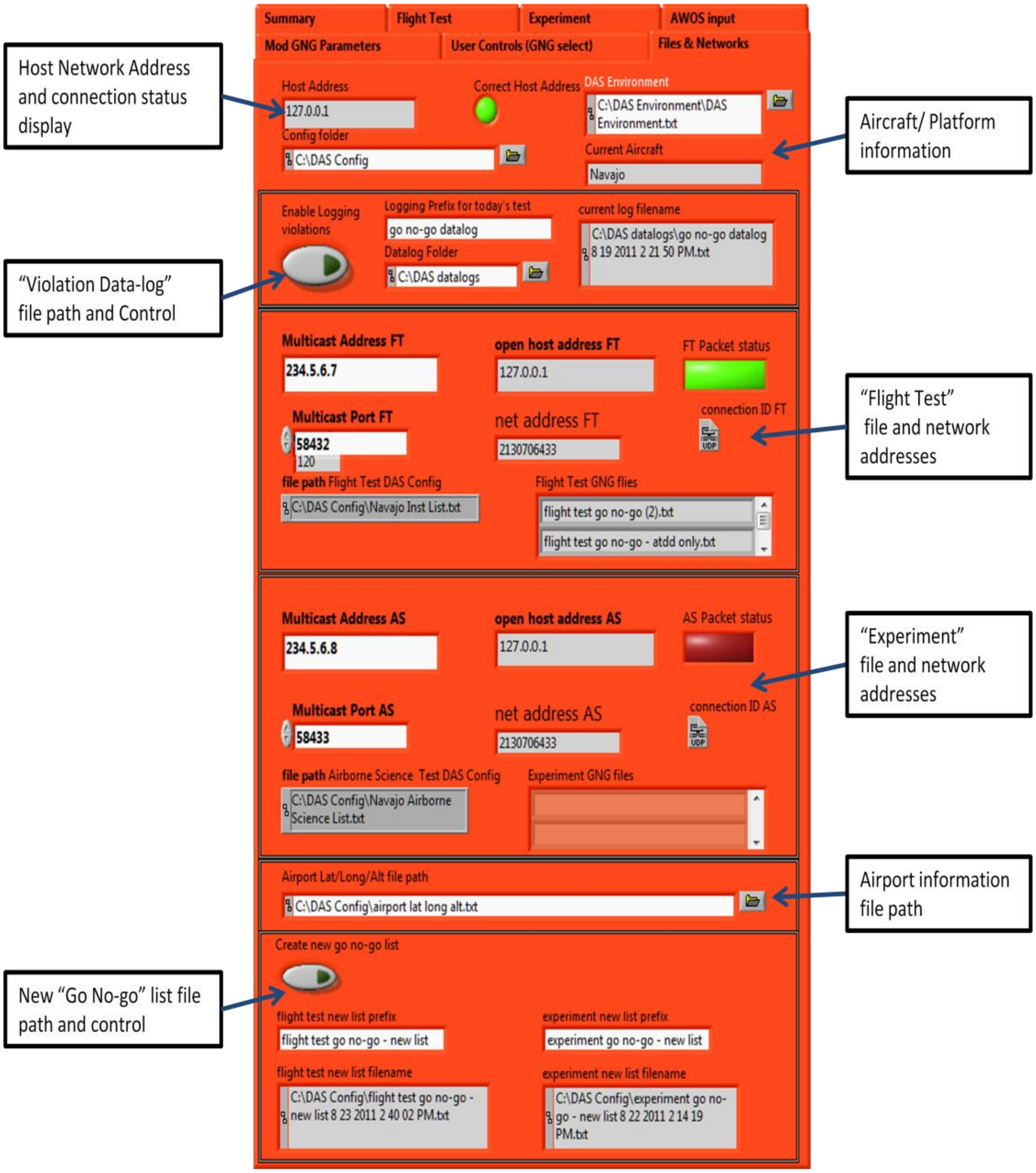
Figure A1.8: "User Controls" page

This page allows the user to select "go no-go" files and data rates.

Discussion of Functions:

- Time Loop Control: Can be used to reset the program
- While Loop Control: Can be used to stop the program
- Data Rate Control: It can be used to set the program cycles to the data acquisition rate. Changes made to this function will reset the program.
- File Select Menus and Controls: Two pull-down menus can be used to select the "Flight-Test" and "Experiment" go no-go files. The Control option has to be selected to enable the program to process the "go no-go" parameters. Changes made to this function will reset the program.
- Config Header Select: The controls provided in this section enable user to select appropriate headers from the configuration files.
- "Go No-go" Header Select: This enables the user to select go no-go file headers for data processing.

Page 7: Files and Networks Page



Host Network Address and connection status display

Aircraft/ Platform information

"Violation Data-log" file path and Control

"Flight Test" file and network addresses

"Experiment" file and network addresses

Airport information file path

New "Go No-go" list file path and control

Host Address: 127.0.0.1
Correct Host Address: 🟩
Config folder: C:\DAS Config
DAS Environment: C:\DAS Environment\DAS Environment.txt
Current Aircraft: Navajo

Enable Logging violations
Logging Prefix for today's test: go no-go datalog
Datalog Folder: C:\DAS datalogs
current log filename: C:\DAS datalogs\go no-go datalog 8 19 2011 2 21 50 PM.txt

Multicast Address FT: 234.5.6.7
open host address FT: 127.0.0.1
FT Packet status: 🟩
Multicast Port FT: 58432
120
net address FT: 2130706433
connection ID FT
file path Flight Test DAS Config: C:\DAS Config\Navajo Inst List.txt
Flight Test GNG flies: flight test go no-go (2).txt; flight test go no-go - atdd only.txt

Multicast Address AS: 234.5.6.8
open host address AS: 127.0.0.1
AS Packet status: 🟥
Multicast Port AS: 58433
net address AS: 2130706433
connection ID AS
file path Airborne Science Test DAS Config: C:\DAS Config\Navajo Airborne Science List.txt
Experiment GNG files

Airport Lat/Long/Alt file path: C:\DAS Config\airport lat long alt.txt

Create new go no-go list
flight test new list prefix: flight test go no-go - new list
experiment new list prefix: experiment go no-go - new list
flight test new list filename: C:\DAS Config\flight test go no-go - new list 8 23 2011 2 40 02 PM.txt
experiment new list filename: C:\DAS Config\experiment go no-go - new list 8 22 2011 2 14 19 PM.txt

Figure A1.9: "Files and Networks" page

This page provides information of data communication and storage performed by the program.

Discussion of functions:

- Host network address and Connection Status: This indicator displays to the user the IP address of the host computer. The connection status Boolean turns green if there is a successful connection. If connection cannot be established, the program is stopped and a dialog box informs the user of connection failure.

- Aircraft/Platform information: This indicator displays the aircraft or platform whose configuration file is being accessed.
- “Config Folder” file path: It shows the location of the “config” folder which contains all files required for the program.
- “Violation Data-log file path and control: Allows the user to initiate/stop violation log. The storage location of log file is shown by the file path. Starting a new log file will reset the program.
- Flight Test file and network address: Gives the connection information required to access flight test data stream and process using go no-go file. “FT packet status” shows green if data flow is positive.
- Experiment file and network address: Gives the connection information required to access experiment data stream and process using go no-go file. “AS packet status” shows green if data flow is positive.
- Airport information file path: This shows the location of file containing airport information.
- New “Go No-go” list file path and control: Allows the user to store a new “go no-go” list. This enables the user to save any changes made to the “go no-go” parameters for future use. New files created will be stored in the “config folder” and thus be accessible by the program for immediate use. If a new go no-go list is created, the program will reset to accommodate the changes.

APPENDIX 2: LabVIEW PROGRAM CODE

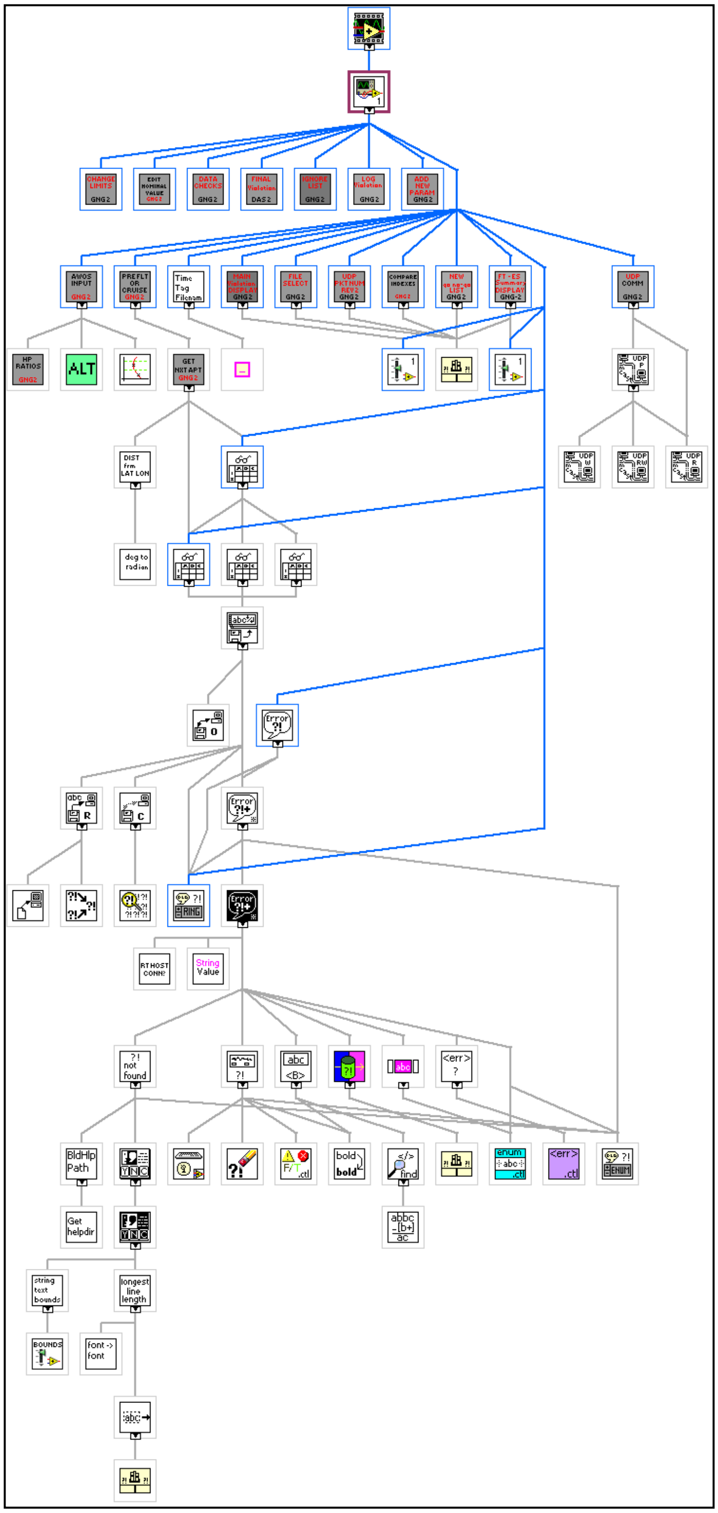


Figure A2.1: “go no-go” Program VI Hierarchy

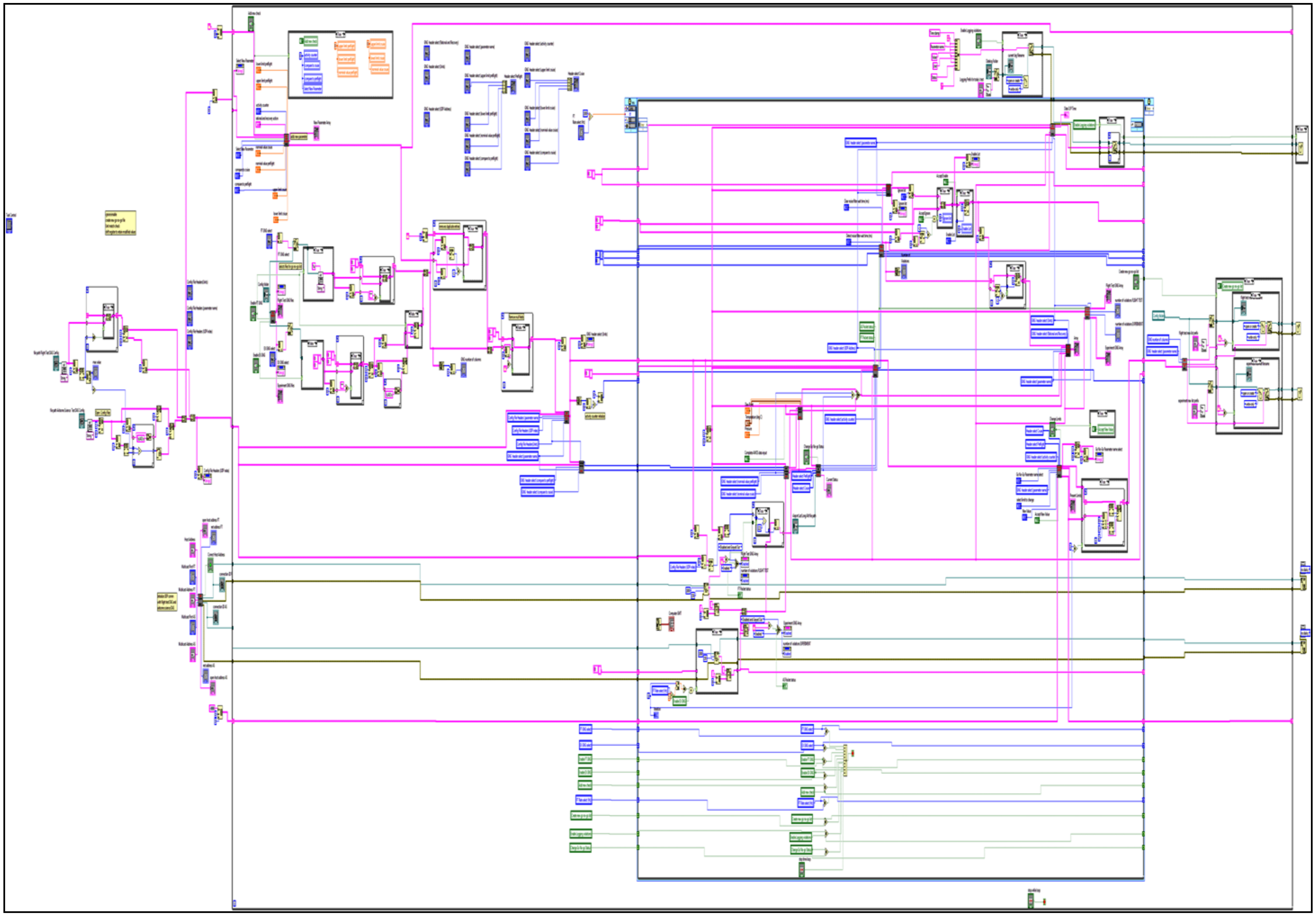


Figure A2.2: “go no-go” Program Block Diagram

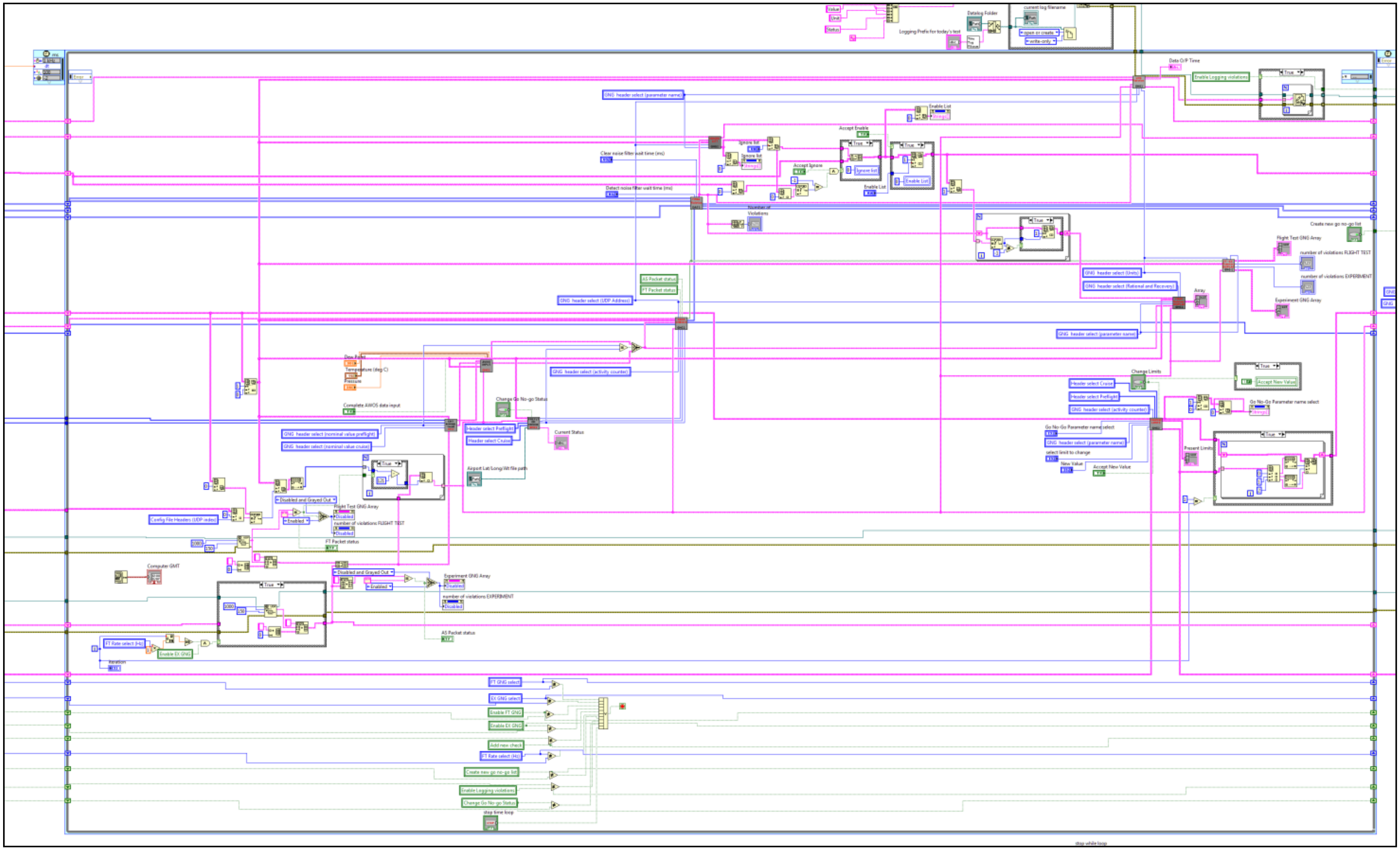


Figure A2.3: “go no-go” Program Timed-Loop structure block diagram

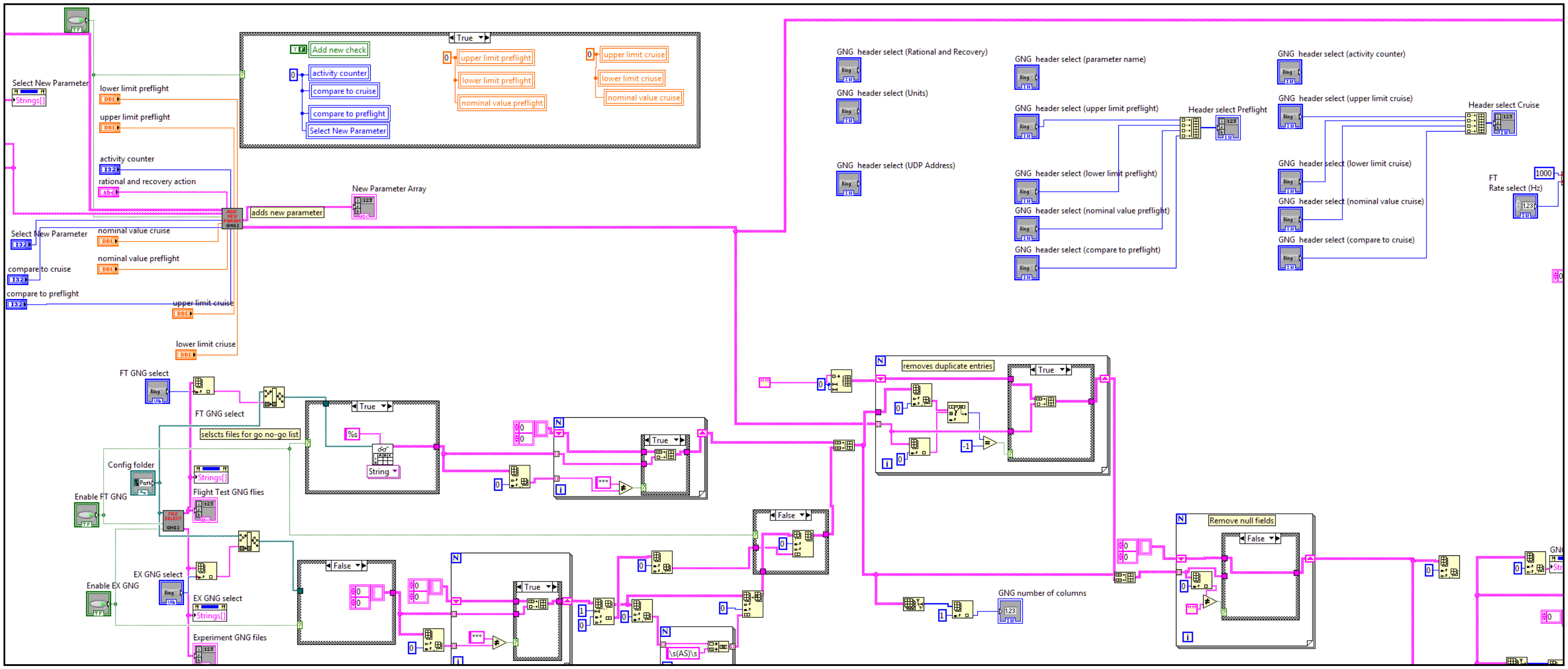


Figure A2.4: Acquiring “go no-go” files block diagram

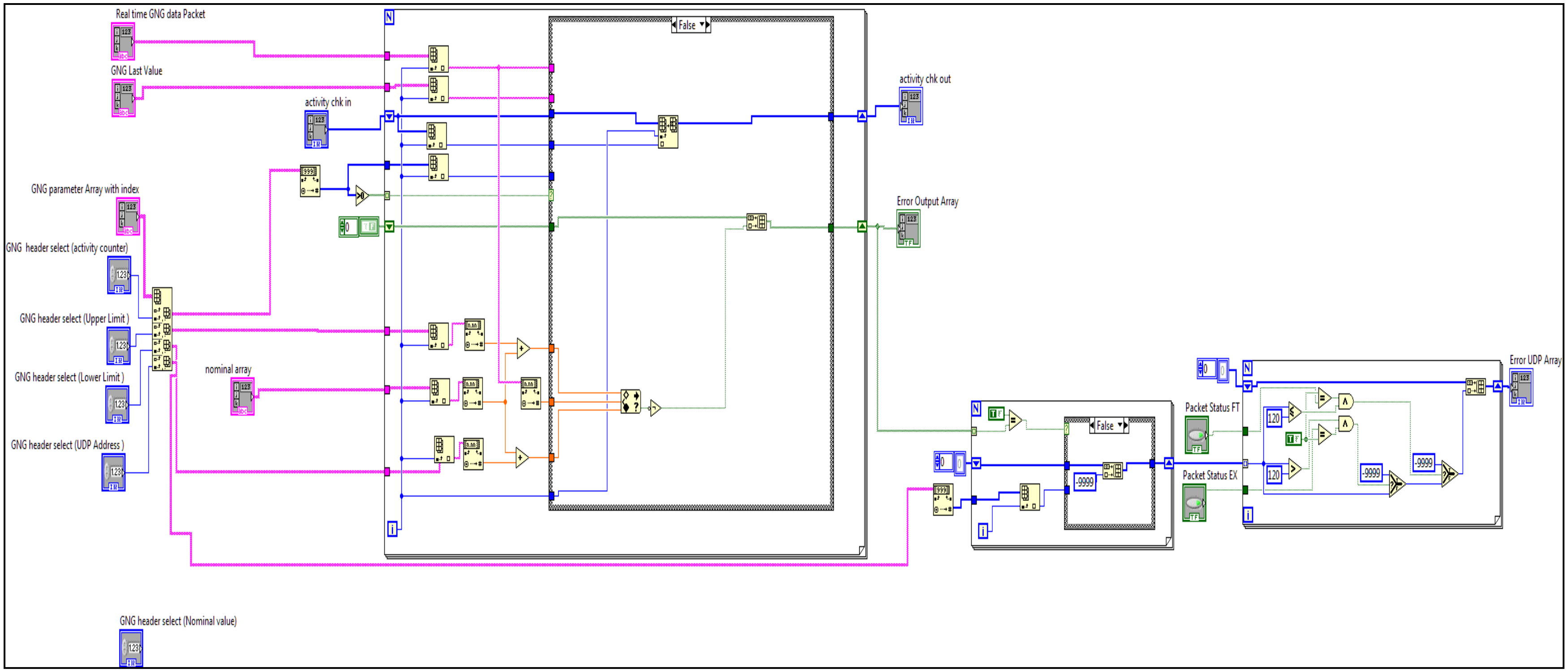

Real time GNG data Packet
GNG Last Value
activity chk in
GNG parameter Array with index
GNG header select (activity counter)
GNG header select (Upper Limit)
GNG header select (Lower Limit)
GNG header select (UDP Address)
nominal array
GNG header select (Nominal value)
False
activity chk out
Error Output Array
Error UDP Array
Packet Status FT
Packet Status EX
False
120
120
-9999
-9999
-9999

Figure A2.5: Data Checks (sub VI) block diagram

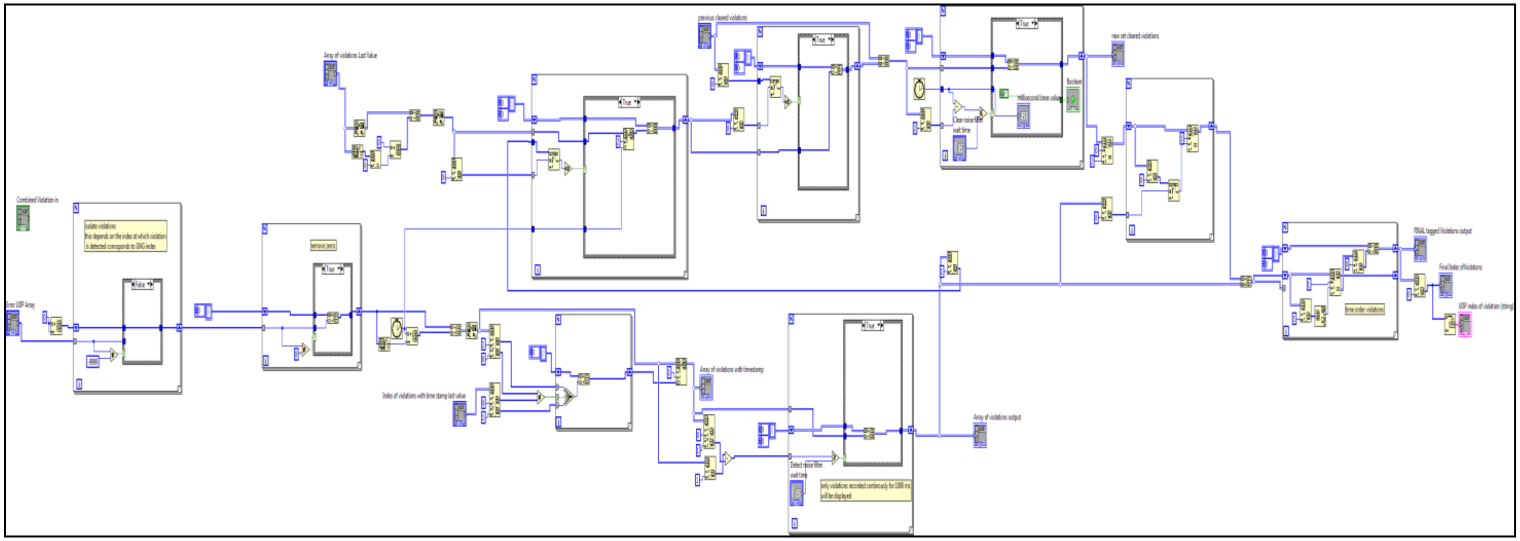

Combined Violation in
Error UDP Array
isolate violations
this depends on the index at which violation is detected corresponds to GNG index
False
remove zeros
True
Array of violations Last Value
True
previous cleared violations
True
True
Clear noise filter wait time
millisecond timer value
Boolean
new set cleared violations
Index of violations with time stamp last value
Array of violations with timestamp
True
Detect noise filter wait time
only violations recorded continously for 1000 ms will be displayed
Array of violations output
time order violations
FINAL tagged Violations output
Final Index of Violations
UDP index of violation (string)

Figure A2.6: Combined Violations (sub VI) block diagram

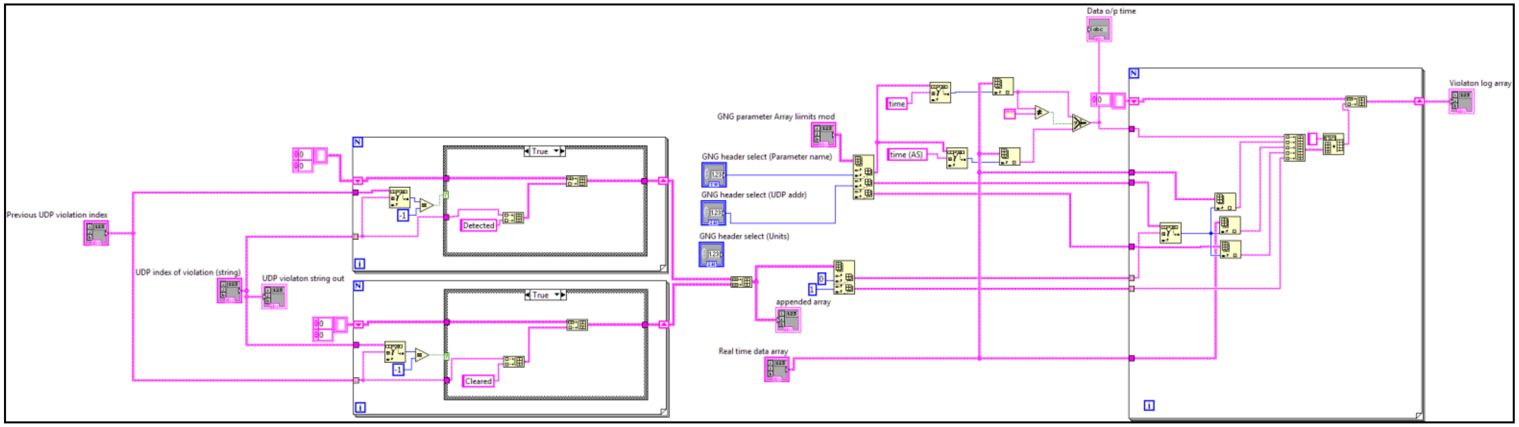

Previous UDP violation index
UDP index of violation (string)
UDP violaton string out
True
Detected
True
Cleared
GNG parameter Array limits mod
GNG header select (Parameter name)
GNG header select (UDP addr)
GNG header select (Units)
appended array
Real time data array
time
time (AS)
Data o/p time
Violaton log array

Figure A2.7: Log Violations (sub VI) block diagram

VITA

Samuel V Williams is a Master of Science in Aviation Systems candidate at UTSI. He holds a Bachelor of Engineering degree in Mechanical Engineering. While at UTSI he has worked on a LabVIEW data acquisition system for multiple aircraft platforms. Samuel has also worked on multiple airborne science missions for NASA and NOAA including over 20 hours as FTE on NOAA ATDD missions. His interests include flight test instrumentation, data processing, handling qualities, flying qualities, performance of fixed wing aircraft, human factors and systems engineering.