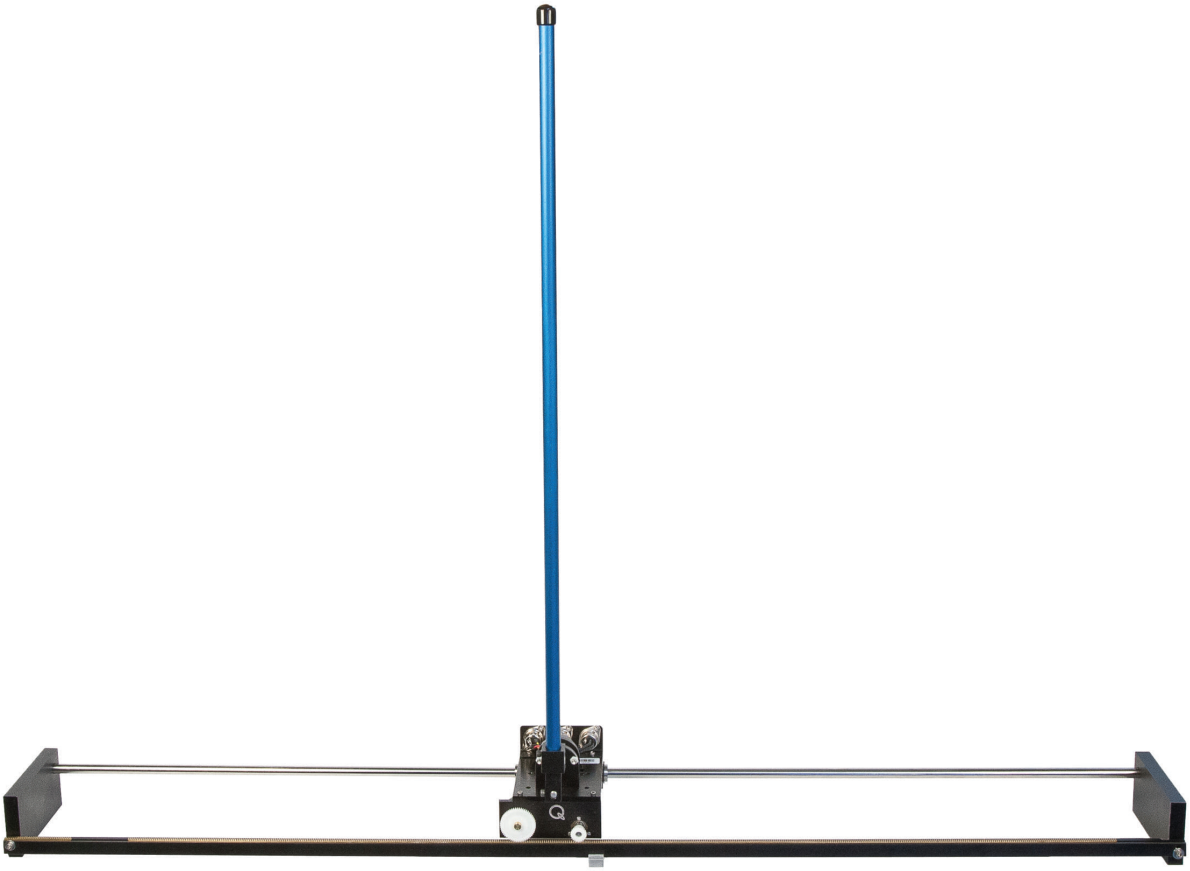
UCSB Controls Lab TA Manual

By Max Crisafulli

September 2024



This document lives at
https://github.com/MaxCrisafulli/UCSB-ControlsLab-Manual-2024

TA Manual for the UCSB ECE Controls Lab

This document is to act as a guide for TAs in setting up the Quanser inverted pendulum cart hardware in the UCSB ECE Controls Lab (Harold Frank Hall 3120A). This guide is an update of the existing manual, located at:

`https://github.com/justinpearson/UCSB-Quanser-Inverted-Pendulum-Lab-Manual`

This updated guide will live (and hopefully be maintained) at

`https://github.com/MaxCrisafulli/UCSB-ControlsLab-Manual-2024`

Hardware In The Lab

The ECE147 labs make use of the Quanser 'Linear Servo Base Unit with Inverted Pendulum'. This manual will detail the setup of just the cart system, with motor voltage as input and linear position as output. There are 3 main components of the hardware setup, with each of which being explained below.

Q4 DAQ Board

The SIMULINK model that interfaces with the hardware interacts with the world via the Q4 HIL control card and DAQ (Data Acquisition Board). The control card is inside the lab PCs connected via PCIe and interfaces with the DAQ board via the two gray ribbon cables. Ensure that these are connected properly and that the red Status LED is lit, indicating that the board is powered.

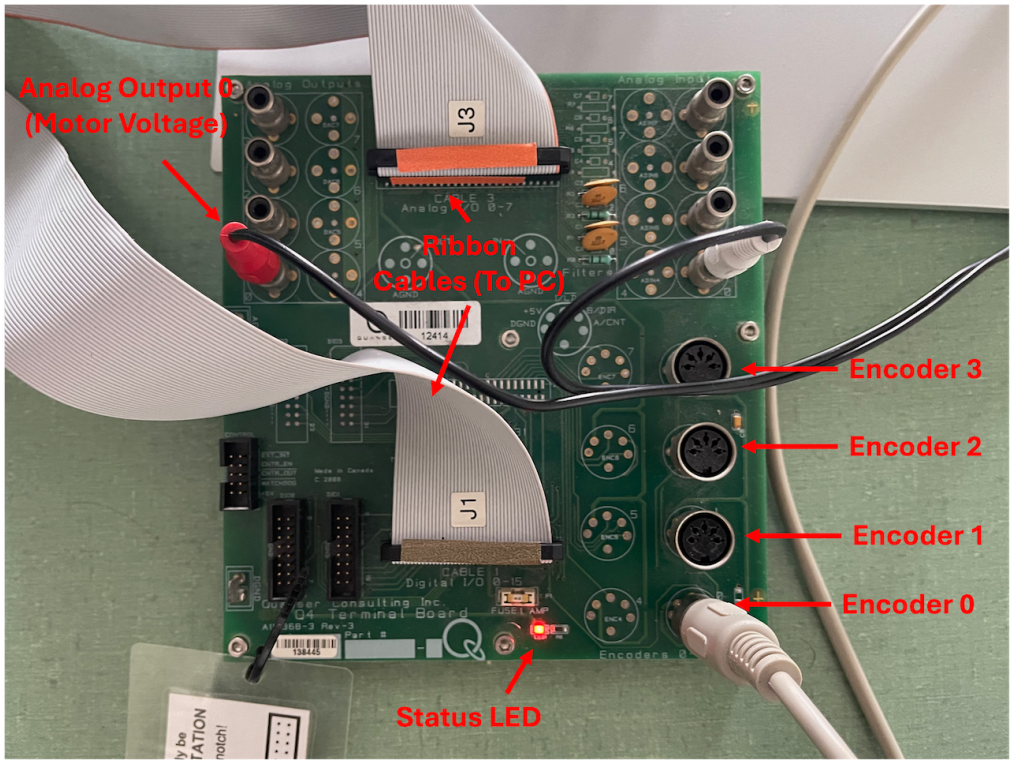


Figure 1: Quanser Q4 Terminal DAQ Board

The DAQ board uses the four encoder ports to read signals from the hardware. Generally, Encoder 0 is used for the linear position encoder. In the top left corner of the board are the Analog Outputs, Analog Output 0 is used to drive the amplifier box which ultimately drives the cart motor.

VoltPAQ-X1 Amplifier

The Quanser VoltPAQ-X1 Amplifier is powered from a standard wall socket, and turned on via a switch at the back of the box. Make sure that the green status LED in the center of the amplifier diagram is lit, indicating that the amplifier is powered on. Also check that the Amplifier Gain switch is set to 1x.

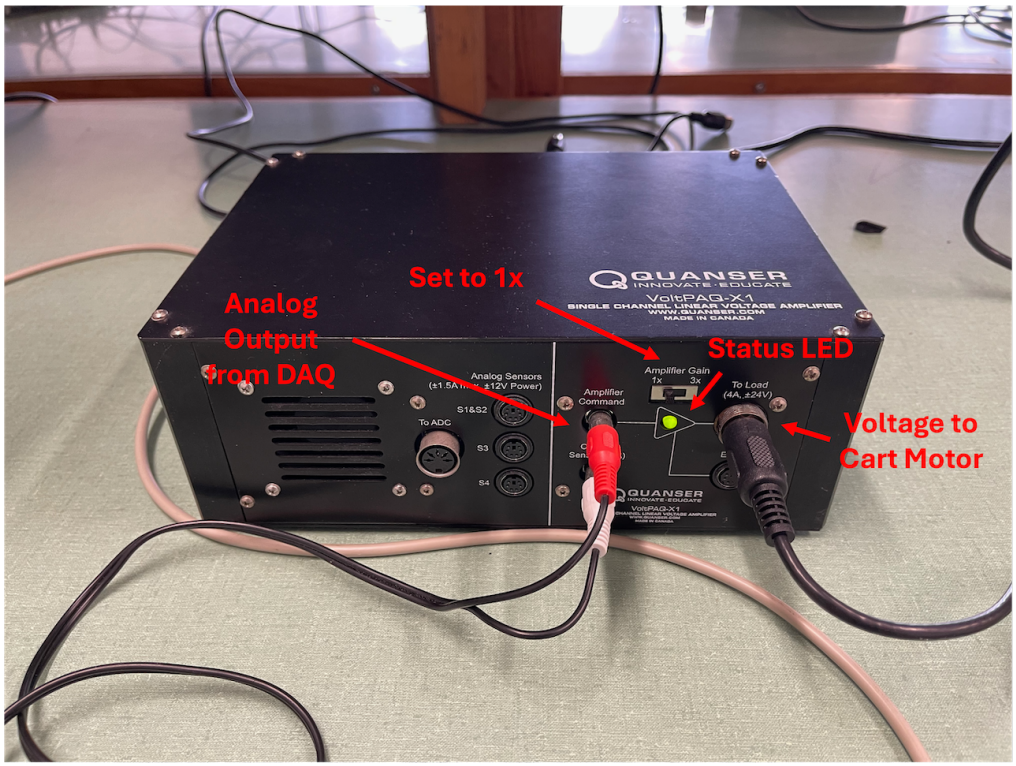


Figure 2: Quanser VoltPAQ-X1 Linear Voltage Amplifier

The red AV cable coming from Analog Output 0 on the DAQ board should be connected to the Amplifier Command (input) port. The white AV cable should be connected to Current Sense on the amplifier and Analog Input 0 on the DAQ board, although this is not necessary.

The output of the amplifier (To Load) is what drives the motor onboard the cart. Check that all the cable connections are correct and that the cable going to the motor has enough slack to allow the cart to move along the track.

Inverted Pendulum Cart System

The cart system has 2 encoder outputs and 1 analog voltage input. The positions of these ports may vary between the carts but in general they are as depicted in Figure 3. You should check what each of the ports connect to before plugging anything in.

The cart controls its position along the track with the smaller motor drive wheel, which is driven by the motor voltage coming from the amplifier. The larger position encoder wheel measures the relative position along the track from the time that the program was initialized in 'ticks'. The conversion ratio from ticks to inches is approximately 5000 ticks per 6 inches.

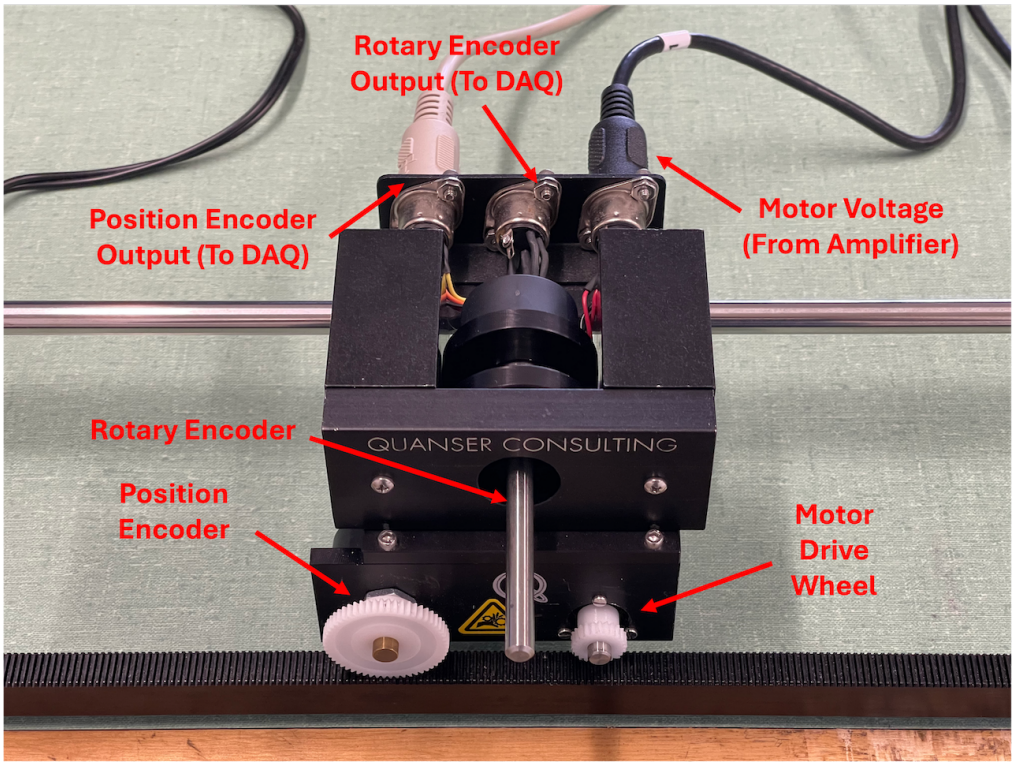


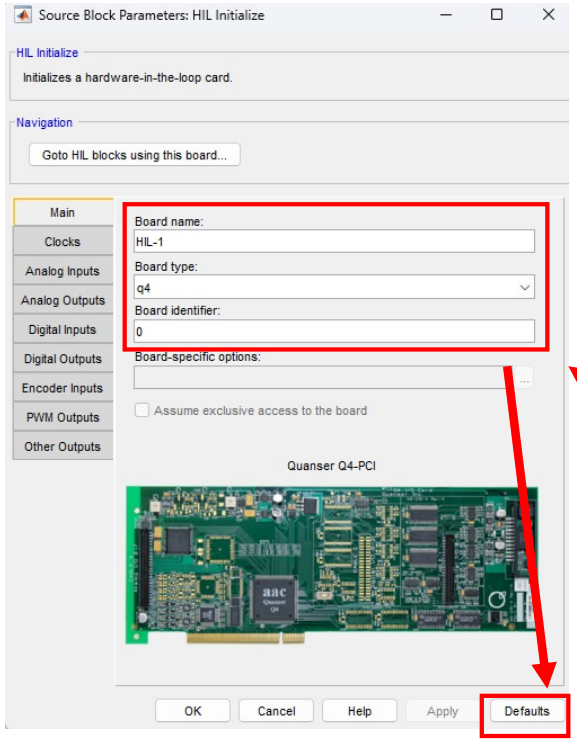
Figure 3: Quanser Linear Servo Base Unit (The 'Cart')

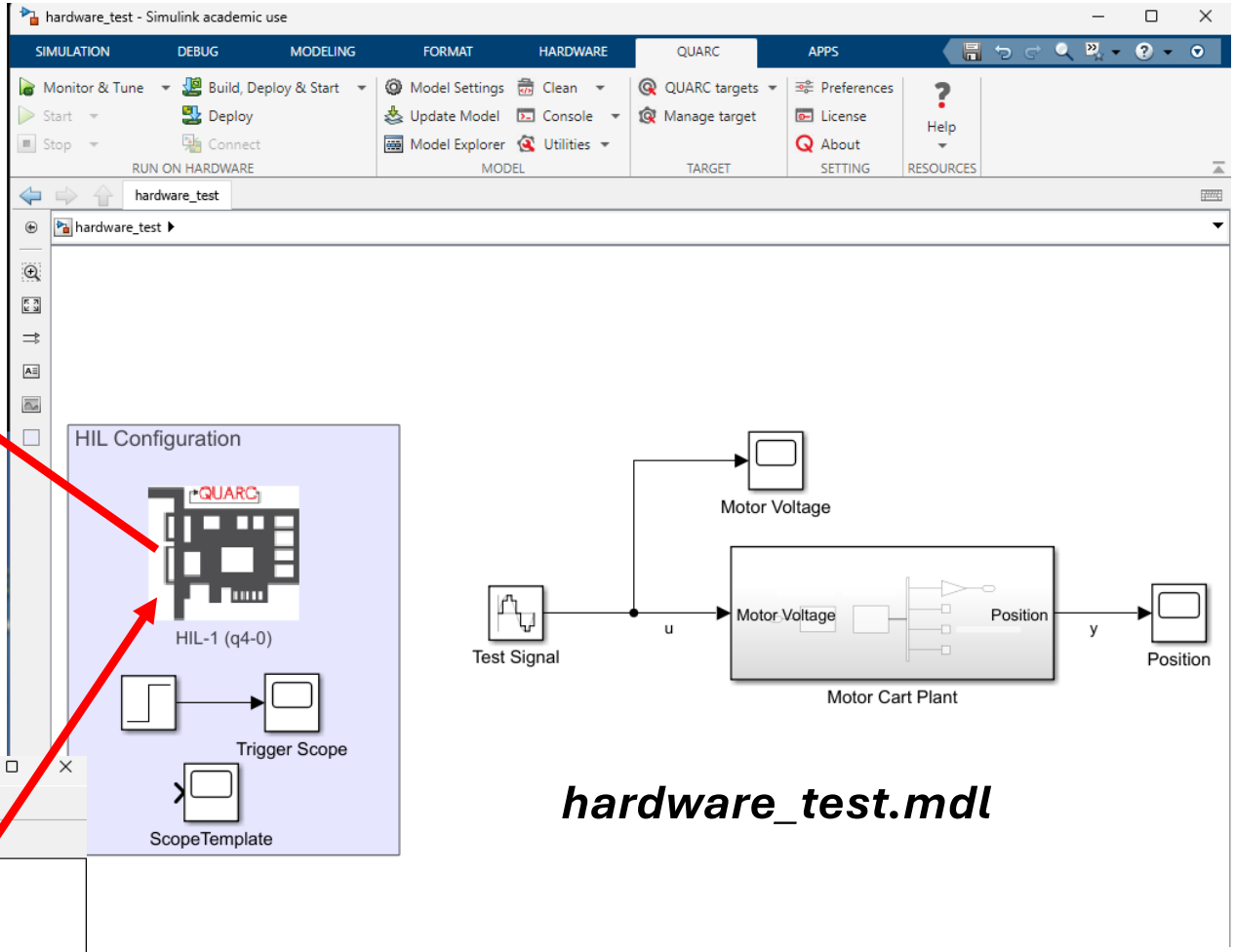
Instructions

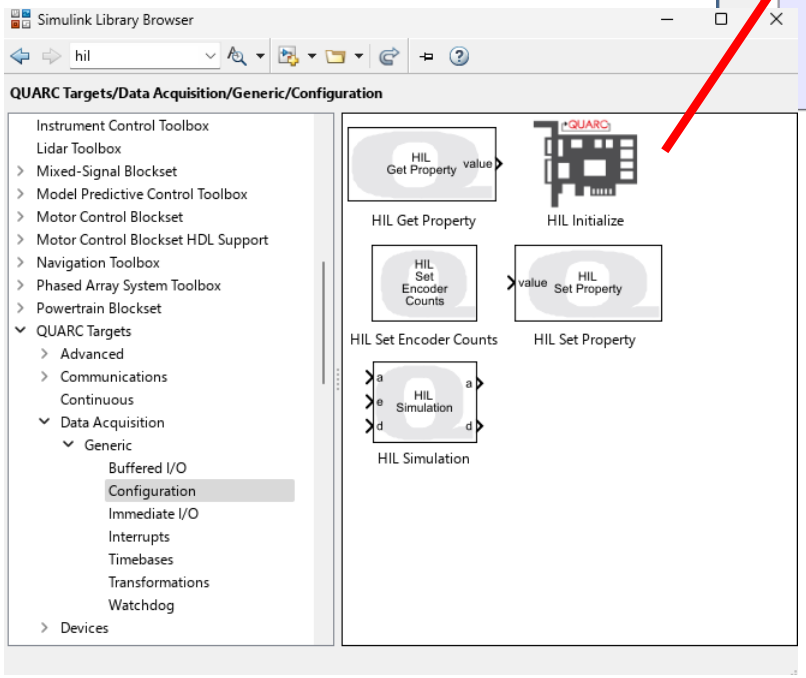
The following slides will detail setting up a minimum working SIMULINK model that can pass a voltage to the motor and read position from the cart encoder.
Ensure that you are using MATLAB R2023a and that the QuaRC package (Quanser Realtime Control) is installed. Run the command `mex -setup` and `mex -setup C++` in the command line to make sure that MATLAB can interface with a compiler, which should be `Microsoft Visual C++ Redistributable 2022`.

The SIMULINK model `hardware_test.mdl` in the Github repo should be configured in such a way that it can be run (assuming all the hardware is setup correctly). It can also be used as a template for other models using the same Hardware.

Configuring the model to work with DAQ



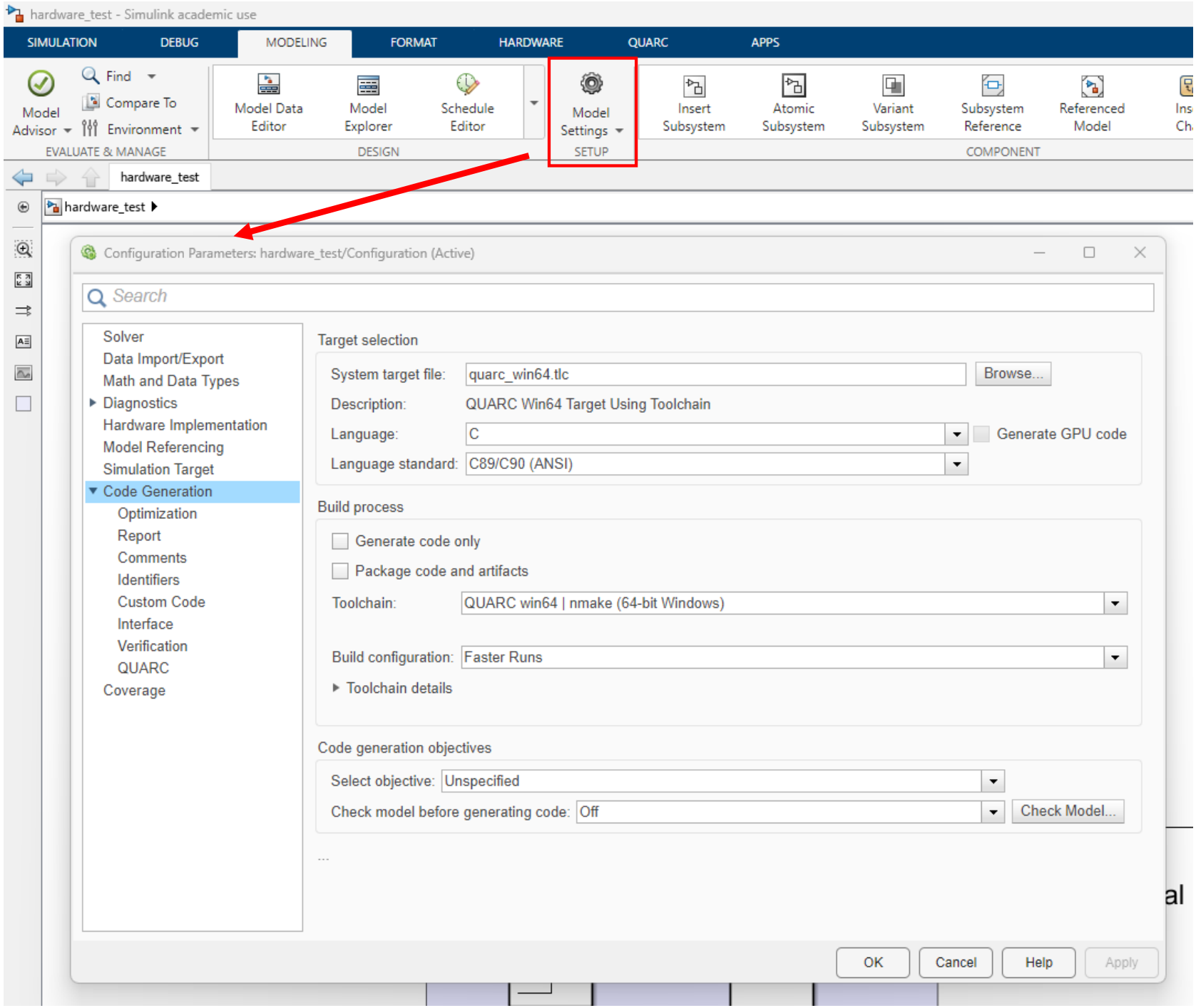


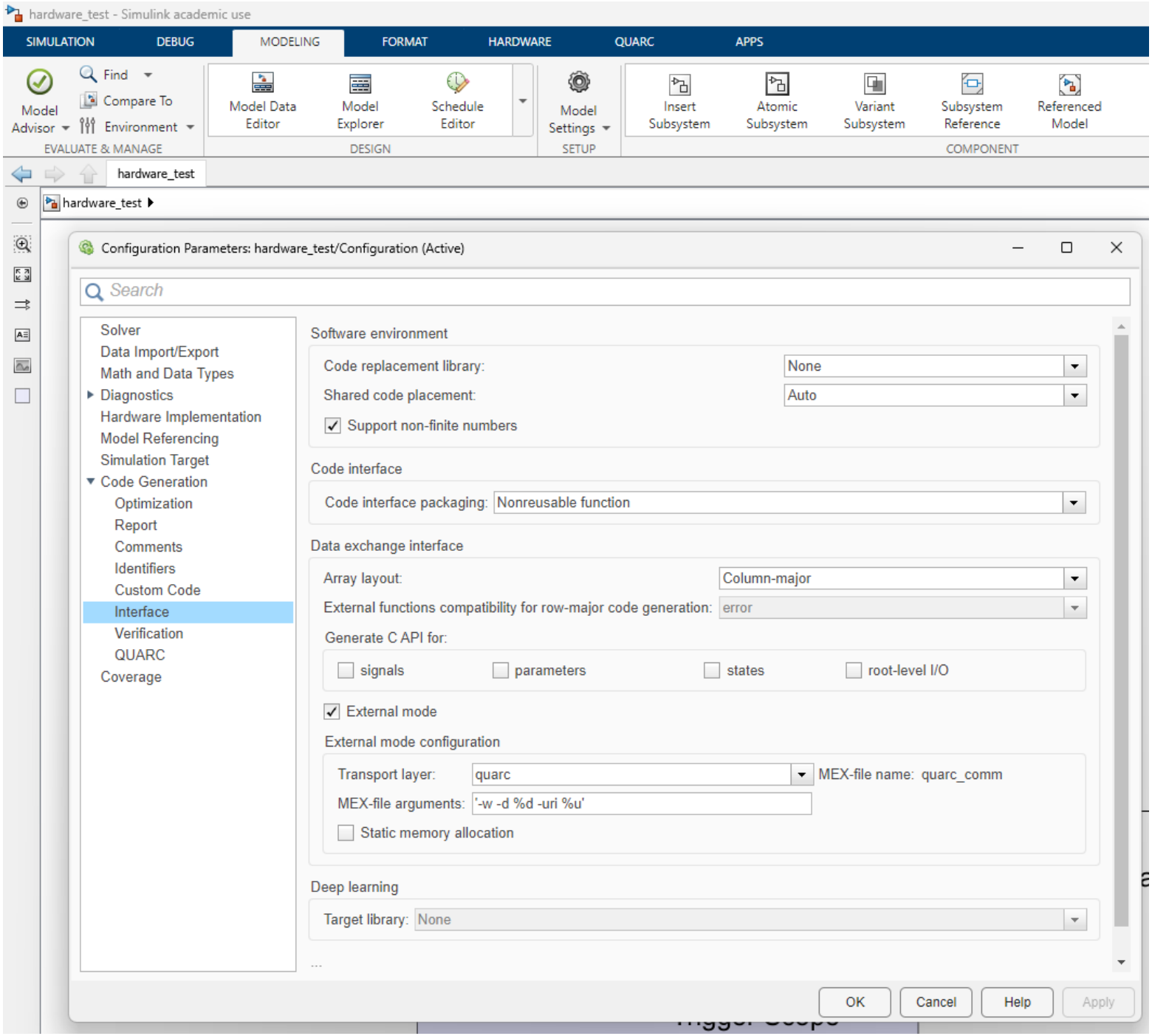


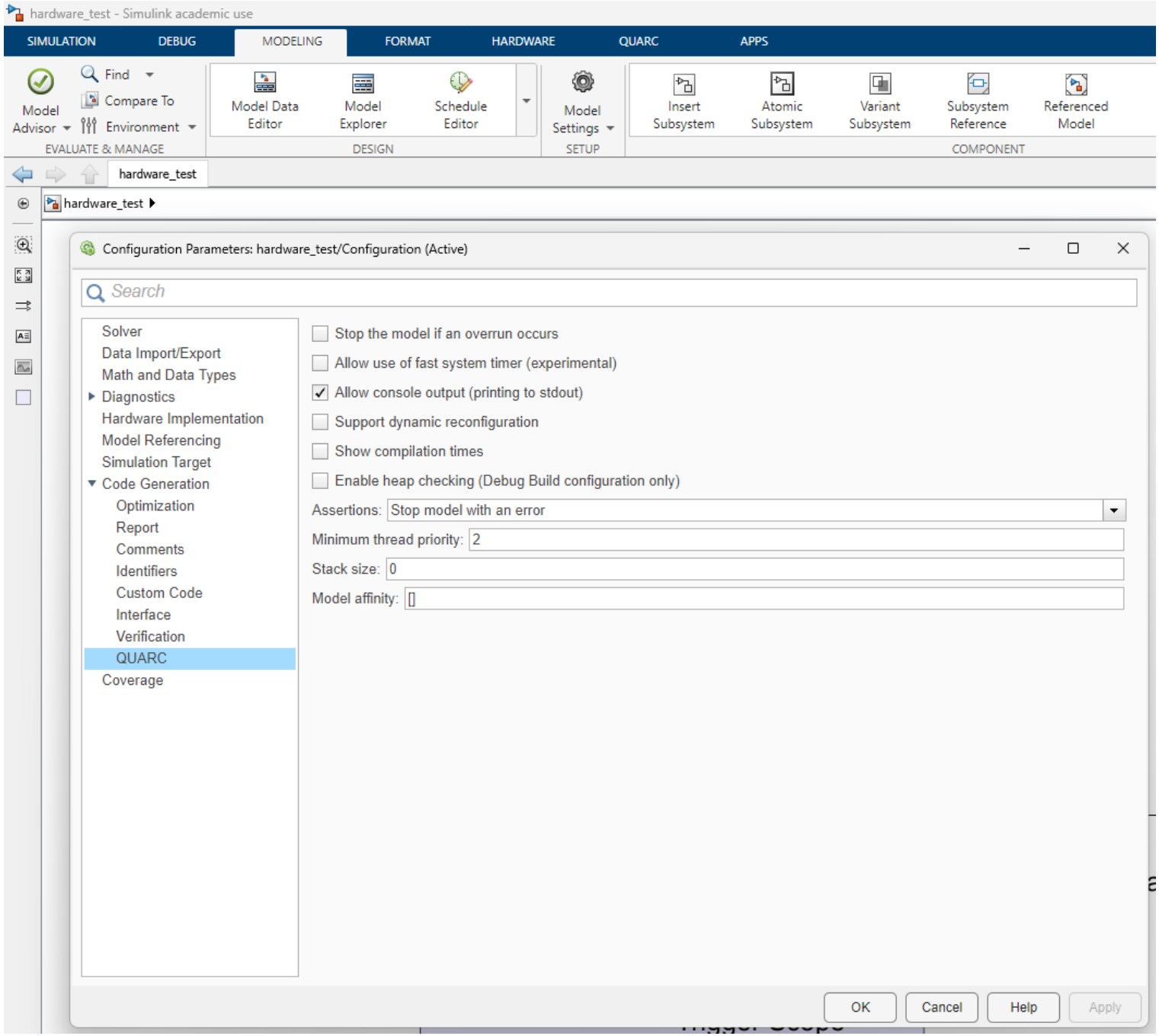
hardware_test.mdl

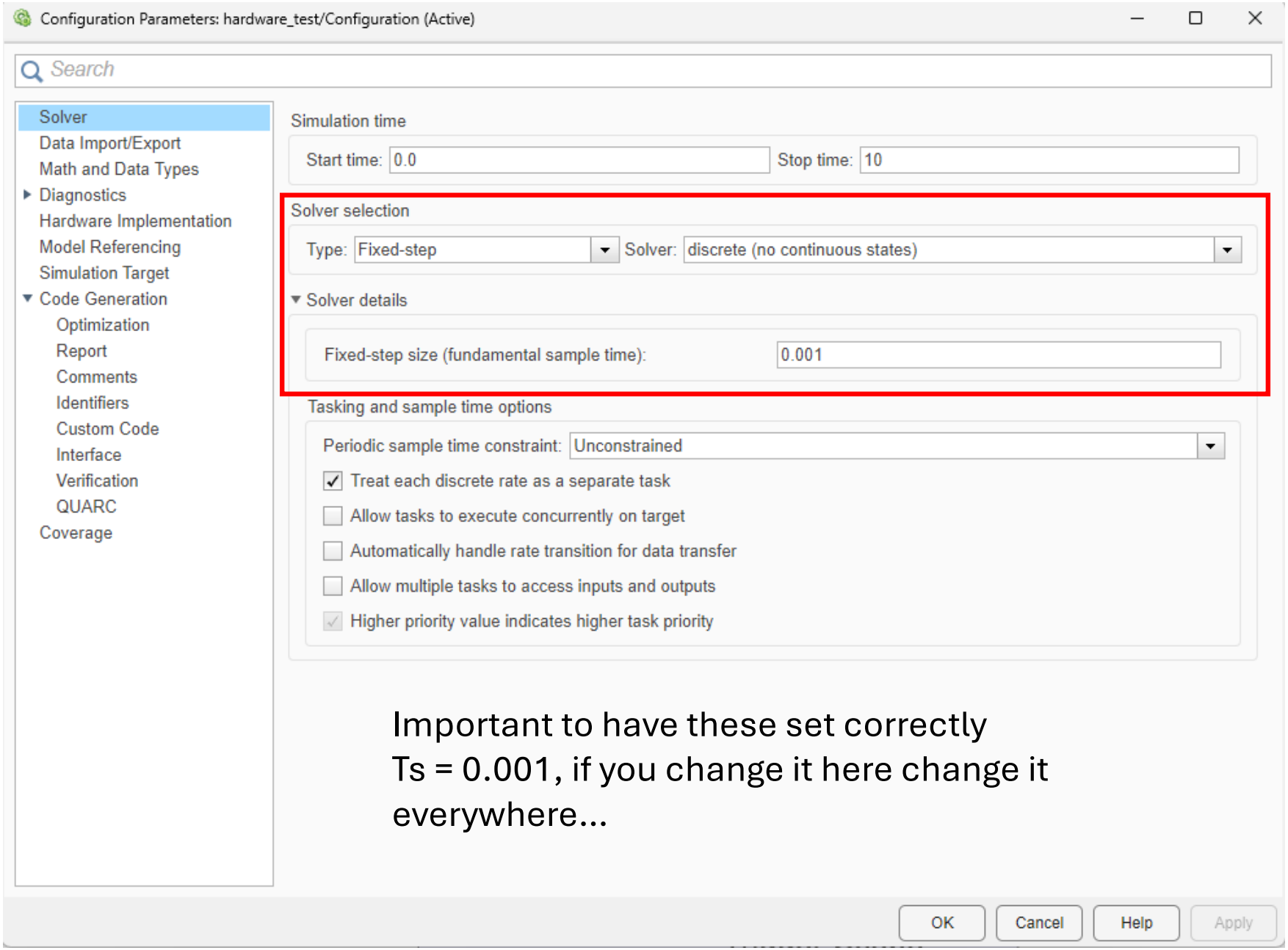
Set the HIL Initialize block to have board name HIL-1 and Board type q4. Click defaults to set up

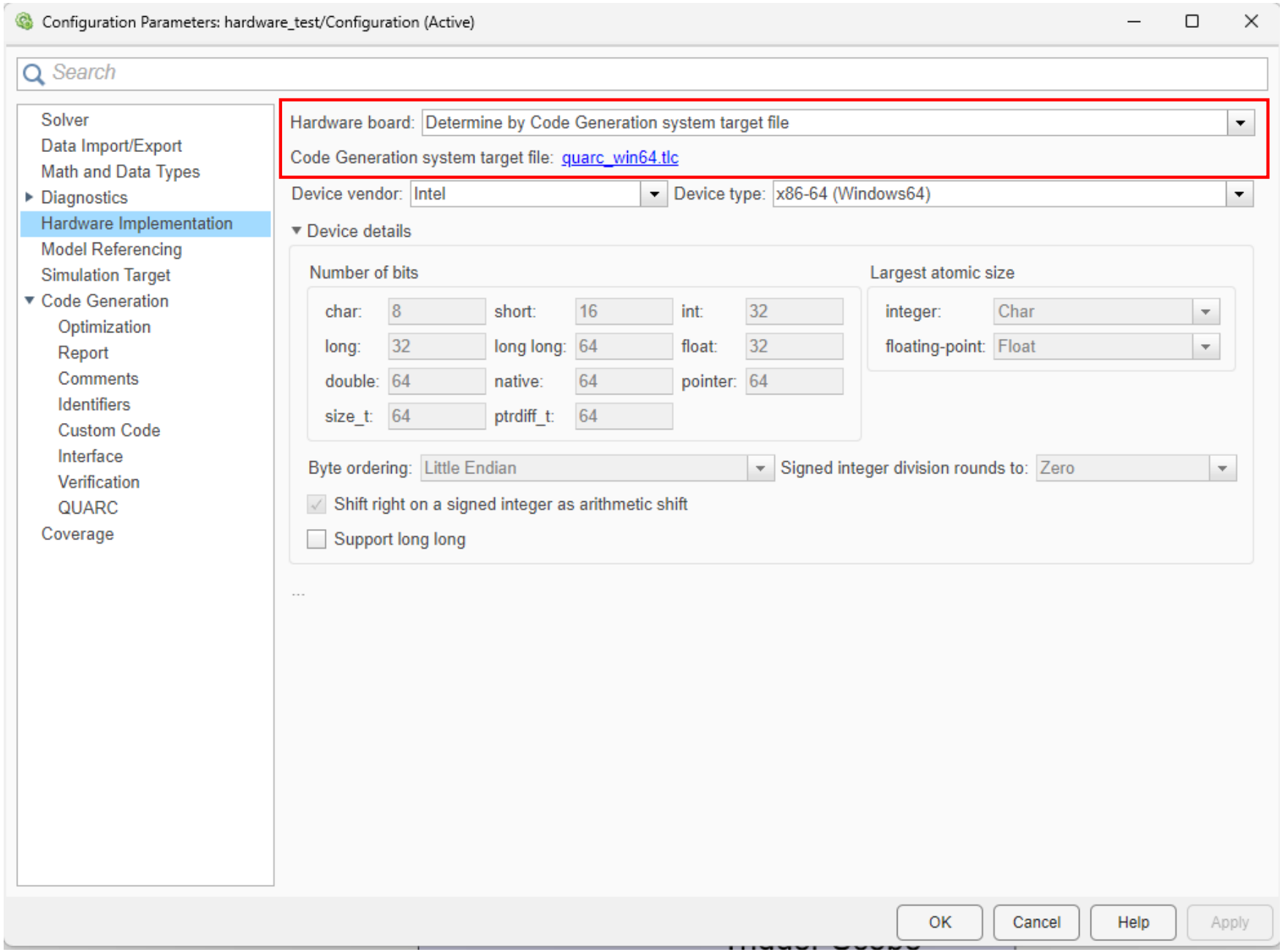
Configuring model parameters for HIL & Quarc



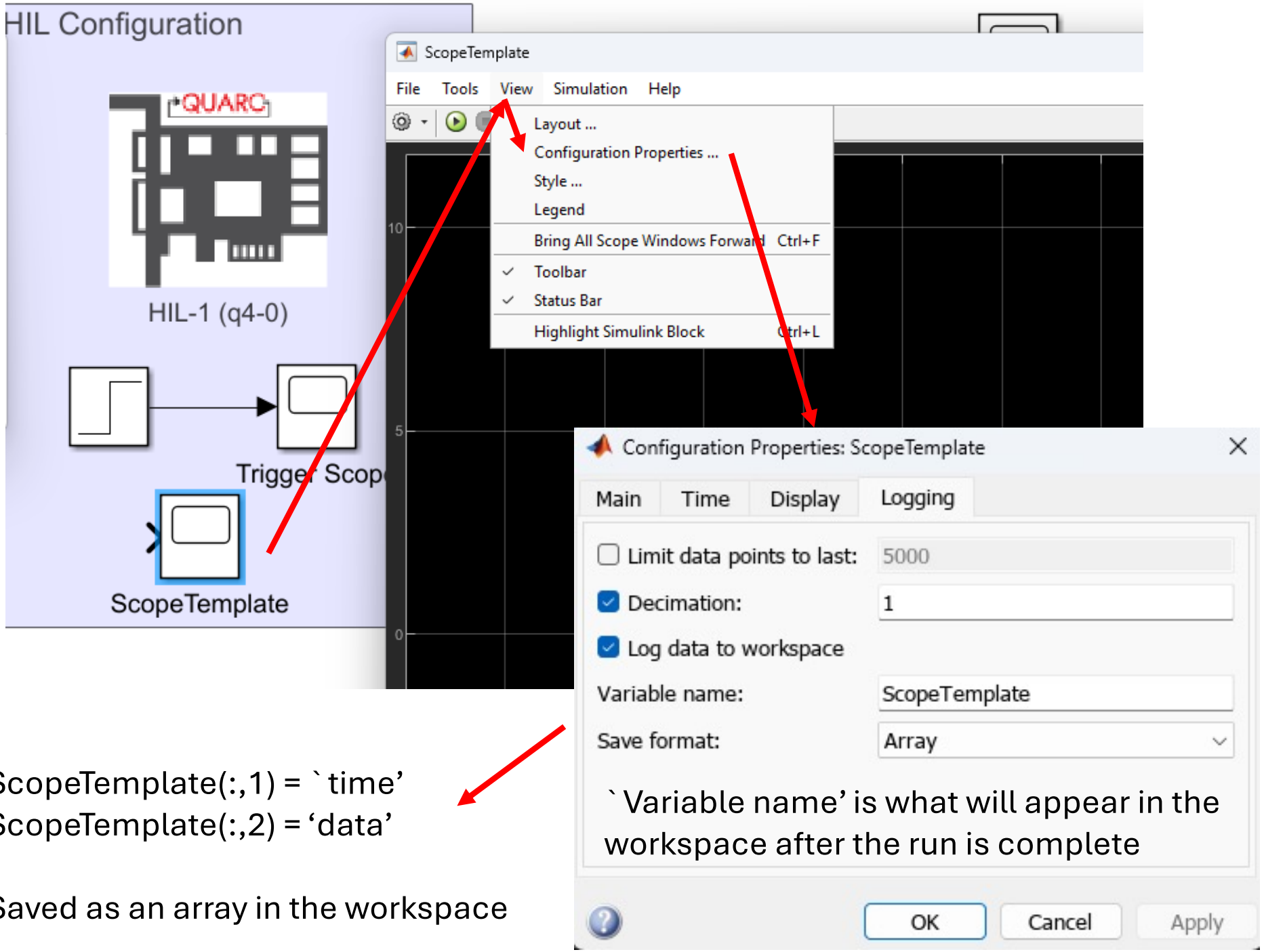

hardware_test - Simulink academic use
SIMULATION
DEBUG
MODELING
FORMAT
HARDWARE
QUARC
APPS
Model Advisor
Find
Compare To
Environment
EVALUATE & MANAGE
Model Data Editor
Model Explorer
Schedule Editor
DESIGN
Model Settings
SETUP
Insert Subsystem
Atomic Subsystem
Variant Subsystem
Subsystem Reference
Referenced Model
COMPONENT
hardware_test
hardware_test
Configuration Parameters: hardware_test/Configuration (Active)
Search
Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Optimization
Report
Comments
Identifiers
Custom Code
Interface
Verification
QUARC
Coverage
Software environment
Code replacement library: None
Shared code placement: Auto
Support non-finite numbers
Code interface
Code interface packaging: Nonreusable function
Data exchange interface
Array layout: Column-major
External functions compatibility for row-major code generation: error
Generate C API for:
signals
parameters
states
root-level I/O
External mode
External mode configuration
Transport layer: quarc
MEX-file name: quarc_comm
MEX-file arguments: '-w -d %d -uri %u'
Static memory allocation
Deep learning
Target library: None
OK
Cancel
Help
Apply


hardware_test - Simulink academic use
SIMULATION
DEBUG
MODELING
FORMAT
HARDWARE
QUARC
APPS
Model Advisor
Find
Compare To
Environment
EVALUATE & MANAGE
Model Data Editor
Model Explorer
Schedule Editor
DESIGN
Model Settings
SETUP
Insert Subsystem
Atomic Subsystem
Variant Subsystem
Subsystem Reference
Referenced Model
COMPONENT
hardware_test
hardware_test
Configuration Parameters: hardware_test/Configuration (Active)
Search
Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Optimization
Report
Comments
Identifiers
Custom Code
Interface
Verification
QUARC
Coverage
Stop the model if an overrun occurs
Allow use of fast system timer (experimental)
Allow console output (printing to stdout)
Support dynamic reconfiguration
Show compilation times
Enable heap checking (Debug Build configuration only)
Assertions: Stop model with an error
Minimum thread priority: 2
Stack size: 0
Model affinity: []
OK
Cancel
Help
Apply


Configuration Parameters: hardware_test/Configuration (Active)
Search
Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Optimization
Report
Comments
Identifiers
Custom Code
Interface
Verification
QUARC
Coverage
Simulation time
Start time: 0.0
Stop time: 10
Solver selection
Type: Fixed-step
Solver: discrete (no continuous states)
Solver details
Fixed-step size (fundamental sample time): 0.001
Tasking and sample time options
Periodic sample time constraint: Unconstrained
Treat each discrete rate as a separate task
Allow tasks to execute concurrently on target
Automatically handle rate transition for data transfer
Allow multiple tasks to access inputs and outputs
Higher priority value indicates higher task priority
Important to have these set correctly
Ts = 0.001, if you change it here change it everywhere...
OK
Cancel
Help
Apply


Configuration Parameters: hardware_test/Configuration (Active)
Search
Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Optimization
Report
Comments
Identifiers
Custom Code
Interface
Verification
QUARC
Coverage
Hardware board: Determine by Code Generation system target file
Code Generation system target file: quarc_win64.tlc
Device vendor: Intel
Device type: x86-64 (Windows64)
Device details
Number of bits
char: 8
short: 16
int: 32
long: 32
long long: 64
float: 32
double: 64
native: 64
pointer: 64
size_t: 64
ptrdiff_t: 64
Largest atomic size
integer: Char
floating-point: Float
Byte ordering: Little Endian
Signed integer division rounds to: Zero
Shift right on a signed integer as arithmetic shift
Support long long
OK
Cancel
Help
Apply

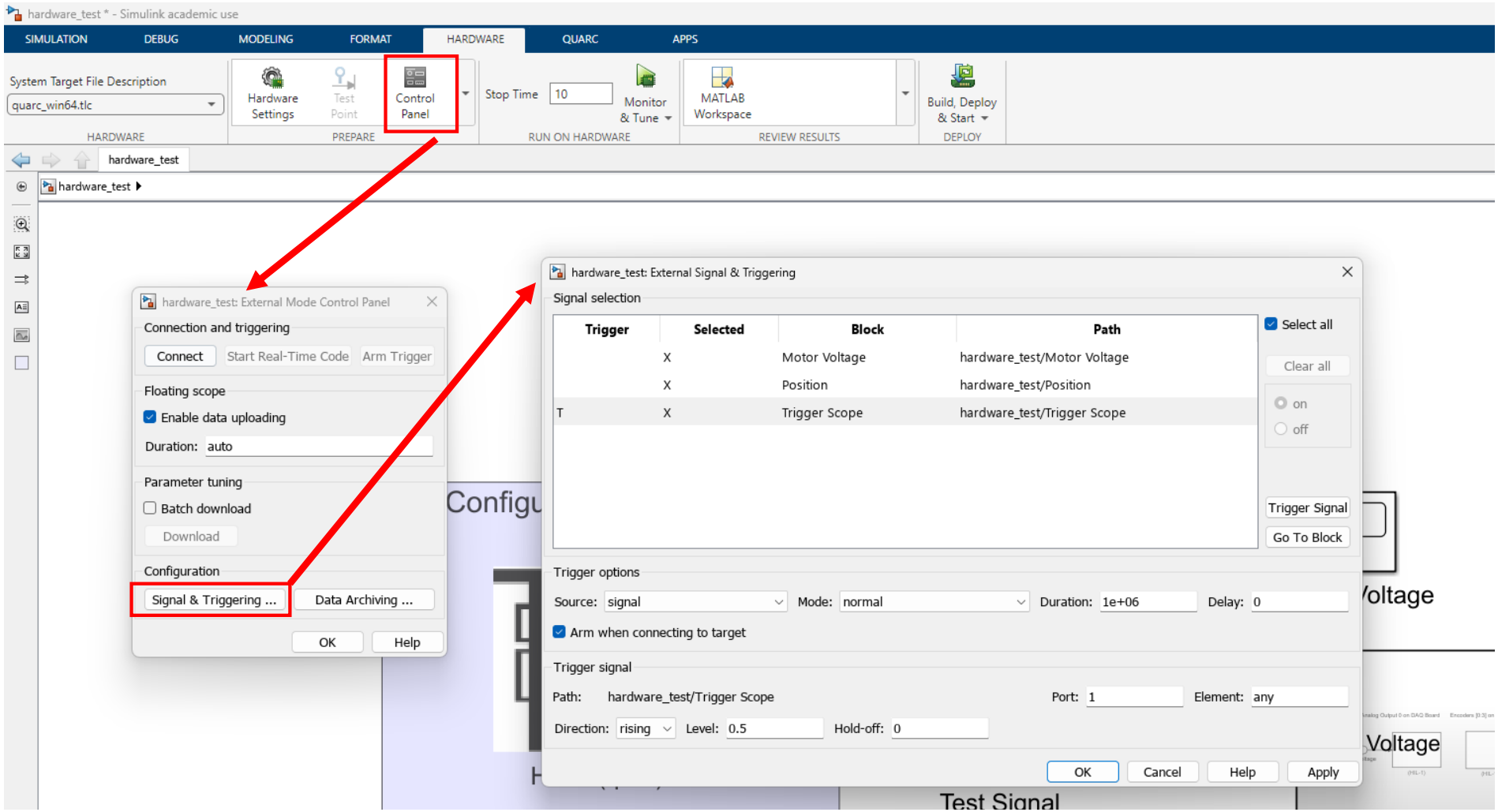
Configuring scopes for data logging:



ScopeTemplate(:,1) = `time'
ScopeTemplate(:,2) = 'data'

Saved as an array in the workspace

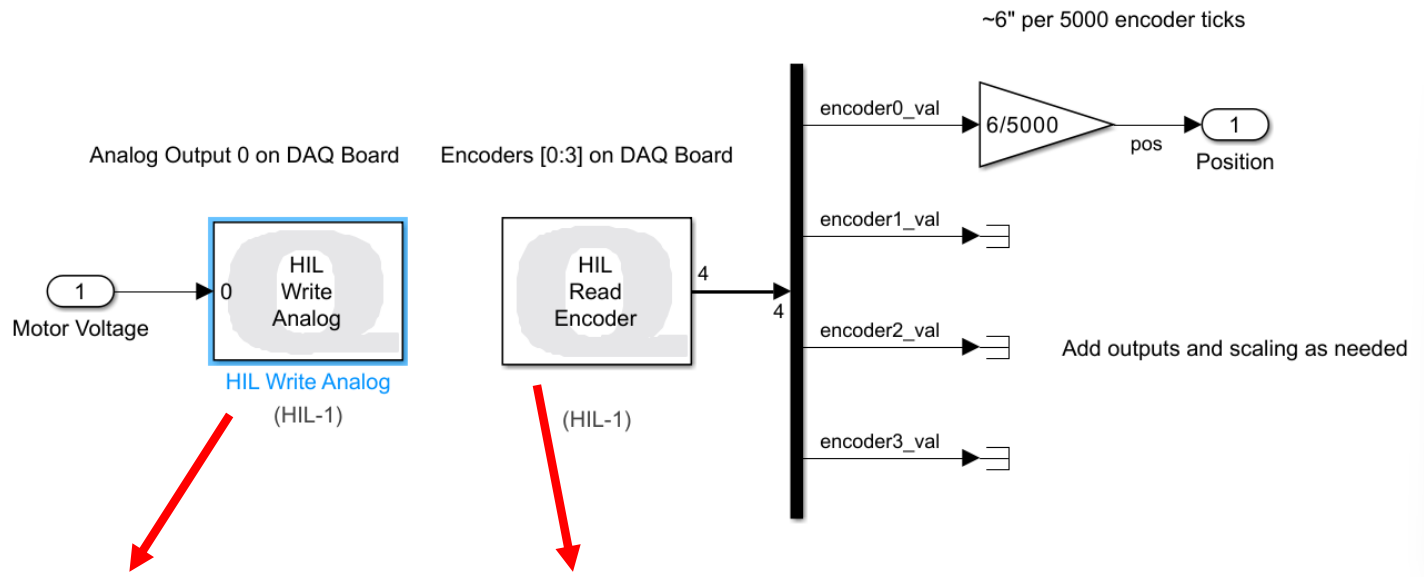
Configuring triggering and model data logging:

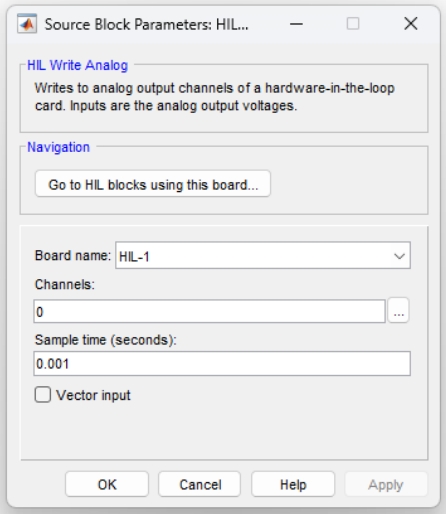


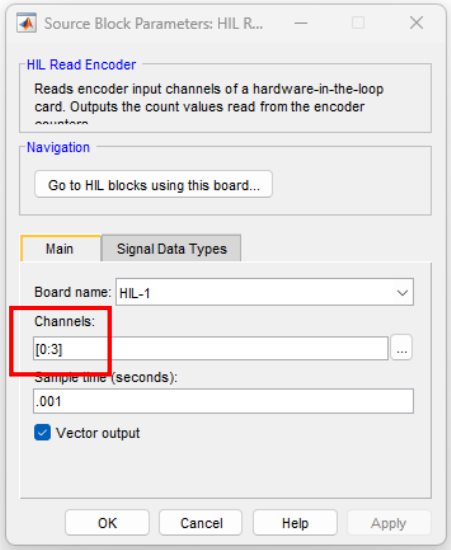
X under Selected means the signal will be logged

T under trigger means that will be used as trigger signal (don't change)

Duration is maximum # of samples logged, *Duration = tStop/Ts*

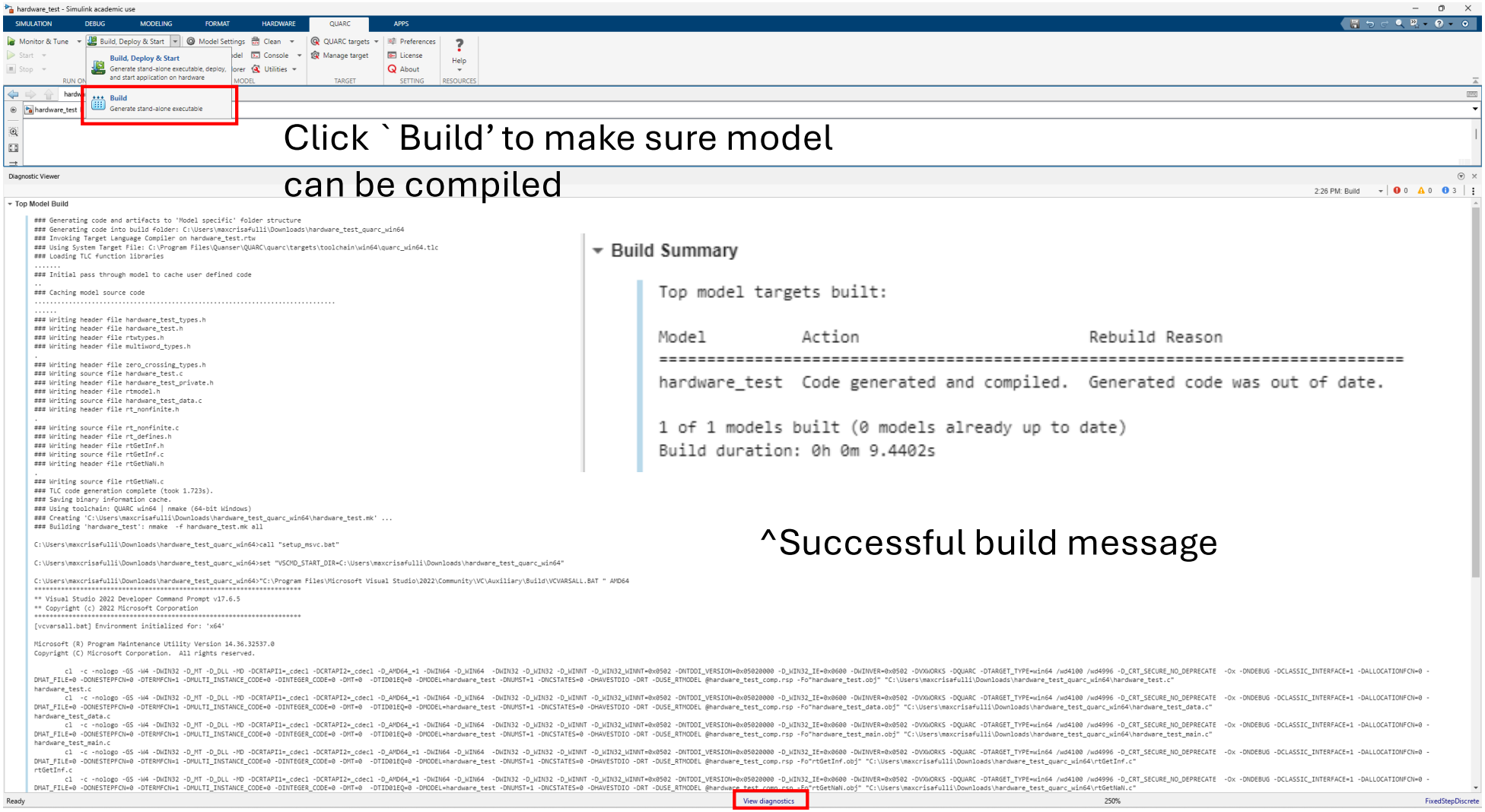






Demo model is reading all 4 encoders as vector and demuxxing (shown above)

Can set to just read a single encode



Click `Build' to make sure model can be compiled

^Successful build message

Monitor & Tune builds & runs the code.
Expected outputs shown