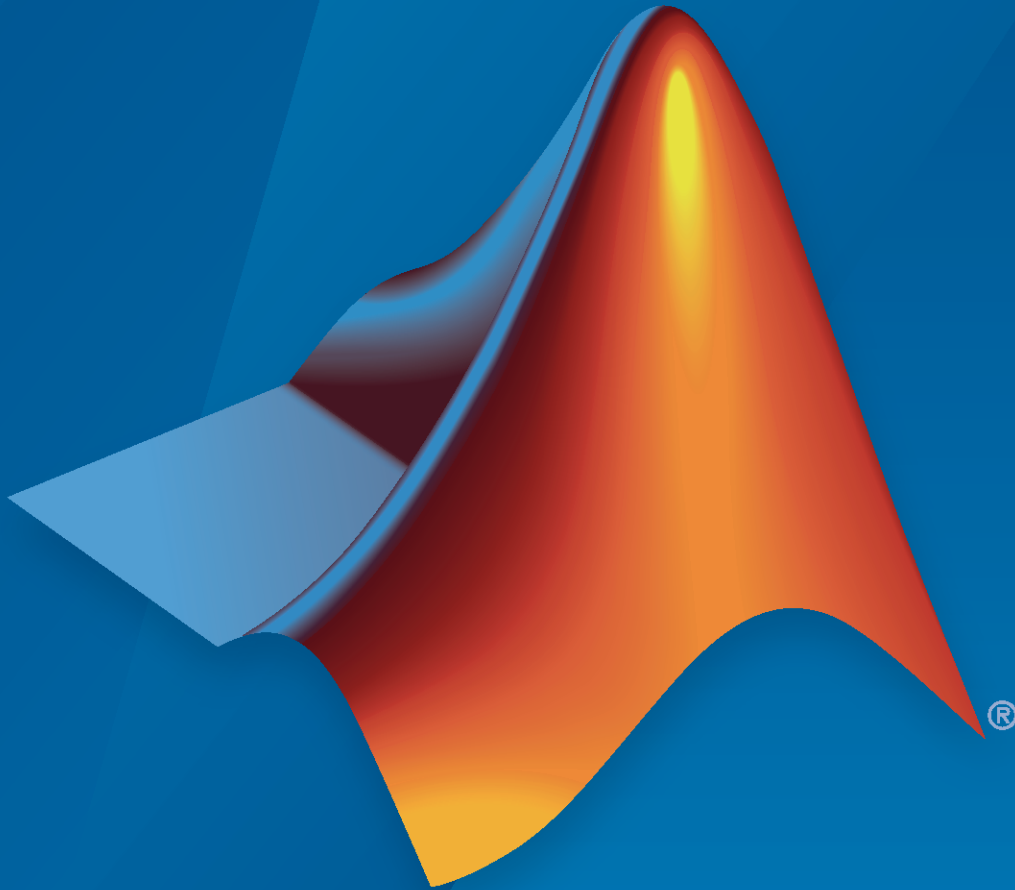


UAV Toolbox Support Package for PX4[®] Autopilots

User's Guide



MATLAB[®]&SIMULINK[®]

R2021a



How to Contact MathWorks



Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

UAV Toolbox Support Package for PX4® Autopilots User's Guide

© COPYRIGHT 2019-2021 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

March 2019	Online only	New for Version 19.1.0 (R2019a)
April 2019	Online only	Revised for Version 19.1.1 (R2019a)
May 2019	Online only	Revised for Version 19.1.2 (R2019a)
September 2019	Online only	Revised for Version 19.2.0 (R2019b)
November 2019	Online only	Revised for Version 19.2.1 (R2019b)
March 2020	Online only	Revised for Version 20.1.0 (R2020a)
April 2020	Online only	Revised for Version 20.1.1 (R2020a)
September 2020	Online only	Revised for Version 20.2.0 (R2020b)
March 2021	Online only	Revised for Version 21.1.0 (R2021a)

Setup and Configuration for UAV Toolbox Support Package for PX4 Autopilots

1

Install Support for UAV Toolbox Support Package for PX4 Autopilots . . .	1-2
Operating System Requirements	1-2
Install, Update, or Uninstall Support Package	1-2
Hardware Setup	1-3
Integration with General PX4 Architecture	1-5
General PX4 Architecture	1-5
Supported Simulink Blocks that Interface with the PX4 Modules	1-7
Supported Modules to be Replaced with User-Defined Algorithms	1-8
Custom Startup Script in UAV Toolbox Support Package for PX4 Autopilots	1-10
Modules Enabled Using Custom Startup Script	1-10
Modules Disabled Using Custom Startup Script	1-10
Impact of Disabling MAVLink, Commander, and Navigator Modules . . .	1-12
Impact of Disabling MAVLink	1-12
Impact of Disabling Commander and Navigator Modules	1-12
Setting Up Cygwin Toolchain and Downloading PX4 Source Code	1-15
Setting up PX4 Tool Chain on Ubuntu 18.04	1-20
Download PX4 Firmware v1.10.2	1-20
Set up PX4 Toolchain on Ubuntu 18.04	1-20
Downloading PX4 Source Code as a Standalone in Windows	1-21
Download PX4 Source Code from Github	1-21
Downloading PX4 Source Code in Ubuntu 18.04	1-22
Download PX4 Source Code from Github	1-22
Selecting PX4 Autopilot Application	1-23
Performing PX4 System Startup from SD Card	1-24
Troubleshooting Test Connection Error	1-26
Troubleshooting Firmware Build Failures	1-28
Troubleshooting Connected I/O	1-29
Troubleshooting Connected I/O with PX4 Host Target	1-29

2

Communicate with Hardware in Normal Mode Simulation Using Connected I/O	2-2
Supported PX4 Boards and Blocks with Connected I/O	2-2
How Connected I/O Works	2-3
Connected I/O in Model-Based Design	2-4
How Connected I/O Differs from External Mode	2-5
When to Use Connected I/O	2-6
How to Enable Connected I/O	2-6

Run on Target for UAV Toolbox Support Package for PX4 Autopilots

3

Supported PX4 Autopilots	3-2
Enabling MAVLink in PX4 Over USB	3-5
Plant and Attitude Controller Model for Hexacopter	3-7
Simulate the Plant Model for Hexacopter	3-7
Generate Code for Controller for Hexacopter	3-8
Adapting the Plant and Attitude Controller for Other Airframes	3-9
Migrating from Pixhawk Pilot Support Package to UAV Toolbox Support Package for PX4 Autopilots	3-10
Integrating uORB Topics in the Simulink Model	3-16
Setting Up the Hardware and Deploying the Model	3-17
Using External Mode over FTDI with Pixhawk 1	3-18
Connecting to NSH Terminal for Debugging	3-20
Accessing NSH from MATLAB	3-20
Accessing NSH using QGroundControl (QGC)	3-21
Deployment and Verification Using PX4 Host Target and jMAVSim	3-22
About jMAVSim	3-22
Preparing PX4 Host Target and jMAVSim Using Hardware Setup Screens	3-22
Launching PX4 Host Target and jMAVSim Simulator from the Simulink Model	3-23
Troubleshooting Deploy to Hardware Issues	3-25
Description	3-25
Action	3-25
Troubleshooting Flash Overflow Issues with Pixhawk 1	3-27
Description	3-27

Action	3-27
Monitor and Tune the Model Running on PX4 Autopilots	3-31
Prepare a Simulink Model for External Mode	3-32
Running the Simulink Model for Monitor and Tune	3-33
Signal Monitoring and Parameter Tuning of Simulink Model	3-34
Stop Monitor and Tune	3-35
Performing Disconnect and Connect	3-36
Performing Connect Operation to Run an Unchanged Simulink Model on Hardware	3-36

MAT-File Logging on SD Card

4

Log Signals on an SD Card	4-2
Prerequisites for Logging Signals	4-4
Configure Model to Log Signals on SD Card	4-5
Settings for To Workspace Block	4-6
Prepare Model for Simulation and Deployment	4-8
Use px4PrepareModelForMATFileLogging to Optimize Memory	4-8
Enable MAVLink	4-8
Run Model on Target Hardware	4-10
Import MAT-Files into MATLAB	4-11
Use getMATFilesFromPixhawk to Retrieve MAT-files	4-11
Combine MAT-files Using px4MATFilestitcher and Analyze Variables	4-12
Troubleshooting Memory Limitations for MAT-file Logging	4-13
Action	4-13
Troubleshooting Dataset Format Usage for MAT-file Logging	4-16
Action	4-16

PX4 SITL Plant Model

5

Integrate Simulator Plant Model Containing MAVLink Blocks with Flight Controller Running on PX4 Host Target	5-2
Introduction	5-2
Controller Model and Plant Model	5-2
Prepare Controller Model and Simulator Plant Model	5-3
Run the Controller and Simulator Plant Model	5-5

Setup and Configuration for UAV Toolbox Support Package for PX4 Autopilots

- “Install Support for UAV Toolbox Support Package for PX4 Autopilots” on page 1-2
- “Integration with General PX4 Architecture” on page 1-5
- “Custom Startup Script in UAV Toolbox Support Package for PX4 Autopilots” on page 1-10
- “Impact of Disabling MAVLink, Commander, and Navigator Modules” on page 1-12
- “Setting Up Cygwin Toolchain and Downloading PX4 Source Code” on page 1-15
- “Setting up PX4 Tool Chain on Ubuntu 18.04” on page 1-20
- “Downloading PX4 Source Code as a Standalone in Windows” on page 1-21
- “Downloading PX4 Source Code in Ubuntu 18.04” on page 1-22
- “Selecting PX4 Autopilot Application” on page 1-23
- “Performing PX4 System Startup from SD Card” on page 1-24
- “Troubleshooting Test Connection Error” on page 1-26
- “Troubleshooting Firmware Build Failures” on page 1-28
- “Troubleshooting Connected I/O” on page 1-29

Install Support for UAV Toolbox Support Package for PX4 Autopilots

Add support for PX4 Autopilots by installing the UAV Toolbox Support Package for PX4 Autopilots.

After you install the support package, you can use:

- Supported hardware and its features.
- Block library to create models.
- Examples that show you how to use the Pixhawk® Series flight controller boards with PX4 flight stack.

Operating System Requirements

UAV Toolbox Support Package for PX4 Autopilots can be installed on both Windows and Linux operating systems. The supported versions are:

- Windows 10
 - Recommended – Windows 10 version 1803 (build 17134) or later

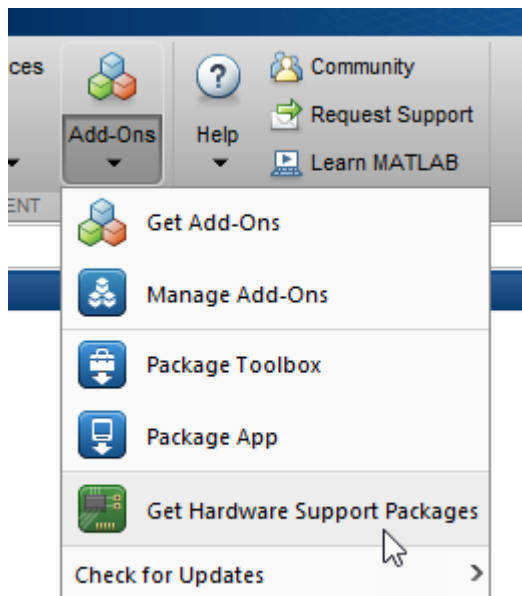
Tip To find the Windows version, enter `winver` at the Windows Start Menu.

- Linux:
 - Ubuntu 18.04 LTS

Install, Update, or Uninstall Support Package

Install Support Package

- 1 On the MATLAB® **Home** tab, in the **Environment** section, select **Add-Ons > Get Hardware Support Packages**.



- 2 In the Add-On Explorer window, click the support package and then click **Install**.

Update Support Package

On the MATLAB **Home** tab, in the **Environment** section, select **Help > Check for Updates**.

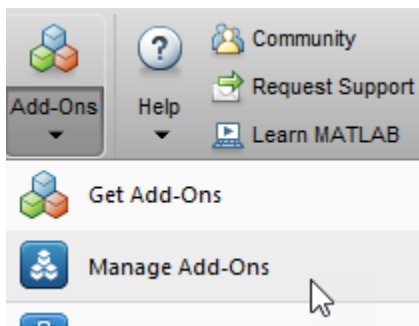
Uninstall Support Package


- 1 On the MATLAB **Home** tab, in the **Environment** section, click **Add-Ons > Manage Add-Ons**.
- 2 In the **Add-On Manager** window, find and click the support package, and then click **Uninstall**.

Hardware Setup

Hardware boards and devices supported by MathWorks® require additional configuration and setup steps to connect to MATLAB and Simulink®. Each support package provides a hardware setup process that guides you through registering, configuring, and connecting to your hardware board.

If the support package is already installed, you can start the hardware setup by opening the Add-On Manager.



In the Add-On Manager, start the hardware setup process by clicking the **Setup** button, .

After starting, the Hardware Setup window provides instructions for configuring the support package to work with your hardware.

Follow the instructions on each page of the Hardware Setup window. When the hardware setup process completes, you can open the examples to get familiar with the product and its features.

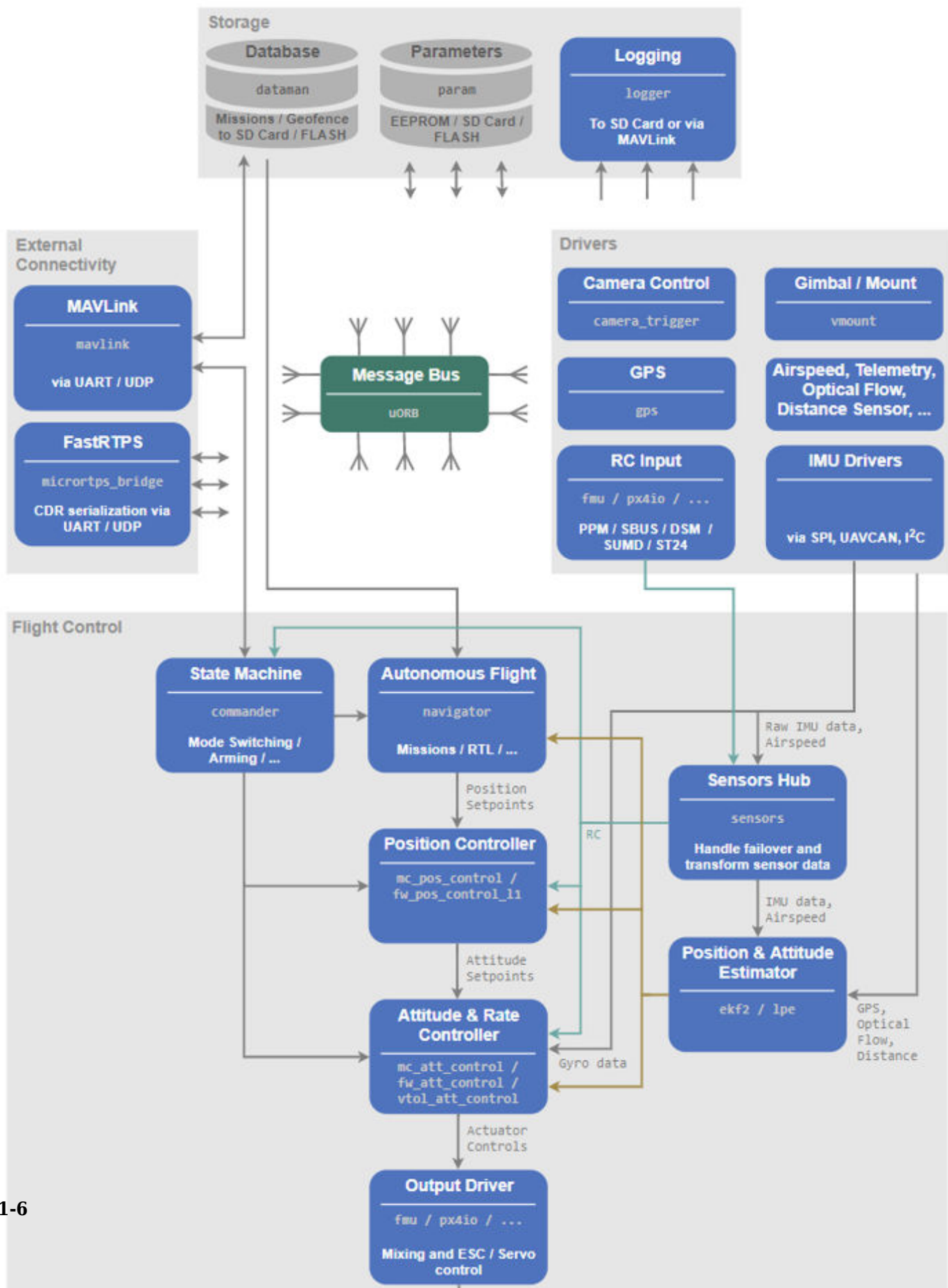
See Also

Integration with General PX4 Architecture

The UAV Toolbox Support Package for PX4 Autopilots enables you to design controllers and estimators in Simulink and deploy to PX4 Autopilot boards. You can integrate the generated code from the Simulink models, with the PX4 flight stack and then deploy the same to the PX4 Autopilots.

General PX4 Architecture

The high-level software architecture of PX4 includes modules for storage, external connectivity, drivers, uORB publish-subscribe message bus, and flight stack components.



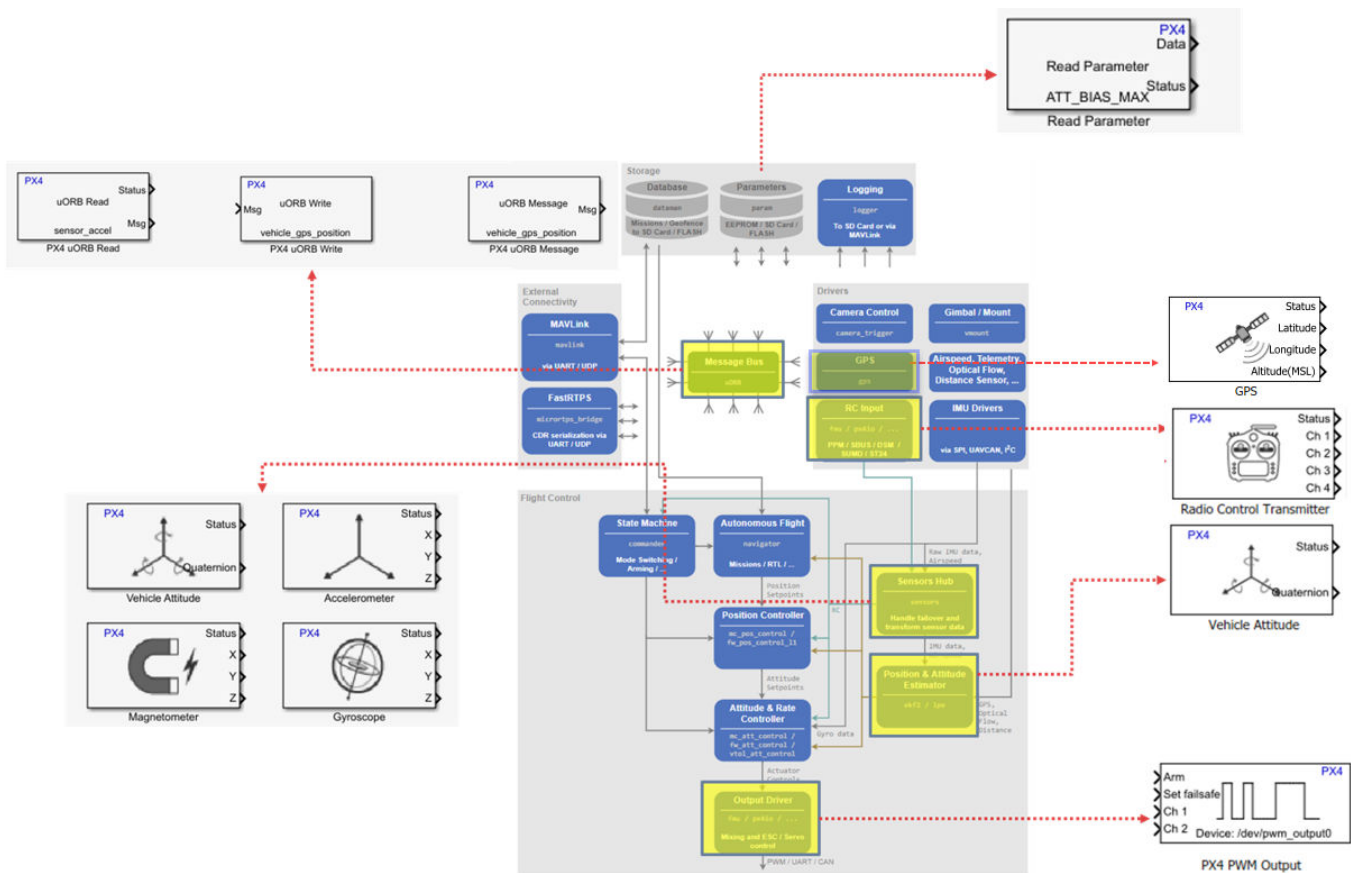
Some of the modules and interfaces of the general PX4 architecture can be integrated with UAV Toolbox Support Package for PX4 Autopilots.

Supported Simulink Blocks that Interface with the PX4 Modules

The UAV Toolbox Support Package for PX4 Autopilots enables you to design controllers and estimators in Simulink and deploy to PX4 Autopilot boards. You can integrate the generated code from the Simulink models, with the PX4 flight stack and then deploy the same to the PX4 Autopilots.

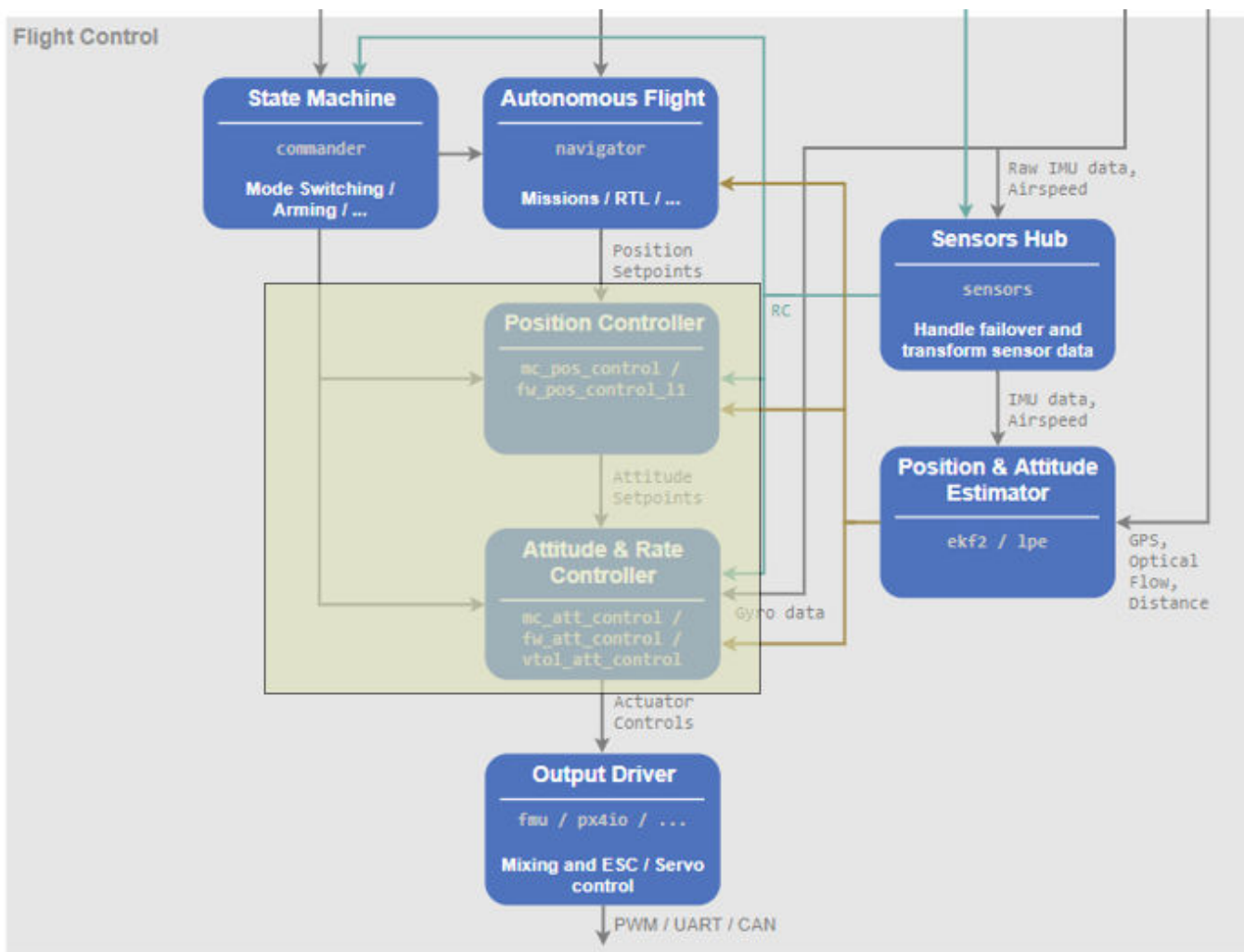
The support package provides interfaces for some of the components in the PX4 architecture by using Simulink blocks. You can use these blocks as input and output for the algorithms in Simulink model.

Component in PX4 Architecture	Simulink Block in the Support Package
Parameters	Read Parameter
uORB Message Bus	PX4 uORB Read PX4 uORB Write PX4 uORB Message
RC Input	Radio Control Transmitter
Sensors Hub	Vehicle Attitude Accelerometer Magnetometer Gyroscope
GPS	GPS
Position & Attitude Estimator	Vehicle Attitude
Output Driver	PX4 PWM Output
External serial communication	Serial Receive Serial Transmit



Supported Modules to be Replaced with User-Defined Algorithms

You can replace the Position Controller and Attitude & Rate Controller modules in the general PX4 architecture with user-defined algorithms that you develop using UAV Toolbox Support Package for PX4 Autopilots.



See Also

|

Custom Startup Script in UAV Toolbox Support Package for PX4 Autopilots

The Hardware Setup process of UAV Toolbox Support Package for PX4 Autopilots contains a step that enables the usage of a custom startup script. This startup script, which needs to be copied to the micro-SD card to be mounted on the Pixhawk Series flight controllers, helps in:

- Enabling or disabling the default PX4 modules as per the support package capabilities
- Avoiding interference with other default PX4 controller modules
- Preparing the PX4 Autopilot hardware to run the application generated by Simulink

For details about using this script, see “Performing PX4 System Startup from SD Card” on page 1-24.

Modules Enabled Using Custom Startup Script

The UAV Toolbox Support Package for PX4 Autopilots uses the custom startup script to enable some of the important modules.

The following table lists the modules that are enabled during the system startup from SD card.

PX4 Modules Enabled Using Custom Startup Script	Usage
uORB	Enables uORB message bus communication
px4io	Starts the firmware in IO processor of PX4 Autopilot and it is responsible for Main PWM Output
fmu	Generating AUX PWM outputs
ekf2	Estimator that outputs vehicle attitude
sensors	The sensors module reads the onboard sensors values and publishes data over uORB message bus
mtd and param	Loading and reading parameter

Modules Disabled Using Custom Startup Script

The UAV Toolbox Support Package for PX4 Autopilots uses the custom startup script to disable few modules.

The following table lists few modules that are disabled at startup.

PX4 Modules Disabled at Startup	Reason for Disabling
Position Controller Attitude & Rate Controller	The support package enables you to design the controller and hence the default Controller module in PX4 is disabled.

PX4 Modules Disabled at Startup	Reason for Disabling
Commander	The default Commander module is disabled to avoid possible interference with controller algorithm
Navigator	The default Navigator module is disabled to avoid possible interference with controller algorithm
MAVLink	The default serial port of MAVLink communication is the USB port of PX4 Autopilot. The MAVLink modules are disabled during startup to enable out-of-the-box External Mode support from Simulink, which uses the same USB port.

However, you can enable MAVLink while configuring the Simulink model. For details, see “Enabling MAVLink in PX4 Over USB” on page 3-5.

To understand the impact of disabling the modules, see .

Impact of Disabling MAVLink, Commander, and Navigator Modules

Impact of Disabling MAVLink

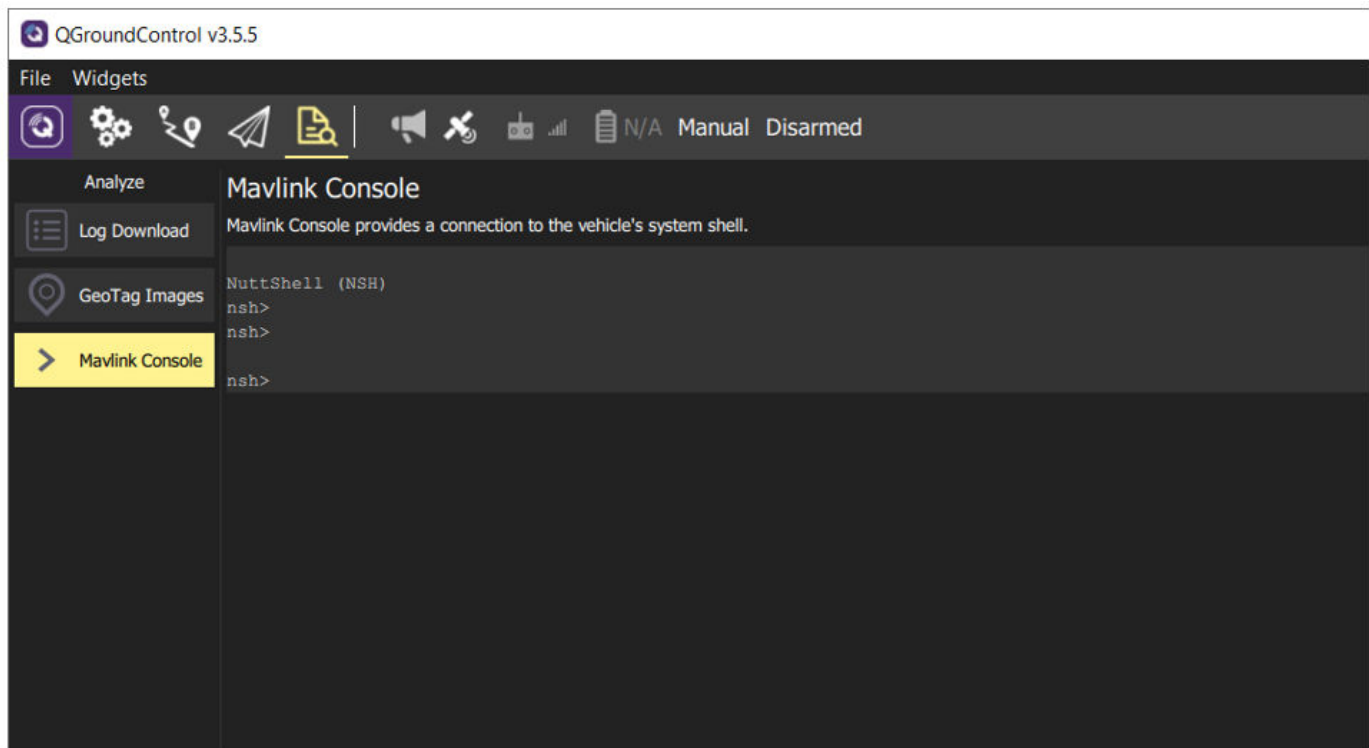
QGroundControl and Mission Planner Communication

QGroundControl (QGC) and Mission Planner communicate over MAVLink to the PX4 Autopilot. By default, MAVLink is disabled when the SD card with the custom startup script is loaded on the PX4 Autopilot. Therefore, the PX4 Autopilot cannot communicate with the QGC or Mission planner.

To enable communication with QGC and Mission planner, MAVLink can be enabled by modifying the setting in the Configuration Parameters dialog box, as described in “Enabling MAVLink in PX4 Over USB” on page 3-5.

MAVLink Console in QGroundControl

If MAVLink is enabled (as described in “Enabling MAVLink in PX4 Over USB” on page 3-5), you can use the **MAVLink Console** in QGroundControl to connect to the PX4 NSH and send commands.



Impact of Disabling Commander and Navigator Modules

Mission Upload from QGroundControl and Mission Planner

Because the Commander and Navigator modules are not started during boot-up, you cannot upload missions planned in QGC or Mission Planner to the PX4 Autopilot (even after you connect QGC to PX4 Autopilot over MAVLink).

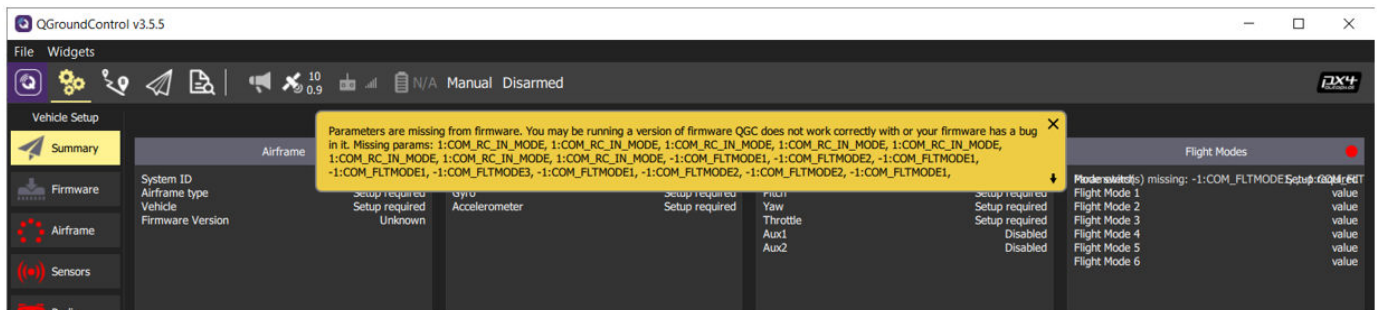
Vehicle Setup in QGroundControl

in QGroundControl, you can perform the Firmware Upgrade option under the **Vehicle Setup**, even though the Commander and Navigator modules are disabled.

However, the following sections under **Vehicle setup** in QGC do not work when Commander and Navigator modules are disabled:

- Airframe Selection
- Sensor Calibration
- Flight Modes
- Power setup

Parameters can be read from PX4 Autopilot, but parameter update from QGC is not available. Additionally, a warning regarding the missing parameters appears in QGC because many of the corresponding modules are not enabled.



Note You can calibrate the Radio even though Commander and Navigator modules are disabled.

Download of Log File

The logger module is not enabled by default in the custom startup script in the SD card. Therefore, the log file is not generated and downloaded in QGC.

Functionalities that are Unavailable upon Disabling Commander Module

Because the Commander module is not started using the custom startup script available in the SD card, the following functionalities that are usually performed by the Commander module are not be available after you deploy the Simulink model developed using UAV Toolbox Support Package for PX4 Autopilots:

- Sensor calibration
- Pre-flight check
- Automatic Arm and Disarm
- Navigator functionalities (Navigator module must be running for the below commands):
 - 1 Take-off, Land, Loiter mode, and so on.
 - 2 Set Mission mode (for example, Manual Mission and Attitude Control)

- Publishing log on state machine status (for example, logging of arming/disarming success over MAVLink).

However, you can implement sensor calibration and pre-flight checks by reading from IMU blocks over uORB in Simulink to replicate the functionality that the Commander is expected to perform. These algorithms can be implemented in a multi-thread model in Simulink just like the Commander module that runs a separate background thread for many of the tasks.

You can also model the arming/disarming feature of the PX4 Autopilot in Simulink by writing to the uORB topic, `actuator_armed`.

See Also

Setting Up Cygwin Toolchain and Downloading PX4 Source Code

Note This section explains the task to be completed as part of the step—Setup Cygwin Toolchain and Download PX4 Source Code—of the Hardware Setup process (using the Hardware Setup screens). Do not perform this as a standalone task.


Note Ensure that your PC is connected to an active internet connection before proceeding with this step.

To set up Cygwin toolchain and download the PX4 source code that is used in UAV Toolbox Support Package for PX4 Autopilots, follow these steps:

- 1 Download version **0.8** of PX4 Cygwin Toolchain MSI Installer, which is compatible with PX4 Firmware v1.10.2, available at this link.

Note UAV Toolbox Support Package for PX4 Autopilots supports only version **v0.8** of PX4 Windows Cygwin Toolchain MSI Installer, even though a latest version may be available.

v0.8


 released this on Nov 29, 2019


- Add python 2 and 3 package "pyros-genmsg" [8f8fbb4](#) (for [PX4/Firmware#13572](#))
- Latest Cygwin package versions as of the build time


This release is designed to build latest PX4 master after [PX4/Firmware#13572](#).

Assets

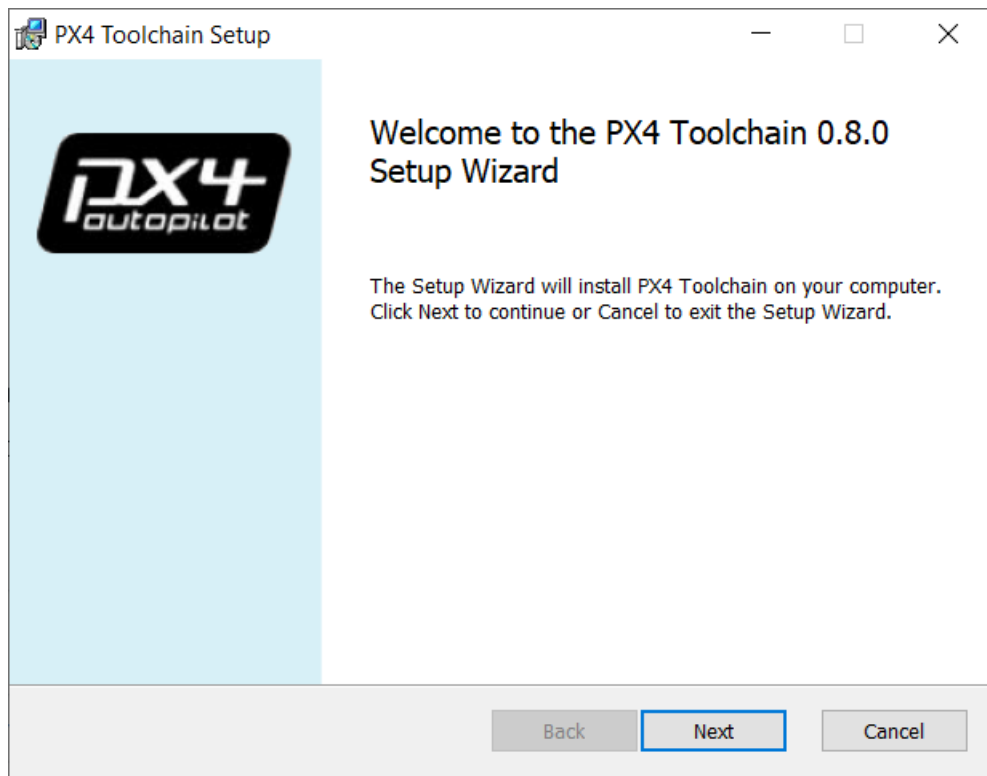
3


[PX4.Windows.Cygwin.Toolchain.0.8.msi](#)

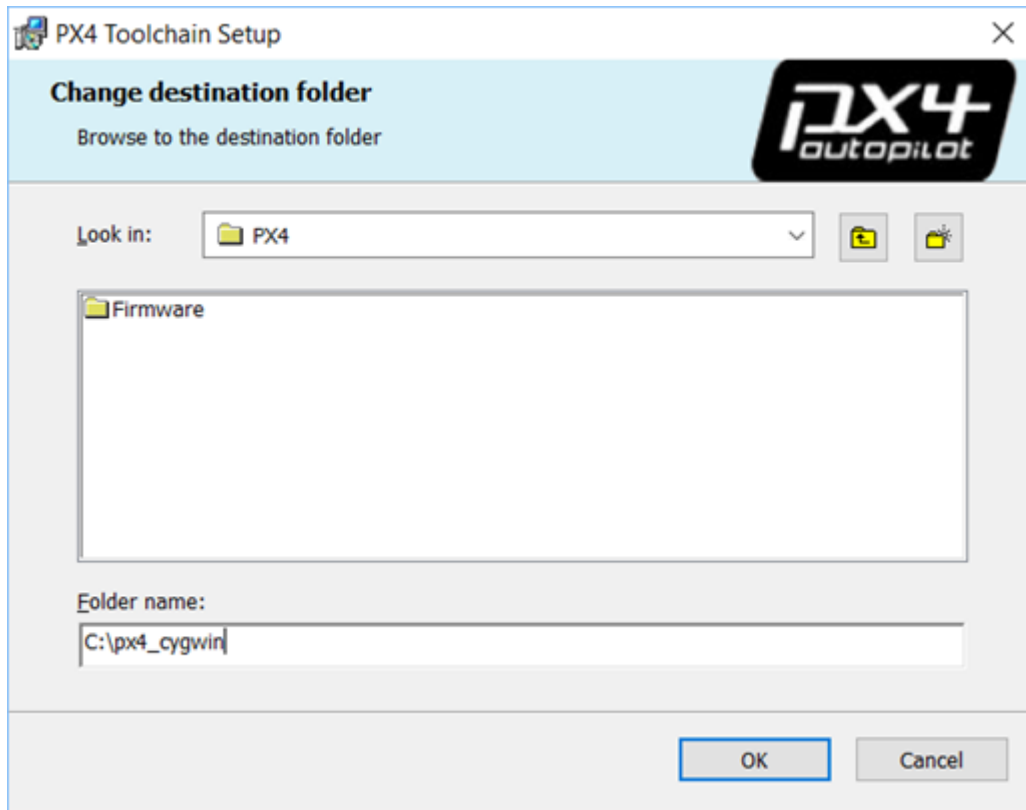

[Source code \(zip\)](#)


[Source code \(tar.gz\)](#)

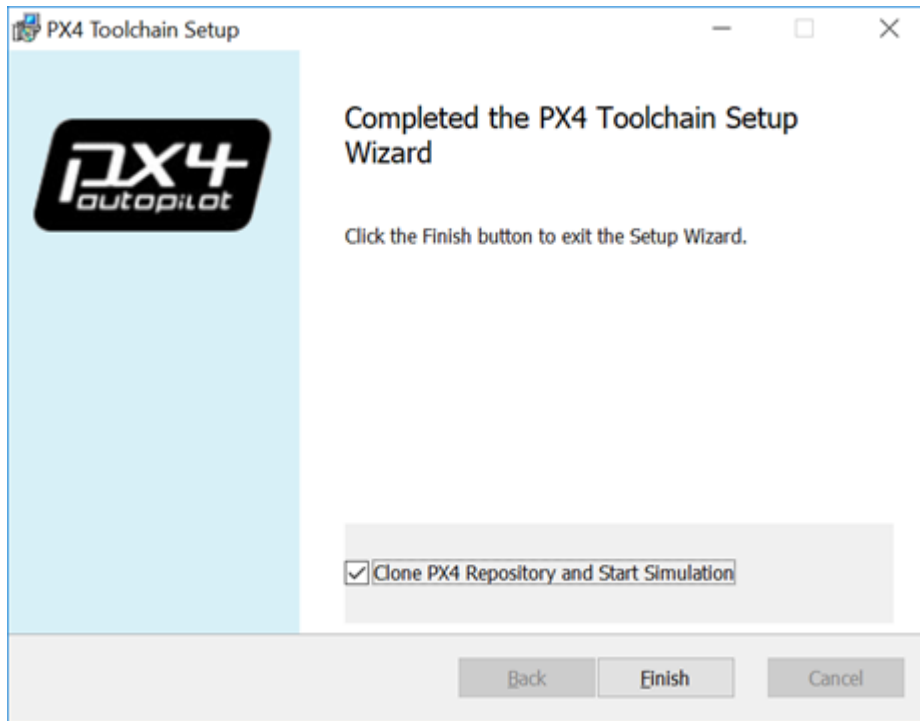
- 2 Run the MSI installer and start the installation of the toolchain.



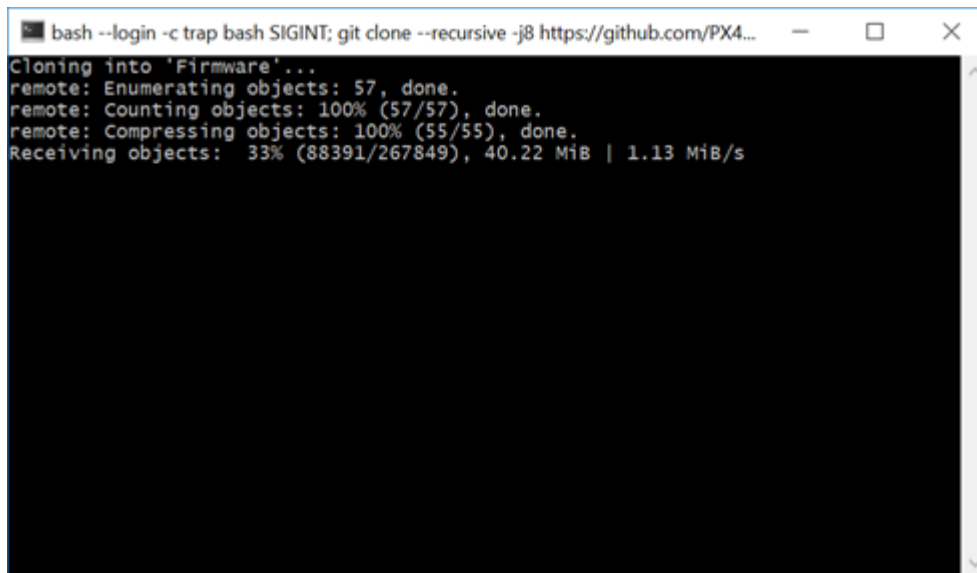
- 3** Change the installation folder for Cygwin to any local folder (for example, C:\px4_cygwin), and then click **OK**.



- 4 At the last step of the PX4 Toolchain Setup wizard, do the following:
- If you do not have the PX4 Source Code (PX4 Autopilot Firmware v1.10.2) downloaded in the host computer, select the option **Clone PX4 repository and Start Simulation**, and then click **Finish**. This option clones the latest master PX4 Firmware. When you click Verify Installation in Step 6 below, the firmware is checked out automatically to v1.10.2.
 - If the PX4 Source Code (PX4 Autopilot Firmware v1.10.2) is already available in the host computer, click **Finish** without selecting the **Clone PX4 repository and Start Simulation** option.



- 5 If you selected **Clone PX4 repository and Start Simulation** and clicked **Finish**, a bash shell is launched which starts cloning the firmware.



Wait for the firmware to finish cloning. After the firmware is cloned, Simulation starts in jMAVSim. You can close the bash shell at this stage.

The PX4 firmware is cloned inside a folder named home, inside the Cygwin folder that you selected during installation (for example, C:\px4_cygwin\home\).

- 6 In the Hardware Setup screen - Setup Cygwin Toolchain and Download PX4 Source Code - enter the path that you used for the Cygwin toolchain installation (same as Step 3), and then click **Verify Installation**.

After the installation is successfully verified, click **Next** to proceed to the next step of the Hardware Setup process.

See Also

Setting up PX4 Tool Chain on Ubuntu 18.04

Note This section explains the task to be completed as part of the step—Set Up the PX4 Toolchain—of the Hardware Setup process (using the Hardware Setup screens). Do not perform this as a standalone task.

Note This section explains the PX4 Toolchain process for Ubuntu 18.04 for PX4 Firmware v1.10.2. This is not applicable for older (for example, v1.8.0) or any other PX4 Firmware version.

UAV Toolbox Support Package for PX4 Autopilots requires installation of a development environment. This development environment is used to build firmware for all the Pixhawk® Series flight controller boards.

Download PX4 Firmware v1.10.2

You need to download PX4 Firmware v1.10.2 as mentioned in “Downloading PX4 Source Code in Ubuntu 18.04” on page 1-22.

Set up PX4 Toolchain on Ubuntu 18.04

After you have downloaded PX4 Firmware v1.10.2, follow the below commands to install the PX4 Toolchain on Ubuntu 18.04.

.

Note Ensure that your host computer is connected to an active internet connection before proceeding.

To install and setup the PX4 toolchain:

- 1 Launch the bash terminal in the Ubuntu 18.04 host computer.
- 2 Go to the PX4 Firmware v1.10.2 directory that you downloaded.

For example: `cd /home/username/mypx4/`

- 3 Navigate to the folder containing the Toolchain setup script.

`cd Firmware/Tools/setup`

- 4 Run the Toolchain setup script.

source ubuntu.sh

- 5 Enter the sudo credentials when prompted, to start the PX4 toolchain setup process.

The `ubuntu.sh` build script installs different third-party utilities like GCC 7.2.1, CMake 3.x, Ninja 1.6, Git, and certain Python packages.

- 6 Reboot the host computer after the `ubuntu.sh` script runs successfully.

Downloading PX4 Source Code as a Standalone in Windows

Note This section explains how to download PX4 Firmware v1.10.2 as a standalone task. This will only be required if you have not cloned PX4 Firmware as part of Cygwin Toolchain installation process as described in the link “Setting Up Cygwin Toolchain and Downloading PX4 Source Code” on page 1-15.

UAV Toolbox Support Package for PX4 Autopilots requires installation of a development environment. This development environment is used to build firmware for all the Pixhawk® Series flight controller boards.

Download PX4 Source Code from Github

To download the PX4 source code (PX4 Autopilot Firmware v1.10.2) on a Windows host computer:

- 1 Ensure that you have installed Cygwin Toolchain in Windows computer as discussed in the link “Setting Up Cygwin Toolchain and Downloading PX4 Source Code” on page 1-15.
- 2 Navigate to the installed Cygwin Toolchain directory in your PC.
- 3 Launch the batch file run-console.bat. This opens the Cygwin console.
- 4 From the Cygwin console, navigate to the location in your PC where you would like to clone the PX4 Firmware.
- 5 Create a directory named mypx4 in the system.

For example:

- `mkdir mypx4`
`cd mypx4`

Note Ensure that your PC is connected to an active internet connection before proceeding with the next step.

- 6 Go to the newly created px4 directory, and run the following commands one-by-one (you may need to wait for a command execution to complete before entering the next command):
 - a `git clone https://github.com/PX4/Firmware.git Firmware`
 - b `cd Firmware`
 - c `git checkout v1.10.2`
 - d `git submodule update --init --recursive`

The download may take several minutes to complete (depending on the internet speed).

Downloading PX4 Source Code in Ubuntu 18.04

Note This section explains the task to be completed as part of the step—Download PX4 Source Code—of the Hardware Setup process (using the Hardware Setup screens). Do not perform this as a standalone task.

UAV Toolbox Support Package for PX4 Autopilots requires PX4 Autopilot Firmware v1.10.2

Download PX4 Source Code from Github

To download the PX4 Sourcecode (PX4 Autopilot Firmware v1.10.2) on the Ubuntu 18.04 host computer:

- 1** Open the bash terminal on Ubuntu 18.04
- 2** Create a directory named mypx4 in the home folder.

For example:

- create a directory in home directory, as described below:

```
cd ~  
  
mkdir mypx4  
  
cd mypx4
```

Note Ensure that your computer is connected to an active internet connection before proceeding with the next step.

- 3** Go to the newly created px4 directory, and run the following commands one-by-one (you may need to wait for a command execution to complete before entering the next command):

```
a  git clone https://github.com/PX4/Firmware.git Firmware  
b  cd Firmware  
c  git checkout v1.10.2  
d  git submodule update --init --recursive
```

The download may take several minutes to complete (depending on the internet speed).

After you complete this step, go back to the Hardware Setup screen again and continue with the next step.

Selecting PX4 Autopilot Application

Note This section explains the step—**Select Application in Simulink**—of the Hardware Setup process (using the Hardware Setup screens).

UAV Toolbox Support Package for PX4 Autopilots provides the option to create either path follower or flight controller algorithm in Simulink. The configuration settings required for each of these are different, and the Hardware Setup process can be used to pre-configure the PX4 firmware and the startup sequence for the required algorithm.

In the **Select Application in Simulink** screen of the Hardware Setup process, select the required option for the PX4 Autopilot algorithm that you are going to design using the support package:

- **Design Flight Controller Algorithm in Simulink** — Select this option to design a flight controller algorithm by disabling the default PX4 controller module that is present in the PX4 firmware that you downloaded. This selection results in an additional step (**Customize PX4 System Startup**) as you proceed further with the Hardware Setup screens. A custom startup script, which is loaded from an SD Card connected to the flight controller, will be used for starting the modules.
- **Design Path Follower Algorithm in Simulink** — Select this option to design a path follower algorithm by disabling the default PX4 navigator module that is present in the PX4 firmware that you downloaded. This selection results in an additional step (**Select Airframe for QGround Control**) as you proceed further with the Hardware Setup screens. The default PX4 startup script, `rCS`, will be used for starting the modules.

See Also

Performing PX4 System Startup from SD Card

Note This section explains the task to be completed as part of the step—Customize PX4 System Startup—of the Hardware Setup process (using the Hardware Setup screens). Do not perform this as a standalone task.

When you deploy custom flight controller algorithms developed using UAV Toolbox Support Package for PX4 Autopilots, you need to suppress the execution of some of the default startup processes and run the application generated by Simulink. This is done by using a start-up script, which is copied to the micro-SD card to be mounted on the Pixhawk® Series flight controllers.

Note This step is required to be performed only once for the micro-SD card that is mounted on the Pixhawk® Series flight controllers.

Note This step is required to be performed only if you selected the option **Design Flight Controller Algorithm in Simulink** in the Select Application in Simulink screen of the Hardware Setup process.

Note Before customizing the PX4 system startup, ensure that the Pixhawk Series flight controller is flashed with PX4 firmware. If you were using ArduPilot firmware before, or if you are not sure about the current firmware type and version of the flight controller (for example, if it is a new Pixhawk flight controller board), flash the latest stable release of the PX4 firmware, using QGroundControl. Refer to QGroundControl documentation for more details on uploading the PX4 firmware. This action needs to be done only once, for a specific board.

To enable Simulink-specific startup sequence from micro-SD card:

- 1 After you install UAV Toolbox Support Package for PX4 Autopilots, go to the `lib\etc` folder under the Support Package Root folder.

Tip You can run the following command at the MATLAB command prompt to go to the specified folder:

```
cd (fullfile(codertarget.pixhawk.internal.getSpPkgRootDir, 'lib', 'etc'))
```

- 2 Copy the file `rc.txt`.
- 3 Connect a micro-SD card to the host computer using the micro-SD card slot on the host computer or by using an external card reader.
- 4 Create a root-level folder named `etc` in the micro-SD card, and paste the `rc.txt` file to this new folder.
- 5 Remove the micro-SD card from the host computer, and insert the same in the Pixhawk Series flight controller that you will be using to deploy the Simulink code.

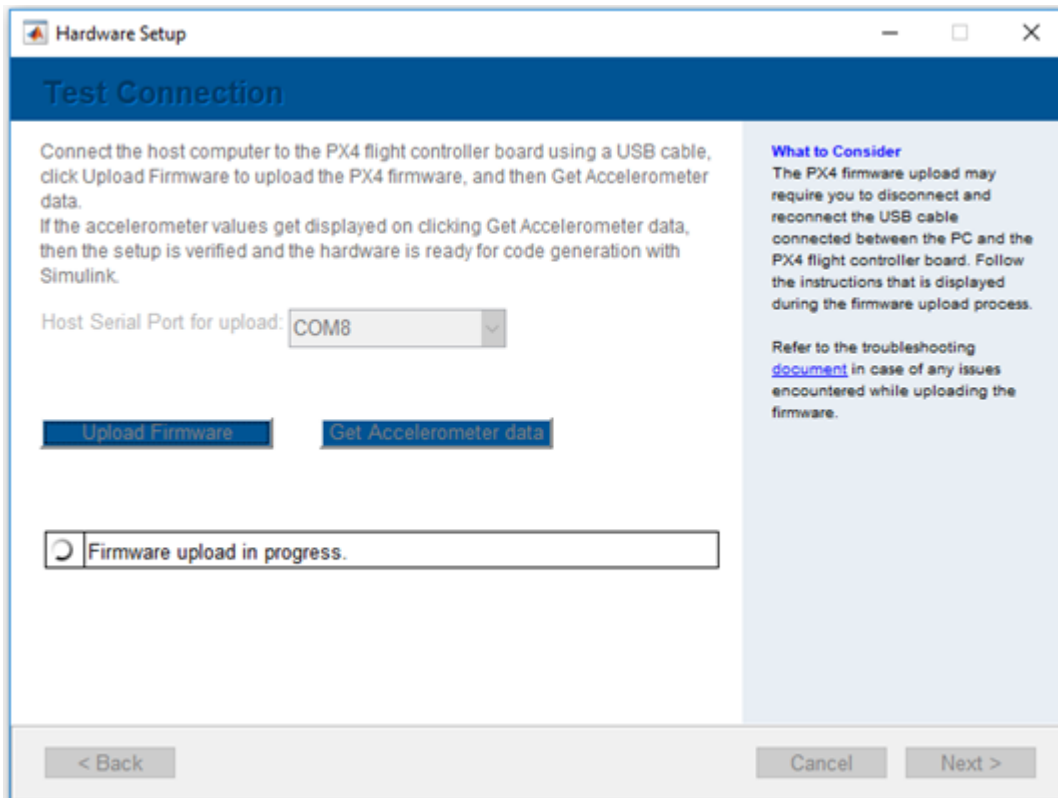
Whenever you start the Pixhawk Series flight controller with a Simulink model deployed in it, the `rc.txt` file on the micro-SD card starts a customized application called `px4_simulink_app`, which executes the code generated through the deployed Simulink model. The `px4_simulink_app` is

located under `Firmware\src\modules` (the `Firmware` directory was created as part of PX4 Sourcecode download step).

After you complete this step, go back to the Hardware Setup screen again and continue with the next step.

Troubleshooting Test Connection Error

- If there are no ports shown in the Host Serial Port for upload drop-down list, ensure that you have connected the Pixhawk® Series flight controller properly to the host computer. Even after connecting the flight controller properly, if the serial port does not appear and it is not detected in the Windows Device Manager, ensure that the Pixhawk Series flight controller is flashed with PX4 firmware. If you were using ArduPilot firmware before, or if you are not sure about the current firmware type and version of the flight controller (for example, if it is a new Pixhawk flight controller board), flash the latest stable release of the PX4 firmware, using QGroundControl. Refer to QGroundControl documentation for more details on uploading the PX4 firmware.
- When you perform the Test Connection step of the Hardware Setup process to upload the PX4 firmware to a Pixhawk® series flight controller, the Hardware Setup screen may not respond even after a long time (the busy spinner remains displayed on the screen).



The reasons for this issue and the corresponding troubleshooting actions are described in the below table:

Reason for the Issue	Action Required
<p>The serial port that you selected in the Test Connection step is not the actual port to which the flight controller is connected. You can verify the issue by checking the serial port to which the Pixhawk Series flight controller is connected.</p>	<p>Close the Hardware Setup screen, and restart the Hardware Setup process again (including the Build Firmware step). Ensure that you select the correct serial port in the Test Connection screen.</p>
<p>On the dialog box that appears instructing you to reconnect the flight controller, you did not click OK within 5 seconds after reconnecting the flight controller (as instructed in the dialog box).</p> <p>You can verify if this is the reason for the issue, if the following line appears at the MATLAB command prompt:</p>	<p>Disconnect and reconnect the Pixhawk Series flight controller to the host computer.</p> <p>Whenever the dialog box appears instructing you to reconnect the flight controller to the serial port, ensure that you click OK on the dialog box within 5 seconds after reconnecting the flight controller.</p>
<p>If the board does not respond, unplug and replug the USB connector.</p>	<p>Tip: The 5-second limit applies only to the time after reconnection. Therefore, you can first disconnect the USB port from the host computer without worrying about the time limit; but, ensure that you click OK within 5 seconds after reconnecting the cable to the USB port of the host computer.</p>

Troubleshooting Firmware Build Failures

If the PX4 Firmware build fails while performing Hardware setup screens, you can analyze the log file `MW_px4_log.txt` to identify the errors and then troubleshoot the issue. This log file is available at the temporary folder for the system. You can run the following command at the MATLAB command prompt to go to the folder and find the file `MW_px4_log.txt` log file.

```
cd (tempdir)
```

Troubleshooting Connected I/O

Description

Simulink timer is very slow while running Connected I/O simulation.

Action

This is due to presence of blocks with very less sample times. There is a two-way communication between Simulink and the Autopilot while running Connected I/O simulation. This introduces an inherent time overhead for a particular block execution. To resolve the issue, set the sample time of blocks to more than the block execution time for connected I/O simulation. For example, if there are two blocks and block execution time is 20 milliseconds for a block. Then set the minimum sample time of the model as $20 \times 2 = 40$ milliseconds. For PX4 Host target this minimum block Execution time is around 10-20 ms and for PX4 hardware boards it is around 20-30 ms.

Troubleshooting Connected I/O with PX4 Host Target

Description

If the system you are using is running slow, following error might appear while starting Connected I/O simulation. This is because PX4 host target and jMAVSim takes more time to launch.

```
Caused by:
  • Cannot create a communication link with the remote server. Please check the input
    arguments(ADDRESS and PORT) and make sure the server is running.
  Additional Information: No connection could be made because the target machine actively refused
  it
```

Action

To resolve the issue:

- Wait till PX4 host target and jMAVSim is launched properly.
- Start connected I/O simulation again by clicking **Run** on **Modeling** tab.

Description

With PX4 v1.10 update, some uORB messages stops updating after an hour. This issue might occur with Connected I/O simulation also.

Action

To resolve the issue:

- Stop simulation and close PX4 Host Target window.
- Start connected I/O simulation again by clicking **Run** on **Modeling** tab.

Connected I/O

Communicate with Hardware in Normal Mode Simulation Using Connected I/O

You can use Connected I/O to communicate with the IO peripherals on the hardware during Normal mode simulation.

Normal mode simulation with Connected I/O is an intermediate step in the Model-Based Design workflow that bridges the gap between simulation and code generation by enabling Simulink to communicate with the hardware before deploying the model on hardware. Connected I/O enables you to modify your model design and monitor the effect of the modified design using the peripheral data from the hardware in a near real-time environment. You are not required to deploy the model on the hardware to monitor the effect of the modified design, which accelerates the simulation process. This Simulink (software) - PX4 (hardware) interaction is available in Normal mode simulation only when you enable Connected I/O.

Note Connected I/O is not based on Simulink code generation, and hence Embedded Coder license is not required to use Connected I/O simulation.

These sections explain:

In this section...
“Supported PX4 Boards and Blocks with Connected I/O” on page 2-2
“How Connected I/O Works” on page 2-3
“Connected I/O in Model-Based Design” on page 2-4
“How Connected I/O Differs from External Mode” on page 2-5
“When to Use Connected I/O” on page 2-6
“How to Enable Connected I/O” on page 2-6

Supported PX4 Boards and Blocks with Connected I/O

The Connected I/O described here applies to the UAV Toolbox Support Package for PX4 Autopilots on these PX4 boards and blocks:

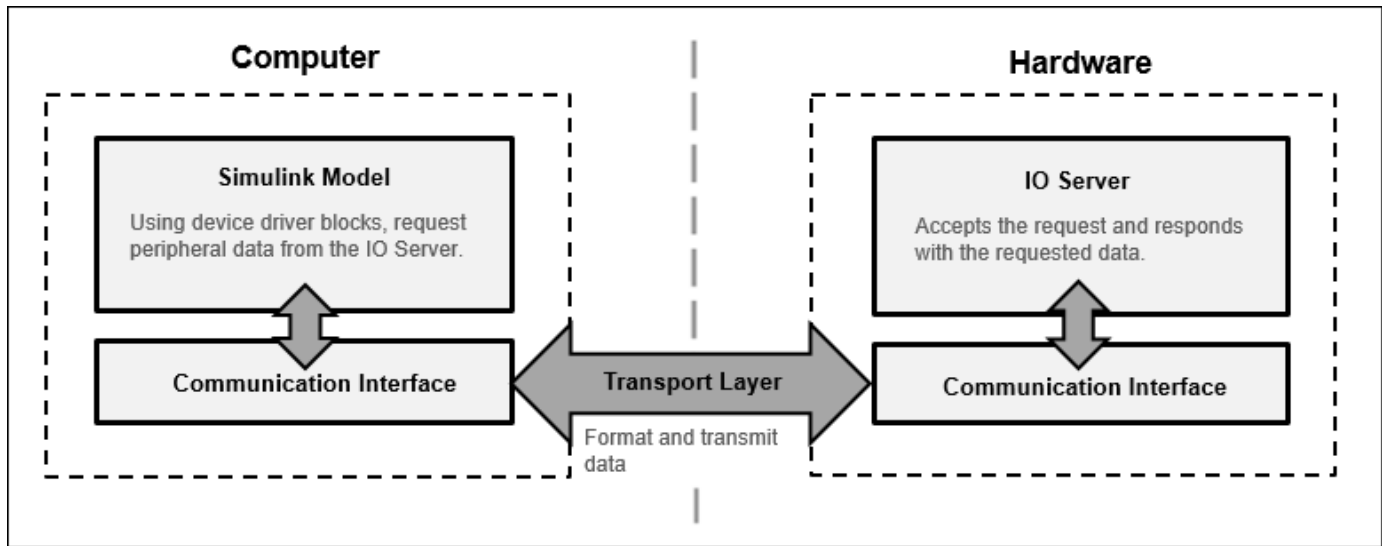
- Source blocks: Without Connected I/O, these source blocks output zero during Normal mode simulation. With Connected I/O, these blocks read data from the peripherals of the hardware during Normal mode simulation.
- Sink blocks: Without Connected I/O, these sink blocks do not have any role during Normal mode simulation. With Connected I/O, these blocks write data to the peripherals of the hardware during Normal mode simulation.

PX4 Boards	Source Blocks	Sink Blocks
PX4 Host Target	PX4 uORB Read	PX4 uORB Write
PX4 Pixhawk 1	Accelerometer	I2C Master Write
PX4 Pixhawk 2.1 (Cube)	Battery	Serial Transmit
PX4 Pixhawk 4	GPS	PWM Output
PX4 Pixracer	Gyroscope	
	Magnetometer	
	Radio Control Transmitter	
	Vehicle Attitude	
	PX4 Analog Input	
	I2C Master Read	
	Read Parameter	
	Serial Receive	

How Connected I/O Works

Connected I/O creates a communication interface that enables the Simulink model and the IO Server to communicate with each other. The Simulink model resides in your computer, and the IO Server is an engine on the hardware that contains all the peripheral functions. The transport layer formats and transmits the data using the communication interface.

This diagram shows the connection that the Connected I/O creates between your computer and the hardware.



Communication in Normal Mode Simulation with Connected I/O

When you simulate a Simulink model in Normal mode with Connected I/O:

- 1 The device driver block (for example, PX4 uORB Read block) in the model request sensor data from the IO Server.
- 2 The IO Server accepts the request and responds with the requested data. You can use any Simulink sink or dashboard block to view the received data. Using the peripheral data received, you can verify that your model design meets the requirements.
- 3 If necessary, you can modify the design by adding, removing, or replacing any block in the Simulink model.
- 4 After the model is modified, resimulate the model. During simulation, the data request from the model is communicated to the hardware. You can continue to modify and simulate the model until the expected behavior is achieved.

Note

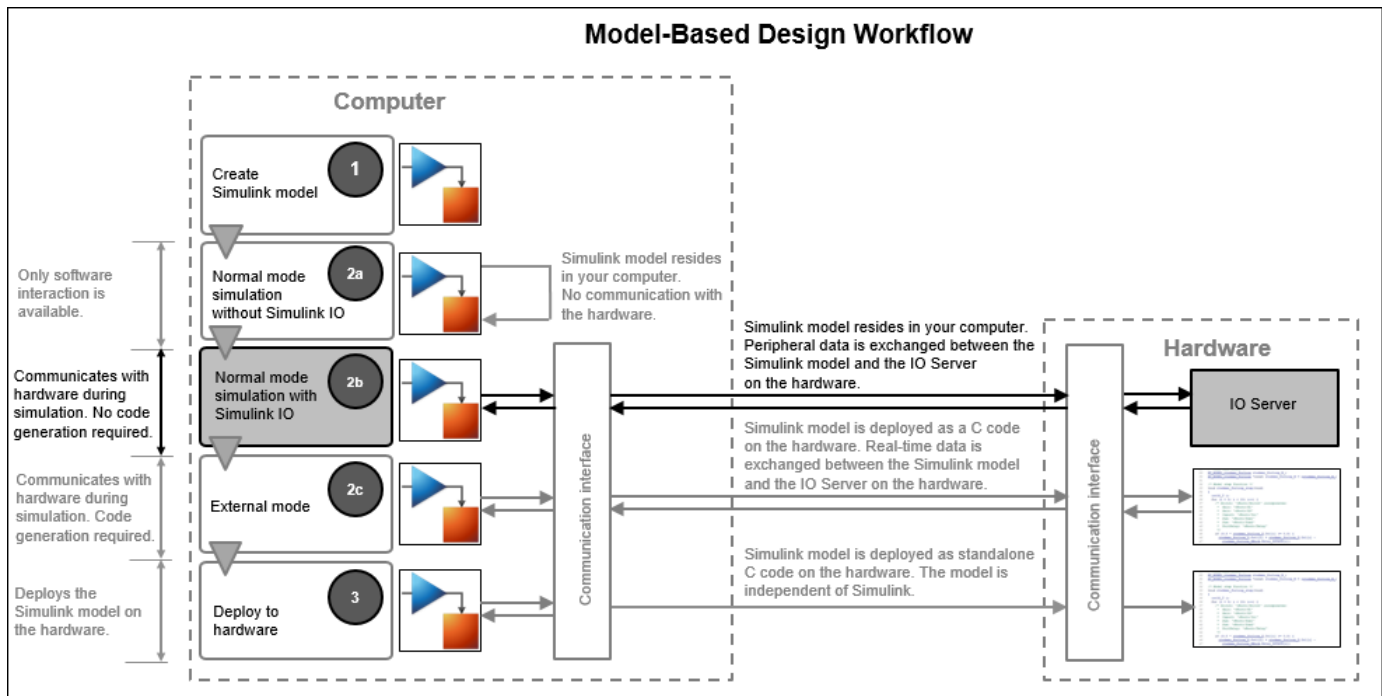
- The communication in Connected I/O is an on-demand process. The hardware sends data only when receiving a data request from the Simulink model.
- You do not have to build, deploy, and run the model on the hardware to monitor the effects of your changes in your model design.

Connected I/O in Model-Based Design

When you simulate a model in Normal mode without Connected I/O, Simulink does not communicate with the hardware. Simulink communicates with the hardware only when the code is generated and the model is deployed on the hardware in External mode. Normal mode simulation with Connected I/O is an intermediate step in the model-based design workflow that bridges the gap between simulation and code generation by enabling Simulink to communicate with the hardware before deploying the model on the hardware.

This Model-Based Design Workflow diagram displays a model-based workflow:

- 1 Create a Simulink model.
- 2 Simulate the model in:
 - a Normal mode simulation without Connected I/O: There is no hardware interaction and no code generation.
 - b Normal mode simulation with Connected I/O: The model communicates with the hardware. There is no code generation.
 - c External mode: The model is deployed on the hardware and generates code.
- 3 Deploy the model to the hardware.



Model-Based Design Workflow

How Connected I/O Differs from External Mode

Connected I/O and External mode both enable you to communicate with the hardware during simulation. However, you use Connected I/O and External mode for different purposes. The table shows the actions that you can perform with each mode.

Action	External Mode	Connected I/O
Obtain real-time data	You can obtain real-time data with External mode.	You can obtain only non real-time data with Connected I/O.
Time required to start simulation	1-2 minutes	Few seconds
Code generation	Code is generated on the hardware. Embedded coder license is required.	No code is generated. Embedded coder license is not required.

When to Use Connected I/O

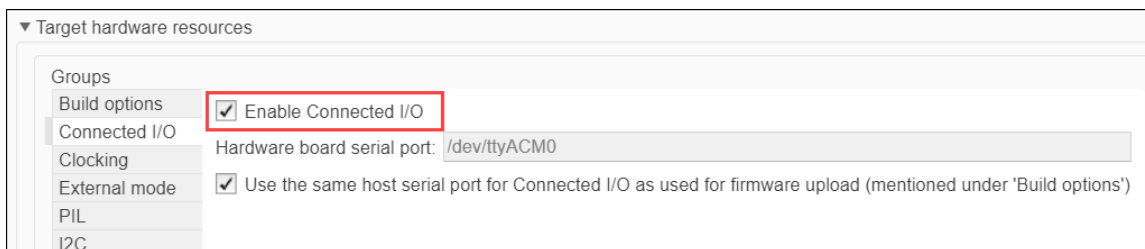
Connected I/O must be used primarily as a Sensor/Actuator block characterization which enables a fast way to get started with a block to understand its outputs/inputs. In Connected I/O every block needs to communicate to the PX4 boards and thus there is a specific minimum time required for a block execution. If the sample time specified is smaller than this block execution time, the block cannot run at the specified sample time during connected I/O simulation. This makes Connected I/O simulation not suitable for designing closed loop controllers and systems with dynamic feedback, where blocks must run at much higher rate.

Note Connected I/O simulation is not ideal for application involving a continuous / dynamic feedback from the plant.

How to Enable Connected I/O

To simulate a model in Connected I/O during Normal mode simulation, you must enable the Connected I/O option:

- 1 Open a Simulink model.
- 2 In the Simulink toolbar, set the Simulation mode to **Normal**.
- 3 In the **MODELING** tab, select **Model Settings**.
- 4 In the Configuration Parameters dialog box, select **Hardware Implementation**.
- 5 Set the **Hardware board** parameter to a PX4 hardware. This selection automatically populates the parameters in the **Hardware board** settings with the default values for the PX4 hardware.
- 6 From the **Groups** list under **Target hardware resources**, select **Connected I/O**.
- 7 Select the **Enable Connected I/O** option.



- 8 Click **Apply**. Click **OK** to close the dialog box.
- 9 Optionally, you can change the rate of simulation by enabling the Simulink Pacing Option as described in “Simulation Pacing”.

Run on Target for UAV Toolbox Support Package for PX4 Autopilots

- “Supported PX4 Autopilots” on page 3-2
- “Enabling MAVLink in PX4 Over USB” on page 3-5
- “Plant and Attitude Controller Model for Hexacopter” on page 3-7
- “Migrating from Pixhawk Pilot Support Package to UAV Toolbox Support Package for PX4 Autopilots” on page 3-10
- “Integrating uORB Topics in the Simulink Model” on page 3-16
- “Setting Up the Hardware and Deploying the Model” on page 3-17
- “Using External Mode over FTDI with Pixhawk 1” on page 3-18
- “Connecting to NSH Terminal for Debugging” on page 3-20
- “Deployment and Verification Using PX4 Host Target and jMAVSim” on page 3-22
- “Troubleshooting Deploy to Hardware Issues” on page 3-25
- “Troubleshooting Flash Overflow Issues with Pixhawk 1” on page 3-27
- “Monitor and Tune the Model Running on PX4 Autopilots” on page 3-31

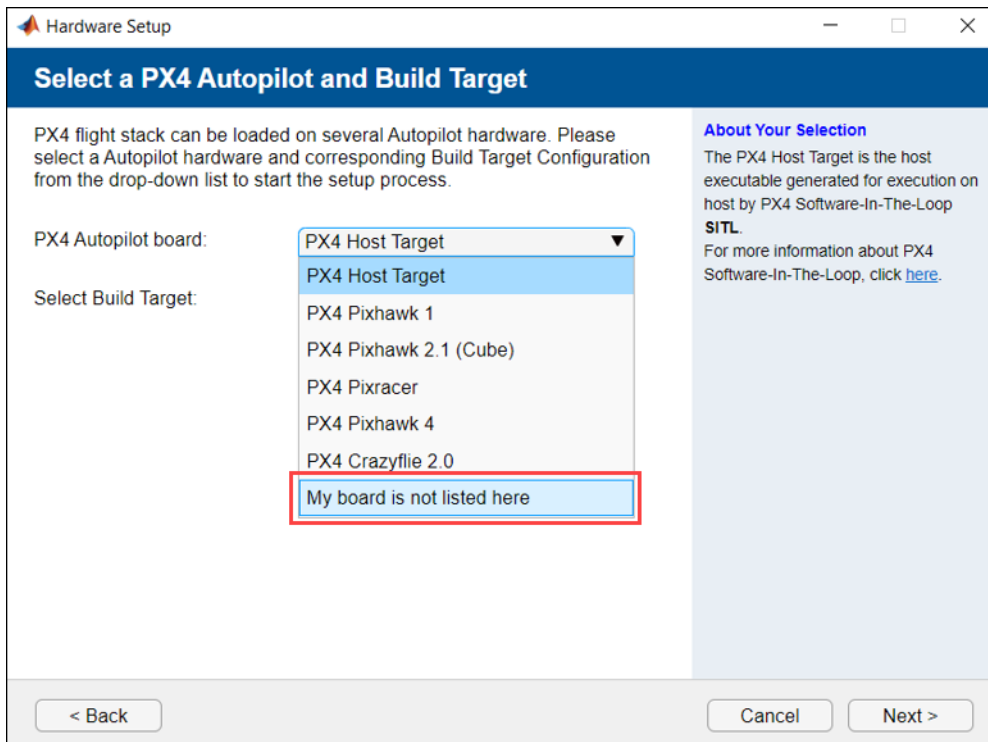
Supported PX4 Autopilots

The UAV Toolbox Support Package for PX4 Autopilots can be used to model flight control algorithms based on PX4 flight stack, and the same can be deployed on the following autopilots:

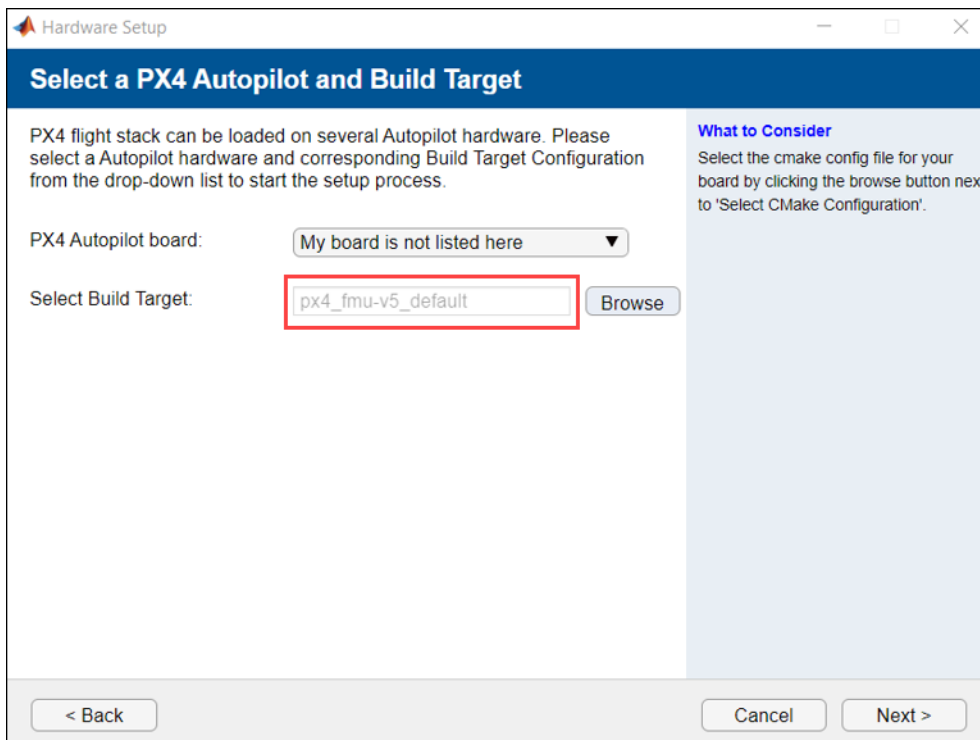
- Pixhawk® 1
- Cube (Pixhawk 2.1)
- Pixhawk 4
- Pixracer

The support package can also be used with other PX4 Autopilots listed here that have been designed based on the Pixhawk FMU project and use a FMU configuration to build the firmware. However, the complete functionality is not tested on those controllers.

To get started with an autopilot that does not belong to set of officially tested autopilots mentioned above, select the option `My board is not listed here` in the hardware setup.

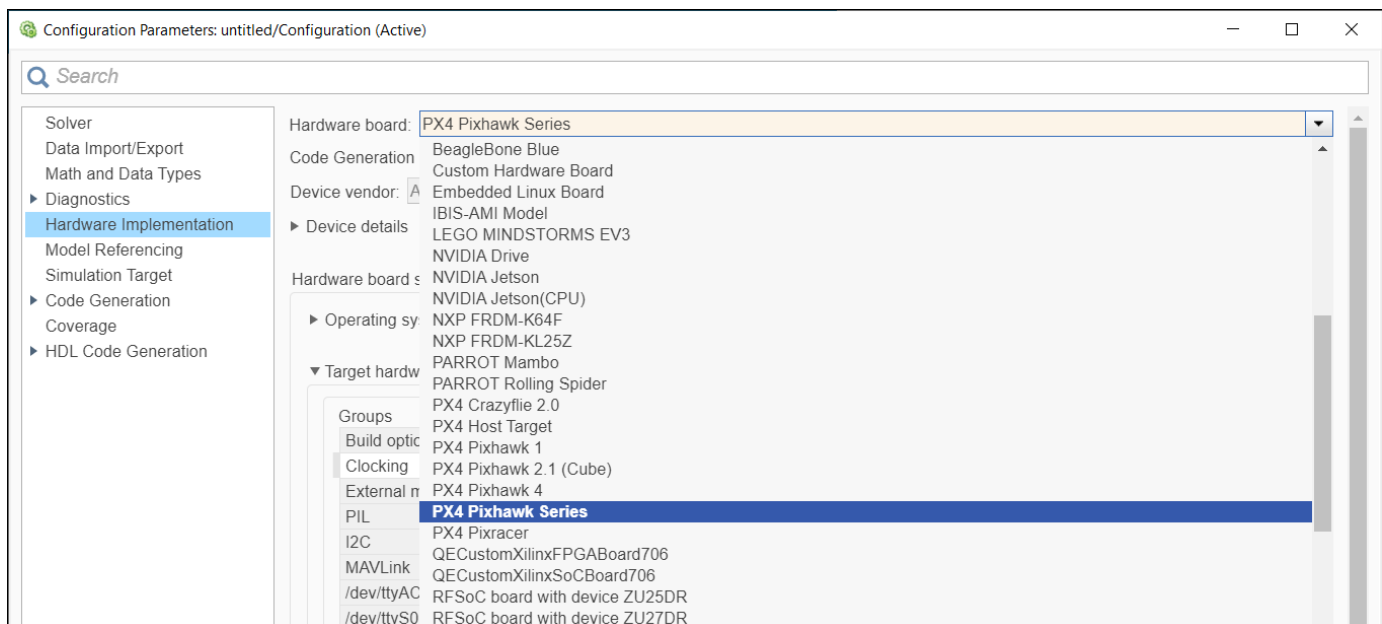


Browse to the corresponding CMake Build target in your PX4 Firmware for building.



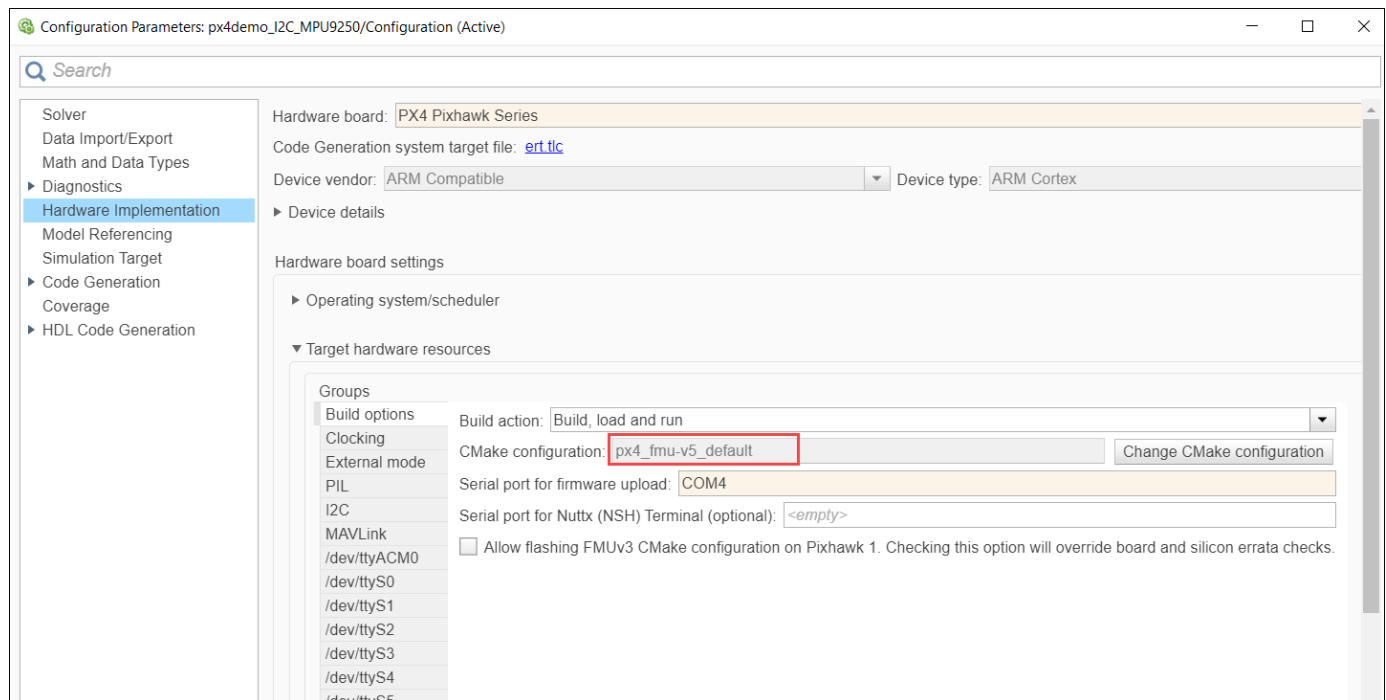
You might also need to enter the corresponding make command to build the firmware for the corresponding target.

After hardware setup is completed and the PX4 Firmware is successfully built for the selected build target, you can use your autopilot in Simulink by selecting PX4 Pixhawk Series as Hardware board in the Simulink model configuration settings.



The selected CMake build target in hardware setup automatically appears in CMake configuration.

3 Run on Target for UAV Toolbox Support Package for PX4 Autopilots



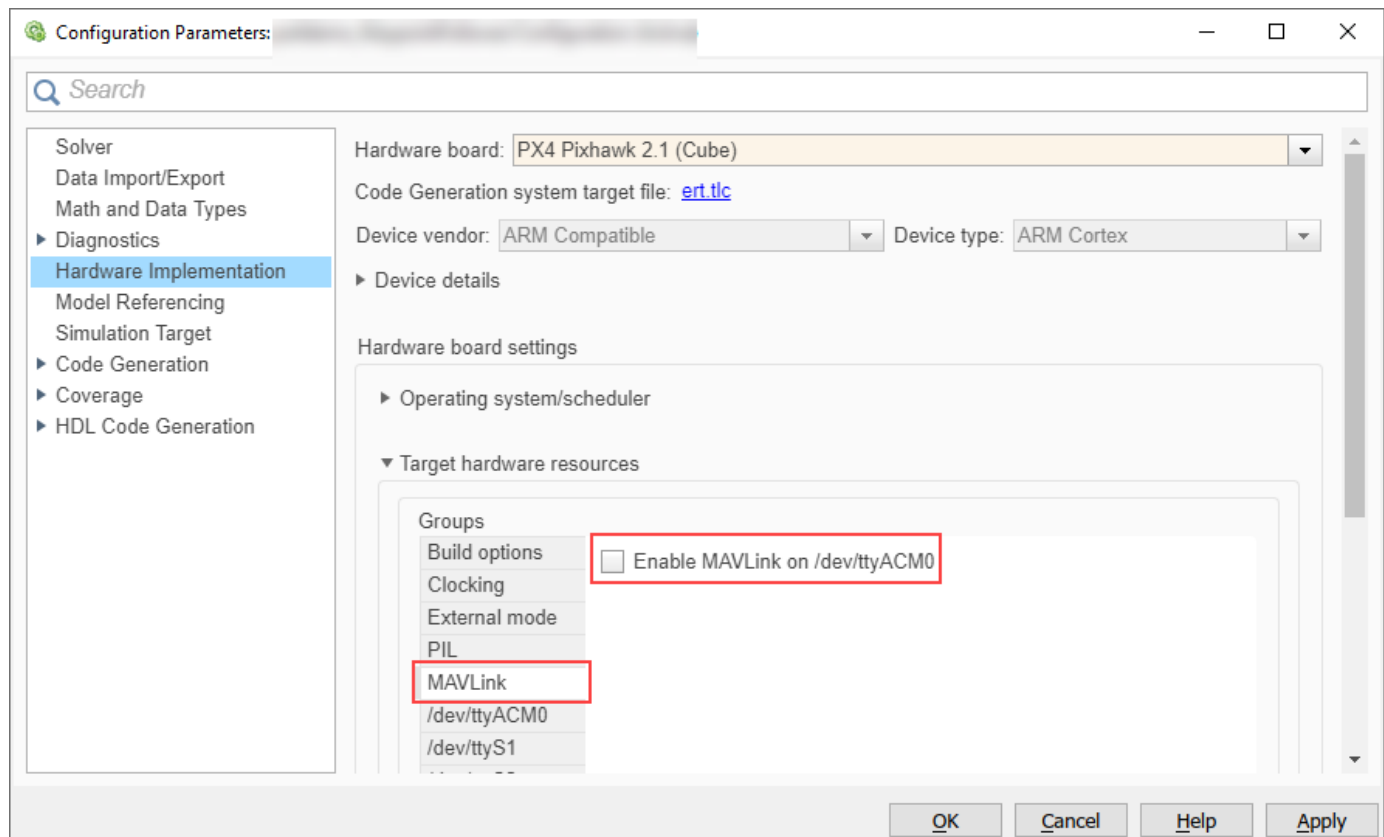
Enabling MAVLink in PX4 Over USB

MAVLink needs to be enabled for establishing connection between the PX4 flight controller and QGroundControl. From QGroundControl, you can access the NSH via USB.

When you complete the hardware setup process for UAV Toolbox Support Package for PX4 Autopilots (using the Hardware Setup screens), MAVLink is turned off by default.

However, UAV Toolbox Support Package for PX4 Autopilots provides the option to enable or disable MAVLink, if required. To enable MAVLink, open the Configuration Parameters dialog box, go to **Hardware Implementation > Target hardware resources > MAVLink**, and click **Enable MAVLink on /dev/ttyACM0**.

Note To enable or disable MAVLink, selecting or clearing the checkbox alone is not sufficient. After you select or clear the checkbox (**Enable MAVLink on /dev/ttyACM0**), go to the **Hardware** tab in the Simulink model window and click **Build, Deploy & Start**. This action regenerates the code with MAVLink enabled or disabled.



Note If you enable MAVLink over USB (/dev/ttyACM0), you cannot perform signal monitoring and parameter tuning (Monitor and Tune feature in Simulink) and PIL simulation over USB (the default port for these features being /dev/ttyACM0). To run the Simulink model using the Monitor and Tune feature or by using PIL, with MAVLink also enabled, you can choose any other serial port for those

features (for example, choose `/dev/ttyS6`, under **Target hardware resources > External mode**). However, in this case, you may need additional serial to USB convertor.

Note For UAV Toolbox Support Package for PX4 Autopilots versions prior to R2020a (20.1.0), you needed to edit `rc.txt` file manually (by adding two lines) to enable MAVLink. Therefore, when you migrate to version R2020a (20.1.0), which provides the option to enable or disable MAVLink in the Configuration Parameters dialog box, ensure that you remove the two lines in the `rc.txt` file in the micro-SD card connected to the PX4 flight controller. The lines that you need to remove from `rc.txt` (if already added) are:

```
mavlink start -d /dev/ttyACM0 -m config -x
```

```
mavlink boot_complete
```

Plant and Attitude Controller Model for Hexacopter

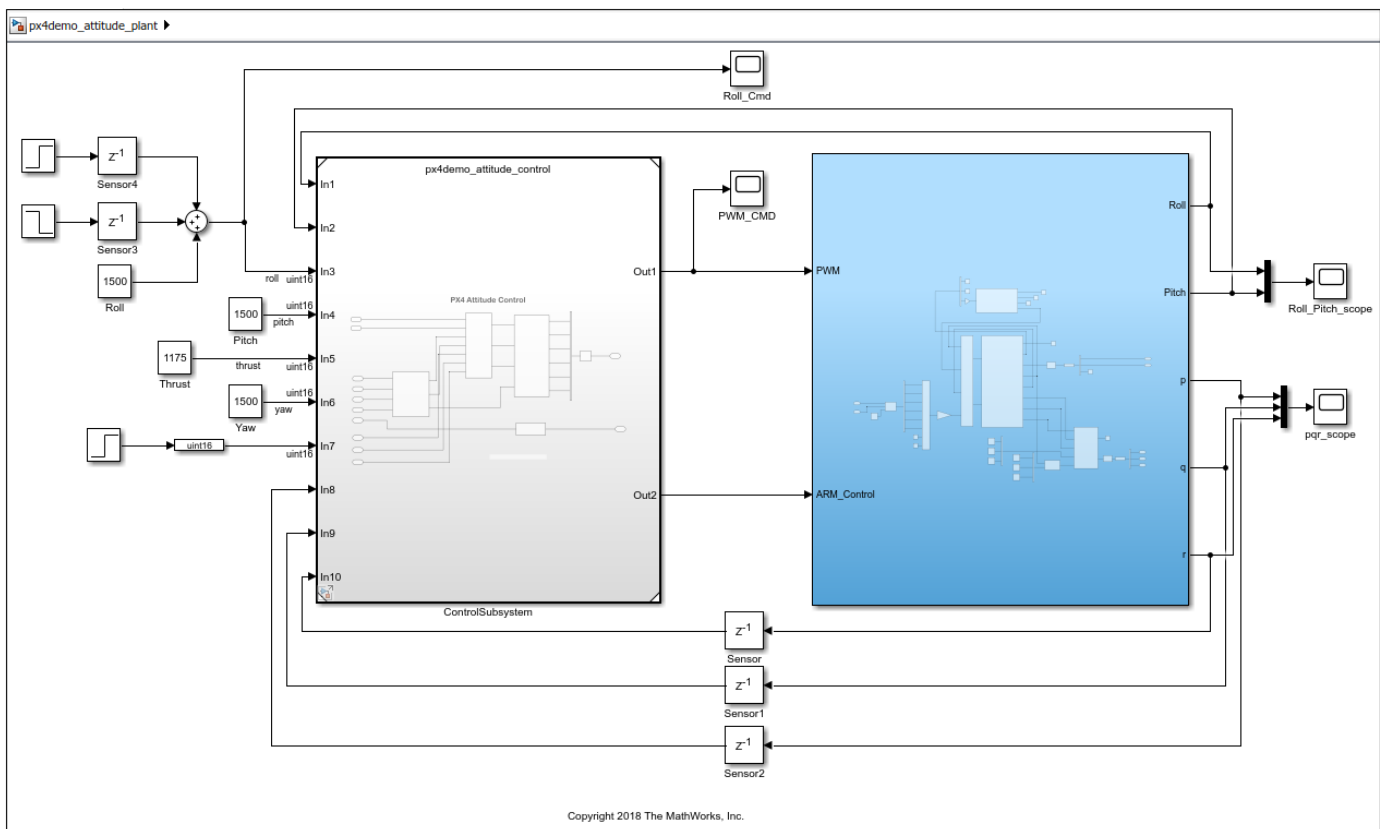
The UAV Toolbox Support Package for PX4 Autopilots contains a plant and an attitude controller model to fly a hexacopter drone that uses a Pixhawk Series flight controller.

Simulate the Plant Model for Hexacopter

The plant model, `px4demo_attitude_plant`, simulates the 6 DOF, and it outputs the roll, pitch, yaw and thrust values, which are then fed to the InputConditioning subsystem in the `px4demo_attitude_control` model. The attitude controller generates PWM pulse widths which are then provided to the plant as feedback.

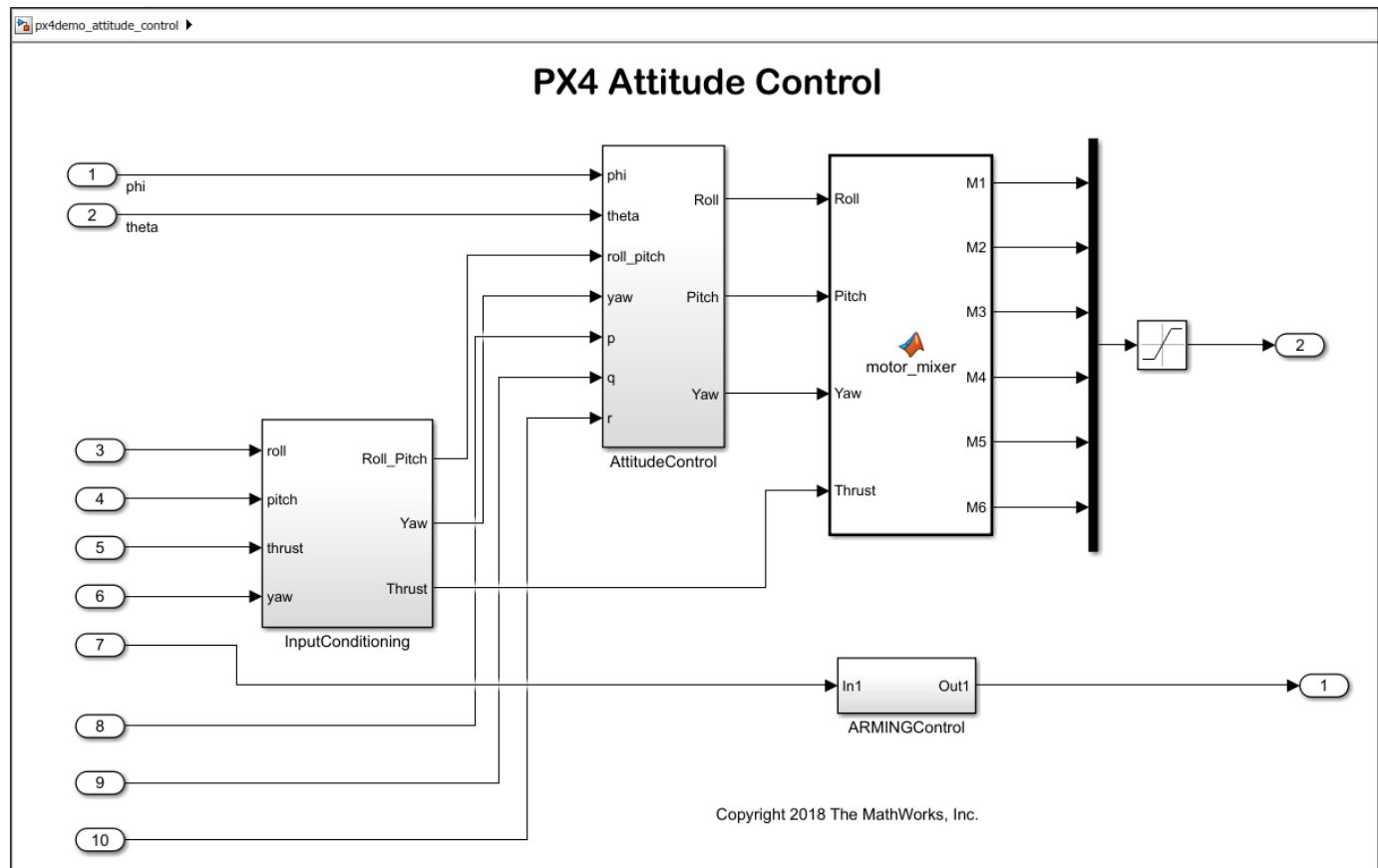
To open the plant model, enter the following command at the MATLAB prompt:

```
px4demo_attitude_plant
```



This plant model simulates the behavior when the roll command from the RC Transmitter varies; the pitch, yaw and thrust values are kept constant. The model can be modified to simulate changes for pitch and yaw as well.

The `px4demo_attitude_control` subsystem takes the roll, pitch, yaw and thrust values as input from user. It also accepts the vehicle attitude and IMU measurements for gyroscope. These values are then fed to the PID controller. The output of PID controller is fed to the mixer matrix for hexacopter to output the PWM pulse widths. The PID controller might need to be tuned according to your airframe. The mixer matrix will vary from airframe to airframe.

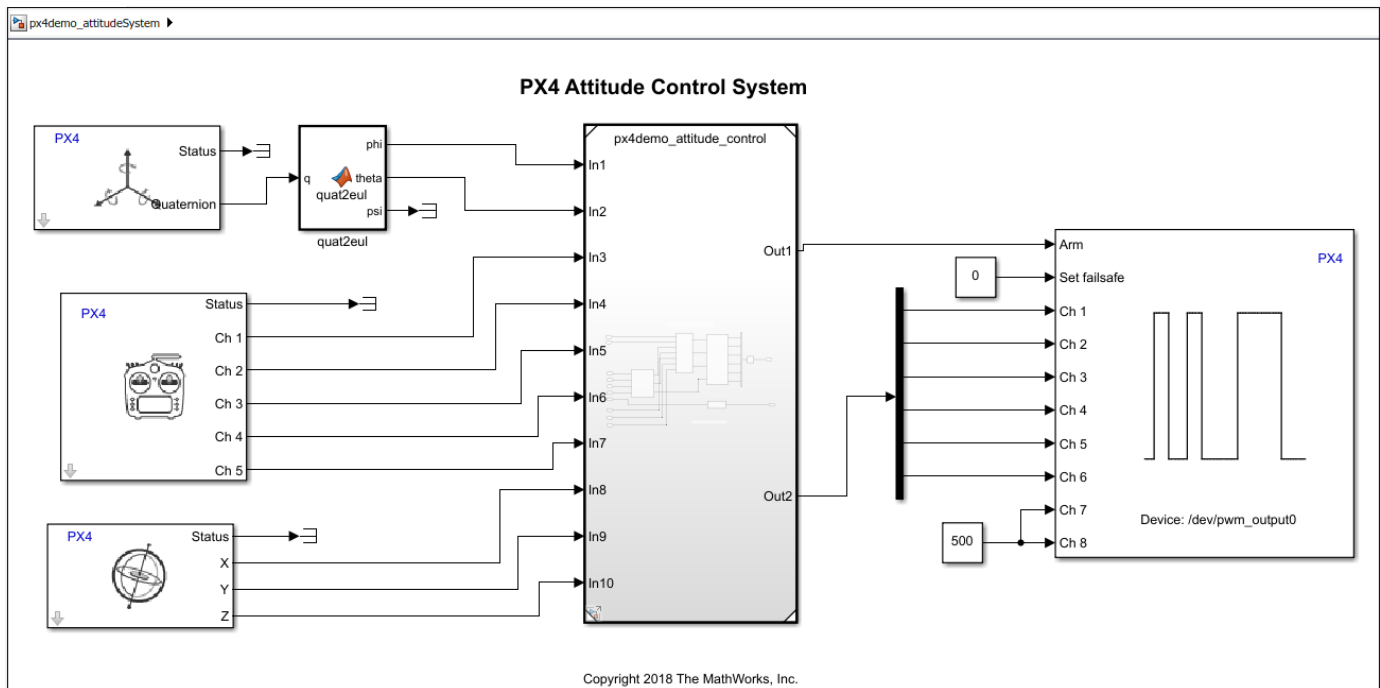


Generate Code for Controller for Hexacopter

The controller model, `px4demo_attitudeSystem`, gathers the inputs from Gyroscope, Radio Controller and Vehicle Attitude block and sends them to the attitude controller `px4demo_attitude_control`, which returns the PWM pulse widths which are then published.

To open the controller model, enter the following command at the MATLAB prompt:

```
px4demo_attitudeSystem
```



Build the above model and deploy it to the selected Pixhawk Series flight controller.

Adapting the Plant and Attitude Controller for Other Airframes

For modifying the controller for any other airframes, the corresponding mixer matrix needs to be added by replacing the existing `motor_mixer` function block in `px4demo_attitude_control` model.

The PID controller also needs to be tuned according to the airframe.

See Also

Migrating from Pixhawk Pilot Support Package to UAV Toolbox Support Package for PX4 Autopilots

Till R2018a, MathWorks provided support for the Pixhawk Pilot Support Package that has been used for developing Simulink models for the Pixhawk FMU using the PX4 toolchain. From R2018b, the official hardware support packages – Embedded Coder® Support Package for PX4 Autopilots (till R2020a) and UAV Toolbox Support Package for PX4 Autopilots (from R2020b) – are available for download.

After you download UAV Toolbox Support Package for PX4 Autopilots, you can migrate Simulink models that you developed using Pixhawk Pilot Support Package to the new support package.

Note The UAV Toolbox Support Package for PX4 Autopilots contains blocks that are not backward compatible with the blocks of the Pixhawk Pilot Support Package. However, you can replace the blocks in older models with the newer blocks and deploy the model using UAV Toolbox Support Package for PX4 Autopilots.

Note Before proceeding with the migration procedure, it is highly recommended that you take a backup of the existing Simulink model that you created using Pixhawk PSP.

The migration process involves the following steps:

- 1 Complete the setup and configuration activities of UAV Toolbox Support Package for PX4 Autopilots using the Hardware Setup screens (see “Setup and Configuration”).

The new support package supports PX4 firmware version 1.10.2 whereas the Pixhawk PSP supports PX4 firmware version 1.6.5. Therefore, you also need to clone PX4 firmware v1.10.2 by following the steps mentioned in the Hardware Setup screens.


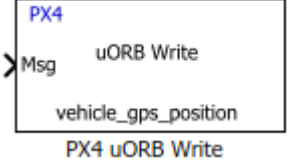
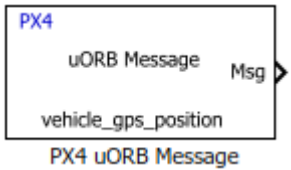
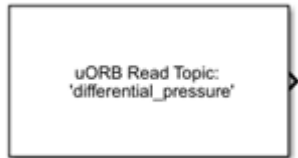
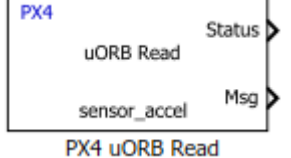
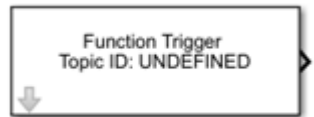
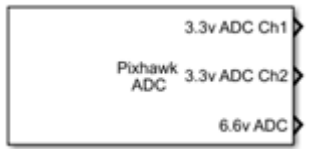
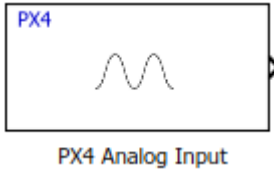
- 2 Launch Simulink and create a new model.
- 3 Open the Configuration Parameters dialog box and select the hardware (go to the Hardware Implementation pane and use the **Hardware board** drop-down list to select the Pixhawk board that you will be using to deploy the Simulink model). The new support package provides support for Pixhawk 1, Pixhawk 2.1, Pixracer, and Pixhawk 4 flight controllers.

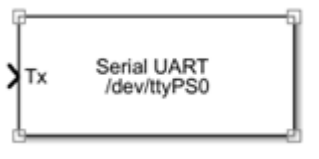
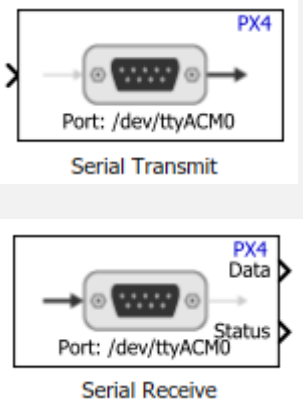

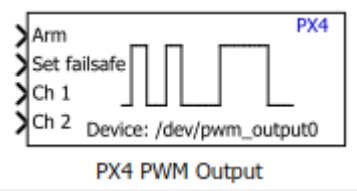

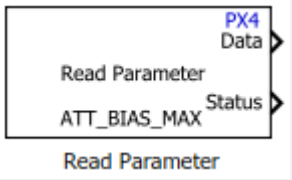
If you are using a board that is not officially supported, select the **PX4 Pixhawk Series** option.

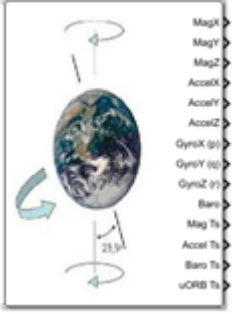
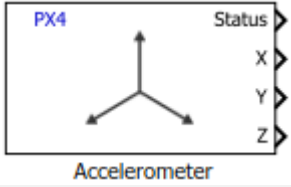
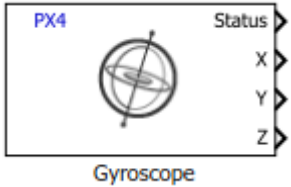
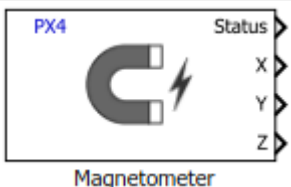

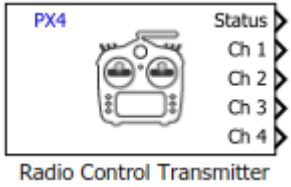
- 4 Copy the contents of the entire Simulink model from Pixhawk PSP to the new Simulink model that you created.
- 5 Replace the Simulink blocks for Pixhawk Pilot Support Package with the new Simulink blocks that are available in the new support package.

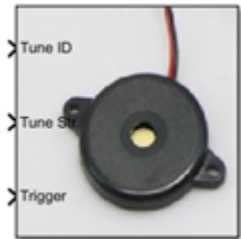
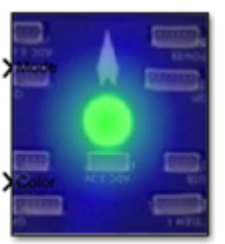

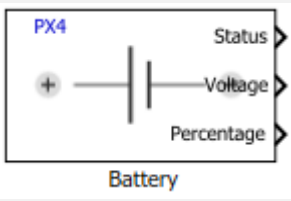
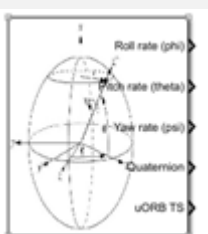
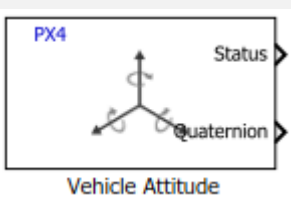
Use the following table to migrate the Simulink blocks to the new UAV Toolbox Support Package for PX4 Autopilots, in R2018b and later releases.

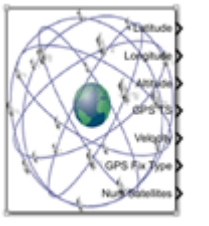


Mapping of Simulink blocks in Pixhawk Pilot Support Package to UAV Toolbox Support Package for PX4 Autopilots

Pixhawk Pilot Support Package Block	New Block in UAV Toolbox Support Package for PX4 Autopilots	Documentation Links
uORB Write / uORB Write Advanced 	PX4 uORB Write and PX4 uORB Message  	PX4 uORB Write and PX4 uORB Message
uORB Read 	PX4 uORB Read 	PX4 uORB Read
Read uORB Function Trigger 	<Not supported>	
ADC 	PX4 Analog Input 	PX4 Analog Input

Pixhawk Pilot Support Package Block	New Block in UAV Toolbox Support Package for PX4 Autopilots	Documentation Links
<p>Serial</p> 	<p>Serial Transmit and Serial Receive</p> 	<p>Serial Transmit and Serial Receive</p>
<p>PWM</p> 	<p>PX4 PWM Output</p> 	<p>PX4 PWM Output</p>
<p>ParamUpdate</p> 	<p>Read Parameter</p> 	<p>Read Parameter</p>

Pixhawk Pilot Support Package Block	New Block in UAV Toolbox Support Package for PX4 Autopilots	Documentation Links
<p>Sensor_Combined</p> 	<p>Accelerometer</p>  <p>Gyroscope</p>  <p>Magnetometer</p> 	<p>Accelerometer</p> <p>Gyroscope</p> <p>Magnetometer</p>
<p>Input_RC</p> 	<p>Radio Control Transmitter</p> 	<p>Radio Control Transmitter</p>

Pixhawk Pilot Support Package Block	New Block in UAV Toolbox Support Package for PX4 Autopilots	Documentation Links
<p>Speaker Tune</p> 	<p>This block can be modelled using uORB Write block and tune_control uORB topic. Refer to px4demo_LEDBuzzer Simulink model.</p>	
<p>LED</p> 	<p>This block can be modelled using uORB Write block and led_control uORB topic. Refer to px4demo_LEDBuzzer Simulink model.</p>	
<p>Battery</p> 		Battery
<p>Vehicle Attitude</p> 		Vehicle Attitude

Pixhawk Pilot Support Package Block	New Block in UAV Toolbox Support Package for PX4 Autopilots	Documentation Links
<p>Vehicle GPS</p> 	<p>This block can be modelled using uORB Read block and vehicle_gps uORB topic. Refer to px4_readGPS Simulink model.</p>	<p>“Reading GPS Data from PX4 Autopilot”</p>
<p>Binary Logger</p> 	<p>Refer to px4_SDCard_logging Simulink model.</p>	
<p>Print</p> 	<p><Not supported></p>	

Integrating uORB Topics in the Simulink Model

Integrating uORB Topics in the Simulink Model

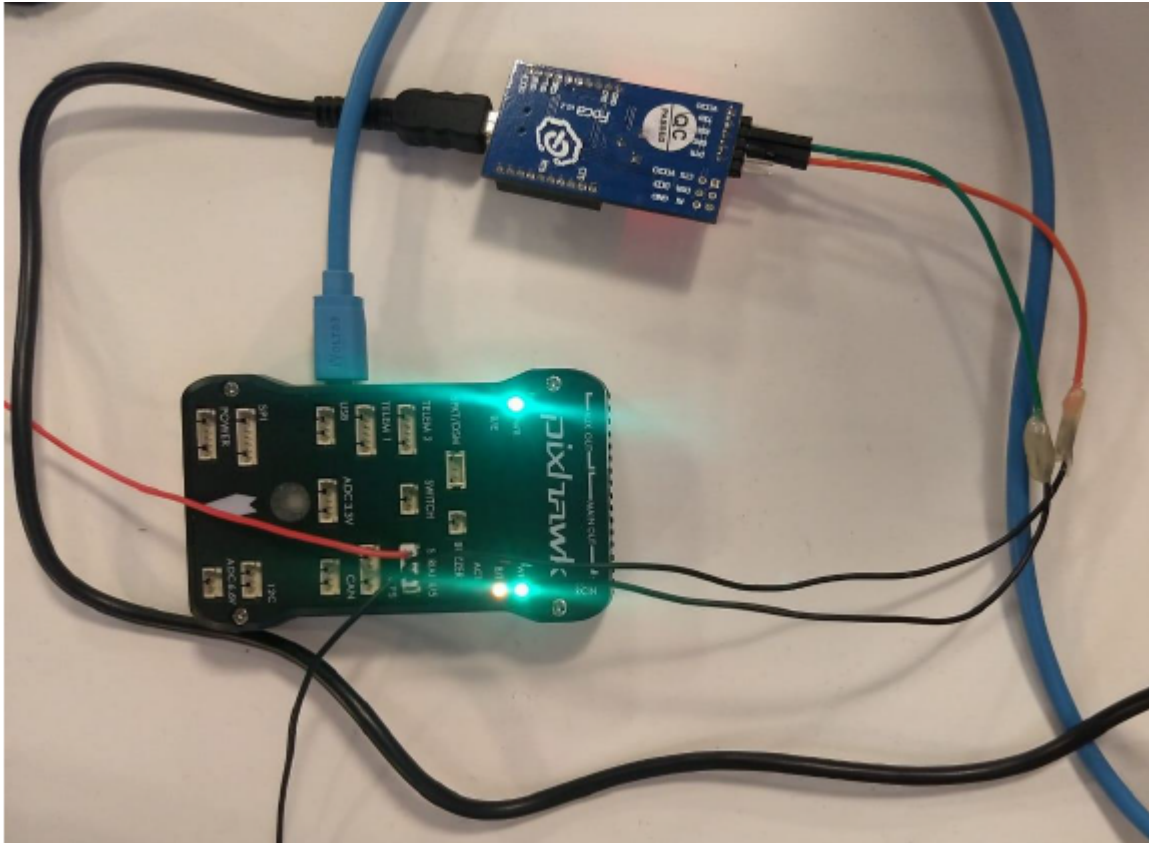
Setting Up the Hardware and Deploying the Model

Setting Up the Hardware and Deploying the Model

Using External Mode over FTDI with Pixhawk 1

In UAV Toolbox Support Package for PX4 Autopilots, serial communication is supported over the native-USB port and non-usb ports (for example, /dev/ttyS6) on the Pixhawk Series boards. To achieve serial communication over non-USB serial port, a serial-to-USB FTDI converter is required.

The below image shows the connection between Serial 4 (/dev/ttyS6) on Pixhawk 1 board and the USB port on host computer, by using an FTDI converter.



The below table lists the serial pins on the Pixhawk 1 board and their FTDI mapping. The pins listed here are from left to right.

Pin and FTDI Mapping for Pixhawk 1

Serial port on Pixhawk 1		FTDI
1	+5V	Not connected
2	Serial 4 Tx	FTDI Rx
3	Serial 4 Rx	FTDI Tx
4	Serial 5 Tx	Not connected
5	Serial 5 Rx	Not connected
6	GND	FTDI GND

Once the above hardware connections are completed, open the Configuration Parameters dialog box in Simulink, and set the serial port to **dev/ttyS6** (go to Hardware Implementation pane > Target Hardware Resources > External Mode, and enter **dev/ttyS6** in the Hardware board serial Port field). Now you can run the model in External mode over FTDI with Pixhawk 1 board.

Connecting to NSH Terminal for Debugging

After you deploy the model created using UAV Toolbox Support Package for PX4 Autopilots, you can use the NSH terminal for the debugging of Pixhawk Series controllers.

NSH is accessed using MAVLink. Therefore, ensure that MAVLink is enabled over USB (for details, see “Enabling MAVLink in PX4 Over USB” on page 3-5).

Note If you enable MAVLink over USB (/dev/ttyACM0), then you cannot use the External mode over USB. You can choose any other serial port (for example, /dev/ttyS6) to run the Simulink model in External mode. However, in this case, you may need additional serial to USB convertor.

Accessing NSH from MATLAB

UAV Toolbox Support Package for PX4 Autopilots provides a pre-defined class named `HelperPX4` that helps you to access NSH and perform certain actions.

Find the serial port on the host computer to which the PX4 Autopilot is currently connected and create a `HelperPX4` object to access NSH. For example:

```
shellObj = HelperPX4('COM9');
```

Here `COM9` value is used as an example. Change it to the actual host serial port.

The `HelperPX4` class provides different methods to send NSH commands and obtain response.

The `system` method helps you to send the commands to NSH and get the shell response. To see the list of available NSH commands under `system` method, use the `help` command:

```
[shellResp, status] = shellObj.system('help')
```

The `status` value 1 indicates a successful execution.

If the `listener` command is listed, you can use it in the `system` method to listen to any uORB topic. For example:

```
[shellResp, status] = shellObj.system('listener vehicle_status')
```

The `getFile` method helps you to get the fault log or any other file in the SD card mounted on the PX4 Autopilot target, without removing SD card from the target. To do this (for fault logs):

- 1 See the list of fault logs available in the SD card using the `system` method:

```
shellObj.system('ls /fs/microsd')
```

- 2 Observe the log list and decide which log file you want to copy to the host computer. Get the file to host computer using the `getFile` method. For example:

```
shellObj.getFile('/fs/microsd/fault_2019_10_24_10_49_04.log');
```

Tip You can use the `getFile` method to edit the startup script `rc.txt` that you have copied to the SD card (for details about `rc.txt`, see “Performing PX4 System Startup from SD Card” on page 1-24). In this case, you can also use the `putFile` method to copy the modified startup script back to the SD card.

After you make all the changes, you can delete the `HelperPX4` object that you created:

```
delete(shellObj);
```

Accessing NSH using QGroundControl (QGC)

For accessing NSH using QGroundControl (QGC), refer to [this link](#).

Deployment and Verification Using PX4 Host Target and jMAVSim

UAV Toolbox Support Package for PX4 Autopilots provides the option to simulate PX4 autopilot algorithms that you develop in Simulink. The workflow is based on PX4 SITL (Software in the Loop). The support package supports simulation of algorithms by generating an executable on the host, referred as **PX4 Host Target**, and the jMAVSim simulator.

Note The **PX4 Host Target** supports code generation and deployment like other supported hardware boards. You can also perform signal monitoring and parameter tuning of the Simulink model by selecting this option.

About jMAVSim

jMAVSim is one of the supported simulators for PX4-based targets. During the hardware setup of UAV Toolbox Support Package for PX4 Autopilots, this simulator is downloaded and installed, and allows you to view the flight of a quadcopter running the PX4 algorithm that you develop in Simulink. By using jMAVSim, you can test the algorithm for the desired take off, flight, and land actions.

Preparing PX4 Host Target and jMAVSim Using Hardware Setup Screens

You can use Hardware Setup process in UAV Toolbox Support Package for PX4 Autopilots to prepare the support package for using the **PX4 Host Target** and testing the connection with jMAVSim simulator.

Perform these steps as part of Hardware Setup to enable **PX4 Host Target** and test the jMAVSim simulator:

- 1 In the **Select a PX4 Autopilot and build configuration** Hardware Setup screen, select **PX4 Host Target** as the PX4 Autopilot board. The supported Build Target file is `posix_sitl_default`.

Proceed with the subsequent step of Hardware Setup process to build the firmware and verify that the firmware build is successful.

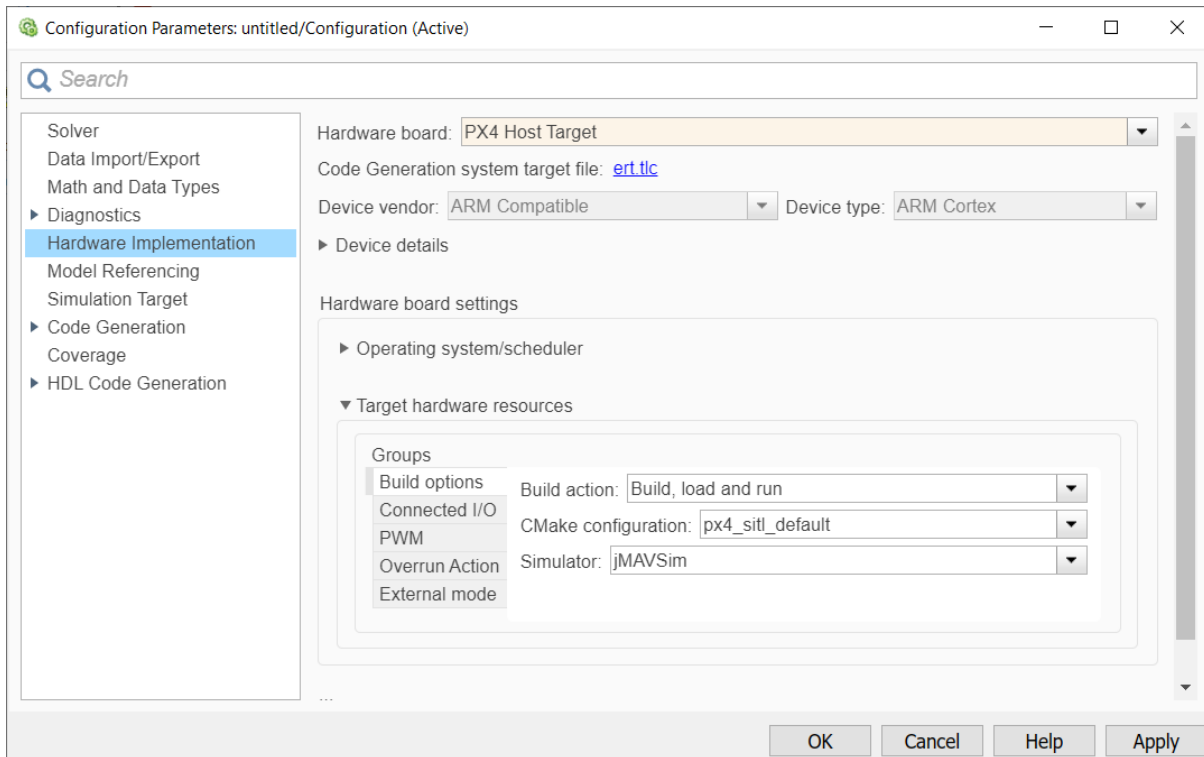
- 2 In the **Test Connection** Hardware Setup screen, click **Launch PX4 Host Target and jMAVSim**. The jMAVSim simulator opens in a new window and shows the flight of the quadcopter. This is a test flight using a pre-configured algorithm in the support package to ensure that the necessary components for the simulator are installed.



Launching PX4 Host Target and jMAVSim Simulator from the Simulink Model

After you prepare the flight controller algorithm using the Simulink blocks available in UAV Toolbox Support Package for PX4 Autopilots, you can launch the jMAVSim simulator to verify the flight. To do this:

- 1 In the **Modeling** tab, click **Model Settings**. In the Configuration Parameters dialog box:
 - a In the **Hardware board** list, select **PX4 Host Target**.
 - b Under **Target Hardware resources > Build Options**, select the Build action as **Build, load and run**.



c Click **Apply** and then **OK**.

2 In the **Hardware** tab, select one of these options to launch jMAVSim simulator:

- To view the flight of the PX4 based flight controller based on the algorithm that is present in the Simulink model, click **Build, Deploy & Start**. The jMAVSim simulator is launched after sometime in a separate window.
- To view the flight of the PX4 based flight controller based on the algorithm and perform signal monitoring and parameter tuning of the Simulink model, click **Monitor & Tune**. The jMAVSim simulator is launched in a separate window. You can change the parameters in the Simulink model and view the change in the simulated flight at runtime in the jMAVSim simulator.

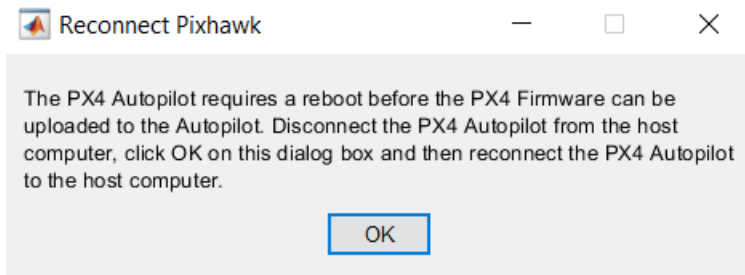
Note With PX4 Firmware v1.10.2, it has been observed that the **PX4 Host Target** may exhibit issues with modules not responding after one hour of simulation time.

Note On Ubuntu 18.04 platform, it is recommended that you set the sample time as 1 millisecond for the Simulink models to be deployed on **PX4 Host Target**. If the sample time is greater than 1 millisecond, issues may occur while writing to some uORB topics such as `actuator_outputs`.

Troubleshooting Deploy to Hardware Issues

Description

When you try to deploy a Simulink model created using UAV Toolbox Support Package for PX4 Autopilots to the selected Pixhawk® Series flight controller using the USB cable, a pop-up dialog box appears with the instruction to disconnect and reconnect the flight controller. The deployment of the model may not happen even though you click **OK** in that dialog box.



Action

To resolve the issue:

- Ensure that you click **OK** within 5 seconds after disconnecting and reconnecting the USB cable connected between the host computer and the Pixhawk Series flight controller.

Tip The 5-seconds limit applies only to the time after reconnection. Therefore, you can first disconnect the USB port from the host computer without worrying about the time limit; but, ensure that you click **OK** within 5 seconds after reconnecting the cable to the USB port of the host computer.

- After you click **OK**, verify the progress in the Diagnostic Viewer in Simulink. The Diagnostic Viewer displays the following lines that indicate the start of a successful deployment.

```
Using COM17 for upload.
Loaded firmware for 9,0, size: 1017552 bytes, waiting fc
Found board 9,0 bootloader rev 4 on COM17
50583400 00ac2600 00100000 00ffffff ffffffff ffffffff ff
ef019ca5 c89bb183 bb00f0c0 06db1a26 7375ff57 1ca41d94 24
Erase : [ ] 0.0%
Erase : [= ] 5.6%
Erase : [== ] 11.1%
Erase : [=== ] 16.7%
Erase : [==== ] 22.3%
Erase : [===== ] 27.8%
Erase : [===== ] 33.4%
Erase : [===== ] 38.9%
Erase : [===== ] 44.5%
Erase : [===== ] 50.1%
Erase : [===== ] 55.6%
Erase : [===== ] 61.2%
Erase : [===== ] 66.7%
Erase : [===== ] 72.3%
Erase : [===== ] 77.9%
Erase : [===== ] 83.4%
Erase : [===== ] 100.0%

Program: [= ] 6.3%
Program: [== ] 12.7%
Program: [=== ] 19.0%
Program: [==== ] 25.4%
Program: [===== ] 31.7%
Program: [===== ] 38.0%
Program: [===== ] 44.4%
Program: [===== ] 50.7%
Program: [===== ] 57.1%
Program: [===== ] 63.4%
Program: [===== ] 69.7%
Program: [===== ] 76.1%
Program: [===== ] 82.4%
Program: [===== ] 88.8%
Program: [===== ] 95.1%
Program: [===== ] 100.0%

Verify : [ ] 1.0%
Verify : [===== ] 100.0%
Rebooting.
```

Note If you do not see the progress of deployment in the Diagnostic Viewer (as shown in the above image), try the process again (disconnect and reconnect the USB cable), and then click **OK** within 5 seconds.

See Also

Troubleshooting Flash Overflow Issues with Pixhawk 1

Description

If you are trying to deploy the Simulink model configured for a Pixhawk 1 flight controller by clicking



the **Build, Deploy & Start** icon () in the **Hardware** tab of Simulink toolstrip, you may see the following flash overflow error towards the end of the build process:

```
Scanning dependencies of target nuttx_px4fmv-v2_default.elf
[100%] Linking CXX executable ../nuttx_px4fmv-v2_default.elf
/home/abose/gcc-arm-none-eabi-7-2017-q4-major/bin/../lib/gcc/arm-none-eabi/7.2.1/../../../../arm-none-eabi/bin/ld: ../nuttx_px4fmv-v2_default.elf section `.text' will not fit in region `flash'
/home/abose/gcc-arm-none-eabi-7-2017-q4-major/bin/../lib/gcc/arm-none-eabi/7.2.1/../../../../arm-none-eabi/bin/ld: address 0x8104160 of ../nuttx_px4fmv-v2_default.elf section `__param' is not within region `flash'
/home/abose/gcc-arm-none-eabi-7-2017-q4-major/bin/../lib/gcc/arm-none-eabi/7.2.1/../../../../arm-none-eabi/bin/ld: address 0x8104160 of ../nuttx_px4fmv-v2_default.elf section `__param' is not within region `flash'
/home/abose/gcc-arm-none-eabi-7-2017-q4-major/bin/../lib/gcc/arm-none-eabi/7.2.1/../../../../arm-none-eabi/bin/ld: region `flash' overflowed by 20680 bytes
collect2: error: ld returned 1 exit status
platforms/nuttx/CMakeFiles/nuttx_px4fmv-v2_default.elf.dir/build.make:217: recipe for target 'nuttx_px4fmv-v2_default.elf' failed
make[3]: *** [nuttx_px4fmv-v2_default.elf] Error 1
CMakeFiles/Makefile2:6363: recipe for target 'platforms/nuttx/CMakeFiles/nuttx_px4fmv-v2_default.elf.dir/all' failed
make[2]: *** [platforms/nuttx/CMakeFiles/nuttx_px4fmv-v2_default.elf.dir/all] Error 2
Makefile:105: recipe for target 'all' failed
make[1]: *** [all] Error 2
Makefile:154: recipe for target 'px4fmv-v2_default' failed
make: *** [px4fmv-v2_default] Error 2
gmake: *** [postdownload_preexecute] Error 2
```

The reasons for this issue can be:

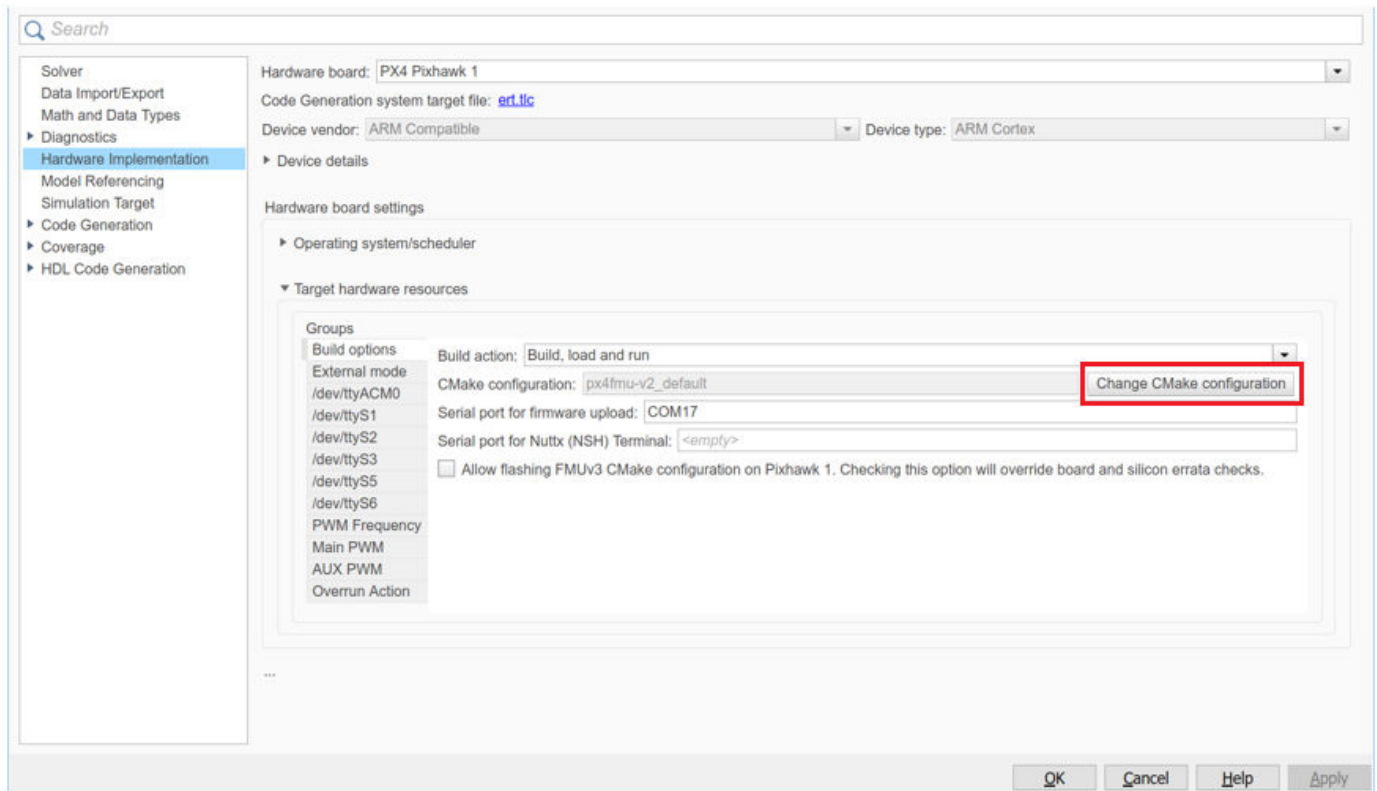
- You have used the CMake file `px4fmv-v2_default`, which is the default for the Pixhawk 1 flight controller, but the 1MB flash memory limit is exceeded.
- The Simulink model is large, and the generated code has exceeded the 1MB flash memory limit.

Action

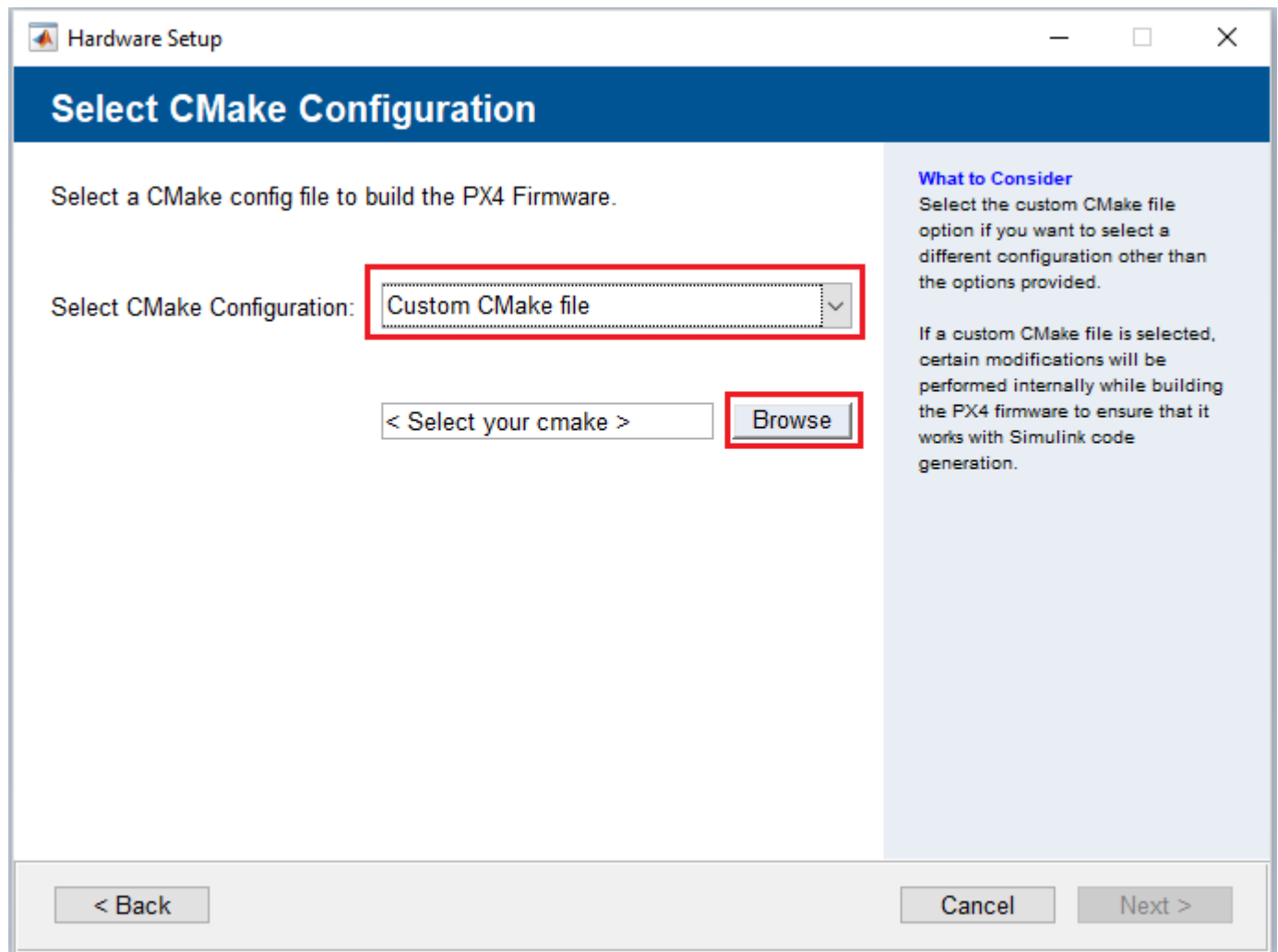
To troubleshoot the issue, there are two options:

- Select another CMake file for the firmware build and configuration. To do this, perform the following tasks:
 - 1 From the Simulink model, open the Configuration Parameters dialog box.
 - 2 Go to Target Hardware Resources > Build options pane, and click **Change CMake configuration** to open the Hardware Setup screen.

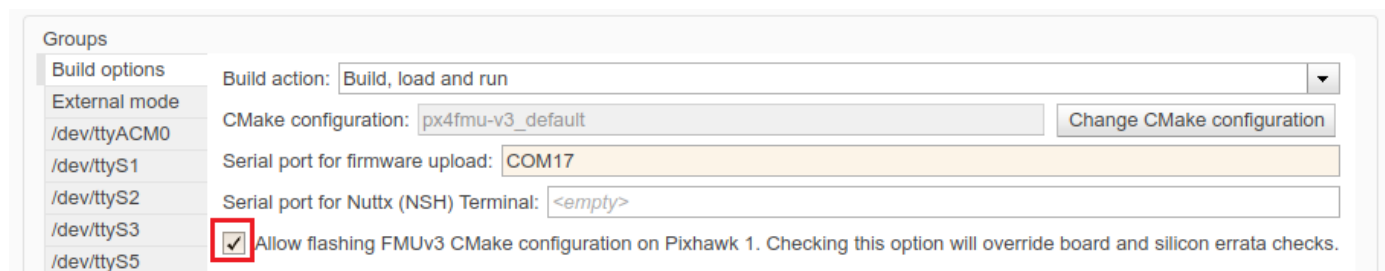
3 Run on Target for UAV Toolbox Support Package for PX4 Autopilots



- 3 In the Hardware Setup screens, go to the **Select CMake Configuration** screen, select **Custom CMake file**, and then click **Browse**.



- 4 Select **nuttx_px4fmu-v3_default.cmake**, and click **Open**.
- 5 Click **Next** and proceed with the remaining steps of Hardware Setup process.
- 6 From the Simulink model, open the Configuration Parameters dialog box again, and select the option **Allow flashing FMUv3 CMake configuration on Pixhawk 1**.




Note This option should be carefully used because enabling this option will result in the following:

- There will not be any check either for the selected CMake compatibility on the connected Pixhawk controller or for any silicon errata on the Pixhawk board.
- If the CMake is not compatible with the Pixhawk controller, the entire board can get into a faulty state.

Hence, only use this option if you have connected Pixhawk 1 and you want to flash FMUv3 CMake on Pixhawk 1.

7 Click **Apply** and then **OK** to close the dialog box.

8

In the **Hardware** tab of the Simulink toolstrip, click **Build, Deploy & Start** () to start the process again.

- If you do not want to change the CMake (px4fmu-v2_default) that is designed for the hardware, you can also troubleshoot the issue if you change the contents of the CMake file. To do this:
 - 1** Go to the location of the CMake files in the px4 directory (for example, C:\mypx4\Firmware\cmake\configs).
 - 2** Edit the **nuttx_px4fmu-v2_default.cmake** file and remove the modules that you do not want to build (you can comment-out those lines).
 - 3** Open the Simulink model and try the **Deploy to Hardware** process again.

See Also

Monitor and Tune the Model Running on PX4 Autopilots

In this section...
“Prepare a Simulink Model for External Mode” on page 3-32
“Running the Simulink Model for Monitor and Tune” on page 3-33
“Signal Monitoring and Parameter Tuning of Simulink Model” on page 3-34
“Stop Monitor and Tune” on page 3-35
“Performing Disconnect and Connect” on page 3-36
“Performing Connect Operation to Run an Unchanged Simulink Model on Hardware” on page 3-36

You can use Monitor and Tune (External Mode) action to tune parameters and monitor a Simulink model running on your target hardware.

Monitor and Tune enables you to tune model parameters and evaluate the effects of different parameter values on model results in real-time. When you change parameter values in a model, the modified parameter values are communicated to the target hardware immediately. You can monitor the effects of different parameter values by viewing the output signals on Sink blocks or in Simulation Data Inspector (SDI). Doing so helps you find the optimal values for performance. This process is called parameter tuning.

Monitor and Tune accelerates parameter tuning. You do not have to rerun the model each time you change parameters. You can also use Monitor and Tune to develop and validate your model using the actual data and hardware for which it is designed. This software-hardware interaction is not available solely by simulating a model.

The support package supports Monitor and Tune simulation over the below communication interfaces when PX4 Host Target is selected as hardware board:

Communication Interface	Description
XCP on TCP/IP	<p>In Universal Measurement and Calibration Protocol (XCP)-based External mode simulation over TCP/IP, you can use:</p> <ul style="list-style-type: none"> “Dashboard” blocks: In addition to “Sources” and Sink blocks, you can use “Dashboard” blocks to change parameter values and to monitor the effects of parameter tuning. The Dashboard library contains set of blocks using which you can interactively control and visualize the model. Simulation Data Inspector (SDI): You can inspect and compare data from multiple simulations to validate model designs using Simulation Data Inspector.
TCP/IP	In External mode simulation over TCP/IP, use the “Sources” blocks to change parameter values and the Sink blocks to monitor the effects of parameter tuning.

The support package supports Monitor and Tune simulation over the below communication interfaces when Pixhawk 1, Pixhawk 2.1 (Cube), Pixhawk 4, Pixhawk Series, or Pixracer is selected as hardware board:

Note When Crazyflie 2.0 is selected as hardware board, only Serial communication interface is supported.

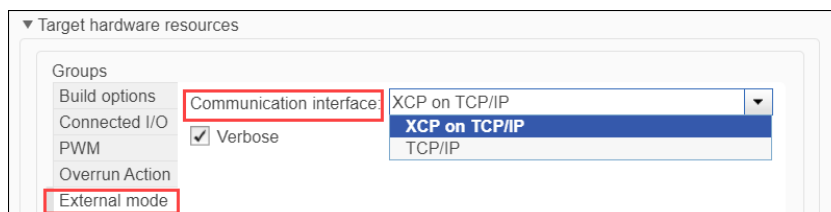
Communication Interface	Description
XCP on Serial	<p>In Universal Measurement and Calibration Protocol (XCP)-based External mode simulation over Serial, you can use:</p> <ul style="list-style-type: none"> “Dashboard” blocks: In addition to “Sources” and Sink blocks, you can use “Dashboard” blocks to change parameter values and to monitor the effects of parameter tuning. The Dashboard library contains set of blocks using which you can interactively control and visualize the model. Simulation Data Inspector (SDI): You can inspect and compare data from multiple simulations to validate model designs using Simulation Data Inspector.
Serial	In External mode simulation over Serial, use the “Sources” blocks to change parameter values and the Sink blocks to monitor the effects of parameter tuning.

Prepare a Simulink Model for External Mode

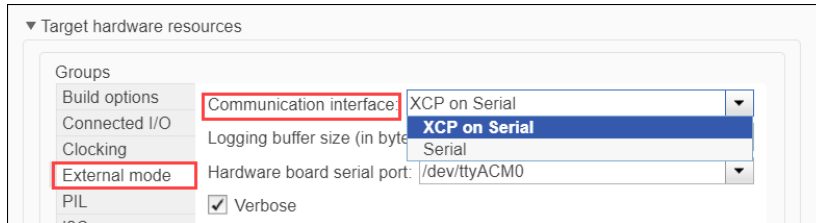
This section explains how to prepare a Simulink model to run in External mode.

- 1 Configure the Model Configuration Parameters to set the Hardware Board as one of the supported PX4 hardware boards, as explained in “Model Configuration Parameters for PX4 Flight Controller”.
- 2 In the Model Configuration Parameters dialog box, under **Hardware board settings > Target Hardware Resources**, select **External Mode** tab.

When PX4 Host Target is selected as hardware board, Communication interface options for External Mode are XCP on TCP/IP and TCP/IP:



When Pixhawk 1, Pixhawk 2.1 (Cube), Pixhawk 4, Pixhawk Series, or Pixracer is selected as hardware board, Communication interface options for External Mode are XCP on Serial and Serial:



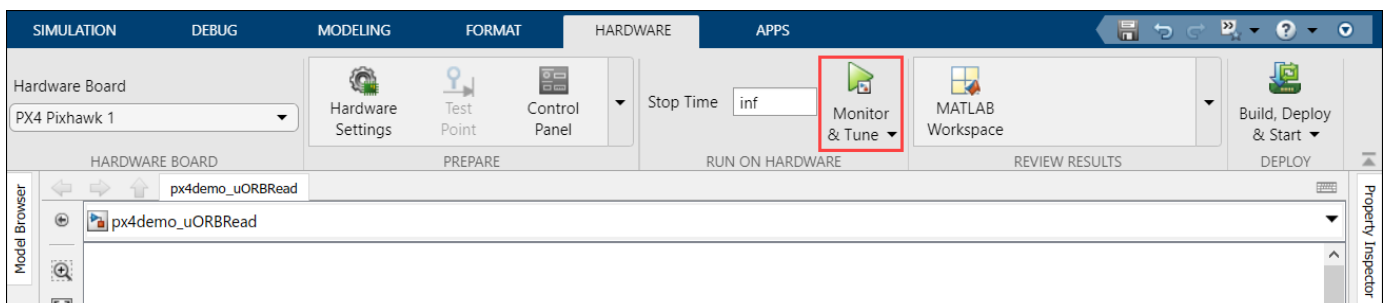
- 3 Ensure that you select the **Verbose** check box to view the external mode execution progress and updates in the Diagnostic Viewer or in the Command Window.
- 4 If the Communication interface is XCP on Serial, specify the number of bytes to allocate for the buffer in the hardware during simulation in the **Logging buffer size (in bytes)** parameter. This parameter is available only when Pixhawk 1, Pixhawk 2.1 (Cube), Pixhawk 4, Pixhawk Series, or Pixracer is selected as hardware board.
- 5 Select the required hardware board serial port. This parameter is available only when Pixhawk 1, Pixhawk 2.1 (Cube), Pixhawk 4, Pixhawk Series, or Pixracer is selected as hardware board.
- 6 If the Serial port for External Mode communication on hardware board is a port other than /dev/ttyACM0 mention the corresponding COM port on host.
- 7 Click **Apply** and then **OK** to close the Model Configuration Parameters dialog box.

Running the Simulink Model for Monitor and Tune

- 1 Connect the PX4 Autopilots hardware to your host computer.
- 2 Open the Simulink model and go to **Hardware** tab.
- 3 Set a value for the **Simulation stop time** parameter. The default value is 10.0 seconds. To run the model for an indefinite period, enter inf.
- 4 Click **Monitor & Tune** to run the model on external mode.

Simulink automatically:

- Runs the model on the target hardware.
- Runs the model on the host computer for Monitor and Tune operation.
- Creates a real-time connection between the model on target hardware and the model on the host computer.



Signal Monitoring and Parameter Tuning of Simulink Model

This section explains how to run:


- “XCP-Based External Mode Simulation over TCP/IP or Serial” on page 3-34
- “External Mode Simulation over TCP/IP or Serial” on page 3-35

XCP-Based External Mode Simulation over TCP/IP or Serial

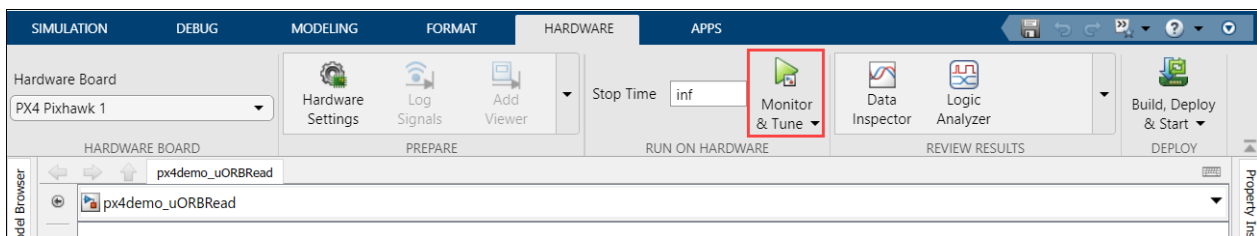
Before you begin, complete the “Prepare a Simulink Model for External Mode” on page 3-32 section.

- 1 In the Simulink model, identify the signals to be logged for monitoring during simulation. Select the identified signal, open its context menu, and click the icon corresponding to **Enable Data Logging**.

For instructions on logging the signal using other methods, refer to “Mark a Signal for Logging”.

Simulink displays a logged signal indicator  for each logged signal.

- 2 (Optional) Place one or more Sink blocks in the model, and then mark the signals connected to them also for logging. For example, connect Display or Scope blocks and mark the signals connected to them for logging.
- 3 To start the simulation, open the **Hardware** tab and click the **Monitor & Tune**.



For instructions on logging the signal, see “Mark a Signal for Logging”.

After several minutes, Simulink starts running the model on the hardware.

During simulation, when new simulation data becomes available in SDI, the Simulation Data

Inspector button  appears highlighted.

- 4 View the simulation output in Sink blocks or in SDI.

Note For XCP- Based External Mode over TCP/IP or Serial, it is recommended to use Signal Data Inspector (SDI) to view and log signals.

- Sink blocks - To view the simulation output, double-click the Sink blocks in the model.
- SDI - To view the new simulation data, perform these steps:
 - a Click the Simulation Data Inspector button.

- b A new simulation run appears in the **Inspect** pane. The **Inspect** pane lists all logged signals in rows, organized by simulation run. You can expand or collapse any of the runs to view the signals in a run. For more information on signal grouping, see .

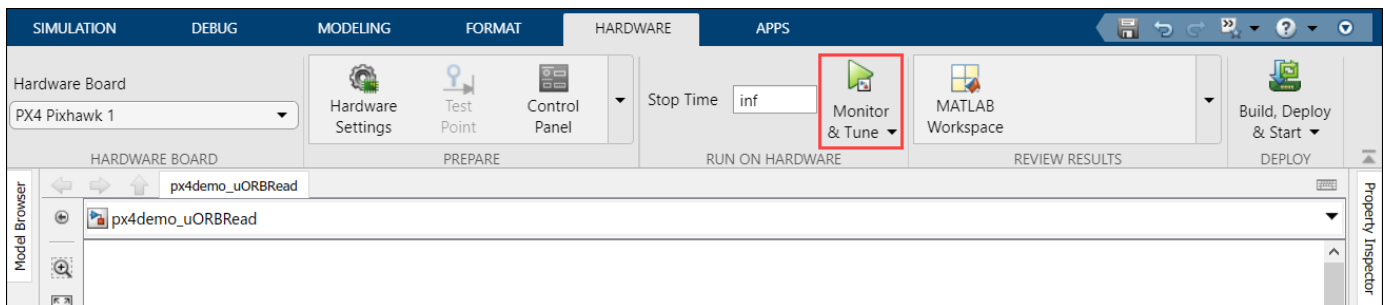
We recommend you use SDI rather than using Sink blocks for the following reasons:

- Streaming data to SDI does not store data in memory, making more efficient use of the memory available on the hardware. Sink blocks such as Scope stores data in buffers before sending the data to the host.
 - Using SDI, you can stream signals from top models and reference models simultaneously. Scope blocks can only log signals from a top-level model.
- 5 Change the parameter values in the model. Observe the corresponding changes in the simulation output.
 - 6 Find the optimal parameter values by adjusting and observing the results in the Sink blocks.
 - 7 After you are satisfied with the results, stop the Monitor and Tune action, and save the model.

External Mode Simulation over TCP/IP or Serial

Before you begin, complete the section “Prepare a Simulink Model for External Mode” on page 3-32 and ensure that you have placed Sink blocks in the model to monitor the simulation output.

- 1 To start the simulation, open the **Hardware** tab and click the **Monitor & Tune**.



After few seconds, Simulink starts running the model on the hardware.

- 2 Change the parameter values in the model. Observe the corresponding changes in the simulation output.
- 3 Find the optimal parameter values by adjusting and observing the results in the Sink blocks.
- 4 After you are satisfied with the results, stop the Monitor and Tune action, and save the model.

Stop Monitor and Tune

To stop the model that is running in Monitor and Tune, open the **Hardware** tab and click the **Stop**



If the Simulation stop time parameter is set to a specific number of seconds, Monitor and Tune stops when that time elapses.

Performing Disconnect and Connect

When you perform Monitor and Tune operation, you can use the **Disconnect** button to temporarily stop transferring the updated parameter values to the PX4 Autopilot. You can click **Connect** again to establish communication to send the updated values to PX4 Autopilot.

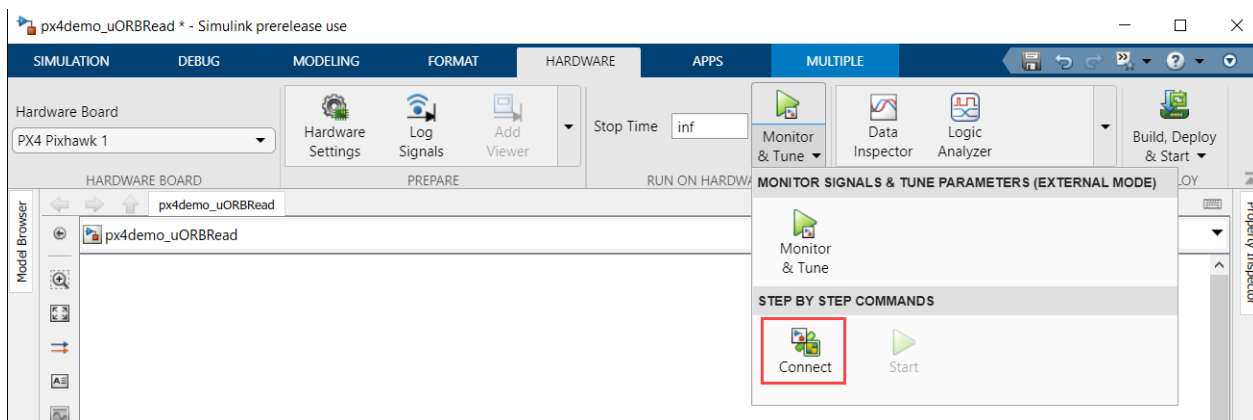
Performing Connect Operation to Run an Unchanged Simulink Model on Hardware

In some cases, the Monitor and Tune operation needs to be performed again even though there is no change to the Simulink model that was running on the PX4 Autopilot hardware. The reasons include:

- Hardware reboot of PX4 Autopilot while Monitor and Tune is in progress
- Disconnection of the port on the host computer while Monitor and Tune is in progress

You can resume Monitor and Tune operation again by:

- Restarting the hardware board.
- If Simulink code generated for **Monitor & Tune** in the previous run is already flashed on hardware, click **Connect** in Simulink instead of clicking **Monitor & Tune** again to establish External mode simulation. By clicking **Connect**, the Simulink code for the model is not generated again, and this avoids the time taken to build the code compared to the whole Monitor and Tune process.

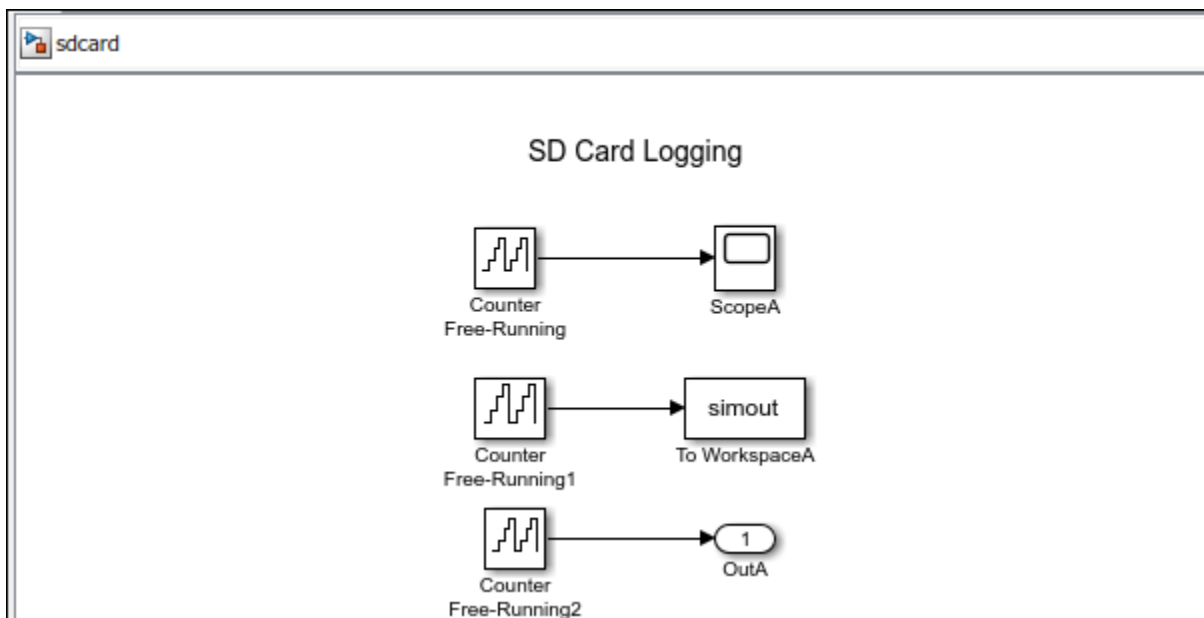


MAT-File Logging on SD Card

Log Signals on an SD Card

You can use SD card logging to save signals from Simulink models on an SD card mounted on a Pixhawk Series hardware. The signals from these models are saved as data points in MAT-files. With the data that you log, you can apply fault analysis, search for transient behavior, or analyze sensor data collected over a long period. The data points can be saved in: **Structure**, **Structure with time**, or **Array** format. Simulink supports logging signals on the target hardware from only these three blocks:

- Scope
- To Workspace
- Outport



Note This topic is for UAV Toolbox Support Package for PX4 Autopilots. The SD card logging feature requires you to additionally install Embedded Coder.

Without MAT-file logging on SD card, you can only log and analyze the data on the target hardware through External mode or by sending the signal to a computer (using Serial) and storing it in MATLAB. These mechanisms need an active connection between a computer and the target hardware when logging data. With SD card logging, you can log data without any need to connect the target hardware to a computer. Other advantages of logging data to SD card over other mechanisms are:

- The ability to collect data over a long duration for analysis.
- The ability to store the data in a well-structured MAT-file, including timestamp information.

Note

- The SD card should be formatted with FAT32 format to support the logging from Pixhawk Series processors. Refer to this [link](#) to see the list of supported SD cards for MAT-File logging.

- MAT-File Logging on SD Card does not support the following modes of operation:
 - PIL (Processor in the loop) Mode simulation
 - Monitor and Tune (External mode)
 - Setting the Hardware board as PX4 Host Target or Crazyflie 2.0
-

Before you start to save the signals from Simulink models on an SD card, complete the steps listed in “Prerequisites for Logging Signals” on page 4-4.

- 1** “Configure Model to Log Signals on SD Card” on page 4-5: SD card logging is supported in models containing To Workspace block. You must specify the values for several block parameters.
- 2** “Prepare Model for Simulation and Deployment” on page 4-8
- 3** “Run Model on Target Hardware” on page 4-10: Simulink deploys code and logs signals on the SD card. These signals are saved as MAT-files on the target hardware.
- 4** “Import MAT-Files into MATLAB” on page 4-11: After logging is complete, you can open MAT-files in MATLAB and use them for further analysis.

See Also

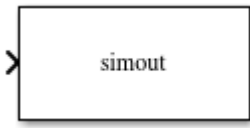
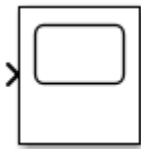

Related Examples

-
-

Prerequisites for Logging Signals

Before logging signals:

- 1 Connect the target hardware to a computer.
- 2 Create or open a Simulink model. To log signals, the model must have at least one of these blocks.

Block Name	Block Icon
To Workspace block	
Scope block	
Outport block	

With UAV Toolbox Support Package for PX4 Autopilots, it is recommended that you use the To Workspace block for logging signals even though the other two blocks are also supported.

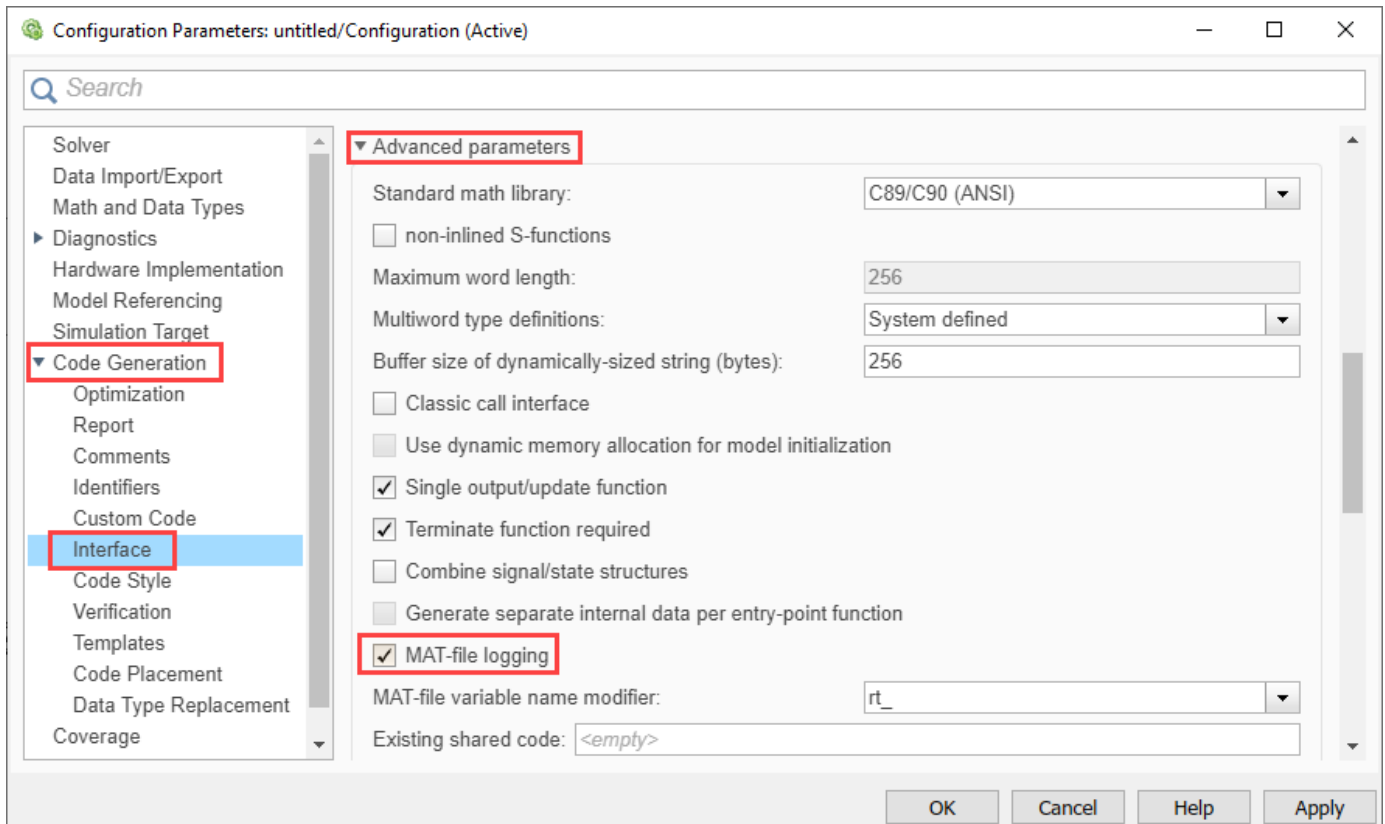
- 3 Save the changes to the Simulink model.

Note Ensure that the file name of the Simulink model does not exceed 22 characters. The filename of the generated MAT file on the SD card consists of this Simulink model's file name and it is also suffixed with the iteration number and run number. The Nuttx (NSH) RTOS, based on which the Pixhawk Firmware is run, has a default file name limit of 32 characters. Therefore, if the filename of the generated MAT-file (modelname_iterationnumber_runnumber) exceeds 32 characters, the MAT-file will not be generated.

Configure Model to Log Signals on SD Card

You need to enable the settings required for MAT-file logging by following these steps:

- 1 In the Simulink window, go to **Modeling > Model Settings** to open the Configuration Parameters dialog box.
- 2 In the **Code Generation** pane, go to **Interface > Advanced Parameters**, and select **MAT-file logging**.



SD card logging is supported in models containing To Workspace block. You must specify the values for several block parameters in the To Workspace block, as described in this section.

To configure a Simulink model to run on the target hardware, perform these steps:

- 1 On the **Modeling** tab, in the **Simulate** section, set the **Stop Time**. The signals are logged for the time period specified in the **Stop Time** parameter. The default value is Inf. Enter time in seconds to log signals for that time period.

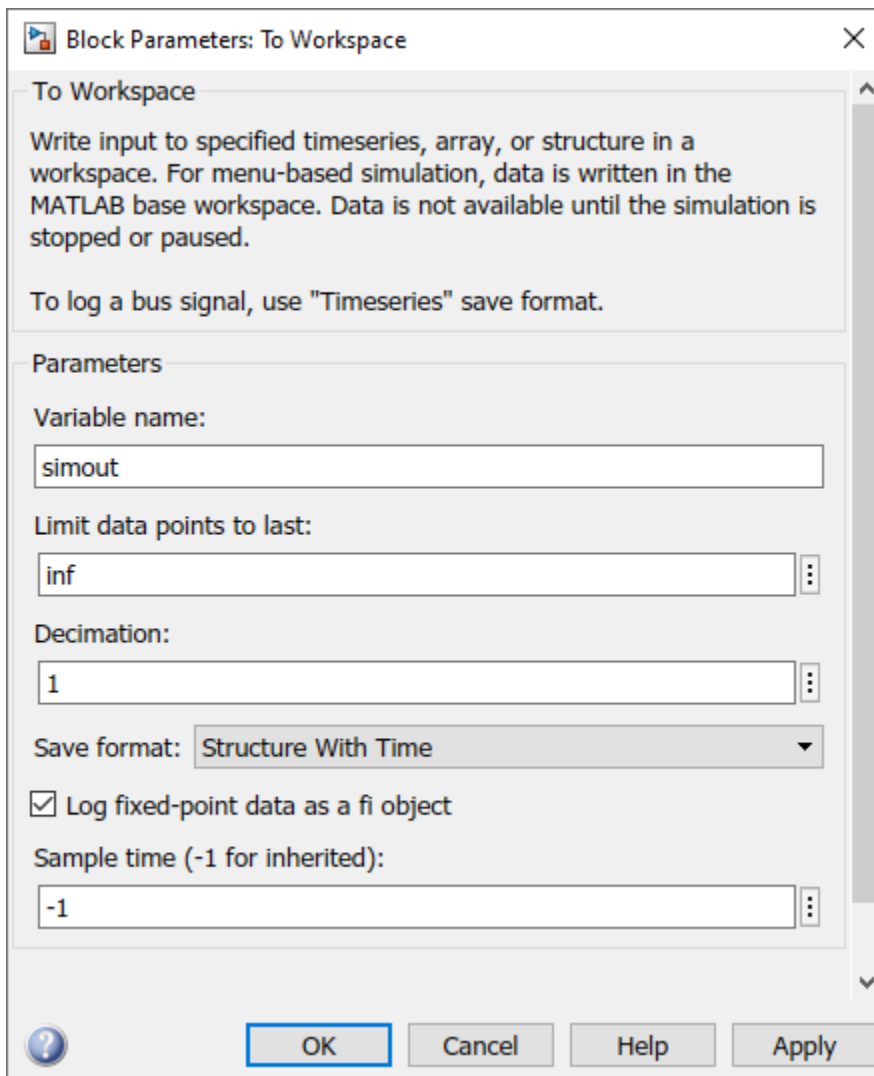
Stop Time

- 2 In the Simulink model, set the parameter values of To Workspace blocks.

Settings for To Workspace Block

Double-click the To Workspace block, and specify these parameters in the **Block Parameter** dialog box.

Parameter	Description
Variable name	Specify a variable name for the logged data.
Limit data points to last	<p>You can use the default value for this parameter.</p> <p>Note Before you simulate the model, you need to run the <code>px4PrepareModelForMATFileLogging()</code> function in MATLAB. This function optimizes the value for Limit data points to last. For details, see “Prepare Model for Simulation and Deployment” on page 4-8.</p>
Decimation	<p>Use this parameter for the block to write data points at every nth sample, where n is the decimation factor. The default decimation, 1, writes data at every time sample.</p> <p>For example, if you specify Decimation as 5, the block writes data at every fifth sample. For example, if the block sample time is 0.1 and Decimation is 5, the data points at 0, 0.5, 1, 1.5, ... seconds are logged. The data points are logged until the Simulation stop time is reached.</p>
Save format	<p>Select a format of the variable to which you save data. SD card logging supports only these three formats: Array, Structure with Time, or Structure.</p> <ul style="list-style-type: none"> • Array: Save data as an array with associated time information. This format does not support variable-size data. • Structure with Time: Save data as a structure with associated time information. • Structure: Save data as a structure. <p>Note Select Save format as Array to use the memory efficiently.</p>
Sample time (-1 for inherited)	Specify an interval at which the block reads data. When you specify this parameter as -1, the sample time is inherited from the driving block.



See Also

Related Examples

-
-

Prepare Model for Simulation and Deployment

After you configure the Simulink model by including the necessary To Workspace blocks, you need to complete the prerequisites for simulation and deployment.

Use `px4PrepareModelForMATFileLogging` to Optimize Memory

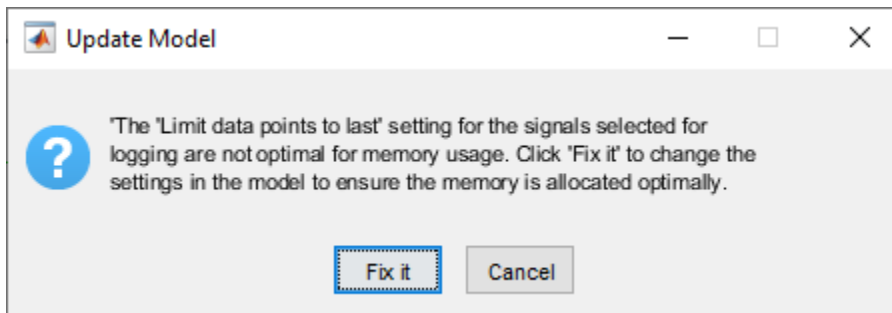
The `px4PrepareModelForMATFileLogging` function needs to be run in the MATLAB command prompt to optimize the memory. The function checks the **Limit data points to last** parameter in the various To Workspace blocks and adjusts its value for memory optimization.

You can use the `px4PrepareModelForMATFileLogging` function by specifying the model name either as character or string:

- `px4PrepareModelForMATFileLogging(bdroot)`
- `px4PrepareModelForMATFileLogging('modelname')`

Here, `modelname` is the file name of the Simulink model.

After you run this function, a dialog box opens requesting your permission to fix the values for memory optimization. Click **Fix it** to optimize the memory.

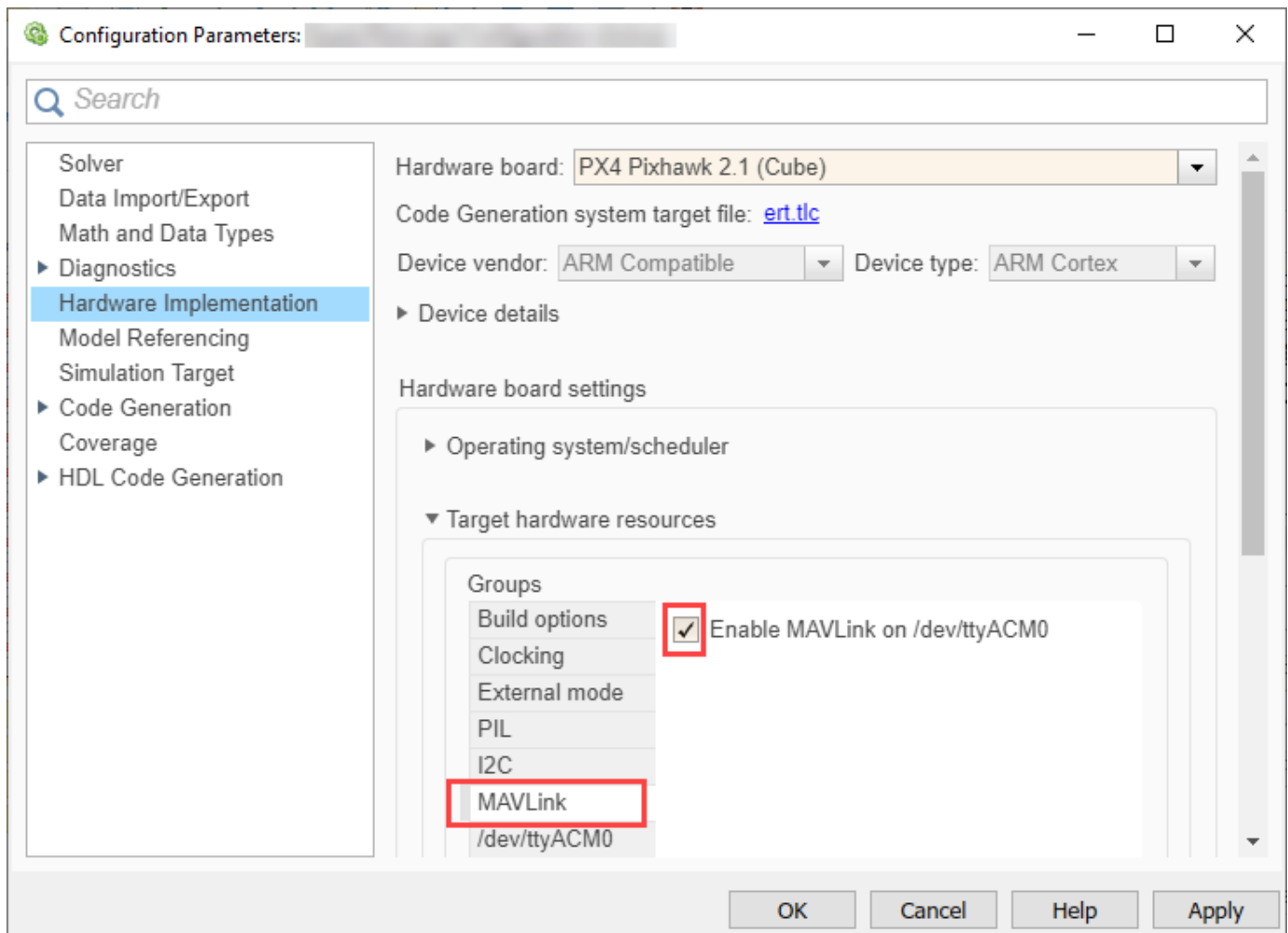


After you click **Fix it**, the value for **Limit data points to last** corresponding to all 'To Workspace' blocks is changed.

It is recommended that you run the function `px4PrepareModelForMATFileLogging` every time you make changes in the model.

Enable MAVLink

MAVLink needs to be enabled to retrieve the log files from SD card inserted on the Pixhawk hardware board (for details about retrieving files, see `getMATFilesFromPixhawk`). To enable MAVLink, open the Configuration Parameters dialog box, go to **Hardware Implementation > Target hardware resources > MAVLink**, and select **Enable MAVLink on /dev/ttyACM0**.

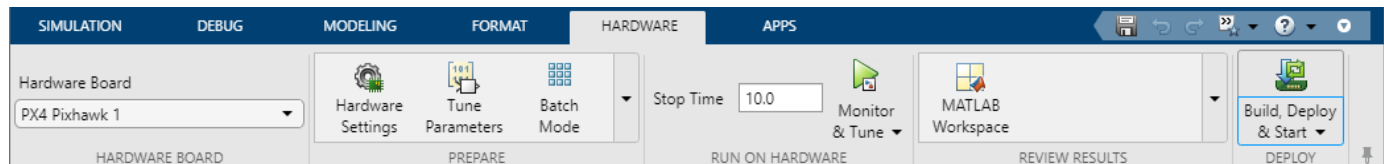


See Also

Run Model on Target Hardware

Simulink deploys code and logs signals on an SD card. These signals are saved as MAT-files on the target hardware.

To deploy the code on the target hardware, in the model window, go to **Hardware** tab. In the **Deploy** section, click the **Build Deploy & Start** button. The lower left corner of the model window displays status while Simulink prepares, downloads, and runs the model on the target hardware. Wait for the logging to stop.



Note After the **Simulation stop time** elapses, the logging of signal stops. However, the model continues to run. For example, if the **Simulation stop time** parameter is specified as **10.0** seconds, the logging stops after 10.0 seconds. However, the model continues to run for an indefinite time. If the **Simulation stop time** parameter is specified as **Inf**, the logging continues until the SD card memory is full, or you remove the SD card from the target hardware.

Import MAT-Files into MATLAB

After logging is complete, you can import MAT-files and open MAT-files in MATLAB, and use them for further analysis. Since the data points are stored in MAT files, you can directly open the files in MATLAB without converting them into any other format.

The files are named as `<modelname>_<runnumber>_<indexnumber>.mat`. The name of your Simulink model is `modelname`. `runnumber` is the number of times the model is run. `runnumber` starts with 1 and is incremented by one for every successive run. `indexnumber` is the MAT-file number in a run. `indexnumber` starts with 1 and is incremented by one for every new file that is created in the same run.

Suppose that the name of the model is `sdcard`. In the first run, Simulink creates `sdcard_1_1.mat` file and starts logging data in this file. After the logging in the first file is completed, Simulink creates `sdcard_1_2.mat` file and continues logging data in this file. Likewise, the logging continues in multiple MAT-files until the **Simulation stop time** is elapsed. If the same model is run again, the new files are named as `sdcard_2_1.mat`, `sdcard_2_2.mat`, and so on.

Note Data loss occurs when:

- The SD card logging mechanism executes in a background task. If the main Simulink algorithm overruns without enough time for the background task, then data loss occurs as the background task gets no time for execution.
 - The signal logging rate is faster than the SD card writing speed. Therefore, it is important that you use an SD card with good write speeds.
-

Use getMATFilesFromPixhawk to Retrieve MAT-files

To retrieve MAT-files from the SD Card, first connect the Pixhawk Series hardware board (with the SD Card still inserted) to the host computer.

Use the `getMATFilesFromPixhawk` function to retrieve the MAT-files. The MAT Files are copied to the current folder in MATLAB.

For details, see `getMATFilesFromPixhawk`.

The `getMATFilesFromPixhawk` function provides two options as name-value pairs:

- `ExtractFilesFromAllRuns` -
 - Set this value to `true` to retrieve all MAT-files corresponding to the model name.
 - Set this value to `false` to retrieve MAT-files corresponding to the model name, but only related to the latest run (only the MAT-files that are updated after you last clicked **Build, Deploy & Start**)
- `DeleteAfterRetrieval` - Set this value to `true` to delete the MAT Files in the SD card after retrieving the same to the host computer.

Note It is recommended that you set this value to `true` to ensure faster retrieval of MAT-Files from SD card in next runs.

Combine MAT-files Using px4MATFilestitcher and Analyze Variables

You need to combine all the MAT-files obtained from the SD card into a single MAT-file by using the `px4MATFilestitcher` function, and then analyze the variables for logged data.

For details, see `px4MATFilestitcher`.

The `px4MATFilestitcher` function combines all the MAT-files starting with the same name into a single file. The order of the stitching is based on the numeric characters found at the end of the file name. The name of the stitched file ends with `__stitched.mat`.

You can use the `px4MATFilestitcher` function at the MATLAB command prompt in these ways:

- To run the stitcher function for all files in the current folder in MATLAB, use `px4MATFilestitcher()` format.
- To run the stitcher function on a specific folder, use `px4MATFilestitcher('folder_name')` format, where `folder_name` is the full path of name of the directory in which the MAT-files are present.

Load the stitched MAT-file. For example, run the below command if the name of the stitched file is `sdcard_1__stitched.mat`:

```
load('sdcard_1__stitched.mat')
```

To view the logged values, double-click the corresponding variable that appears in the **Workspace** window.

Troubleshooting Memory Limitations for MAT-file Logging

If you deploy the Simulink model configured for MAT-File logging by clicking **Build, Deploy & Start** in the Hardware tab of Simulink toolstrip, the Diagnostic Viewer sometimes displays the following RAM overflow errors towards the end of the build process:

```
NuttX/nuttx/mm/libmm.a NuttX/nuttx/sched/libsched.a -lm -lgcc msg/libuorb_msgs.a
src/lib/perf/libperf.a src/lib/parameters/tinybson/libtinybson.a && :
c:/px4_cygwin_5/toolchain/gcc-arm/bin/./lib/gcc/arm-none-eabi/7.2.1/./.././.././arm-none-
eabi/bin/ld.exe: nuttx_px4fmu-v2_default.elf section `.bss' will not fit in region `sram'
c:/px4_cygwin_5/toolchain/gcc-arm/bin/./lib/gcc/arm-none-eabi/7.2.1/./.././.././arm-none-
eabi/bin/ld.exe: region `sram' overflowed by 4786332 bytes
collect2.exe: error: ld returned 1 exit status
ninja: build stopped: subcommand failed.
make[1]: *** [Makefile:154: px4fmu-v2_default] Error 1
make[1]: Leaving directory .../Firmware'
PX4 Cygwin returned an error of 2
gmake: *** [postdownload_preexecute] Error 1
...>echo The make command returned an error of
2
The make command returned an error of 2
...>exit 1
### Build procedure for x aborted due to an error.

Error(s) encountered while building "x"
Component: Simulink | Category: Block diagram error
```

This error occurs if the static memory allocated for all the signals marked for logging exceeds the RAM size for Pixhawk board.

```
...>exit 0
### Build procedure for x aborted due to an error.

The following error occurred during deployment to your hardware board:
Signal logging to SD card takes up 185.2148 KB memory which is more than 50% of available memory
for PX4 Pixhawk 1. This may cause performance issues. Reduce the number of signals to be logged,
or change the data type, or increase the sample time at which the signals are logged.

Component: Simulink | Category: Block diagram error
```

This error occurs if the entire PX4 application along with the PX4 stack and integrated Simulink code takes up more than 50% of the available board RAM. With less than 50% of the RAM available, the chances of a hard fault because of Stack/Heap collision increases.

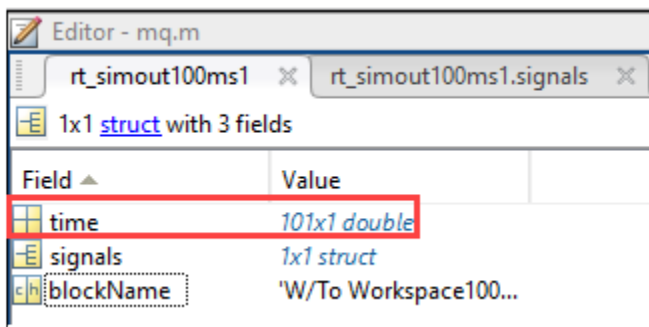
Action

Reduce the memory by following any one of these steps, or a combination of these steps:

- Reduce the number of signals to be logged.
- Change the data type of the signal to be logged to achieve a smaller size for data. By default, the data type of signals in Simulink is `double`, which takes twice the amount of memory compared to a signal with `single` data type. Additionally, the `double` data type takes 8 times the amount of memory compared to an `int8` signal.

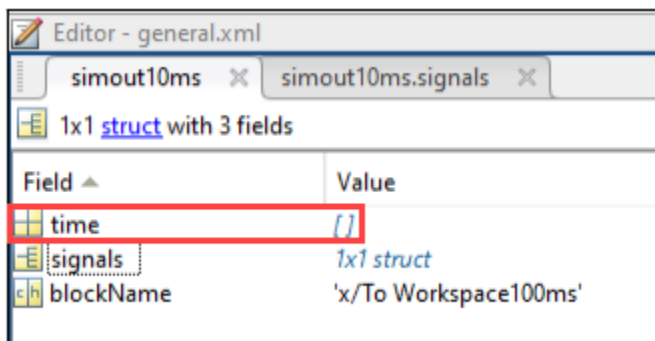
- Increase the sample time at which the signals are logged by using the Rate Transition block. The memory required to log a signal at the rate of 5ms is twice the amount of memory required to log the same signal at 10ms.
- Increase the rate at which the signal is logged by increasing the decimation of the signal. The memory required to log a signal with decimation set to 1 is twice the amount of memory required to log the same signal with its decimation set to 2.
- If there are multiple signals selected for logging at the same Sample time, it is recommended that the *time* variable is logged for only one signal. For the other blocks that support MAT-file logging, choose the **Save Format** as either Structure or Array, instead of Structure with Time.

For each signal with Structure with Time format, the *time* variable is duplicated (the Structure with Time format logs both the signal data and the time stamp).



Field	Value
time	101x1 double
signals	1x1 struct
blockName	'W/To Workspace100...'

However, if you choose the format as either Structure or Array, only the signal values are logged.



Field	Value
time	[]
signals	1x1 struct
blockName	'x/To Workspace100ms'

The timestamp from the first signal with Structure with Time format can be used for other signals that are logged in the same Sample time, so that the same *time* variable is not logged repeatedly.

- To reduce memory overhead, consider combining signals with the same Sample Time using a Mux block, and then feeding the values to To Workspace or Scope blocks. Each signal, which is sent to the To Workspace or Scope block for logging, has a memory overhead apart from the memory required for signal values and timestamp. Combining the signals using a Mux block reduces this overhead.

Continue to modify the model according to the above suggestions, and execute the `px4PrepareModelForMATFileLogging` function until this warning message no longer appears:

```
>> px4PrepareModelForMATFileLogging(bdroot)
Warning: The memory required to log the selected signals to SD card
may cause any of the below memory issues:
  1. Linker RAM overflow issues.
  2. Less RAM available to run the application. This may increase the
chances of a stack/heap collision that will cause a hard fault on the
AutoPilot.
Use any one or a combination of the following methods to reduce the
memory required.
  1. Reduce the number of signals to be logged.
  2. Change the data type of the signal to be logged to a data type of
smaller size.
  3. Increase the sample time at which the signals are logged.
  4. Increase the decimation of the signals which are logged.
  5. Set the Save Format to 'Structure' or 'Array', instead of
'Structure with Time'. For each signal with 'Structure with Time'
format, there will be a duplication of time variable created.
  6. Combine the signals to be logged using a Mux, and feed to a 'To
Workspace', or 'Scope' block to avoid overheads.
Continue to tweak the model according to the above suggestions until
this warning message no longer appears.
Refer the link for more information on how to log Simulink signals
into SD Card.
> In codertarget.pixhawk.registry.staticMemorySizeforSDCard (line 286)
In px4PrepareModelForMATFileLogging (line 7)
```

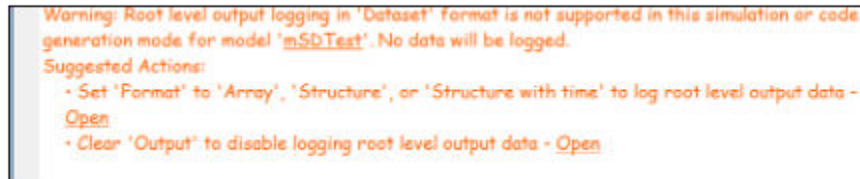
Note It may take multiple iterations to reduce the memory footprint. You need to call the `px4PrepareModelForMATFileLogging` function repeatedly until the warning goes away. This process for optimizing the memory footprint needs to be done every time you make these specific changes to the model:

- Change in the number of signals being logged
- Change to the sample time of signals being logged

See Also

Troubleshooting Dataset Format Usage for MAT-file Logging

If you use Outport blocks for MAT-file logging in the Simulink Model, with the default configuration settings, the below warning may appear:



Action

The above warning appears when Outport blocks are used and the data format is set to **Dataset**. This setting is available as part of Configuration Parameters for the Simulink model (in the Configuration Parameters dialog box, go to **Data Import/Export > Save to workspace or file > Format**).

Use the below methods to get rid of the warning:

- Use the To Workspace block instead of Outport block for logging signals to SD card.
- If you still want to use the Outport blocks for logging signals, avoid using the **Dataset** option for **Format** (in the Configuration Parameters dialog box). Change the option to one of these: **Array**, **Structure** or **Structure with Time**.
- If the signals connected to the Outport blocks need not be logged even though the blocks are present, clear the **Time** and **Output** check boxes in Configuration Parameters dialog box (**Data Import/Export > Save to workspace or file**).

See Also

PX4 SITL Plant Model

Integrate Simulator Plant Model Containing MAVLink Blocks with Flight Controller Running on PX4 Host Target

In this section...

“Introduction” on page 5-2

“Controller Model and Plant Model” on page 5-2

“Prepare Controller Model and Simulator Plant Model” on page 5-3

“Run the Controller and Simulator Plant Model” on page 5-5

The UAV Toolbox Support Package for PX4 Autopilots provides the option to simulate PX4 autopilot algorithms that you develop in Simulink. The workflow is based on PX4 SITL (Software in the Loop). The support package supports simulation of algorithms by generating an executable on the host, **PX4 Host Target**, and the simulator designed in Simulink.

Introduction

PX4 supports running the PX4 flight code to control a computer-modeled vehicle in a simulated world (for more details, refer to this link). In this mode, the sensor data from the simulator is written to PX4 uORB topics. All motors and actuators are blocked, but internal PX4 software is fully operational. As described in the Simulator MAVLink API documentation, the PX4 flight stack and simulator exchange a specific set of MAVLink messages.

The PX4 flight stack sends motor and actuator values to the simulator using the `HIL_ACTUATOR_CONTROLS` MAVLink message. The simulator receives the `HIL_ACTUATOR_CONTROLS` message and sends the sensor, GPS, and quaternion data to the PX4 flight stack using `HIL_SENSOR`, `HIL_GPS` and `HIL_STATE_QUATERNION` MAVLink messages. These messages are exchanged via a TCP/IP connection as described here. The simulator is the TCP server on port 4560 and the PX4 Host Target is the client on the same IP address that connects to the server.

For details about how the PX4 Host Target simulates using the jMAVSIM simulator, see “Deployment and Verification Using PX4 Host Target and jMAVSIM” on page 3-22.

Controller Model and Plant Model

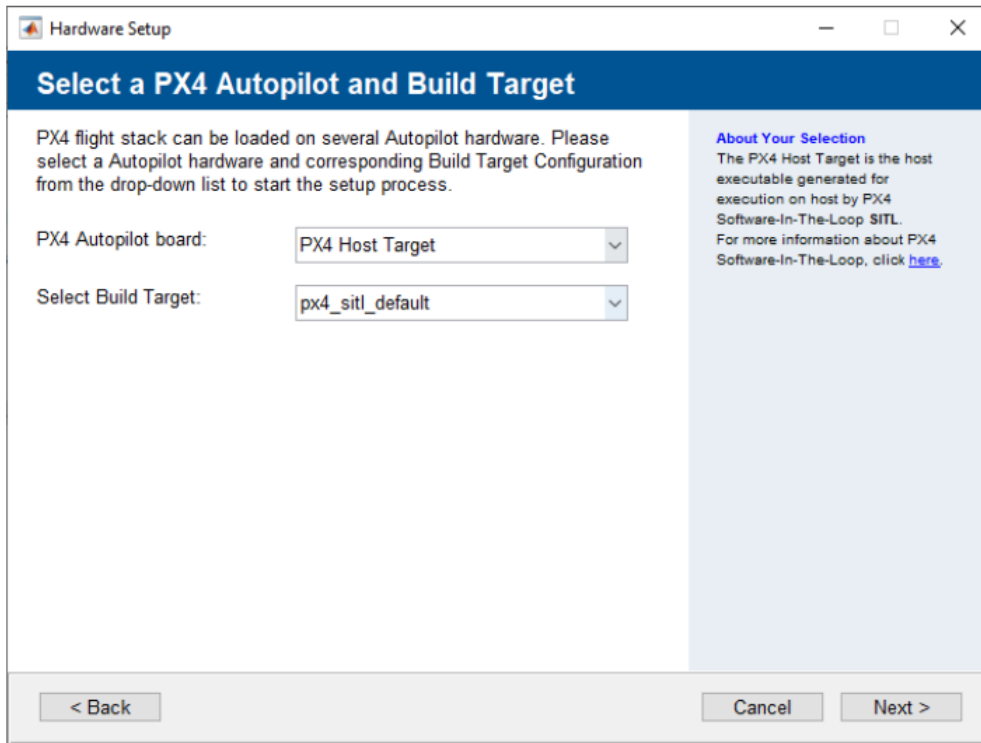
To use the PX4 Host Target with the simulator plant designed in Simulink, follow a two-model approach.

- **Controller model** — In this model, you use the sensors, attitude, and position data to design a controller and estimator and to output the motor and actuator values using the PX4 uORB Read block.
- **Simulator Plant model** — In this model, you design the dynamics of the plant. Use the MAVLink Deserializer block to extract the `HIL_ACTUATOR_CONTROLS` message sent from the controller. The dynamics and sensors designed in the plant model output the sensors and attitude values to the MAVLink Serializer block in the form of `HIL_SENSOR`, `HIL_GPS`, and `HIL_STATE_QUATERNION` messages, and the block in turn sends the serialized data to TCP Send block.

Prepare Controller Model and Simulator Plant Model

Perform these steps, which are a part of the Hardware Setup process in the UAV Toolbox Support Package for PX4 Autopilots, to enable PX4 Host Target.

- 1 In the **Select a PX4 Autopilot and Build Target** Hardware Setup screen, select **PX4 Host Target** as the PX4 Autopilot board. The Build Target file is `px4_sitl_default`.

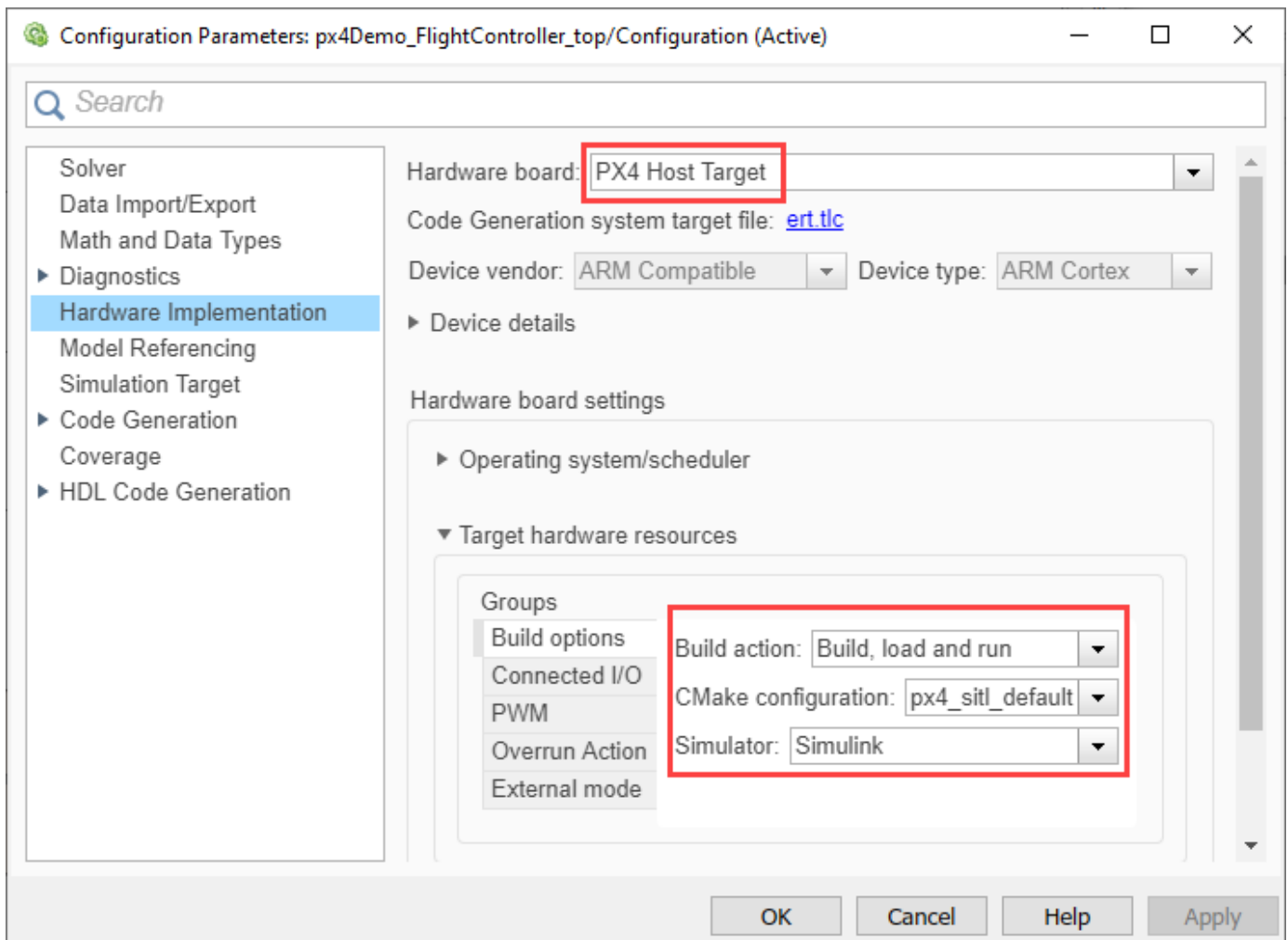


- 2 Proceed with the subsequent steps in the Hardware Setup process to build the firmware and verify that the build was successful.

Prepare PX4 Host Target Controller Model

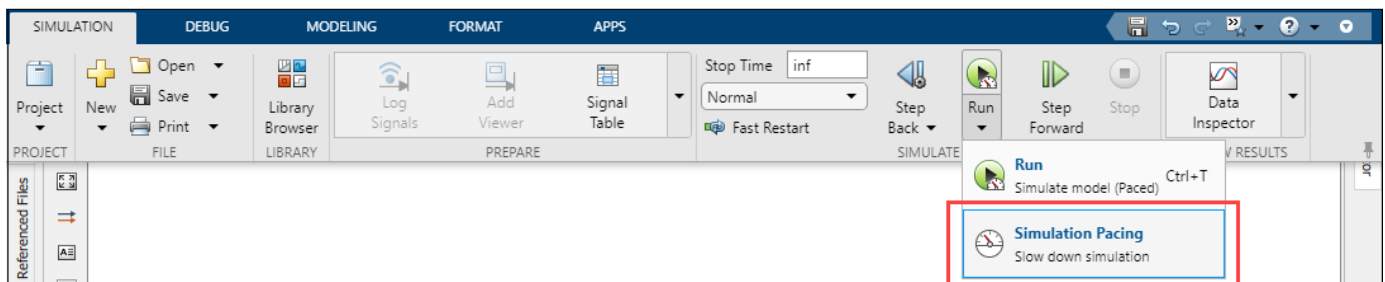
After completing the Hardware setup process, prepare the flight controller algorithm using the Simulink blocks available in the UAV Toolbox Support Package for PX4 Autopilots.

- 1 In the **Modeling** tab, click **Model Settings**.
- 2 In the Configuration Parameters dialog box, choose PX4 Host Target for the Hardware board.
- 3 Go to **Target hardware resources > Build Options**, and choose Build, load and run for the **Build action** parameter.
- 4 Choose Simulink for the **Simulator** parameter.
- 5 Click **Apply** and then **OK**.



Prepare the Simulator Plant Model

- 1 Use the MAVLink Deserializer block to extract the HIL ACTUATOR CONTROLS data sent from the controller model. Design the plant dynamics and send the data using MAVLink Serializer blocks to the controller model.
- 2 In the **Simulation** tab, select **Run > Simulation Pacing**.



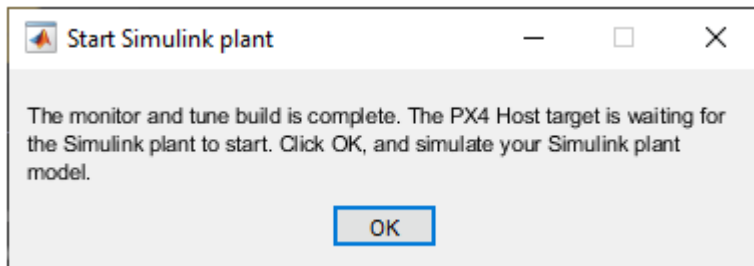
- 3 In the Simulation Pacing Options window, select **Enable pacing to slow down simulation**. For more information, refer to "Simulation Pacing".

Run the Controller and Simulator Plant Model

- 1 In the Controller Simulink model, go to the **Hardware** tab.
- 2 Set a value for the **Simulation stop time** parameter. The default value is 10.0 seconds. To run the model for an indefinite period, enter `inf`.
- 3 Click **Monitor & Tune** to run the model on external mode.

Simulink automatically creates a real-time connection between the model on the PX4 Host Target application and the model on the host computer.

- 4 Wait for the code generation to be completed. Whenever the dialog box appears mentioning that the build is complete, ensure that you click **OK**.



- 5 Open the Simulator Plant model.
- 6 The Controller model is yet to start executing, and you can see the simulation time waiting at 0.000.
- 7 In the Simulator Plant model, click **Run** to compare how both models execute in lockstep simulation.

Tip If you encounter any performance issues while executing the two Simulink models, run the models in two separate MATLAB sessions.

See Also

“Monitor and Tune PX4 Host Target Flight Controller with Simulink-Based Plant Model”

