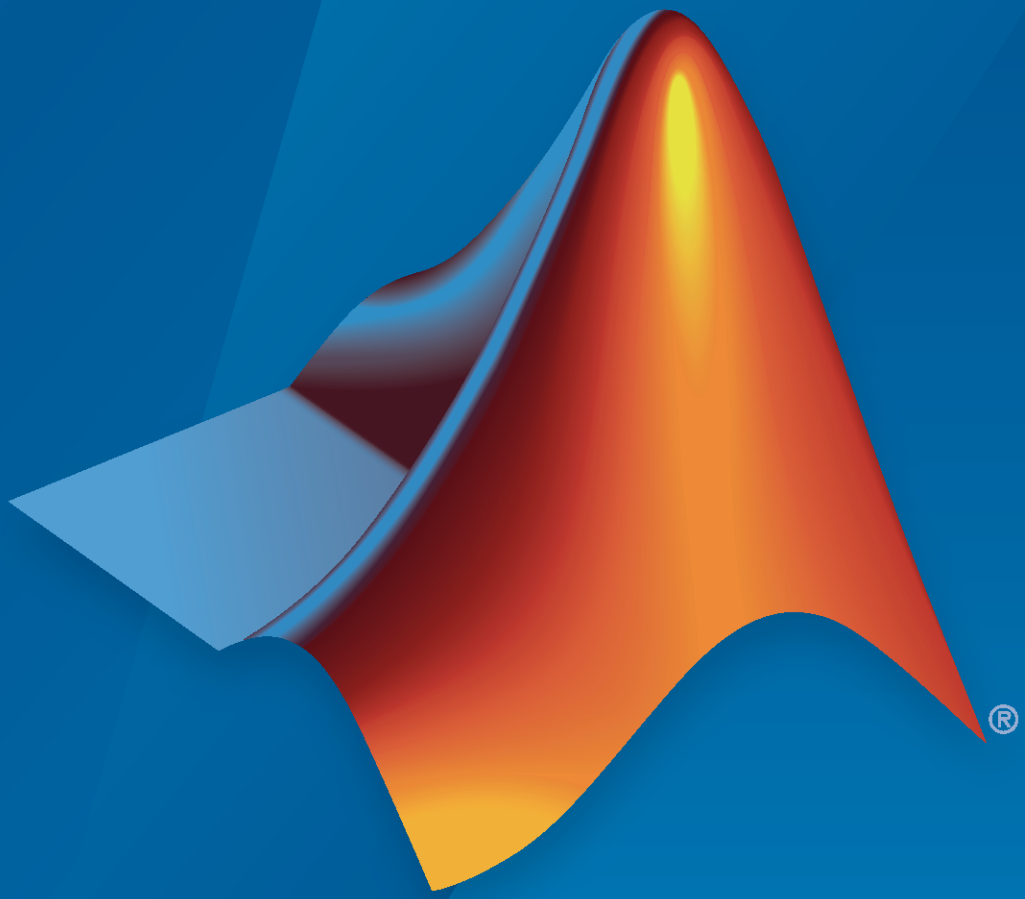
UAV Toolbox

Support Package for PX4® User's Guide



MATLAB® & SIMULINK®

R2025a

MathWorks®

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

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)
September 2021	Online only	Revised for Version 21.2.0 (R2021b)
March 2022	Online only	Revised for Version 22.1.0 (R2022a)
September 2022	Online only	Revised for Version 22.2.0 (R2022b)
March 2023	Online only	Revised for Version 23.1.0 (R2023a)
September 2023	Online only	Revised for Version 23.2.0 (R2023b)
March 2024	Online only	Revised for Version 24.1.0 (R2024a)
September 2024	Online only	Revised for Version 24.2.0 (R2024b)
March 2025	Online only	Revised for Version 25.1.0 (R2025a)

Contents

Setup and Configuration for UAV Toolbox Support Package for PX4 Autopilots

1

2 Connected I/O

3 Connected I/O

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

5 MAT-File Logging on SD Card

6 PX4 SITL Plant Model

Setup and Configuration for UAV Toolbox Support Package for PX4 Autopilots

Operating System Requirements

UAV Toolbox Support Package for PX4 Autopilots can be installed these versions of Windows® and Linux® operating systems:

- Windows
  - Windows 11 (recommended)
  - Windows 10

  Tip To find the Windows version, enter `winver` at the Windows Start Menu.
- Linux:
  - Ubuntu® 20.04 LTS,
  - Ubuntu 22.04 LTS

See Also

“Supported PX4 Autopilots” on page 4-2

Install UAV Toolbox Support Package for PX4 Autopilots in Windows

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 these versions of Windows:

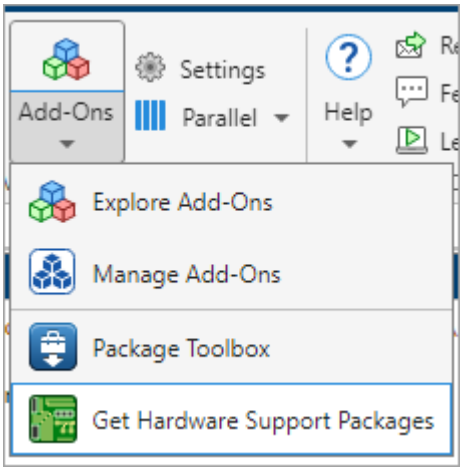
- Windows 11 (recommended)
- Windows 10

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

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 **Resources** section, select **Help > Check for Updates**.

Uninstall Support Package

To uninstall the support package, in the **Add-Ons** panel, click the **Options** button ⋮ next to the installed support package, and then click **Uninstall**.

To open the Add-Ons panel, click the Add-Ons icon on the left sidebar.

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-Ons** panel. To open the Add-Ons panel, click the Add-Ons icon on the left sidebar.

In the Add-Ons panel, click the **Options** ⋮ button next to the installed support package, and click **Setup** to start the hardware setup process.

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.

Additional Information for Hardware Setup

These pages provide additional information for the Hardware Setup process

- “Install Python 3.8.2 on Windows for Firmware Upload” on page 4-83
- “Install Windows Subsystem for Linux (WSL2)” on page 1-28
- “Set Up Windows Subsystem for Linux (WSL2)” on page 1-30
- “Download PX4 Source Code in Windows Subsystem for Linux (WSL2)” on page 1-26
- “Set Up PX4 Tool Chain on Windows Subsystem for Linux” on page 1-32

See Also

“Install UAV Toolbox Support Package for PX4 Autopilots in Linux” on page 1-5

Install UAV Toolbox Support Package for PX4 Autopilots in Linux

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

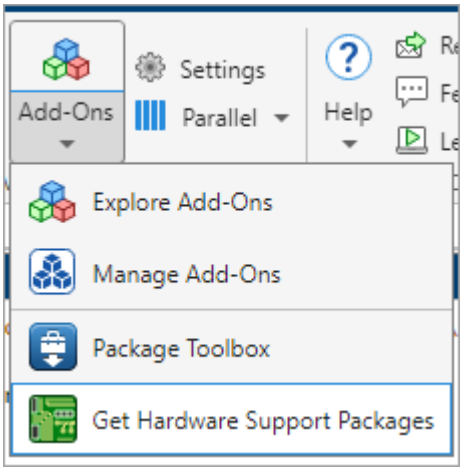
UAV Toolbox Support Package for PX4 Autopilots can be installed these versions of Linux operating systems:

- Ubuntu 20.04 LTS,
- Ubuntu 22.04 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 **Resources** section, select **Help > Check for Updates**.

Uninstall Support Package

To uninstall the support package, in the **Add-Ons** panel, click the **Options** button ⋮ next to the installed support package, and then click **Uninstall**.

To open the Add-Ons panel, click the Add-Ons icon on the left sidebar.

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-Ons** panel. To open the Add-Ons panel, click the Add-Ons icon on the left sidebar.

In the Add-Ons panel, click the **Options** ⋮ button next to the installed support package, and click **Setup** to start the hardware setup process.

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.

Additional Information for Hardware Setup

These pages provide additional information for the Hardware Setup process

- “Download PX4 Source Code in Ubuntu 20.04 and 22.04” on page 1-24
- “Set up PX4 Tool Chain on Ubuntu 20.04 and 22.04” on page 1-22

See Also

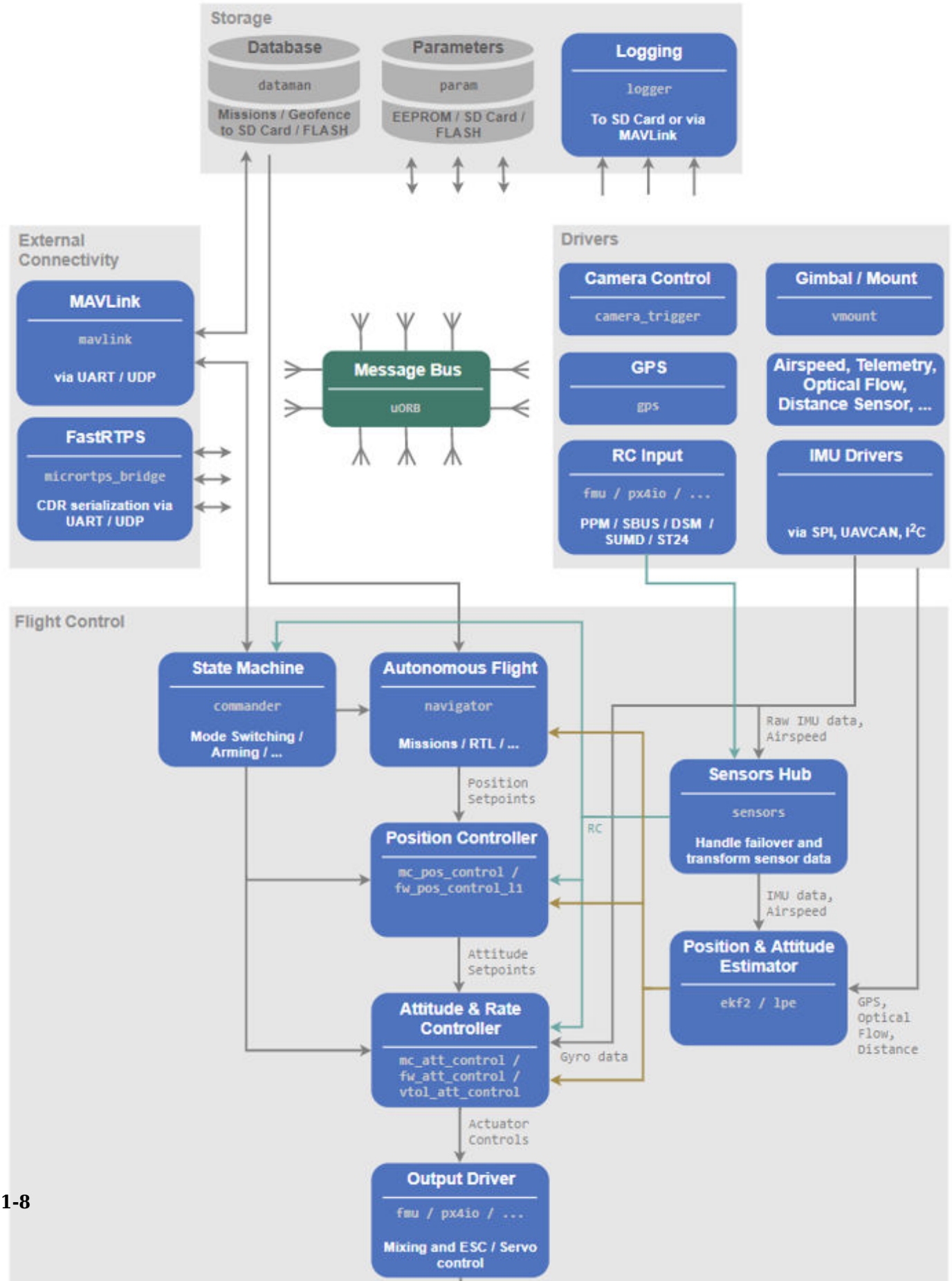
“Install UAV Toolbox Support Package for PX4 Autopilots in Windows” on page 1-3

Integration with General PX4 Architecture

The UAV Toolbox Support Package for PX4 Autopilots enables you to design controllers, estimators, and navigators 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.


Storage
Database
dataman
Missions / Geofence to SD Card / FLASH
Parameters
param
EEPROM / SD Card / FLASH
Logging
logger
To SD Card or via MAVLink
External Connectivity
MAVLink
mavlink
via UART / UDP
FastRTPS
micrortps_bridge
CDR serialization via UART / UDP
Message Bus
uORB
Drivers
Camera Control
camera_trigger
Gimbal / Mount
vmount
GPS
gps
Airspeed, Telemetry, Optical Flow, Distance Sensor, ...
RC Input
fmu / px4io / ...
PPM / SBUS / DSM / SUMD / ST24
IMU Drivers
via SPI, UAVCAN, I2C
Flight Control
State Machine
commander
Mode Switching / Arming / ...
Autonomous Flight
navigator
Missions / RTL / ...
Raw IMU data, Airspeed
Sensors Hub
sensors
Handle failover and transform sensor data
Position Setpoints
RC
Position Controller
mc_pos_control / fw_pos_control_l1
IMU data, Airspeed
Position & Attitude Estimator
ekf2 / lpe
GPS, Optical Flow, Distance
Attitude Setpoints
Attitude & Rate Controller
mc_att_control / fw_att_control / vtol_att_control
Gyro data
Actuator Controls
Output Driver
fmu / px4io / ...
Mixing and ESC / Servo control

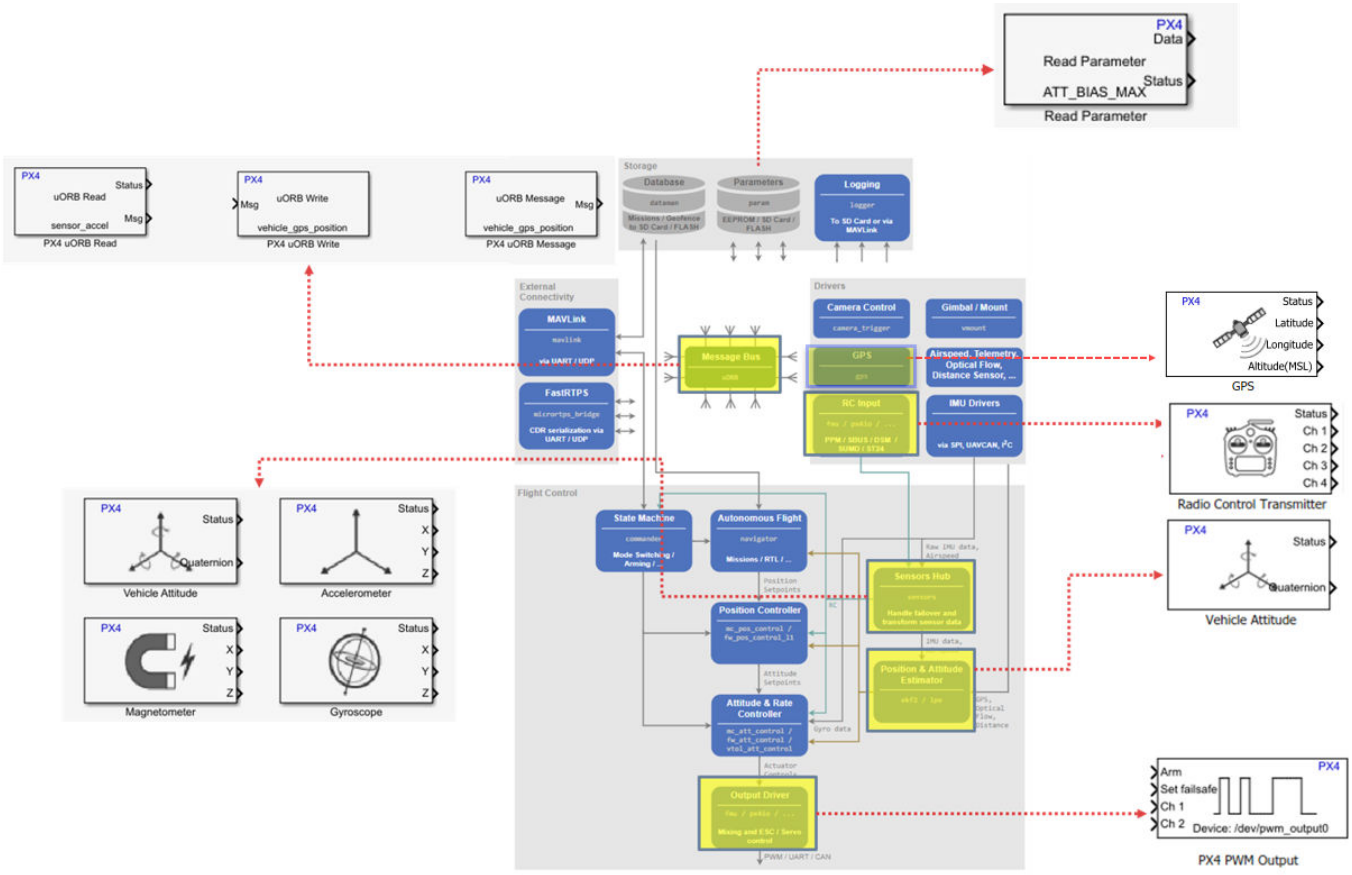
Some of the modules and interfaces of the general PX4 architecture can be integrated with UAV Toolbox Support Package for PX4 Autopilots.

Supported Simulink Blocks That Interface with the PX4 Modules

The UAV Toolbox Support Package for PX4 Autopilots enables you to design controllers, estimators, and navigators 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.

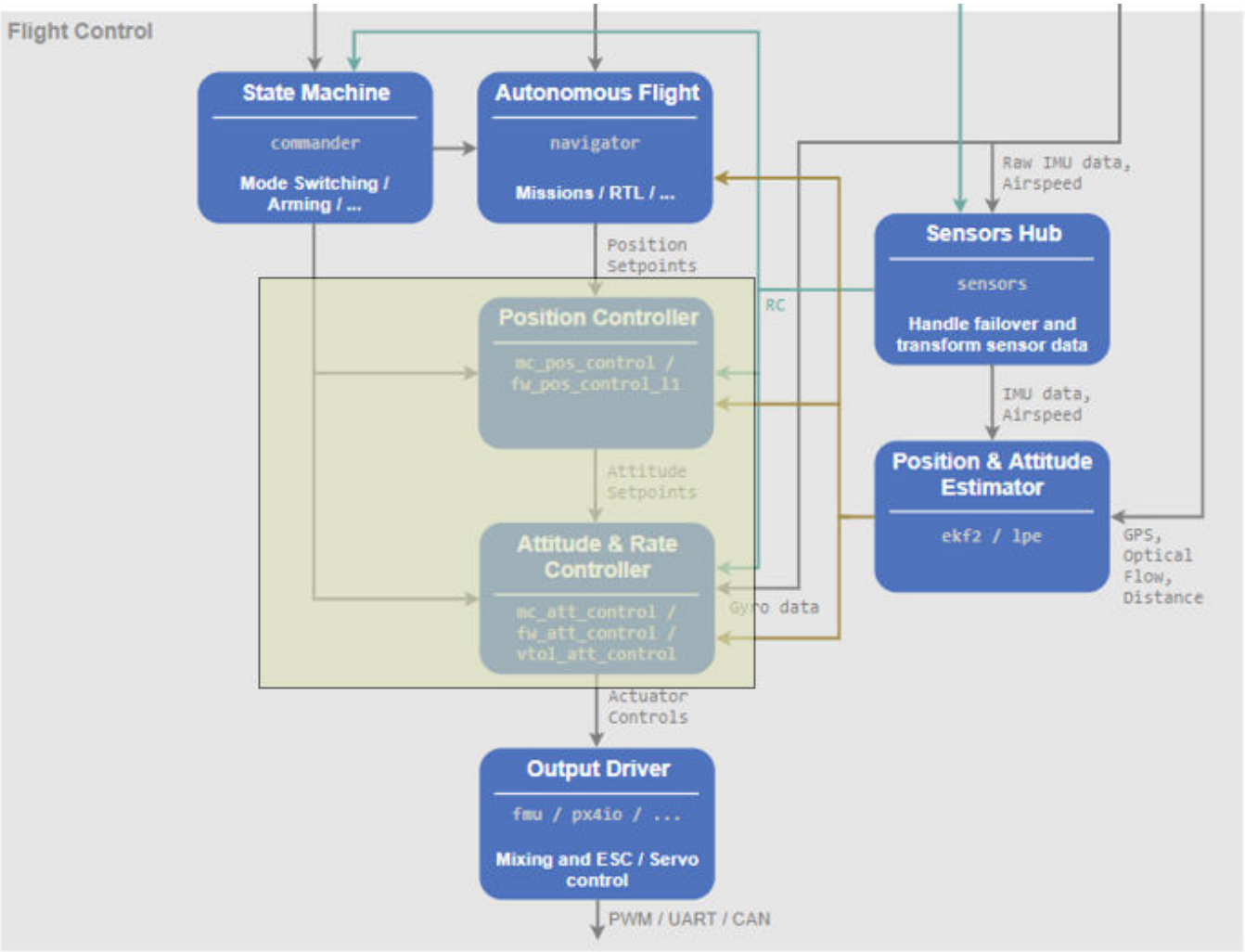
The support package provides interfaces for some of the components in the PX4 architecture by using Simulink blocks. You can use these blocks as input and output for the algorithms in Simulink model.

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
I2C communication	I2C Controller Read I2C Controller Write
CAN communication	PX4 CAN Receive PX4 CAN Transmit
Time stamp	PX4 Timestamp



Supported Modules That Can 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 Controller algorithms that you develop using UAV Toolbox Support Package for PX4 Autopilots.



User-Defined Algorithms in Simulink as a PX4 Module

Embedded Coder® allows you to generate code for the algorithms designed in your Simulink model. The UAV Toolbox Support Package for PX4 Autopilots integrates the generated code as a new module called **px4_simulink_app** in the PX4 Firmware. This new module from Simulink is also added the CMake build configuration for the desired hardware so that the module is compiled and integrated with the PX4 executable when the PX4 Firmware is built for the corresponding CMake build configuration. This module is added to the PX4 startup scripts also so that it is started automatically after the Autopilots boots up. For more information on PX4 startup scripts and how they are managed in the UAV Toolbox Support Package for PX4 Autopilots, see “Load and Start Modules on PX4 Autopilot After Boot-Up” on page 1-40.

See Also

“Custom Startup Script in UAV Toolbox Support Package for PX4 Autopilots” on page 1-12 | “Impact of Disabling MAVLink, Commander, and Navigator Modules” on page 1-14

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 1-38.

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
px4_simulink_app	The Simulink generated code that is integrated in PX4 Firmware as a separate module

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.
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 4-6.

To understand the impact of disabling the modules, see “Impact of Disabling MAVLink, Commander, and Navigator Modules” on page 1-14.

See Also

More About

- “Selecting Startup Script for PX4 Autopilot” on page 1-41
- “Load and Start Modules on PX4 Autopilot After Boot-Up” on page 1-40

Impact of Disabling MAVLink, Commander, and Navigator Modules

Note Refer to this topic only if you are using the custom startup script for the Autopilot as mentioned in the topic "Custom Startup Script in UAV Toolbox Support Package for PX4 Autopilots" on page 1-12. If you are using the PX4 default startup script rcS, then all the modules mentioned here are enabled by default and you can skip this topic.

Impact of Disabling MAVLink

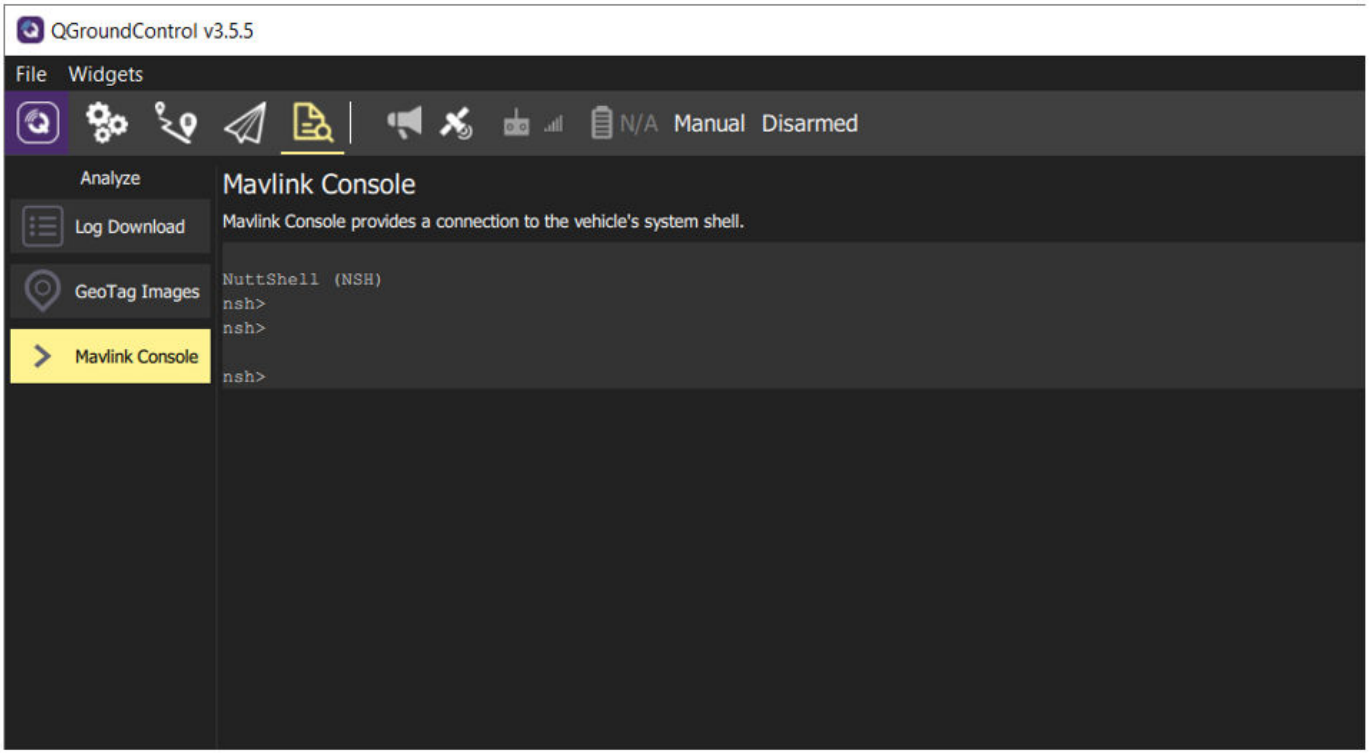
QGroundControl and Mission Planner Communication

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 4-6.

MAVLink Console in QGroundControl

If MAVLink is enabled (as described in "Enabling MAVLink in PX4 Over USB" on page 4-6), 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).

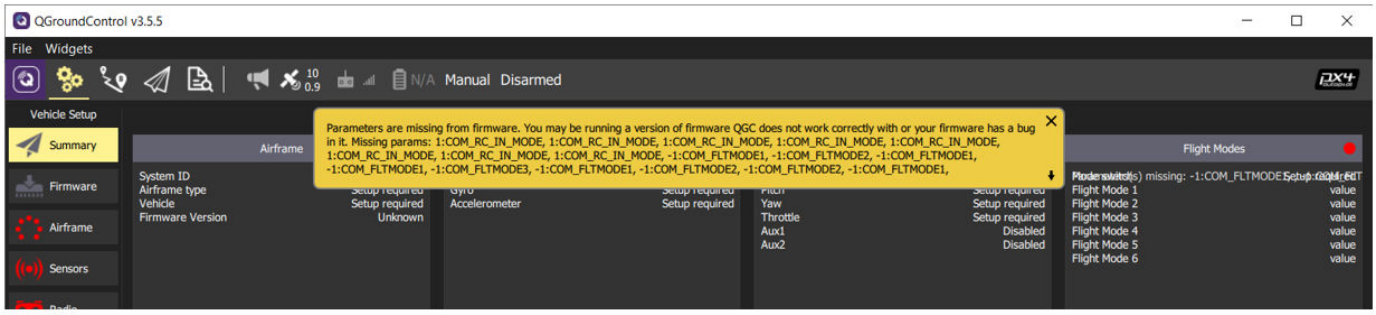
Vehicle Setup in QGroundControl

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`.

Setting Up Cygwin Toolchain and Downloading PX4 Source Code

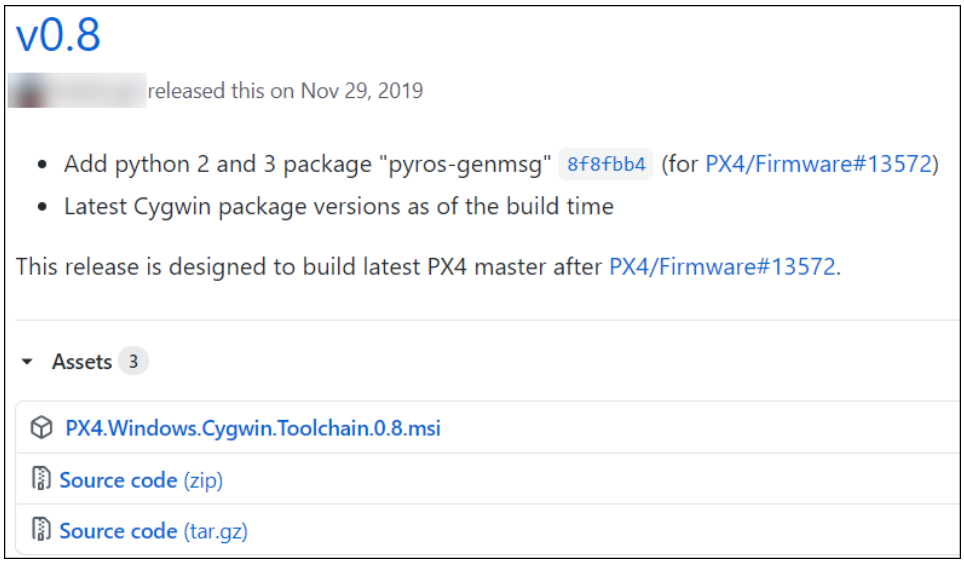
Note This section explains the task to be completed as part of the step—`Setup Cygwin Toolchain and Download PX4 Source Code`—of the Hardware Setup process (using the Hardware Setup screens). Do not perform this as a standalone task.

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

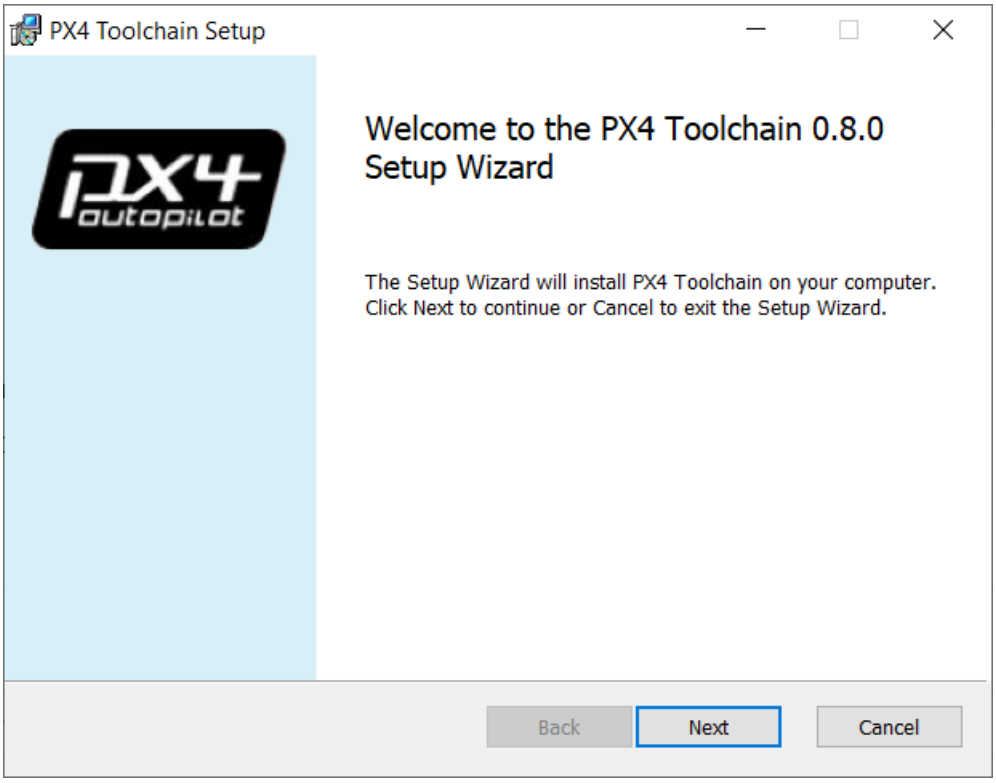
To set up Cygwin toolchain and download the PX4 source code that is used in UAV Toolbox Support Package for PX4 Autopilots, follow these steps:

1. Download version **0.8** of PX4 Cygwin Toolchain MSI Installer, which is compatible with PX4 Firmware v1.14.3, available at this link.

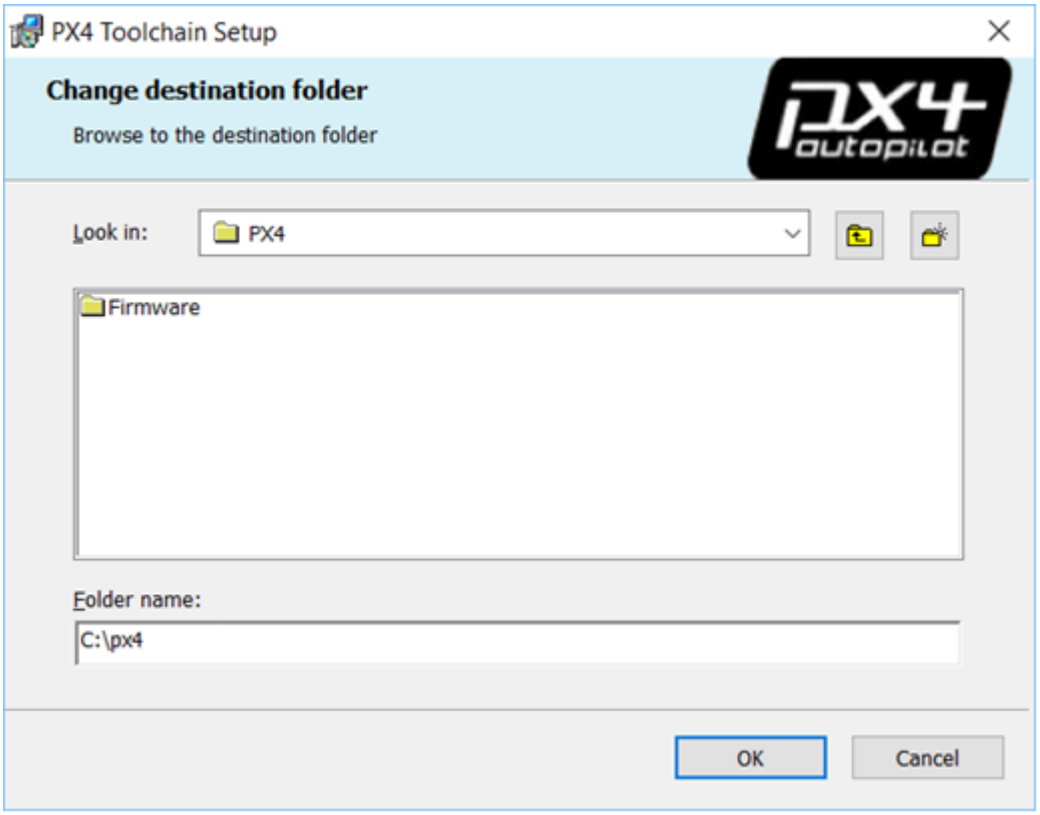
   Note UAV Toolbox Support Package for PX4 Autopilots supports only version **v0.8** of PX4 Windows Cygwin Toolchain MSI Installer, even though a latest version may be available.



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

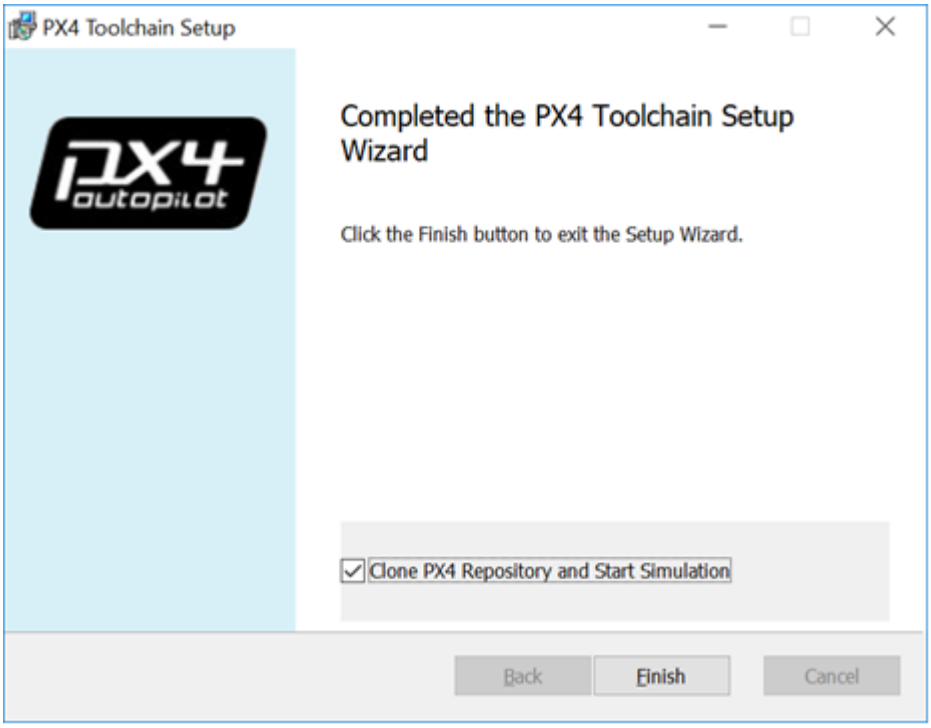


3 Change the installation folder for Cygwin to any local folder (for example, `C:\px4`), and then click **OK**.

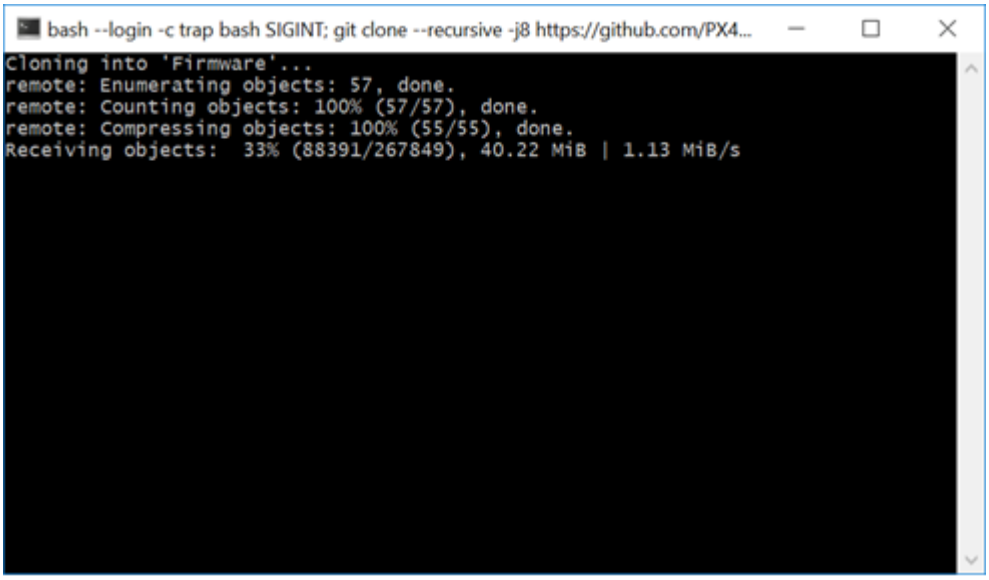


4 At the last step of the PX4 Toolchain Setup wizard, do the following:

- If you do not have the PX4 Source Code (PX4 Autopilot Firmware v1.14.3) downloaded in the host computer, select the option **Clone PX4 repository and Start Simulation**, and then click **Finish**. This option clones the current PX4 main Firmware. When you click Verify Installation in Step 6 below, the firmware is checked out automatically to v1.14.3.
- If the PX4 Source Code (PX4 Autopilot Firmware v1.14.3) 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\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.

Set up PX4 Tool Chain on Ubuntu 20.04 and 22.04

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 step requires an active internet connection.

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.

After you have downloaded PX4 Firmware v1.14.3, follow the below commands to install the PX4 Toolchain on Ubuntu 22.04.

To install and setup the PX4 toolchain:

1. Launch the bash terminal in the Ubuntu 22.04 host computer.
2. Go to the PX4 Firmware v1.14.3 directory that you downloaded.

   For example: `cd /home/username/mypx4/PX4-Autopilot`
3. Navigate to the folder containing the Toolchain setup script.

   ```
   cd Tools/setup
   ```
4. Run the Toolchain setup script.

   ```
   bash ./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 9.3.1, CMake 3.x, Ninja 1.6, Git™, and certain Python packages.
6. Reboot the host computer after the ubuntu.sh script runs successfully.

See Also

More About

- “Download PX4 Source Code in Ubuntu 20.04 and 22.04” on page 1-24

Downloading PX4 Source Code as a Standalone in Windows

Note This section explains how to download PX4 Firmware v1.14 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 1-17.

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 v1.14) 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 1-17.
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 v1.14`

   d. `git submodule update --init --recursive`

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

Download PX4 Source Code in Ubuntu 20.04 and 22.04

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 v1.14.3

Download PX4 Source Code from GitHub

To download the PX4 Sourcecode (PX4 Autopilot Firmware v1.14.3) on the Ubuntu 20.04 or 22.04 host computer:

1. Open the bash terminal on Ubuntu 20.04 or 22.04.
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. Install Git in Ubuntu using this command.

   ```
   sudo apt install git -y
   ```
4. 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/PX4-Autopilot.git --recursive`

   b. `cd PX4-Autopilot`

   c. `git checkout v1.14.3 -f`

   d. `git submodule update --init --recursive`

   The download might take several minutes to complete (depending on the internet speed) and the downloaded firmware can be found in the location, `~/home/mypx4/PX4-Autopilot`.

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

See Also

More About

- “Set up PX4 Tool Chain on Ubuntu 20.04 and 22.04” on page 1-22

Download PX4 Source Code in Windows Subsystem for Linux (WSL2)

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).

UAV Toolbox Support Package for PX4 Autopilots requires PX4 Autopilot Firmware v1.14.3

Download PX4 Source Code from GitHub

To download the PX4 source code (PX4 Autopilot Firmware v1.14.3) on a Windows Subsystem for Linux:

1. Launch the WSL2 shell.
2. Navigate to the home directory by executing the cd ~.

   Warning Cloning the firmware in WSL2 home directory is crucial. If you clone it outside of the WSL file system, then you will encounter slow execution issues and access right / permission errors.

   Note Ensure that your PC is connected to an active internet connection before proceeding with the next step.
3. Download the PX4 source code using git.

   ```
   git clone https://github.com/PX4/PX4-Autopilot.git --recursive
   ```

   The download may take several minutes to complete (depends on the internet speed).
4. After the firmware is cloned, run the below commands one by one in WSL2:
   a. `cd PX4-Autopilot`
   b. `git checkout v1.14.3 -f`
   c. `git submodule update --init --recursive`
5. Run this command in WSL2 to verify that the required version of PX4 Firmware (v1.14.3) is cloned:
   a. `cd PX4-Autopilot`
   b. `git describe --tage`

   If the output of git describe `--tags is v1.14.3`, then you have set up the right firmware version required for the current version of the UAV Toolbox Support Package for PX4 Autopilots.

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

See Also

More About

- “Set Up PX4 Tool Chain on Windows Subsystem for Linux” on page 1-32

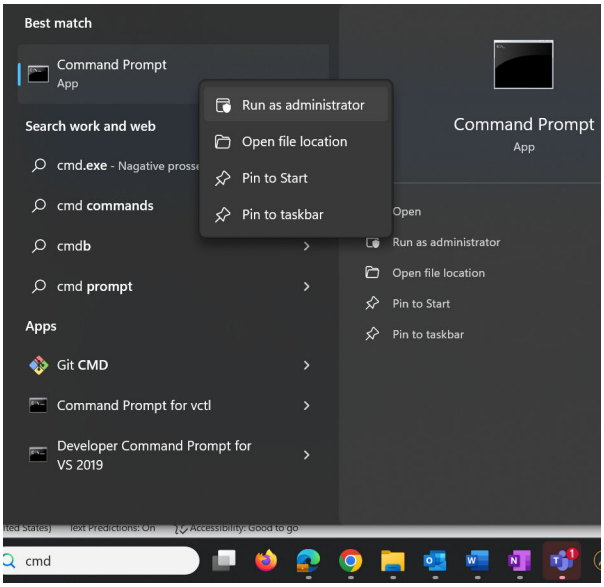
Install Windows Subsystem for Linux (WSL2)

This section explains the task to be completed as part of the step—`Install Windows Subsystem for Linux (WSL2)`—of the Hardware Setup process (using the Hardware Setup screens).

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

To set up a new installation of Windows Subsystem for Linux (WSL2) that is used in UAV Toolbox Support Package for PX4 Autopilots, perform these steps.

1. Ensure that your computer's virtualization feature is enabled in the BIOS. It is usually referred as "Virtualization Technology", "Intel VT-x" or "AMD-V" respectively.
2. Open Windows Command prompt as administrator. This can be done by typing `cmd` in Windows Search, right-clicking on the Command prompt entry and selecting **Run as administrator**.



3. Execute the following command on Windows Command prompt to install WSL2.

```
wsl --install -d Ubuntu-22.04
```

Note To ensure compatibility with the UAV Toolbox Support Package for PX4 Autopilots, you must install Ubuntu `22.04`, as Ubuntu `24.04` is not compatible.

4. After the installation is complete, restart the computer.
5. After computer restarts, launch Windows command prompt again. Run the following command to set Ubuntu `22.04` distribution as default.

```
wsl --set-default Ubuntu-22.04
```

See Also

More About

- “Set Up Windows Subsystem for Linux (WSL2)” on page 1-30
- “Upgrade WSL1 to WSL2” on page 1-33
- “Locate PX4 Firmware Path in WSL 2” on page 1-35

Set Up Windows Subsystem for Linux (WSL2)

Note This step is required only if you have not setup the username and password for your WSL2 installation after you have installed it for the first time.

Configure Ubuntu 22.04 in WSL2

1. After the WSL2 is installed for the first time in your computer by following the instructions from this on page 1-28 documentation, you need to configure the WSL2 with a username and password.
2. Open the Windows Command Prompt again as a regular user (not as an administrator). You can do this by pressing the Windows **Start** key, typing `cmd`, and then pressing **Enter** key.
3. Run the command `wsl` on Windows command prompt to launch the WSL shell.

   This will start the Ubuntu 22.04 installation.
4. After the Ubuntu is installed, WSL will prompt you to setup an username and password for the Ubuntu installation. Ensure to remember these credentials.
5. Set the "Ubuntu" distribution as default. Run the following command to set Ubuntu distribution as default.

Launch WSL2 shell

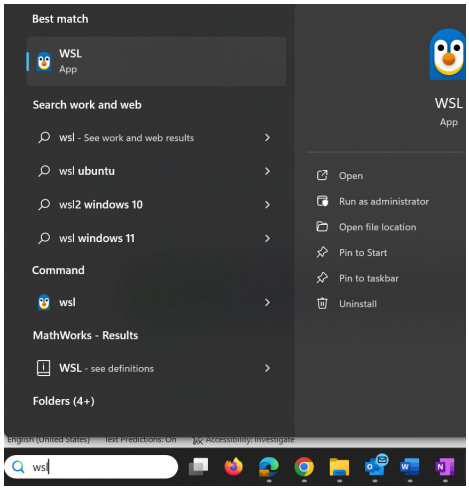
There are multiple ways to launch a WSL2 shell.

Option 1: Using Windows command prompt.

In Windows command prompt, type `wsl` to launch the WSL2 shell.

Option 2: Search wsl or Ubuntu in Windows Search

In Windows Search look for WSL app or Ubuntu app.



See Also

More About

- “Install Windows Subsystem for Linux (WSL2)” on page 1-28
- “Upgrade WSL1 to WSL2” on page 1-33
- “Locate PX4 Firmware Path in WSL 2” on page 1-35

Set Up PX4 Tool Chain on Windows Subsystem for Linux

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

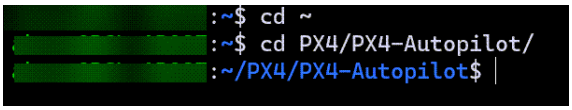
Note This step is only applicable for PX4 Firmware v1.14.3, and not applicable for older PX4 firmware versions such as v1.12.3, v1.10.2, or any other PX4 Firmware versions.

Note This step requires an active internet connection.

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.

After you have downloaded PX4 Firmware v1.14.3, follow the below commands to install the PX4 Toolchain on WSL2.

1. Launch WSL2 shell.
2. Go to the `PX4 Firmware v1.14.3` directory that you downloaded.



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

   ```
   cd Tools/setup
   ```

4. Run the Toolchain setup script.

   ```
   bash ./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 9.3.1, CMake 3.x, Ninja 1.6, Git, and certain Python packages.

6. Restart your WSL environment after the ubuntu.sh script completes successfully.

See Also

More About

- “Download PX4 Source Code in Windows Subsystem for Linux (WSL2)” on page 1-26
- “Locate PX4 Firmware Path in WSL 2” on page 1-35

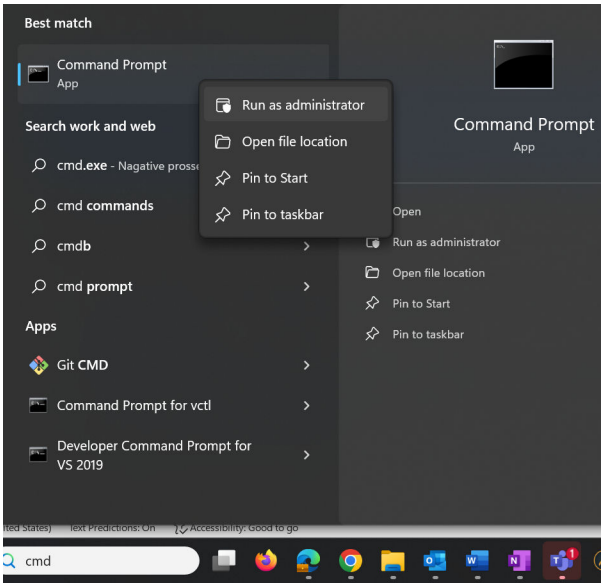
Upgrade WSL1 to WSL2

Note This section explains the task to be completed as part of the step—`Install Windows Subsystem for Linux (WSL2)`—of the Hardware Setup process (using the Hardware Setup screens).

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

The Windows Subsystem for Linux (WSL 2) is required with the UAV Toolbox Support Package for PX4 Autopilots to build the PX4 Firmware. If you have the older version, you can upgrade WSL version by performing these steps.

1. Open Windows Command prompt as administrator. This can be done by typing `cmd` in Windows Search, right-clicking on the Command prompt entry and selecting **Run as administrator**.



2. Check your current WSL version by running the below command in the command prompt.

```
wsl -l -v
```

If the command output mentions the WSL version 1 (as shown below), then you have WSL version 1. In this case follow the next steps to upgrade your WSL.

```
NAME STATE VERSION

* Ubuntu Stopped 1
```

3. Execute the following commands on Windows Command prompt to upgrade your WSL to version 2.

   a. `wsl.exe --update`

b `wsl --set-default-version 2`

c `wsl --set-version Ubuntu 2`

4 Reboot your WSL after the above commands have been executed. Launch WSL shell again. Check that the version of WSL is now version 2 by running the command `wsl -l -v`. The expected output is as shown below

```
NAME STATE VERSION

* Ubuntu Stopped 2
```

See Also

More About

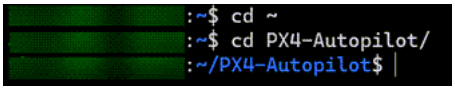
- “Install Windows Subsystem for Linux (WSL2)” on page 1-28
- “Set Up Windows Subsystem for Linux (WSL2)” on page 1-30
- “Locate PX4 Firmware Path in WSL 2” on page 1-35

Locate PX4 Firmware Path in WSL 2

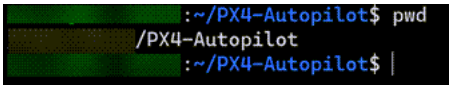
Note This section explains the task to be completed as part of the step—`Validate PX4 Source Code`—of the Hardware Setup process (using the Hardware Setup screens).

After the PX4 Firmware is cloned in WSL as mentioned in “Download PX4 Source Code in Windows Subsystem for Linux (WSL2)” on page 1-26, the path needs to be validated in the Hardware setup process. To know how to locate and specify the Firmware path, follow these steps.

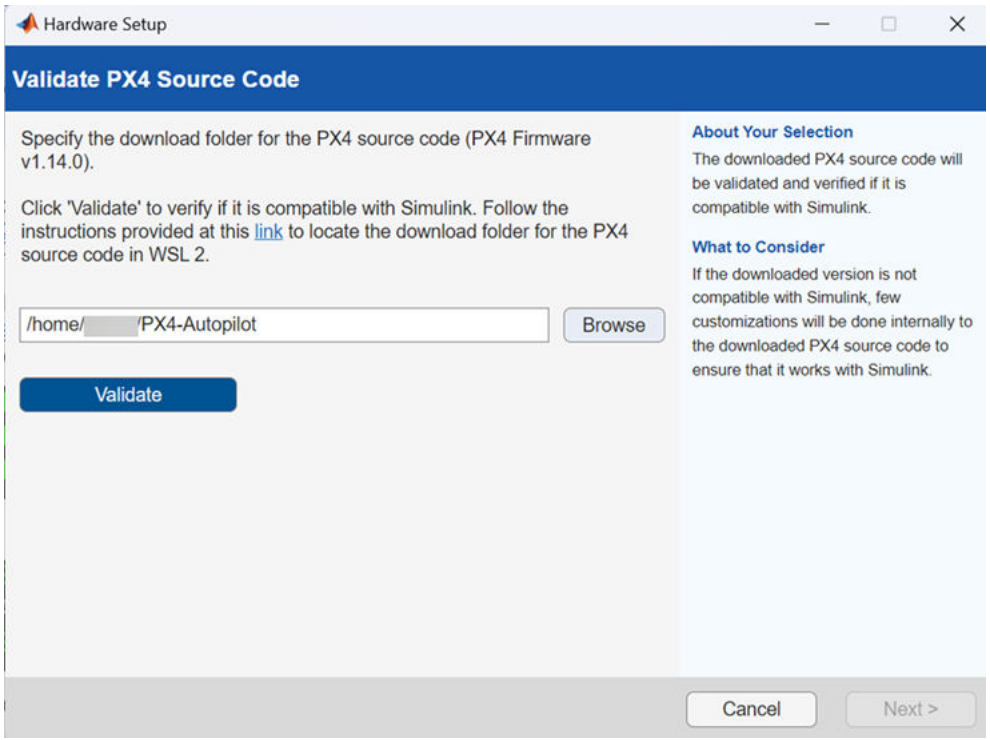
1. Make sure that you have followed the instructions to clone the PX4 Firmware in “Download PX4 Source Code in Windows Subsystem for Linux (WSL2)” on page 1-26. To find the firmware path, launch WSL shell and navigate to the downloaded folder, as shown here.



2. Run the command PWD to print out the path value

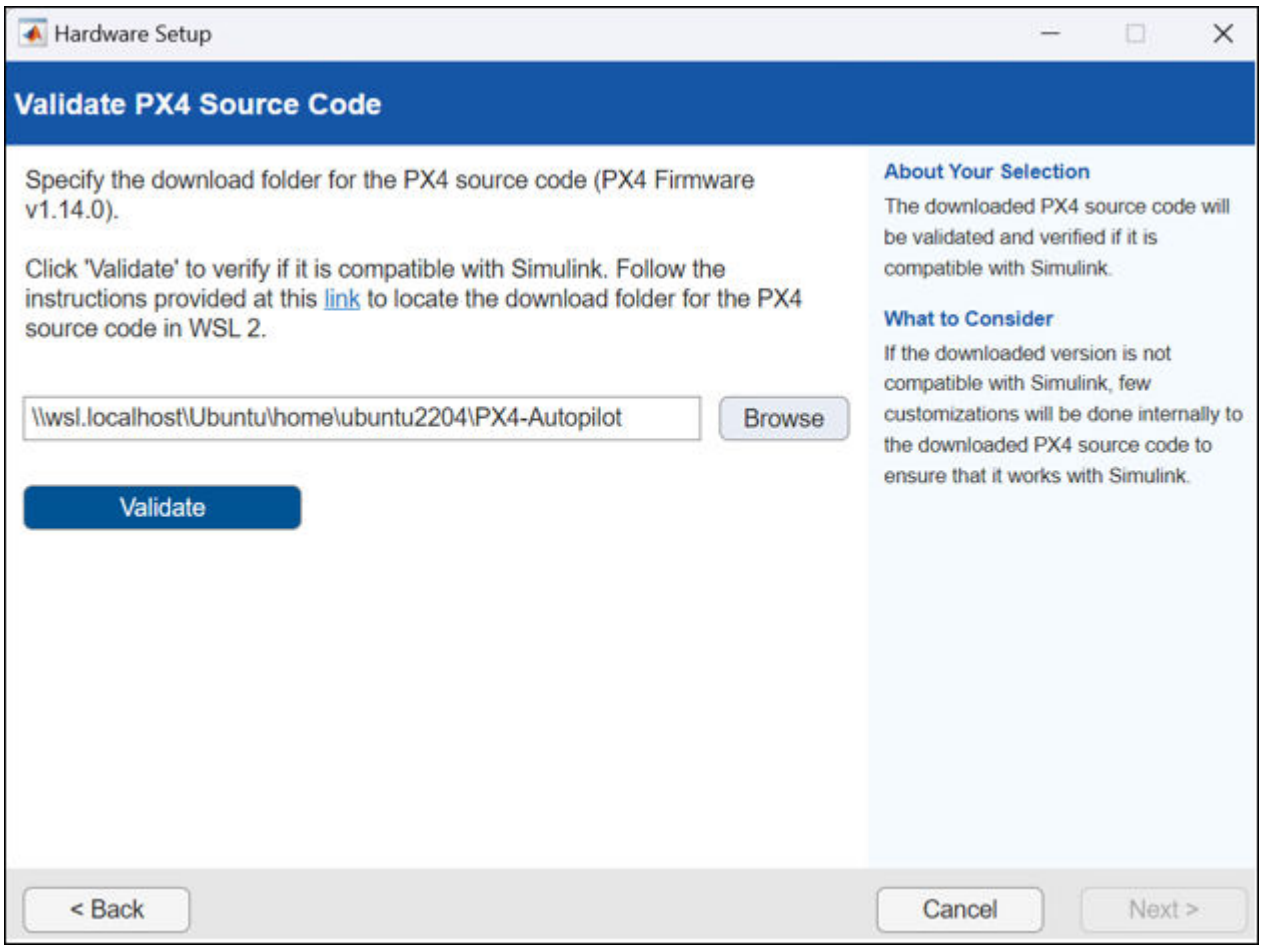


3. Copy the path value and enter it in the edit field in the `Validate PX4 Source code` step of the hardware setup.



4. Click Validate.

5 Alternatively, you can also specify the Windows UNC path for your cloned firmware location in WSL2 as shown below. If your WSL Ubuntu distribution name is "Ubuntu", then `/home` in Unix corresponds to `\\wsl.localhost\Ubuntu\home` in Windows. Thus, the path `/home/<your_WSL_username>/PX4-Autopilot` is equivalent to `\\wsl.localhost\Ubuntu\home\<your_WSL_username>\PX4-Autopilot`



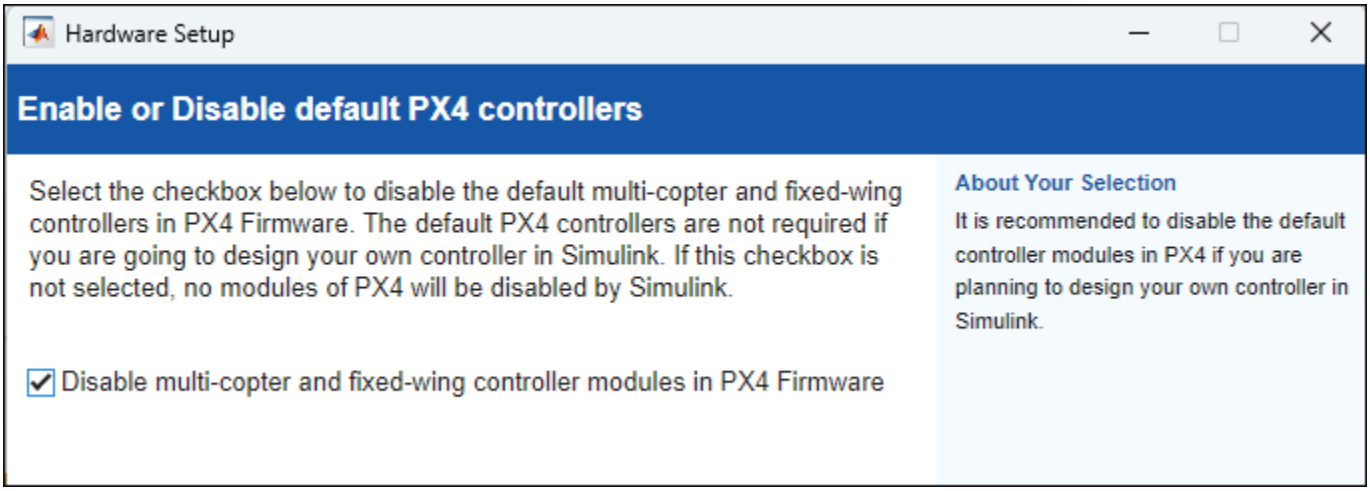
See Also

More About

- “Install Windows Subsystem for Linux (WSL2)” on page 1-28
- “Set Up Windows Subsystem for Linux (WSL2)” on page 1-30
- “Upgrade WSL1 to WSL2” on page 1-33

Enabling or Disabling Default PX4 Controllers

Note This section explains the step—`Enable or Disable default PX4 controllers`—of the Hardware Setup process (using the Hardware Setup screens).



The UAV Toolbox Support Package for PX4 Autopilots provides the option to create custom flight controller algorithms in Simulink. In the **Enable or Disable default PX4 controllers** screen of the Hardware Setup process, select **Disable multicopter and fixed-wing controller modules in PX4 Firmware** to disable the default multicopter and fixed-wing controllers in PX4 Firmware. This setting enables the creation of custom flight controller algorithms in Simulink. If this option is not selected, Simulink does not disable any modules of PX4.

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.

See Also

More About

- “Selecting Startup Script for PX4 Autopilot” on page 1-41

Load and Start Modules on PX4 Autopilot After Boot-Up

The startup sequence of the PX4 modules during boot-up of the hardware is controlled by the startup script contained in the PX4 firmware. The hardware setup process of the UAV Toolbox Support Package for PX4 Autopilots you to select either the default or custom startup script.

The default startup script, rcS, loads the modules and the desired airframe mixers after the Autopilot is booted up. If you select `rcS` as the startup script, all PX4 modules are available for you to use in your Autopilot. The `px4_simulink_app` module which contains the Simulink generated code is also added as an additional module in `rcS` so that it is started after the Autopilot is booted up.

Alternatively, you can use a custom startup script, `rc.txt`, that you place on the SD card of the PX4 Autopilot. The custom start up script allows you to select which PX4 modules to enable in accordance with your controller design requirements. If the `rc.txt` is not present on the SD card, the default startup script is executed on startup.

See Also

More About

- “Selecting Startup Script for PX4 Autopilot” on page 1-41
- “Custom Startup Script in UAV Toolbox Support Package for PX4 Autopilots” on page 1-12

Selecting Startup Script for PX4 Autopilot

Note This section explains the step—`Select System Startup Script in PX4`—of the Hardware Setup process (using the Hardware Setup screens).

The UAV Toolbox Support Package for PX4 Autopilots provides the option to select the startup script for PX4 Autopilot. You can select either the default PX4 Startup Script `rCS` or a custom startup script provided by the support package.

In the **Select System Startup Script in PX4** screen of the Hardware Setup process, select the desired option for the PX4 Autopilot startup script.

- **Use default startup script (rcS)** — Select this option to use the PX4 default startup script `rCS` as the startup script for the Autopilot.

  If you select `Design Flight Controller Algorithm in Simulink` as the algorithm choice in the Select Application in Simulink Hardware setup screen, the default PX4 multicopter controller modules, `mc_pos_control` and `mc_att_control` modules are disabled in the `rCS` startup script.

  If you select `Design Path Follower Algorithm in Simulink`, no modules are disabled in the `rCS`. This selection results in two additional steps, Download QGroundControl and Airframe in QGroundControl as you proceed further with the Hardware Setup screens.

  Note If you use the PX4 default startup script `rcS` as the startup script for the Autopilot and if you also have an SD card connected to the Autopilot, then ensure that the SD card does not contain any custom startup script. If custom startup script exists in the SD card, it will override the PX4 default startup script `rcS`.

- **Use custom startup script (rc.txt)** — Select this option to use the custom startup script `rc.txt` as the startup script for the Autopilot.

  This selection redirects you to the Hardware setup screen. To place the custom startup script in the SD card for the autopilot, you must also complete the steps in “Performing PX4 System Startup from SD Card” on page 1-38.

  To learn more about the modules which are enabled and disabled by the custom startup script, visit “Custom Startup Script in UAV Toolbox Support Package for PX4 Autopilots” on page 1-12

See Also

More About

- “Load and Start Modules on PX4 Autopilot After Boot-Up” on page 1-40

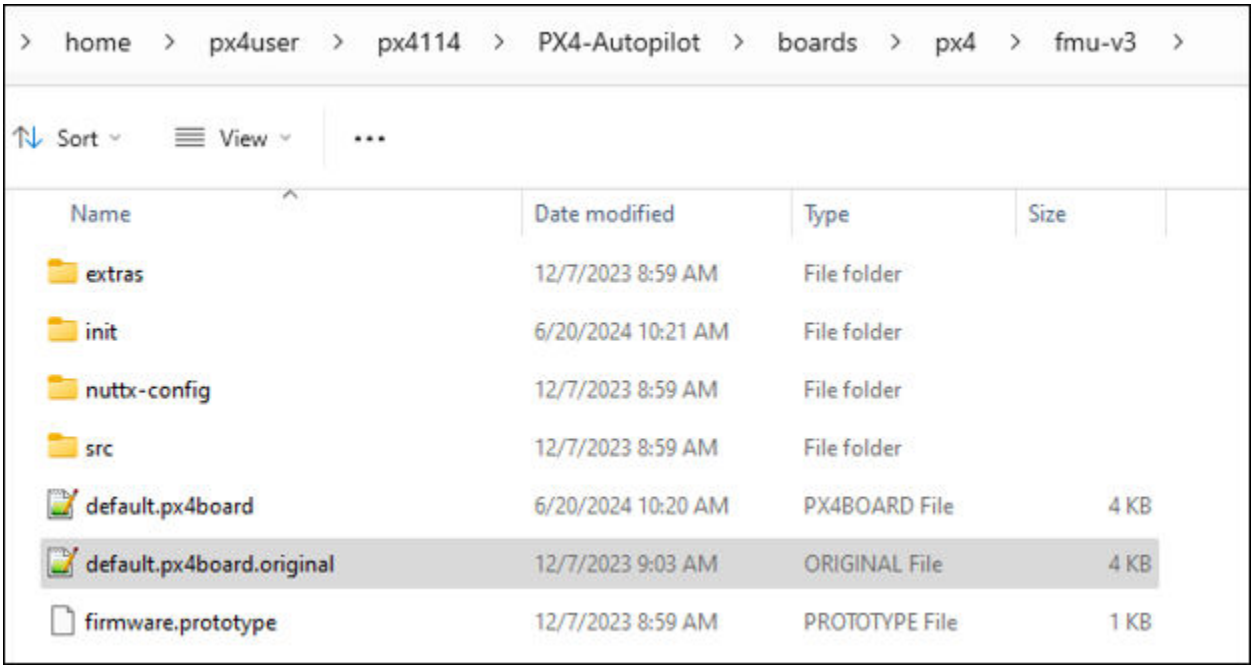
Disabling Modules in PX4board Build Target File

With the PX4 firmware v1.14.3 upgrade, the PX4 source size has increased. If you enable all PX4 modules for the build, the build might fail due to flash overflow. This guide shows you how to disable certain modules based on your use case.

Remove the modules that you don not need for your specific requirement. For example, if you are designing a multicopter, remove the Fixed-wing controller modules, and vice versa. Perform these steps to disable modules in the px4board build target file.

Step1: Open px4board build target file

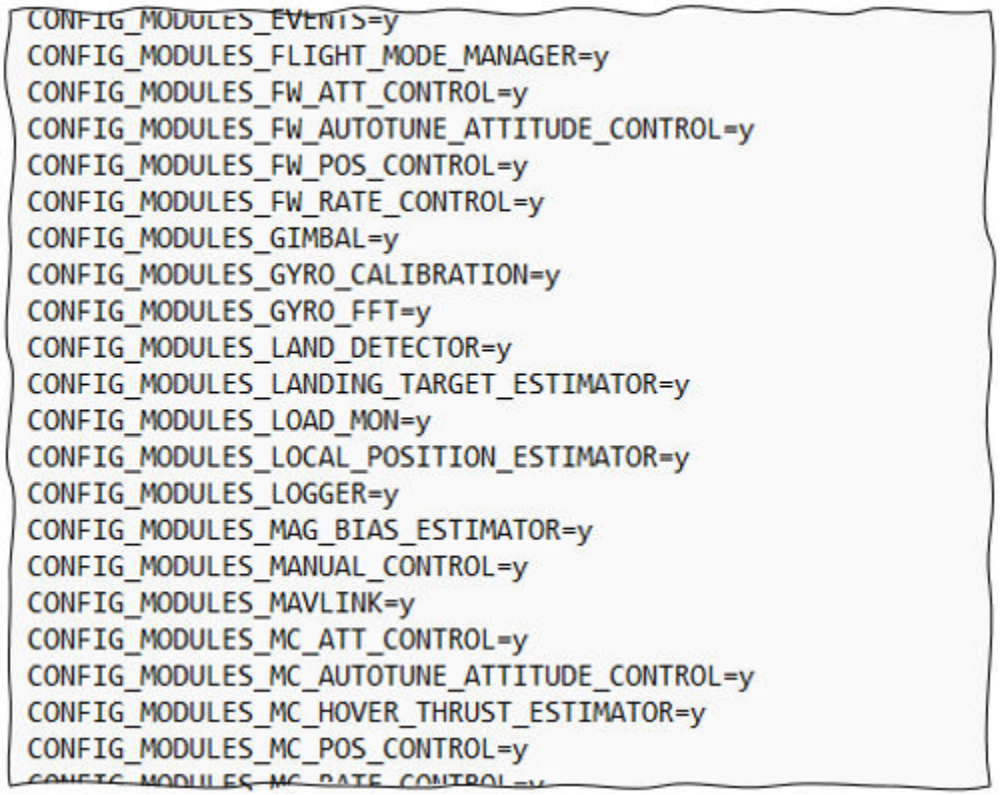
For each PX4 autopilot there is a corresponding px4board build target file. For example, if you are using PX4 Cube Black, the build target is `px4_fmu-v3_default`. Navigate to the corresponding px4board build target file location.



If a default original file exists, open that; otherwise, open the default file.

Step2: Disable control modules from build

All the modules that will be added to the build are listed in the px4board file.



By defining '=y', a module will be enabled for the build. To disable a specific module, define it as '=n'. The following recommendations indicate which modules to disable based on the airframe you want to design.

Multicopter

'CONFIG_MODULES_FW_ATT_CONTROL',

'CONFIG_MODULES_FW_AUTOTUNE_ATTITUDE_CONTROL',

'CONFIG_MODULES_FW_POS_CONTROL',

'CONFIG_MODULES_FW_RATE_CONTROL',

'CONFIG_MODULES_VTOL_ATT_CONTROL',

'CONFIG_MODULES_AIRSPEED_SELECTOR'

Fixed-wing

'CONFIG_MODULES_MC_HOVER_THRUST_ESTIMATOR',

'CONFIG_MODULES_MC_RATE_CONTROL',

'CONFIG_MODULES_VTOL_ATT_CONTROL',

'CONFIG_MODULES_MC_AUTOTUNE_ATTITUDE_CONTROL'

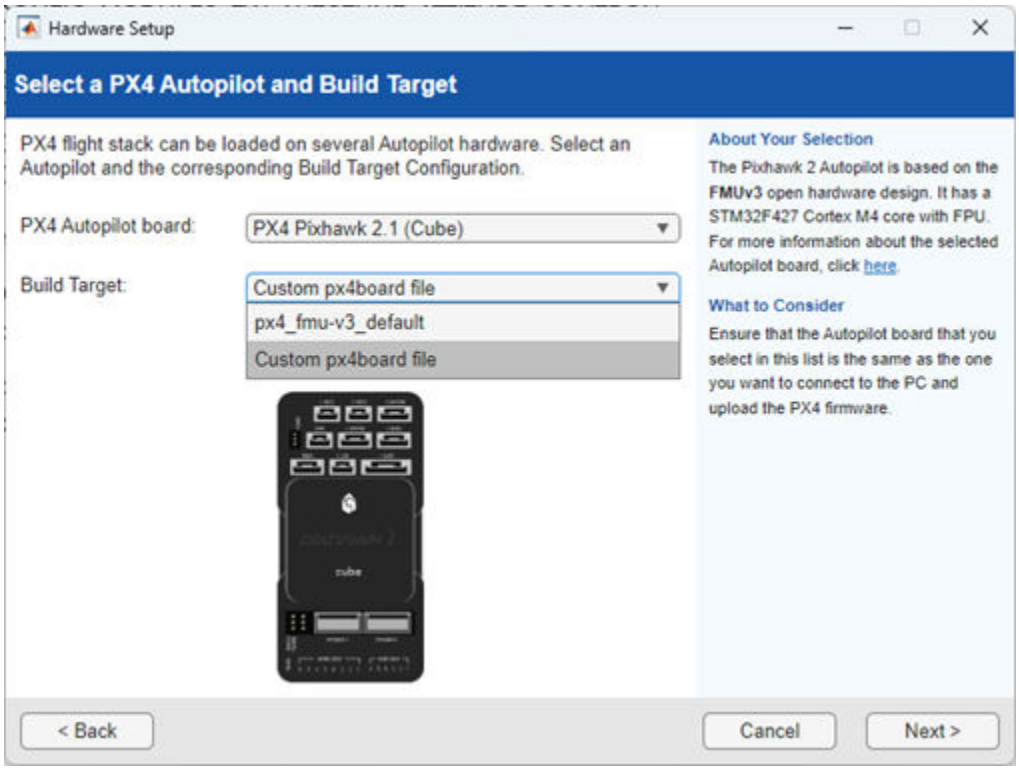
VTOL

```
'CONFIG_MODULES_FW_ATT_CONTROL',

'CONFIG_MODULES_FW_AUTOTUNE_ATTITUDE_CONTROL',

'CONFIG_MODULES_FW_POS_CONTROL',

'CONFIG_MODULES_MC_HOVER_THRUST_ESTIMATOR',

'CONFIG_MODULES_MC_RATE_CONTROL',

'CONFIG_MODULES_VTOL_ATT_CONTROL',

'CONFIG_MODULES_MC_AUTOTUNE_ATTITUDE_CONTROL'
```

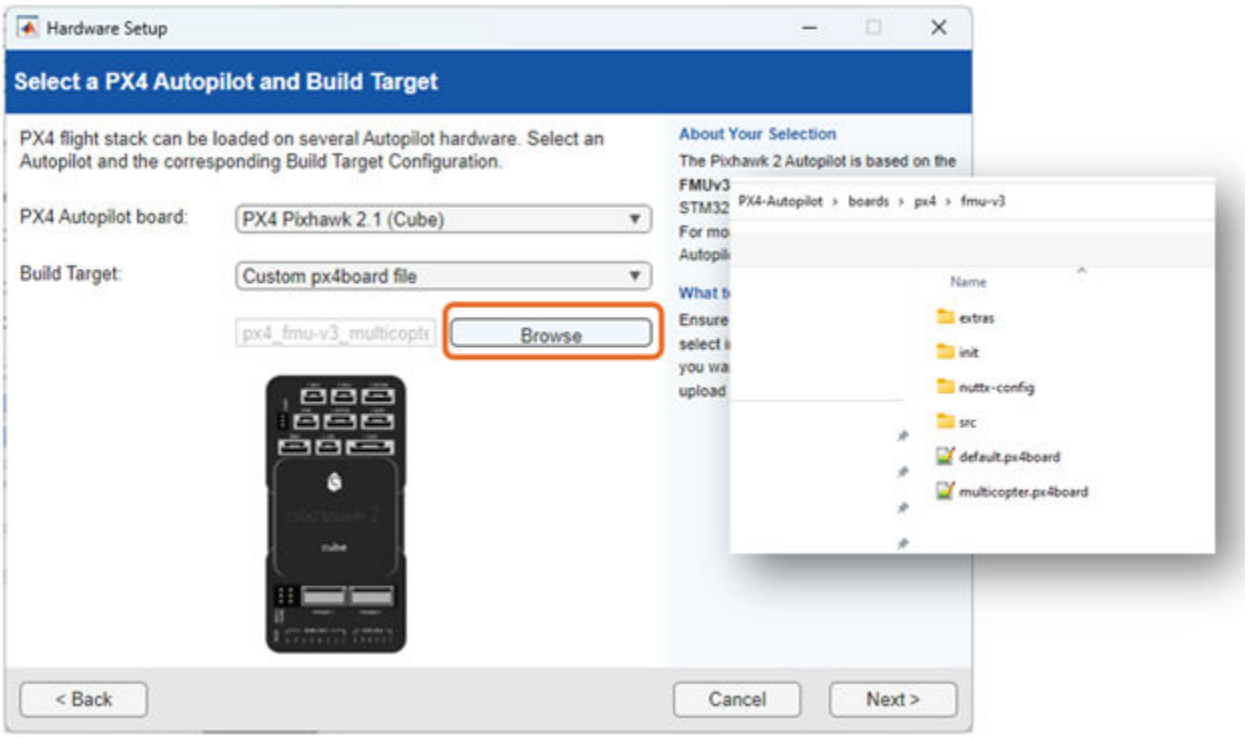
Step3: Save build target file

Once the necessary changes are made, save the build target file. Consider saving the file with a new name, such as `multicopter.px4board`.

In the setup screen, choose the newly saved px4board build target file using the `custom px4board file` option.

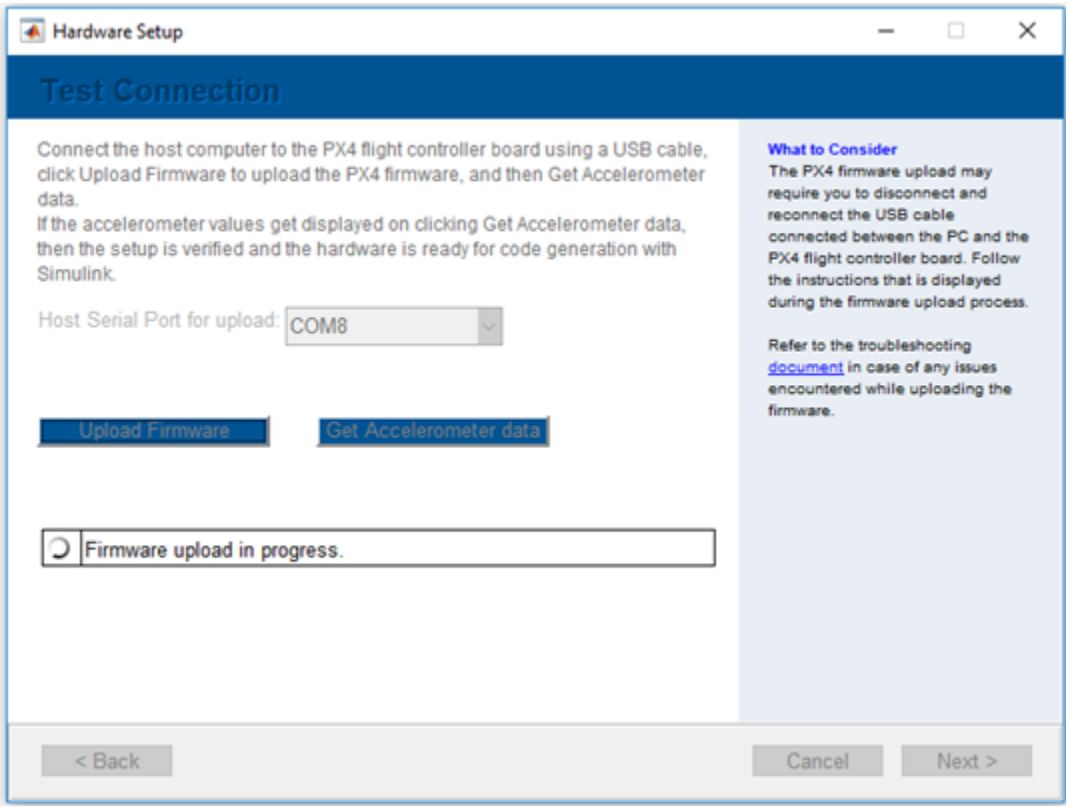


Click **Browse** and navigate to your px4board file location.


Hardware Setup
Select a PX4 Autopilot and Build Target
PX4 flight stack can be loaded on several Autopilot hardware. Select an Autopilot and the corresponding Build Target Configuration.
PX4 Autopilot board:
PX4 Pixhawk 2.1 (Cube)
Build Target:
Custom px4board file
px4_fmu-v3_multicopte
Browse
About Your Selection
The Pixhawk 2 Autopilot is based on the
FMUv3
STM32
For mo
Autopil
What t
Ensure
select i
you wa
upload
PX4-Autopilot › boards › px4 › fmu-v3
Name
extras
init
nuttx-config
src
default.px4board
multicopter.px4board
< Back
Cancel
Next >

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
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.	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.
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). You can verify if this is the reason for the issue, if the following line appears at the MATLAB command prompt: `If the board does not respond, unplug and replug the USB connector.`	Disconnect and reconnect the Pixhawk Series flight controller to the host computer. 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. **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.

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)
```

See Also

More About

- “Troubleshooting PX4 Firmware Build Failure Due to Flash Memory Overflow on the Hardware” on page 4-31

Troubleshooting Connected I/O

Description

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

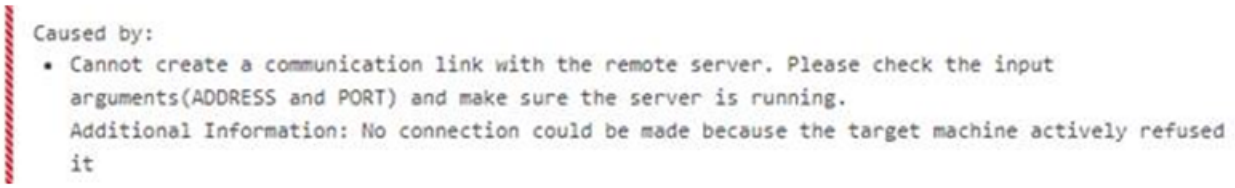
Action

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*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.



Action

To resolve the issue:

- Wait till PX4 host target and jMAVSim is launched properly.
- Start connected I/O simulation again. On the **Hardware** tab, in the **Mode** section, select **Connected IO** and then click **Run with IO**.

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 Using Connected I/O

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

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 only when you run the model in Connected I/O mode.

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 2-2
“Connected I/O in Model-Based Design” on page 2-3
“How Connected I/O Differs from Monitor & Tune (External Mode)” on page 2-4
“When to Use Connected I/O” on page 2-5
“Run Simulink Model in Connected I/O” on page 2-5

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

All PX4 boards and blocks support Connected I/O. For more information, see:

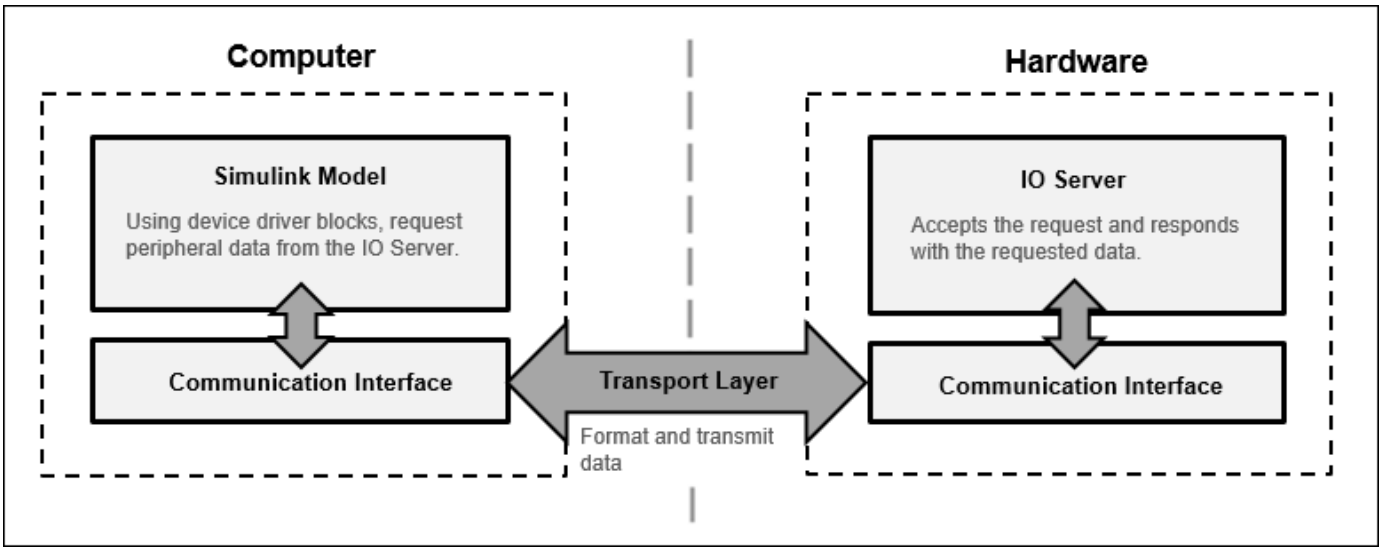
- “Supported PX4 Autopilots” on page 4-2
- Supported PX4 Blocks

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.



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

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 in Connected I/O mode. 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.

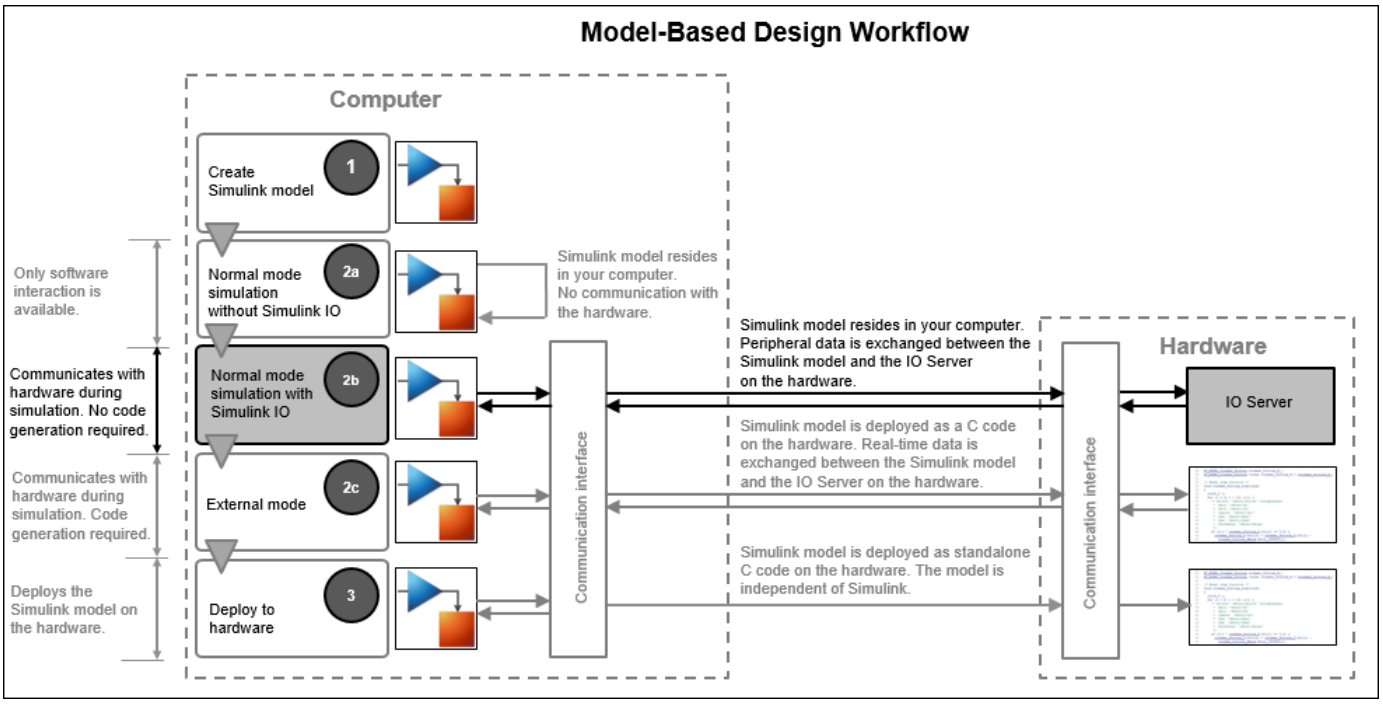
Connected I/O in Model-Based Design

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. Connected I/O simulation 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. Simulation without Connected I/O: There is no hardware interaction and no code generation.
   b. Simulation with Connected I/O: The model communicates with the hardware. There is no code generation.
   c. Monitor & Tune (External mode): The model is deployed on the hardware and generates code.
3. Deploy the model to the hardware.



How Connected I/O Differs from Monitor & Tune (External Mode)

Connected I/O and Monitor & Tune (External Mode) both enable you to communicate with the hardware during simulation. However, you use Connected I/O and Monitor & Tune 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 Monitor & Tune.	You can obtain only non real-time data with Connected I/O.
Time required to start simulation	1-2 minutes	Few seconds

Action	External Mode	Connected I/O
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

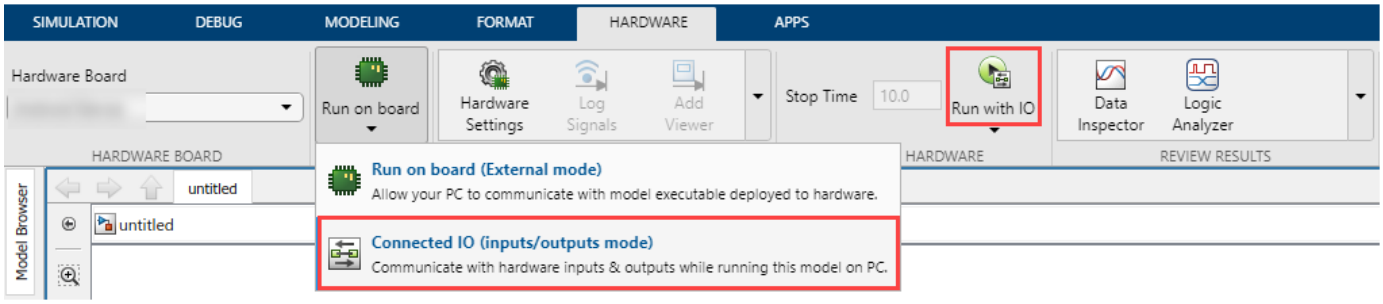
Connected I/O must be used primarily as a Sensor/Actuator block characterization which enables a fast way to get started with a block to understand its outputs/inputs. In Connected I/O every block needs to communicate to the PX4 boards and thus there is a specific minimum time required for a block execution. 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.

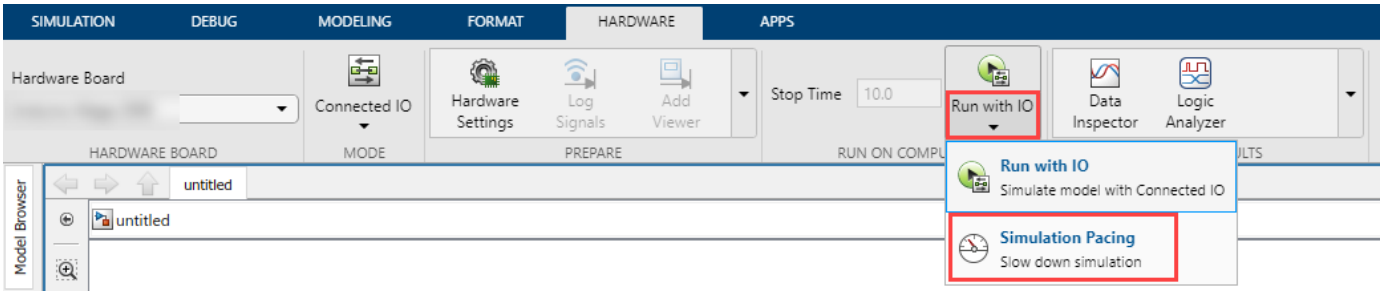
Run Simulink Model in Connected I/O

To simulate a model in Connected I/O during Normal mode simulation, perform these steps:

1. Open a Simulink model.
2. In the **Modeling** tab, select **Model Settings**.
3. In the Configuration Parameters dialog box, select **Hardware Implementation** from the left pane and select the target hardware in the **Hardware board** parameter.
4. On the **Hardware** tab of the model, in the **Mode** section, select `Connected IO` and then click **Run with IO**.



5. Additionally, you can change the rate of simulation by enabling the simulation pacing. For more information, see Simulation Pacing Options.



See Also

“Getting Started with Connected IO for PX4 Host Target” | “Getting Started with Connected IO for PX4 Autopilot”

Connected I/O

Communicate with Hardware Using Connected IO

You can use Connected IO to communicate with the IO peripherals on the hardware.

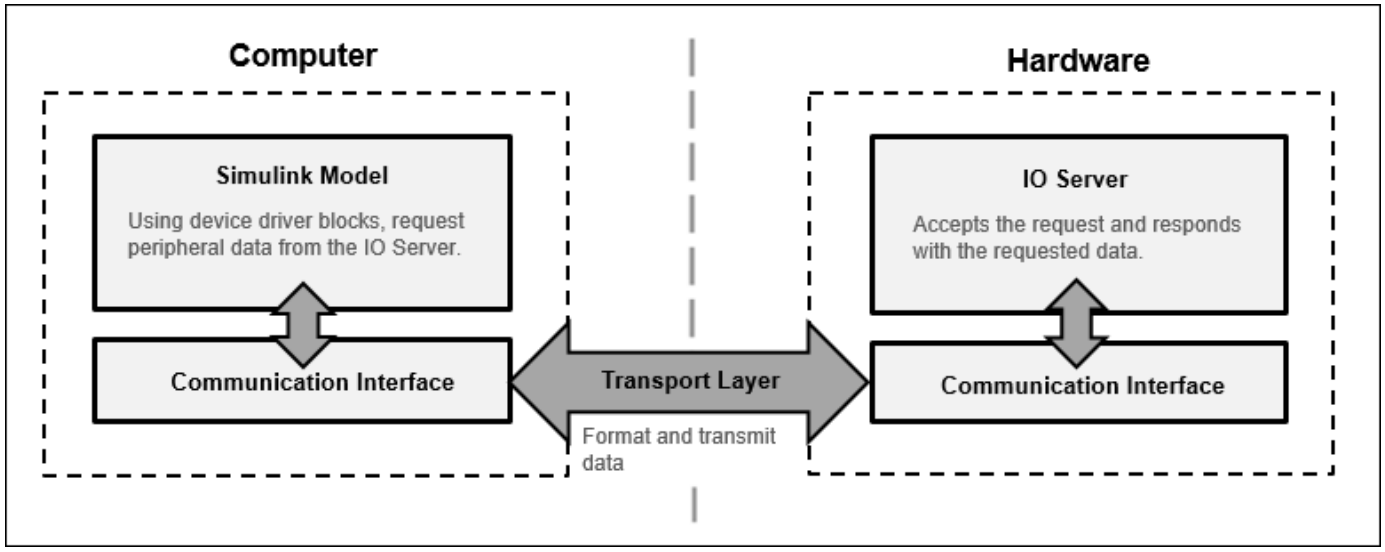
These sections explain:

In this section...

How Connected IO Works

Connected IO 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 IO creates between your computer and the hardware.



Communication in Connected IO

When you simulate a Simulink model with Connected IO:

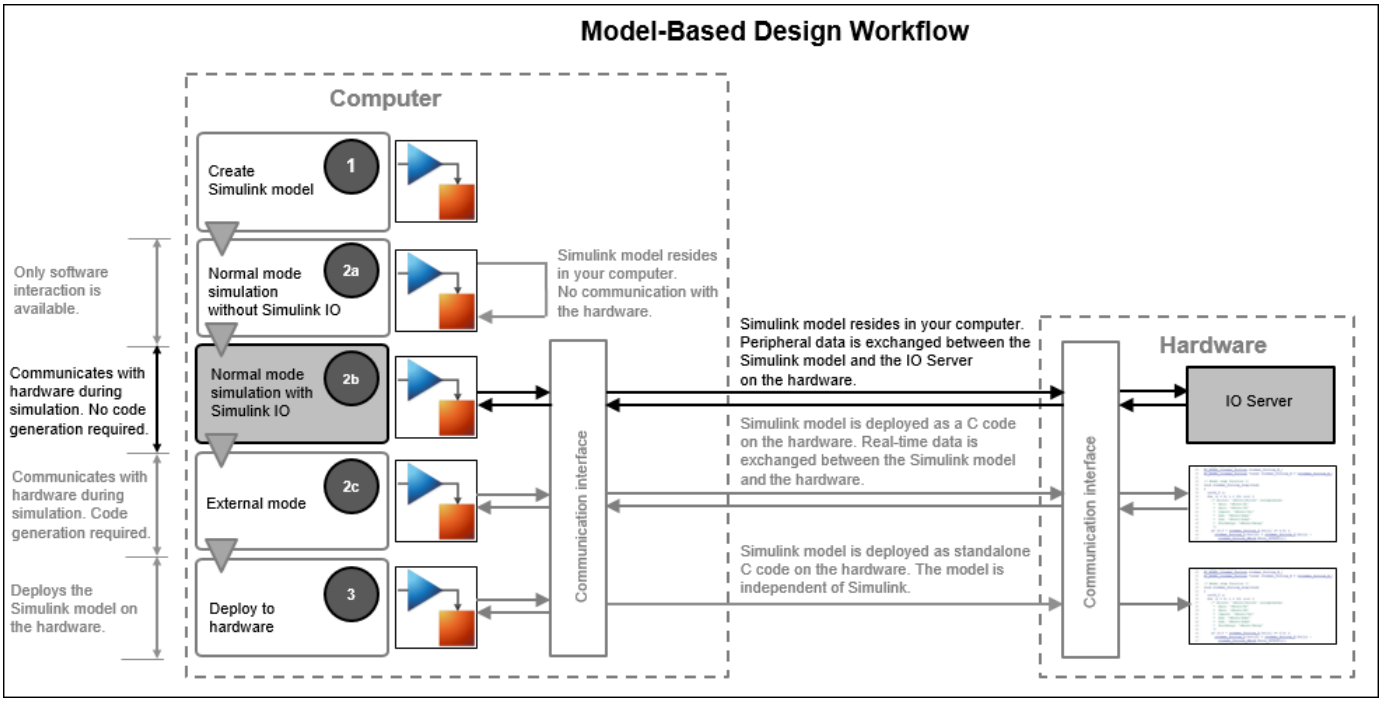
1. If necessary, you can modify the design by adding, removing, or replacing any block in the Simulink model.
2. After the model is modified, resimulate the model. During simulation, the data request from the model is communicated to the hardware. You can continue to modify and simulate the model until the expected behavior is achieved.

Connected IO in Model-Based Design

When you simulate a model without Connected IO, 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. Connected IO 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. Simulation without Connected IO: There is no hardware interaction and no code generation.
   b. Simulation with Connected IO: The model communicates with the hardware. There is no code generation.
   c. External mode (Monitor & Tune): The model is deployed on the hardware and generates code.
3. Deploy the model to the hardware.



Model-Based Design Workflow

How Connected IO Differs from External Mode

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

Run on Target for UAV Toolbox Support Package for PX4 Autopilots

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:

- Cube Blue H7
- Cube Orange
- Cube Orange Plus
- CUAV X7+
- Pixhawk 6c
- Pixhawk 6x
- Pixhawk 4
- Cube (Pixhawk 2.1)
- Pixhawk 1 (mRo)
- Pixracer
- UVify IFO-S

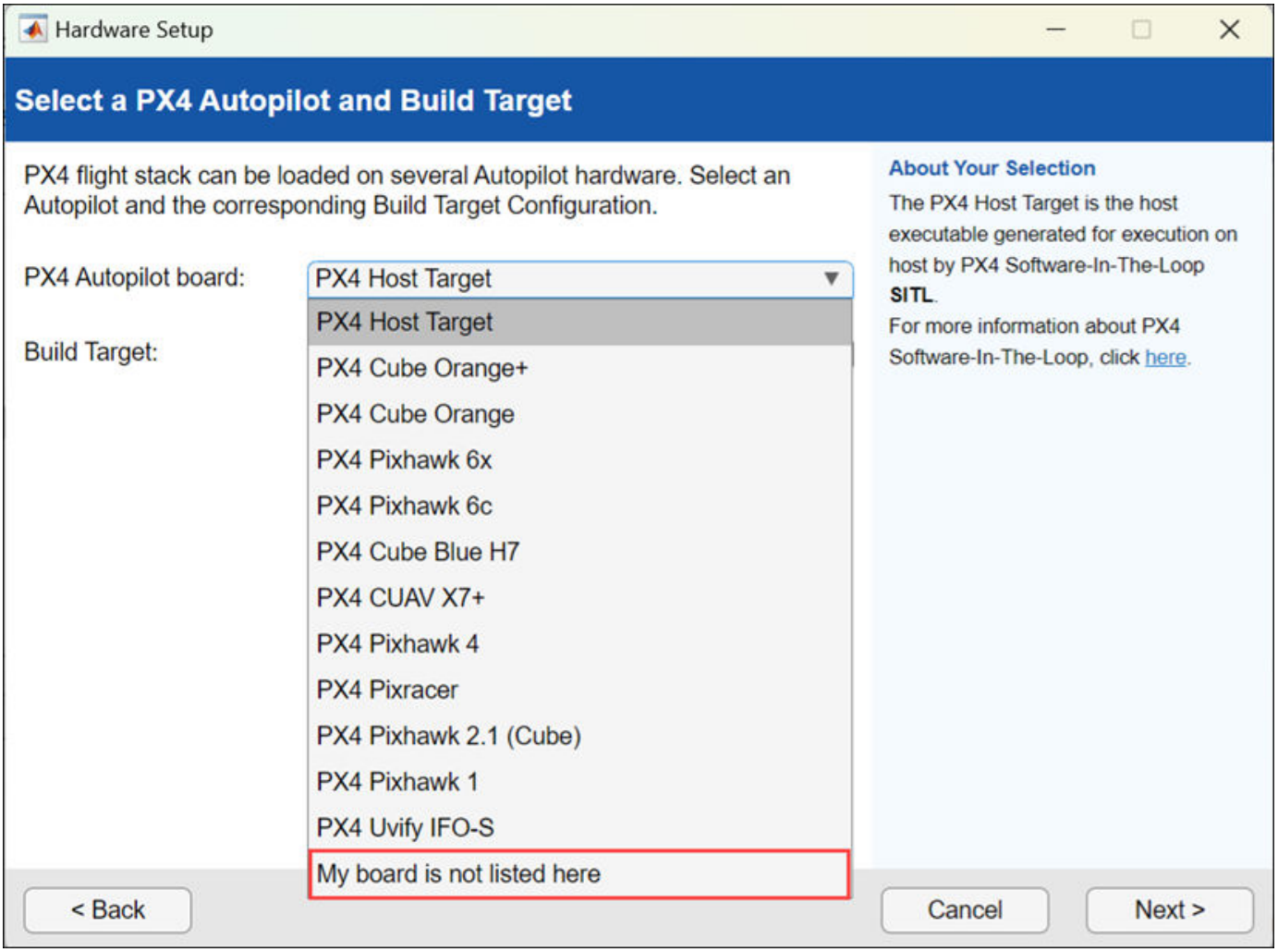
The support package can also be used with other PX4 Autopilots listed here that have been designed based on the Pixhawk FMU project and use a FMU configuration to build the firmware. However, the complete functionality is not tested on those controllers.

Support for PX4 Autopilots not listed above

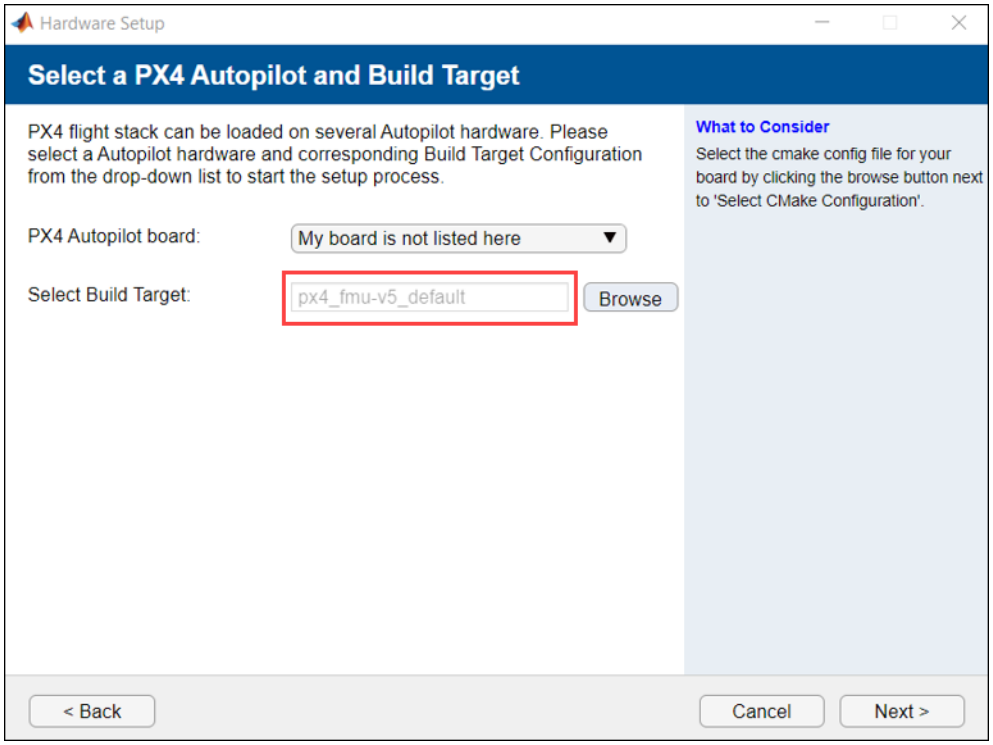
If you have a PX4 Autopilot that is not listed above, check if that hardware is supported on the PX4 version that Simulink currently supports (v1.14.3).

If your Autopilot is in the supported list of Autopilots for PX4 v1.14.3, you can consider using it with the support package. However, the complete functionality is not tested on those controllers. You might have a higher chance of getting it to work if you use Pixhawk Standard / Supported Autopilots. It is not recommended to use Manufacturer-Supported Autopilots. Experimental and Discontinued autopilots are not supported.

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.

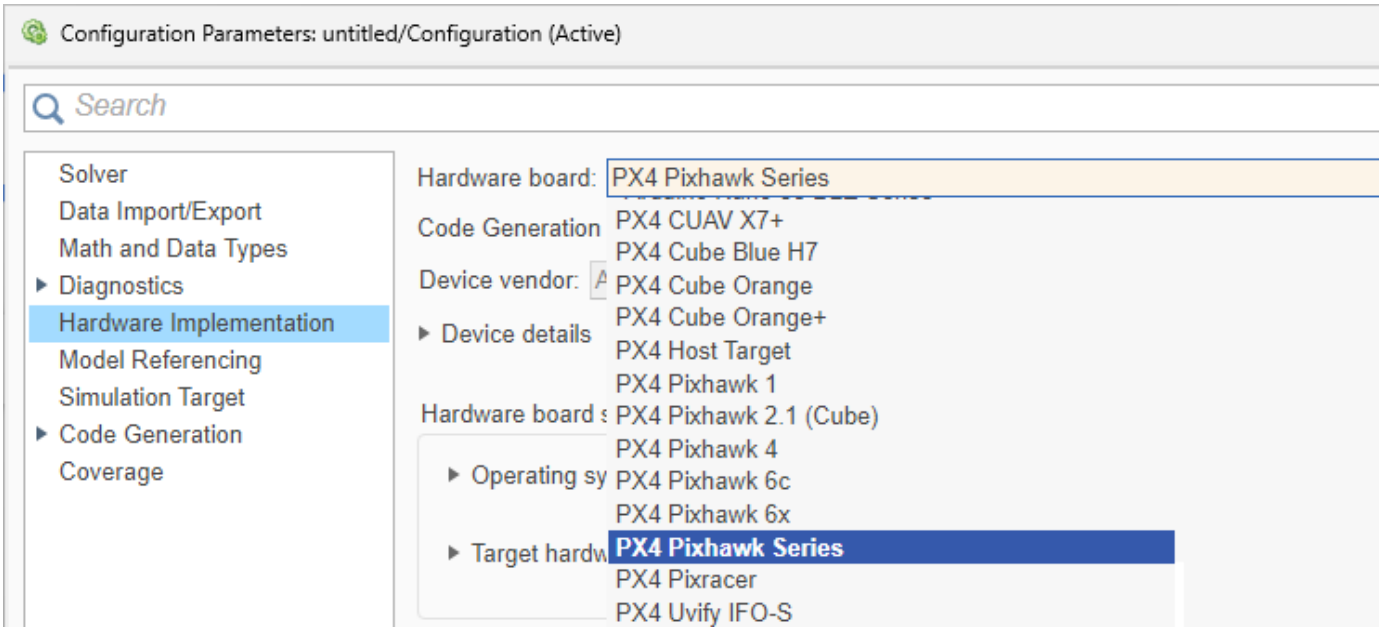


Browse to the corresponding CMake Build target in your PX4 Firmware for building.

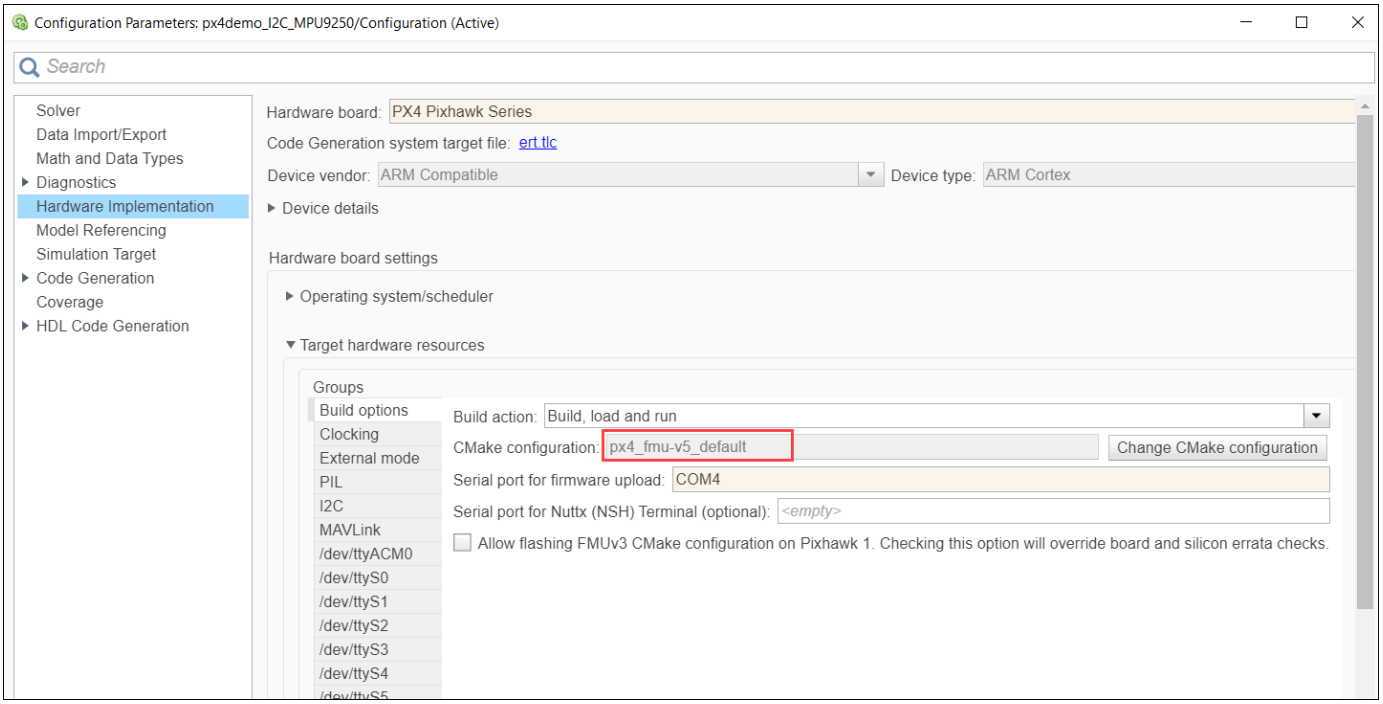


You might also need to enter the corresponding make command to build the firmware for the corresponding target.

After hardware setup is complete 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.



See Also

“Operating System Requirements” on page 1-2

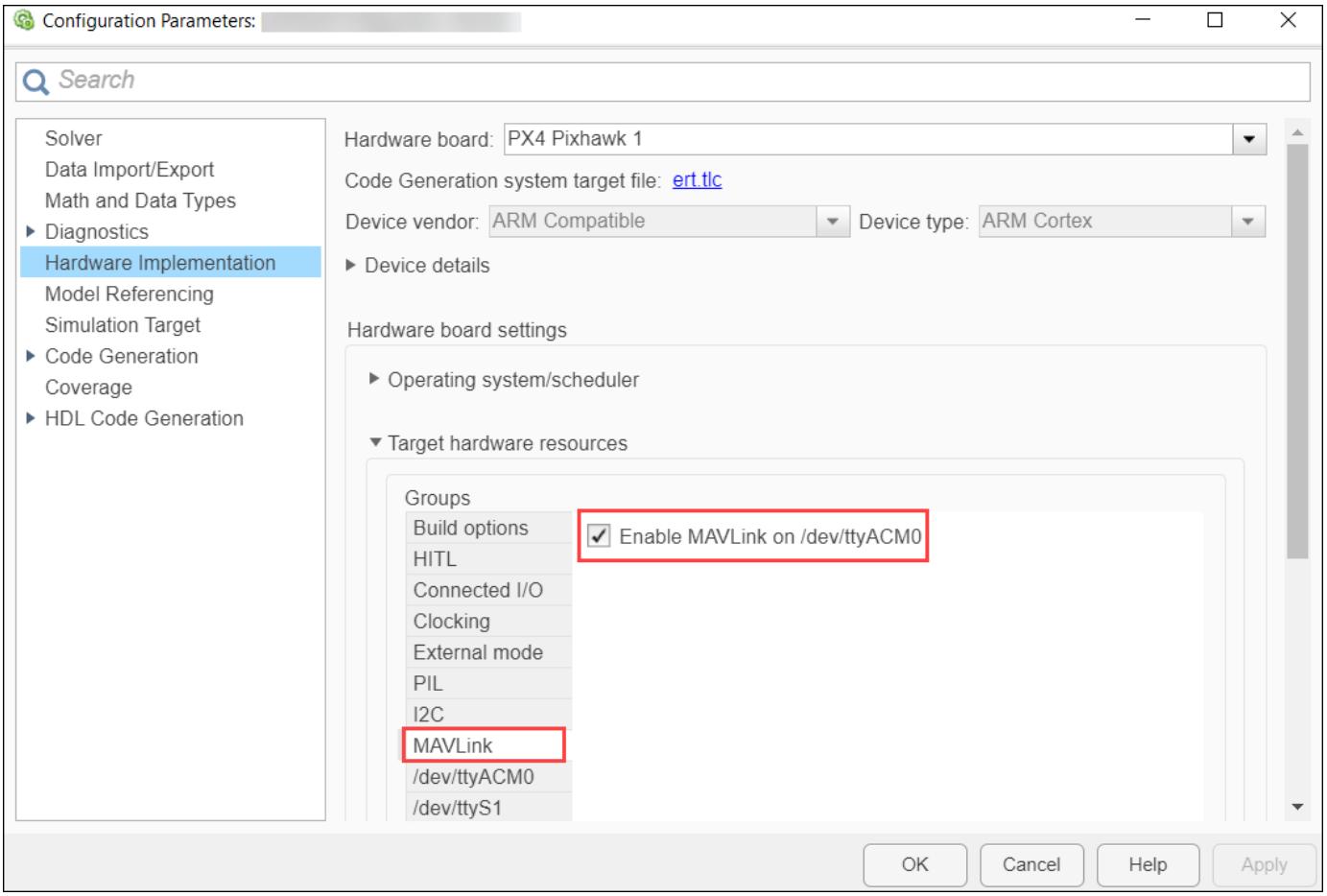
Enabling MAVLink in PX4 Over USB

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 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

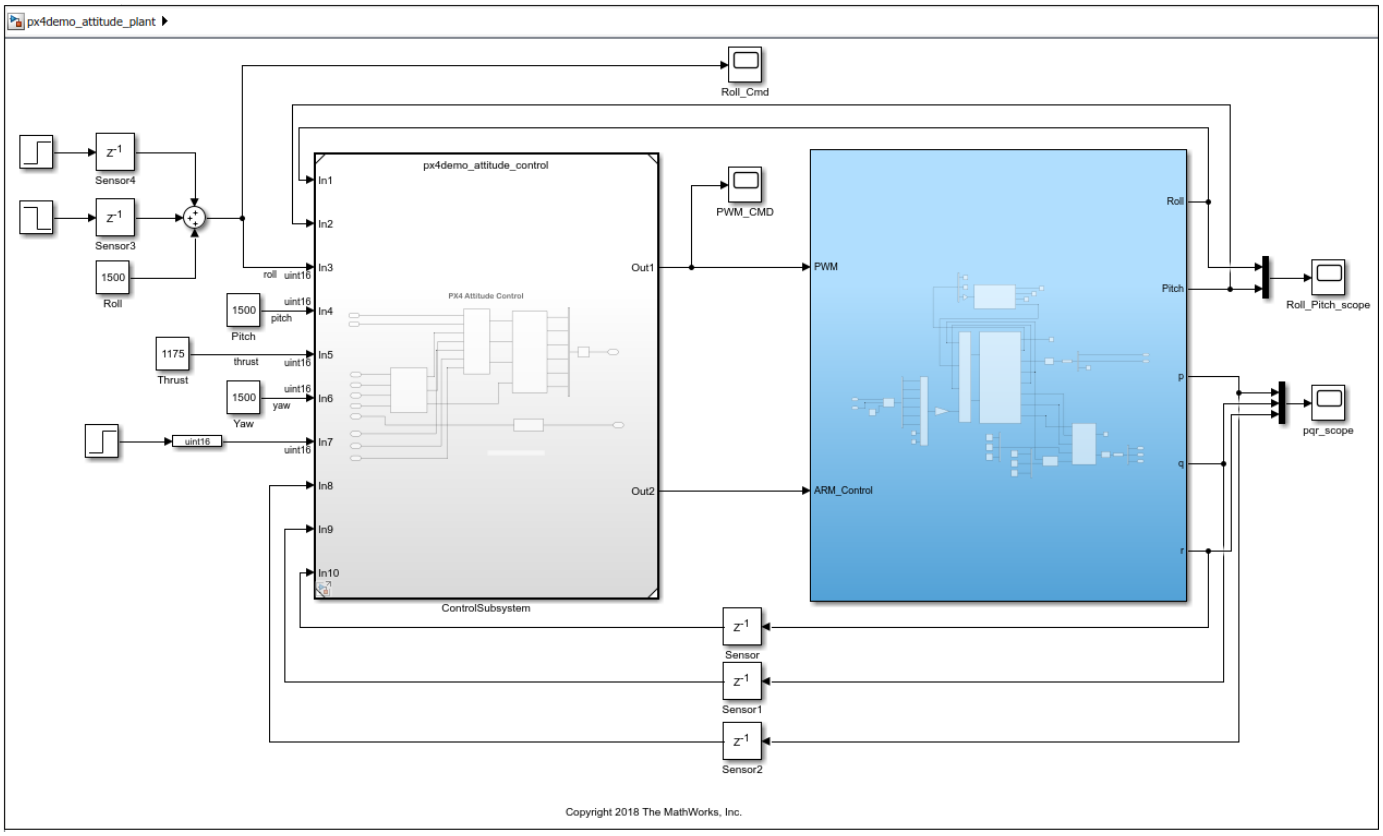
The UAV Toolbox Support Package for PX4 Autopilots contains a plant and an attitude controller model to fly a hexacopter drone that uses a Pixhawk Series flight controller.

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.

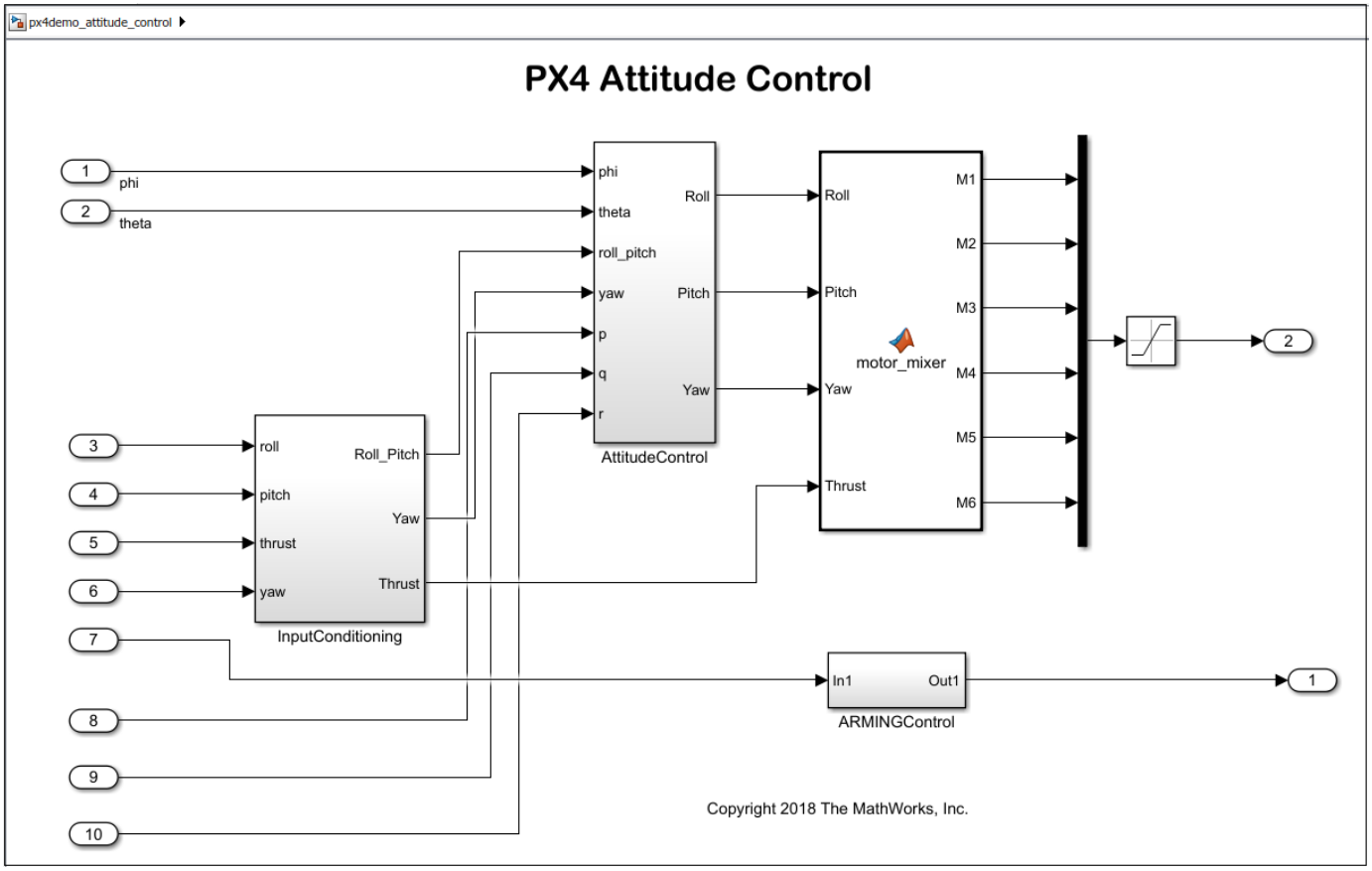
To open the plant model, enter the following command at the MATLAB prompt:

```
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.

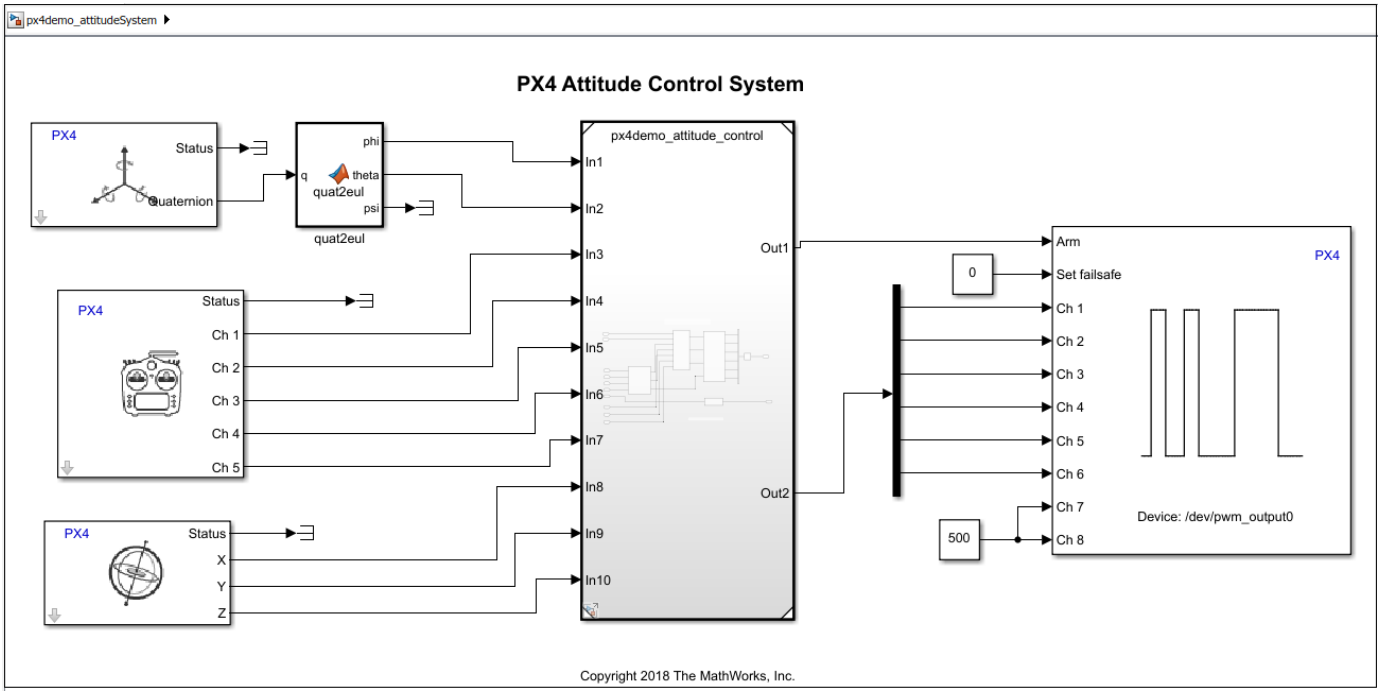


Generate Code for Controller for Hexacopter

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.

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:

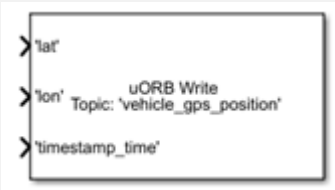
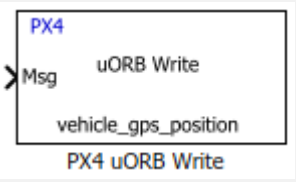
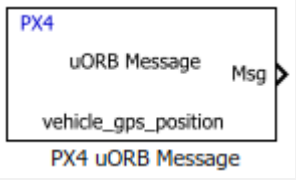
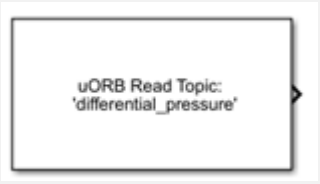
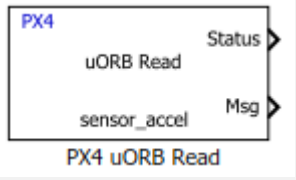
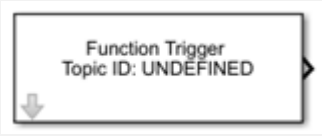
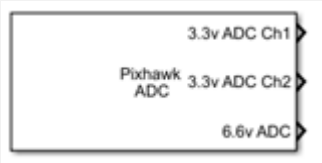
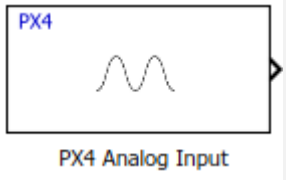
1. Complete the setup and configuration activities of UAV Toolbox Support Package for PX4 Autopilots using the Hardware Setup screens (see “Setup and Configuration”).

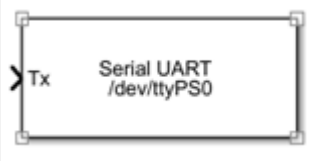
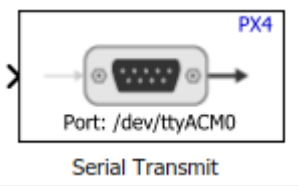
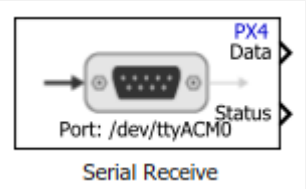
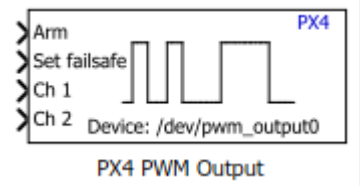
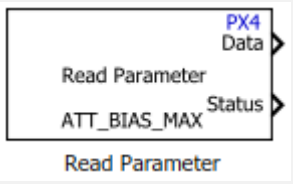
   The new support package supports PX4 firmware version v1.14 whereas the Pixhawk PSP supports PX4 firmware version 1.6.5. Therefore, you also need to clone PX4 firmware v1.14 by following the steps mentioned in the Hardware Setup screens.
2. Launch Simulink and create a new model.
3. Open the Configuration 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.

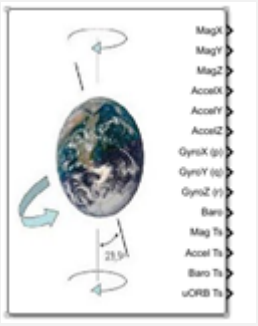
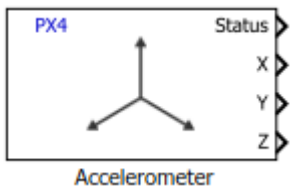
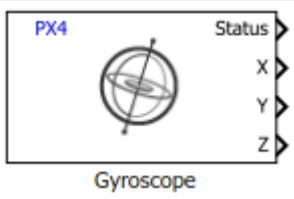
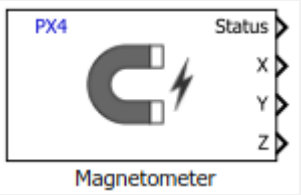
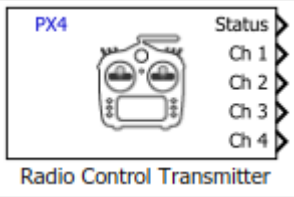
   If you are using a board that is not officially supported, select the `PX4 Pixhawk Series` option.
4. Copy the contents of the entire Simulink model from Pixhawk PSP to the new Simulink model that you created.
5. Replace the Simulink blocks for Pixhawk Pilot Support Package with the new Simulink blocks that are available in the new support package.

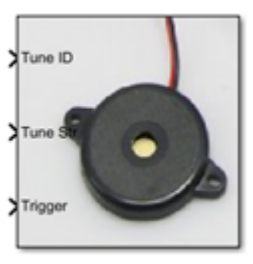
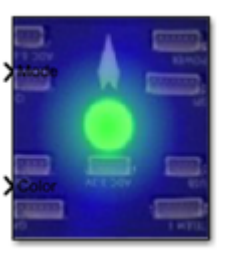
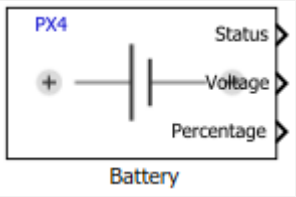
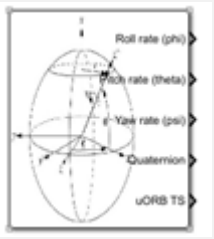
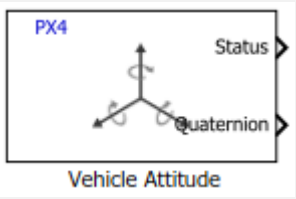
   Use the following table to migrate the Simulink blocks to the new UAV Toolbox Support Package for PX4 Autopilots, in R2018b and later releases.

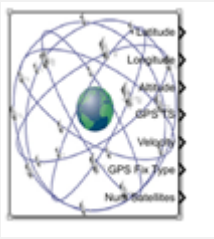
Mapping of Simulink blocks in Pixhawk Pilot Support Package to UAV Toolbox Support Package for PX4 Autopilots

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
Serial 	Serial Transmit and Serial Receive  	Serial Transmit and Serial Receive
PWM 	PX4 PWM Output 	PX4 PWM Output
ParamUpdate 	Read Parameter 	Read Parameter

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

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

Pixhawk Pilot Support Package Block	New Block in UAV Toolbox Support Package for PX4 Autopilots	Documentation Links
Vehicle GPS 	This block can be modelled using uORB Read block and vehicle_gps uORB topic. Refer to `px4_readGPS` Simulink model.	“Reading GPS Data from PX4 Autopilot”
Binary Logger 	Refer to `px4demo_log` Simulink model.	“MAT-file Logging on SD Card for PX4 Autopilots”
Print 	<Not supported>	

Integrating uORB Topics in the Simulink Model

Integrating uORB Topics in the Simulink Model

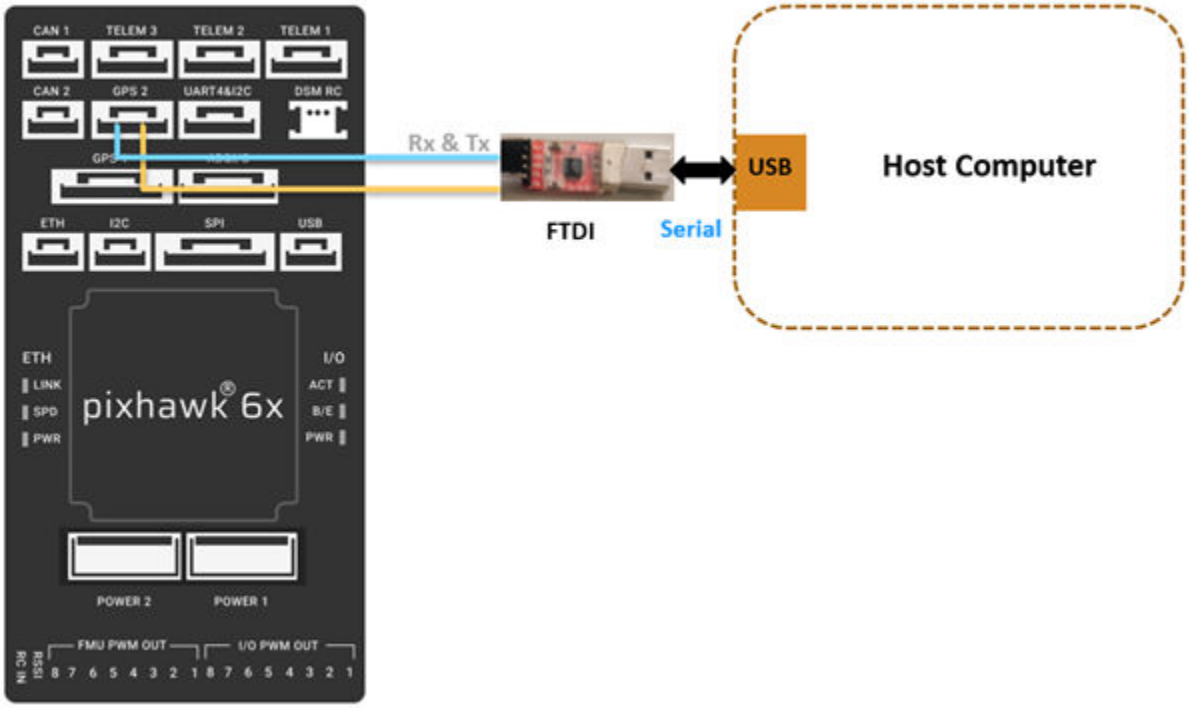
Setting Up the Hardware and Deploying the Model

Setting Up the Hardware and Deploying the Model

Running Monitor & Tune Simulation over FTDI with Pixhawk 6x

In UAV Toolbox Support Package for PX4 Autopilots, serial communication is supported over the native-USB port and non-usb ports on the Pixhawk Series boards. To achieve Monitor & Tune Simulation (External mode) over non-USB serial port, a serial-to-USB FTDI converter is required.

The below image shows the connection between GPS2 (/dev/ttyS7) on Pixhawk 6x board and the USB port on host computer, by using an FTDI converter.



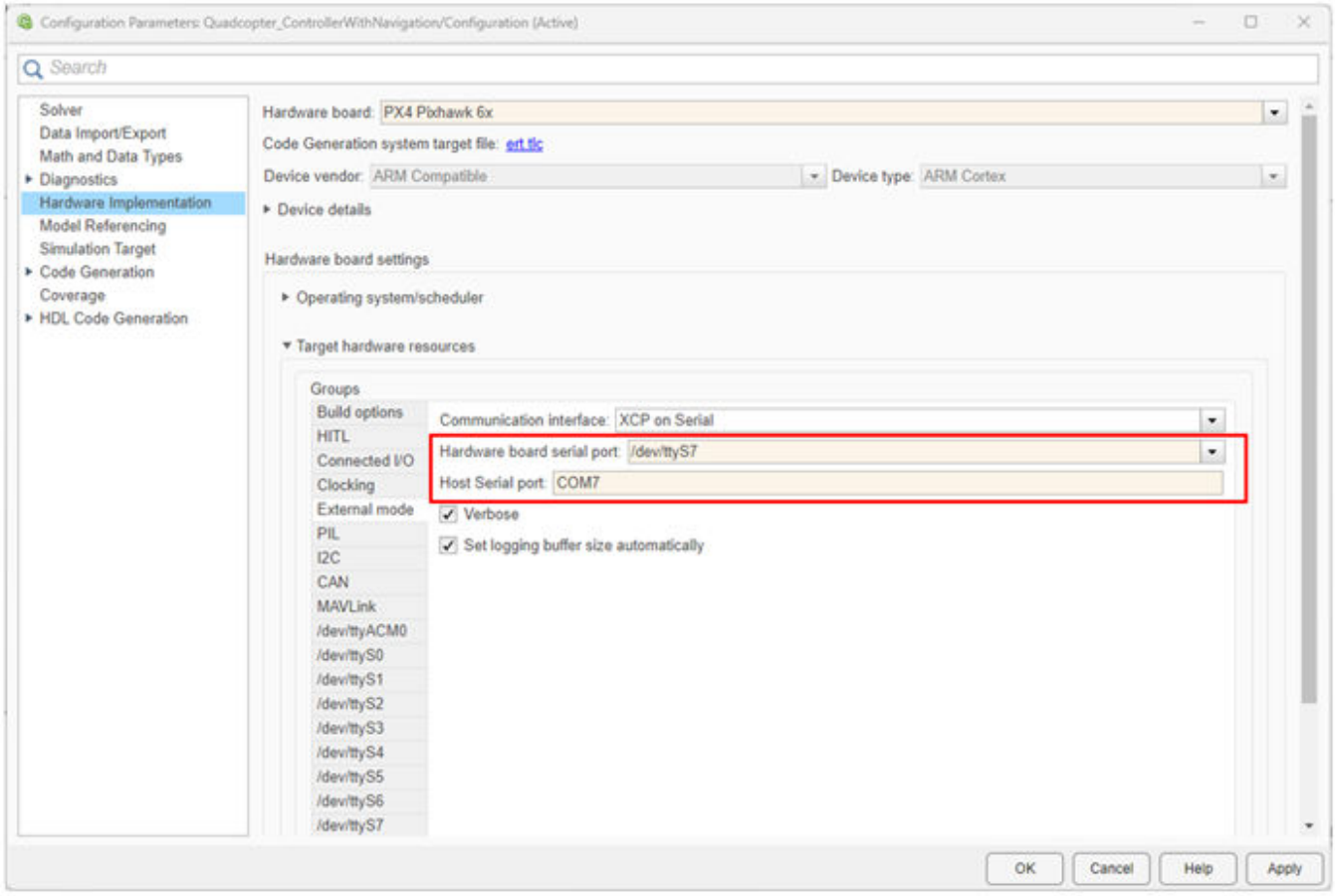
The below table lists the serial pins of the GPS2 on the Pixhawk 6x board. The pins listed here are from left to right.

Serial port mapping for GPS2 on Pixhawk 6x	
1	+5V
2	Tx
3	Rx
4	SCL2
5	SDA2
6	GND

Connect the **Rx** an **Tx** of GPS2 port to the **Tx** and **Rx** of the FTDI device (Serial to USB Converter) respectively. Connect the FTDI device to host computer over another USB port.

Note the Host serial port for the FTDI (for example, COM7). In Windows operating system, you can find this in 'Device manager'.

Once the above hardware connections are completed, open the Configuration Parameters dialog box in Simulink, and set the serial port to **dev/ttyS7** (go to `Hardware Implementation` pane > `Target Hardware Resources` > `External Mode`, and enter **dev/ttyS7** in the `Hardware board serial Port` field). Also enter the Host Serial port of your FTDI device in the **Host Serial Port** dialog box. A sample screen is shown below.



Now you can run the model in Monitor & Tune Simulation over FTDI with Pixhawk 6x 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 4-6).

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 1-38). 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/Simulink

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 Simulation-In-Hardware (SIH) (Software in the Loop). The support package supports simulation of algorithms by generating an executable on the host, referred as **PX4 Host Target** using the SIH in Host Target simulator, with jMAVSim or Simulink for visualization.

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

About Simulation-In-Hardware (SIH) Simulator

Simulation-In-Hardware (SIH) is an alternative to Hardware-In-The-Loop (HITL) simulation for quadrotors, fixed-wing vehicles (airplanes), and VTOL tailsitters. SIH is beneficial for new PX4 users to become familiar with PX4, explore its various modes and features, and learn to fly a vehicle using an RC controller within a simulation,something that is not possible with Software-In-The-Loop (SITL). For more information, see Simulation-In-Hardware (SIH).

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. Currently, jMAVSim is primarily used as a visualizer, not as a simulator. By using jMAVSim, you can test the algorithm for the desired take off, flight, and land actions.

Preparing PX4 Host Target Using Hardware Setup Screens

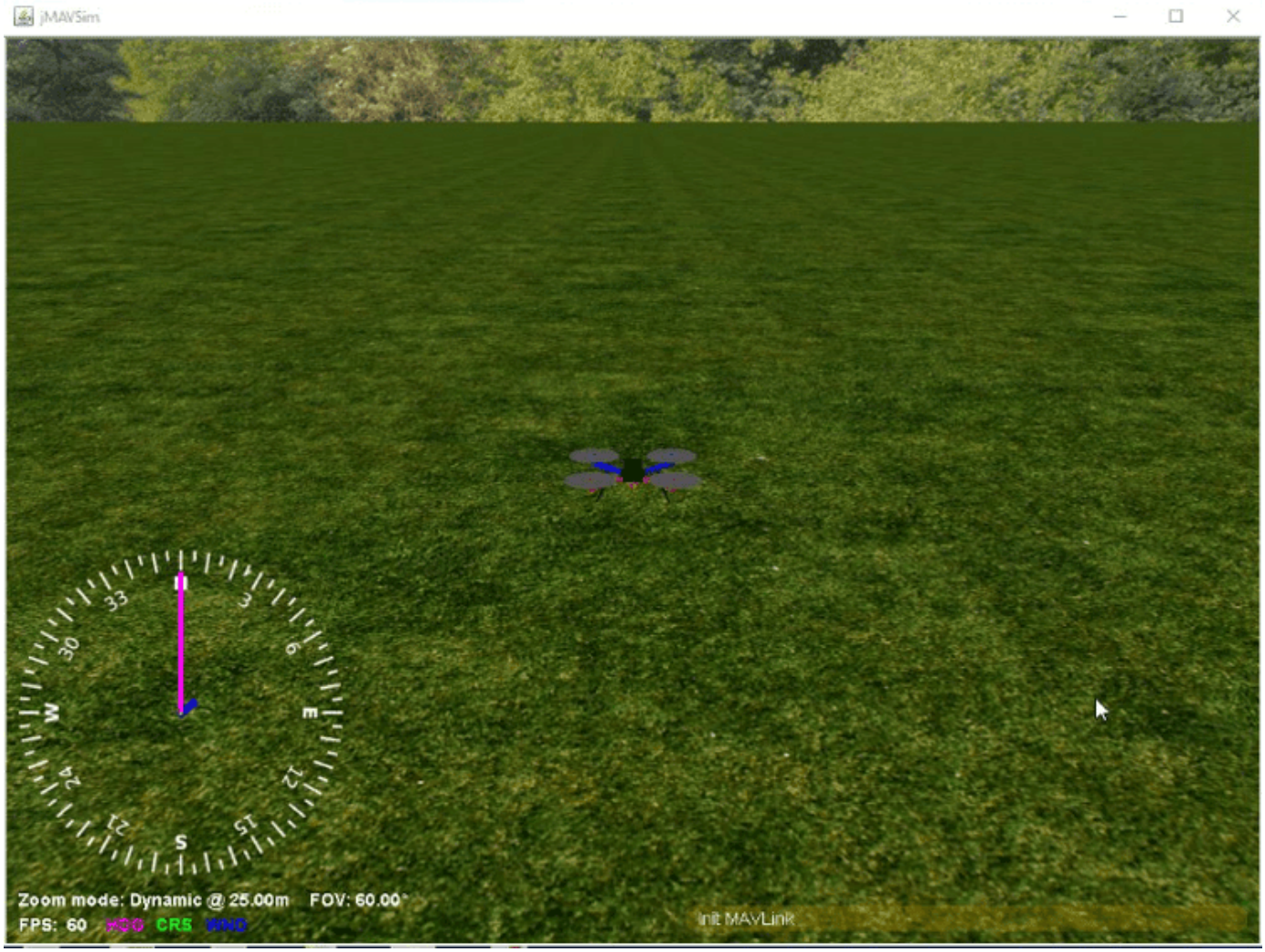
You can use Hardware Setup process in UAV Toolbox Support Package for PX4 Autopilots to prepare the support package for using the **PX4 Host Target** and testing the connection with SIH simulator and jMAVSim visualizer.

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

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 `px4_sitl_default`.

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

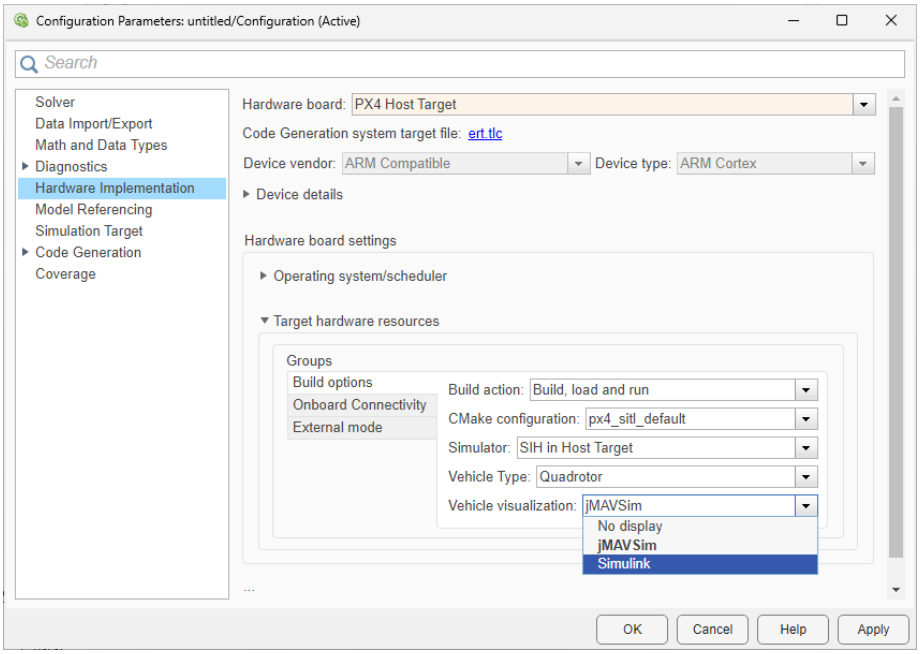
2. In the **Test Connection** Hardware Setup screen, click **Launch PX4 Host Target and jMAVSim**. The jMAVSim visualizer opens in a new window and shows the flight of the quadcopter. This is a test flight using a pre-configured algorithm in the support package to ensure that the necessary components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



Running PX4 Host Target 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 SIH 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 Select `SIH in Host Target` as the simulator.

Vehicle type selection is determined by the chosen airframe configuration.

2 **Vehicle Visualization:** You can select `jMAVSim` or `Simulink` for vehicle visualization.

For more information on using jMAVSim as visualizer, see “Simulate Manual Control for Fixed-Wing with PX4 Host Target”. For information on using Simulink as visualizer, see “Visualize 3D Scenarios in Unreal Engine with PX4 Host Target Simulation”.

If you select `No display`, you can use QGroundControl (QGC) to visualize the trajectory. For more information to configure QGC, see “Configuring QGC for Vehicle Visualization Without a Display” on page 4-26.

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

3 On the **Hardware** tab, in the **Mode** section, select **Run on board**, and then select one of these options to launch SIH 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 visualizer is launched in a separate window. The visualizer depends on the selection for Vehicle visualization option in the Configuration Parameters dialog box.

  If required, use jMAVSim as the visualizer and the SIH simulator for your setup. For more information, see “Visualize 3D Scenarios in Unreal Engine with PX4 Host Target Simulation”.

Note It has been observed that on some Ubuntu 18.04 Virtual Machines, there are issues while launching jMAVSim and PX4 Host Target from MATLAB. You might encounter a java crash dialog on

these systems in place of the expected jMAVSim and PX4 Host Target Windows. To workaround these behavior, follow the below steps.

1. Build the Simulink model for PX4 Host Target and wait for the build to complete.
2. If the jMAVSim and PX4 Host Target are not launched automatically after the build is complete, then launch a Terminal.
3. Navigate to the PX4 Firmware directory in the launched Terminal.
4. Execute the following command in Terminal to manually launch both PX4 Host Target and jMAVSim.

   ```
   make px4_sitl
   ```
5. To tune parameters and monitor signals, click **Connect** from **Hardware** tab of Simulink toolstrip after PX4 Host Target and jMAVSim are launched.

Note

- With PX4 Firmware v1.12.3, it has been observed that the **PX4 Host Target** may exhibit issues with modules not responding after one hour of simulation time.
- With PX4 Firmware v1.12.3, it has been observed that the drone in SITL mode crashes sporadically.

  Set the number of EKF instances to `1` to reduce the crashes substantially. To do this, set `EKF2_MULTI_IMU` and `EKF2_MULTI_MAG` parameters to `1` in QGC.

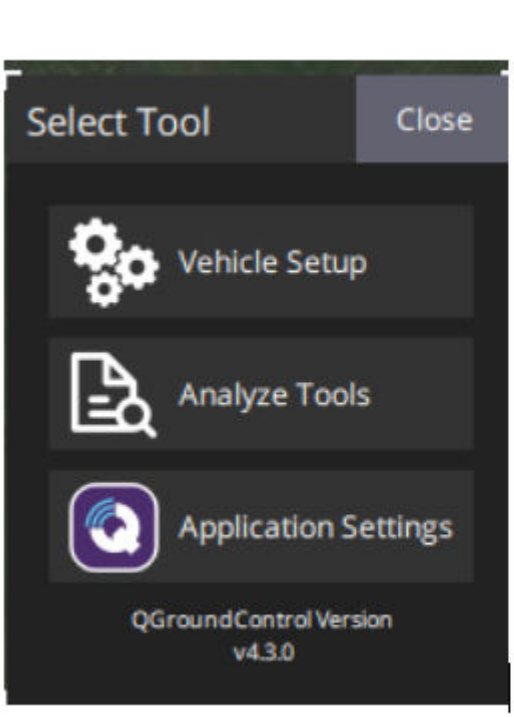
Note On Ubuntu 18.04 platform, it is recommended that you set the sample time as 1 millisecond for the Simulink models to be deployed on **PX4 Host Target**. If the sample time is greater than 1 millisecond, issues may occur while writing to some uORB topics such as `actuator_outputs`.

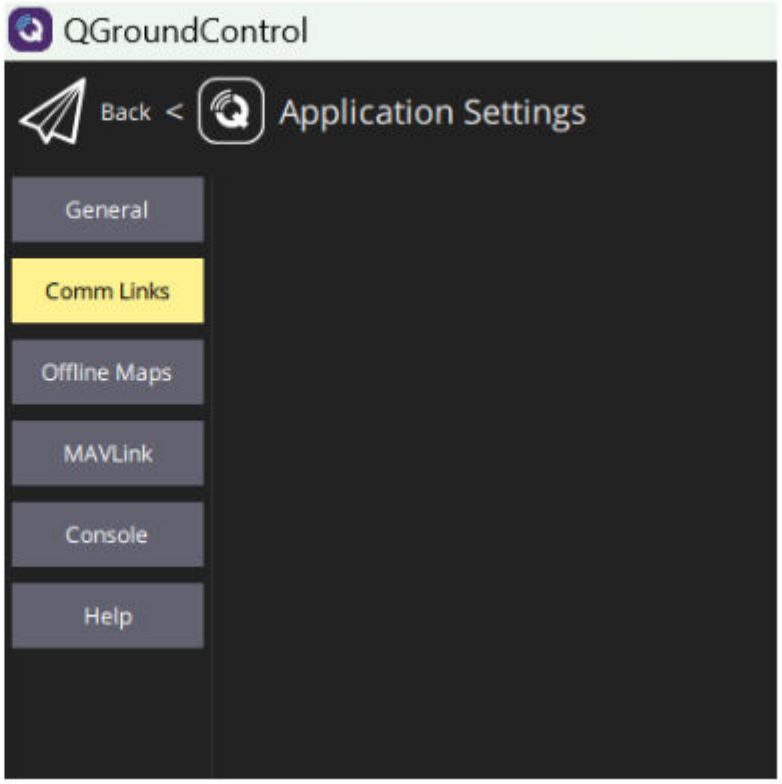
Configuring QGC for Vehicle Visualization Without a Display

Perform these steps if you select `No display` for vehicle visualization.

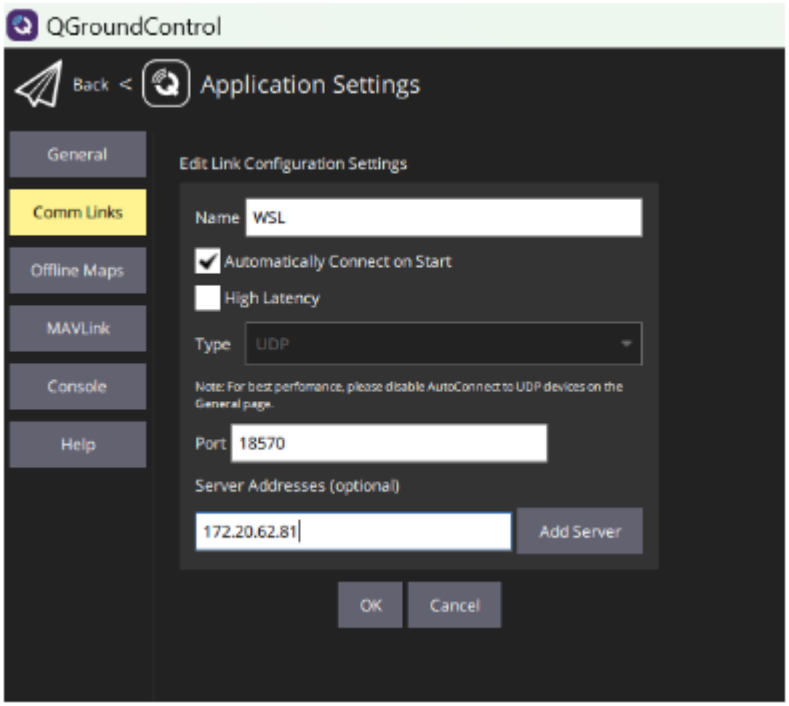
Note It is recommended to use QGC QGC version 4.3.0.

1. In the QGC, click the gear icon and select **Application Settings** and go to **Comm Links**.





2 Click **Add** and add link configuration settings.



3 Set the IP address information. Check the WSL IP address from Windows.

a
- Open Command Prompt in Windows.
- Execute the following command:

```
wsl hostname -I
```

4 Click **Add Server** and **OK**.

Note Select **Automatically Connect on Start** option to avoid connecting manually each time.

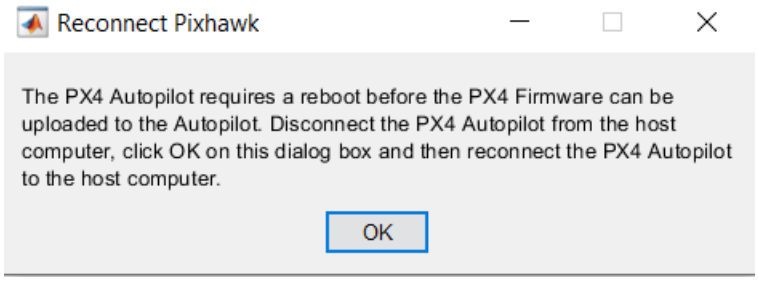
5 Select the comm link you created and click **Connect**.

6 Restart the QGC for the changes to take effect.

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.



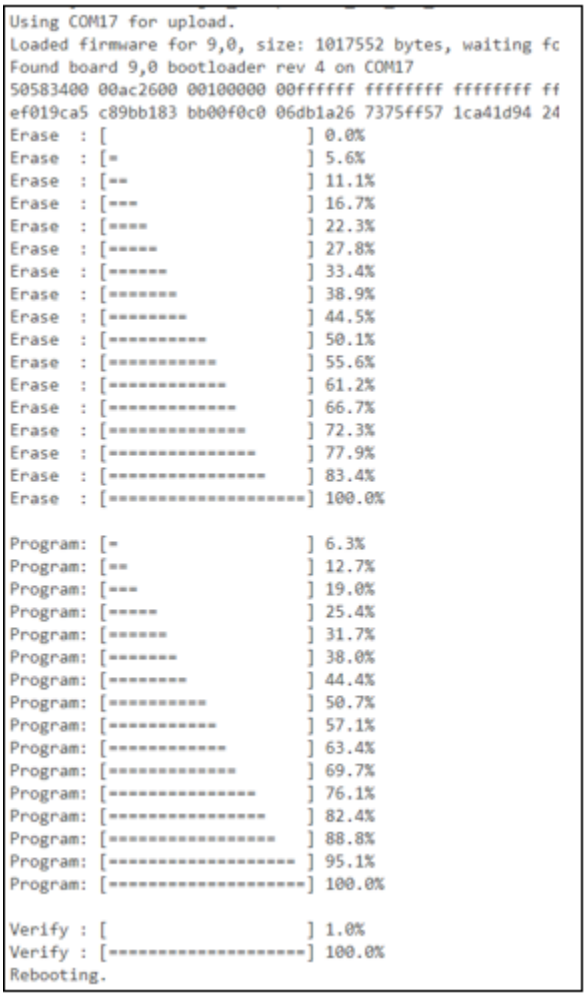
Action

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.



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 Running Out of File Descriptor Issues” on page 4-32 | “Troubleshooting USB Issues with Cube Orange on Windows” on page 4-33 | “Troubleshooting PX4 Firmware Build Failure Due to Flash Memory Overflow on the Hardware” on page 4-31

Troubleshooting PX4 Firmware Build Failure Due to Flash Memory Overflow on the Hardware

Description

If you are using the default PX4 board build target file, such as `cubepilot_cubeorange_default`, the build might fail due to flash memory overflow on the hardware with the following error message.

The reason for this issue is because the source code size increases with PX4 firmware v1.14. If all modules are enabled for build, it might result in flash memory overflow on the hardware.

Action

To troubleshoot the issue disable some modules from the default px4 board build target file. For example, if you are designing a Multicopter, consider removing fixed-wing controller modules and vice versa.

For some autopilots, the Multicopter, fixed-wing, and VTOL controllers are already selectively disabled, and variants of the default PX4 board build target files are available. If there is a Multicopter or fixed-wing variant of the PX4 board build target file for the PX4 autopilot you are using, choose that file instead of the default one.

For example, if you are using Cube Orange and facing build failure due to flash memory overflow on the hardware with the `cubepilot_cubeorange_default` PX4 board build target file, consider using `cubepilot_cubeorange_multicopter` for designing multicopter, `cubepilot_cubeorange_fixedwing` for -wing and `cubepilot_cubeorange _VTOL` for VTOL airframe.

If no Multicopter, fixed-wing, or VTOL variants exist for the autopilot you are using, manually create them by editing the default PX4 board build target file. For more information, see “Disabling Modules in PX4board Build Target File” on page 1-42.

See Also

“Troubleshooting Deploy to Hardware Issues” on page 4-29 | “Troubleshooting Running Out of File Descriptor Issues” on page 4-32 | “Troubleshooting USB Issues with Cube Orange on Windows” on page 4-33

Troubleshooting Running Out of File Descriptor Issues

Description

File descriptor running out warning or error messages (for example, `WARN [load_mon] <pthread> low on FDs! (1 FDs left)`) in the PX4 Console might cause undesirable behaviors.

Background

File descriptors are used to open and store uORB message handles, device driver handles and so on. By default, the number of File descriptors is limited to 12. PX4 runs out of file descriptors when large number of uORB Read / Write blocks or any other device driver blocks are added.

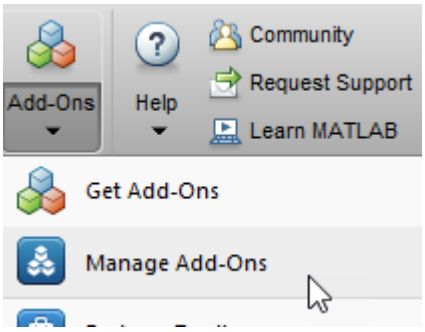
When PX4 runs out of File descriptors, the serial port opening is also handled using a file descriptor, thus Monitor and Tune simulation (external mode) connection might also fail.

Workaround

Increase number of File Descriptors by performing these steps.

For the CMake config you are using, go to the corresponding `nuttx` config folder. For example, for `fmu-v5` go to `Firmware\boards\px4\fmu-v5\nuttx-config\nsh`

1. Open the `defconfig` file.
2. Modify the property `CONFIG_NFILE_DESCRIPTORS` to the desired number (for example, 20).
3. Build the complete PX4 Firmware by navigating through the Hardware Setup screens. You can initiate the hardware setup process by opening the Add-On Manager.



   In the Add-On Manager, start the hardware setup process by clicking  icon.
4. Try building and deploying / running Monitor and Tune Simulation (external mode) from Simulink.

See Also

“Troubleshooting USB Issues with Cube Orange on Windows” on page 4-33 | “Troubleshooting PX4 Firmware Build Failure Due to Flash Memory Overflow on the Hardware” on page 4-31 | “Troubleshooting Deploy to Hardware Issues” on page 4-29

Troubleshooting USB Issues with Cube Orange on Windows

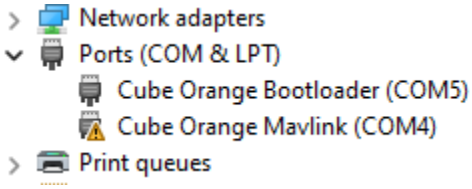
Description

Serial port issues while using Cube orange with Simulink on a Windows computer.

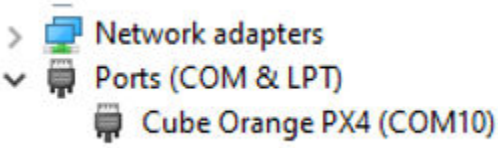
Action

When you use Cube Orange with Simulink on a Windows computer, ensure that you install the version 4.3.0 of QGroundControl and able to upload latest stable version of PX4 Firmware from QGroundControl

After uploading the PX4 firmware, open the Device Manager. if the Cube Orange hardware appears as two different COM ports in your machine as shown below, then install Mission planner and do a clean re-installation of latest drivers as described in this page.



If the driver installations are correct, Cube orange will appear as a single COM port as shown below.



If, Cube Orange is not shown as a single port even after installing Mission Planner, perform a clean re-installation of the latest drivers as explained in this page.

See Also

“Troubleshooting PX4 Firmware Build Failure Due to Flash Memory Overflow on the Hardware” on page 4-31 | “Troubleshooting Deploy to Hardware Issues” on page 4-29 | “Troubleshooting Running Out of File Descriptor Issues” on page 4-32

Monitor and Tune the Model Running on PX4 Autopilots

In this section...

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 (Simulink) blocks or in Simulation Data Inspector (SDI) (Simulink). 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 XCP on TCP/IP when PX4 Host Target is selected as hardware board:

Communication Interface	Description
`XCP on TCP/IP`	In Universal Measurement and Calibration Protocol (XCP)-based External mode simulation over TCP/IP, you can use: • “Dashboard” (Simulink) blocks: In addition to “Sources” (Simulink) and Sink (Simulink) blocks, you can use “Dashboard” (Simulink) 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) (Simulink): You can inspect and compare data from multiple simulations to validate model designs using Simulation Data Inspector.

The support package supports Monitor and Tune simulation over XCP on Serial when Pixhawk 1, Pixhawk 2.1 (Cube), Pixhawk 4, Pixhawk Series, or Pixracer is selected as hardware board:

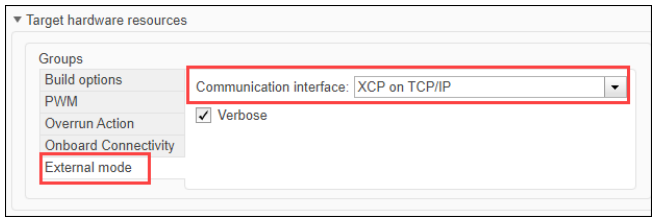
Communication Interface	Description
`XCP on Serial`	In Universal Measurement and Calibration Protocol (XCP)-based External mode simulation over Serial, you can use: • "Dashboard" (Simulink) blocks: In addition to "Sources" (Simulink) and Sink (Simulink) blocks, you can use "Dashboard" (Simulink) blocks 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) (Simulink): You can inspect and compare data from multiple simulations to validate model designs using Simulation Data Inspector.

Prepare a Simulink Model for External Mode

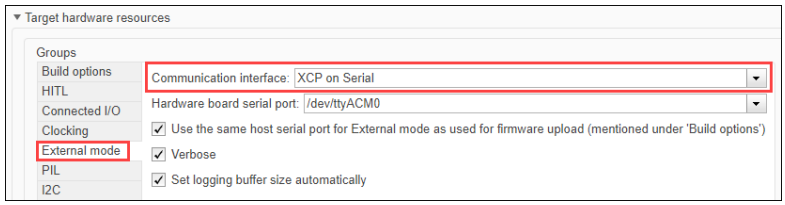
This section explains how to prepare a Simulink model to run in External mode.

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 is `XCP on TCP/IP`.

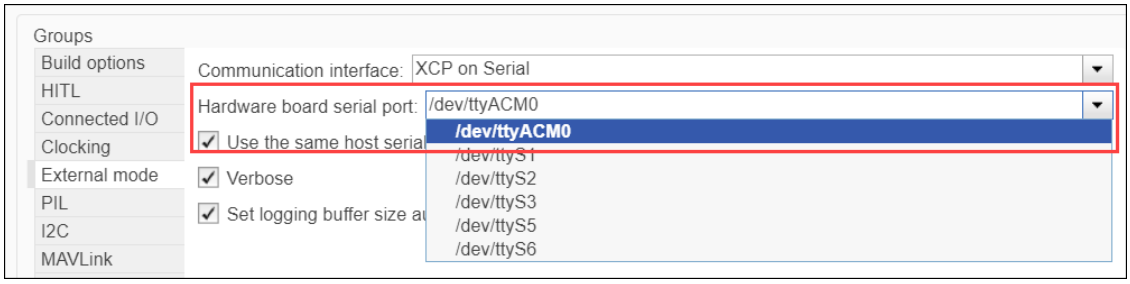


   When Pixhawk 1, Pixhawk 2.1 (Cube), Pixhawk 4, Pixhawk Series, or Pixracer is selected as hardware board, Communication interface is `XCP on Serial`.

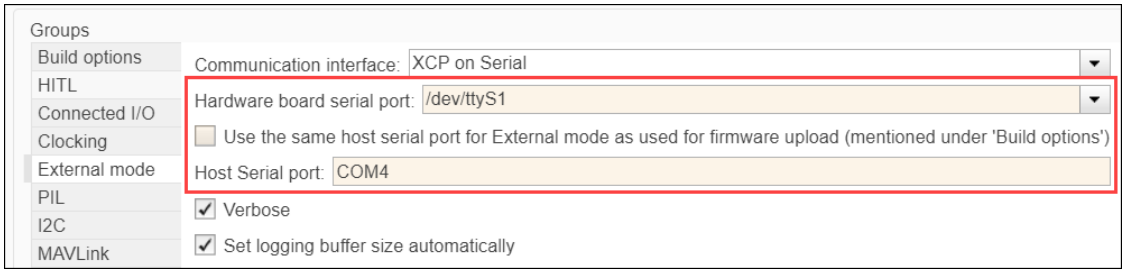


3. Ensure that you select the **Verbose** check box to view the external mode execution progress and updates in the Diagnostic Viewer or in the Command Window.

4. Select **Use the same host serial port for External mode as used for firmware upload** parameter to set the same host serial port for External mode as the one used for serial port for firmware upload. By default, this parameter is selected. If you clear this parameter, **Host Serial port** parameter becomes available, where you can explicitly define the serial port for External mode on the host computer. **Use the same host serial port for External mode as used for firmware upload** and **Host Serial port** parameters are available only when Pixhawk 1, Pixhawk 2.1 (Cube), Pixhawk 4, Pixhawk Series, or Pixracer is selected as hardware board.
5. If the communication interface is `XCP on Serial`, the **Set logging buffer size automatically** parameter becomes available. Select this parameter to automatically set the number of bytes to preallocate for the buffer in the hardware during simulation. By default, the **Set logging buffer size automatically** parameter is selected. If you clear this parameter, **Logging buffer size (in bytes)** parameter becomes available, where you can manually specify the memory buffer size for XCP-based External mode simulation. **Set logging buffer size automatically** and **Logging buffer size (in bytes)** parameters are available only when Pixhawk 1, Pixhawk 2.1 (Cube), Pixhawk 4, Pixhawk Series, or Pixracer is selected as hardware board.
6. 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.

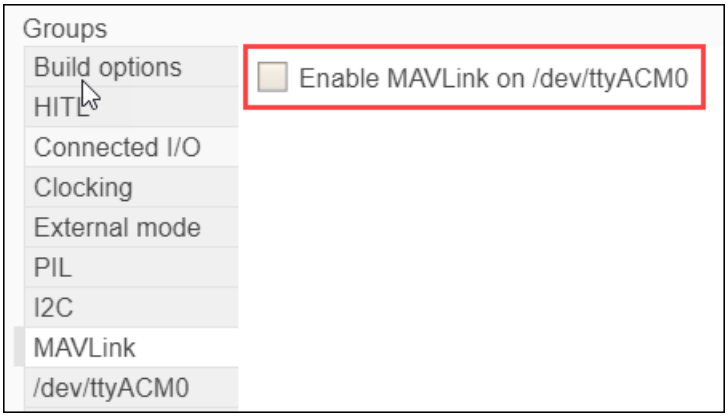


7. If the Serial port for External Mode communication on hardware board is a port other than `/dev/ttyACM0` mention the corresponding COM port on host.



Note If the serial port used for Monitor and Tune communication on hardware board is a port other than the USB port, then make sure that PX4 is not using that port. For example, MAVLink is enabled by default in `TELEM1` port and if you want to use the `TELEM1` port for Monitor and Tune communication, then disable MAVlink on `TELEM1`. To do so, set the parameter `MAV_0_CONFIG` to `0` on QGC. Similarly, to use the `GPS1` port, set the parameter `GPS_1_CONFIG` to `0`.

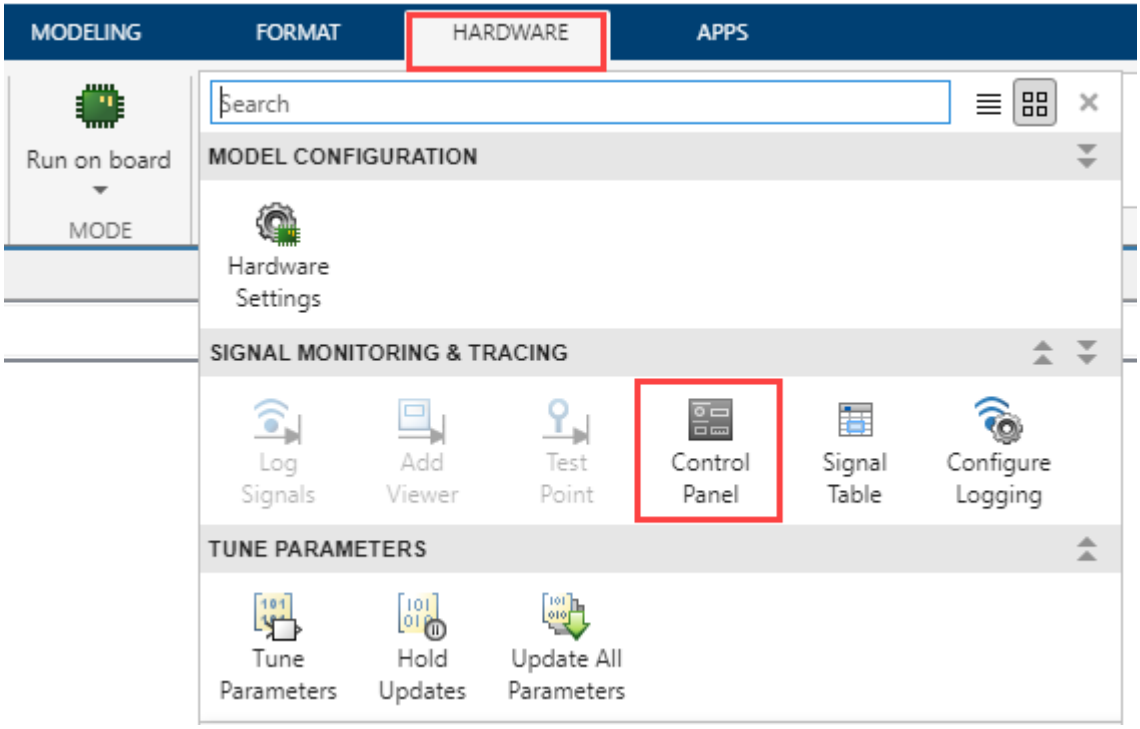
8. Ensure that **Enable MAVLink on /dev/ttyACM0** is cleared if the hardware board serial port that is selected for External Mode Communication is /dev/ttyACM0



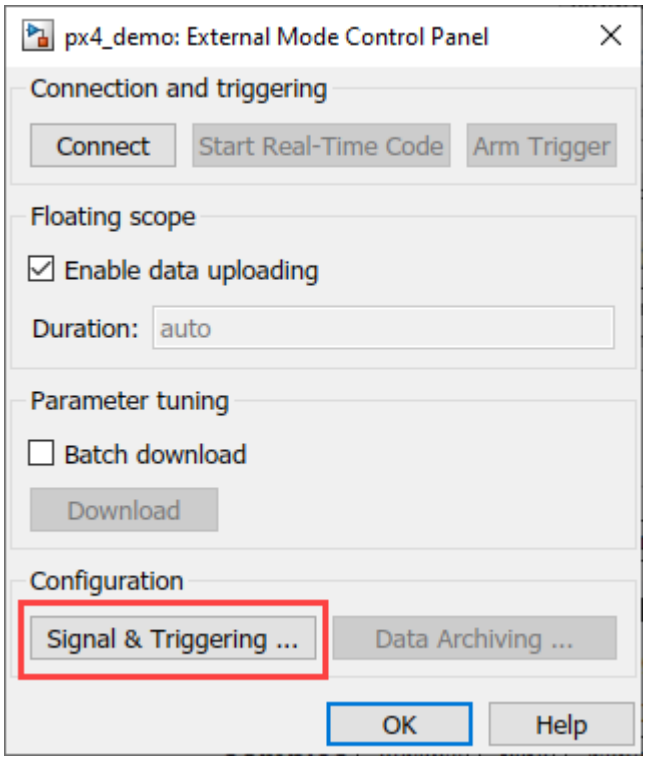
9 Click **Apply** and then **OK** to close the Model Configuration Parameters dialog box.

10 If the communication interface is `XCP on Serial` or `XCP on TCP/IP`, you can send multiple contiguous samples in same packet to enhance signal logging performance in models containing signals of high sample rates. To do so, perform these steps.

a Click **Hardware** tab, in the **Prepare** gallery, select **Control Panel**.

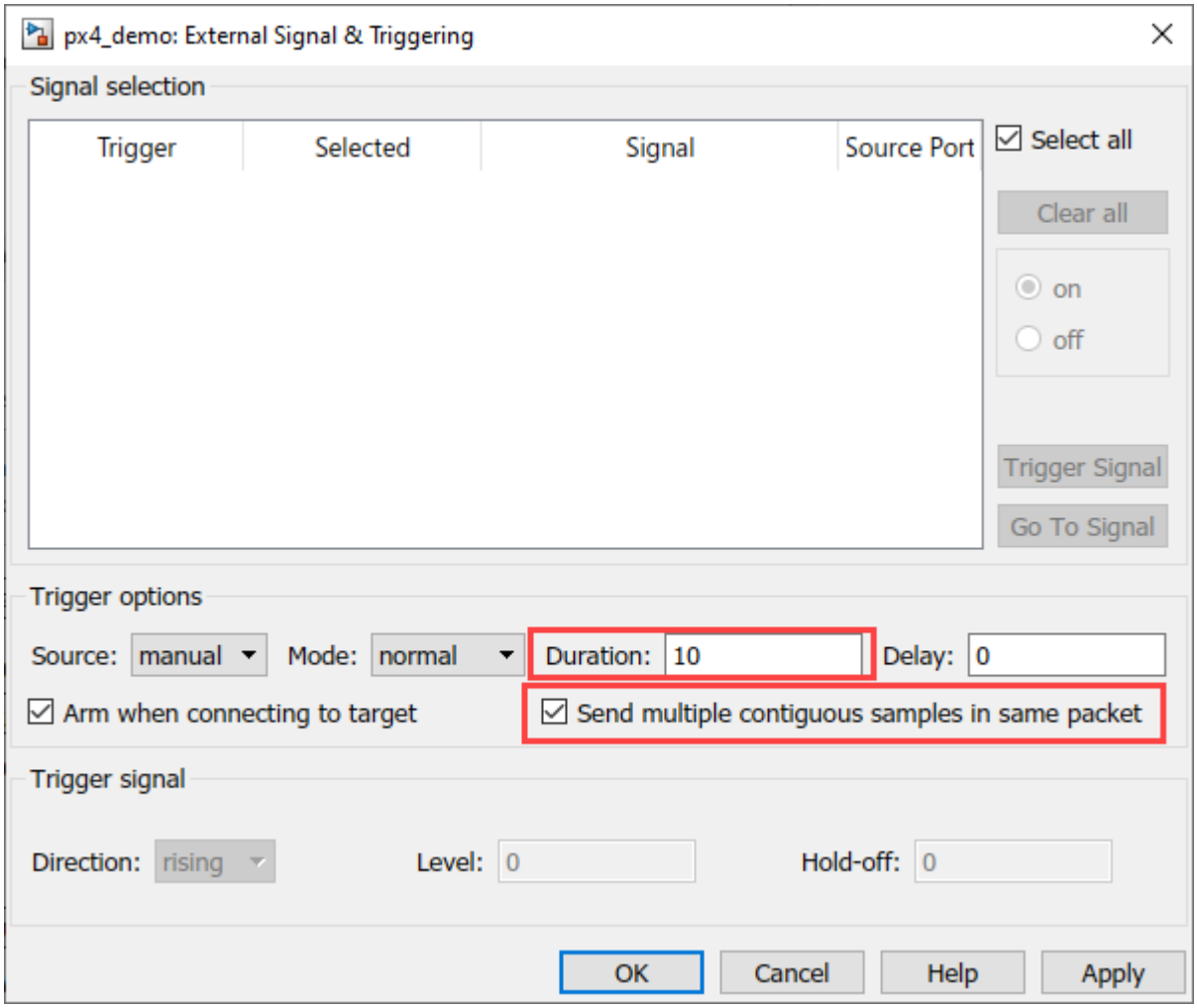


b In the **External Mode Control Panel** screen click **Signal & Triggering**.



c In the **External & Signal Triggering** dialog box, select **Send multiple contiguous samples in same packet** and enter a value for **Duration**.

This enables sending multiple contiguous samples in a single packet whose length is the value specified for **Duration** parameter.



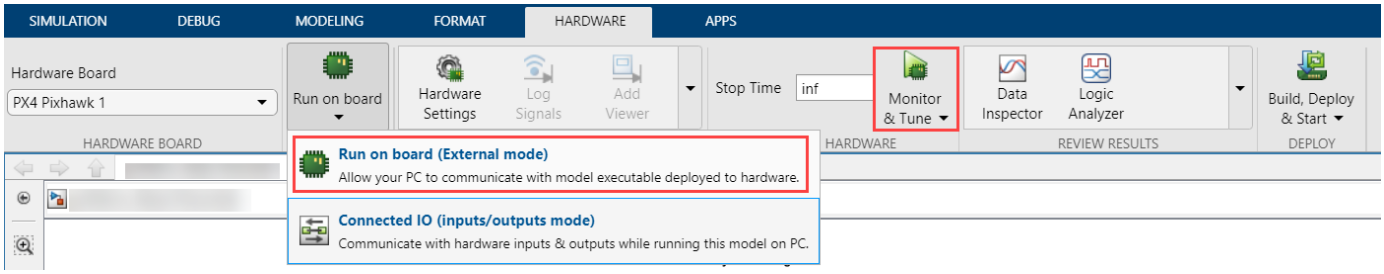
d Click **Apply** and **OK**.

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. On the **Hardware** tab, in the **Mode** section, select **Run on board** and then 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

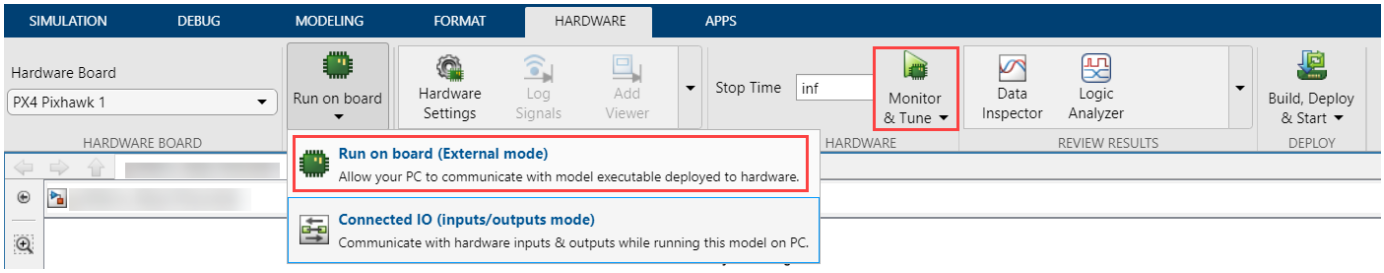
This section explains how to run:

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

Before you begin, complete the “Prepare a Simulink Model for External Mode” on page 4-35 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 Signals for Logging” (Simulink). Simulink displays a logged signal indicator  for each logged signal.
2. (Optional) Place one or more Sink (Simulink) blocks in the model, and then mark the signals connected to them also for logging. For example, connect Display or Scope (Simulink) blocks and mark the signals connected to them for logging.
3. To start the simulation, on the **Hardware** tab, in the **Mode** section, select **Run on board** and then click **Monitor & Tune**.



   For instructions on logging the signal, see “Mark Signals for Logging” (Simulink).

   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 “Signal Groups” (Simulink).

  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 (Simulink) 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.

Stop Monitor and Tune

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

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

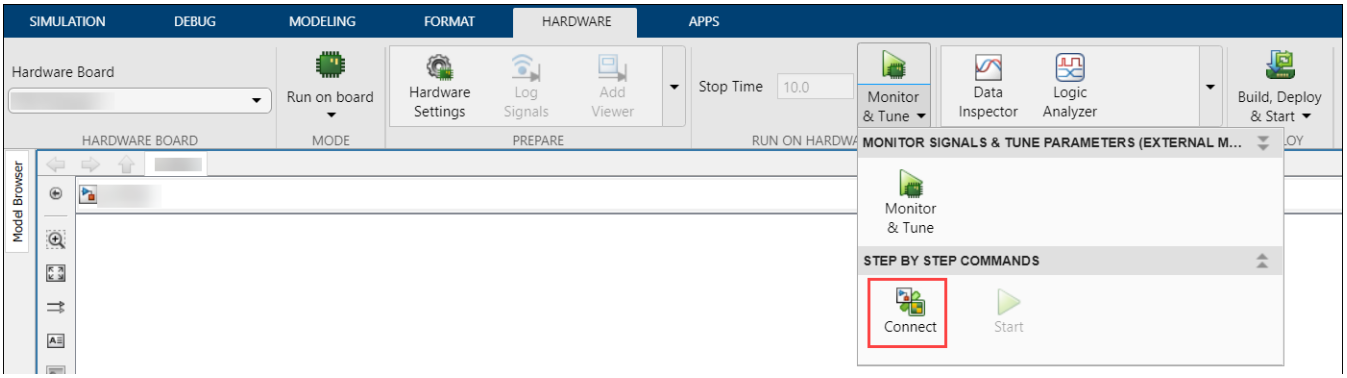
In some cases, the Monitor and Tune operation needs to be performed again even though there is no change to the Simulink model that was running on the PX4 Autopilot hardware. The reasons include:

- 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.

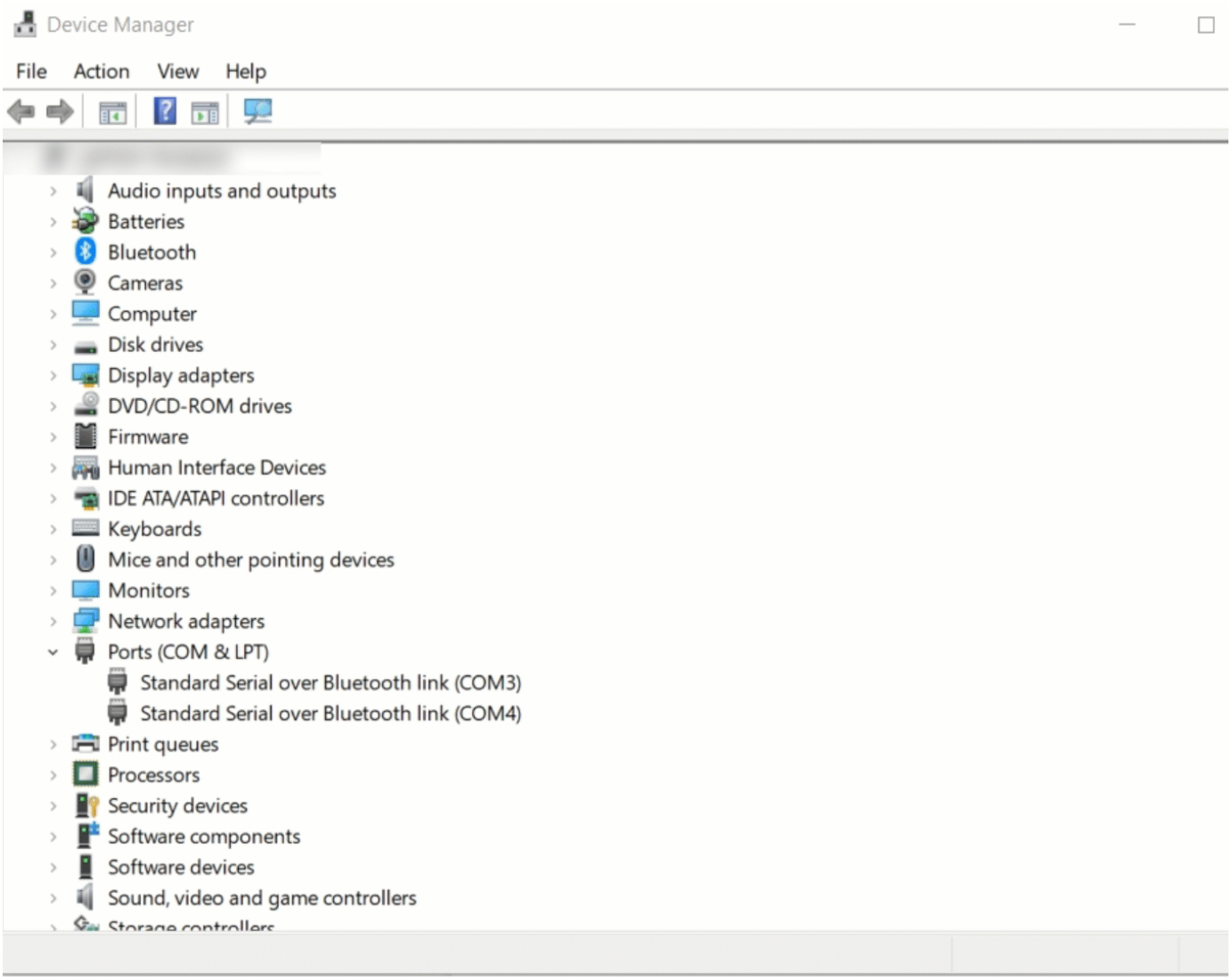
- If Simulink code generated for **Monitor & Tune** in the previous run is already flashed on hardware, click **Connect** in Simulink instead of clicking **Monitor & Tune** again to establish External mode simulation. By clicking **Connect**, the Simulink code for the model is not generated again, and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



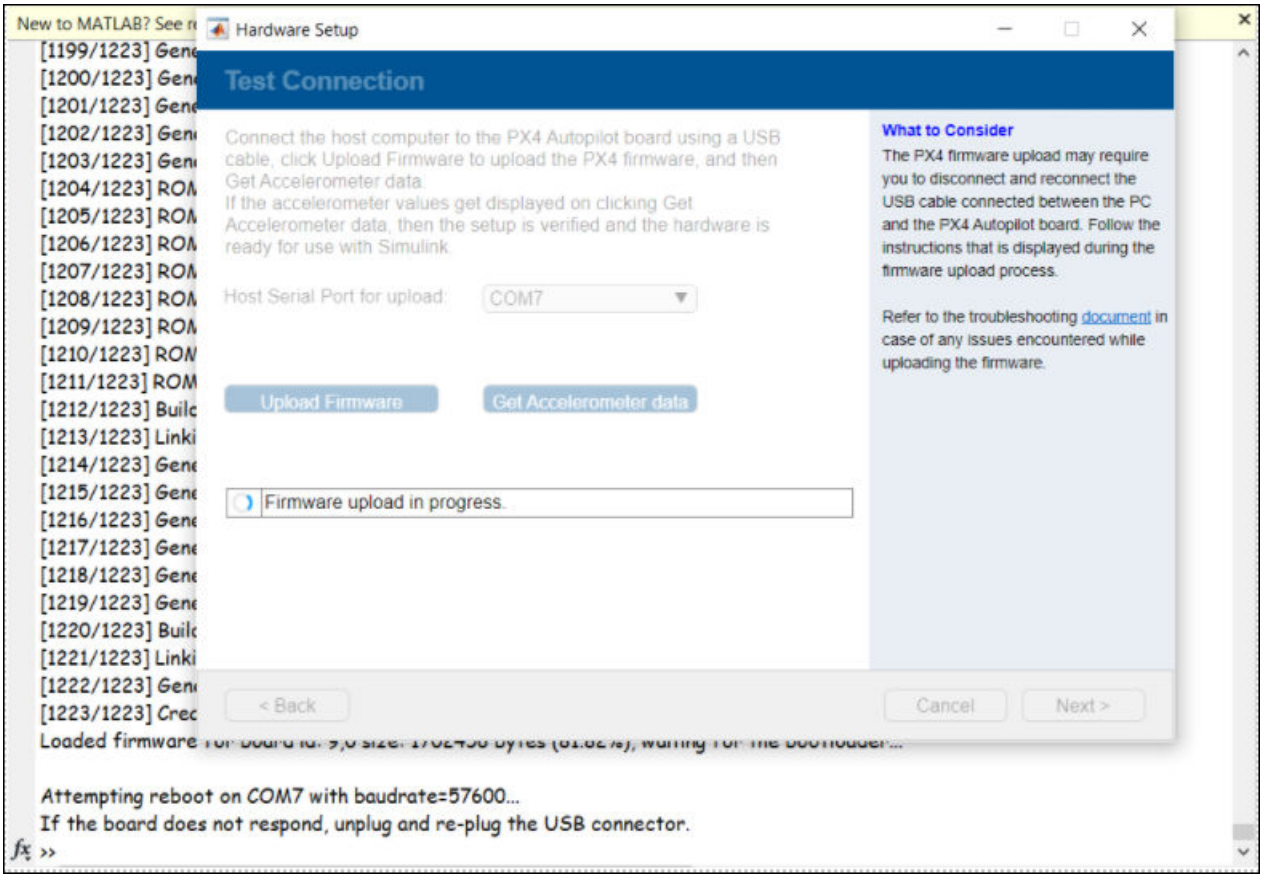
Troubleshooting Unresponsive Firmware Upload

Description

Some Pixhawk boards have different bootloader communications port and main communications port. When you connect to such boards, the bootloader port appears first in the Device Manager. The port shows for a few seconds and then the main communications port appears. A sample GIF image is shown here. The bootloader COM port is needed for firmware upload and the communication port is needed for MAVLink communication.



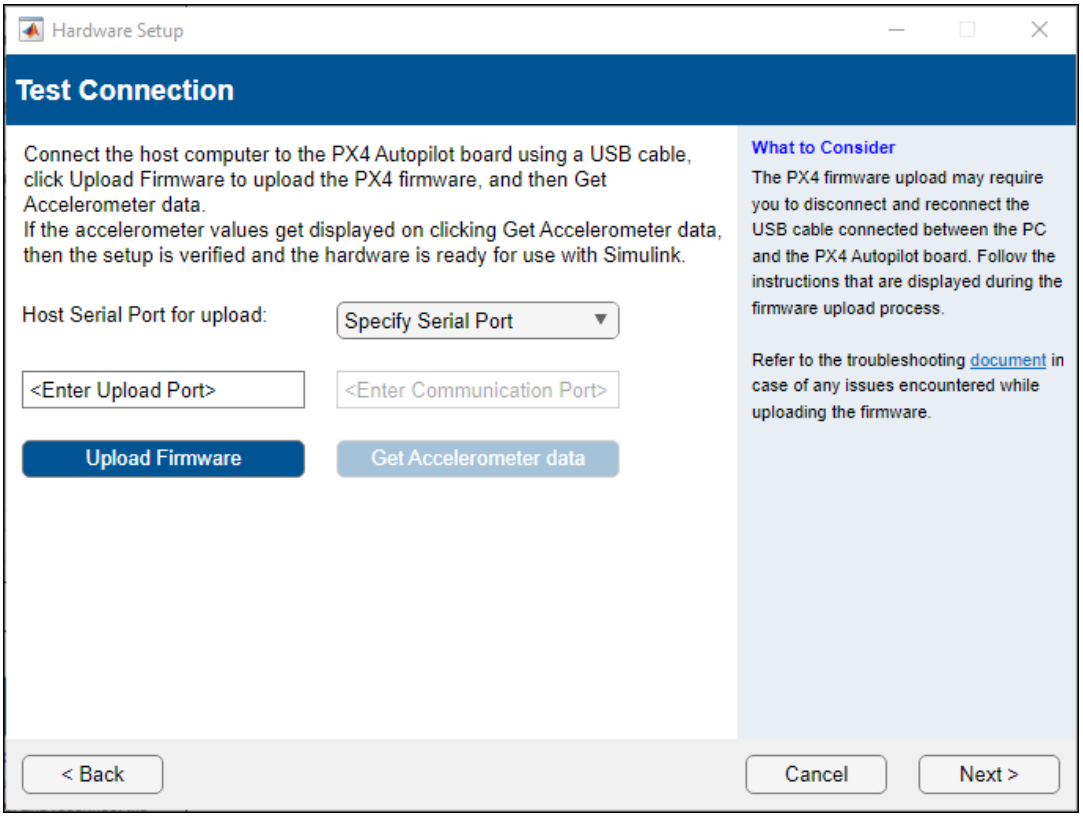
Ensure that you specify the correct COM ports for firmware upload and MAVLink communication. You will be unable to upload the firmware if you specify the communication COM port value instead of bootloader COM port value for firmware upload.



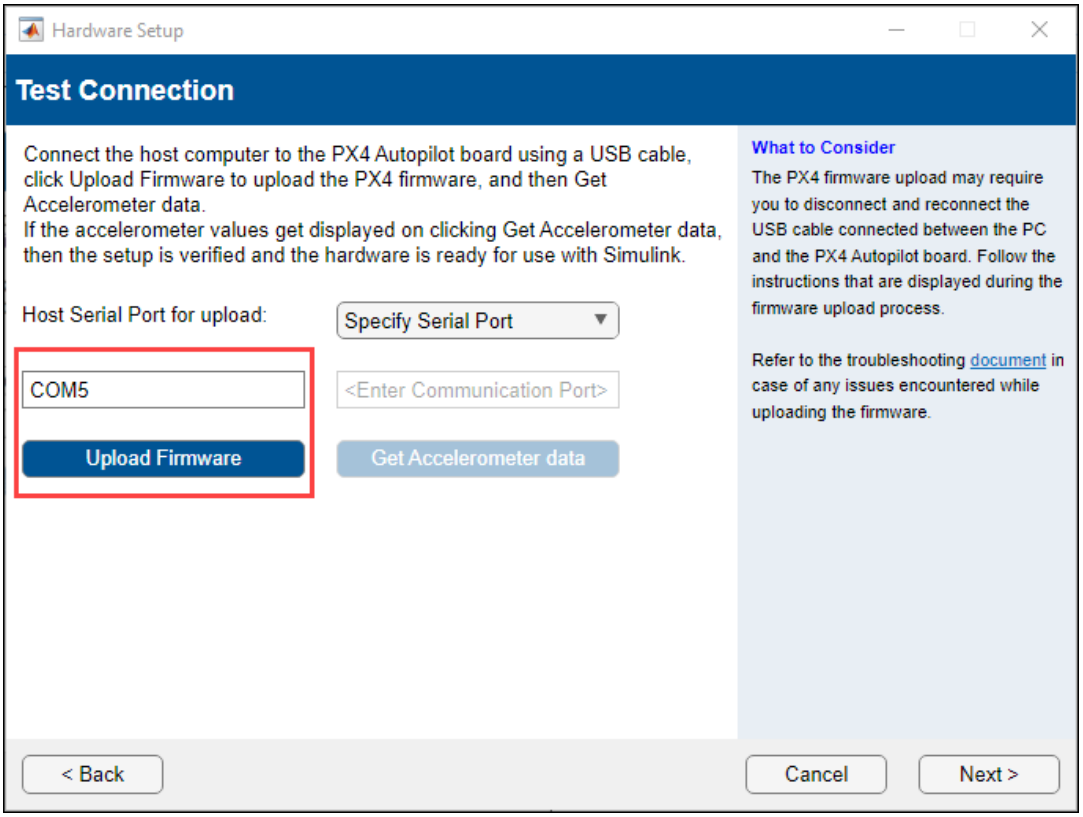
Action

Follow these steps to successfully upload the firmware.

1. Determine the bootloader COM port and the communication COM port value, by performing these steps.
   a. Disconnect the Pixhawk hardware from your computer.
   b. Open the Device Manager.
   c. Connect the Pixhawk hardware to your computer.
   d. The Pixhawk hardware now shows in the Device manager. The corresponding COM Port associated with that, seen in Device Manager is the bootloader COM Port.
   e. Observe that the Pixhawk device disappears automatically from the Device Manager in five seconds. This is because, at this point the device transitions from bootloader to the main PX4 firmware.
   f. Observe that the Pixhawk device reappears in the Device Manager after one or two seconds. Now the main firmware is running and the COM Port you see is the main communications port.
2. In the **Hardware Setup** screen, select `Specify Serial Port` in Test Connection.



3 Enter the bootloader COM port value in the **Enter Serial Port** box.

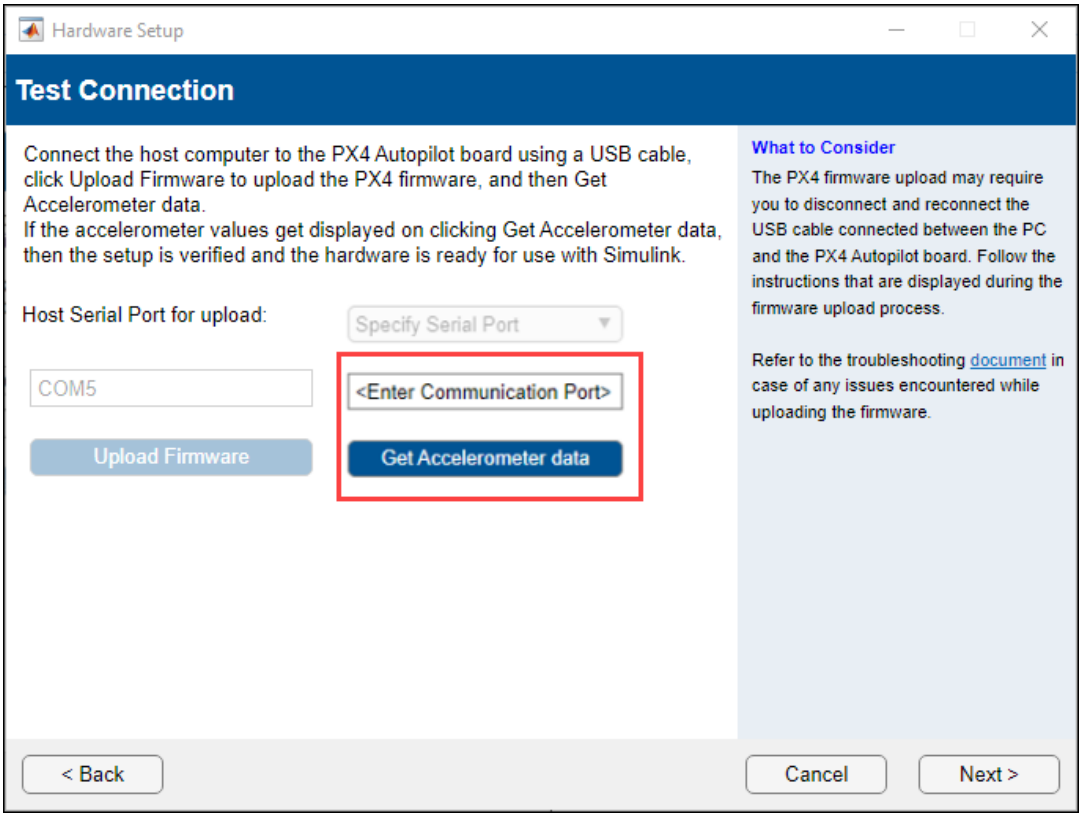


4 Disconnect the Pixhawk board and reconnect it to the host computer.

5 Click **Upload Firmware** immediately after reconnecting the board. The dialog box that appears prompts you to disconnect and reconnect the board. Do that and click **OK** to start uploading the firmware.

   If the upload does not start, try reconnecting the board again.

6 Once you have successfully uploaded the firmware, enter the COM port value in the **Enter Serial Port** box and then click **Get Accelerometer data**. The accelerometer value is displayed and the hardware is now to ready to use with Simulink.



7 Once you have set the COM ports for upload and communication in the Simulink model, deploy the model on the Pixhawk hardware and execute the algorithm on autopilot.

Troubleshooting PX4 Firmware Build Failure While Using createCustomPX4Parameter Function

Description

When you manually change the CMake file and then use the `createCustomPX4Parameter` function to build PX4 firmware through the **Add Parameter Information** dialog box, the PX4 firmware build might stop and display errors.

Action

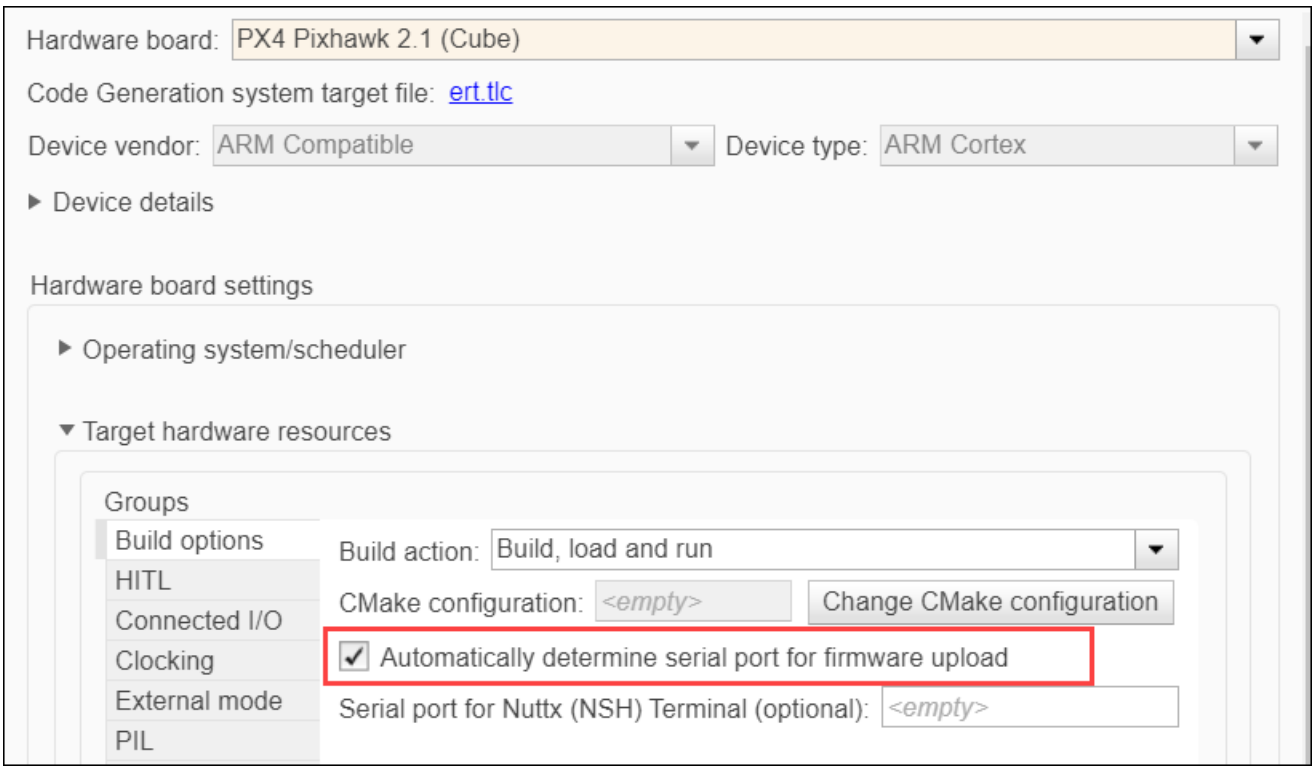
To resolve this issue, perform the hardware setup process again and ensure that you select the correct CMake file in the **Select a PX4 Autopilot and Build Target** hardware setup screen. Then, use the `createCustomPX4Parameter` function.

See Also

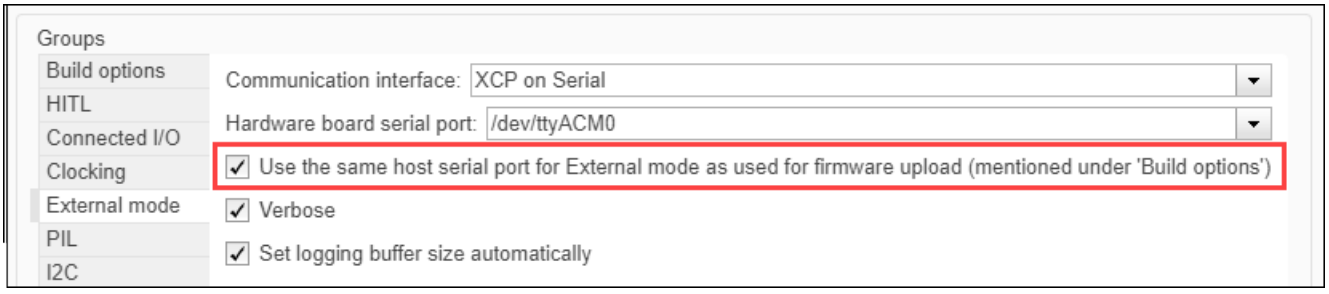
“Troubleshooting Firmware Build Failures” on page 1-48 | “Troubleshooting Unresponsive Firmware Upload” on page 4-43 | “Troubleshooting PX4 Firmware Build Failure Due to Flash Memory Overflow on the Hardware” on page 4-31

Set COM Port for Upload and Communication in Simulink

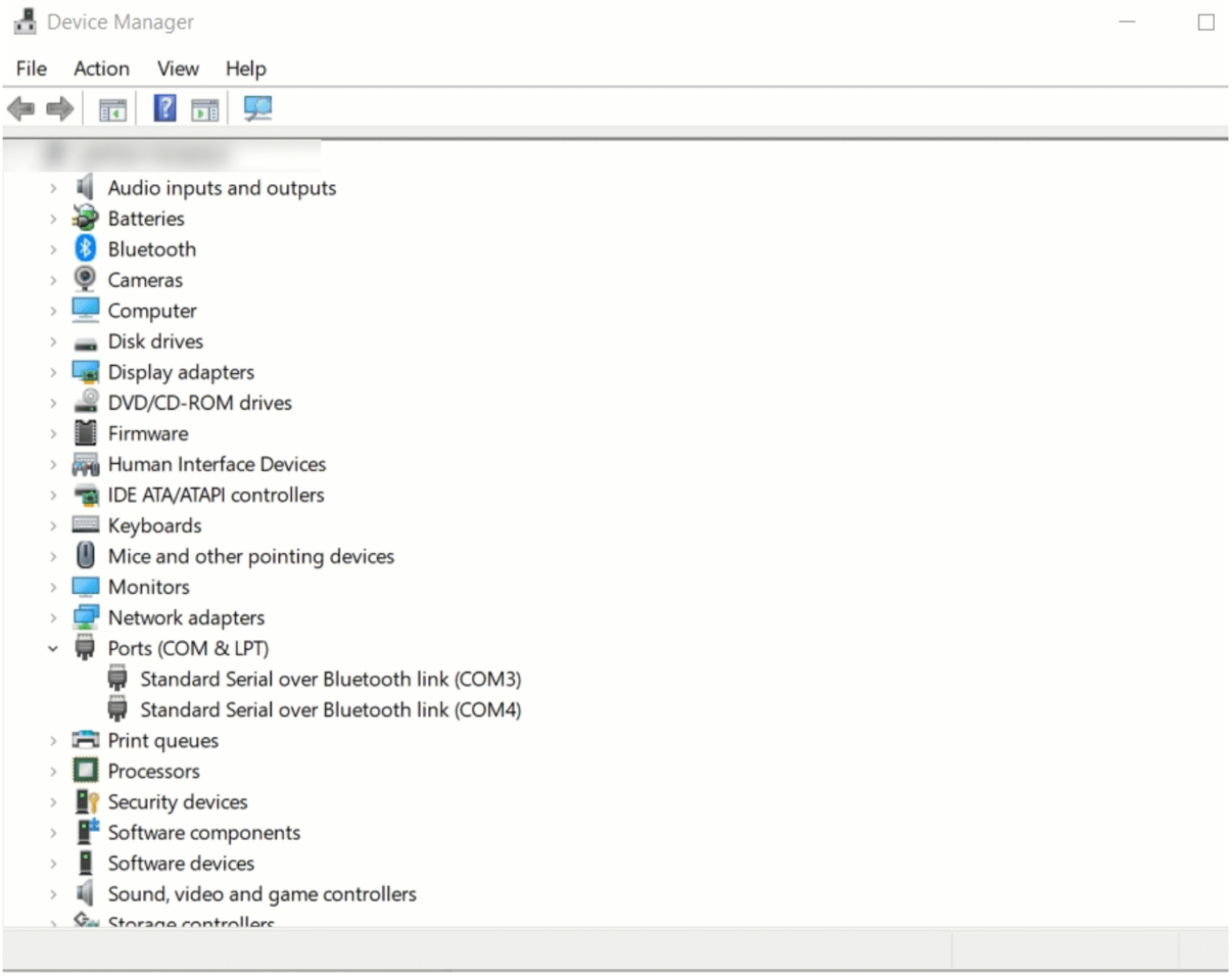
In most of the Pixhawk boards, the COM port required to upload the firmware (bootloader COM port) and the COM port required to establish the connection with computer over MAVLink (Communications COM Port) are the same. You can use the same port to upload the firmware and communicate with the Pixhawk board after the firmware is uploaded. In such cases, you can use the **Automatically determine serial port for firmware upload** option in Simulink Model settings to detect the COM port.



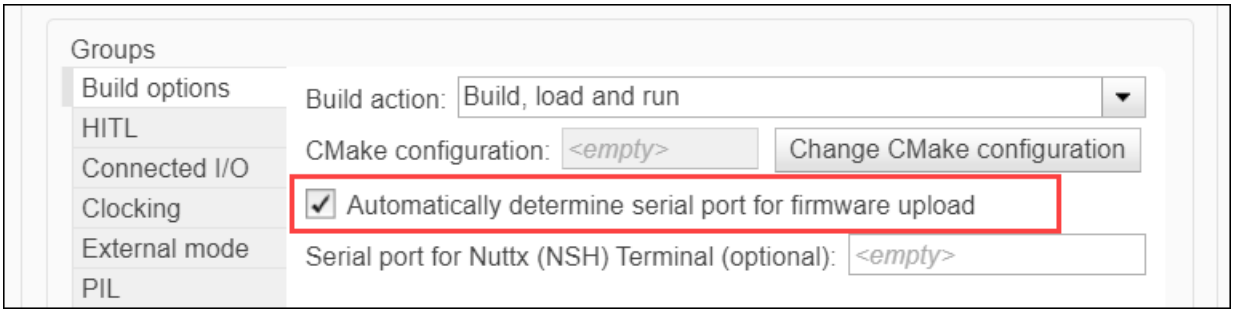
No additional settings are required in External Mode, Connected I/O or PIL tab for the corresponding simulation modes. You can continue to use the same serial port.



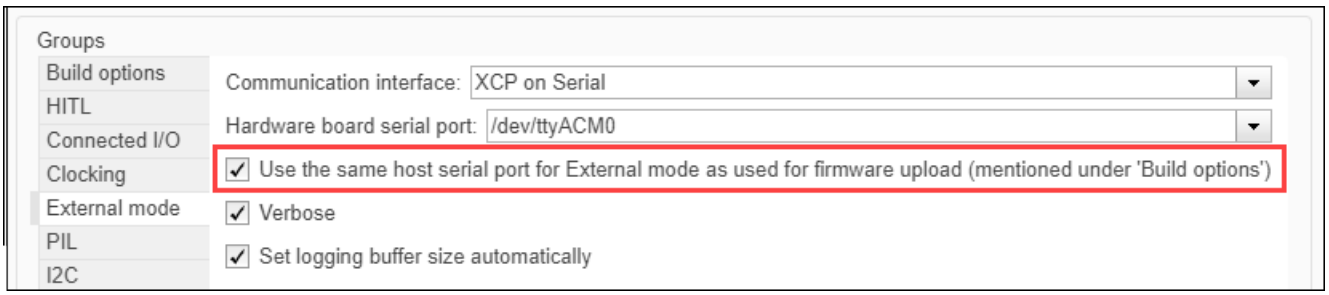
However, for some Pixhawk Series boards, the bootloader COM Port and the main communications COM Port are different. When you connect to such boards, the bootloader port appears first in the Device Manager. The port shows for a few seconds and then the main communications port appears. A sample GIF image is shown here. The bootloader COM port is needed for firmware upload and the communication port is needed for MAVLink communication.



From R2023a, if your Pixhawk board has different bootloader and communication serial ports, enabling the **Automatically determine serial port for firmware upload** option in Simulink Model settings detects the correct bootloader serial port for upload. You do not have to manually enter the upload serial port.



For Pixhawk boards having different Communication serial port and upload port, you can still enable the **Use the same host serial port** option for **External Mode**, **Connected I/O** or **PIL** tabs for the corresponding simulation modes. If you enable this option, the correct serial port for communication is automatically detected even if the Pixhawk hardware has different Communication port other than the upload port.

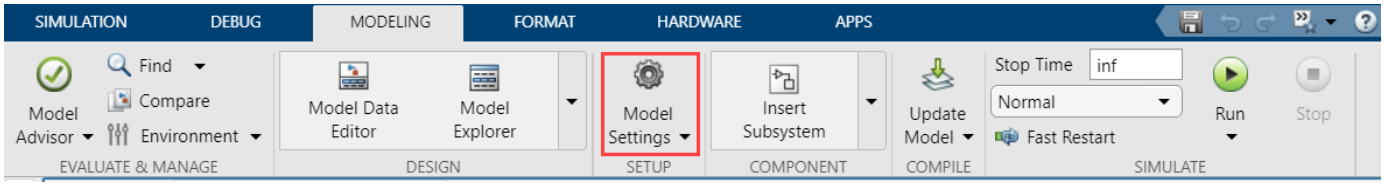


Note In the **PX4 Pixhawk Series** hardware target, the automatic serial port detection is not supported. If you are using **PX4 Pixhawk Series** as the hardware target or if the auto detect option is not working, manually set the upload and communication ports as mentioned in “Manually Set COM Port for Upload and Communication” on page 4-51.

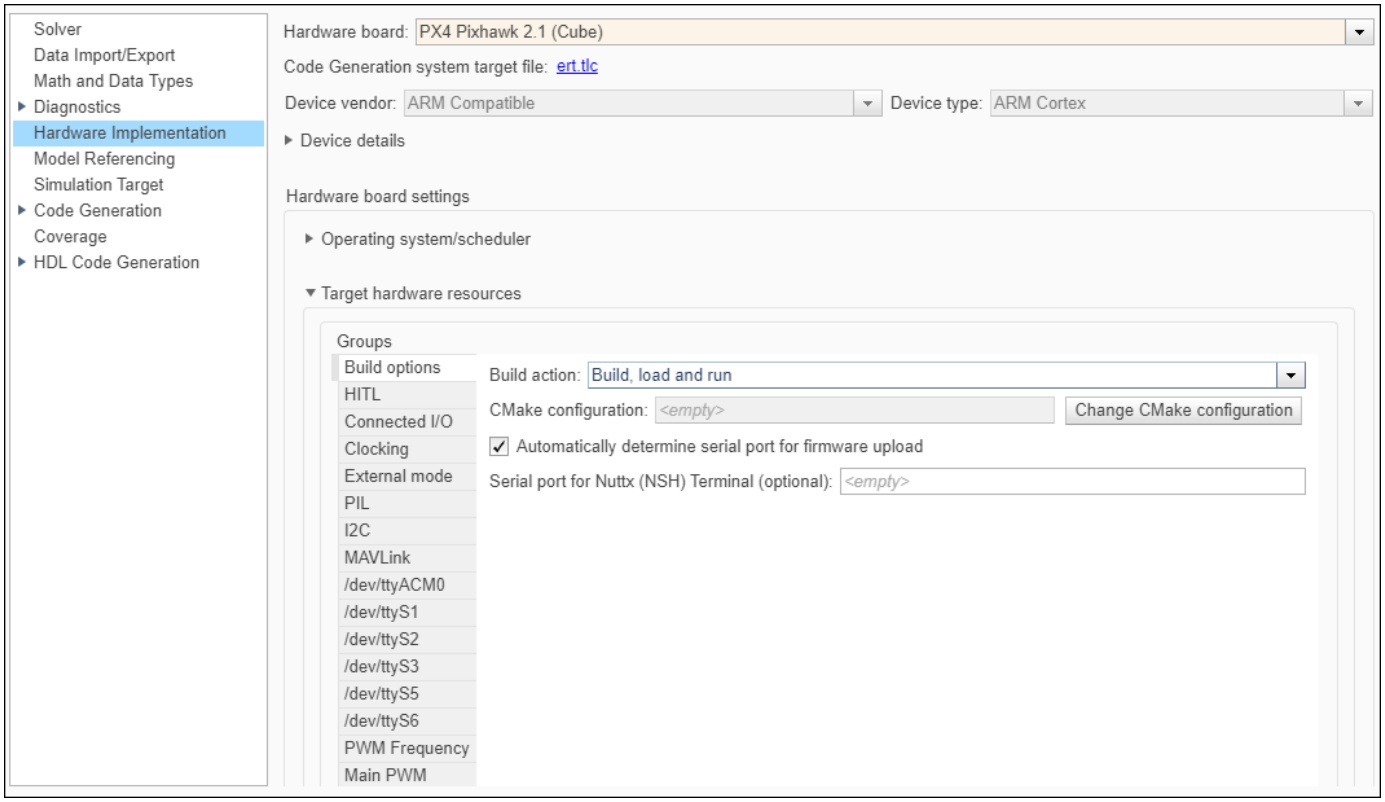
Manually Set COM Port for Upload and Communication

Set Bootloader COM Port for Firmware Upload and Deploy

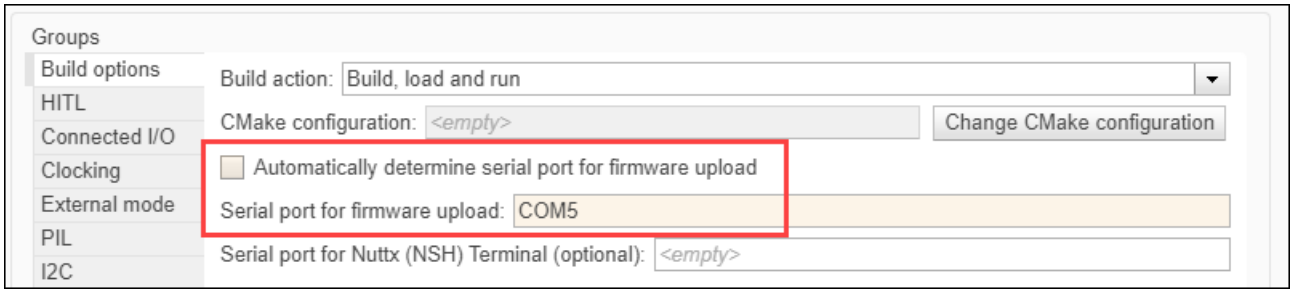
1 For the Simulink model, launch the **Model Settings**.



A sample image of the default settings for hardware implementation is shown here.



2 Clear the **Automatically determine serial port for firmware upload** option and enter the bootloader serial port in **Serial Port for firmware upload** (This should be the bootloader port).

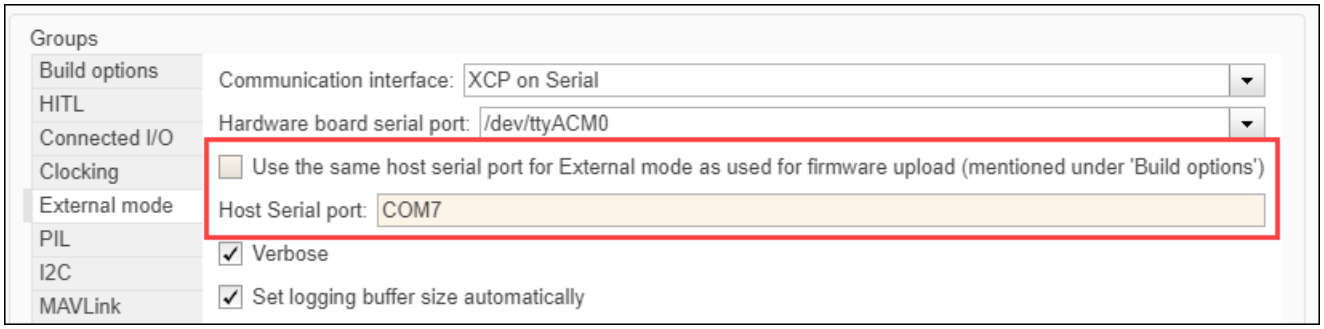


3 Deploy the Simulink model. On the **Hardware** tab, in the **Mode** section, select **Run on board** and then click **Build, Deploy & Start**.

Set Communication COM Port for Monitor & Tune (External Mode)

1 Perform the settings for Build options as shown in the topic “Set Bootloader COM Port for Firmware Upload and Deploy” on page 4-51.

2 In **External mode** tab, clear the **Use the same host serial port for External Mode as used for firmware upload** option and mention the main communications port in **Host Serial Port** (this must be different from the bootloader port you mentioned earlier in **Build options**).

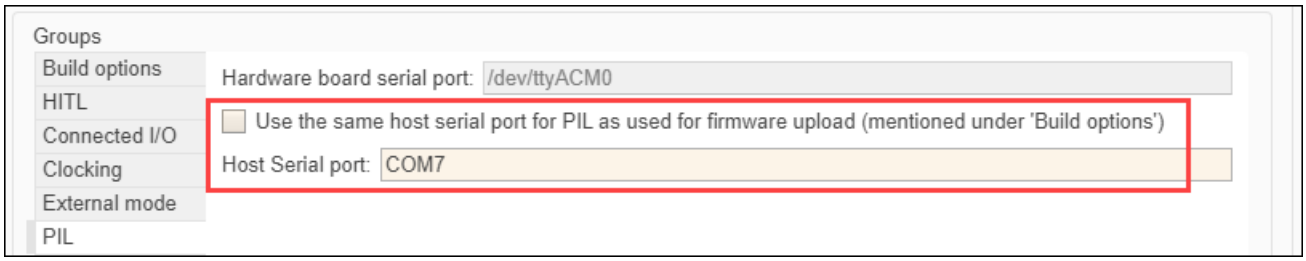


3 Run External Mode for the model.

Set Communication COM Port for PIL

1 Perform the settings for Build options as shown in the topic “Set Bootloader COM Port for Firmware Upload and Deploy” on page 4-51.

2 In **PIL** Tab, clear the **Use the same host serial port for PIL as used for firmware upload** option and mention the main communications port in **Host Serial port** (this must be different from the bootloader port you mentioned earlier in **Build options**).



Set Communication COM Port for Connected IO

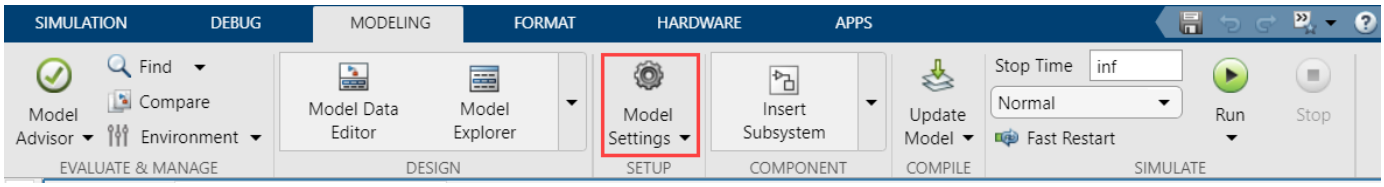
1 Perform the settings for Build options as shown in the topic “Set Bootloader COM Port for Firmware Upload and Deploy” on page 4-51.

2 In **Connected I/O** tab, clear the **Use the same host serial port for Connected I/O as used for firmware upload** option and mention the main communications port in **Host Serial Port** (this must be different than the bootloader port you mentioned earlier in **Build options**).

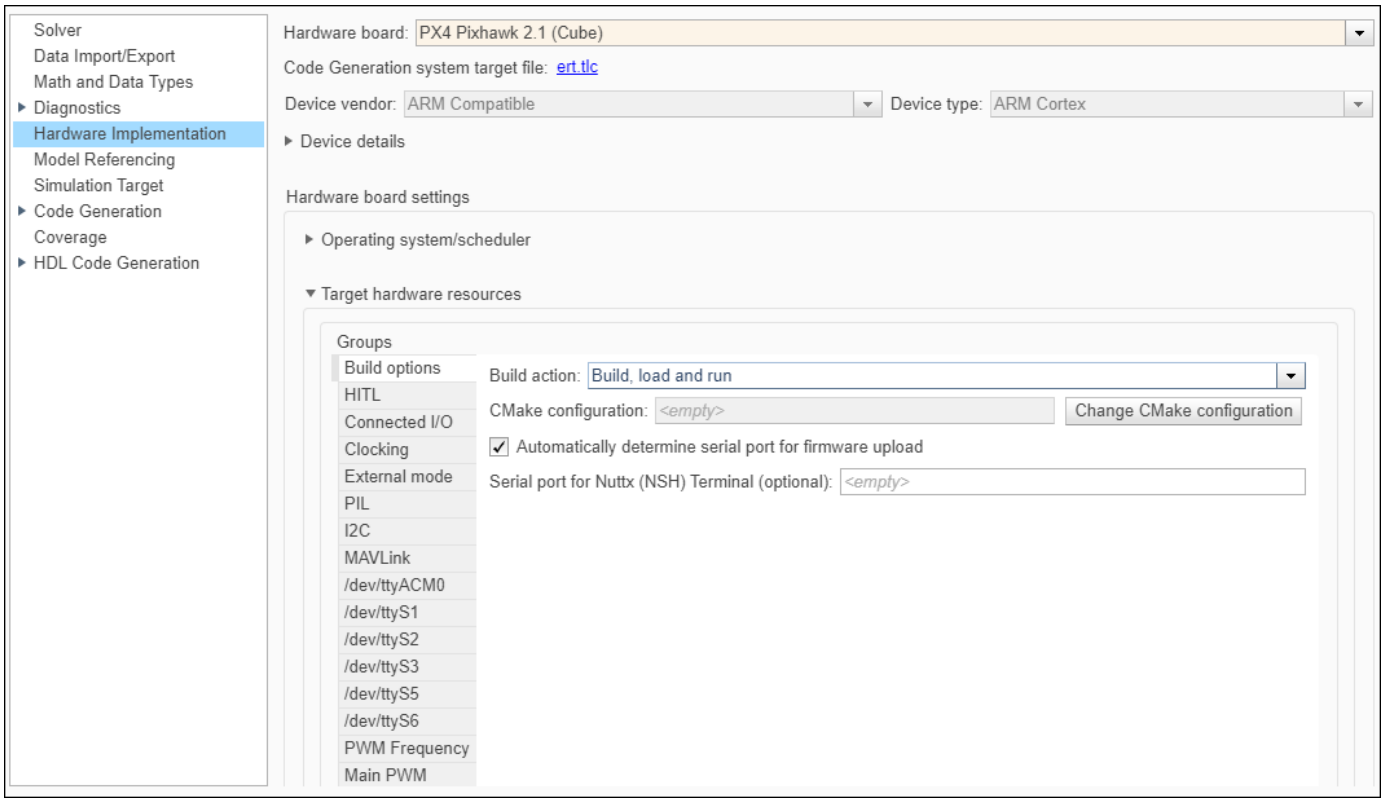
Manually Set COM Port for Upload and Communication

Set Bootloader COM Port for Firmware Upload and Deploy

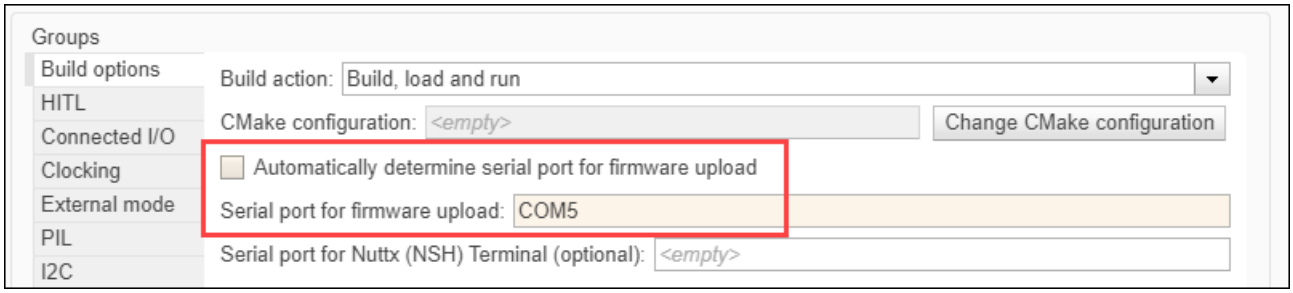
1 For the Simulink model, launch the **Model Settings**.



A sample image of the default settings for hardware implementation is shown here.



2 Clear the **Automatically determine serial port for firmware upload** option and enter the bootloader serial port in **Serial Port for firmware upload** (This should be the bootloader port).

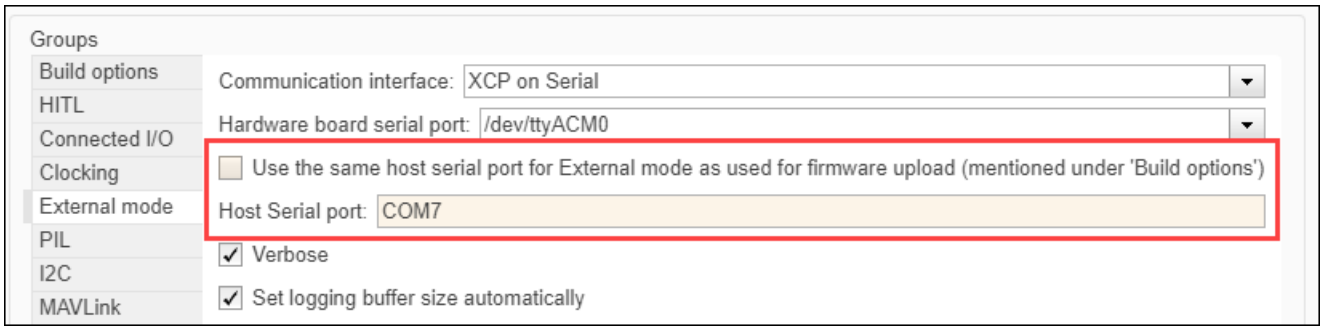


3 Deploy the Simulink model. On the **Hardware** tab, in the **Mode** section, select **Run on board** and then click **Build, Deploy & Start**.

Set Communication COM Port for Monitor & Tune (External Mode)

1 Perform the settings for Build options as shown in the topic “Set Bootloader COM Port for Firmware Upload and Deploy” on page 4-51.

2 In **External mode** tab, clear the **Use the same host serial port for External Mode as used for firmware upload** option and mention the main communications port in **Host Serial Port** (this must be different from the bootloader port you mentioned earlier in **Build options**).

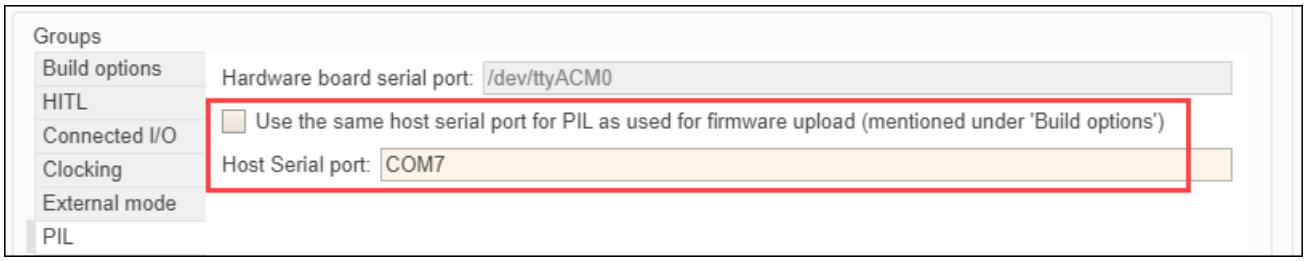


3 Run External Mode for the model.

Set Communication COM Port for PIL

1 Perform the settings for Build options as shown in the topic “Set Bootloader COM Port for Firmware Upload and Deploy” on page 4-51.

2 In **PIL** Tab, clear the **Use the same host serial port for PIL as used for firmware upload** option and mention the main communications port in **Host Serial port** (this must be different from the bootloader port you mentioned earlier in **Build options**).



Set Communication COM Port for Connected IO

1 Perform the settings for Build options as shown in the topic “Set Bootloader COM Port for Firmware Upload and Deploy” on page 4-51.

2 In **Connected I/O** tab, clear the **Use the same host serial port for Connected I/O as used for firmware upload** option and mention the main communications port in **Host Serial Port** (this must be different than the bootloader port you mentioned earlier in **Build options**).


Groups
Build options
HITL
Connected I/O
Clocking
External mode
Hardware board serial port: /dev/ttyACM0
Use the same host serial port for Connected I/O as used for firmware upload (mentioned under 'Build options')
Host Serial port: COM7

Code Execution Profiling on PX4 Targets

Sample times you specify in a Simulink model determine the time schedule for running generated code on target hardware. With enough computing power on the hardware, the code runs in real-time according to the specified sample times. With real-time execution profiling, you can check if the generated code meets your real-time performance requirements.

You can use code execution profiling results to enhance the design of your system. For example, if the code easily meets the real-time requirements, you can consider adding more functionality to your system to exploit available processing power. If the code does not meet real-time requirements, you can look for ways to reduce execution time. For example, you can identify the tasks that require the most time and then investigate whether trade-off between functionality and speed is possible.

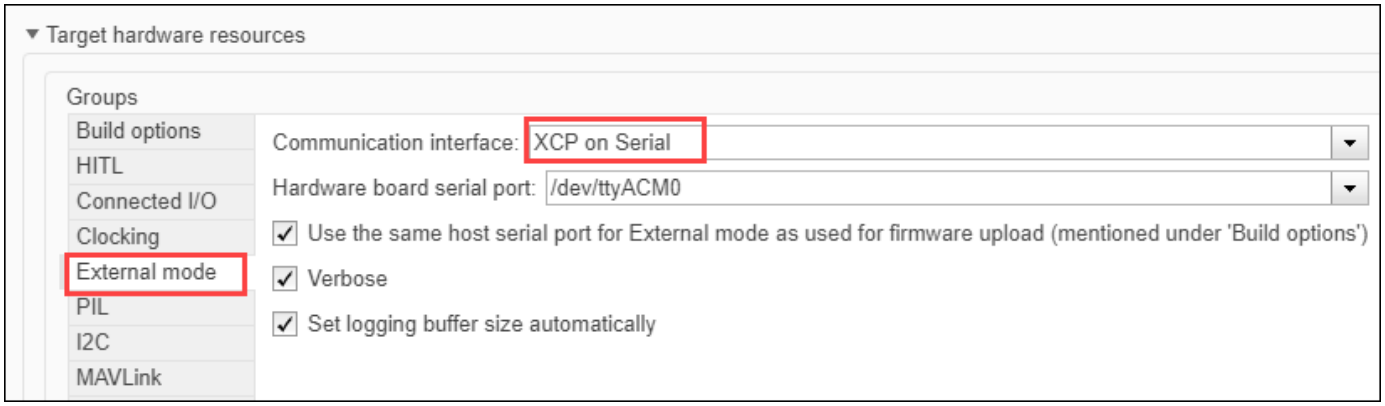
This example introduces a workflow for real-time code execution profiling by showing you how to:

- Configure the model for code execution profiling, and generate code.
- Run generated code on target hardware.
- Analyze performance through code execution profiling plots and reports.

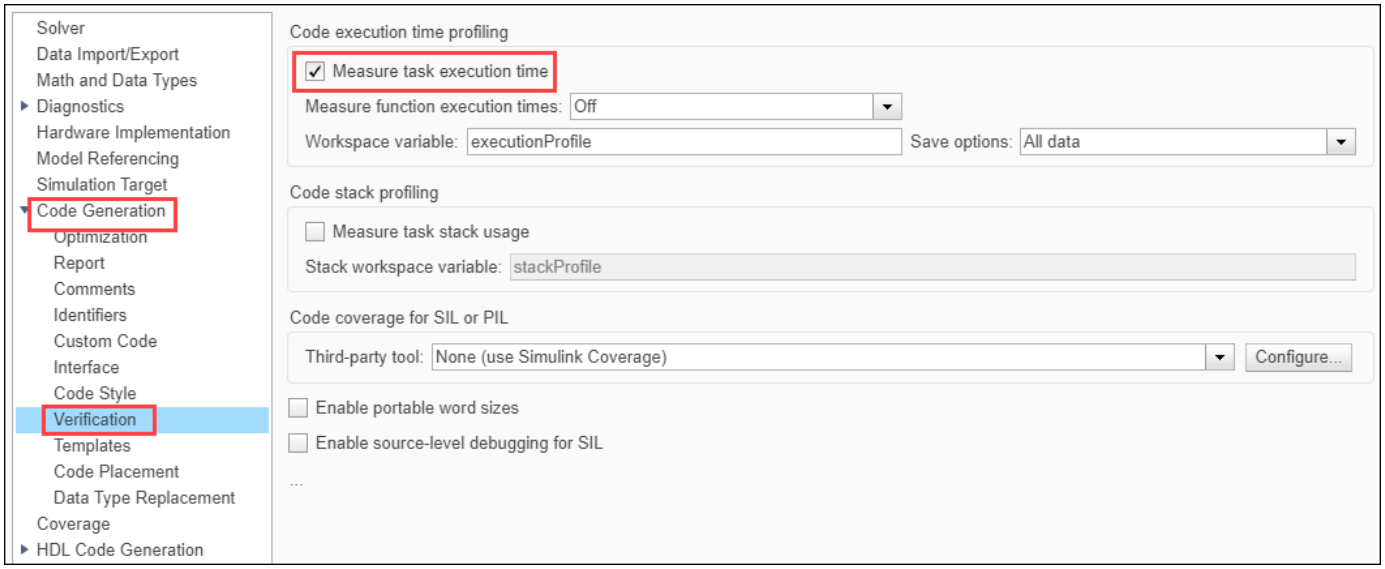
Profiling with XCP External Mode

Real time profiling data can be obtained by using the XCP External Mode infrastructure. To configure a Simulink model for real-time profiling perform these steps:

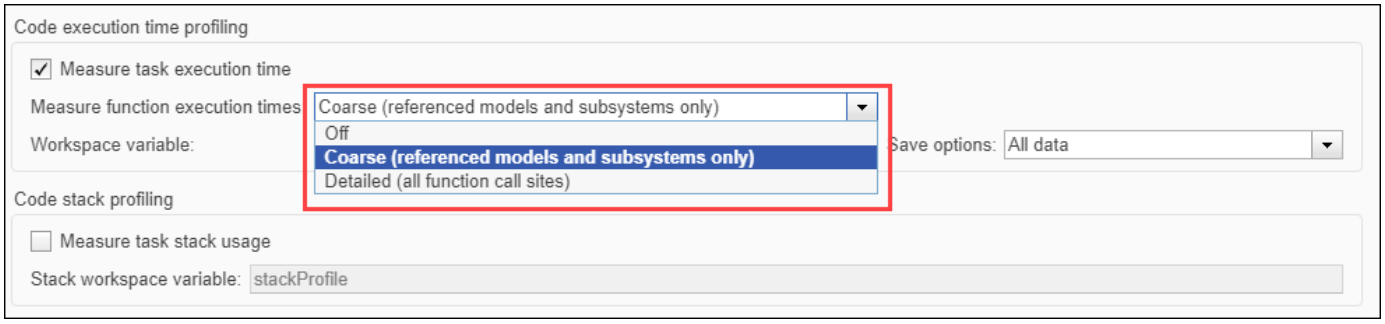
1 In the Simulink Editor, select **Modeling > Model Configuration**. In the **Configuration Parameter** dialog box, click **External mode**.



2 Navigate to **Code generation > Verification** and select **Measure task execution time**.



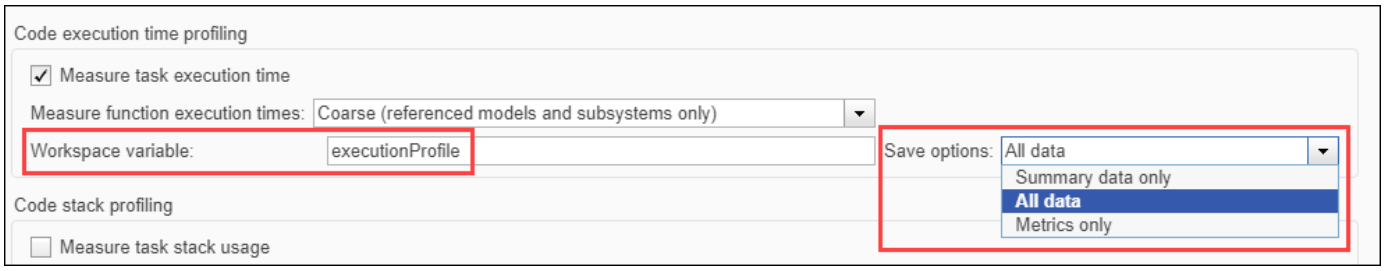
3 Select the required option for **Measure function execution time**.



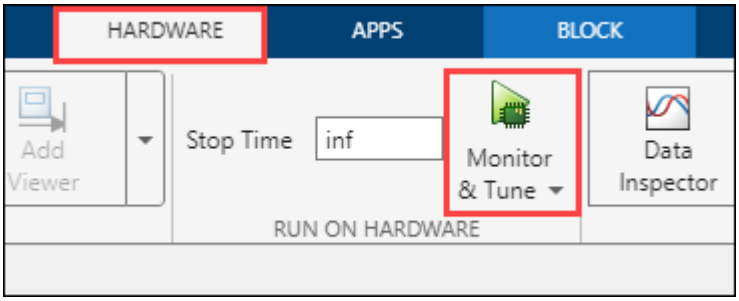
- `Off` – Select this option to disable Profiling. Only Task profiling is available in this option.
- `Coarse (referenced models and subsystems only)` – Select this option to analyse generated function code for the main model components.
- `Detailed (all function call sites)` – Select this option to analyse generated function code for all blocks in the model

4 Enter the required value for **Workspace variable**. It is the variable in the MATLAB workspace used for storing data received from the target.

5 Select the required option for **Save options**. For help on selecting the save options, see “Save Options” on page 4-59.



6 Click **Monitor & Tune** from the **Hardware** tab of Simulink toolstrip to generate the profiling report.



After the simulation ends, a profiling report is generated with profiling metrics of different tasks/functions that are being profiled. For more information, see “Code Execution Profiling on PX4 Target in Monitor & Tune Simulation”.

For information on code execution profiling with SIL and PIL, see “Create Execution-Time Profile for Generated Code” (Embedded Coder).

Save Options

Save options are used to select the required type of report. The following table explains the differences between save options.

	Summary data only	All data	Metrics only
Real-time data	Available	Available	Not available. Target sends profiling data only at the end of the simulation.
Host memory requirement	This option requires less memory as the host stores only summary metrics of Profiling Data. For example, 11 KB data for a model running for 50 seconds.	This option requires large memory as the host stores all the data sent by the target . For example, 1500 KB data for a model running for 50 seconds.	This option requires less memory as the host stores only the metrics data sent by the target. For example, 12 KB of data for a model running for 50 seconds.
SDI streaming	Available	Available	Not available
Bandwidth requirement	Requires additional bandwidth.	Requires additional bandwidth.	Does not require additional bandwidth.

Selecting save options

This section helps you to select the recommended save options in different scenarios.

- `All data` – Select this option, if the host has enough memory and the target has the required bandwidth to stream data.
- `Summary data Only` – Select this option, if the simulation is running for a long time and host does not have a lot of memory.
- `Metrics Only` – Select this option, if the target does not have enough time/bandwidth to stream profiling data.

Troubleshooting

Data Drop in Signal Logging or Code Execution Profiling

Description

Data drops can occur either in signal logging or profiling.

Action

Both Signal Logging and Profiling data streaming use the same communication channel to send data from the target. As channel bandwidth is limited, there could be data drops at high sample rates. This issue can be mitigated by streaming only the data you need. If only the profiling data is required, disable signal logging by clearing all the check boxes in **Configuration Parameters > Data Import/Export > Save to workspace or file**.

See Also

More About

- “Code Execution Profiling on PX4 Target in Monitor & Tune Simulation”

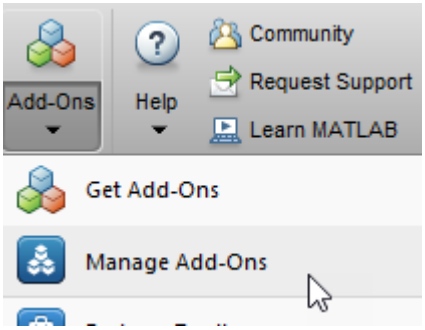
Deployment on Cube Orange Autopilot from Simulink

This topic helps you to get started with Cube Orange Autopilot. Before starting with Simulink, ensure that you install the version 4.3.0 of QGroundControl and able to upload latest stable version of PX4 Firmware from QGroundControl. If you are facing issue with USB connection, It is recommended to install Mission planner and do a clean re-installation of latest drivers as described in this page.

After you successfully upload the PX4 firmware from QGroundControl, get started with UAV Toolbox Support Package for PX4 Autopilots.

Select PX4 Cube Orange in Hardware Setup

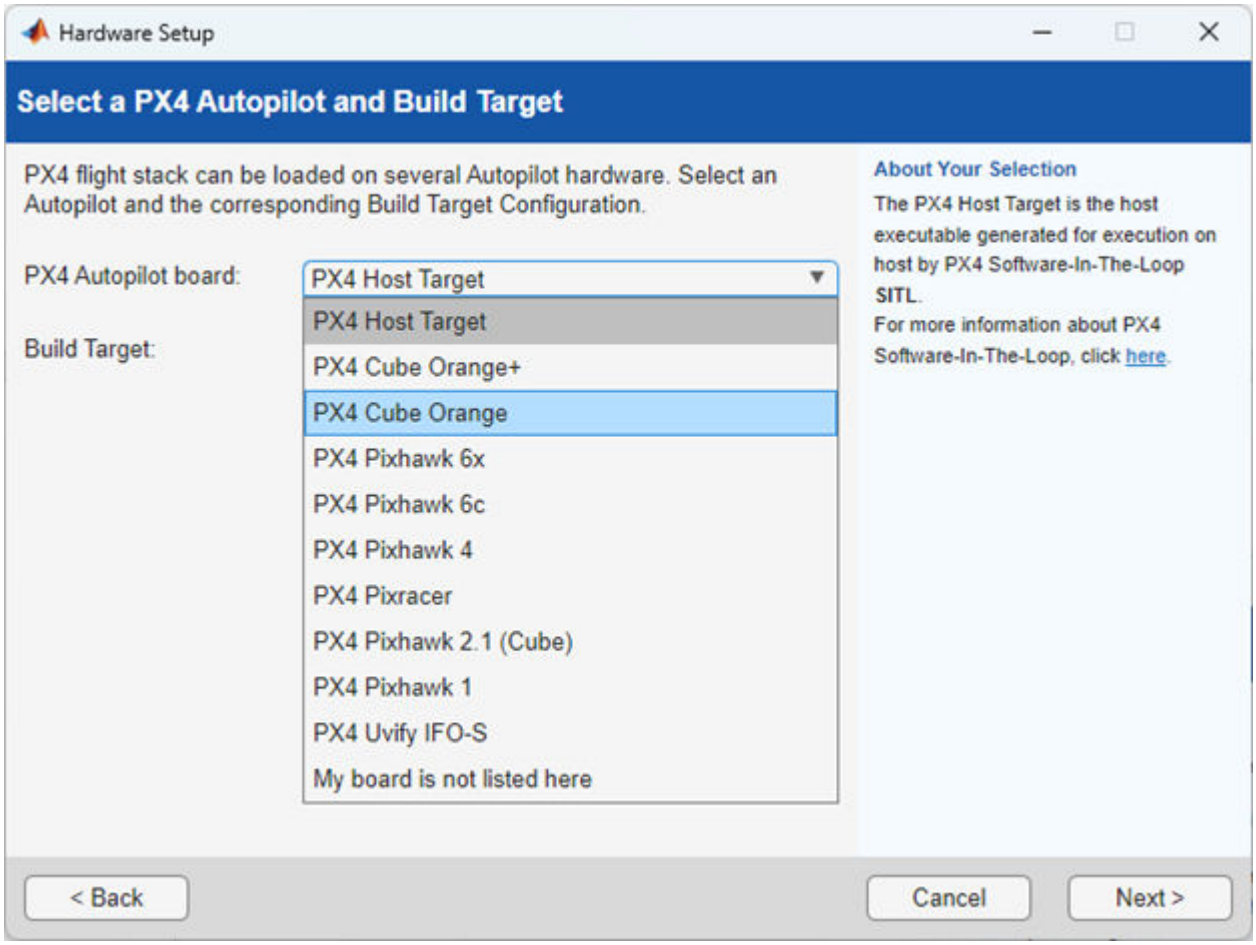
1. If the support package is already installed, start the hardware setup by opening the Add-On Manager.



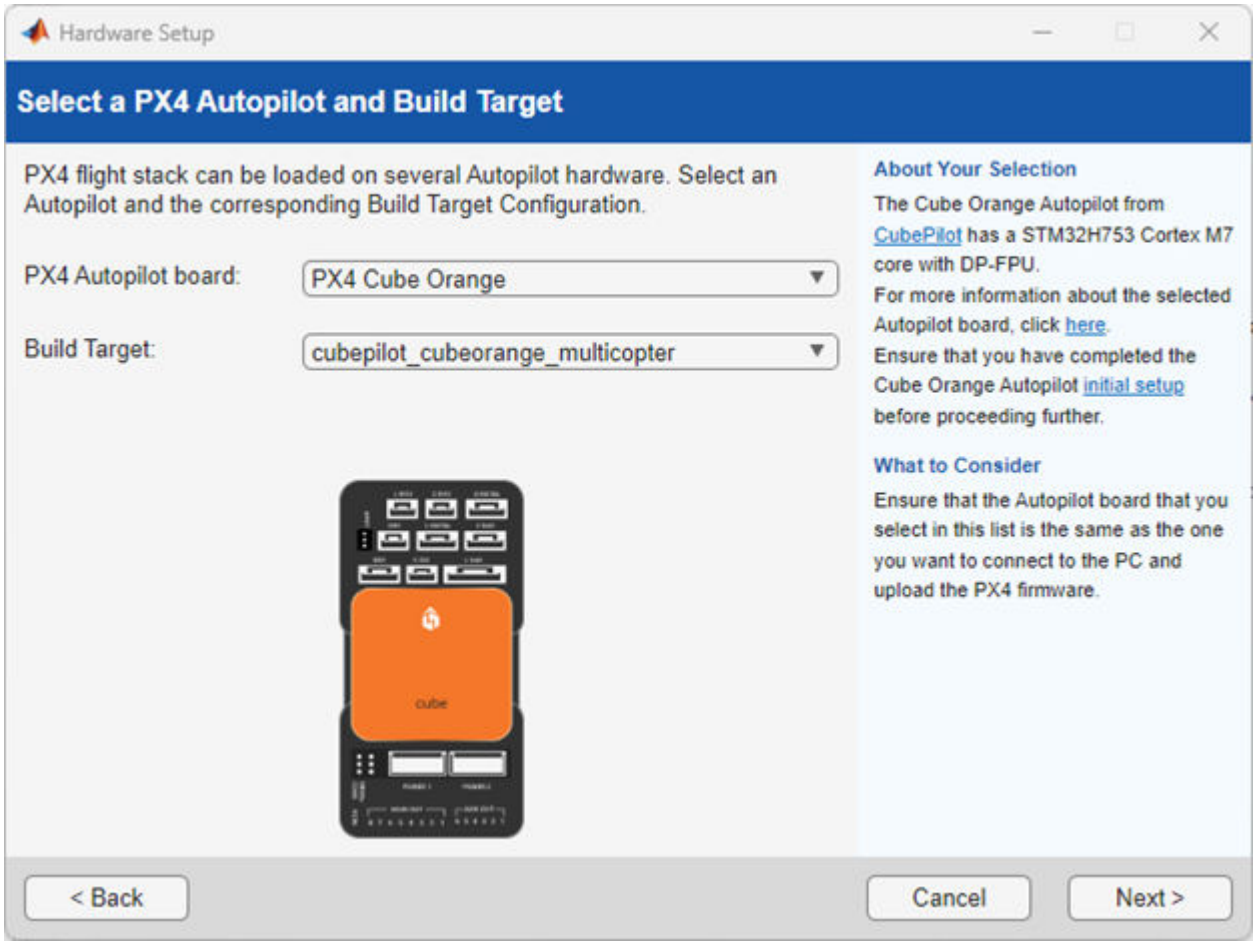
2. In the Add-On Manager, start the hardware setup process by clicking the **Setup** button,  .

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

3. In the **Select a PX4 Autopilot and Build Target** screen, select `PX4 Cube Orange`.

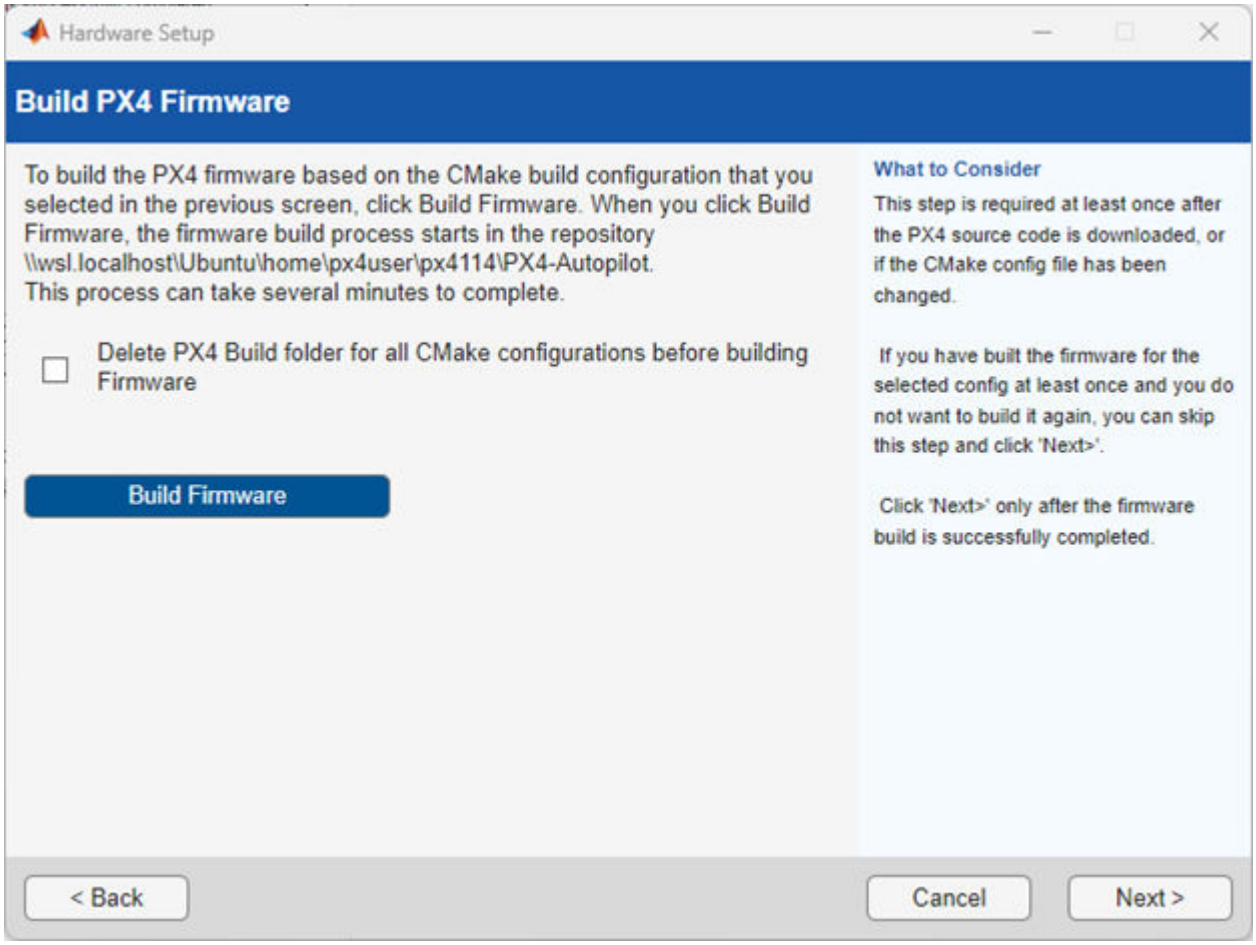


4 Select the corresponding CMake Build target (`cubepilot_cubeorange_multicopter`) from **Build Target** drop-down list.



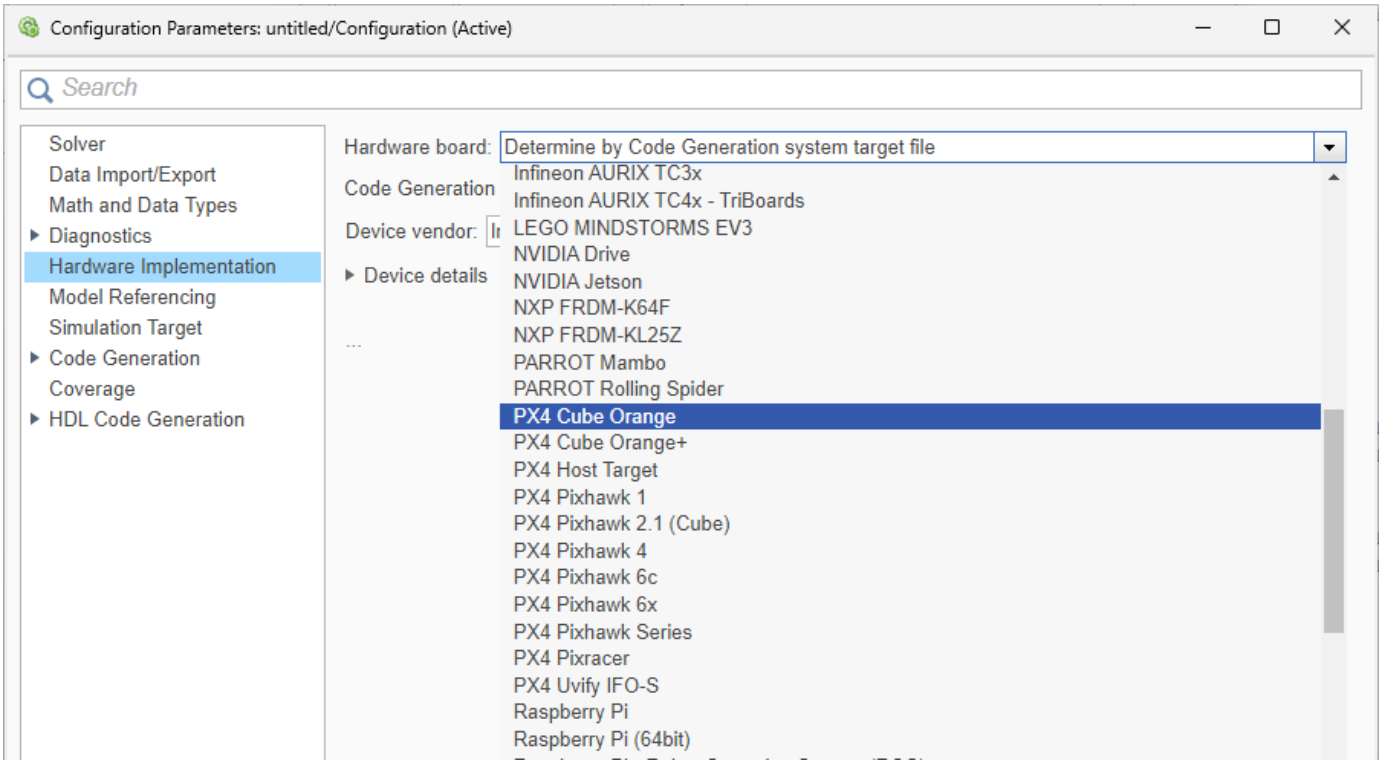
5 Click **Next** and continue with onscreen instructions.

6 Click **Build PX4 Firmware** in the **Build PX4 Firware** screen to complete the setup.

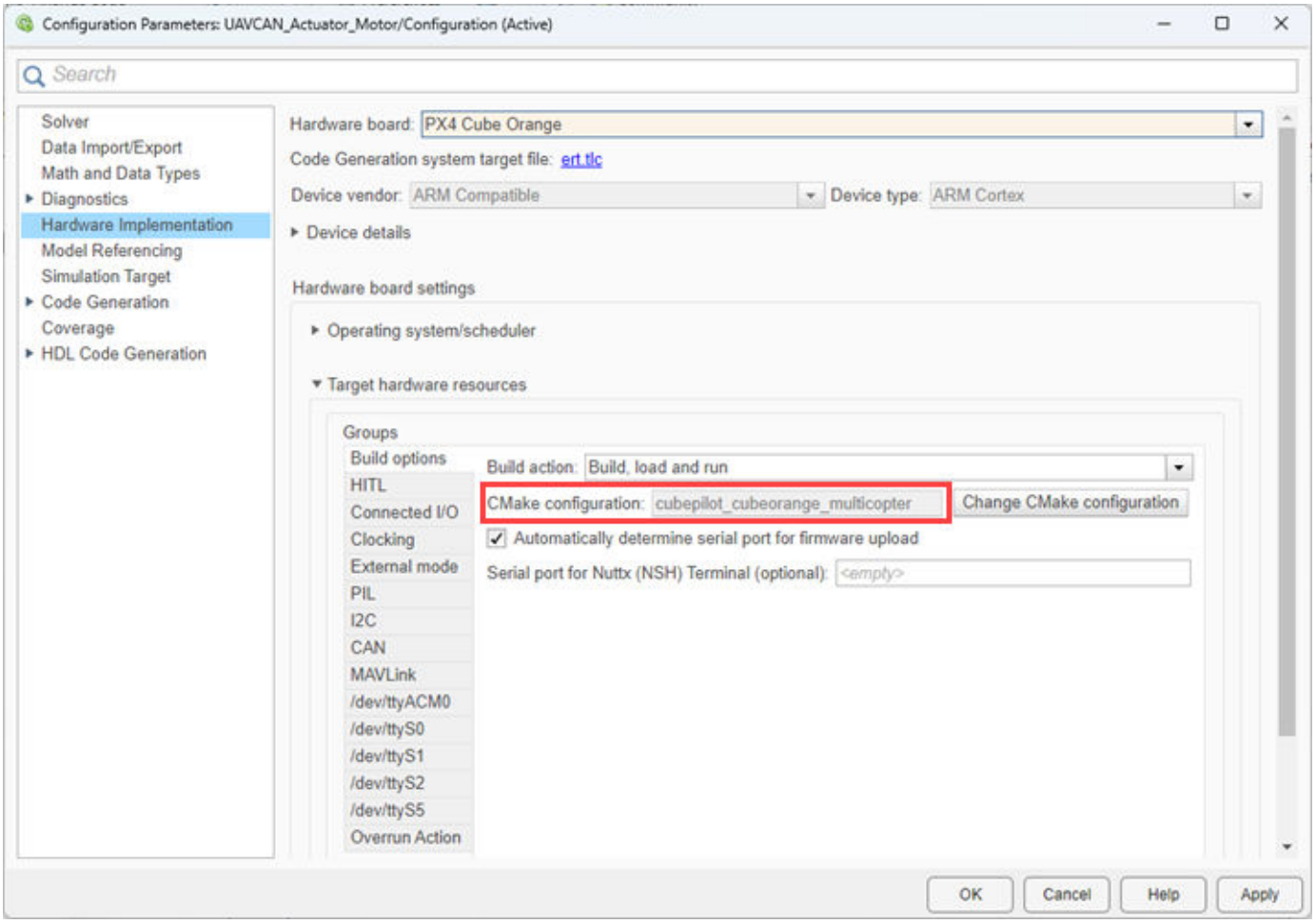


Select PX4 Cube Orange as the Simulink Target Hardware

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



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



Note If you are using Cube Orange autopilot, then use `GPS2` for monitor and tune communication over non-USB port connection, as there might be issues using `TELEM1` or `TELEM2` ports.

Deployment on Cube Blue H7 Autopilot from Simulink

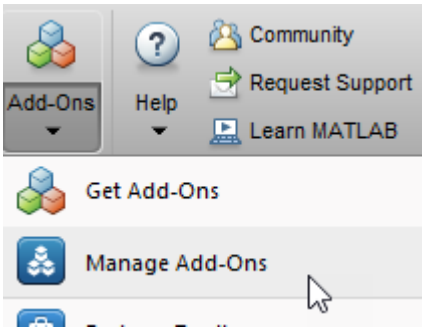
This topic helps you to get started with Cube Blue H7 Autopilot. Before starting Simulink, install the version 4.3.0 of QGroundControl and upload latest stable version of PX4 Firmware from QGroundControl. If you are unable to create a USB connection, install Mission planner and install the latest drivers as described here.

After you successfully upload the PX4 firmware from QGroundControl, get started with UAV Toolbox Support Package for PX4 Autopilots.

Once the hardware setup is completed and the PX4 Firmware is successfully built for the selected build target, you can use your Cube Blue H7 autopilot in Simulink by selecting PX4 Cube Blue H7 as the hardware board in the Simulink model configuration settings.

Select PX4 Cube Blue H7 in Hardware Setup

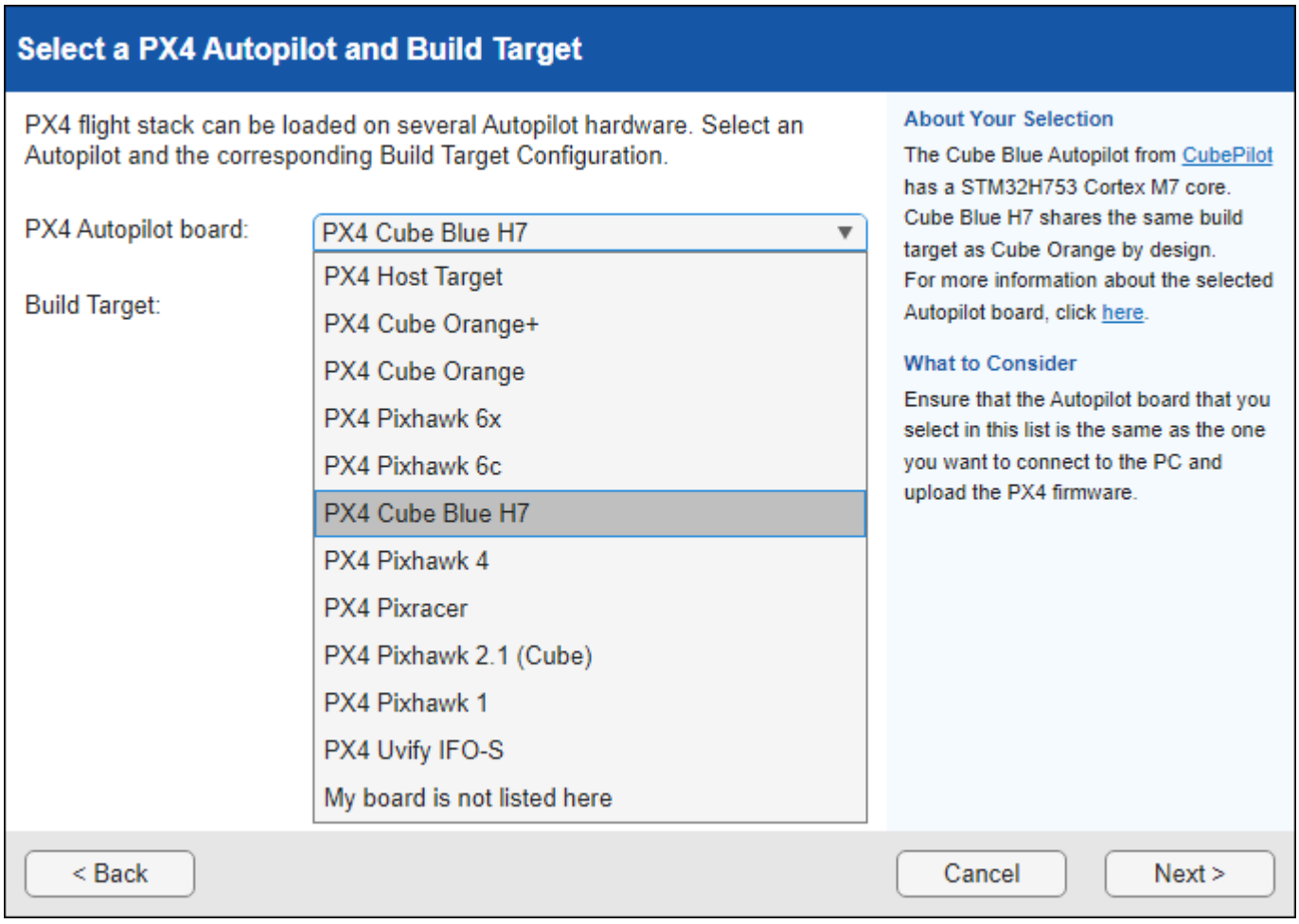
1. If the support package is already installed, start the hardware setup by opening the Add-On Manager.



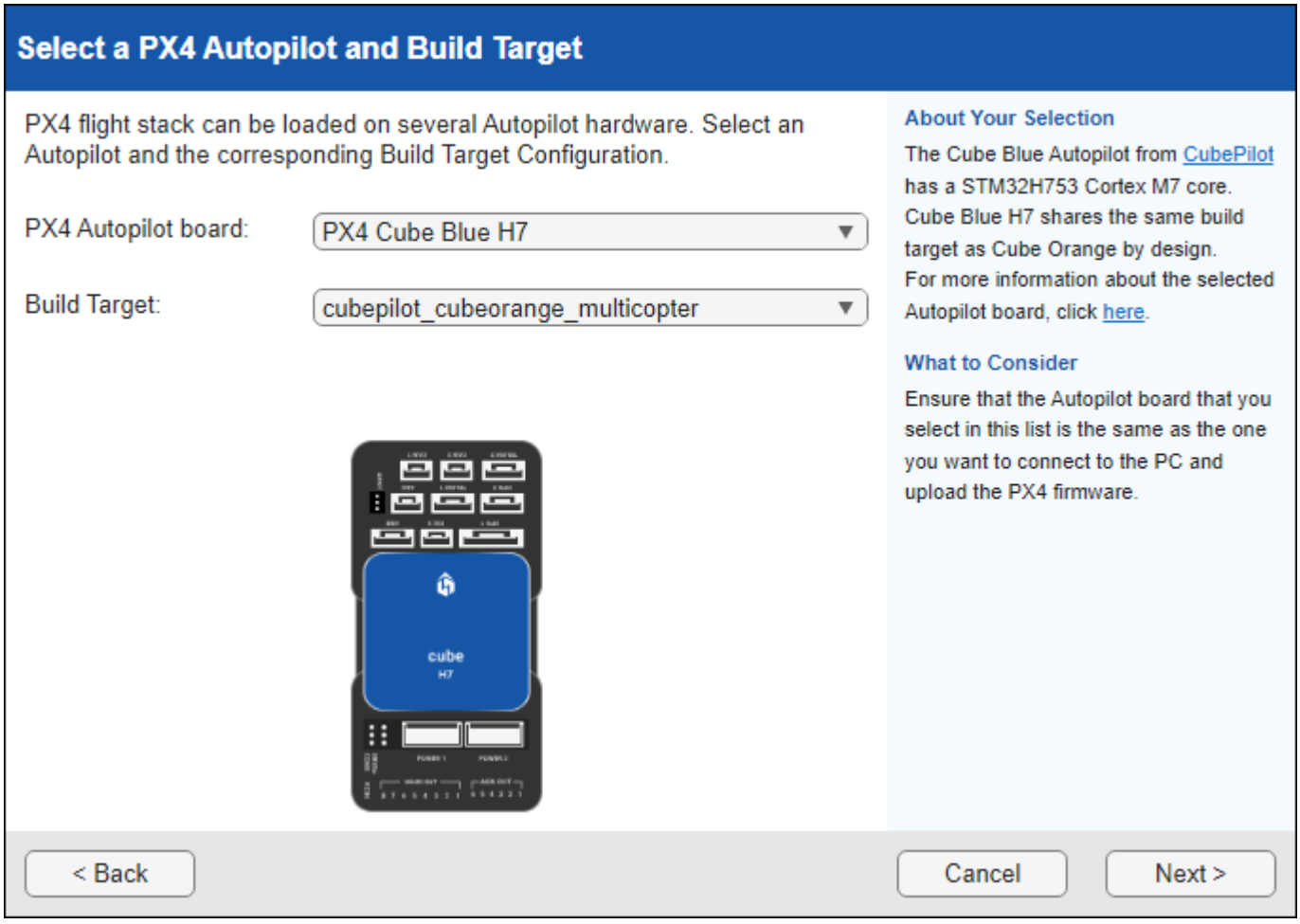
2. In the Add-On Manager, start the hardware setup process by clicking the **Setup** button  .

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

3. On the **Select a PX4 Autopilot and Build Target** page, select `PX4 Cube Blue H7`.

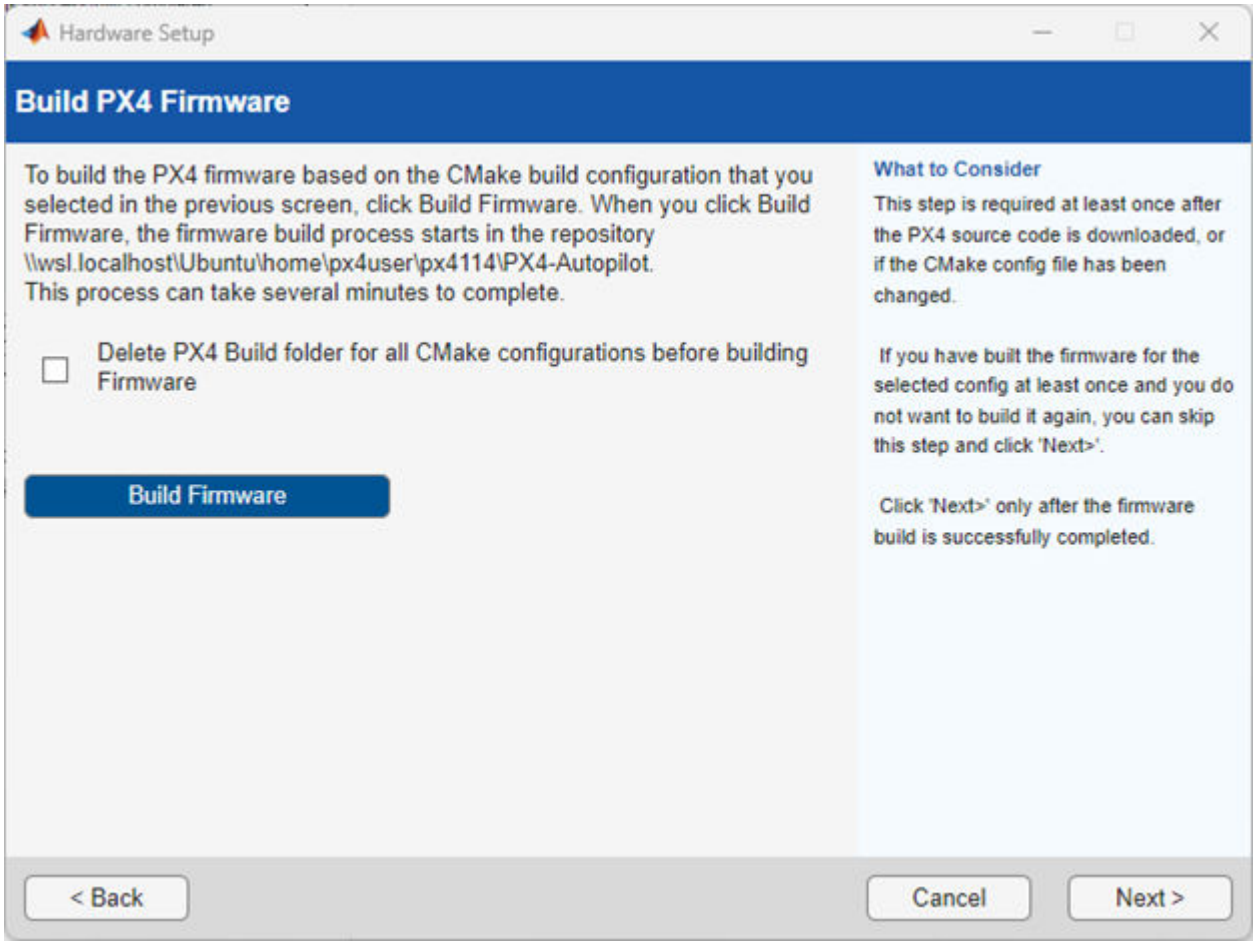


4 Select the corresponding CMake Build target (`cubepilot_cubeblue_multicopter`) from the **Build Target** drop-down list.



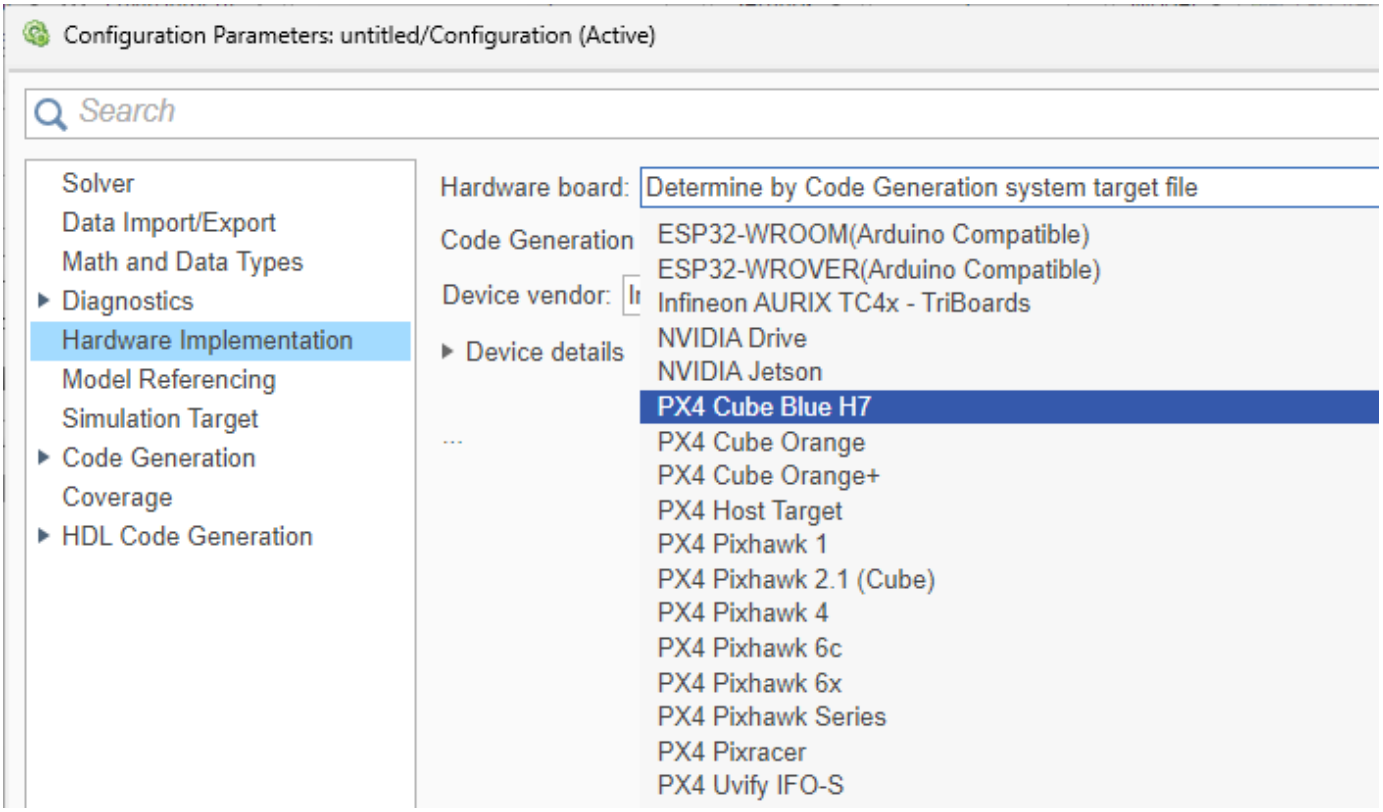
5 Click **Next** and complete the instructions on the page.

6 Click **Build PX4 Firmware** on the **Build PX4 Firware** page to complete the setup.

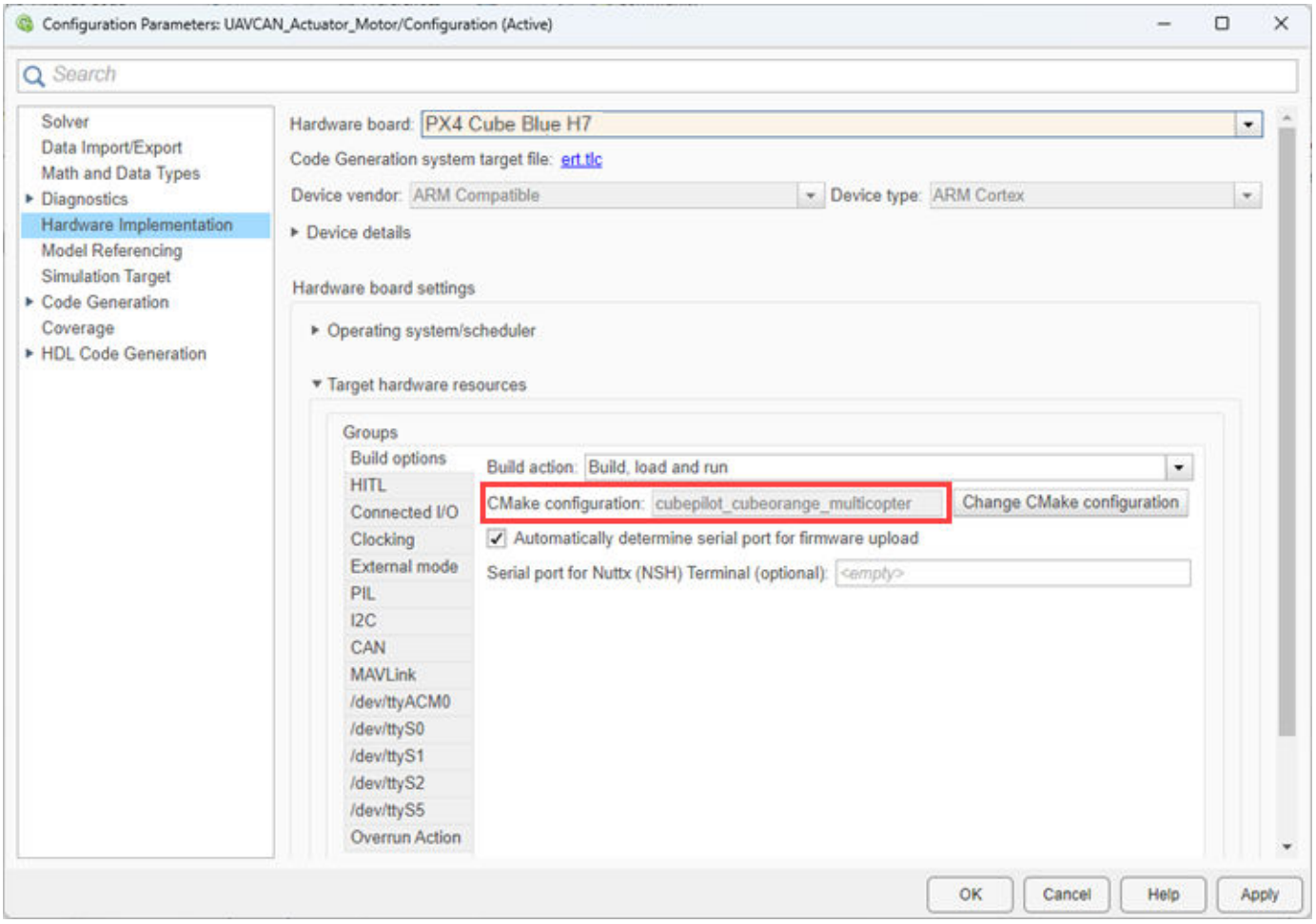


Select PX4 Cube Blue H7 as Simulink Target Hardware

After hardware setup is completed and the PX4 Firmware is successfully built for the selected build target, you can use your Cube Blue H7 autopilot in Simulink by selecting PX4 Cube Blue H7 as the hardware board in the Simulink model configuration settings.



The CMake build target the you select in the hardware setup automatically appears in the CMake configuration.



Note If you are using Cube Blue H7 autopilot, use the GPS2 port for monitoring and tuning parameters over a non-USB port connection, as there might be issues with the `TELEM1` or `TELEM2` ports.

Deployment on Unsupported PX4 Autopilots from Simulink

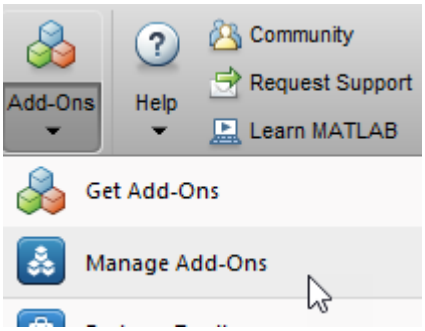
This topic helps you to get started with an autopilot that does not belong to set of officially tested autopilots. This topic uses Cube Orange as an example for deploying Simulink generated code. However, Cube Orange is an officially supported PX4 Autopilot. For information on deployment on Cube Orange Autopilot from Simulink, see “Deployment on Cube Orange Autopilot from Simulink” on page 4-61.

Before starting with Simulink, ensure that you install the version 4.3.0 of QGroundControl and able to upload latest stable version of PX4 Firmware from QGroundControl. If you are facing issue with USB connection, It is recommended to install Mission planner and do a clean re-installation of latest drivers as described in this page.

After you successfully upload the PX4 firmware from QGroundControl, get started with UAV Toolbox Support Package for PX4 Autopilots.

Select Custom PX4 Autopilot Hardware Setup

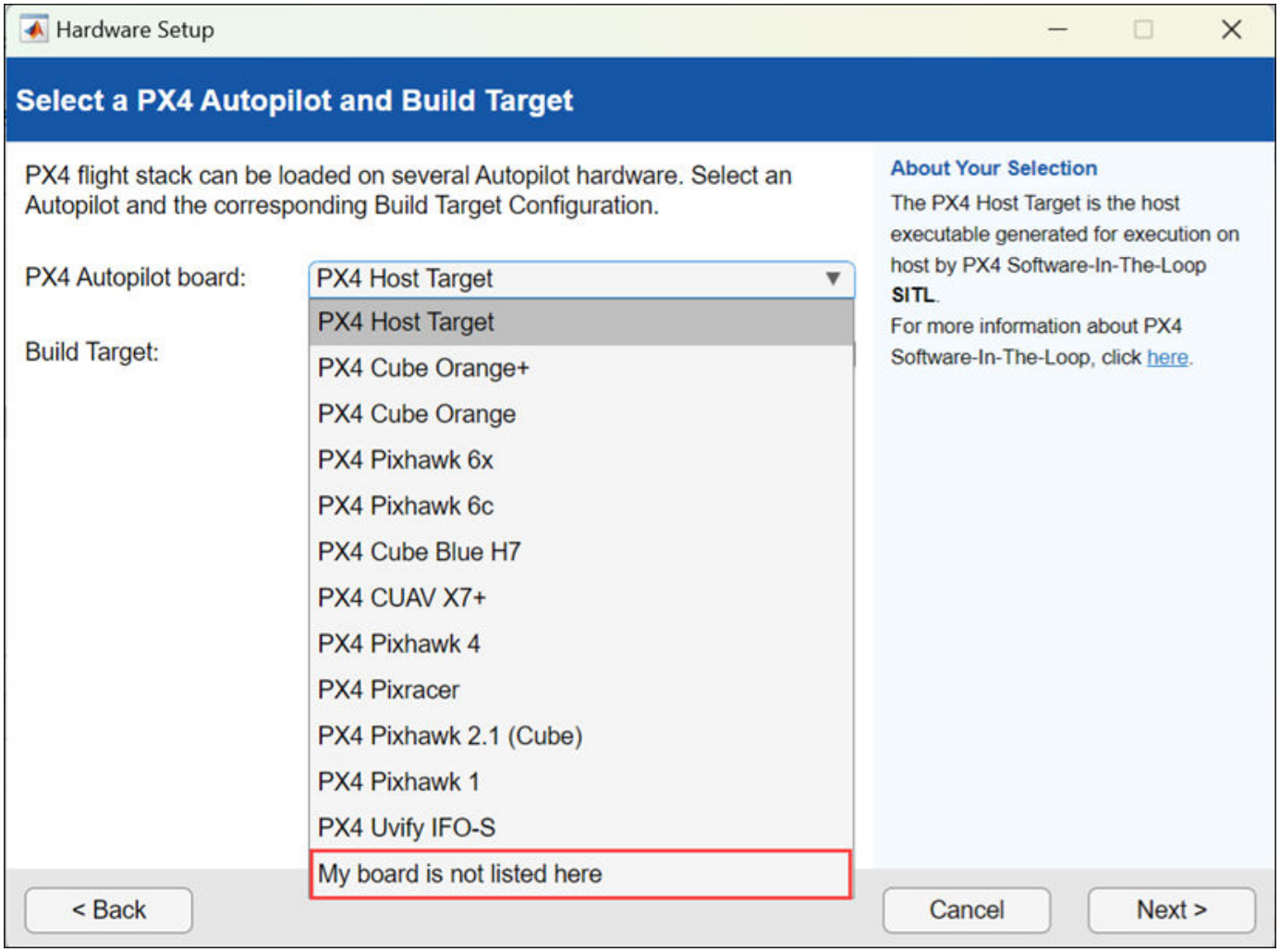
1. If the support package is already installed, start the hardware setup by opening the Add-On Manager.



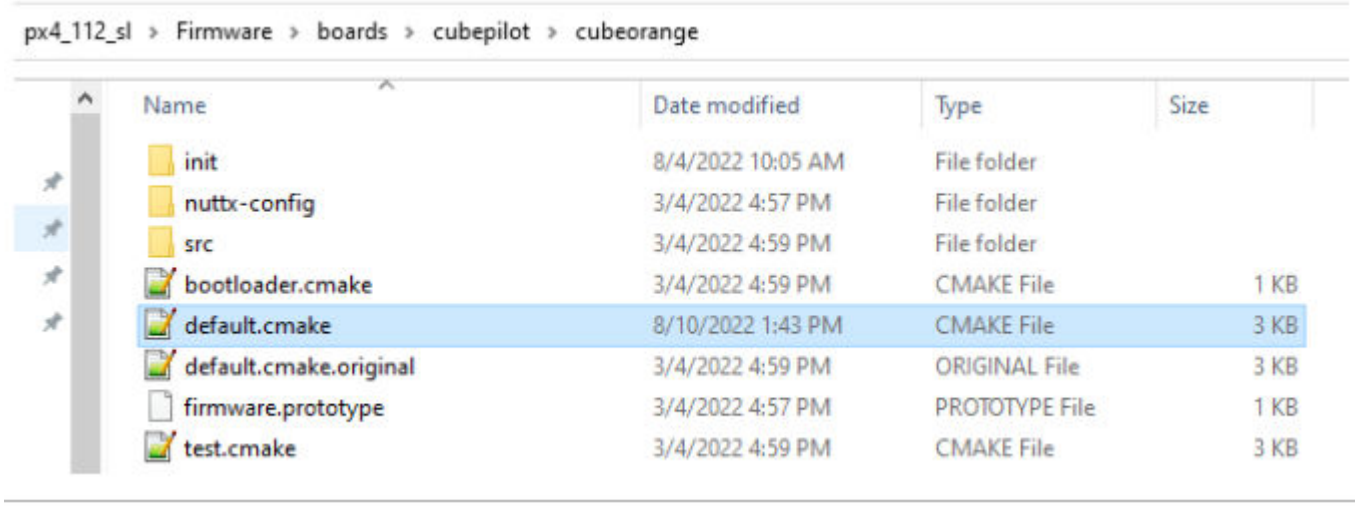
2. In the Add-On Manager, start the hardware setup process by clicking **Setup** button,  .

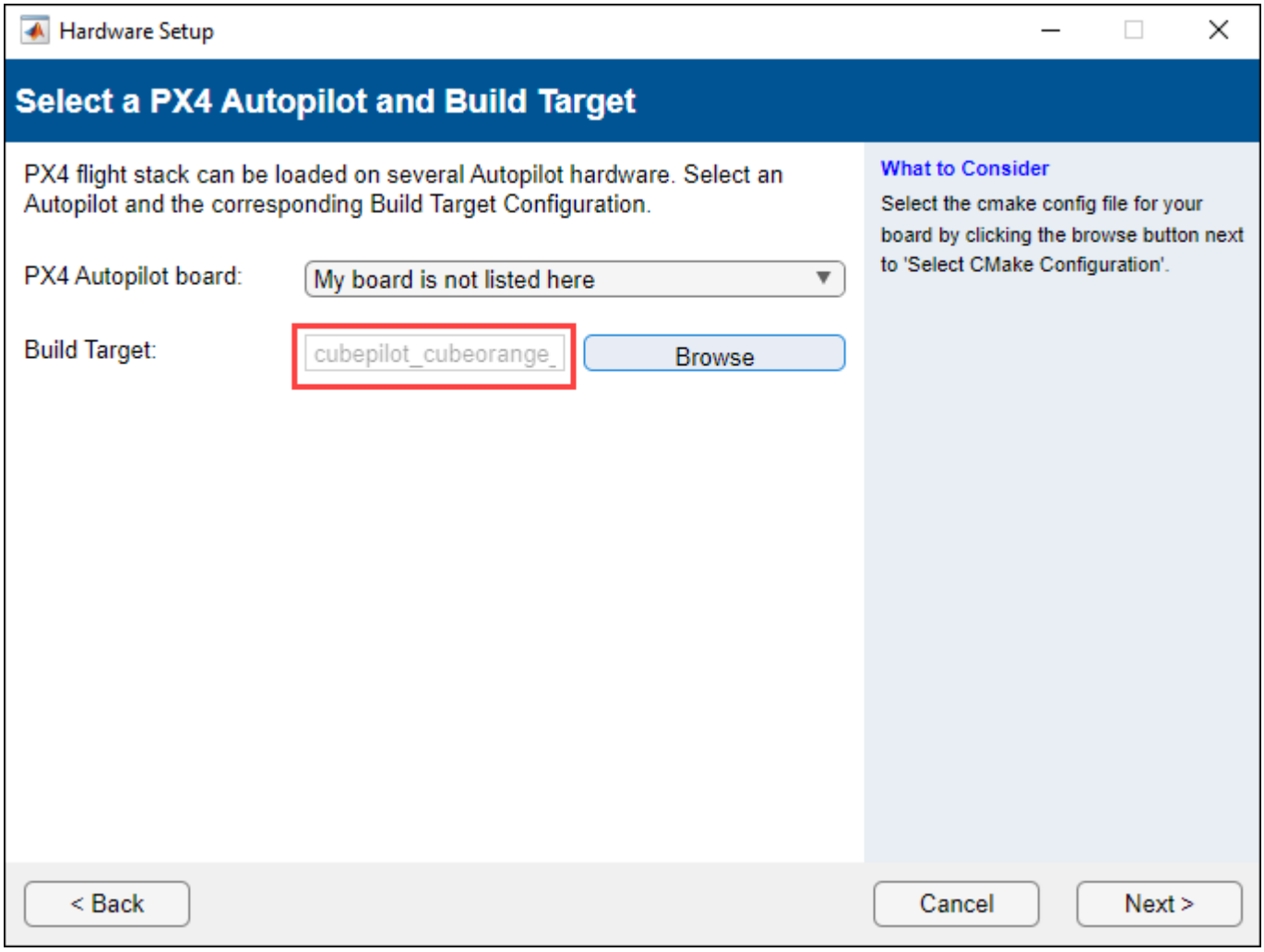
   After starting, the Hardware Setup window provides instructions for configuring the support package to work with your hardware.

3. In the **Select a PX4 Autopilot and Build Target** screen, select `My board is not listed here`.

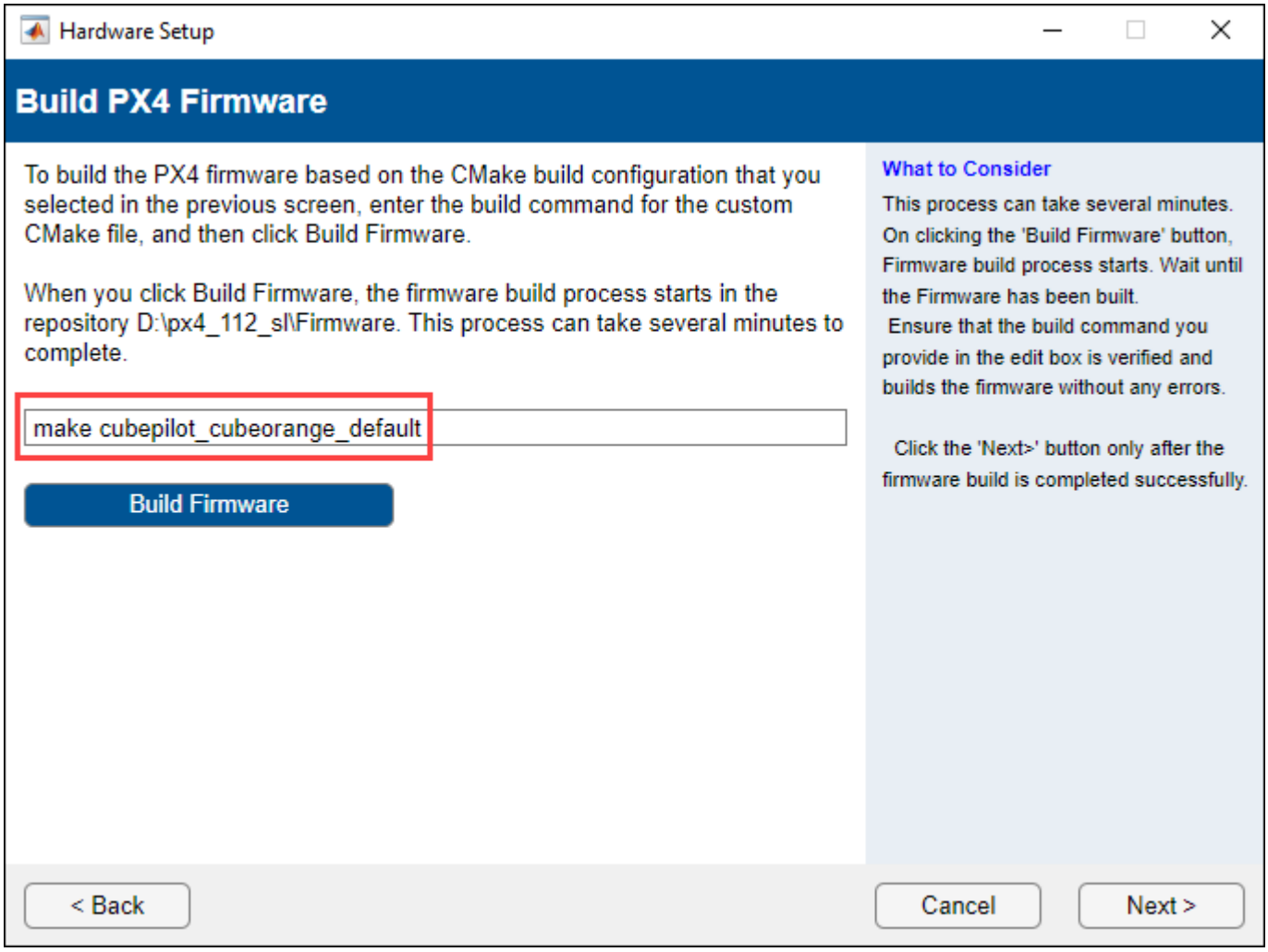


4 Browse to the corresponding CMake Build target (`cubepilot_cubeorange_default`) in your PX4 Firmware for building.

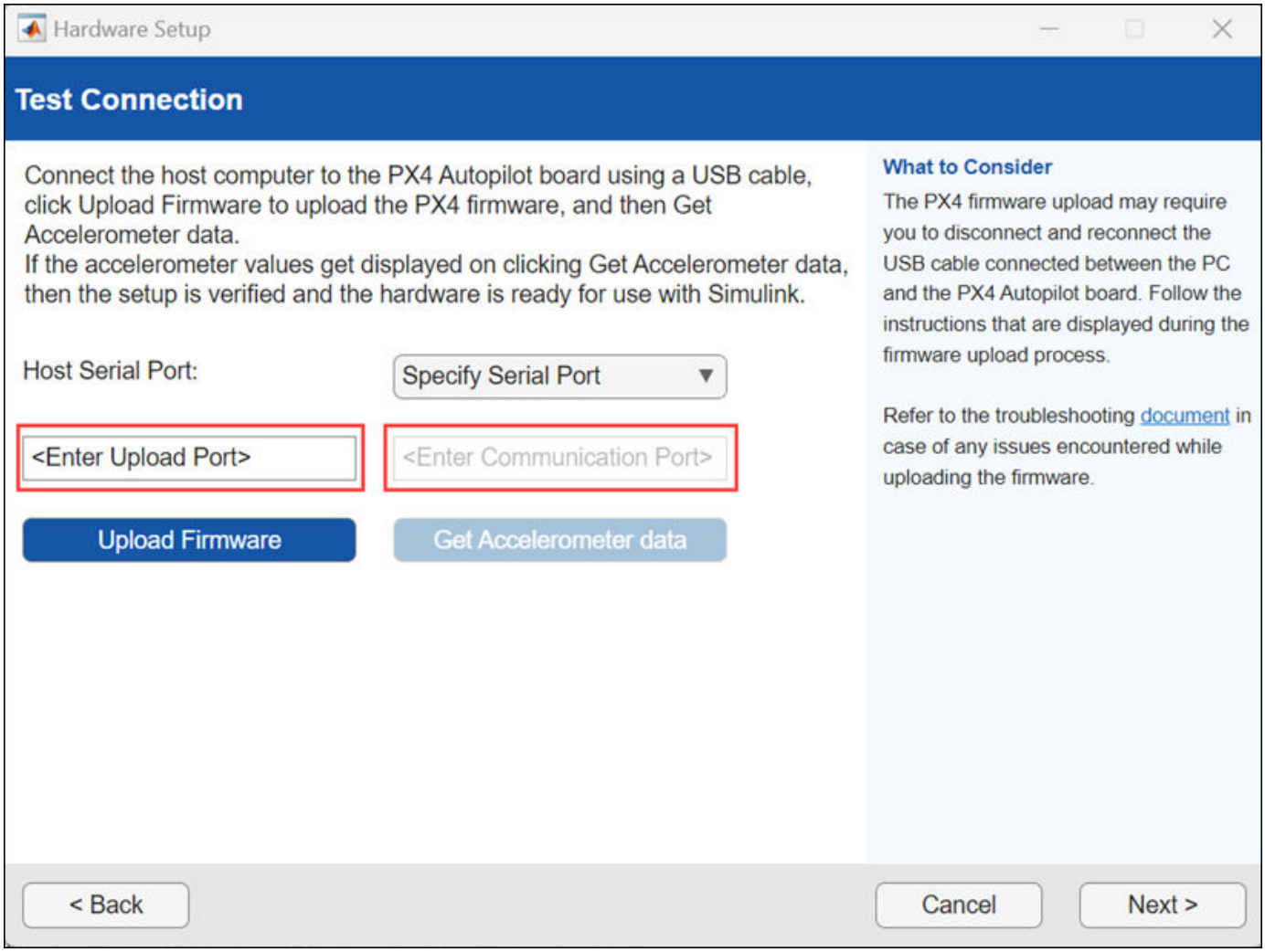




5 Verify the correct make command to build the firmware for Cube Orange and then click **Build Firmware**.



6 For Cube Orange, the bootloader COM Port and the main communications COM Port are different. In the **Test Connection** hardware setup screen, specify the Upload port in `<Enter Upload Port>` text box manually and click **Upload Firmware** button, to upload the PX4 firmware.

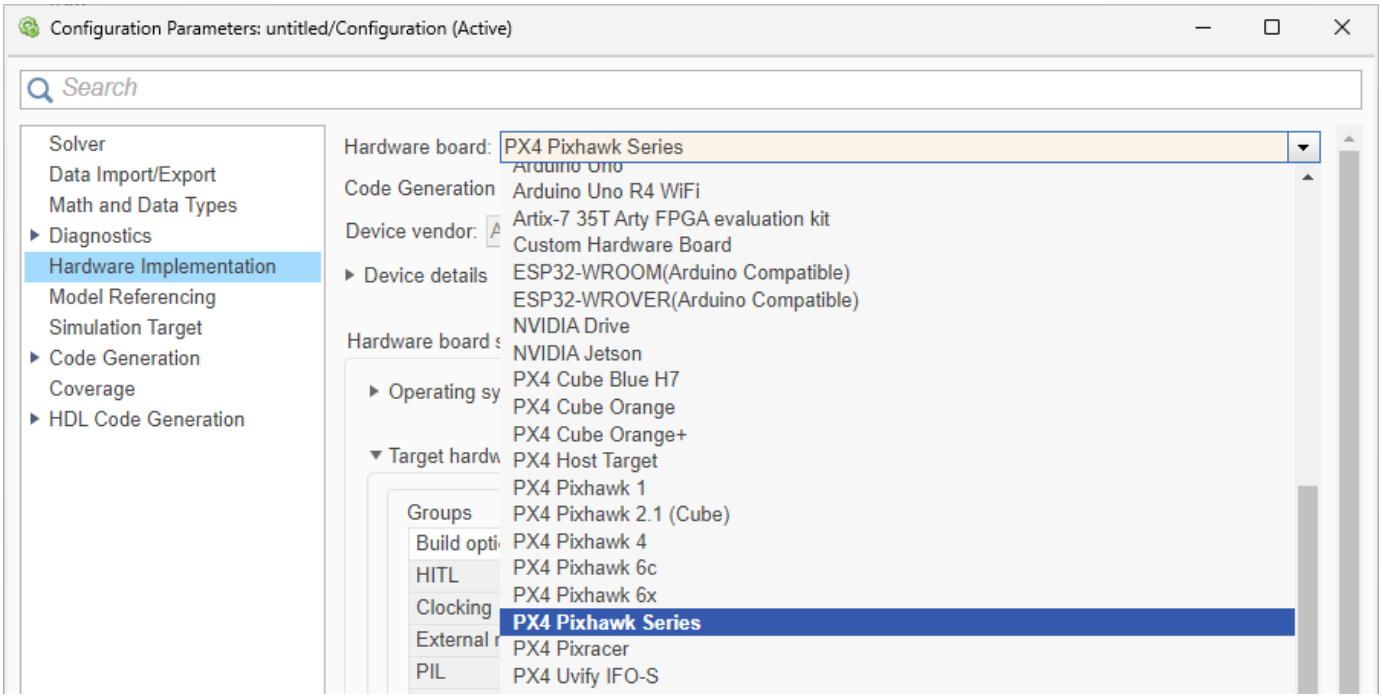


7 Specify the communication port and click **Get Accelerometer Data** to confirm that the hardware is ready for use with Simulink.

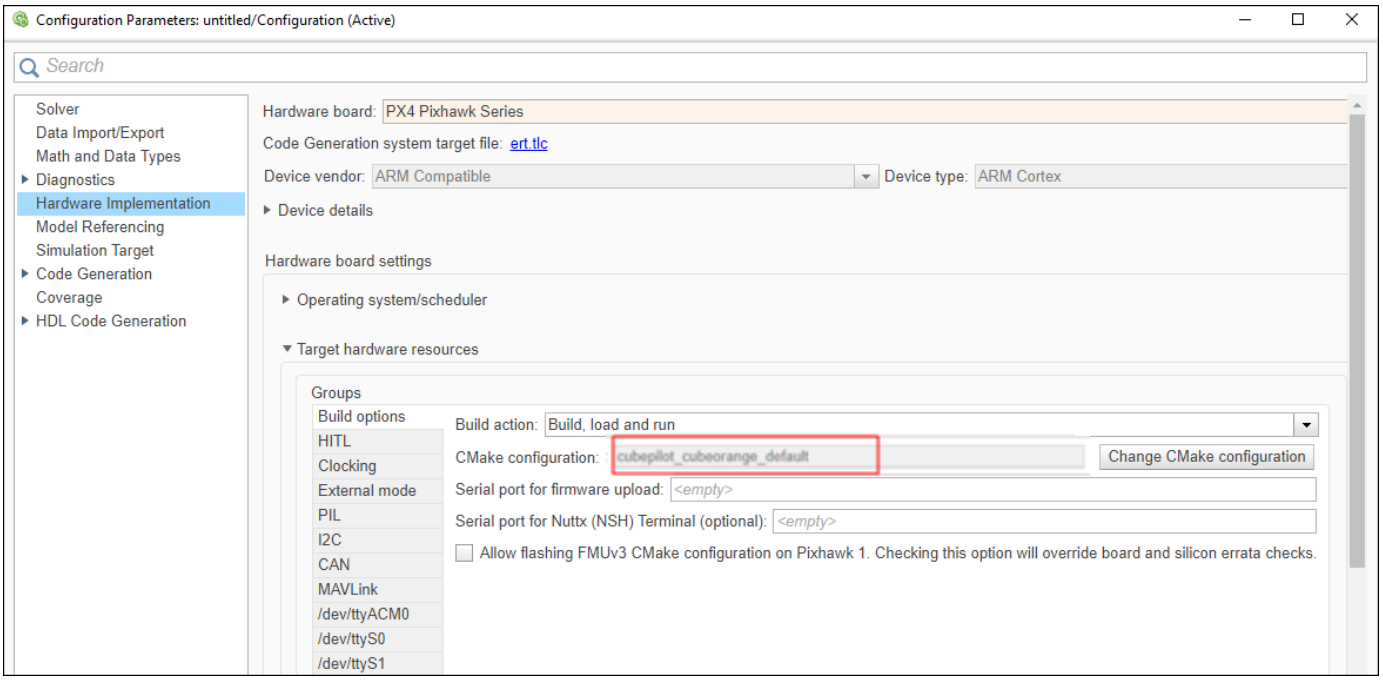
For getting Upload port and Communication port check the Device Manager, The bootloader port appears first in the Device Manager. The port shows for a few seconds and then the main communications port appears. The bootloader COM port is needed for firmware upload and the communication port is needed for MAVLink communication or Monitor and Tune simulation.

Select PX4 Pixhawk Series as the Simulink Target Hardware

After hardware setup is completed and the PX4 Firmware is successfully built for the selected build target, you can use your Cube Orange 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.

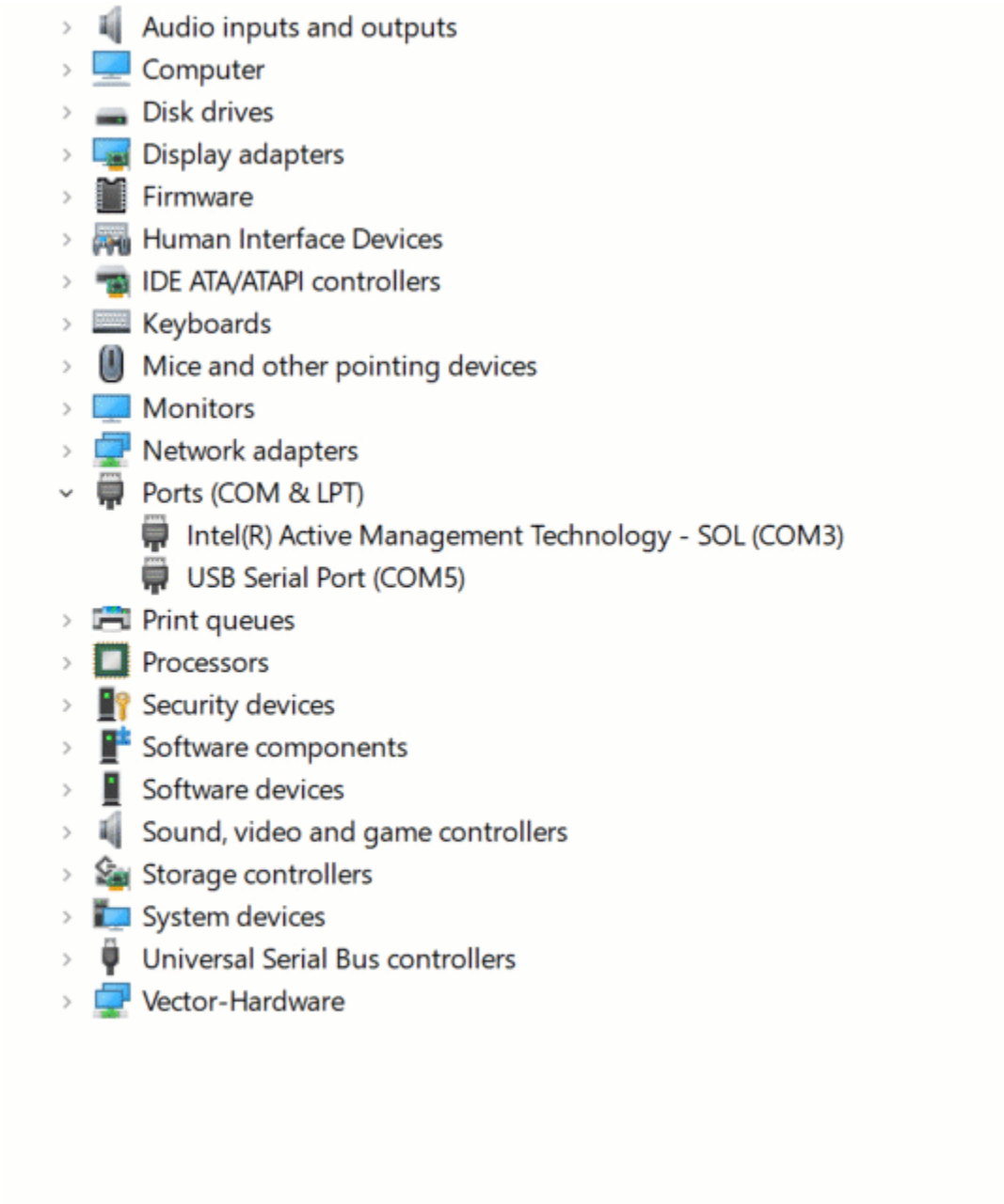


Set COM Port for Upload and Communication in Simulink

In some Cube Orange boards, the bootloader COM Port for uploading the firmware and the main communications COM Port are different. When you connect to such boards, the bootloader port appears first in the Device Manager. The port shows for a few seconds and then the main

communications port appears. The bootloader COM port is needed for firmware upload and the communication port is needed for MAVLink communication or Monitor and Tune simulation.

For example, below is a snapshot of the Device Manager window when a Cube Orange is connected. After connecting, a COM port called ProfiCNC Cube (COM13) comes up under ports. This is the bootloader port needed for firmware upload. This port stays for a while after which the board loads the main Firmware and the main communication port Cube Orange PX4 (COM14) comes up. Hence, for this board COM13 is the bootloader port and COM14 is the communications port.

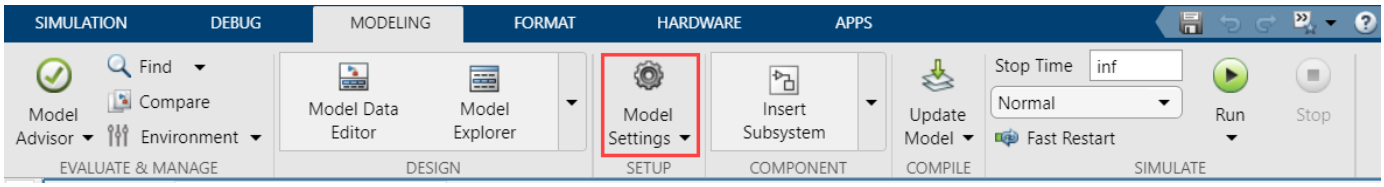


Ensure that you specify the correct COM ports for firmware upload and MAVLink communication. You will be unable to upload the firmware if you specify the communication COM port value instead of

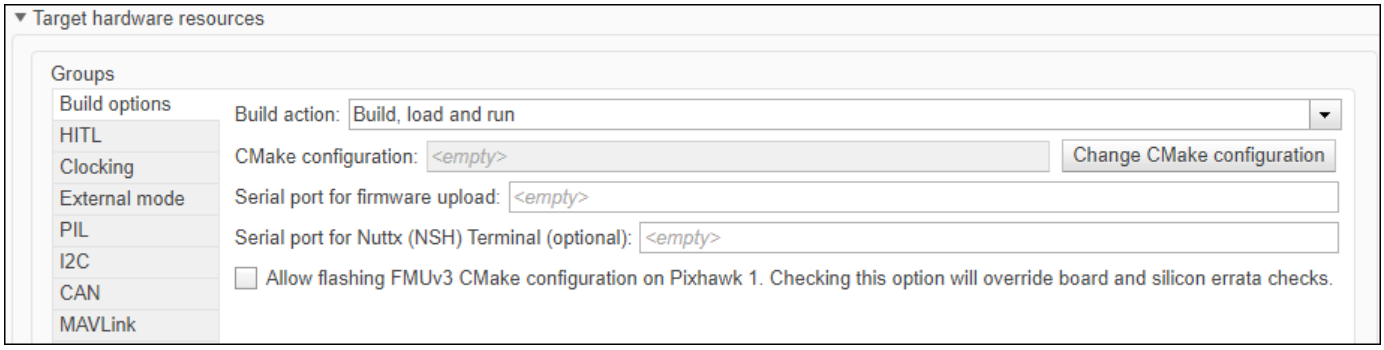
bootloader COM port value for firmware upload. In such cases, perform the following steps to set the upload and communication port correctly for a Pixhawk board in the Simulink model for different modes of simulation.

Set Bootloader COM Port for Firmware Upload and Deploy

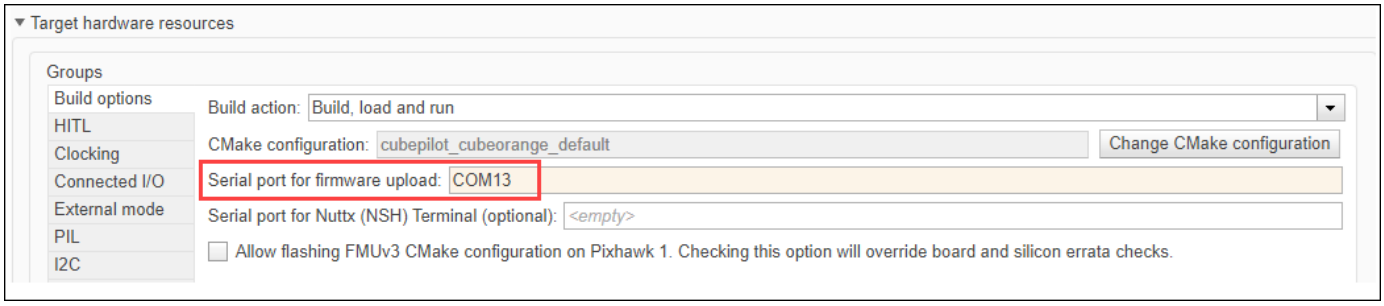
1 For the Simulink model, launch the **Model Settings**.



A sample image of the default settings for hardware implementation is shown here.



2 Enter the bootloader serial port in **Serial Port for firmware upload** (This should be the bootloader port).

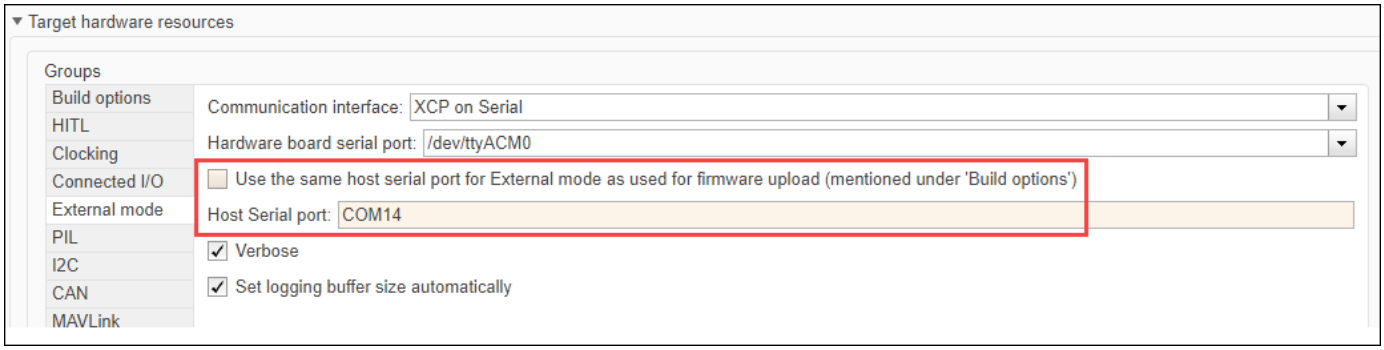


3 Deploy the Simulink model. On the **Hardware** tab, in the **Mode** section, select **Run on board** and then click **Build, Deploy & Start**.

Set Communication COM Port for Monitor & Tune (External Mode)

1 Perform the settings for Build options as shown in the topic “Set Bootloader COM Port for Firmware Upload and Deploy” on page 4-51.

2 In **External mode** tab, clear the **Use the same host serial port for External Mode as used for firmware upload** option and mention the main communications port in **Host Serial Port** (this must be different from the bootloader port you mentioned earlier in **Build options**).

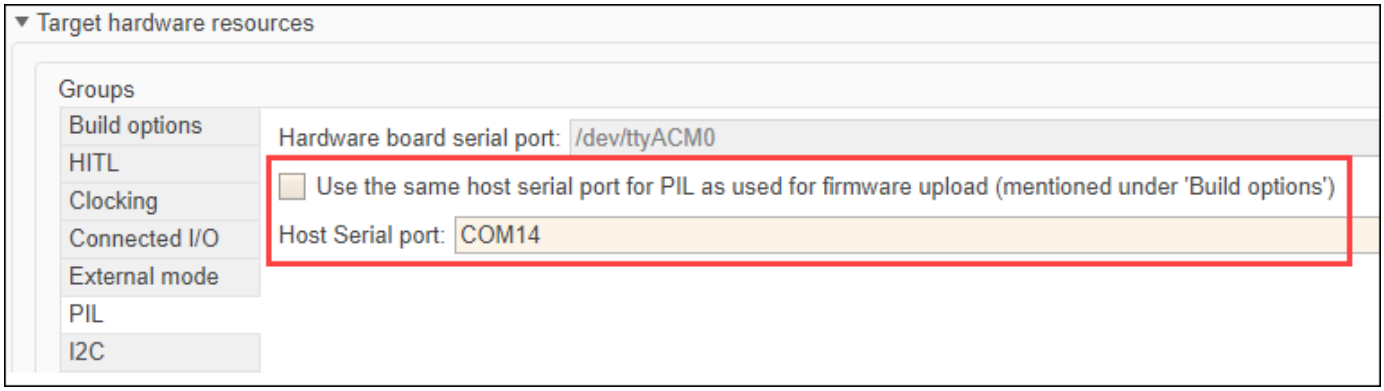


3 Run the model in External Mode.

Set Communication COM Port for PIL

1 Perform the settings for Build options as shown in the topic “Set Bootloader COM Port for Firmware Upload and Deploy” on page 4-51.

2 In **PIL** Tab, clear the **Use the same host serial port for PIL as used for firmware upload** option and mention the main communications port in **Host Serial port** (this must be different from the bootloader port you mentioned earlier in **Build options**).



Limitations

- Connected I/O Simulation is not supported with PX4 Pixhawk Series boards.
- In the “MAT-file Logging on SD Card for PX4 Autopilots” example, provide communication COM port as an input argument to the `getMATFilesFromPixhawk` function. Automatic detection of COM port is not supported with Cube orange.

```
getMATFilesFromPixhawk('px4demo_log','DeleteAfterRetrieval',
true,'COMPort','COM13')
```

Set COM Port for Upload and Communication in Simulink

In some Cube Orange boards, the bootloader COM Port for uploading the firmware and the main communications COM Port are different. When you connect to such boards, the bootloader port appears first in the Device Manager. The port shows for a few seconds and then the main communications port appears. The bootloader COM port is needed for firmware upload and the communication port is needed for MAVLink communication or Monitor and Tune simulation.

For example, below is a snapshot of the Device Manager window when a Cube Orange was connected. After connection, a COM port called Cube Orange PX4 (COM14) comes up under ports. This is the bootloader port needed for firmware upload. This port stays for a while after which the board loads the main Firmware and hence the main communication port ProfiCNC Cube (COM13) comes up. Hence, for this board COM14 is the bootloader port and COM13 is the communications port.

Ensure that you specify the correct COM ports for firmware upload and MAVLink communication. You will be unable to upload the firmware if you specify the communication COM port value instead of bootloader COM port value for firmware upload. In such cases, perform the following steps to set the upload and communication port correctly for a Pixhawk board in the Simulink model for different modes of simulation.

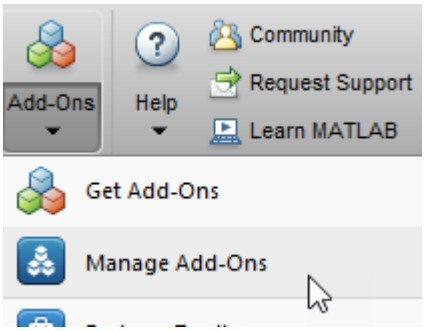
Install Python 3.8.2 on Windows for Firmware Upload

The UAV Toolbox Support Package for PX4 Autopilots requires Python `3.8.2` and pySerial `3.4` to upload PX4 Firmware on the Autopilot. The installation of Python `3.8.2` and pySerial `3.4` can be done from the hardware setup screen itself (if not already installed).

Note Internet connection is required to install Python `3.8.2` in your Windows computer

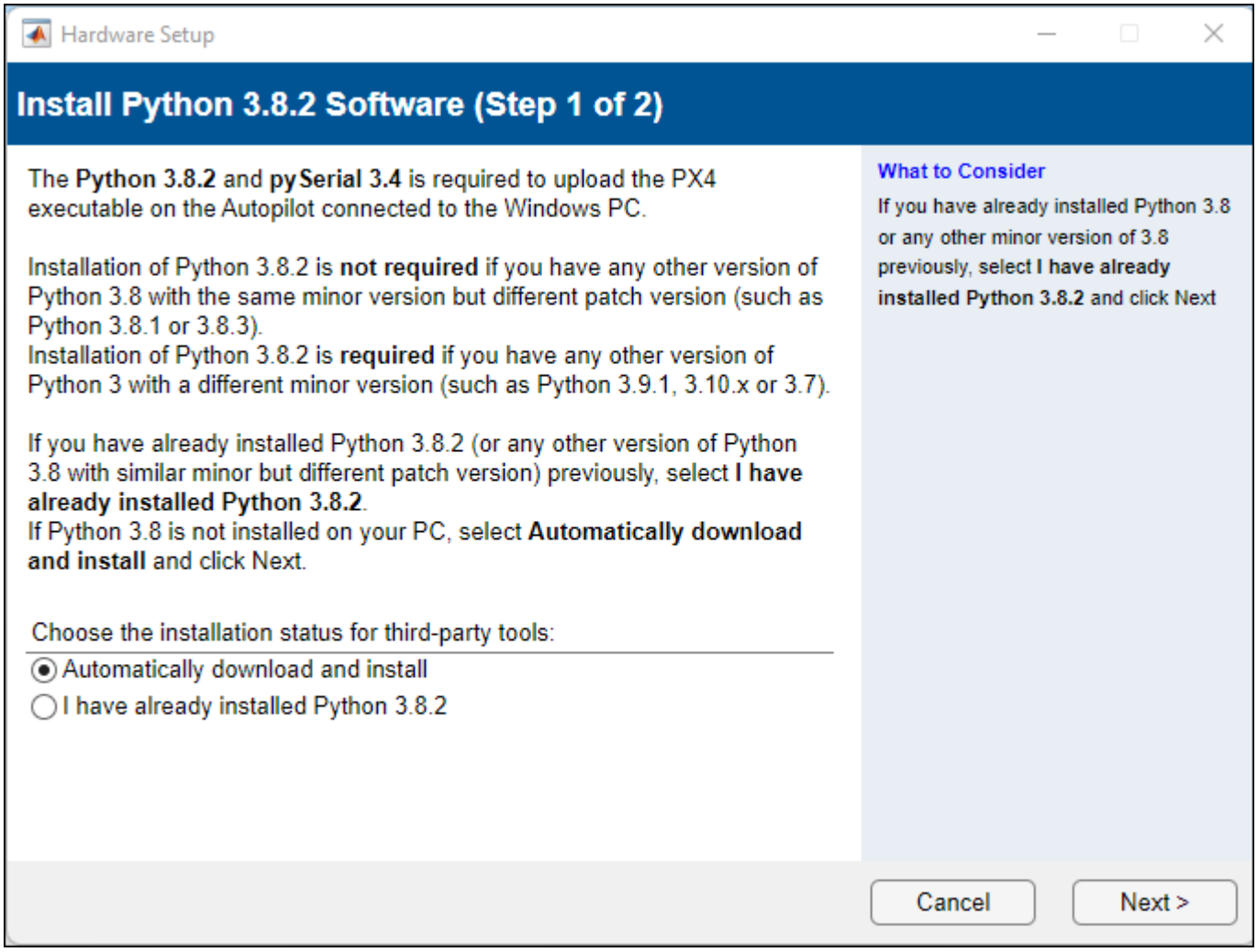
To install Python 3.8.2 from hardware setup, follow the below steps.

1. After installing the UAV Toolbox Support Package for PX4 Autopilots, start the hardware setup by opening the Add-On Manager.



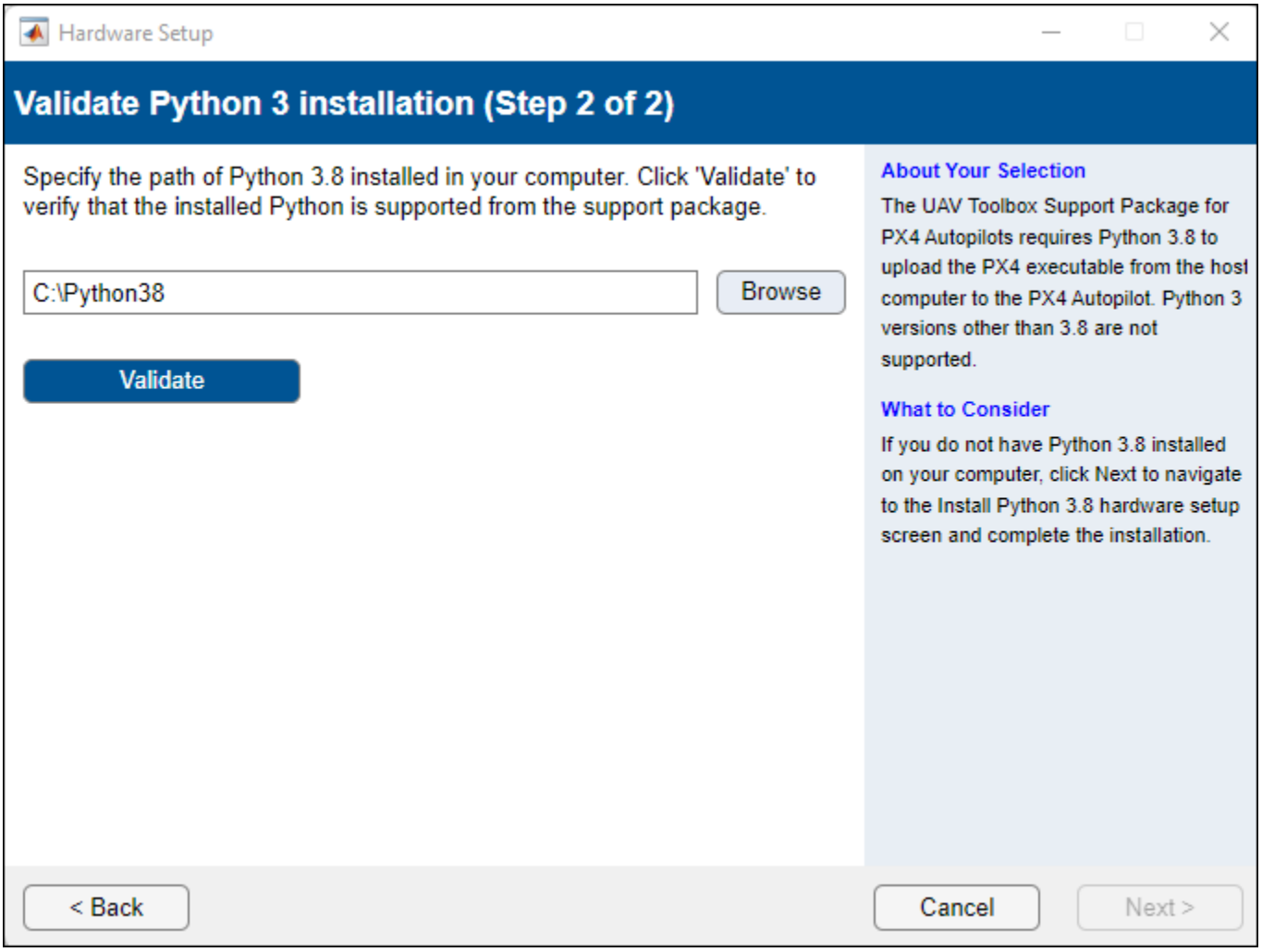
2. In the Add-On Manager, start the hardware setup process by clicking the **Setup** button,  .

   The following screen is launched if hardware setup for PX4 is being launched for the first time in current installation of MATLAB.

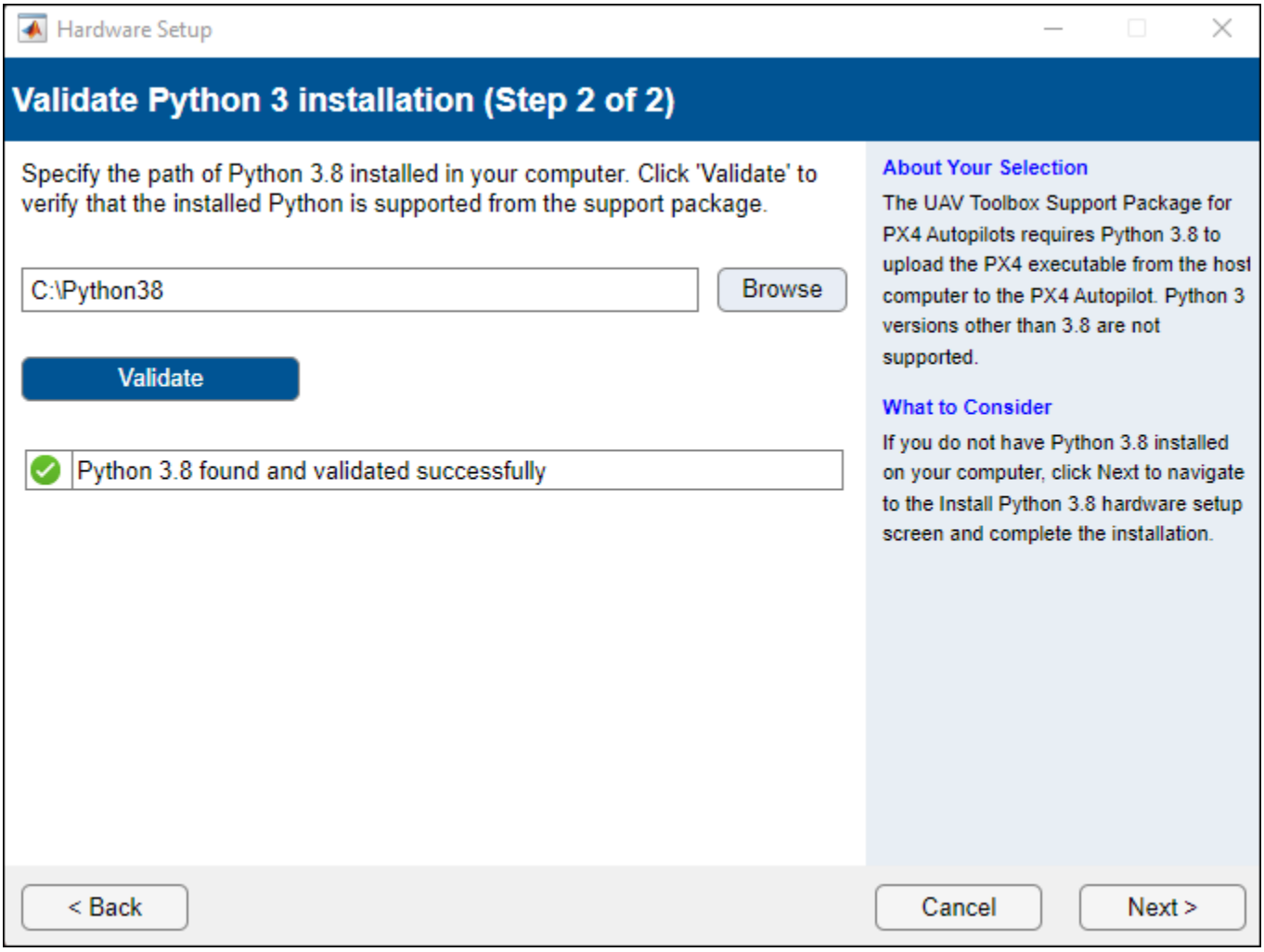


Scenario 1: Python 3.8.2 is already installed on Windows computer

If Python `3.8.2` is already installed on your computer, select **I have already installed Python 3.8.2** and click **Next**. The next hardware setup requires a valid path where you have Python `3.8.2` installed.



After you provide the path where Python `3.8.2` is installed. Click **Validate**. Upon successful validation, you can move to subsequent hardware setup screens.



If validation fails, reinstall Python 3.8.2. For more information, see Scenario 2 on page 4-86.

Scenario 2: Python 3.8.2 is not installed on Windows computer

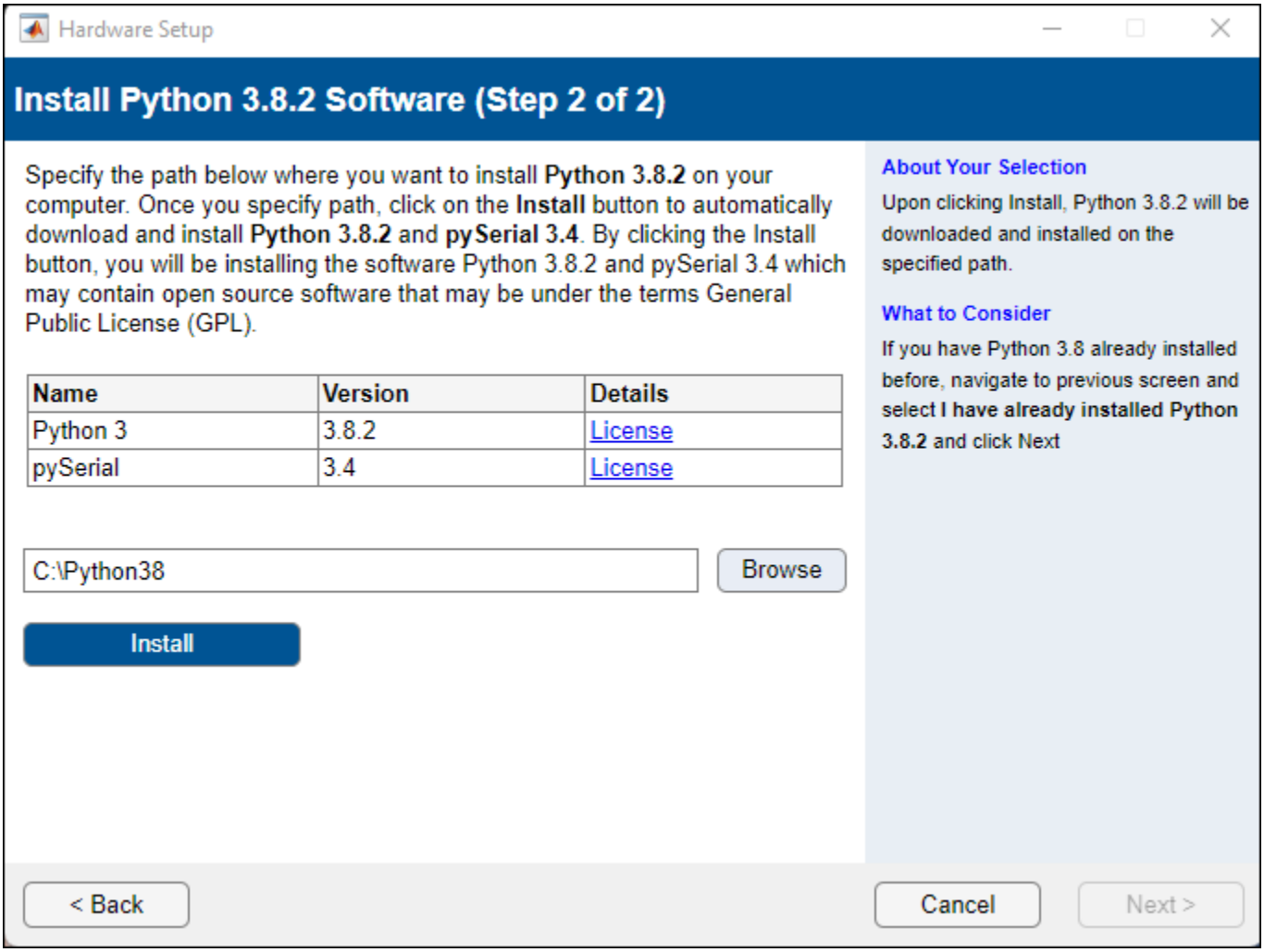
If Python `3.8.2` is not installed on the computer, select **Automatically Download and install** and click **Next**.

Note The installation of Python `3.8.2` is not required if you have any other version of Python `3.8` with the same minor version (8) but different patch version. For example, if you have Python `3.8.1` or `3.8.3` already installed on your PC, this is same as Scenario 1 on page 4-84 and you can select I have already installed Python `3.8.2` and provide the path of Python `3.8.1` or `3.8.3` for validation.

However, installation of Python 3.8.2 is required if you have any other version of Python 3 with a different minor version other than 8, such as Python `3.9.1`, `3.10.x`, or `3.7`).

Also, installation of Python `3.8.2` will fail if there is another Python with same minor version such as Python `3.8.2`, `3.8.3`, or `3.8.5` and so on is already installed. However, you can install Python `3.8.3` if the other Python version has a different minor version other than 8, such as Python `3.9.1`, `3.10x`, or `3.7`.

After clicking Next, the following screen appears.



Specify the path where you want to install Python `3.8.2` in your computer and click **Install**. Python `3.8.2` and pySerial `3.4` is installed on that path. After successful installation, click **Next** for subsequent screens.

See Also

More About

- “Troubleshooting Python 3.8.2 Installation Issues” on page 4-88

Troubleshooting Python 3.8.2 Installation Issues

- Python `3.8.2` is already installed in your PC.

  Verify this by launching **Programs and Features** in Windows Control Panel and checking if Python `3.8.2` is present in list of installed programs. In this case, select **I have already installed Python 3.8.2** in **Install Python 3.8.2** hardware setup screen and provide the current Python `3.8.2` installation path in **Validate Python 3.8.2 installation** screen.
- Another Python with same minor version as `3.8.2` (8) but different patch version, such as `3.8.1` or `3.8.5` is installed in your computer.

  Verify this by launching **Programs and Features** in Windows Control Panel and seeing if Python `3.8` is present in list of installed programs. In this case, select **I have already installed Python 3.8.2** in **Install Python 3.8.2** hardware setup screen and provide the current Python `3.8` (such as `3.8.1` or `3.8.5`) installation path in “Validate Python `3.8.2` installation” screen.
- You do not administrative privileges in your computer to install Python `3.8.2`.

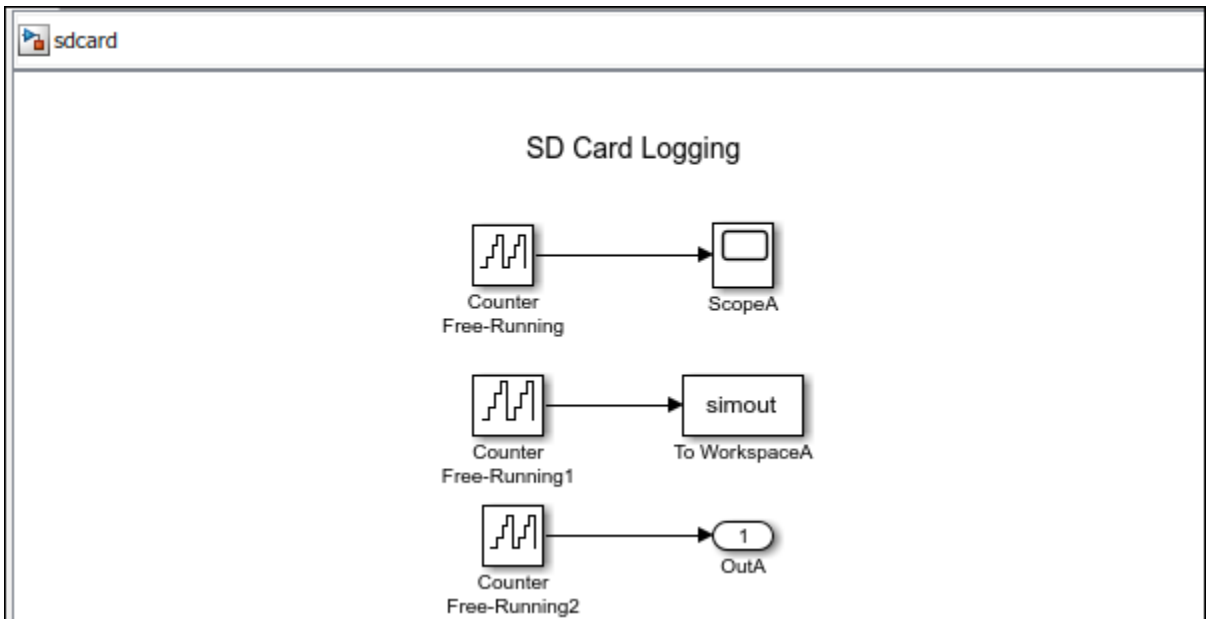
  In this case, you need to obtain administrative privileges to carry out the installation.

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`

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 5-4.

1. “Configure Model to Log Signals on SD Card” on page 5-5: 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 5-8
3. “Run Model on Target Hardware” on page 5-10: 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 5-11: After logging is complete, you can open MAT-files in MATLAB and use them for further analysis.

See Also

Related Examples

- “MAT-file Logging on SD Card for PX4 Autopilots”

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.

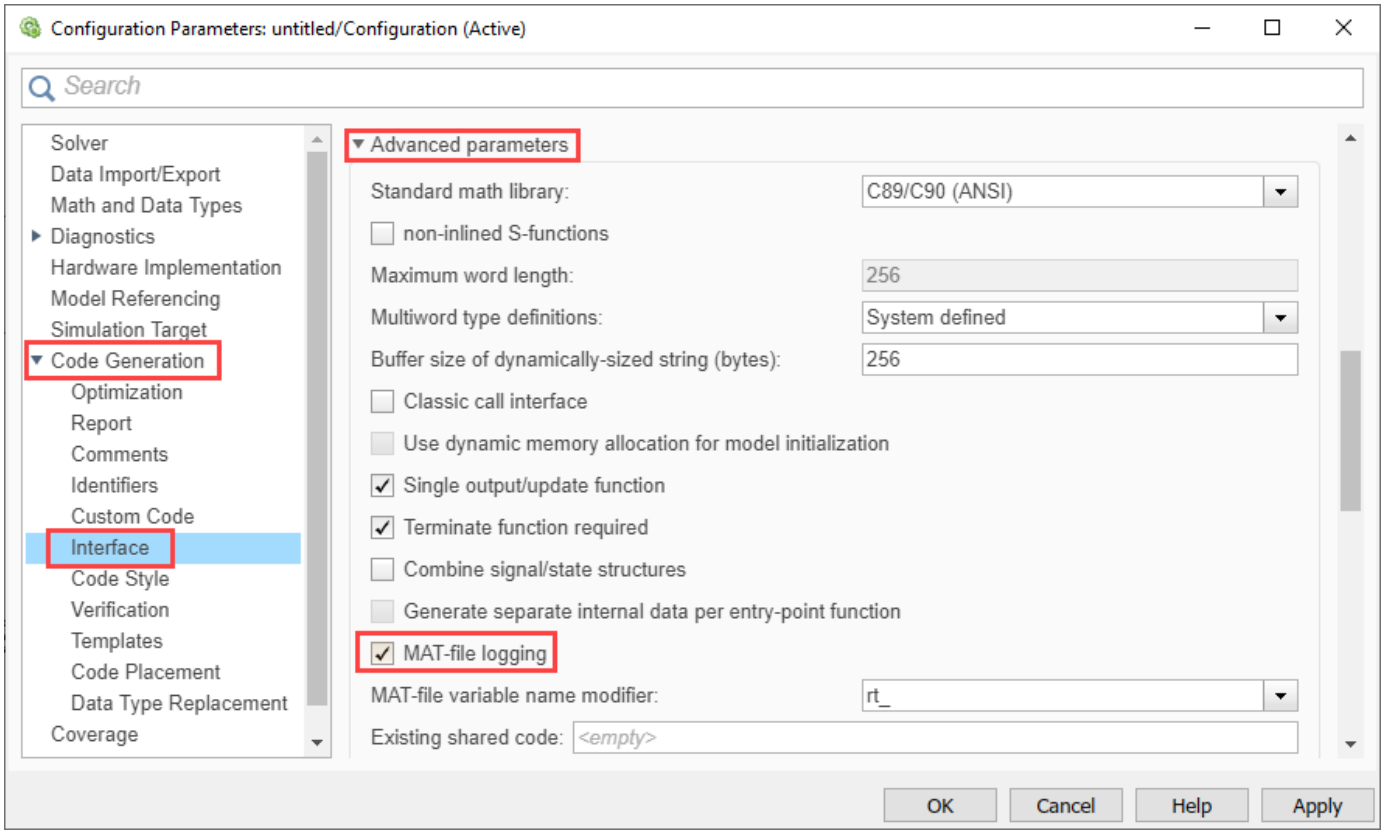
3. Save the changes to the Simulink model.

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.

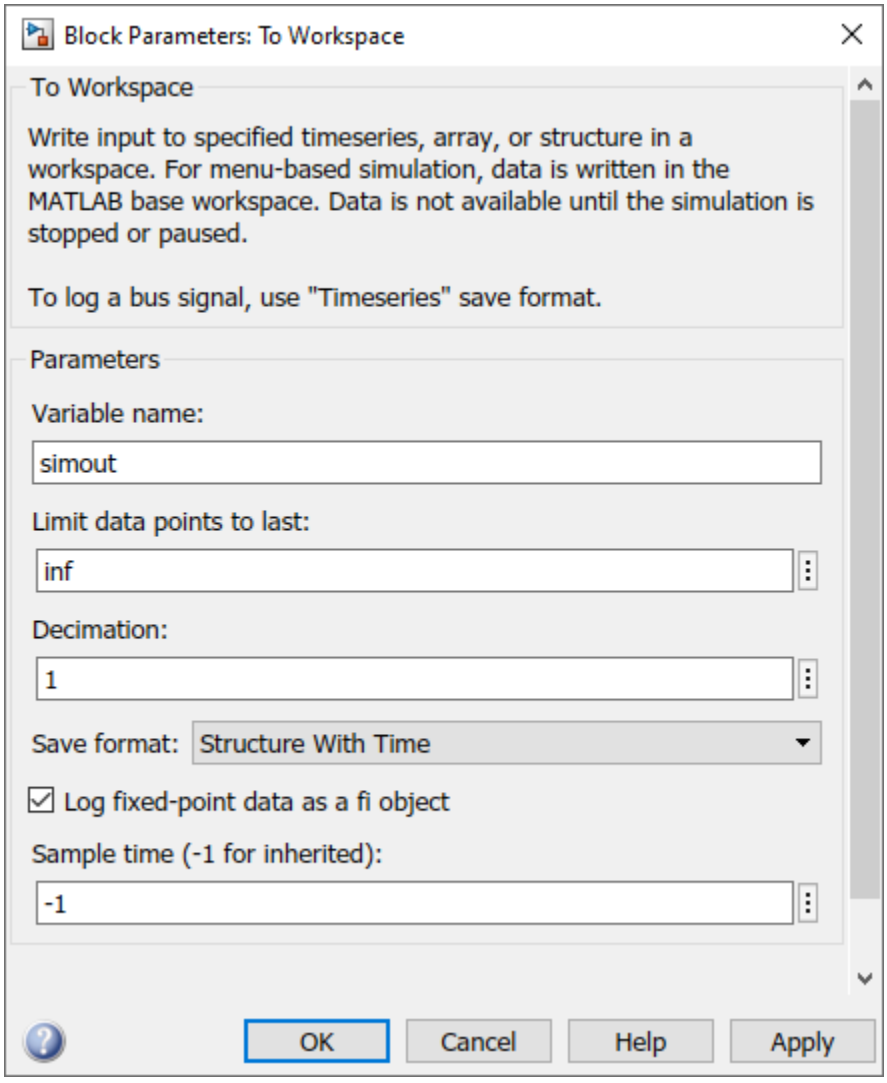
   

2. In the Simulink model, set the parameter values of To Workspace blocks.

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	You can use the default value for this parameter. **Note** Before you simulate the model, you need to run the `px4PrepareModelForMATFileLogging()` 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 5-8.
Decimation	Use this parameter for the block to write data points at every *n*th sample, where *n* is the decimation factor. The default decimation, `1`, writes data at every time sample. 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.
Save format	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`. • `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. **Note** Select **Save format** as `Array` to use the memory efficiently.
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.


Block Parameters: To Workspace
To Workspace
Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.
To log a bus signal, use "Timeseries" save format.
Parameters
Variable name:
simout
Limit data points to last:
inf
Decimation:
1
Save format:
Structure With Time
Log fixed-point data as a fi object
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply

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

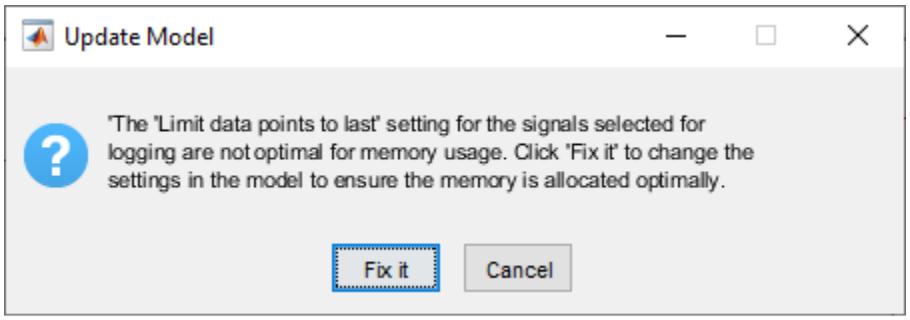
The `px4PrepareModelForMATFileLogging` function needs to be run in the MATLAB command prompt to optimize the memory. The function checks the **Limit data points to last** parameter in the various `To Workspace` blocks and adjusts its value for memory optimization.

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.

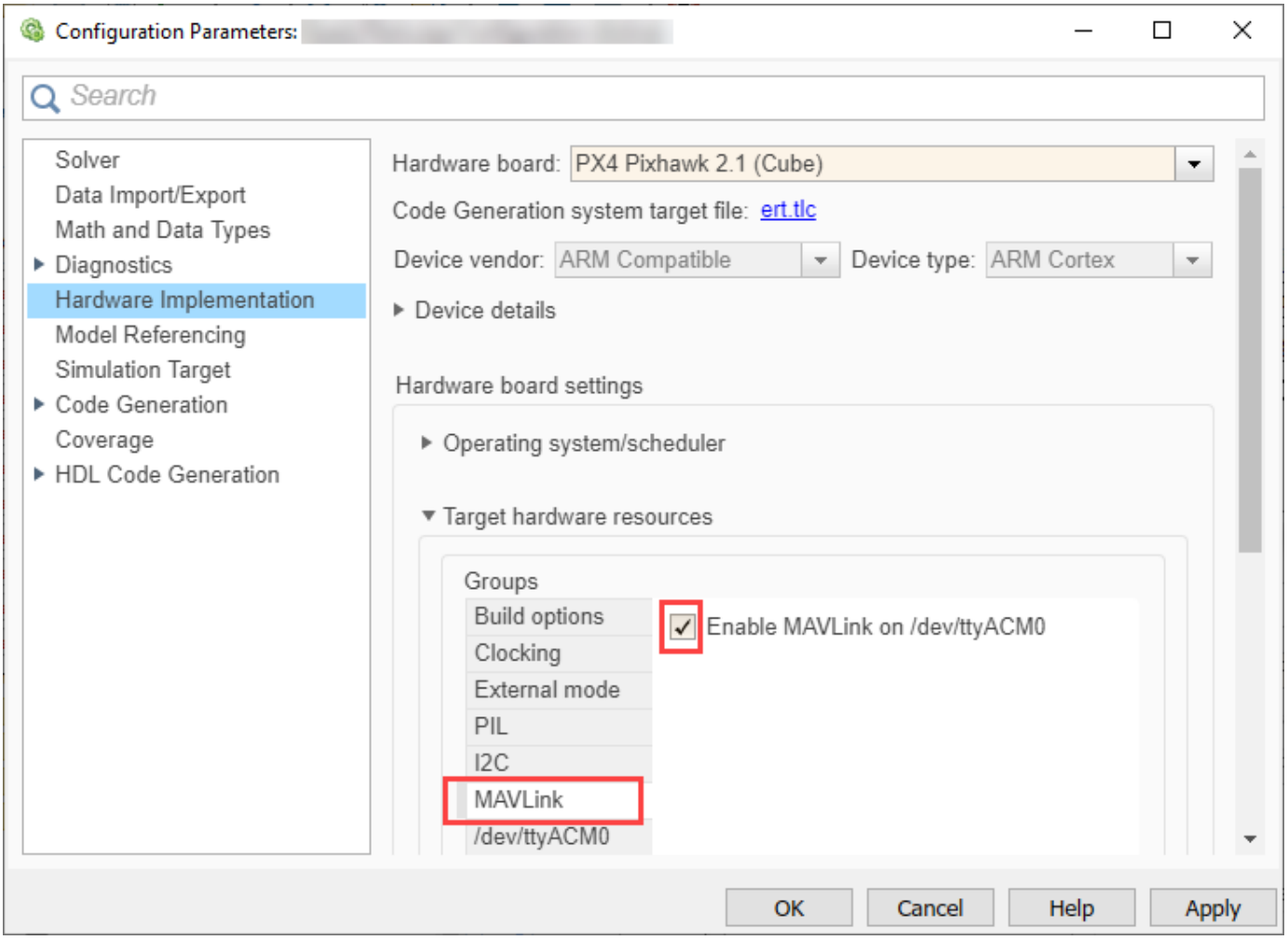


After you click **Fix it**, the value for **Limit data points to last** corresponding to all 'To Workspace' blocks is changed.

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

Enable MAVLink

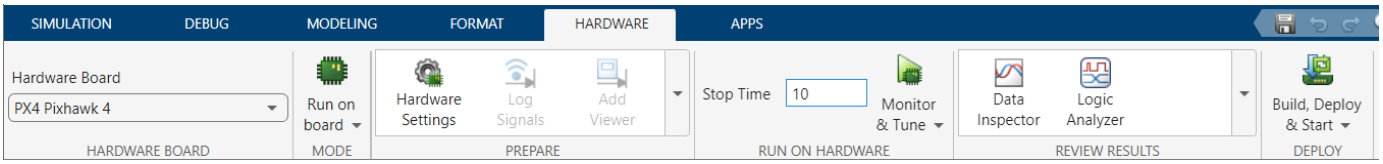
MAVLink needs to be enabled to retrieve the log files from SD card inserted on the Pixhawk hardware board (for details about retrieving files, see `getMATFilesFromPixhawk`). To enable MAVLink, open the Configuration Parameters dialog box, go to **Hardware Implementation** > **Target hardware resources**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


Configuration Parameters:
Search
Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Coverage
HDL Code Generation
Hardware board: PX4 Pixhawk 2.1 (Cube)
Code Generation system target file: ert.tlc
Device vendor: ARM Compatible
Device type: ARM Cortex
Device details
Hardware board settings
Operating system/scheduler
Target hardware resources
Groups
Build options
Clocking
External mode
PIL
I2C
MAVLink
/dev/ttyACM0
Enable MAVLink on /dev/ttyACM0
OK
Cancel
Help
Apply

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.

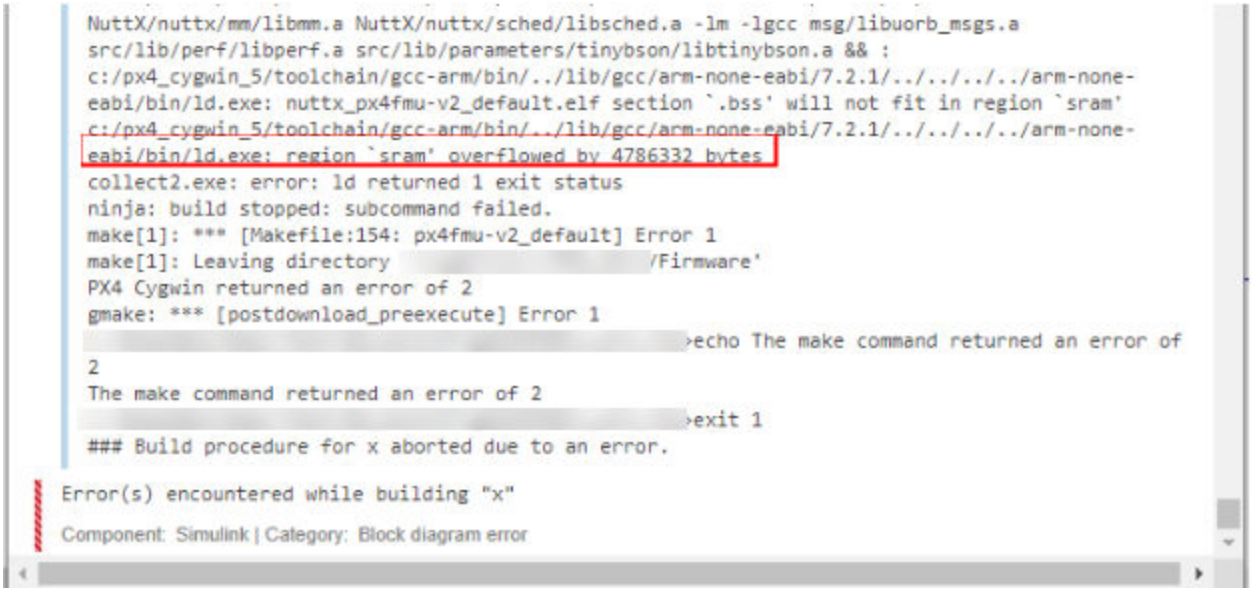
Load the stitched MAT-file. For example, run the below command if the name of the stitched file is `sdcard_1__stiched.mat`:

```
load('sdcard_1__stiched.mat')
```

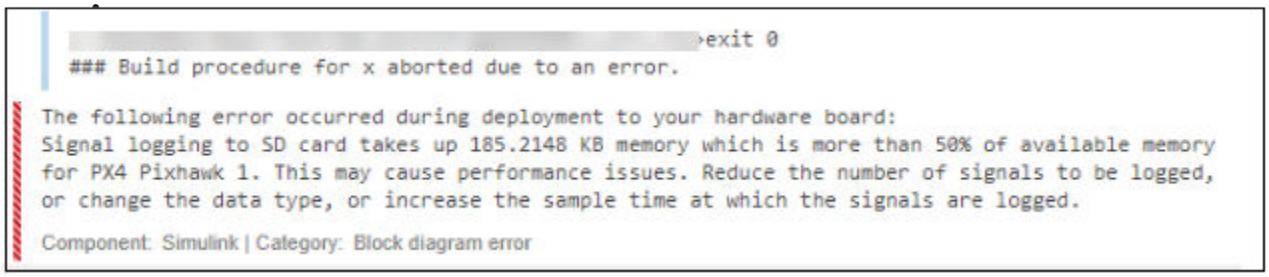
To view the logged values, double-click the corresponding variable that appears in the **Workspace** window.

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:



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



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.

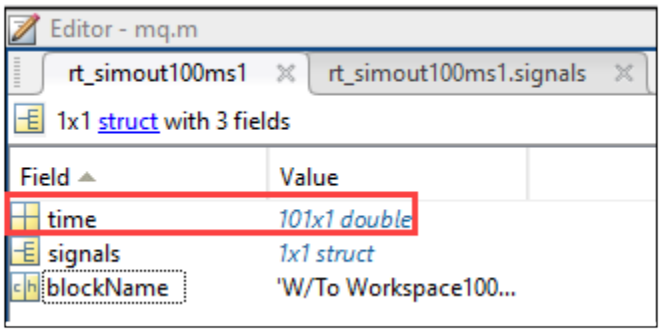
Action

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

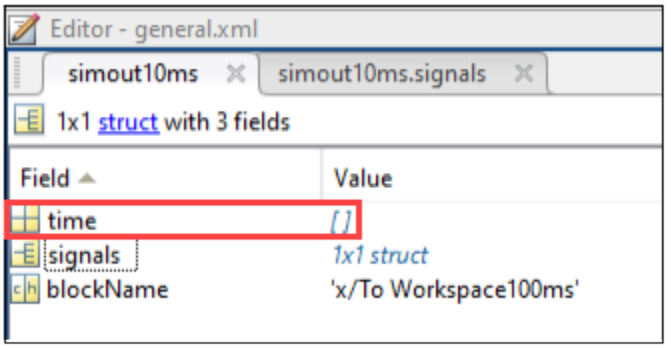
- Reduce the number of signals to be logged.
- Change the data type of the signal to be logged to achieve a smaller size for data. By default, the data type of signals in Simulink is `double`, which takes twice the amount of memory compared to a signal with `single` data type. Additionally, the `double`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



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



  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.
- Te 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:

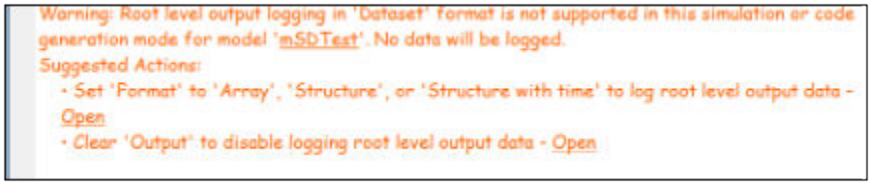
```
>> px4PrepareModelForMATFileLogging(bdroot)
Warning: The memory required to log the selected signals to SD card
may cause any of the below memory issues:
 1. Linker RAM overflow issues.
 2. Less RAM available to run the application. This may increase the
 chances of a stack/heap collision that will 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

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**).

PX4 SITL Plant Model

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

In this section...

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/Simulink” on page 4-23.

Controller Model and Plant Model

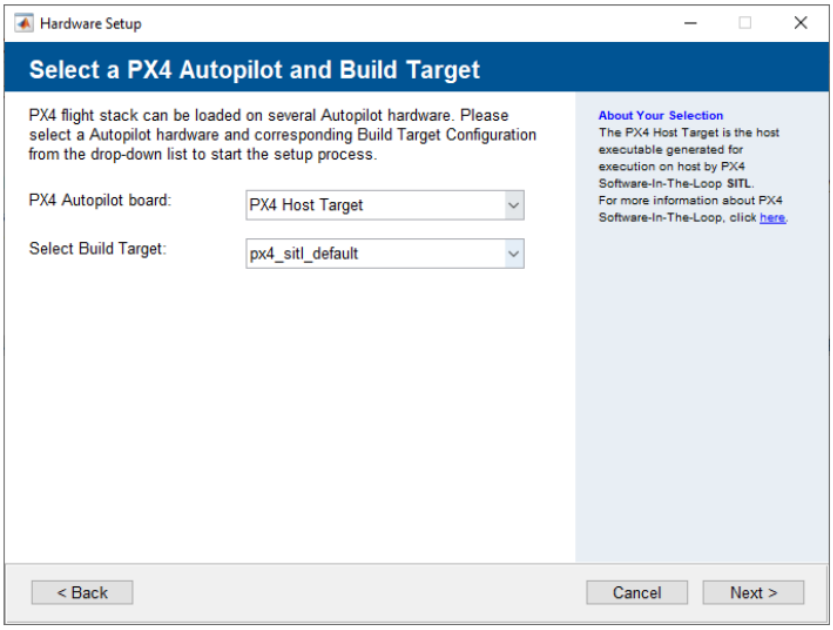
To use the PX4 Host Target with the simulator plant designed in Simulink, follow a two-model approach.

- 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 the 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`.

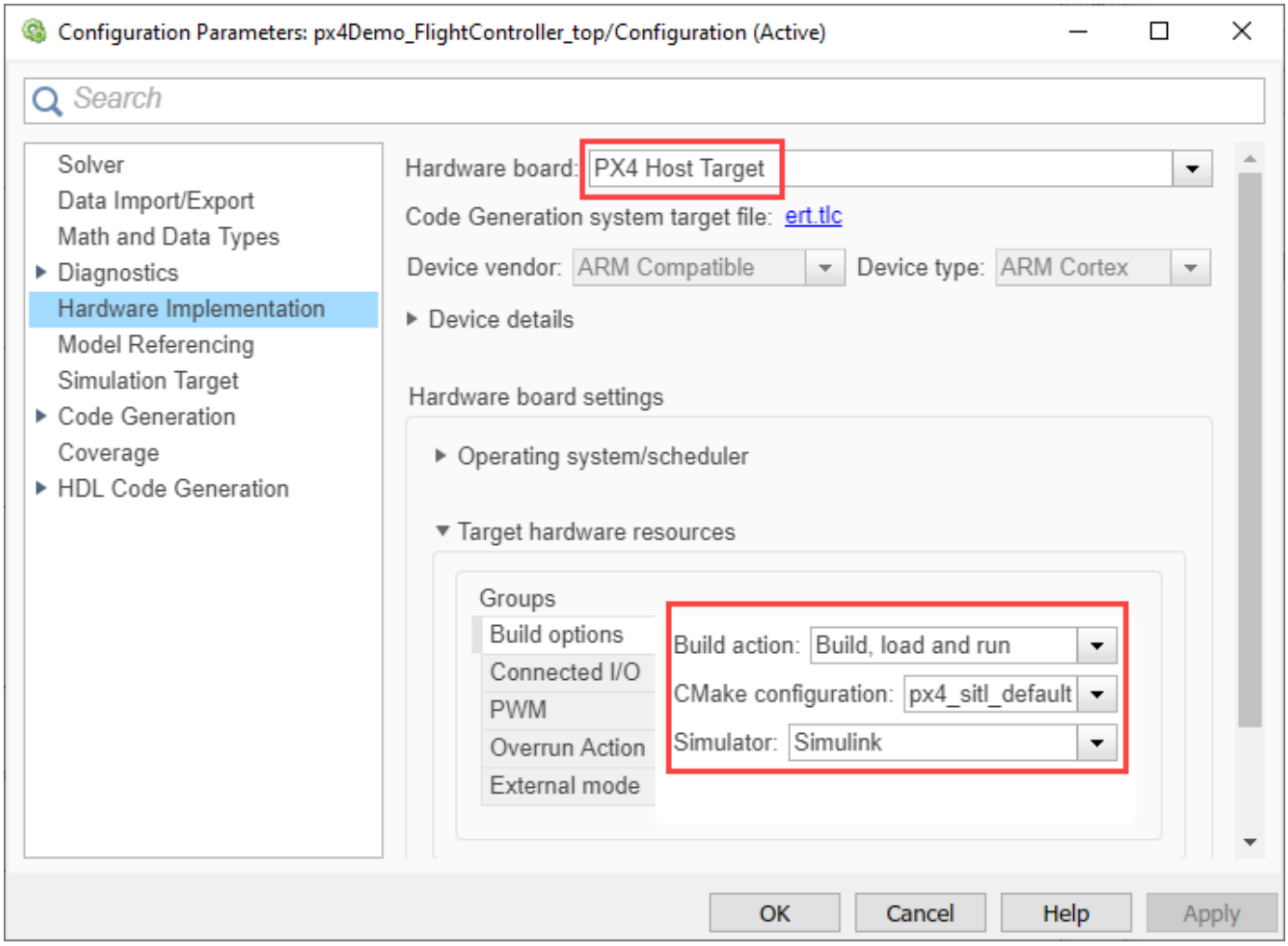


2. Proceed with the subsequent steps in the Hardware Setup process to build the firmware and verify that the build was successful.

Prepare PX4 Host Target Controller Model

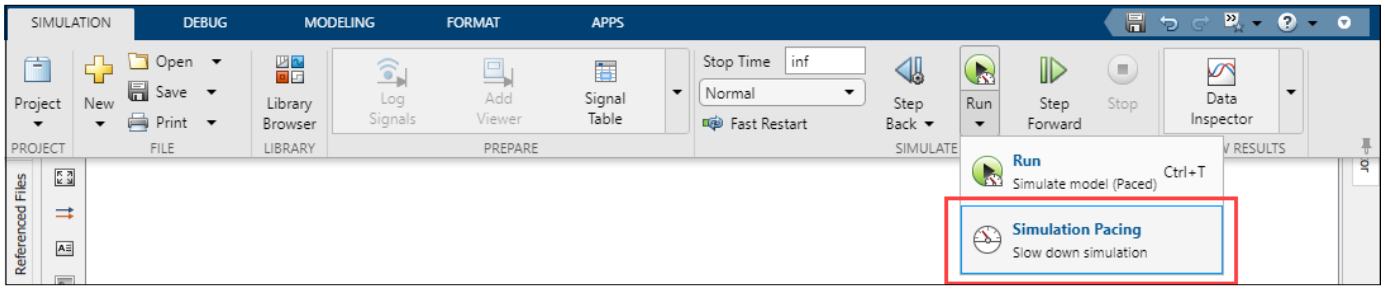
After completing the Hardware setup process, prepare the flight controller algorithm using the Simulink blocks available in the UAV Toolbox Support Package for PX4 Autopilots.

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, click the **Run** button arrow then select **Simulation Pacing**.

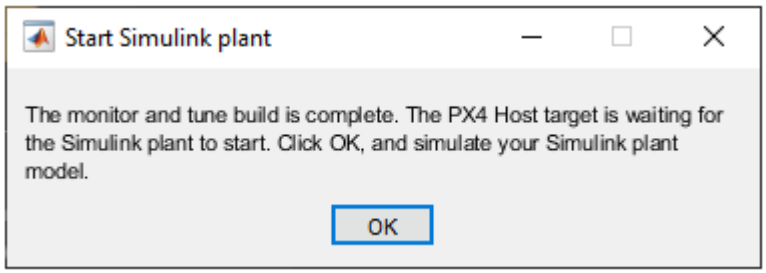


3. In the Simulation Pacing Options dialog box, select **Enable pacing to slow down simulation**. For more information, see Simulation Pacing Options.

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”

Set Up PX4 Firmware for Hardware-in-the-Loop (HITL) Simulation

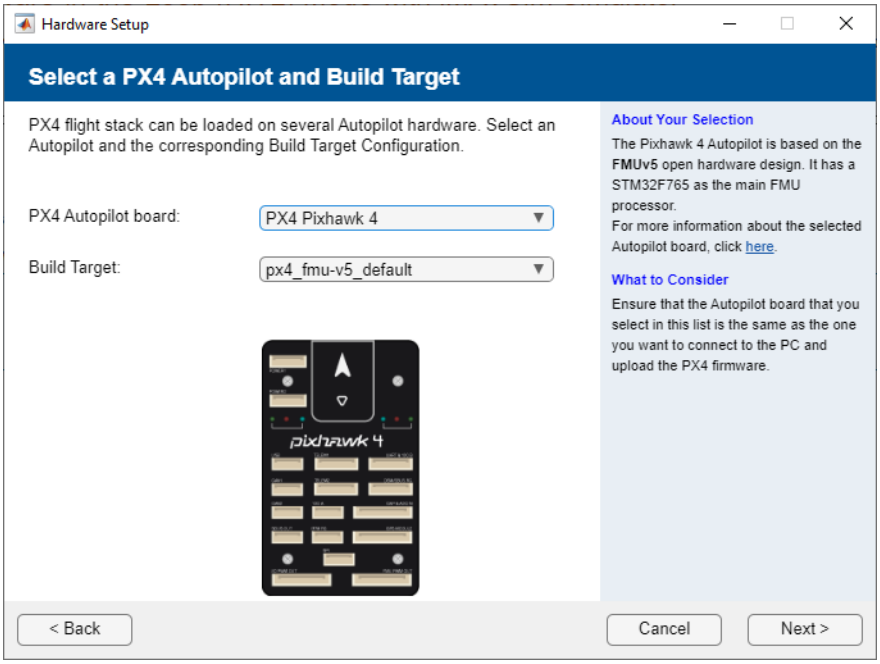
The UAV Toolbox Support Package for PX4 Autopilots provides the option to deploy the controller on the PX4 Autopilot and verify the algorithm by running Hardware-in-the-Loop (HITL) simulation. The workflow is based on Hardware in the Loop Simulation (HITL).

Setting up PX4 Firmware

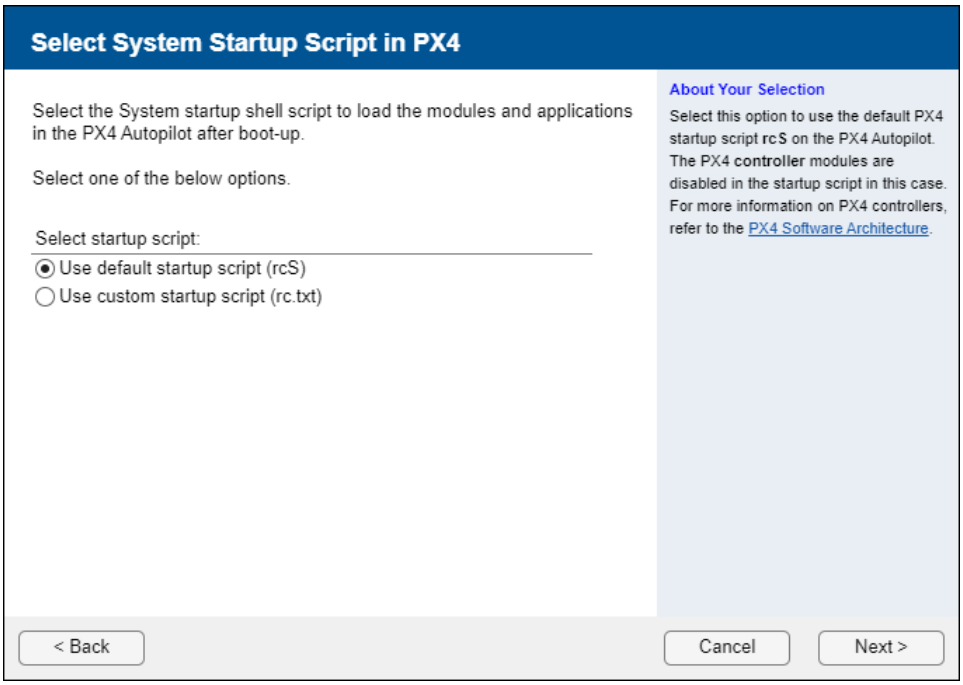
Perform these steps, which are a part of the Hardware Setup process in the UAV Toolbox Support Package for PX4 Autopilots, to enable the PX4 autopilot for HITL simulation.

1 In the **Select a PX4 Autopilot and Build Target** Hardware Setup screen, select any Pixhawk Series board as the Hardware board. For example, **PX4 Pixhawk 4**. Click **Next**.

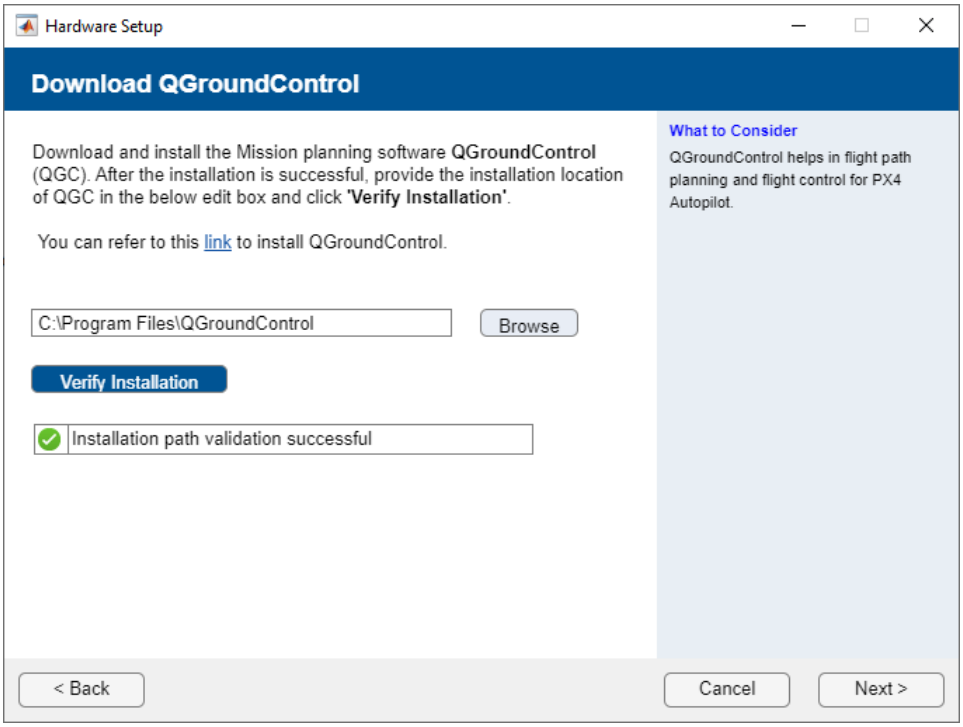
Note This feature supports only PX4 boards and does not support PX4 Host Target.



2 In the **Select System Startup Script in PX4** Hardware Setup screen, select **Use default startup script(rcS)** and click **Next**.



3 In the **Download QGroundControl** Hardware Setup screen, download and install QGC. After installation, click **Verify Installation** and then click **Next**



4 Proceed with the subsequent steps in the Hardware Setup process to build the firmware and verify that the build is successful.

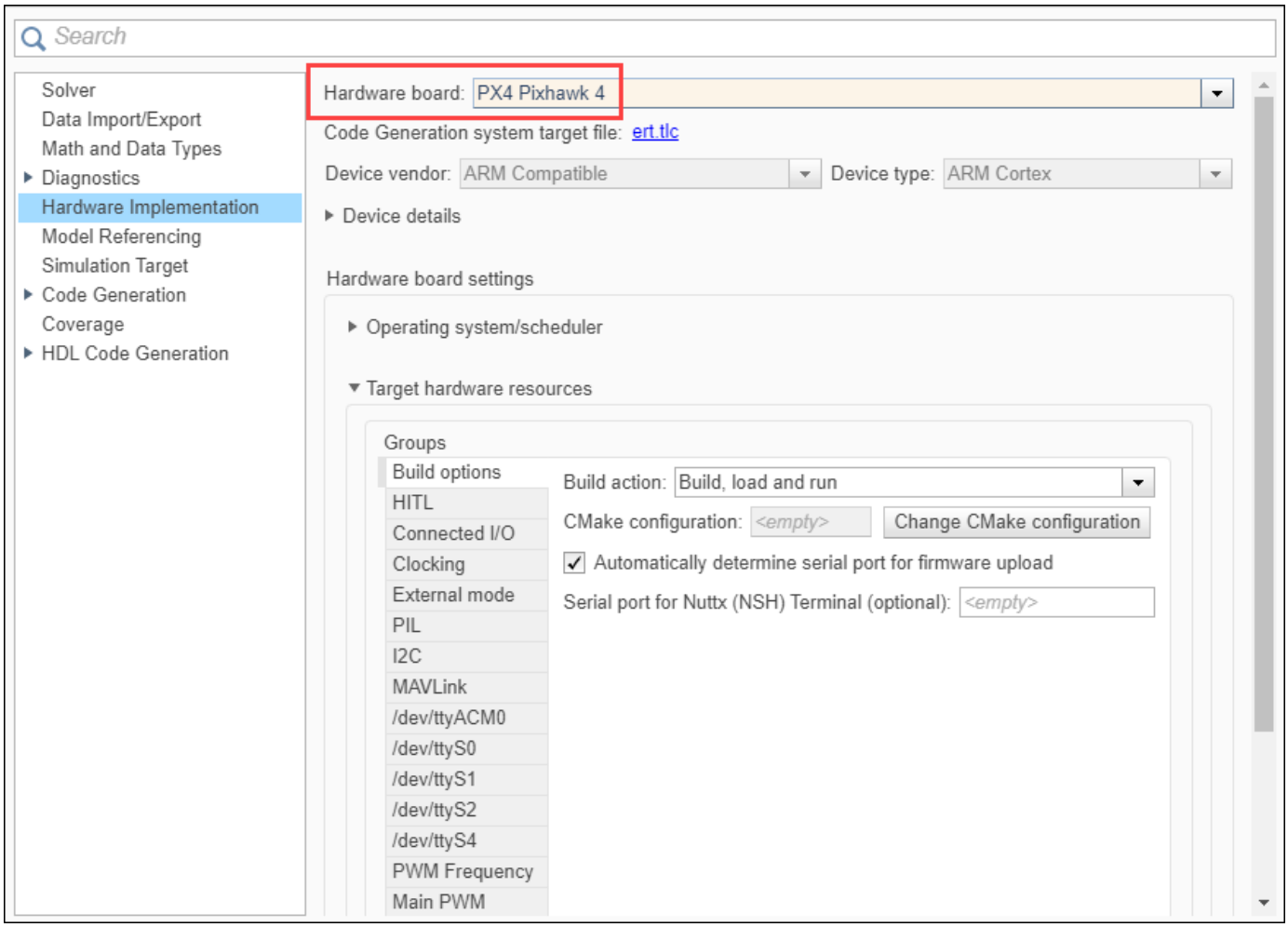
See Also

“Configure Simulink Model for Deployment in Hardware-in-the-Loop (HITL) Simulation” on page 6-12 | “Configure Simulink Model for Monitor & Tune Simulation with Hardware-in-the-Loop (HITL)” on page 6-9

Configure Simulink Model for Monitor & Tune Simulation with Hardware-in-the-Loop (HITL)

Configure PX4 Controller Model in Simulink for Monitor & Tune Simulation

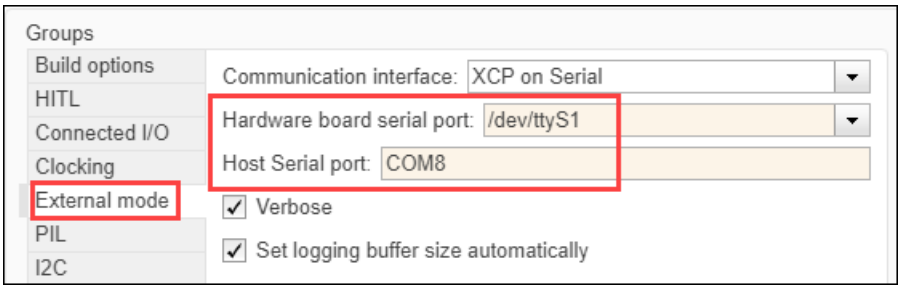
1. In the **Modeling** tab, click **Model Settings**.
2. In the Configuration Parameters dialog box, choose any Pixhawk Series board.



3. Click **External mode** and select `/dev/ttyS1` as the **Hardware board serial port**. The serial port `/dev/ttyS1` corresponds to TELEM1 on Pixhawk 4 over which External Mode communication would be established.

Note If you select PX4 Pixhawk 4 hardware board, then configure **/dev/ttyS1** tab. If you select any other Pixhawk Series board, configure the appropriate **/dev/tty** tab. For example, for PX4 Pixhawk 1 as hardware board, configure **/dev/ttyS6** tab.

For more information, see “Running Monitor & Tune Simulation over FTDI with Pixhawk 6x” on page 4-19.

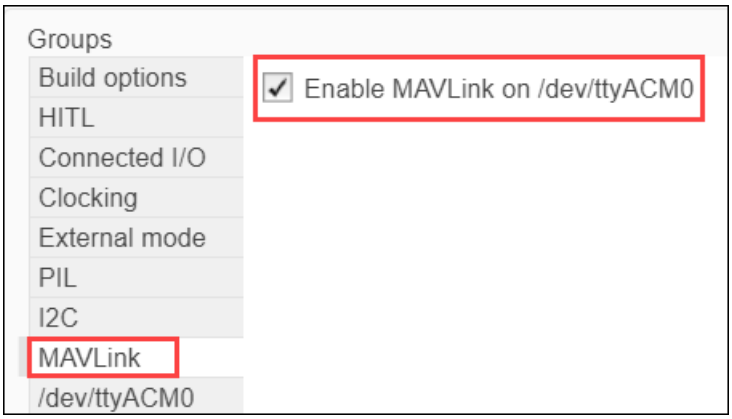


4 Ensure that MAVLink is not enabled in the TELEM1 port of Pixhawk 4 (or any other Pixhawk 4 serial port that you want to use for External Mode communication) as part of MAV_0_CONFIG or MAV_1_CONFIG or MAV_2_CONFIG.

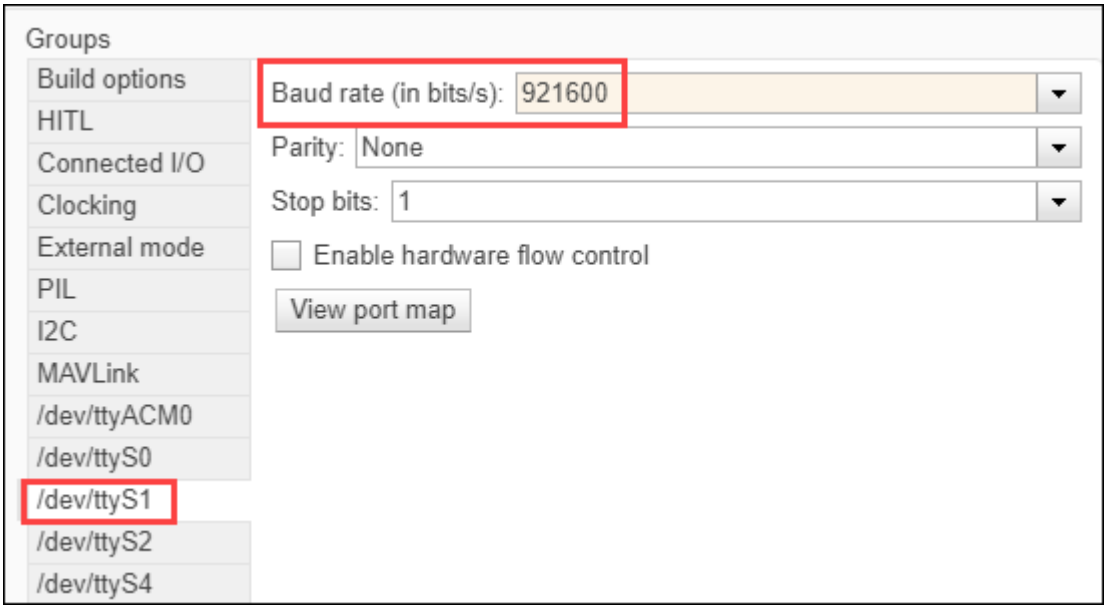
5 Clear the **Use the same host serial port for External mode as used for firmware upload** option.

6 In the **Host Serial port** parameter, enter the host serial port number on the host computer for External mode communication. This is usually the COM port of the FTDI that is connected with the /dev/ttyS1 (TELEM1) serial port on Pixhawk 4.

7 Click **MAVLink** and ensure that **Enable MAVLink on /dev/ttyACM0** option is selected.



8 Click **/dev/ttyS1** and set the **Baud rate** to `921600`. Do not change any other parameters.



9 Click **Apply** and then click **OK**.

See Also

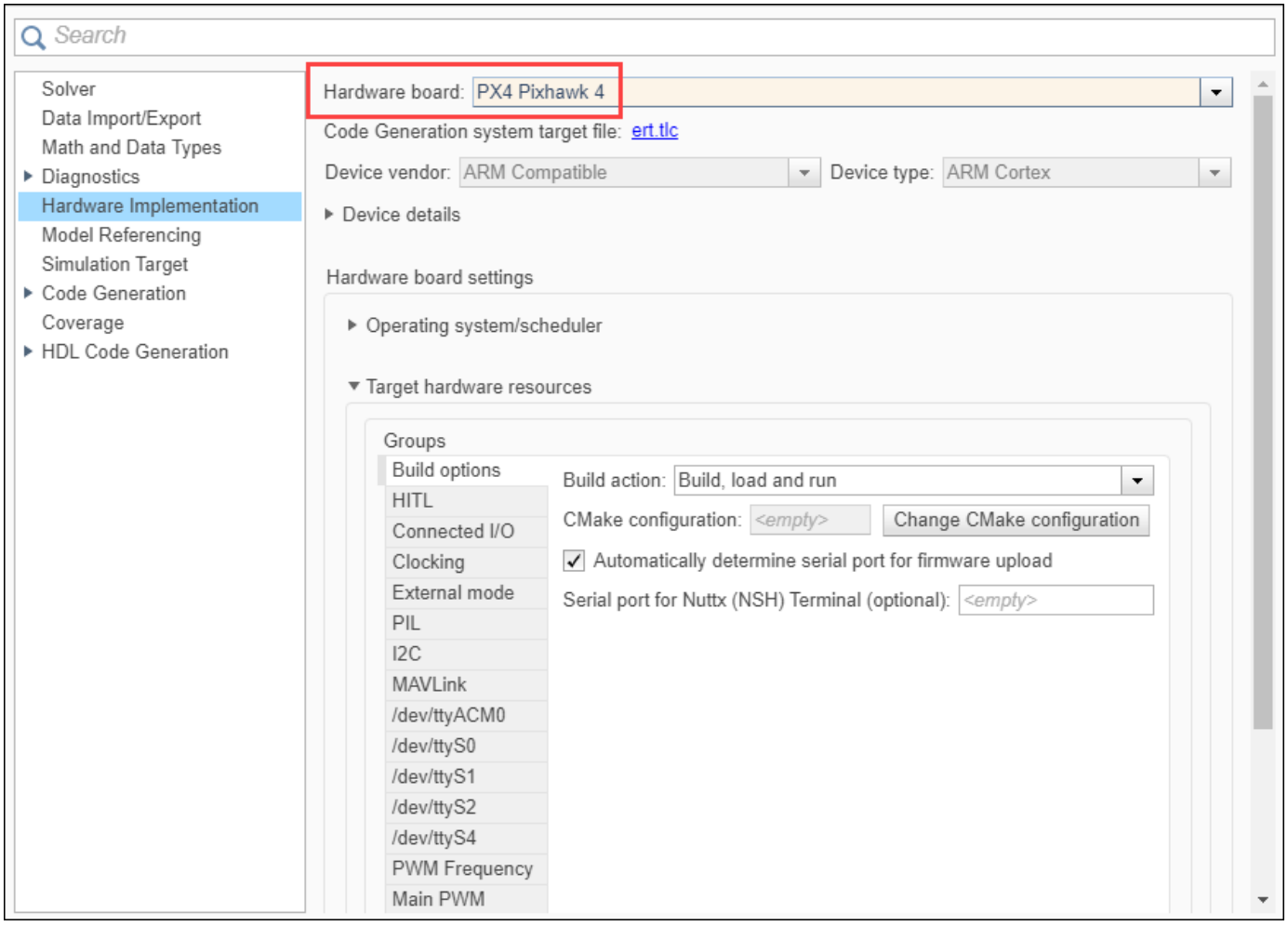
“Set Up PX4 Firmware for Hardware-in-the-Loop (HITL) Simulation” on page 6-6 | “Configure Simulink Model for Deployment in Hardware-in-the-Loop (HITL) Simulation” on page 6-12

Configure Simulink Model for Deployment in Hardware-in-the-Loop (HITL) Simulation

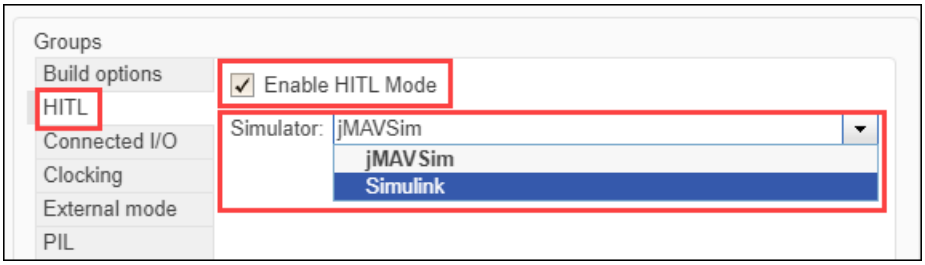
Configure PX4 Controller Model in Simulink

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. Perform these steps to configure the Simulink model.

1. In the **Modeling** tab, click **Model Settings**.
2. In the Configuration Parameters dialog box, choose any Pixhawk Series board.

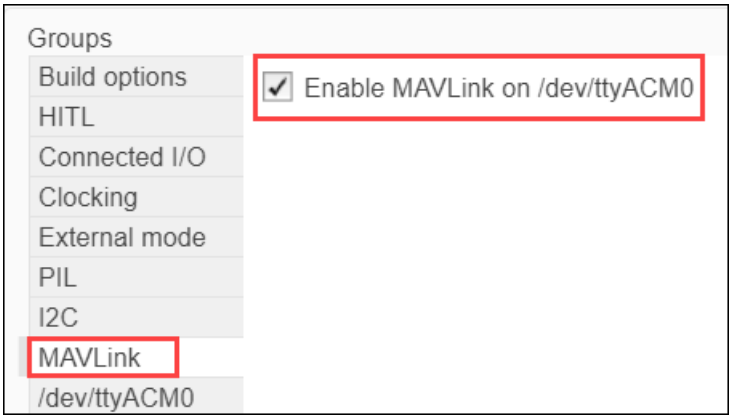


3. Click **HITL** and then select **Enable HITL Mode**.



4 Select `Simulink` as Simulator.

5 Click **MAVLink** and ensure that **Enable MAVLink on /dev/ttyACM0** option is selected.



6 Click **Apply** and then click **OK**.

See Also

"Set Up PX4 Firmware for Hardware-in-the-Loop (HITL) Simulation" on page 6-6 | "Configure Simulink Model for Monitor & Tune Simulation with Hardware-in-the-Loop (HITL)" on page 6-9

Related Examples

- "PX4 Hardware-in-the-Loop (HITL) Simulation with VTOL UAV Tilt-Rotor Plant in Simulink"

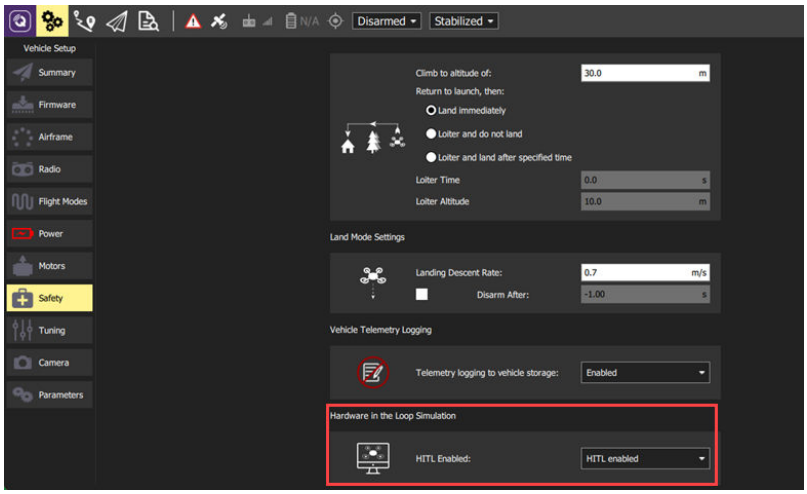
Setting Up PX4 Autopilot in Hardware-in-the-Loop (HITL) Mode from QGroundControl

The PX4 Autopilot must be configured in Hardware-in-the-Loop (HITL) mode before setting it up for HITL Simulation. The workflow is based on Hardware in the Loop Simulation (HITL).

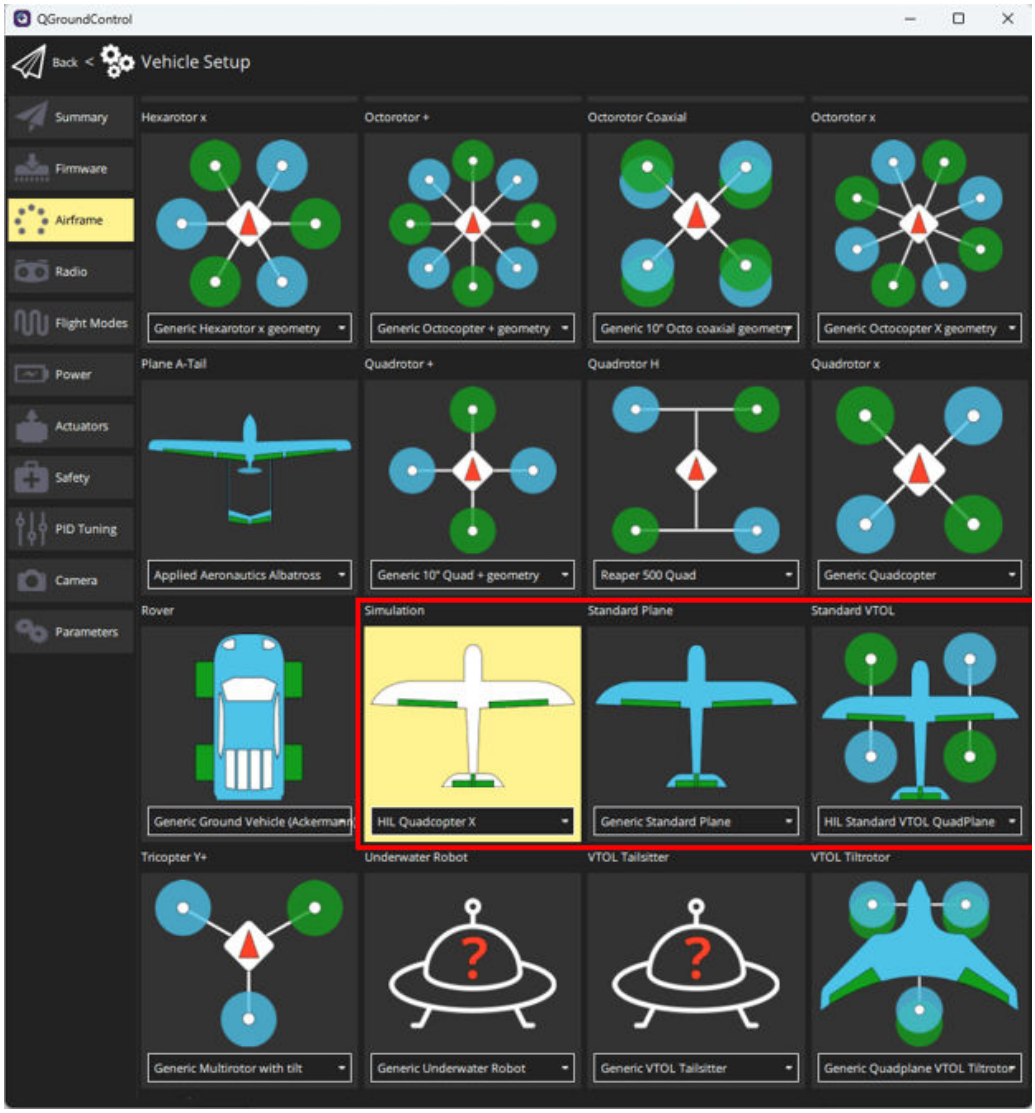
Setting up PX4 Autopilot in HITL Mode

Perform these steps to configure and setup QGroundControl (QGC).

1. Connect the autopilot to QGC via USB.
2. Enable HITL mode.
   a. Navigate to **Setup** > **Safety** section.
   b. Select **Enabled** from the **HITL Enabled** list.



3. Select Airframe.
   a. Navigate to **Setup** > **Airframes**.
   b. Select `HIL QuadCopter x` to simulate a quadcopter or select `Generic Standard Plane)` to simulate a fixed-wing plane. For VTOL, select `HIL Standard VTOL QuadPlane`. Click **Apply and Restart** on top-right of the **Airframe Setup** page.



4 In the **General** tab of the settings menu, clear all **AutoConnect** options except **UDP**.

Note This selection ceases the communication between QGC and PX4 Autopilot over USB. The **QGC can now communicate only over UDP** (this is usually taken care by a Simulator which acts as a bridge between QGC and Autopilot and routes MAVLink data between QGC and Autopilot over UDP).

Note This selection is done only for HITL mode. To run Speedgoat® models, select **Pixhawk** option also.

---

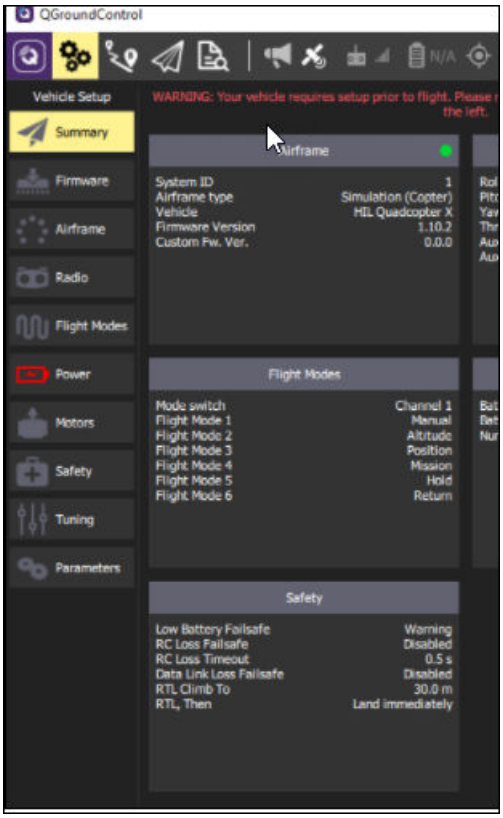
5 Select Virtual Joystick.

a In the **General** tab, from **Fly View** of the settings menu, select Virtual Joystick option.

6 Configure Joystick and Failsafe. Set the following parameters to use a joystick instead of an RC remote control transmitter:

- COM_RC_IN_MODE to `Joystick only`. This allows joystick input and disables RC input checks.
- COM_DISARM_LAND to `-1`. This disables the timeout for auto-disarm when QGC detects a landing. This helps in avoiding a potential PX4 failsafe when drone motors stops at highest take-off point momentarily.
- NAV_ACC_RAD to `10`.

7 Ensure there are no warnings in the **Vehicle Settings** tab of QGC. Only the **Power** tab can be in red (with warning). If you see warnings for the **Flight Modes** tab, set it up by picking a mode for each of the channels. Sometimes, the **Sensor Calibration** tab can be in red. You can go ahead with it.



Note Ensure that SD card is inserted in the hardware for the mission upload from QGC to work.

8 If you plan to run Monitor & Tune simulation for the Controller model on the TELEM1, TELEM2, or TELEM3 port of the Pixhawk hardware, ensure that MAVLink is not enabled on these ports. You can verify this by checking the values of the parameters `MAV_0_CONFIG` and `MAV_1_CONFIG`. The port value in these parameters must be disabled when running the Monitor & Tune simulation.

9 Close the QGC.

See Also

“Deployment and Verification Using PX4 Host Target and jMAVSim/Simulink” on page 4-23

Configure and Assign Actuators in QGC

For Firmware version 1.14, the PWM & AUX pins need to be configured in QGroundControl (QGC) before the PX4 Actuator Write block can be used in Simulink. The configurations for QGroundControl described here are based on version 4.3.0.

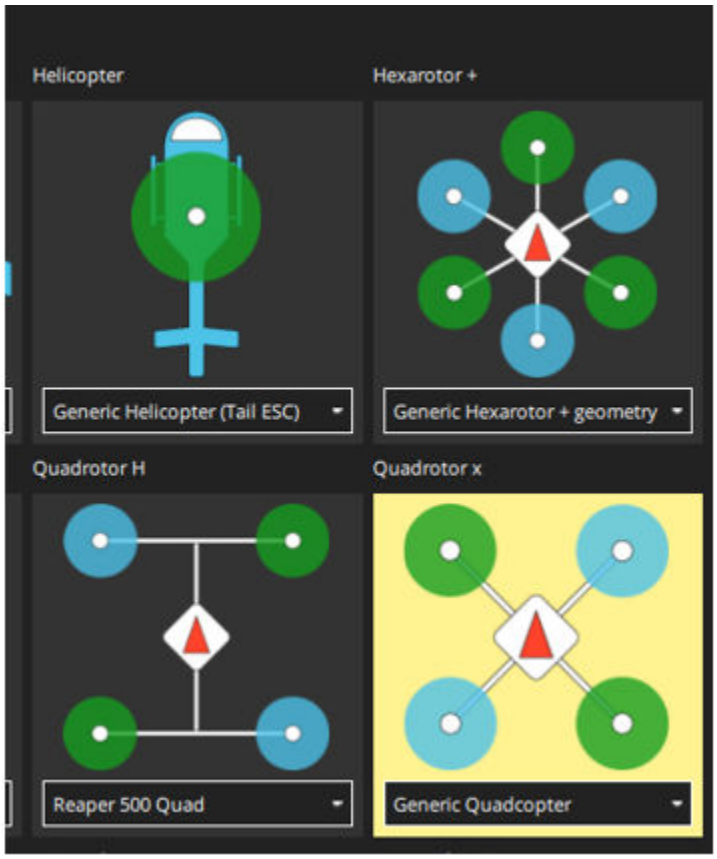
Depending on your vehicle's airframe, you might need to configure and allocate actuators for motors and servos. The following sections provide a detailed explanation of the necessary steps.

Step1: Open QgroundControl and establish connection with PX4 autopilot

1. Enable MAVLink over USB (/dev/ttyACM0) for QGC connectivity. For more information, see “Enabling MAVLink in PX4 Over USB” on page 4-6.
2. Launch QGroundControl (QGC) and allow it to connect to your Autopilot.

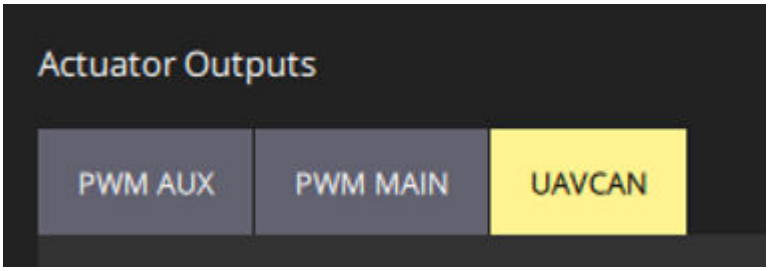
Step2: Choose Airframe

Select an Airframe for your vehicle from **Vehicle Setup** -> **Airframe** tab, if you have not chosen one yet. Reboot the hardware to apply this change.



Step3: Actuator Configuration and Actuator outputs assignment

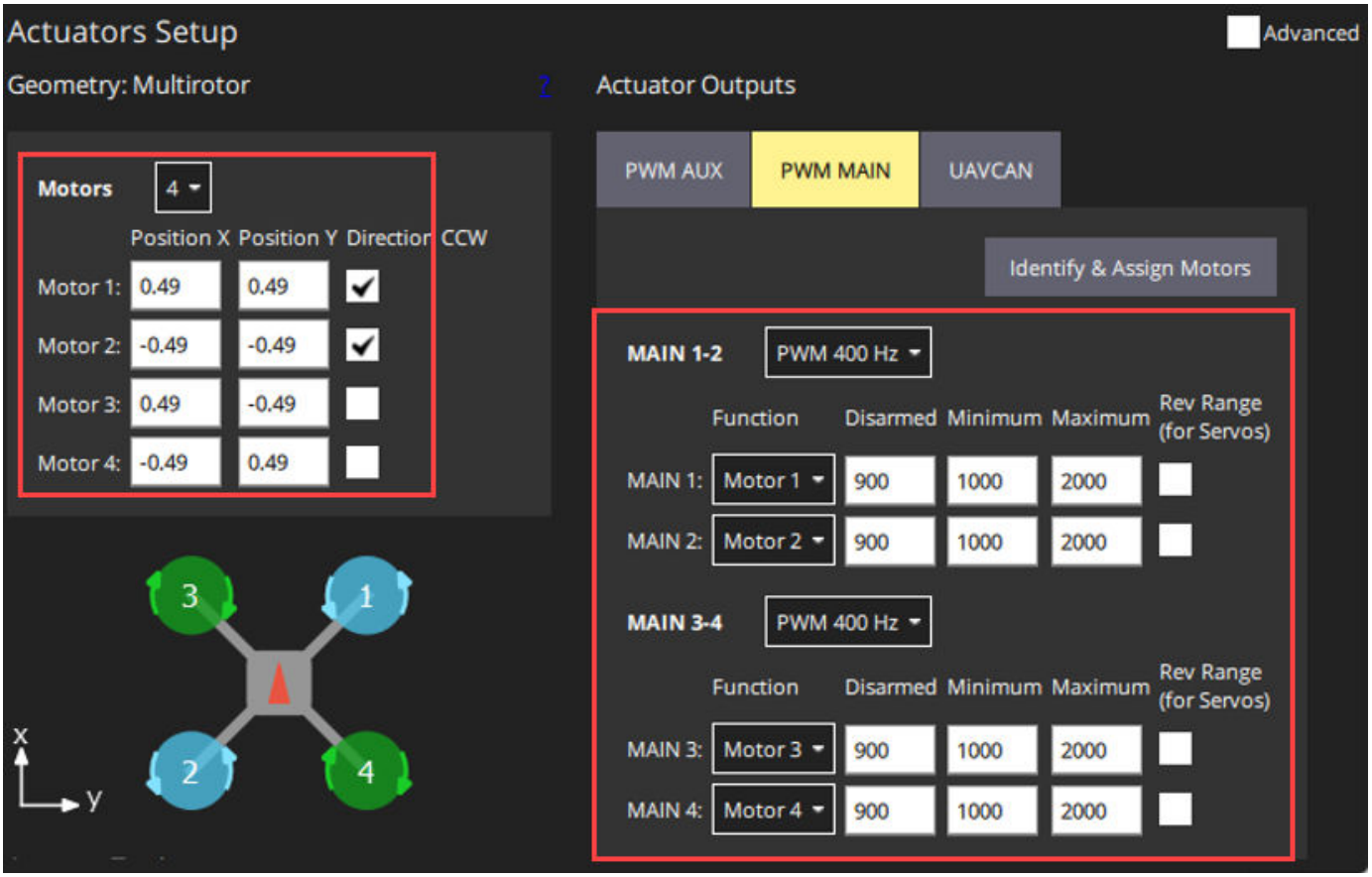
1. Once QGC is connected, go to **Vehicle Setup** -> **Actuators** tab.
2. Based on the Airframe selection, you will see the corresponding number of motors and servos displayed. The interface also shows tabs for actuator output assignment (PWM AUX, PWM MAIN, UAVCAN, and so on.). For more information on actuator configuration, see PX4 documentation.



Depending on whether you want to write to motors or servos and choose to use the PWM or UAVCAN/DroneCAN interface for actuator communication, select the appropriate output assignment in the corresponding tabs. Refer to the following sections for more information.

Configure Motors over PWM MAIN

If you select the `Generic Quadcopter` airframe, an option to configure four motors appears. In this scenario, you can configure the four motors using PWM Main channels `1-4`.

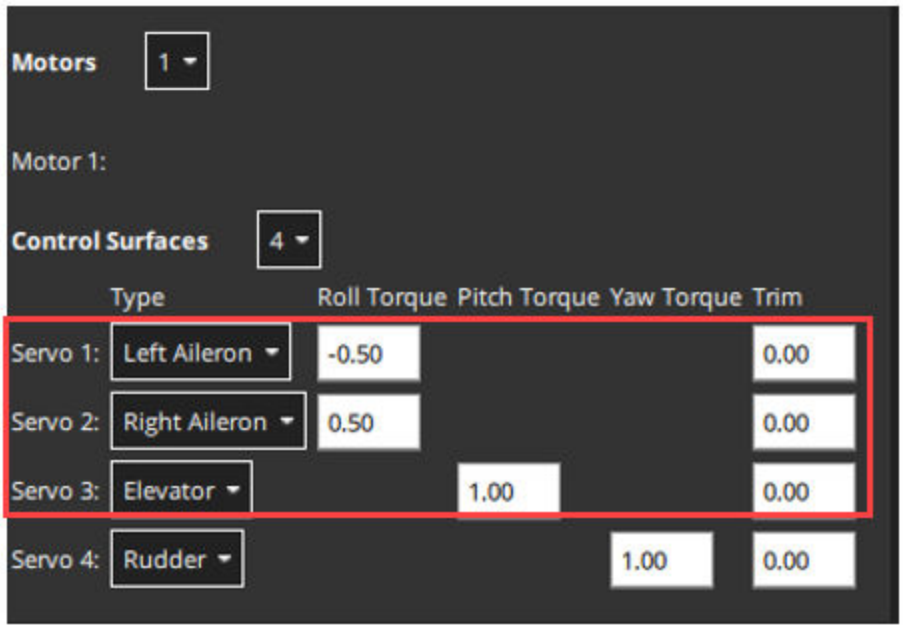


The number of Main channels used should match the required number of motors.

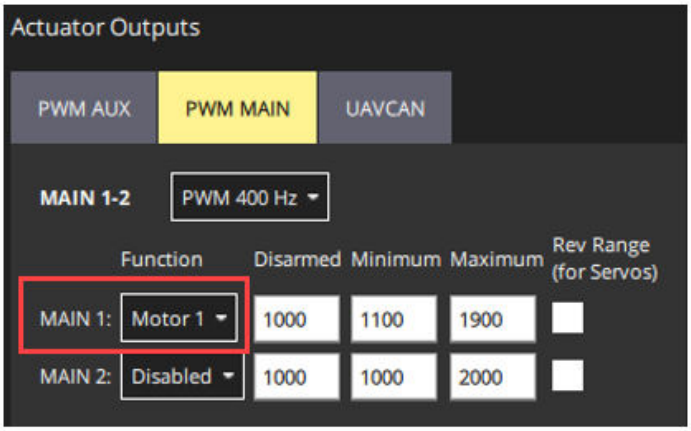
It recommended to use only PWM Main channels for Motors connected over PWM interface.

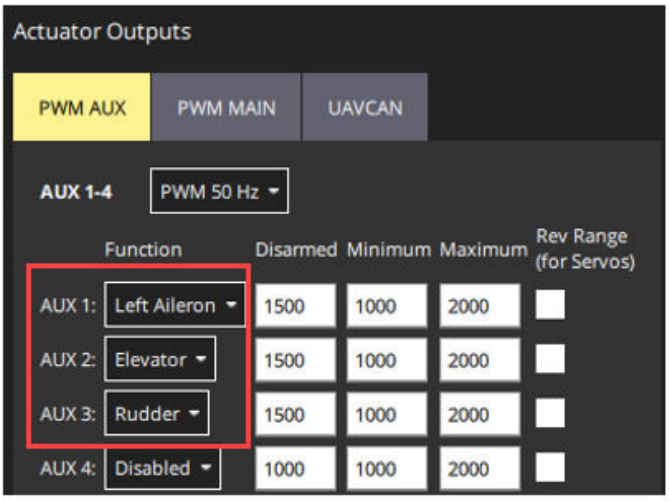
Configure Servos over AUX PWM

If you select the `Generic Standard Plane` airframe, an option to configure one motor and four servos appears by default. If you need only three servos, you can change the configuration to three.



Motor can be assigned to PWM Main channels 1 as described in step3.1. Servos can be assigned to Aux channel `1-3`.



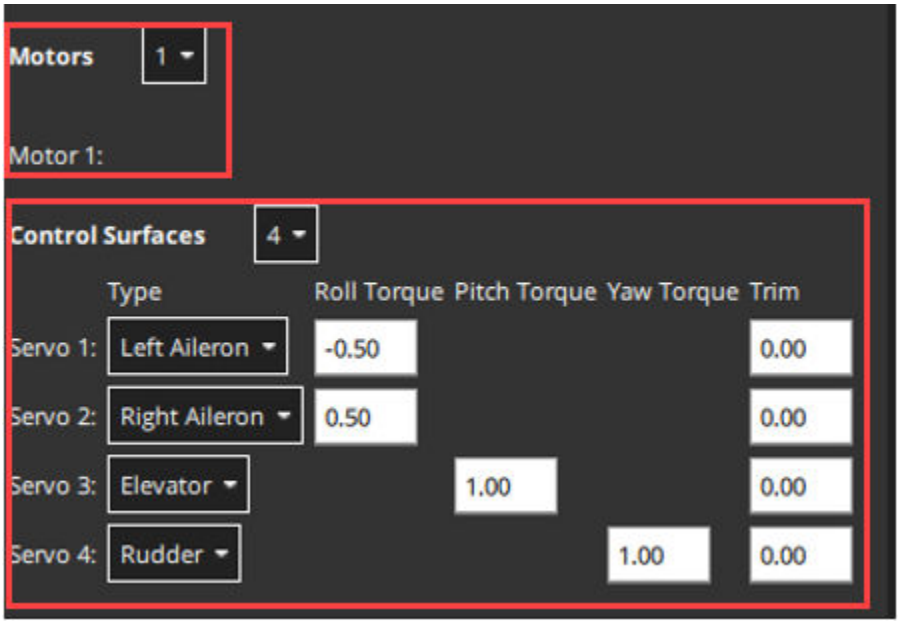


Depending on the number of servos needed, use the corresponding number of Aux channels. It is recommended to use only PWM Aux channels for servos connected over the PWM interface.

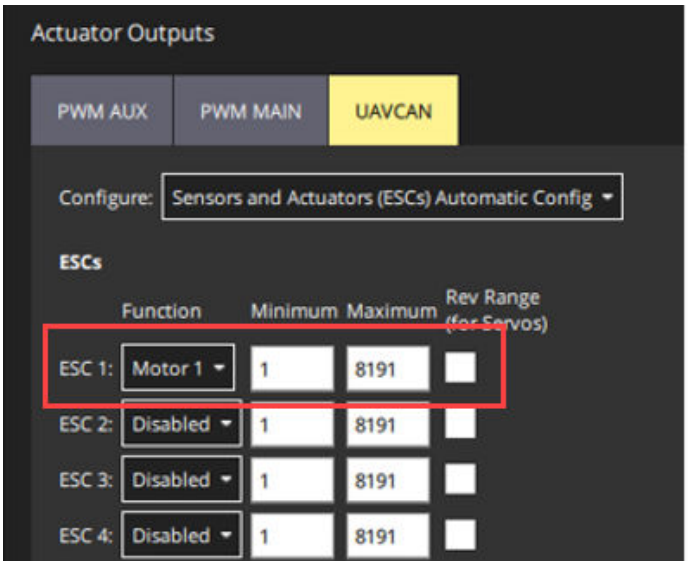
Configure Motors and servos over UAVCAN

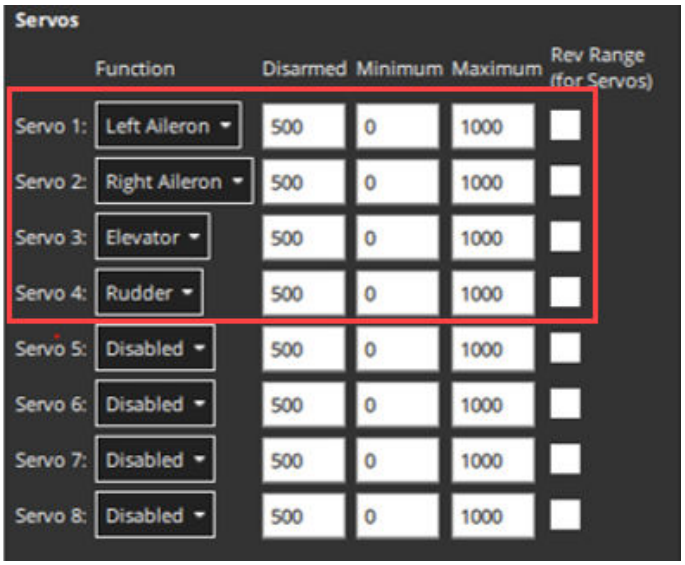
Note UAVCAN interface is supported only with PX4 Actuator Write block and not with PX4 PWM output block.

For example, if you choose `Generic Standard Plane` airframe you will see an option to configure one motor and four servos by default.



The motor can be assigned to UAVCAN ESC channel 1, while servos can be assigned to Servo channels 1-4.



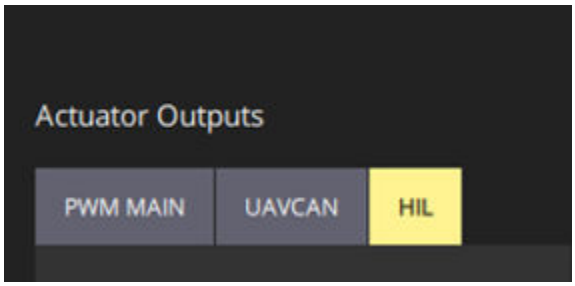


Depending on the number of motors and servos needed, use the corresponding number of ESC channels and Servo channels respectively. It is recommended to use only ESC channels for motors and servos channels for Servos connected over the UAVCAN interface.

Configure Motors and Servos for HITL

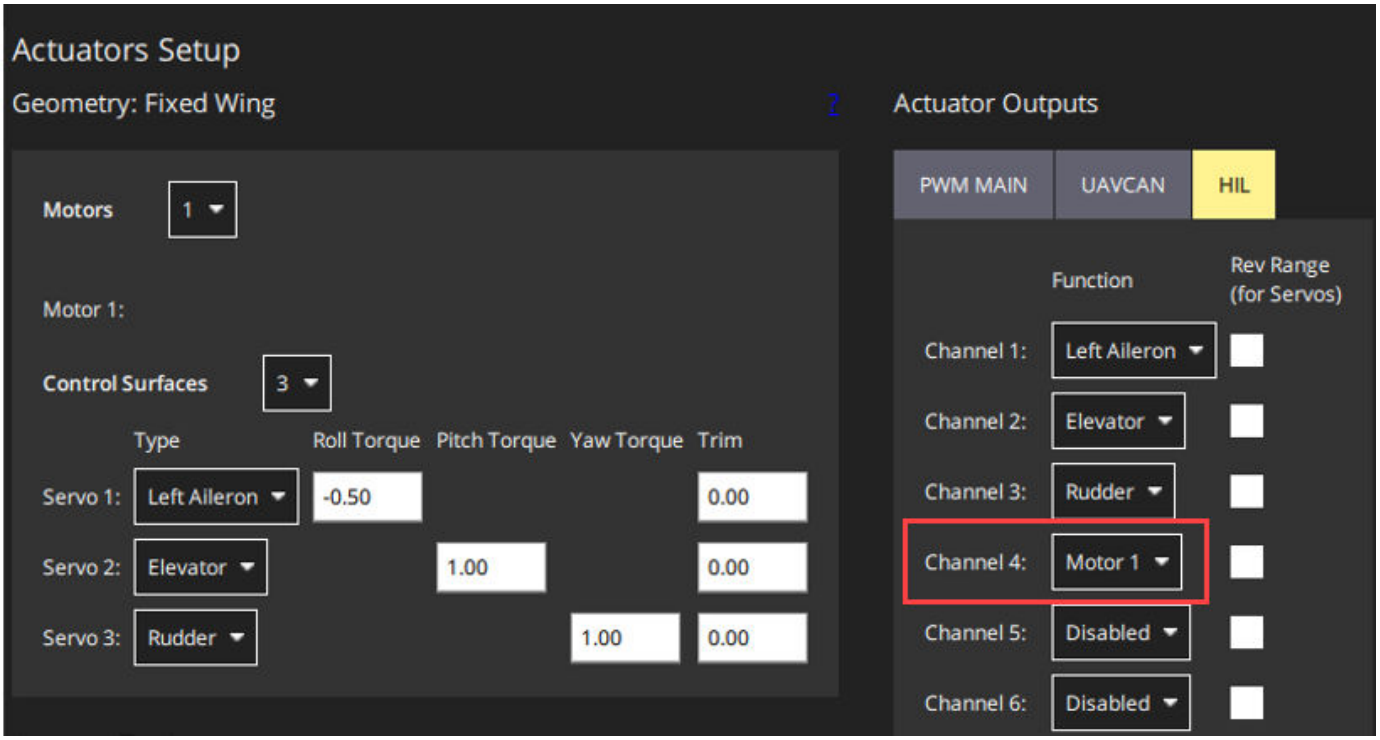
Note HITL actuator configuration is necessary only for the PX4 Actuator Write block and is not required for the PX4 PWM Output block.

For example, selecting the `Generic Standard Plane` airframe and enabling HITL on page 6-14 will show a new tab labeled **HIL** in the Actuator Outputs section.



By default, the option to configure one motor and four servos is shown. If you require only three servos, you can adjust this to three.

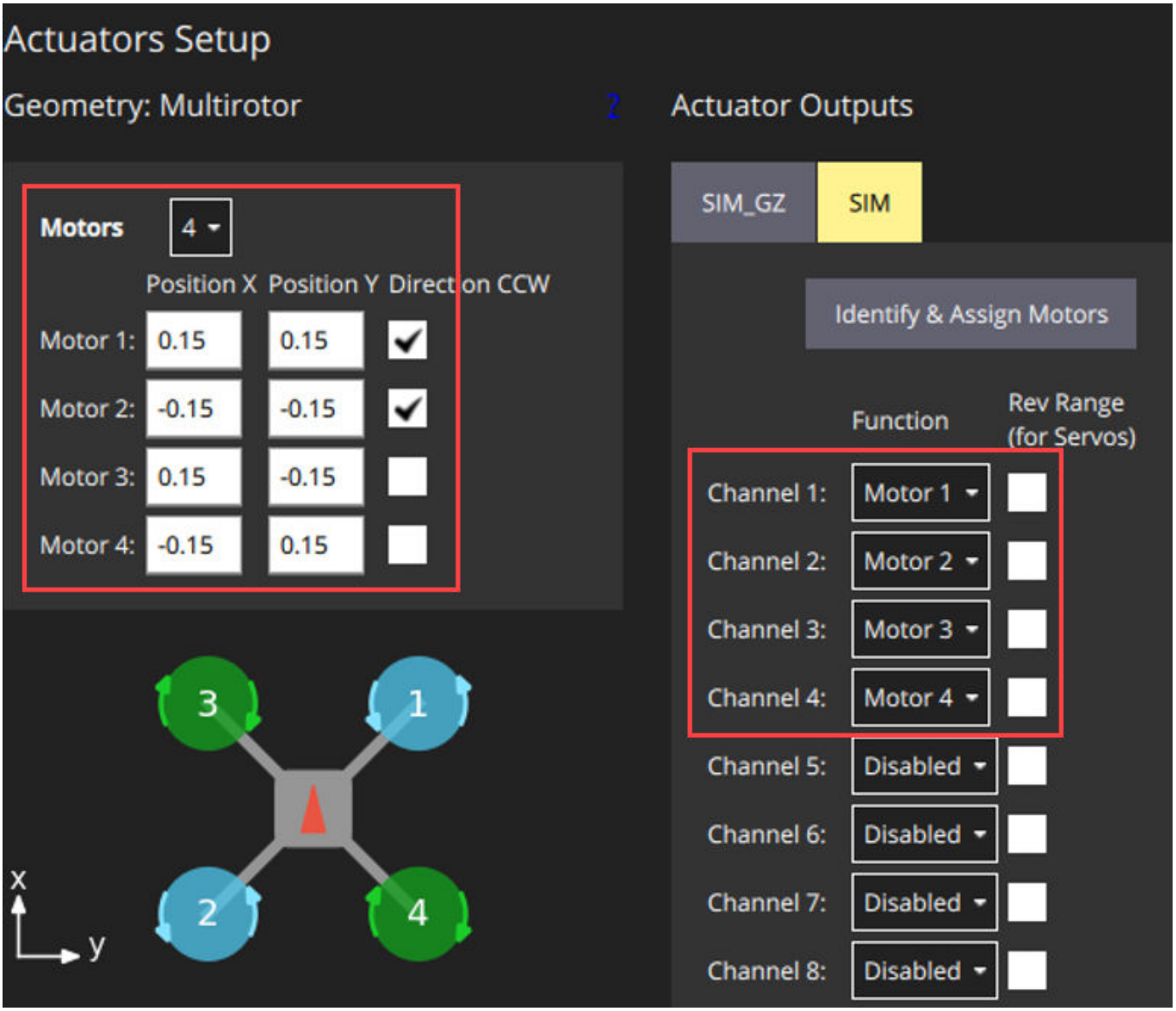
Motors and Servos can be assigned to the HIL channels based on the specific needs of your UAV vehicle physics simulation and how Simulation reads the actuator data. For example, if your physics simulation uses control channel 4 of the HIL_ACTUATOR_CONTROLS MAVLink message to simulate motor thrust, then the motor should be assigned to the fourth channel accordingly.



Configure Motors and Servos for PX4 Host Target

Note The PX4 Host Target actuator configuration is necessary only for the PX4 Actuator Write block and not for the PX4 PWM Output block.

By default, in the PX4 host target, an option to configure four motors is shown. These four motors will also be automatically assigned to the first 4 channels in the **SIM** tab, by default.



If you select a different airframe, then you must assign the motors and servos accordingly in the **SIM** tab.

See Also

PX4 Actuator Write

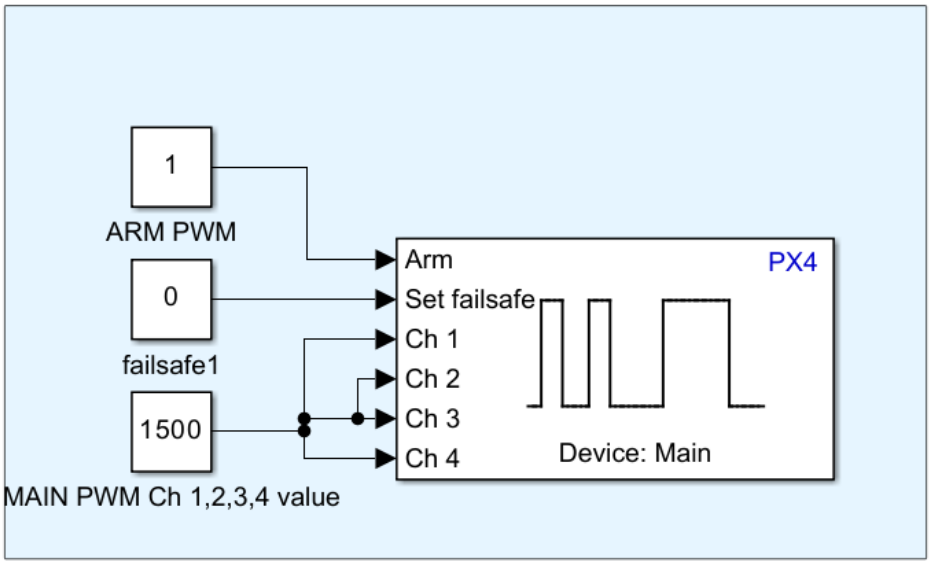
Convert PX4 PWM Output Block to PX4 Actuator Write Block

The PX4 PWM Output block will be removed in a future release. It is recommended to use the PX4 Actuator Write block as a replacement. To convert existing PX4 PWM Output blocks in your Simulink model to PX4 Actuator Write blocks, follow the steps listed in this section.

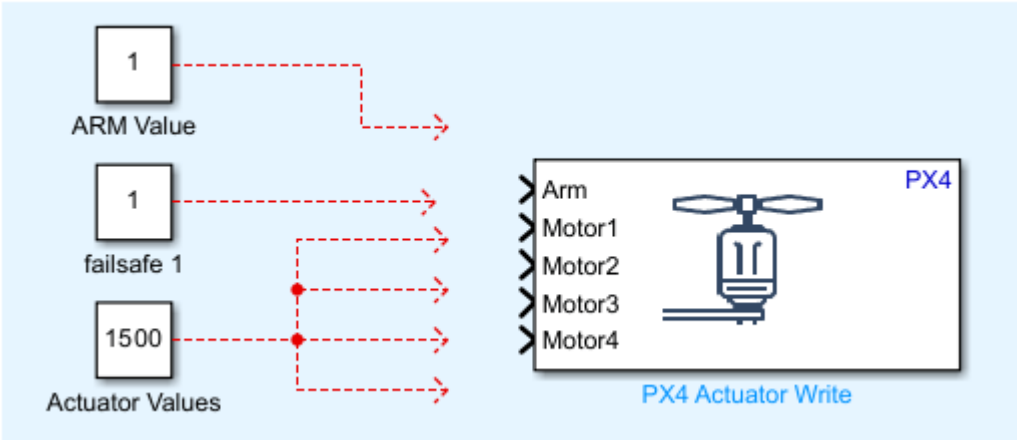
The key difference between these blocks is in the input value range and type. For the PX4 PWM Output, inputs are PWM on-time values with a `uint16` type. However, the PX4 Actuator Write block requires inputs to be normalized values of type `single`, within a range of `[0, 1]` for motors and `[-1, 1]` for servos.

To convert the block, perform these steps:

1. Open your Simulink model that includes the PX4 PWM Output block. Here is an example model with an **ARM PWM** signal, **failsafe**, and four **MAIN PWM** channels connected to PWM Output block.



2. Select the PX4 PWM Output block and press the **Delete** key on your keyboard to remove it.
3. Add a PX4 Actuator Write block into your model and establish the necessary connections.

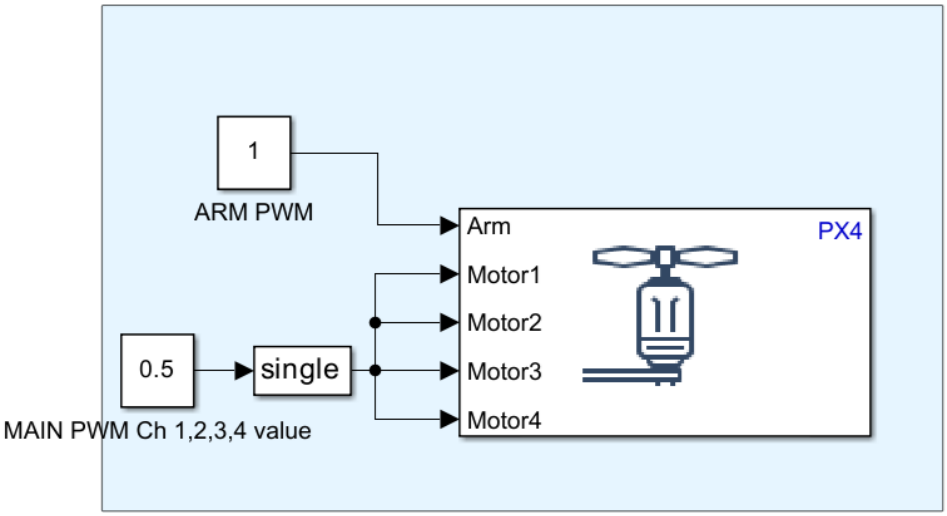


a Link the **ARM PWM** signal to the Arm input of the PX4 Actuator Write block.

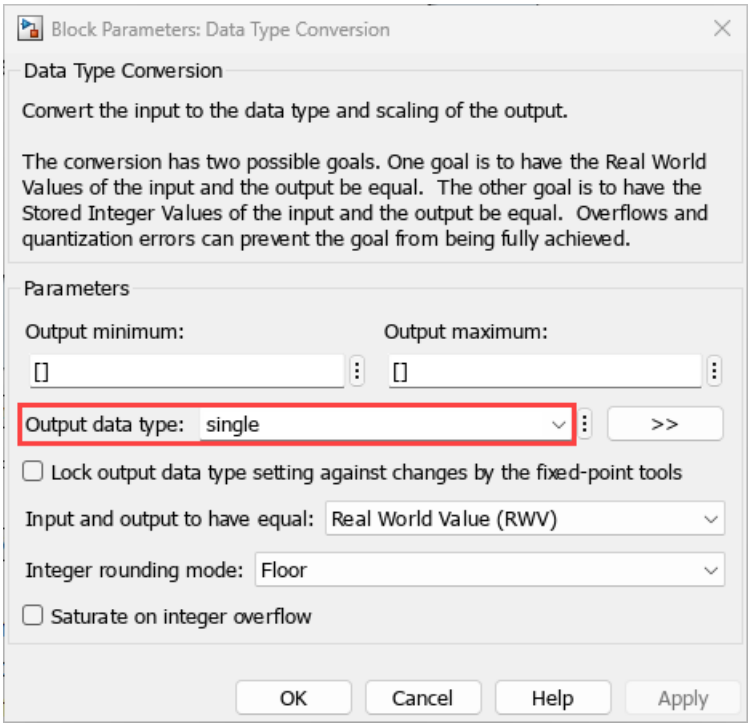
b Remove the **failsafe** signal, as it is not used by the PX4 Actuator Write block.

c Adjust the **MAIN PWM** signal values. For the minimum and maximum values from `1000` to `2000`, the corresponding actuator values will range from `0` to `1`. So for the value `1500`, actuator value will be `0.5`.

4 Map the PWM Output block channels to the Motors inputs on the Actuator Write block. Since the Motors inputs only accept Single scalar values, insert a Data Type Conversion block to modify the channel values accordingly.



5 Open the Data Type Conversion block settings by double-clicking it. In the Block Parameters dialog, set the **Output data type** to `single`, then click **Apply** and **OK**.



Your model should now reflect the changes, with the PX4 PWM Output block successfully converted to a PX4 Actuator Write block.

See Also

PX4 Actuator Write

MAVLink Connectivity for QGC, On-board Computer and Simulink Plant

Introduction

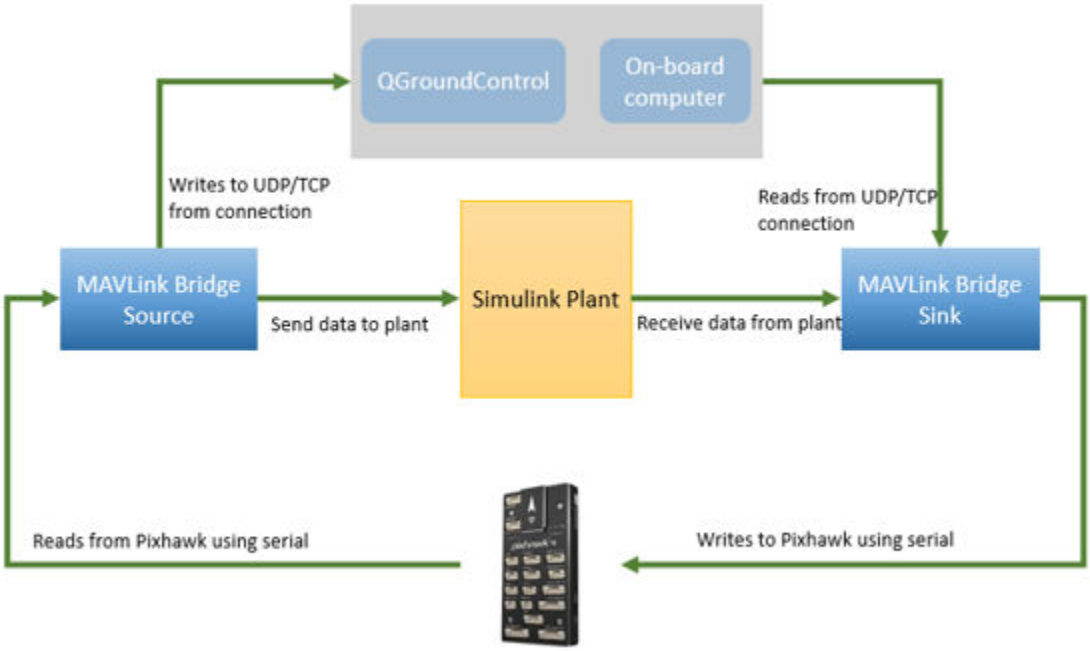
During the HIL simulation, both QGroundControl(QGC) and the plant model must communicate with the Pixhawk hardware over serial. However, only one of these applications can hold the serial port and communicate with the PX4 hardware during the HIL Simulation. Hence, the application which communicates with the Pixhawk hardware must also bridge the MAVLink communication between the Pixhawk and the other applications such as onboard computer workflow.

In PX4 HITL Workflow, the simulator (Gazebo or jMAVSim) acts as a bridge to share MAVLink data between Pixhawk and QGroundControl.

To replace the plant dynamics with the Simulink Plant for HIL simulation, an internal bridge to share the MAVLink data with the other applications is required. MAVLink connectivity provides a way to route MAVLink data stream from Pixhawk hardware to the mission planning software QGroundControl(QGC), On-board computer hardware, Simulink Plant, and vice versa in HITL workflow.

PX4 HITL Workflow with Simulink Plant and MAVLink Bridge Blocks

The following diagram illustrates high level interface between multiple components used in PX4 HITL workflow.



See Also

MAVLink Bridge Source | MAVLink Bridge Sink

PX4 Hardware-in-the-Loop System Architecture

Hardware-in-the-Loop simulation (HITL) in PX4 is a simulation mode in which normal mode PX4 Firmware is run on real PX4 hardware. The workflow is based on Hardware-in-the-Loop simulation (HITL).

PX4 HITL System Diagram

When the autopilot is configured in the HITL configuration,

- No sensors are started on hardware and the actuators are disabled.
- The autopilot is connected to a UAV dynamics simulator (jMAVSim, Gazebo and so on) over an USB.
- The simulator also bridges the data between autopilot and QGroundControl over UDP.
- The simulator can also bridge data between autopilot and another computer.

The various modules in PX4 HITL are connected to each other as shown in this image.

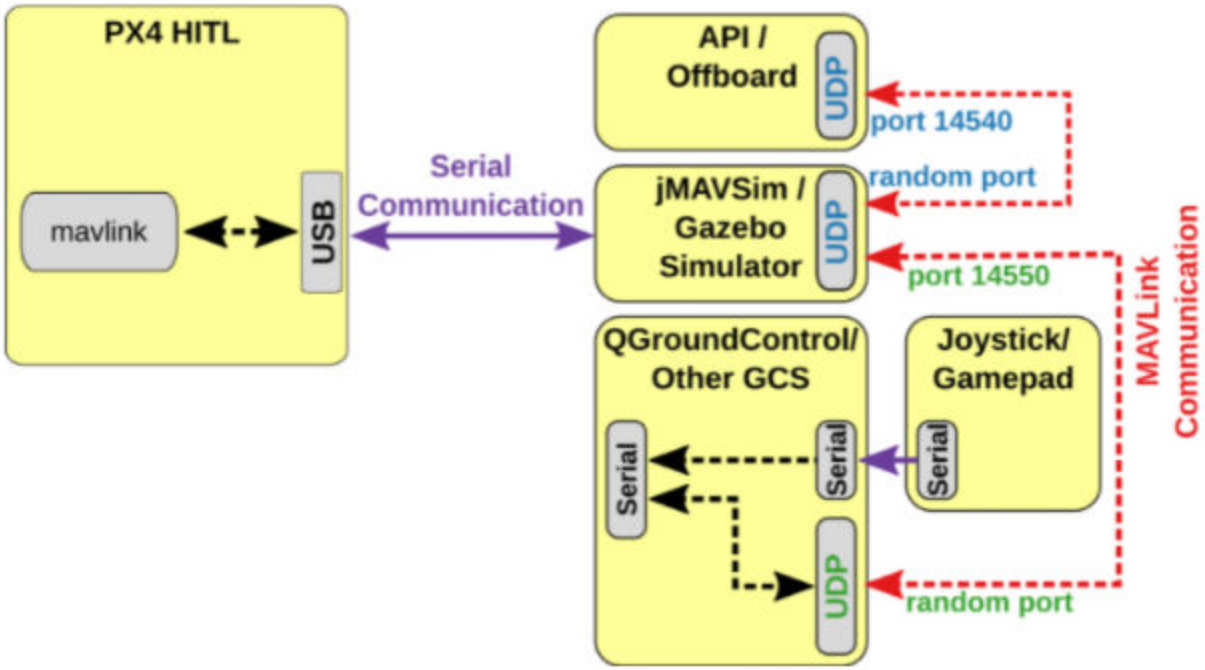


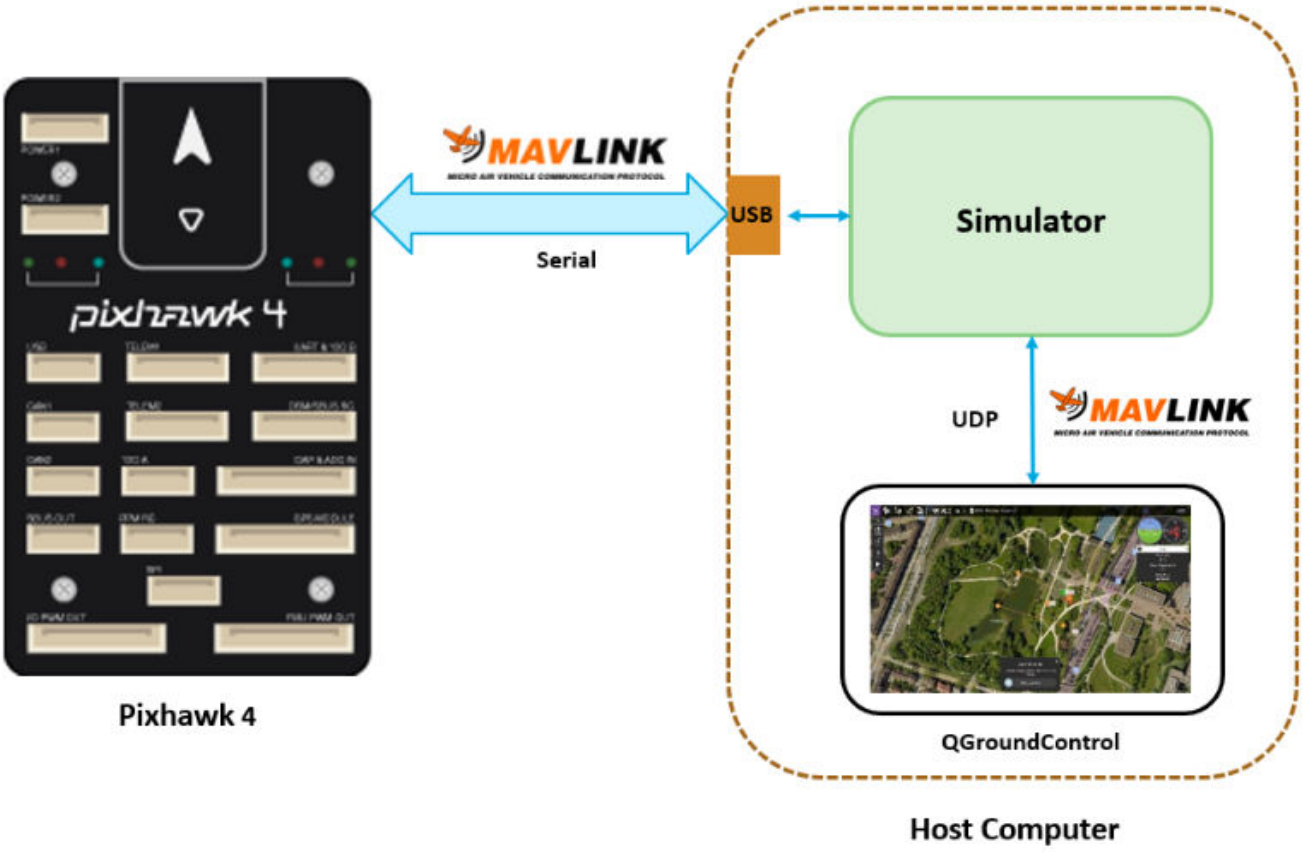
Image courtesy: https://discuss.px4.io/t/how-can-i-change-the-usb-port-to-telem1-when-using-hil/20251

PX4 HITL Physical Communication Diagram

The following diagram shows a simplified version of the above diagram for establishing HITL between Pixhawk 4 and Host Computer.

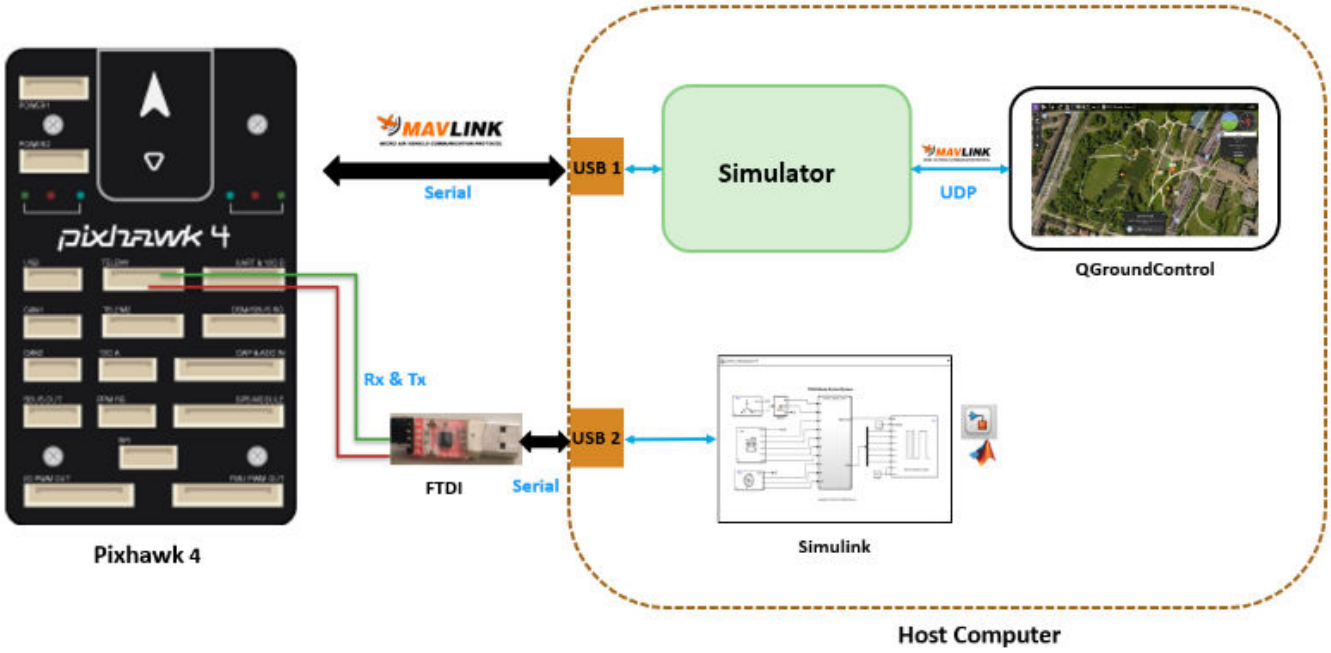
- The Pixhawk 4 is configured in HITL mode.

- The Simulator and QGroundControl is running on Host Computer.
- The Pixhawk 4 communicates to the Simulator via USB connection between Pixhawk 4 and Host Computer.
- The Simulator bridges the MAVLink data between Host Computer and Pixhawk 4 via UDP.



PX4 HITL Physical Communication Diagram for Monitor & Tune (External Mode) Simulation in Simulink

The flight controller algorithm can be deployed from Simulink on the PX4 Autopilot using the UAV Toolbox Support Package for PX4 Autopilots. In this scenario, you can run Monitor & Tune Simulation on page 4-34 to tune the algorithm in real time on the autopilot and log the real time signals in Simulink. In this case an additional serial port is required to establish Monitor & Tune communication with the Autopilot. Since the USB port is already occupied for transferring MAVLink data between host computer and Autopilot in HITL mode, you need to use any of the other available serial ports on the Autopilot. For example, in the below diagram, TELEM1 (/dev/ttyS1) is used on Pixhawk 4 to communicate with Simulink running on Host Computer using Monitor & Tune Simulation. The Tx and Rx from the TELEM1 port on Pixhawk 4 can be connected to the Rx and Tx of the FTDI (Serial to USB Converter) device which is connected to host computer over another USB port. The pinout for the TELEM1 port of Pixhawk 4 can be found here.



Consider the following points while configuring a serial port on Pixhawk Autopilot for Monitor & Tune.

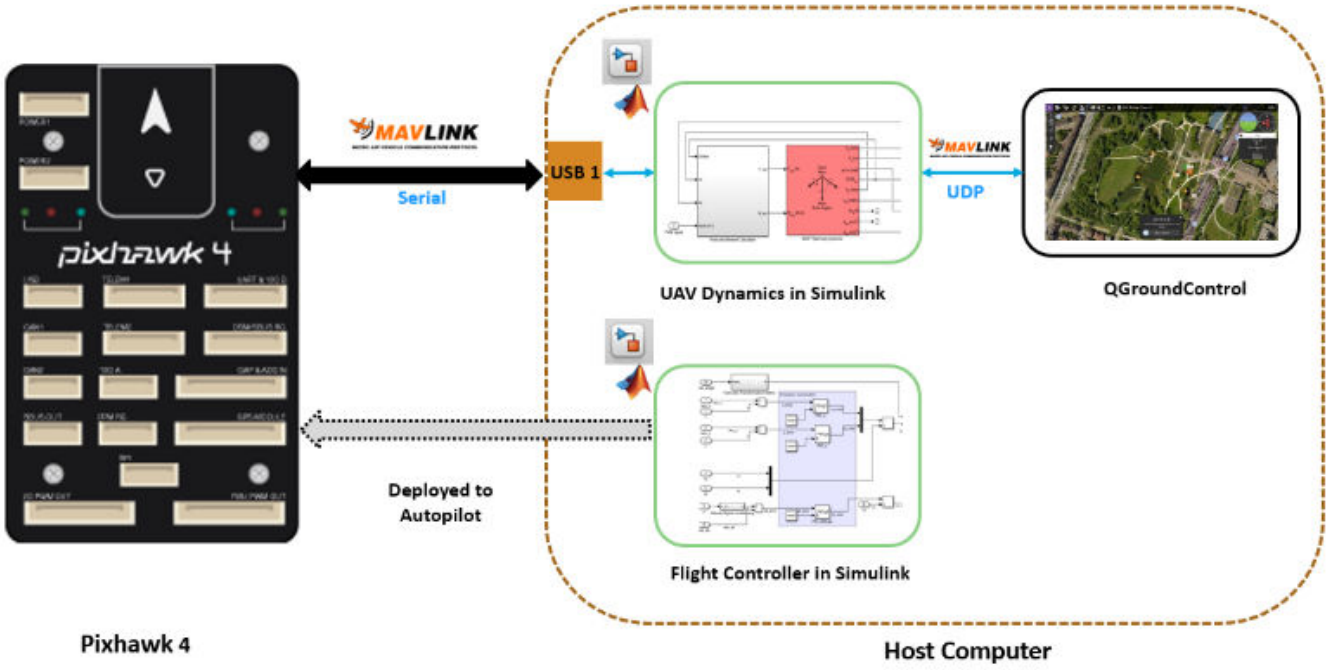
Note The following points are documented for Pixhawk 4 and TELEM1 port, however the points are also valid for any other Autopilot and any other serial port on the Autopilot.

- Ensure that the MAVLink is not enabled on the port where you want to enable Monitor & Tune. The MAVLink can be disabled on such port by removing the port from `MAV_0_CONFIG` or `MAV_1_CONFIG` or `MAV_2_CONFIG` from QGroundControl.
- Use a Serial to USB Converter (FTDI) to communicate with the host. Ensure that the **Tx** and **Rx** of TELEM1 are connected to **Rx** and **Tx** of FTDI (Tx of TELEM1 -> Rx of FTDI and vice-versa). Follow the pinout diagram for Pixhawk 4 found here. For other Autopilots, search for correct pinout and do the connection accordingly.
- For some Autopilots, the maximum baud rate that you can set for communication is `115200`. It is recommended to set `921600` for advanced Autopilots or `115200` if you see connection or data drop issues in Monitor & Tune.

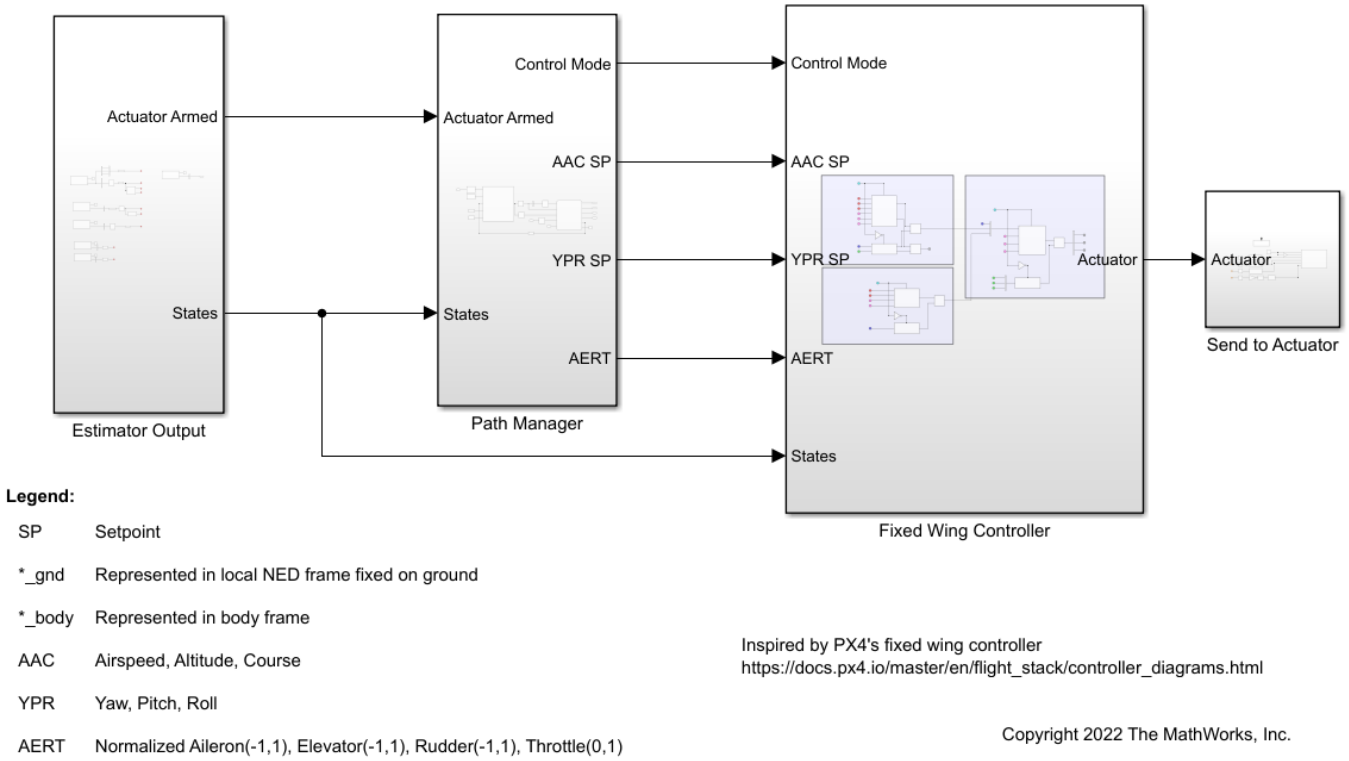
Fixed-Wing Plant and Controller Architecture

This page helps you to understand the high level architecture of reference plant and controller models of fixed-wing UAV simulated in HITL. To get started with the example, see “PX4 Hardware-in-the-Loop (HITL) Simulation with Fixed-Wing Plant in Simulink”.

The diagram below illustrates the workflow of HITL with fixed-wing plant in Simulink.



Guidance Navigation and Controller Model



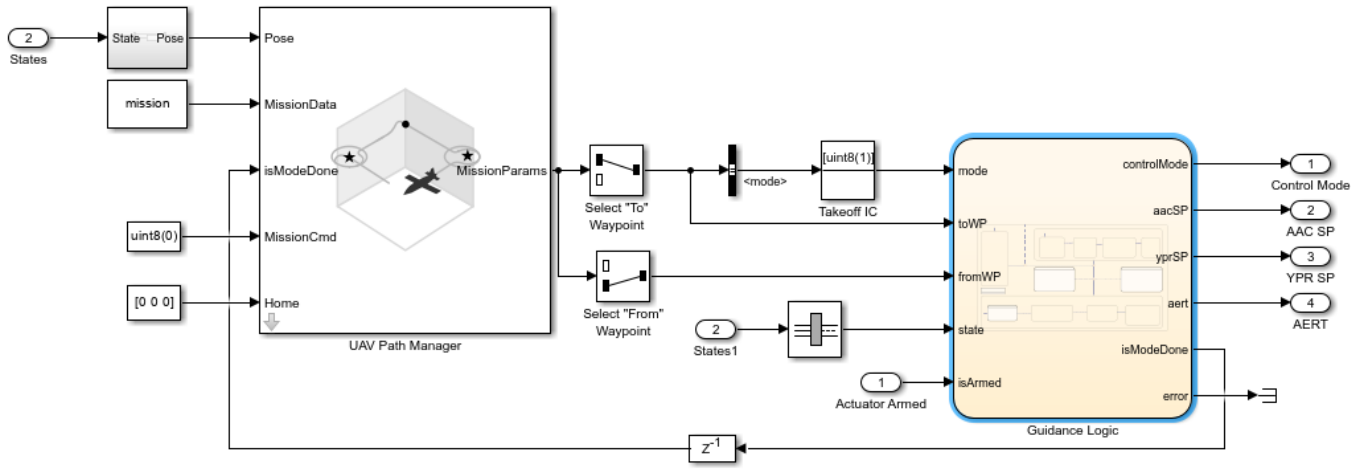
State estimates are obtained through uORB topics. States are populated into a non-virtual Simulink Bus and sent to the Path Manager.

Path manager takes the hardcoded waypoints from a workspace variable and computes the high-level commands to the controller. Depending on the flight mode (take-off, loiter, approach, land, waypoint following), certain control loops are enabled or disabled by the Path Manager. The control modes and high-level set-points are sent via non-virtual bus to Fixed Wing Controller subsystem.

Fixed-Wing controller computes these actuator commands - Throttle, Aileron, Elevator, Rudder.

The Send to Actuator scales actuator commands into pulse widths and sends it to specified channels.

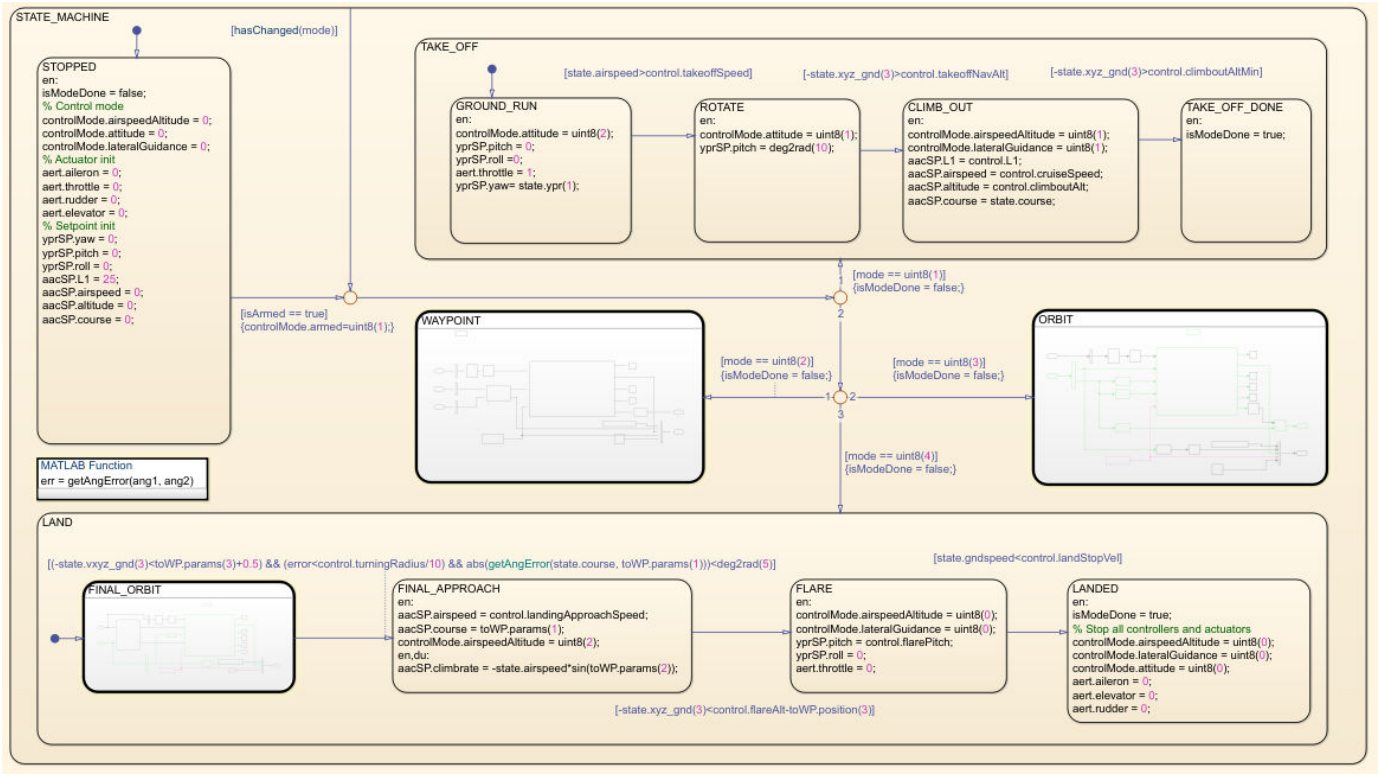
Path Manager



Mission

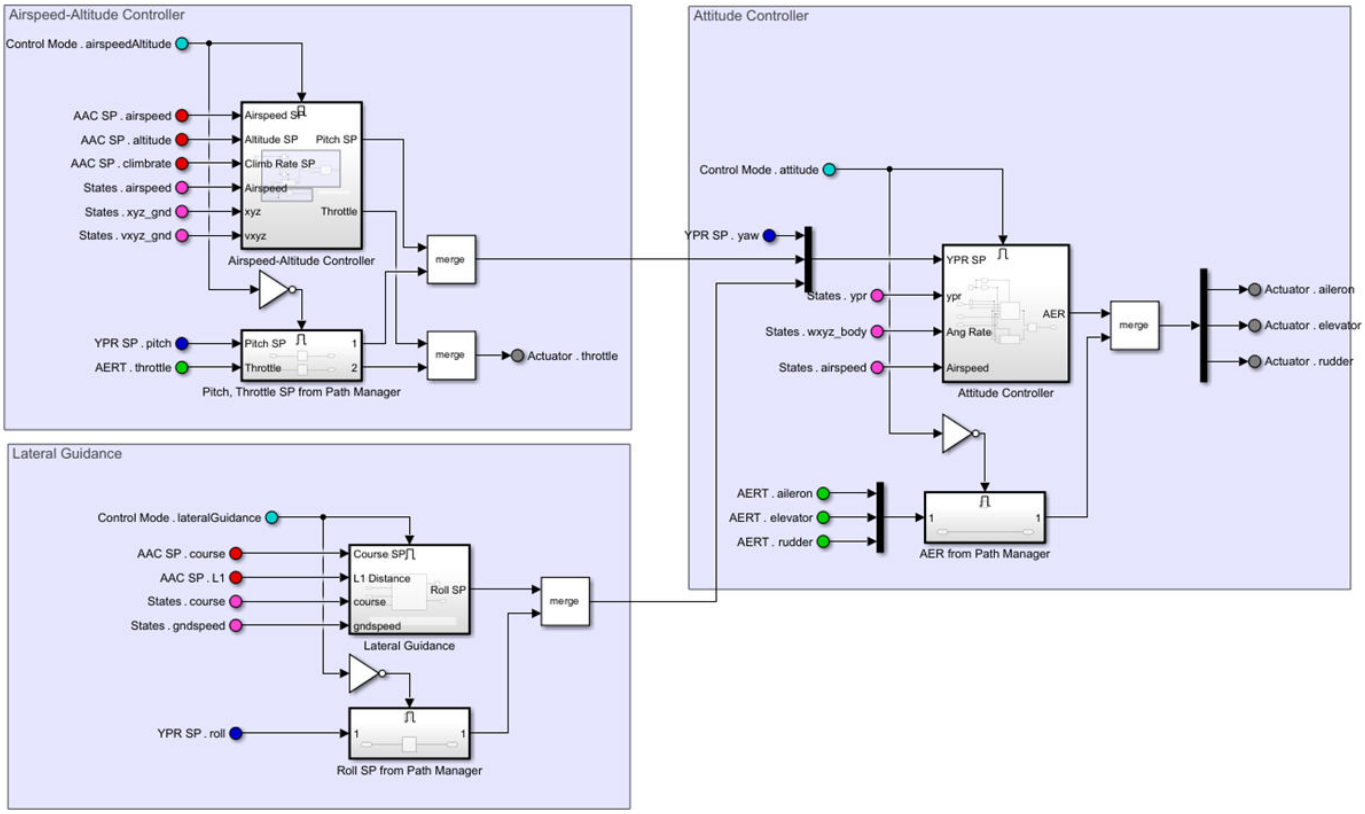
Mission is a MATLAB Struct array that captures flight mode, position, and parameters such as runway orientation, turning radius, flight speed. This is set in `InitWaypoints.m` file. For more information, see UAV Path Manager.

Guidance Logic



Mission is fed to the UAV Path Manager block which outputs the active waypoint and previous waypoint. When a waypoint mission item is completed, the block switches to the next waypoint. Mission items have particular characteristics and they all cannot be handled in the same way. For example, during take-off you must not try to maneuver towards waypoints as the plane is low on airspeed and altitude. Guidance logic for each mission item is modelled with a state machine. Guidance Logic chart takes in the mode and waypoint information from UAV Path Manager and outputs high level commands to controller.

Fixed-Wing Controller

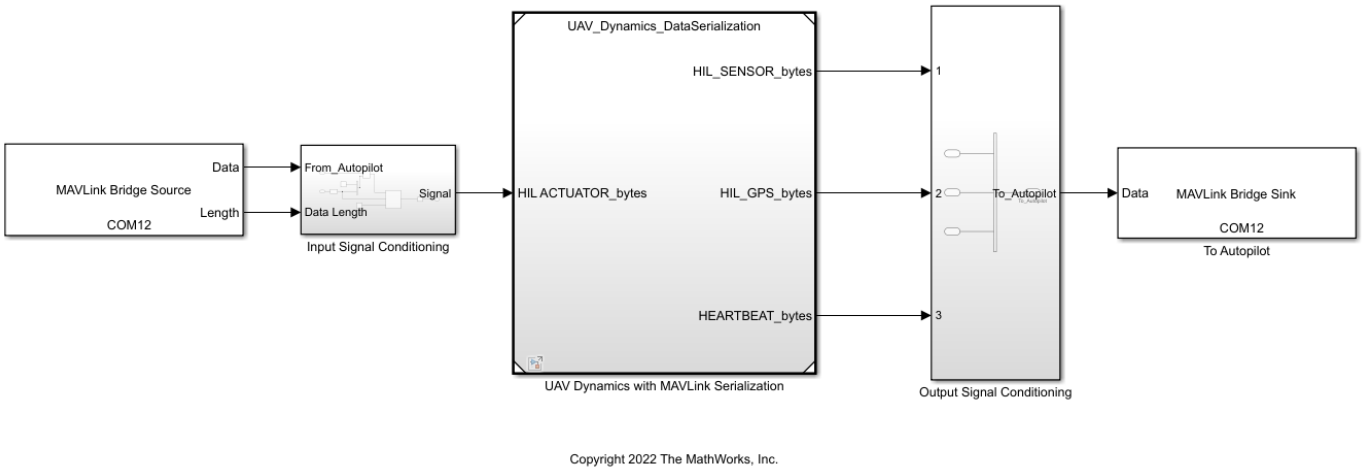


A cascaded control structure is formulated for controlling the following:

- Airspeed
- Altitude & climb rate
- Course
- Attitude & angular rate

Control loops are modelled as Enabled Subsystems and are enabled or disabled depending on flight phase.

Plant Model



The plant model performs the following functions.

- Communication via MAVLink Bridge block
  - Reading MAVLink packets from Pixhawk board via serial port.
  - Relaying MAVLink packets to QGC from Simulink.
- Encoding and decoding.
  - HIL actuator commands are decoded using MAVLink Deserializer block
  - HIL sensors and heartbeat data is encoded using MAVLink Serializer block.
- Plant simulation
  - Models forces and moments from aerodynamics, propulsion, ground contact, gravity.
  - Models IMU, GPS, barometer, and airspeed sensors.

References

- Px4 Fixed-Wing Landing. Retrieved June 20, 2022, Landing Algorithm.
- PX4 Controller Diagrams. Retrieved June 20, 2022, Controller Diagram.
- S. Park, J. Deyst, and J. P. How, "A New Nonlinear Guidance Logic for Trajectory Tracking," Proceedings of the AIAA Guidance, Navigation and Control Conference, Aug 2004. AIAA-2004-4900.
- M. Lizarraga. Design, Implementation and Flight Verification of a Versatile and Rapidly Reconfigurable UAV GNC Research Platform. PhD Dissertation, UC, Santa Cruz, Santa Cruz, CA, USA, 2009.

PX4 Hardware-in-the-Loop (HITL) Simulation with Fixed-Wing Plant and Hardcoded Mission in Simulink

This topic shows how to use the UAV Toolbox Support Package for PX4 Autopilots to verify the controller design by deploying the design on the Pixhawk hardware board. This is done in HITL mode with fixed-wing UAV Dynamics contained in Simulink. The mission is hardcoded in MATLAB and is executed by the GNC model once armed.

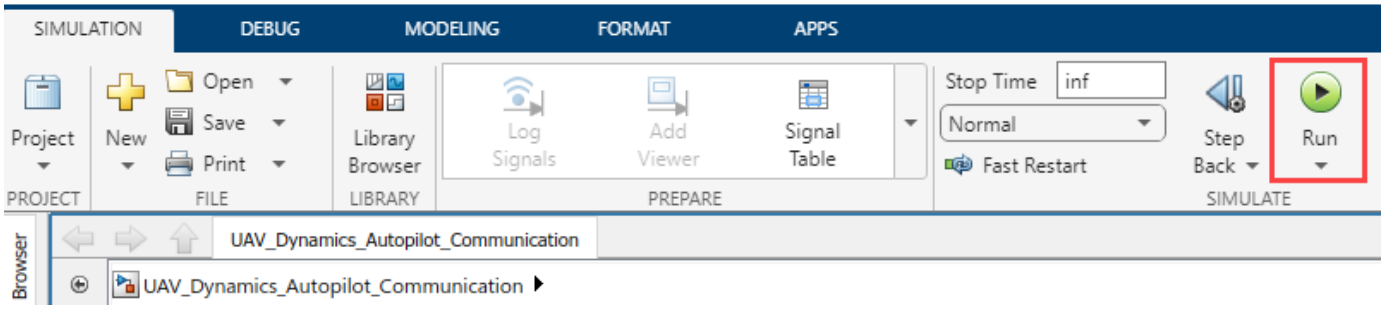
Before starting with Simulink, ensure that you go through “PX4 Hardware-in-the-Loop (HITL) Simulation with Fixed-Wing Plant in Simulink” example and perform Step 1, Step 2, and Step 3.

Run the UAV Dynamics Model, Upload Mission from QGroundControl and Fly UAV*

1. To use the mission hardcoded in MATLAB, run the following command in MATLAB command prompt.

   ```
   isMissionFromQGC = 0;
   ```

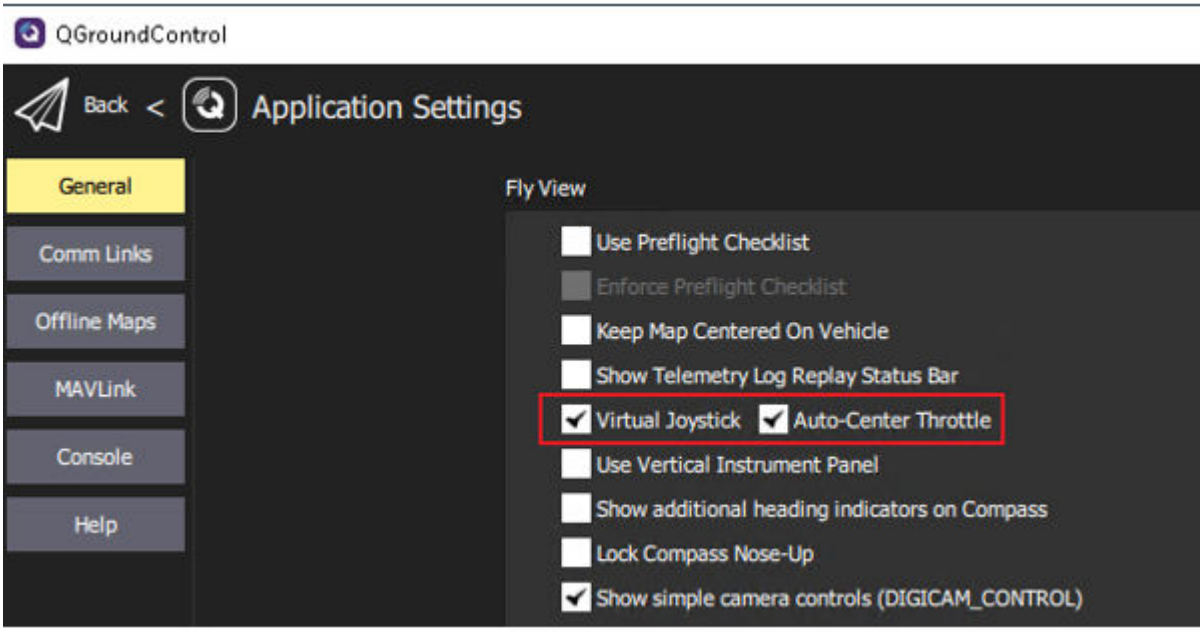
2. In the Simulink toolstrip of the Plant model (*UAV_Dynamics_Autopilot_Communication*), on the **Simulation** tab, click **Run** to simulate the model.



3. Enable the Joystick in QGC.

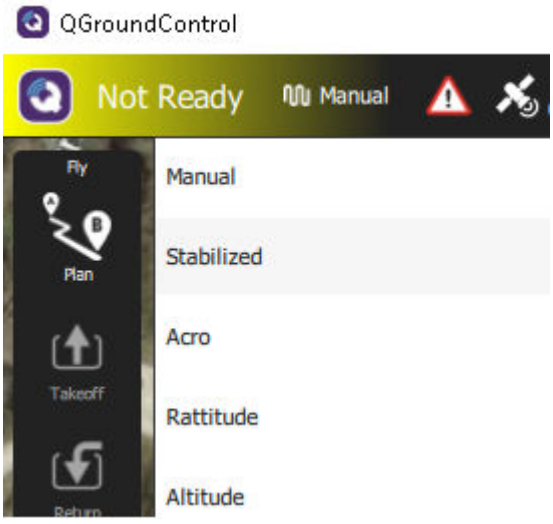
   Waypoints are hardcoded and stored in a *InitWaypoints.m* data file. For more information on the structure, see Mission Data UAV Path Manager. As the waypoints are hardcoded, do not add a

mission in QGC. Enable Virtual Joystick in QGC Application Settings as shown below.



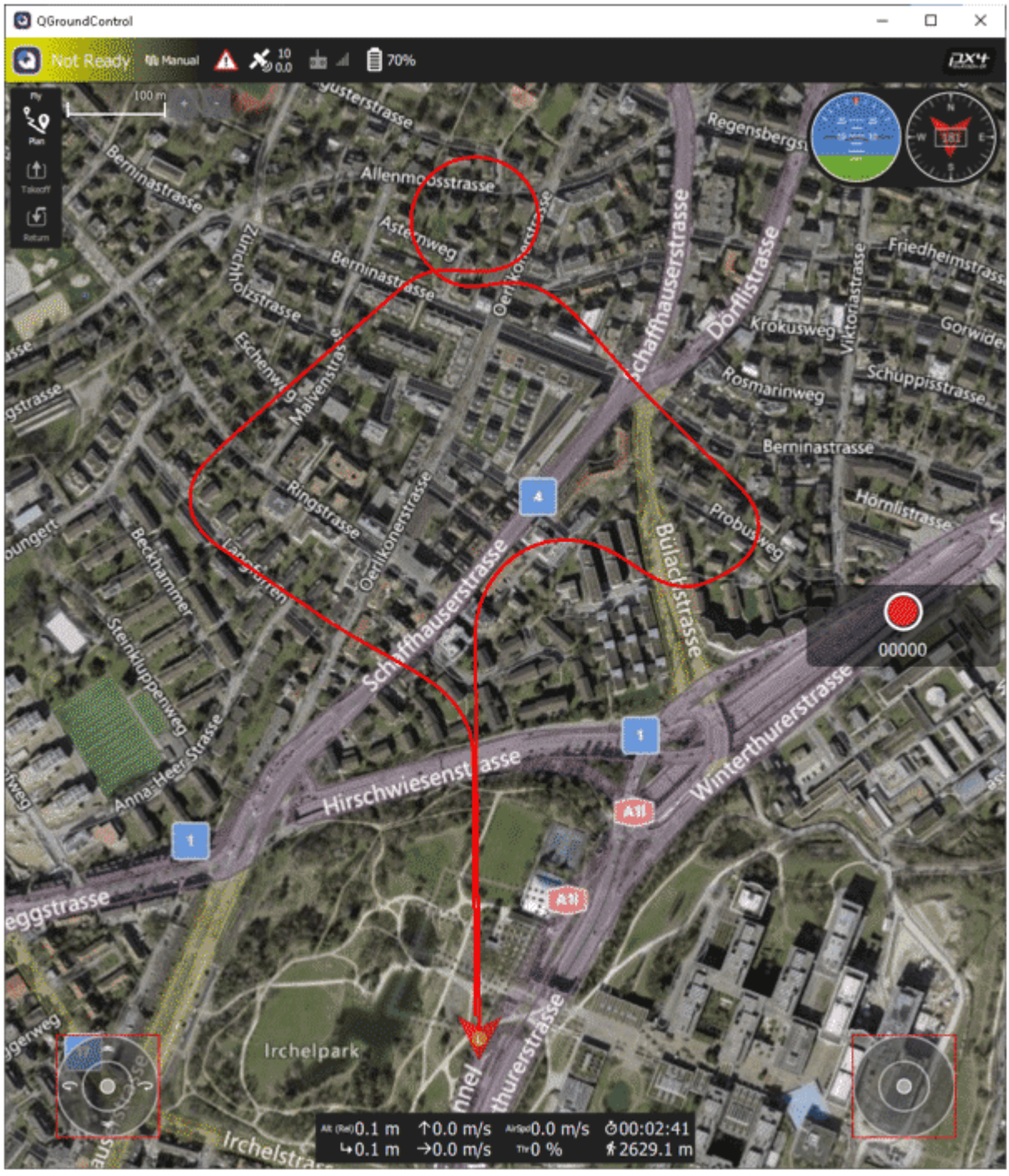
4 Change the flight mode to Manual in the QGC.

Note The mission is hardcoded in MATLAB and is executed by the GNC model once armed. It is not possible to arm the UAV in the mission mode.



5 Drag the left joystick to the bottom-right corner for a couple of seconds to arm the UAV. The flight starts to execute.

An image of the expected flight trajectory is shown here.



See Also

Related Examples

- “PX4 Hardware-in-the-Loop (HITL) Simulation with VTOL UAV Tilt-Rotor Plant in Simulink”

- “Visualize PX4 Hardware-in-the-Loop (HITL) Simulation with VTOL UAV Over Urban Environment”

PX4 Hardware-in-the-Loop (HITL) Simulation with Manual Control for Fixed-Wing Plant in Simulink

This example shows how to use the UAV Toolbox Support Package for PX4 Autopilots and take manual control inputs from RC Transmitter and control the fixed-wing flight.

Before starting with Simulink, ensure that you go through “PX4 Hardware-in-the-Loop (HITL) Simulation with Fixed-Wing Plant in Simulink” example and perform Step 1.

Launch MATLAB

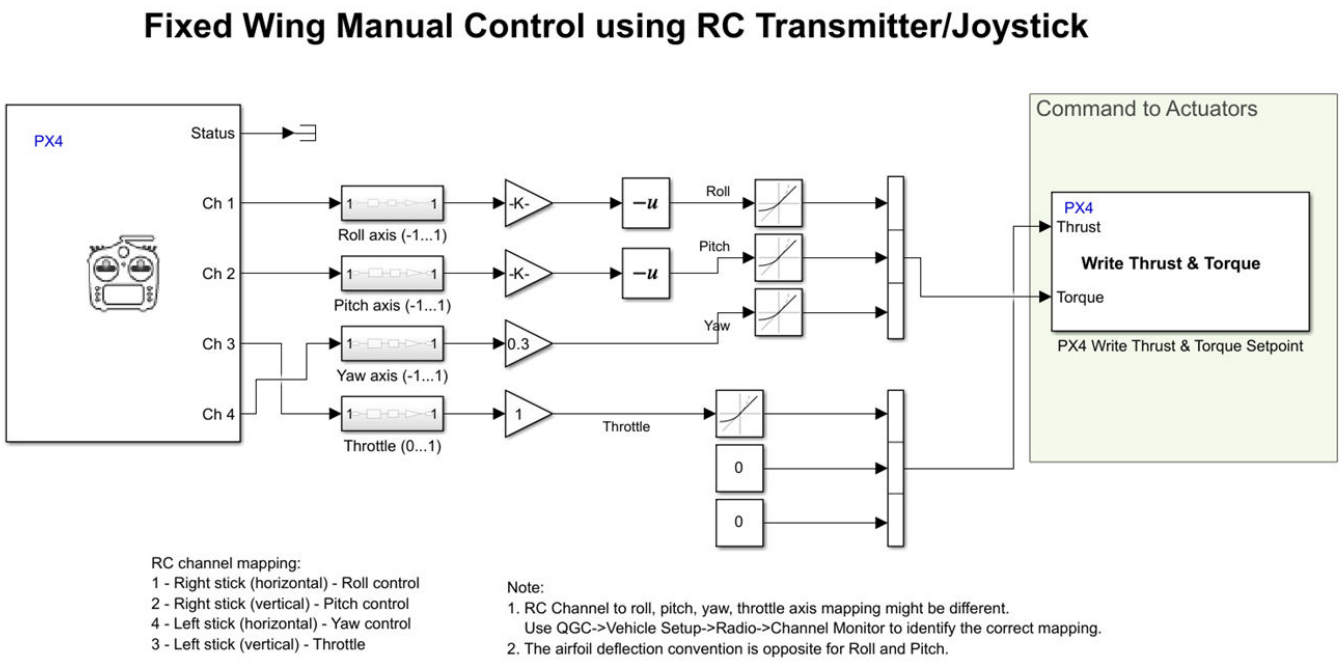
1. Open MATLAB.
2. Open the example MATLAB project by executing this command at the MATLAB command prompt:

   ```
   openExample('px4/PX4HITLSimulationFixedWingPlantSimulinkExample')
   ```

3. Once you open the project, click the **Project Shortcuts** tab on the MATLAB toolstrip and click **Launch Manual Control Mode** to launch the *ManualControl* controller.

   Alternatively, execute this command at the MATLAB command prompt.

   ```
   open_system('ManualControl')
   ```



4. In the **Project Shortcuts** tab, click **Launch UAV Dynamics** to launch the Simulink plant model named *UAV_Dynamics_Autopilot_Communication*.

   Alternatively, execute this command at the MATLAB command prompt.

   ```
   open_system('UAV_Dynamics_Autopilot_Communication')
   ```

5. In the Simulink plant model, set the serial ports in MAVLink Bridge Source and MAVLink Bridge Sink blocks.

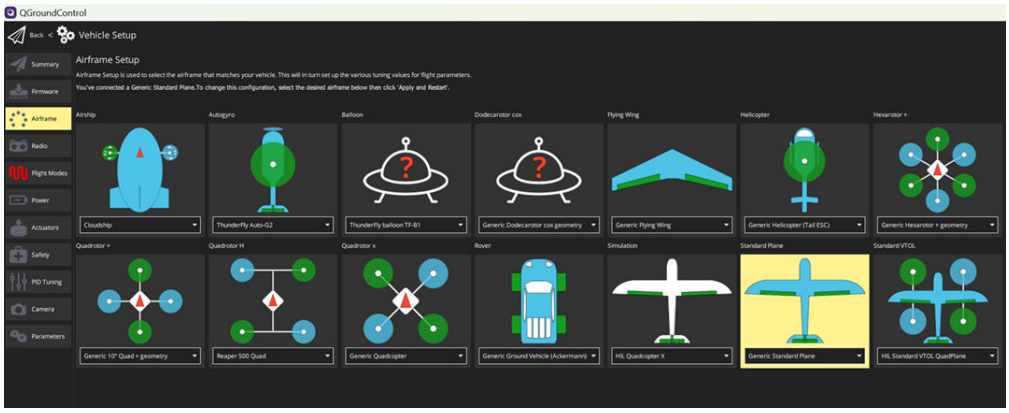
6 Click **Build, Deploy & Start** from **Deploy** section of **Hardware** tab in the Simulink Toolstrip of the controller model *ManualControl*.

Run the Simulink Plant Model, Configure QGroundControl and Fly UAV

1 Run the Simulink plant model (*UAV_Dynamics_Autopilot_Communication*).

2 Open QGC and connect to PX4.

3 In the QGC, navigate to **Vehicle Setup** > **Airframe**.



a Select the **Generic Standard Plane**.

b Click **Apply and Restart** at the top right corner of the window.

c Reboot the Vehicle for the changes to take effect. This stops the Simulink plant model.

d Run the Simulink plant model again.

4 In the QGC, navigate to **Vehicle Setup** > **Safety** and perform these steps.

a In the **Hardware in the Loop Simulation** section, select `HITL enabled` option.

c Close the QGC and Reboot the vehicle manually.

This will stop the Simulink plant model

d Run the Simulink plant model again and relaunch the QGC.

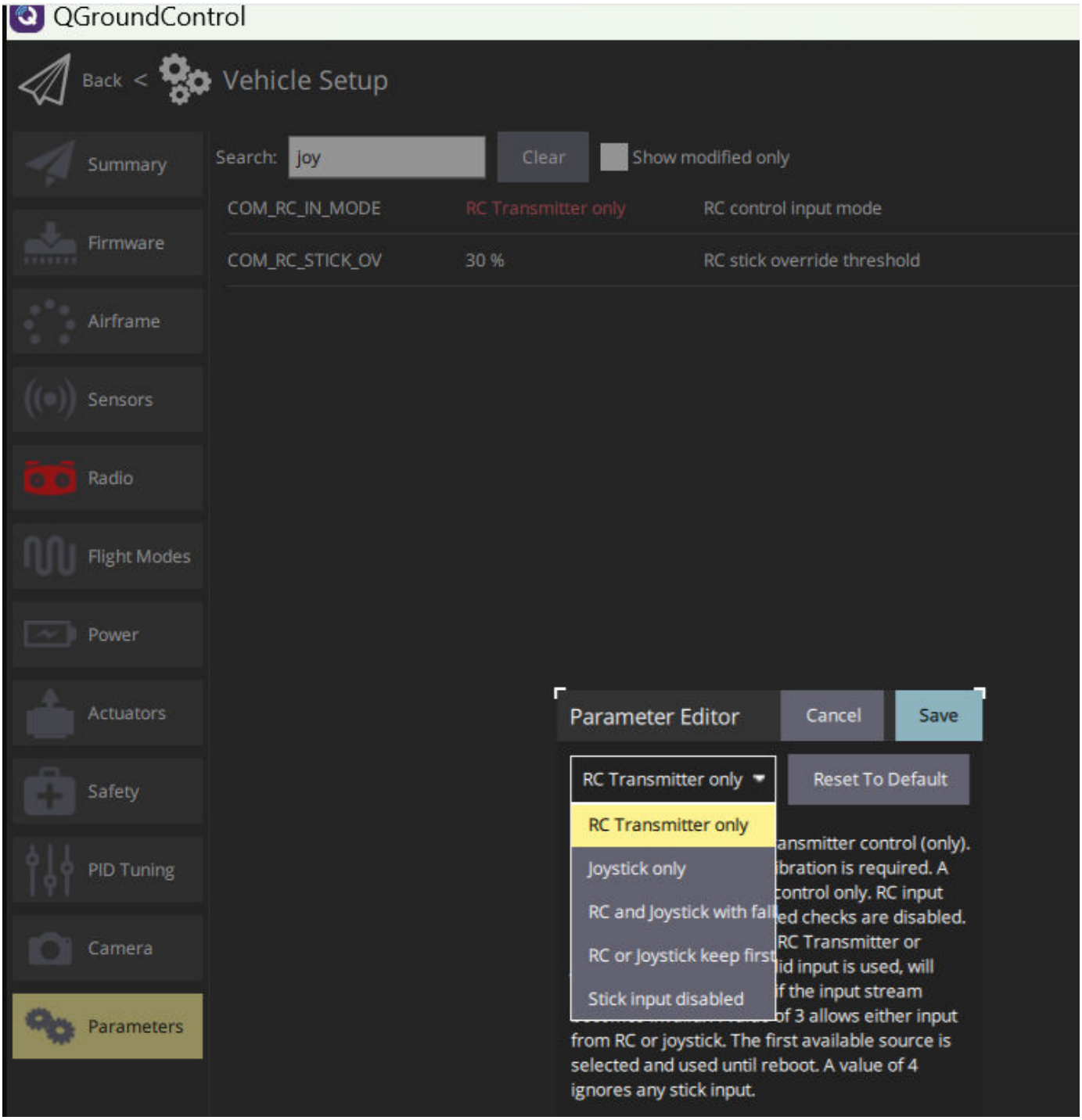
5 In the QGC, navigate to **Vehicle Setup** > **Actuators** and choose the servos and motors for the channels as shown in this image.

Note The sequence should match the sequence used in the Simulink plant model. You can see the sequence under `UAV_Dynamics_Autopilot_Communication/UAV Dynamics with MAVLink Serialization UAV_Dynamics_DataSerialization/UAV Dynamics (UAV_Dynamics)/Dynamics model/Force and Moment Calculation/Actuator Model` subsystem.

6 In the QGC, navigate to **Vehicle Setup** > **Parameters**.

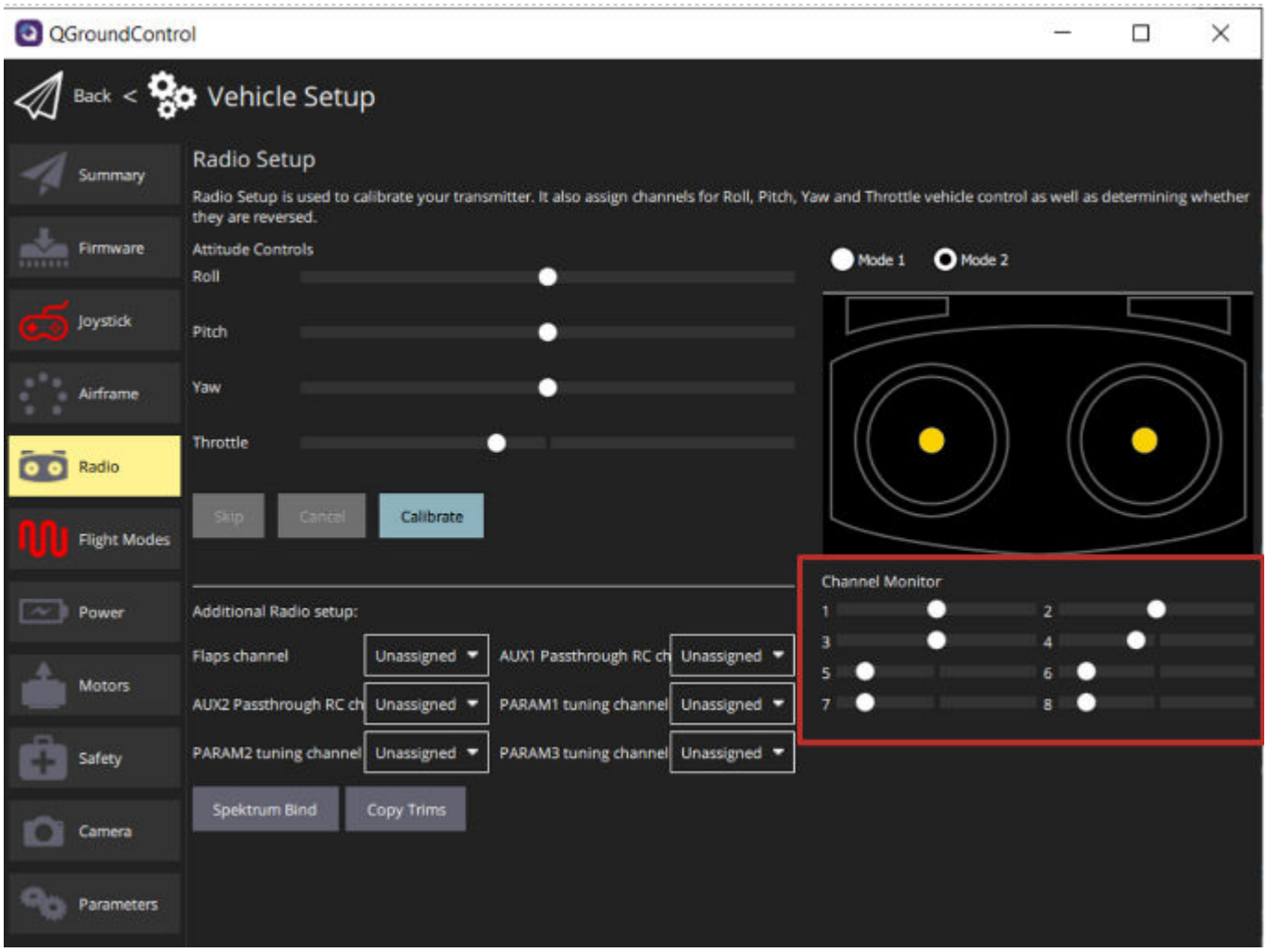
   a Set the input device in `COM_RC_IN_MODE` parameter.

      - Select `RC Transmitter only` option. For information on connection and calibration, see Radio (Remote Control) Setup.



b Set the parameters `COM_DISARM_LAND` and `COM_DISARM_PRFLT` to `-1`.

7 Click **Radio** and move each sticks that control aileron, elevator, rudder, and throttle. Observe the channel value changes in the **Channel Monitor**.



If the mapping is incorrect in the example model or between the channel number and controlled axes, correct it.

8 Arm the vehicle from QGC and move the left stick towards right-bottom to arm the vehicle. Provide only the throttle for the drone to takeoff.

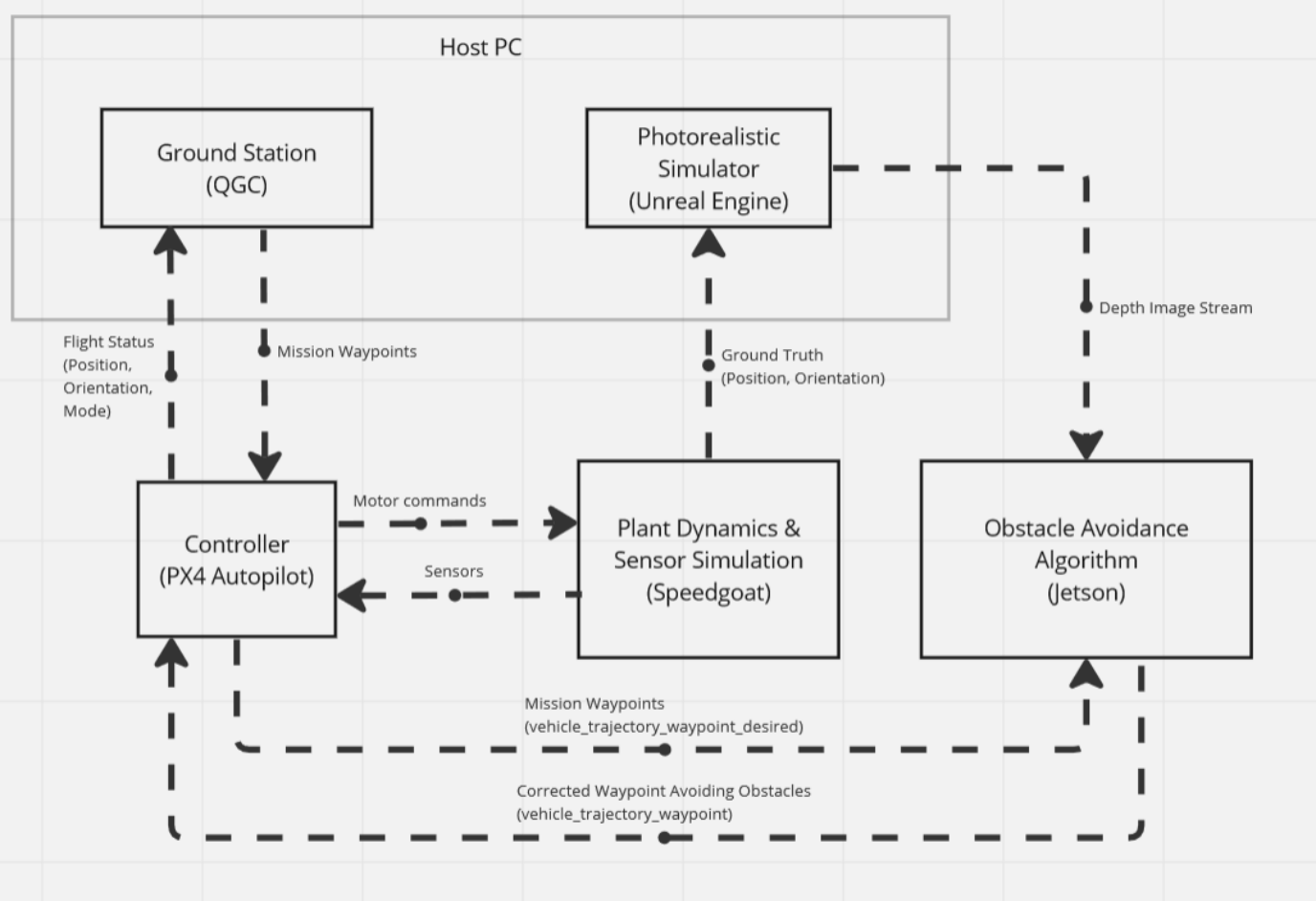
Limitations

The moments along roll, pitch, and yaw axis exerted by the ground is not modelled. High control inputs can make the simulated plant turn to improbable angles.

Speedgoat and Simulink Real-Time Setup

This topic helps you with setting up Speedgoat and Simulink Real-Time for “PX4 Autopilot in Hardware-in-the-Loop (HITL) Simulation with Speedgoat in Simulink” example. Hardware-in-the-loop simulation allows you to validate your controller design code running on an embedded system in normal and adverse real-world conditions.

The following image shows an overview of hardware connections and information flow between PX4 Autopilot, Speedgoat, and QGC.



In the “PX4 Autopilot in Hardware-in-the-Loop (HITL) Simulation with Speedgoat in Simulink” example, a quadcopter plant model, along with its sensors, are modeled in Simulink. The generated code is deployed to Speedgoat Real-Time Target machines to perform HITL simulation. The example demonstrates the following workflows.

- Real-Time simulation of plant and sensors in Speedgoat.
- Controller deployment to PX4 Autopilot.
- Photorealistic simulation using unreal engine, based on plant states obtained from Speedgoat.
- Running an obstacle avoidance algorithm on Jetson™ using the depth camera data simulated in Unreal Engine.

The plant model communicates with PX4 Autopilot to share the sensor and actuator data. The plant model additionally shares the UAV ground truth (position and orientation) to host PC running photorealistic simulation.

The photorealistic simulator provides a 3D visualization of the UAV's movement along with simulating its depth camera. Depth data is sent over to a Jetson onboard computer running an obstacle avoidance algorithm, which corrects the path accounting for any obstacles and sends back corrected path over to PX4 Autopilot.

Required Hardware

To run this example, you will need the following hardware:

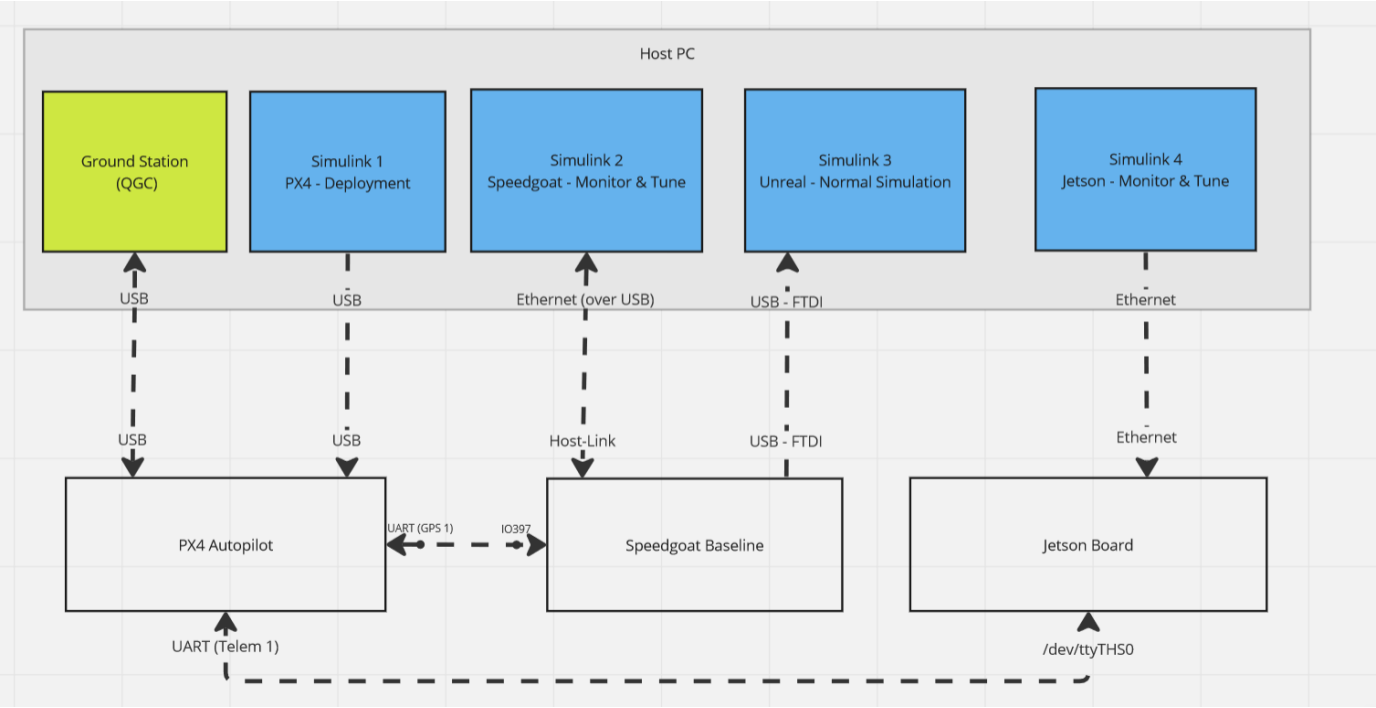
- Speedgoat Baseline with IO397 module
- IO397 Terminal board
- A supported PX4 Autopilot on page 4-2 board with a class 10 microSD card.
- Micro USB (Universal Serial Bus) type-B cable
- USB to Ethernet converter
- Pixhawk UART to jumper male converter
- 2 x FTDI units

To run onboard obstacle avoidance, you need the following,

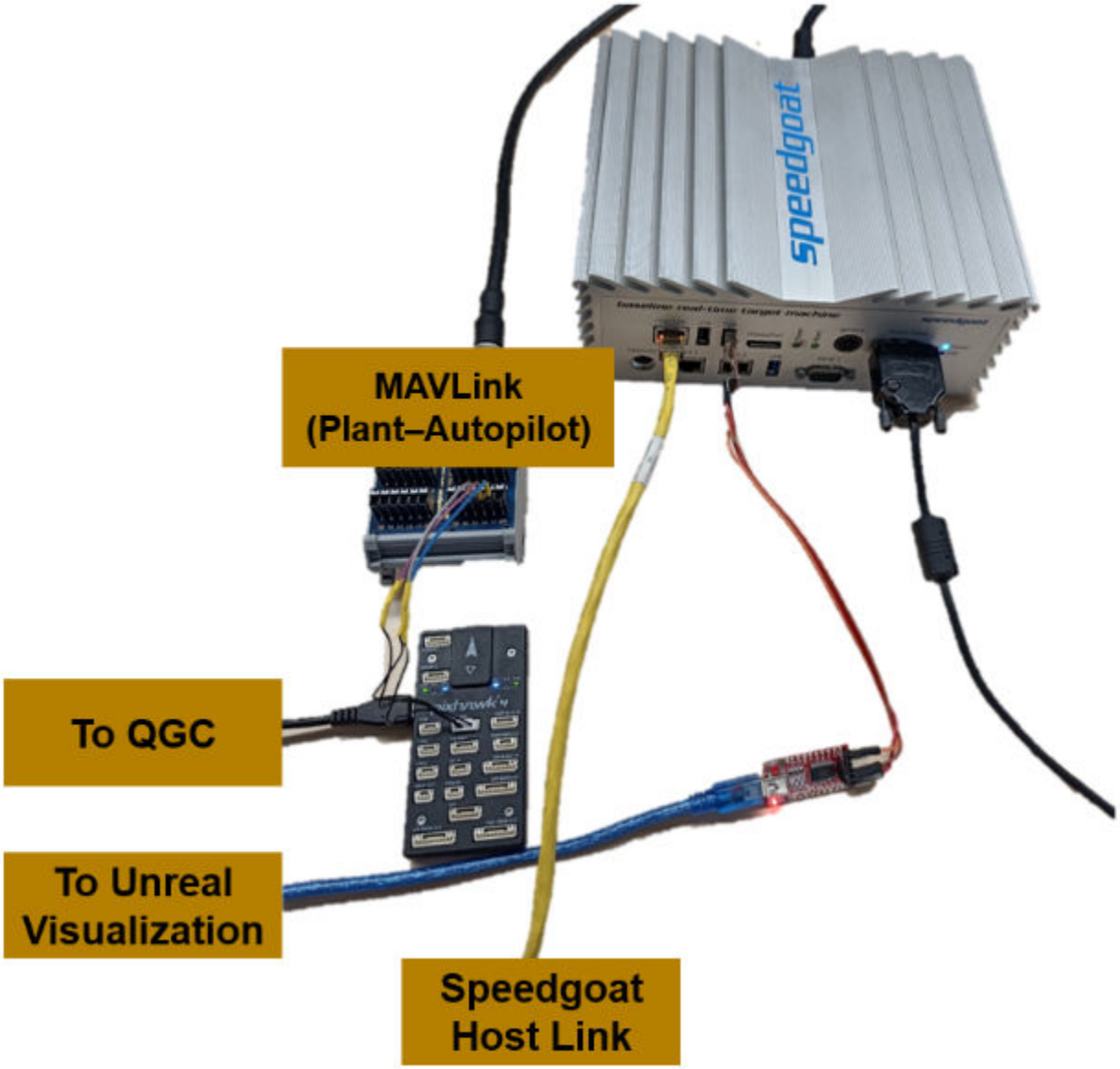
- Jetson Xavier / Nano connected to autopilot over UART

Make Connections for Speedgoat

Connect the hardware as shown in this diagram. You can skip the Jetson connection if you are not running onboard obstacle avoidance.



A sample image of Speedgoat connected to PX4 Autopilot, QGC, host computer is shown below.



Connect the following ports between Pixhawk and Speedgoat's IO397 Terminal board.

Pixhawk Speedgoat connections

Pixhawk (GPS)	IO397 Terminal board
Tx	16b
Rx	15b
GND	17b

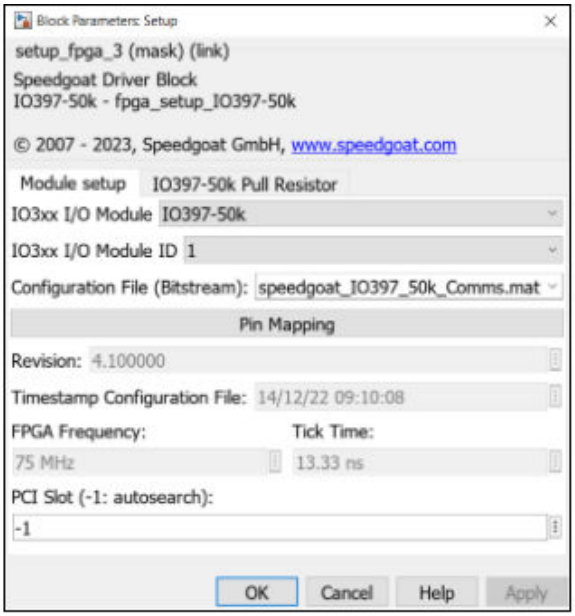
Pixhawk GPS 1 port is connected to IO397 Terminal board.

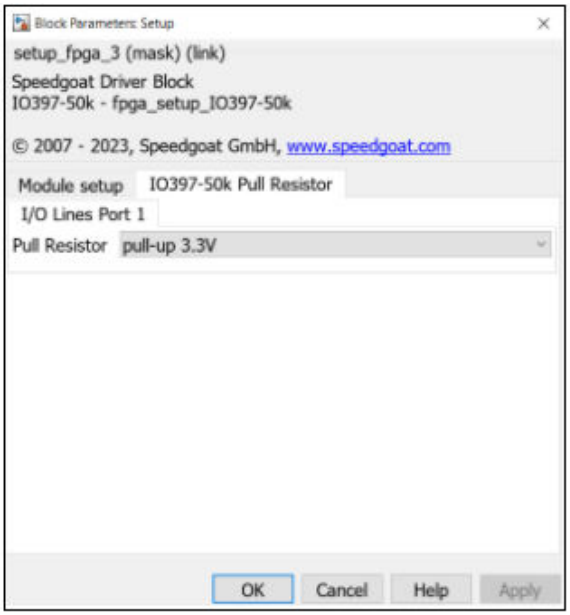
To establish a connection with unreal visualization model, perform these steps.

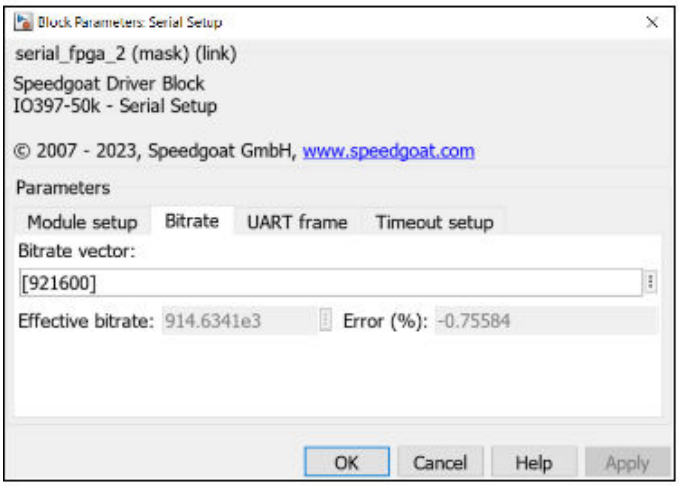
1. connect a FTDI port to the right USB port.
2. Connect the TX of FTDI to RX of another FTDI and vice-versa. Also connect gnd.
3. Connect the second FTDI unit to host PC that runs unreal visualization model.

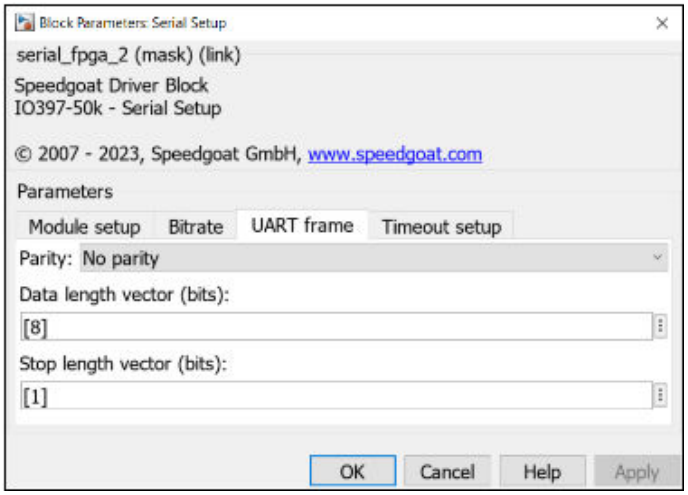
Speedgoat Configuration

To configure the Serial ports and IO397 module, refer to the following image. If you have a different Speedgoat configuration such as a different IO module, then use this information to configure Speedgoat. Serial ports and IO397 modules are available in Speedgoat Setup section of Simulink plant model (*UAV_Dynamics_Speedgoat*). For more information on opening the Simulink model, see "PX4 Autopilot in Hardware-in-the-Loop (HITL) Simulation with Speedgoat in Simulink".









Software Setup

Along with the required products mentioned in the example, you must also download and install the following software.

- Simulink IO Blockset
- IO397 Configuration package (The example uses v4.1)
- QGroundControl (QGC)
- Simulink Real-Time Target Support Package.

Create an account in Speedgoat to download and install the Simulink IO Blockset and IO397 configuration package.

For more information, see Speedgoat Software.

Connections for Speedgoat

1. Connect the Speedgoat machine to host computer over an Ethernet-USB converter.
2. Depending on the operating system installed on the host computer, use the links provided below to configure your network adapter to communicate with Speedgoat.
   - For Windows: Windows
   - For Linux: Linux
3. Open `slrtExplorer` and follow these instructions to configure the IP address and update the firmware of Speedgoat.
4. If you are not yet connected to the target computer, click **Disconnected** to change it to **Connected**.

PX4 Parameter Setup in QGC

Set the parameters values in QGroundControl (QGC), as shown in the table below.

QGC Parameters

Parameter	Value
SER_GPS1_BAUD	921600 8N1
SER_TEL1_BAUD	230400 8N1
MAV_0_CONFIG	TELEM 1
MAV_0_FORWARD	1
MAV_0_MODE	Onboard
MAV_0_RADIO_CTL	1
MAV_0_RATE	1200 B/s
MAV_1_CONFIG	GPS1
MAV_1_FORWARD	0
MAV_1_MODE	Config
MAV_1_RATE	0 B/s

See Also

“PX4 Autopilot in Hardware-in-the-Loop (HITL) Simulation with Speedgoat in Simulink”