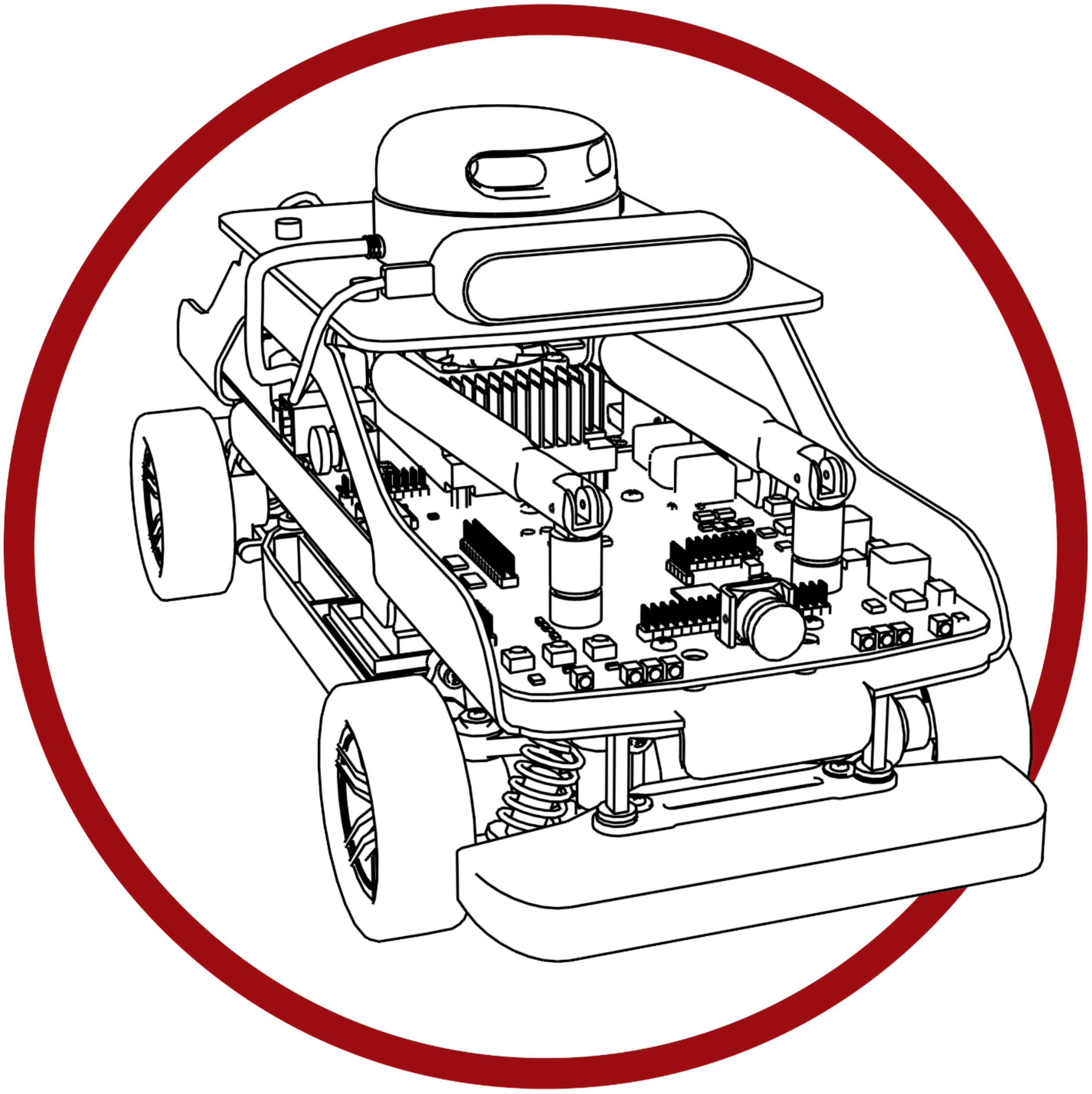
QUANSER
INNOVATE · EDUCATE

Self-Driving Car Research Studio



User Guide – Software – Simulink

V 1.1 (November 2020)

Caution **This equipment is designed to be used for educational and research purposes and is not intended for use by the general public.** The user is responsible to ensure that the equipment will be used by technically qualified personnel only.

Table of Contents

A. Overview

The overall process is described in Figure 1. Simulink is used as an open canvas to design your applications as you see fit. To deploy the code, you first build a run-time application (C code) from the Simulink diagram, and then deploy it to your QCar using QUARC. Simulink then connects to the application and displays any output connected to Simulink sinks such as scope or display blocks to the screen of the host machine that is running Simulink.

For more details, see the following sections.

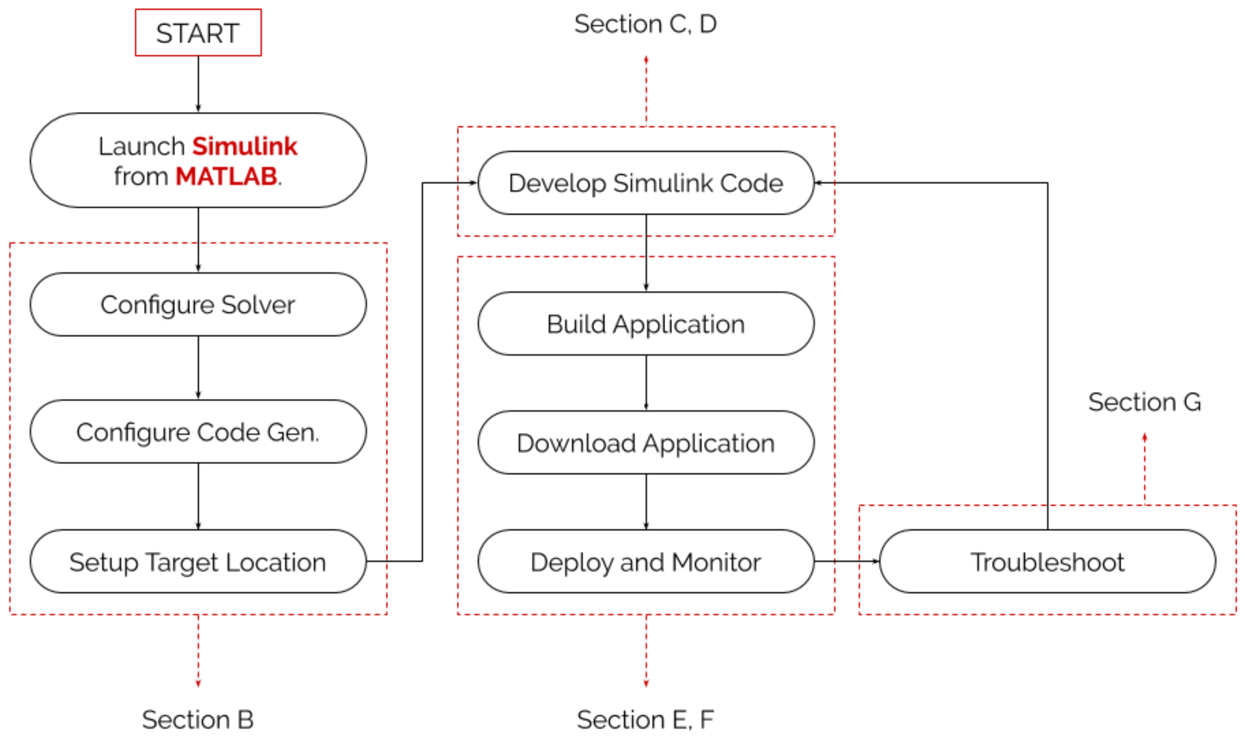


Figure 1. Process diagram for Simulink code deployment

B. Model Configuration Settings

1. Once you open **Simulink**, open the **Model Configuration Settings** using the icon ⚙ or pressing `Ctrl+E`.

2. Under **Solver > Solver selection**, set the **Type** to **Fixed-step** and set the **Solver** to the desired one. If you are unsure, select `ode1 (Euler)`.

3. Under **Solver > Solver details**, set the **Fixed-step size (fundamental sample time)** to the desired value. For example, for 500 Hz, set the value to 0.002 s.

4. Under **Code Generation > Target Selection**, set **System target file** as applicable to your deployed application, for example:
   a. For applications running on the development machine, use `quarc_win64.tlc` for a windows-based platform (e.g. the provided Ground Control Station). This may be used for real-time simulations, robust communications proxies to Qcars, or infrastructure servers to coordinate the actions of multiple QCars.
   b. For applications to be deployed to a QCar target, select the `quarc_linux_nvidia.tlc` for the QCar platform. This will be the typical target for most of the Simulink examples.

5. If the target is the QCar, you must also specify the location of the platform on the network so that the application can be downloaded there automatically by QUARC. To do this, navigate to **Code Generation > Interface**, and edit the **MEX-file arguments** by adding the following code:

```
'-w -d /tmp -uri %u', 'tcpip://IP_ADDRESS:PORT'
```

   Do not exclude the comma or the single quotations. Here, `IP_ADDRESS` refers to the IPv4 address of the QCar shown on the car's LCD screen and `PORT` refers to a number between 17001 and 17999. Simulink will use the `PORT` to communicate with the deployed application and display any data connected to Simulink sinks in your code on your local machine itself. For example, if the IP_ADDRESS of the QCar is **192.168.2.12** and we are targeting the model to port **17001**, then the MEX-file arguments string will be:

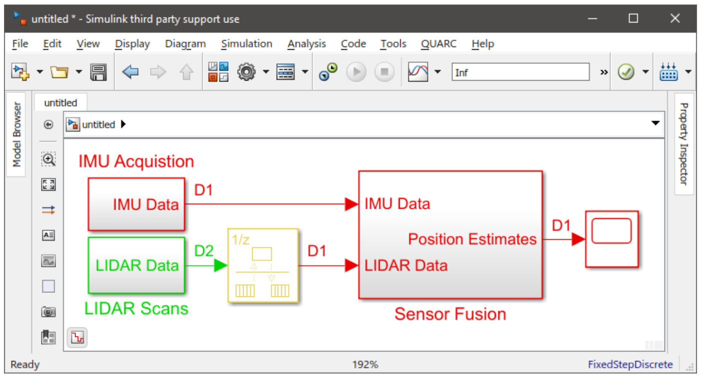
```
'-w -d /tmp -uri %u', 'tcpip://192.168.2.12:17001'
```

   Typically port 17001 will always be used, but if you are running multiple Simulink models on the QCar simultaneously, then you will need to specify a separate port for each simultaneous model you run.

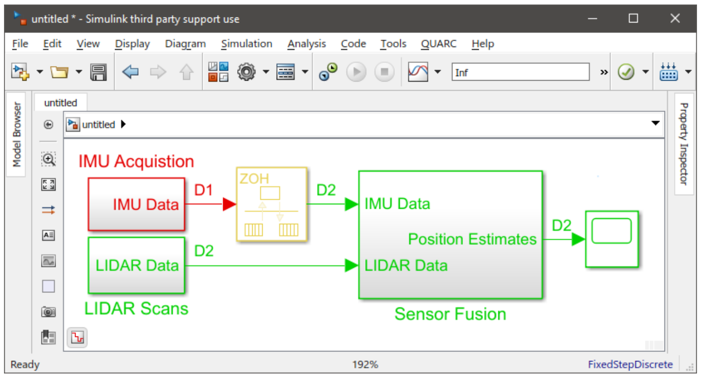
C. Multiple sampling rates

The **Fixed-step size (fundamental sample time)** parameter in the **Model Configuration Settings** corresponds to the fastest sample time in your model, denoted as D1 in Figure 2. This also corresponds to the **continuous** sample time. The current value of this parameter can be accessed in the model by using the constant **`qc_get_step_size`.**

Most Simulink/QUARC blocks have a **Sample Time** parameter. When set to -1, this makes the block automatically acquire its sample time based on other blocks that are connected to it. However, it is good practice to explicitly specify the sample time, which makes it easier to read the diagram for future developers and debugging. Integer multiples of the time step constant **`qc_get_step_size`** can be used to specify other sample rates.



a. Sensor Fusion at 500Hz



b. Sensor Fusion at 10Hz

Figure 2. Using Rate Transition to combine signals at various rates

Tip: Under **Display > Sample Times**, click **All** to turn on rate colors

Consider the example shown in Figure 2 where the **LIDAR Scans** subsystem provides data at 10Hz and the **IMU Acquisition** subsystem provides data at 500Hz. The subsystem labelled **Sensor Fusion** needs to use both the signals to estimate position. To do so without Simulink errors, you must use a **Rate Transition** block on the signal depending on the rate at which you expect sensor fusion to run. For example, if **Sensor Fusion** must be executed at 500Hz, insert a **Rate Transition** on the lidar signal line with sample time set to -1 as in Figure 2a. To run **Sensor Fusion** at 10Hz, insert a **Rate Transition** block on the IMU signal line, with sample time set to -1 as shown in Figure 2b.

D. Core Modules

In addition to the supplied **Hardware Tests** and **Application** examples, the **Self Driving Car Research Studio** comes with **Core** Simulink models, equipped with a list of Simulink subsystems and MATLAB functions commonly used throughout the provided examples. These are broken down into 6 functional groups**:** Essential, Control, Interpretation, User Interface, Decision & Planning, and Miscellaneous, which are summarized in Table 1 in the context of an autonomous driving example illustrated in Figure 3. Use these methods (found under Core) as you see fit.

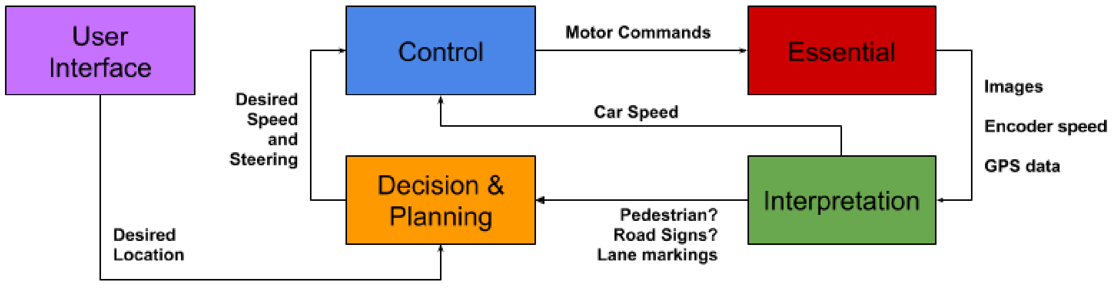


Figure 3: Autonomous driving example

Functional Units	Description
User Interface (UI)	Serves as the entry point for the user - manual switches, sliders, constants etc. that are expected to be tunable online. In Figure 3, this corresponds to the user's desired coordinates.
Essential	This core represents all the I/O (inputs and outputs) in regards to the QCar. Motor commands and LED values written to the device, and measurements such as IMU, motor current, battery voltage, image acquisition from the RGBD camera, LIDAR scans etc. are included here. The Essential core represents the platform itself and without these, the model is purely a simulation. In Figure 3, this represents motor commands written and image capture.
Interpretation	This includes methods or functions that interpret sensor data according to the application at hand. Analyzing images to detect obstacles or road signs, detecting lanes, etc. all belong here. In Figure 3, this involves analyzing images for pedestrians or stop signs, and detecting lanes.
Control	This serves as the classic control group of methods. Cruise control, steering-based speed reduction, etc. get included here. In Figure 3, this involves a cruise controller to follow the desired speed command.
Decision & Planning	These modules include the higher-level decision making that is involved with autonomous or intelligent systems. In the example in Figure 3, this subsystem calculates the desired speed to pass to the Control unit depending on the desired position from the UI unit and pedestrian/road sign data from the Interpretation unit.
Miscellaneous	This includes other support methods, such as stream communications, communication checks, etc. that act as a glue for the other units.

Table 1. Core units and their usage

E. Code generation, deployment and monitoring

Once you are done with code development, you must generate C code before deploying it to the target. Follow these steps:

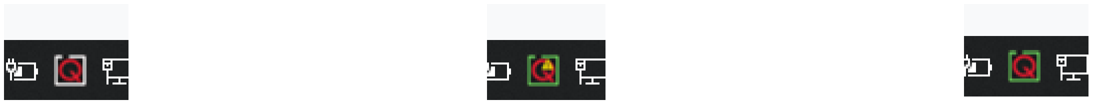
1. **Build and download your code**: click build icon or pressing **`Ctrl+B`**. Open the **Diagnostic Viewer** (View menu) to view the progress. In case you are deploying the code to an external target, this also downloads the code.
2. **Connect to the target**: click on the connect icon or press **`Ctrl+Shift+O`**.
3. **Start your model**: click on the start icon or press **`Ctrl+Shift+Q`**. Your model should start running and the simulation time should advance (bottom right corner of the model).
4. **Stop your model**: When ready to stop execution, click on the stop icon or press **`Ctrl+Shift+W`**.

F. QUARC Monitor and Console

From the **Start** menu, find and launch the **QUARC Monitor** app. It will appear on the bottom-right corner of your screen as a tray icon as shown in Figure 4a. Right-click this icon, and set the **Target** to **Remote** or **Local Windows** depending on where your QUARC application is being deployed. For the **Remote** case, set the **Target URI** to the following,

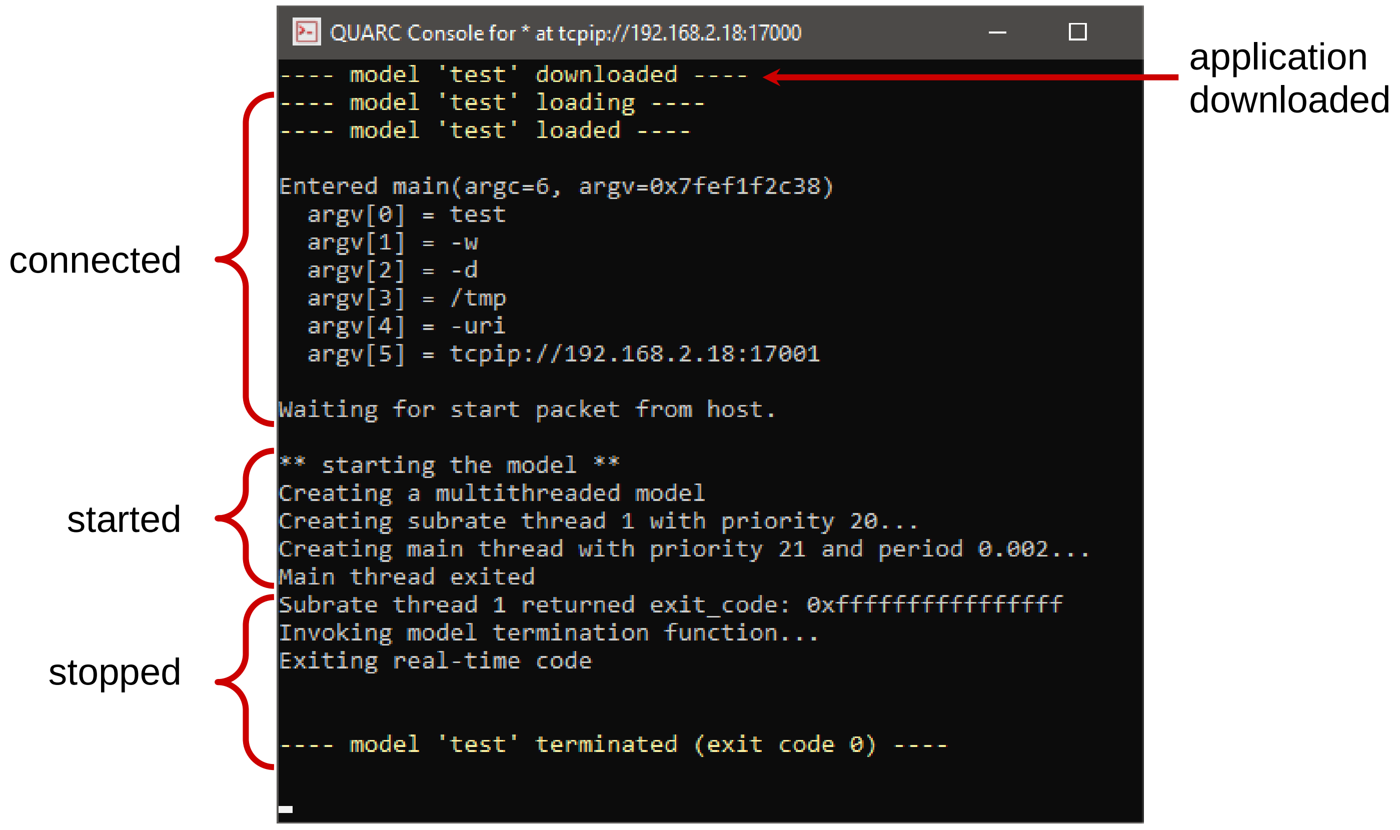
```
tcpip://IP_ADDRESS:17000
```

where IP_ADDRESS corresponds to the IPv4 address of the QCar (e.g., 192.168.2.11). When the monitor searches for the target, a yellow warning symbol is also displayed as shown in Figure 4b. Once connected, the warning sign disappears as in Figure 4c. Right-click the icon now and click on **Console…**. This opens a QUARC console that shows additional information to validate the model being downloaded, connected to and started (Figure 5).



a. QUARC monitor tray icon b. connecting to remote target c. connected to remote target

Figure 4. QUARC Monitor tray icon and connecting to target



```
---- model 'test' downloaded ----
---- model 'test' loading ----
---- model 'test' loaded ----

Entered main(argc=6, argv=0x7fef1f2c38)
  argv[0] = test
  argv[1] = -w
  argv[2] = -d
  argv[3] = /tmp
  argv[4] = -uri
  argv[5] = tcpip://192.168.2.18:17001

Waiting for start packet from host.

** starting the model **
Creating a multithreaded model
Creating subrate thread 1 with priority 20...
Creating main thread with priority 21 and period 0.002...
Main thread exited
Subrate thread 1 returned exit_code: 0xffffffffffffffff
Invoking model termination function...
Exiting real-time code


---- model 'test' terminated (exit code 0) ----
```

Figure 5. QUARC Console for additional debugging information

G. Upgrading QUARC

Quanser regularly updates QUARC with new features. Many of these features only require that you update your development PC with the latest version of QUARC. The target device (the QCar) can typically continue to operate with the original version of QUARC, however, if a target update of QUARC is require this will be noted in the change log of this content package. To upgrade QUARC on the QCar, please see the user guide on **Customizing Your QCar**.

H. Basic Troubleshooting

This non-exhaustive list consists of common errors when building code, or connecting to and starting applications. If your software error is not present here, please contact tech@quanser.com.

1. **The 'Build' step completes successfully but the MATLAB Command Window displays**
   ```
   '??? Model ***model_name*** cannot be downloaded to target 'tcpip://IP_ADDRESS:17000?keep_alive=1'. It
   was not possible to connect to the specified URI.'
   ```

   This indicates that even though the build succeeded and the application is ready, the 'download' step has failed due to QUARC not being able to find the target.

   a. Check your connection to the target - both your local machine and the target should be on the same network. For more details, see the **III - Connectivity** user guide.

   b. Check the **IP_ADDRESS** of your target in the **MEX-file arguments**
   c. Check the Windows Firewall setting to ensure the requested port numbers in the model URI and in the QUARC Stream API blocks such as Stream Server and Stream Client (e.g., 17000-17020 and 18000-18999) are allowed to pass through (for both incoming and outgoing TCP/IP and UDP traffic).

   Once your connection to the target resumes, download the application already built by clicking on **Download** in the **QUARC** menu in Simulink.

2. **The 'Connect' step returns the following error in the Diagnostic Viewer**
   ```
   'Error occurred while executing External Mode MEX-file 'quarc_comm':
   Unable to establish connection with QUARC Target manager for external mode communications. It was not
   possible to connect to the specified URI. Verify that the target is serving on URI
   tcpip://IP_ADDRESS:17000?keep_alive=1'. Use the QUARC Console for debugging.'
   ```

   This mirrors issue (1) described above. Communication with the **Quarc Target Manager** application could not be established because the target could not be found. Follow the instructions given in the troubleshooting steps for issue (1) to establish the connection.

3. **The 'Connect' step returns the following error in the Diagnostic Viewer**
   ```
   'Error occurred while executing External Mode MEX-file 'quarc_comm':
   Unable to establish connection with real-time model for external mode communications. The remote peer
   refused the connection, most likely because no server application was listening for connections. Verify
   that your real-time model is serving on URI 'tcpip://IP_ADDRESS:PORT'.
   ```

   The connection between your development machine and the target was successful, however, the built application was not downloaded successfully and hence cannot be found. Click on **Download** in the **QUARC** menu to download the built application to your target and **Connect** again.

   Note: It is common to get this error after the error in 1.b. If the connection to the target wasn't established when you used Build to build your application, it will fail to download the model to the target. After resuming connection, trying to Connect directly will also fail, resulting in this error. Try downloading the model again should resolve this issue.

4. **Error occurred while executing External Mode MEX-file 'quarc_comm': The card specific option specified is not recognized.**

   This sometimes occurs when models are moved between Matlab versions. Find the HIL Initialize block in the Essentials subsystem and double click on it to open the dialog. Confirm that QCar is selected then click on the Defaults button at the bottom of the dialog to restore the default board specific options.