



Autopilot Developers Kit

Kestrel Limited Source Code License



Procerus has developed a set of tools and capabilities designed specifically for Developers and Integrators to help facilitate and reduce "time to flight" implementation of the Kestrel Autopilot System. Whether you're a developer wanting to add your own algorithms or you have a need to integrate the Kestrel autopilot into your airframe, these tools will assist and smooth the process. The Procerus Developers Kit consists of the items as listed below and is governed by License Agreement and by paying appropriate fees as applicable. Contact Procerus to obtain a copy of the License Agreement.

| | |
|-----------------|-----------------|
| License Fee | \$7,500.00 (yr) |
| Universities | \$5,000.00 (yr) |
| Yearly Renewal: | \$3,500.00 |

DEVELOPMENT TOOLS

Consists of:

- Kestrel Autopilot Software Libraries – certain libraries provided in source format, others encrypted
- Communication Protocol (serial)
- TCP/IP Development Interface (open source applet provided)
- Hardware-In-The-Loop Simulator

Services Fees:

NOTE: The cost of the Kestrel SDK does not include engineering support. Fees to support customers using our SDK are invoiced at \$150 an hour. A purchase order is required in order to accommodate those charges. Procerus tracks time used and provides monthly reporting to the customer.

Customers outside US: For customers outside the US, the customer estimates amount of time they feel they may need in engineering support and provides funds up front to Procerus to receive that support. As funds get low, Procerus will report to the customer who will then choose to increase funds for further support, as they may choose.



Export:

Kestrel Source Code is export controlled as technical data based on the International Trade in Arms Regulations (ITAR) under the United States Department of State. Domestic customers are responsible for understanding and following all applicable export rules, including rules related to code access by foreign national employees.

Kestrel Autopilot Software Libraries

The Kestrel Autopilot system uses a Rabbit Semiconductor microprocessor for all its onboard processing requirements. The microprocessor can be programmed using a language called Dynamic C. Dynamic C is a variant of ANSI C programming with special operators that take advantage of the Rabbit microprocessor's abilities. Each Dynamic C program has a main file that has an entry point for the software. A user can also include a number of libraries for support. These libraries can be plain text source code or they can be encrypted. Note: Dynamic C is provided by Rabbit Semiconductor and requires a separate purchase through Rabbit Semiconductor to use their integrated development environment.

With the Procerus Development Kit, source code is provided to customers as noted in Table A below. This allows the developer to have control of what source code is loaded onto the autopilot. As an example, developers may choose to replace existing control algorithms with their own. In this case, a developer would comment out the function call to the PID algorithm and reference the new replacement function. The control functions are called from the main file. The replacement function can either be placed in the main file or in a new library created by the developer.

The developer can use the unencrypted libraries as templates to create new libraries. The main file, the communications library, and the header library are provided un-encrypted. These libraries handle the packets being sent from the ground station and the definitions for the autopilot variables. With these two libraries a number of modifications can be made to add customization to the existing ground station/autopilot protocol.

Autopilot Software Library Descriptions

The Kestrel Autopilot software libraries are listed below with a designation indicating either source code format (unencrypted) or non-source library format (encrypted). The purposes of supplying the Kestrel libraries in this way are; 1) to provide LICENSEE the ability to interact with and modify various functions of the autopilot (source libraries), add their own algorithms, and to compile autopilot binaries that can be loaded onto the Kestrel autopilot, and 2) to protect Procerus Intellectual Property.

These libraries, and all technical data contained therein, are the intellectual property of Procerus Technologies and/or its licensors and are subject to the terms of the License Agreement. No unauthorized distribution is permitted in any form.

Table A

| # | Library Name | Description | Encrypted |
|----|-----------------------|--|-----------|
| 1 | Kestrel.c | Main autopilot file that calls out functions from libraries. Source is compiled from this file. (SOURCE CODE) | No |
| 2 | KA_Header.lib | Contains the #define information. (SOURCE CODE) | No |
| 3 | KA_Comm.lib | Contains functions relating to decoding and encoding communication packets between the ground station and autopilot. (SOURCE CODE) | No |
| 4 | KA_Control.lib | Contains feedback control, auto trim, and gain scheduling algorithms. | Yes |
| 5 | KA_Actuator.lib | Contains the functions relating to servo control. | Yes |
| 6 | KA_GPS.lib | Contains functions for reading and parsing data from GPS. | Yes |
| 7 | KA_Hardware_V20.lib | Contains hardware dependent sensor & IO functions | Yes |
| 8 | KA_Hardware_V21.lib | Contains hardware dependent sensor & IO functions | Yes |
| 9 | KA_Hardware_V145.lib | Contains hardware dependent sensor & IO functions | Yes |
| 10 | KA_Mag.lib | Contains functions relating to reading and processing information from the magnetometer. | Yes |
| 11 | KA_Math.lib | Contains matrix and trig manipulation functions | Yes |
| 12 | KA_Modem_Aero.lib | Contains functions for re-programming Aerocomm modems. | Yes |
| 13 | KA_Nav.lib | Contains navigation functions | Yes |
| 14 | KA_Nonlin_Control.lib | Contains failsafes, altitude tracker, and airborne detection algorithms | Yes |
| 15 | KA_Sensor.lib | Contains sensor processing and attitude estimation functions. | Yes |
| 16 | KA_Uutilities.lib | Miscellaneous functions. | Yes |



Communication Protocol (serial)

The Kestrel Autopilot and Virtual Cockpit talk through a proprietary communications protocol by sending data packets serially over a communications link. Developers may choose to replace some or all of the functionality of the Virtual Cockpit by understanding the published communication protocol specification provided in the Developers Kit. Doing so requires a programmer capable of opening up a computer serial port and sending data packets to the Commbus. A disadvantage of working with the pure serial protocol is that only one application can talk to the Commbus at a time so developers would have to replace all of the required capabilities that are already offered in the Virtual Cockpit (i.e. Mapping Tools, PID tuning and configuration). However, this disadvantage is overcome by using the TCP/IP development interface built into the Virtual Cockpit. **Made available under NDA in conjunction with the TCP/IP Development Interface Tools as noted below.**

TCP/IP Development Interface Protocol

Built into the Virtual Cockpit is a TCP/IP development interface socket server. This server accepts connections from outside applications and depending on the data received, the Virtual Cockpit will react to the commands. Using the TCP/IP interface simplifies development time because the Virtual Cockpit will take care of the underlying low level serial communication and packet generation. This also allows developers to continue to use all the inherent features provided by the Virtual Cockpit while still allowing developers to add their own special modifications needed for custom projects.

To help developers get familiar with the TCP/IP development interface, an open source Visual C++ 6.0 MFC application is provided in the Development Kit. This application shows how to interface to the Virtual Cockpit in a simple modular manner and issue some of the common commands such as pass-through packets that will be sent directly through the Virtual Cockpit to the Kestrel autopilot.

An example of using the TCP/IP interface and Developers Kit might entail a developer adding a new sensor to their UAV. Code could be written on the autopilot to talk to the sensor and gather critical information. The information could potentially be requested by the developer's custom written application that sends a request packet through the Virtual Cockpit to the autopilot. The autopilot would respond with the data to the Virtual Cockpit and the Virtual Cockpit would pass it through to the TCP/IP interface. **Made available under NDA in conjunction with the Communications Protocol as noted above.**

Hardware-In-The-Loop Simulator

A necessity of almost all airborne applications is to execute many scenarios of a new project in simulation. The development kit allows developers to test their customized changes in a 3D graphics engine with a 6-degree-of-freedom physics simulation. Not only are developers able to test using a 3D simulator but the simulator interacts with the actual autopilot hardware so developers can know that they are testing as close to the real flight as possible.

The Developers Kit uses an Open Source 3D simulator called Aviones (Aviones Simulator, the open source simulator developed by Brigham Young University, is governed by Open Source GPL. www.gnu.org/licenses) which can be downloaded online but is also provided for convenience on the Developers Kit CD. Aviones uses the concept of dynamic loadable libraries (DLL) to interact between the physics engine and the Kestrel autopilot. This ability allows users to write their own physics equations of motions and load them in Aviones. Procerus Technologies has created a pre-compiled DLL that will allow the Kestrel Autopilot to communicate with the Open Source Aviones simulator.

The physics DLL that comes with the Aviones simulator allows users to adjust airplane coefficients. This allows developers to adapt the simulated aircraft to fly more closely to their particular airframe. The default coefficients provided with Aviones are adjusted to simulate a flying wing aircraft.

A possible scenario in which developers can use the Hardware-In-The-Loop simulator is if they create a new reactive search pattern based off of sensor information. Developers could augment the autopilot code to talk to a new sensor such as a smoke bloom detector and based off of the sensor reading fly a search pattern in Aviones. The entire mission could be simulated on the ground to refine it as much as possible before actual flight.



Kit Contents

Included in the Developers Kit are the following:

- Communications protocol documentation (serial, TCP/IP)
- Software libraries (encrypted, unencrypted)
- Sample open source Virtual Cockpit TCP/IP plug-in application
- Autopilot programming cable with autopilot extension pigtail. (Dynamic C required)
- Aviones Simulator and Kestrel DLL for hardware-in-the-loop simulation
- CDROM (software libraries, open source tools, and manuals)

Not included in the Kit: (must be purchased separately)

- Kestrel Autopilot hardware or software
- Virtual Cockpit ground control software and Commbot ground station wireless communications device.
- Dynamic C interface development environment. www.rabbitsemiconductor.com

The Developers Kit is used with the Kestrel Autopilot System only as per our Software License Agreement and Terms and Conditions.

TCP/IP Developers Kit (included in the above but also made available separately)

Includes Communication Protocol and TCP/IP Development Interface Protocol. Made available free of charge.

Please see: www.procerusuav.com/productsintegrationtools.php

