Erkam ÇANKAYA

MATLAB Kıdemli Uygulama Mühendisi

erkam.cankaya@figes.com.tr

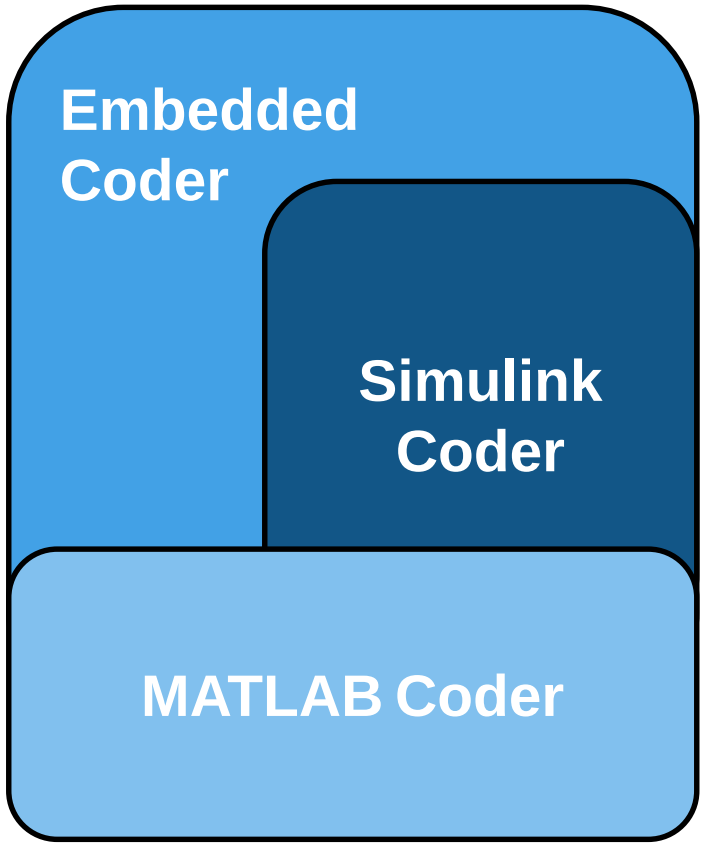
06.11.2019

Agenda

- ✓ Code Generation Family
  - ✓ MATLAB Coder
  - ✓ Simulink Coder
  - ✓ Embedded Coder
  - ✓ Demos

Code Generation Family

Coder Products for Generating C and C++



MATLAB® Coder™
Generate C and C++ from MATLAB

Simulink® Coder™
Generate C and C++ from Simulink and Stateflow

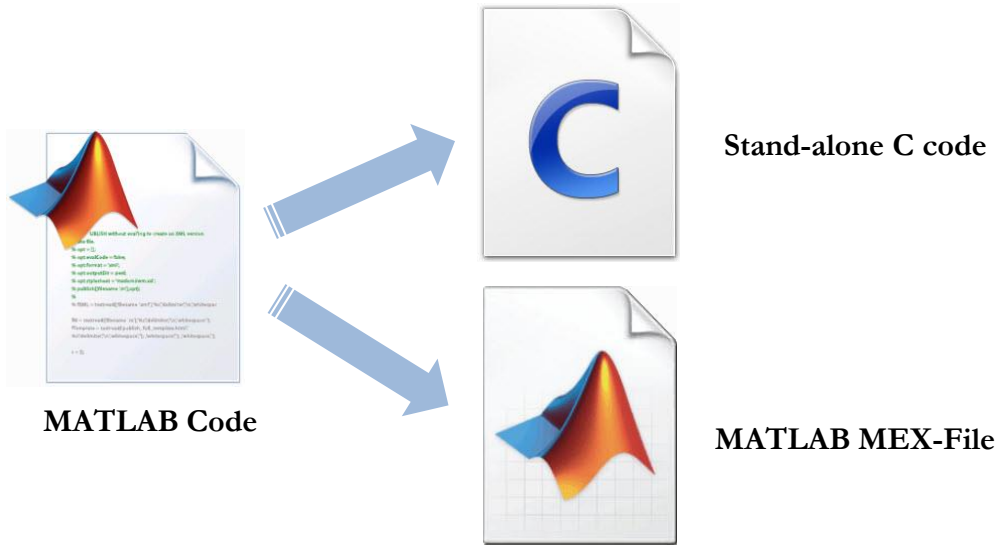
Embedded Coder™
Generate C and C++ from MATLAB and Simulink optimized for embedded systems

Coders are also available for HDL and PLCs

MATLAB Coder Overview

MATLAB Coder supports efficient code generation for prototyping and accelerating algorithms.

- Generate ANSI/ISO compliant C and C++ code (readable, efficient, and standalone) from MATLAB code.
- Generate MEX functions from MATLAB Code
- Integrate custom C/C++ code into MATLAB.



Daimler Designs Cruise Controller for Mercedes-Benz Trucks

> "MathWorks tools for modeling and code generation enabled us to quickly and seamlessly perform design and test iterations, and release our product within a hard deadline of only 18 months."
>
> — *Mario Wünsche, Daimler*

Mercedes-Benz truck.

Products Used

MATLAB
Request trial | Get pricing | Contact sales

Simulink
Request trial | Get pricing | Contact sales

Embedded Coder
Request trial | Get pricing | Contact sales

Fixed-Point Designer
Request trial | Get pricing | Contact sales

MATLAB Coder
Request trial | Get pricing | Contact sales

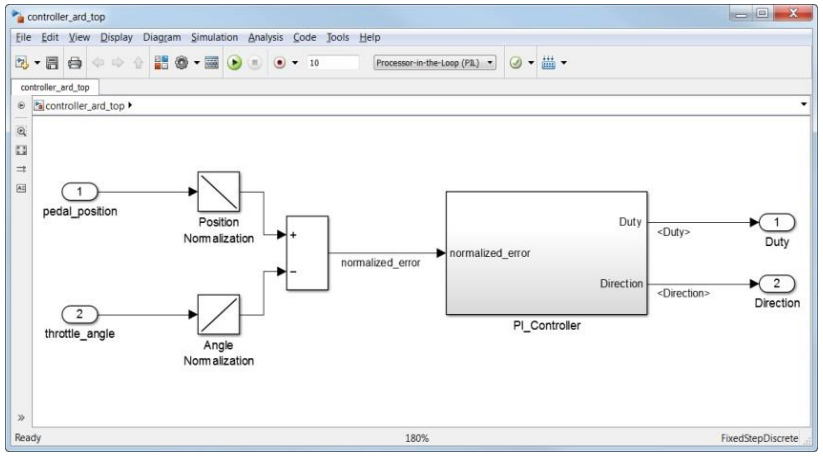
Simulink Coder
Request trial | Get pricing | Contact sales

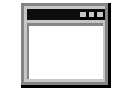
Stateflow
Request trial | Get pricing | Contact sales

Why Automatic Code Generation with Simulink Coder?

Simulink Coder™ automatically generates C code from a model for development of a platform-specific application.

- By automating the development process, you can
- Eliminate manual coding errors
- Maintain code quality in a consistent manner
- Reproduce code at the click of a button





controller.exe

Alstom Generates Production Code for Safety-Critical Power Converter Control Systems

> "We used MathWorks tools to design, test, modify, and implement a control system for a permanent magnet drive in one year. Given the resources available to us, it would have been impossible to deliver this on schedule without MathWorks tools."
>
> — *Han Geerligs, Alstom*

Pendolino tilting train.

Products Used

MATLAB
Request trial | Get pricing | Contact sales

Simulink
Request trial | Get pricing | Contact sales

Control System Toolbox
Request trial | Get pricing | Contact sales

Embedded Coder
Request trial | Get pricing | Contact sales

Simscape Electrical
Request trial | Get pricing | Contact sales

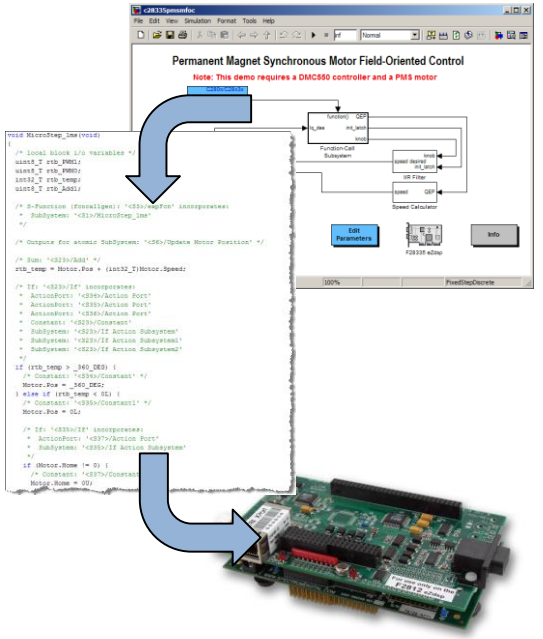
Simulink Coder
Request trial | Get pricing | Contact sales

Stateflow
Request trial | Get pricing | Contact sales

Introduction to Embedded Coder

Generates readable, compact, and fast C and C++ code for:

- Standalone execution
- Microprocessors
- On-target rapid prototyping boards

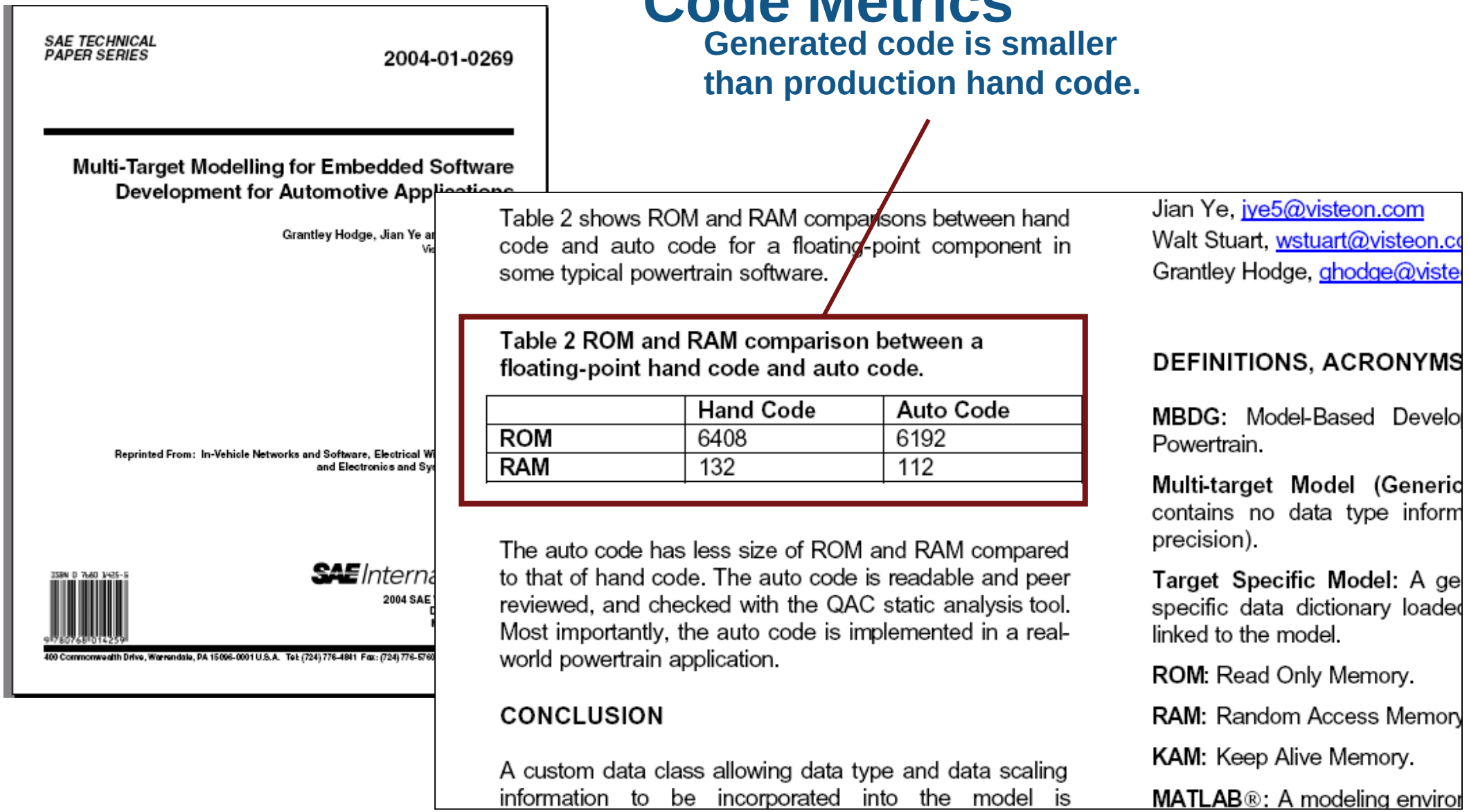


Key Features

- Processor-specific code optimization
- Code verification
  - SIL and PIL testing
  - Code reports with tracing of models to and from code and requirements
- Integration of third-party embedded development environments
- Supports standards including ASAP2, AUTOSAR, DO-178, IEC 61508, ISO 26262, and MISRA C® in Simulink

Code Metrics

Generated code is smaller than production hand code.



SAE TECHNICAL PAPER SERIES

2004-01-0269

Multi-Target Modelling for Embedded Software Development for Automotive Applications

Grantley Hodge, Jian Ye

Reprinted From: In-Vehicle Networks and Software, Electrical

and Electronics and Sy

SAE Interna

2004 SAE

400 Commonwealth Drive, Warrendale, PA 15096-0001 U.S.A. Tel: (724) 776-4841 Fax: (724) 776-5

Table 2 shows ROM and RAM comparisons between hand code and auto code for a floating-point component in some typical powertrain software.

Table 2 ROM and RAM comparison between a floating-point hand code and auto code.

	Hand Code	Auto Code
ROM	6408	6192
RAM	132	112

The auto code has less size of ROM and RAM compared to that of hand code. The auto code is readable and peer reviewed, and checked with the QAC static analysis tool. Most importantly, the auto code is implemented in a real-world powertrain application.

CONCLUSION

A custom data class allowing data type and data scaling information to be incorporated into the model is

Jian Ye, jye5@visteon.com
Walt Stuart, wstuart@visteon.c
Grantley Hodge, ghodge@viste

DEFINITIONS, ACRONYMS

MBDG: Model-Based Develo Powertrain.

Multi-target Model (Generic contains no data type inform precision).

Target Specific Model: A ge specific data dictionary loade linked to the model.

ROM: Read Only Memory.

RAM: Random Access Memory

KAM: Keep Alive Memory.

MATLAB®: A modeling enviro

SAE Technical Paper 2004-01-0269, March 2004

www.mathworks.com/mason/tag/proxy.html?dataid=4361&fileid=20307

Code Speed

THALES

Software Development with Real-Time Workshop Embedded Coder

Missile Electronics

PS

- Similar pilot study evaluating Model-Based Design carried out at a Thales sister company in Belfast
  - Automatically generated fixed point code ran 30% faster than the hand written fixed point code

Where did we use Model-Based Design

- Two projects used MBD
  - P1: Data processing for a single channel pulsed proximity sensor + timing algorithm
    - TME designed custom hardware for TDP
    - Software developed for 2 x dual-core 16-bit fixed-point DSPs
      - Serial and parallel I/O required with DMA
      - FPGA + analogue front-end
  - P2: Control algorithms for a gimbal assembly with mounted pulsed laser and PIR dual mode sensing
    - COTS hardware with 4 x floating-point DSPs
    - Single DSP used to run model
      - Parallel I/O
      - FPGA – gateway to rest of the system
    - Vendor board support library

Generated code is faster than hand code.

MADC :Thales Dual Core Code Generation, March 2008

http://www.mathworks.com/mason/tag/proxy.html?dataid=10532&fileid=48884

Airbus Helicopters Accelerates Development of DO-178B Certified Software with Model-Based Design

"We use our system design model in Simulink for ARP4754 to establish stable, objective requirements. We save time by using the model as the basis for our software design model for DO-178—from which we generate flight code—and reusing validation tests for software verification."

— *Ronald Blanrue, Airbus Helicopters*

The Airbus Helicopters EC130 helicopter.

Products Used

Simulink
Request trial | Get pricing | Contact sales

Embedded Coder
Request trial | Get pricing | Contact sales

Simulink Check
Request trial | Get pricing | Contact sales

Simulink Coverage
Request trial | Get pricing | Contact sales

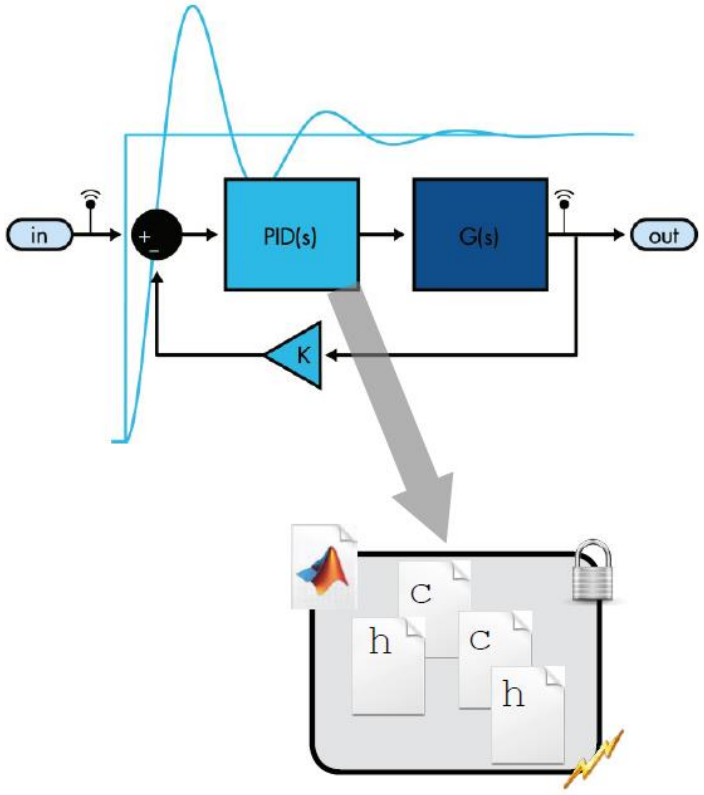
https://www.mathworks.com/company/user_stories/airbus-helicopters-accelerates-development-of-do-178b-certified-software-with-model-based-design.html

Code Verification

After you complete your detailed design, you generate production code. You should test the generated code with your environment or plant model to verify a successful conversion of the model to code.

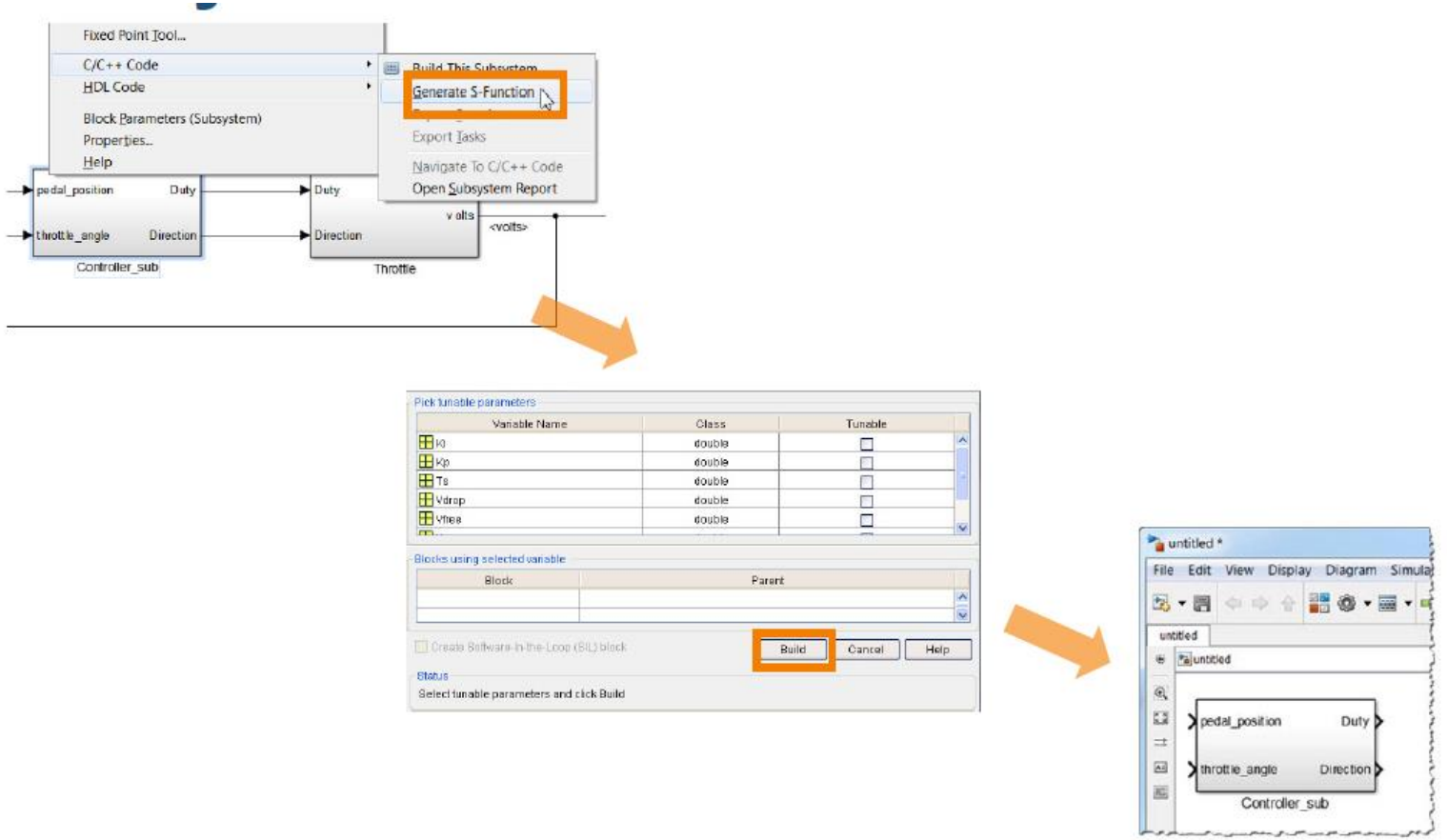
S-Function

- Create a reusable component. With proper parameterization, you can reuse the generated algorithm even if the model is using multiple instances of the S-function.
- Enhance simulation performance. An S-function typically performs more efficiently compared to the original block diagram construct.
- Protect intellectual property. By generating an S-function, you can share the component without having to provide the original construct or the generated source code. Providing the MEX-file object is sufficient.

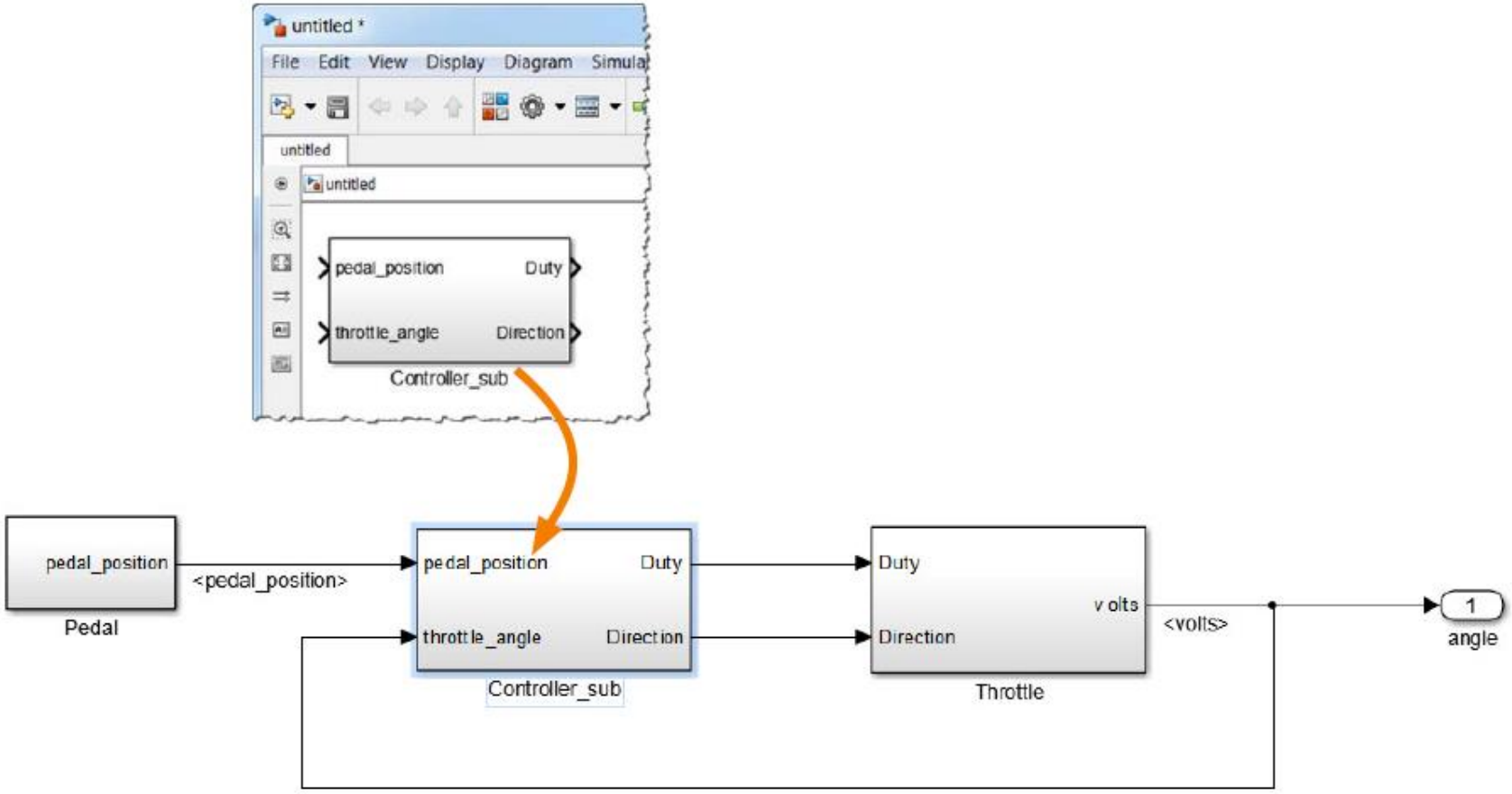


Generating an S-Function from a Subsystem

S-function wrapper. An S-function wrapper is an S-Function block that calls your generated C or C++ code from within a Simulink model.

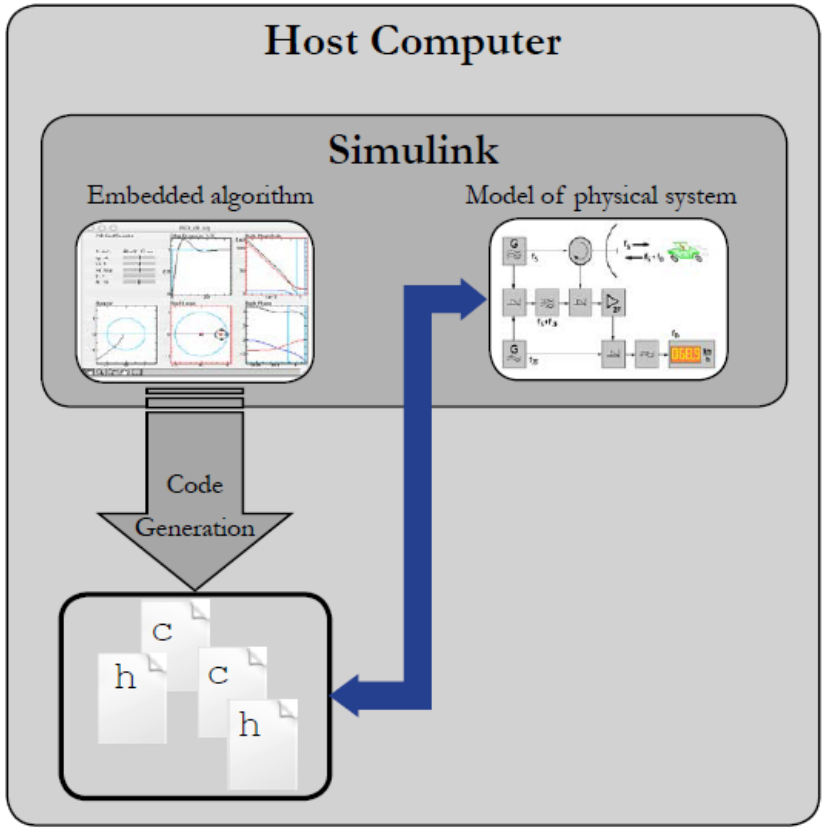


Using a Generated an S-Function from a Subsystem

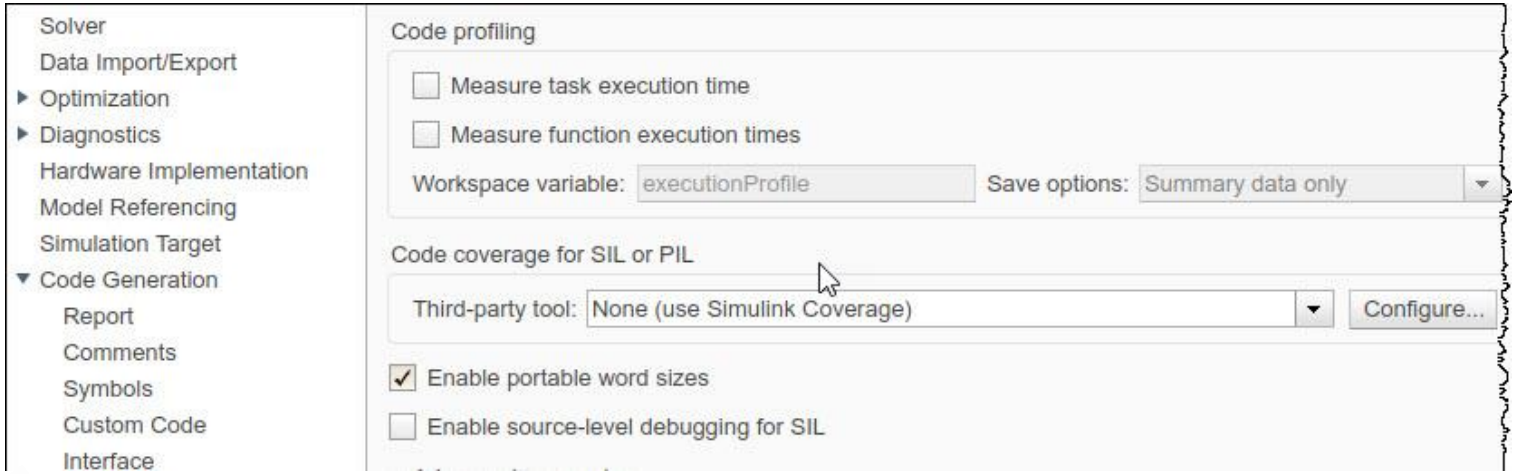


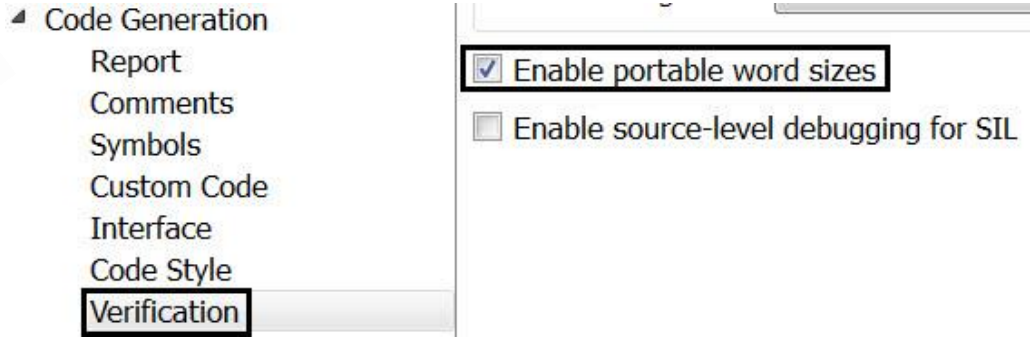
Software-in-the-loop (SIL) Verification

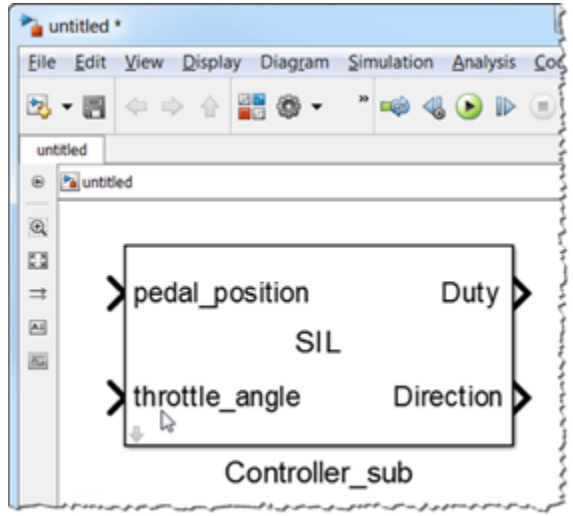
A SIL simulation involves compiling and running production **source code on your host computer to verify the source code**. SIL provides a convenient alternative to processor-in-the-loop (PIL) simulation as no target hardware (for example, an evaluation board or instruction set simulator) is required.

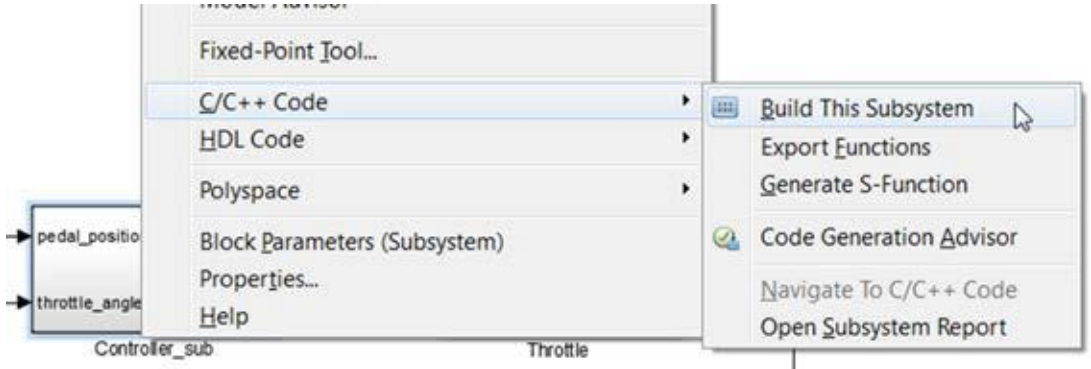


Generating a SIL Block from a Subsystem

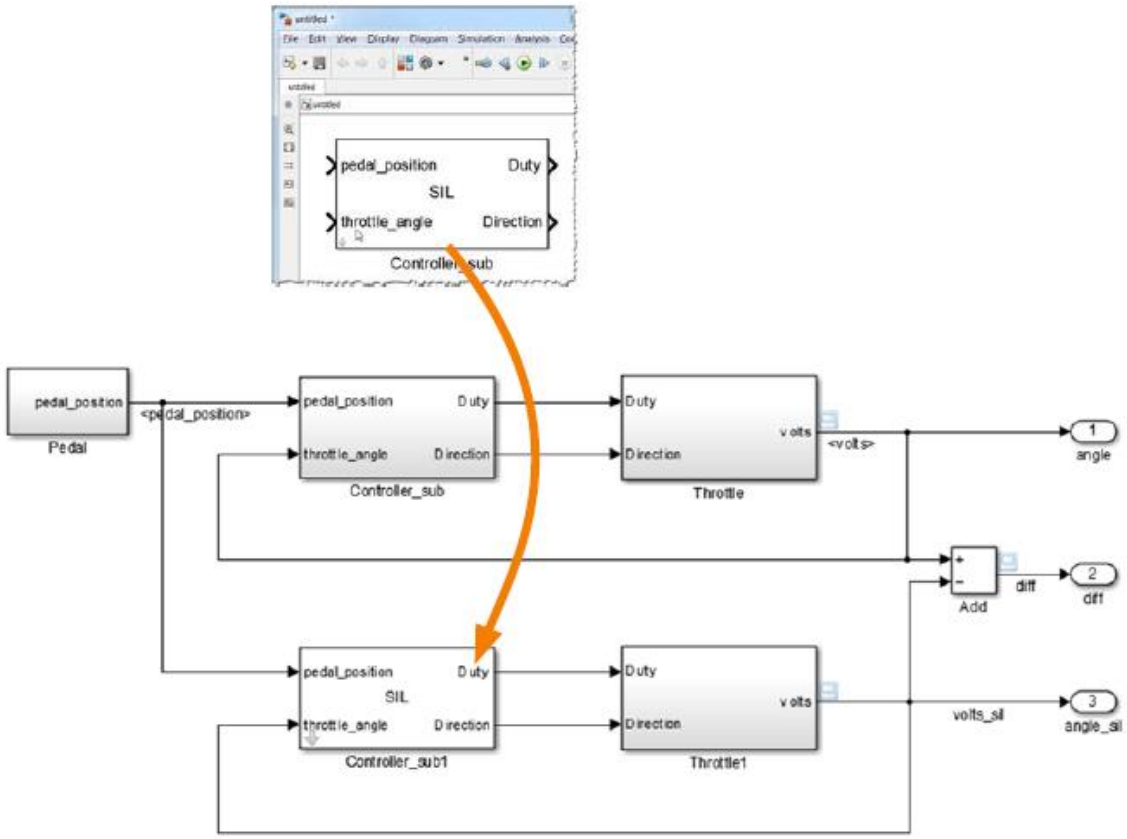


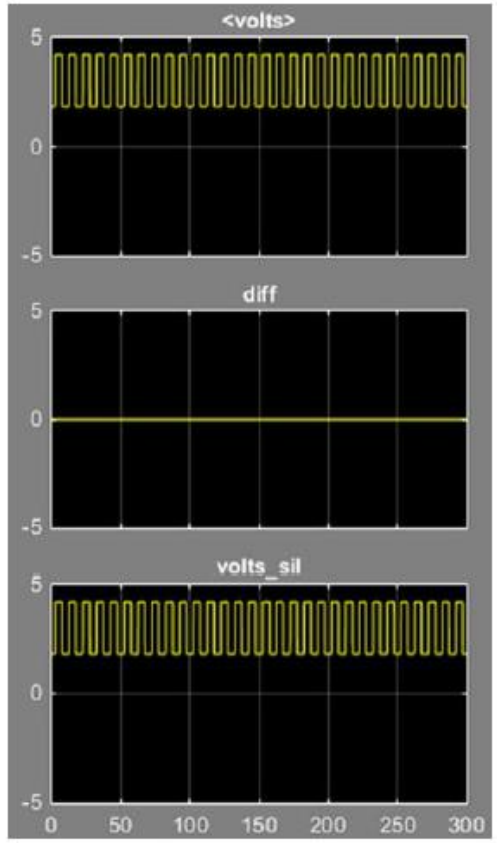






Verifying Subsystem Code with SIL Block

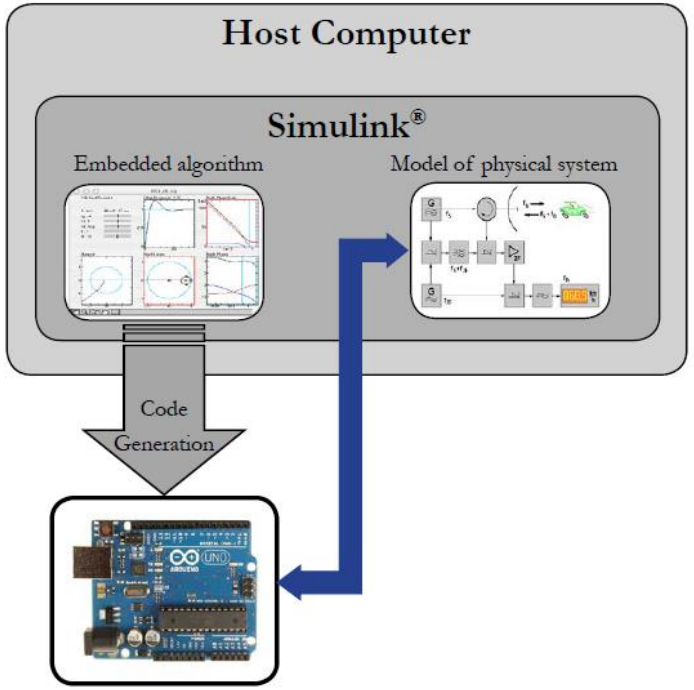




Processor-in-the-loop (PIL) Verification

After you generate the production code for a component design, you need to integrate, compile, link, and deploy the code as a complete application on the embedded system. One approach for verifying the generated code is to use processor-in-the-loop (PIL) verification.

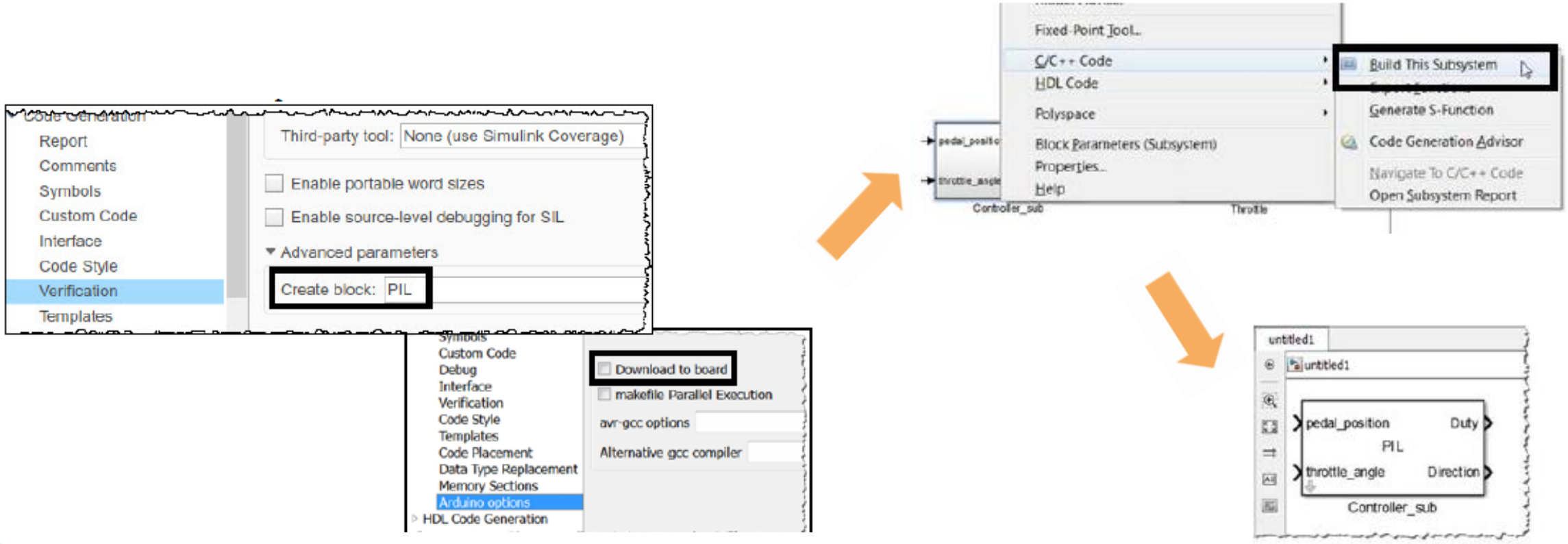
PIL is a technique designed to help you evaluate the behavior of a candidate algorithm on the target processor (or an instruction set simulator) selected for the application. You can compare the output of regular simulation mode and PIL simulation mode to check that your generated code performs as intended.



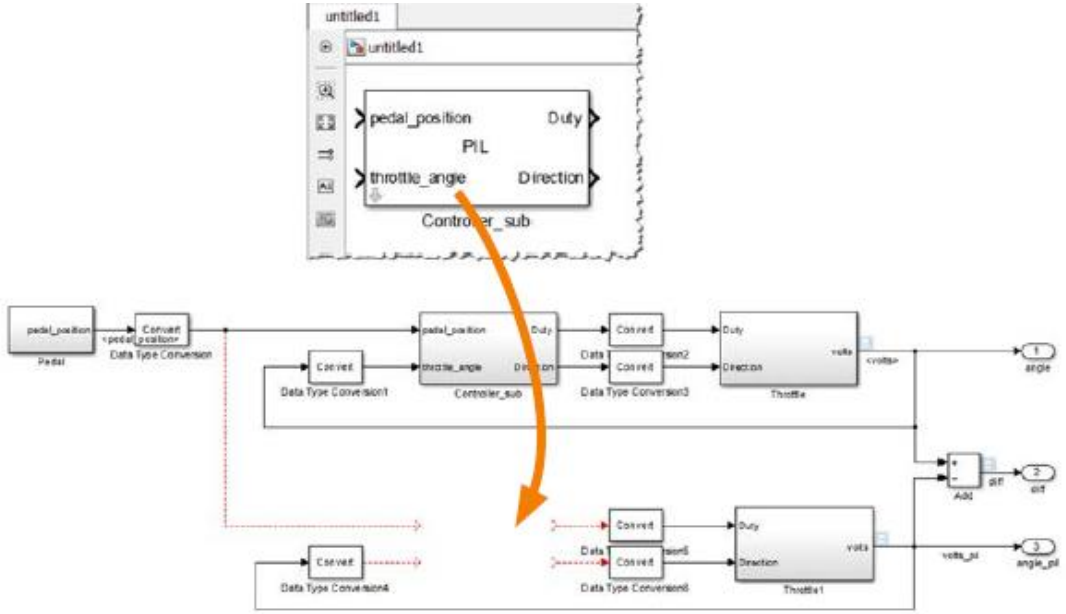
[Code Verification and Validation with PIL and External Mode]

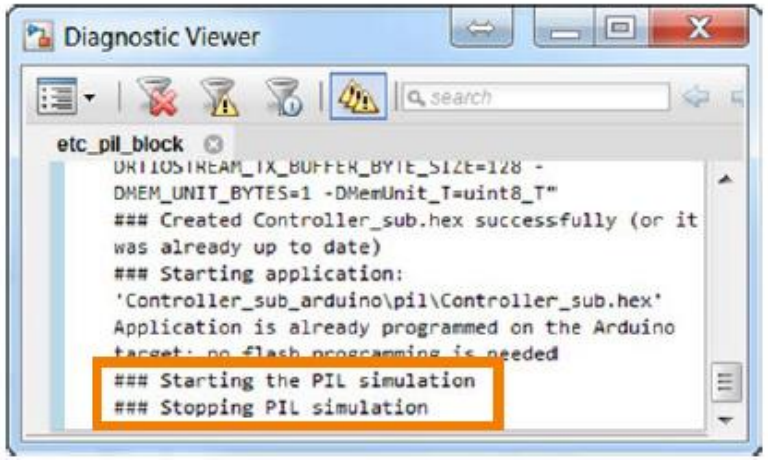
Generating a PIL Block from a Subsystem

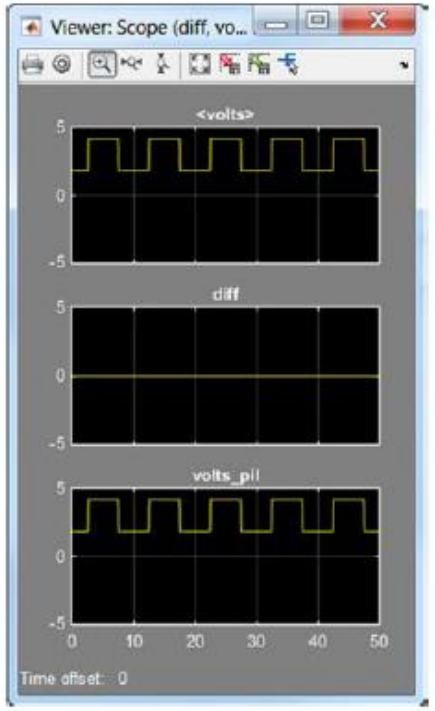
The PIL block workflow is very similar to the SIL block workflow. When you run the model, the subsystem code will run on the embedded processor and an S-function wrapper will communicate data between the PIL block and the rest of the model in Simulink.



Verifying Subsystem Code with PIL Block







Code Report / Traceability