A High-Throughput Processor for Flight Control Research Using Small UAVs

Robert H. Klenke[*] and W. C. Sleeman IV[†]
Virginia Commonwealth University, Richmond, VA, 23284

and

Mark A. Motter[‡]
NASA Langley Research Center, Hampton, VA, 23681

There are numerous autopilot systems that are commercially available for small (<100 lbs) UAVs. However, they all share several key disadvantages for conducting aerodynamic research, chief amongst which is the fact that most utilize older, slower, 8- or 16-bit microcontroller technologies. This paper describes the development and testing of a flight control system (FCS) for small UAV's based on a modern, high throughput, embedded processor. In addition, this FCS platform contains user-configurable hardware resources in the form of a Field Programmable Gate Array (FPGA) that can be used to implement custom, application-specific hardware. This hardware can be used to off-load routine tasks such as sensor data collection, from the FCS processor thereby further increasing the computational throughput of the system.

I. Introduction

The use of small unmanned aerial vehicles (UAVs) for aerodynamic and controls system research has obvious benefits, chief amongst which are low initial system and vehicle cost, low operating cost, and low risk to life, property, and the research project itself, in case of a crash. The largest disadvantages of this application of UAVs are related to the small size of the vehicle itself, and are primarily the lack of space, weight lifting capacity, and power system capacity inherent in the vehicle. These limitations impose severe constraints on the flight control system (FCS) used to control the vehicle and record data during the experiment. On the other hand, the significant processing requirements and low latency required for testing multidimensional flight control systems and associated control algorithms dictate that the FCS have a significant computational throughput and that this processing capability actually be on-board the aircraft.

In addition to performing the calculations necessary to execute the algorithms used to control the aircraft and monitor the experiment, the flight control system needs to perform other functions, chief among these are interfacing to sensors and other hardware aboard the aircraft, and manipulating the flight control surfaces of the aircraft. Both of these tasks, if handled directly by the FCS processor, can place a significant additional, processing burden on the FCS processor.

A project was started at NASA Langley Research Center that was designed to allow controls systems researchers to conceive, develop, implement, and flight test highly experimental and perhaps even controversial flight control technologies in a relatively low cost and low risk platform. Simulations provide a useful development tool but the focus here is to develop a testbed to demonstrate these control approaches in actual flight. The Flying Controls Testbed (FLiC)[1], which was developed as part of this project, is a relatively small and inexpensive unmanned aerial

[*] Associate Professor, Department of Electrical and Computer Engineering, School of Engineering, 601 West Main St.
[†] Graduate Research Assistant, Department of Electrical and Computer Engineering, School of Engineering, 601 West Main St.
[‡] Controls Research Engineer, Electronic Systems Branch, Building 1202/Mail Stop 488.

vehicle developed specifically to test highly experimental flight control approaches. The most recent version of the FLiC is configured with 16 independent aileron segments allowing for experimentation with multidimensional flight control problems.

Candidate control applications intended to be tested on the FLiC include a multiple-model controller with dynamic models based on Kohonen's self organizing map (SOM)[2]. This approach, based on a neural network type model, has been shown to provide an effective technique for modeling highly nonlinear dynamic systems[3]. This type of controller however, is very computationally expensive, utilizing matrix manipulations requiring large numbers of integer or floating point operations within the inner control loop.

At the current time, commercial autopilot systems designed for small UAVs emphasize ease of operational use, small size and weight, and expandability in order to hit their target market. As a result, they tend to be based upon commercially available microcontrollers, that while having ever increasing computational power, tend to lag somewhat behind the state-of-the-art in embedded processor development.

For example, the Phoenix AX IMU/GPS Autopilot Module offered by O-Navi (http://www.o-navi.com/phoenix_ax.htm) contains a 32 MHz Motorola MMC-2114 processor. The MMC-2114 is a 32-bit microcontroller from Motorola's M·CORE family of processors and is capable of 31 Dhrystone[4] 2.1 MIPS performance using a 33 MHz clock. In addition to the MMC-2114 processor, the Phoenix module contains 256k Bytes of non-volatile Flash memory for executable program storage and 32k Bytes of volatile SRAM for data (variables) storage. The Phoenix module does provide a serial port for communications with off-board resources such as a communications modem, but only one. Likewise, pulse width modulation (PWM) ports are provided to control the servos used to move the flight control surfaces on a small UAV, but only 6 are provided.

Another example is the Piccolo autopilot system from Cloud Cap Technology (http://www.cloudcaptech.com). The Piccolo system has been used in a number of successful operational UAVs including versions of the Aerosonde Robotic Aircraft (http://www.aerosonde.com). The Piccolo autopilot system uses a version of the Motorola MPC555 microcontroller. While the MPC555 is a 32-bit processor, is compliant with the PowerPC instruction set, and includes a hardware floating point unit, it still has only a 40 MHz clock speed and contains only 448k Bytes of Flash memory and 26k Bytes of SRAM.

While more than sufficient for operational control of most small UAV's, the processors described above do not have the computational throughput to support the requirements of aerodynamic and controls system research using these types of vehicles. Thus in order to offer the researcher the most computational power reasonably available, it is necessary to develop a custom flight control system for some aerodynamic and controls system research projects.

The goal of this project was to develop a flight control system that would be easily capable of executing these complex calculations within the timing constraints of a real-time flight control system. This goal requires that the system have several significant characteristics:

- a state-of-the-art, 32-bit processor with a clock speed as high as possible, and if available, a built-in floating point unit,
- a large amount of memory, both volatile and non-volatile, to hold the executable code for a large application and the large amount of data used in these complex algorithms, and
- a large amount of user-configurable I/O resources necessary to interface to a large variety of sensors used in aerodynamic and flight control system research.

In addition, to further increase performance and ease the development of applications for this system, the following characteristics are desirable:

- available user-customizable logic which can be used to implement complex logic and state machines that can off-load repetitive or long time scale operations from the processor thereby further increasing available processor resources,
- a standard operating system that can provide basic system resource management and functions which can reduce application development time and increase the portability of applications, and
- a robust, powerful, and readily available hardware and software development environment with standard hardware cores and software libraries to reduce application development effort and time.

The following section provides details on the two processor architectures that were evaluated for this flight control system development effort. Section III describes the architecture of the final system that was developed. Section IV describes the flight testing of the system and the current level of functionality, and Section V provides conclusions and outlines future efforts being implemented or considered.

II. Candidate System Evaluation

In considering candidate processors for this autopilot, the search was restricted to include only those that were available from third-party vendors on completely designed and fabricated processor boards. While it would be possible to design and construct a custom board for the system and it would doubtless be smaller and lighter than using an off-the-shelf board, this was deemed not practical for this project. High-performance microcontrollers and microprocessors have large number of I/O pins, which requires them to be placed in complex physical packages like ball grid arrays. Furthermore, in order to provide sufficient memory for complex applications, additional external memory, both Flash and SRAM or SDRAM, must be added to the system. The I/O busses to these external memories are usually complex and fairly high speed, requiring careful design of the circuit board that connects them. These requirements make the design of a processor board a complex and time consuming process and several iterations might be required simply to achieve a working processor system. Thus it was decided early on to restrict the search to processors for which off-the-shelf boards were available.

The majority of the implementation effort in flight control system research involves software development for the on-board processor, so it was vitally important that the selected processor have an efficient and effective software development environment. While all manufacturers of processors make software development environments for the processor available, either from themselves or third-parties, not all are equally efficient, robust, and easy to use. In addition, the development of complex application code is much easier if the processor has a standard operating system (OS) available upon which the application can be executed. A standard OS provides service routines for input/output and system functions, applications for transferring, loading, and executing application code, debugging and performance monitoring capabilities, and advanced features like threads, which can make application development easier. Therefore, part of the processor evaluation effort involved gathering and evaluating information on the software development environment and operating systems available for each candidate.

Finally, each candidate processor board had to be evaluated for its physical characteristics like size, weight, and power consumption. In addition, while not the overriding concern, the cost and availability of each candidate was taken into consideration. As mentioned previously, one of the benefits of the use of small UAVs, like the FLiC, for research applications is their low cost, which reduces the risk to a project should an aircraft be lost. Just as the airframe should be low-cost for this reason, so should the processor board. It should be noted that there are many processor boards with higher performance than the candidates that were ultimately chosen for further evaluation. However, it makes little sense to utilize a processor board that costs significantly more than the remaining equipment and the airframe itself, unless it is absolutely necessary to achieve the required performance goals.

Using these criteria, two primary candidates were chosen for additional evaluation including the development of a prototype FCS.

The first candidate was the Connex 400 MX board from GumStix (http://www.gumstix.com). The 400 MX contains an Intel XScale® PXA255 400MHz processor. The PXA255 is an Intel implementation of the popular 32-bit ARM (Advanced RISC Machine – http://www.arm.com) instruction set architecture. The 400 MX board includes 64 M Bytes of SDRAM memory and 16 M Bytes of Flash memory. It requires a 3.4V to 5V power supply and draws only 250 mA at 400 MHz. The board itself is 80 mm by 20 mm, and includes only a single serial port for communications but two expansion connectors are provided to allow connections to boards with additional resources. The Connex MX board does not include any user-customizable logic. The PXA255 processor on the 400 MX runs version 2.6 of the Linux operating system and a full GNU cross-compiler is available. The Connex 400 MX is shown in Figure 1.

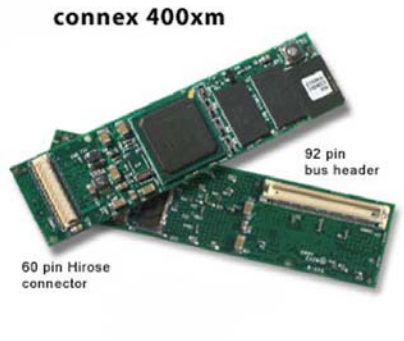

Figure 1. The Gumstix Connex 400 MX processor board

The second candidate was the Suzaku V board from Atmark Techno, Inc. The Suzaku V is based on the Xilinx Virtex-II Pro XC2VP4-FG256 field programmable gate array (FPGA). The Virtex-II FPGA contains a PowerPC 405 32bit RISC processor operating at 270MHz. In addition, being that it is an FPGA, the Virtex-II device contains user-customizable logic that can be used to implement application-specific functions. The logic can be interfaced to the processor via a built-in On-chip Peripheral Bus (OPB).

In addition to the Virtex-II device, the Suzaku V board contains 32 M Bytes of SDRAM and 8 M Bytes of Flash memory and a 10Base-T/100Base-T Ethernet interface. The PowerPC in the Virtex-II runs Version 2.4 of the Linux kernel. Used-defined hardware can be developed for implementation in the FPGA portion of the Virtex-II using Version ISE6.3i of Xilinx's Logic Design Tools. Like the GumStix Connex 400MX, software development for the PowerPC is done using the GNU tool chain including a full cross-compiler. The Suzaku V processor board is shown in Figure 2.

Figure 2. The Suzaku V processor board

The UAV research group at Virginia Commonwealth University (VCU) has developed an operational FCS for small UAVs based on embedded processors[5]. Currently this system utilizes the Suzaku board based on the Xilinx Spartan-3 FPGA. The Spartan-3-based Suzaku board utilizes a 50 MHz, 32-bit processor called the Microblaze. The Microblaze is a *soft-core* processor. That is, the processor is not actually built in dedicated silicon on the device like the PowerPC processor in the Virtex-II, but is implemented using standard FPGA logic resources just like user-defined custom logic. This approach makes the processor architecture more flexible in that the application designer can actually change it, but results in lower overall processor performance.

In addition to the Spartan-3 device upon which the Microblaze processor can be implemented, the Suzaku board contains 16 M Bytes of SDRAM and 8 M Bytes of Flash memory. The Spartan-3 device on the Suzaku comes pre-configured with Microblaze processor on it with a micro version of Linux called uCLinux. The GNU tool chain can also be used to develop applications for the Microblaze processor on the Suzaku.

The salient characteristics of the two candidate processors are listed in Table 1. For comparison, the characteristics of the Suzaku board used in the VCU autopilot are also included. In order to help assess the performance of the processors on each board, the Drystone integer and Linpack floating point benchmarks were compiled and executed on each board. Note that none of the processors on these boards have a built-in hardware floating point unit, so all floating point operations are performed in software using numerical library routines.

Table 1. Candidate Processor Board Comparison

Processor Board	Processor	Memory	Operating System	Drystone 2.1 MIPS	Linpack MFLOPS
Connex 400 MX	400 MHz 32-bit Intel XScale® PXA255	64 M Bytes SDRAM 16 M Bytes Flash	Linux 2.6	282	4.06
Suzaku-V	270 MHz PowerPC 405 32-bit RISC	32 M Bytes SDRAM 8 M Bytes Flash	Linux 2.4	154	1.04
Suzaku	50 MHz 32-bit Microblaze RISC	16 M Bytes SDRAM 8 M Bytes Flash	uCLinux	NA	0.085

The PXA255 processor on the Connex 400MX board has almost twice the integer performance and almost 4 times the floating point performance of the PowerPC on the Suzaku V board. Part of this increased performance can be explained by the 400MHz clock speed of the PXA255 vs. the 270 MHz of the PowerPC, but at least some of the difference must be the result of more efficient instruction execution in the ARM architecture than in the PXA255. Both processors outperform the Microblaze processor in the Suzaku board by at least an order of magnitude. The 50 MHz Microblaze processor on the Suzaku is in the same class as the processors used in most current commercial autopilots for small UAVs.

III. System Architecture

In order to thoroughly evaluate the two candidate processors for this application, a prototype FCS was implemented and flight tested using each. The prototype was based on, and utilized the flight control and ground station software of, the VCU operational autopilot. However, to reduce system cost and complexity, the VCU autopilot system uses infrared (IR) sensors to detect the horizon and provide pitch and roll information. While adequate for simple aircraft control, this angle measurement system is not precise enough for the aerodynamic and controls system research application. For this application, a Crossbow AHRS400 Attitude & Heading Reference System (http://www.xbow.com) was added in lieu of the IR sensors to provide roll and pitch information to the FCS.

As mentioned previously, in order to maximize the utilization of the processor for executing the control algorithms, user-programmable logic was required in the system to be used to implement interfaces to the on-board sensors and perform other tasks as necessary. The Suzaku V board already contains user-programmable logic in the FPGA. For the Connex 400 MX board, an external FPGA prototyping board was interfaced to the processor board to provide this capability.

In all cases, the design of the system hardware implemented in the user-programmable logic was handled similarly. The VHDL hardware description language[6], was used to specify and simulate the custom logic. Each custom logic block, or *core*, was designed to interface to the processor over a standard I/O bus. Therefore, the processor's interface to each core consisted of simple I/O read and write operations. After the core was designed and simulated using VHDL to ensure proper operation, it was mapped to the resources in the FPGA using the Xilinx ISE6.3i FPGA development tools.

The architecture of the FCS is shown in Figure 3. The processor, its associated memory, and the Linux operating system is shown as a separate subsystem in the diagram. With the exception of device drivers needed to interface to peripherals, no modifications to the processor/memory/OS subsystem were made for this application. All of the custom hardware added for this application was interfaced to the processor via an external I/O bus, in the case of the Suzaku V, the OPB bus.

A total of three UART modules were needed for this application to provide RS-232 interfaces to the Ublox Antaris GPS receiver, 900 MHz radio modem to communicate with the ground control station (GCS), and the Crossbow AHRS400 Attitude & Heading Reference System. Depending on the specific processor board, some of these UARTs were available within the processor itself. Additional UARTs were added to the system by implementing them in the FPGA as required.

In addition to the UARTs, an interface was required to the Analog to Digital Converters (ADCs) used to digitize the outputs of the analog sensors. These analog sensors included the pressure sensors for airspeed and altitude, the IR sensors, where applicable, and the measurement of the state of the onboard batteries. The ADCs are designed interface to other devices via a Serial Peripheral Interface (SPI) bus. An SPI interface core was implemented in the FPGA resources to manage the SPI communications between the processor and the ADCs. This core allowed the processor to control and received data from the ADCs by performing simple I/O register read and write operations.

In addition to the sensors, the FCS needs to monitor the values of the input commands from the safety pilot. An external UAV switch handles the switching of control from the safety pilot to the FCS. This device, developed by Electrodynamics, Inc. (http://electrodynam.com) allows completely independent operation of the UAV by the safety pilot regardless of the state of the FCS. Switching of the inputs to the servos from the safety pilot receiver to the FCS is controlled by one of the channels from the safety pilot receiver. Thus, the safety pilot can turn over control of the UAV to the FCS and take it back at any time. The requirement for the capability to do this is in a system intended to test unproven hardware, software, and flight control algorithms is obvious.

Even with the UAV switch, however, the FCS still needs to read the inputs from the safety pilot receiver. The state of the channel which controls the safety pilot switch (i.e., the manual/autopilot mode switch) is used to tell the FCS when it should begin execution of the algorithms for control of the aircraft. The values on the channels which control the UAVs control surfaces, such as elevator, aileron, rudder, and throttle, are used by the FCS to determine neutral trim positions, and are telemetered to the ground control station by the FCS for debugging purposes.

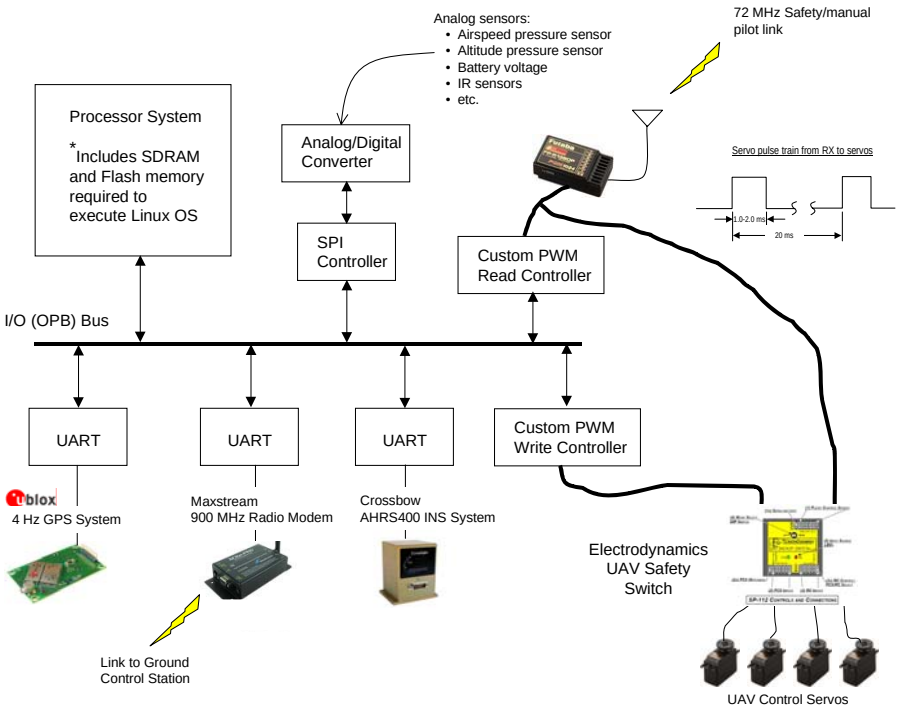

Figure 3. Architecture of the Flight Control System

The pulse train output by the safety pilot receiver, which normally goes directly to the servos, is shown in Figure 3 as well. This pulse train has a period of 20 ms and the individual pulses are 1.0 ms to 2.0 ms wide. The servo output position is proportional to the width of the individual pulses. Because the period, and even the width of the pulses, is very long with respect to the processor's clock period, it is not efficient for the processor to be directly involved in reading the width of the individual pulses. Therefore, a custom core was developed using VHDL and the FPGA development tools, to read the incoming pulses from the safety pilot receiver. This core contains a 16-bit counter for each incoming channel. When the start of a control pulse is detected, this counter is started, and when the end of the pulse occurs, the counter is stopped. The input clock to the counter was setup with a period such that the difference between the resulting count for a 1.0 ms pulse and a 2.0 ms pulse resulted in 10 bits of resolution. Once the counter has stopped, the contents of the counter is placed in a register corresponding to the specific channel. To obtain the value of the current command pulse width for any incoming channel from the receiver, all the processor has to do is perform a simple I/O read from a specific I/O address.

In a similar manner, a custom core was developed to generate the required output pulses to operate the UAVs servos when the FCS is operating the aircraft. In this case, the processor performs a simple I/O write operation to the specific address for the servo in question. The core then uses this value to load a counter every 20 ms. This counter is then started and outputs a pulse that begins when the count begins and ends when the count reaches zero. The core repeats this process every 20 ms using the current values stored in each register. Therefore, the processor only needs to write a value out to a specific servo's control location when it desires to change the value of the current pulse width (i.e., when it wants to move that servo).

In addition to the components described above, additional hardware resources can be added to the user-programmable logic inside the FPGA as required to assist the processor in performing the flight control function. Examples might include custom cores to operate additional types of actuators such as pneumatic or electromagnetic components used in different types of "morphing" aerodynamic structures, custom cores to generate pulse width modulated signals for AC motor control for electric aircraft, or interfaces to complex sensors that use some other

communications protocol besides RS-232 or SPI. These cores can be designed and simulated using VHDL and added to the FPGA as long as additional resources are available within it.

IV. Flight Testing

Each processor board was tested by porting the flight control software from the VCU autopilot system to it. Because this autopilot system had already been ported to uCLinux on the Suzaku board as part of another project, the port to Linux on the candidate processors was fairly straight forward. In addition to being ported to the candidate processor boards, the flight control software was modified to interrogate the Crossbow AHRS400 Attitude & Heading Reference System as required to obtain the current pitch and roll angles. Each system was bench tested using the hardware-in-the-loop simulation capability present in the VCU autopilot system.

Once bench testing of each system was completed, it was installed in a UAV for field testing. The UAV was a modified FQM-117B Radio Controlled Miniature Aerial Target (RCMAT). The FQM-117B is a foam aircraft built in the form of 1/9th-scale model of a MiG-27. Several modifications were incorporated in the UAV, including strengthening the fuselage and wings for the greater weight load, installation of a larger engine, and installation of tricycle landing gear. Figure 4 shows one of the UAVs used for testing and the Suzaku V-based FCS installed in an aircraft.

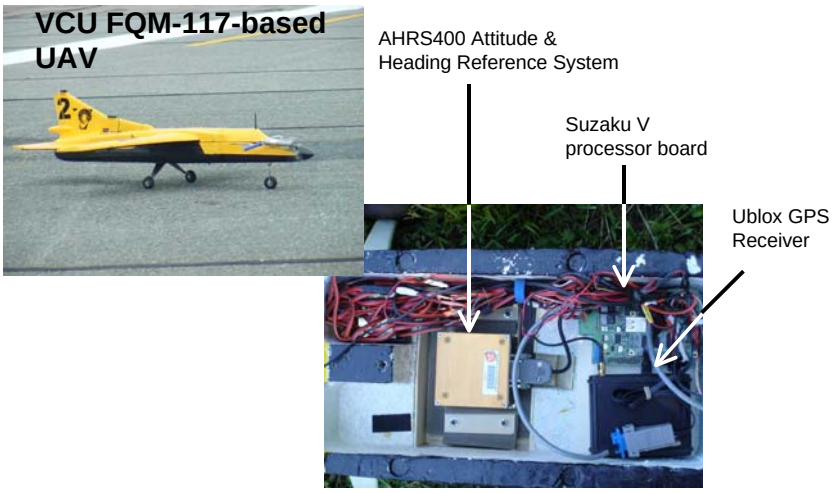

Figure 4. Test setup for the Suzaku V-based autopilot

During flight testing, the ground control station (GCS) developed as part of the VCU autopilot system was utilized to monitor the flight of the aircraft, the performance of the FCS, tune the FCS parameters, and record flight data. Several test flights were conducted for each system. Figure 5 shows an example GCS display from a test flight and an example of the data collected.

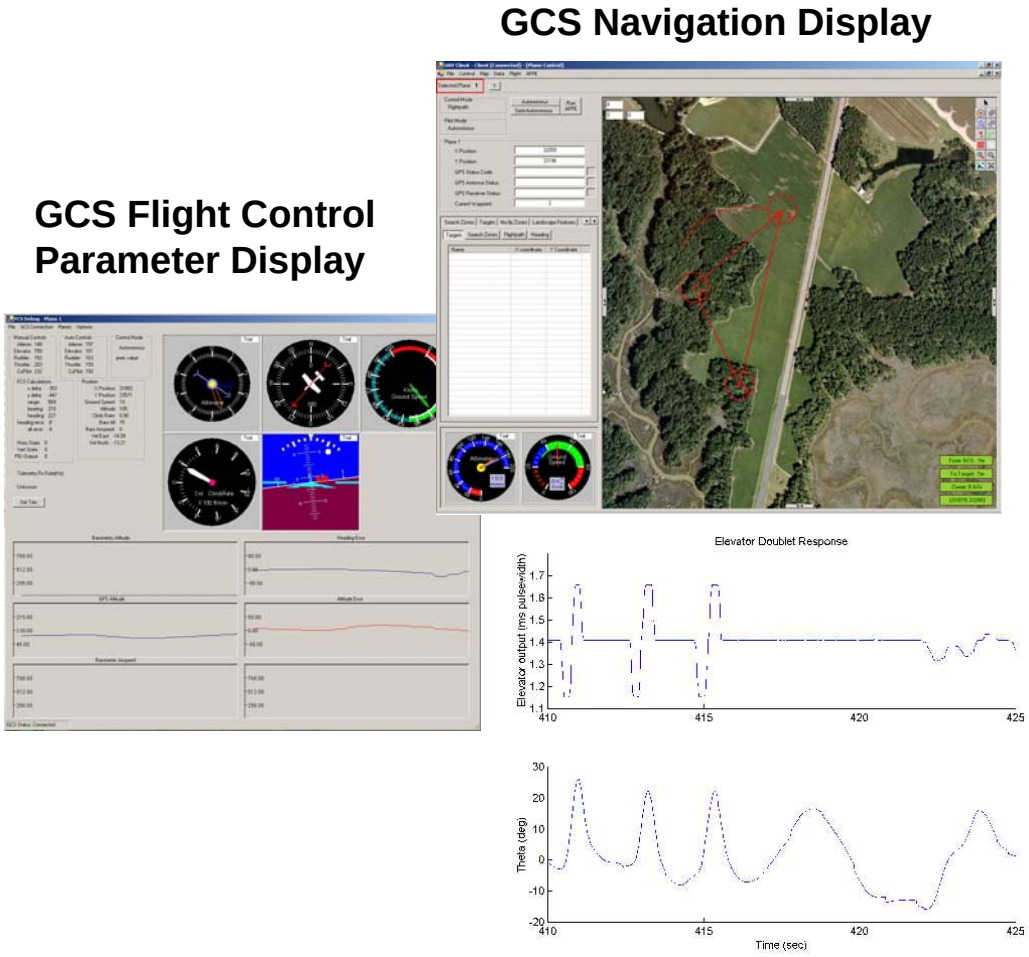

Figure 5. Ground Control Station (GCS) display and example recorded flight data

V. Conclusions and Future Work

The authors have developed an FCS for small UAVs that has the processor throughput necessary to execute complex flight control algorithms. As such, it is ideally suited for use in conducting aerodynamic and control systems research. Two candidate processor boards were identified and prototype FCSs based on each were constructed and flight tested. At this point in time, it is intended that the research will move forward with the Suzaku V-based system. Although the Suzaku V processor board has a lower computational throughput than the Connex 400 MX board, the Suzaku V has the advantage of having built-in user programmable logic on-board whereas the 400 MX board requires these resources to be added on an expansion board. Experience has shown that having the ability to implement additional logic for such functions as sensor interfaces, in user-programmable logic within the FCS can offload a significant amount of work from the processor.

One of the major benefits of the approach used to develop this system is portability. Because the system and associated software were developed around off-the-shelf boards using an industry standard operating system and tool chain, porting the system to other processors that use the same OS and development tools will be a relatively easy task. This was shown in the development of the systems described herein in that the application was ported to both candidate boards within a relatively short time period. This portability will allow researchers to utilize the latest in processor technology as it is developed.

Future work of course includes implementing research algorithms such as the SOM-based approach described earlier, on the system. Additional improvements to increase processor performance are also planned. For example, it is possible to add a floating point co-processor to the PowerPC in the Suzaku V system by implementing it in the FPGA. This improvement could increase the MFLOP rating of the Suzaku V system by an order of magnitude or more. In addition, the development of a sensor expansion board, which includes FPGA resources for use with the Connex 400 MX, is also being considered. This development would allow the use of the greater performance of the PXA255 processor on the 400 MX board in the system without requiring the additional weight and complexity of an external FPGA board as part of the system.

References

[1]Motter, M. A., "Autonomous Flying Controls Testbed," *AUVSI Unmanned Systems North America* , AUVSI, Baltimore, MD, 2005.

[2] Kohonen, T., *Self-Organizing Maps*. NewYork: Springer-Verlag, 1995.

[3]Principe, J.C., Wang, L., Motter, M.A., "Local Dynamic Modeling with Self-Organizing Maps and Applications to Nonlinear System Identification and Control," *Proceedings of the IEEE*, Vol.86, No. 11, November 1998.

[4]Weicker, R. P., "Dhrystone: a synthetic systems programming benchmark,"*Communications of the ACM*, Volume 27, Issue 10, October 1984, pp. 1013-1030.

[5]Klenke, R. H., A. Handa, J. E. Quinones, H. K. Van, B. Wynne, "An FPSLIC-based UAV Control Platform," Proceedings of the International Conference on Military and Aerospace Programmable Logic Devices (MAPLD), September, 2004 (CD-ROM).

[6]IEEE, "IEEE Standard VHDL Language Reference Manual," New York, NY, IEEE Std. 1076-2002, August 7, 2002.