This document will help you to set up your AscTec UAV to be used with MATLAB/Simulink®. Please read the manual carefully before you start using the software with your hardware.

Please be aware that all experiments done using the AscTec SDK and the described framework can not be controlled by Ascending Technologies. It is your own responsibility to carefully check all systems before flight and to take all necessary safety precautions. The proven flight controller on the low level processor can help you to recover from a lot of flight situations caused by a failure in the implemented high level controller. But be aware, that it might not handle extreme situations.

Content:
1. Index of contents
2. Introduction
3. System requirements
4. Initialization of the onboard model
5. Wireless serial connection (Xbee)
6. Onboard model
7. Parameters, control and debug channels
8. Setting parameters
9. Debug channel configuration
10. UART communication (monitoring)
11. UART communication (send commands)
12. Example: Simple attitude control
13. Generating C-code for the AscTec SDK
14. Setting up Eclipse and the AscTec SDK
15. Connect to the microprocessor (JTAG)
16. Flashing the microprocessor (JTAG)
17. Notes
Introduction

This framework enables you to get your Simulink® model on the high level processor (HLP) of the AscTec AutoPilot. It works on all AscTec UAVs with 2012 firmware.

The Simulink® framework works in combination with the AscTec SDK. It enables the user to develop control systems in Simulink® and translates the code automatically in C-code, which can be flashed to the HLP with the help of the AscTec SDK. The framework also provides a model for UART communication, which helps you to send live commands from your ground PC and to monitor custom debug data.

After simulating the Simulink® algorithms on the ground PC, it is possible to test them directly onboard of the flight system. To activate the control of the UAV by the code on the HLP, the pilot only needs to flick a switch on his remote control. In case of any flight problems, the safety pilot can switch back to the proven flight controller on the LLP to recover the UAV and provide a soft landing.
The Simulink® framework is compatible to the AscTec AutoPilot HL-Simulink SDK and the 2012 low level firmware for the AscTec AutoPilot.

All tools and software versions are tested for Windows 7 on 32/64 bit systems. The Simulink® models are tested for MATLAB/Simulink® version 2011b. Other releases may require minor changes for code generation.

To use the framework, you will need MATLAB Simulink® with the following toolboxes:

- MATLAB
- Simulink
- Embedded Coder
- Fixed-Point Toolbox
- MATLAB Coder
- Simulink Coder
- Simulink Fixed Point
- (Control System Toolbox)
- (Simulink Control Design)
The central part of the framework is the Simulink® model `onboard_matlab.mdl`. This model is the template for all Simulink® based control algorithms for your AscTec UAV.

Please follow the next steps to set up the model for the first time:

1. Unzip „AscTec_HL-Simulink_SDK“. The folder `onboard_matlab` contains all necessary files.
2. Open MATLAB® and change the workspace to the subfolder `custom_mdl`.
3. Run `InitOnboardMatlab.m` to initialize all needed parameters.
4. Open `onboard_matlab.mdl` with Simulink®. It is important not to rename this model. The model in use needs to have exactly the name `onboard_matlab.mdl`.

Please follow the next steps, every time you restart MATLAB®:

1. Open MATLAB® and change the workspace to `custom_mdl`.
2. Run `InitOnboardMatlab.m` to initialize all needed parameters (more information on page 7).
3. Open `onboard_matlab.mdl` with Simulink®. The model in use needs to have exactly the name `onboard_matlab.mdl`. If you are working on more than one model and/or save backups, it’s best to rename them individually and store them in separate folders. If you want to use an older model, copy it back to the subfolder `custom_mdl` and change the name back to `onboard_matlab.mdl`.
The communication to receive debug information and to send commands or parameters from a ground PC is done via Xbee wireless modules.

One pair of Xbees is used for the communication. The UFO-Xbee is connected to the HL Serial 0 port and the other one to any USB port of your ground PC. The USB-Xbee needs to be programmed with the name *Quad_USB1* using *Mprog*:

http://www.ftdichip.com/Support/Utilities/MProg3.5.zip

For the Xbee modules, the FTDI driver is necessary.

Go to: http://www.ftdichip.com/Drivers/D2XX.htm and download the D2XX driver, not the VCP driver.
The Simulink® subsystem **Onboard_Matlab_Controller** is the place to implement your own control ideas and algorithms.

The grey subsystems are used for in- and output mapping. These subsystems convert e.g. the provided attitude data to so called “real world values” using Simulink® fixed-point data types.

The subsystem **Output_Mapping_Debug** is used to select and configure debug channels for transmission to a ground station computer. To monitor this data you can use the Simulink® model **UART_Communication**.
There are three different types of data to send to the HLP from the ground station and back to the ground station:

**Parameters:** Parameters are constant inputs set by the ground station. They can e.g. be used to tune your control algorithm. The parameters are only sent once and therefore can take time for transmission. If a correct parameter set is found, parameters can also be stored permanently on the HLP. On page 8 is shown how to set and store parameters.

```matlab
% parameter definition and calculations
Mass = 0.5;
ACC_Gravity = 9.805;

% map calculated parameters to tunable parameters 1 ... 40
uart_up_p_01 = Mass;
uart_up_p_02 = ACC_Gravity;
uart_up_p_03 = 10;
uart_up_p_04 = 10;
uart_up_p_05 = 10;
```

**Controls:** Controls are sent at runtime to control the system by the ground PC and not by the pilot. A high transmission rate is required and therefore the number and size limited. You can set the controls in **UART_communication**.

**Debug channels:** Debug channels are used to debug the system, to monitor data or as inputs for distributed control tasks. Therefore, there are different requirements to the number and transmission rate, which can be set in the subsystem **Output_Mapping_Debug**.
By default, all parameters are set to zero or to the last stored set of parameters. So before you can test a new controller, the parameters need to be set.

The orange blocks in the Simulink® model are parameter inputs. The parameters are defined in `InitOnboardMatlab.m`.

To send the parameters, first create a comport object in MATLAB®:

```matlab
com = serial('COMxy', 'BaudRate', 57600)
```

You need to enter the comport number of the module `Quad_USB1` (page 5).

To set the parameters, run `parameter_send.m`. The parameters are acknowledged by the beeper. Make sure the model `UART_communication` is stopped before you send the parameters!

**Note**: power cycling the UAV will reset the parameters. You can store the parameters permanently using the script `parameter_store_onboard.m`. This write cycle takes a moment and is only possible for about 2000 times. **Don’t do it in flight, it will lead to a short freeze of the microprocessor!** We also recommend not to change parameters during flight.

For the parameters, a 32 Bit fixed-point data type is used with 15 Bit fraction length. The scaling is done in the `parameter_send.m` on the ground station side and by the data type definition in the “constant” input block in the Simulink® onboard model.

-> You can directly enter and use “real world values” for parameters.
The debug channels are set using the yellow “Go to” blocks. The debug channels are sorted and scaled in the subsystem `Output_Mapping_Debug`. The scaling for transmission by a 16 Bit fixed-point data type is also done in this subsystem.

For transmission, 60 channels are separated in 6 blocks. The picture on this page shows the debug configuration in the subsystem `Output_Mapping_Debug`. The first block, the channels 1…10, are transmitted every cycle (in the standard configuration at 50 Hz). Furthermore, in every cycle one additional block of 10 channels can be transmitted. Channel 10 chooses which additional block is transmitted. This information is used on the ground PC to map the channels accordingly.

The selection of the transmitted block can be done automatically as defined in list iteration or can be selected by the control channel 10. As long as the control channel 10 transmits 0, the list iteration is active. So the blocks change every cycle. Any other number (1 to 5) selects one of the 5 other blocks of 10 channels.

The mapping of the signals itself is done by the “Go to” and “From” blocks. You can copy the “Go to” blocks from `Output_Mapping_Debug` anywhere in the model and name the signal. With the “From” blocks, you can select these signals. For unused “From” Blocks, “Go to” dummies are in the subsystem to avoid error messages.

The scaling of the signals for fixed-point transmission is done by the gains. If you change a scaling here, you also need to change it in `UART_communication`.
To receive the data on the ground PC you first have to compile the s-Functions for your system. It is tested with the compiler of Microsoft® Visual Studio Express which is available for free. The Lcc-win compiler is not working!

Use `mex -setup` to set the compiler and use the script `recompile_sfunctions.m` in the folder `onboard_control/QuadCom` to compile the s-Functions.

To show data, open the Simulink® model `Uart_communication` and start its simulation. Make sure to stop this model before you send any parameters to the UAV as shown on page 8. After flashing the microprocessor with the AscTec SDK, the microprocessor is paused automatically. Then you either need to start the microprocessor in the debug window or restart your AscTec UAV.

The 60 debug channels are all available in the `Quad_data` bus, which comes out of `Debug_Mapping_Extended`. In the standard model, you will find the block `Attitude Control` as an example how to display debug data.

Besides the debug channels, the battery voltage and the onboard CPU-Load (0-1) is displayed. E.g. 0.21 means 21% CPU-Load. A constant 1 means overload, so the update rate will drop below 1 kHz.
To send data to the UAV, open the Simulink® model `Uart_communication` and start its simulation. Make sure to stop this model before you send any parameters to the UAV as shown on page 8.

**Control channels**

The control channels are available in the model `UART_communication`. Controls can be set by a constant block or be replaced by any signal. They are sent during runtime, so due to the high data rate only a limited number of channels is provided.

The control channels can be found in the onboard model in `Signal_Mapping_UART`.

-> To familiarize with the system, send a signal to the UAV by a control channel and receive it by a debug channel. -> play around with different signals, channels and transmission settings. Remember: ctrl_10 can be used to select debug blocks.
To help you getting familiar with the framework, we implemented a simple attitude control example.

The picture shows the **Onboard_Matlab_Controller** subsystem of the sample application:

A simple attitude controller is implemented using a two loop cascade structure.

The subsystem **Command Filter** is used to integrate the yaw rate commanded by the pilot to a commanded heading, taking into account the flip at 2*pi and a reset to the IMU heading measurement at zero thrust. The roll and pitch command is scaled to +- 30°.

The attitude controller is separated in two loops, the rate loop as inner loop and the attitude loop as outer loop. In the subsystem **Control Allocation**, the roll, pitch, yaw and thrust signals are “mixed” to the four motors.

Please note: in this example nonlinearities and scaling to “real world values” are neglected and the controller is adjusted by empirically set controller gains.

The output of the system are the commands for the four rotors in a range of 0….200 which are mapped in the motor controller to the rotational speed up to 8000 rpm. **A value of 0 will stop the motor. That is/should be avoided by a limit.**

In **Output_Mapping_CMD** the control mode is set to individual motor commands for this example. We will publish further examples in the future, which will show how to use the other available command modes.
The control algorithms can now be implemented in the Simulink® model. Doing so, it is important to keep in mind that the generated code will be executed on a microprocessor hardware. Therefore, in the example, all signals are implemented as fixed-point data types. Please refer to the Simulink® fixed-point tutorial for more details.

In general it is also possible to use floating point data types (better float than double). Doing so, check the microprocessor CPU-load in UART_communication. In the default setting, floating point data types are not allowed in the Real Time Workshop options. Also not all floating point operations are supported on the microprocessor target.

Some general hints:

- Do intense calculations as far as possible only once in the model and reuse the data. E.g. the trigonometric functions of the attitude angles.

- Do the fixed-point conversion based on physical considerations of signal range and signal precision. Limit signals to reasonable values. Doing so, it is much easier to either select fixed-point data types by hand or to use the fixed point tool/advisor.

- Check all fixed-point data types by displaying the data types on the signals and avoid data types using more than 32 Bits.

- If any functions are not supported for fixed-point data types or on the microprocessor, use look-up tables to approximate these functions. Use e.g. the function fixpt_look1_func_approx to generate the table data.

To generate C-Code, go to: Tools -> Real Time Workshop -> Build Model

The embedded coder now creates C-code of the model and stores it to onboard_matlab_ert_rtw.

A report of the generated code is displayed afterward.

-> The auto generated C-code is ready to be flashed to the high level processor with the AscTec AutoPilot HL-Simulink SDK in the Eclipse environment.
Install the `armsdk_setup_v0.5.exe` package, which comes with the AscTec SDK 2.0. It will install Eclipse, ARM-GCC, OpenOCD and JTAG-drivers on your system.

**Eclipse**

Choose your workspace and move the folder “custom.mdl” into your workspace folder.

Go to: File -> Import Project

General -> Existing Projects into Workspace

Select root directory -> Browse… -> select AutoPilot_HL-Simulink_SDK_v2.0 -> Finish

Eclipse automatically starts to build the project after the import. If an error message appears, restart Eclipse and build again pressing ctrl+B.

It is important to generate any C-code by MATLAB® before importing the Eclipse project. If not, the script for combining both the MATLAB® and the framework code won’t be able to locate the MATLAB® code and an error will appear.
Connect the JTAG adapter to the processor and switch on your AscTec UAV.

The first time you work in this workspace, set the configuration as follows:

1. Select OpenOCD Asctec – JTAG in “External Tools Configurations…”

2. Press “Run”: Eclipse now automatically builds the project and connects the JTAG adapter. In most cases, the first time starting the JTAG after connecting it to the USB port an error message is displayed. -> Start the JTAG again by pressing “Run As -> OpenOCD Asctec – JTAG”
Select Zylin Embedded debug (Native) -> Asctec JTAG Debug “Debug Configurations…”

Eclipse now flashes the code and changes to the debug mode. After that, start your code in the debug window or restart your AscTec UAV, that the code is running on the HLP.
The provided tools are based on a cooperation between

- Institute of Flight System Dynamics, Technische Universität München
- Ascending Technologies GmbH

We like to thank our partners at the Technische Universität München for their great help concerning this framework. We also like to thank The Mathworks for their cooperation and support.

The hardware, the UAV and accessories, can be purchased from Ascending Technologies.

The functionality to have support for the Simulink® framework on the low level processor is available from Ascending Technologies.