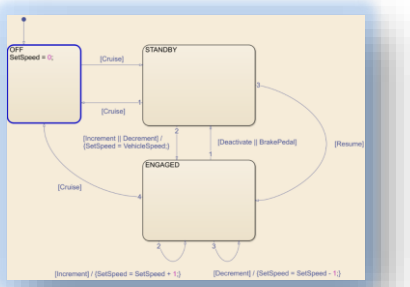
From Models to Embedded Code

Morgan Fremovici – *Application Engineer*
mfremovi@mathworks.com

Model-Based Design Workflow



Component and system testing

Equivalence checking

Equivalence testing

Review and static analysis

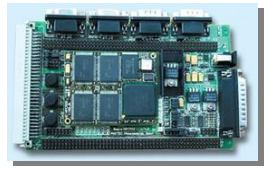
Static Code Analysis

Textual Requirements

Executable Specification

Model used for production code generation

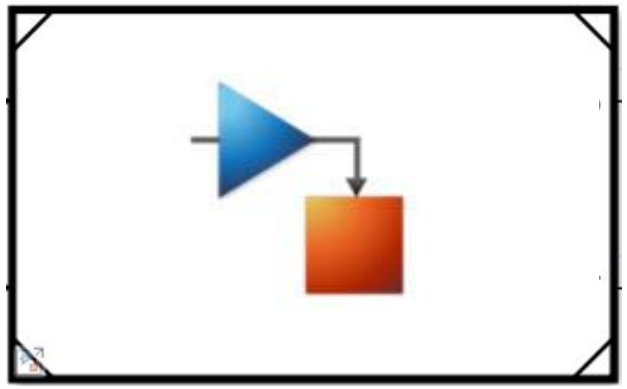
Generated code



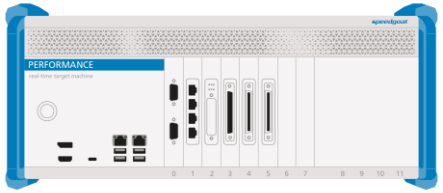
Modelling

Code Generation

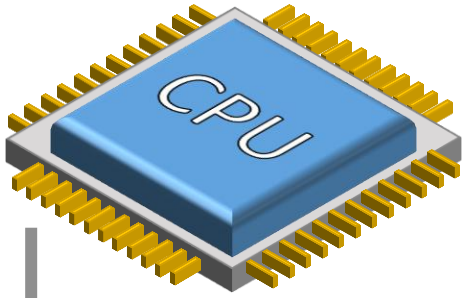
Overall picture



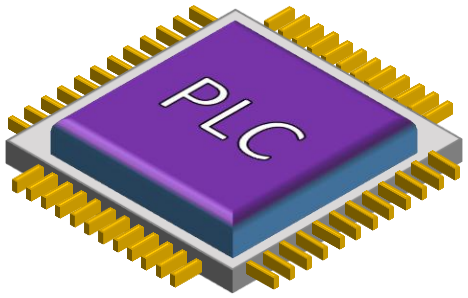
Simulink Coder



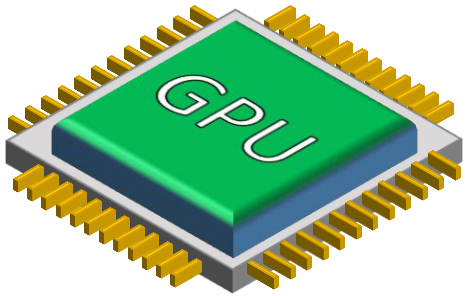
Embedded Coder



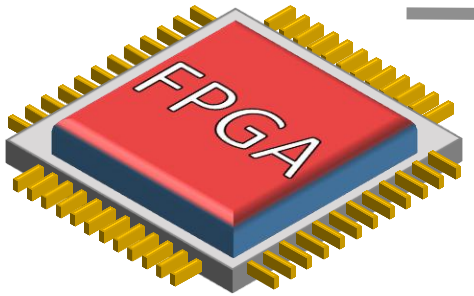
PLC coder

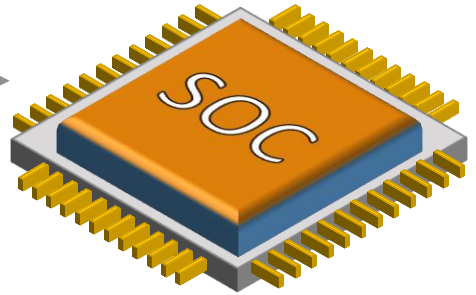


GPU coder



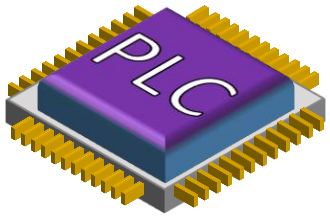
HDL Coder





Soc Blockset

Model-Based Design of Magnetic Levitation Guide Control System for Ultraprecision Machining



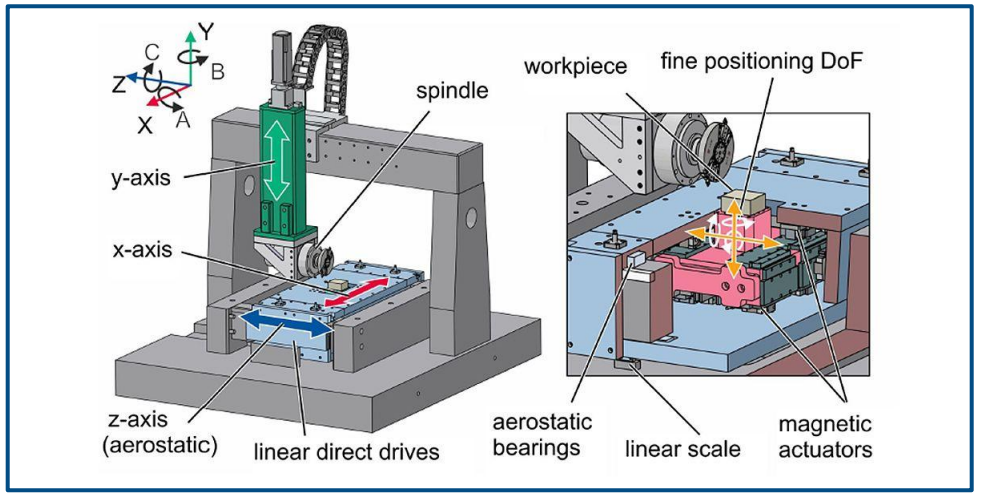
Researchers at Leibniz University Hannover used Model-Based Design to build a high-precision machining prototype that uses electromagnetic levitation guides.

Results/Key Outcomes

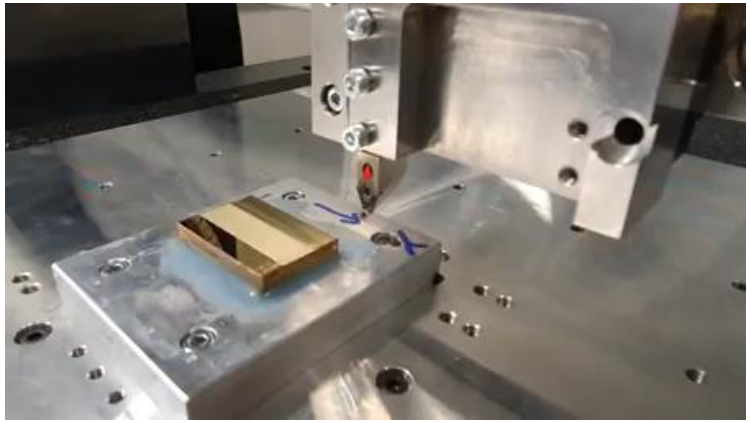
- Model-Based Design enabled modeling and simulation of a high-precision machining prototype with extra degrees of freedom and active vibration control
- Simulink PLC Coder was used to streamline development by automating IEC 61131-3 Structured Text generation for a Beckhoff Industrial PC

"Model-Based Design was instrumental in achieving our initial objective: demonstrating a first-of-its-kind, fully functional prototype for ultraprecision machining with magnetic levitation guides."

- Per Schreiber, Institute of Production Engineering and Machine Tools (IFW), Leibniz University Hannover

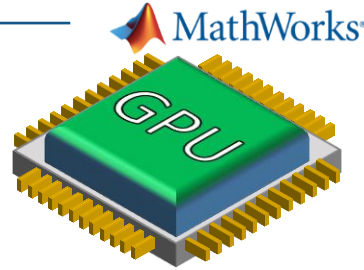


The ultraprecision machining prototype showing larger scale positioning axes on the left (x, y, and z) and small-scale positioning using electromagnetic actuators on the right.



[Link to article]

Drass Develops Deep Learning System for Real-Time Object Detection in Maritime Environments



Challenge

Help ship operators monitor sea environments and detect objects, obstacles, and other ships

Solution

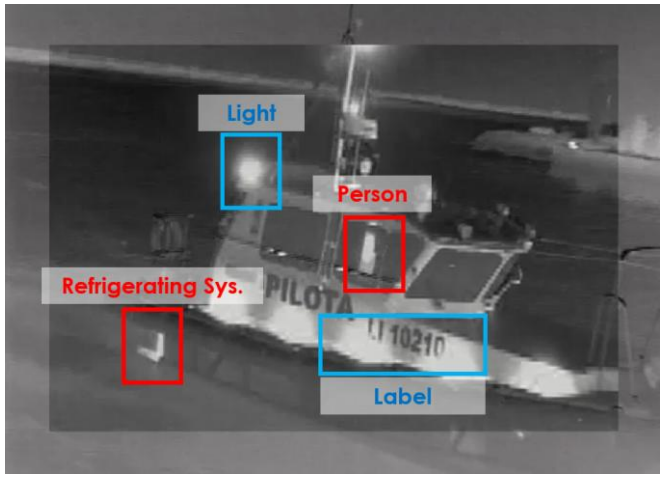
Create an object-detection deep learning model that can be deployed on ships and run in real time

Results

- Data labeling automated
- Development time reduced
- Flexible and reproducible framework established

> *"From data annotation to choosing, training, testing, and fine-tuning our deep learning model, MATLAB had all the tools we needed—and GPU Coder enabled us to rapidly deploy to our NVIDIA GPUs even though we had limited GPU experience."*
>
> *- Valerio Imbriolo, Drass Group*

Object detection tests with optronic system prototype

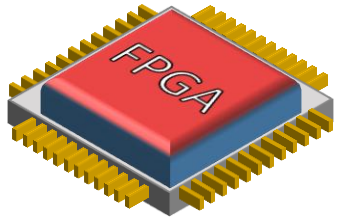


Drass

Maritime object tracking on NVIDIA GPU

Link to user story

KEK Develops Power Converter Control Software for the J-PARC Particle Accelerator with Model-Based Design

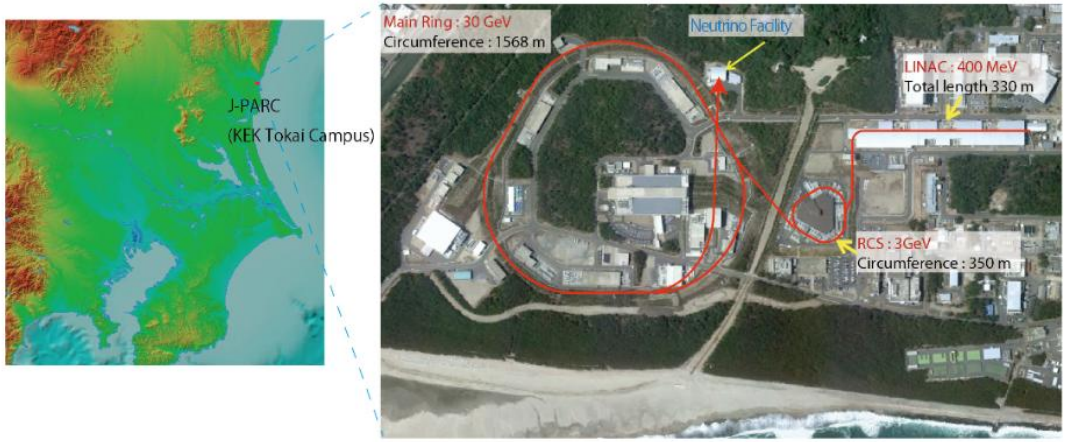


Challenge

Engineer a power converter capable of delivering more than 100 MW to electromagnets used to control proton beams in J-PARC's particle accelerator

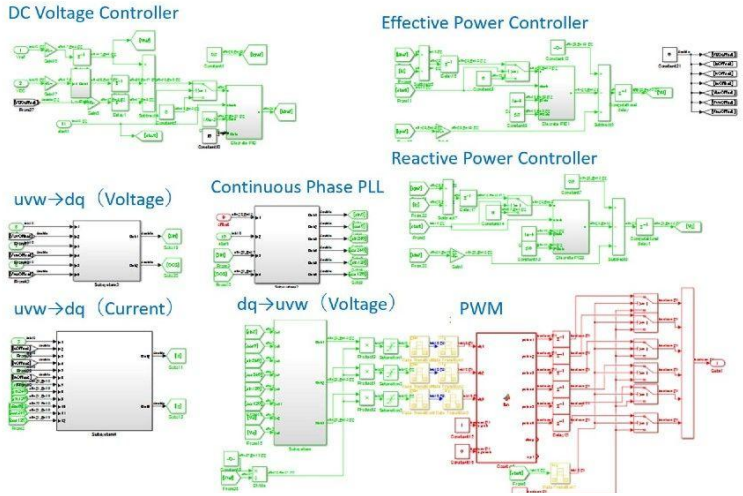
Solution

Use Simulink to model and simulate the power converter control software and plant and generate synthesizable Verilog code from the controller model using HDL Coder



Birds-eye view of J-PARC and its main ring particle accelerator.

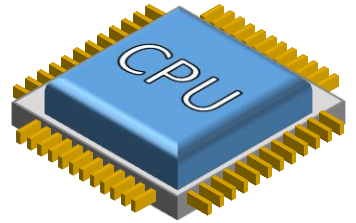
> *"Model-Based Design enabled us to develop the control software at a cost 60% less than the estimates provided by major manufacturers and to cut development time by more than 50%."*
>
> *- Yoshinori Kurimoto,*
> *High Energy Accelerator Research Organization (KEK)*



Simulink model of controller subsystems.

[Link to article]

Dyson Accelerates New Product Development with System-Level Simulation and Code Generation



Model-Based Design enabled faster innovation and reduced time to market for Dyson's first-ever wet floor cleaner, the WashG1.

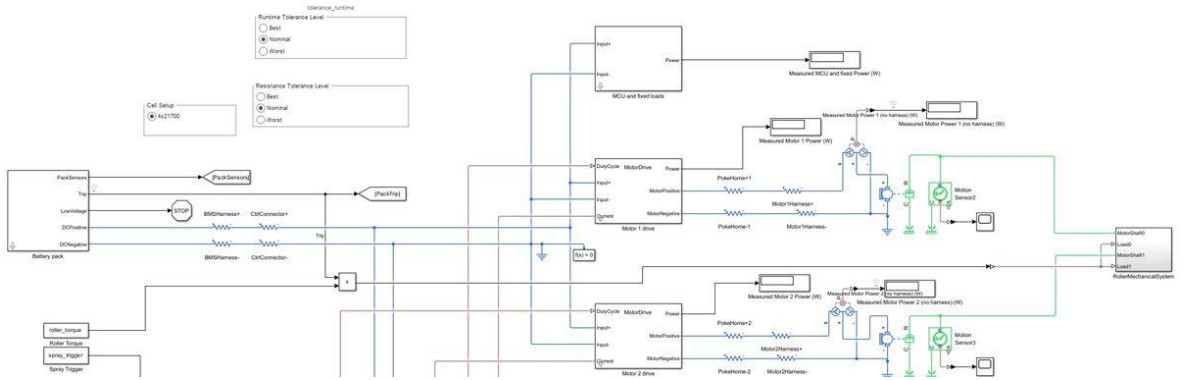
Key Outcomes

- By leveraging Model-Based Design, Dyson accelerated the testing of new design iterations, achieving a 100% increase in speed.
- Using Embedded Coder, Dyson shrunk its software release cycle from 10 weeks to nine days.
- Using Requirements Toolbox, Dyson tested their requirements before the hardware testing phase.

Components of Dyson roller technology.

"We needed to explore many different concepts and directions. Using Model-Based Design and Simulink models gave us the ability to be agile and turn new ideas around twice as fast compared to our document-based development process."

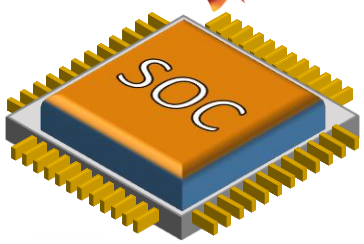
- Romain Guicherd, lead advanced control systems engineer at Dyson

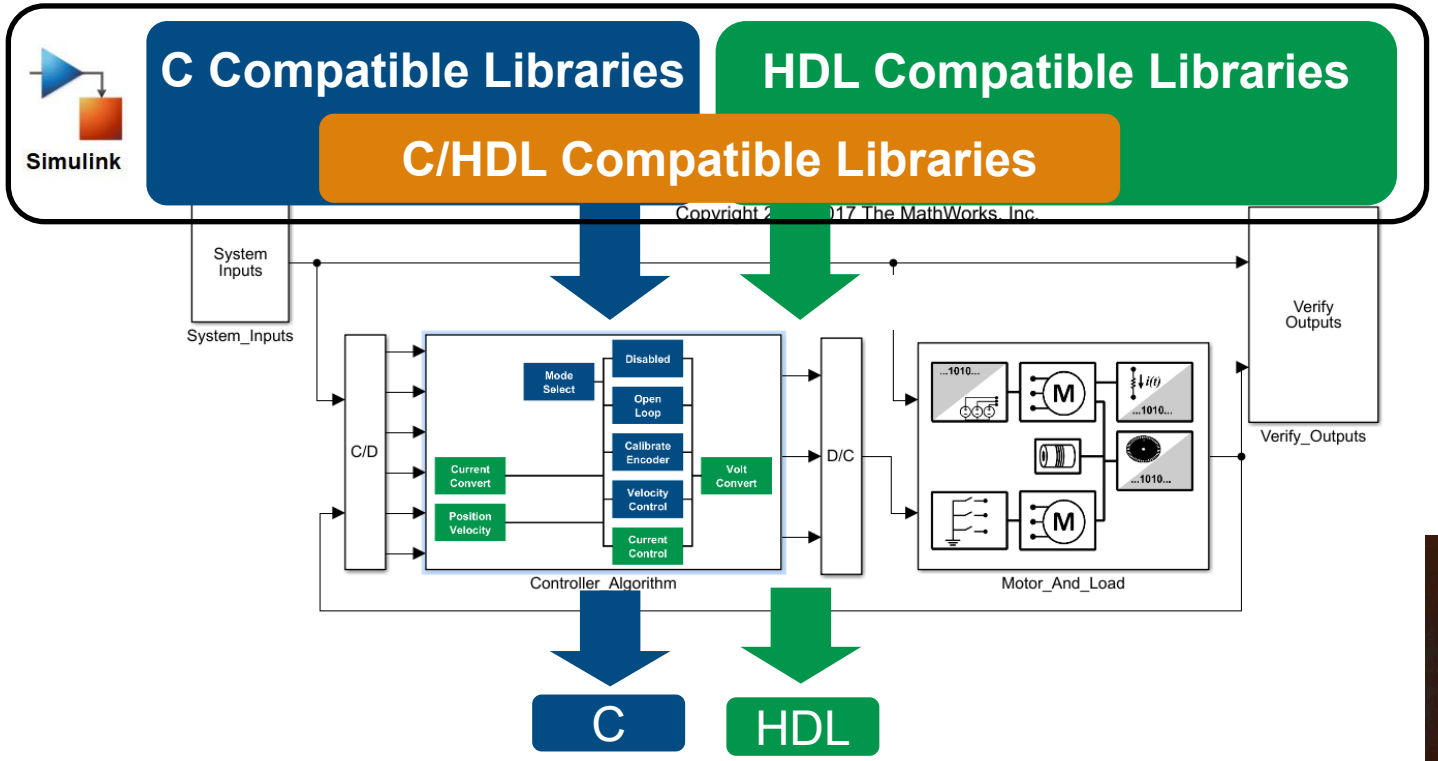


Dyson roller technology modeled in Simscape. (Image credit: Dyson)

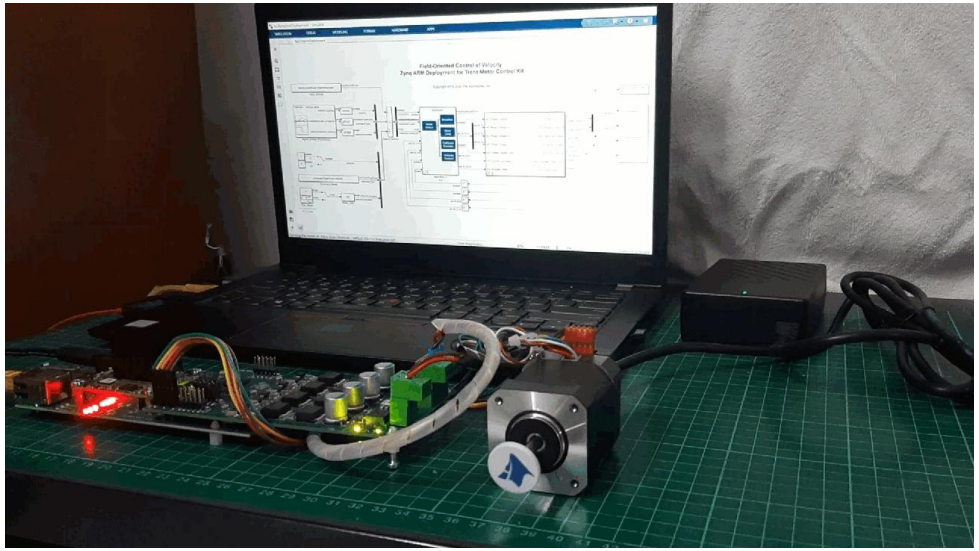
Link to story

Use Simulink to Deploy Motor Control Algorithms on SoC





Trenz Electronic Motor Control Dev Kit

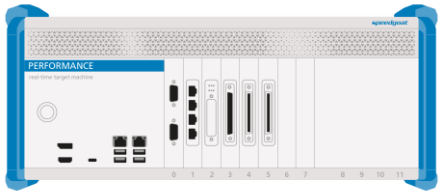


Main application aera

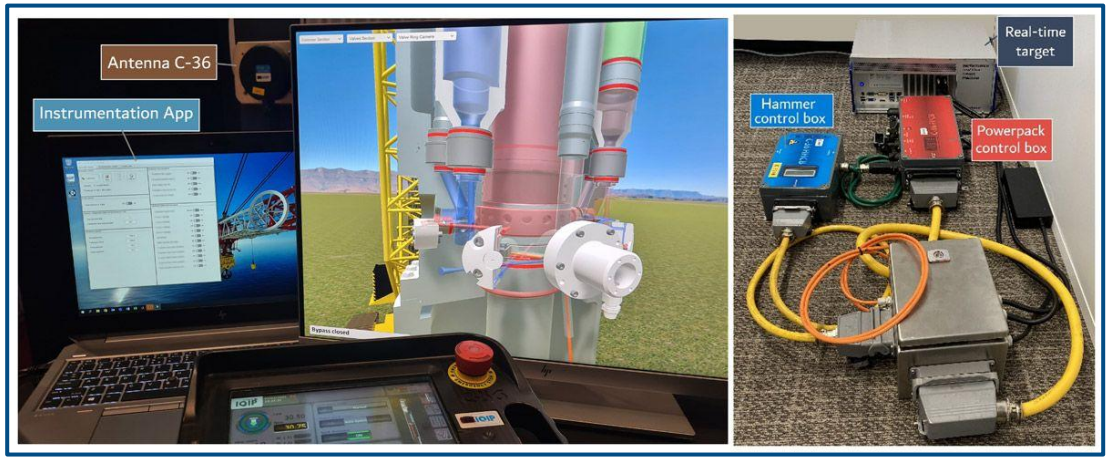
- Wireless Communication
- Radar
- Vision
- Motor Control

openExample('shared_mcb_soc_c2b/ImplementFieldOrientedControlOnFPGASoCExample')

IQIP Uses HIL for Virtual Commissioning of Offshore Machines



Using MATLAB, Simulink, and Simulink Real-Time, IQIP created an HIL setup for virtual commissioning of a pile driver for offshore foundations. The setup can also inject faults impossible to reproduce on real hardware because of the resulting damage, enabling operator training under realistic conditions.



Hardware-in-the-loop setup with 3D visualization (left) and the actual control hardware connected to the real-time target (right).

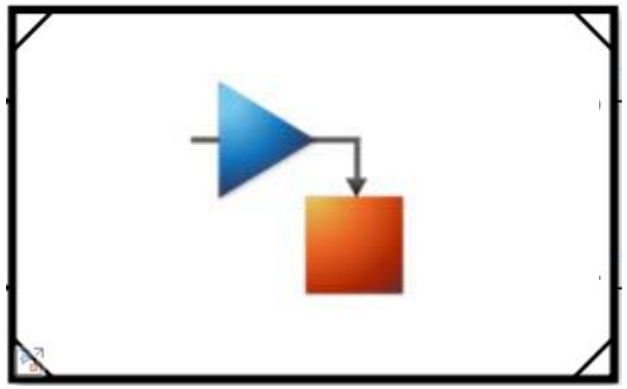
Key Outcomes/Advantages:

- Achieved faster development, integration, and testing of offshore pile driving machinery using virtual commissioning
- Improved product reliability through comprehensive fault injection testing
- HIL setup enabled realistic training environments for operators with fault injection that is impossible to recreate on the real machine

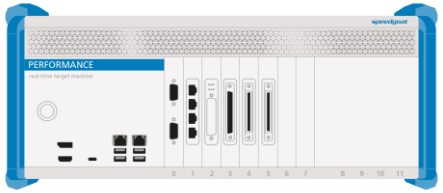
"Without the HIL system, operator training was performed with an actual Hydrohammer, which is costly. With the HIL system, we can now simulate all sorts of faults occurring in reality in the field. On an actual pile driving machine, many of them would lead to damage."
- Michael Schaap, technical director, IQIP

Link to user story

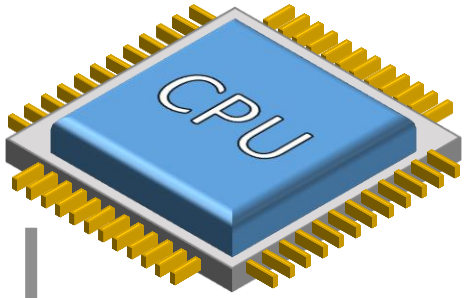
Overall picture



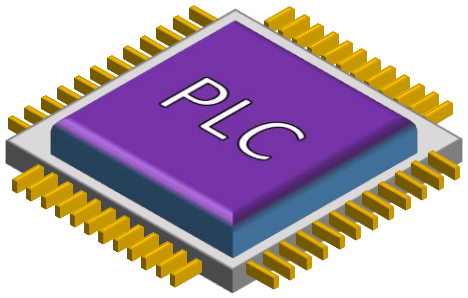
Simulink Coder



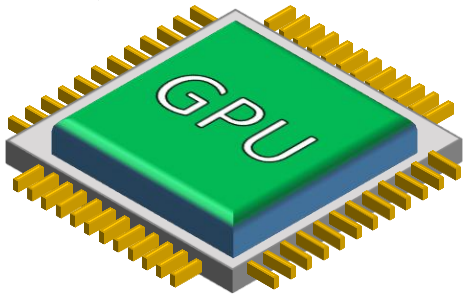
Embedded Coder



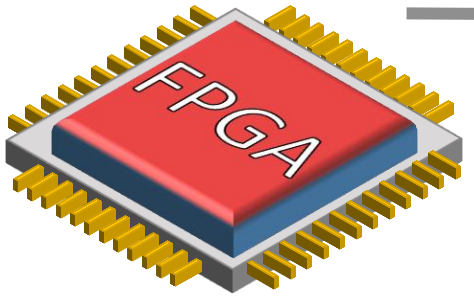
PLC coder

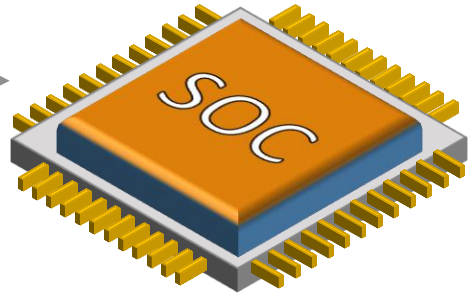


GPU coder



HDL Coder





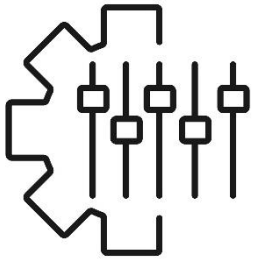
Soc Blockset

Default generated code

- Basic generated functions format:
  - void modelname_initialize(void) : to call at system initialization
  - Void modelname_step(void) : to call each processing step (on timer or interrupt)
  - Void modelname_terminate(void) : to call at system shutdown.

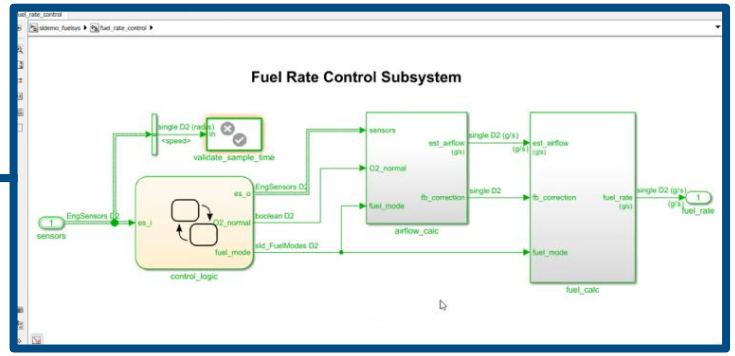
- Code optimization & customization:
  - None or limited

How to customize / optimize my code?



Customization

Integrate Generated Controller Code with Your Hand-Coded Software Project



Communication Interfaces

Comm Drivers

Core Software Algorithms and Logic

Output Drivers

M

Actuators

A B C

Input Drivers

Special Device Drivers

Sensors

Special Interfaces

CCP

ASAP2

Scheduler/Operating System And Support Utilities

Integrate Generated Controller Code with Your Hand-Coded Software Project

Model | **Hand**

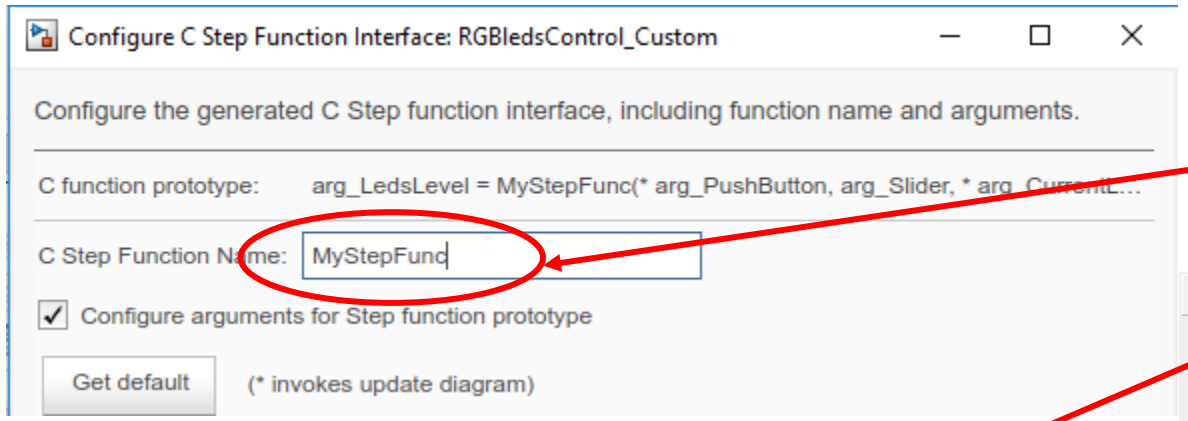
Embedded Software Project Pseudo-Code

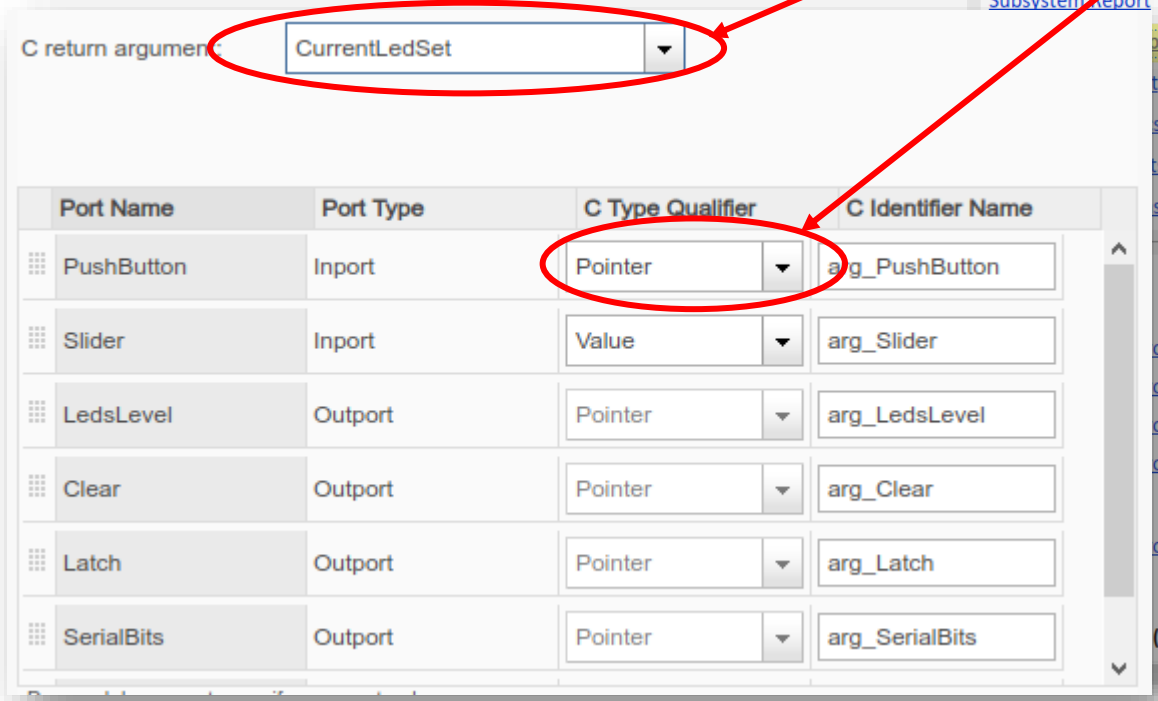
```
main()
{
    adcInit();
    encoderInit();
    pwmInit();

    controllerInit();

    while(1) {
    }
}
```

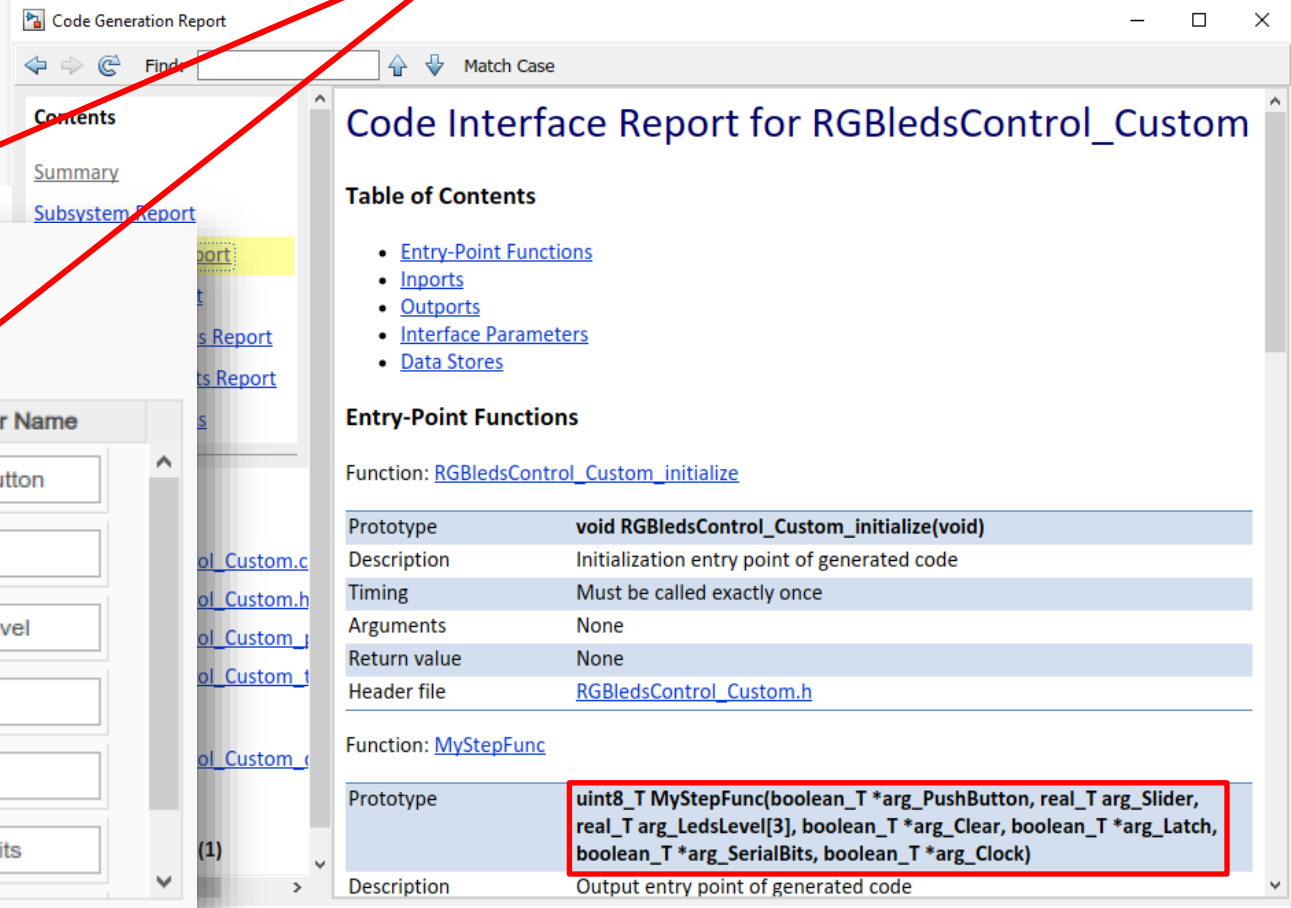
```
interruptServiceRountine()
{
    AdcStruct = readAdcCountFromDriver();
    EncoderStruct = readEncoderCountFromDriver();

    PwmStruct = controllerStep(AdcStruct, EncoderStruct);

    writePwmCountToDriver(PwmStruct);
}
```

Basic generated code architecture & interface

- Customsation
  - **Names**
  - **Arguments Type qualifier**

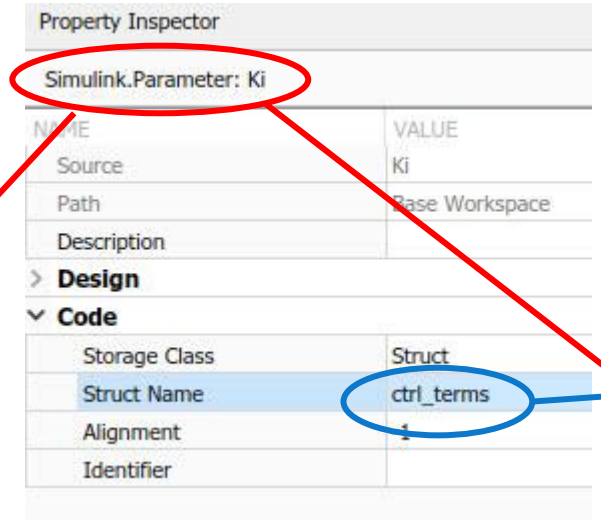


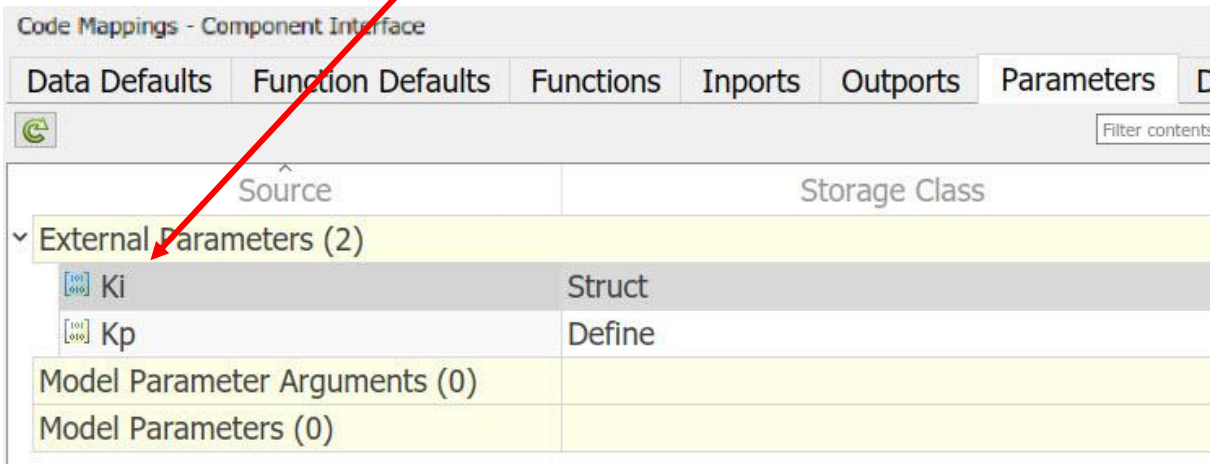


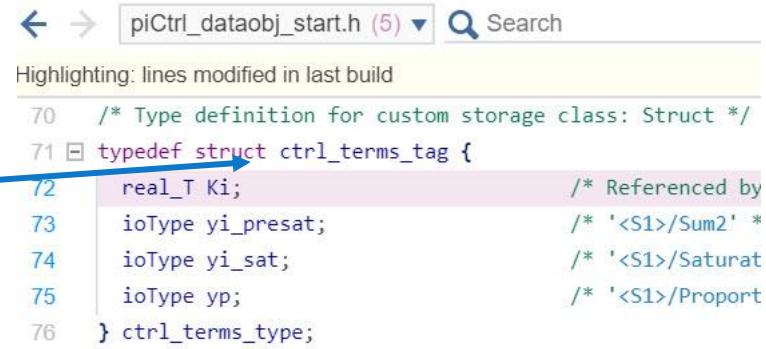


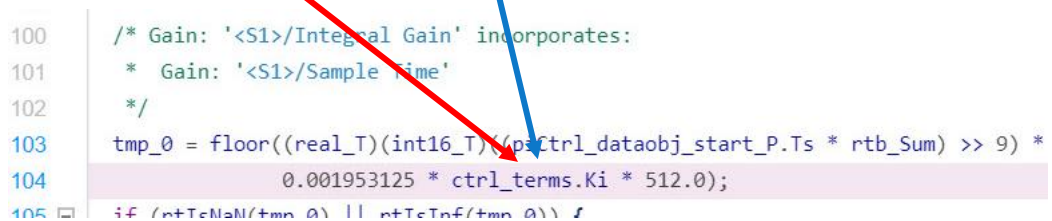
Basic generated code architecture & interface

- Customization
  - **Simulink.Parameter object**
  - **Storage class**



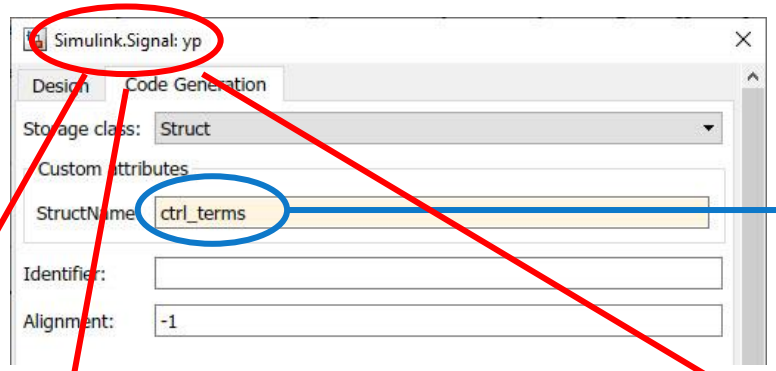


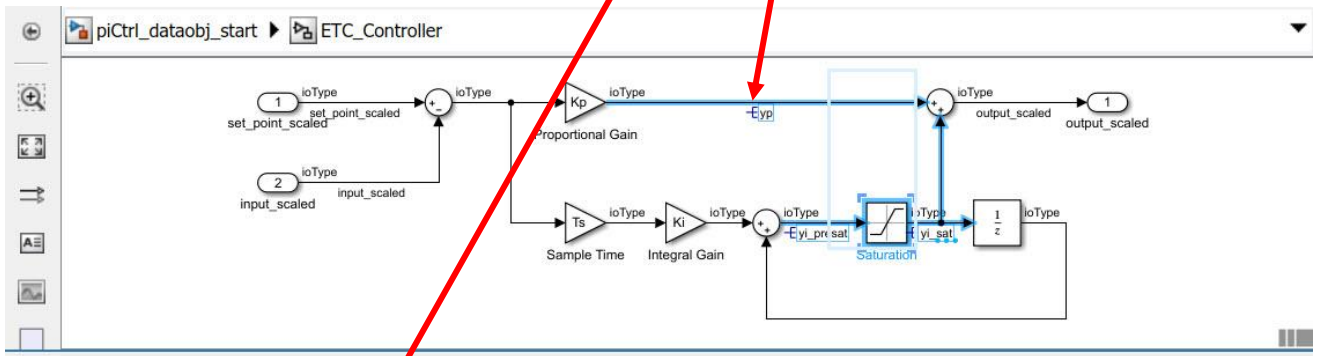


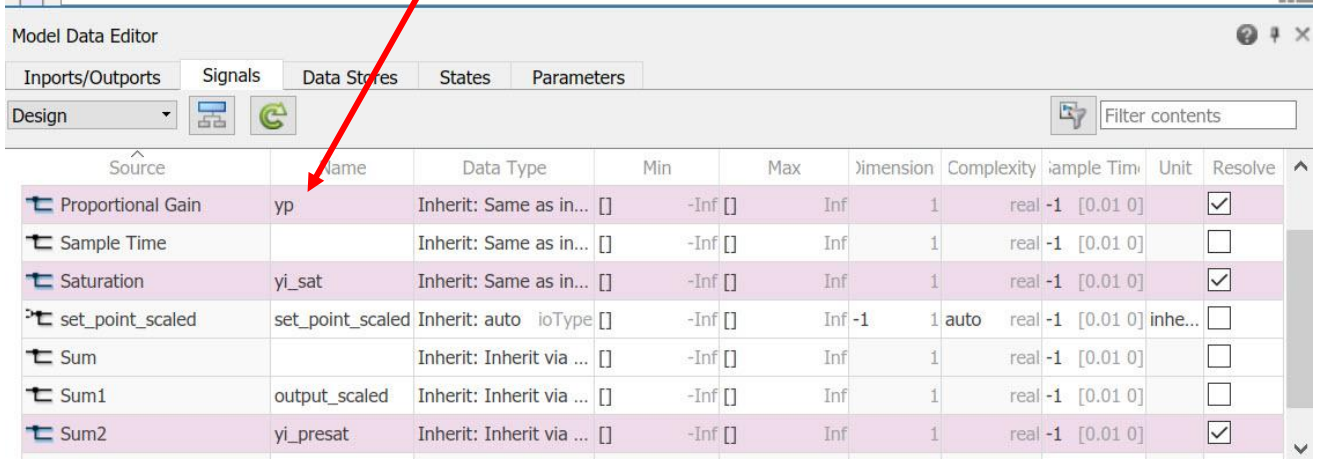


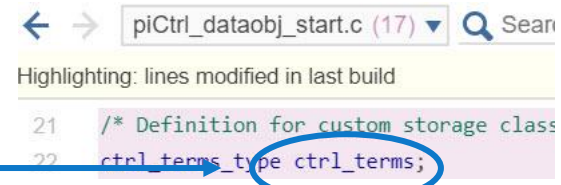
Basic generated code architecture & interface

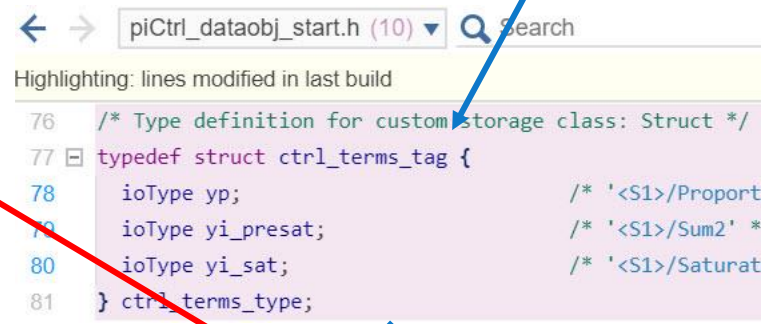
- Customization
  - **Simulink.Signal object**
  - **Storage class**

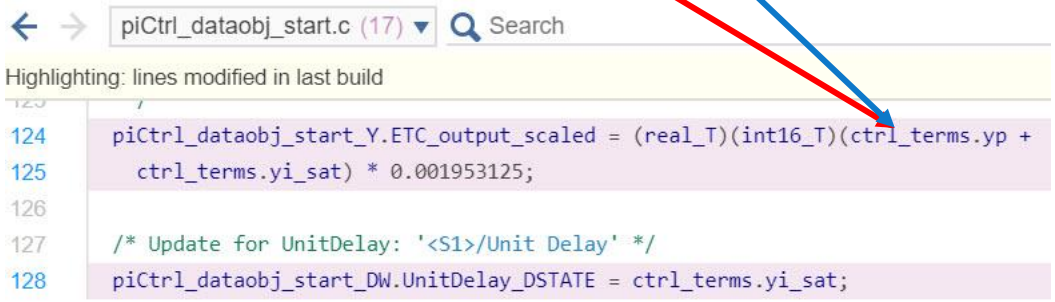










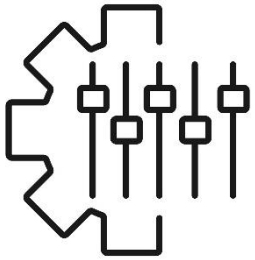


Basic generated code architecture & interface

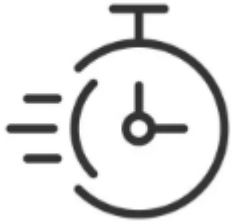
- Basic generated functions format:
  - void modelname_initialize(void) : to call at system initialization
  - Void modelname_step(void) : to call each processing step (on timer or interrupt)
  - Void modelname_terminate(void) : to call at system shutdown.

- Several possible customisation using Embedded coder
  - Functions / files names
  - Function interface (return & argument passing mode).
  - Several step functions for concurrent tasking.
  - …

example : int **MyModel_step**(int inputs, *int params, *int dworks)

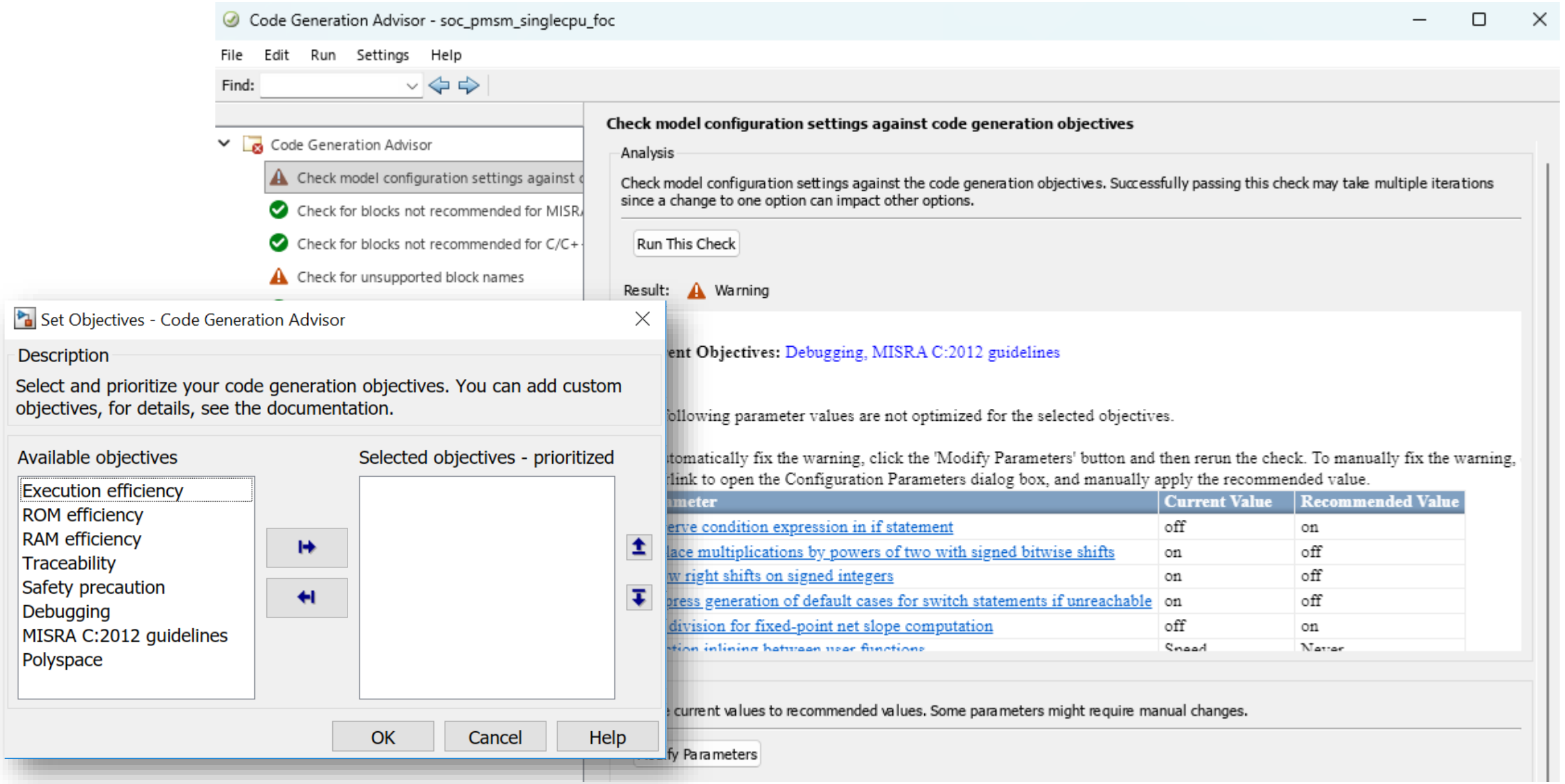


Customization

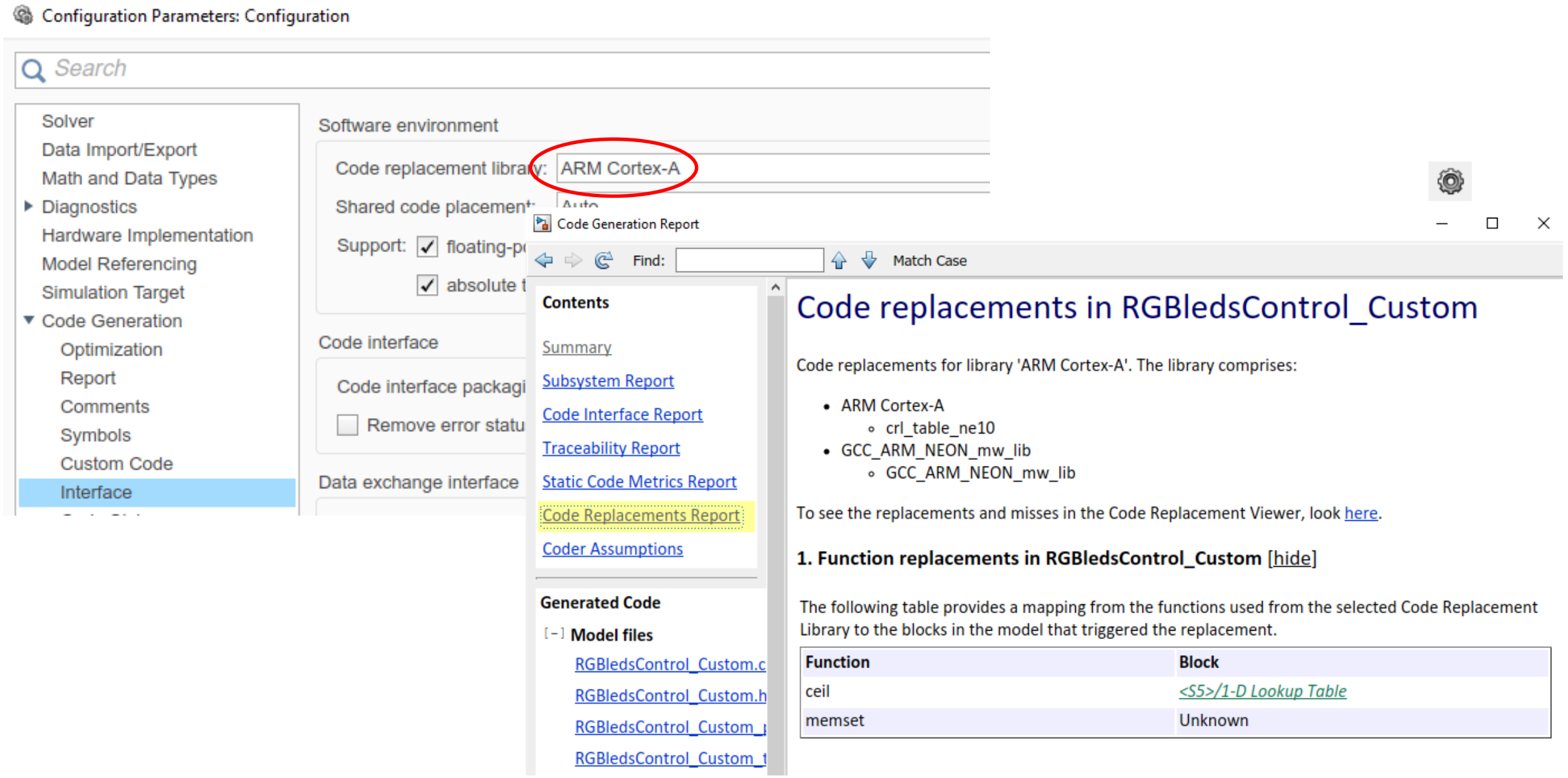


Optimization

Configuration Panel & Code generation objectives

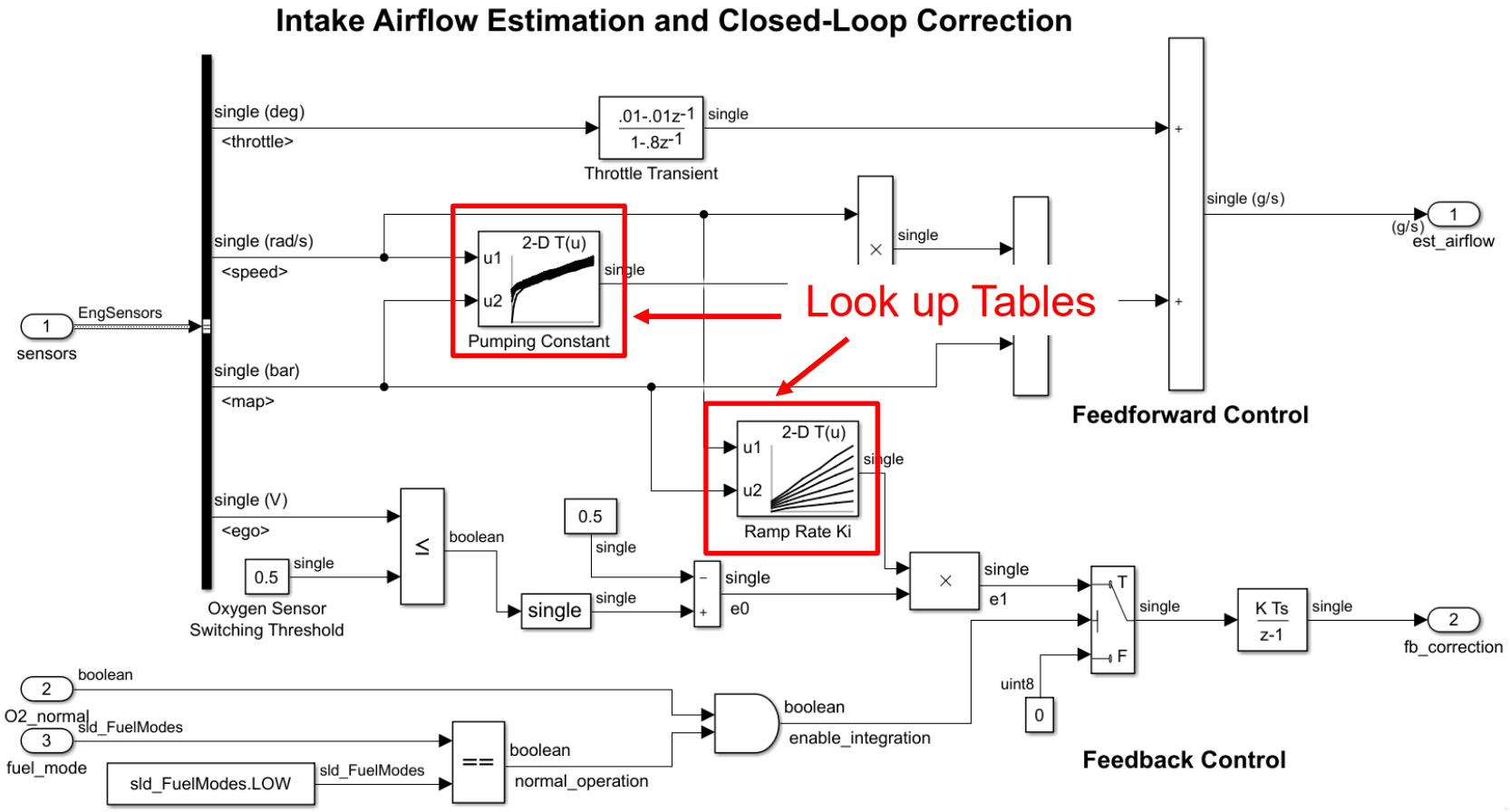


Code Replacement & Legacy Code

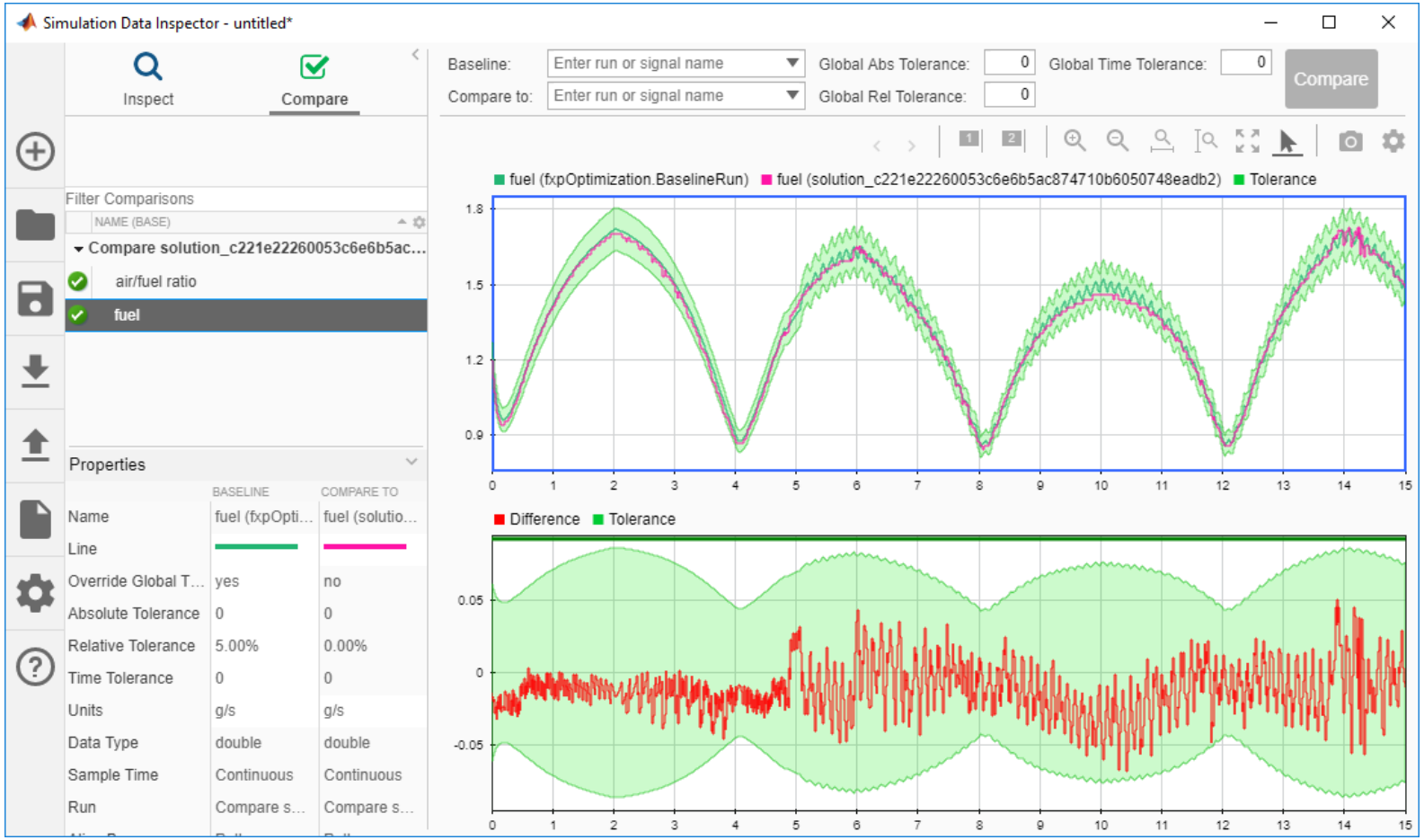


Memory usage reduction & data quantitization

Objective: Code generation for micro-processor ARM 7 minimizing Memory usage



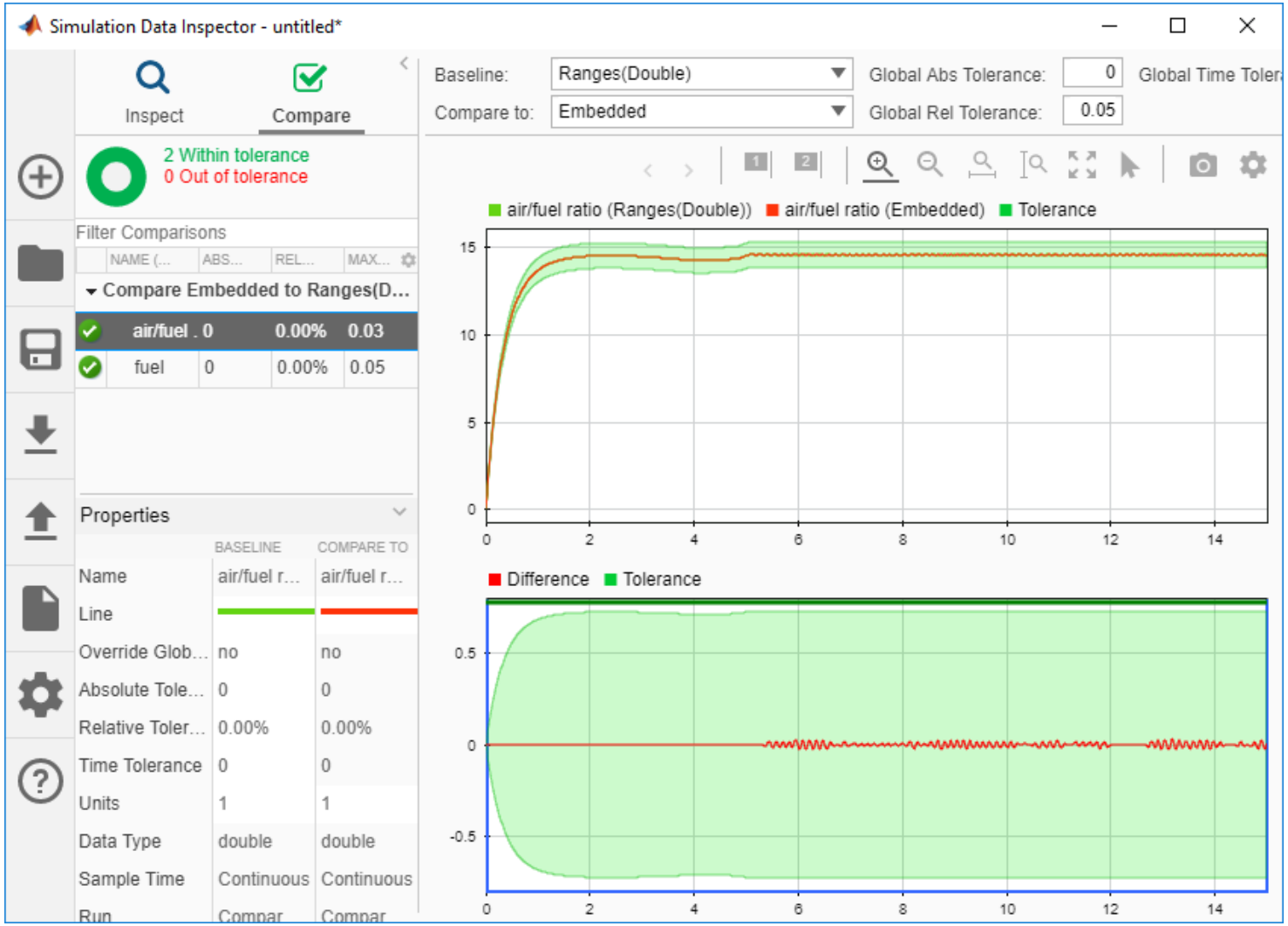
Single Precision Conversion Result



Data Type	Memory Usage (Bytes)
Double	12048
Single	6024

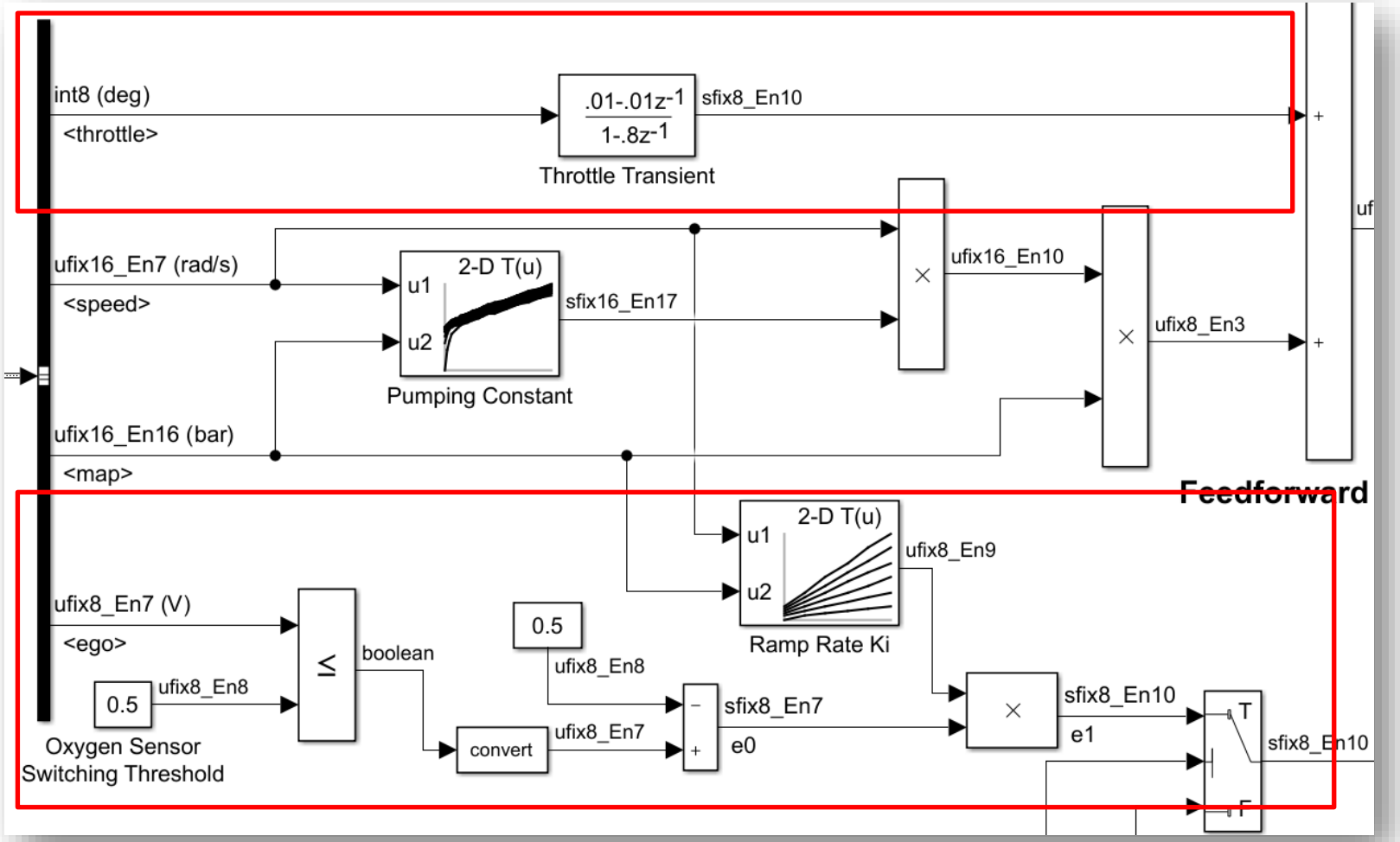
Fixed Point Conversion Result

Data Type	Memory Usage (Bytes)
Double	12048
Single	6024
16 bit Fixed	3012



* Need Fixed Point Designer

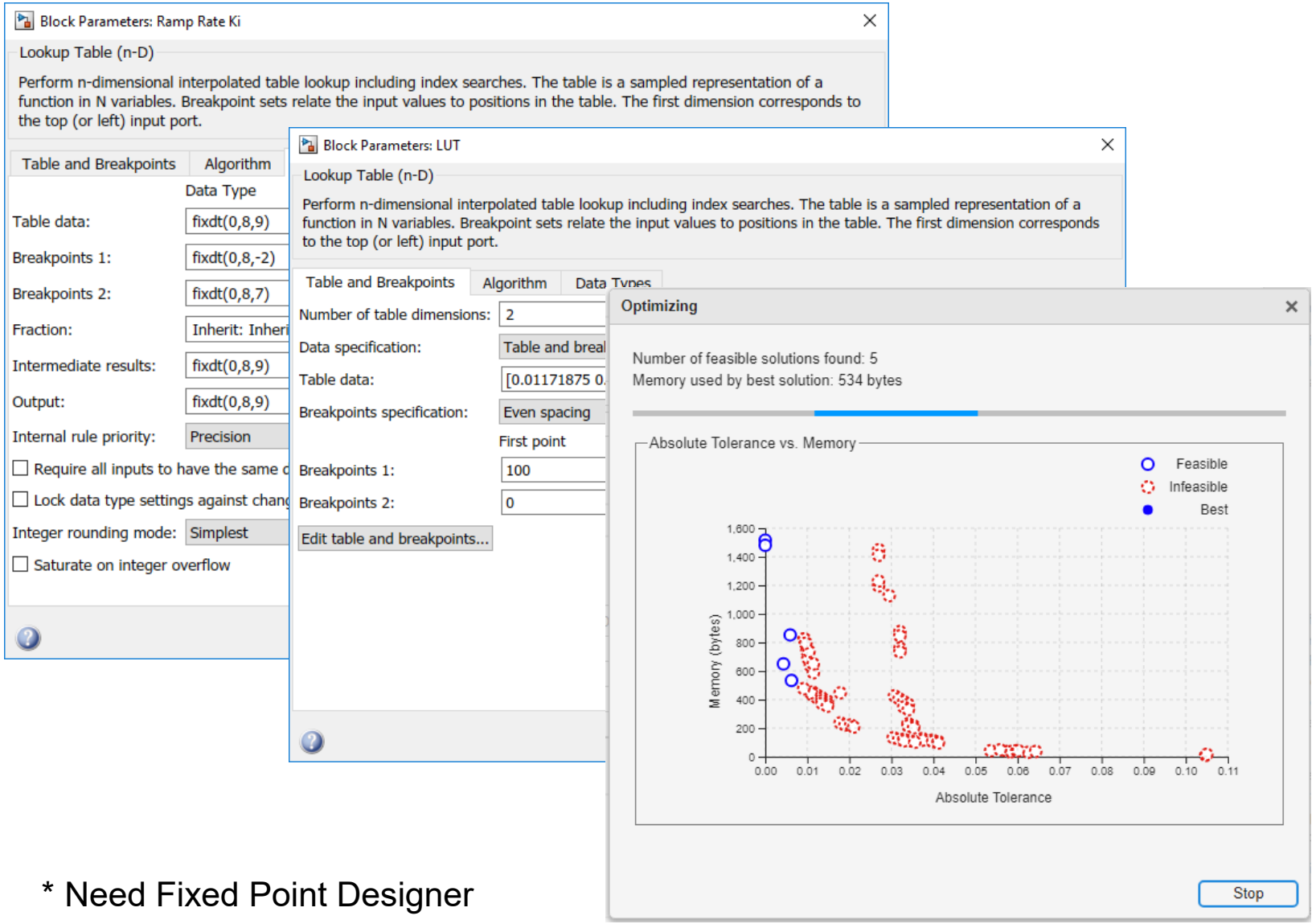
Data Type Optimization Result



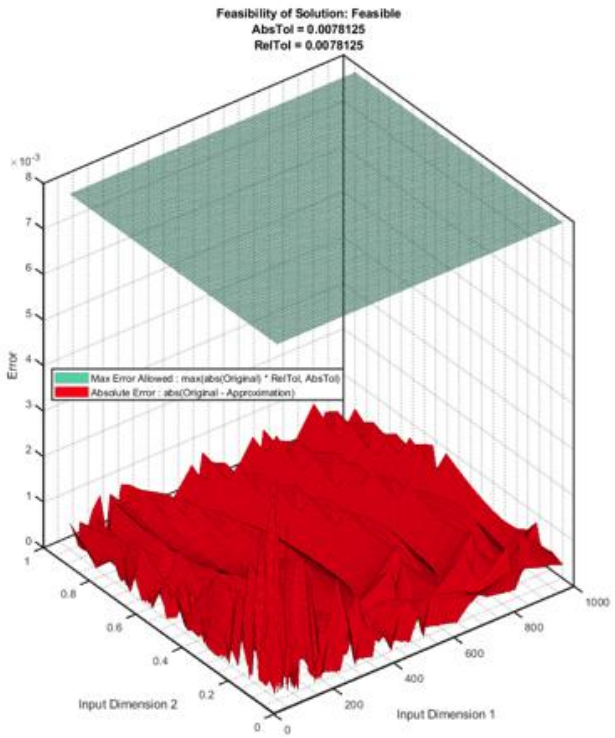
Data Type	Memory Usage (Bytes)
Double	12048
Single	6024
16 bit Fixed	3012
Optimized	2224

* Need Fixed Point Designer

Lookup Table Optimization Result



Data Type	Memory Usage (Bytes)
Double	12048
Single	6024
16 bit Fixed	3012
Optimized	2224
LUTO	411



* Need Fixed Point Designer

Performance of automatic model generation code

Delphi HV motor control software

Task / Module	Throughput (uSec)	
	Model	Hand-Code
Current Magnitude and Phase Process	1.42	1.31
ABC to dq0 Frame Transformation	0.76	0.52
Resolver Harmonic Learn	0.48	0.22
Angle Position Determination	0.93	0.84
PI-Current Regulator	7.62	7.51
Torque Mode	4.82	4.72
dq0 Rotating to Stationary Frame Transformation	0.94	0.82
Complete 100 uSec Task	**65.37**	**63.83**

※ MathWorks Automotive Conference Michigan 2015

Visteon powertrain control software

		Code Size
Hand Code		928
Auto Code	No overflow/underflow check	904
	Check OF/UF everywhere	1562
	Check only where necessary	934

*Based on Tasking Compiler for ST10

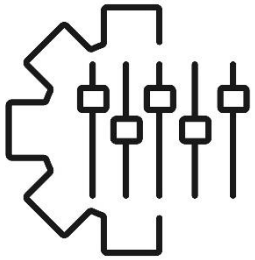
Table 2 ROM and RAM comparison between a floating-point hand code and auto code.

	Hand Code	Auto Code
ROM	6408	6192
RAM	132	112

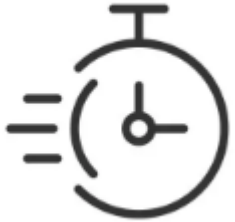
※ SAE Technical Paper 2004-01-0269, March 2004

It is possible to auto-generate C code comparable to hand code!

- A poorly designed model will generate inefficient code
- Highly efficient code generation requires tool knowledge and right modeling pattern and architecture.



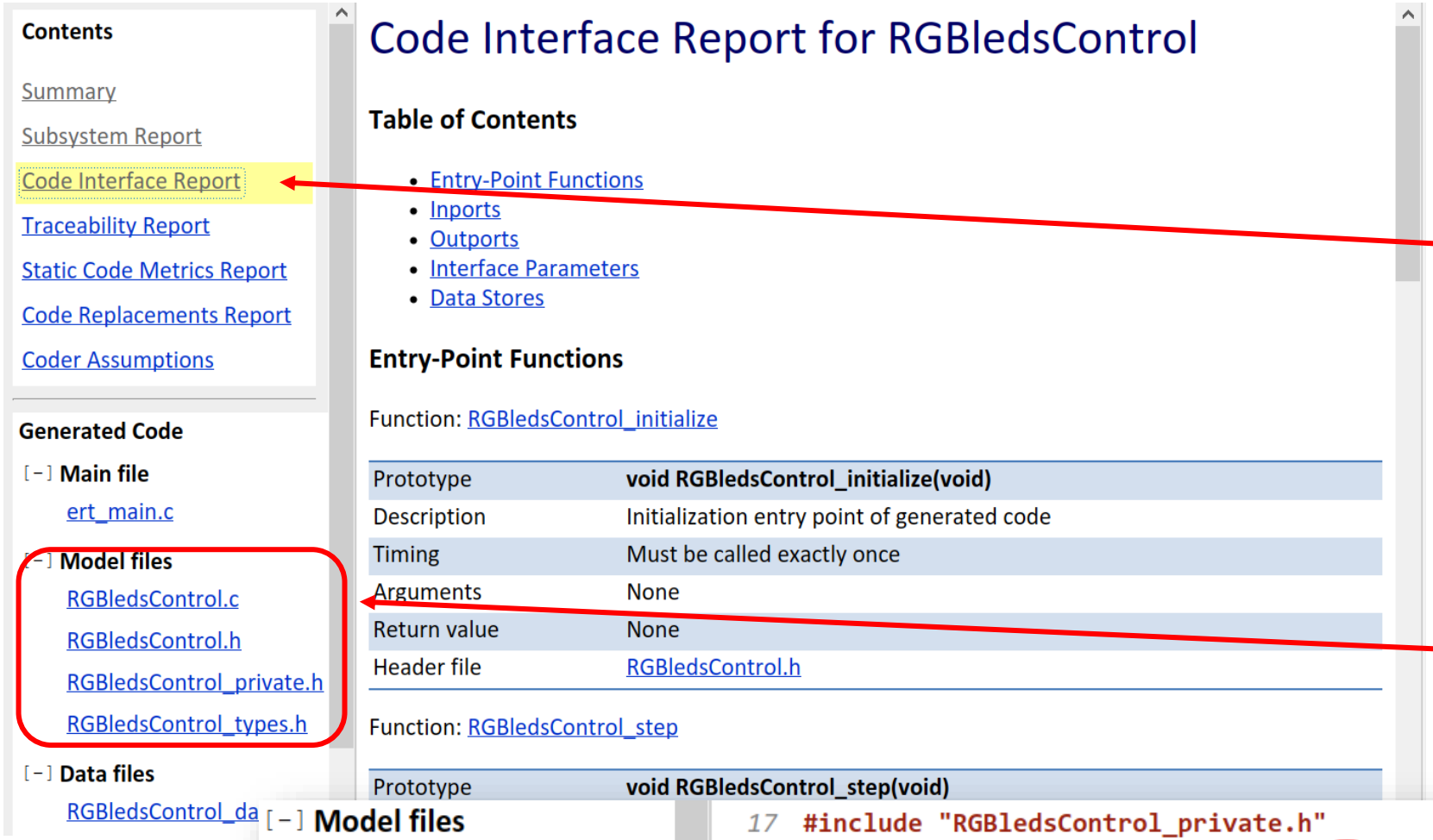
Customization

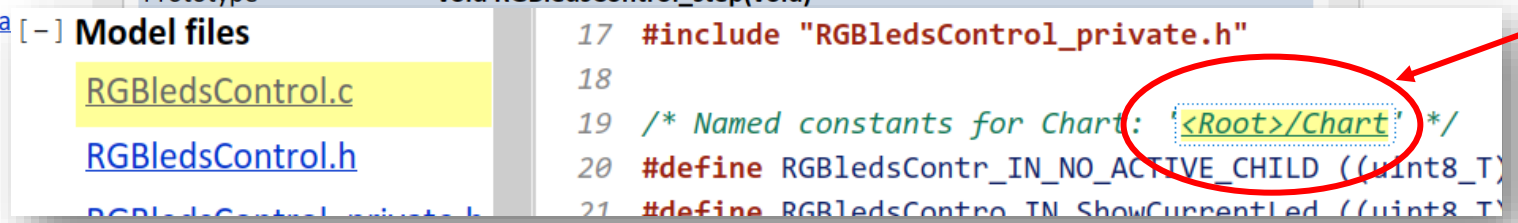


Optimization

Report

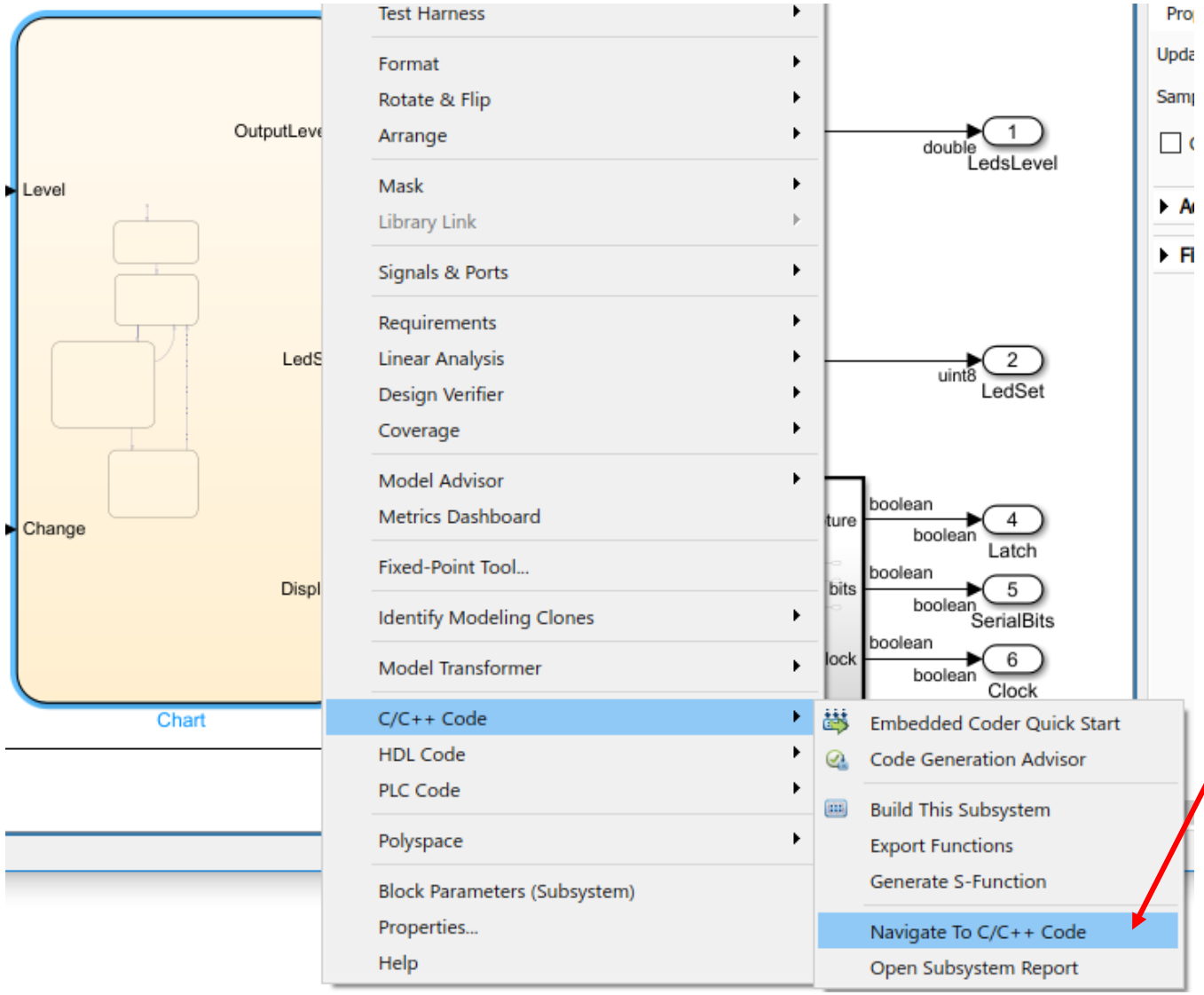
Tracability & Code generation report





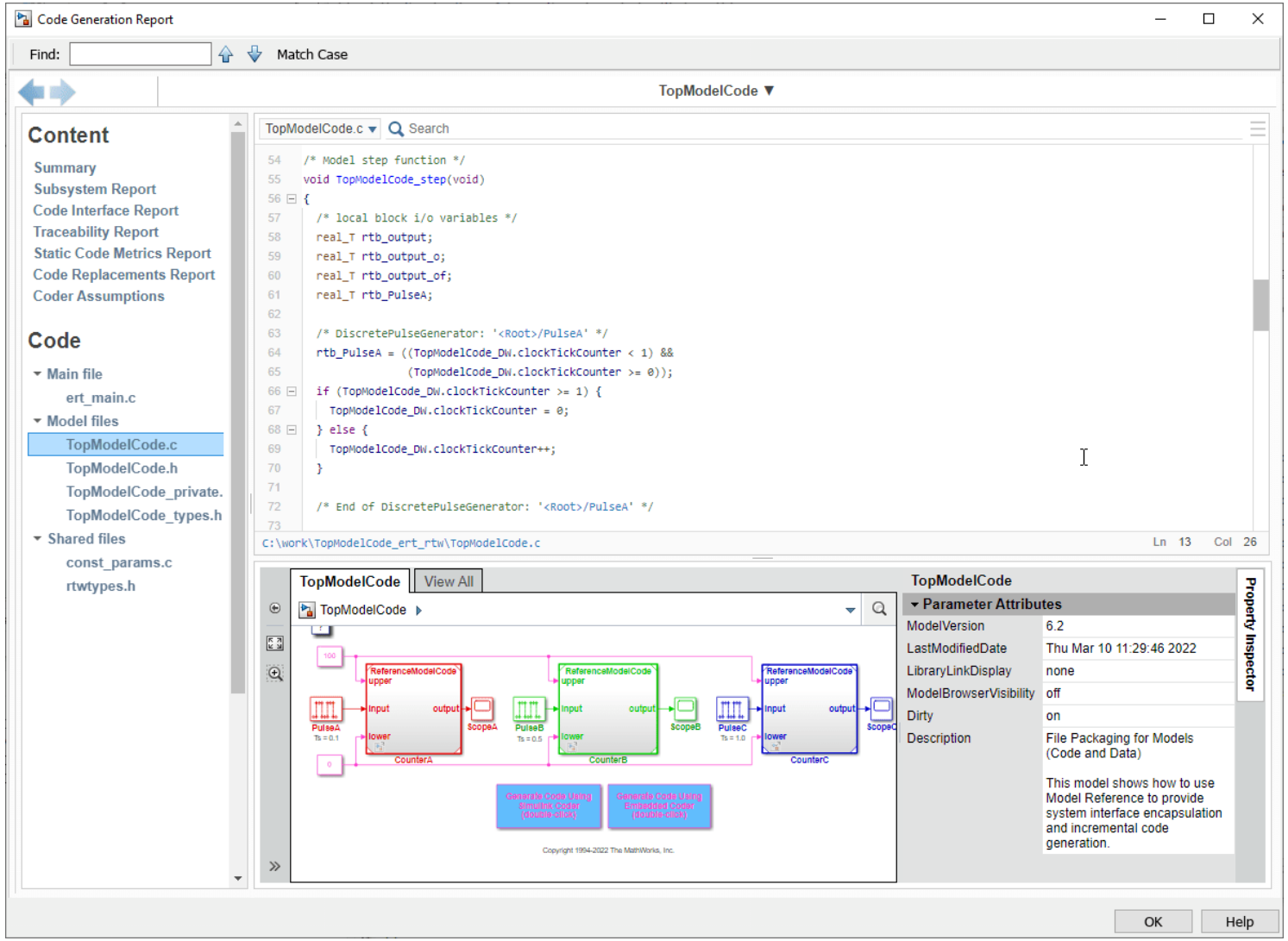
- In « Code interface report » tab you can see the main functions generated by Embedded Coder:
  - **RGBledsControl_Initialize:** contains code used during program initialization.
  - **RGBledsControl_ Step**: core of the algorithm called at each time step.
  - **RGBledsControl_ Terminate** Code to be called when program ends.
- You can navigate into the generated files.
- Hyperlinks allow you to navigate to corresponding blocks in model

Tracability & Code generation report



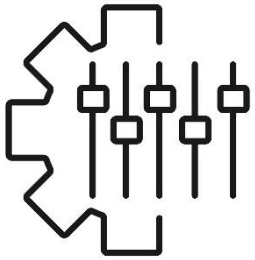
- You can navigate from model to code by right clicking on a block and selecting: « C/C++ Code > Navigate To C/C++ Code ».
- You are then re-directed on the corresponding line in code report.

Tracability & Code generation report

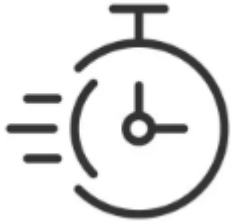


- Include Webview* for people who do not have access to Simulink license

* Need Simulink Report Generator



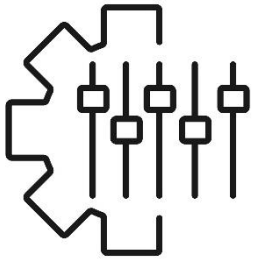
Customization



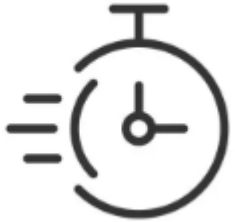
Optimization

Report

Code
Testing / Check



Customization



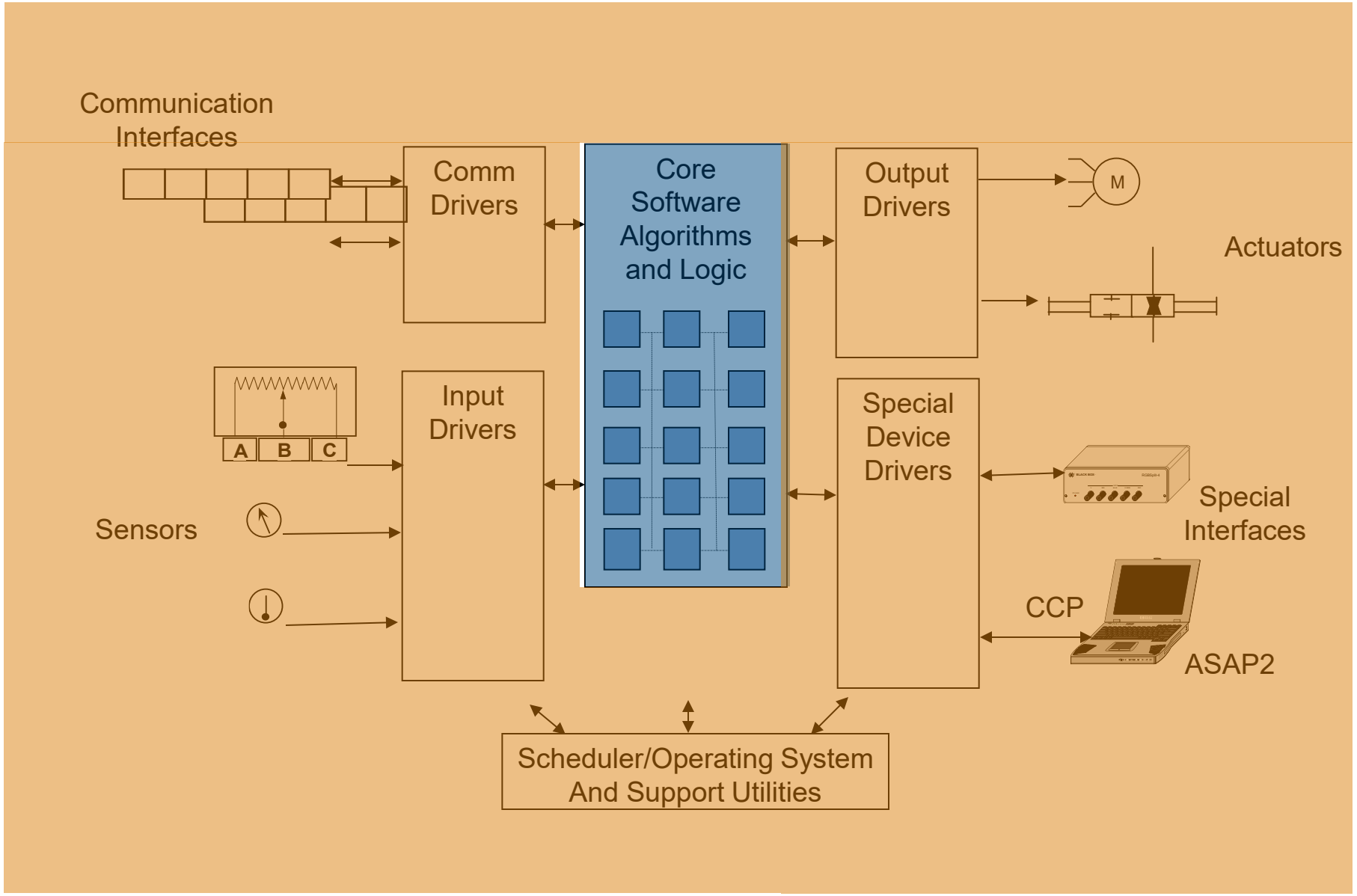
Optimization

Report

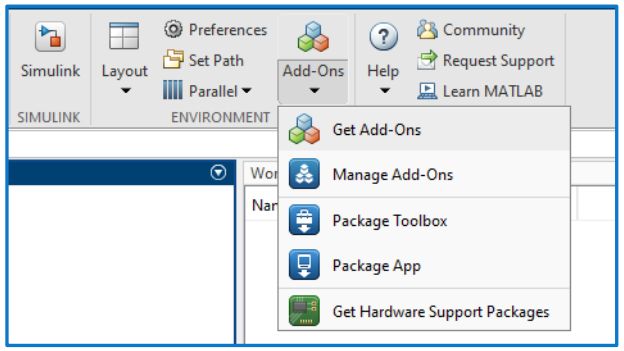
Code
Testing / Check

Code deployment

Simple Embedded Software Architecture



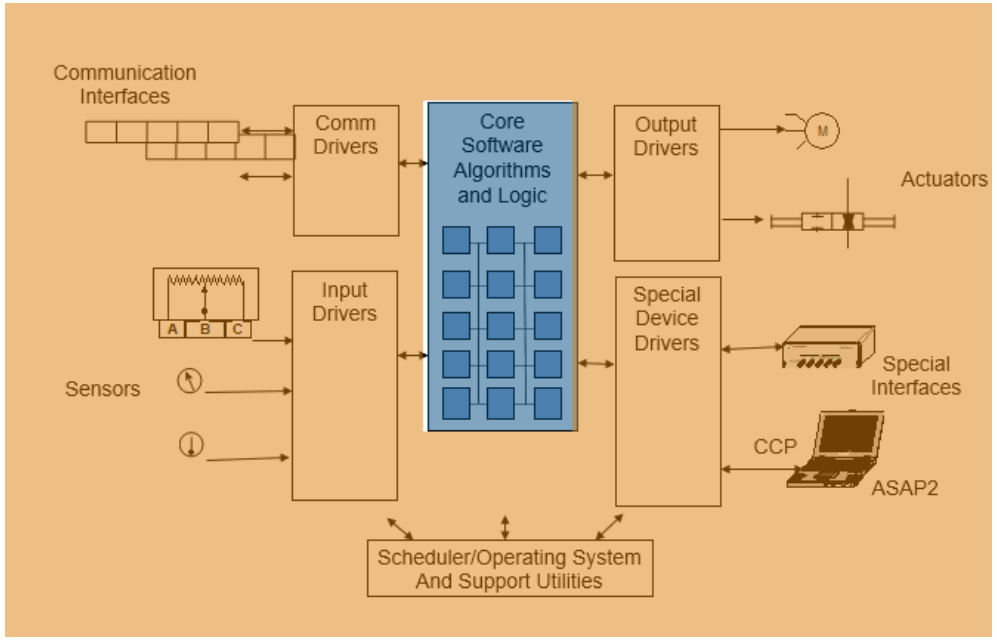
Custom Target, Hardware Support Package or Specialized toolbox



Simulink Support Package for Arduino Hardware by MathWorks
Simulink Team STAFF

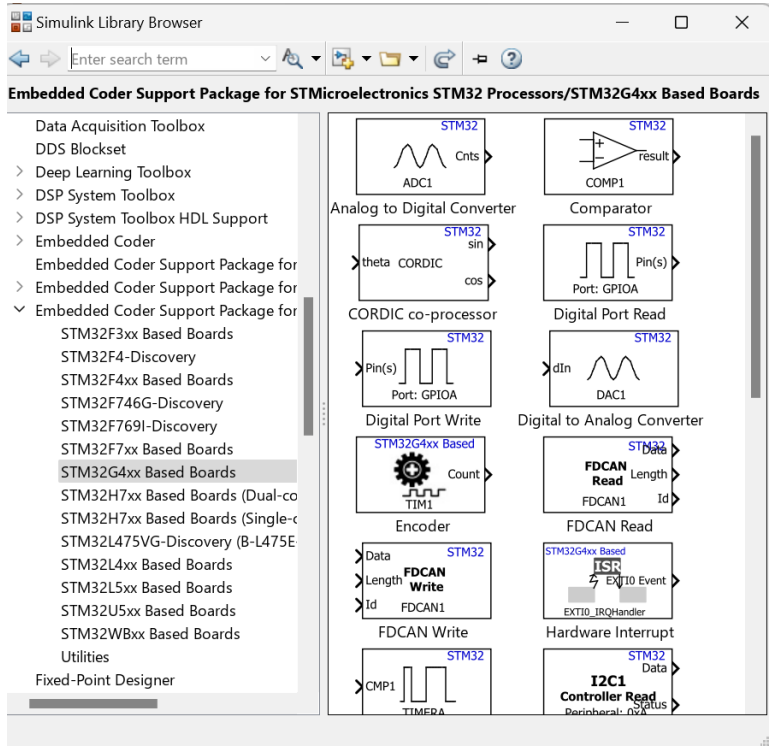
Run models on **Arduino** boards

	Author
Texas Instruments, ST, Infineon, Arduino, Raspberry Pi, Qualcomm	MathWorks
NXP, Microchip, Renesas	Partner

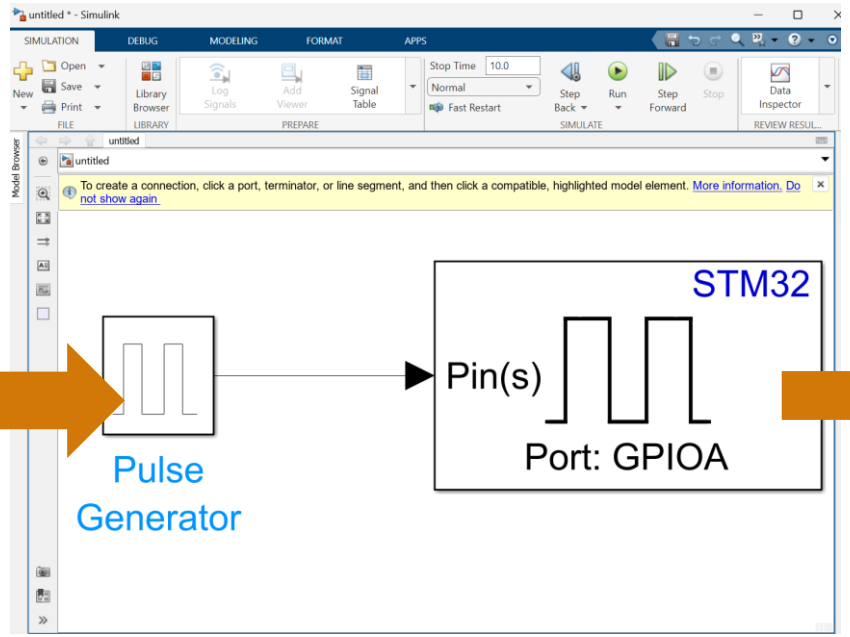


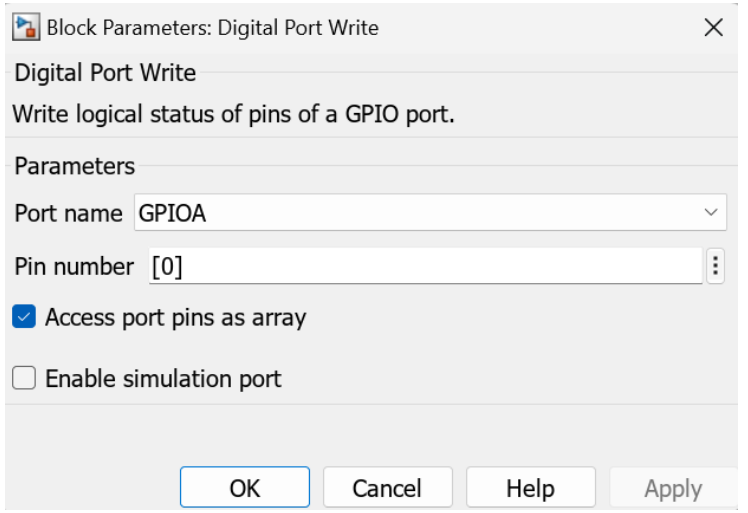
C2000 Microcontoller Blockset / STM32 Microcontroller Blockset / Raspberry Pi Blockset / Hardware Support Packages

Custom Target, Hardware Support Package or Specialized toolbox



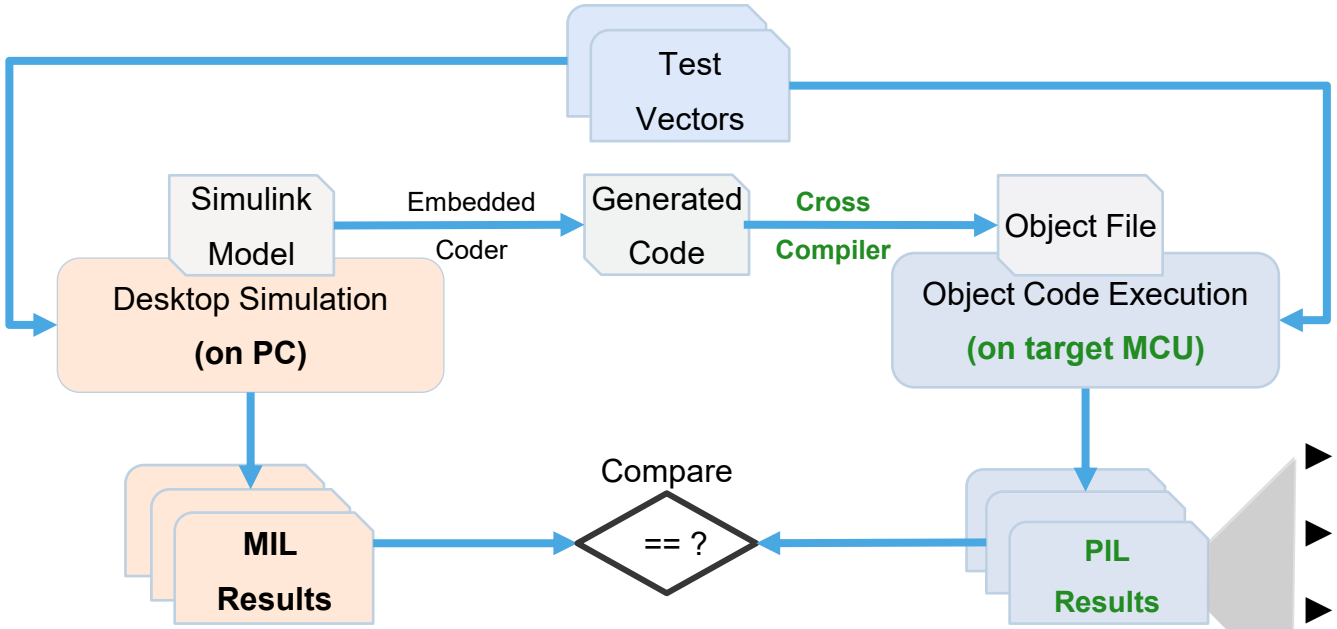
Drag & drop Peripheral Blocks



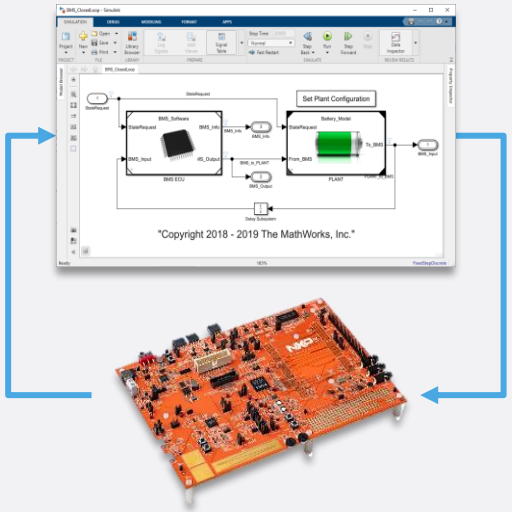


Configure Peripheral Blocks

PIL (Processor in the loop)



Plant model with test signals



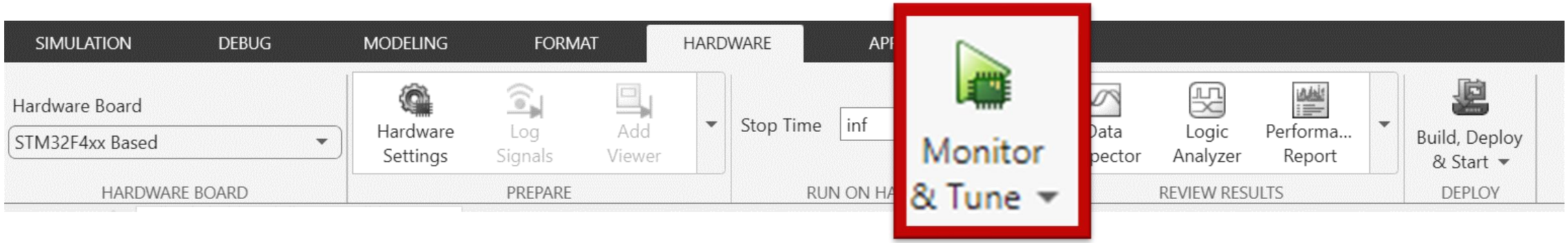
C Code on target MCU

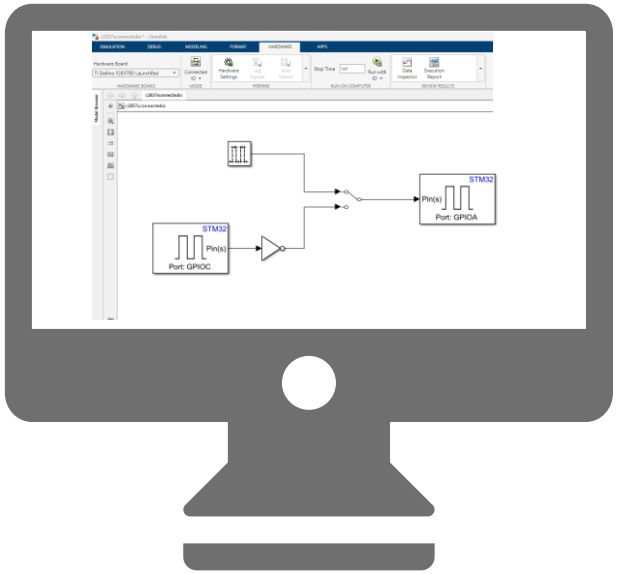
- Verify numerical equivalence
- **Profile target execution time**
- Collect on target code coverage
- Must for code replacement library

Section	Maximum Execution Time in ns	Average Execution Time in ns	Max
[+] Current_initialize	2260	2260	
Current_step [5e-05 0]	5135	5067	
Current_terminate	540	540	

Non Real-Time Execution

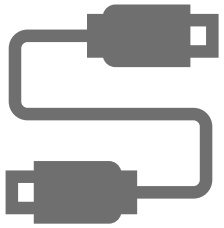
Monitor and Tune (Also called External Mode)



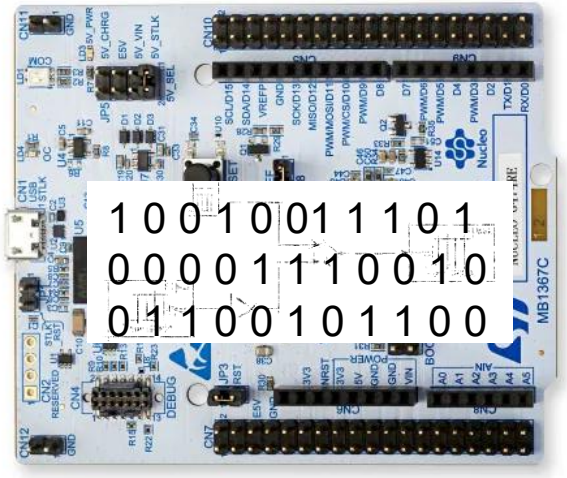


Simulink Model
(Monitoring only)

Real-Time Execution

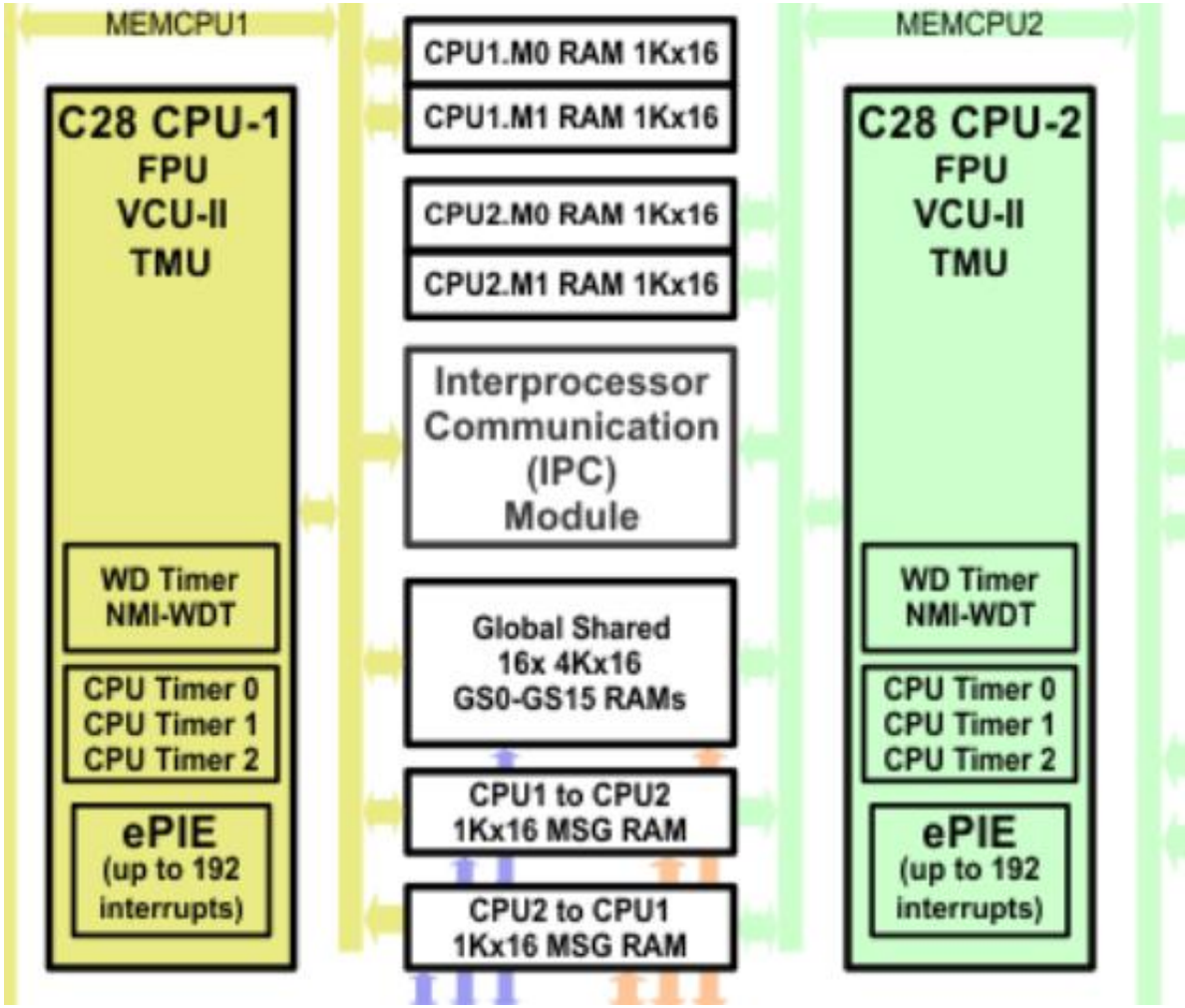


Data Exchange
over
XCP over Serial or CAN
or TCP/IP



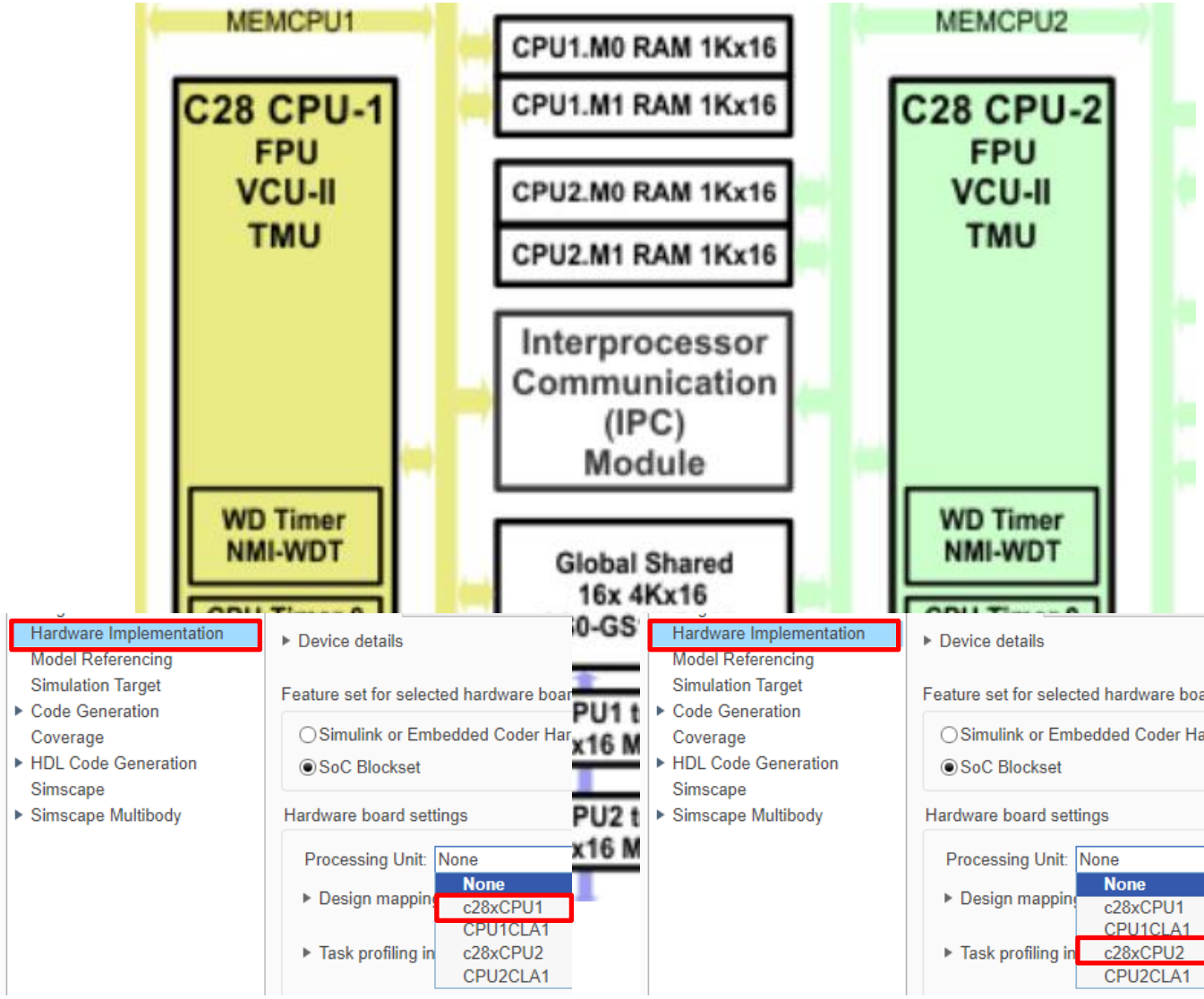
Model running on hardware
with additional code

How to deal with multi-processor board?



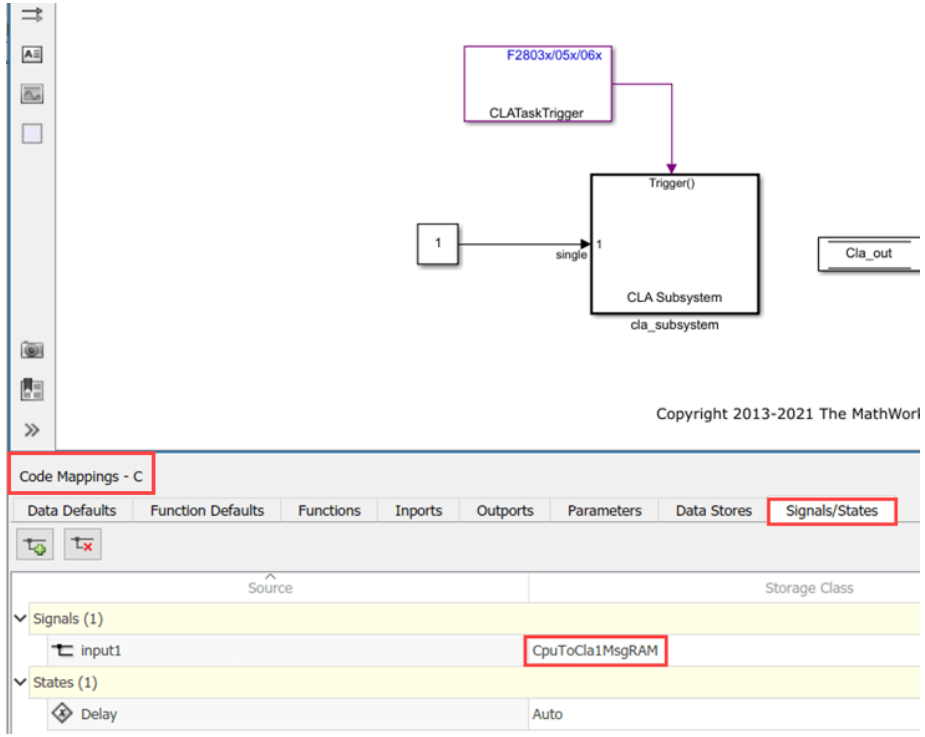
Embedded Coder workflow and configuration

- Pattern: « Expert Mode »
- Simulation: not easy
- When generating code for multiple processor
  - One processor = one model



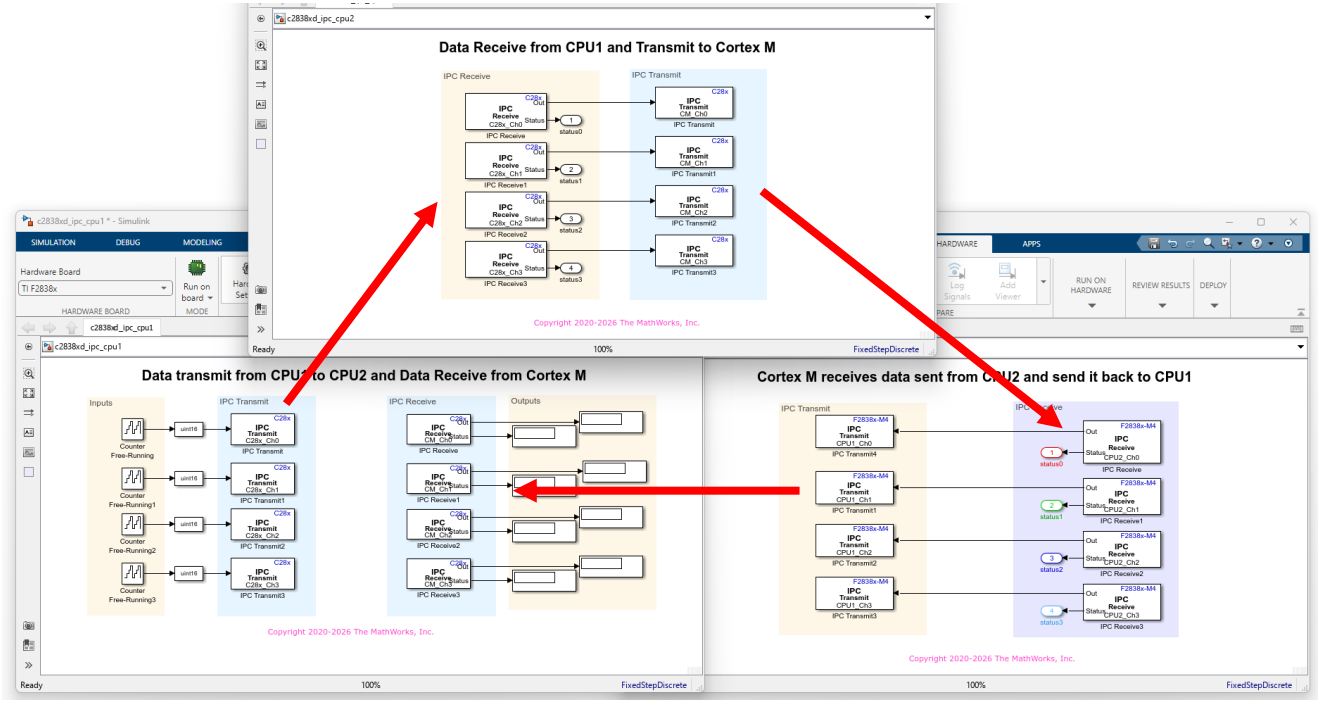
Embedded Coder workflow and configuration

- Pattern: « Expert Mode »
- Simulation: not easy
- When generating code for multiple processor
  - One processor = one model
  - Each models have
    - Its own hardware configuration
    - Modeling pattern to achieve processor interaction



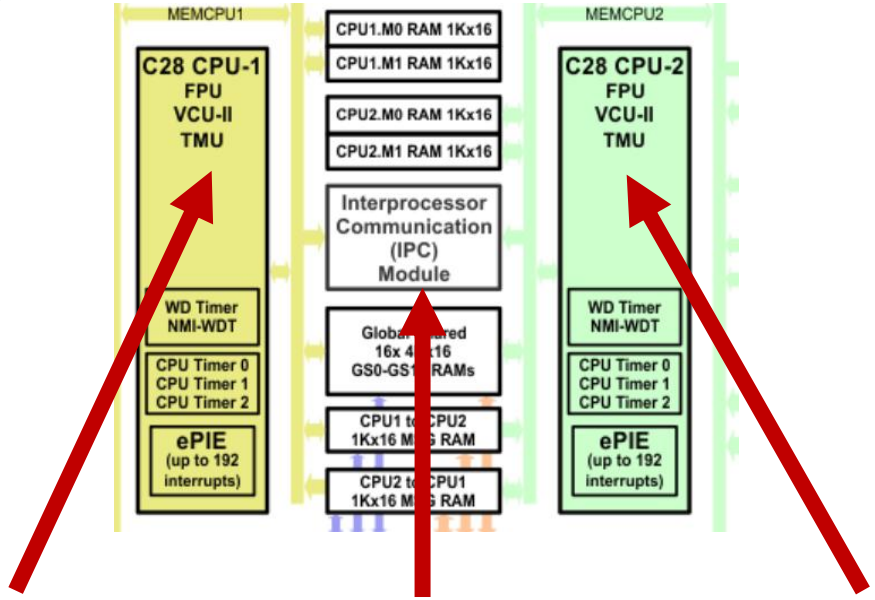
Embedded Coder workflow and configuration

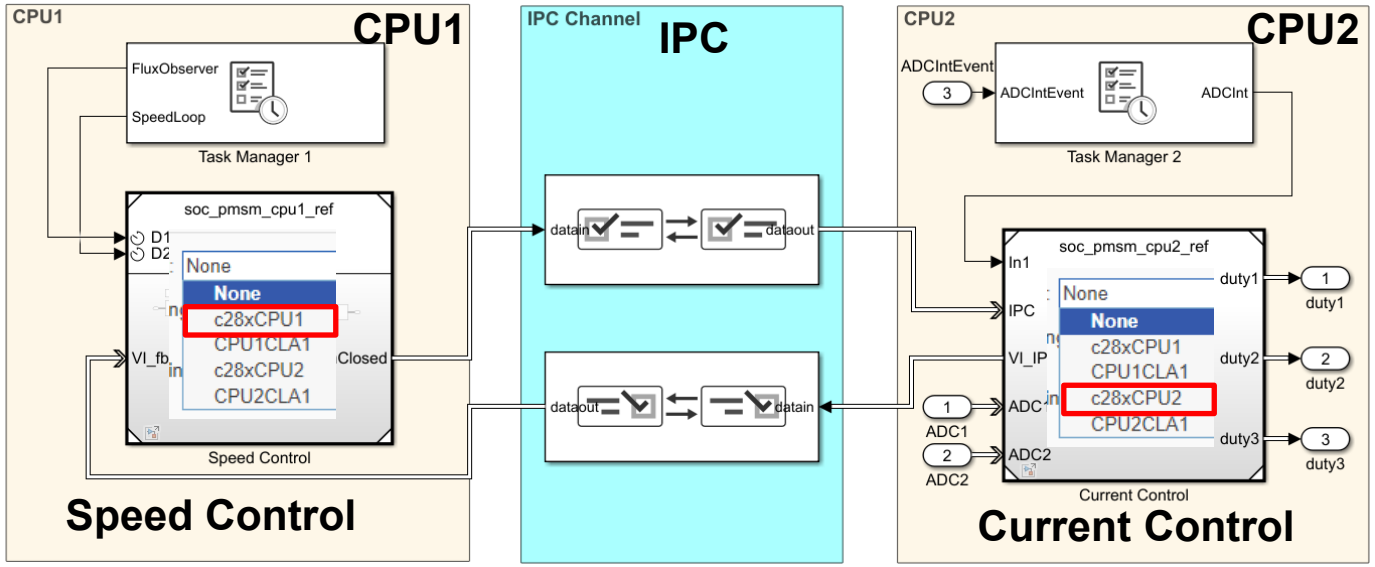
- Pattern: « Expert Mode »
- Simulation: not easy
- When generating code for multiple processor
  - One processor = one model
  - Each models have
    - Its own hardware configuration
    - Modeling pattern to achieve processor interaction
  - Each models need to be aligned with others



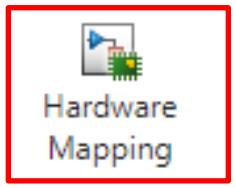
Embedded Coder workflow and configuration

- Pattern: « Simplified Mode »
- Simulation: easy & designed for
- When generating code for multiple processor
  - One processor = one model
  - One global model for the multi-processor architecture
    - Modeling patterns to achieve processor interaction

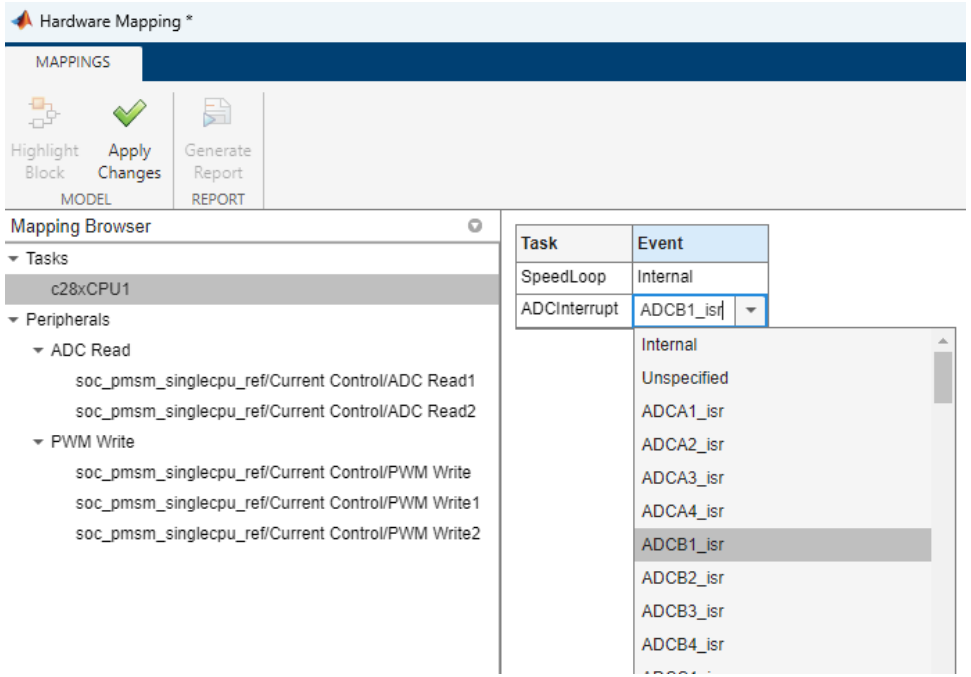


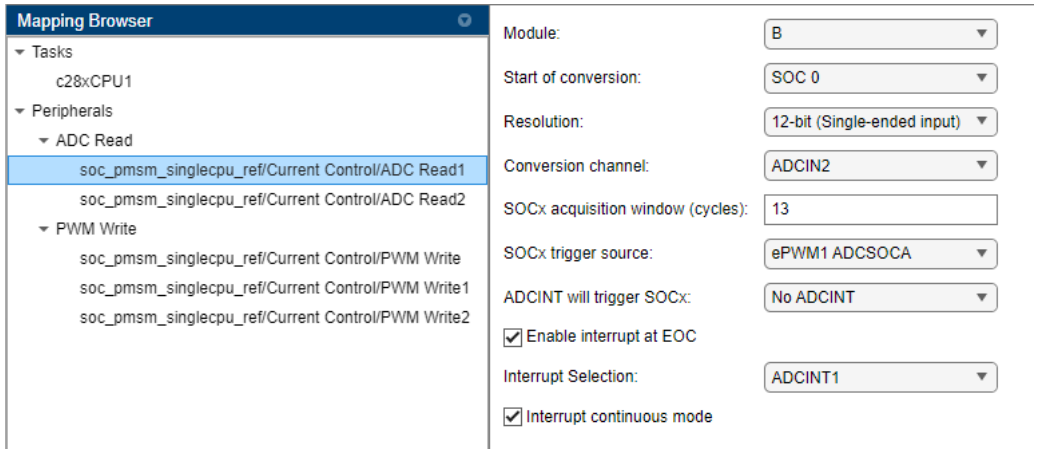


Embedded Coder workflow and configuration



- Pattern: « Simplified Mode »
- Simulation: easy & designed for
- When generating code for multiple processor
  - One processor = one model
  - One global model for the multi-processor architecture
    - Modeling patterns to achieve processor interaction
    - To create the global hardware configuration

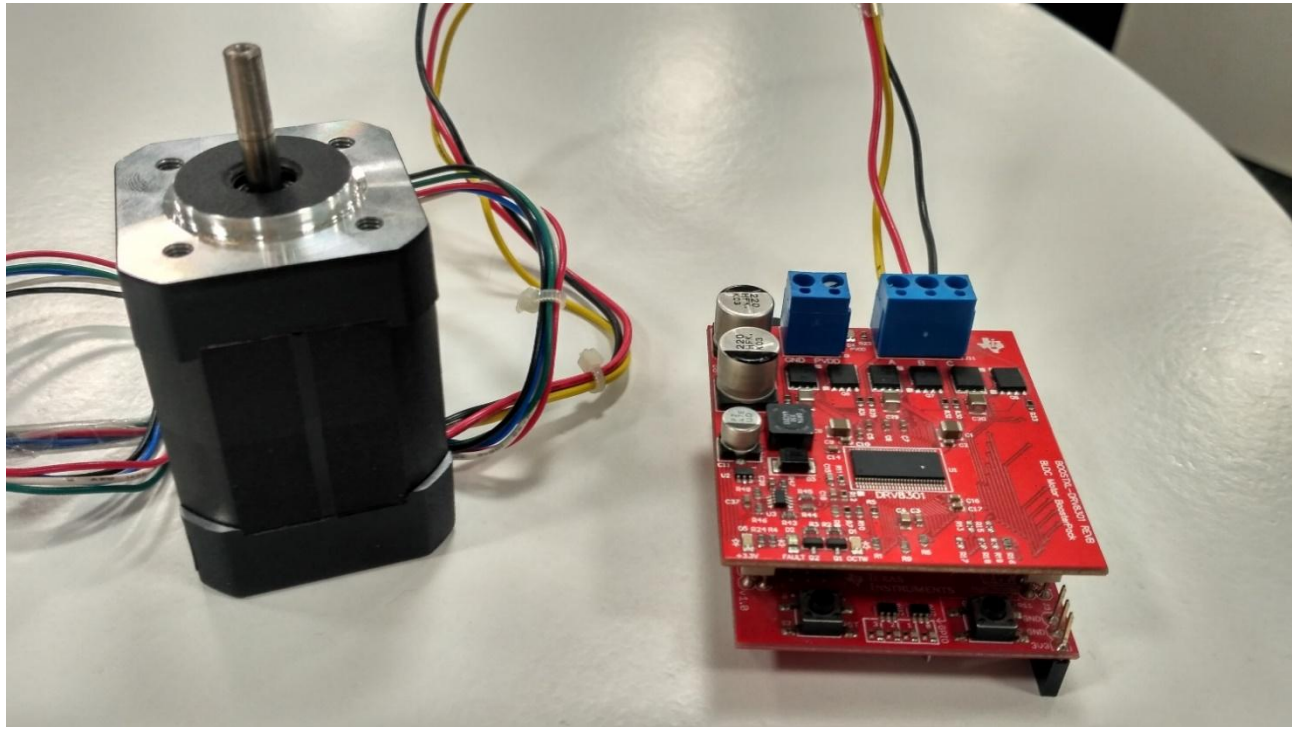




Workshop Example

- FOC sensorless Motor control on TI Launchpad F28379D
  - Existing Single processor algorithm running with PWM at 20 Khz
  - Need to achieve 40 Khz

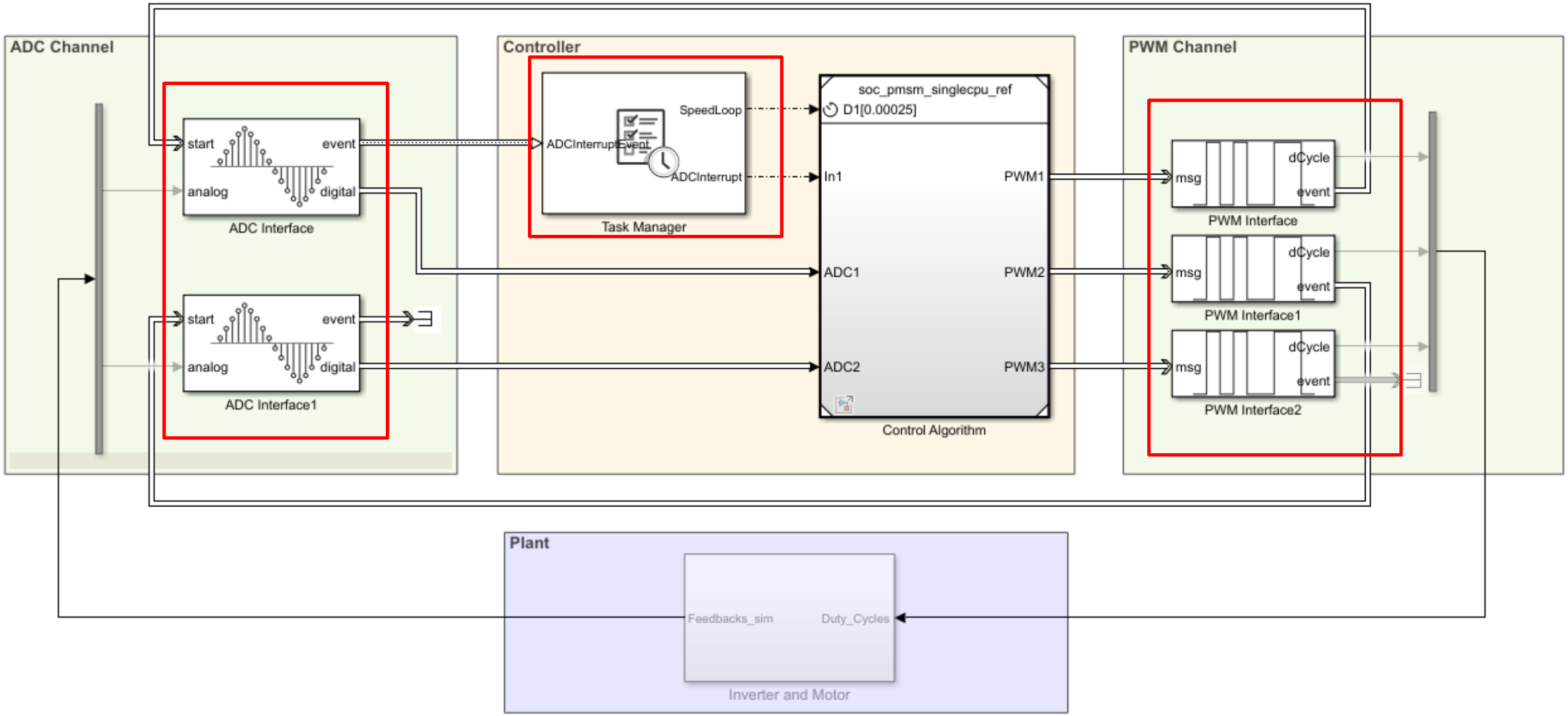
- Is single processor possible ?
- How to transform our algorithm to multi-processor if required?



openExample('shared_mcb_soc_c2b/PartitionMotorControlForMultiprocessorMCUsExample')

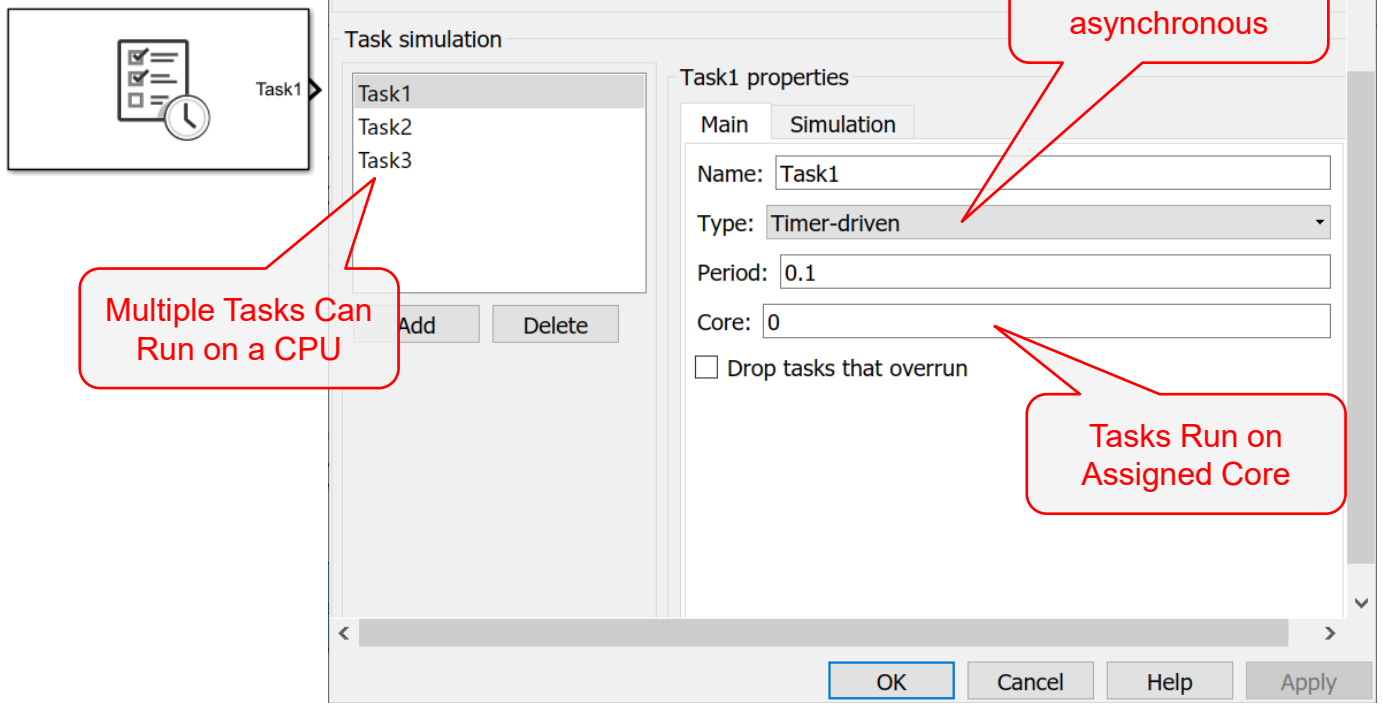
Field Oriented Control In Single CPU

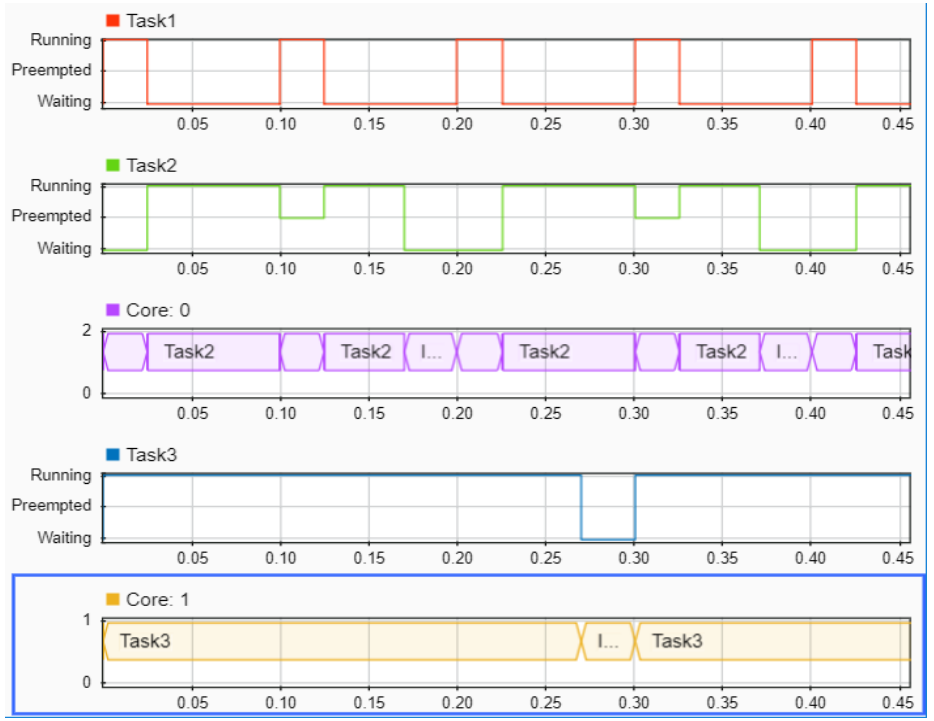
Single processor model

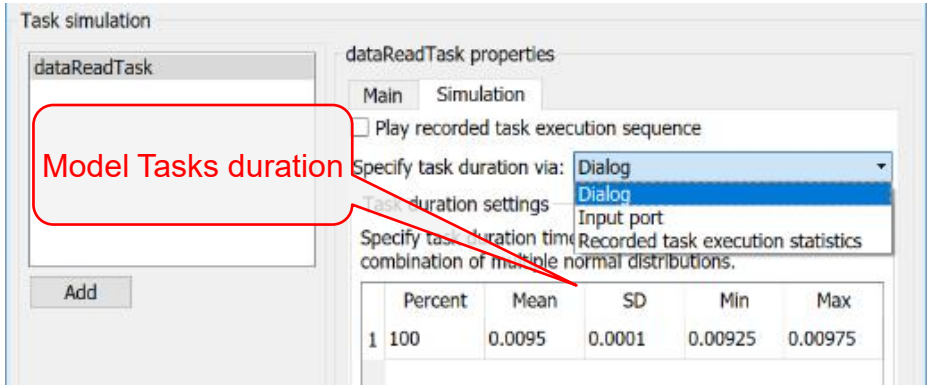


Task Partition and Management with **Task Manager Block**

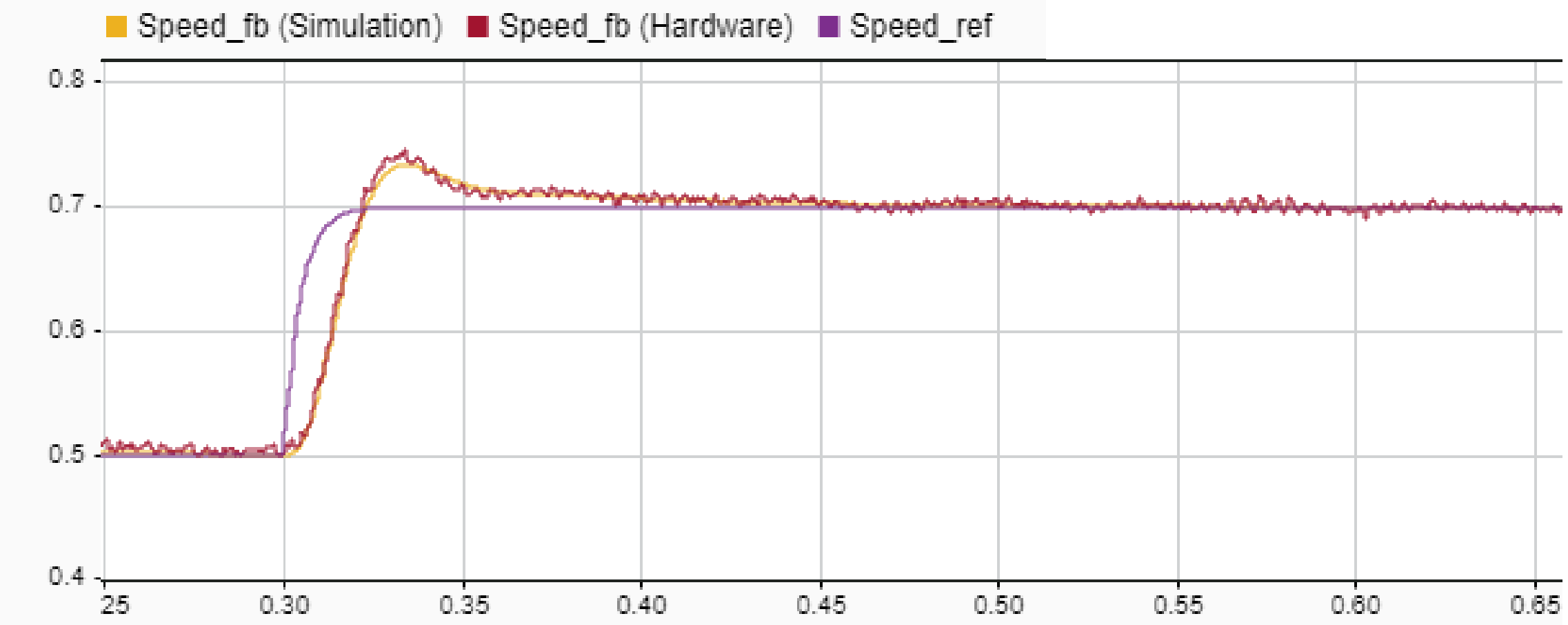
- Model task execution timing impact





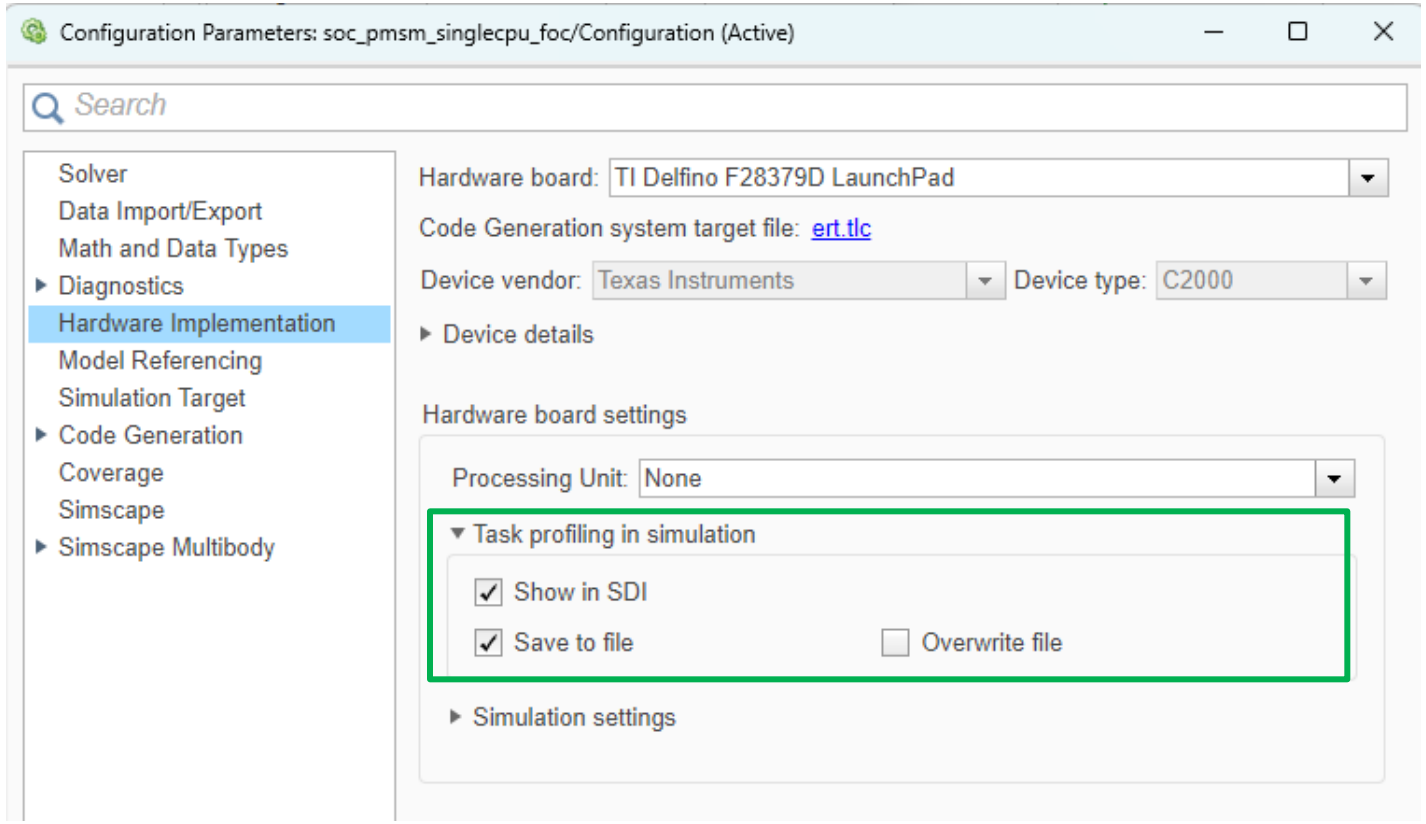


Comparison Simulation vs Hardware



How to do profiling?

- External mode required
- Activate Task profiling in Configuration Parameter

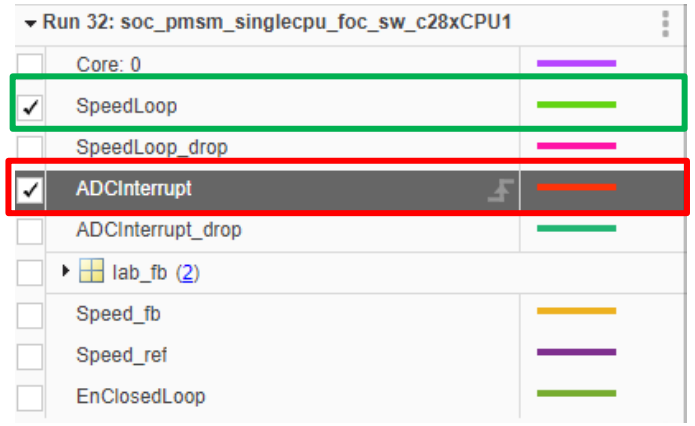


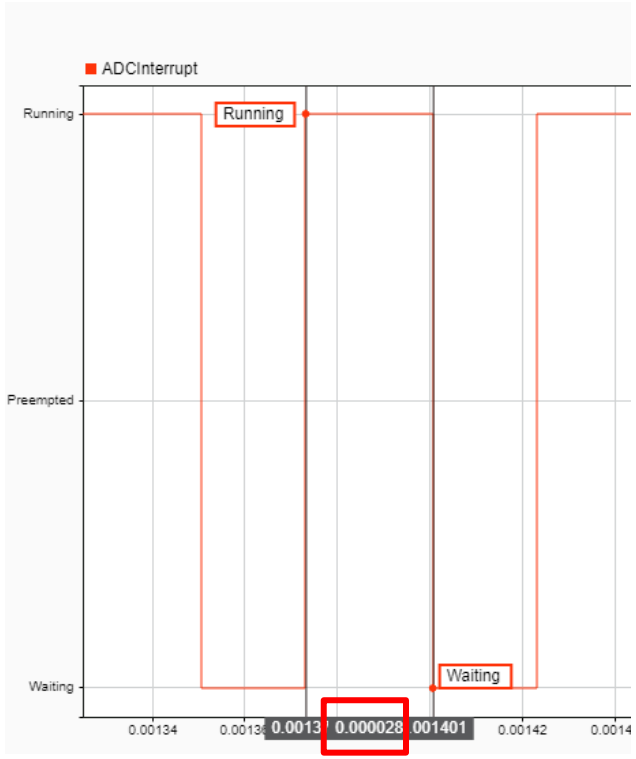
[Documentation page with details and example]

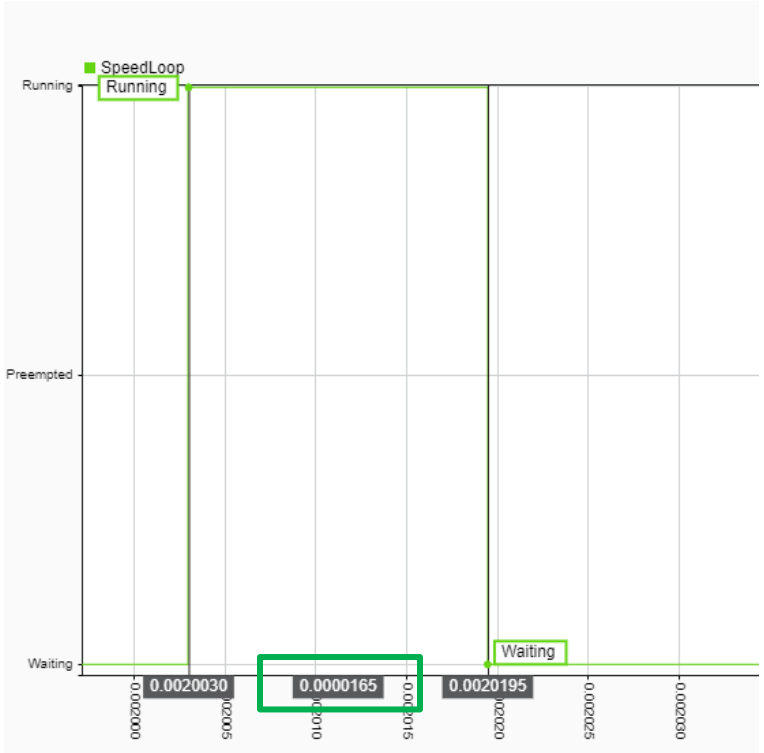
How to do profiling?

- 2 ways to look the results:
  - Profiling report generated
  - Task signals in SDI

Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls	
soc_pmsm_singlecpu_foc_sw_c28xCPU1_initialize	2034340	2034340	2034340	2034340	1	
soc_pmsm_singlecpu_foc_sw_c28xCPU1_step [0.0005 0]	18670	13184	18670	13184	28	
ADCInterrupt	32580	27713	32580	27713	271	



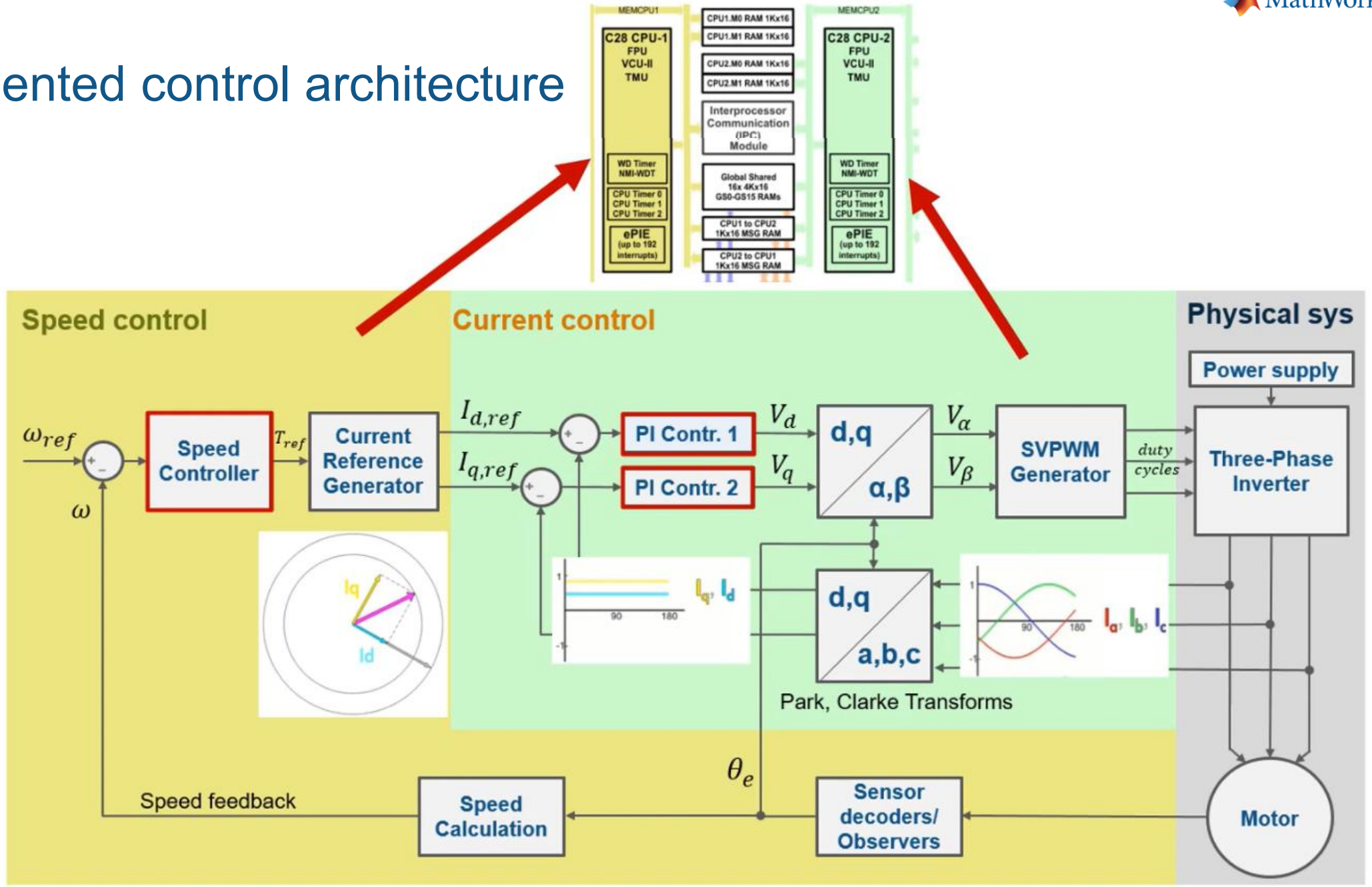




Workshop Example

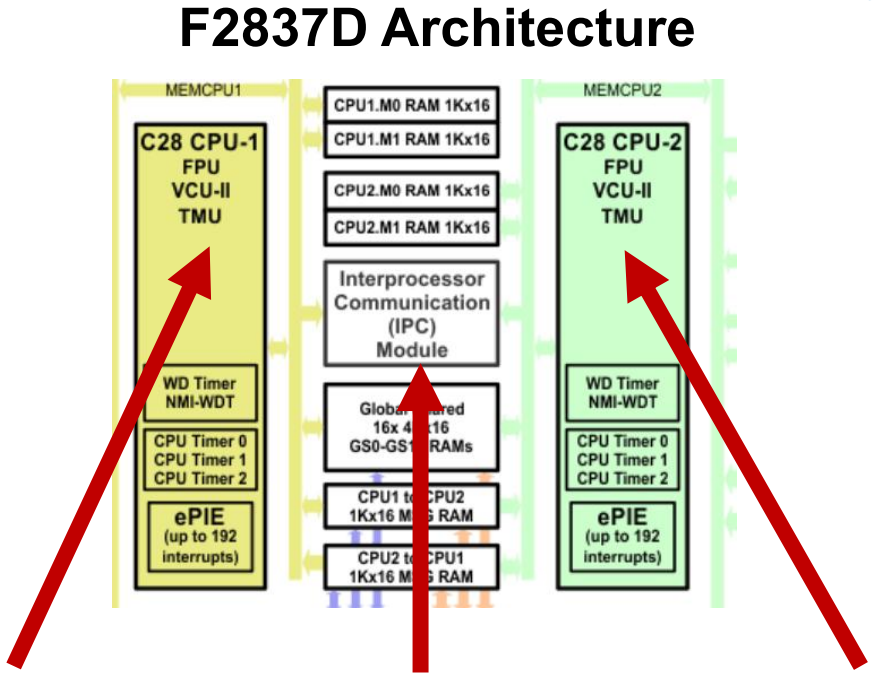
- FOC sensorless Motor control on TI Launchpad F28379D
  - Existing Single processor algorithm running with PWM at 20 Khz
  - Need to achieve 40 Khz
- Is single processor possible ?
  - NO: asynchronous task take already 27 µs which is higher than 25 µs (40 kHz)
- How to transform our algorithm to multi-processor if required?

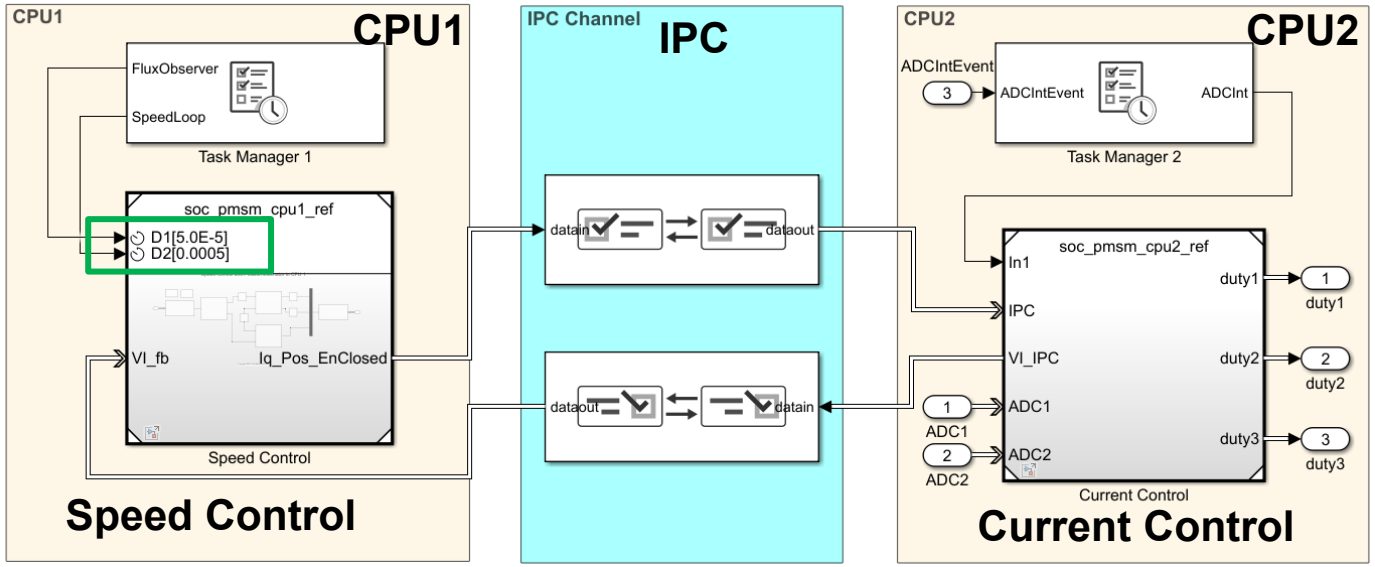
Field-oriented control architecture



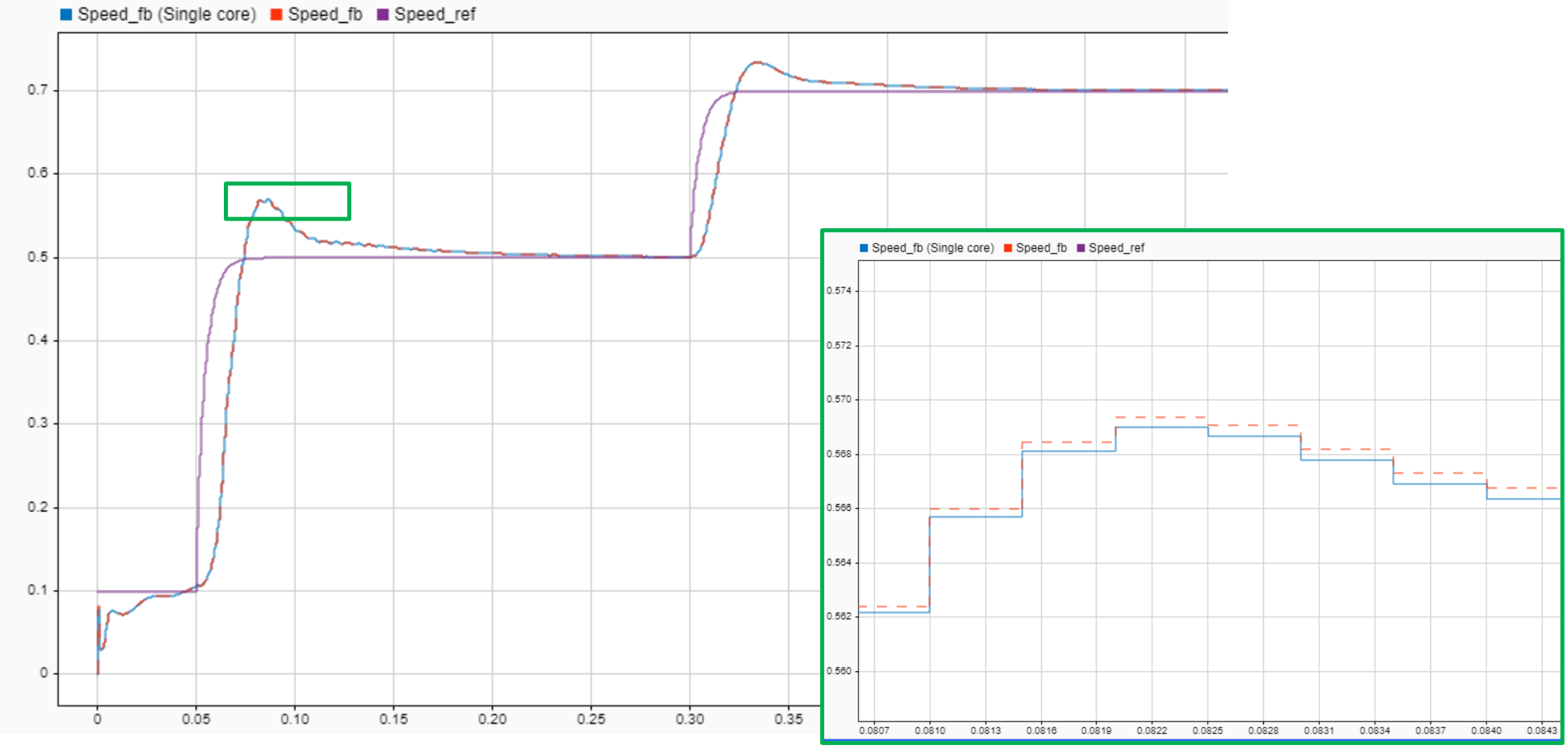
Design for F28379D multi-processor

- F28379D Architecture: 2xCPU + IPC module
- Distribute current control and speed control



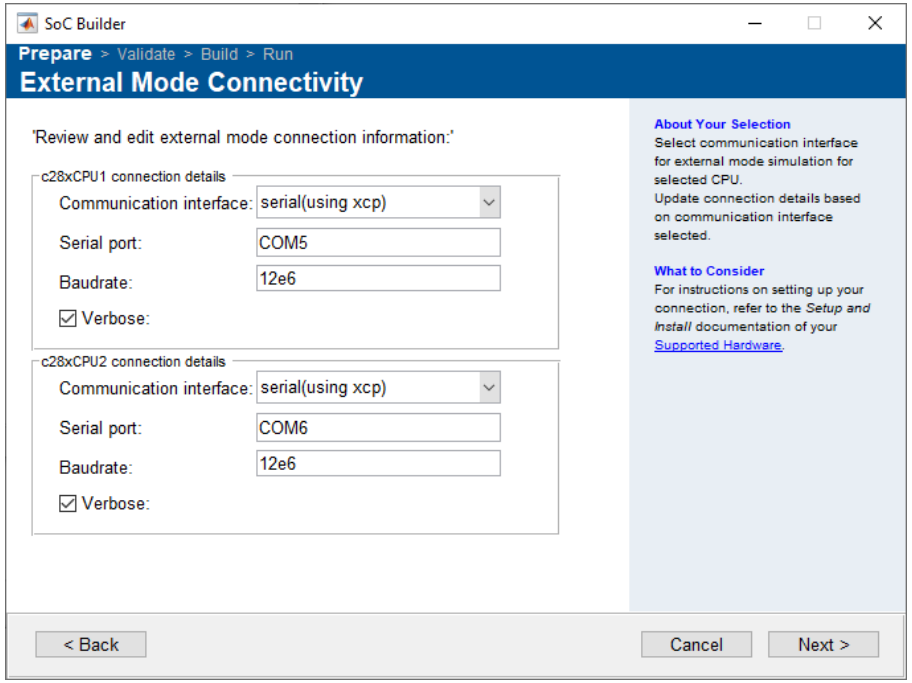


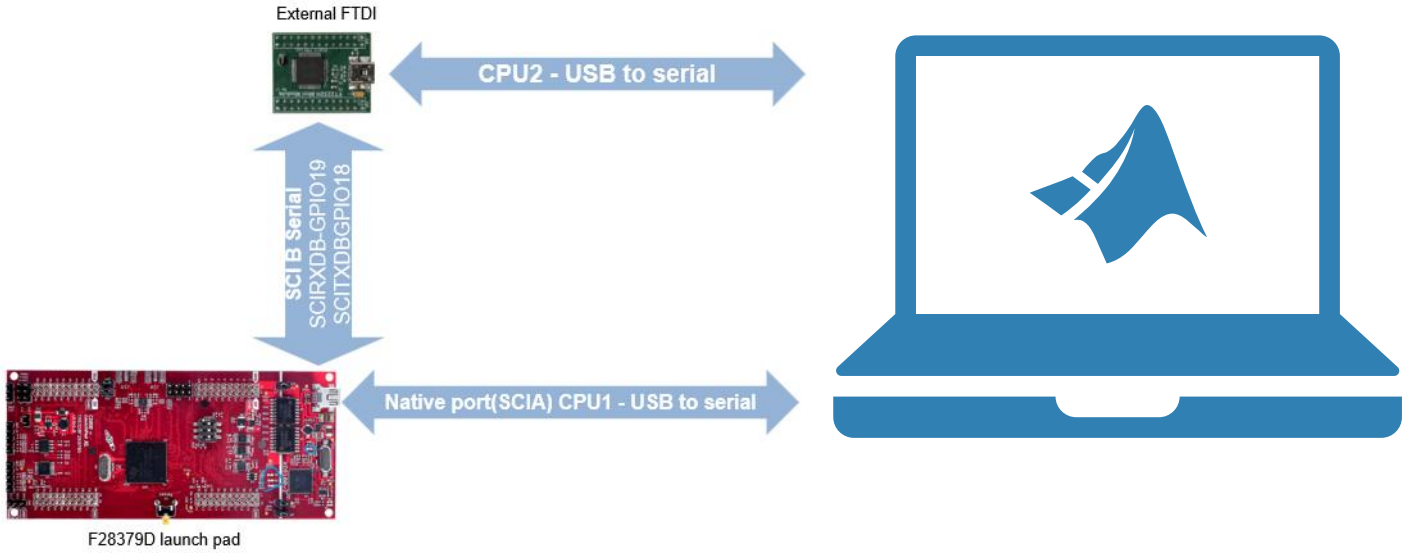
Comparison Single vs Multi processor



Dual External mode and Profiling

- Live task execution information from device to Simulink





Profiling results

- CPU1

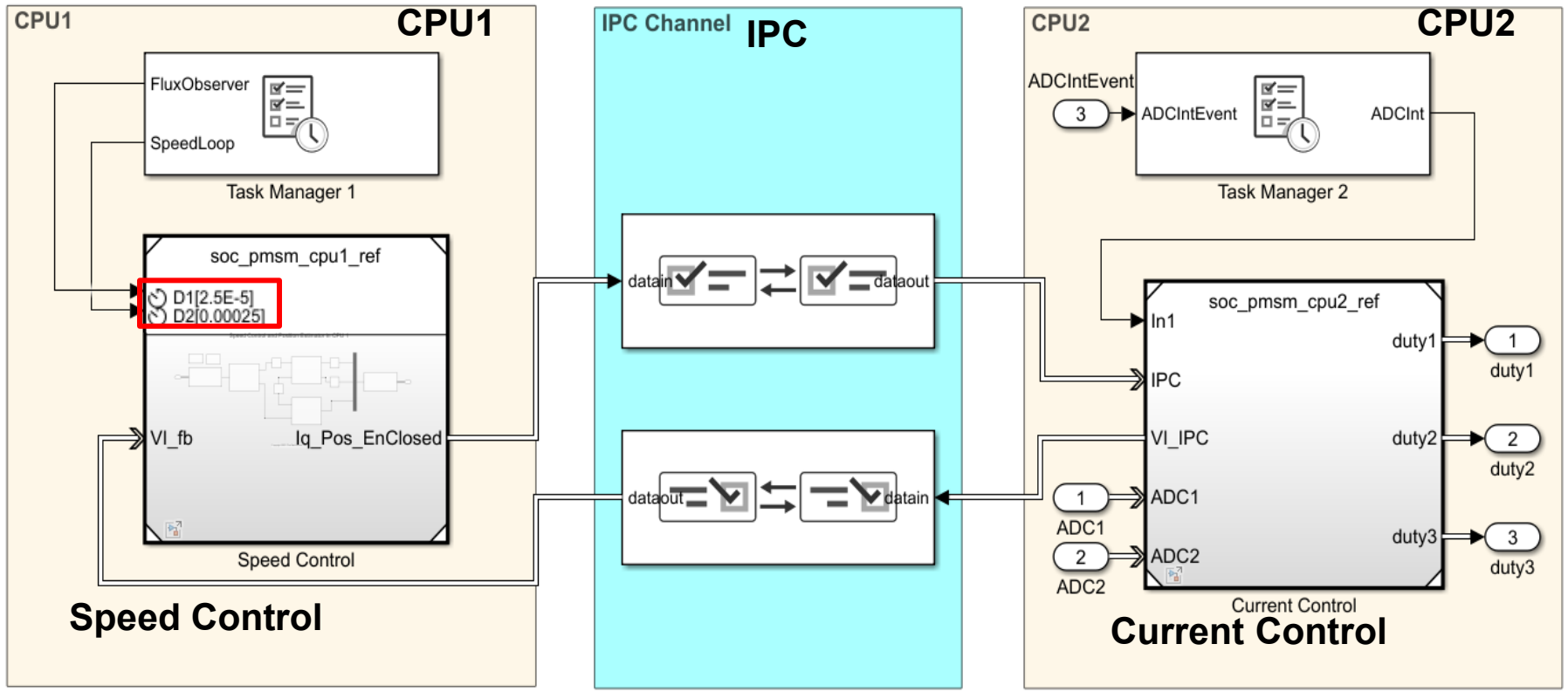
Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls
soc_pmsm_dualcpu_foc_sw_c28xCPU1_initialize	25350	25350	25350	25350	1
soc_pmsm_dualcpu_foc_sw_c28xCPU1_step0 [5e-05 0]	23070	22230	23070	22230	290
soc_pmsm_dualcpu_foc_sw_c28xCPU1_step1 [0.0005 0]	19120	14813	19120	14813	29

- CPU2

Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls
soc_pmsm_dualcpu_inv_foc_sw_c28xCPU1_initialize	2065220	2065220	2065220	2065220	1
soc_pmsm_dualcpu_inv_foc_sw_c28xCPU1_step [0.2 0]	7720	7720	7720	7720	1
ADCInt	35400	29886	35400	29886	378

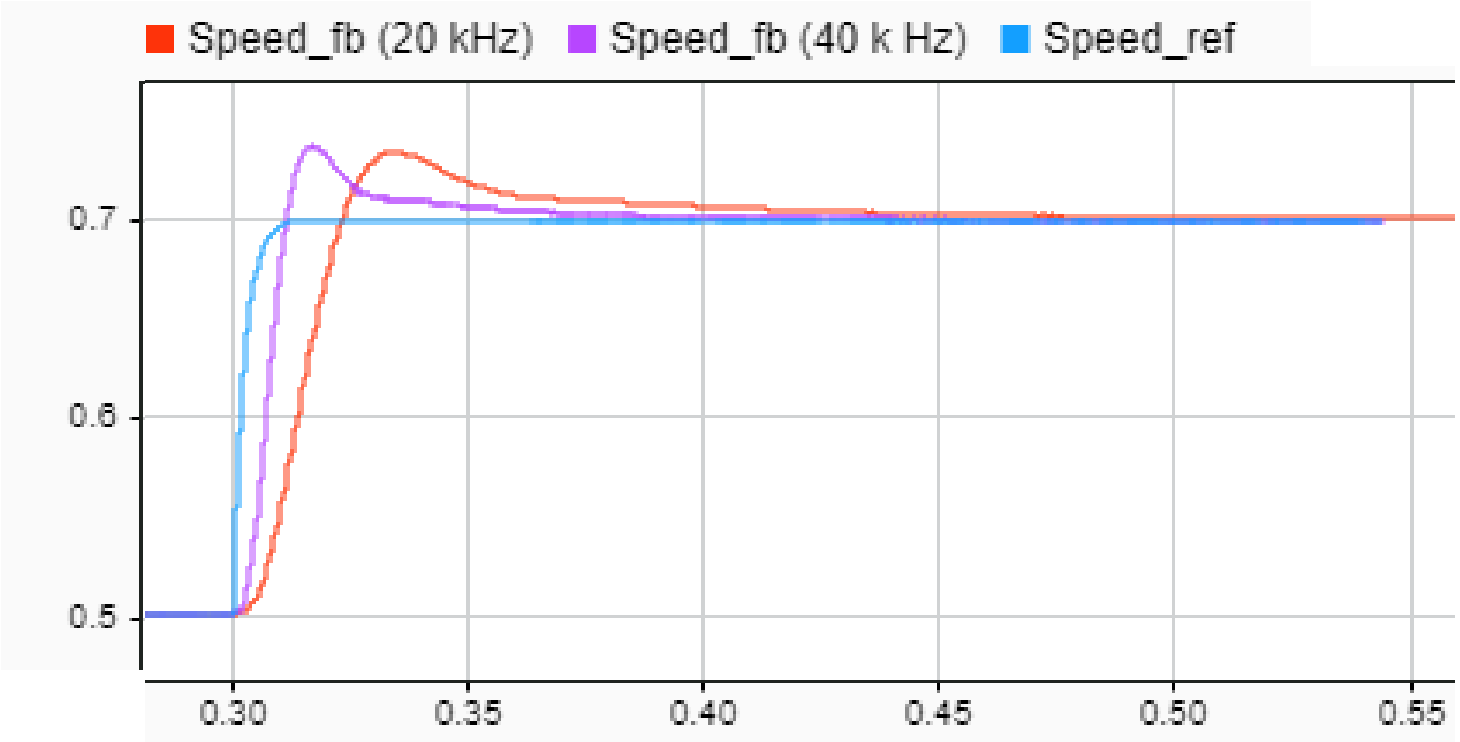
Switch from 20 kHz to 40 kHz

- As 23 µs is less than 25 µs (40 kHz) we can increase CPU1 frequency



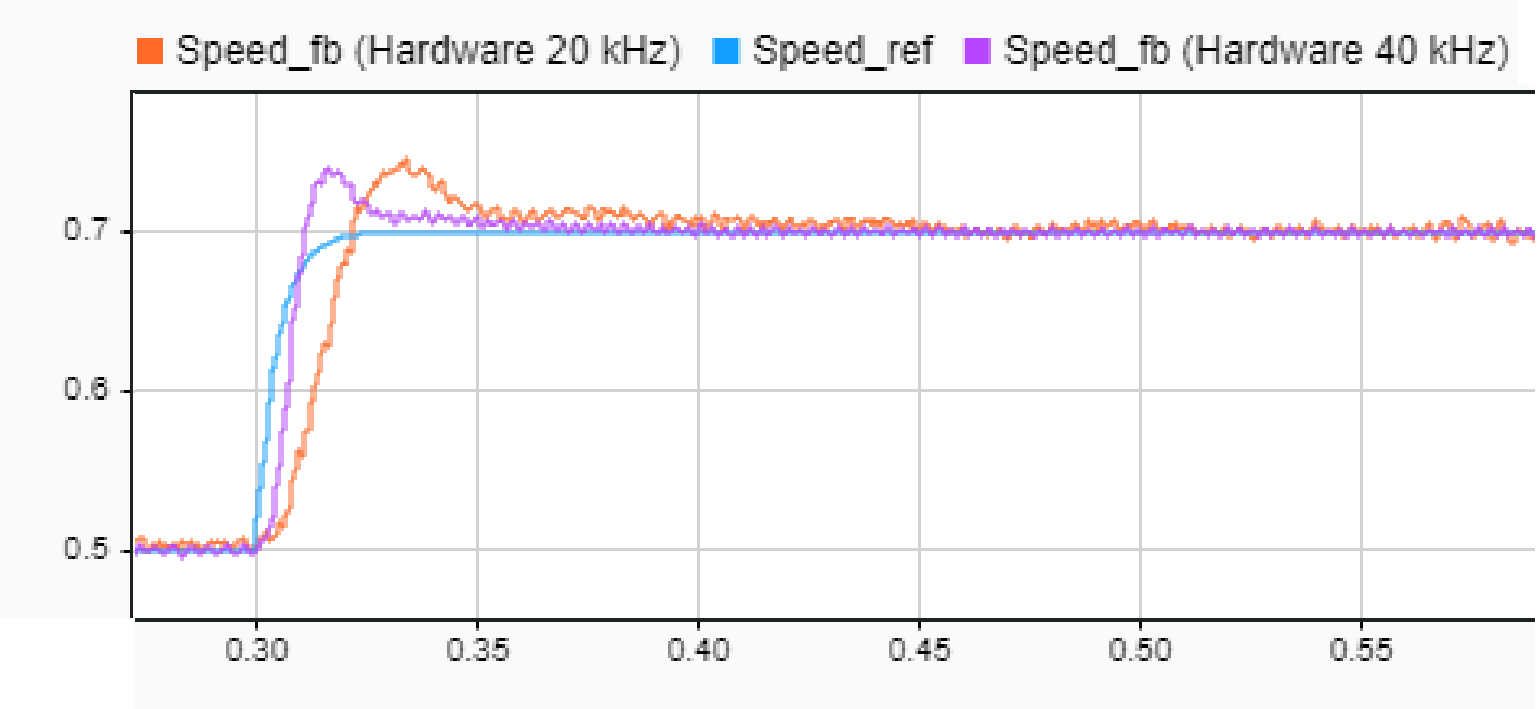
Comparison simulation Dual processor 20 kHz vs 40 kHz

- As expected, the response time seems improved



Comparison Hardware Dual processor 20 kHz vs 40 kHz

- It is confirmed on hardware & no overrun occurred



My final thought

- « Expert Mode »
  - + Used for many years now
  - + Faster
  - ± Freer, less guided
  - – More possibility of error
  - – No Simulation
- « Simplified Mode »
  - + Enable Simulation
  - + Less possibility of error
  - ± Very guided
  - – Slower

- Simulink enables Model-Based design workflow for Multi Processor board with :
  - – FPGA & CPU
  - – ARM & not ARM CPU
  - – Control Law Accelerator
- Depending on the board it is possible to do
  - – Bare-metal
  - – OS based

Which boards do you want to target?

C2000 Microcontoller Blockset / STM32 Microcontroller Blockset / Raspberry Pi Blockset / Hardware Support Packages / Contact us!

Thank you

Q & A