MODEL-BASED DESIGN TOOLBOX
ENABLING FAST PROTOTYPING AND DESIGN

ON-TARGET RAPID PROTOTYPING FOR
MODEL-BASED DESIGN AND
MOTOR CONTROL APPLICATION DEVELOPMENT

SECURE CONNECTIONS
FOR A SMARTER WORLD

Agenda

- **Overview:**
  - Introduction and Objectives
  - Model-Based Design Toolbox: Library blocks, FreeMASTER, and Bootloader
- **Hands-On Demo:**
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Convert simple model to run on Motor Kit with MCD Toolbox and use FreeMASTER
- **Model-Based Design:**
  - Model-Based Design Steps: Simulation, SIL, PIL and ISO 26262
  - SIL/PIL Hands-On Demo Step 2 & 3 of MBD
- **Trapezoidal Motor Control**:
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Trapezoidal control and how to use it to turn a motor
- **Trapezoidal Motor Control Hands-on Demo**:
  - Implement Trapezoidal Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor and tune parameters
- **FOC Motor Control**:
  - FOC Sensor-less control and how to use it to turn a motor
- **FOC Motor Control Hands-On Demo:**
  - Implement FOC Sensor-less Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor
- **Summary and Q&A:**

Introduction: Model-Based Design (MBD)

- Model-Based Design is becoming more common during the normal course of software development to explain and implement the desired behavior of a complex system. The challenge is to take advantage of this approach and get an executable that can be simulated and implemented directly from the model to help you get the product to market in less time and with higher quality. This is especially true for electric motor controls development in this age of hybrid/electric vehicles and the industrial motor control application space.

- Many companies model their controller algorithm and the target motor or plant so they can use a simulation environment to accelerate their algorithm development.

- The final stage of this type of development is the integration of the control algorithm software with target MCU hardware. This is often done using hand code or a mix of hand code and model-generated code. Model-Based Design Toolbox allows this stage of the development to generate 100% of the code from the model.

Introduction: Model-Based Design Toolbox

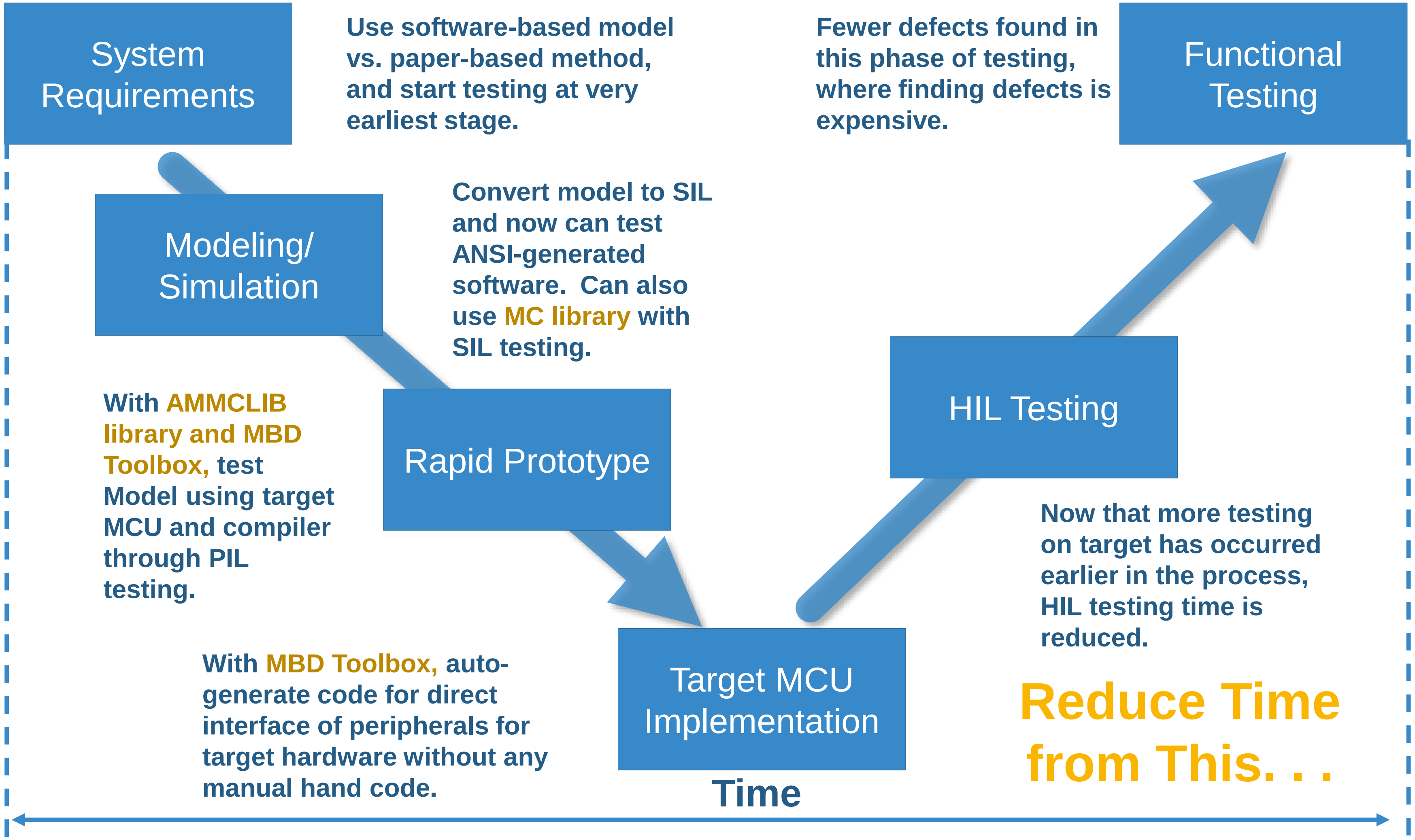
- The Model-Based Design Toolbox includes an embedded target supporting NXP MCUs, Simulink™ plug-in libraries which provide engineers with an integrated environment and tool chain for configuring and generating the necessary software, including initialization routines, device drivers, and a real-time scheduler to execute algorithms specifically for controlling motors.
- The toolbox also includes an extensive Math and Motor Control Function Library developed by NXP's renowned Motor Control Center of Excellence. The library provides dozens of blocks optimized for fast execution on NXP MCUs with bit-accurate results compared to Simulink™ simulation using single-precision math.
- The toolbox provides built-in support for Software and Processor-in-the-Loop (SIL and PIL), which enables direct comparison and plotting of numerical results.

MathWorks products required for MBD Toolbox:

- MATLAB (32-Bit or 64-Bit)*
- Simulink
- MATLAB Coder
- Simulink Coder
- Embedded Coder

*Earlier released products only support 32-bit

Reduce Development Time with Model-Based Design



Using NXP's Model-Based Design Toolbox you can reduce development time from this.

Objectives

- Exposure to NXP’s hardware/software enablement

e.g.:MTRCKTSBNZVM128

BLDC Motor Control Dev Kit

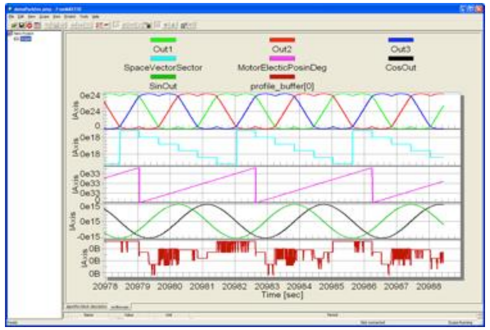
Model-Based Design Toolbox
with Simulink ™

MATLAB
SIMULINK

Model-Based Design
Driver configuration
Assignment to pins
Initialization setup

FREEMASTER

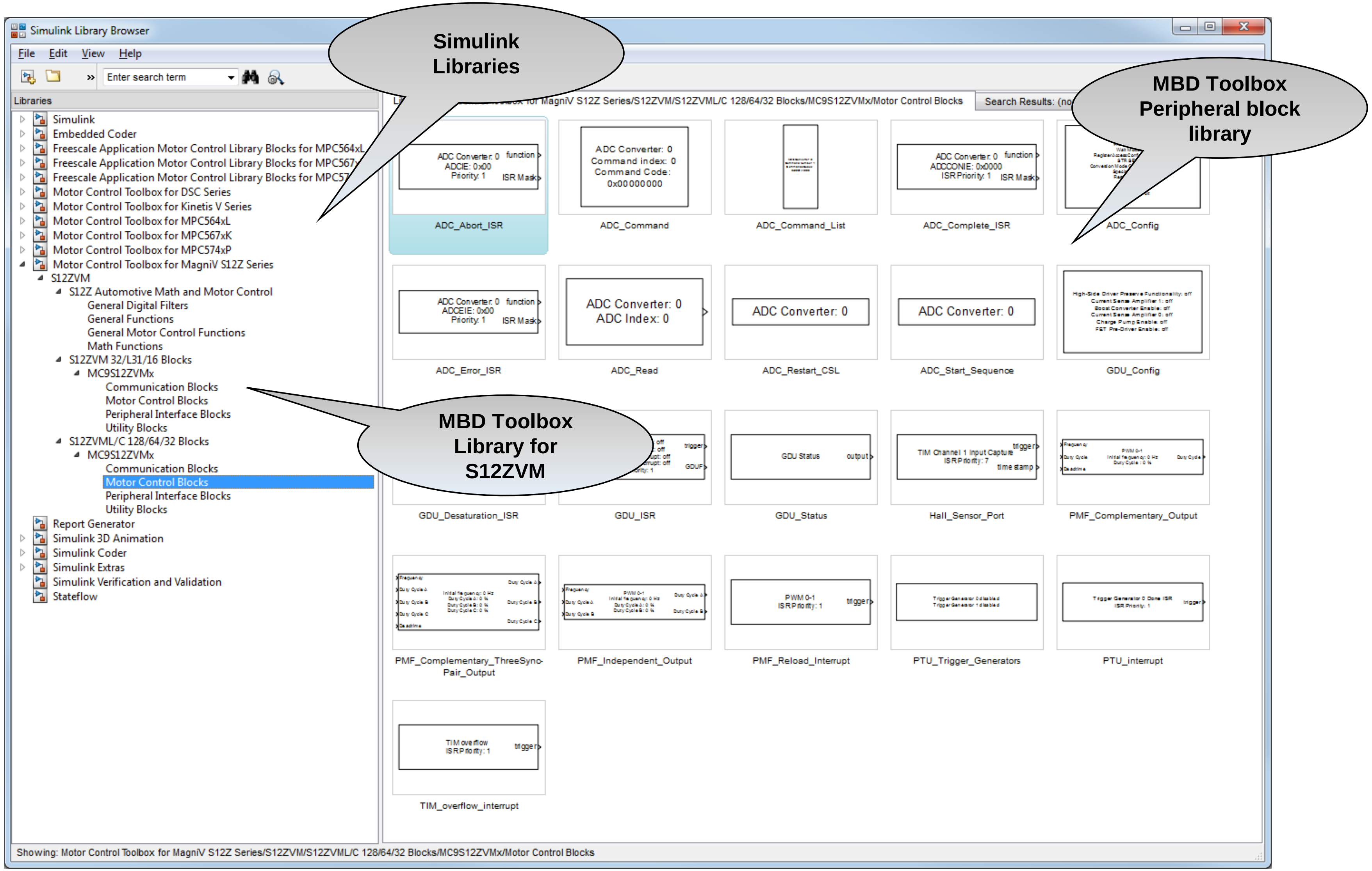
Signal Visualization
and Data Acquisition Tool



Agenda

- **Overview:** 20 minutes
  - Introduction and Objectives
  - ***Model-Based Design Toolbox: Library blocks, FreeMASTER, and Bootloader***
- **Hands-On Demo:** 50 minutes
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Convert simple model to run on Motor Kit with MBD Toolbox and use FreeMASTER
- **Model-Based Design:** 10 minutes
  - Model-Based Design Steps: Simulation, SIL, PIL and ISO 26262
  - SIL/PIL Hands-On Demo Step 2 & 3 of MBD
- **Trapezoidal Motor Control**: 30 minutes
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Trapezoidal control and how to use it to turn a motor
- **Trapezoidal Motor Control Hands-on Demo**: 80 minutes
  - Implement Trapezoidal Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor and tune parameters
- **FOC Motor Control**: 20 minutes
  - FOC Sensor-less control and how to use it to turn a motor
- **FOC Motor Control Hands-On Demo:** 80 minutes
  - Implement FOC Sensor-less Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor
- **Summary and Q&A:** 10 minutes

MBD Toolbox: Library Contents



Model-Based Design Toolbox: Toolbox Contents

On-Chip Peripherals

- General
  - ADC conversion
  - Digital I/O
  - PIT timer
  - ISR
- Communication Interface
  - CAN driver
  - SPI driver
  - I2C
  - UART
- Motor Control Interface
  - Cross triggering unit
  - PWM
  - eTimer block(s)
  - Sine wave generation
  - ADC Command List
  - GDU (Gate Drive Unit)
  - PTU (Programable Trigger Unit)
  - TIM Hall Sensor Port
  - FTM (Flex Timer Module)
  - PDB (Programmable Delay Block)

Configuration/Modes

- Compilers Supported
  - CodeWarrior
  - Wind River DIAB
  - Green Hills
  - Cosmic
  - IAR
  - GCC
  - RAM/FLASH targets
- Simulation Modes
  - Normal
  - Accelerator
  - Software in the Loop (SIL)
  - Processor in the Loop (PIL)
- MCU Option
  - Multiple packages
  - Multiple Crystal frequencies

Utilities

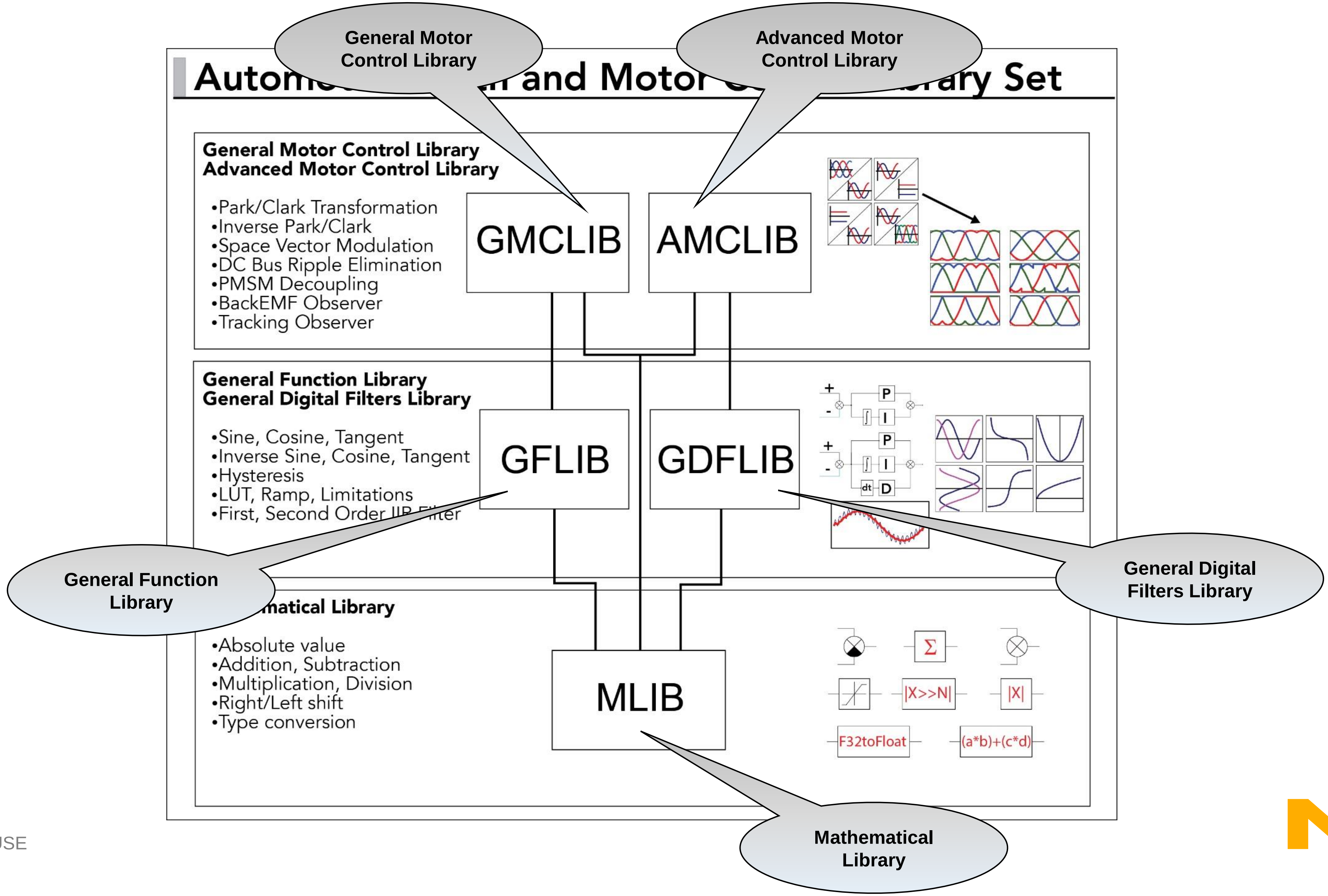
- FreeMASTER Interface
  - Data acquisition
  - Calibration
  - Customize GUI
- Profiler Function
  - Exec. time measurement
  - Available in PIL
  - Available in standalone

Embedded MCU Support

- MPC5643L
- MPC567xK
- MPC574xP
- S12ZVM
- KV10Z
- 56F82xx
- KV31/30/40/50
- S32K

NOTE: Peripheral Blocks and compiler support is dependent on MCU use.

Automotive Math and Motor Control Library Set - Architecture



Automotive Math and Motor Control Library Set – Content

MLIB

- **Absolute Value, Negative Value**
  - MLIB_Abs, MLIB_AbsSat
  - MLIB_Neg, MLIB_NegSat
- **Add/Subtract Functions**
  - MLIB_Add, MLIB_AddSat
  - MLIB_Sub, MLIB_SubSat
- **Multiply/Divide/Add-multiply Functions**
  - MLIB_Mul, MLIB_MulSat
  - MLIB_Div, MLIB_DivSat
  - MLIB_Mac, MLIB_MacSat
  - MLIB_Mnac, MLIB_Msu
  - MLIB_VMac
- **Shifting**
  - MLIB_ShL, MLIB_ShLSat
  - MLIB_ShR
  - MLIB_ShBi, MLIB_ShBiSat
- **Normalisation, Round Functions**
  - MLIB_Norm, MLIB_Round
- **Conversion Functions**
  - MLIB_ConvertPU, MLIB_Convert

GFLIB

- **Trigonometric Functions**
  - GFLIB_Sin, GFLIB_Cos, GFLIB_SinCos, GFLIB_Tan
  - GFLIB_Asin, GFLIB_Acos, GFLIB_Atan, GFLIB_AtanYX
  - GFLIB_AtanYXShifted
- **Limitation Functions**
  - GFLIB_Limit, GFLIB_VectorLimit
  - GFLIB_LowerLimit, GFLIB_UpperLimit
- **PI Controller Functions**
  - GFLIB_ControllerPIr, GFLIB_ControllerPIrAW
  - GFLIB_ControllerPIp, GFLIB_ControllerPIpAW
- **Interpolation**
  - GFLIB_Lut1D, GFLIB_Lut2D
- **Hysteresis Function**
  - GFLIB_Hyst
- **Signal Integration Function**
  - GFLIB_IntegratorTR
- **Sign Function**
  - GFLIB_Sign
- **Signal Ramp Function**
  - GFLIB_Ramp
- **Square Root Function**
  - GFLIB_Sqrt

GDFLIB

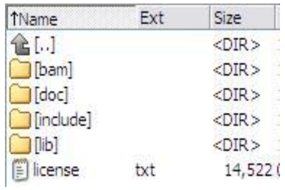
- **Finite Impulse Filter**
  - GDFLIB_FilterFIR
- **Moving Average Filter**
  - GDFLIB_FilterMA
- **1st Order Infinite Impulse Filter**
  - GDFLIB_FilterIIR1init
  - GDFLIB_FilterIIR1
- **2nd Order Infinite Impulse Filter**
  - GDFLIB_FilterIIR2init
  - GDFLIB_FilterIIR2

GMCLIB

- **Clark Transformation**
  - GMCLIB_Clark
  - GMCLIB_ClarkInv
- **Park Transformation**
  - GMCLIB_Park
  - GMCLIB_ParkInv
- **Duty Cycle Calculation**
  - GMCLIB_SvmStd
- **Elimination of DC Ripples**
  - GMCLIB_ElimDcBusRip
- **Decoupling of PMSM Motors**
  - GMCLIB_DecouplingPMSM

AMCLIB

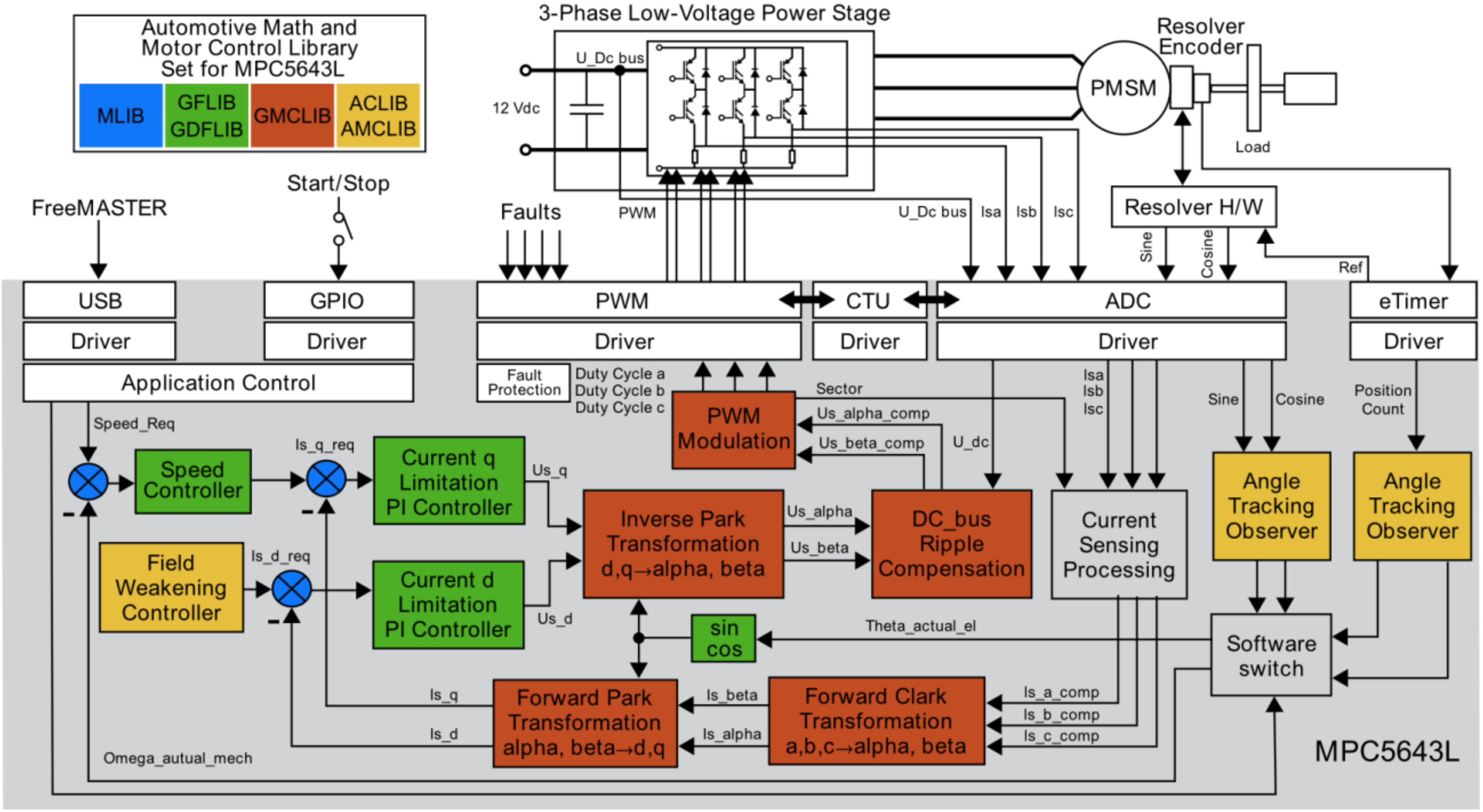
- **BEMF Observer DQ**
  - AMCLIB_BemfObsrvDQ
- **Tracking Observer**
  - AMCLIB_TrackObsrv



Delivery Content

→ Matlab/Simulink Bit Accurate Models
→ User Manuals
→ Header files
→ Compiled Library File
→ License File (to be accepted at install time)

AMMCLib Application Example for MPC5643L PMSM Field Oriented Control



Auto Math and Motor Control Library Set – Supported Devices

Target Platform	GreenHills Multi	CodeWarrior	WindRiver Diab	Cosmic	IAR	GCC	S32DS PPC
	Version 2015.1.4	Version 10.6.4	Version 5.9.4.8	Version 4.3.4	Version 8.11	Version 4.9.3	Version 1.2
MPC560xP MPC560xB MPC564xL MPC567xF MPC567xK	Available	Available	Available	Not supported[1]	N/A[2]	N/A[2]	Available
MPC574xC MPC574xG MPC574xP MPC574xR MPC577xC MPC577xK MPC577xM	Available	N/A[2]	Available	Not supported[1]	N/A[2]	N/A[2]	Available
S12ZVM	N/A[2]	Available	N/A[2]	Available	N/A[2]	N/A[2]	N/A[2]
S32K14x	Available	Not supported[1]	N/A[2]	Not supported[1]	Available	Available	N/A[2]
KEAx	Available	Available	N/A[2]	Not supported[1]	N/A[2]	N/A[2]	N/A[2]

1) Not supported: The compiler contains the support of selected device, however the AMMCLib does not support this compiler.
2) N/A: The compiler (or the compiler version) does not support selected device.

MBD Toolbox: RAppID Bootloader Utility

The RAppID Bootloader works with the built-in Boot Assist Module (BAM) included in Freescale Qorivva MCUs or can be resident in FLASH. The Bootloader provides a streamlined method for programming code into FLASH or RAM on either target EVBs or custom boards. Once programming is complete, the application code automatically starts.

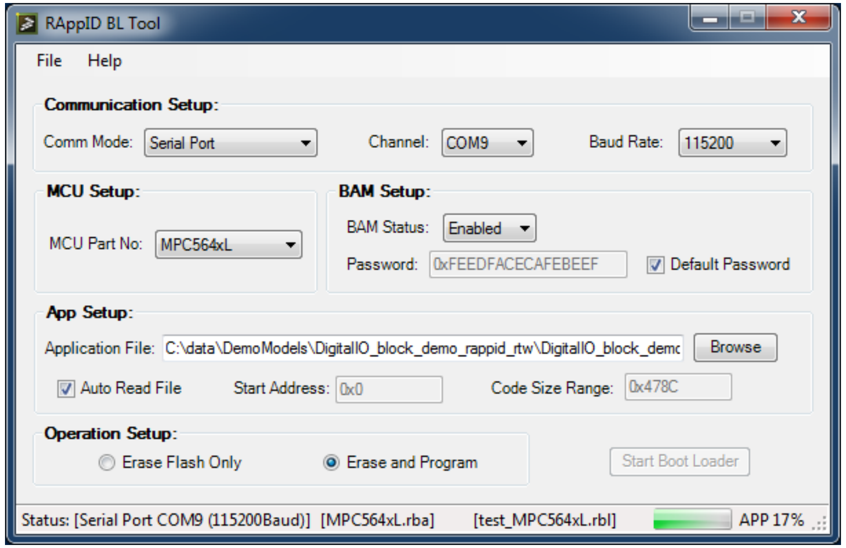
Modes of Operation

The Bootloader has two modes of operation: for use as a stand-alone PC desktop GUI utility, or for integration with different user required tools chains through a command line interface (i.e. Eclipse Plug-in, MATLAB/Simulink, …)

MCUs Supported

MPC5534, MPC5601/2D, MPC5602/3/4BC, MPC5605/6/7B, MPC564xB/C, MPC567xF, MPC567xK, MPC564xA, MPC5605/6/7BK, MPC564xL, MPC5604/3P, MPC574xP, MPC5746R, MPC5746C, MPC5748G, MPC5777C, MPC5775K, S12ZVC, S12ZVL, S12ZVM, S12VR, KEAZN16/32/64, KEAZ64/128, S32K144, 56F82xx, KV10Z, and KV3x/KV4x/KV5x.

Graphical User Interface



Command Line

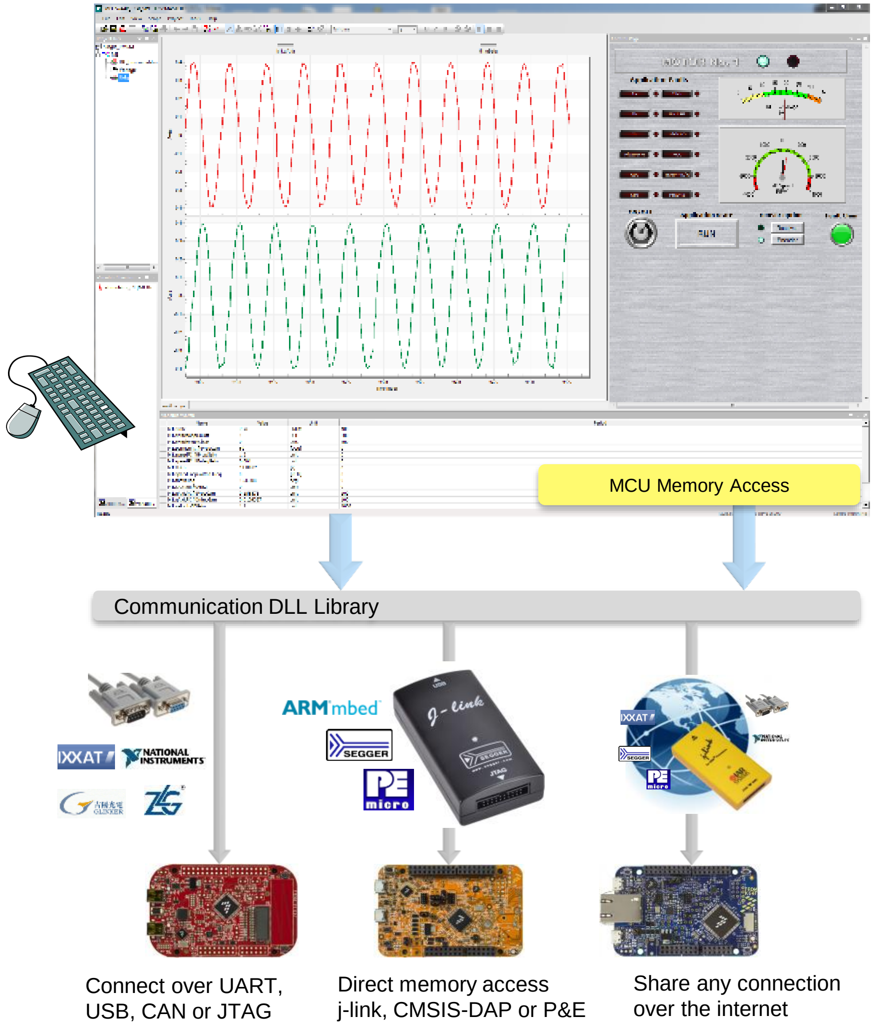
Status given in two stages: Bootloader download, then application programming

What is FreeMASTER?

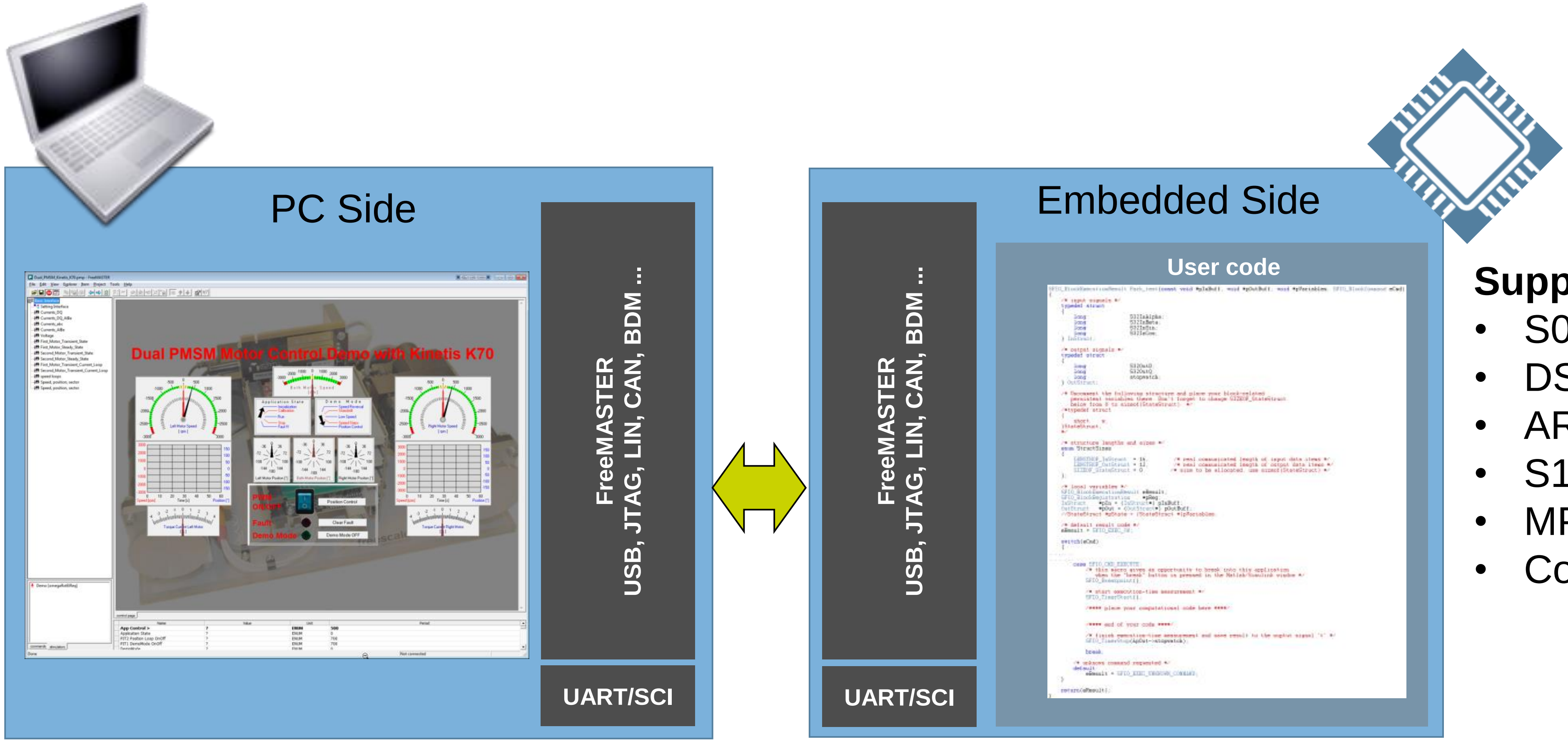
- Runtime configuration & tuning tool for embedded software applications
- Graphical Control Panel
- Data Capture tool, interface to custom processing in Matlab, Excel etc.

What do we do with FreeMASTER?

- **Connect**: to target MCU over UART, CAN, BDM, JTAG
- **Monitor**: read & show variables in run-time
- **Control**: set variables, send commands
- **Share**: enable Excel, Matlab or a script engine to add hardware to the control loop



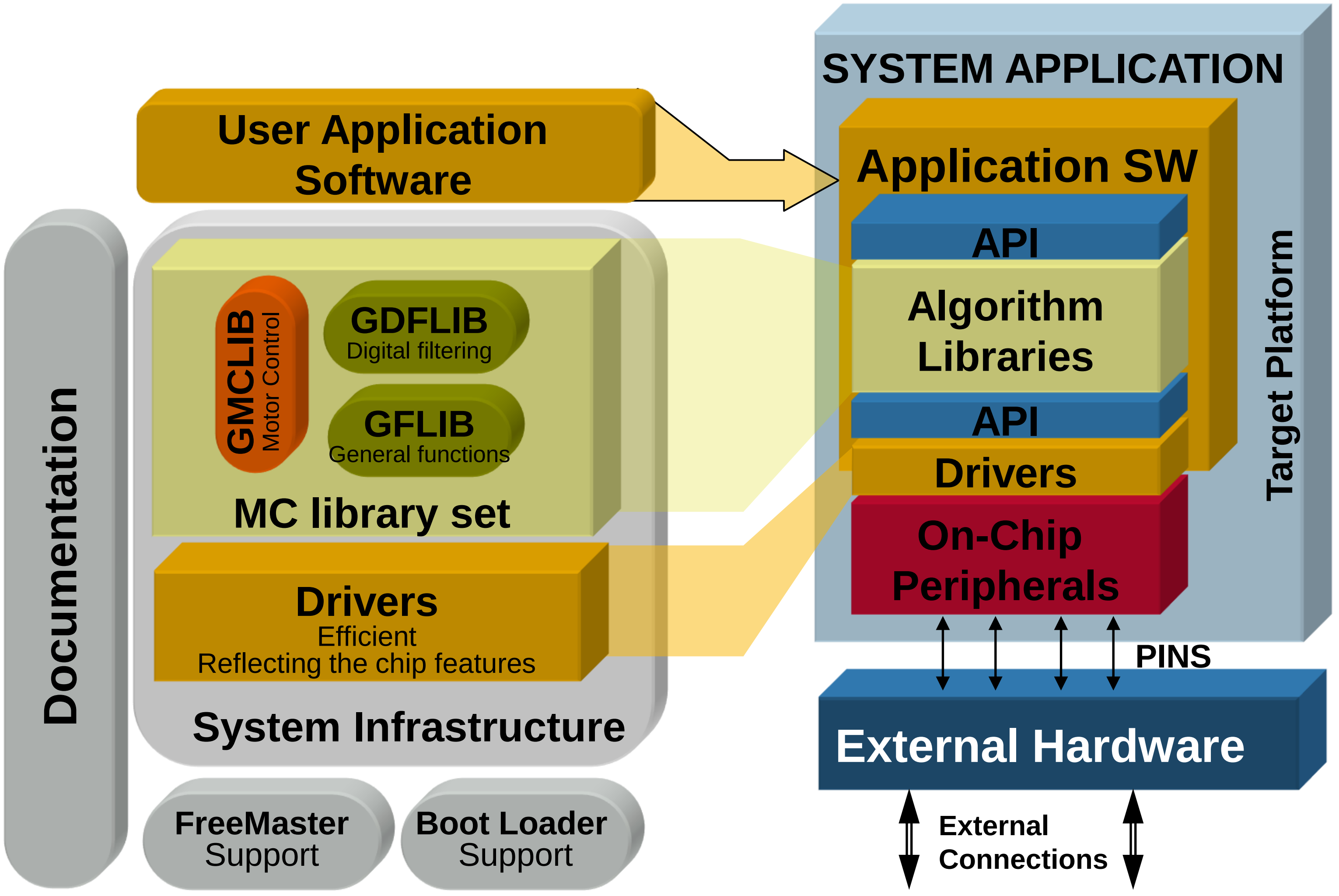
FreeMASTER Topology and Platforms Support



Supported platforms:

- S08
- DSC
- ARM Cortex-M (Kinetis/S32)
- S12/S12X/S12Z(MagniV),
- MPC56xx, MPC57xx
- ColdFire V1/V2

MBD Toolbox: Summary of Customer Application Support



Agenda

- **Overview:**
  - Introduction and Objectives
  - Model-Based Design Toolbox: Library blocks, FreeMASTER, and Bootloader
- **Hands-On Demo:**
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Convert simple model to run on Motor Kit with MCD Toolbox and use FreeMASTER
- **Model-Based Design:**
  - Model-Based Design Steps: Simulation, SIL, PIL and ISO 26262
  - SIL/PIL Hands-On Demo Step 2 & 3 of MBD
- **Trapezoidal Motor Control**:
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Trapezoidal control and how to use it to turn a motor
- **Trapezoidal Motor Control Hands-on Demo**:
  - Implement Trapezoidal Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor and tune parameters
- **FOC Motor Control**:
  - FOC Sensor-less control and how to use it to turn a motor
- **FOC Motor Control Hands-On Demo:**
  - Implement FOC Sensor-less Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor
- **Summary and Q&A:**

Hands-On Demo: Motor Kit

Features:
- MC9S12ZVML/C12MKH
- BDM interface
- On-board OSBDM
- Hall Sensor
- Resolver interface
- SINCOS interface
- LIN/CAN
- USB-to-SCI serial port
- Phase and DC-bus current sensing circuits
- FAULT indicator
- Over-voltage and over-current FAULT indicator with potentiometer adjustments
- 2 User LEDs
- 2 push buttons
- 1 switch
- 4 MHz oscillator
- 1 Potentiometer

MTRCKTSBNZVM128

BLDC Motor

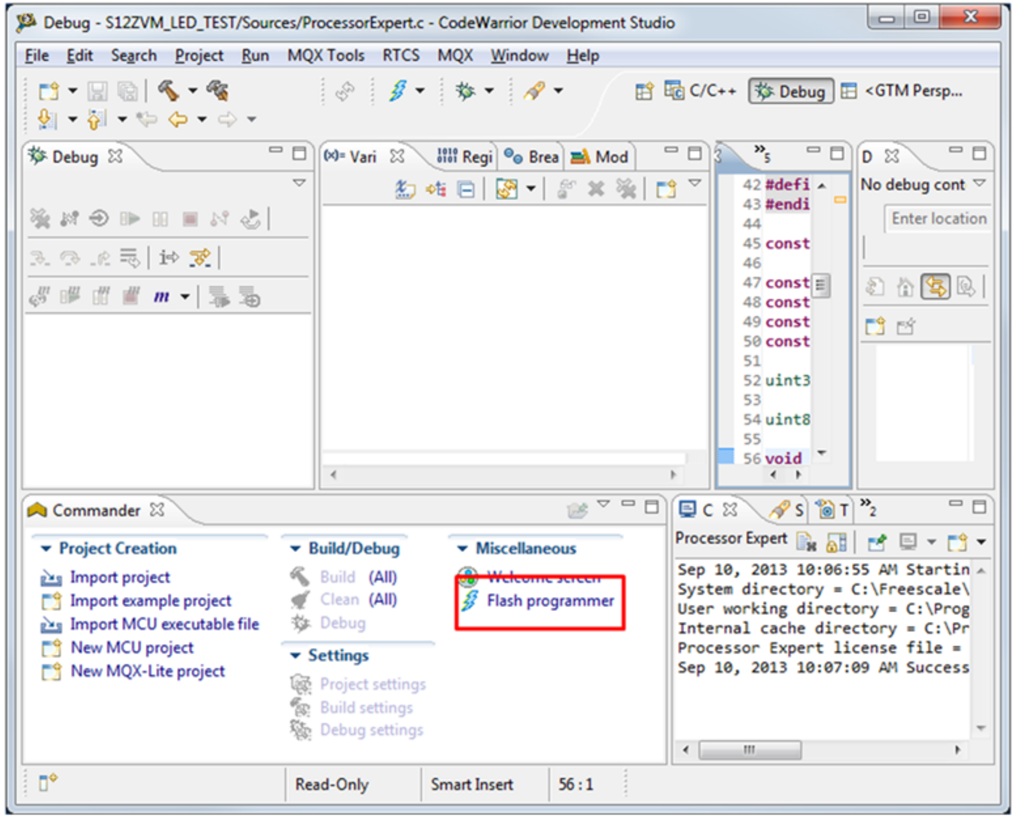
Control Dev Kit

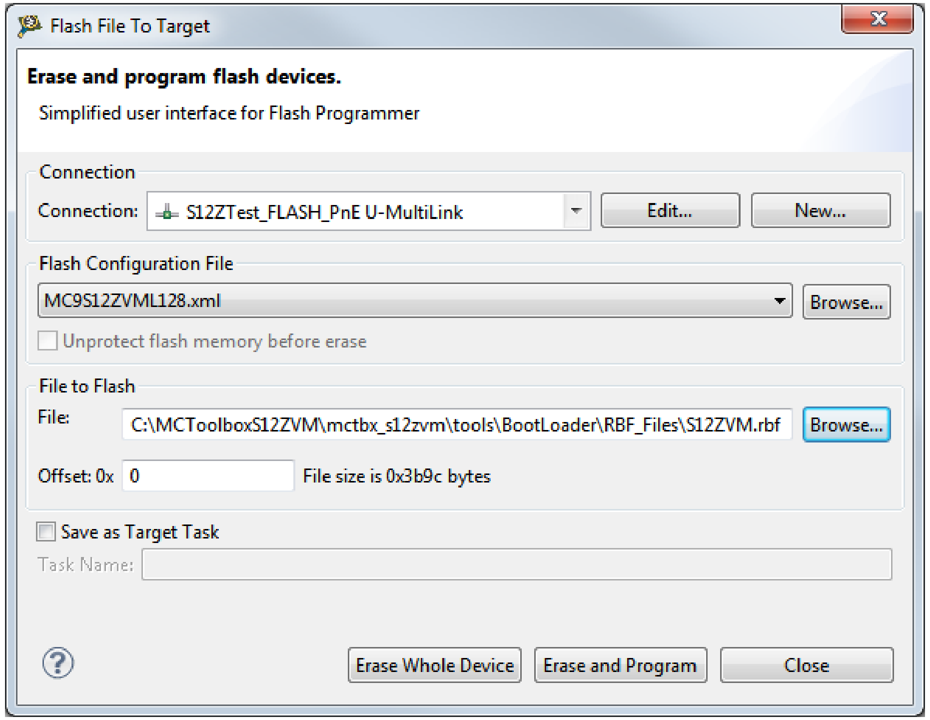
Motor and Drive Features:
- Input voltage 12–24 V DC
- Output current 5–10 Amps
- 3-phase MOSFET inverter using 6 N-channel Power MOSFETs
- 4 pole-pair BLDC motor with Hall sensors (9450 RPM rated speed at 24 V)

Hands-On Demo: Read A/D and Toggle LED Simple Model

Load in Flash Bootloader using CodeWarrior Flash Programmer

Use the Flash programmer in CW IDE to program the Flash Bootloader.

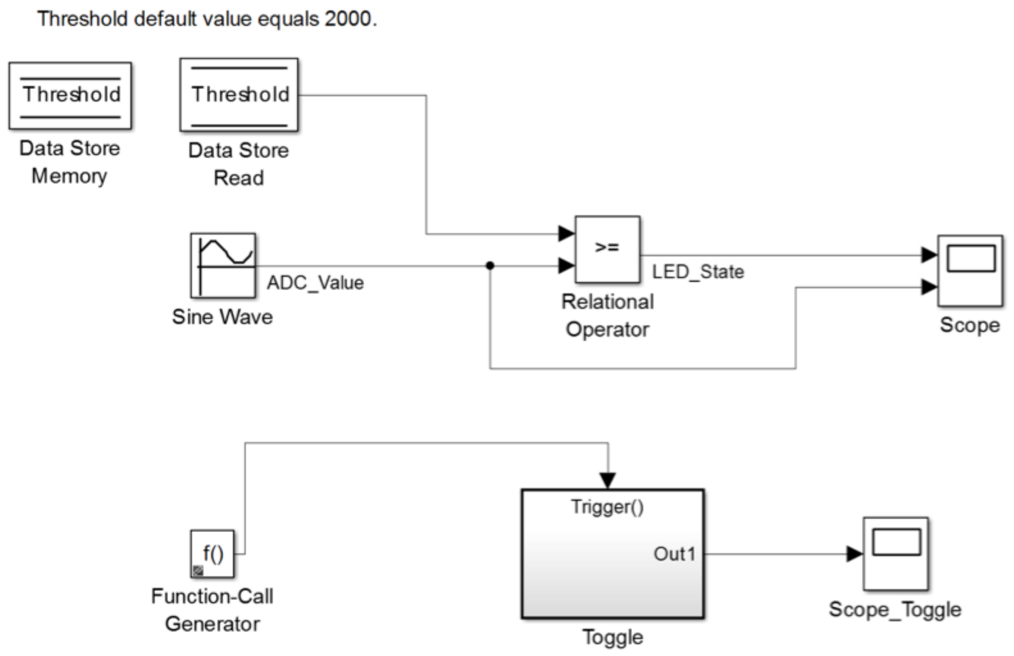


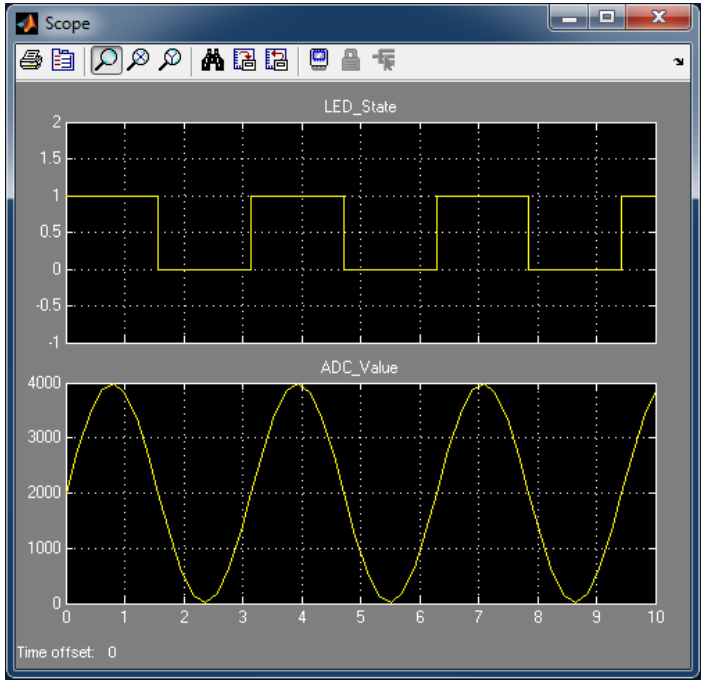


Hands-On Demo: Read A/D and Toggle LED Simple Model

Run Simple Model Simulation

1. Open Model “Simple_ADC.slx and save it as S12ZVM_Simple_ADC.slx”
2. You will see a model that changes the output state of a relational operator based on an input value as compared to a data value.
3. Run simulation and open the scope. You should see the following on the scope:





Hands-On Demo: Read A/D and Toggle LED Simple Model

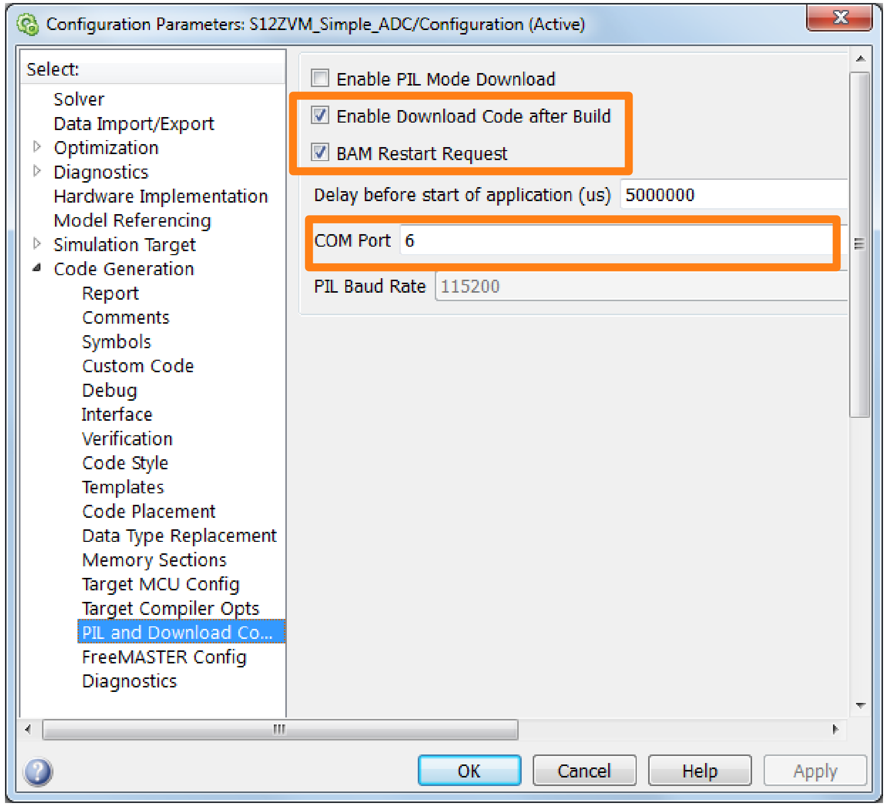
Convert Simple Model and Run

1. Save Model as “S12ZVM_Simple_ADC.slx”
2. Select system target file “mcd_s12zvm.tlc” to configure model for the MCU
3. Open Simulink Library
4. Go to Motor Control Toolbox for MC9S12ZVMx -> MC9S12ZVMx -> MCD_MC9S12ZVMx_Config_Information Block
5. Drag the block into the model
6. Open block and go to PIL and Download Config
7. Check Enable Download Code after Build and BAM Restart Request
8. Enter the COM port number that you are using from PC
9. Enable Freemaster to run on SCI 1 at 115200 Baud
10. Delete Sine Wave block and both Scopes
11. Also delete line that was going to second input of scope
12. Go back to library under Motor Control Blocks and drag in an ADC Config block, ADC Command Sequence List block and a ADC Read block which will connect to the ADC_Value line

Hands-On Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run

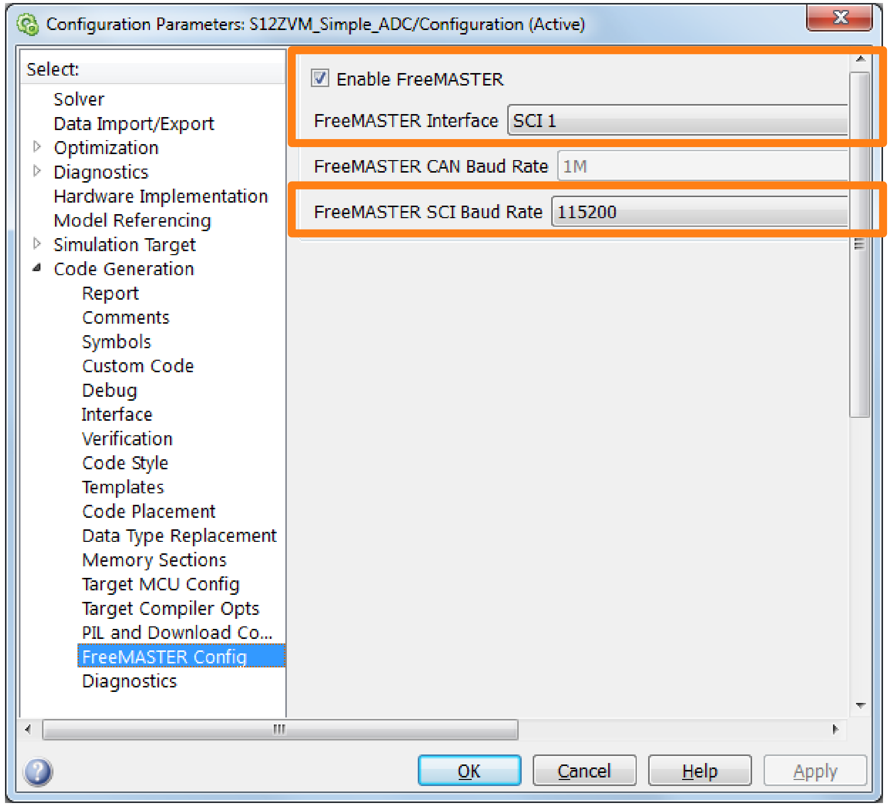
13. Open "Configuration Parameters" and go to PIL/BAM Setup tab.
14. Enter the COM port number that you are using from PC



Hands-On Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run

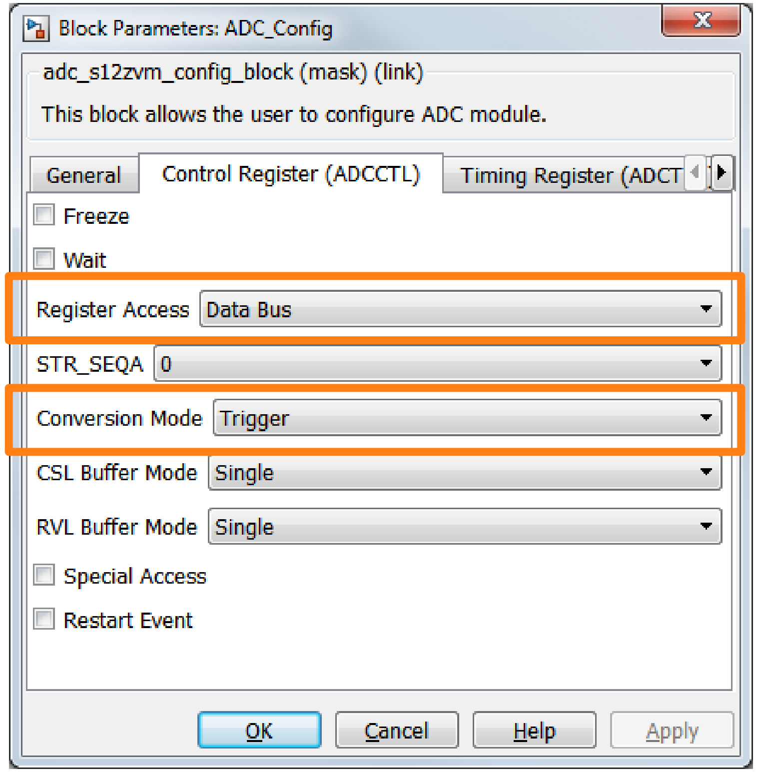
15. Open “Configuration Parameters” and go to FreeMASTER Config tab.
16. Enable Freemaster to run on SCI 1 at 115200 Baud

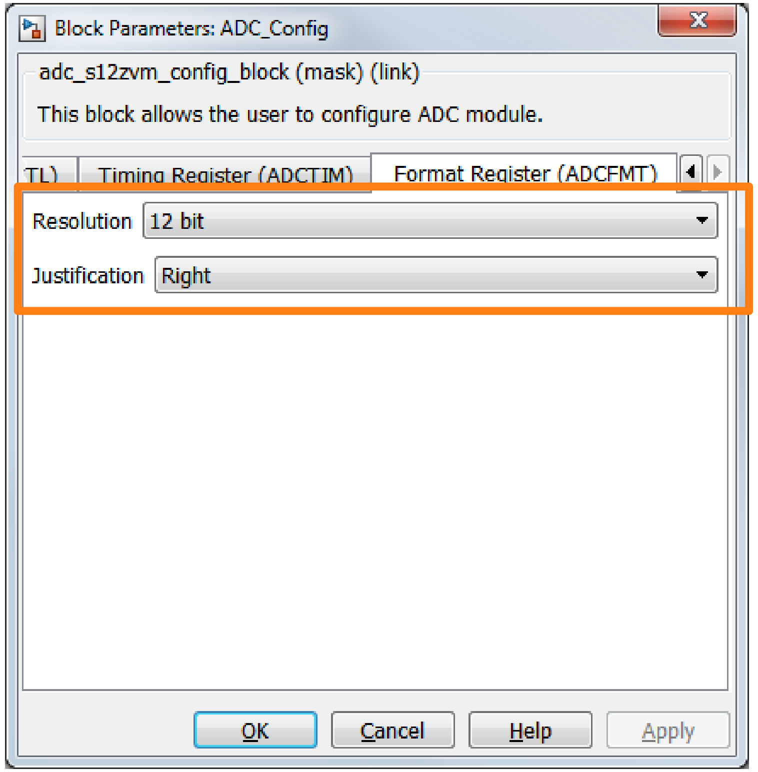


Hands-On Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run

17. Open "ADC Config block" and set Conversion Mode to Trigger.
18. Also select Data Bus register access,12-bit resolution and right justification

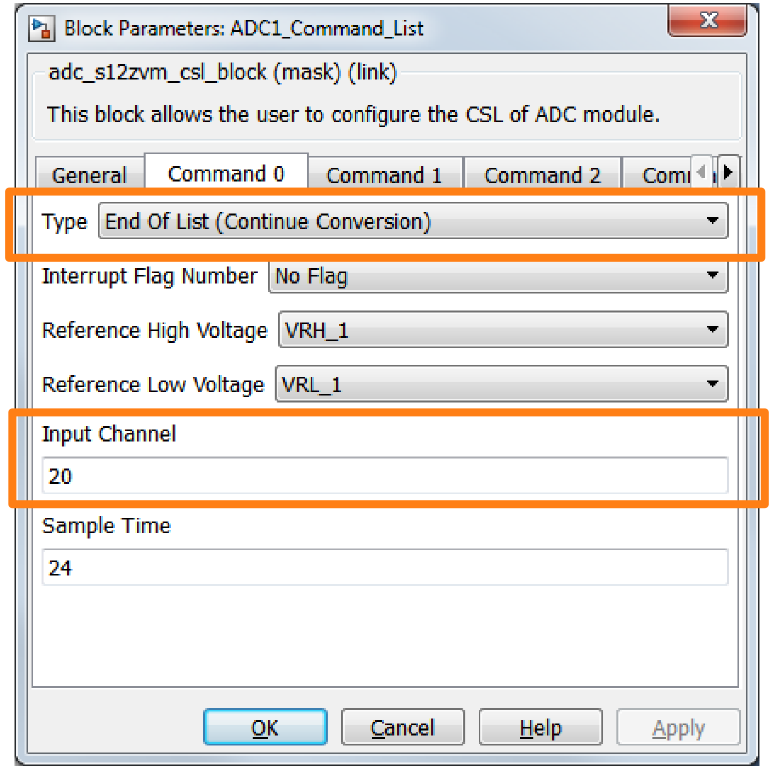




Hands-On Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run

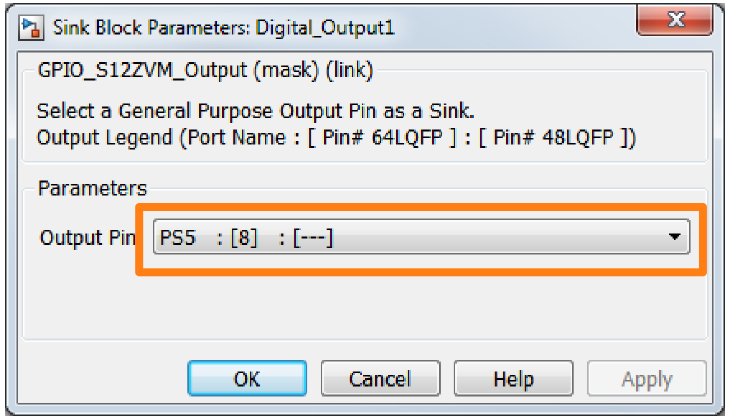
19. Open “ADC Command List” and set input channel to 20.

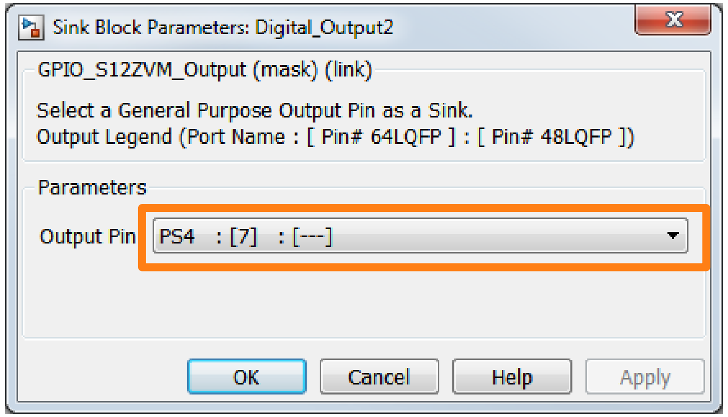


Hands-On Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run

20. Go back to library under Peripheral Interface Blocks, drag in two Digital Output blocks and connect one to output of the comparator and the other to Toggle subsystem.
21. Open Digital Output blocks and select pins

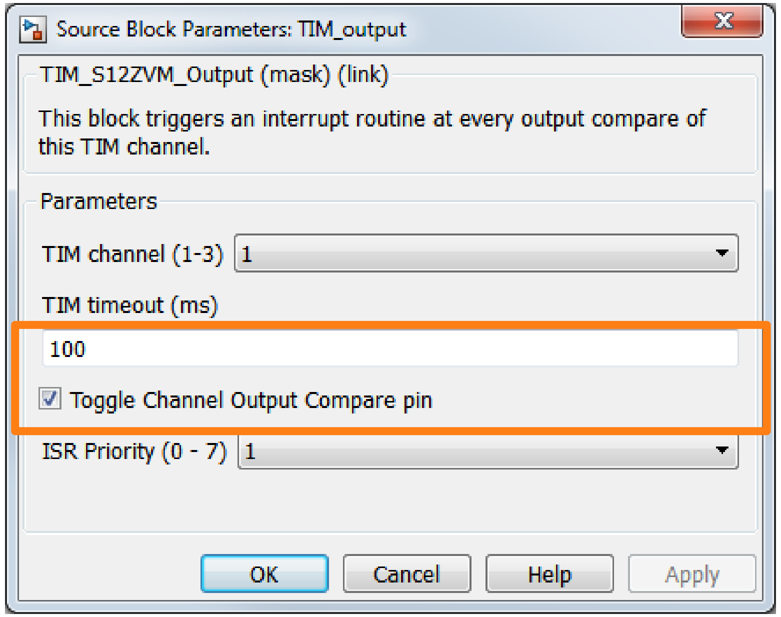




Hands-On Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run

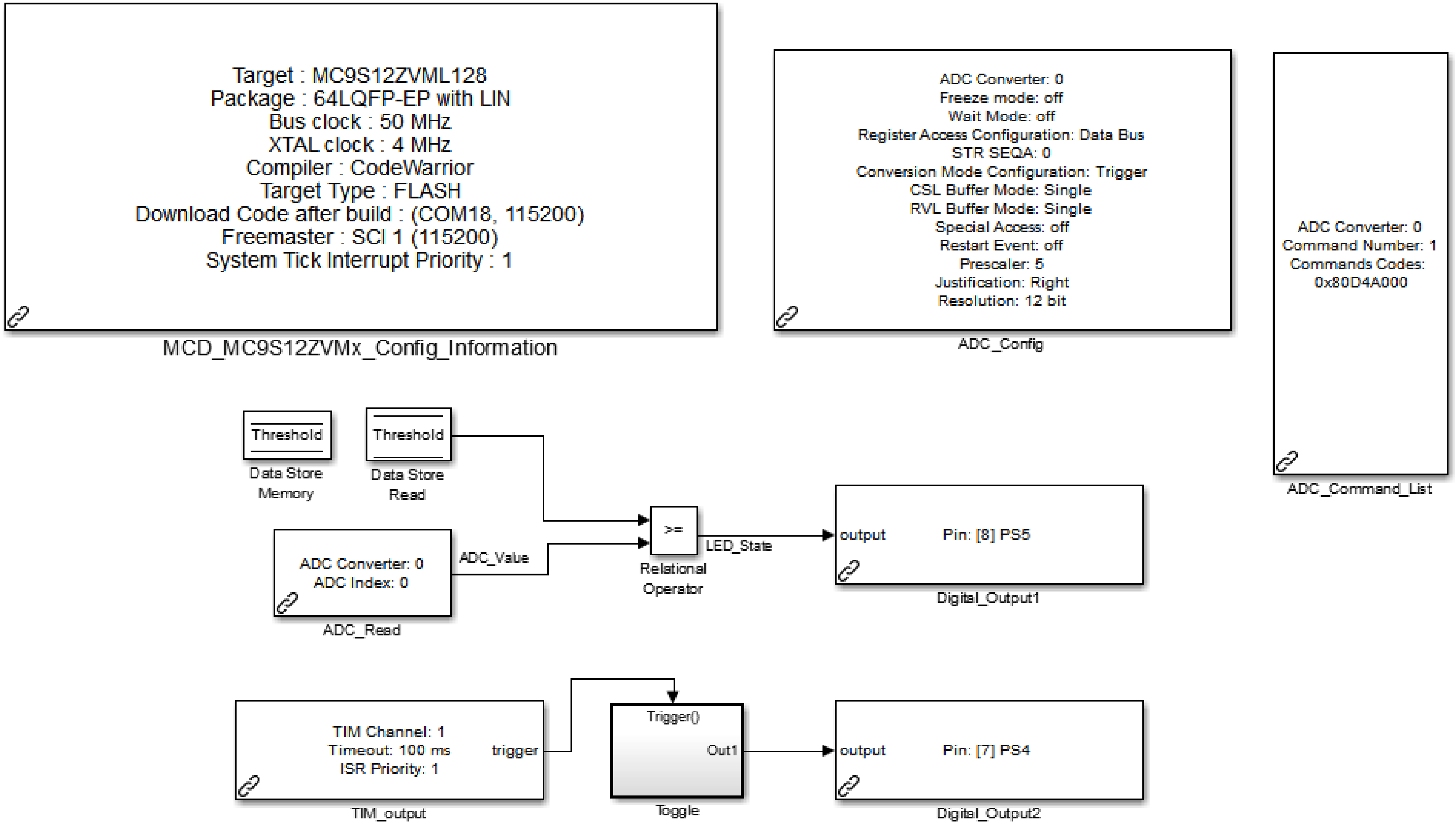
22. Go back to model and delete Function Call block
23. Go back to library under Utility Blocks, drag in a TIM Output block and connect to Trigger of Toggle subsystem.
24. Open TIM output block and set the timeout to 100 ms.



Hands-On Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run

- This is what the model should look like after step 24



Hands-On Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run

25. Go to Code -> C/C++ Code pull down menu and then select Build Model.
26. Wait for model to generate code and then a prompt from the RAppID Bootloader Utility will appear. Reset the MCU and then select “OK”.
27. Once the download is complete you should observe an LED blinking.
28. Turn the Potentiometer on the Motor Kit from right to left. You should observe the LED turn ON and OFF when turning the POT from one stop to the other. Conversion of the model is complete!

Hands-On Demo: Read A/D and Toggle LED Simple Model

Using FreeMASTER with Hands-On Demo

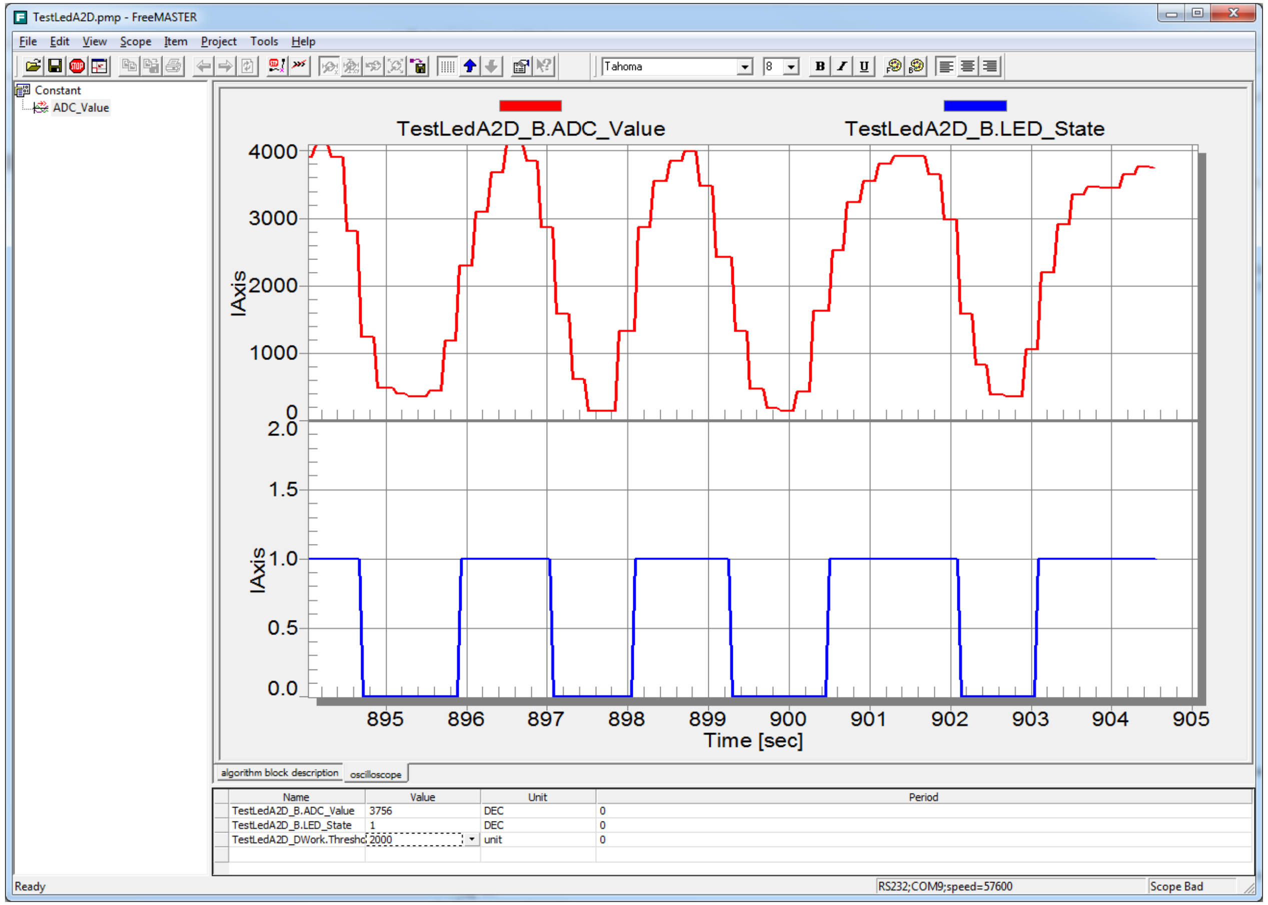
29. Start FreeMASTER and open project TestLedA2D.pmp. Just press OK if a message comes up that the map file has been updated.
30. Go to Project Options Pull Down and select “Options”. Verify that COM settings are the same as what were set in your model.
31. Once the COM settings are correct, press the STOP button and start turning the Potentiometer back and forth. You should see the following (next slide):

Note: You should be able to change the threshold value to something other than 2000. Try changing it and see if the LED_State changes state.

Hands-On Demo: Read A/D and Toggle LED Simple Model

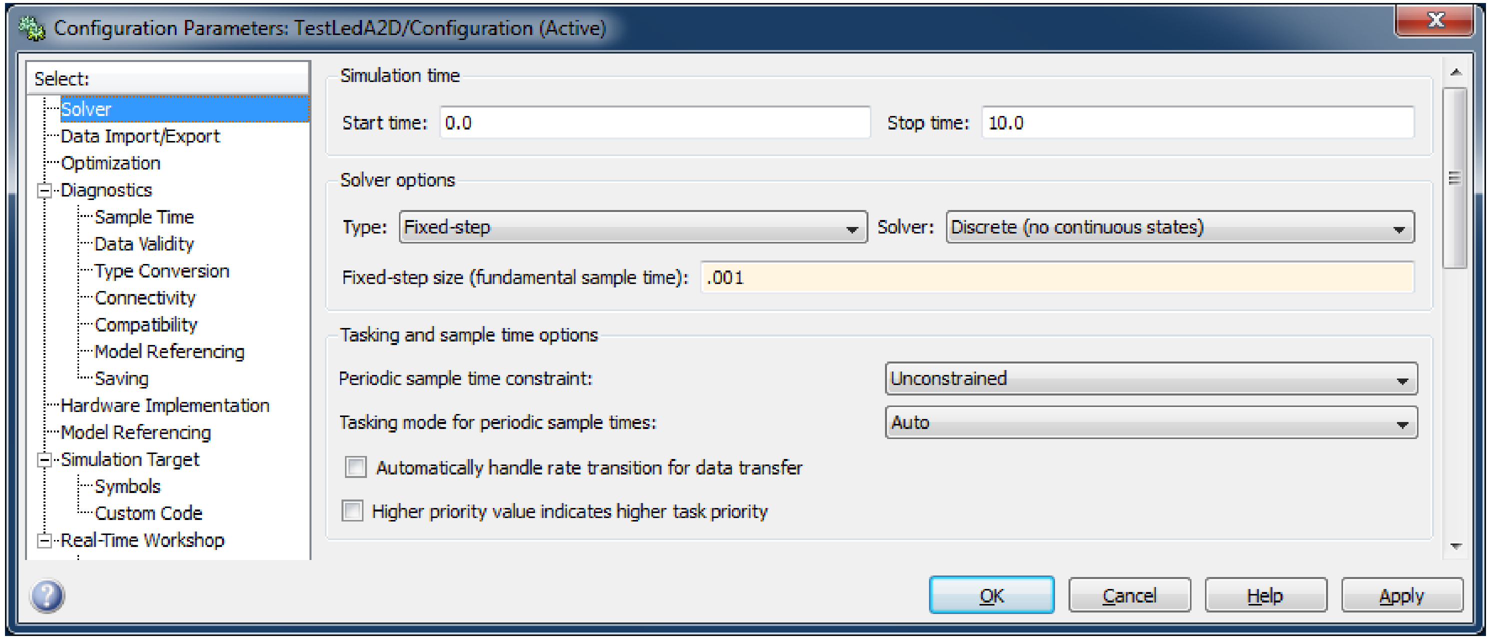
Using FreeMASTER with Hands-On Demo

- This is what you should see after step 31



Using FreeMASTER with Hands-On Demo

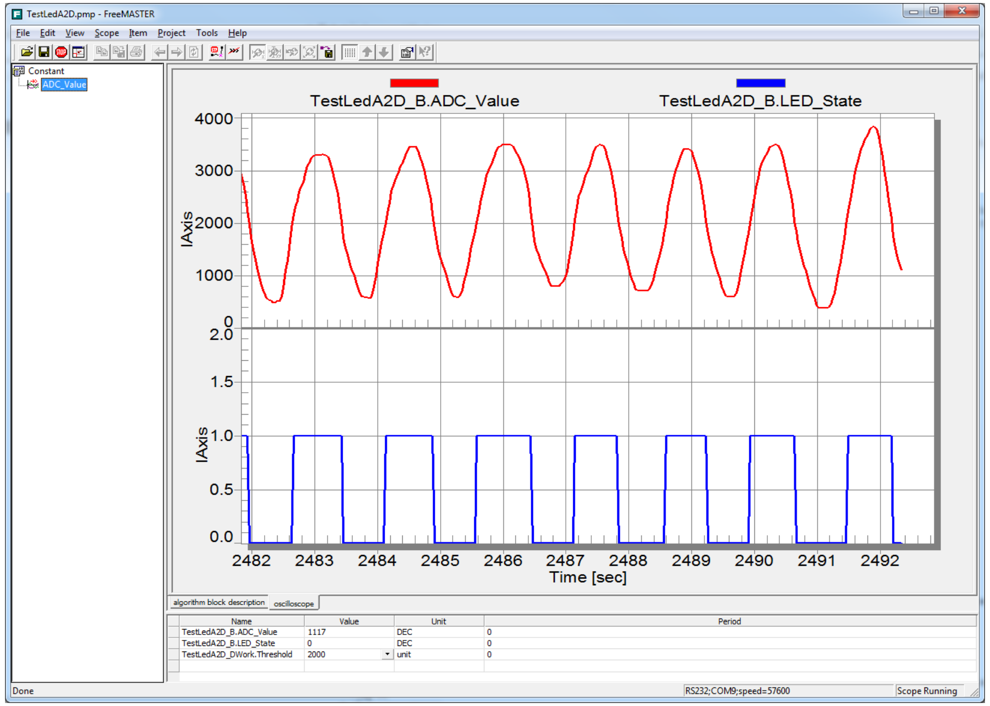
32. You will notice that there is dither in the A2D reading as you change the Potentiometer. This is because the system tick time in the model is too slow. To change this, go to the model and select the Simulation pull down menu. Then select Configuration parameters. Change the Fixed-step size from “auto” to “.001”



Hands-On Demo: Read A/D and Toggle LED Simple Model

Using FreeMASTER with Hands-On Demo

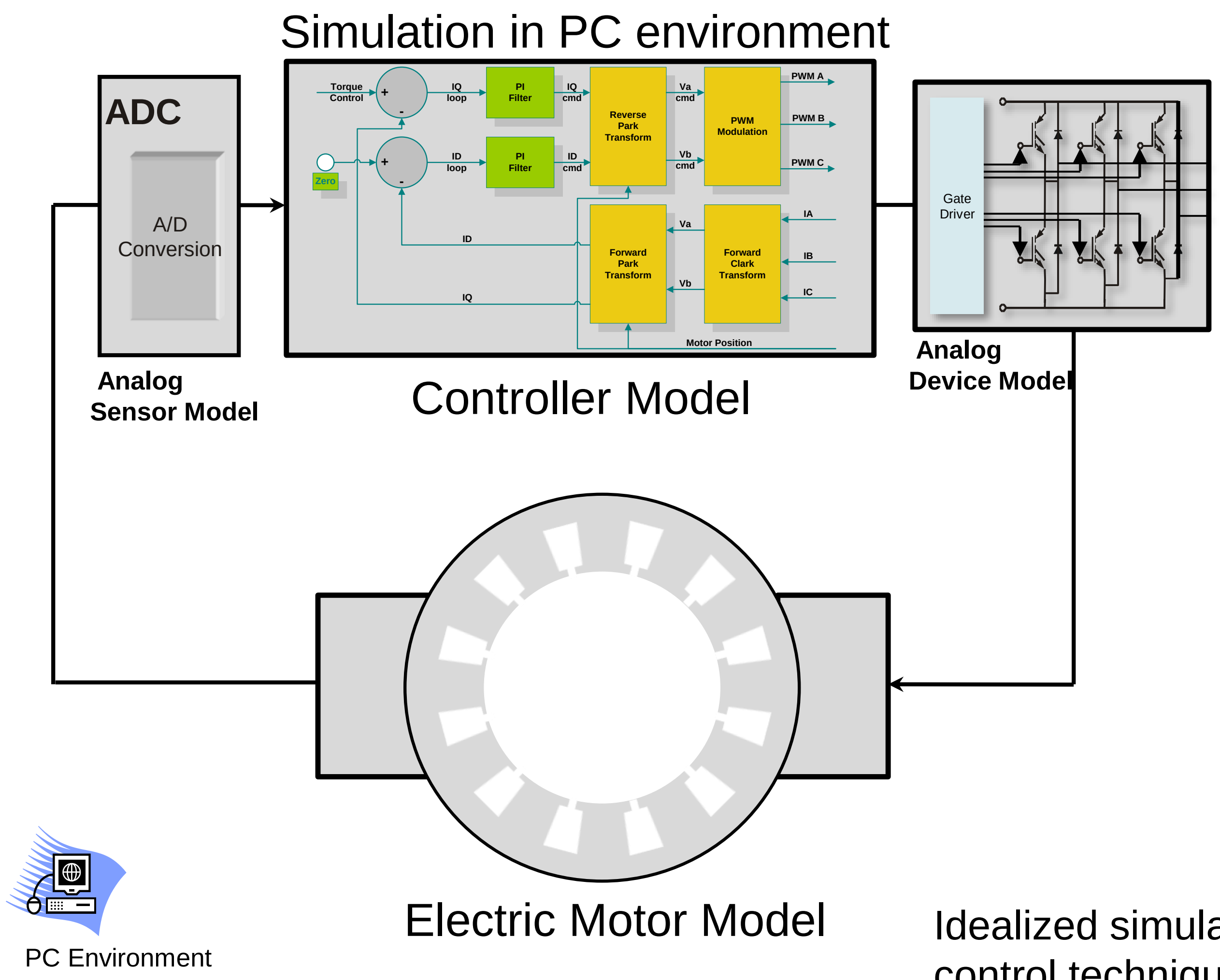
33. Disconnect FreeMASTER by pressing the STOP button. Then rebuild the model and have the bootloader download the software to the MCU. Re-Connect FreeMASTER and turn the Pot. You should see the following:

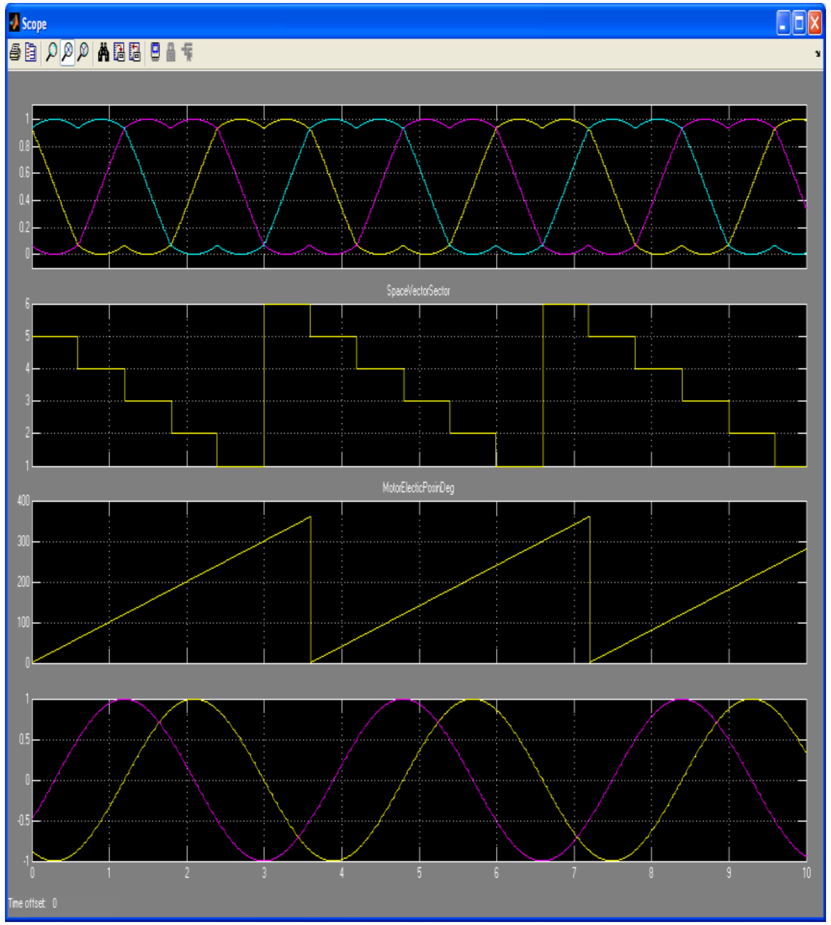


Agenda

- **Overview:**
  - Introduction and Objectives
  - Model-Based Design Toolbox: Library blocks, FreeMASTER, and Bootloader
- **Hands-On Demo:**
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Convert simple model to run on Motor Kit with MCD Toolbox and use FreeMASTER
- **Model-Based Design:**
  - Model-Based Design Steps: Simulation, SIL, PIL and ISO 26262
  - SIL/PIL Hands-On Demo Step 2 & 3 of MBD
- **Trapezoidal Motor Control**:
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Trapezoidal control and how to use it to turn a motor
- **Trapezoidal Motor Control Hands-on Demo**:
  - Implement Trapezoidal Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor and tune parameters
- **FOC Motor Control**:
  - FOC Sensor-less control and how to use it to turn a motor
- **FOC Motor Control Hands-On Demo:**
  - Implement FOC Sensor-less Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor
- **Summary and Q&A:**

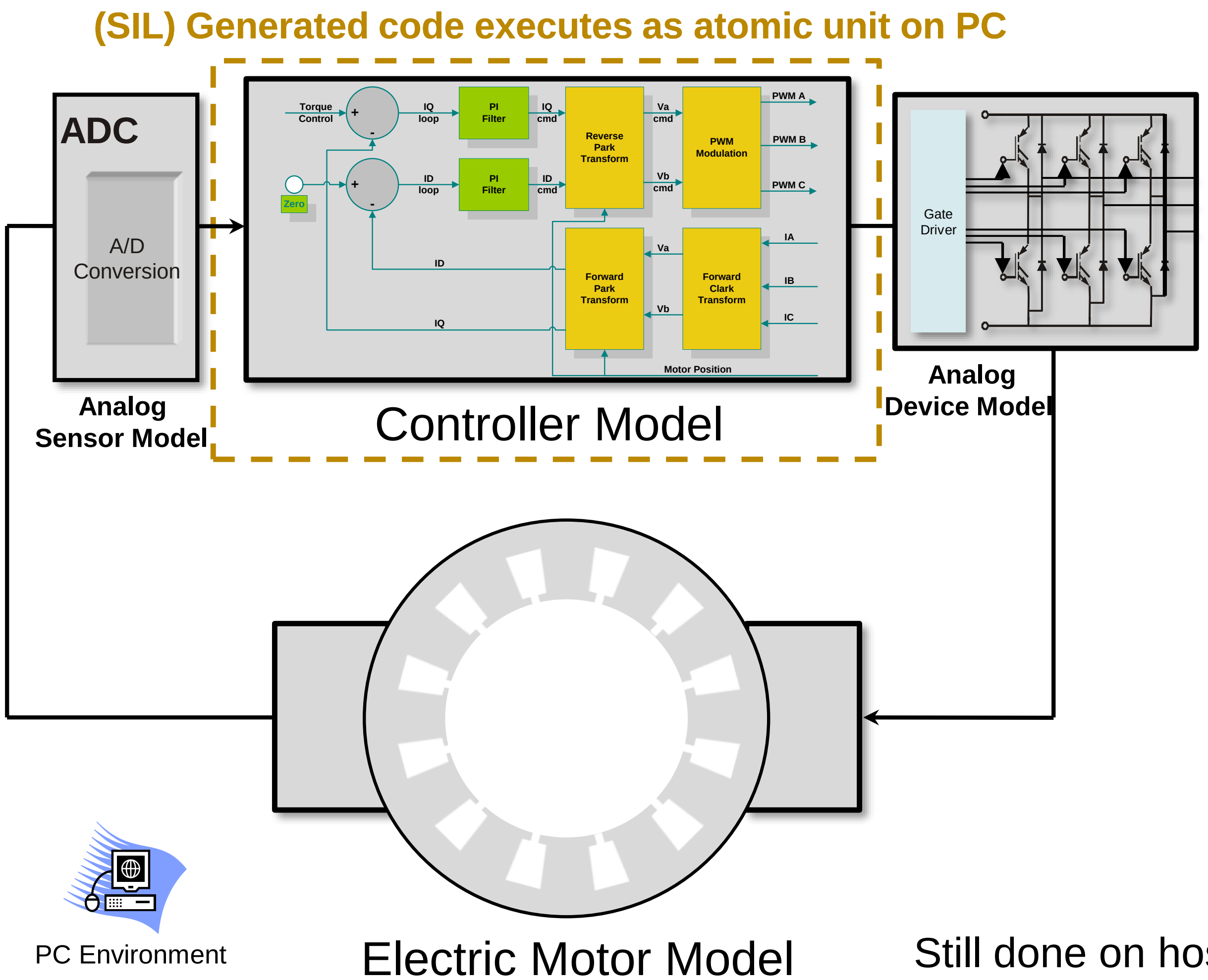
Model-Based Design Steps: Step 1 (Simulation)

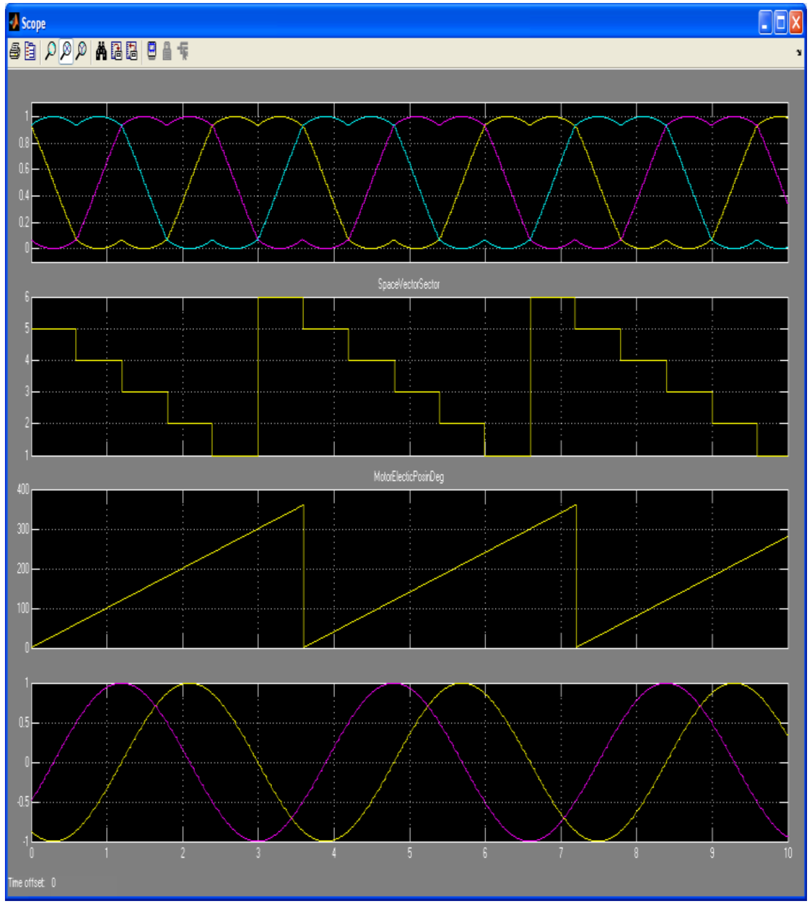




Idealized simulation of the controller and the motor to refine the control technique. Done on host PC without regard for embedded controller. Can optionally add analog device models for fault detection and signal control.

Model-Based Design Steps: Step 2 – Software in the Loop (SIL)

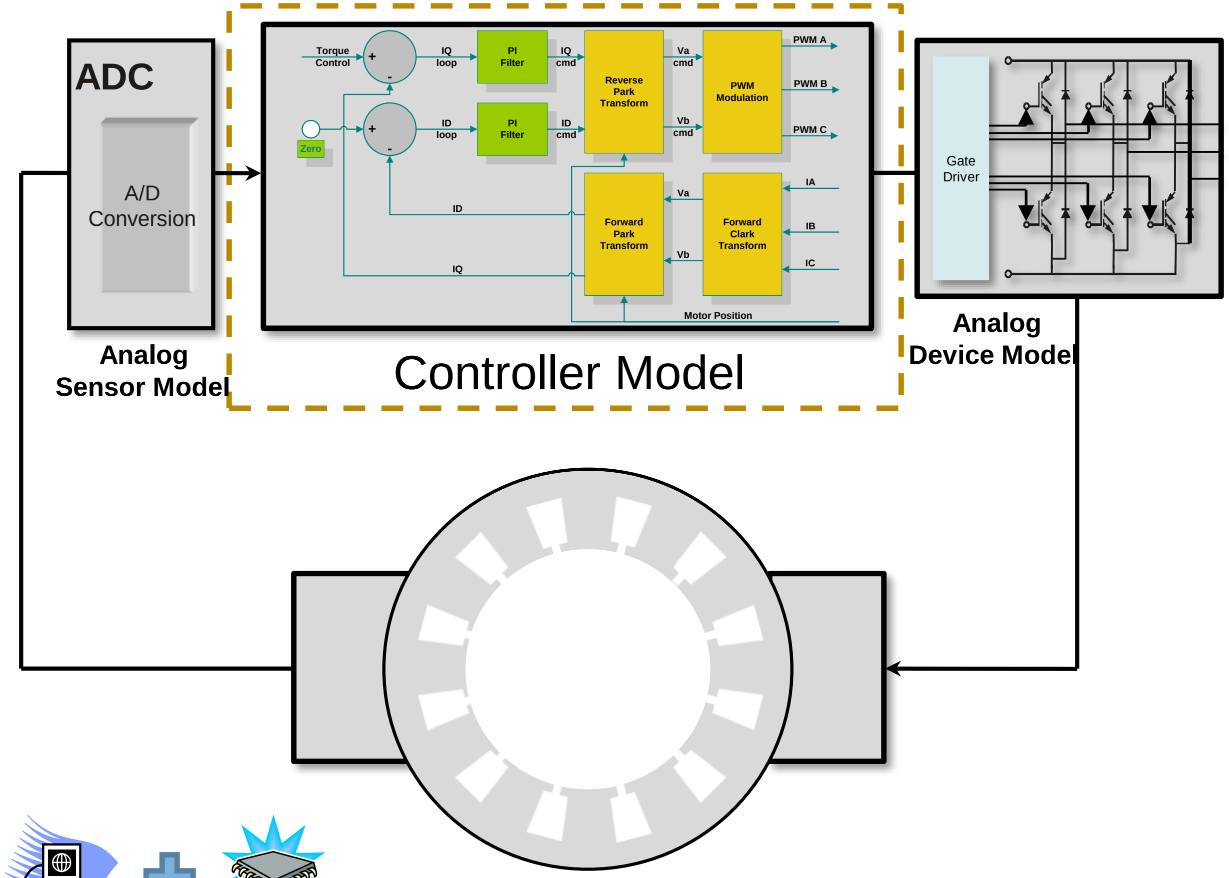


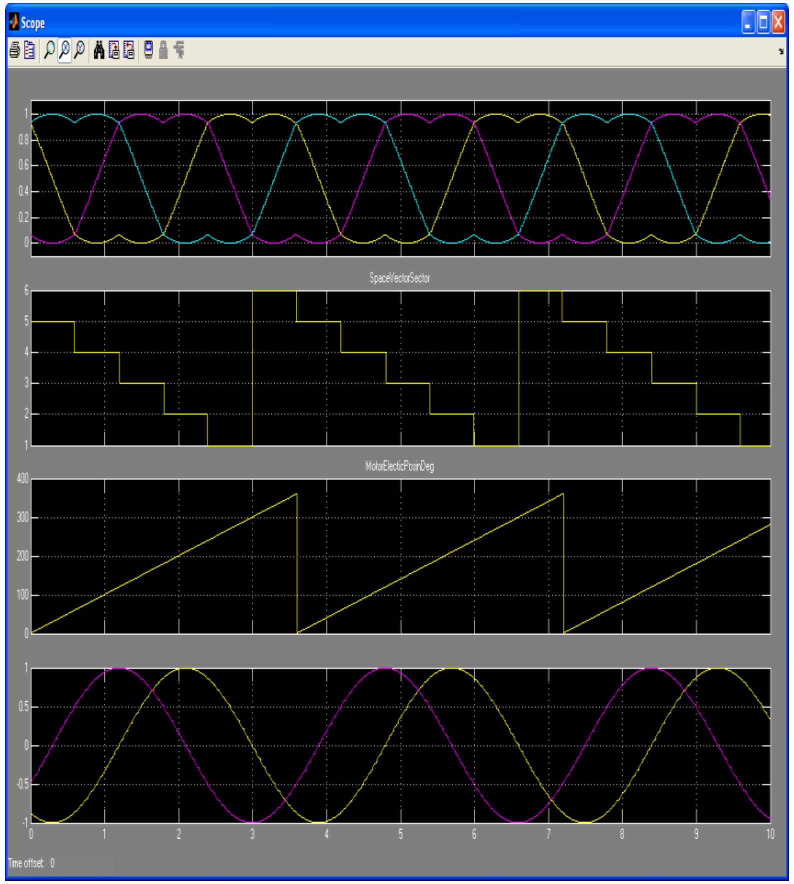


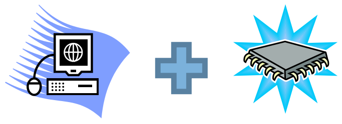
Still done on host PC without regard for embedded controller. Instead using generated C code that is compiled using a PC-based compiler. Run same test vectors as in simulation for C Code Coverage analysis and verify functionality.

Model-Based Design Steps: Step 3 – Processor in the Loop (PIL)

(PIL) Executes generated code on the target MCU







PC Environment + MCU

Electric Motor Model

Execute the model on the target MCU and perform numeric equivalence testing. Co-execution with MCU and Model-Based Design working together while collecting execution metrics on the embedded controller of control algorithm. Validate performance on MCU.

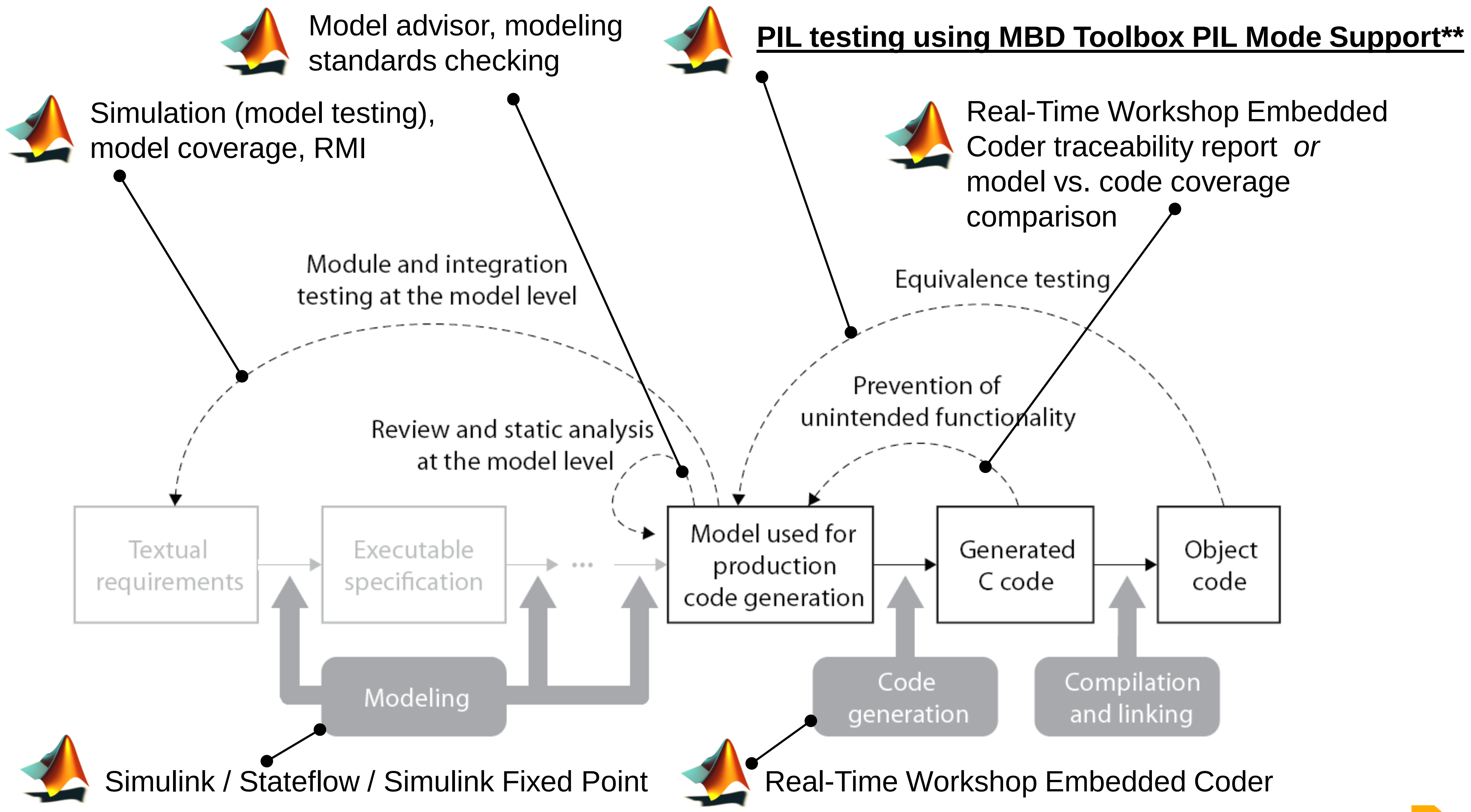
Model-Based Design Steps: Step 3 (PIL)

Verification and Validation at Code Level

- This step allows:
  - Translation validation through systematic testing
  - To demonstrate that the execution semantics of the model are being preserved during code generation, compilation, and linking with the target MCU and compiler

- Numerical Equivalence Testing:
  - Equivalence Test Vector Generation
  - Equivalence Test Execution
  - Signal Comparison

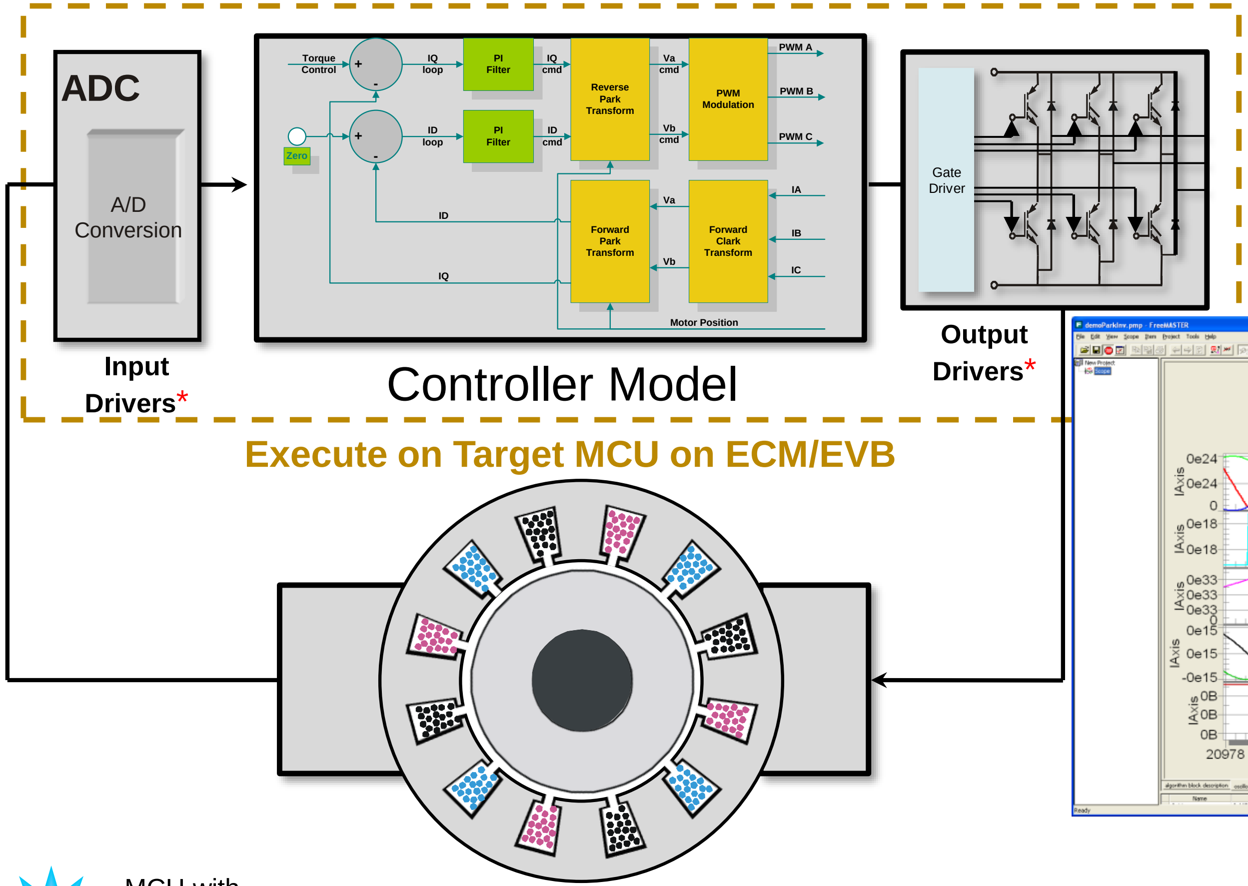
Example IEC 61508 and ISO 26262 Workflow for Model-Based Design with MathWorks Products*



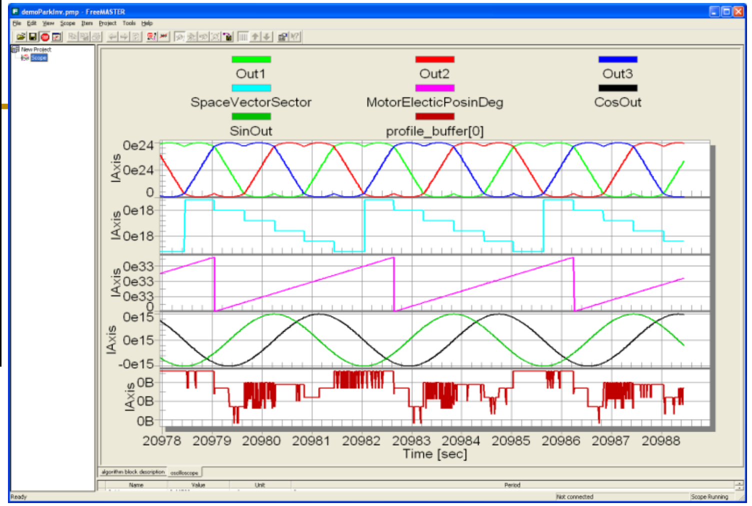
*Workflow from The Mathworks™ Presentation Material Model-Based Design for IEC 61508 and ISO 26262

** NXP MBD Toolbox is part of Mathworks Workflow outlined in The Mathworks™ Material Model-Based Design for IEC 61508 and ISO 26262 as well as part of certification qualification tool suite.

Model-Based Design Steps: Step 4 (Target MCU)*



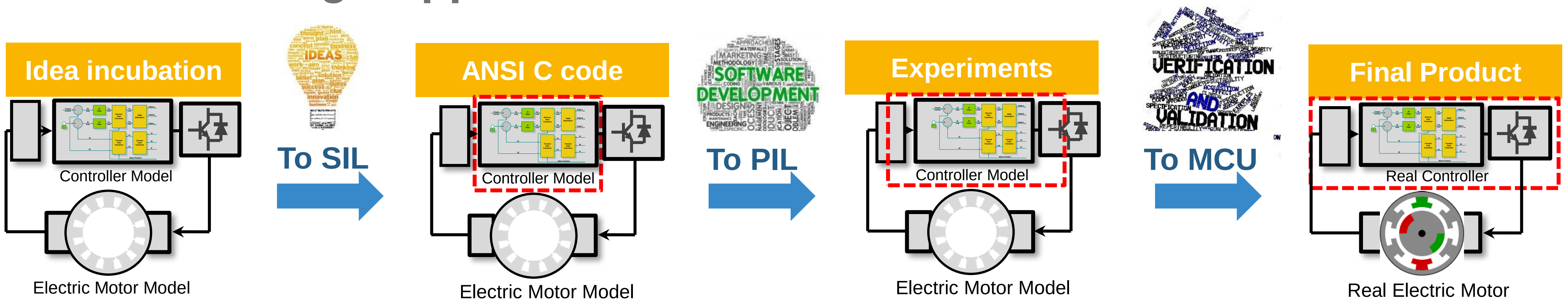
* I/O peripheral driver blocks can be included in the model, providing the analog driver interfaces needed to directly interface to devices external from the MCU.



MCU with Embedded Control Module (ECM)

Generate production code to run on embedded MCU with real motor while collecting execution metrics on the embedded controller of control algorithm. Validate performance on MCU and use FreeMASTER to tune control parameters and perform data logging.

Model-Based Design Approach



PC Environment

PC Environment

PC Environment + MCU

MCU with Embedded Control Module (ECM)

Step 1 – System Requirements:

MBD Simulation Only

- Software requirements
- Control system requirements
- Overall application control strategy

- Modeling style guidelines applied
- Algorithm functional partitioning
- Interfaces are defined here

Step 2 – Modeling/Simulation:

MBD Simulation with ANSI C Code using SIL

- Control algorithm design
- Code generation preparation
- Control system design
- Overall application control strategy design
- Start testing implementation approach

- Testing of functional components of algorithm
- Test harness to validate all requirements
- Test coverage of model here
- Creates functional baseline of model

Step 3 – Rapid Prototype:

MBD Simulation with ANSI C Code using PIL

- Controller code generation
- Determine execution time on MCU
- Verify algorithm on MCU
- See memory/stack usage on MCU
- Start testing implementation approach
- Target testing controls algorithm on MCU

- Refine model for code generation
- Function/File partitioning
- Data typing to target environment done here
- Scaling for fixed point simulation and code gen Testing of functional components of algorithm
- Test harness to validate all requirements
- Test coverage of model here
- Creates functional baseline of model
- Equivalence testing

Step 4 – Target MCU Implementation

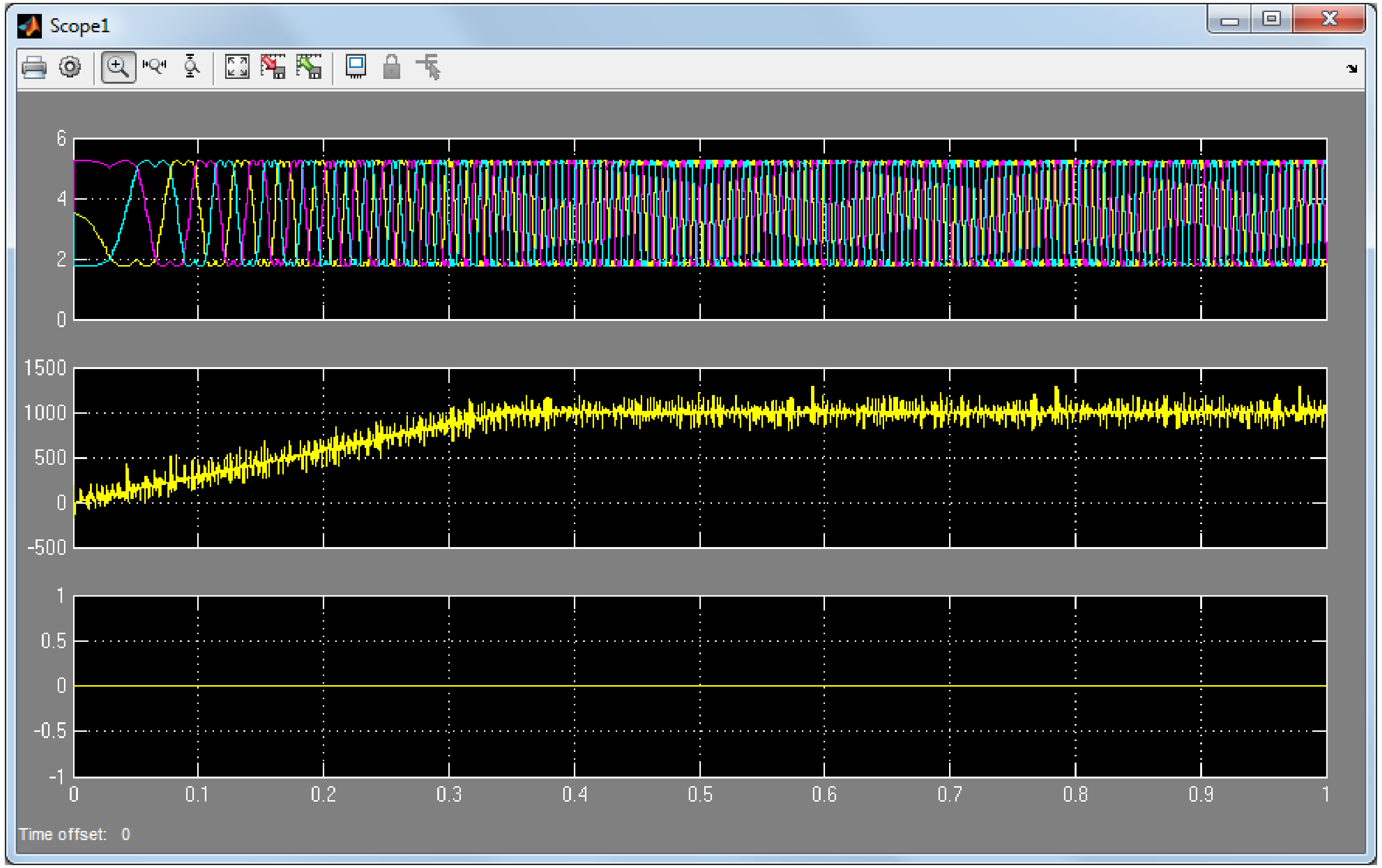
ANSI C Code Running on Target HW & MCU

- Validation/verification phase
- Controller code generation
- Determine execution time on MCU
- Start testing implementation on target ECM
- Code generate control algorithm
- Test system in target environment Utilize calibration tools for data logging and parameter tuning

- Execute code on target MCU
- Functional testing in target environment
- Ensure execution on target is correct as well as code generation on target is performing as desired.

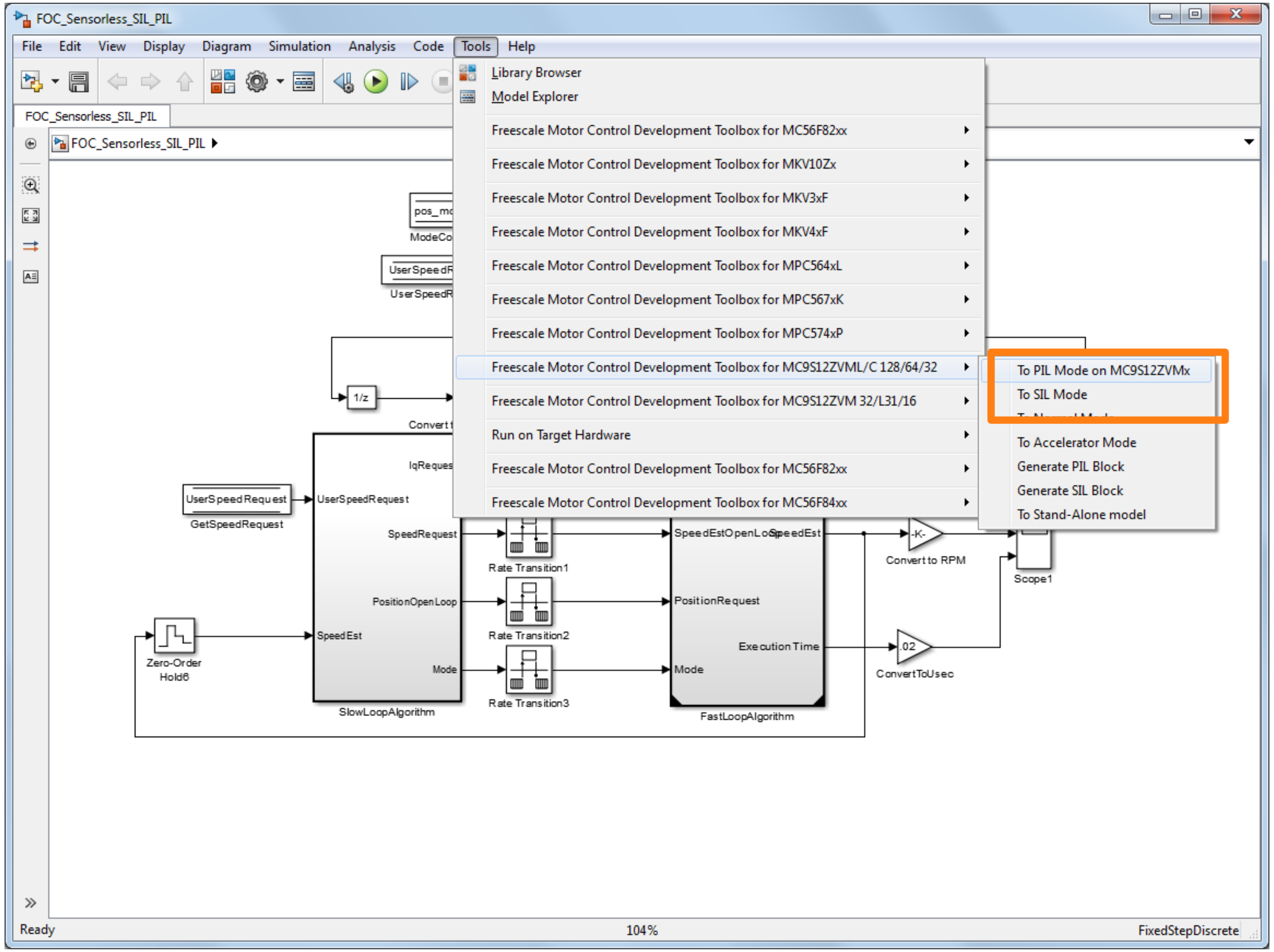
Demo: SIL/PIL Step 2 & 3 of MBD

1. Open Model “FOC_Sensorless_SIL_PIL.slx
2. You will see a motor simulation of an FOC control algorithm
3. Will Run model and view the results.



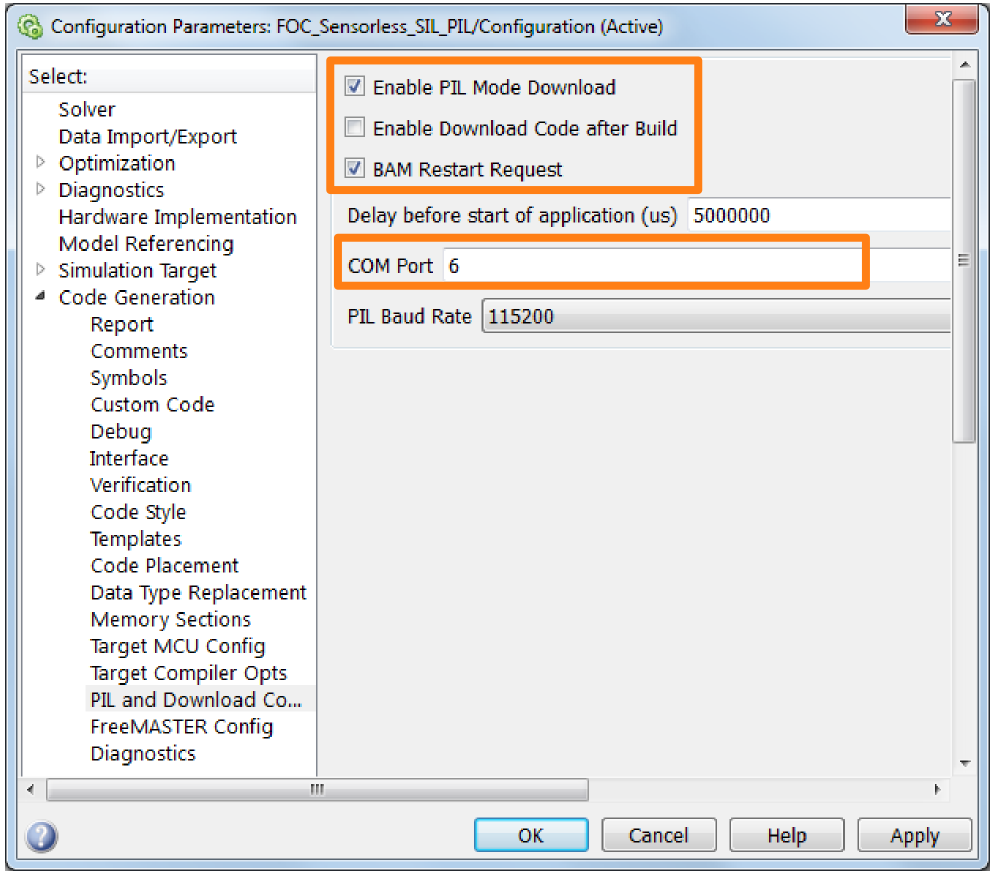
Demo: SIL/PIL Step 2 & 3 of MBD

4. You can switch between SIL and PIL thru using the tools menu.



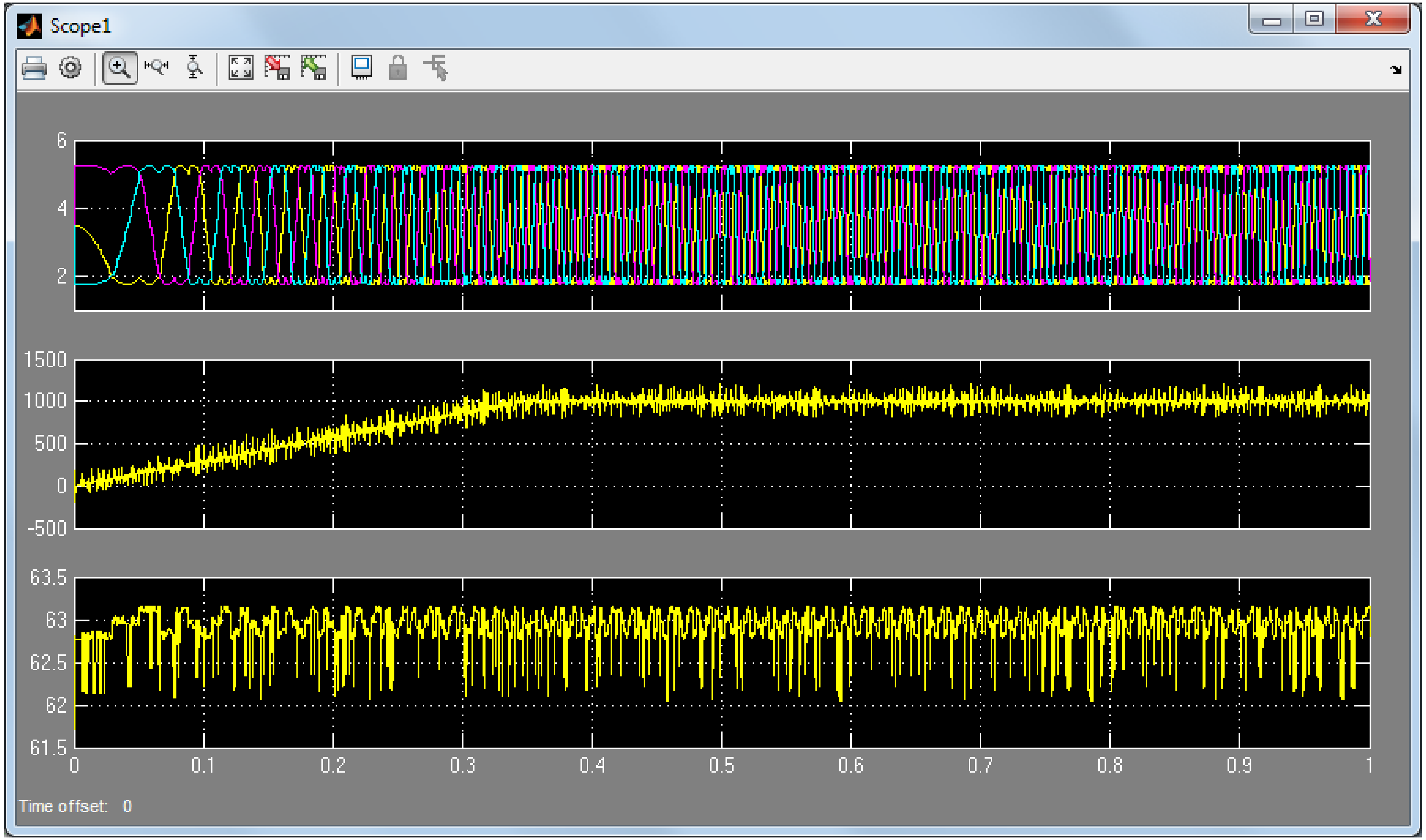
Demo: SIL/PIL Step 2 & 3 of MBD

5. Open “Configuration Parameters” in the reference model.
6. Go to PIL/BAM Setup tab.
7. Enter the COM port number that you are using from PC.



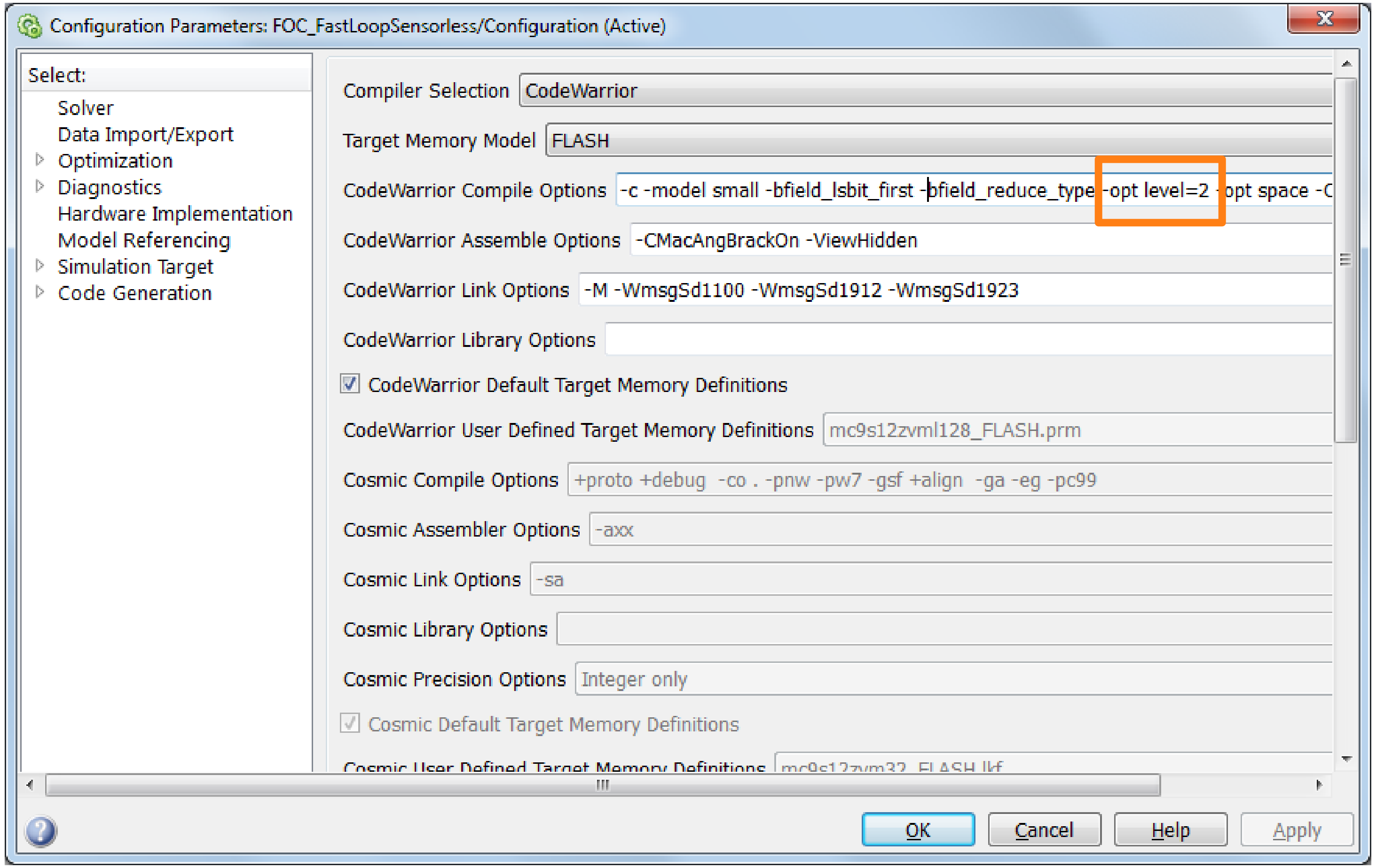
Demo: SIL/PIL Step 2 & 3 of MBD

8. - Will Run model and view the results.



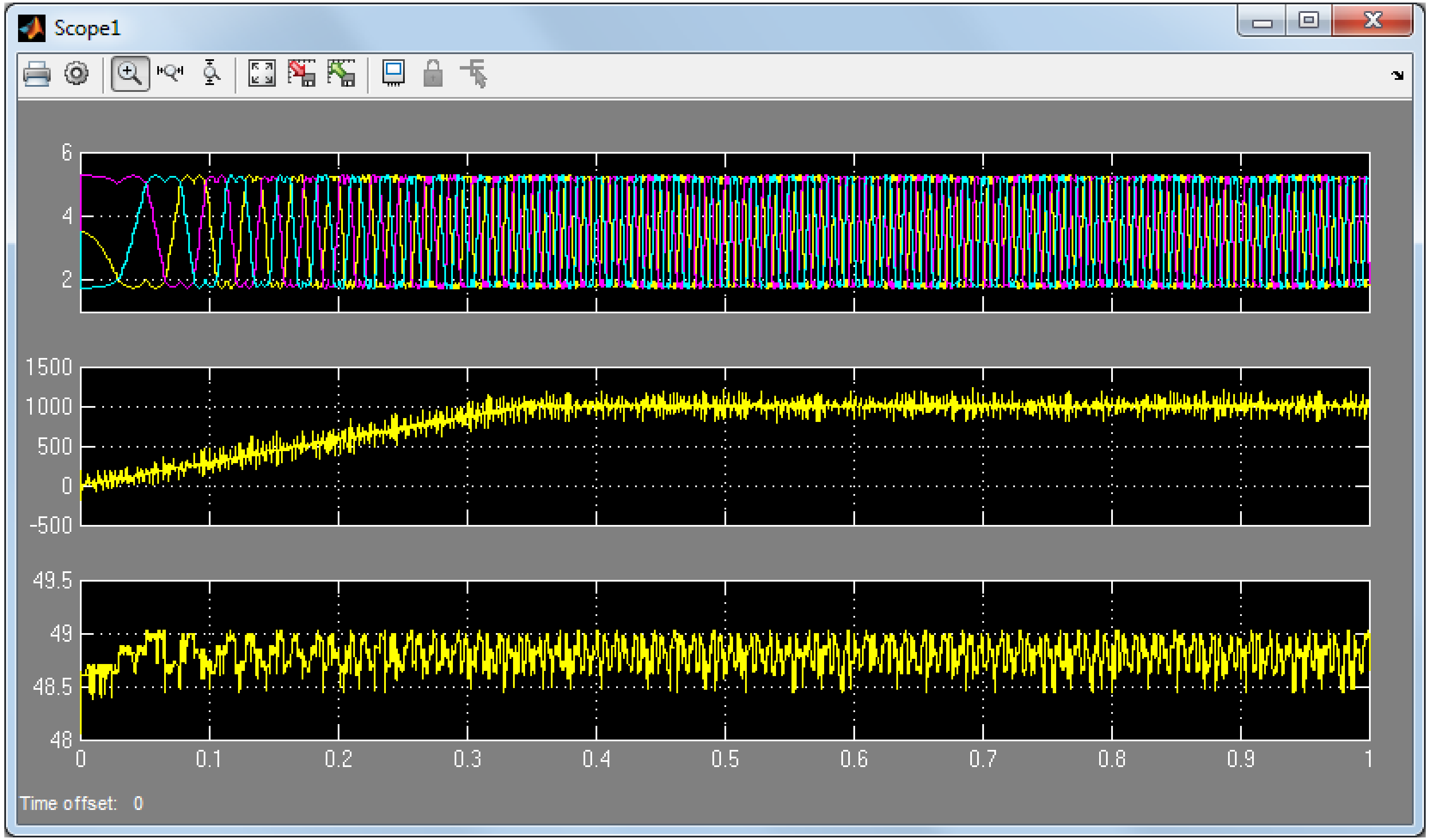
Demo: SIL/PIL Step 2 & 3 of MBD

9. Let us try improving the execution time by changing the compiler options
10. Change the optimization level from 0 to 2.



Demo: SIL/PIL Step 2 & 3 of MBD

11. Run model and view the results.



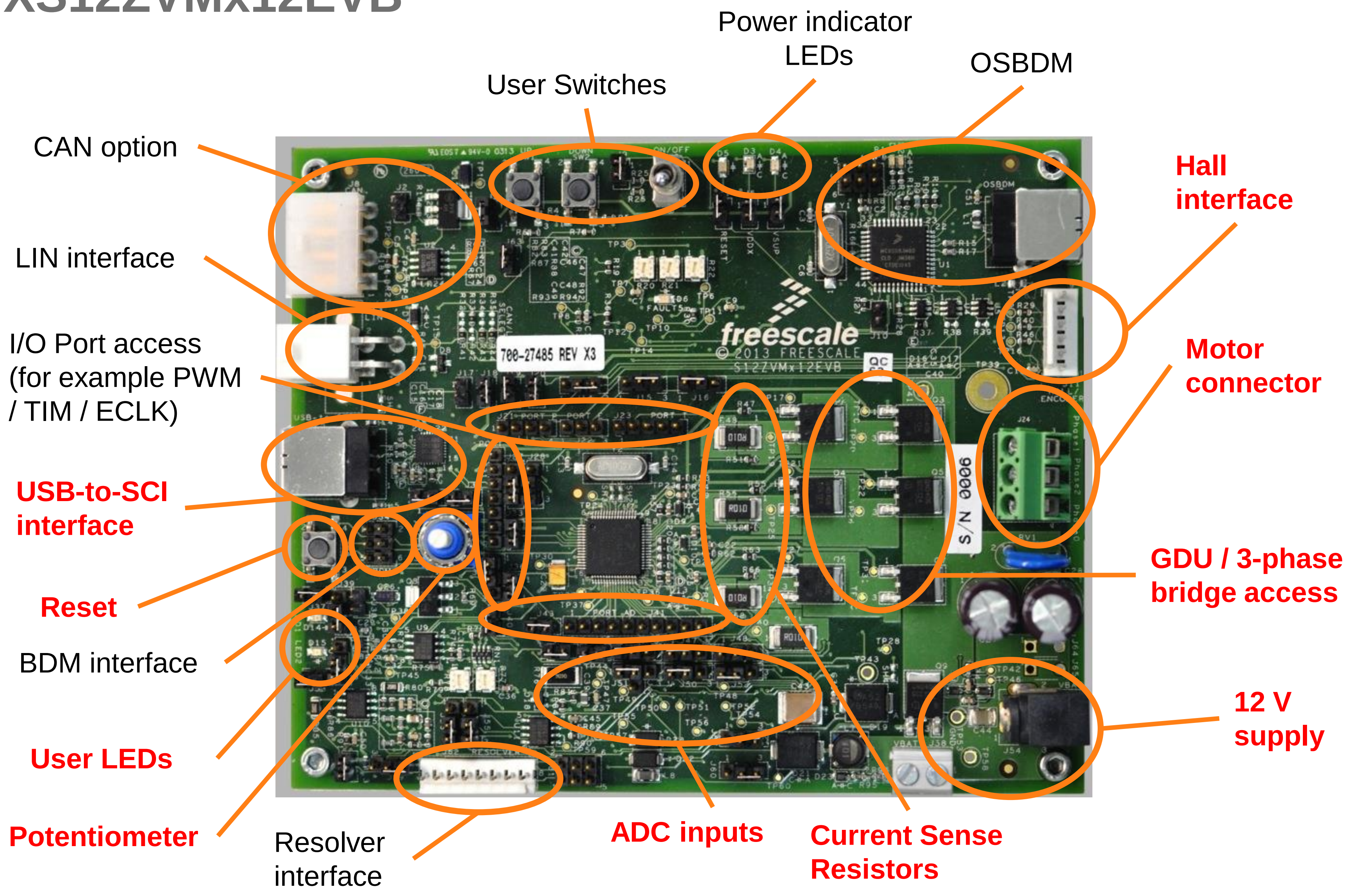
Agenda

- **Overview:**
  - Introduction and Objectives
  - Model-Based Design Toolbox: Library blocks, FreeMASTER, and Bootloader
- **Hands-On Demo:**
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Convert simple model to run on Motor Kit with MCD Toolbox and use FreeMASTER
- **Model-Based Design:**
  - Model-Based Design Steps: Simulation, SIL, PIL and ISO 26262
  - SIL/PIL Hands-On Demo Step 2 & 3 of MBD
- **Trapezoidal Motor Control**:
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Trapezoidal control and how to use it to turn a motor
- **Trapezoidal Motor Control Hands-on Demo**:
  - Implement Trapezoidal Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor and tune parameters
- **FOC Motor Control**:
  - FOC Sensor-less control and how to use it to turn a motor
- **FOC Motor Control Hands-On Demo:**
  - Implement FOC Sensor-less Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor
- **Summary and Q&A:**

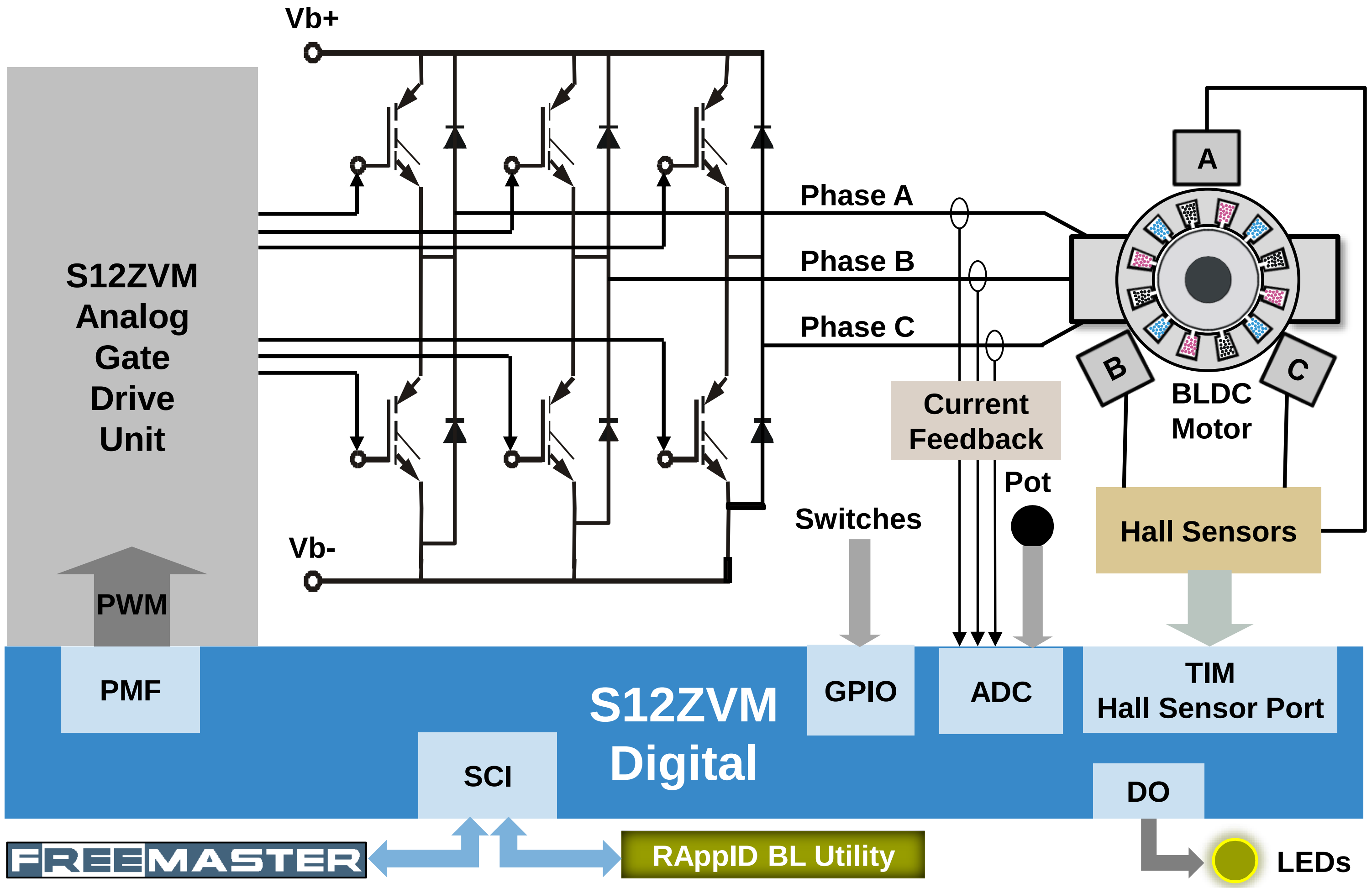
Motor Kit: MTRCKTSBNZVM128 BLDC Motor Control Kit

- The kit includes a 4 pole-pair count motor, which means that every single mechanical revolution equals four electrical revolutions. State changes in Hall sensors is every 60 degrees electrical.

Motor Kit: XS12ZVMx12EVB



Motor Kit: System Diagram



Motor Kit: S12ZVM for BLDC Motor Control

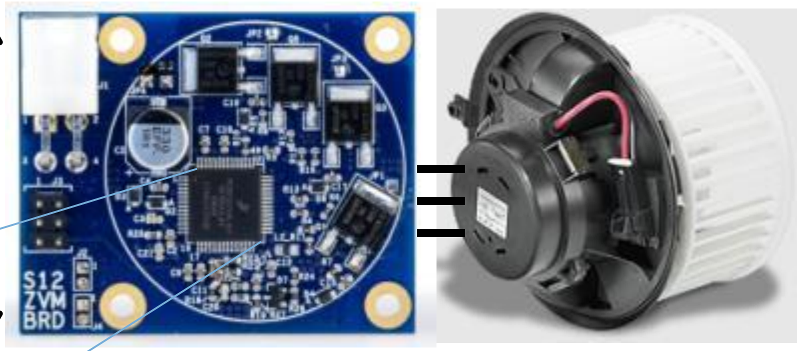
Discrete Solution

VREG (8pin) — 3+ — MCU or DSC (48pin) — 20+ — Gate Driver (48pin)

LIN phy (8pin) — 2+ — MCU or DSC (48pin)

Op-amps

4 cm
~1 ½ in.

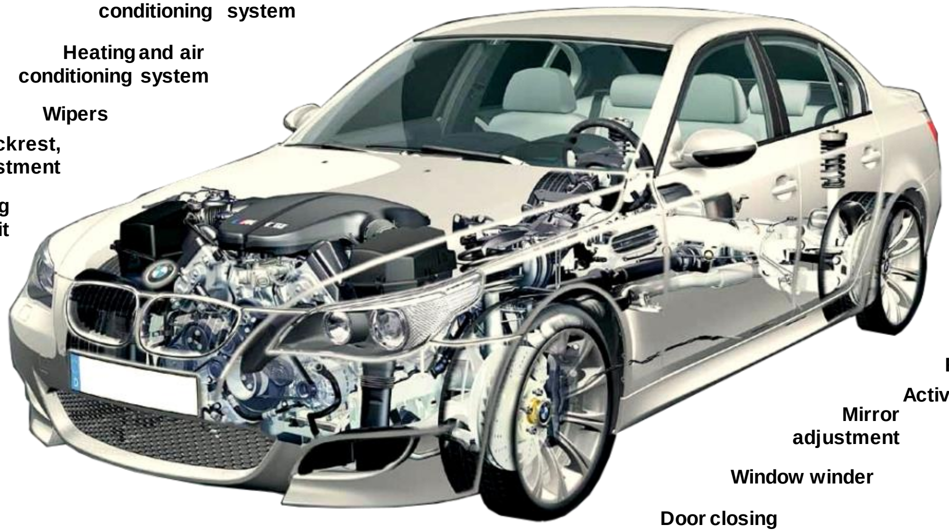


64 pin

S12ZVM Solution:

- ~ 50 fewer solder joints
- - 4 to 6 cm2 PCB space

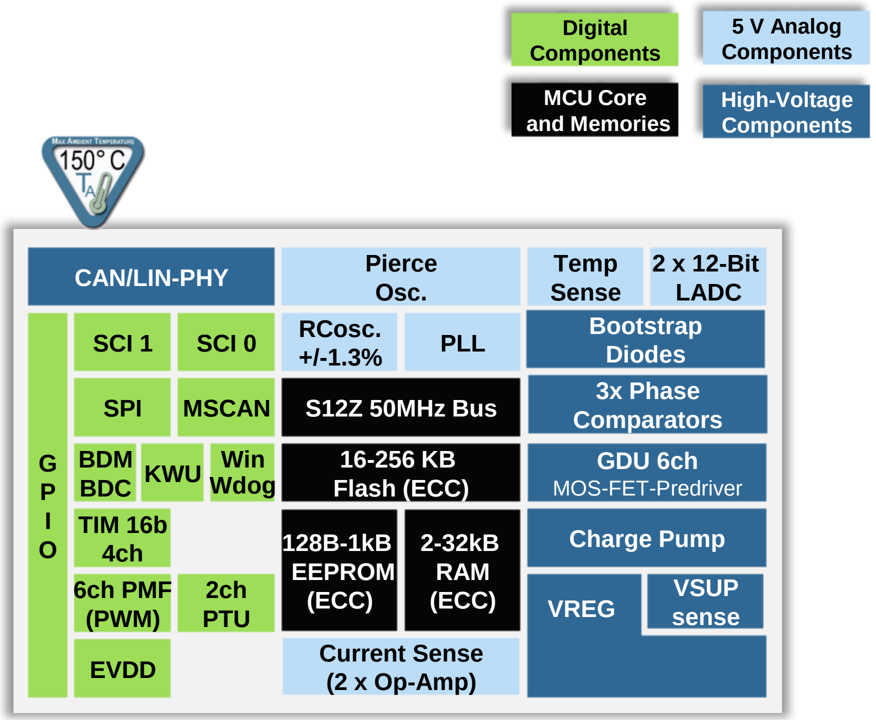
10 Billion Electric Motors Shipped Globally in 2013
2.5 Billion in Automobiles, 30 Per Car Average



Motor Control

Motor Kit: S12ZVM Family

BLDC/PMSM/SR motor control

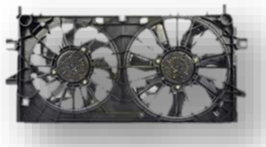


Key Features:

- S12Z CPU @ 50 MHz bus speed
- 6 ch. Gate Drive Unit (GDU) with 50-150 nC total Gate Charge drive capability, incl charge pump for High-Side, Bootstrap diodes for charging external bootstrap capacitors
- Embedded Vreg with switchable 5V/20 mA sensor supply
- LIN PHY, LIN2.1 / 2.2 / J2602 compliant
- Dual 12-bit list-based ADC (LADC), synch with PWM through Programmable Trigger Unit (PTU)
- 2x Op-amp for current sensing

Target applications:

- Sensorless BLDC or PMSM motor control
- Switched Reluctance Motor
- Bidirectional DC motors (H-Bridge)
- Various pumps (oil, fuel, water, vacuum)
- Cooling fan, HVAC blower, Turbocharger



Options:

- Package: 64-LQFP-EP, 48 LQFP-EP, 80-LQFP-EP
- Memory: 16 kB / 32 kB / 64 kB / 128 kB / 256 kB Flash
- Spec-Options:
  - L with LIN phy
  - C with CAN-PHY (256 kB only)
  - C with 2nd Vreg for external CAN phy (128/64 kB)
  - “ “ with High Voltage PWM-communication interface
- Temperature: V / M / W (up to 150 °C Ta per AEC-Q100 Grade 0)

Motor Kit: S12ZVML *(LIN Version) — Details*

Digital Components
5V Analog Components
MCU Core and Memories
High-Voltage Components

2x UARTs
One linked to LIN Phy, 2nd as independant Test Intf.

SPI
As alternative test Interf or for peripherals (sensors)

MSCAN 2.0A/B
CAN Controller

Up to 18 Wake-up pins
Combined with Analog Input pins

4 ch. 16-bit Timer
Hall Inputs, software timing

6-ch. PMF
15-bit PWM for motor control with dead time, fault mgmt

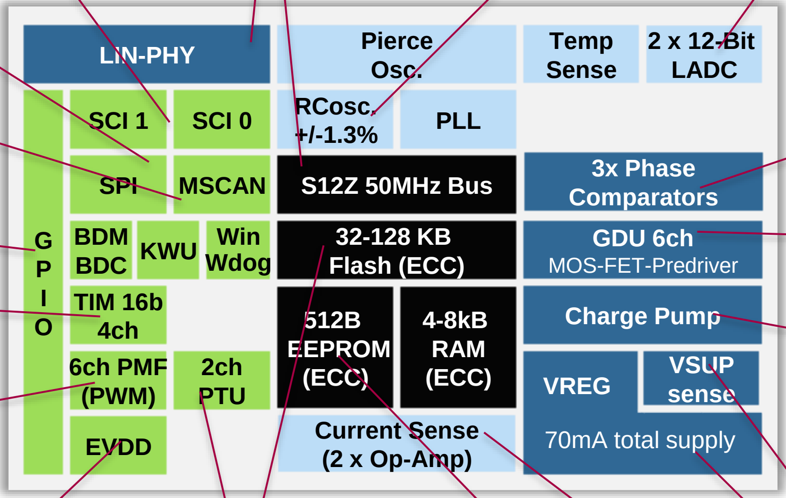
External Supply
5 V / 20 mA switchable for local (same PCB), over current protected.
Eg. supplying Hallsensors

LIN Physical Layer
LIN2.2 and SAE J2602 compliant +/- 8 kV ESD capability

S12Z CPU
16-bit, 32-bit MAC, linear addressing Harvard architech compatible within S12 MagniV family

On-chip RC OSC
factory-trimmed to +/-1.3%, meets LIN -needs

2 x 12-bit list based ADC
Simultaneous measurement 5+4 ch. external. Plus 8 ch. int (temp sence, GDU phase, Ref voltages) with DMA



3x Phase Comparators
for BEMF zero crossing detection in sensorless BLDC

6-ch. GDU
Low side and high side FET pre-drivers for each phase with 100-150 nC total gate Charge

Charge Pump
To support reverse battery protection and boostrap assist for 100% duty-cycle

Vsup sense
Monitoring supply voltage

Voltage Regulator
5V/70 mA for whole system

2x Op-Amp for current measurement / sensing

PTU
Enables synchronization between PMF and ADC

Flash (32/64/128 kB)
512 B erasable. 10 K p/e cycles. Can be used for Data

EEPROM
4 byte eraseable, 100 K program / erase cycles

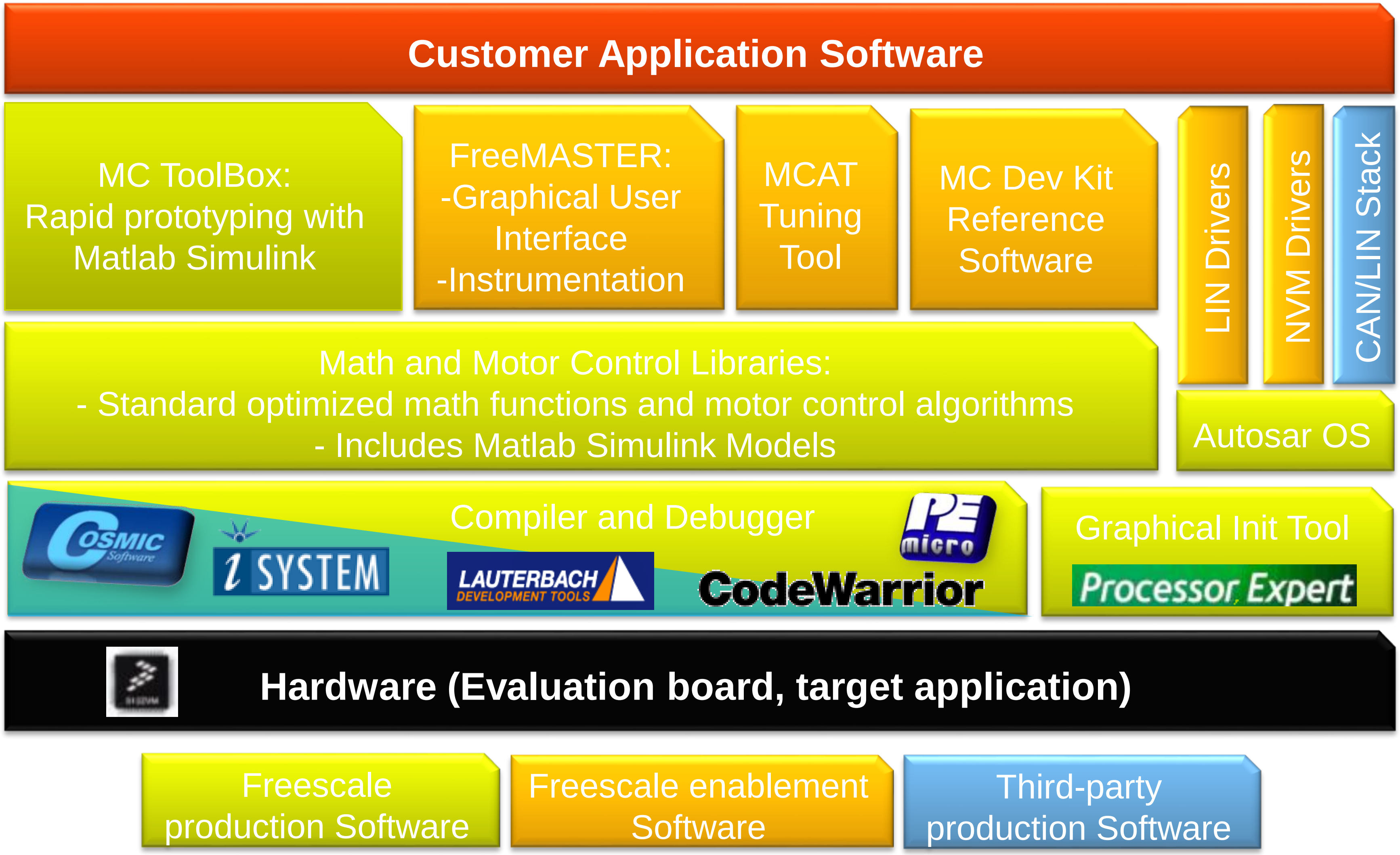
AEC-Q100 Grade 0
Qual'ed up to 150 °C Ta

Packaging Option
64LQFP-EP

Motor Kit: S12ZVM Family Feature Set Summary

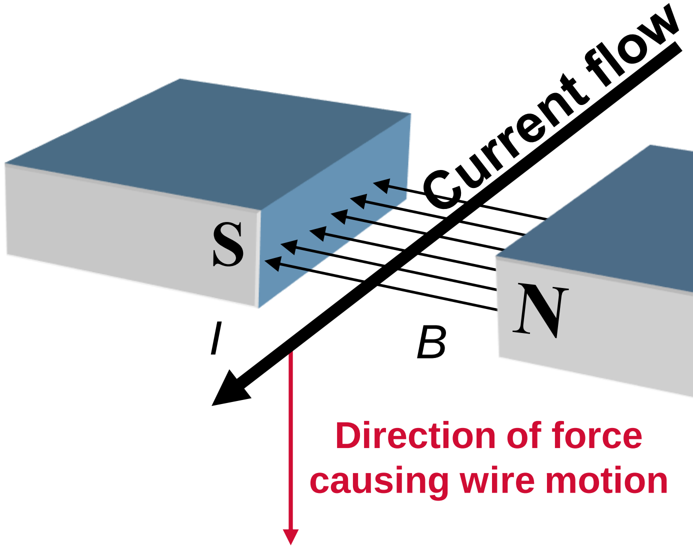
Connectivity	CAN	LIN	CAN	LIN	CAN	LIN			PWM			
Product Name	**VMC256**	**VML128**	**VMC128**	**VML64**	**VMC64**	**VML32**	**VML31**	**VML31**	**VM32**		**VM16**	
Package	80LQFP-EP	64LQFP-EP	64LQFP-EP	64LQFP-EP	64LQFP-EP	64LQFP-EP	64LQFP-EP	48LQFP-EP	64LQFP-EP	48LQFP-EP	64LQFP-EP	48LQFP-EP
EEPROM (bytes)	1K	512	512	512	512	512	128	128	128	128	128	128
PHY	CAN	LIN	0	LIN	0	LIN	LIN	LIN	HV	HV	HV	HV
Separate VREG	1+1	0	1	0	1	0	0	0	0	0	0	0
GDU (HS / LS)	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3
Bootstrap Diodes	0	0	0	0	0	0	3	3	3	3	3	3
Op Amp	2	2	2	2	2	2	2	1	2	1	2	1
ADC (ext. channels)	8 + 8	4 + 5	4 + 5	4 + 5	4 + 5	4 + 5	4 + 5	1 + 3	4 + 5	1 + 3	4 + 5	1 + 3
MSCAN	1	1	1	1	1	1	0	0	0	0	0	0
SCI	2	2	2	2	2	2	2	1	2	1	2	1
SPI	1	1	1	1	1	1	1	0	1	0	1	0
TIM (IC/OC channels)	4	4	4	4	4	4	4	3	4	3	4	3
PWM channels	6+4	6	6	6	6	6	6	6	6	6	6	6
Internal timers	RTI+API	RTI+API	RTI+API	RTI+API	RTI+API	RTI+API	RTI+API	RTI+API	RTI+API	RTI+API	RTI+API	RTI+API
External FET Nominal Total Gate Charge (nC)	100-150	100-150	100-150	100-150	100-150	100-150	50-80	50-80	50-80	50-80	50-80	50-80
Package Size	12 mm x 12 mm	10 mm x 10 mm	10 mm x 10 mm	10 mm x 10 mm	10 mm x 10 mm	10 mm x 10 mm	10 mm x 10 mm	7 mm x 7 mm	10 mm x 10 mm	7 mm x 7 mm	10 mm x 10 mm	7 mm x 7 mm
Samples availability	H2 2015	Now	Now	Now	Now	Now	Now	Q2 2015	Now	Q2 2015	Now	Q2 2015
Production release	H2 2016	Q1 2014	Q1 2014	Q1 2014	Q1 2014	Q1 2014	Q1 2016	Q3 2016	Q1 2016	Q3 2016	Q1 2016	Q3 2016

Motor Kit: S12ZVM Ecosystem — The Complete Solution

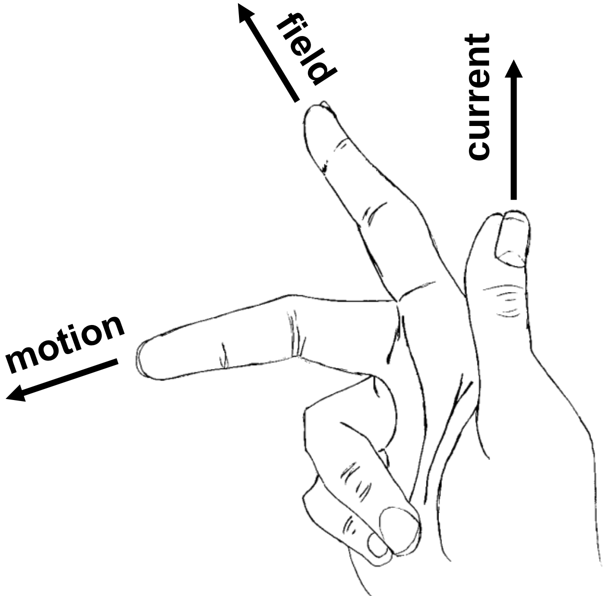


Motor Control: Motion Force Generation

- Current flowing in a magnetic field results in a force on the conductor
- Direction of generated force is governed by “Right Hand Rule” (Lorentz Force Law)

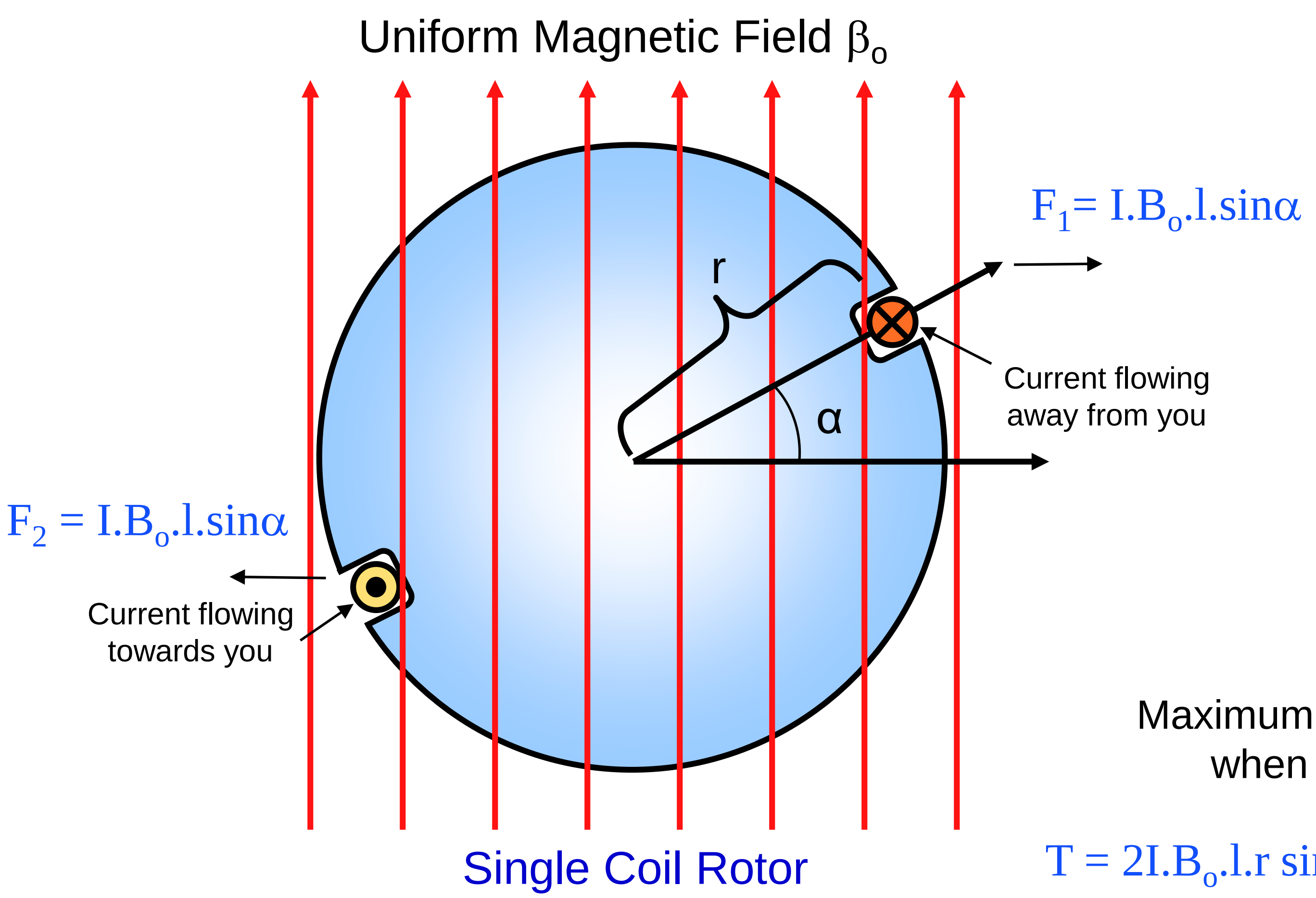


$$F = I.B.l.\sin\alpha$$



Right Hand Rule

Electromagnetic Force Creates Torque

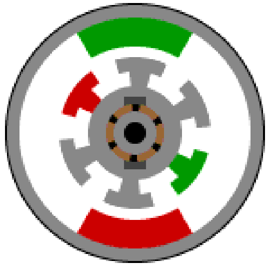


Maximum torque occurs when $\alpha = \pm 90°$

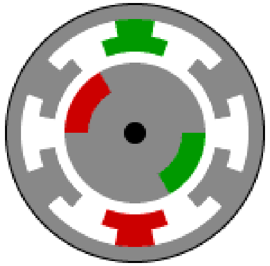
$T = 2I.B_o.l.r \sin\alpha$ (clockwise)

l = length of wire

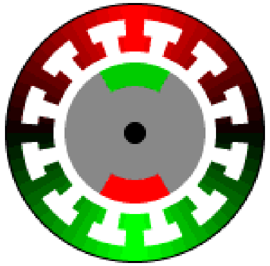
Motor Types



- DC Motors
  - Two or more permanent magnets in stator
  - Rotor windings connected to mechanical commutator



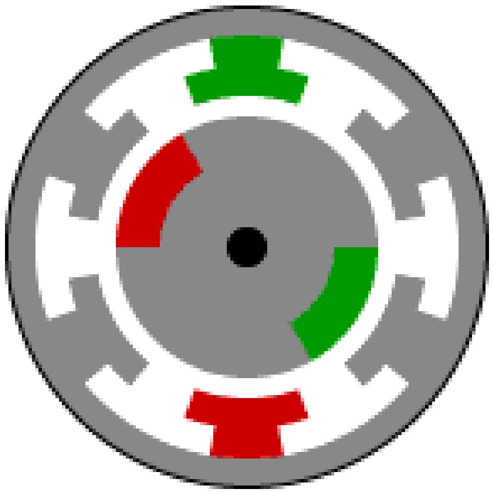
- BLDC Motors
  - PM in rotor, 3-phase conductors in stator
  - Trapezoidal back-EMF

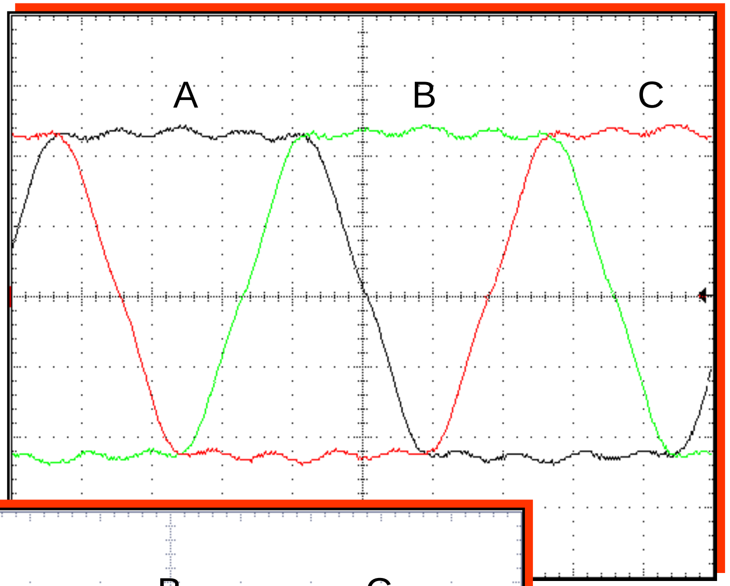


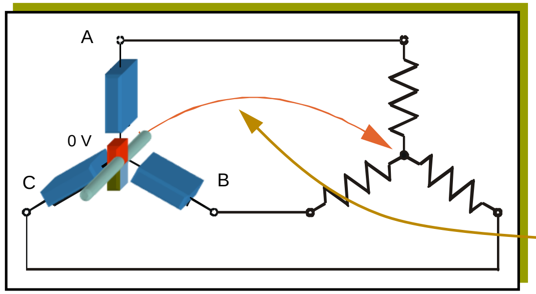
- Permanent Magnet Synchronous Motors
  - Similar to BLDC in construction
  - Sinusoidal back-EMF

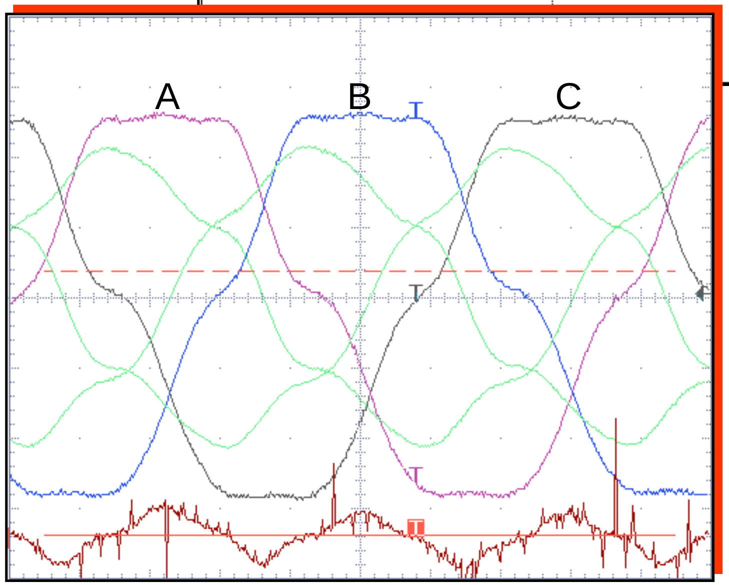
BLDC Motor = Trapezoidal Back-EMF

- ▶ BLDC Motor Commutation
  - one phase is un-powered at any given time









PM Machines – Trapezoidal vs. Sinusoidal

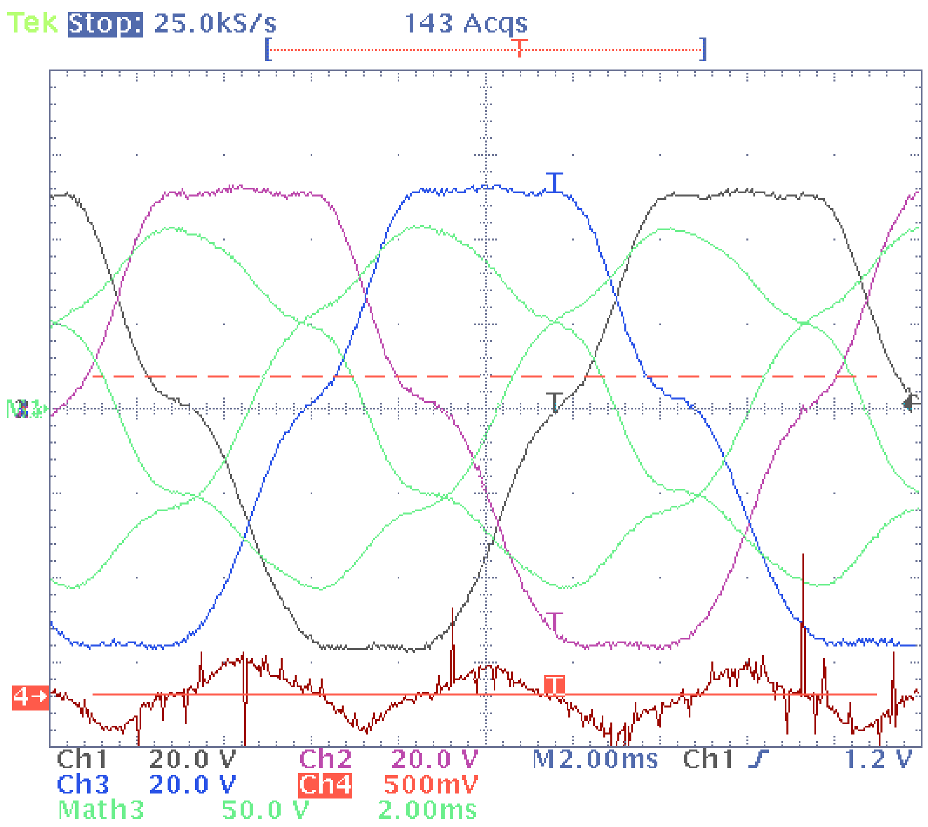
- The characteristic "Trapezoidal" or "Sinusoidal" is linked with the shape of the Back-EMF of the Permanent Magnet motor.
  - "Sinusoidal" means Synchronous (PMSM) motors
  - "Trapezoidal" means Brushless DC (BLDC) motors

- BLDC motor control (6-step control)
  - Only 2 of the 3 stator phases are excited at any time
  - 1 unexcited phase used as sensor (sensorless control)

- Synchronous motor (Field-oriented control)
  - All 3 phases are persistently excited at any time

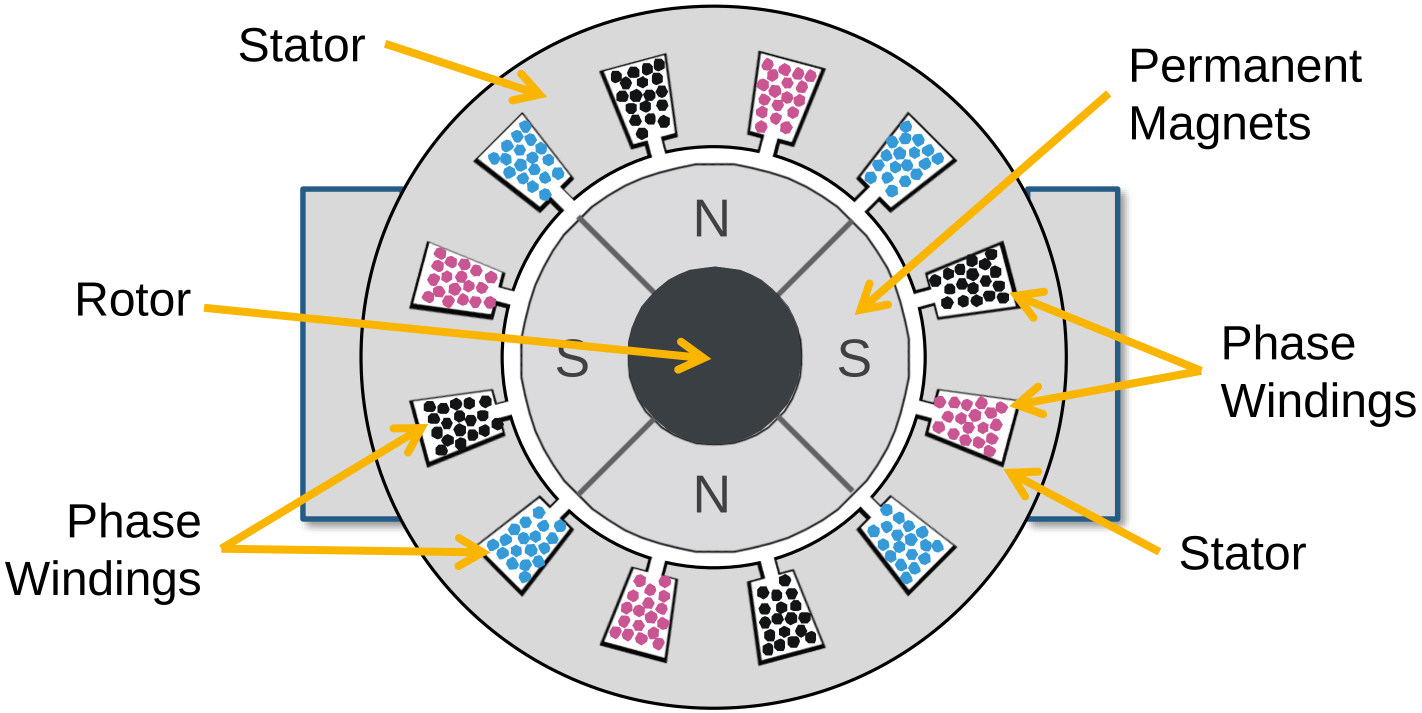
Trapezoidal vs. Sinusoidal PM Machine

- Sinusoidal” or “Sinewave” machine means Synchronous (PMSM)
- Trapezoidal means brushless DC (BLDC) motors
- Differences in flux distribution
- Six-Step control vs. **F**ield-**O**riented **C**ontrol
- Both requires position information
- BLDC motor control
  - 2 of the 3 stator phases are excited at any time
  - 1 unexcited phase used as sensor (BLDC Sensorless)
- Synchronous motor
  - All 3 phases persistently excited at any time
  - Sensorless algorithm becomes complicated

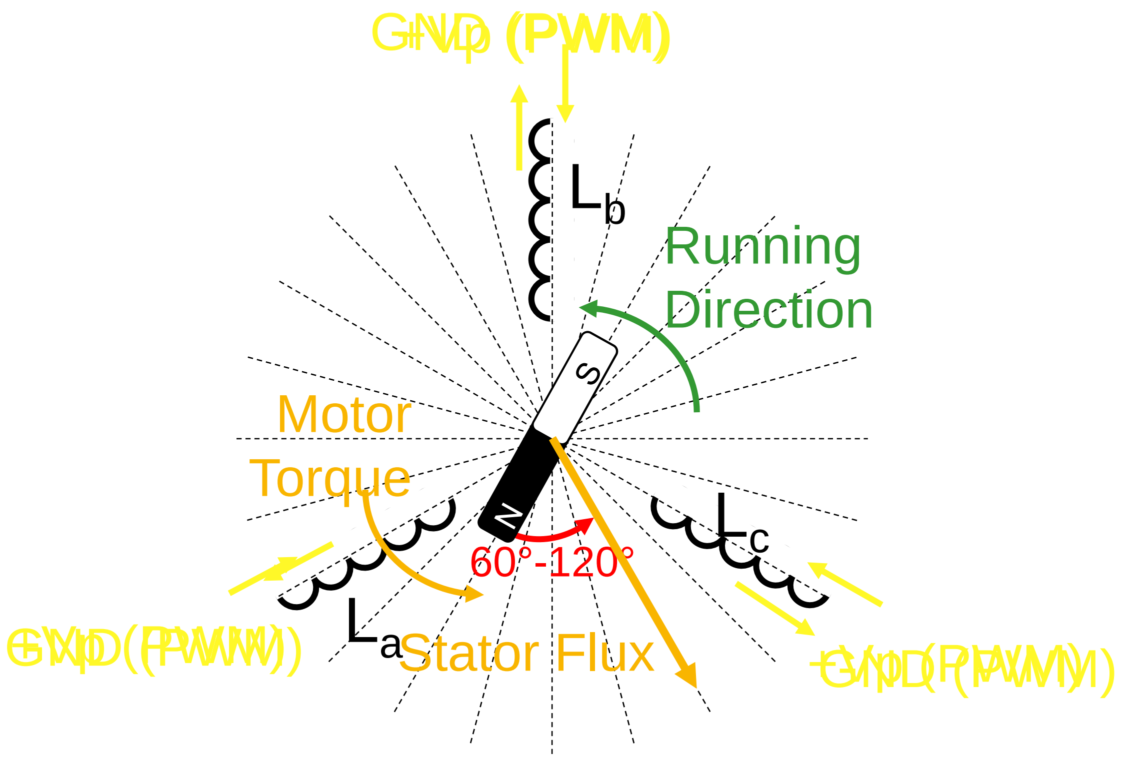


Trapezoidal Control: Brushless DC Motor

A BLDC motor consists of a rotor with permanent magnets and a stator with phase windings. A BLDC motor needs electronic commutation for the control of current through its three phase windings.



Trapezoidal Control: BLDC Commutation Method

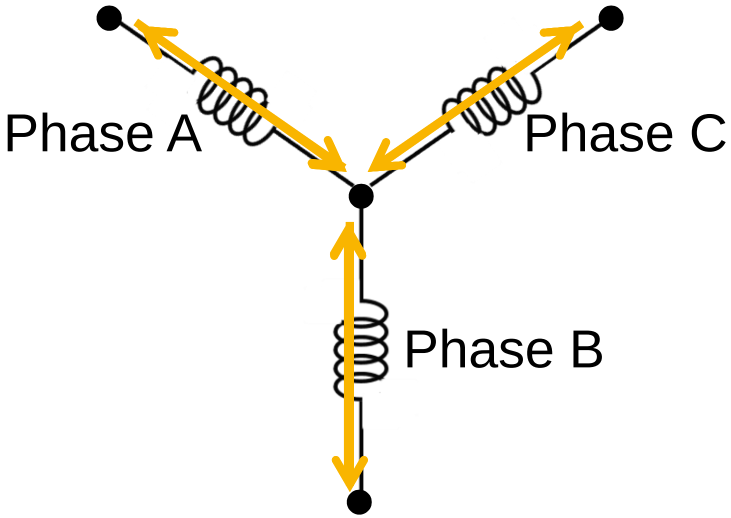


- Stator Field is generated between 60° to 120° to rotor field to get maximal torque and energy efficiency
- Six Flux Vectors defined to create rotation

Trapezoidal Control: Commutation Method

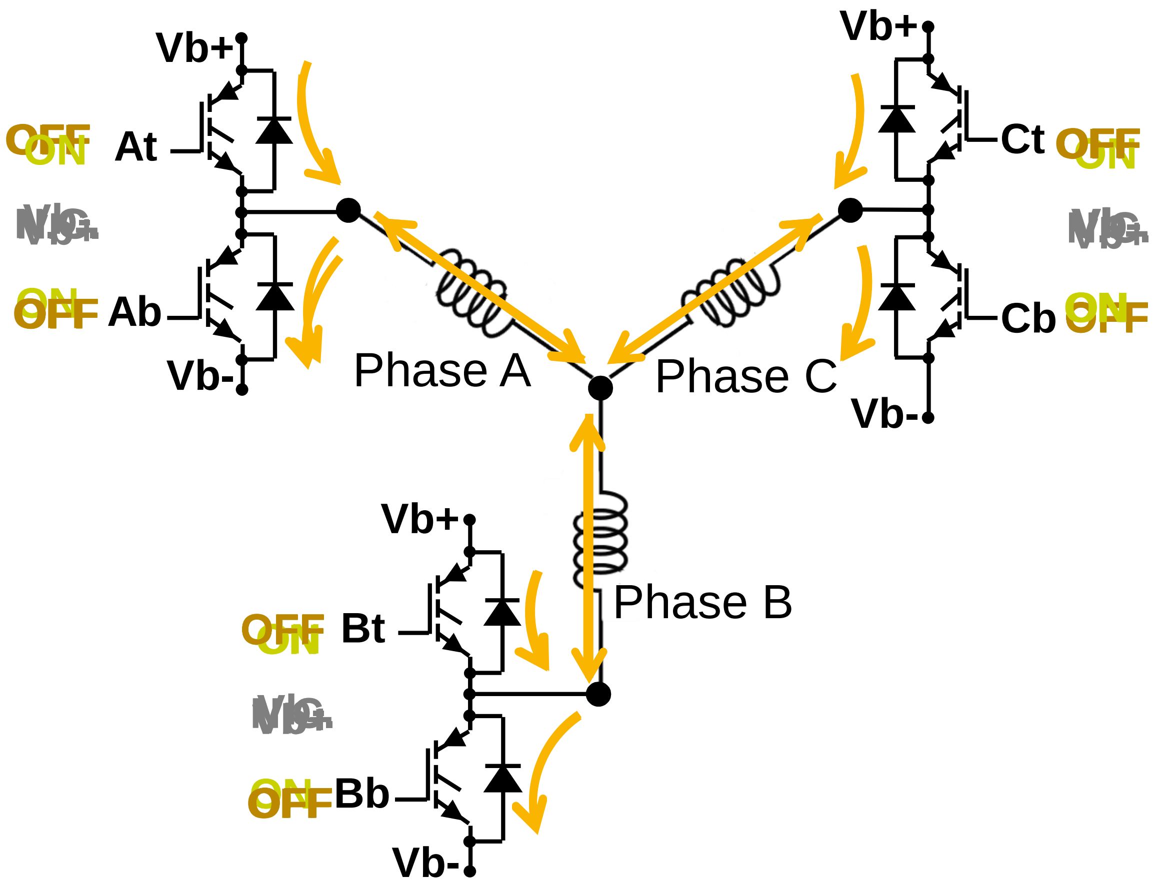
Trapezoidal control is one type of commutation method used to turn a motor where only two phase windings will conduct current at any one time. With direction also to consider, that leaves six possible patterns.

Circuit Representation
of BLDC Stator Windings



Trapezoidal Control: Commutation Control

By adding switches, the current flow can be controlled by a MCU to perform trapezoidal control.

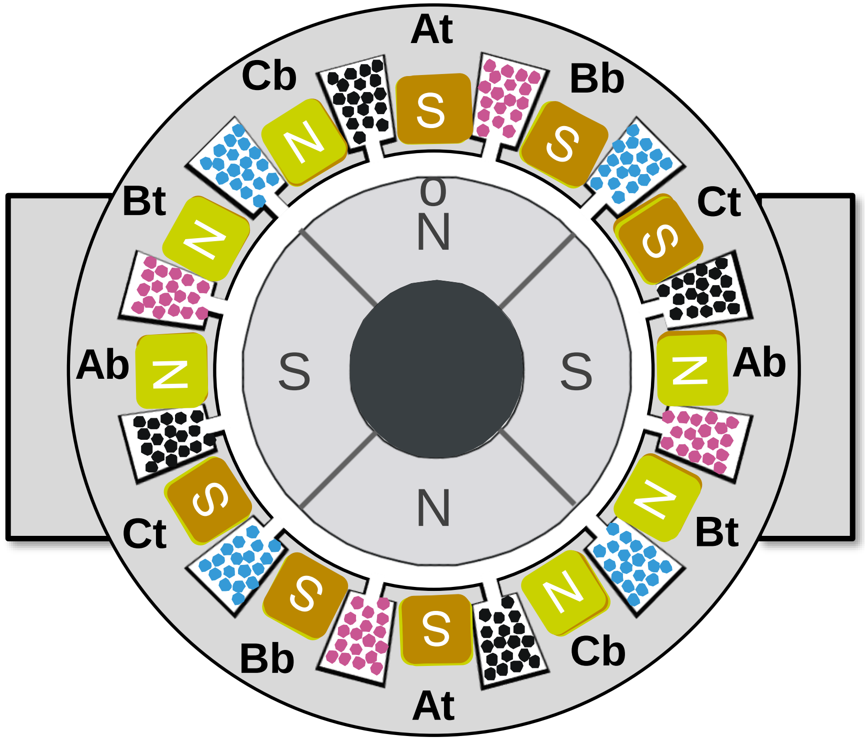


Trapezoidal Control: Turning the Motor CW

With the switches, the stator can be used to turn the motor to the desired direction and location by creating a magnetic field that affects the magnets on the rotor.

CW	Phase A	Phase B	Phase C
0/180°	Vb-	NC	Vb+
30°	Vb-	Vb+	NC
60°	NC	Vb+	Vb-
90°	Vb+	NC	Vb-
120°	Vb+	Vb-	NC
150°	NC	Vb-	Vb+

	Vb+	Vb-	NC
Top Switch	On	Off	Off
Bottom Switch	Off	On	Off

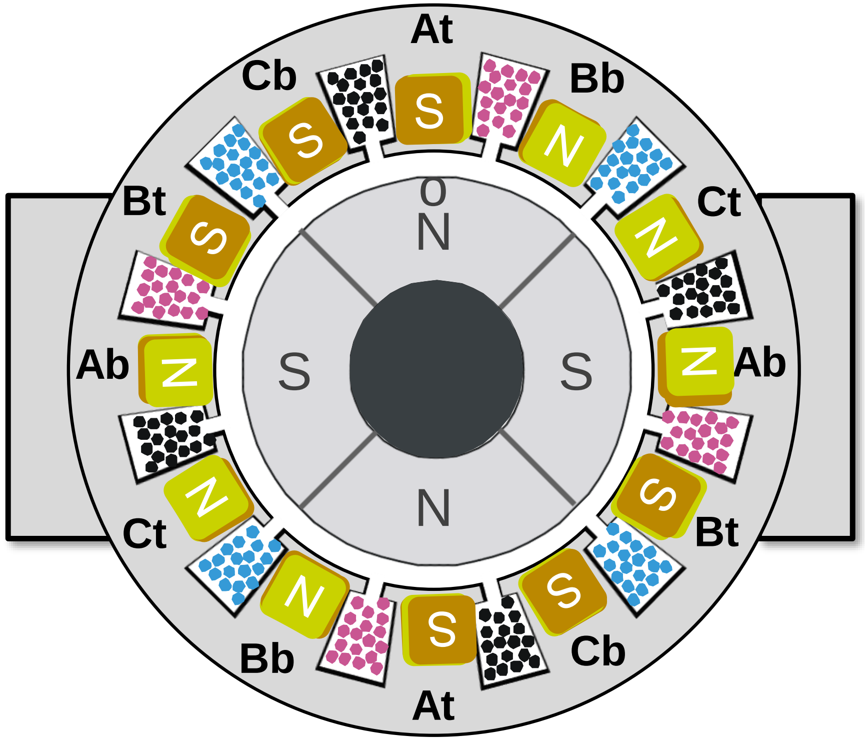


Trapezoidal Control: Turning the Motor CCW

With the switches, the stator can be used to turn the motor to the desired direction and location by creating a magnetic field that affects the magnets on the rotor.

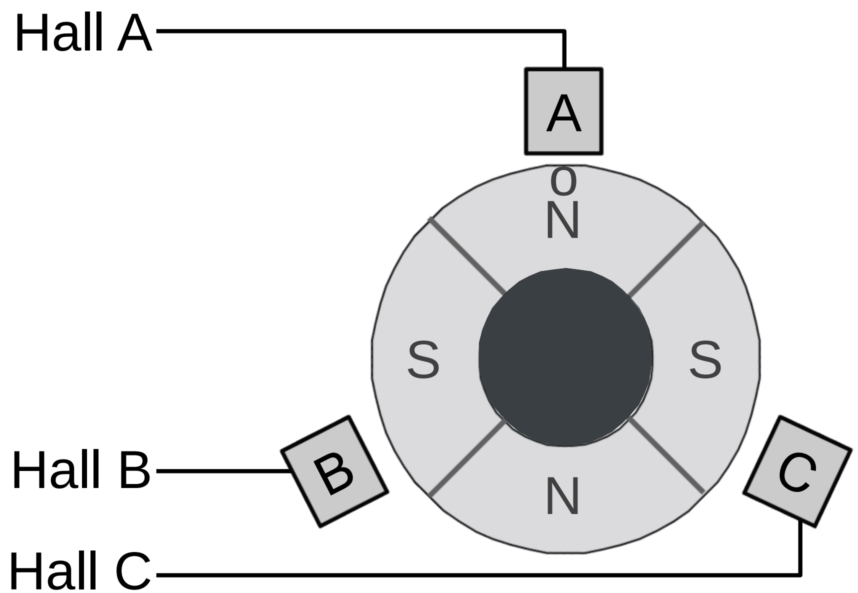
CCW	Phase A	Phase B	Phase C
0/180°	Vb+	NC	Vb-
30°	Vb+	Vb-	NC
60°	NC	Vb-	Vb+
90°	Vb-	NC	Vb+
120°	Vb-	Vb+	NC
150°	NC	Vb+	Vb-

	Vb+	Vb-	NC
Top Switch	On	Off	Off
Bottom Switch	Off	On	Off



Trapezoidal Control: Motor Position

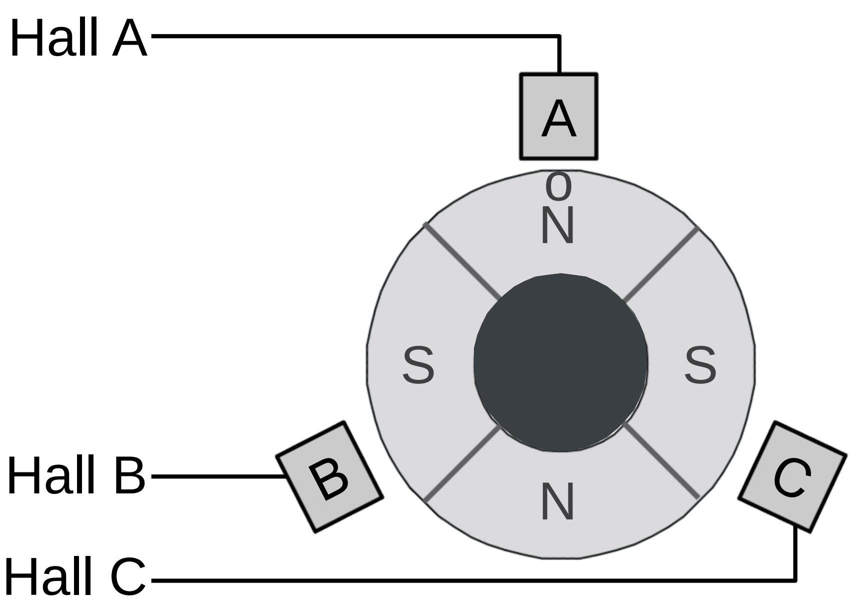
In order to commutate correctly for trapezoidal control, motor position information is required for proper motor rotation. The motor position information enables the MOSFETs or IGBTs in the inverter to properly be switched ON and OFF to ensure proper direction of current flow through the phase windings. Therefore, Hall sensors are used as position sensors for trapezoidal control. Each Hall sensor is placed 120 degrees apart and delivers a “high” state when facing a “north pole” and a “low” state when facing a “south pole”.



Trapezoidal Control: Motor Position CW

With three Hall sensors, it is possible to have eight states with two invalid states. That leaves six valid states that can be used to determine which two phase coils to drive the current through and in which direction. The six states are generated due to rotation of the motor.

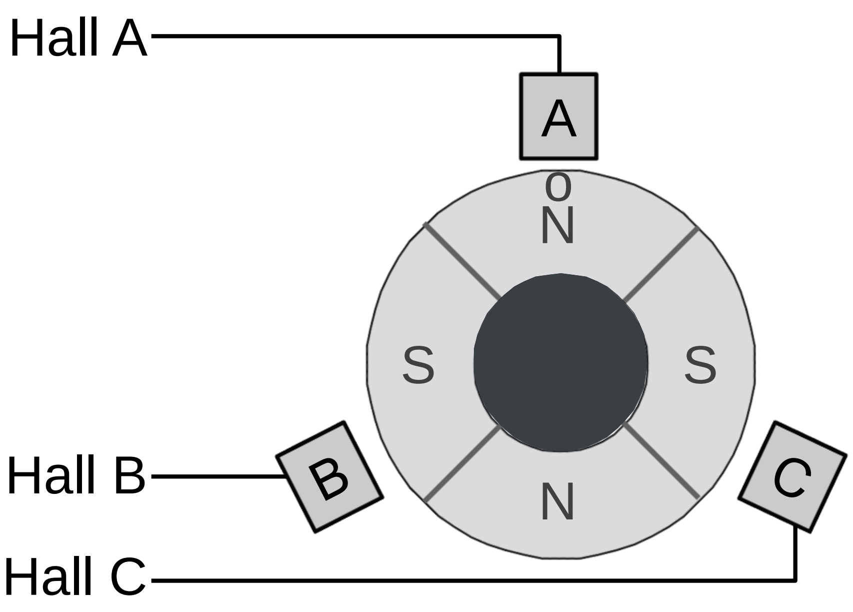
Hall A	Hall B	Hall C	State	CW
1	0	0	4	0/180°
1	1	0	6	30°
0	1	0	2	60°
0	1	1	3	90°
0	0	1	1	120°
1	0	1	5	150°
0	0	0	Invalid	n/a
1	1	1	Invalid	n/a



Trapezoidal Control: Motor Position CCW

With three Hall sensors, it is possible to have eight states with two invalid states. That leaves six valid states that can be used to determine which two phase coils to drive the current through and in which direction. The six states are generated due to rotation of the motor.

Hall A	Hall B	Hall C	State	CCW
1	0	0	4	0/180°
1	0	1	5	30°
0	0	1	1	60°
0	1	1	3	90°
0	1	0	2	120°
1	1	0	6	150°
0	0	0	Invalid	n/a
1	1	1	Invalid	n/a



Trapezoidal Control: Bringing It All Together

With the commutation table and the motor position table, a full trapezoidal control algorithm can be developed.

Motor Position Table Input

Hall A	Hall B	Hall C	State	CW
1	0	0	4	0/180°
1	1	0	6	30°
0	1	0	2	60°
0	1	1	3	90°
0	0	1	1	120°
1	0	1	5	150°
0	0	0	Invalid	n/a
1	1	1	Invalid	n/a

Commutation Table Output

CW	Phase A	Phase B	Phase C
0/180°	Vb+	NC	Vb-
30°	Vb+	Vb-	NC
60°	NC	Vb-	Vb+
90°	Vb-	NC	Vb+
120°	Vb-	Vb+	NC
150°	NC	Vb+	Vb-

	Vb+	Vb-	NC
Top Switch	On	Off	Off
Bottom Switch	Off	On	Off

Trapezoidal Control: Bringing It All Together

With the commutation table and the motor position table, a full trapezoidal control algorithm can be developed.

Trapezoidal Control Algorithm Clockwise Rotation

Hall A	Hall B	Hall C	State	CW	Phase A	Phase B	Phase C
1	0	0	4	0/180°	Vb+	NC	Vb-
1	1	0	6	30°	Vb+	Vb-	NC
0	1	0	2	60°	NC	Vb-	Vb+
0	1	1	3	90°	Vb-	NC	Vb+
0	0	1	1	120°	Vb-	Vb+	NC
1	0	1	5	150°	NC	Vb+	Vb-
0	0	0	Invalid	n/a			
1	1	1	Invalid	n/a			

	Vb+	Vb-	NC
Top Switch	On	Off	Off
Bottom Switch	Off	On	Off

Trapezoidal Control: Bringing It All Together

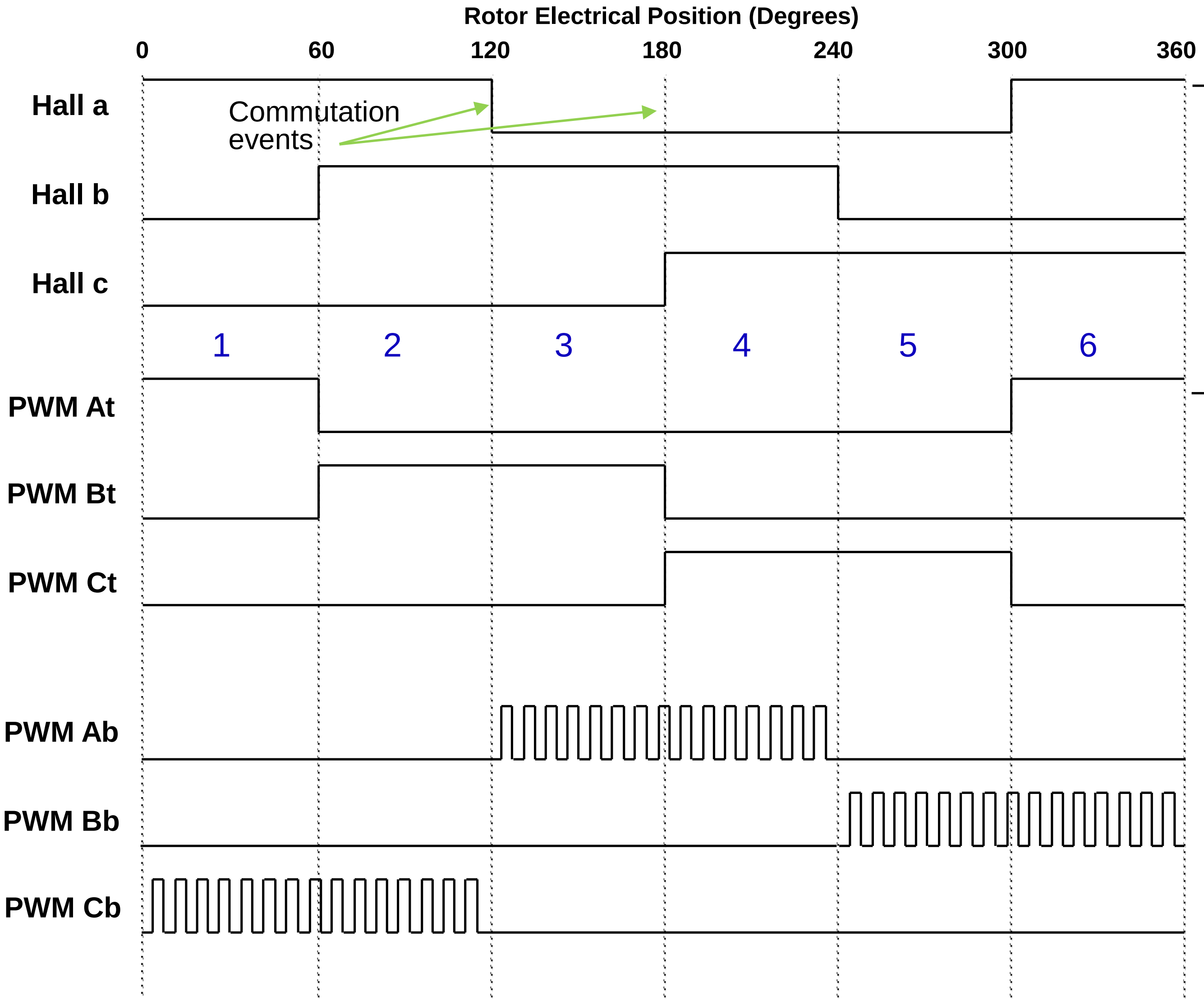
With the commutation table and the motor position table, a full trapezoidal control algorithm can be developed.

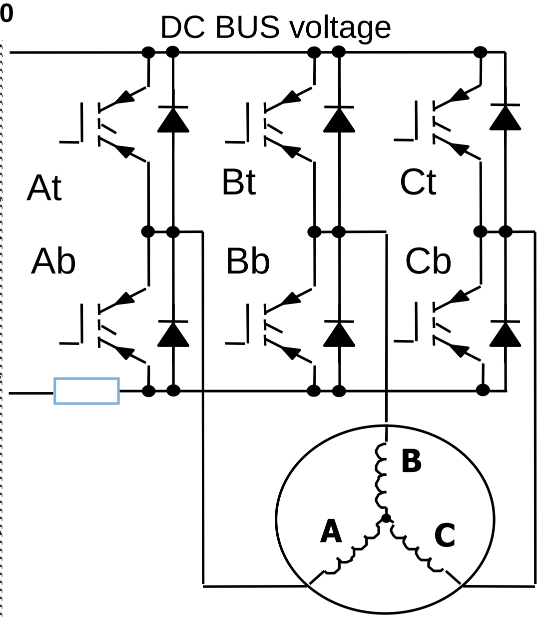
Trapezoidal Control Algorithm Counter Clockwise Rotation

Hall A	Hall B	Hall C	State	CW	Phase A	Phase B	Phase C
1	0	0	4	0/180°	Vb-	NC	Vb+
1	0	1	5	30°	Vb-	Vb+	NC
0	0	1	1	60°	NC	Vb+	Vb-
0	1	1	3	90°	Vb+	NC	Vb-
0	1	0	2	120°	Vb+	Vb-	NC
1	1	0	6	150°	NC	Vb-	Vb+
0	0	0	Invalid	n/a			
1	1	1	Invalid	n/a			

	Vb+	Vb-	NC
Top Switch	On	Off	Off
Bottom Switch	Off	On	Off

Sensor-based Commutation

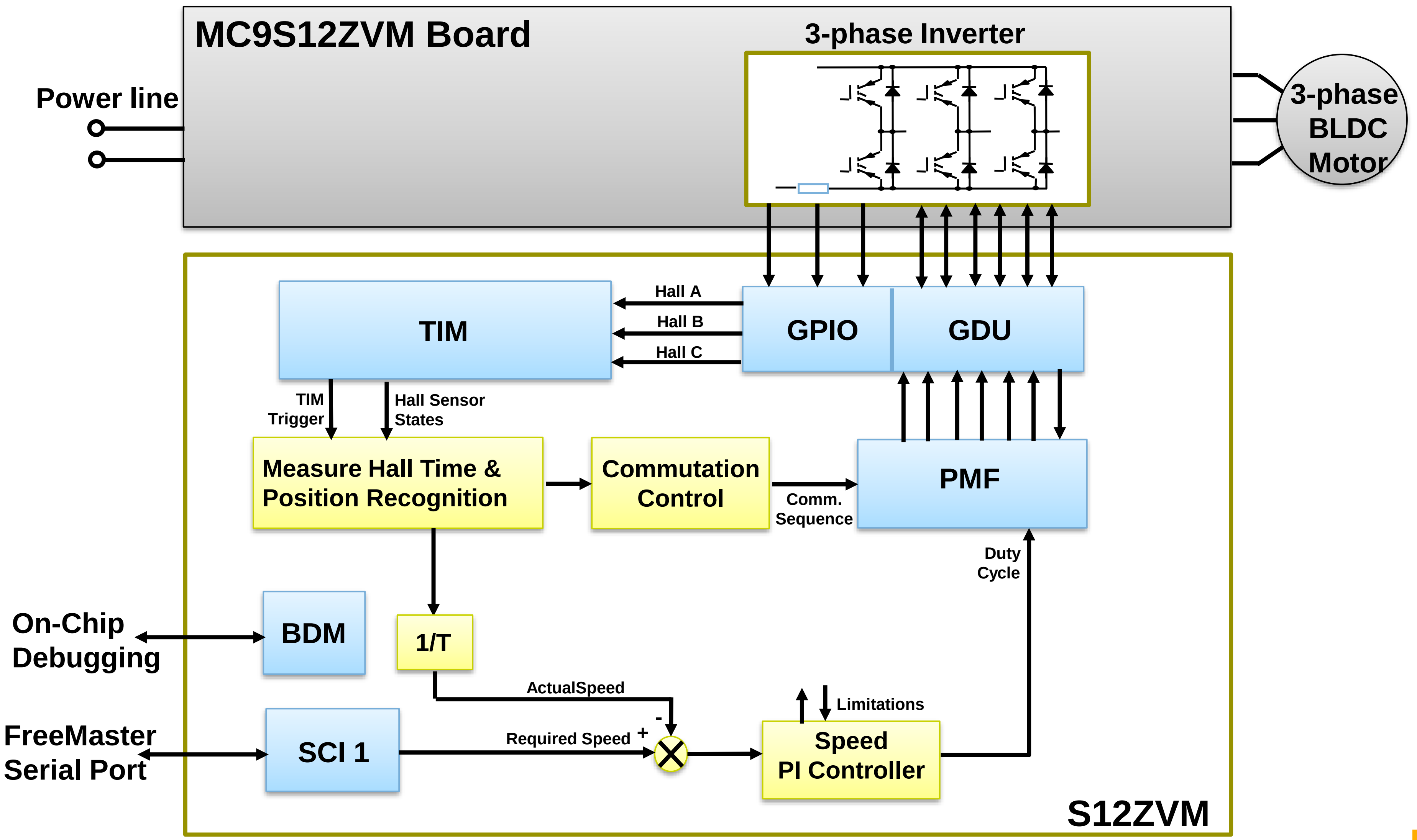




One phase powered by complementary PWM signal,
second phase grounded:

- Low MOSFET switching losses
- Low EMC noise

Trapezoidal Work-Shop Control Block diagram



Agenda

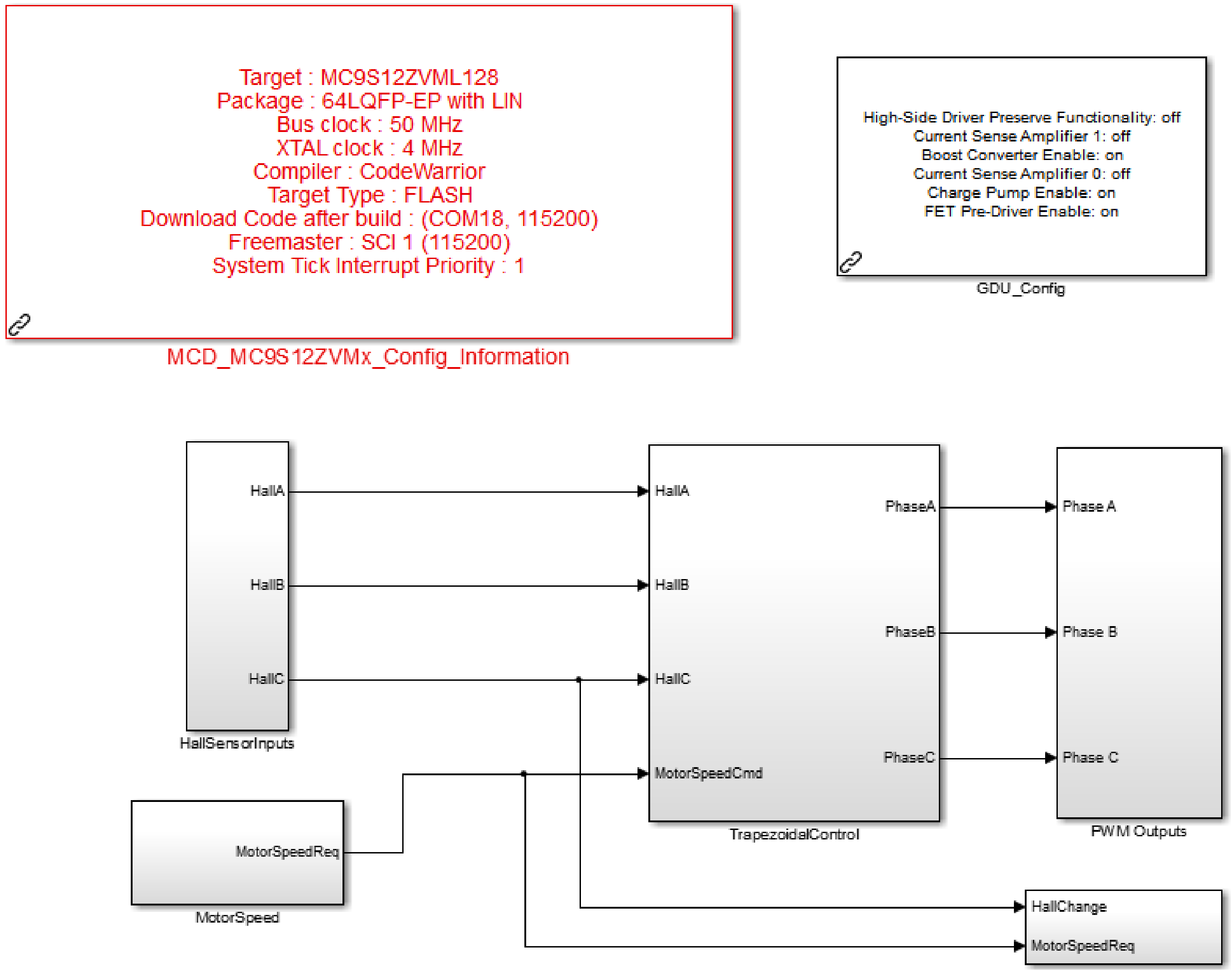
- **Overview:**
  - Introduction and Objectives
  - Model-Based Design Toolbox: Library blocks, FreeMASTER, and Bootloader
- **Hands-On Demo:**
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Convert simple model to run on Motor Kit with MCD Toolbox and use FreeMASTER
- **Model-Based Design:**
  - Model-Based Design Steps: Simulation, SIL, PIL and ISO 26262
  - SIL/PIL Hands-On Demo Step 2 & 3 of MBD
- **Trapezoidal Motor Control**:
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Trapezoidal control and how to use it to turn a motor
- **Trapezoidal Motor Control Hands-on Demo**:
  - Implement Trapezoidal Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor and tune parameters
- **FOC Motor Control**:
  - FOC Sensor-less control and how to use it to turn a motor
- **FOC Motor Control Hands-On Demo:**
  - Implement FOC Sensor-less Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor
- **Summary and Q&A:**

Hands-on Demo: Implement Trapezoidal Motor Control on Motor Kit

Summary Trapezoidal Motor Control on MPC5643L steps:

1. Open TrapCtrl.mdl
2. Save model as MPC564xL_TrapCtrl.mdl
3. Configure MPC5643L configuration block
4. Configure Input port blocks to read motor hall position state
5. Configure output blocks to monitor motor position with LEDs
6. Configure eTimer Blocks to detect change in motor position sensors
7. Configure eTimer Capture to measure Hall sensor pulse width for RPM calculation
8. Configure ADC block for monitoring potentiometer input for RPM Request
9. Configure Digital Input for use in controlling RPM Request
10. Configure DSPI blocks to interface to Freescale 3PP driver
11. Connect and configure Flex PWM blocks for output to switches
12. Configure PIT Timer and ADC blocks to read phase voltages.

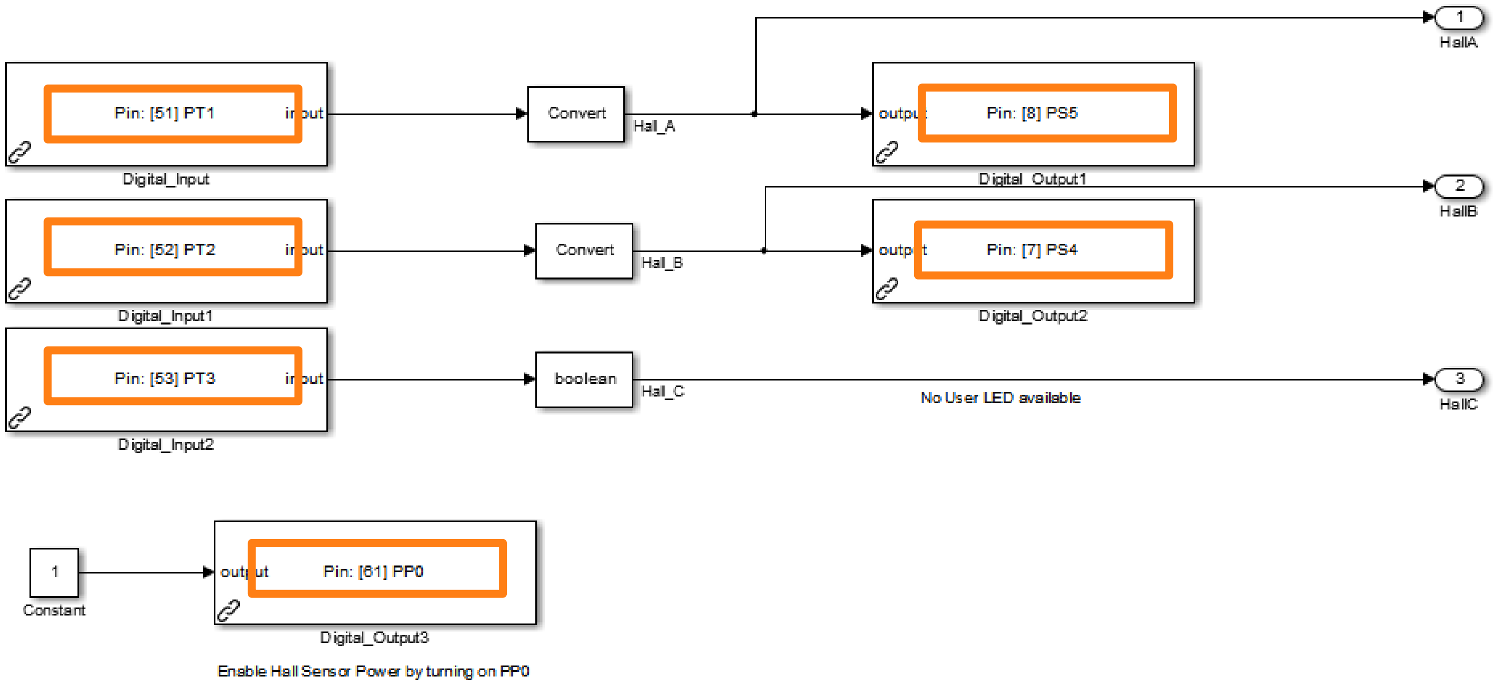
Hands-on Demo: Implement Trapezoidal Motor Control on Motor Kit



Hands-on Demo: Implement Trapezoidal Motor Control on Motor Kit

Configure Hall Sensor Input Block using Digital I/O steps:

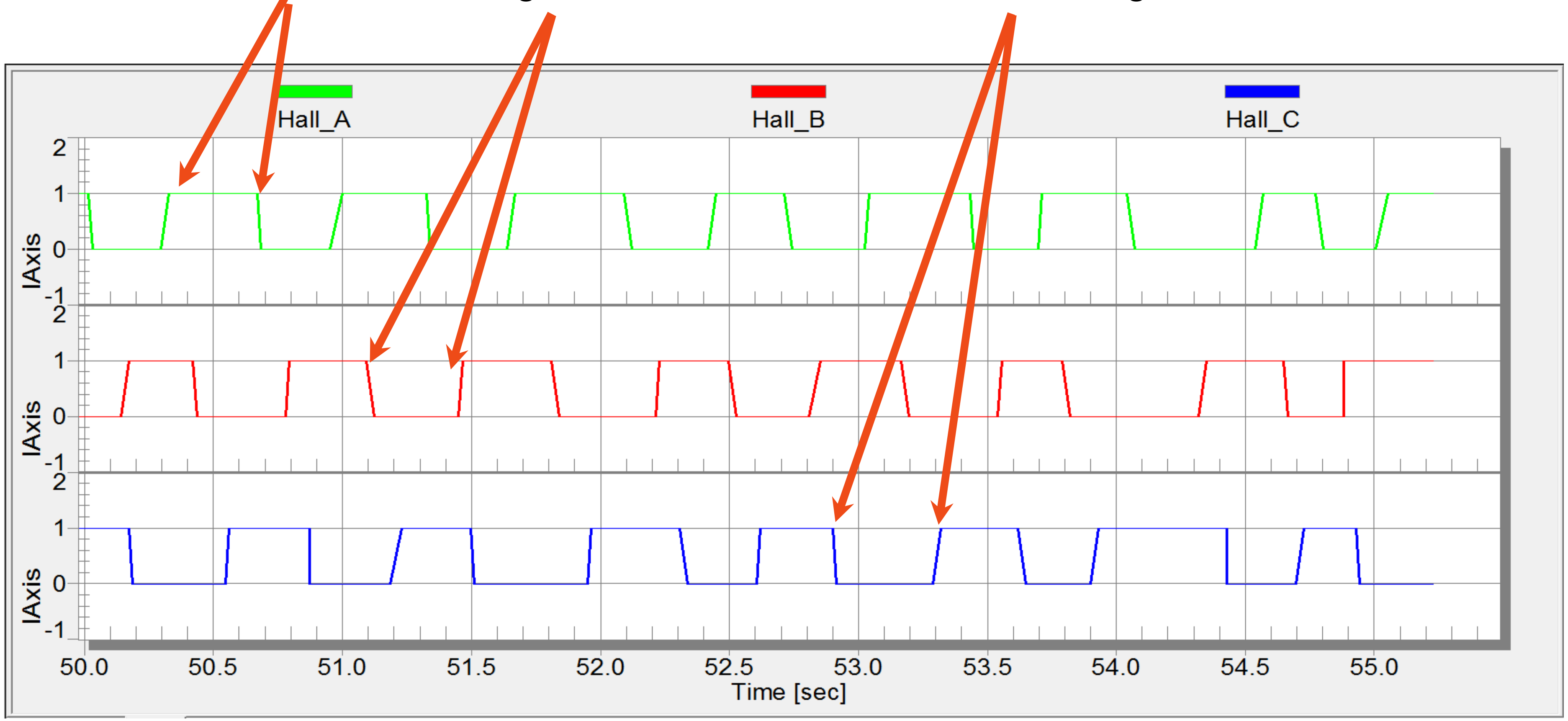
Remove termination blocks and pull 3 output blocks to replace them and then set them to the correct MCU pins. Set input blocks to correct pins.



Hands-on Demo: Implement Trapezoidal Motor Control on Motor Kit

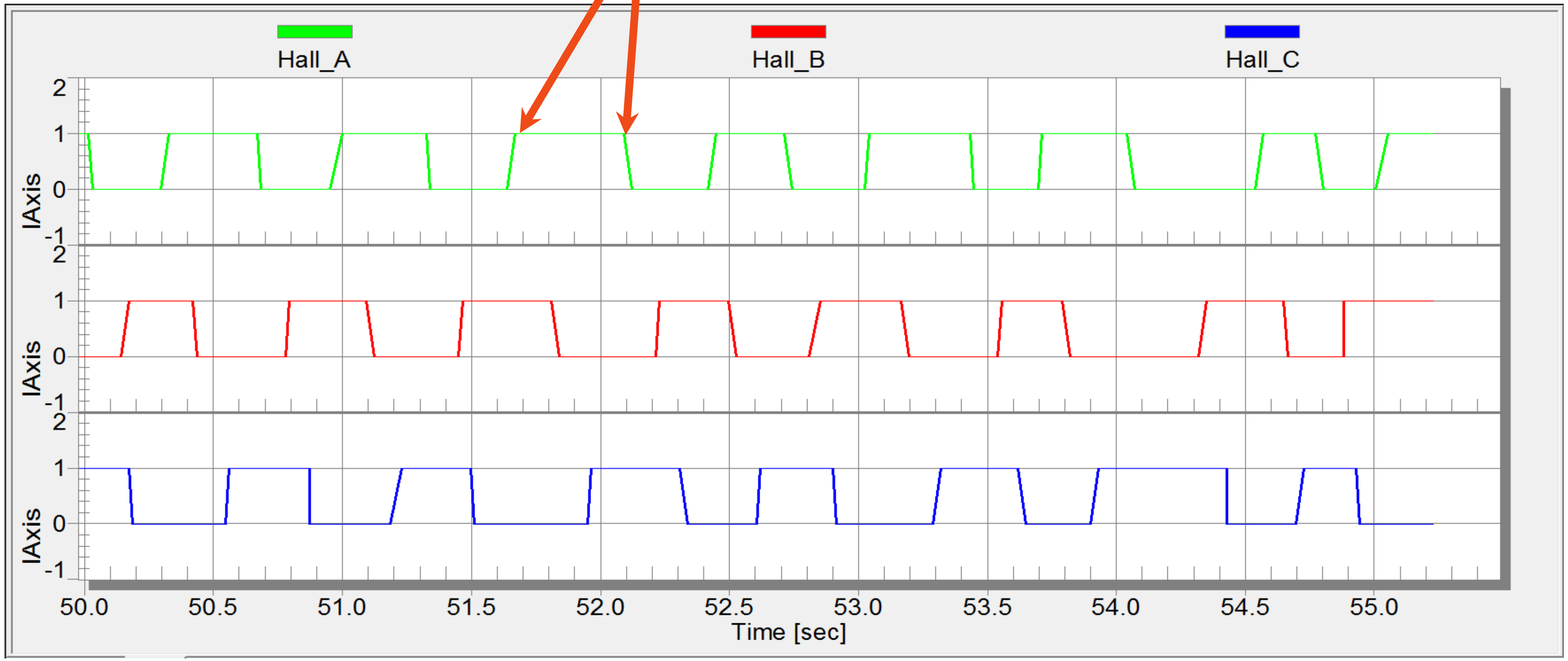
Commutation Change event

Commutation Change event



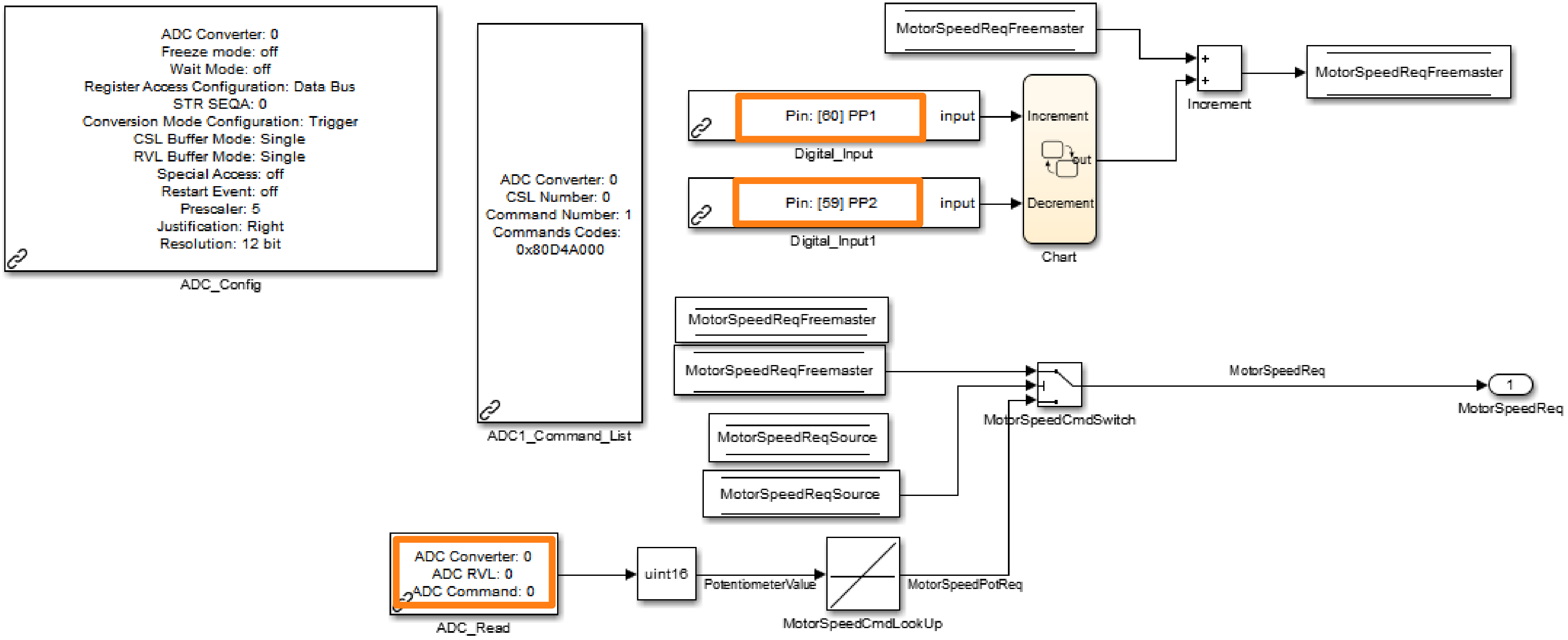
Hands-on Demo: Implement Trapezoidal Motor Control on Motor Kit

Measure the pulse width of a hall sensor so that motor speed can be calculated



Hands-on Demo: Implement Trapezoidal Motor Control on Motor Kit

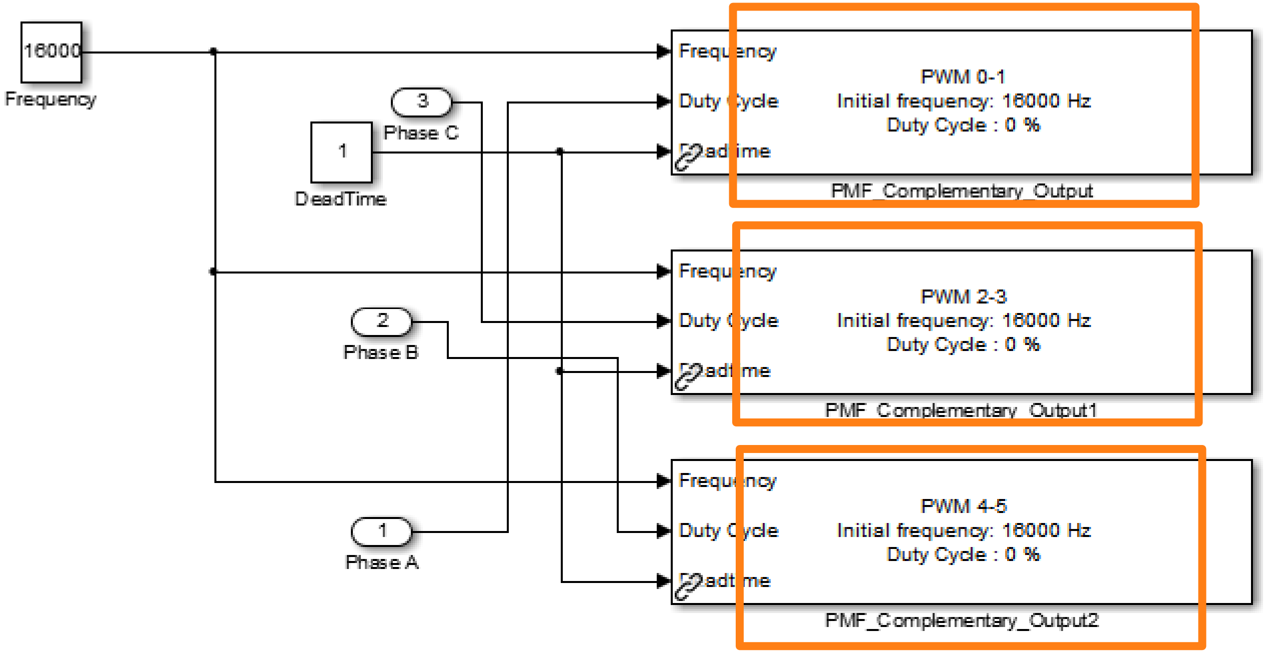
MotorSpeed Block with ADC and Digital outputs steps:



Hands-on Demo: Implement Trapezoidal Motor Control on Motor Kit

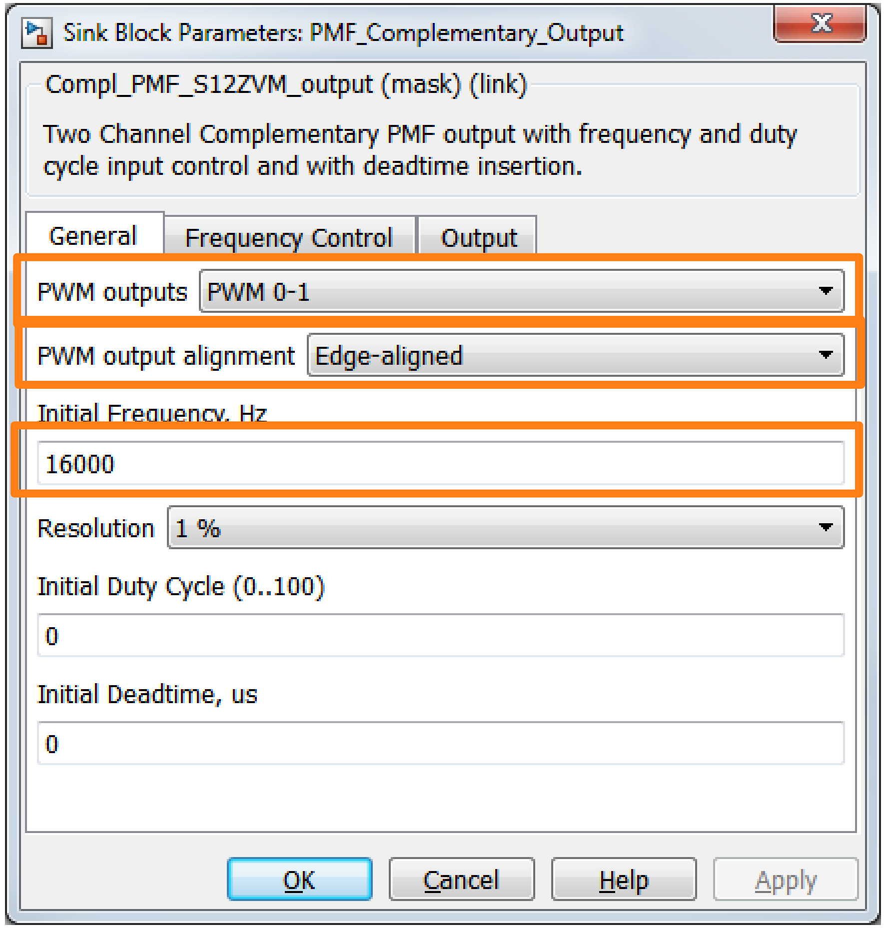
3PhaseDutyCycleOut Block with Flex PWM Blocks steps:

Pull Simple PWM phase block from library, connect to phase A and configure.



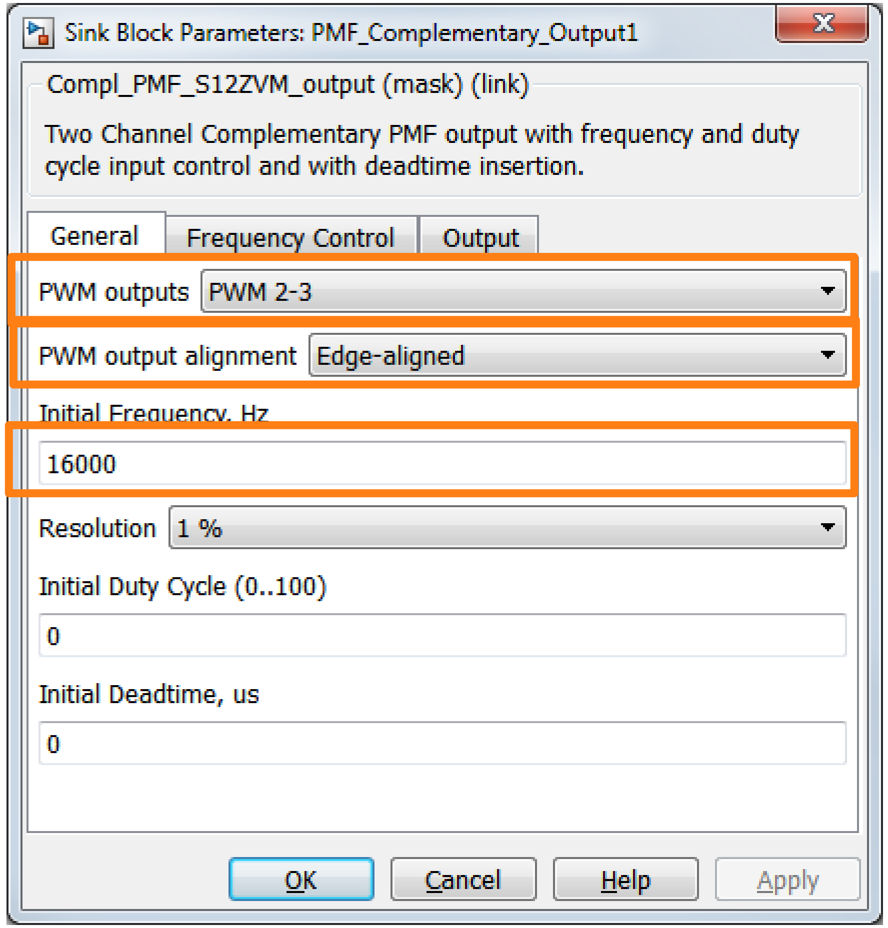
Hands-on Demo: Implement Trapezoidal Motor Control on Motor Kit

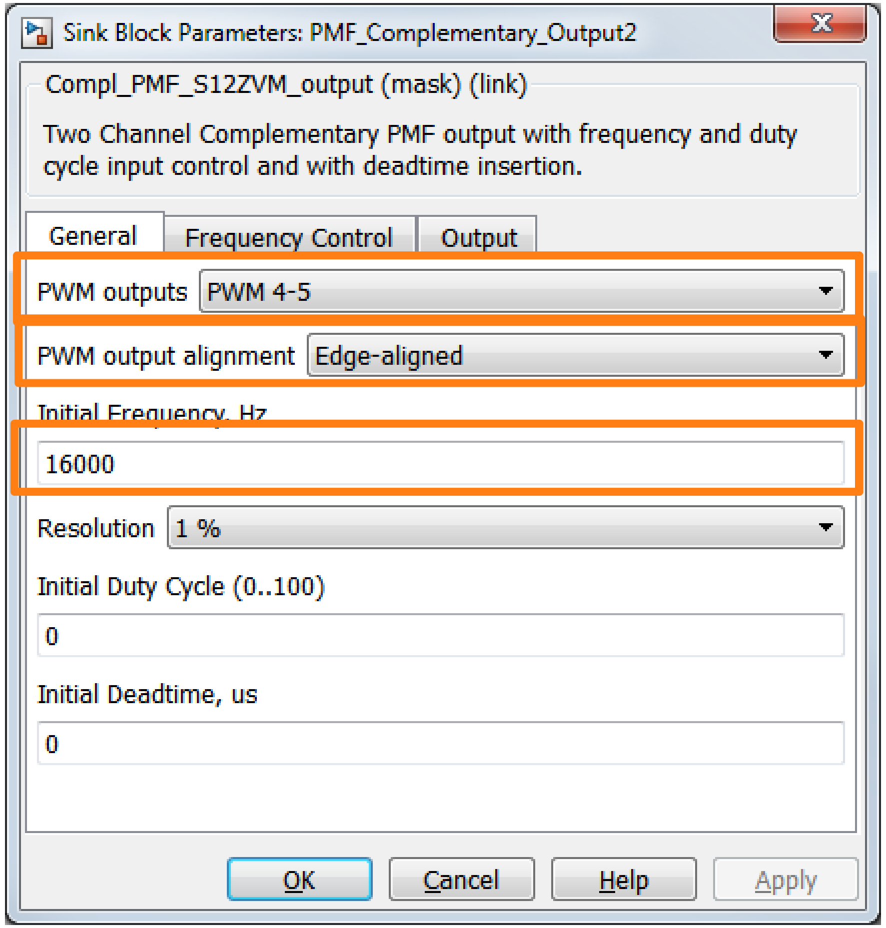
PMF Block steps:



Hands-on Demo: Implement Trapezoidal Motor Control on Motor Kit

PMF Block steps:

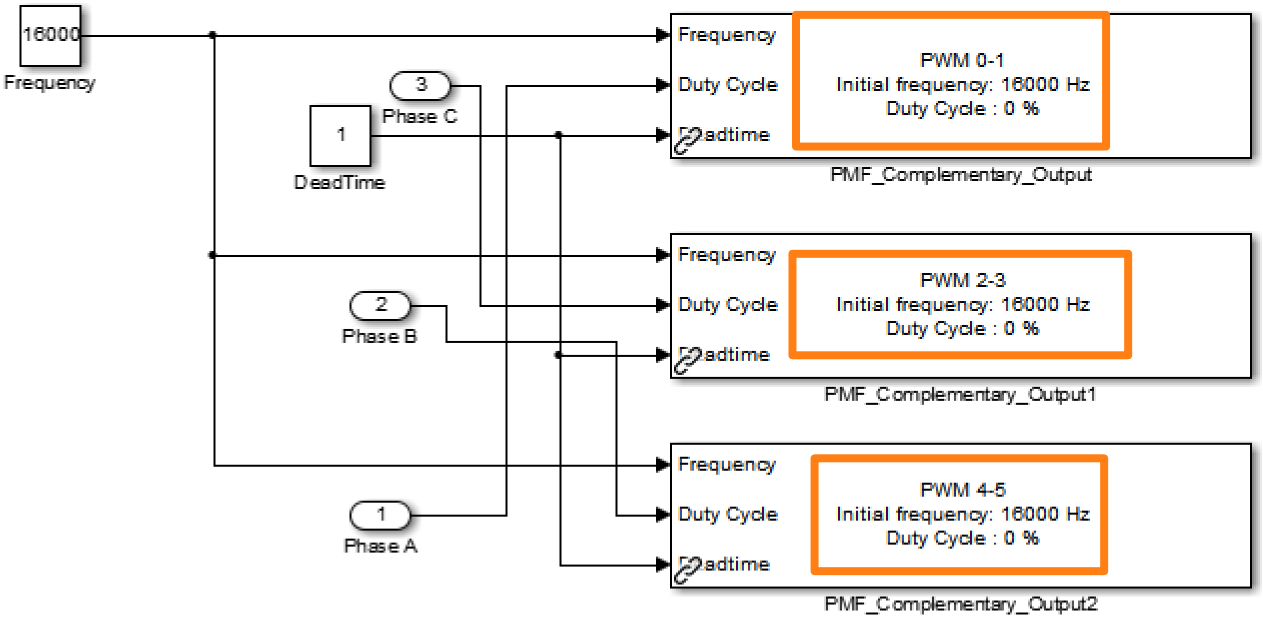




Hands-on Demo: Implement Trapezoidal Motor Control on Motor Kit

3PhaseDutyCycleOut Block with Flex PWM Blocks steps:

Copy phase C block, paste twice and connect to phase A & B.



Hands-on Demo: FreeMASTER to Monitor and Tune Parameters

Using FreeMASTER with Hands-on Demo

1. Start FreeMASTER and open project S12ZVM_TrapCtrl.pmp. Press OK if a message comes up that the map file has been updated.
2. Go to Project Options pull-down and select "Options". Verify that COM settings are the same as what were set in your model.
3. Once the COM settings are correct, press the STOP button.
4. Change MotorSpeedReqFreemaster Variable to 1000 RPM.

Agenda

- **Overview:**
  - Introduction and Objectives
  - Model-Based Design Toolbox: Library blocks, FreeMASTER, and Bootloader
- **Hands-On Demo:**
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Convert simple model to run on Motor Kit with MCD Toolbox and use FreeMASTER
- **Model-Based Design:**
  - Model-Based Design Steps: Simulation, SIL, PIL and ISO 26262
  - SIL/PIL Hands-On Demo Step 2 & 3 of MBD
- **Trapezoidal Motor Control**:
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Trapezoidal control and how to use it to turn a motor
- **Trapezoidal Motor Control Hands-on Demo**:
  - Implement Trapezoidal Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor and tune parameters
- **FOC Motor Control**:
  - FOC Sensor-less control and how to use it to turn a motor
- **FOC Motor Control Hands-On Demo:**
  - Implement FOC Sensor-less Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor
- **Summary and Q&A:**

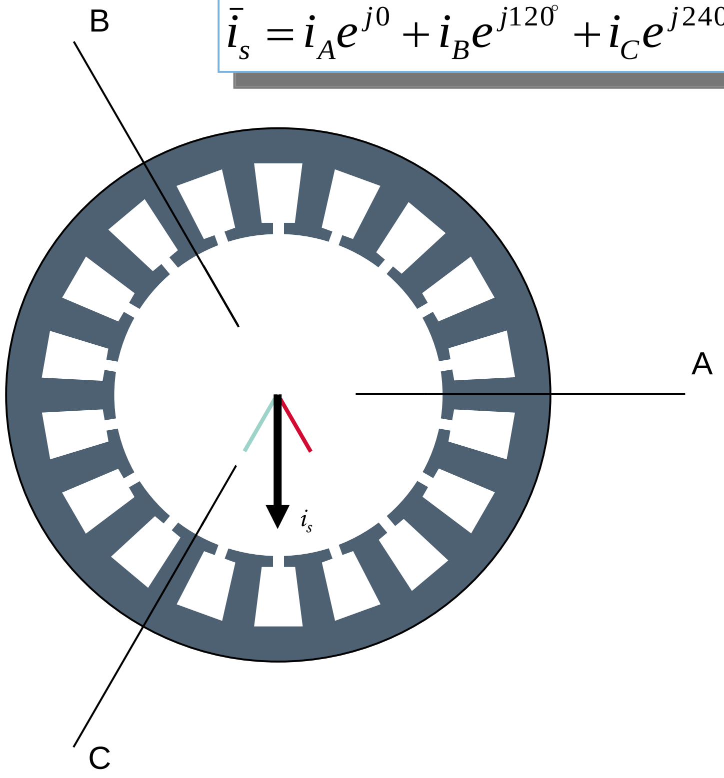
Motor Control: Why FOC over Trapezoidal

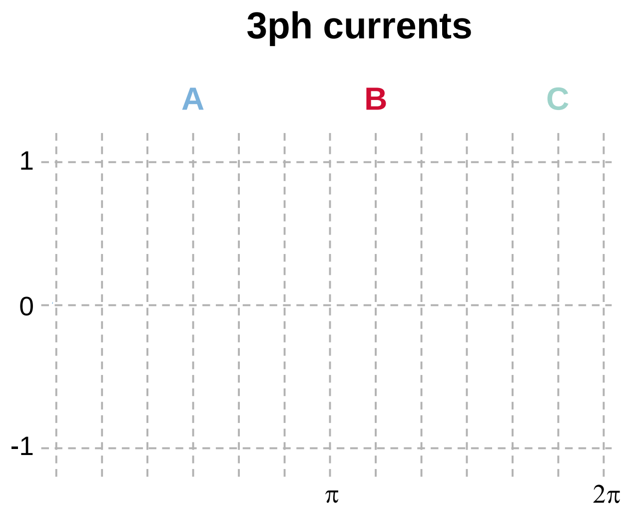
- FOC inherently better at aligning rotor and stator flux which results in a more efficient way of generating motor torque.
- Since FOC continuously pulls the rotor to a new position torque ripple is reduced making it ideal for application like electric steering where low speeds are required.
- FOC uses sinusoidal commutation therefore reducing EMC noise that trapezoidal control can create.
- Finally, FOC can enable a motor to go above its rated speed at the expense of torque. This is called Field Weakening where the stator windings are energized at an angle where the rotor's magnetic field weaker therefore increasing the magnetic field vector.

Motor Control: Creation of Rotating Magnetic Field

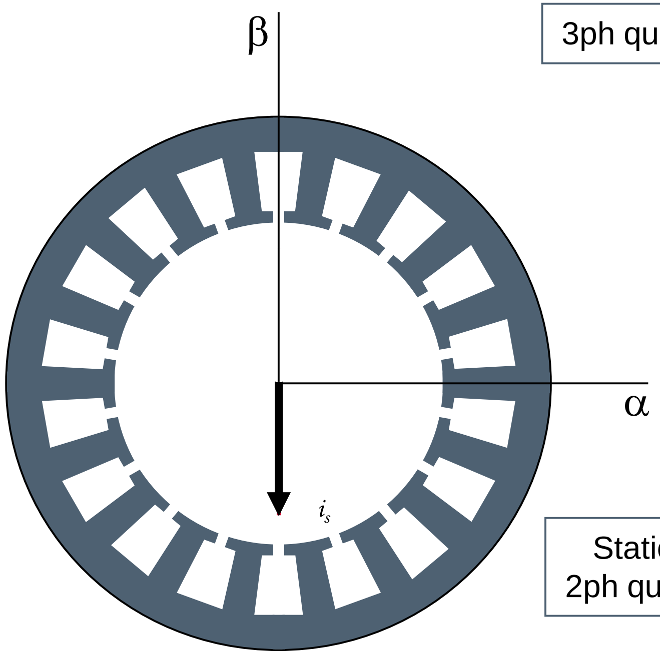
- The space-vectors can be defined for all motor quantities

$$\bar{i}_s = i_A e^{j0} + i_B e^{j120^\circ} + i_C e^{j240^\circ}$$





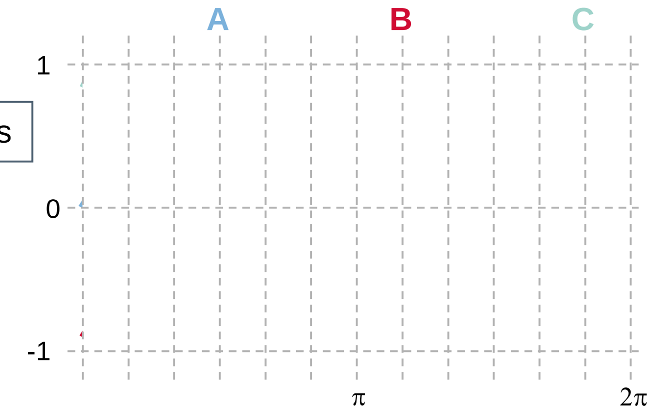
Motor Control: Transformation to 2-ph Stationary Frame

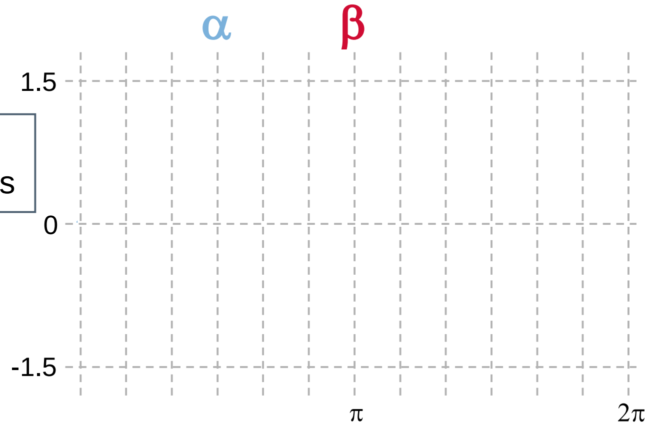


3ph quantities

Stationary 2ph quantities

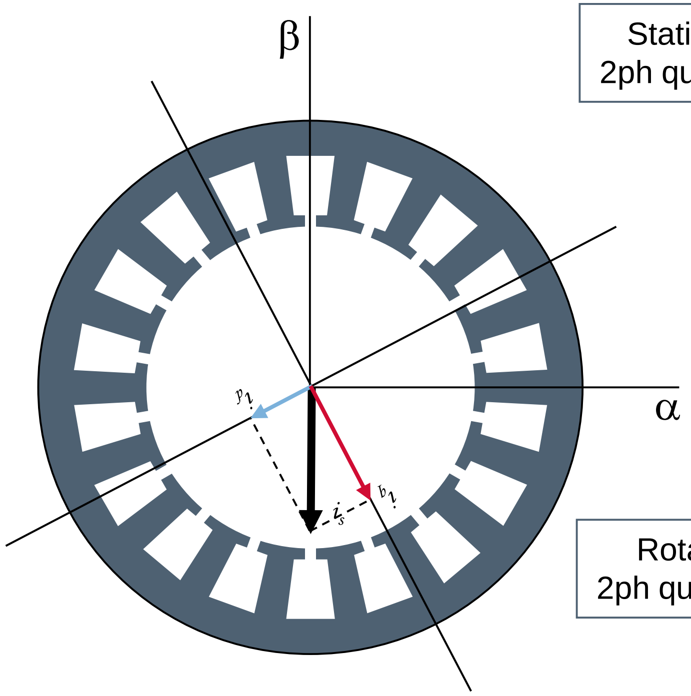
3ph currents / MMF



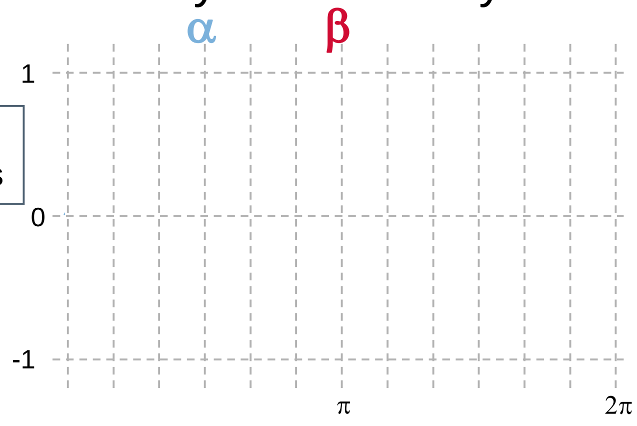


Motor Control: Transformation to 2-ph Synchronous Frame

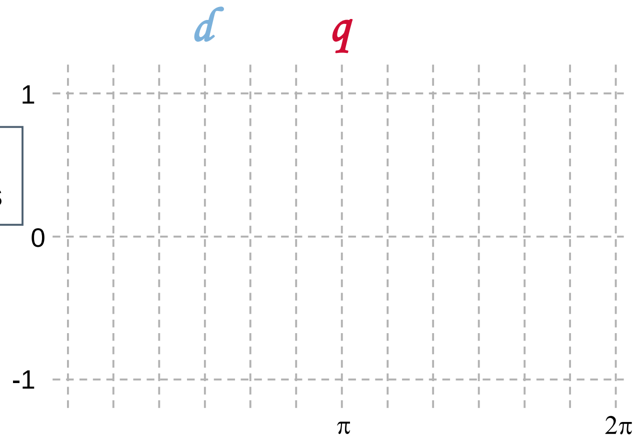
- Position and amplitude of the stator flux/current vector is fully controlled by two DC values

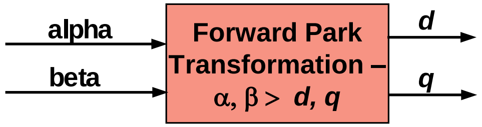


Stationary
2ph quantities

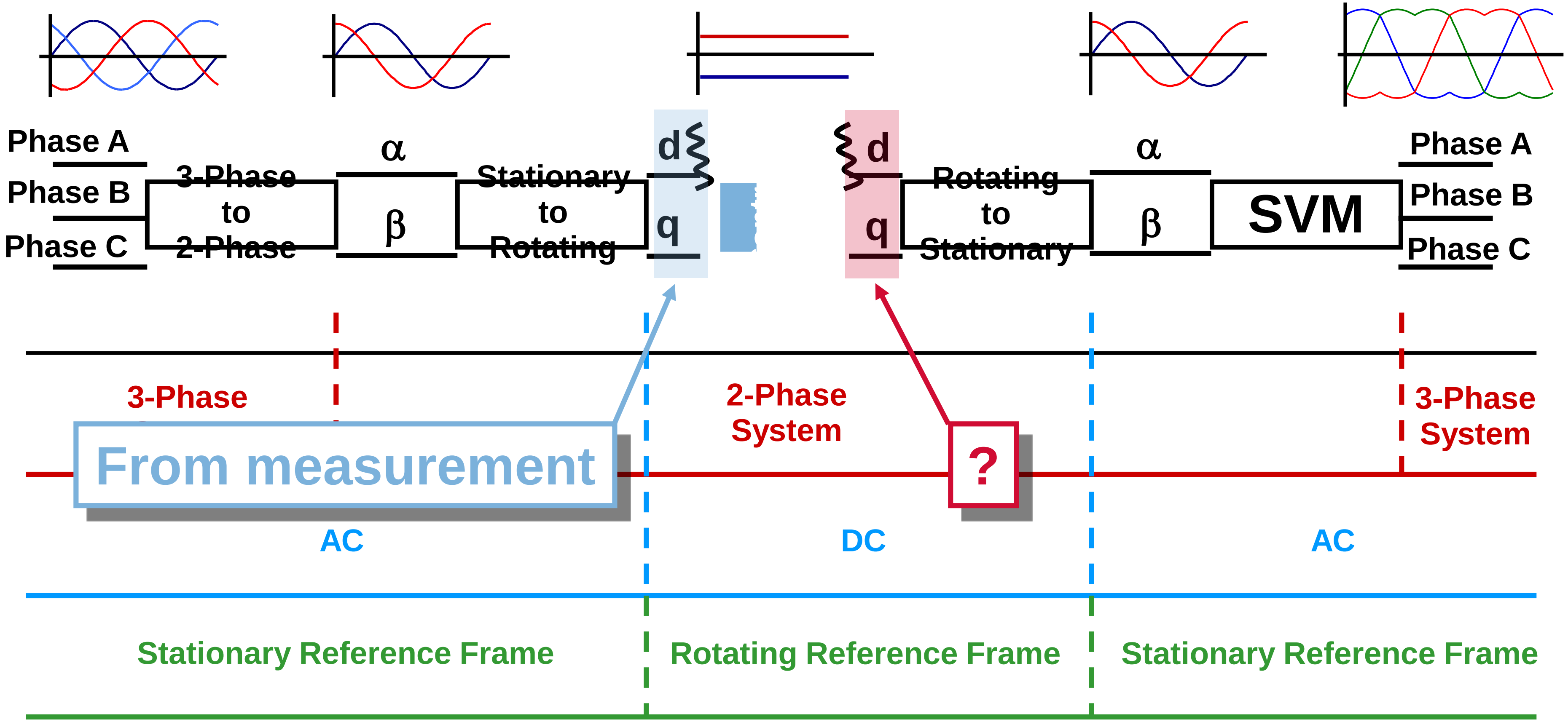


Rotating
2ph quantities

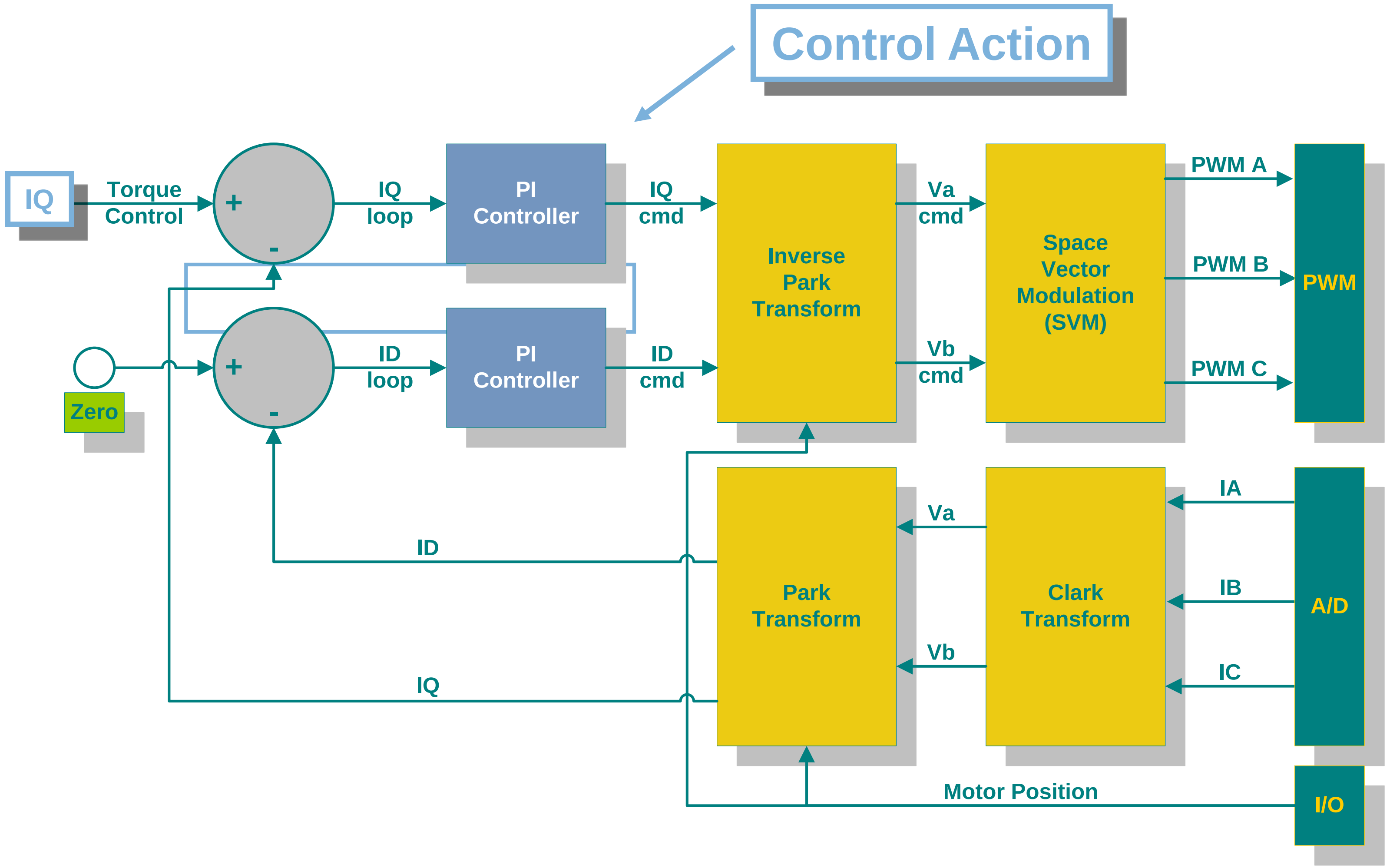




Motor Control: FOC Transformation Summary



Motor Control: FOC Transformation Summary



Motor Control: Field Oriented Control in Steps

1. Measure obtain state variables quantities
   (e.g. phase currents, voltages, rotor position, rotor speed …).

2. Transform quantities from 3-phase system to 2-phase system (Forward Clark Transform) to simplify the math - lower number of equations

3. Transform quantities from stationary to rotating reference frame -
   “rectify” AC quantities, thus in fact transform the AC machine to DC machine

4. Calculate control action (when math is simplified and machine is “DC”)

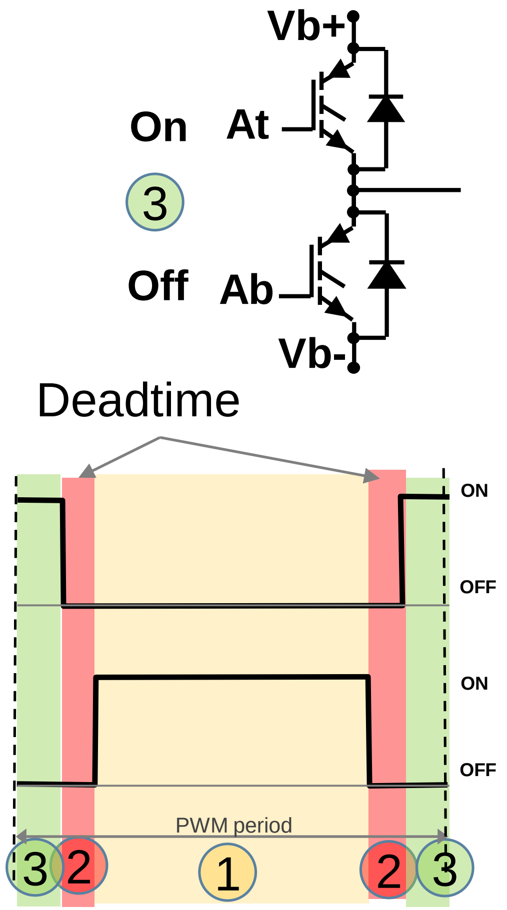
5. Transform the control action (from rotating) to stationary reference frame

6. Transform the control action (from 2-phase) to 3-phase system

7. Apply 3-phase control action to el. motor

Motor Control: Commutation Control Methods

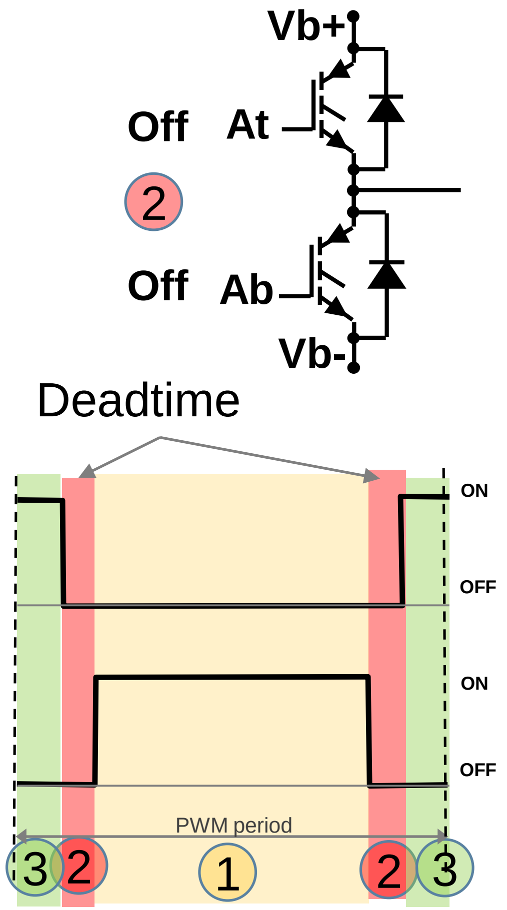
- All three inverter legs (6 transistors) are managed at any time – transistors are either switched on or off
- PWM pairs are set to complementary mode
  - Top transistor – ON
  - Bottom transistor – OFF
  - Or vice versa
  - Deadtime is inserted to protect inverter against short circuit



Motor Control: Commutation Control Methods

- All three inverter legs (6 transistors) are managed at any time – transistors are either switched on or off

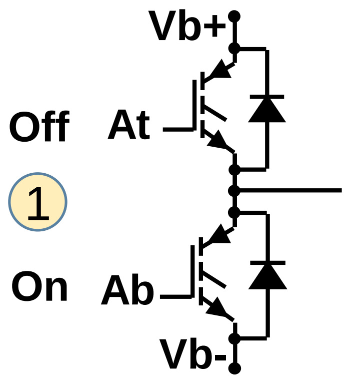
- PWM pairs are set to complementary mode
  - Top transistor – ON
  - Bottom transistor – OFF
  - Or vice versa
  - Deadtime is inserted to protect inverter against short circuit

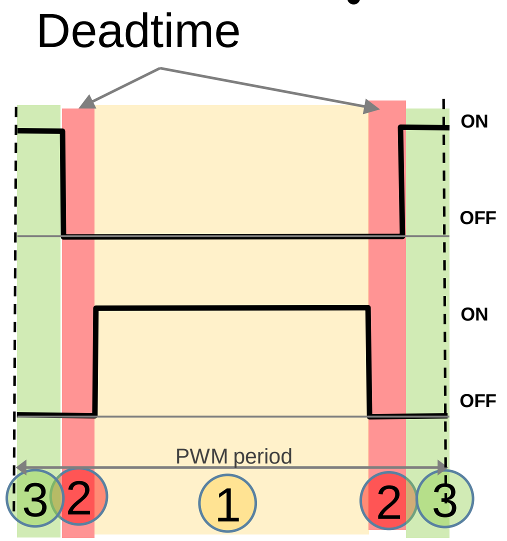


Motor Control: Commutation Control Methods

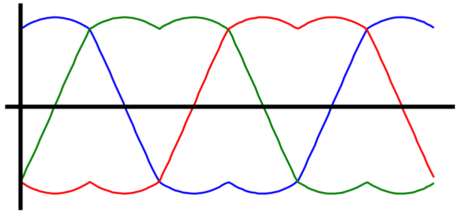
- All three inverter legs (6 transistors) are managed at any time – transistors are either switched on or off

- PWM pairs are set to complementary mode
    - Top transistor – ON
    - Bottom transistor – OFF
    - Or vice versa
    - Deadtime is inserted to protect inverter against short circuit



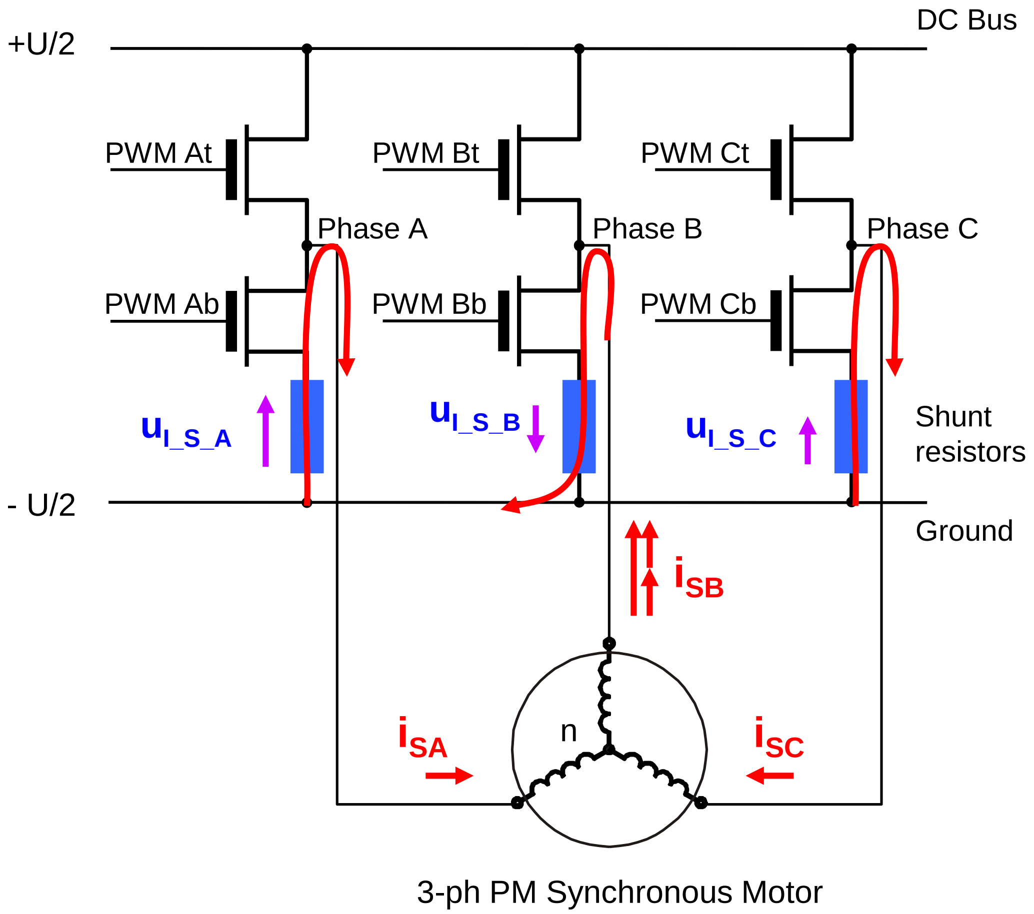


- All PMSM phases are always supplied creating sinusoidal voltages.



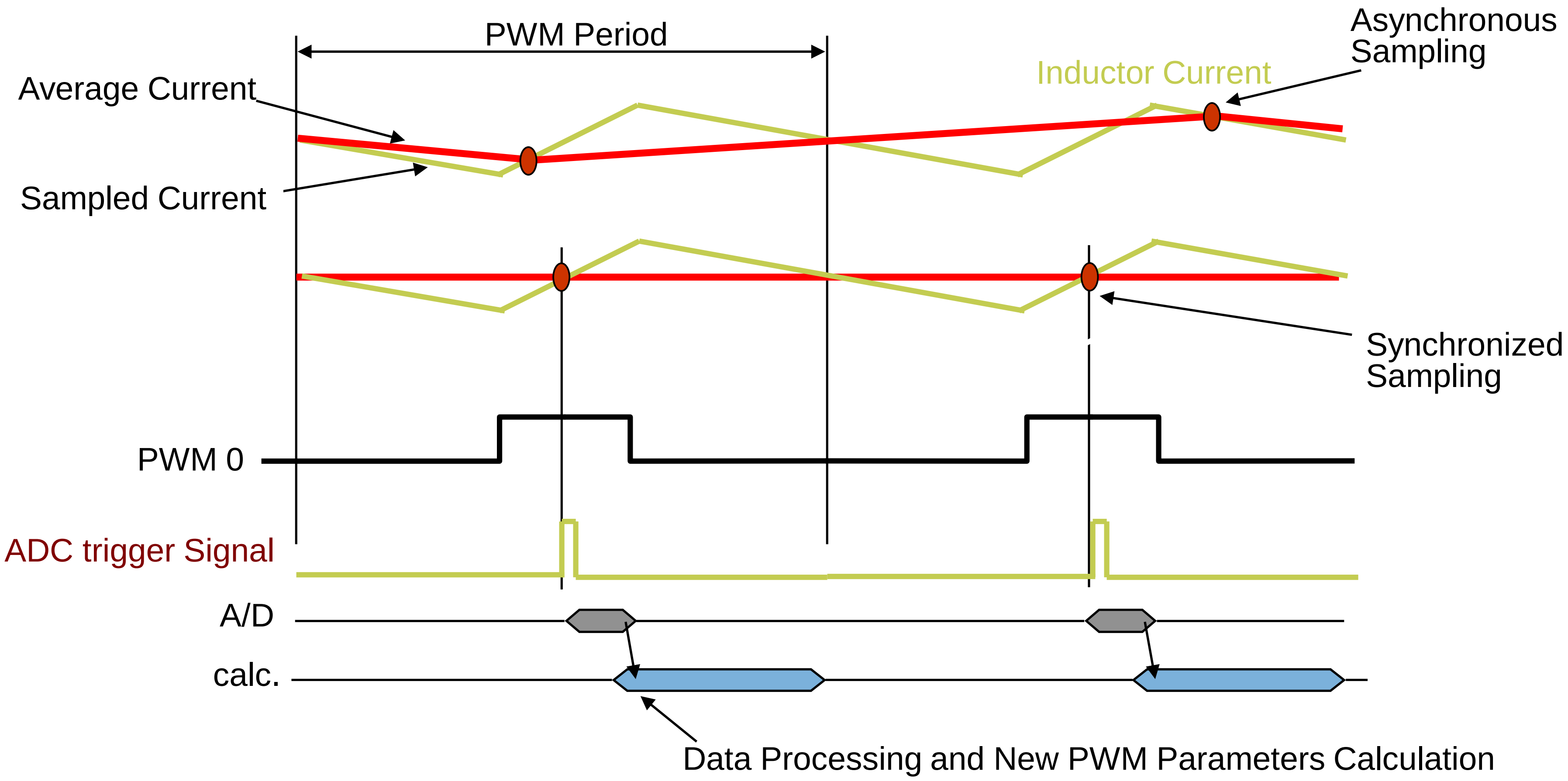
Motor Control: Current Sensing with Shunt Resistors

- Shunt resistors voltage drop measured
- SW calculation of all 3 phase currents needed
- Adding all 3 phase currents equals zero allowing that only two phase currents needed to be sampled by the MCU.
- Dual-sampling required



ADC to PWM Synchronization - Why Needed?

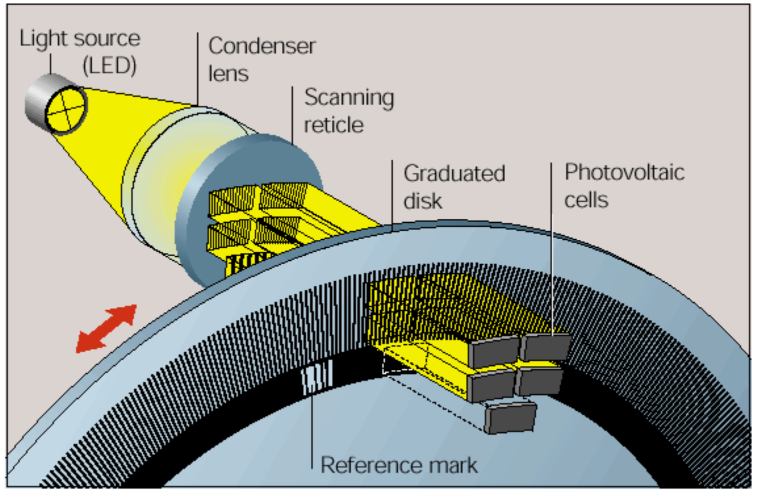
- ADC sampling helps to filter the measured current - antialiasing



Motor Control: Incremental Encoders

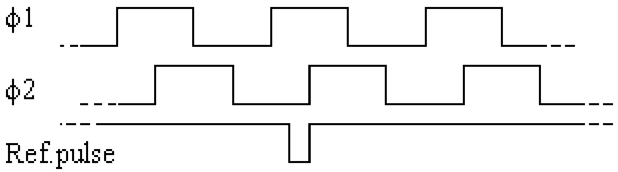
- The current position is calculated by incrementing/decrementing the pulse edges.
- The direction of counting is determined by phase shift of two quadrature pulses.
- The reference pulse is used to denote start point.
- Rotary encoders output the position values over the binary TTL square-waves or serial data interfaces (EnDat, SSI, PROFIBUS-DP)

Scanning Principle



Source: Heidenhain

Incremental Encoder Pulses

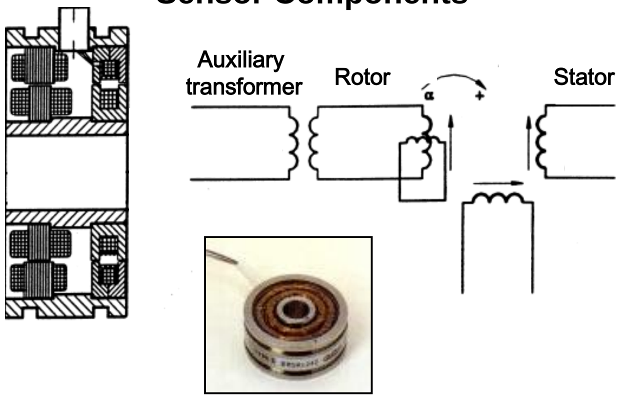


There are 4 phases within one pulse cycle. You need for example (360/0.5)/4=180 pulses per rotation if 0.5deg resolution is wanted.

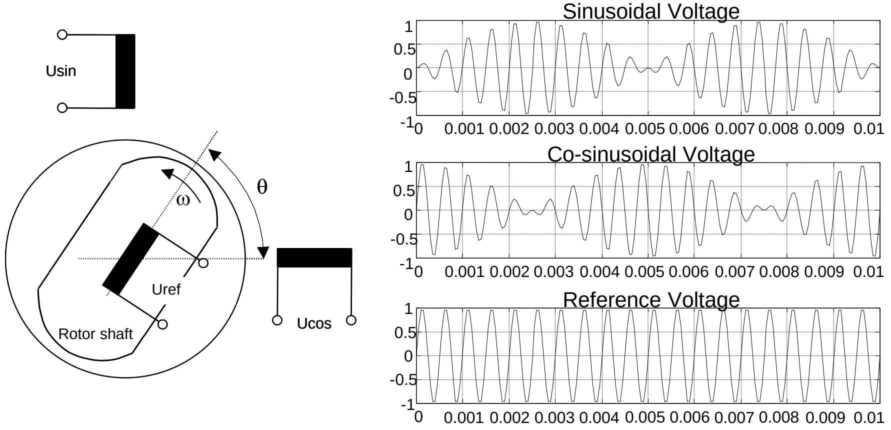
Motor Control: Resolvers

- Rotor is put directly on the drive’s shaft
- Stator is fixed on drive’s shield
- Simple assembly and maintenance
- No bearings — “unlimited” durability
- Resist well against distortion, vibration, deviation of operating temperature and dust
- Worldwide consumption millions of pieces at present time
- Widely used in precious positioning applications
- The number of generated sine and cosine cycles per one mechanical revolution depends on the number of resolver pole-pairs (usually 1-3 cycles)

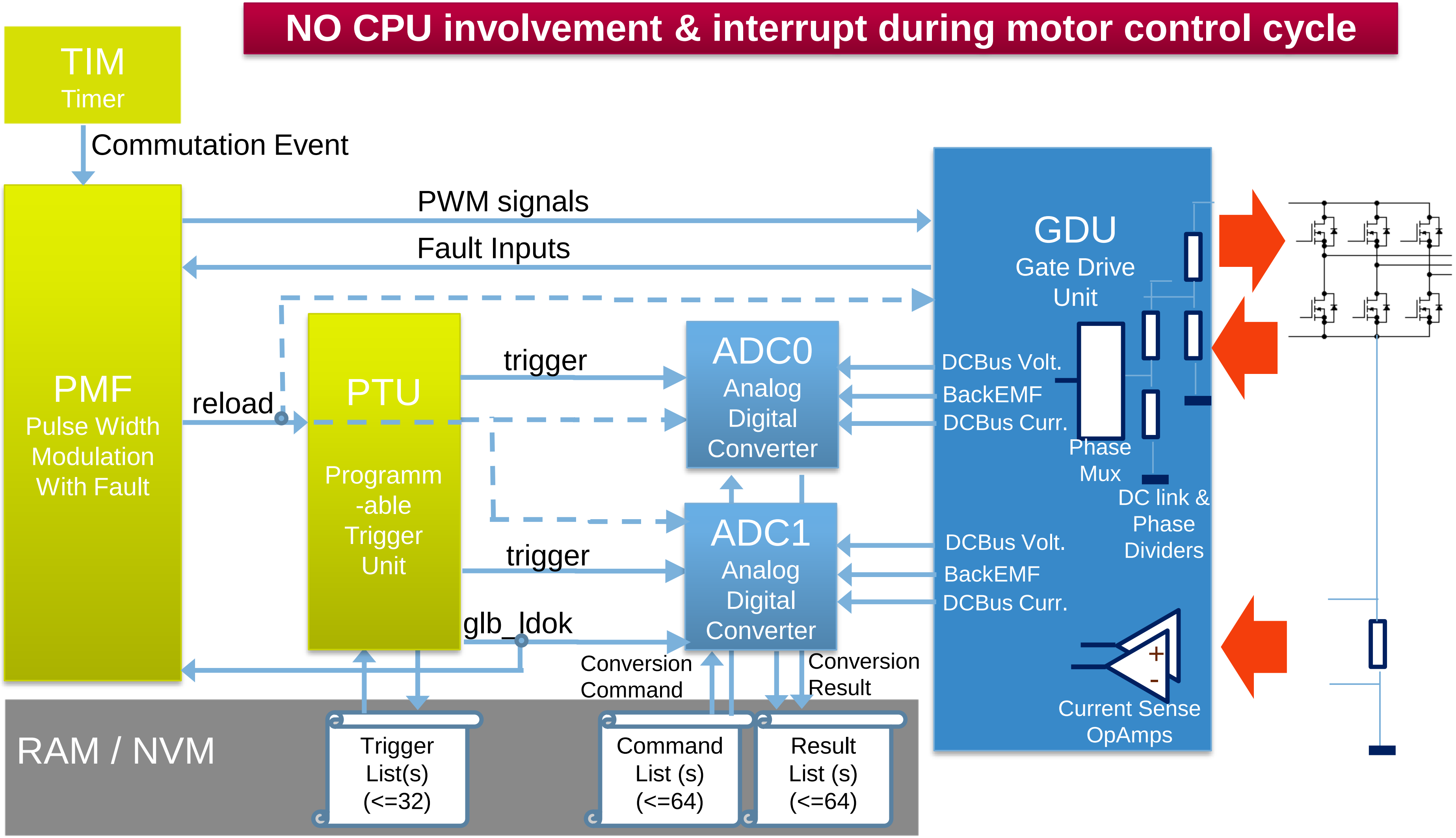
Sensor Components



Sensor Principle



Motor Control: Autonomous Motor Control Loop Implementation

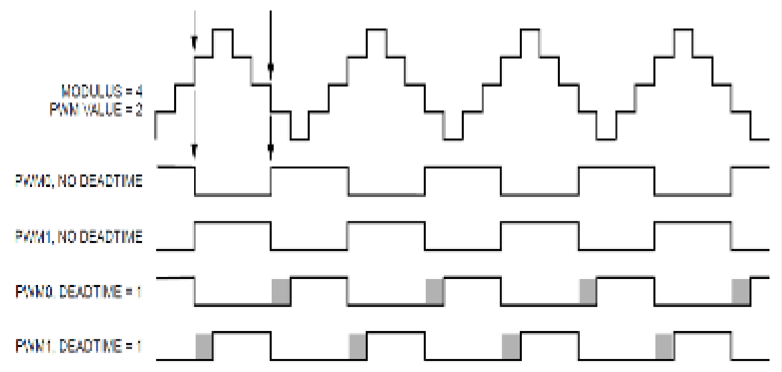


Motor Control: Pulse Width Modulator Module (PMF)

- **6 PWM channels, 3 independent counters**
  - Up to 6 independent channels or 3 complementary pairs
- **Based on core clock (max. 100 MHz)**
- **Complementary operation:**
  - Dead time insertion
  - Top and Bottom pulse width correction
  - Double switching
  - Separate top and bottom polarity control
- **Edge- or center-aligned PWM signals**
- **Integral reload rates from 1 to 16**
- **6-step BLDC commutation support, with optional link to TIM Output Compare**
- **Individual software-controlled PWM outputs (+ easy masking feature per output)**
- **Programmable fault protection**

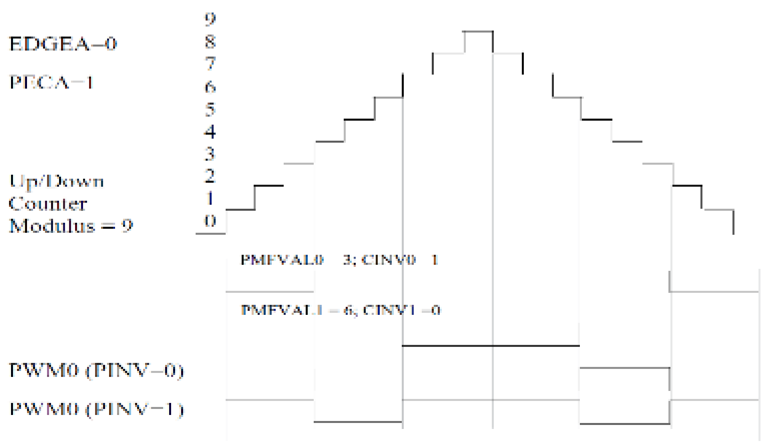
Complementary Mode

with / without dead time insertion

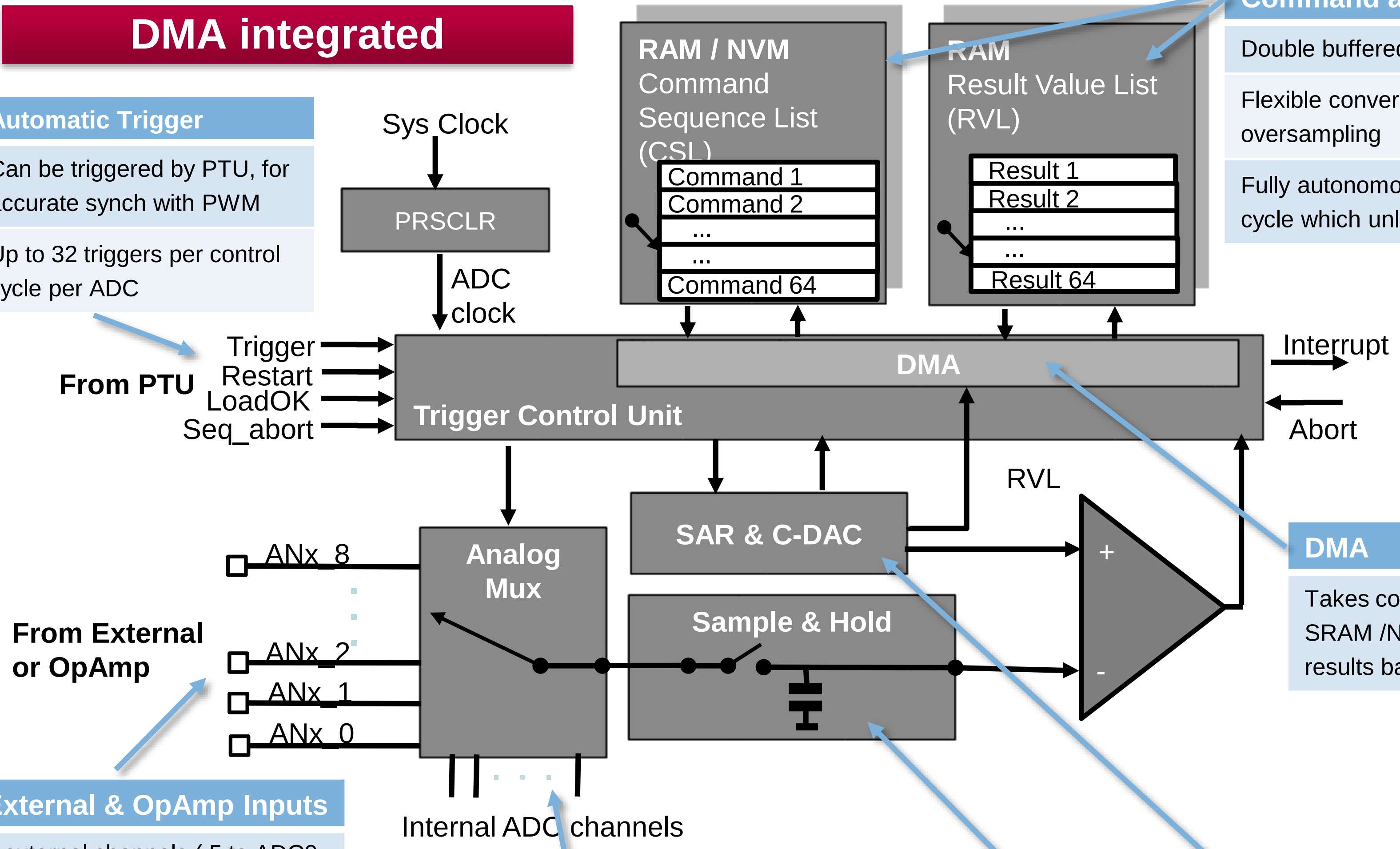


Double-Switching Mode

for single shunt system



Motor Control: 2 x 12 bit Analog Digital Converter



Command and Result List

- Double buffered
- Flexible conversion sequence and oversampling
- Fully autonomous motor control cycle which unloads CPU

Automatic Trigger

- Can be triggered by PTU, for accurate synch with PWM
- Up to 32 triggers per control cycle per ADC

DMA

- Takes commands from SRAM /NVM and stores results back into SRAM

External & OpAmp Inputs

- 9 external channels (5 to ADC0 and 4 to ADC1)
- OpAmp output shared with ADC external channel

Monitoring Internal Signals

- DC link, phase voltages, Vsup
- Vreg & ADC temp sensors
- Bandgap voltage

Sample Time

- Selectable: 480 ns to 2.88 µs

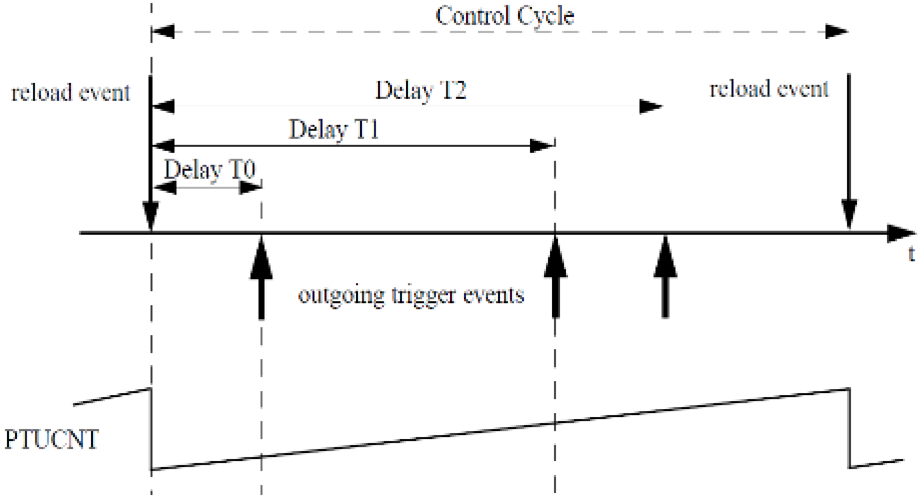
Conversion Time

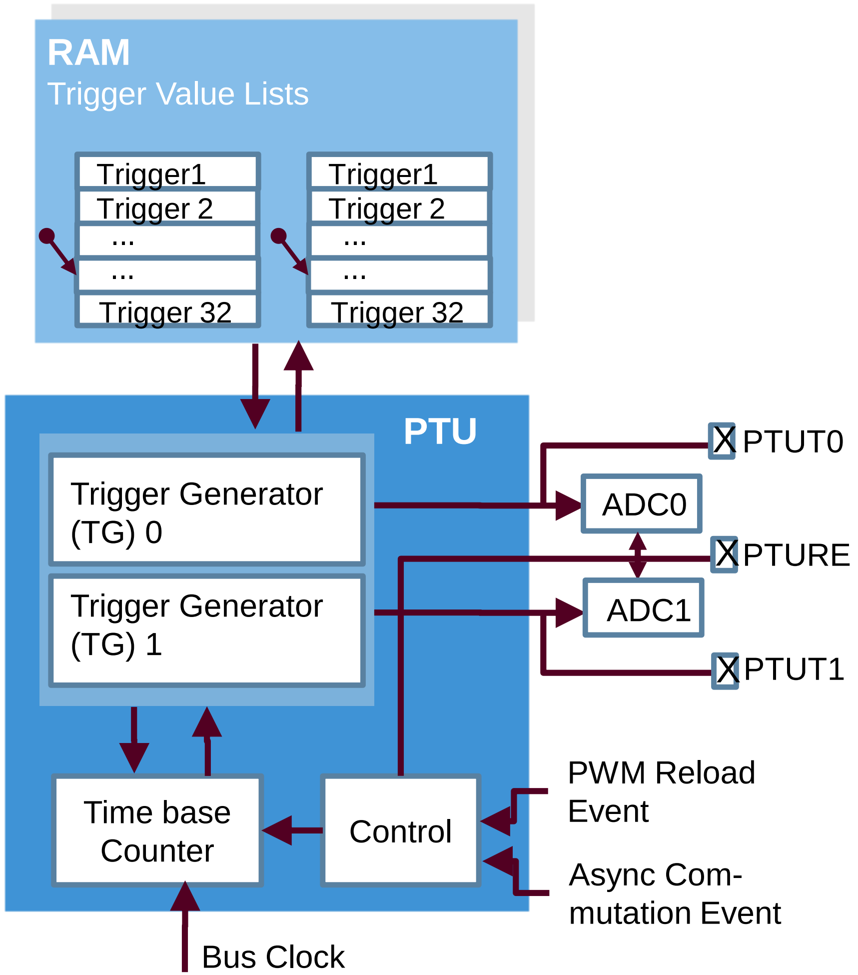
- 1.8 µs @ max. ADC clock for 12 bit

Motor Control: Programmable Trigger Unit (PTU)

Completely avoids CPU involvement to trigger ADC during the control cycle

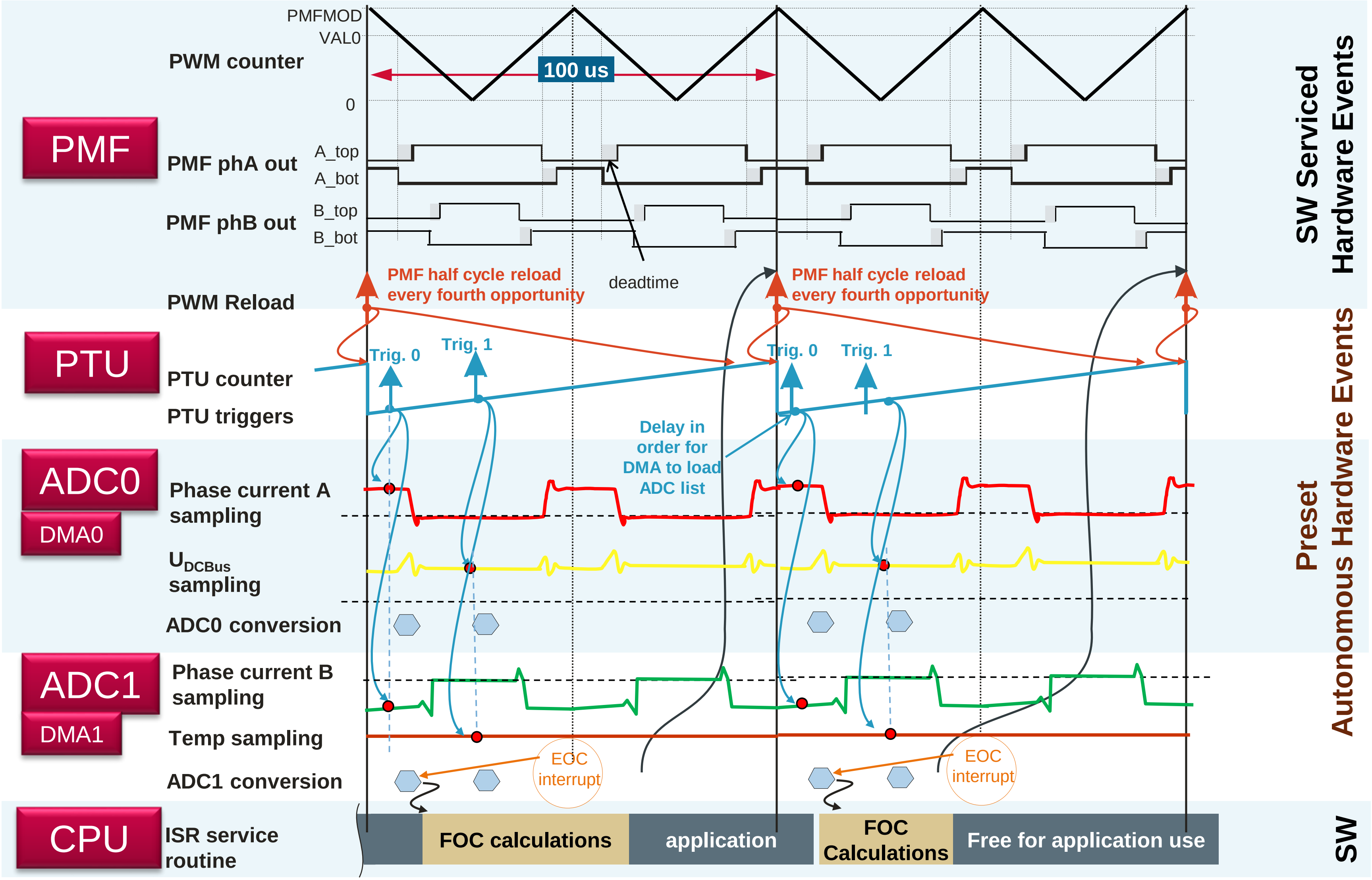
- One 16-bit counter as time base
- Two independent trigger generators (TG)
- Up to 32 trigger events per trigger generator
- Trigger Value List stored in system memory
- Double buffered list, so that CPU can load new values in the background
- Software generated “Reload” & trigger event
- Synchronized with PMF and ADC to guarantee coherent update of all control loop modules





Motor Control: Autonomous PMSM Application Timing

Two shunts current sensing



Motor Control: Rotor Position Sensor Elimination

- ***FOC*** requires accurate position and velocity signals
- Conventional motion control systems uses resolvers or encoders
- Sensor, wirings, connectors increase the cost of the system and decrease the reliability
- Application Sensorless PM Motor Control In
  - Lower overall drive cost by eliminating mechanical position sensor
    - cost sensitive application
    - increase system performance for the same price
  - Increase position resolution in collaboration of estimator and low cost position sensor
    - increase system performance
    - back-up sensor
  - Independent position sensing together with mechanical
    - safety critical application
    - increase system redundancy
- A sensorless motor strategy is good for applications that don’t have the motor to stop and does not change direction (ex. Fuel pumps).

Motor Control: What is Back EMF and how to use it

- An electric motor acts like a generator and can generate a secondary force that opposes the original electromotive force (EMF) called back EMF. There is a direct correlation between the back EMF and position of a motor.
- Since there is a direct correlation between motor position and back EMF voltage amplitude we can use it to get motor position needed for FOC.
- In sensor-less trapezoidal the back EMF zero crossings are detected by measuring the phase voltage of the coil that is not energized, but in FOC that is not the case.
- All coils are energized thru the full commutation cycle. Therefore a back EMF observer is used to estimate the back EMF using a simplified model of a PMSM motor, phase current feedback, and command voltage information.
- To measure the back EMF a certain amount of motor speed is required so that enough back EMF voltage is generated for measurement. Therefore a minimum speed is required for sensorless FOC algorithms to operated.

Motor Control: Simplified Sensorless PM Control

- **Force Mode –** Use the requested speed and position as the estimated speed and position for the back-EMF observer and FOC control. FOC always uses phase currents from the shunt resisters. Speed control uses requested speed and position to close the loop on speed.

- **Tracking Mode –** The back-EMF observer closes the loop and uses it own estimated position and speed vs. the requested position and speed for feedback. Speed control uses requested speed and position to close the loop on speed.

- **Sensorless Mode –** Speed control uses back-EMF observer speed and position to close the loop on speed.

- Basically each mode transition happens at a different requested speed value to eventually have the motor spin fast enough to generated enough back-EMF to accurately estimate the motor position and speed.

Agenda

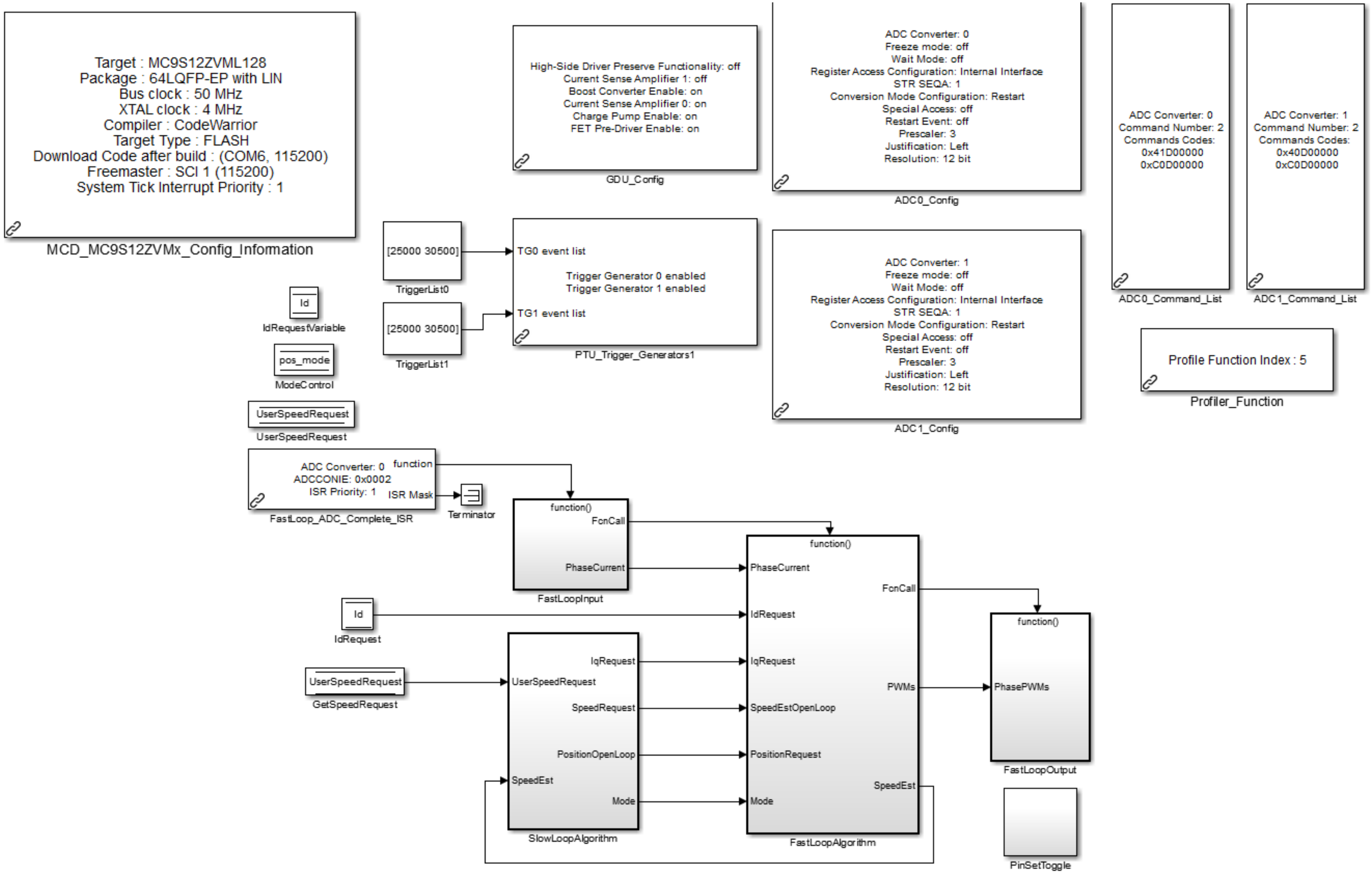
- **Overview:**
  - Introduction and Objectives
  - Model-Based Design Toolbox: Library blocks, FreeMASTER, and Bootloader
- **Hands-On Demo:**
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Convert simple model to run on Motor Kit with MCD Toolbox and use FreeMASTER
- **Model-Based Design:**
  - Model-Based Design Steps: Simulation, SIL, PIL and ISO 26262
  - SIL/PIL Hands-On Demo Step 2 & 3 of MBD
- **Trapezoidal Motor Control**:
  - Motor Kit (Describe Freescale 3-Phase Motor Kit)
  - Trapezoidal control and how to use it to turn a motor
- **Trapezoidal Motor Control Hands-on Demo**:
  - Implement Trapezoidal Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor and tune parameters
- **FOC Motor Control**:
  - FOC Sensor-less control and how to use it to turn a motor
- **FOC Motor Control Hands-On Demo:**
  - Implement FOC Sensor-less Motor Control on Motor Kit
  - Run software from the model and use FreeMASTER to monitor
- **Summary and Q&A:**

Demo: Implement FOC Sensor-Less Motor Control

Summary FOC Sensor-less Motor Control on S12ZVM steps:

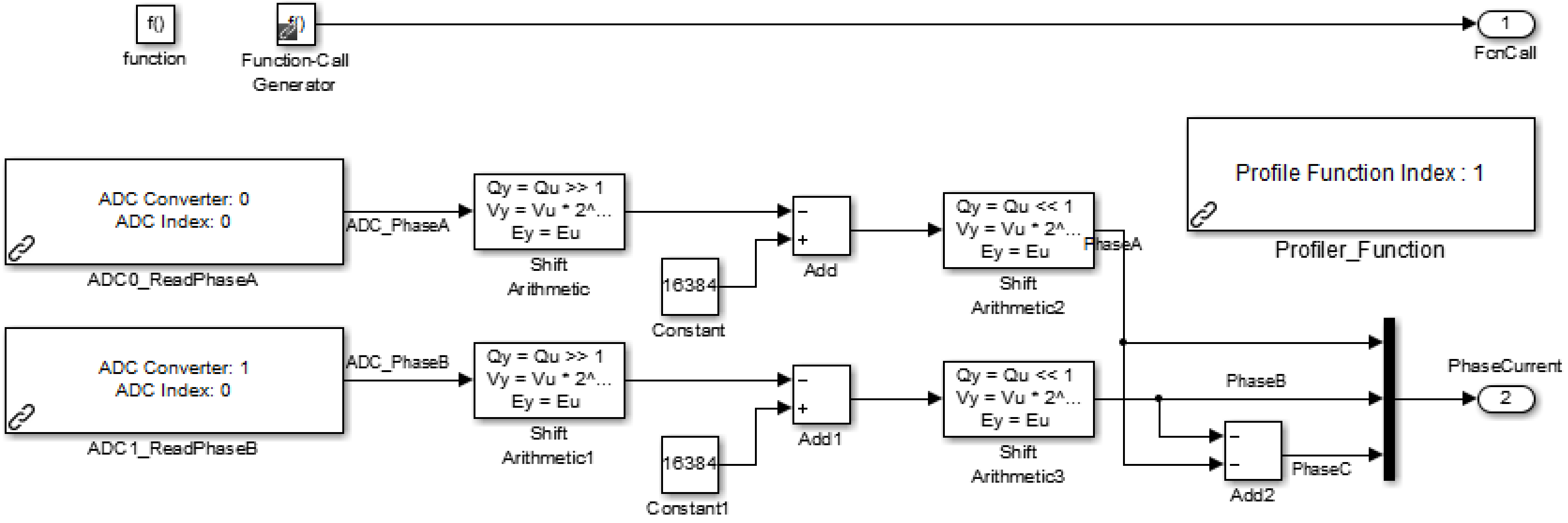
1. Open FOC_Sensorless.slx
2. Save model as S12ZVM_FOC_Sensorless.slx
3. Configure S12ZVM thru model configuration parameters
4. Configure the Gate Driver Unit with the GDU Configuration Block
5. Configure ADC blocks to read phase currents
6. Set up PTU triggers to synchronize current readings with PWM
7. Setup ADC interrupt to start FOC Fast Loop
8. Configure Digital Input for Toggling LED
9. Connect and configure PMF PWM blocks for output to switches thru GDU

Demo: Implement FOC Sensor-Less Motor Control



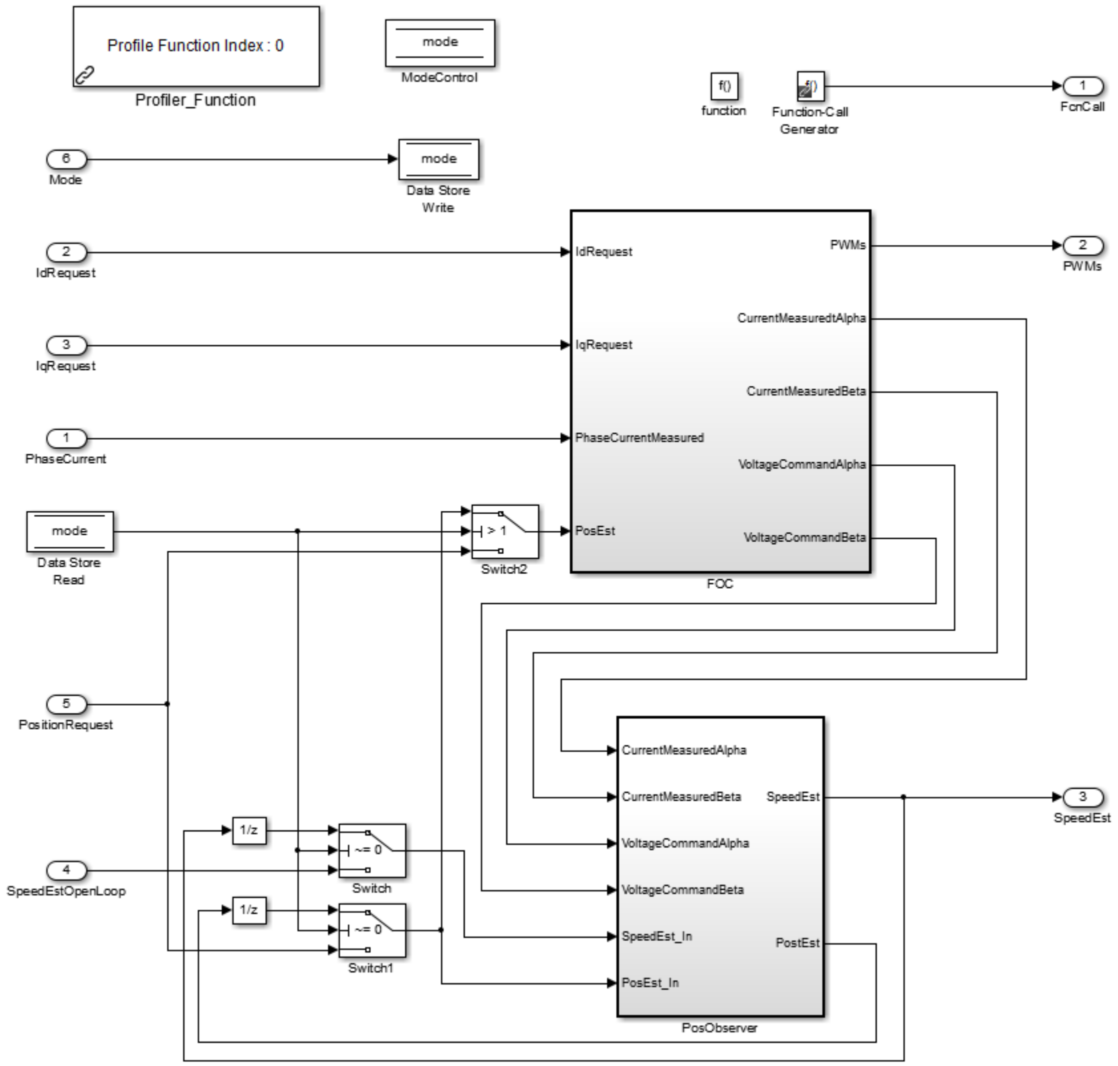
Demo: Implement FOC Sensor-Less Motor Control

FastLoopInput Block



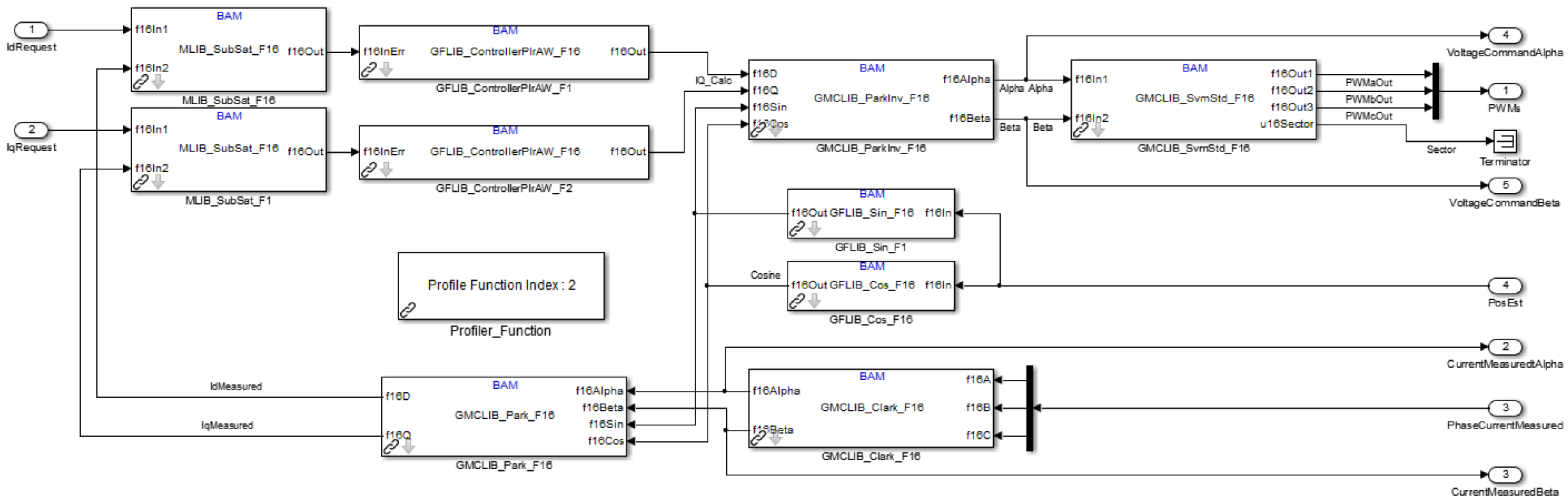
Demo: Implement FOC Sensor-Less Motor Control

FastLoopAlgorithm Block



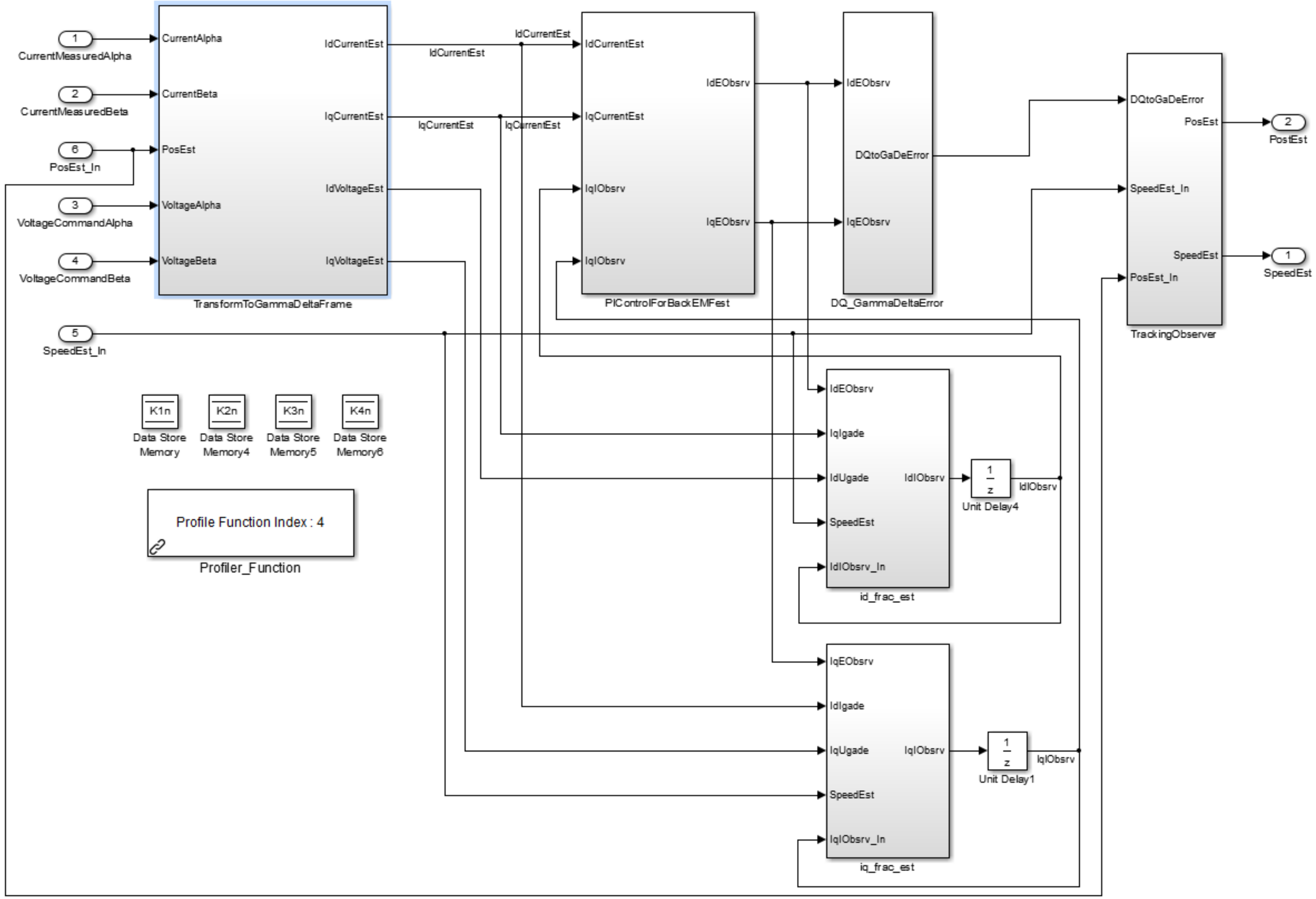
Demo: Implement FOC Sensor-Less Motor Control

FastLoopAlgorithm FOC Block



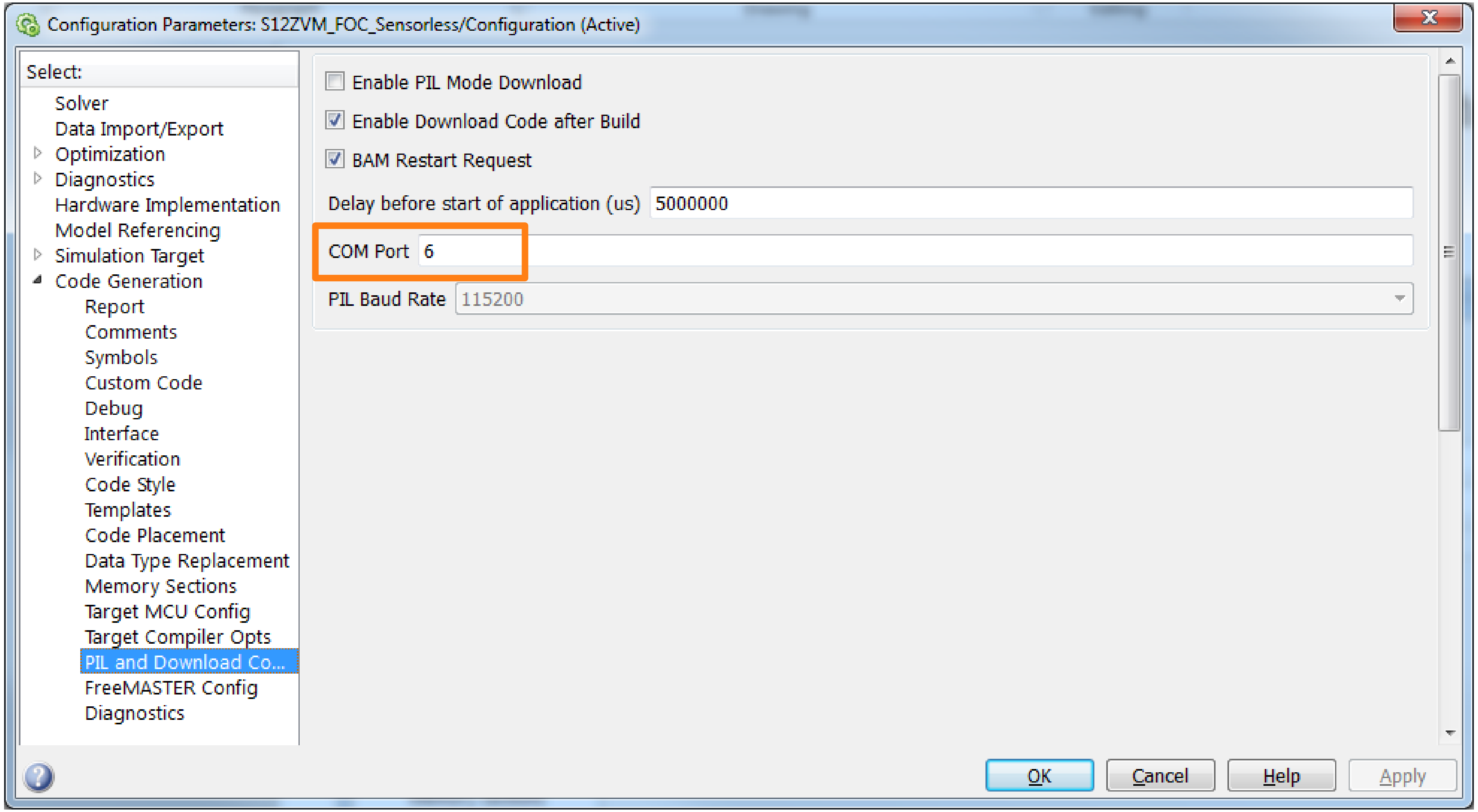
Demo: Implement FOC Sensor-Less Motor Control

FastLoopAlgorithm PosObserver Block

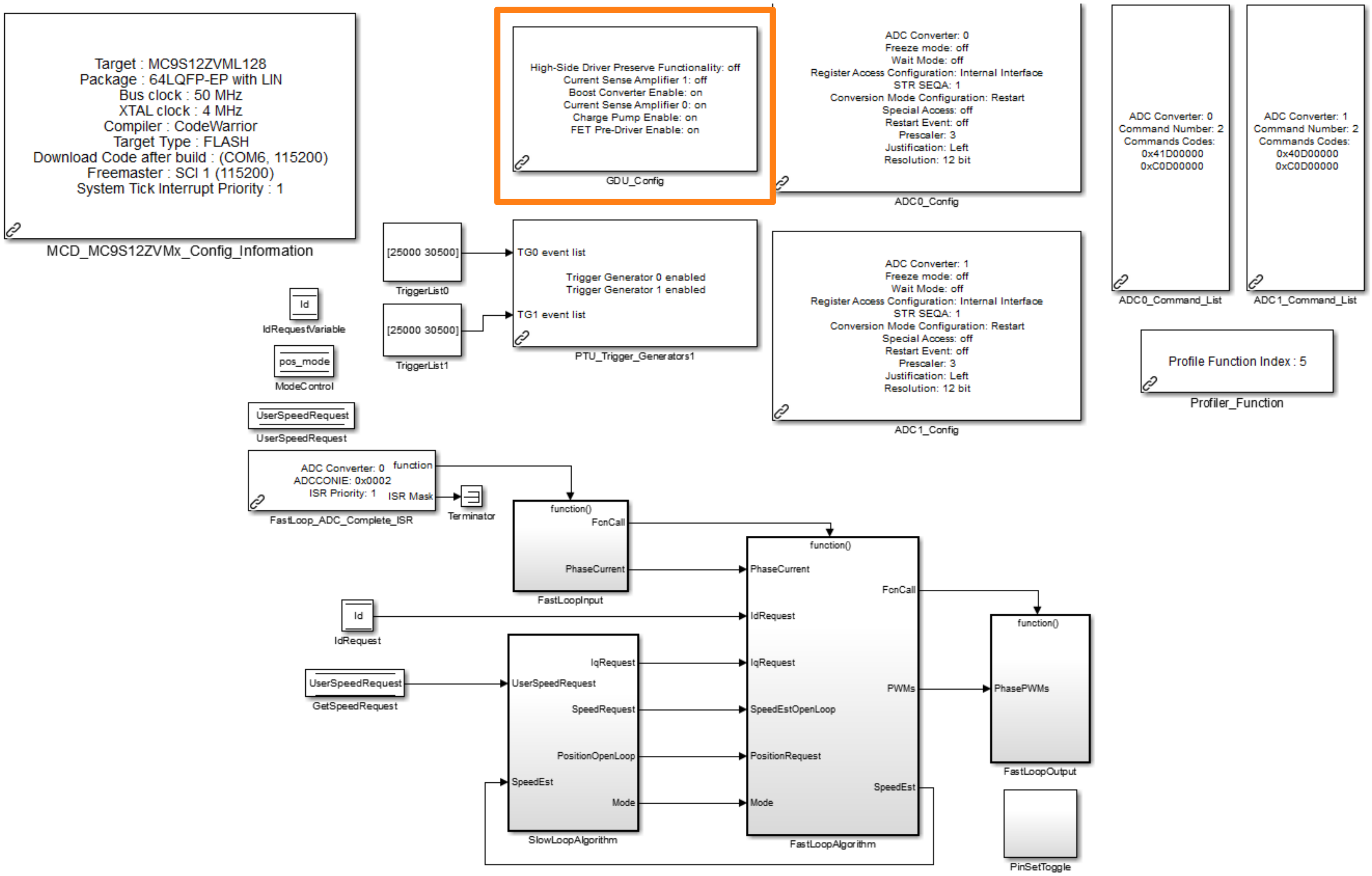


Demo: Implement FOC Sensor-Less Motor Control

COM Port Setup:

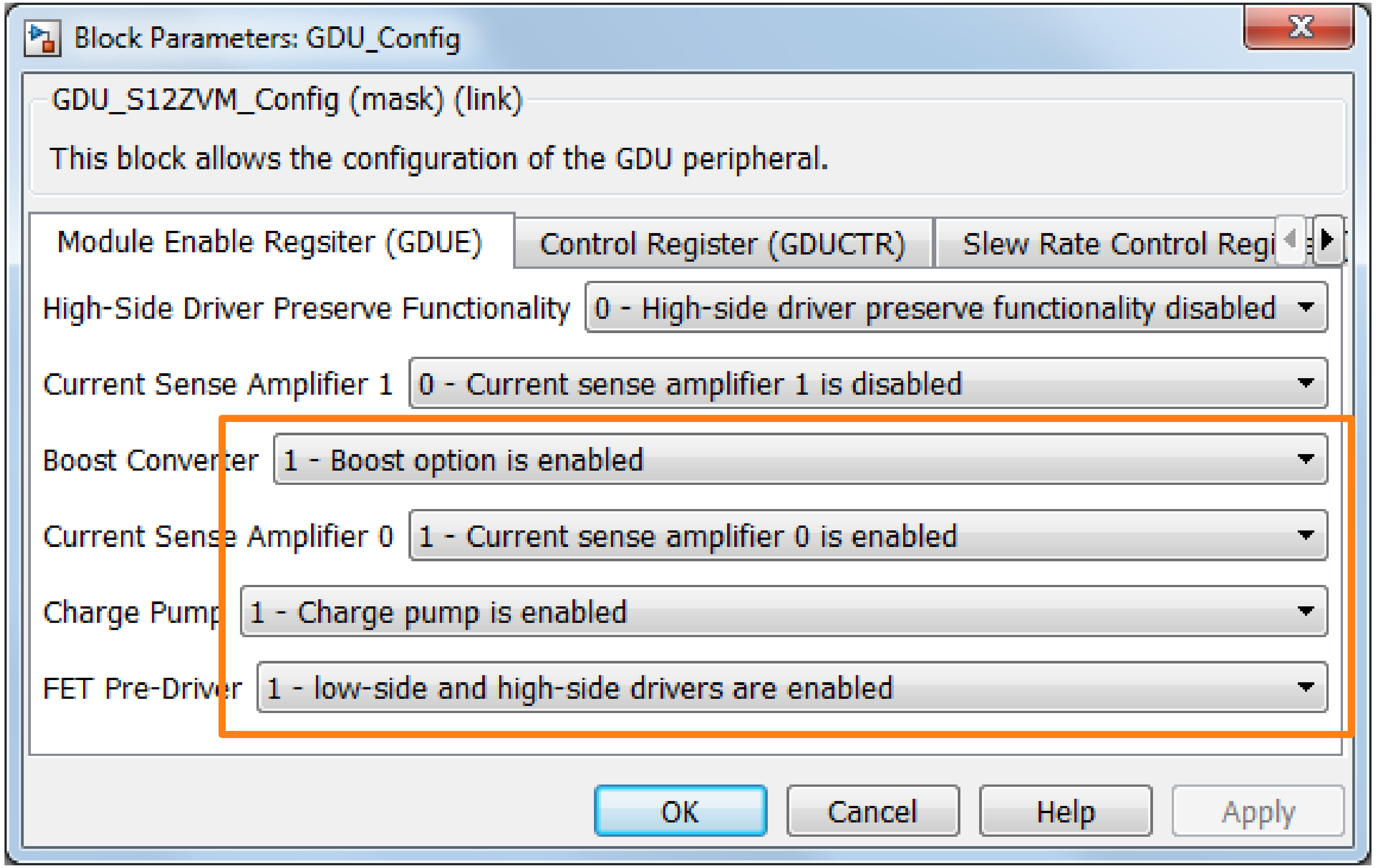


Demo: Implement FOC Sensor-Less Motor Control



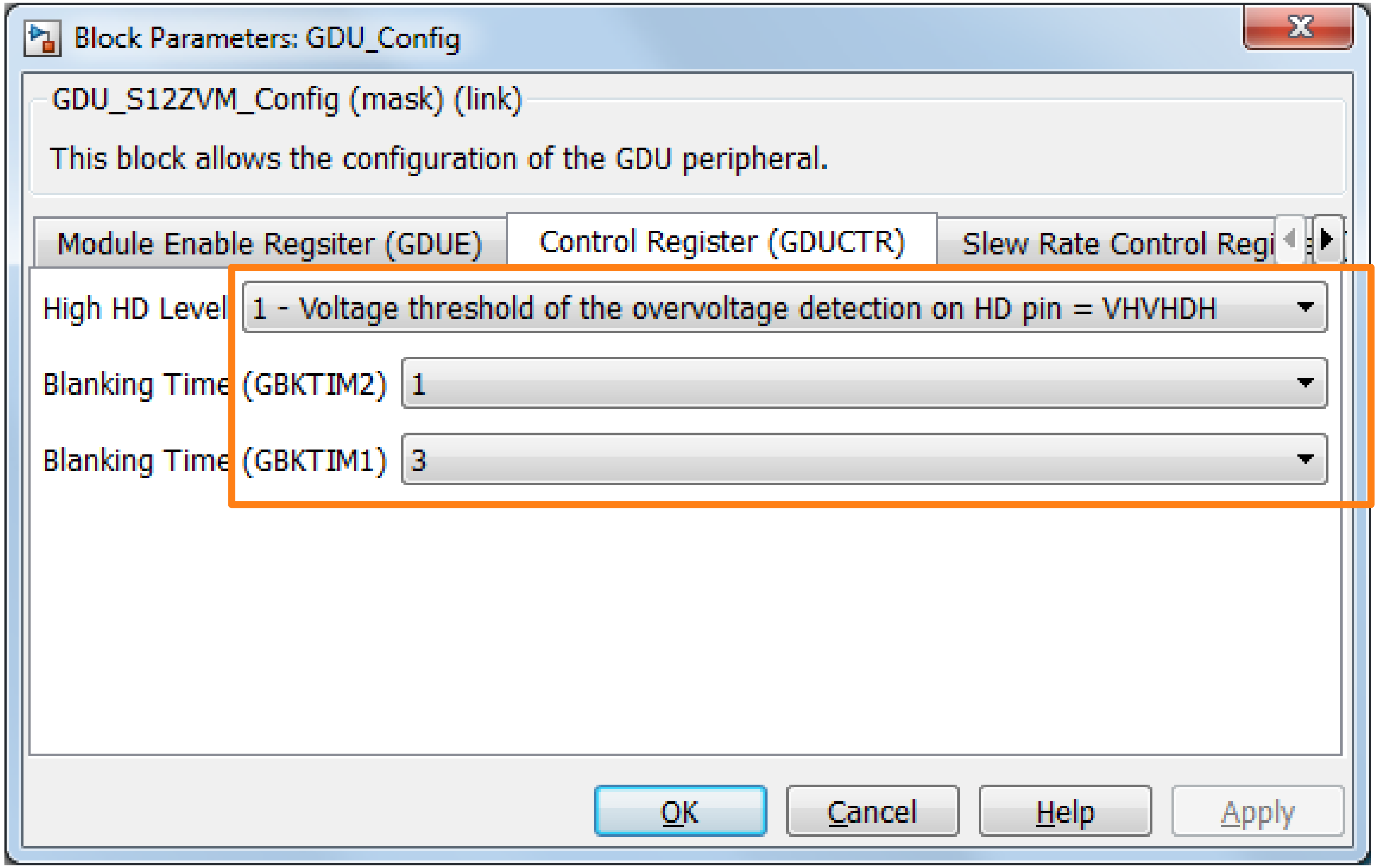
Demo: Implement FOC Sensor-Less Motor Control

Gate Drive Unit configuration steps:



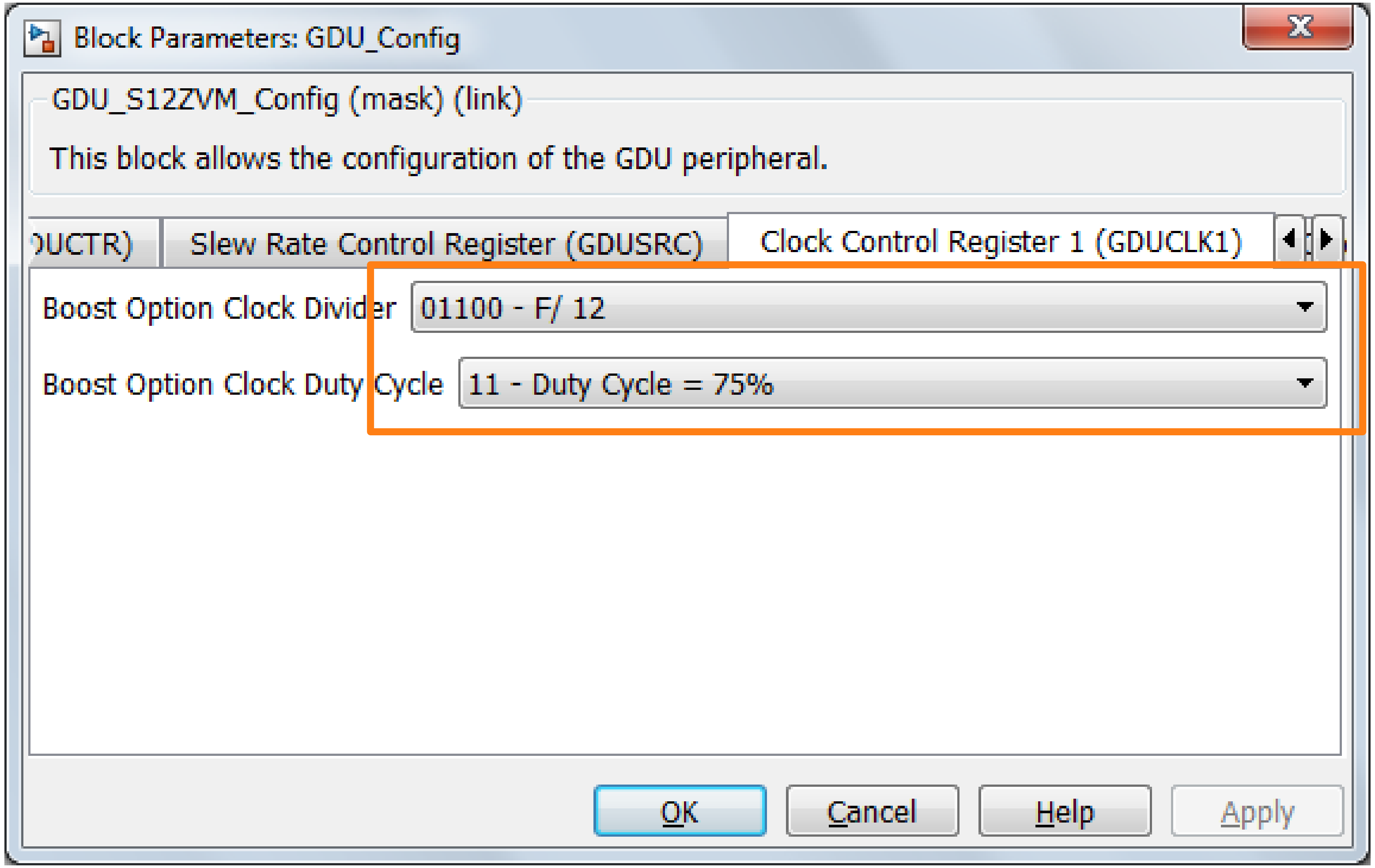
Demo: Implement FOC Sensor-Less Motor Control

Gate Drive Unit configuration steps (the settings = 720 nsec):



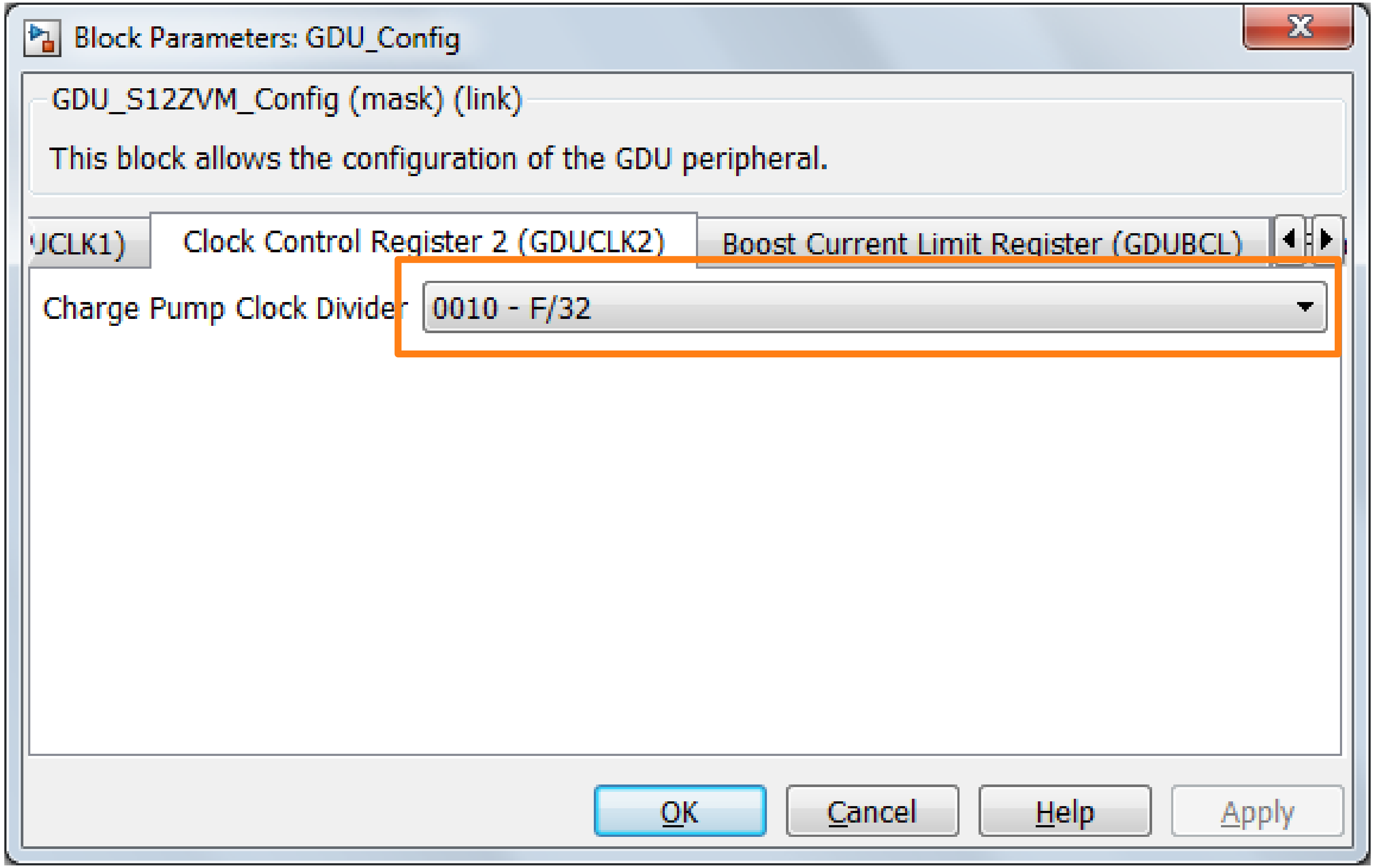
Demo: Implement FOC Sensor-Less Motor Control

Gate Drive Unit configuration steps:



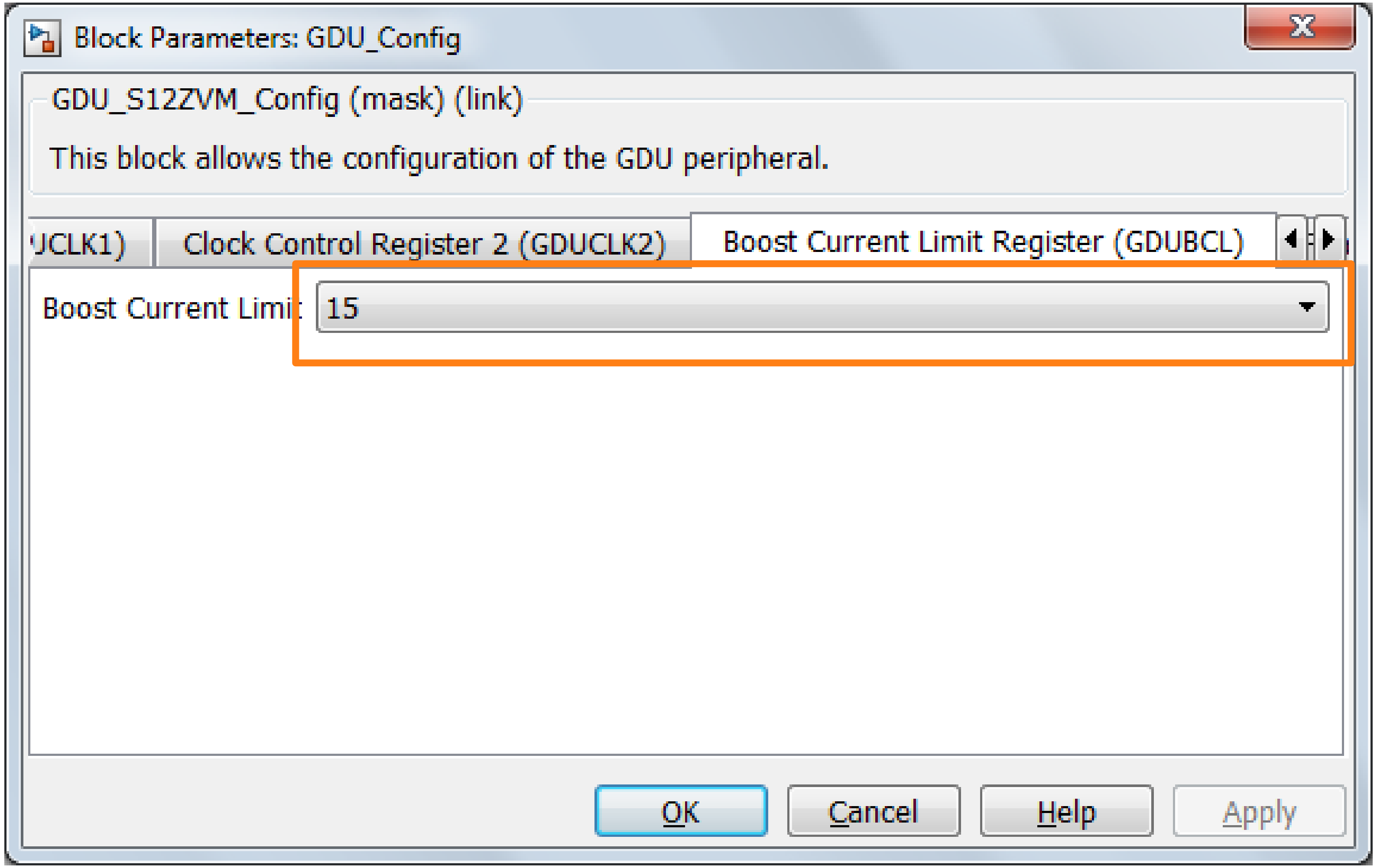
Demo: Implement FOC Sensor-Less Motor Control

Gate Drive Unit configuration steps:



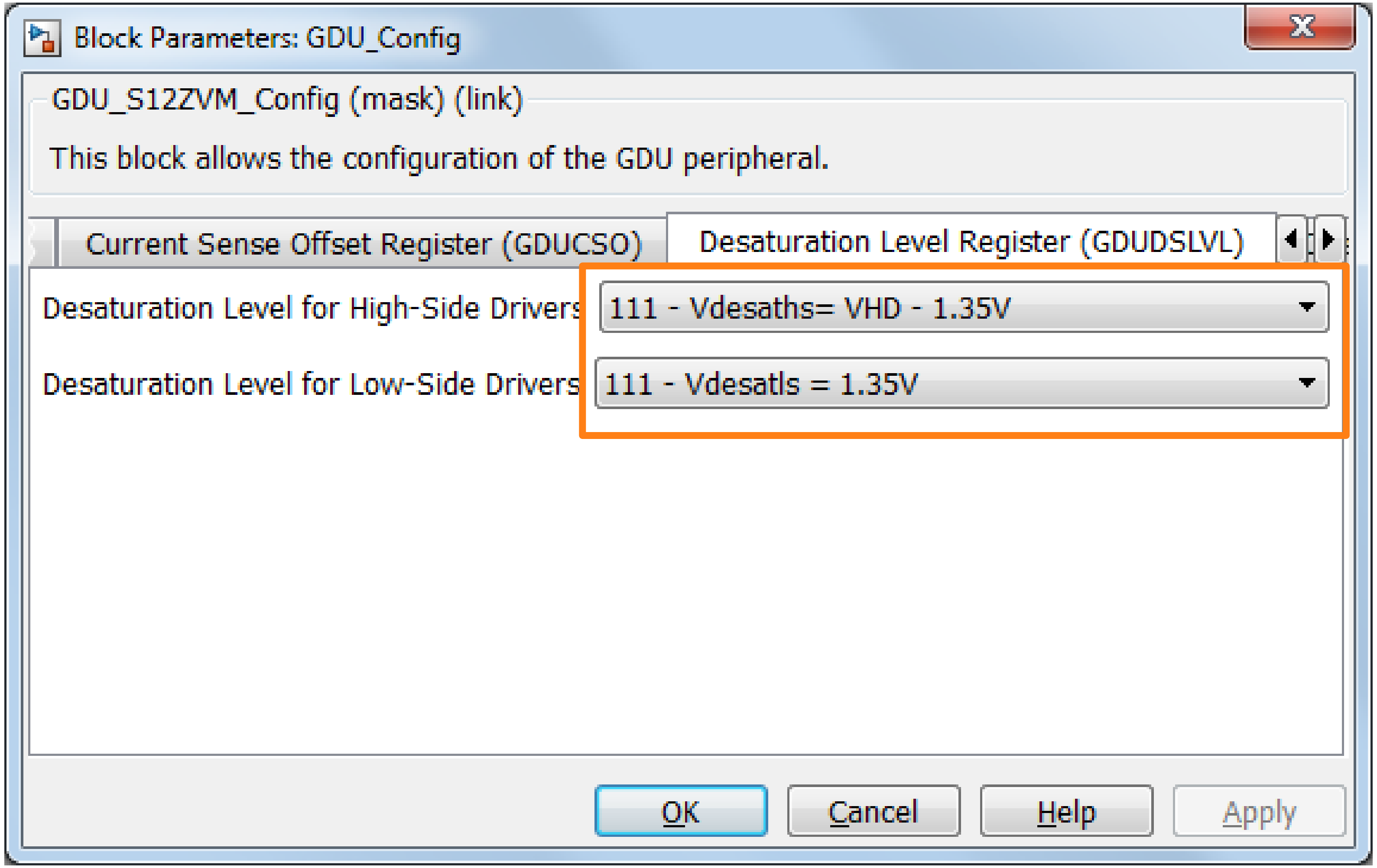
Demo: Implement FOC Sensor-Less Motor Control

Gate Drive Unit configuration steps:

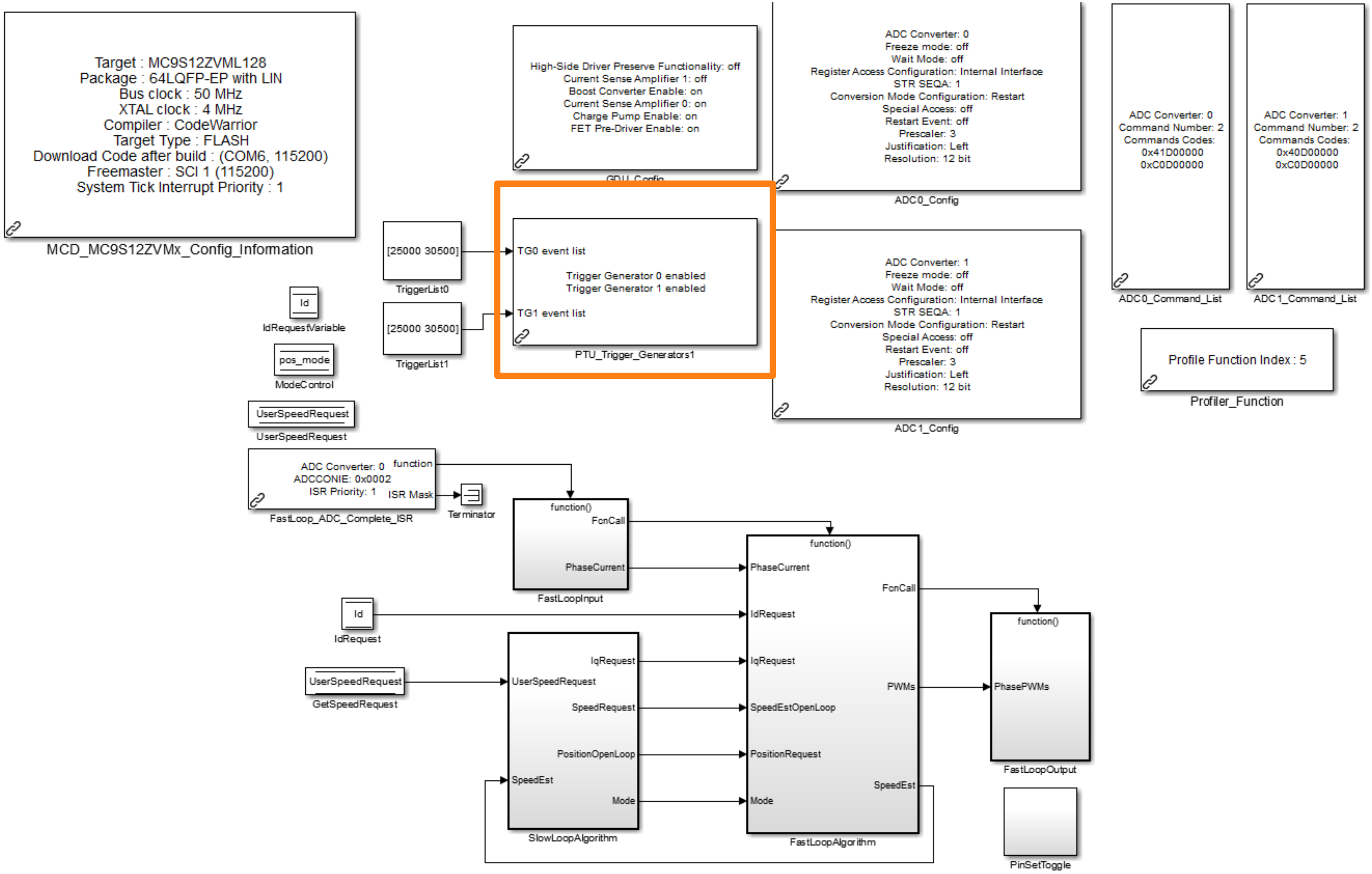


Demo: Implement FOC Sensor-Less Motor Control

Gate Drive Unit configuration steps:

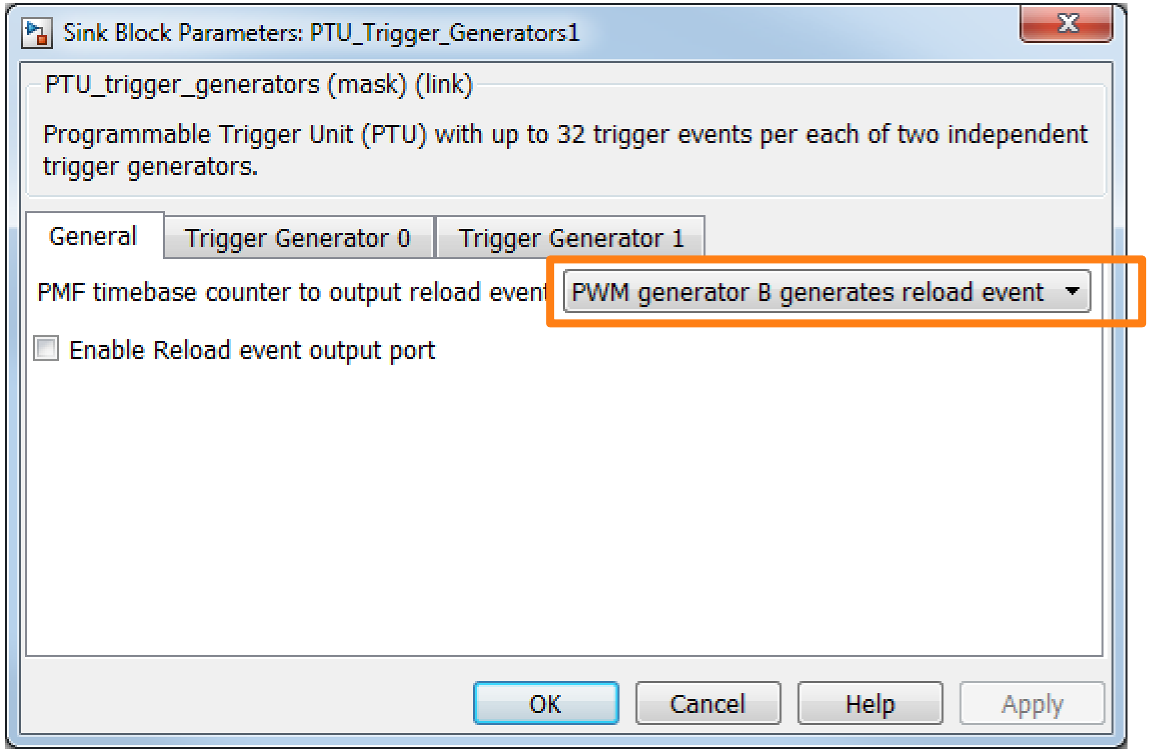


Demo: Implement FOC Sensor-Less Motor Control



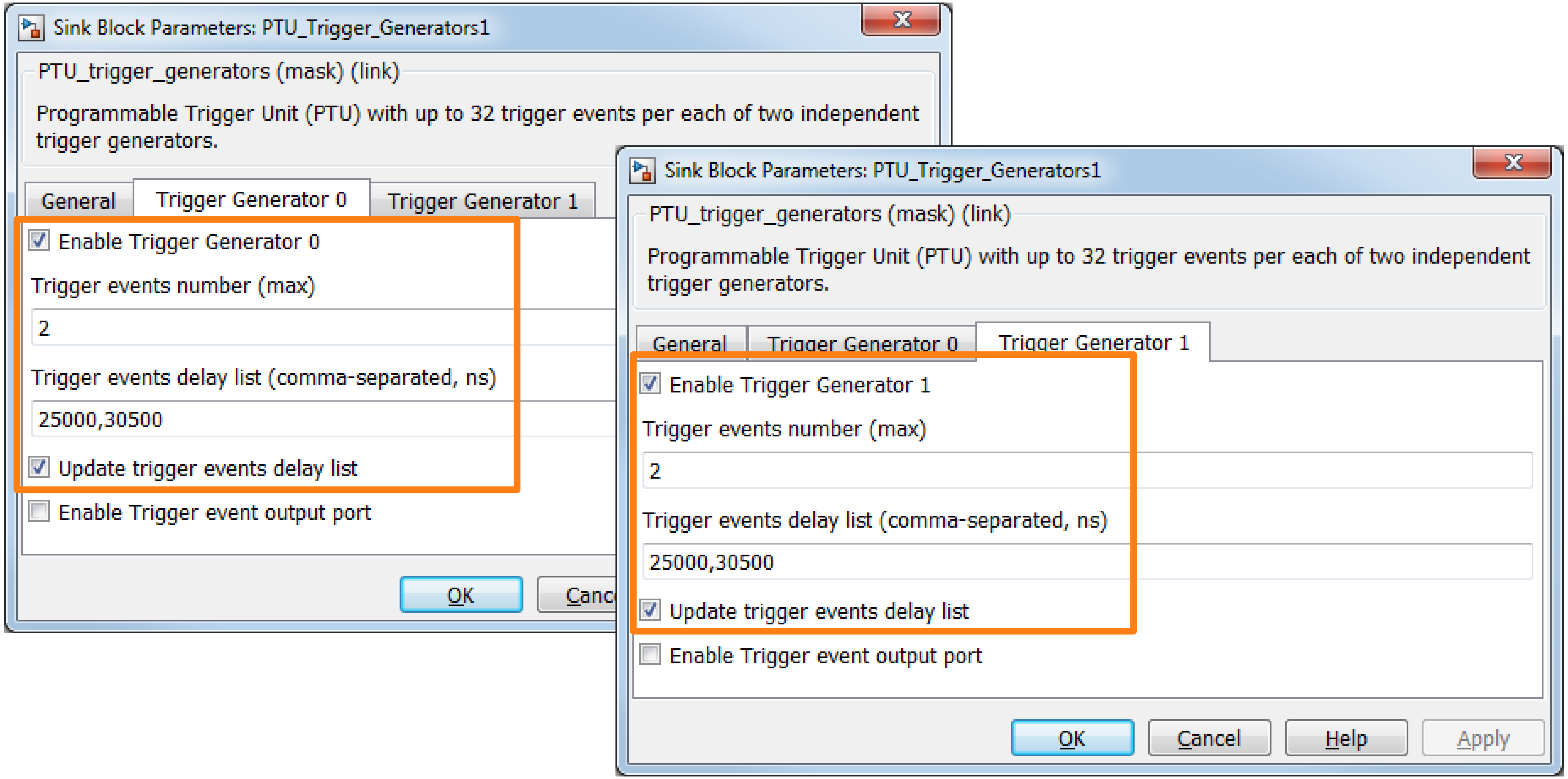
Demo: Implement FOC Sensor-Less Motor Control

PTU Unit configuration steps:



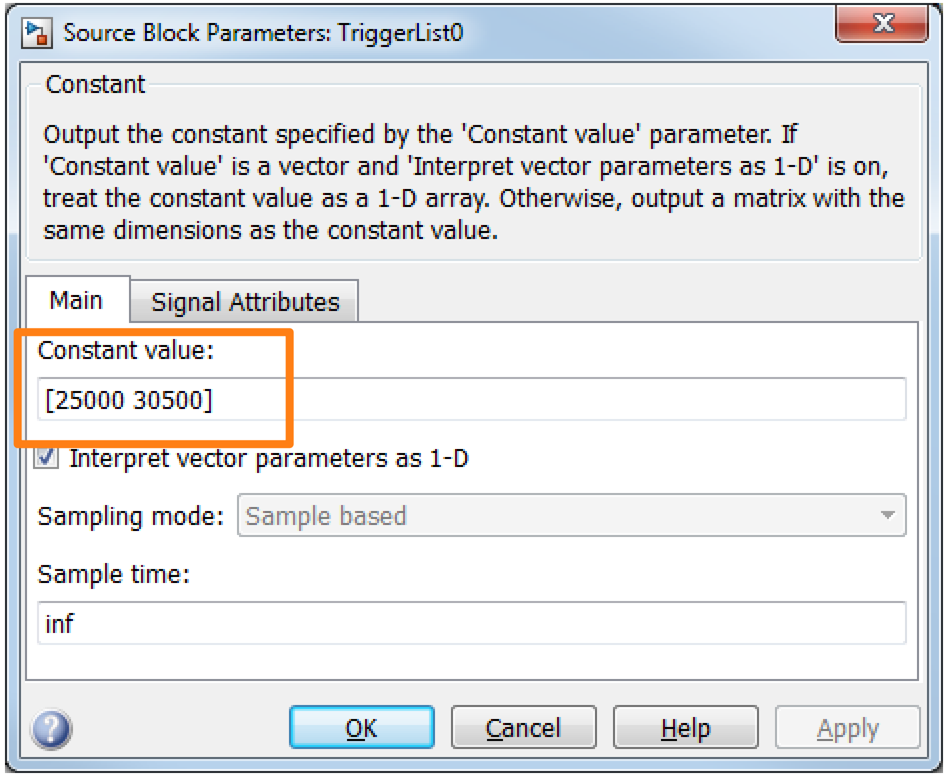
Demo: Implement FOC Sensor-Less Motor Control

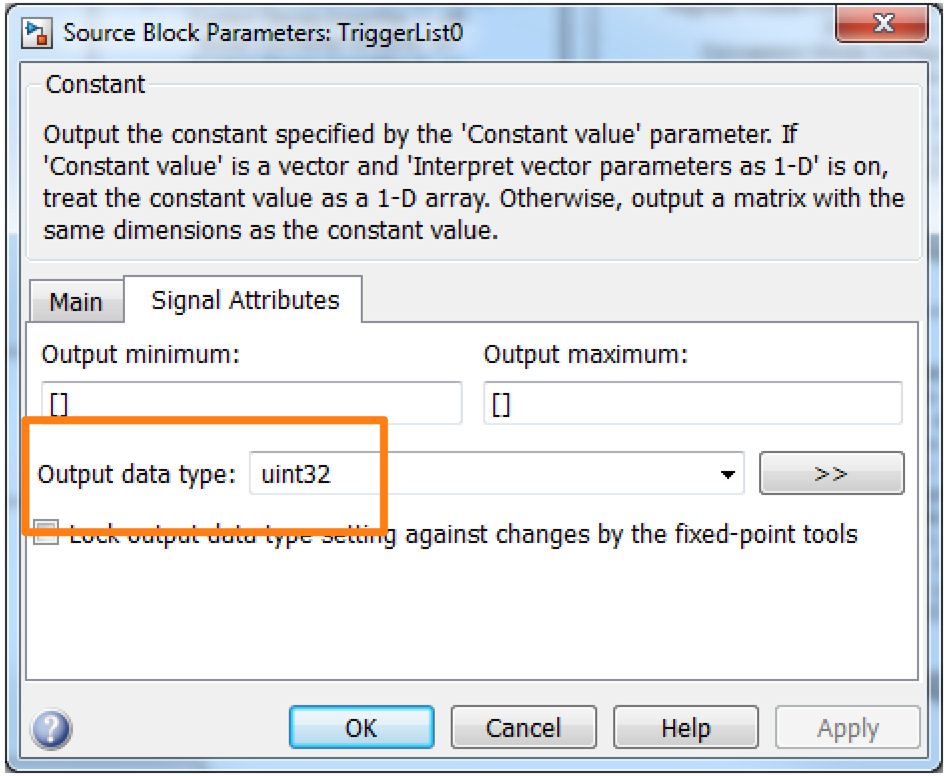
PTU Unit configuration steps:



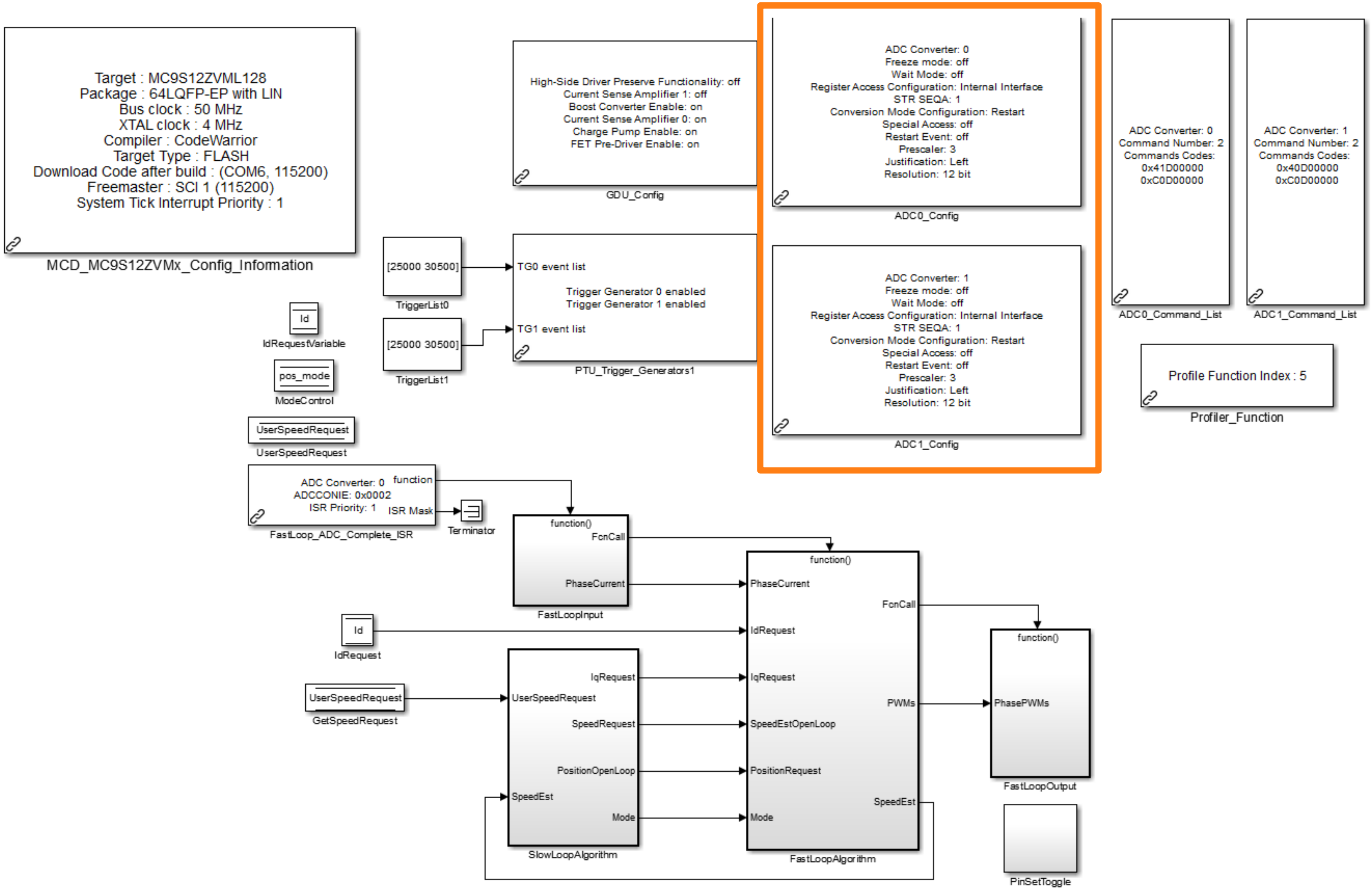
Demo: Implement FOC Sensor-Less Motor Control

PTU Unit configuration steps:



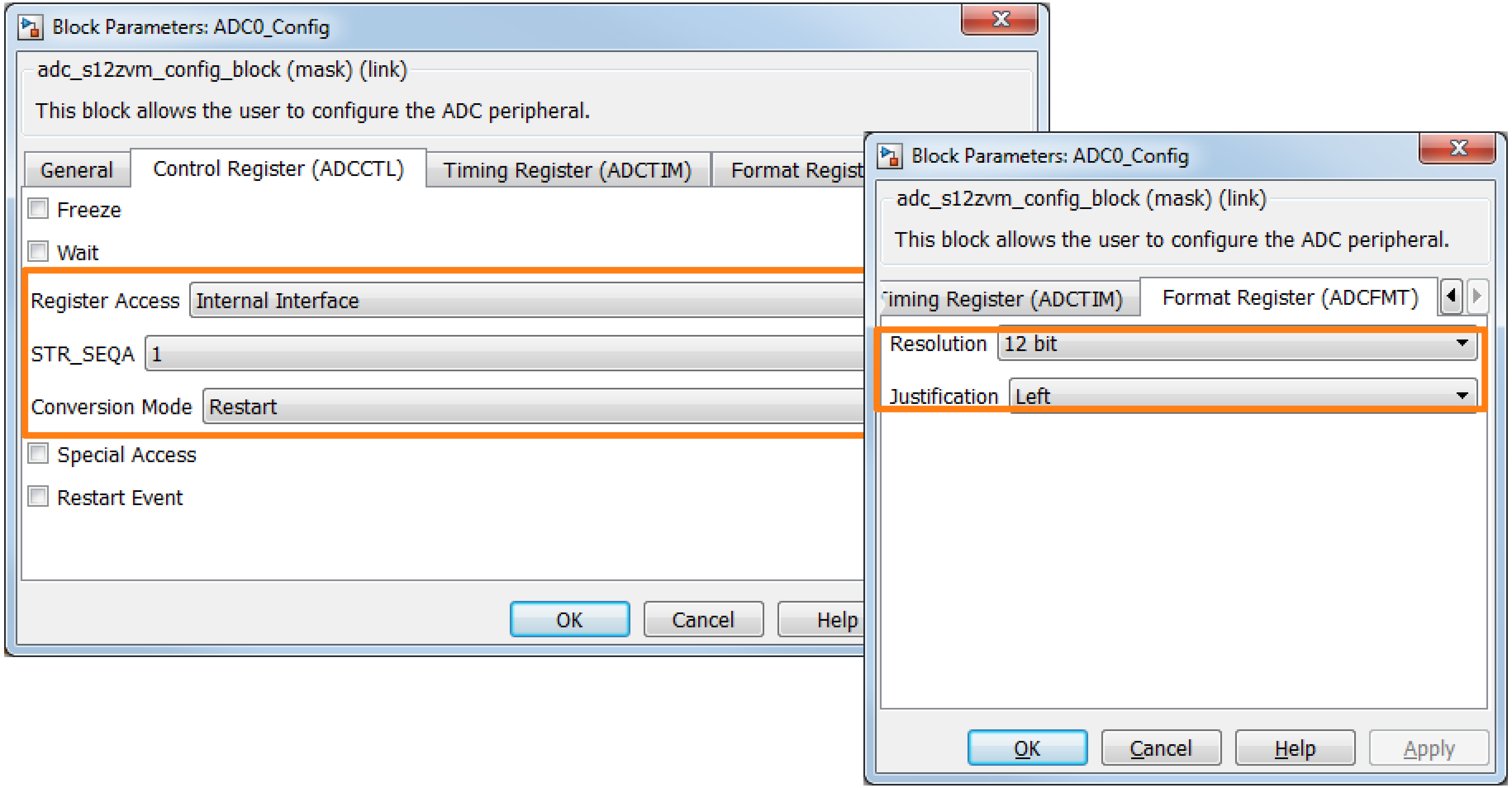


Demo: Implement FOC Sensor-Less Motor Control



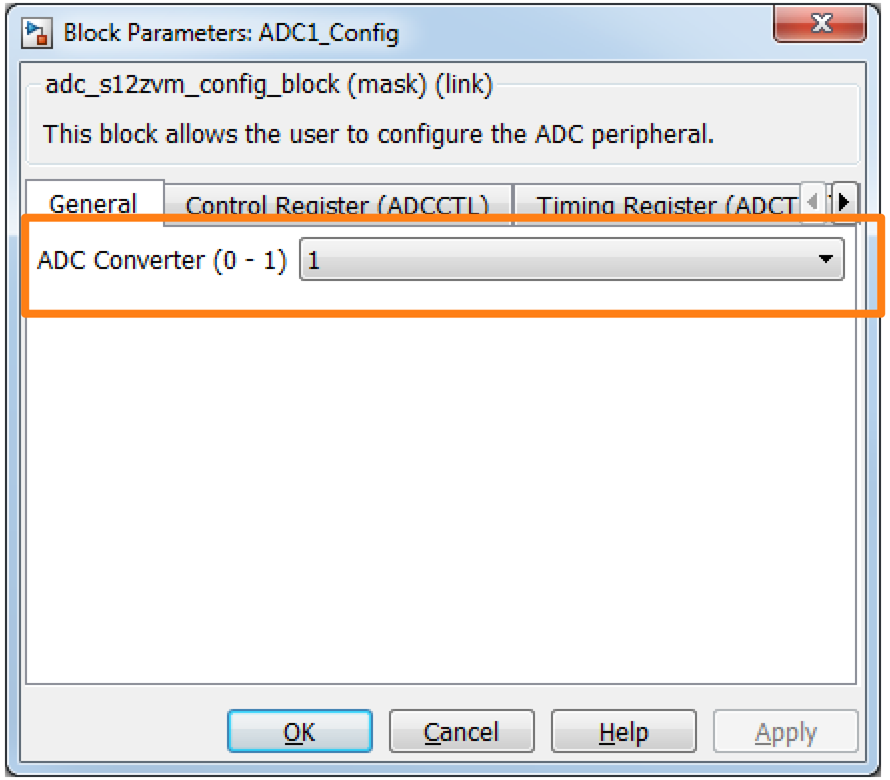
Demo: Implement FOC Sensor-Less Motor Control

ADC Unit configuration steps:



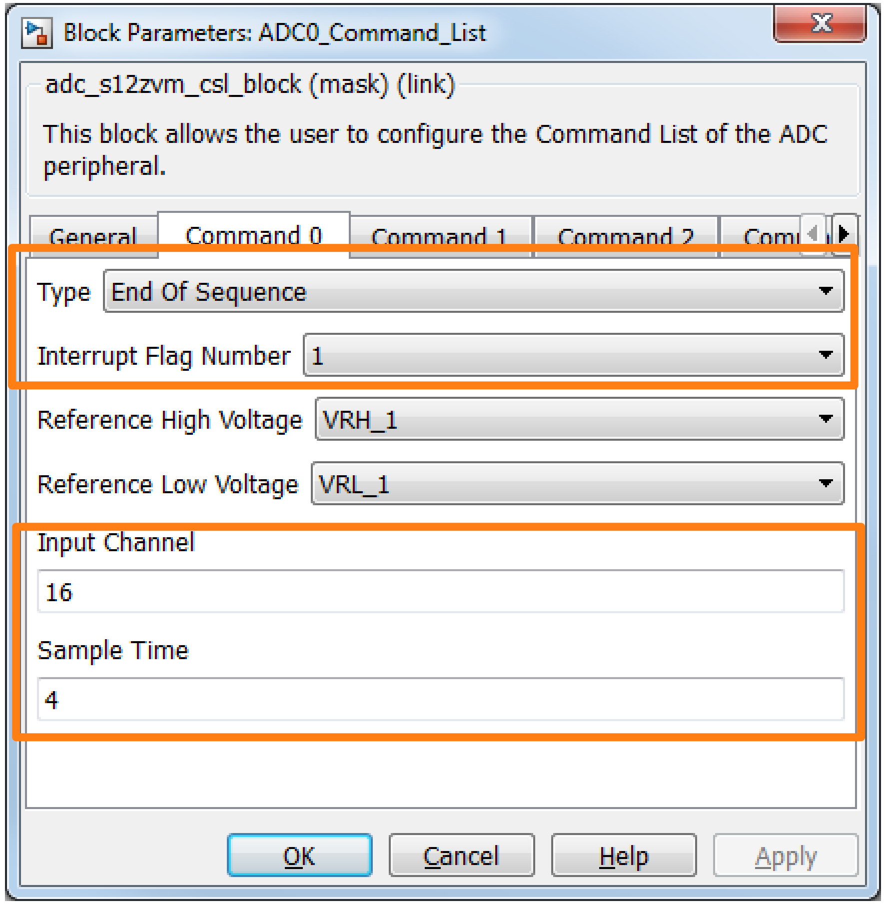
Demo: Implement FOC Sensor-Less Motor Control

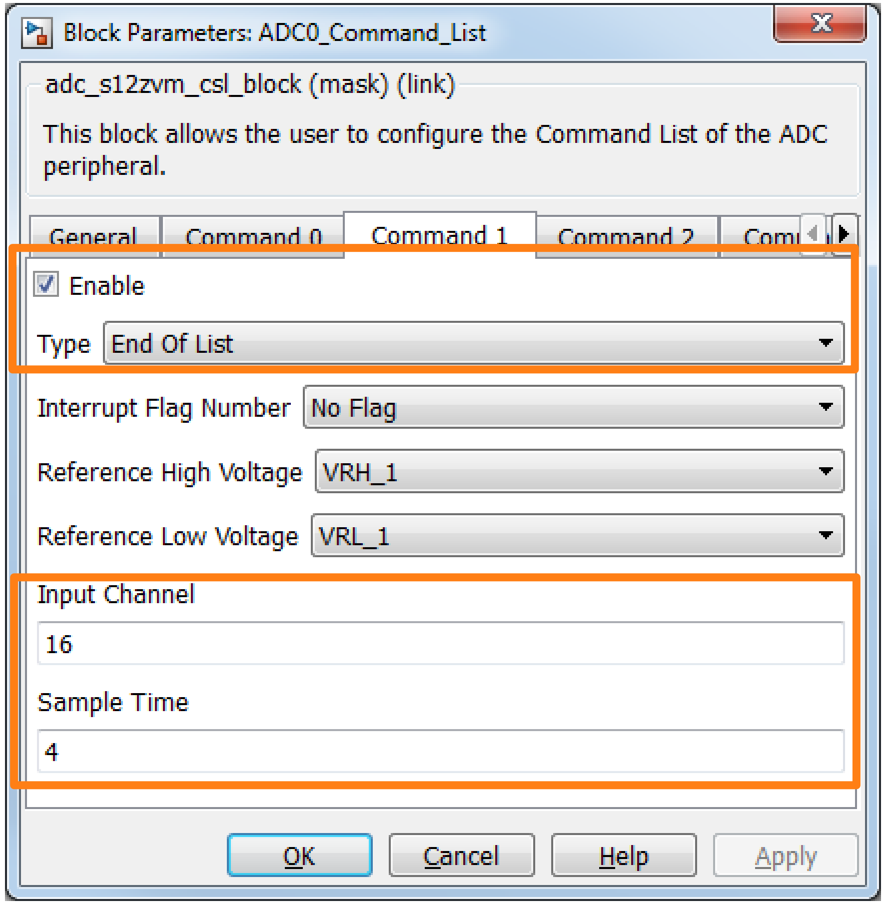
ADC Unit configuration steps: Copy ADC0_Config



Demo: Implement FOC Sensor-Less Motor Control

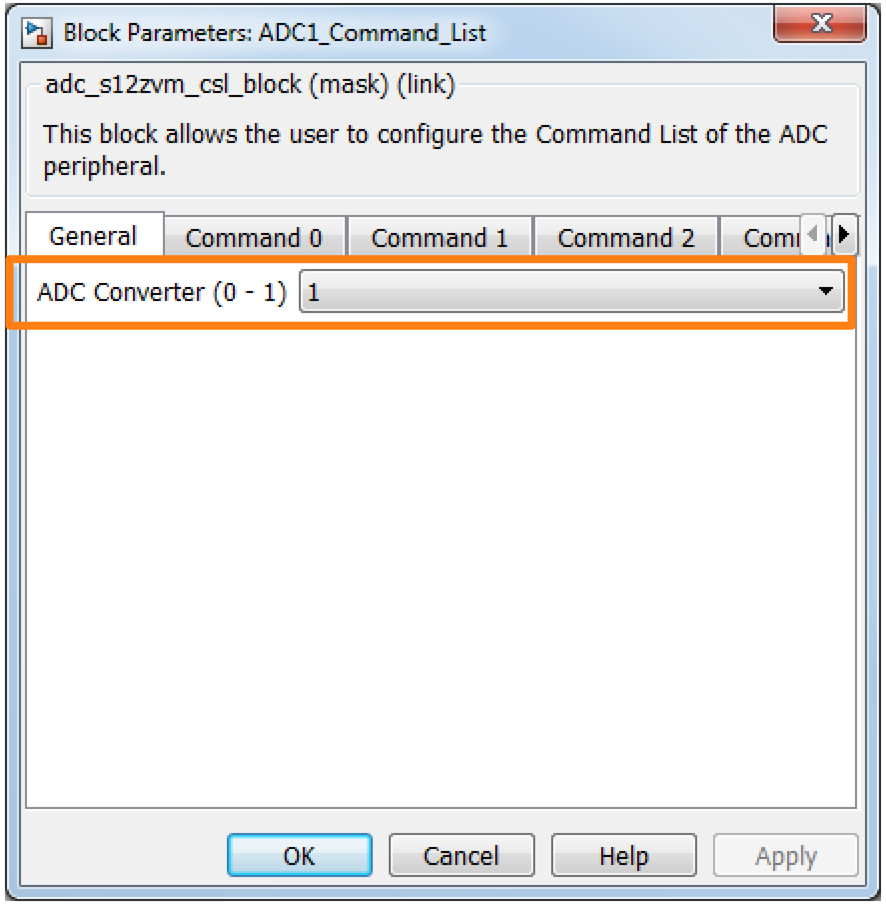
ADC Command List Block steps:



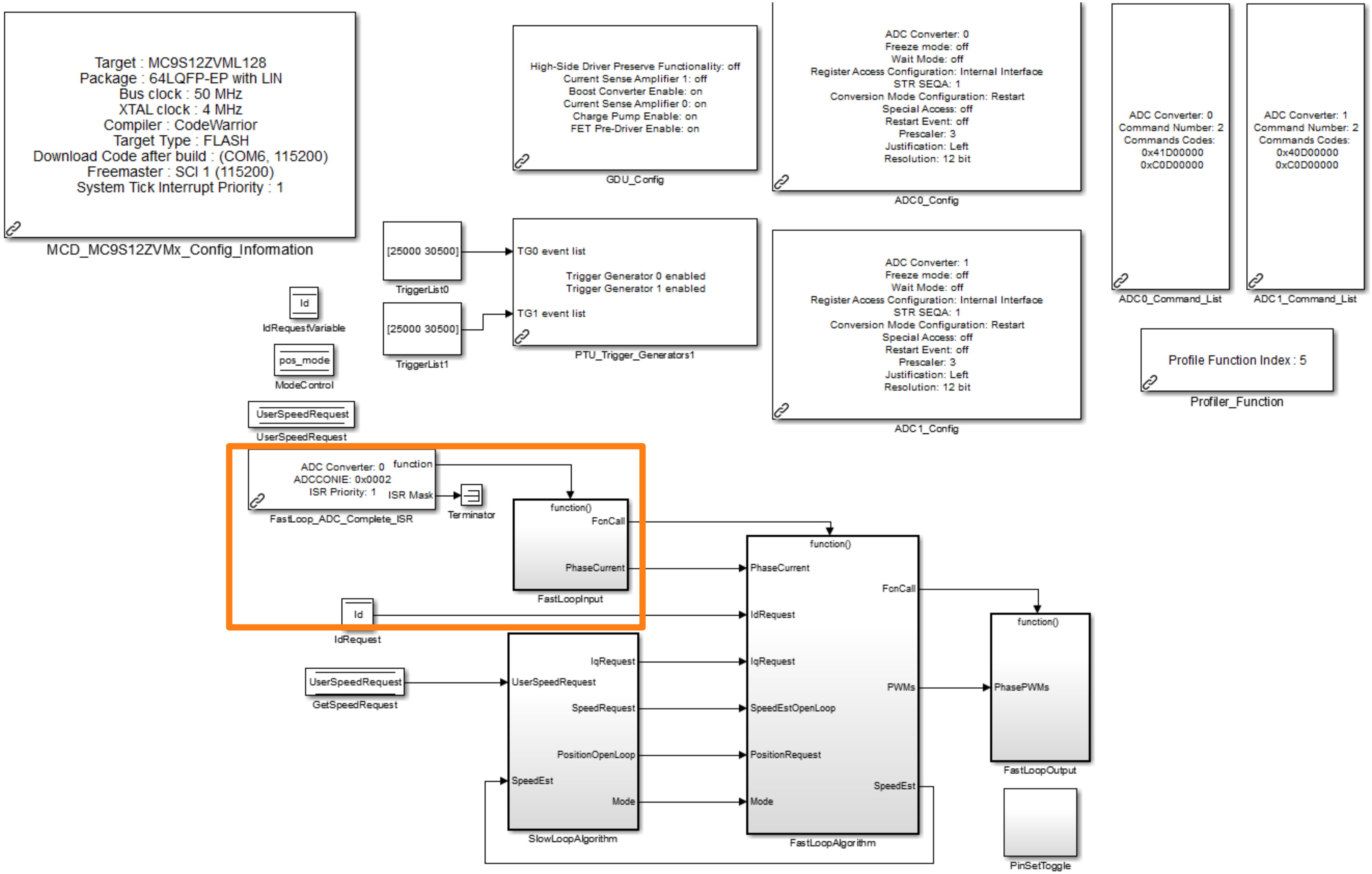


Demo: Implement FOC Sensor-Less Motor Control

ADC Command List Block steps: Copy ADC0_Command_List

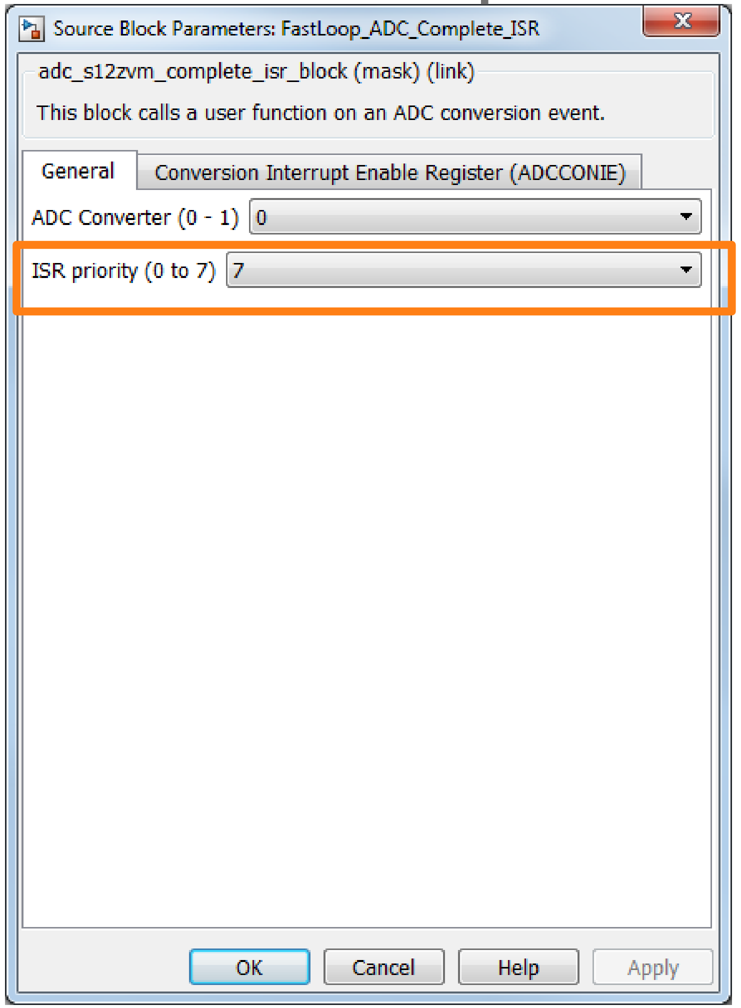


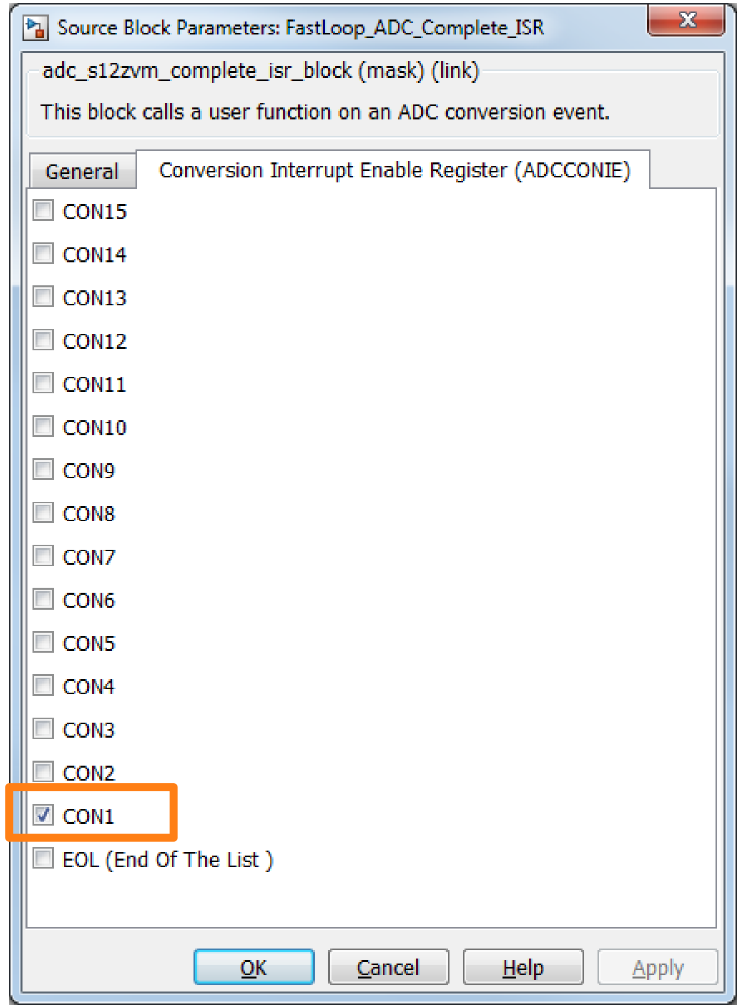
Demo: Implement FOC Sensor-Less Motor Control



Demo: Implement FOC Sensor-Less Motor Control

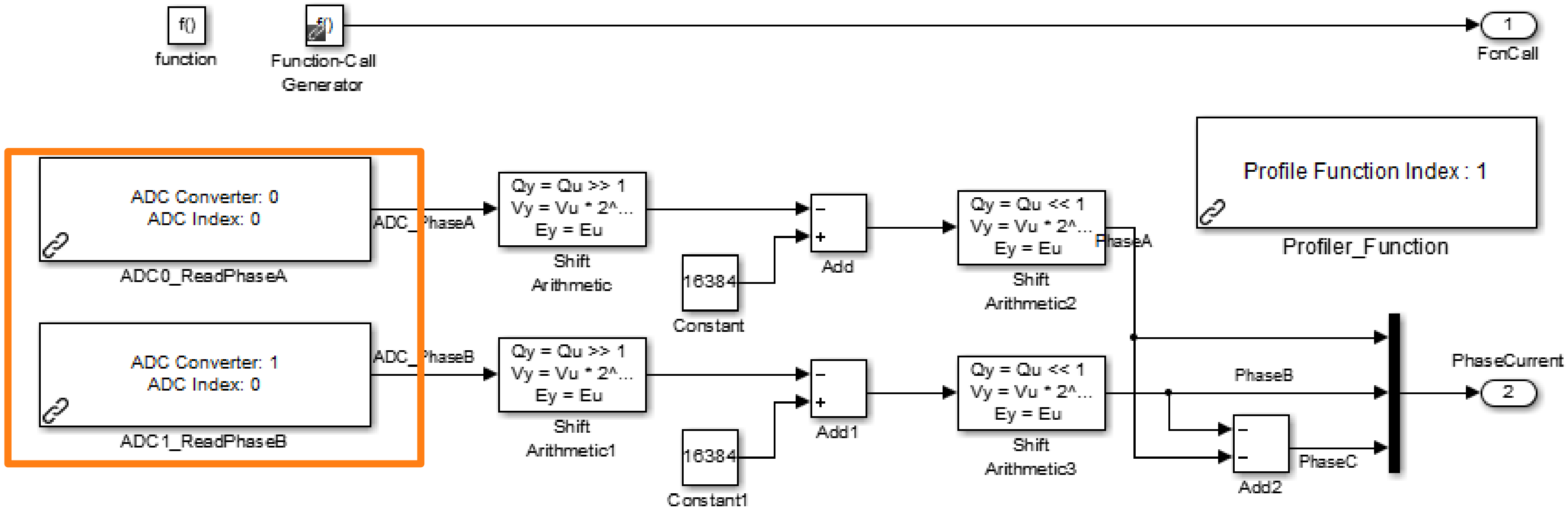
ADC ISR Block steps:





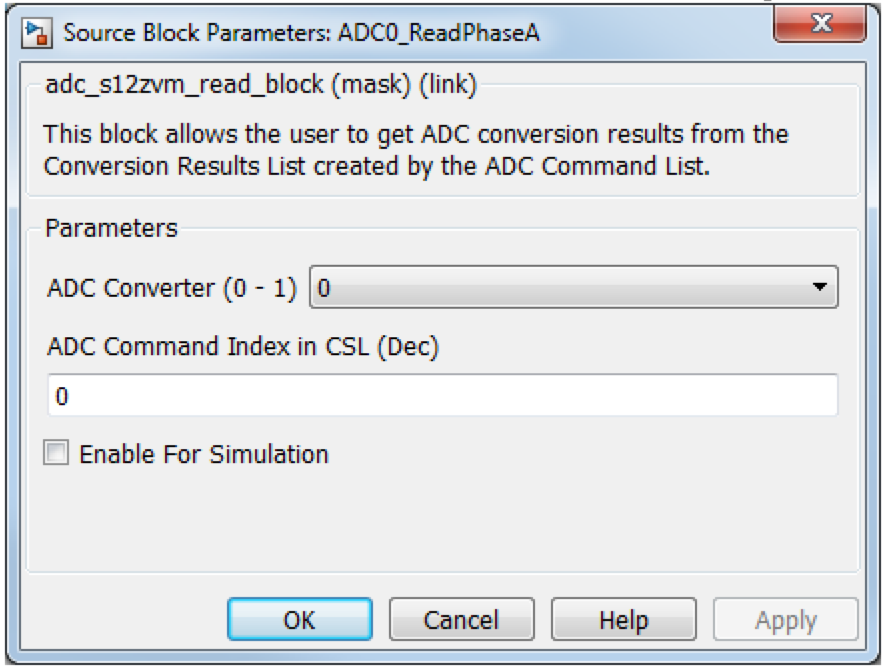
Demo: Implement FOC Sensor-Less Motor Control

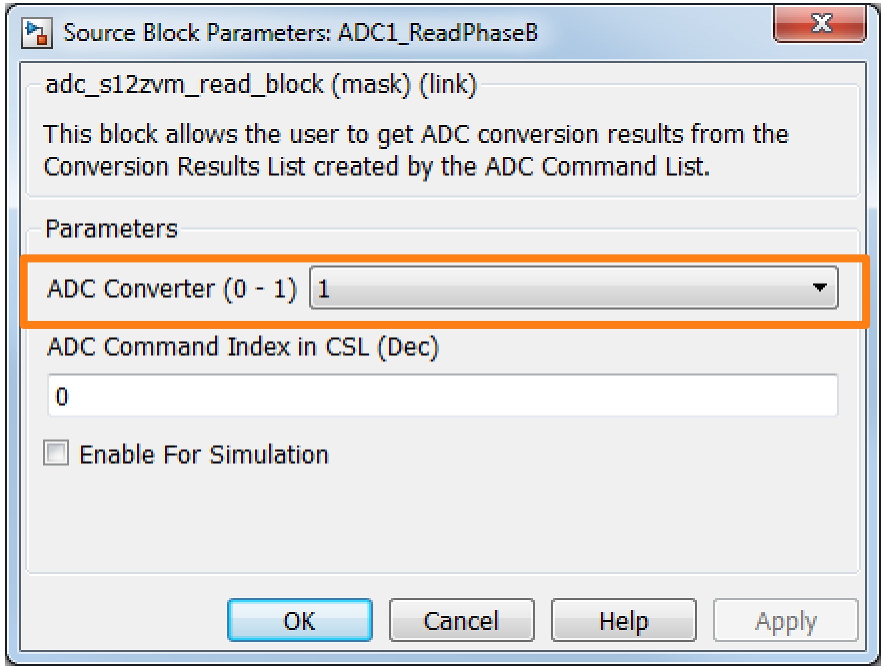
FastLoopInput



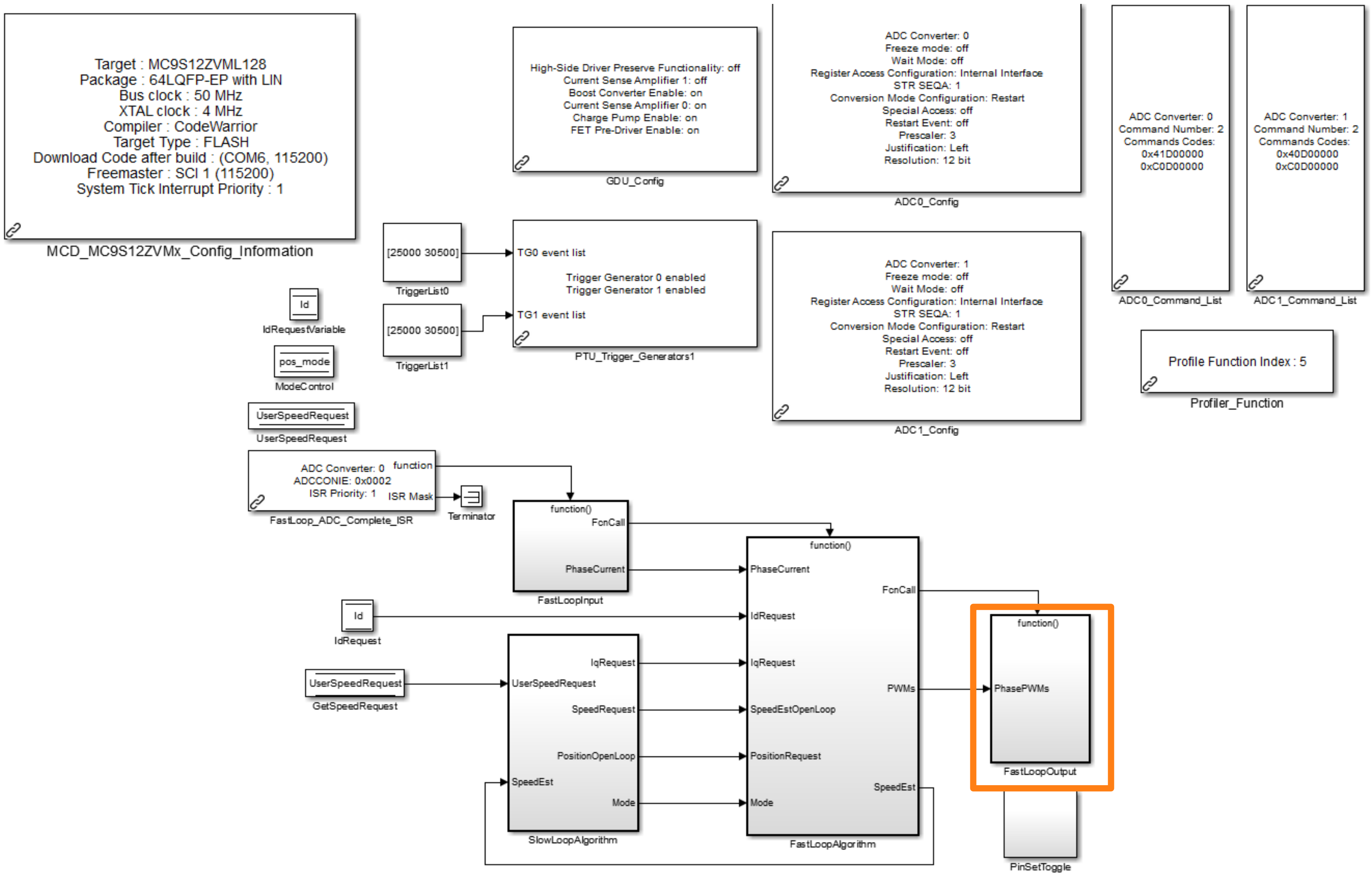
Demo: Implement FOC Sensor-Less Motor Control

ADC Command List Block steps:





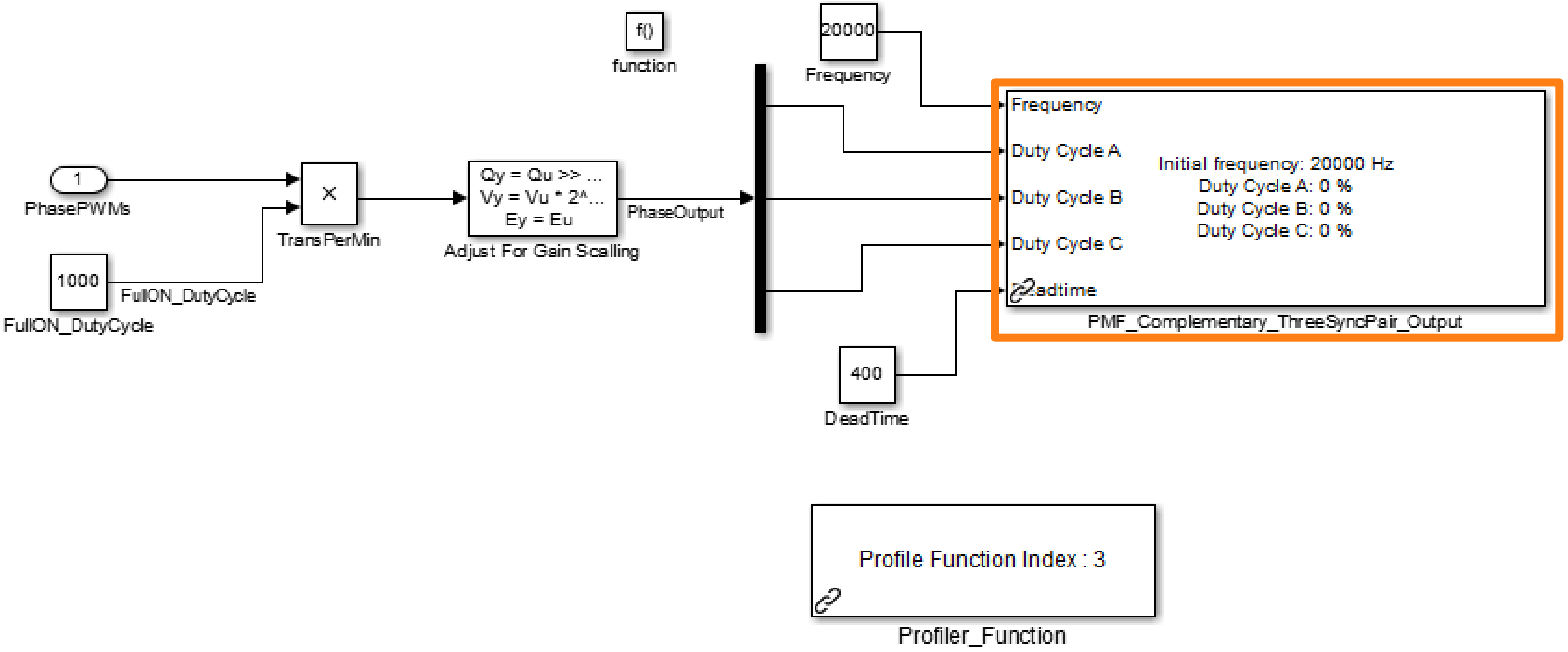
Demo: Implement FOC Sensor-Less Motor Control



Demo: Implement FOC Sensor-Less Motor Control

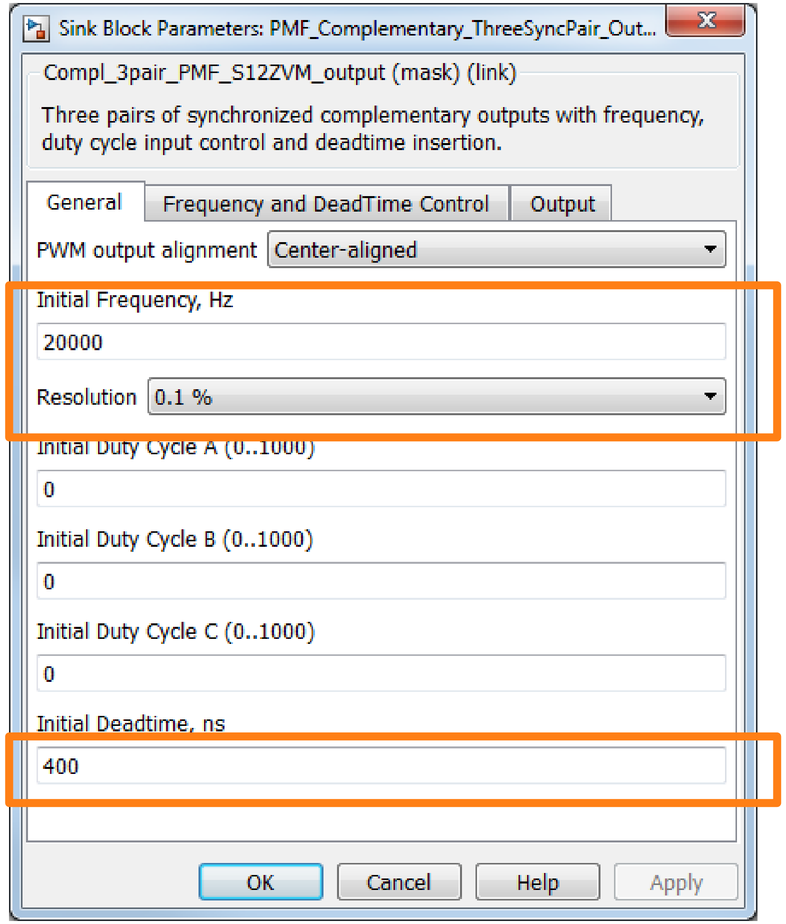
3PhaseDutyCycleOut Block with Flex PWM Blocks steps:

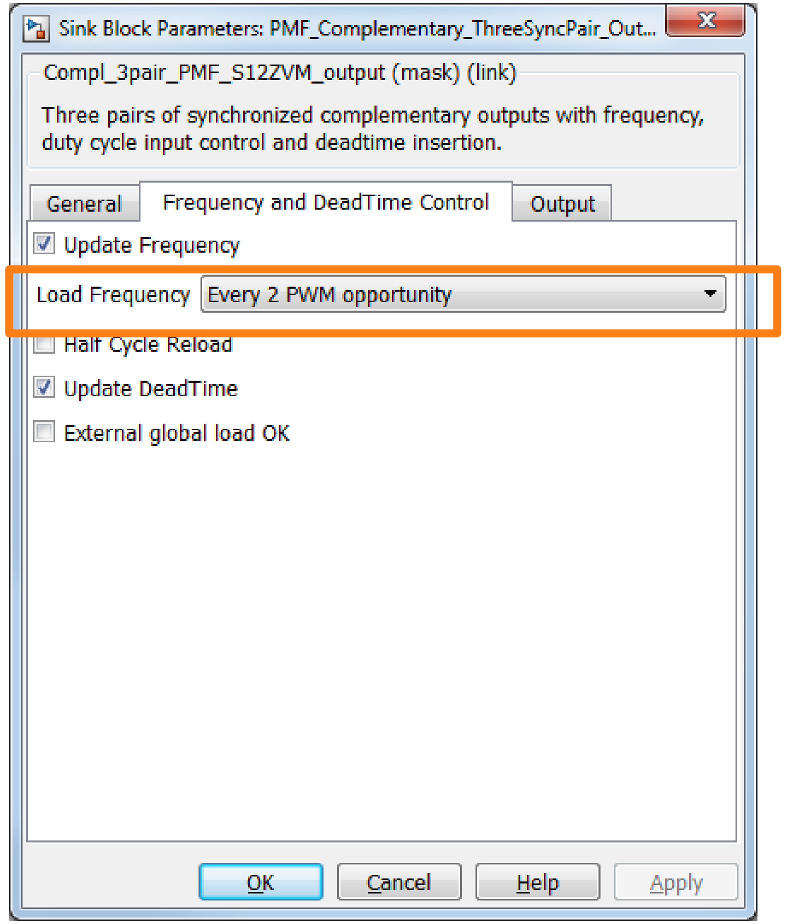
Pull Simple PWM phase block from library, connect to phase A and configure.



Demo: Implement FOC Sensor-Less Motor Control

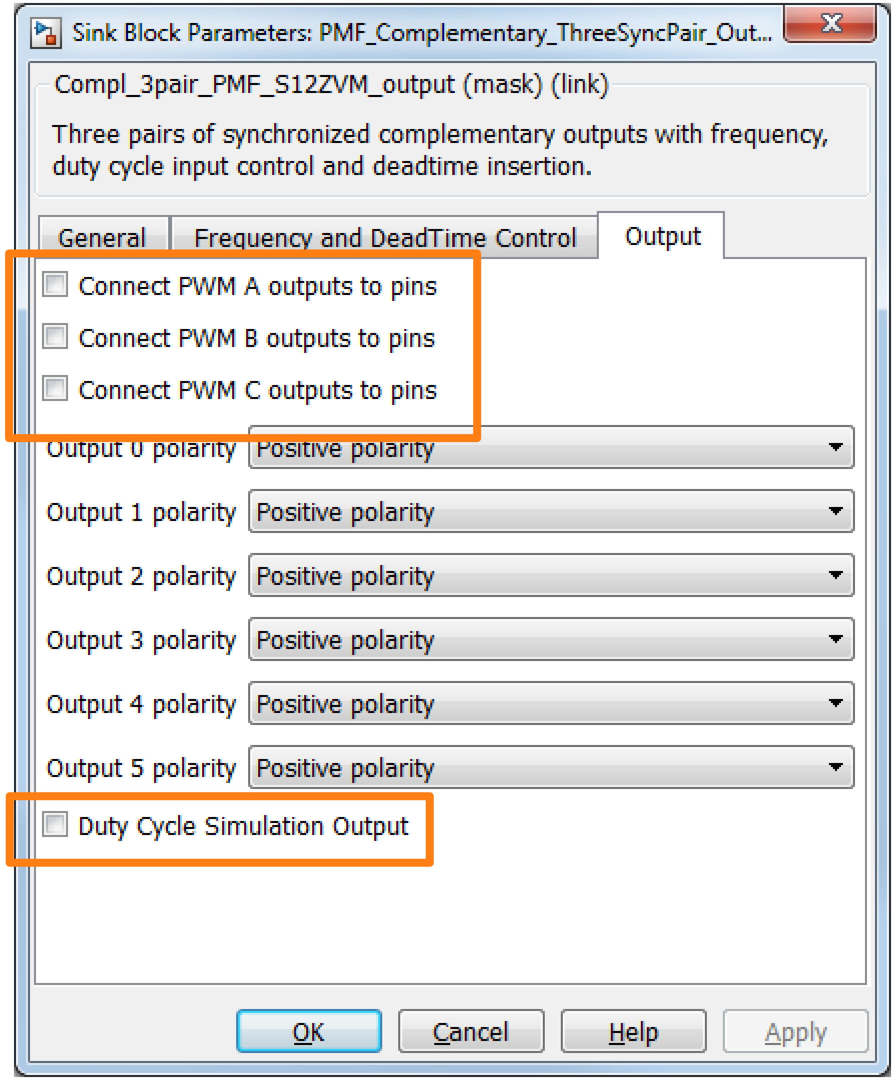
PMF Block steps:





Demo: Implement FOC Sensor-Less Motor Control

PMF Block steps:

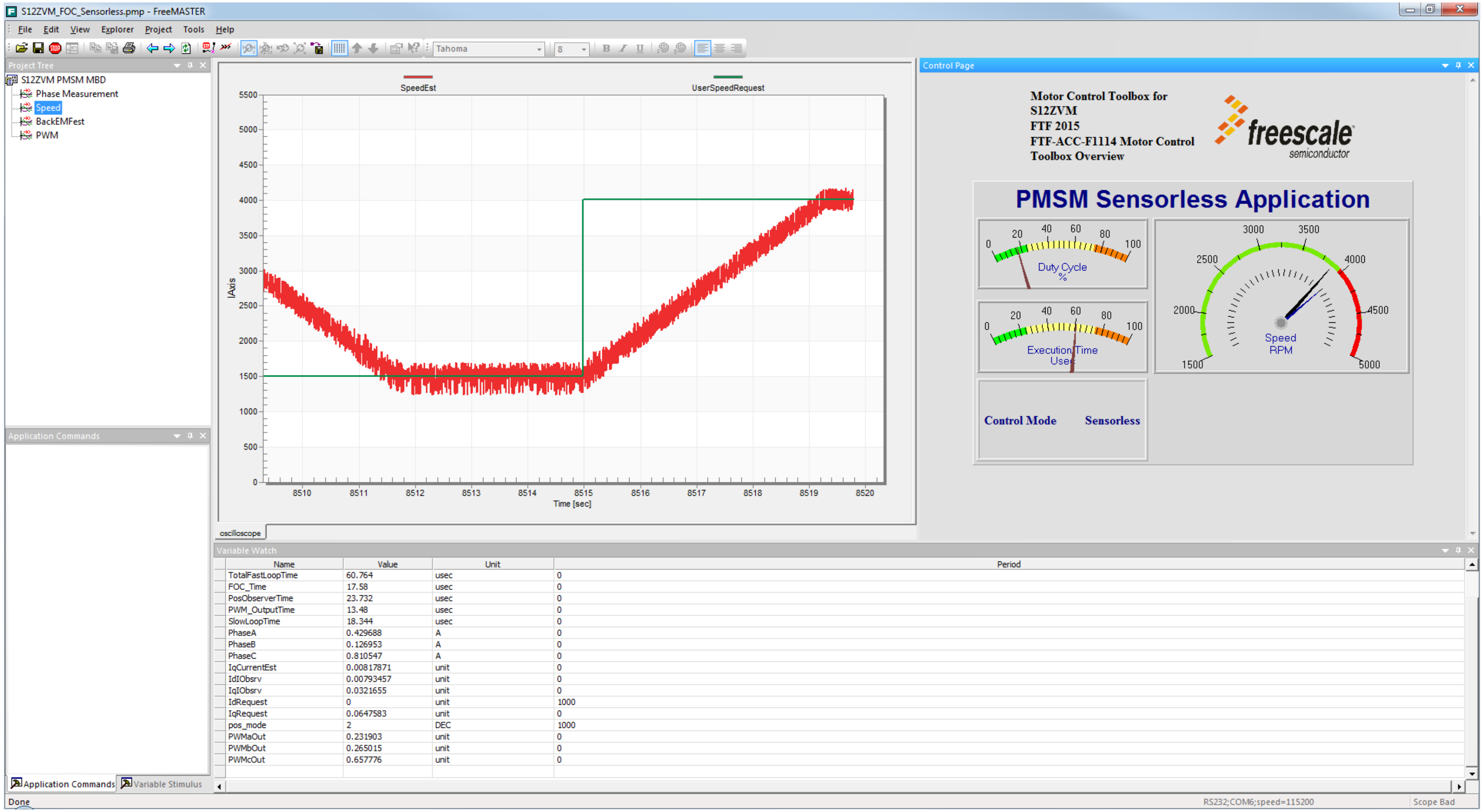


Demo: FreeMASTER to Monitor and Tune Parameters

Using FreeMASTER with Hands-On Demo

1. Start FreeMASTER and open project S12ZVM_FOC_Sensorless.pmp. Press OK if a message comes up that the map file has been updated.
2. Go to Project Options pull-down and select “Options”. Verify that COM settings are the same as what were set in your model.
3. Once the COM settings are correct, press the STOP button.
4. Change MotorSpeedReqFreemaster Variable to 1000 RPM.

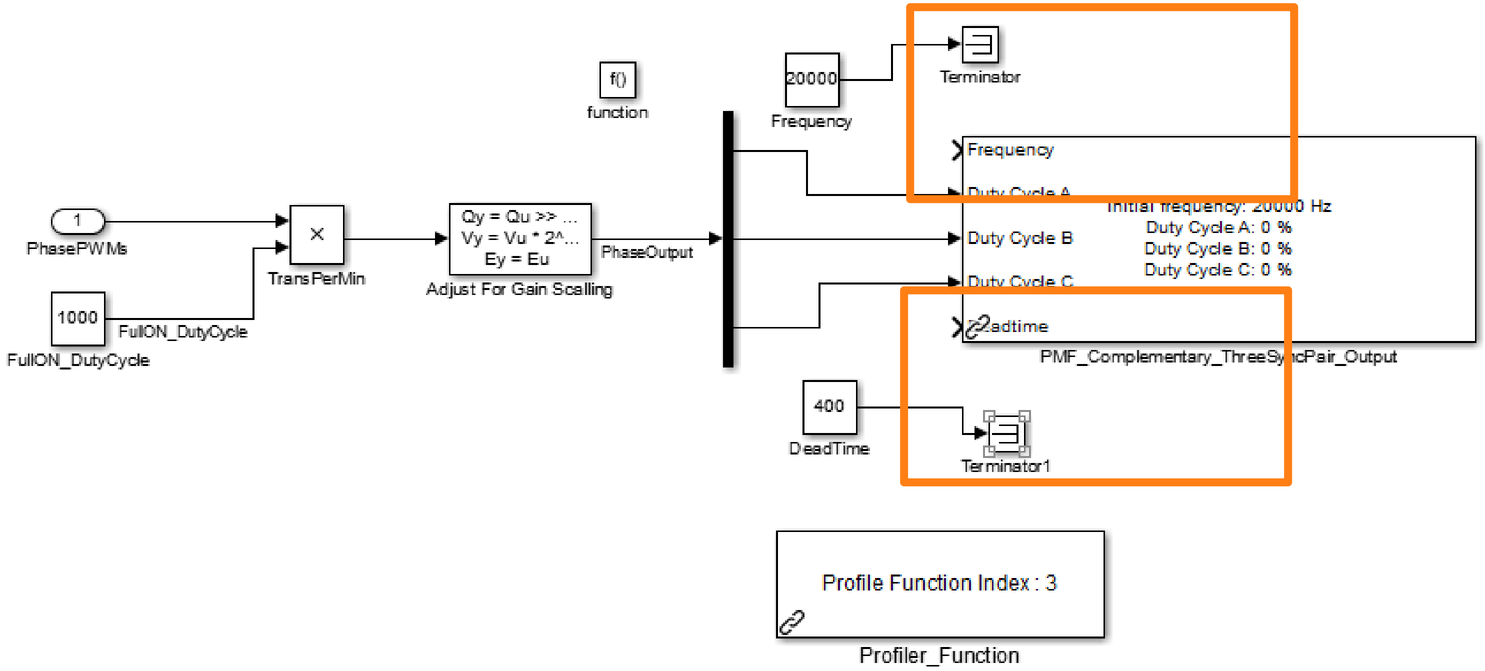
Demo: FreeMASTER to Monitor and Tune Parameters



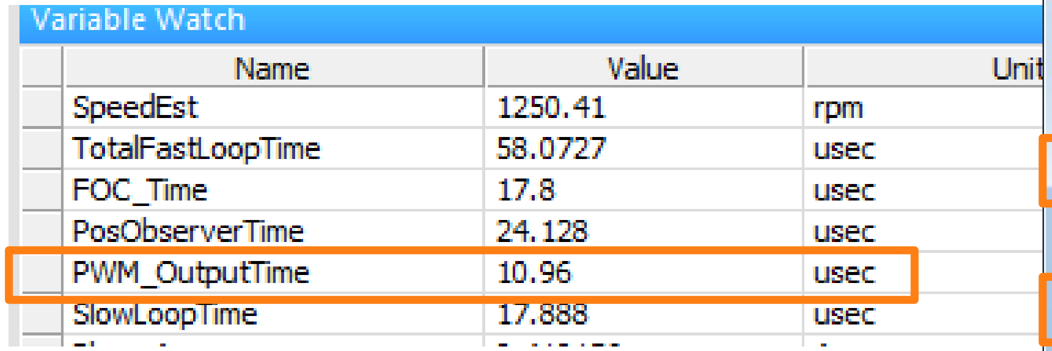
Demo: FreeMASTER to Monitor and Tune Parameters

Variable Watch

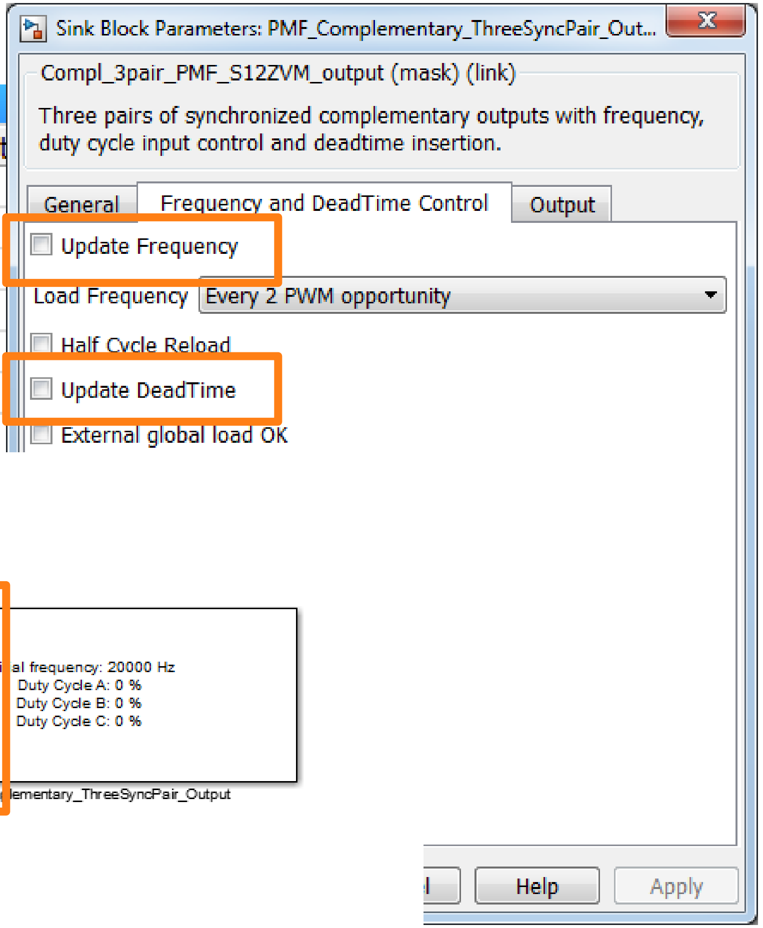
Name	Value	Unit	
SpeedEst	1695.72	rpm	0
TotalFastLoopTime	60.73	usec	0
FOC_Time	17.876	usec	0
PosObserverTime	23.892	usec	0
PWM_OutputTime	13.64	usec	0
SlowLoopTime	17.828	usec	0

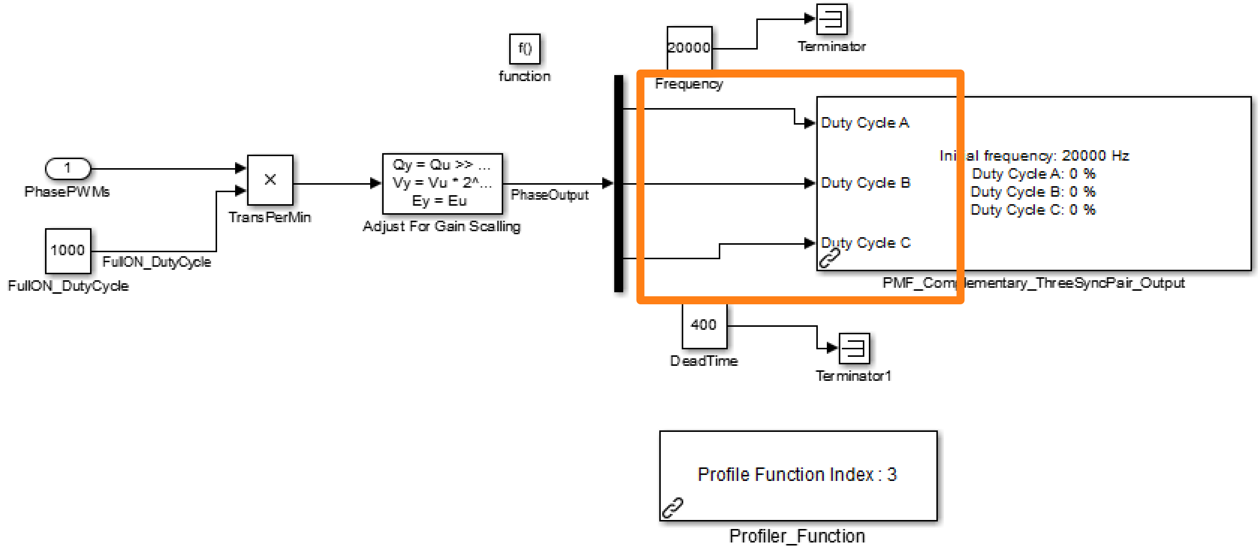


Demo: Implement FOC Sensor-Less Motor Control


Variable Watch

Name	Value	Unit
SpeedEst	1250.41	rpm
TotalFastLoopTime	58.0727	usec
FOC_Time	17.8	usec
PosObserverTime	24.128	usec
PWM_OutputTime	10.96	usec
SlowLoopTime	17.888	usec





SECURE CONNECTIONS
FOR A SMARTER WORLD