



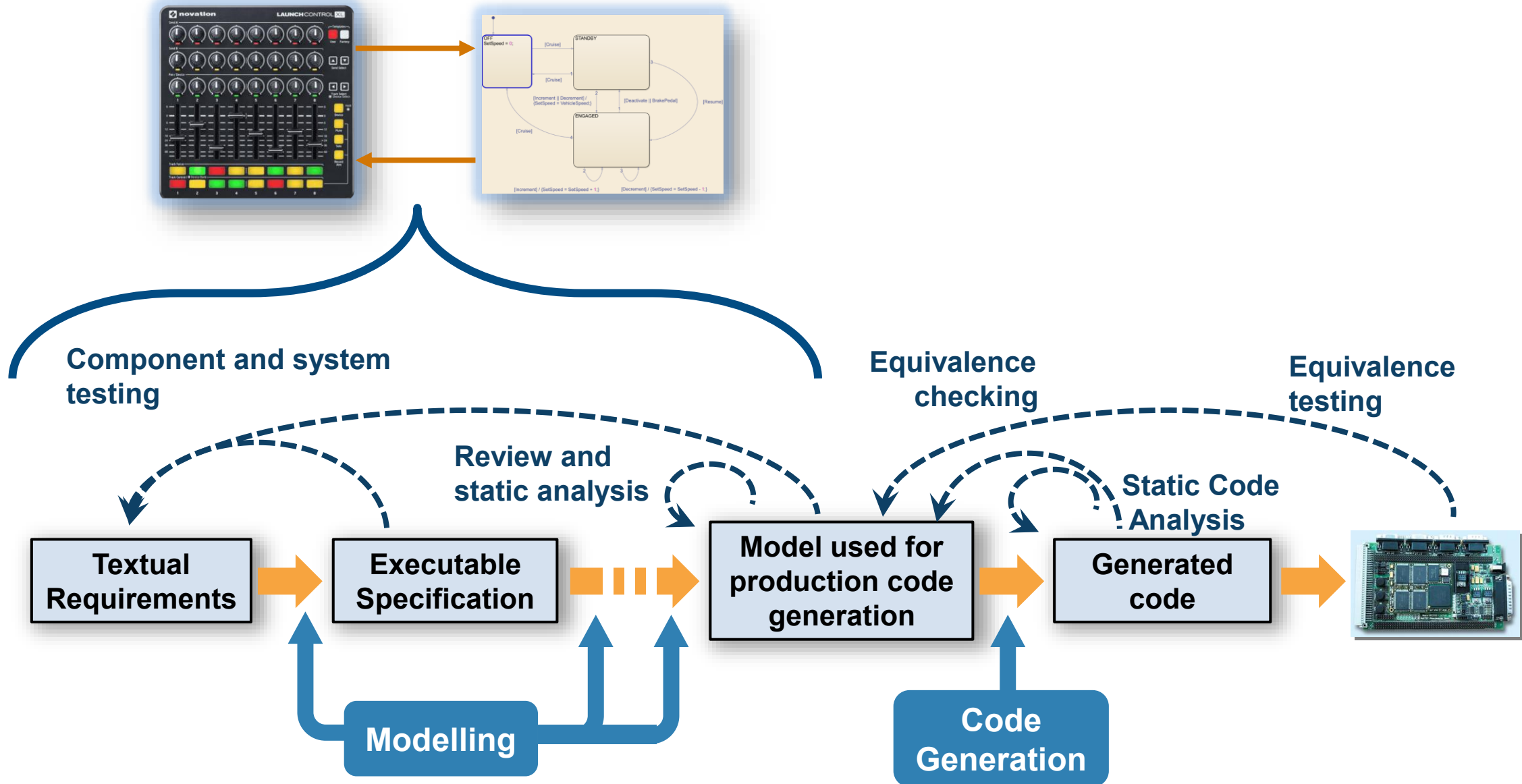
From Models to Embedded Code



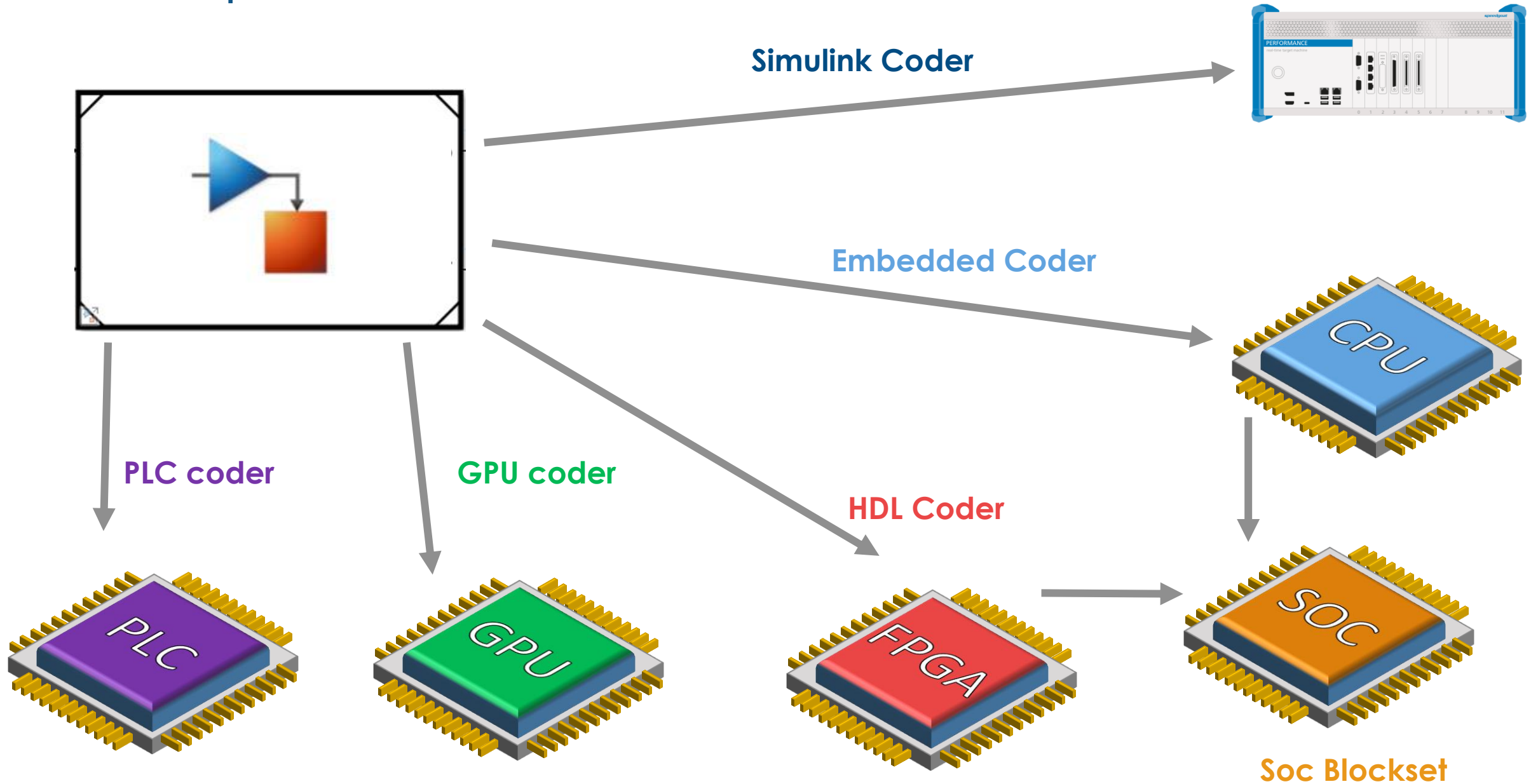
Morgan Fremovici – *Application Engineer*
mfremovi@mathworks.com



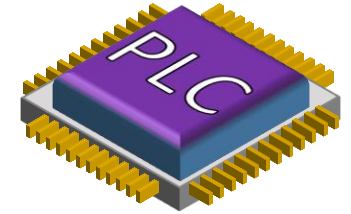
Model-Based Design Workflow



Overall picture



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



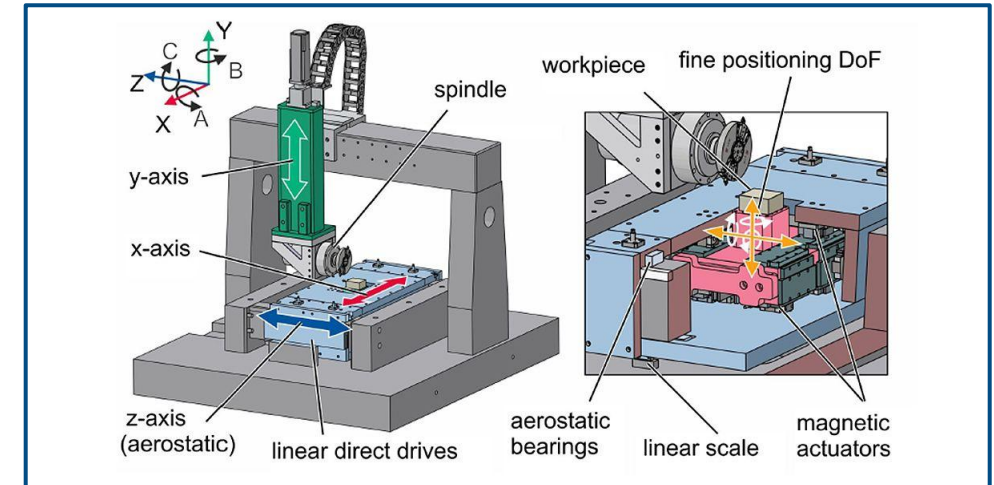
Researchers at Leibniz University Hannover used Model-Based Design to build a high-precision machining prototype that uses electromagnetic levitation guides.

Results/Key Outcomes

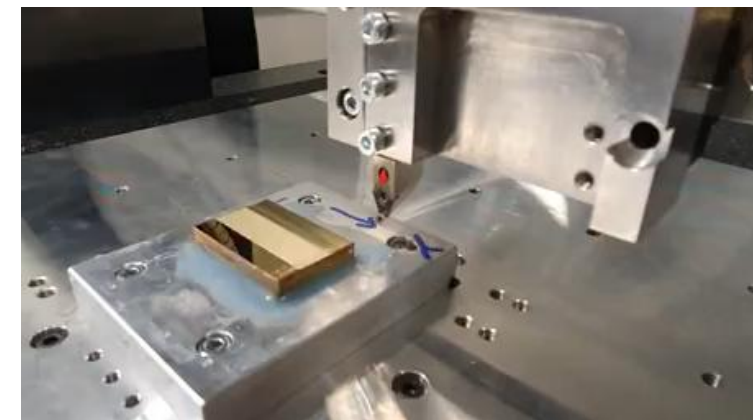
- Model-Based Design enabled modeling and simulation of a high-precision machining prototype with extra degrees of freedom and active vibration control
- Simulink PLC Coder was used to streamline development by automating IEC 61131-3 Structured Text generation for a Beckhoff Industrial PC

“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.”

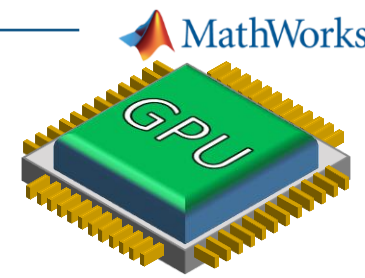
- Per Schreiber, Institute of Production Engineering and Machine Tools (IFW), Leibniz University Hannover



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.



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

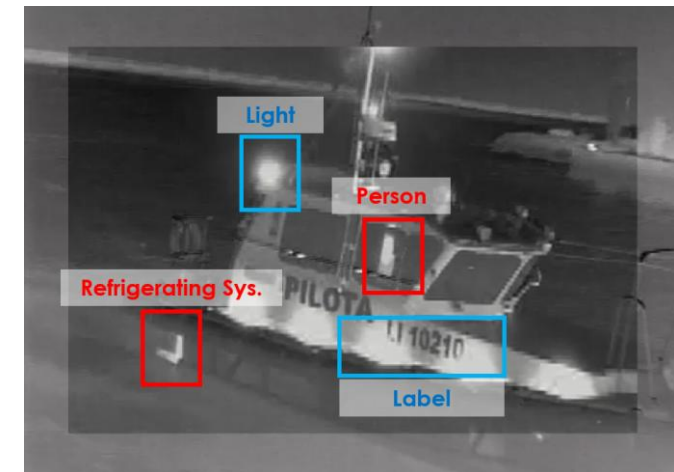
- Data labeling automated
- Development time reduced
- Flexible and reproducible framework established

“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



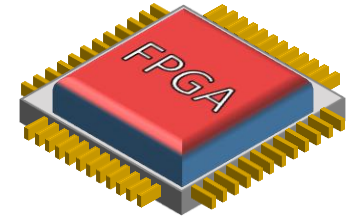
Object detection tests with optronic system prototype



Drass

Maritime object tracking on NVIDIA GPU

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

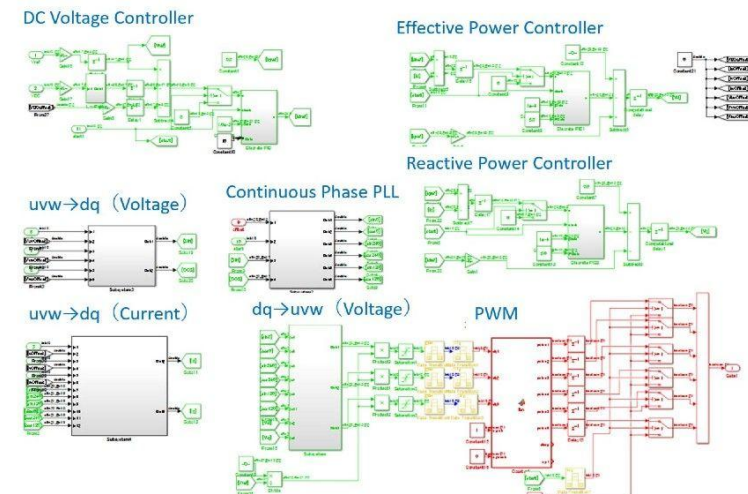
Use Simulink to model and simulate the power converter control software and plant and generate synthesizable Verilog code from the controller model using HDL Coder



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

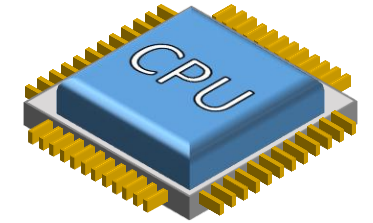
“Model-Based Design enabled us to develop the control software at a cost 60% less than the estimates provided by major manufacturers and to cut development time by more than 50%.”

*- Yoshinori Kurimoto,
High Energy Accelerator Research Organization (KEK)*



Simulink model of controller subsystems.

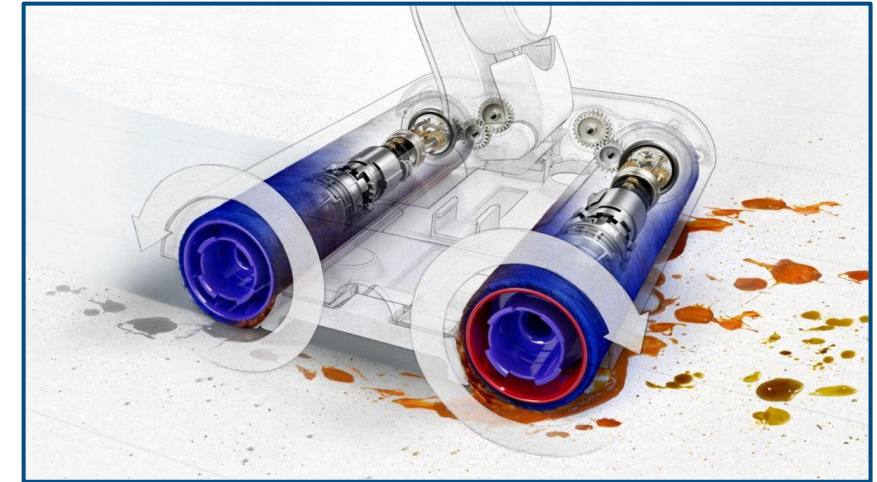
Dyson Accelerates New Product Development with System-Level Simulation and Code Generation



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

Key Outcomes

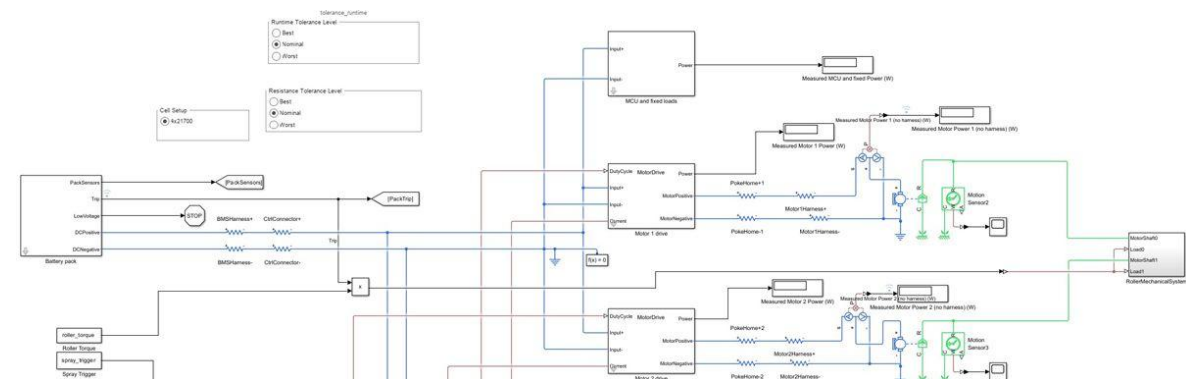
- By leveraging Model-Based Design, Dyson accelerated the testing of new design iterations, achieving a 100% increase in speed.
- Using Embedded Coder, Dyson shrunk its software release cycle from 10 weeks to nine days.
- Using Requirements Toolbox, Dyson tested their requirements before the hardware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



Components of Dyson roller technology.

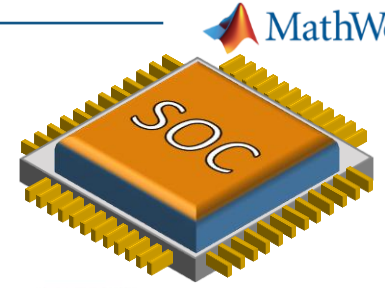
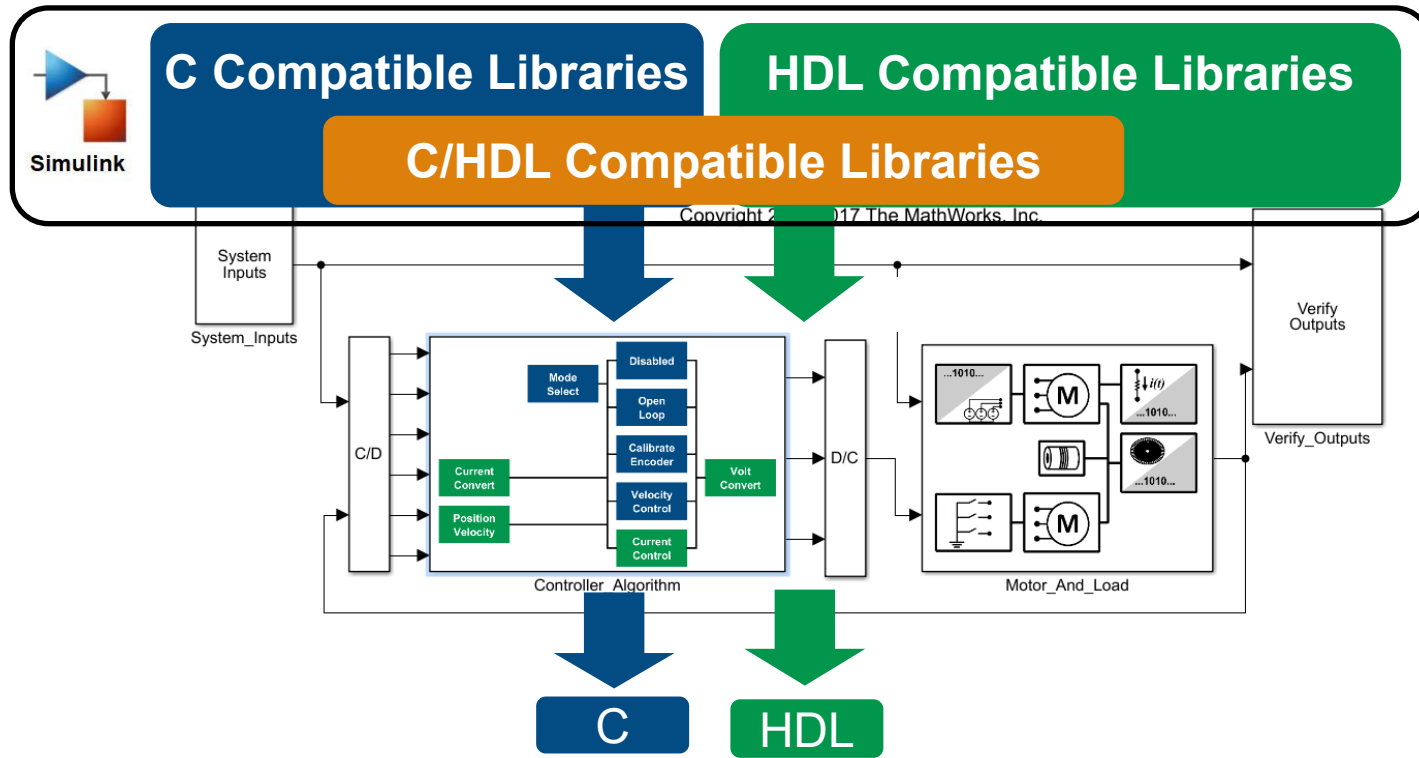
"We needed to explore many different concepts and directions. Using Model-Based Design and Simulink models gave us the ability to be agile and turn new ideas around twice as fast compared to our document-based development process."

- Romain Guicherd, lead advanced control systems engineer at Dyson



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

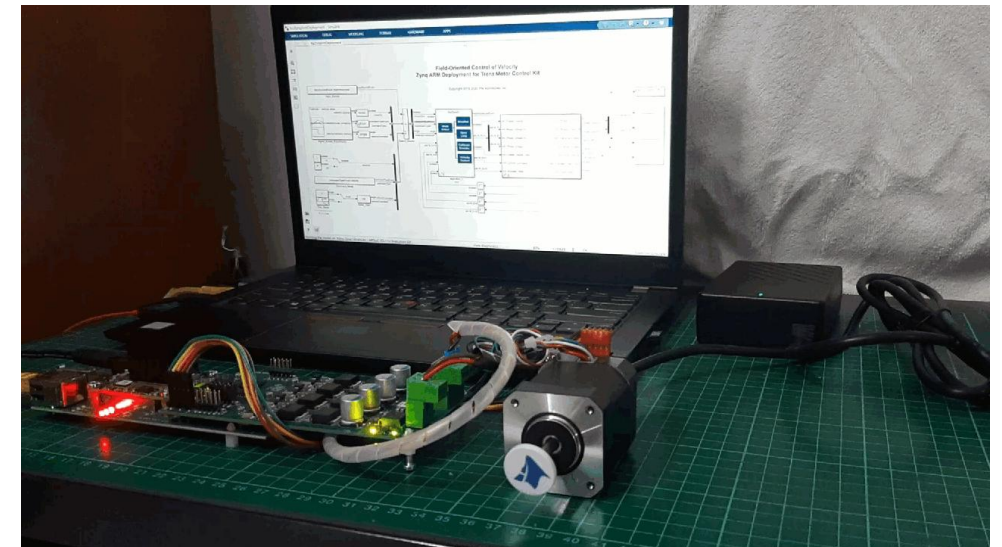
Use Simulink to Deploy Motor Control Algorithms on SoC



Trenz Electronic Motor Control Dev Kit

Main application area

- [Wireless Communication](#)
- [Radar](#)
- [Vision](#)
- [Motor Control](#)



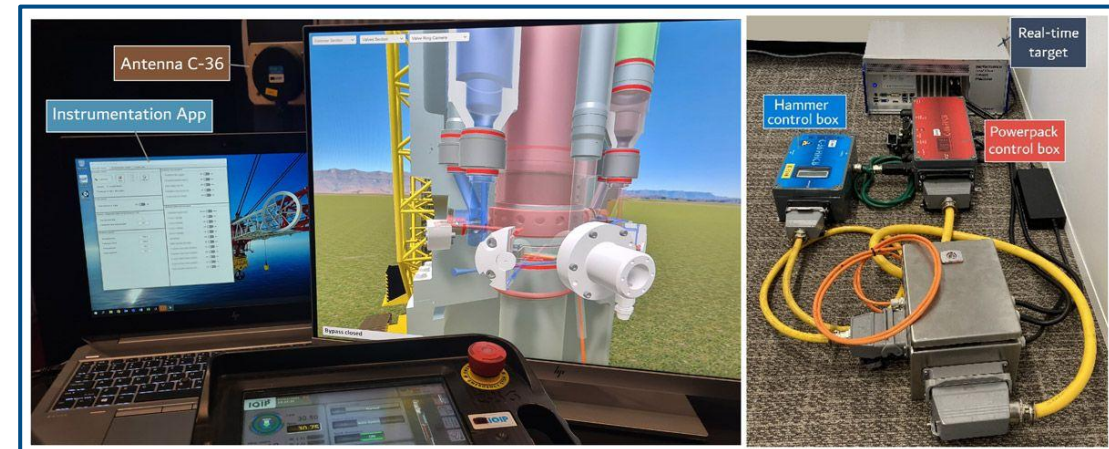
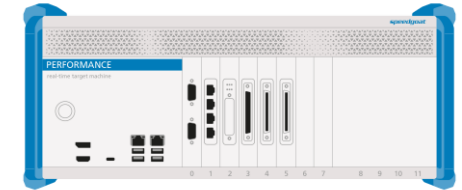
`openExample('shared_mcb_soc_c2b/ImplementFieldOrientedControlOnFPGASoCExample')`

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.

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

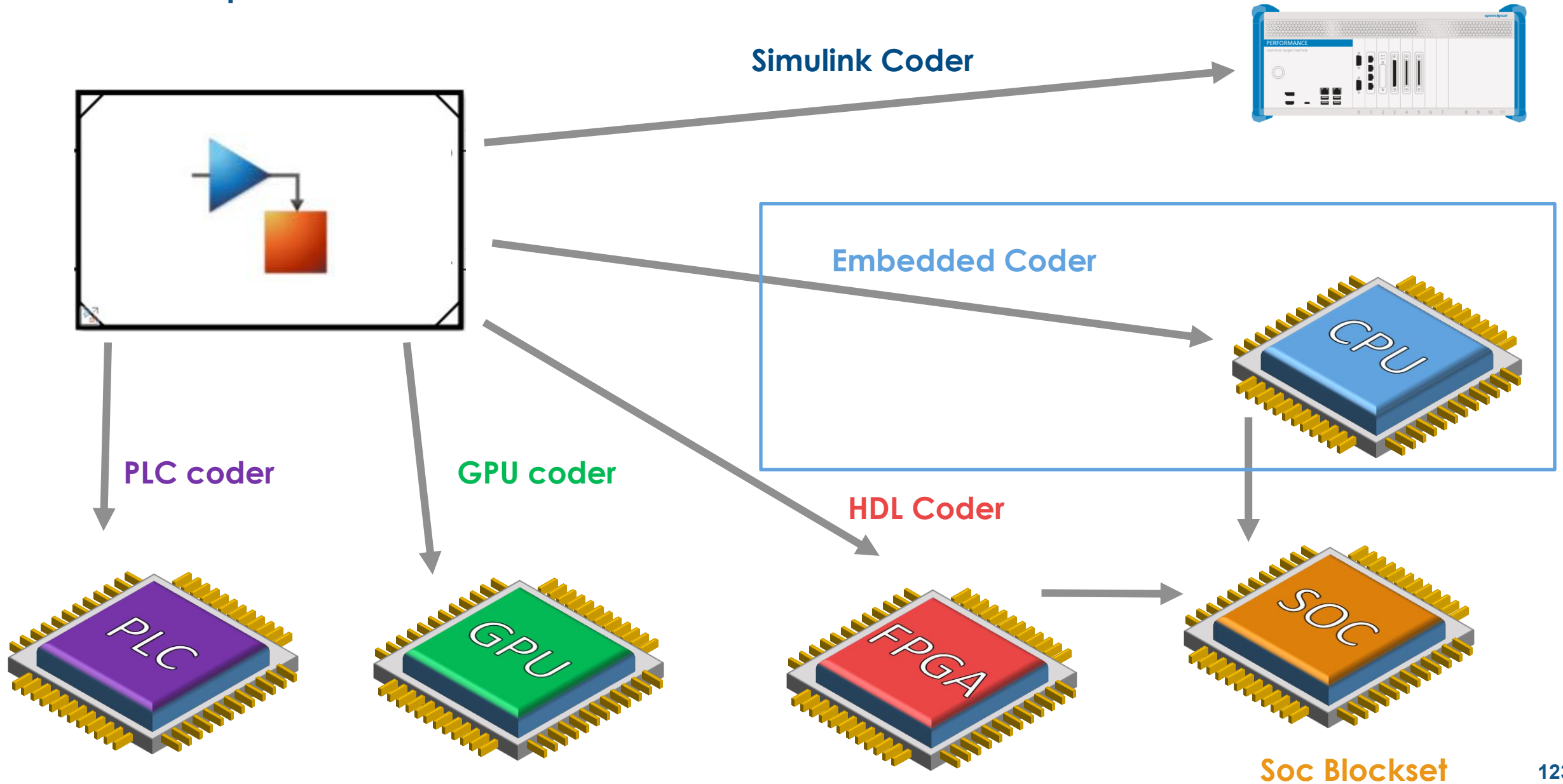


Hardware-in-the-loop setup with 3D visualization (left) and the actual control hardware connected to the real-time target (right).

“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

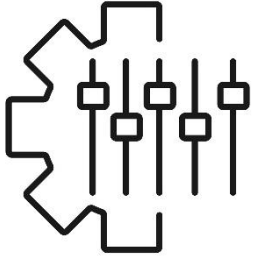
Overall picture



Default generated code

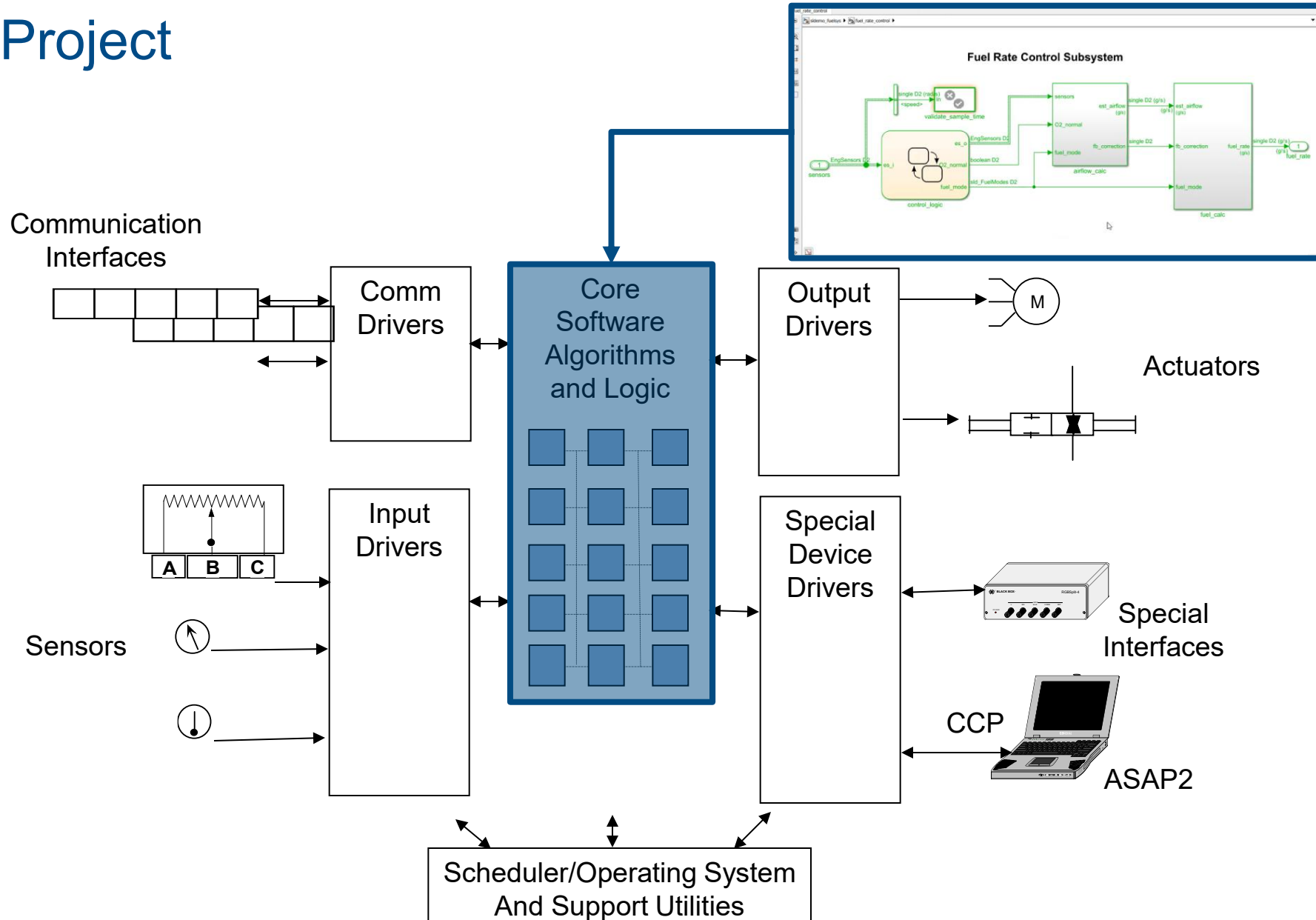
- Basic generated functions format:
 - `void modelName_initialize(void)` : to call at system initialization
 - `Void modelName_step(void)` : to call each processing step (on timer or interrupt)
 - `Void modelName_terminate(void)` : to call at system shutdown.
- Code optimization & customization:
 - None or limited

How to customize / optimize my code?



Customization

Integrate Generated Controller Code with Your Hand-Coded Software Project



Integrate Generated Controller Code with Your Hand-Coded Software Project

Model

Hand

Embedded Software Project Pseudo-Code

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

    controllerInit();

    while(1) {
    }
}
```

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

    PwmStruct = controllerStep(AdcStruct, EncoderStruct);

    writePwmCountToDriver(PwmStruct);
}
```

Basic generated code architecture & interface

- Customisation
 - Names
 - Arguments Type qualifier

Configure C Step Function Interface: RGBledsControl_Custom

Configure the generated C Step function interface, including function name and arguments.

C function prototype: `arg_LedsLevel = MyStepFunc(* arg_PushButton, arg_Slider, * arg_CurrentL...`

C Step Function Name:

Configure arguments for Step function prototype

(* invokes update diagram)

C return argument:

Port Name	Port Type	C Type Qualifier	C Identifier Name
PushButton	Inport	Pointer	arg_PushButton
Slider	Inport	Value	arg_Slider
LedsLevel	Output	Pointer	arg_LedsLevel
Clear	Output	Pointer	arg_Clear
Latch	Output	Pointer	arg_Latch
SerialBits	Output	Pointer	arg_SerialBits

Code Generation Report

Code Interface Report for RGBledsControl_Custom

Table of Contents

- [Entry-Point Functions](#)
- [Inports](#)
- [Outports](#)
- [Interface Parameters](#)
- [Data Stores](#)

Entry-Point Functions

Function: [RGBledsControl_Custom_initialize](#)

Prototype	<code>void RGBledsControl_Custom_initialize(void)</code>
Description	Initialization entry point of generated code
Timing	Must be called exactly once
Arguments	None
Return value	None
Header file	RGBledsControl_Custom.h

Function: [MyStepFunc](#)

Prototype	<code>uint8_T MyStepFunc(boolean_T *arg_PushButton, real_T arg_Slider, real_T arg_LedsLevel[3], boolean_T *arg_Clear, boolean_T *arg_Latch, boolean_T *arg_SerialBits, boolean_T *arg_Clock)</code>
Description	Output entry point of generated code

Basic generated code architecture & interface

- Customization
 - Simulink.Parameter object
 - Storage class

Property Inspector

Simulink.Parameter: Ki

NAME	VALUE
Source	Ki
Path	Base Workspace
Description	

> Design

Code

Storage Class	Struct
Struct Name	ctrl_terms
Alignment	
Identifier	

piCtrl_dataobj_start.h (5) Search

Highlighting: lines modified in last build

```

70  /* Type definition for custom storage class: Struct */
71  typedef struct ctrl_terms_tag {
72      real_T Ki; /* Referenced by
73      ioType yi_presat; /* '<S1>/Sum2' *
74      ioType yi_sat; /* '<S1>/Saturat
75      ioType yp; /* '<S1>/Proport
76  } ctrl_terms_type;
    
```

```

100  /* Gain: '<S1>/Integral Gain' incorporates:
101  * Gain: '<S1>/Sample Time'
102  */
103  tmp_0 = floor((real_T)(int16_T)((p>Ctrl_dataobj_start_P.Ts * rtb_Sum) >> 9) *
104  0.001953125 * ctrl_terms.Ki * 512.0);
105  if (rtIsNaN(tmp_0) || rtIsInf(tmp_0)) {
    
```

Code Mappings - Component Interface

Data Defaults Function Defaults Functions Inports Outports Parameters

Filter contents

Source	Storage Class
External Parameters (2)	
Ki	Struct
Kp	Define
Model Parameter Arguments (0)	
Model Parameters (0)	

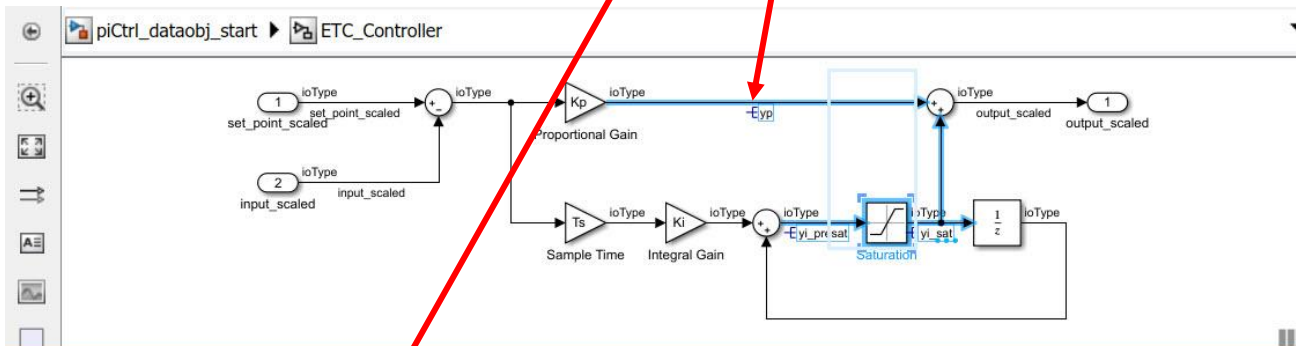
Basic generated code architecture & interface

- Customization
 - Simulink.Signal object
 - Storage class

Simulink.Signal: yp

Storage class: Struct

StructName: ctrl_terms



Model Data Editor

Inports/Outports | Signals | Data Stores | States | Parameters

Design

Source	Name	Data Type	Min	Max	Dimension	Complexity	Sample Time	Unit	Resolve
Proportional Gain	yp	Inherit: Same as in...	-Inf	Inf	1	real	-1 [0.01 0]		<input checked="" type="checkbox"/>
Sample Time		Inherit: Same as in...	-Inf	Inf	1	real	-1 [0.01 0]		<input type="checkbox"/>
Saturation	yi_sat	Inherit: Same as in...	-Inf	Inf	1	real	-1 [0.01 0]		<input checked="" type="checkbox"/>
set_point_scaled	set_point_scaled	Inherit: auto ioType	-Inf	Inf	-1	auto	real -1 [0.01 0]	inhe...	<input type="checkbox"/>
Sum		Inherit: Inherit via ...	-Inf	Inf	1	real	-1 [0.01 0]		<input type="checkbox"/>
Sum1	output_scaled	Inherit: Inherit via ...	-Inf	Inf	1	real	-1 [0.01 0]		<input type="checkbox"/>
Sum2	yi_presat	Inherit: Inherit via ...	-Inf	Inf	1	real	-1 [0.01 0]		<input checked="" type="checkbox"/>

```
piCtrl_dataobj_start.c (17) Search
Highlighting: lines modified in last build
21 /* Definition for custom storage class
22 ctrl_terms_type ctrl_terms;
```

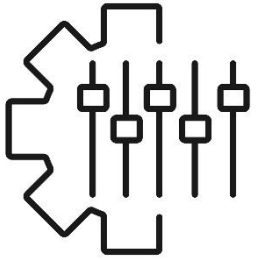
```
piCtrl_dataobj_start.h (10) Search
Highlighting: lines modified in last build
76 /* Type definition for custom storage class: Struct */
77 typedef struct ctrl_terms_tag {
78     ioType yp; /* '<S1>/Proport
79     ioType yi_presat; /* '<S1>/Sum2' *
80     ioType yi_sat; /* '<S1>/Saturat
81 } ctrl_terms_type;
```

```
piCtrl_dataobj_start.c (17) Search
Highlighting: lines modified in last build
124 piCtrl_dataobj_start_Y.ETC_output_scaled = (real_T)(int16_T)(ctrl_terms.yp +
125     ctrl_terms.yi_sat) * 0.001953125;
126
127 /* Update for UnitDelay: '<S1>/Unit Delay' */
128 piCtrl_dataobj_start_DW.UnitDelay_DSTATE = ctrl_terms.yi_sat;
```

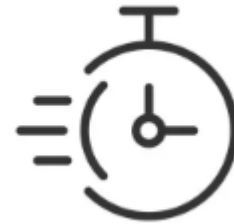
Basic generated code architecture & interface

- 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 Generation Advisor - soc_pmsm_singlecpu_foc

File Edit Run Settings Help

Find:

- Code Generation Advisor
 - Check model configuration settings against code generation objectives
 - Check for blocks not recommended for MISRA C:2012 guidelines
 - Check for blocks not recommended for C/C++ code generation
 - Check for unsupported block names

Check model configuration settings against code generation objectives

Analysis

Check model configuration settings against the code generation objectives. Successfully passing this check may take multiple iterations since a change to one option can impact other options.

Run This Check

Result: **Warning**

Set Objectives - Code Generation Advisor

Description

Select and prioritize your code generation objectives. You can add custom objectives, for details, see the documentation.

Available objectives	Selected objectives - prioritized
Execution efficiency	
ROM efficiency	
RAM efficiency	
Traceability	
Safety precaution	
Debugging	
MISRA C:2012 guidelines	
Polyspace	

OK Cancel Help

Selected Objectives: [Debugging](#), [MISRA C:2012 guidelines](#)

The following parameter values are not optimized for the selected objectives.

To automatically fix the warning, click the 'Modify Parameters' button and then rerun the check. To manually fix the warning, click the link to open the Configuration Parameters dialog box, and manually apply the recommended value.

Parameter	Current Value	Recommended Value
Prevent unreachable condition expression in if statement	off	on
Prefer multiplications by powers of two with signed bitwise shifts	on	off
Prefer right shifts on signed integers	on	off
Prefer generation of default cases for switch statements if unreachable	on	off
Prefer division for fixed-point net slope computation	off	on
Prefer function inlining between user functions	Speed	Memory

Modify Parameters

Code Replacement & Legacy Code

Configuration Parameters: Configuration

- Solver
- Data Import/Export
- Math and Data Types
- ▶ Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- ▼ Code Generation
 - Optimization
 - Report
 - Comments
 - Symbols
 - Custom Code
 - Interface

Software environment

Code replacement library: ARM Cortex-A

Shared code placement+ Auto

Support: floating-p absolute t

Code interface

Code interface packagi

Remove error statu

Data exchange interface

Code Generation Report

Find: Match Case

Contents

- [Summary](#)
- [Subsystem Report](#)
- [Code Interface Report](#)
- [Traceability Report](#)
- [Static Code Metrics Report](#)
- [Code Replacements Report](#)
- [Coder Assumptions](#)

Code replacements in RGBledsControl_Custom

Code replacements for library 'ARM Cortex-A'. The library comprises:

- ARM Cortex-A
 - crl_table_ne10
- GCC_ARM_NEON_mw_lib
 - GCC_ARM_NEON_mw_lib

To see the replacements and misses in the Code Replacement Viewer, look [here](#).

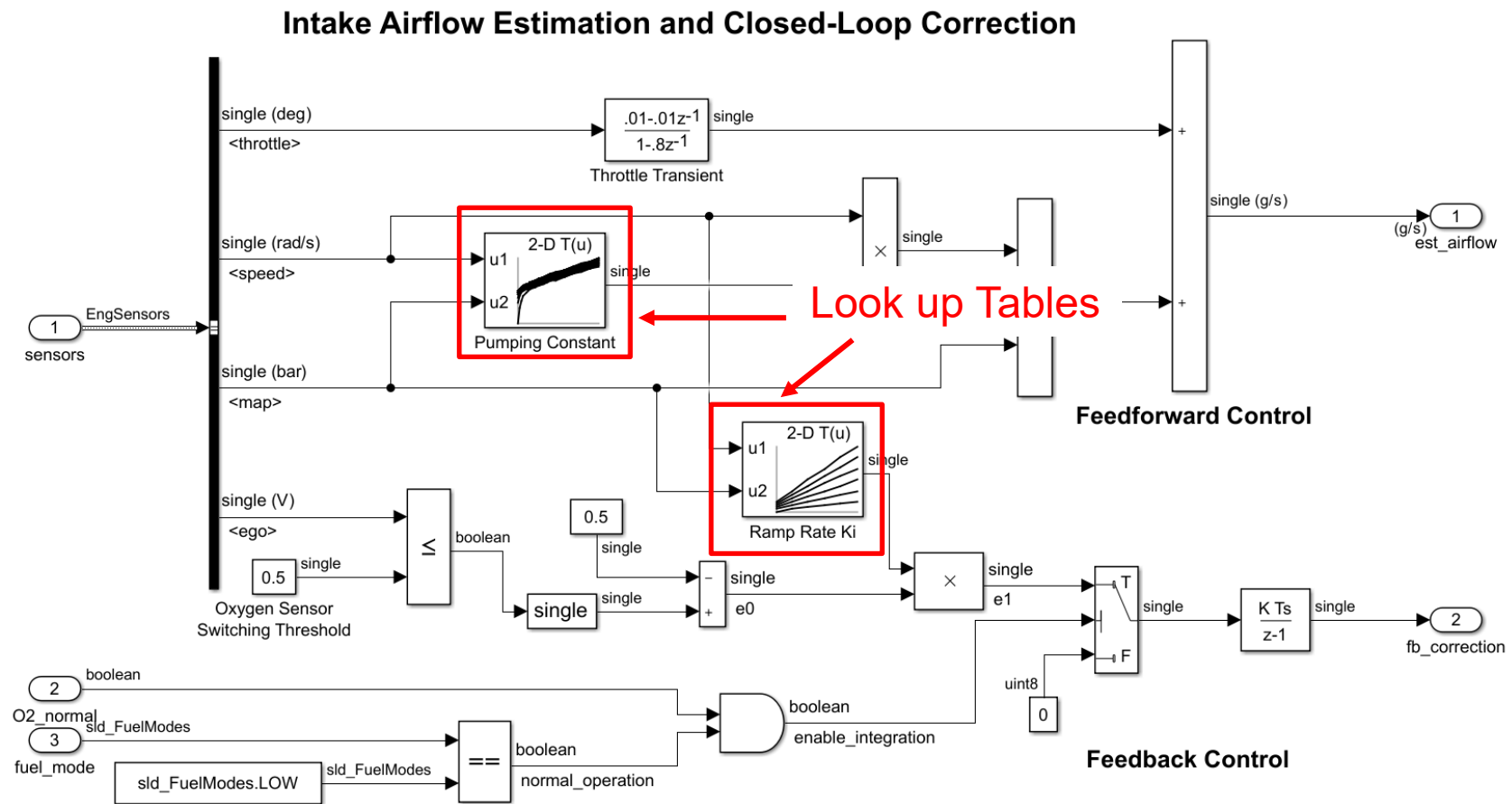
1. Function replacements in RGBledsControl_Custom [hide]

The following table provides a mapping from the functions used from the selected Code Replacement Library to the blocks in the model that triggered the replacement.

Function	Block
ceil	<S5>/1-D Lookup Table
memset	Unknown

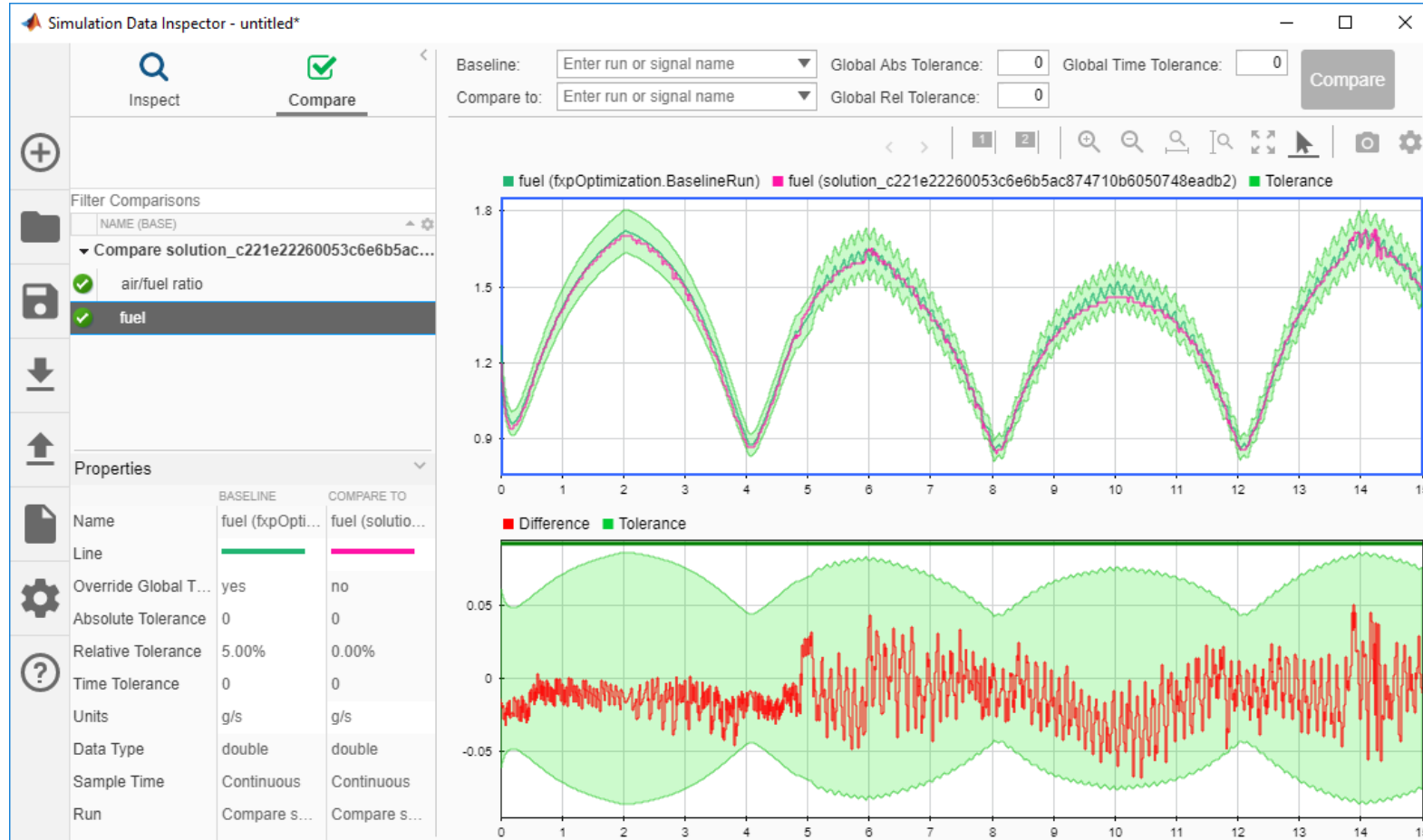
Memory usage reduction & data quantization

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



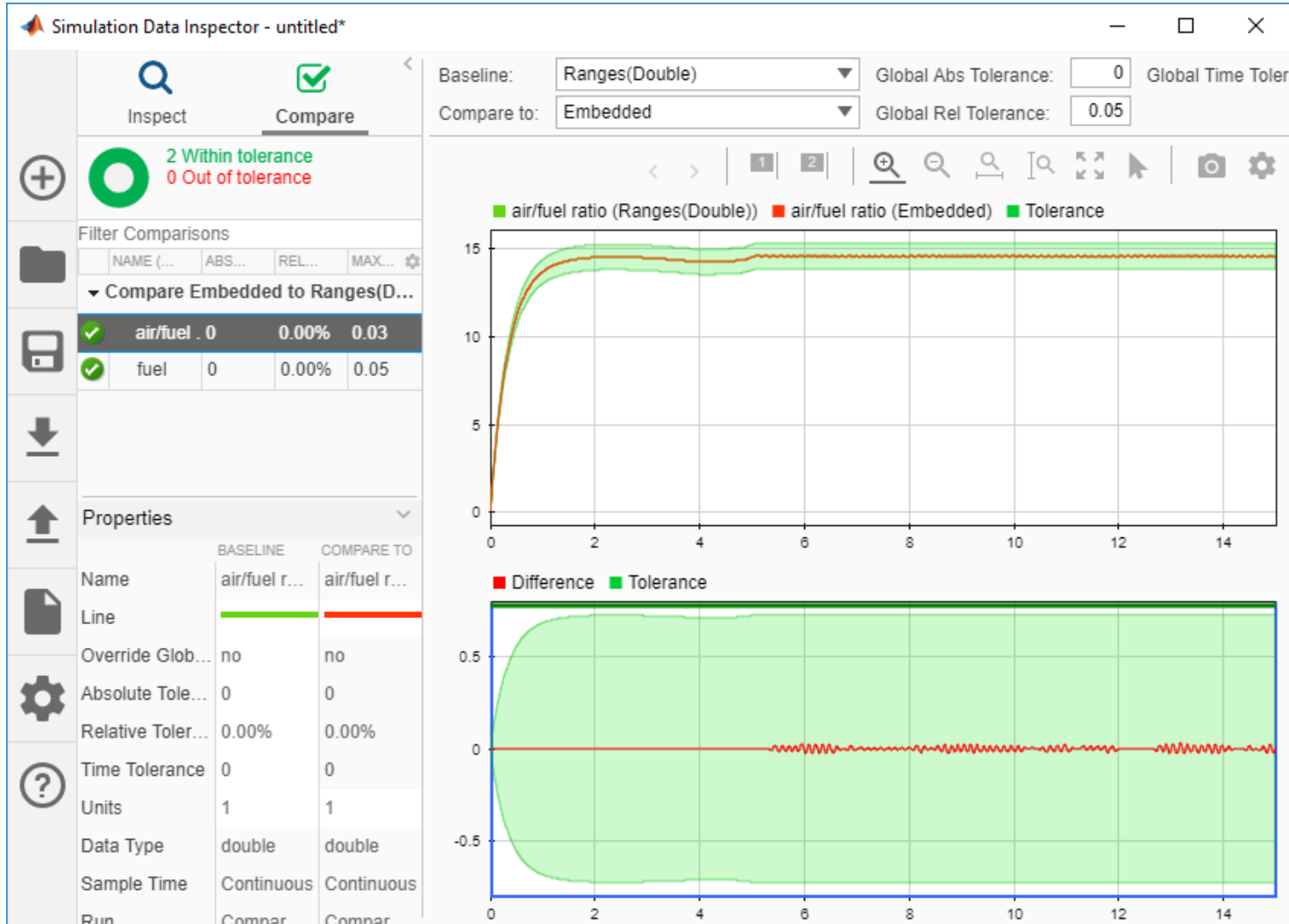
Single Precision Conversion Result

Data Type	Memory Usage (Bytes)
Double	12048
Single	6024



Fixed Point Conversion Result

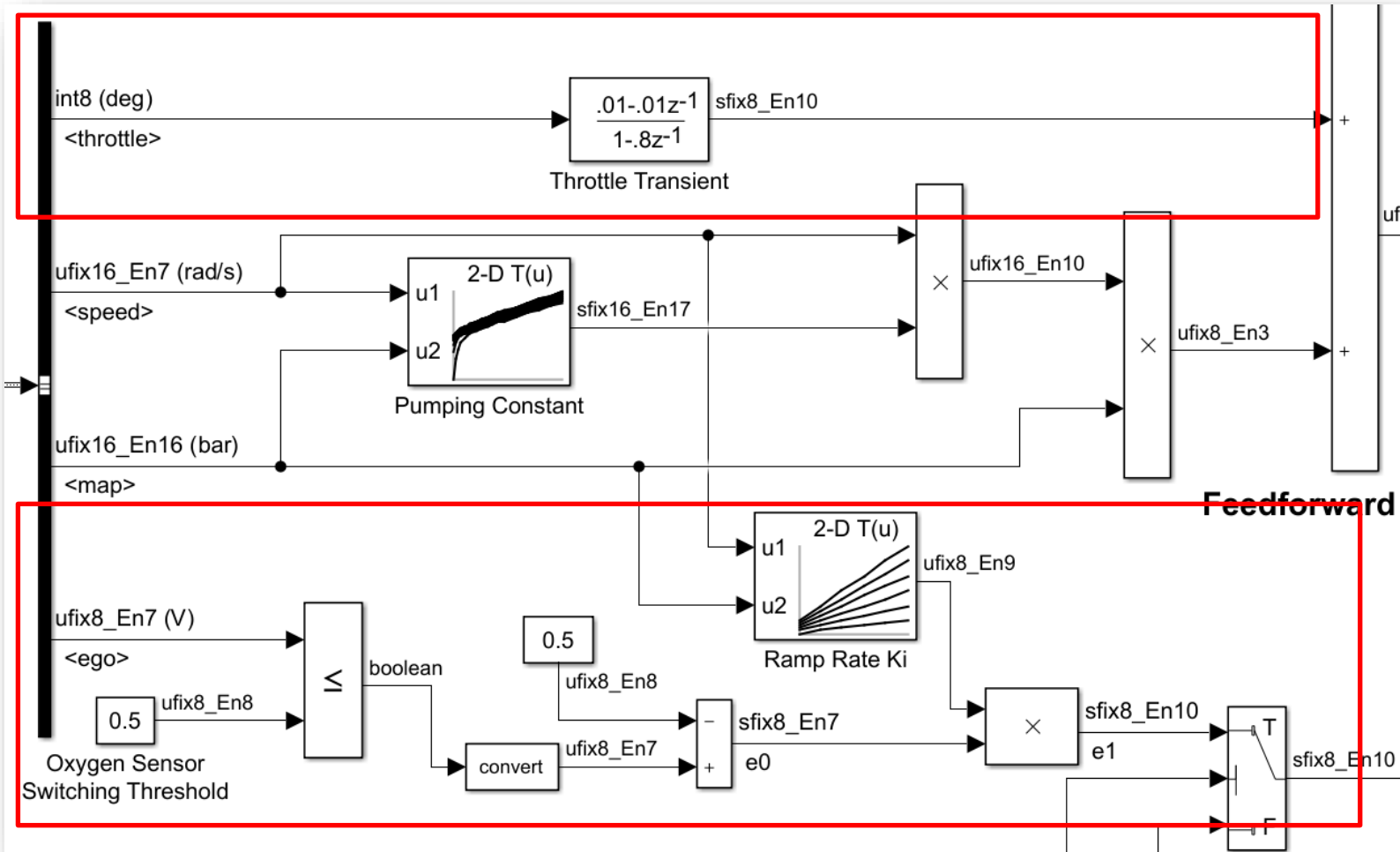
Data Type	Memory Usage (Bytes)
Double	12048
Single	6024
16 bit Fixed	3012



* 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

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

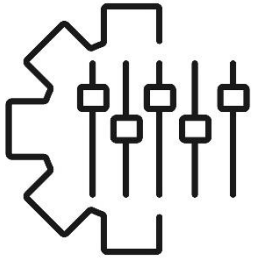
Table 2 ROM and RAM comparison between a floating-point hand code and auto code.

	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

Code Interface Report for RGBledsControl

Table of Contents

- [Entry-Point Functions](#)
- [Inports](#)
- [Outports](#)
- [Interface Parameters](#)
- [Data Stores](#)

Entry-Point Functions

Function: [RGBledsControl_initialize](#)

Prototype	<code>void RGBledsControl_initialize(void)</code>
Description	Initialization entry point of generated code
Timing	Must be called exactly once
Arguments	None
Return value	None
Header file	RGBledsControl.h

Function: [RGBledsControl_step](#)

Prototype	<code>void RGBledsControl_step(void)</code>
-----------	---

Generated Code

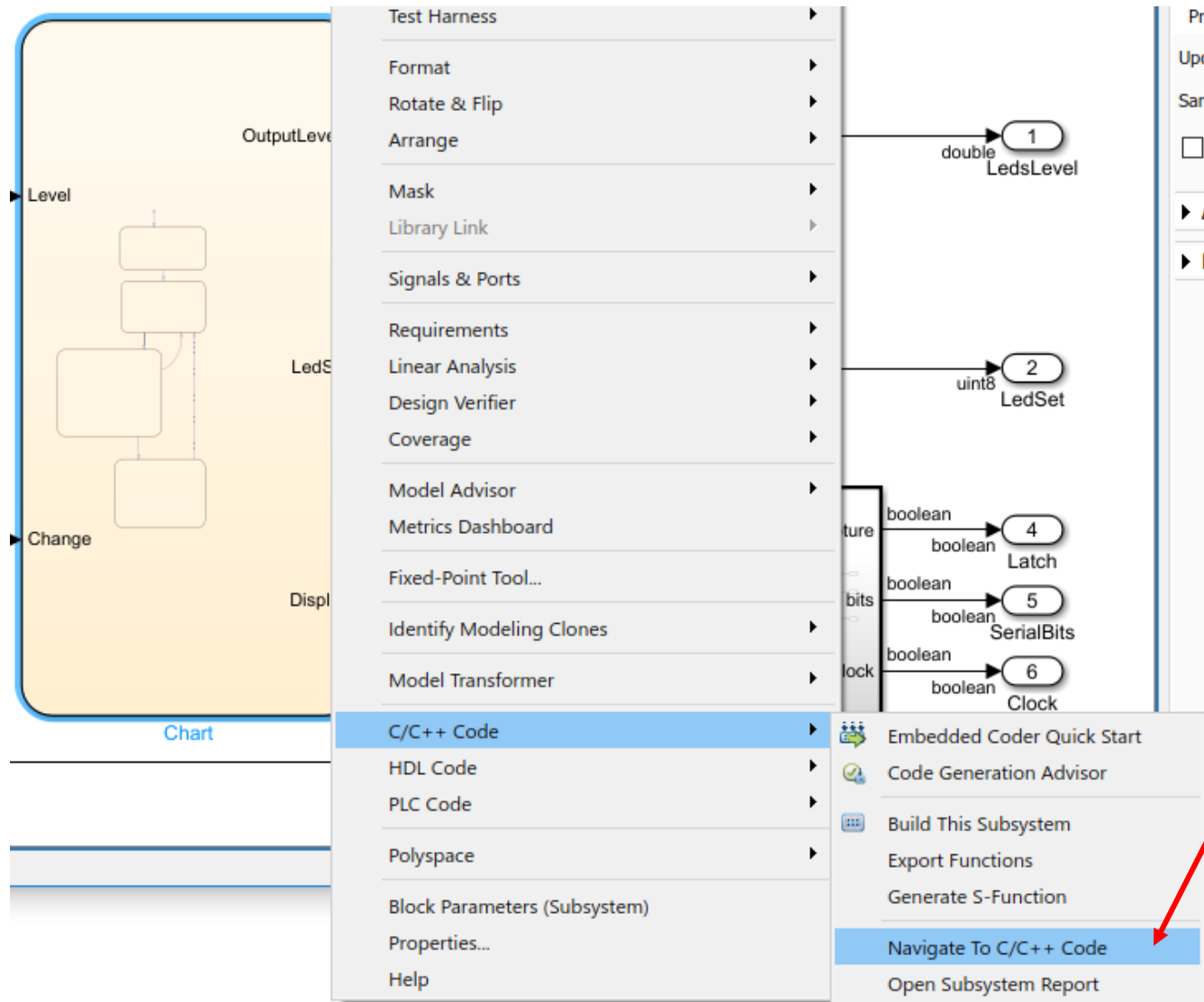
- [-] Main file
 - [ert_main.c](#)
- [-] Model files
 - [RGBledsControl.c](#)
 - [RGBledsControl.h](#)
 - [RGBledsControl_private.h](#)
 - [RGBledsControl_types.h](#)
- [-] Data files
 - [RGBledsControl_da](#)

```

17 #include "RGBledsControl_private.h"
18
19 /* Named constants for Chart: '<Root>/Chart' */
20 #define RGBledsContr_IN_NO_ACTIVE_CHILD ((uint8_T)
21 #define RGBledsContr_IN_ShowCurrentLed ((uint8_T)
    
```

- 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



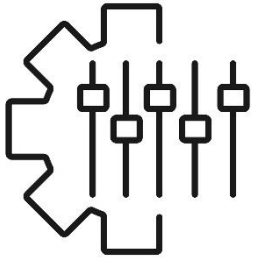
- You can navigate from model to code by right clicking on a block and selecting: « C/C++ Code > Navigate To C/C++ Code ».
- You are then re-directed on the corresponding line in code report.

Tracability & Code generation report

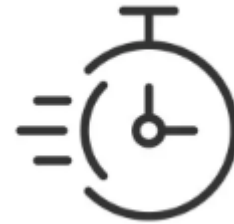
The screenshot shows the 'Code Generation Report' window. On the left, the 'Content' sidebar lists various reports, with 'Traceability Report' selected. The main pane displays the C code for 'TopModelCode.c', showing a model step function with local variables and discrete logic for a pulse generator. Below the code, a block diagram visualizes the model structure, including three counter blocks (CounterA, CounterB, CounterC) and their associated scopes. A 'Property Inspector' on the right shows parameters for 'TopModelCode', such as ModelVersion (6.2) and LastModifiedDate (Thu Mar 10 11:29:46 2022).

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

* Need Simulink Report Generator



Customization



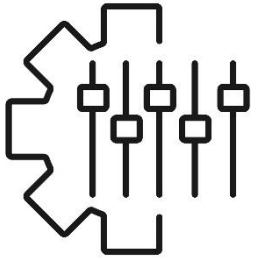
Optimization



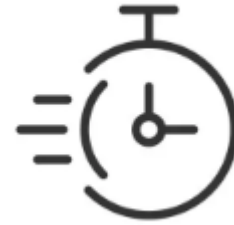
Report



Code
Testing / Check



Customization



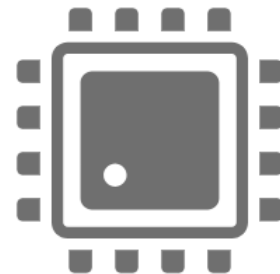
Optimization



Report

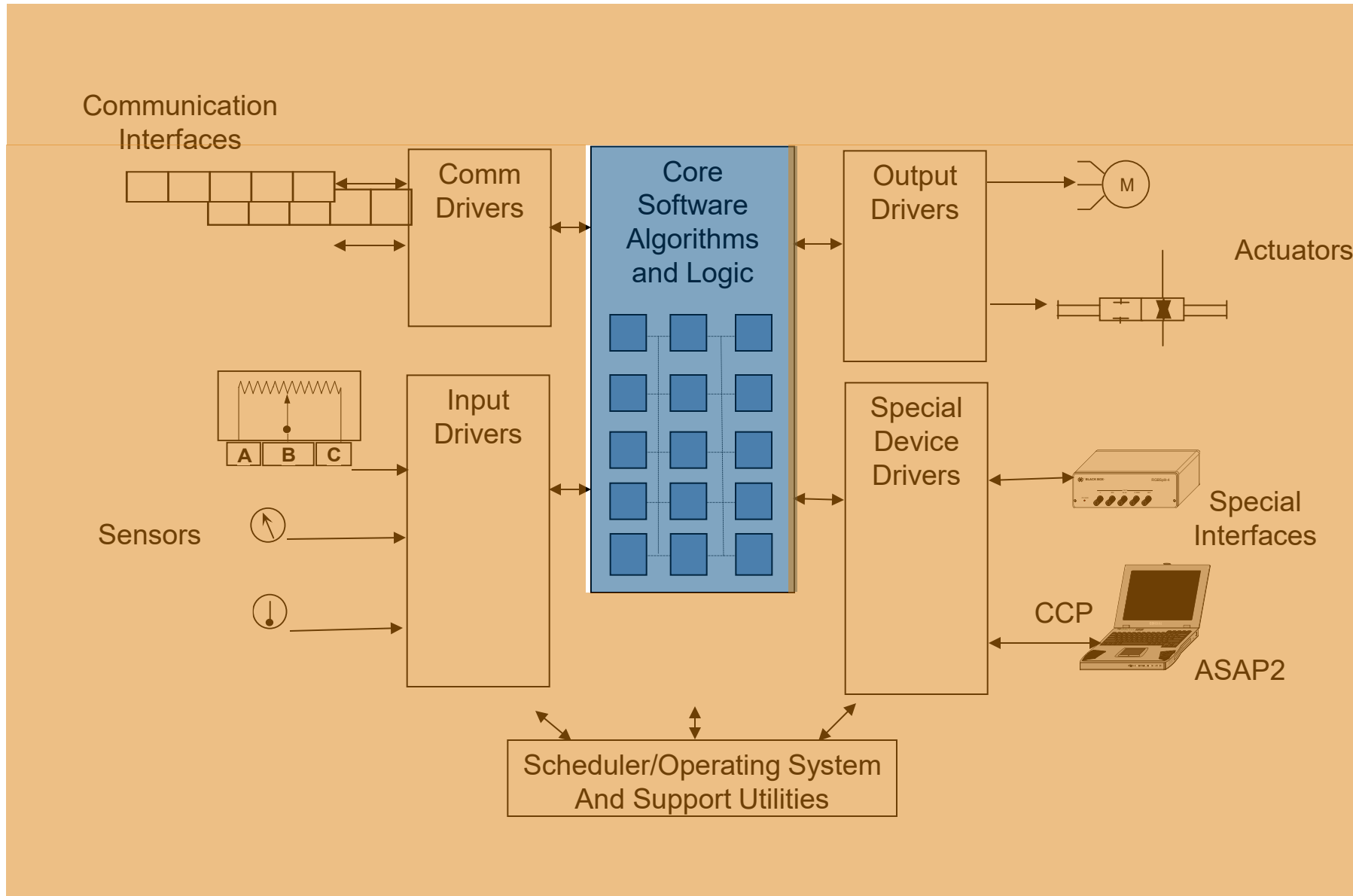


Code
Testing / Check

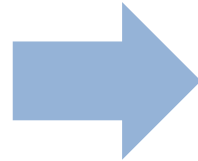
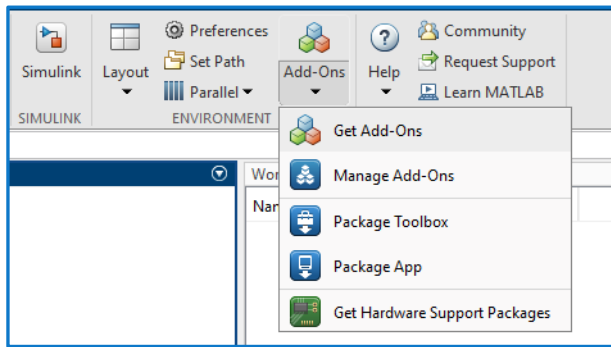


Code deployment

Simple Embedded Software Architecture



Custom Target, Hardware Support Package or Specialized toolbox

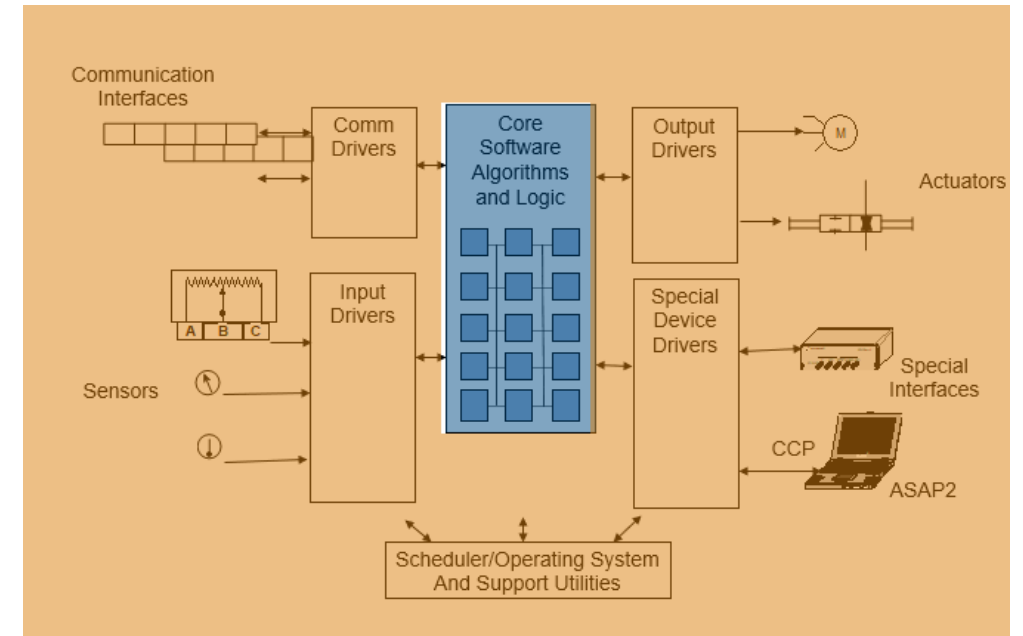


Simulink Support Package for Arduino Hardware by MathWorks

Simulink Team **STAFF**

Run models on **Arduino** boards

				Author
				Partner



[C2000 Microcontroller Blockset](#) / [STM32 Microcontroller Blockset](#) / [Raspberry Pi Blockset](#) / [Hardware Support Packages](#)

Custom Target, Hardware Support Package or Specialized toolbox

Simulink Library Browser

Embedded Coder Support Package for STMicroelectronics STM32 Processors/STM32G4xx Based Boards

- Data Acquisition Toolbox
- DDS Blockset
- Deep Learning Toolbox
- DSP System Toolbox
- DSP System Toolbox HDL Support
- Embedded Coder
 - Embedded Coder Support Package for
 - Embedded Coder Support Package for
 - Embedded Coder Support Package for STM32F3xx Based Boards
 - STM32F4-Discovery
 - STM32F4xx Based Boards
 - STM32F746G-Discovery
 - STM32F769I-Discovery
 - STM32F7xx Based Boards
 - STM32G4xx Based Boards
 - STM32H7xx Based Boards (Dual-co
 - STM32H7xx Based Boards (Single-c
 - STM32L475VG-Discovery (B-L475E
 - STM32L4xx Based Boards
 - STM32L5xx Based Boards
 - STM32U5xx Based Boards
 - STM32WBxx Based Boards
- Utilities
- Fixed-Point Designer

Grid of blocks including: ADC1, COMP1, CORDIC, Digital Port Write, FDCAN, Encoder, Hardware Interrupt, etc.

Block Parameters: Digital Port Write

Digital Port Write

Write logical status of pins of a GPIO port.

Parameters

Port name: GPIOA

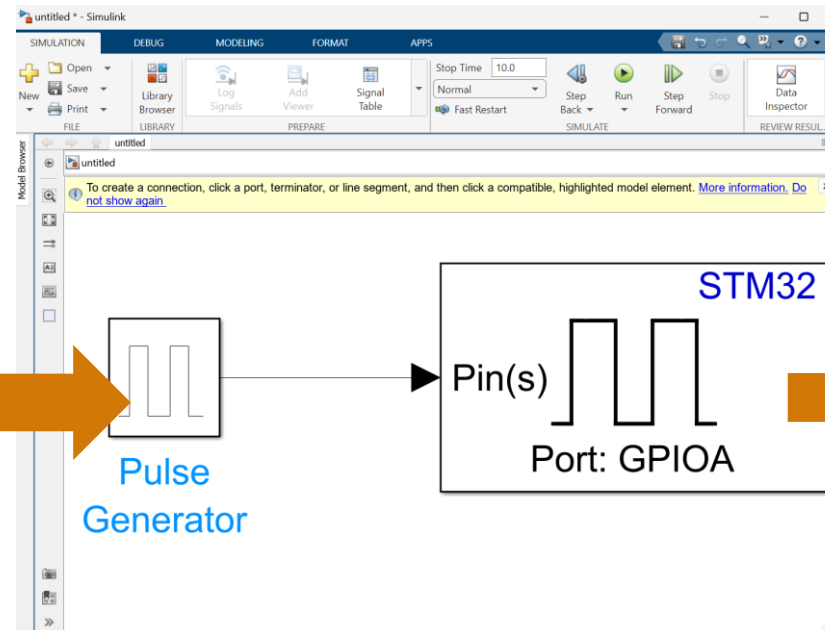
Pin number: [0]

Access port pins as array

Enable simulation port

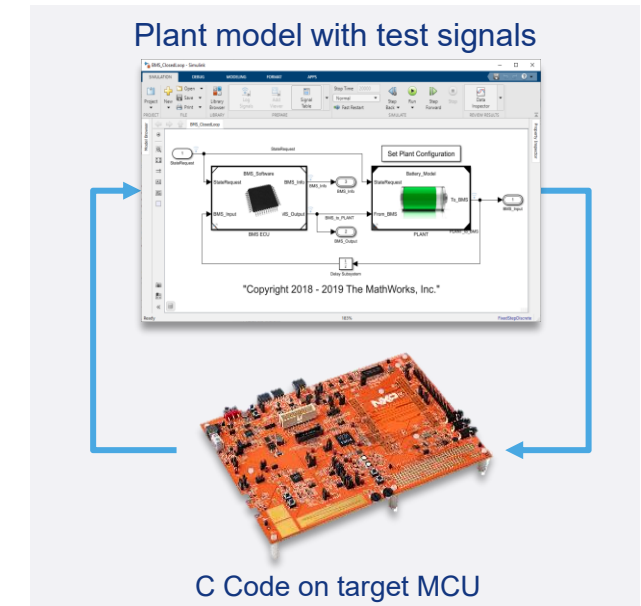
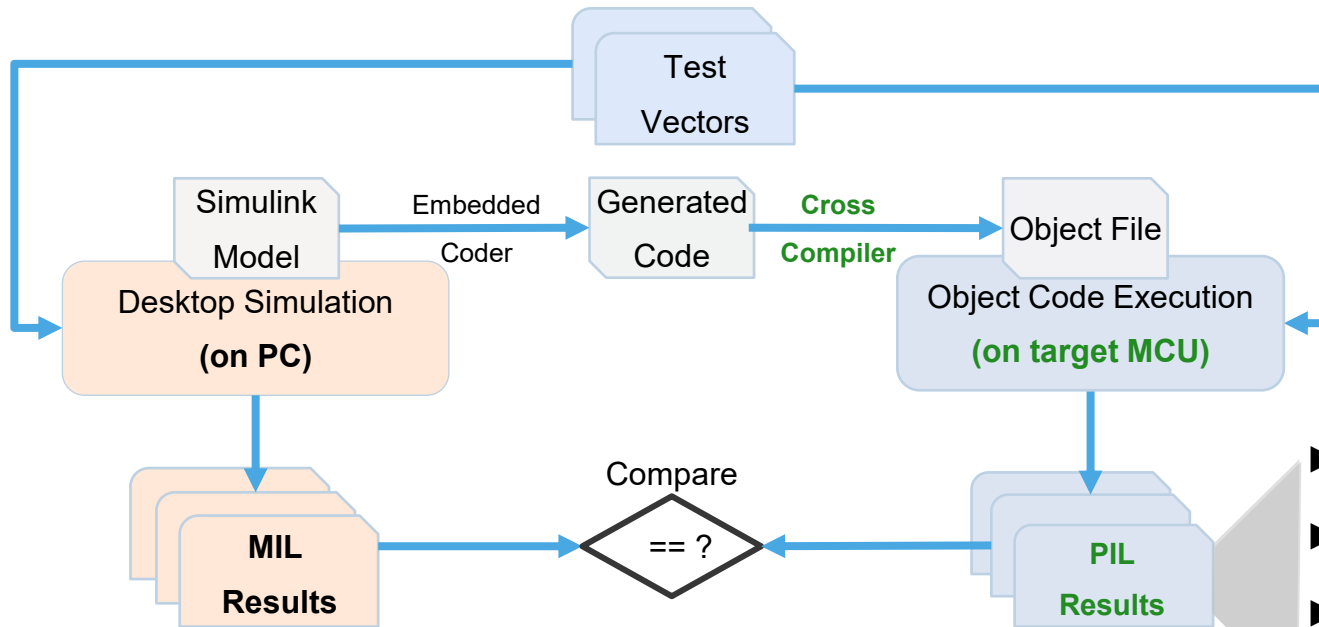
OK Cancel Help Apply

Drag & drop Peripheral Blocks



Configure Peripheral Blocks

PIL (Processor in the loop)

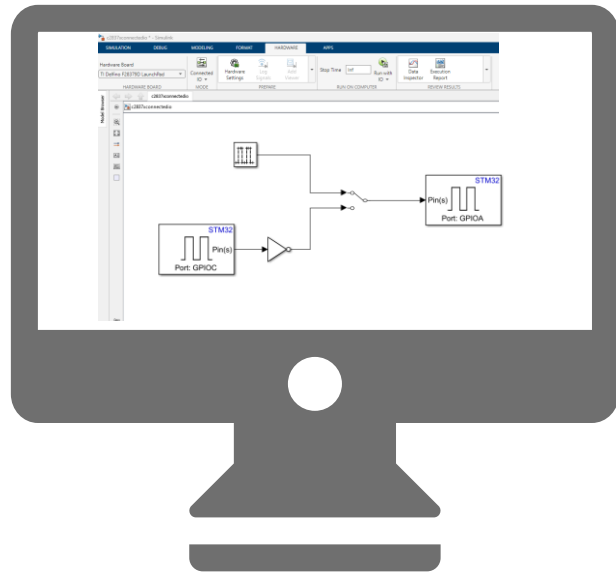
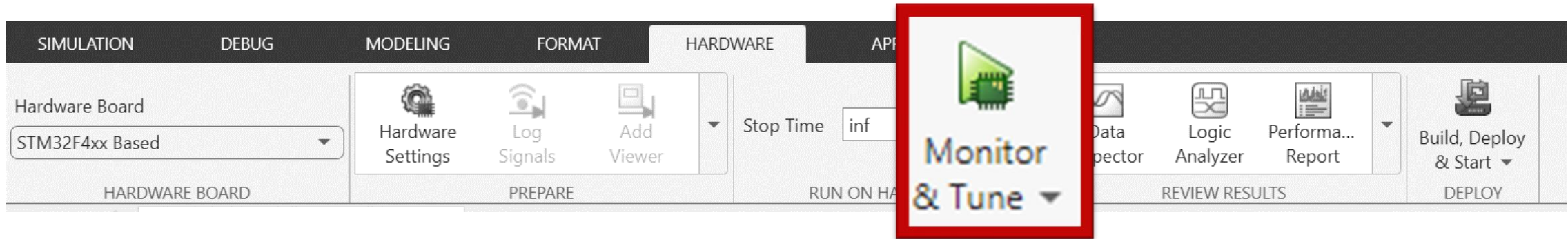


- ▶ Verify numerical equivalence
- ▶ **Profile target execution time**
- ▶ Collect on target code coverage
- ▶ Must for code replacement library

Non Real-Time Execution

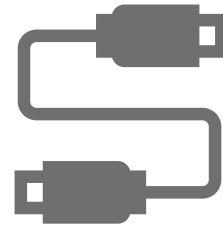
Section	Maximum Execution Time in ns	Average Execution Time in ns	Max
[+] Current_initialize	2260	2260	
Current_step [5e-05 0]	5135	5067	
Current_terminate	540	540	

Monitor and Tune (Also called External Mode)

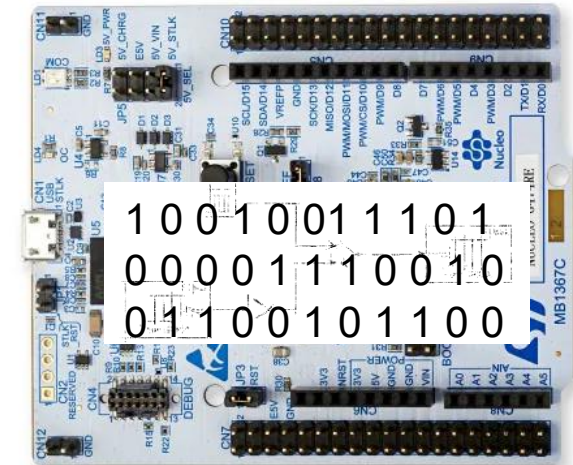


Simulink Model
(Monitoring only)

Real-Time Execution

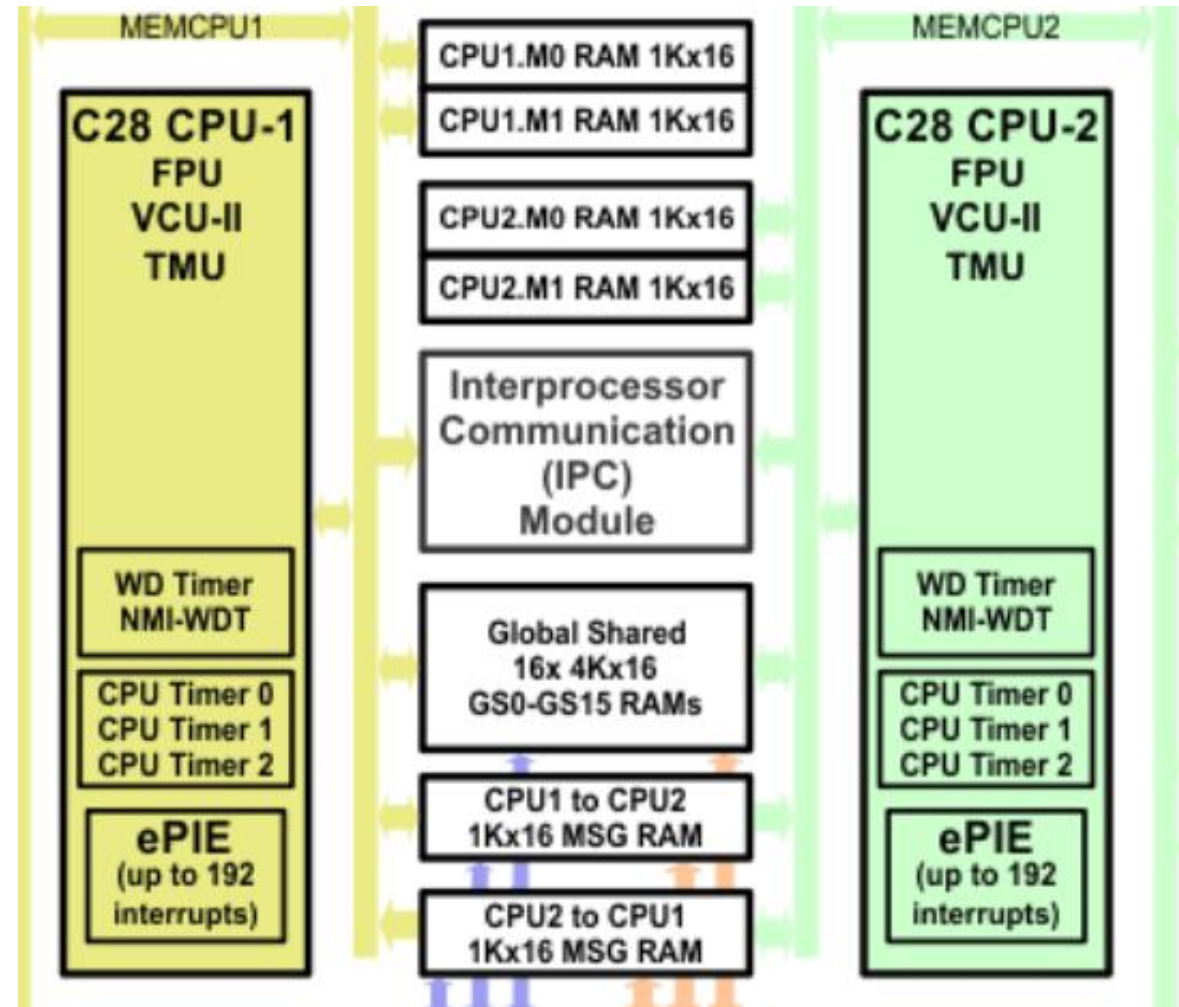


Data Exchange
over
XCP over Serial or CAN
or TCP/IP



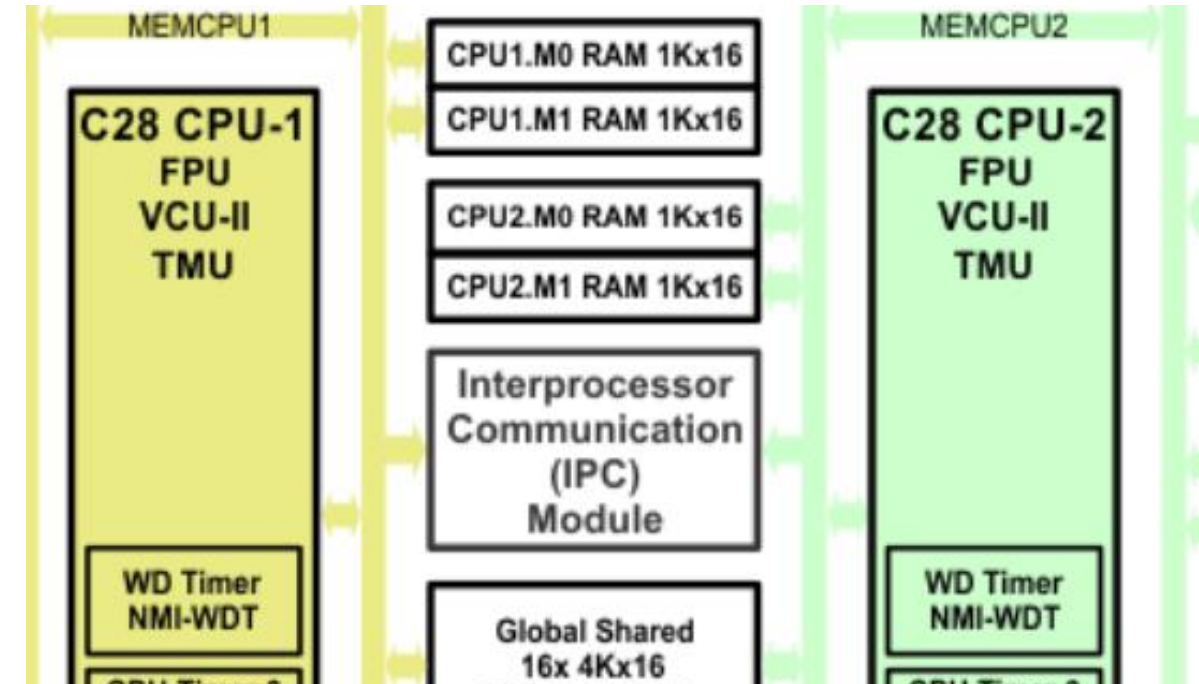
Model running on hardware
with additional code

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



Hardware Implementation

- Model Referencing
- Simulation Target
- ▶ Code Generation
- Coverage
- ▶ HDL Code Generation
- Simscape
- ▶ Simscape Multibody

▶ Device details

Feature set for selected hardware board:

Simulink or Embedded Coder Hardware

SoC Blockset

Hardware board settings

Processing Unit: None

▶ Design mapping: **None**

▶ Task profiling in: c28xCPU1

Hardware Implementation

- Model Referencing
- Simulation Target
- ▶ Code Generation
- Coverage
- ▶ HDL Code Generation
- Simscape
- ▶ Simscape Multibody

▶ Device details

Feature set for selected hardware board:

Simulink or Embedded Coder Hardware

SoC Blockset

Hardware board settings

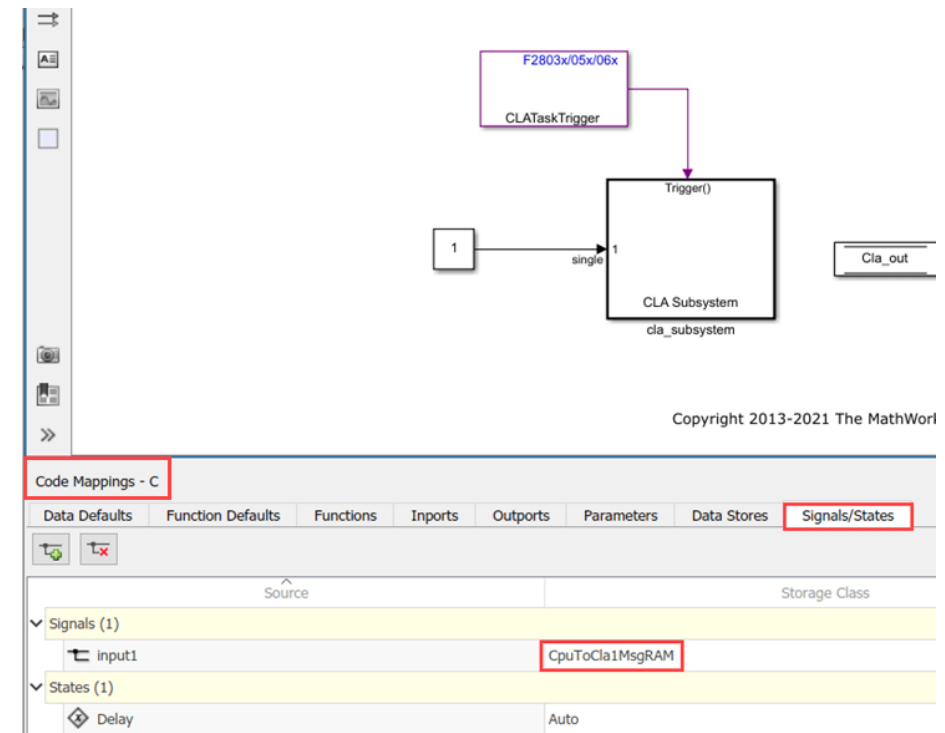
Processing Unit: None

▶ Design mapping: **None**

▶ Task profiling in: **c28xCPU2**

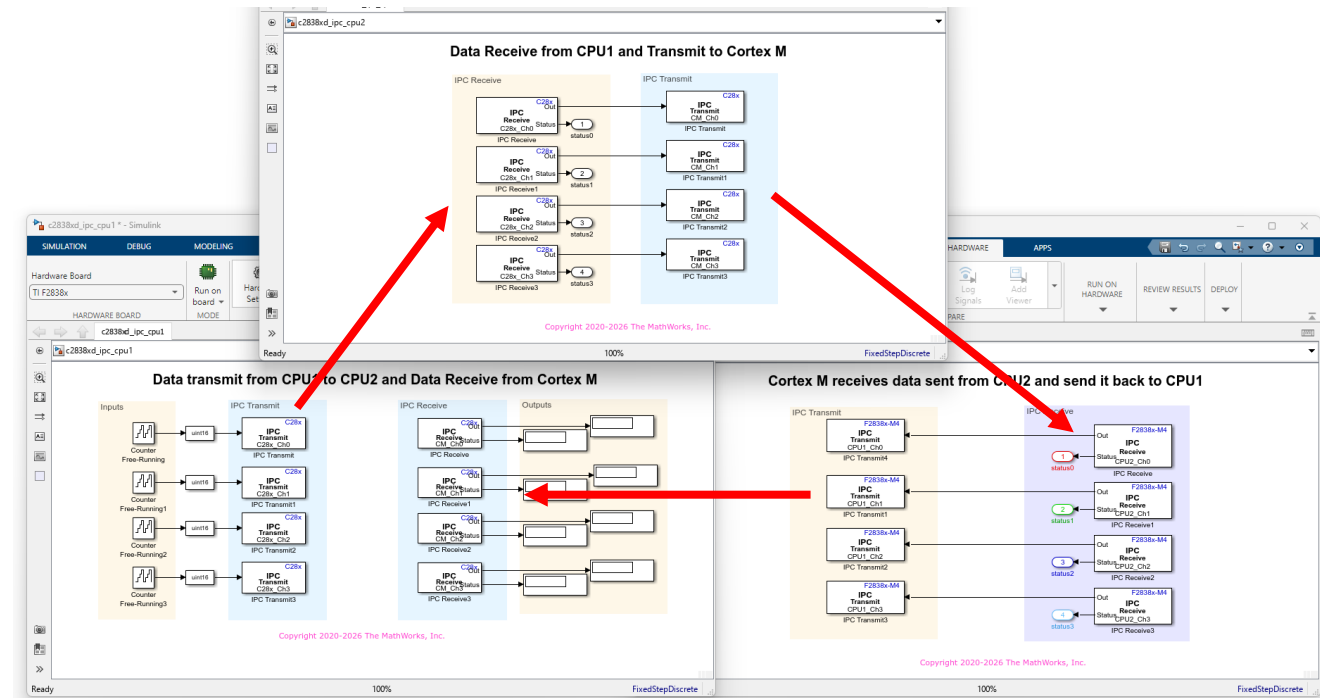
Embedded Coder workflow and configuration

- **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



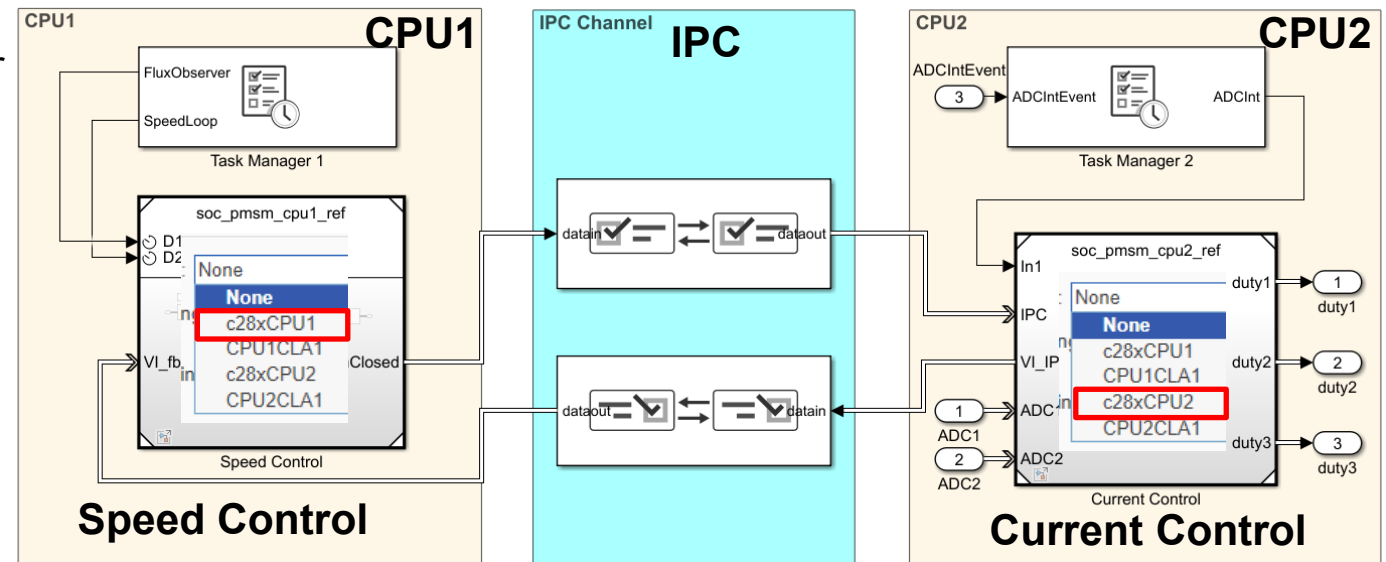
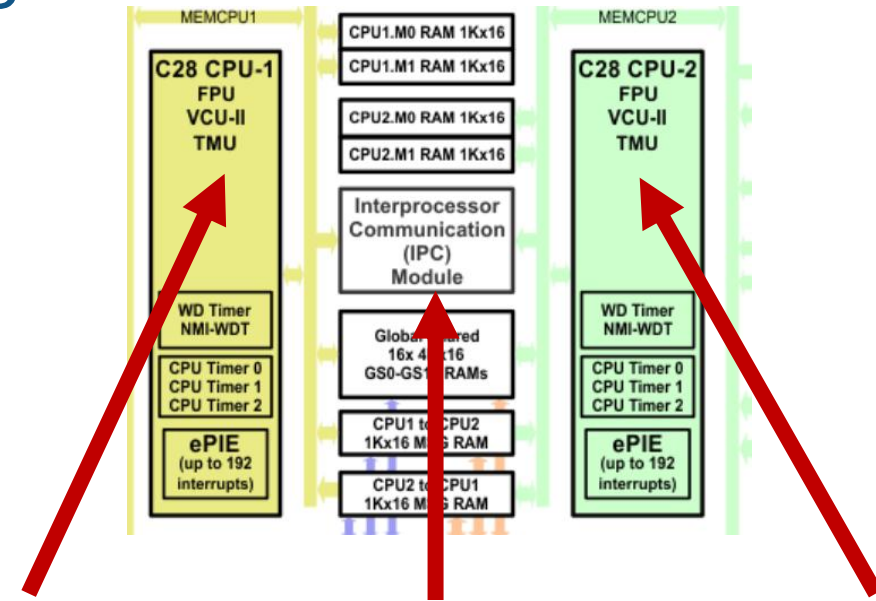
Embedded Coder workflow and configuration

- **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

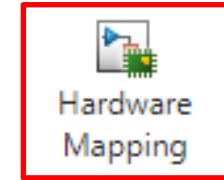


Embedded Coder workflow and configuration

- 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

The screenshot shows the 'Hardware Mapping' tool interface. The 'Mapping Browser' on the left lists tasks and peripherals. The 'ADC Read' task is selected, and the 'ADC Interrupt' configuration is shown on the right. The configuration includes a dropdown for the event (ADC1_isr) and a list of ADC channels (ADCA1_isr to ADCB4_isr). Below this, a detailed configuration panel for the selected task is shown, including fields for Module, Start of conversion, Resolution, Conversion channel, SOCx acquisition window, SOCx trigger source, ADCINT will trigger SOCx, and Interrupt Selection.

Task	Event
SpeedLoop	Internal
ADCInterrupt	ADCB1_isr

Mapping Browser

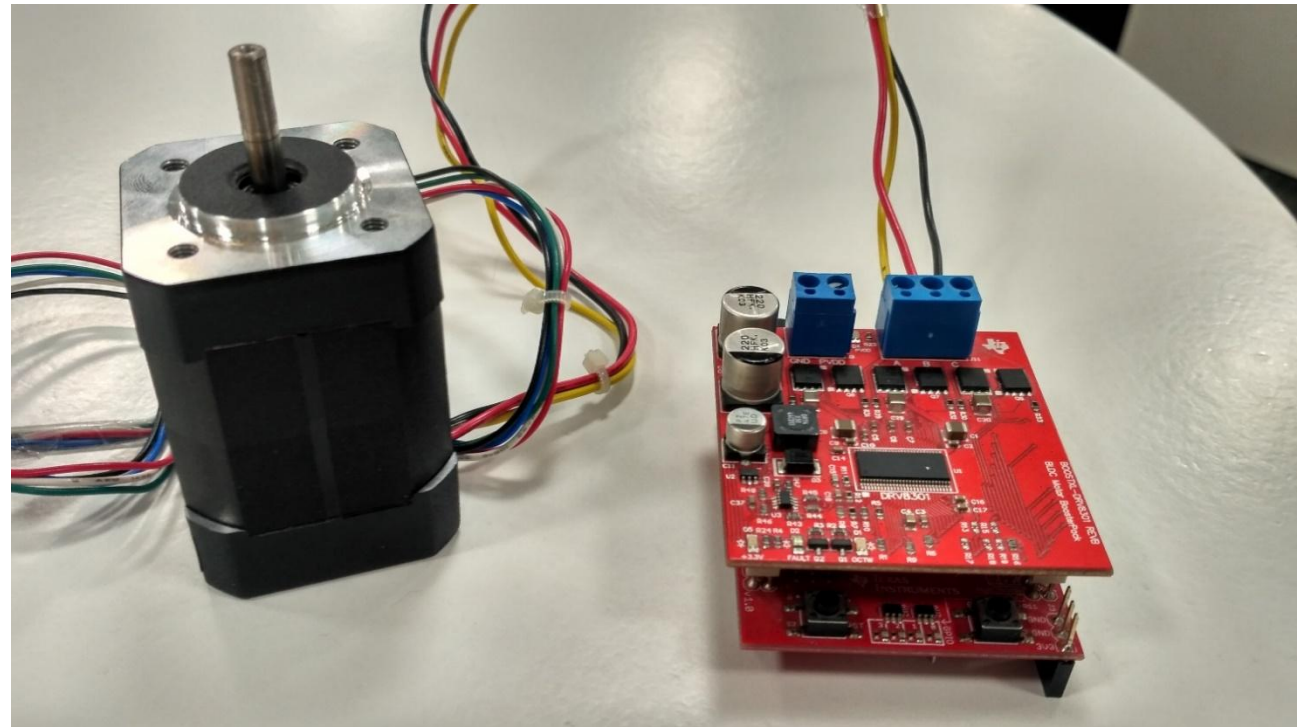
- Tasks
 - c28xCPU1
- Peripherals
 - ADC Read
 - soc_pmsm_singlecpu_ref/Current Control/ADC Read1
 - soc_pmsm_singlecpu_ref/Current Control/ADC Read2
 - PWM Write
 - soc_pmsm_singlecpu_ref/Current Control/PWM Write
 - soc_pmsm_singlecpu_ref/Current Control/PWM Write1
 - soc_pmsm_singlecpu_ref/Current Control/PWM Write2

Configuration Panel:

- Module: B
- Start of conversion: SOC 0
- Resolution: 12-bit (Single-ended input)
- Conversion channel: ADCIN2
- SOCx acquisition window (cycles): 13
- SOCx trigger source: ePWM1 ADCSOCA
- ADCINT will trigger SOCx: No ADCINT
- Enable interrupt at EOC
- Interrupt Selection: ADCINT1
- Interrupt continuous mode

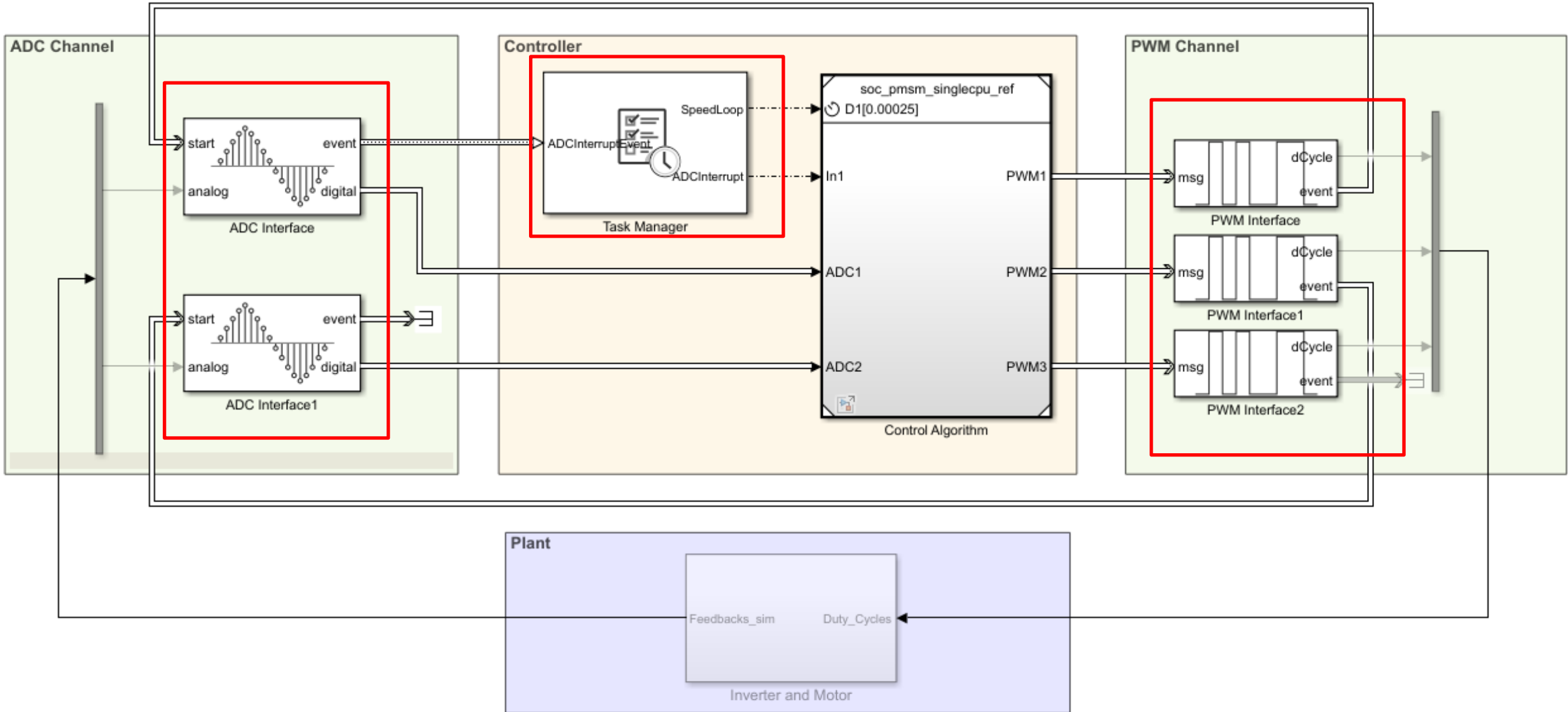
Workshop Example

- 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?



Field Oriented Control In Single CPU

Single processor model



Task Partition and Management with Task Manager Block

- Model task execution timing impact

Block Parameters: Task Manager

task starts and stops from a recorded file. Otherwise, task duration in simulation is determined via dialog, input port, or summary statistics from a recorded file, and the task start times are determined by settings in the Main tab.

Enable task simulation

Task simulation

- Task1
- Task2
- Task3

Add Delete

Task1 properties

Main Simulation

Name: Task1

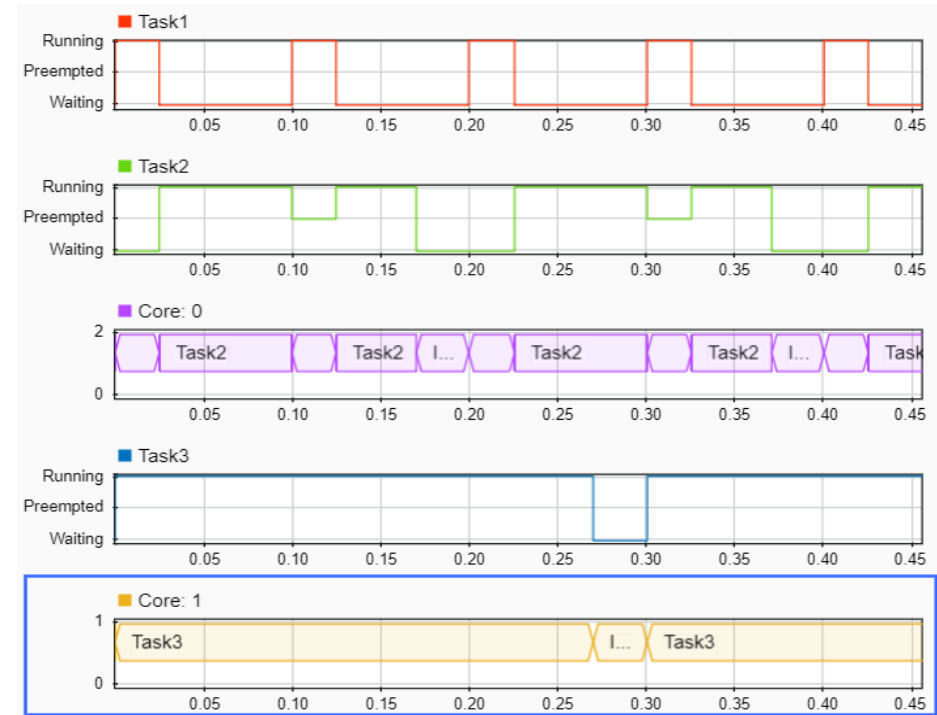
Type: Timer-driven

Period: 0.1

Core: 0

Drop tasks that overrun

OK Cancel Help Apply



Task simulation

dataReadTask properties

Main Simulation

Play recorded task execution sequence

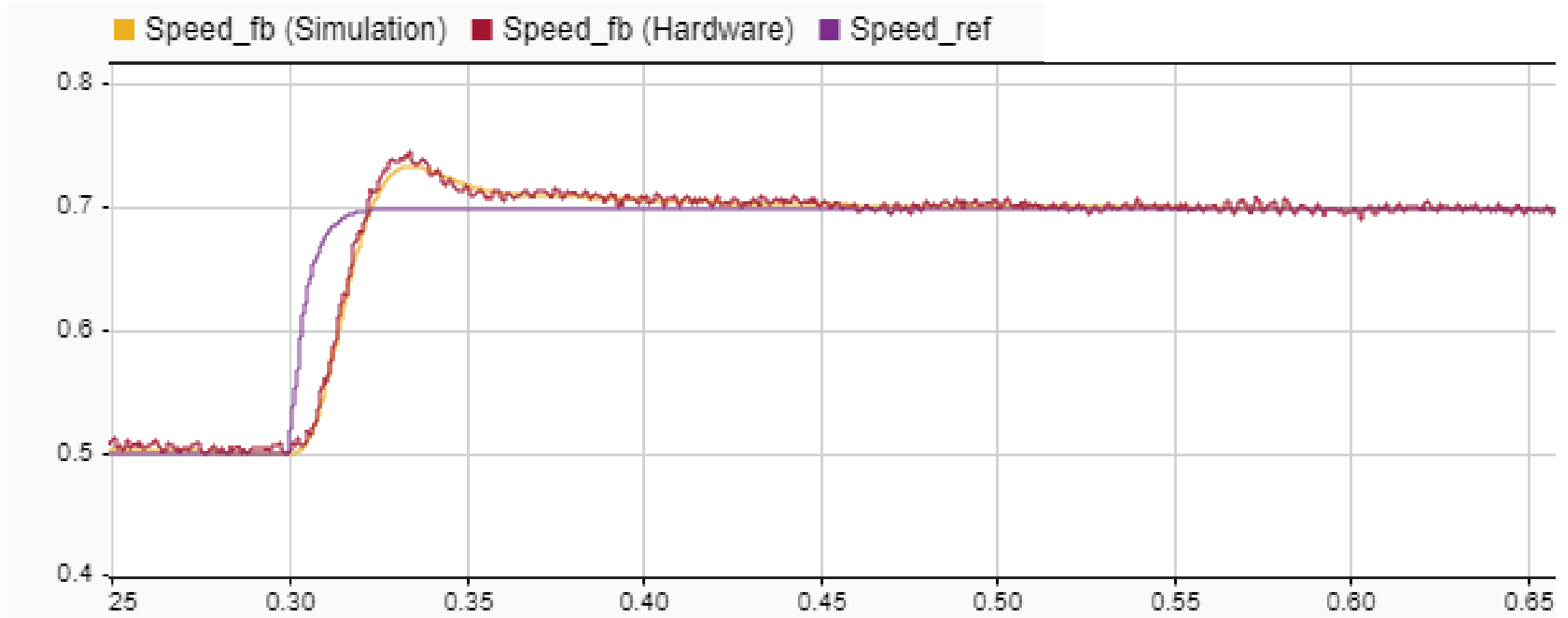
Specify task duration via: Dialog

Task duration settings

Specify task duration time Recorded task execution statistics combination of multiple normal distributions.

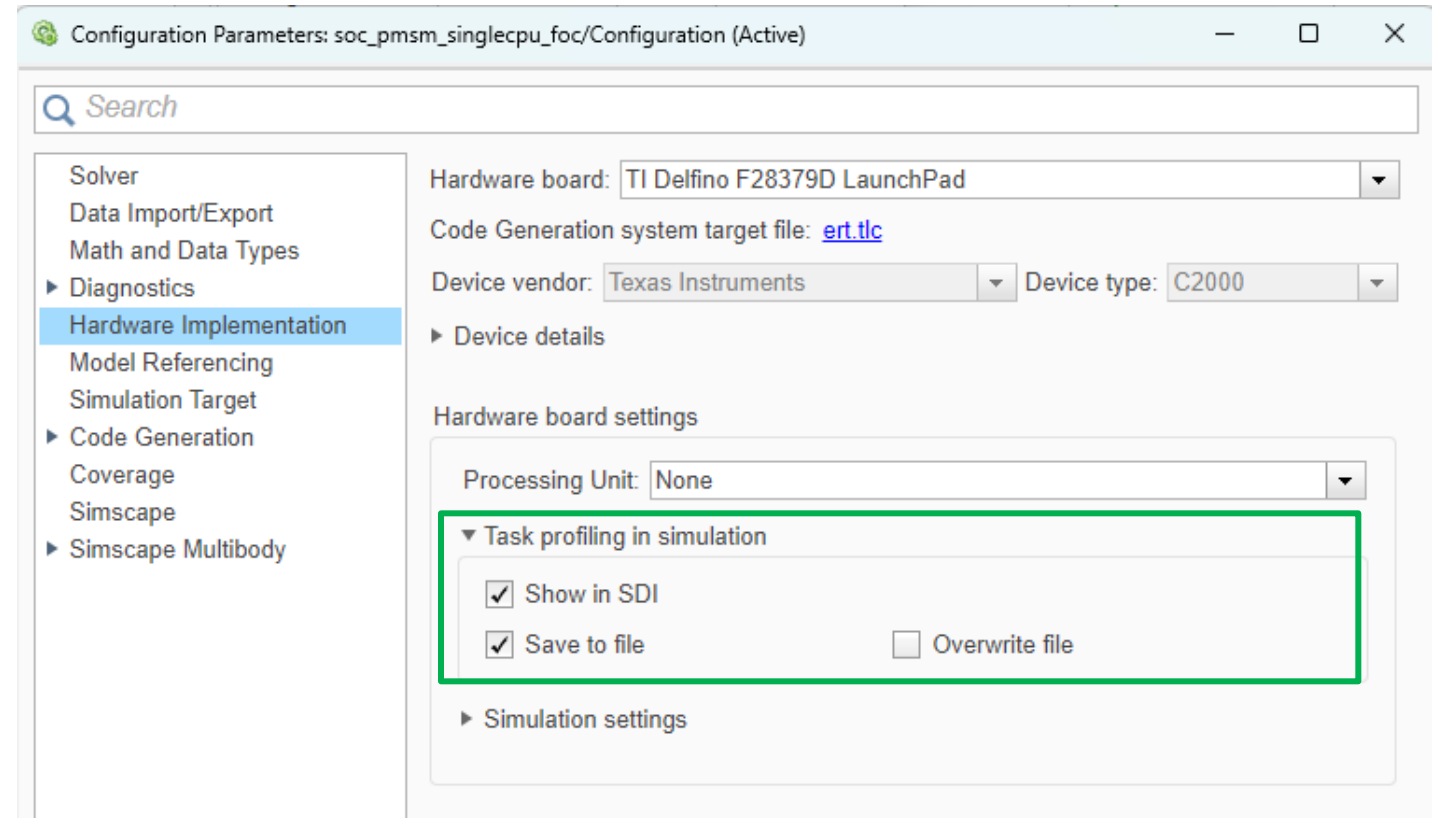
Percent	Mean	SD	Min	Max
1	100	0.0095	0.0001	0.00925 0.00975

Comparison Simulation vs Hardware



How to do profiling?

- External mode required
- Activate Task profiling in Configuration Parameter



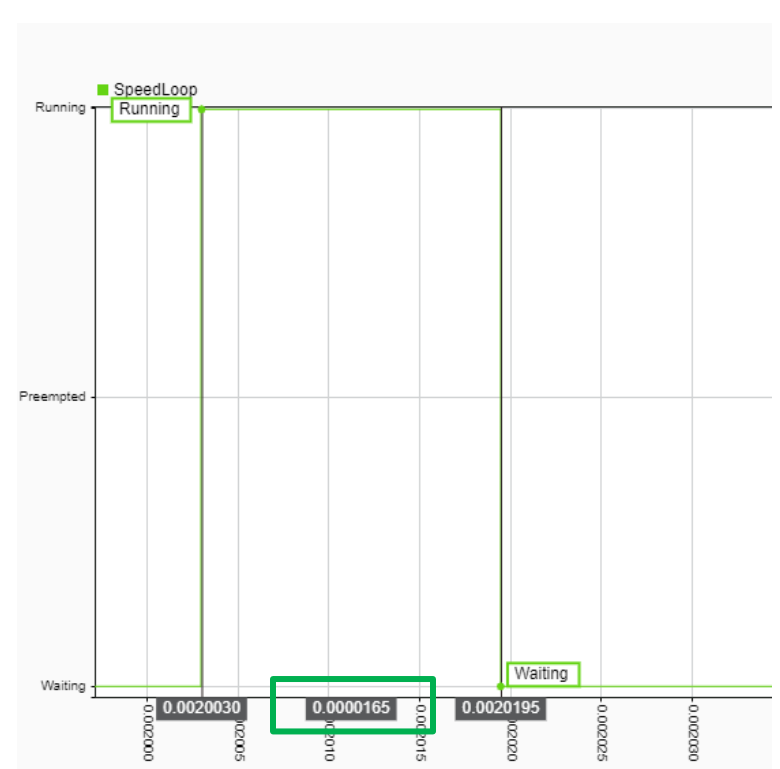
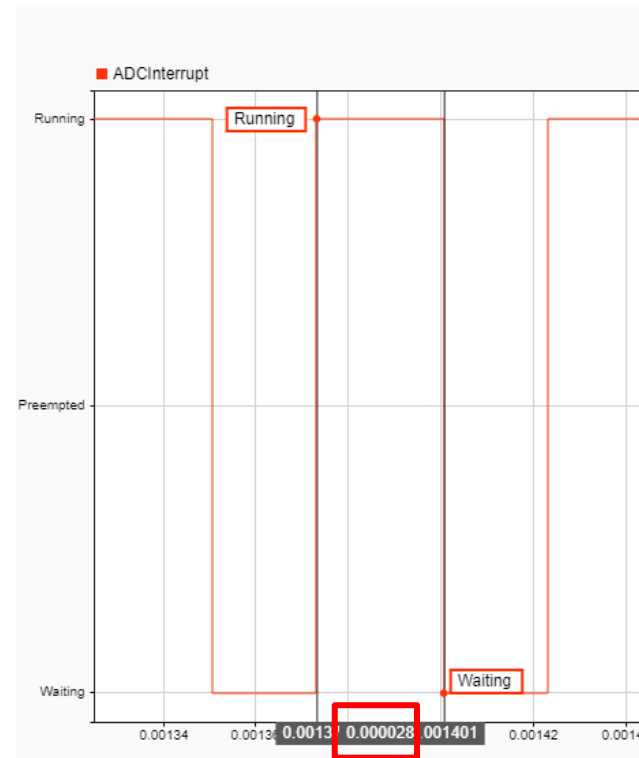
How to do profiling?

- 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	1
soc_pmsm_singlecpu_foc_sw_c28xCPU1_step [0.0005_0]	18670	13184	18670	13184	28
ADCInterrupt	32580	27713	32580	27713	271

▼ Run 32: soc_pmsm_singlecpu_foc_sw_c28xCPU1

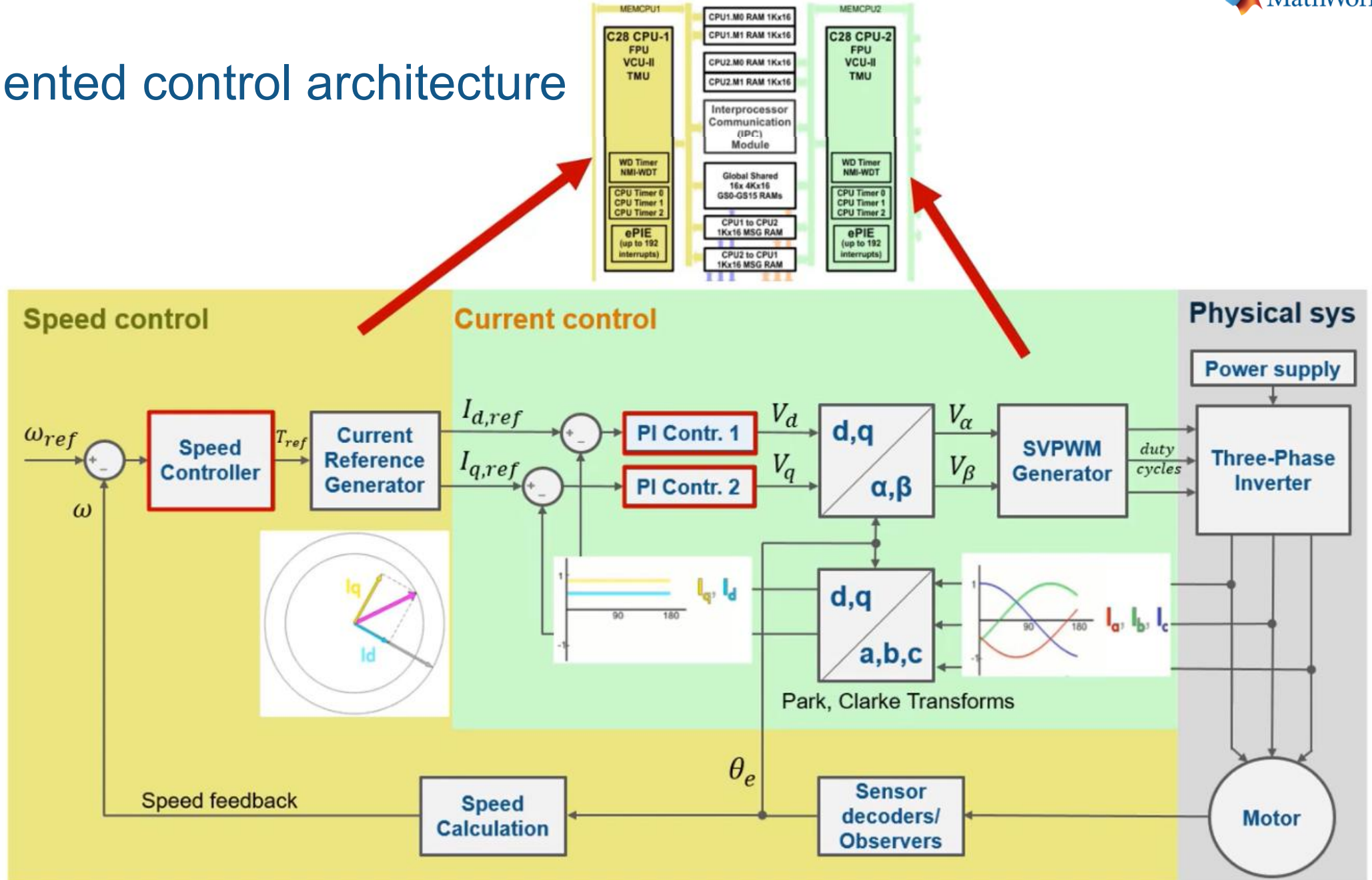
- Core: 0
- SpeedLoop
- SpeedLoop_drop
- ADCInterrupt
- ADCInterrupt_drop
- lab_fb (2)
- Speed_fb
- Speed_ref
- EnClosedLoop



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 μ s which is higher than 25 μ s (40 kHz)
- How to transform our algorithm to multi-processor if required?

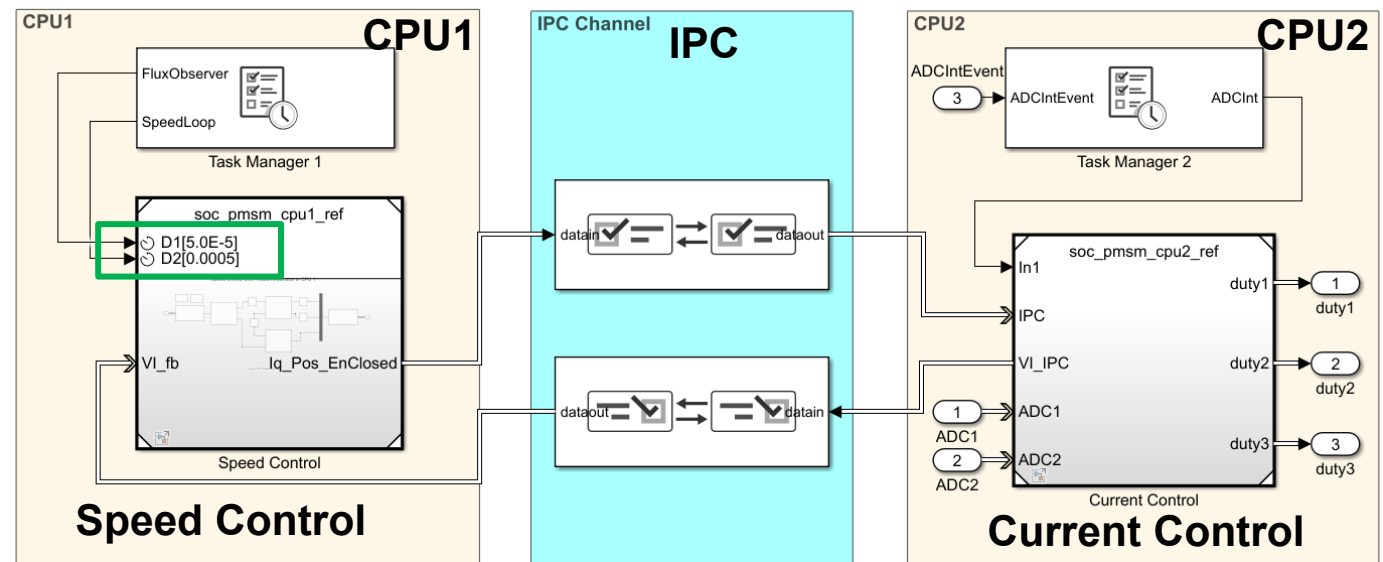
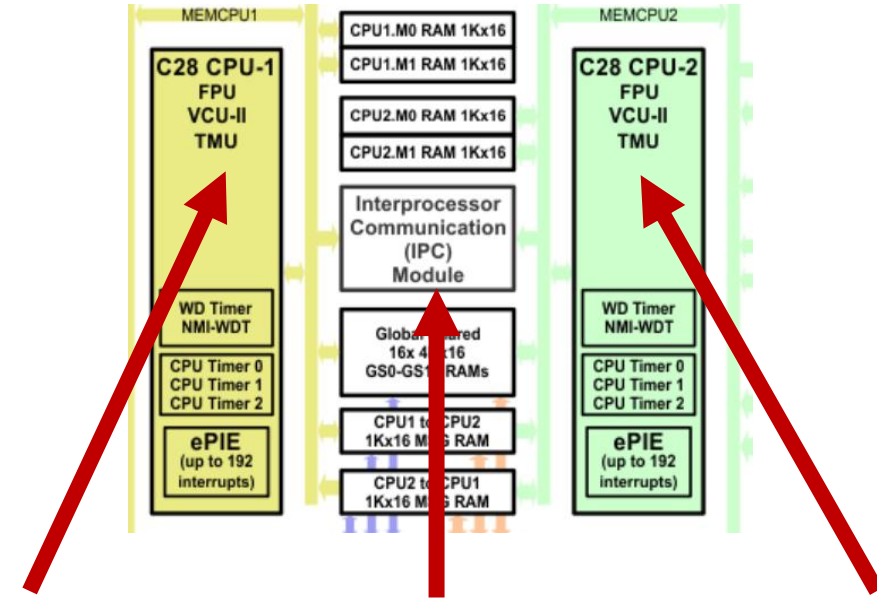
Field-oriented control architecture



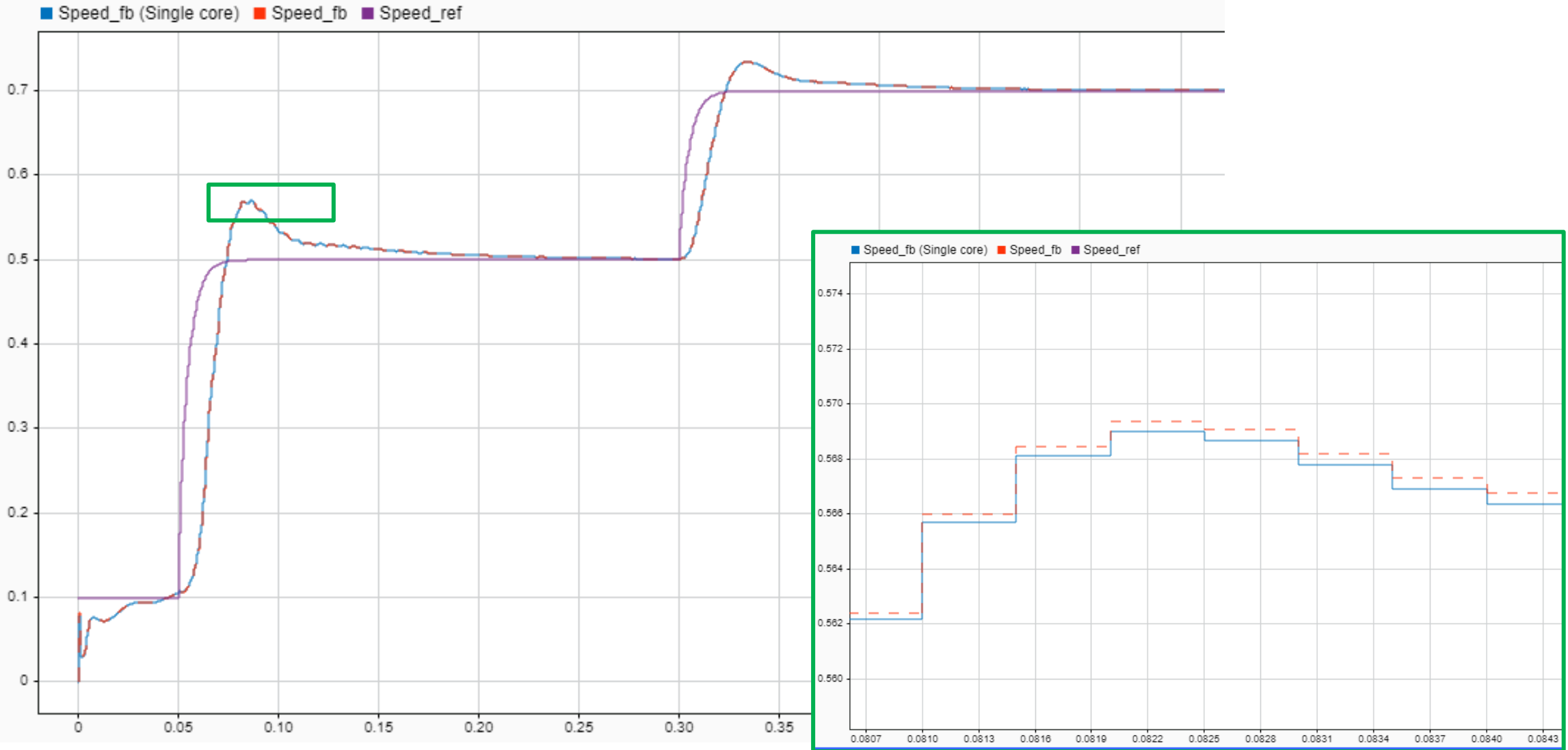
F2837D Architecture

Design for F28379D multi-processor

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

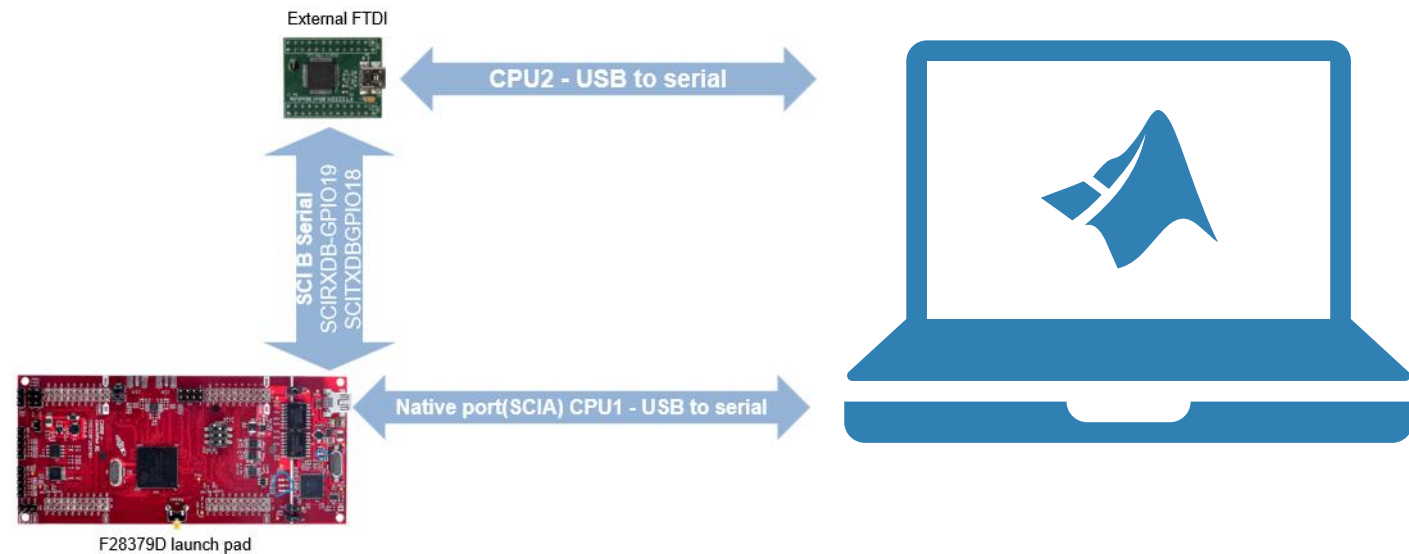
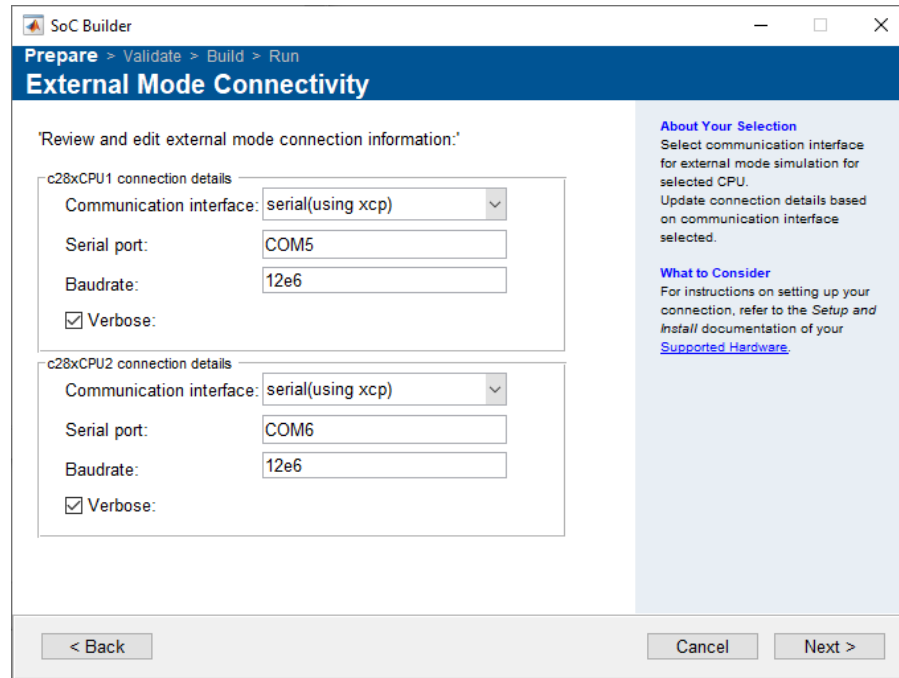


Comparison Single vs Multi processor









Dual External mode and Profiling

- Live task execution information from device to Simulink









Profiling results

- CPU1

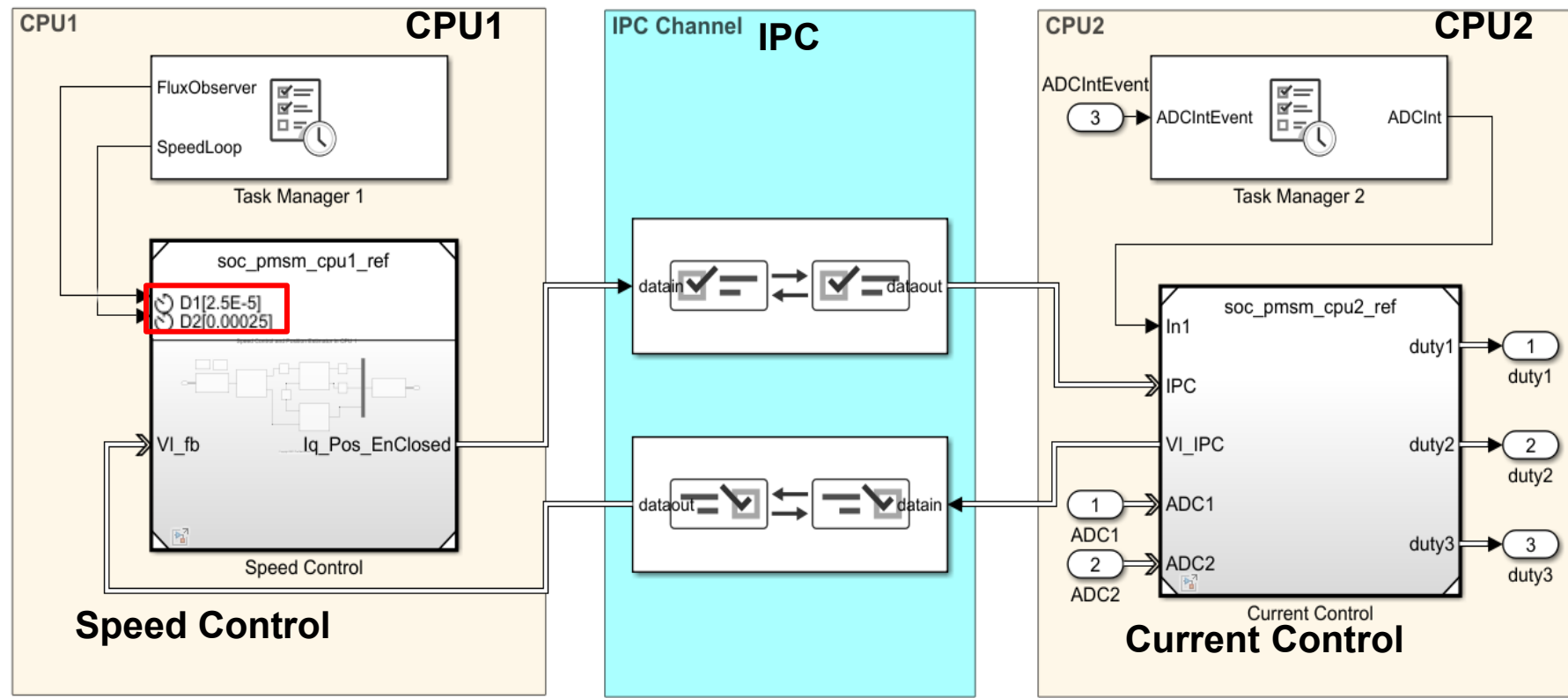
Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls	
soc_pmsm_dualcpu_foc_sw_c28xCPU1_initialize		25350	25350	25350	25350	1  
soc_pmsm_dualcpu_foc_sw_c28xCPU1_step0 [5e-05 0]		23070	22230	23070	22230	290  
soc_pmsm_dualcpu_foc_sw_c28xCPU1_step1 [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	29886	378  

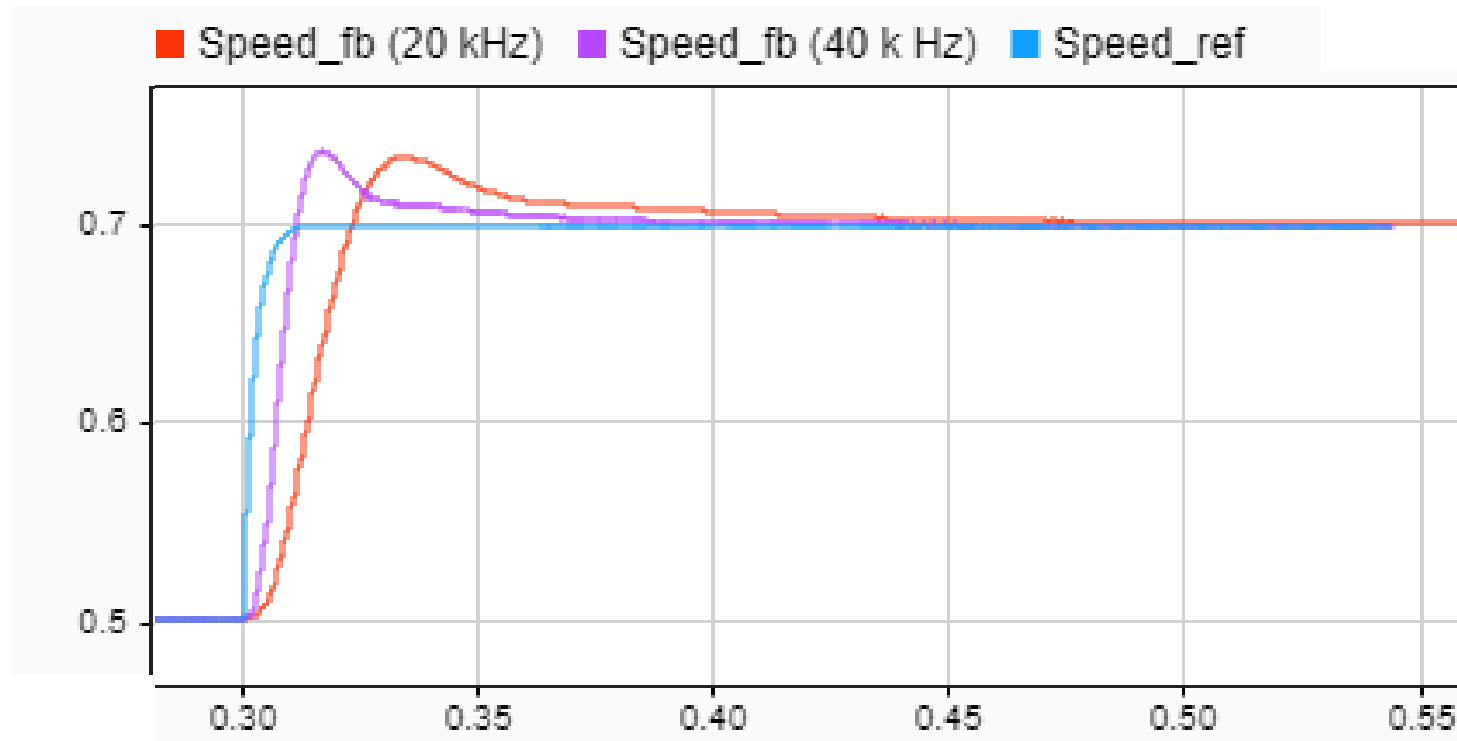
Switch from 20 kHz to 40 kHz

- As $23 \mu\text{s}$ is less than $25 \mu\text{s}$ (40 kHz) we can increase CPU1 frequency



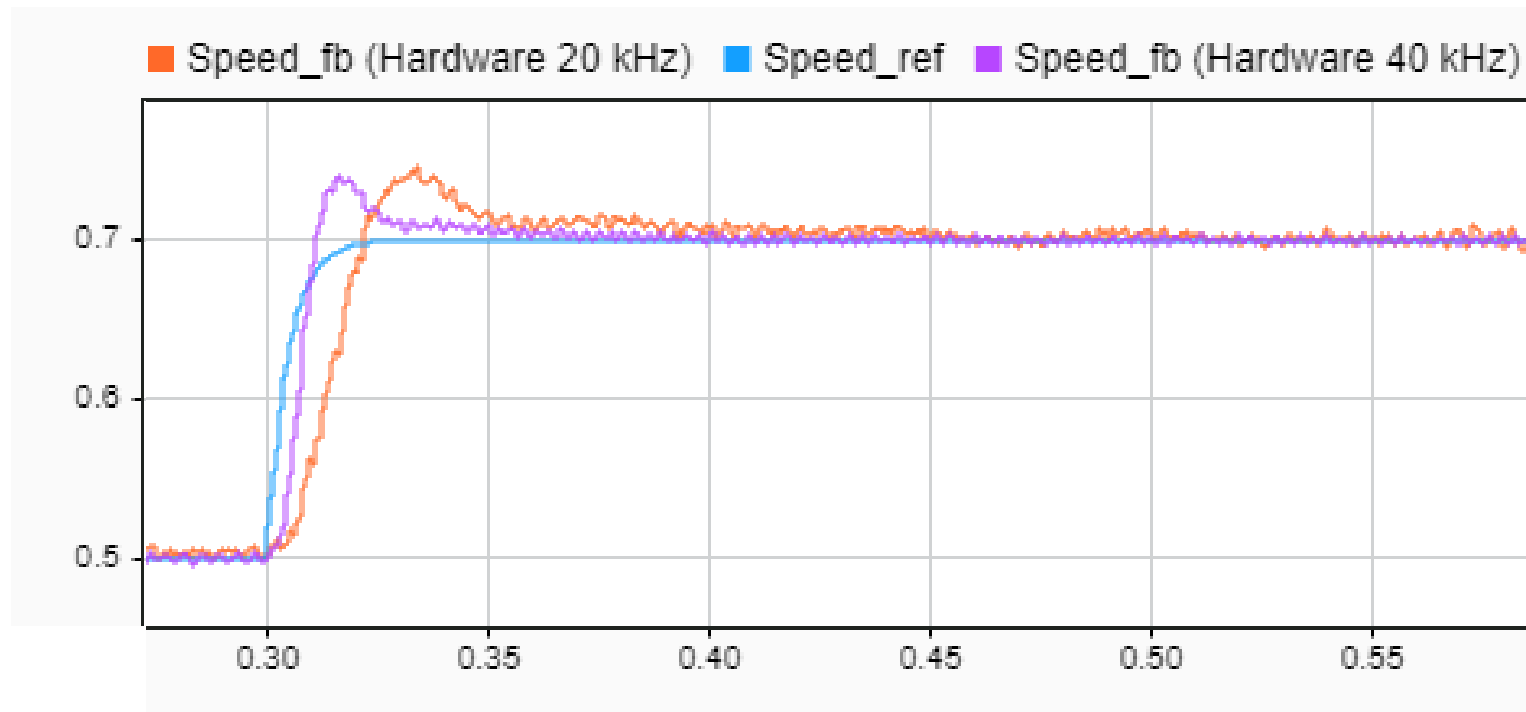
Comparison simulation Dual processor 20 kHz vs 40 kHz

- 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 Microcontroller Blockset](#) / [STM32 Microcontroller Blockset](#) / [Raspberry Pi Blockset](#) / [Hardware Support Packages](#) / Contact us!

Thank you

Q & A