MathWorks®

MATLAB EXPO 2017

NXP Motor Control Toolbox For MATLAB/Simulink Modeling and Code Generation

Mike Cao , NXP Automotive Senior FAE

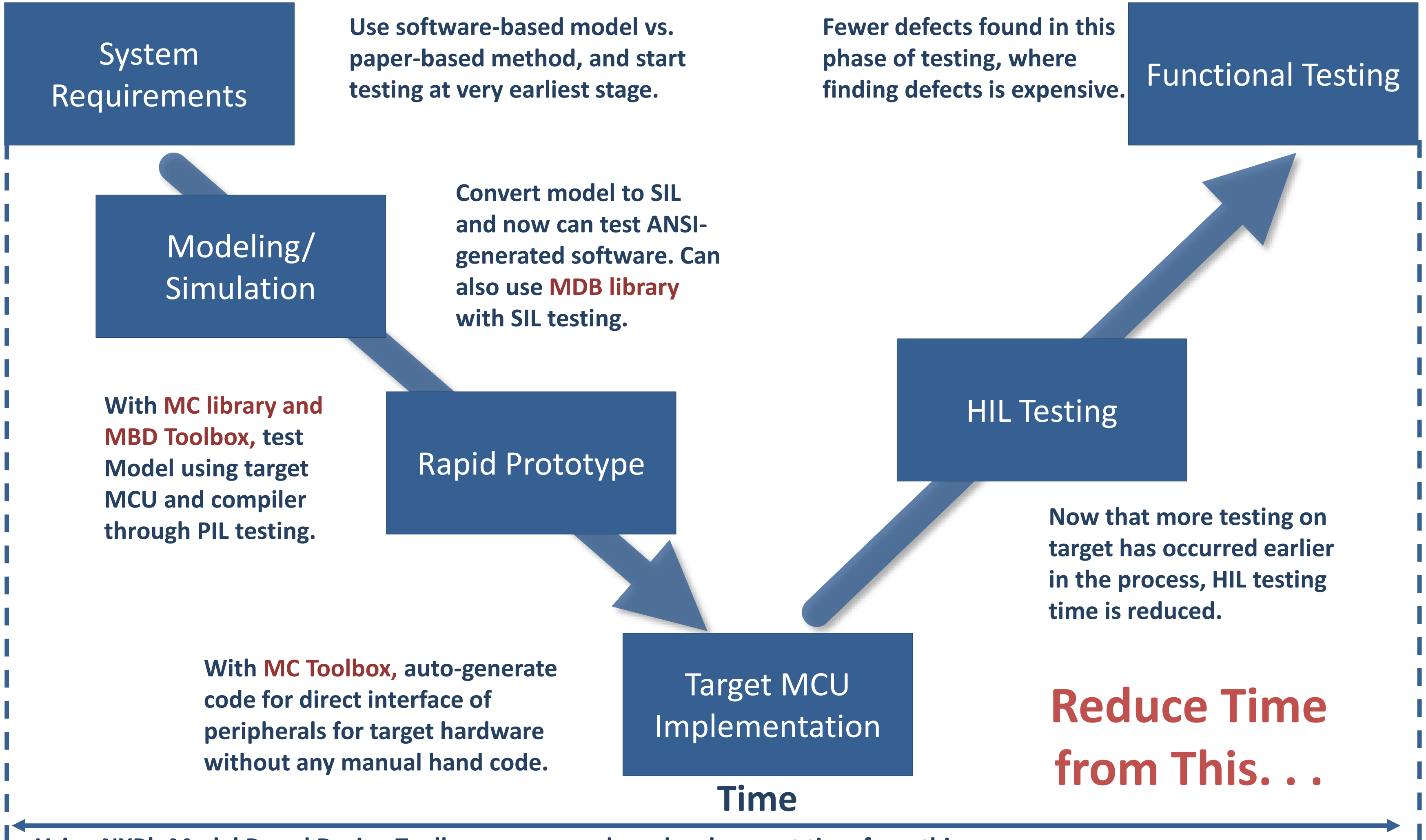
Introduction: Model Based Design (MBD)

- Model Based Design is becoming more common during the normal course of software development to explain and implement the desired behavior of a 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.

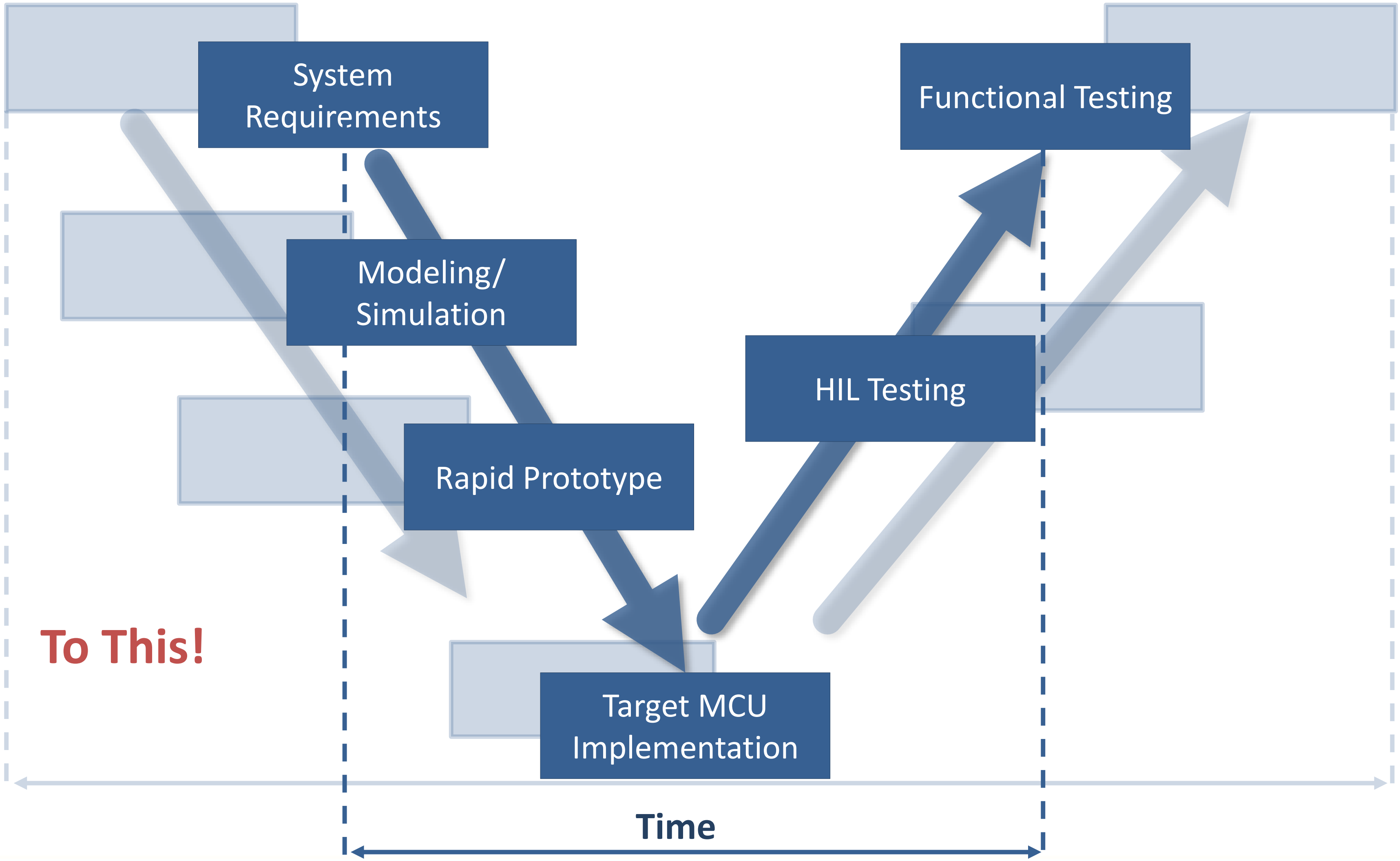
- The final stage of this type of development is the integration of the control algorithm software with target MCU hardware. This is often done using hand code or a mix of hand code and model-generated code. NXP's Model Based Design Toolbox allows this stage of the development to generate 100% of the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

Introduction: Reduce Development Time With MBD Toolbox



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

Introduction: Reduce Development Time With MBD Toolbox



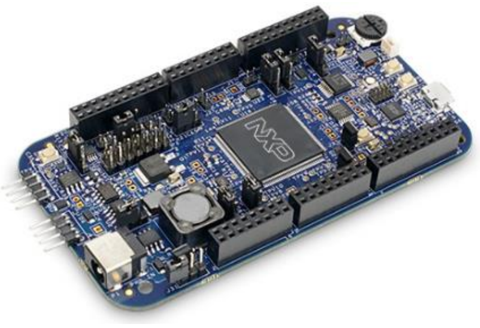
Introduction: What Do We Do?

- One of the Automotive Tools Enablement & Engineering group' s objectives is to develop software enablement tools to assist our customers with rapid prototyping and accelerate algorithm development on their target NXP MCU

- This includes software tools that automatically generate peripheral initialization code through GUI configuration, to generating peripheral driver code from a Model Based Design environment like Simulink™

Exposure to NXP' s hardware/software enablement

NXP DEVKIT-MPC5744P



NXP S32K EVB Kit

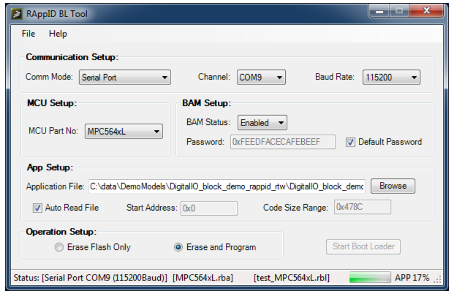
Model Based DesingToolbox
with Simulink ™

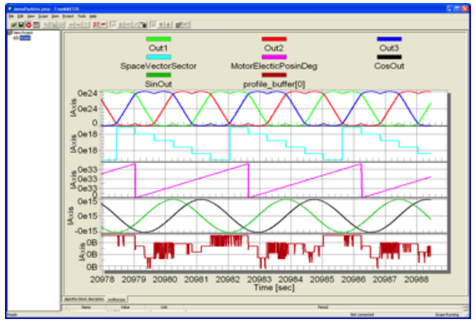
MATLAB
SIMULINK

Model-based design
Driver configuration
Assignment to pins
Initialization setup

MTRCKTSBNZVM128

RAppID Bootloader Utility





FREEMASTER

Signal Visualization
and Data Acquisition Tool

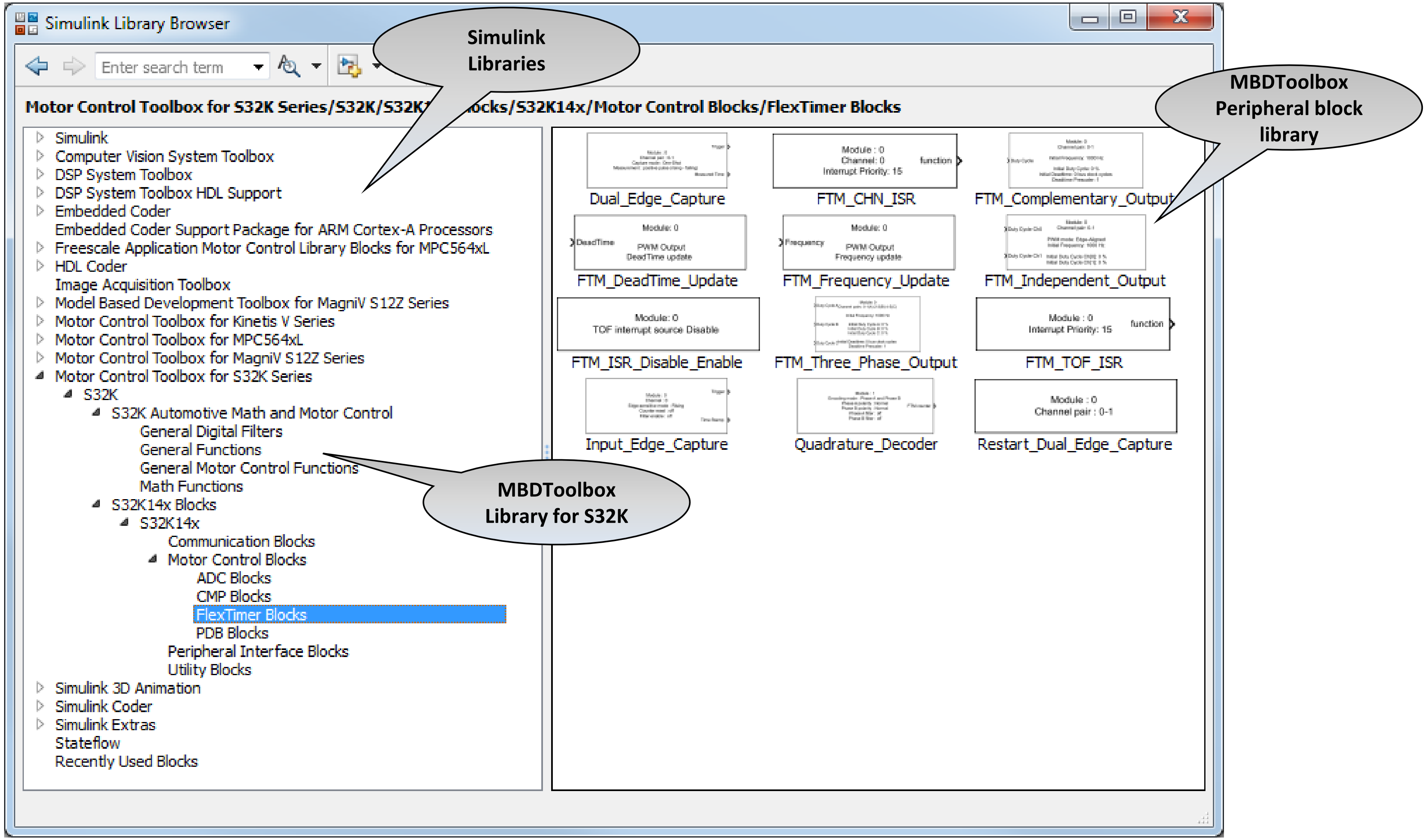
Model Based Design Toolbox Overview

- The Model Based Design Toolbox includes an embedded target supporting NXP MCUs and 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 Automotive 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

MBD Toolbox: Toolbox Library Contents



MBD Toolbox: Toolbox Library Contents

Peripherals

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

Configuration/Modes

- Compiler Options
  - 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

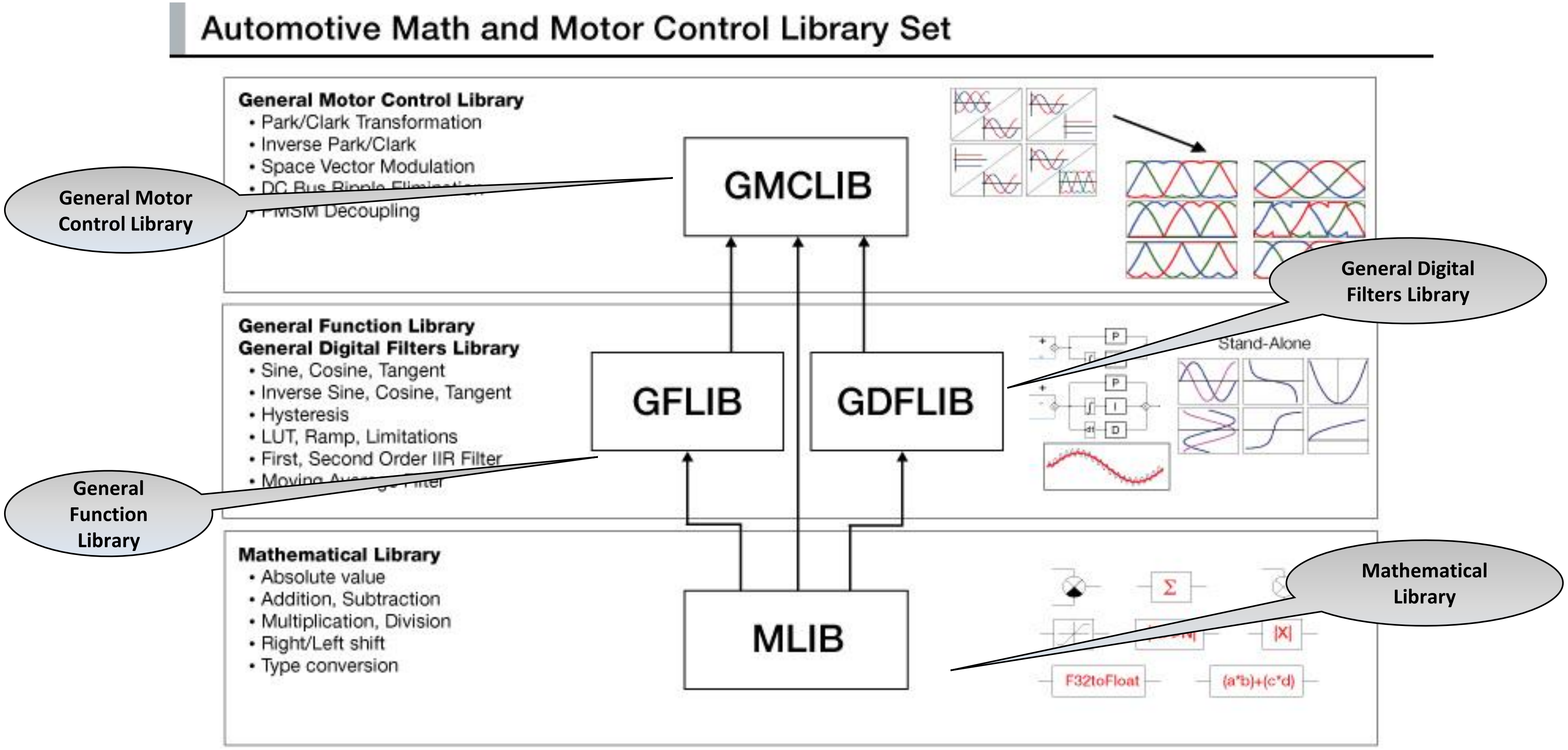
Utility

- FreeMASTER Interface
  - Data acquisition / Calibration
  - Customize GUI
- Profiler Function
  - Exec. time measurement
  - Available in PIL
  - Available in standalone
- Memory Read and Write

MCUs Supported

- **MPC5643L**
- **MPC567xK**
- **MPC574xP**
- **S12ZVM**
- **S32K**

MBD Toolbox: Auto Math and Motor Control Library Contents



MBD Toolbox: Auto Math and Motor Control Library Contents

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_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

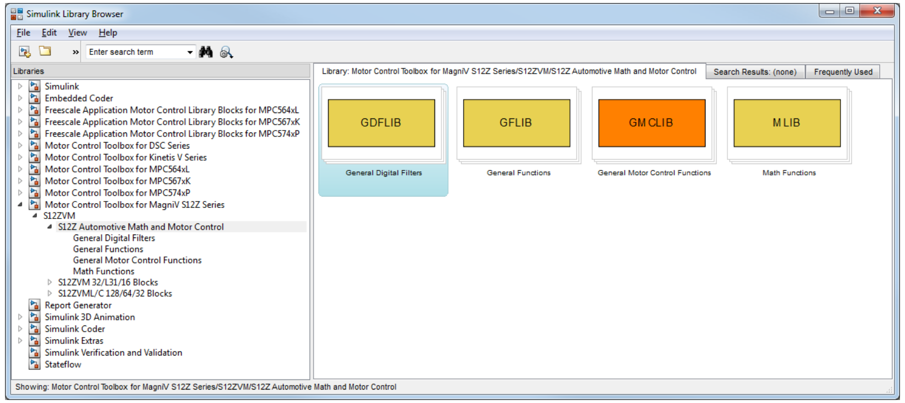
- **Trigonometric Functions**
  - GFLIB_Sin, GFLIB_Cos, GFLIB_Tan
  - GFLIB_Asin, GFLIB_Acos, GFLIB_Atan, GFLIB_AtanYX
  - GFLIB_AtanYXShifted
- **Limitation Functions**
  - GFLIB_Limit, GFLIB_VectorLimit
  - GFLIB_LowerLimit, GFLIB_UpperLimit
- **PI Controller Functions**
  - GFLIB_ControllerPIr, GFLIB_ControllerPIrAW
  - 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



MBD Toolbox: RAppID Bootloader Utility

The RAppID Bootloader works with the built-in Boot Assist Module (BAM) included in the NXP Qorivva and also supports S12 MagniV, Kinetis, and DSCs family of parts. 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.

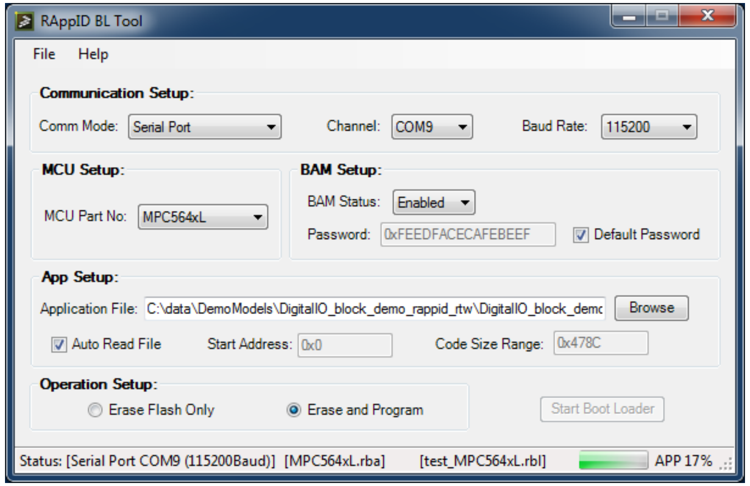
Modes of Operation

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, MPC564xL, MPC5604/3P, MPC574xP, S12ZVM, S32K, KV10, KV3x, KV4x, KV5x, 56F82xx and 56F84xx.

Graphical User Interface



Command

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

FreeMASTER — Run Time Debugging Tool

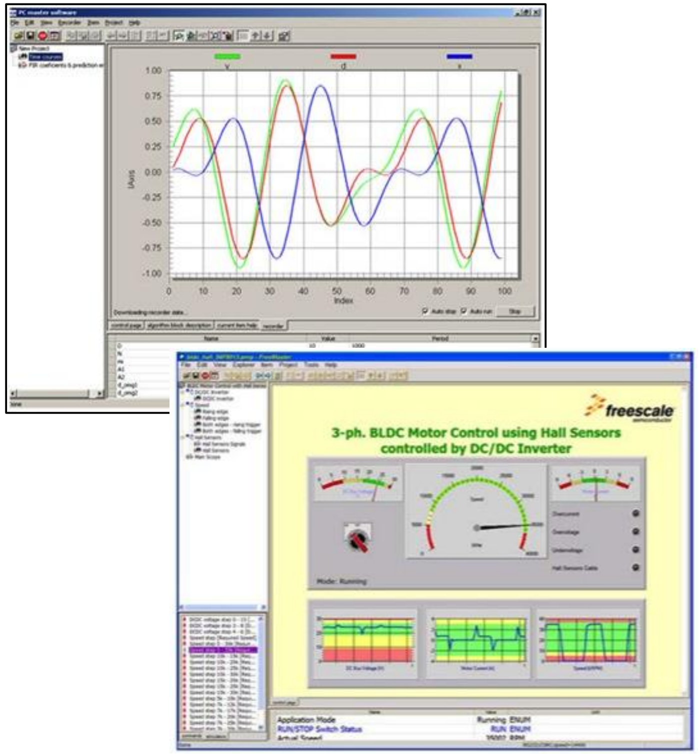
User-friendly tool for real-time debug monitor and data visualization

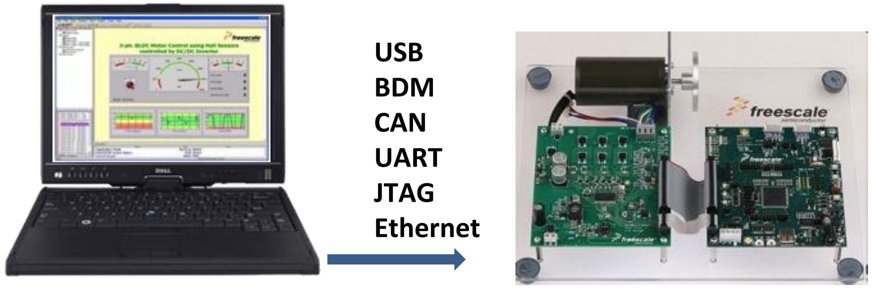
- Completely non-intrusive monitoring of variables on a running system
- Display multiple variables changing over time on an oscilloscope-like display, or view the data in text form
- Communicates with an on-target driver via USB, BDM, CAN, UART

Establish a Data Trace on Target

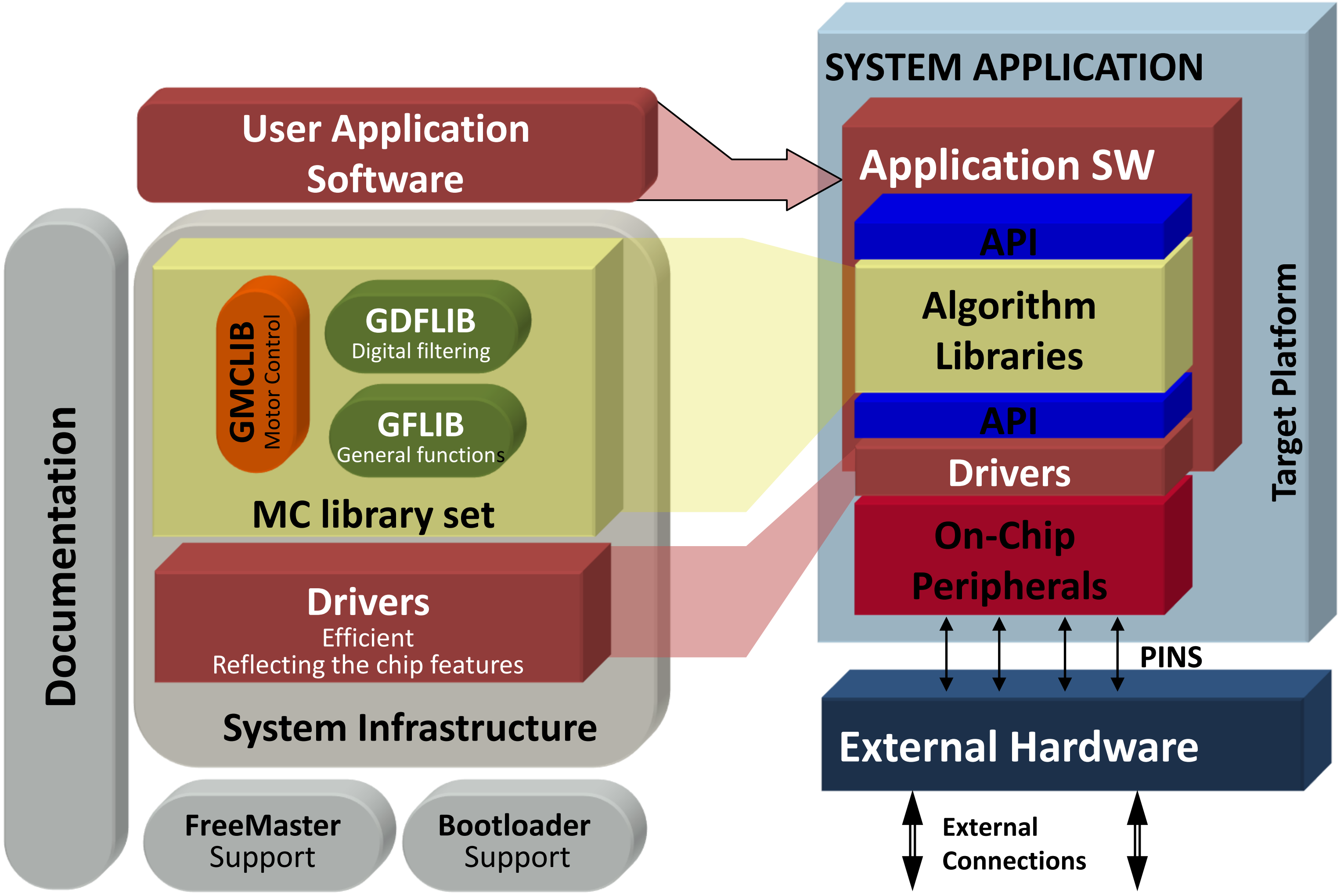
- Set up buffer (up to 64 KB), sampling rate and trigger
- Near 10-µs resolution

http://www.nxp.com/freemaster

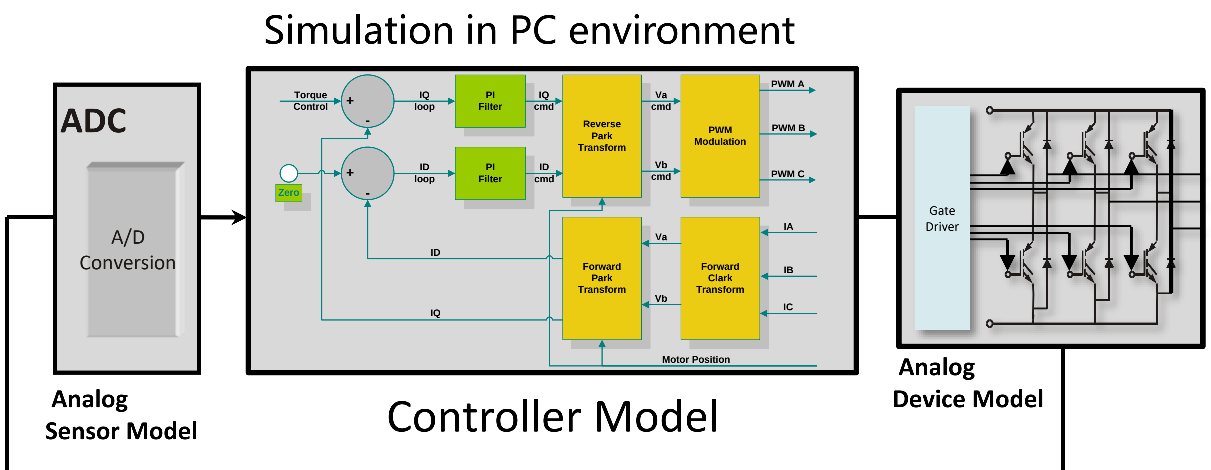


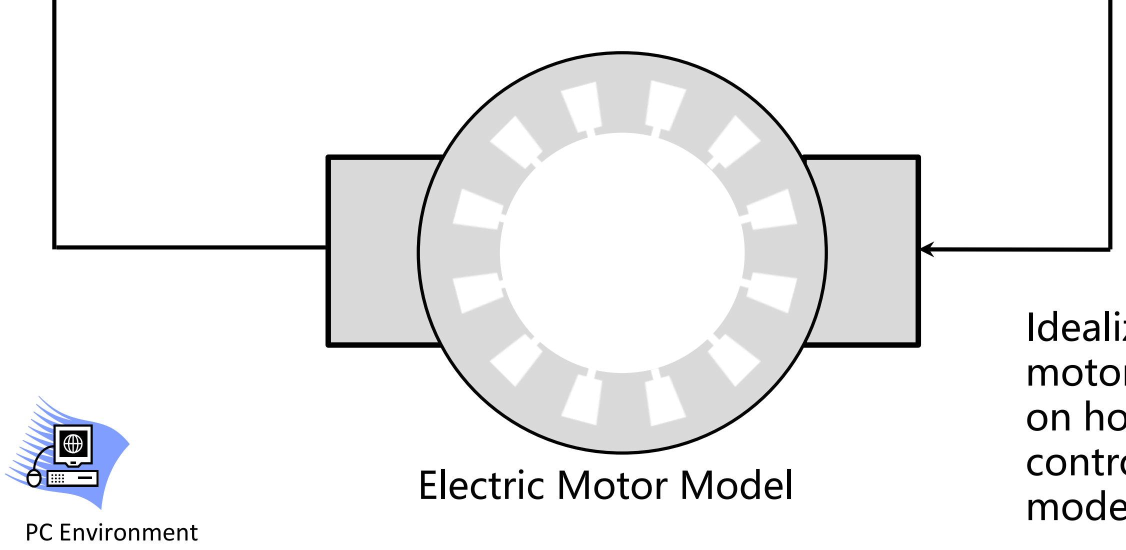


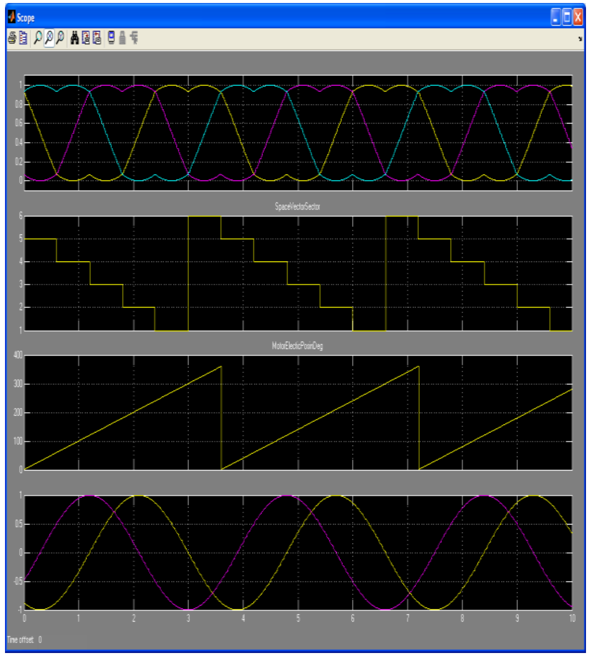
MBD Toolbox: Summary of Application Support



Model Based Design Steps: Step 1 (Simulation)



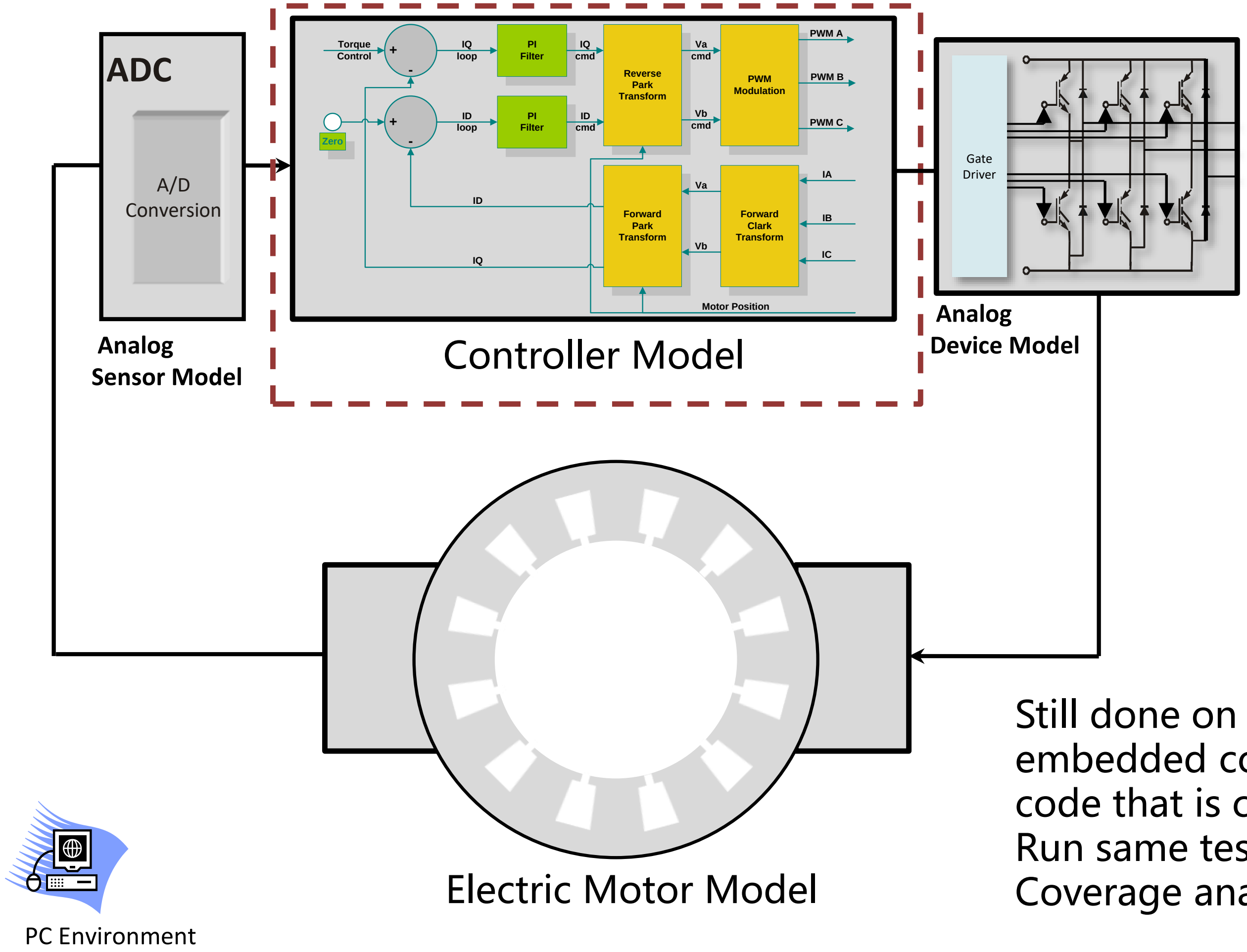


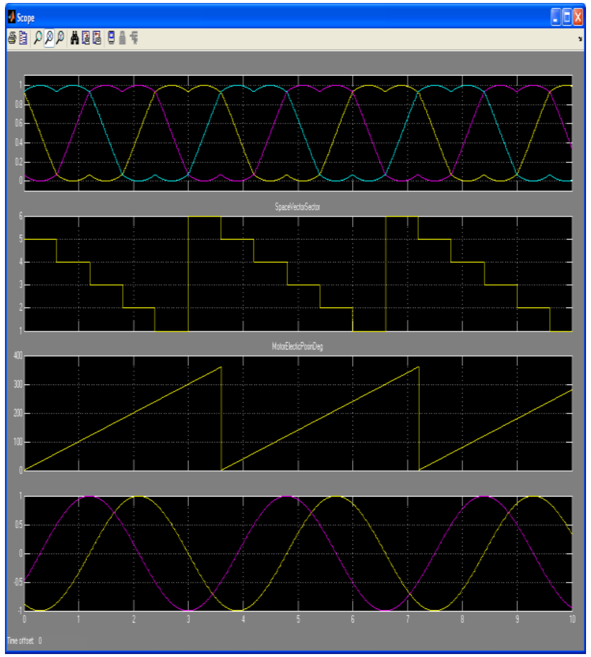


Idealized simulation of the controller and the motor to refine the control technique. Done on host PC without regard for embedded controller. Can optionally add analog device models for fault detection and signal control.

Model Based Design Steps: Step 2 (SIL)

(SIL) Generated code executes as atomic unit on PC

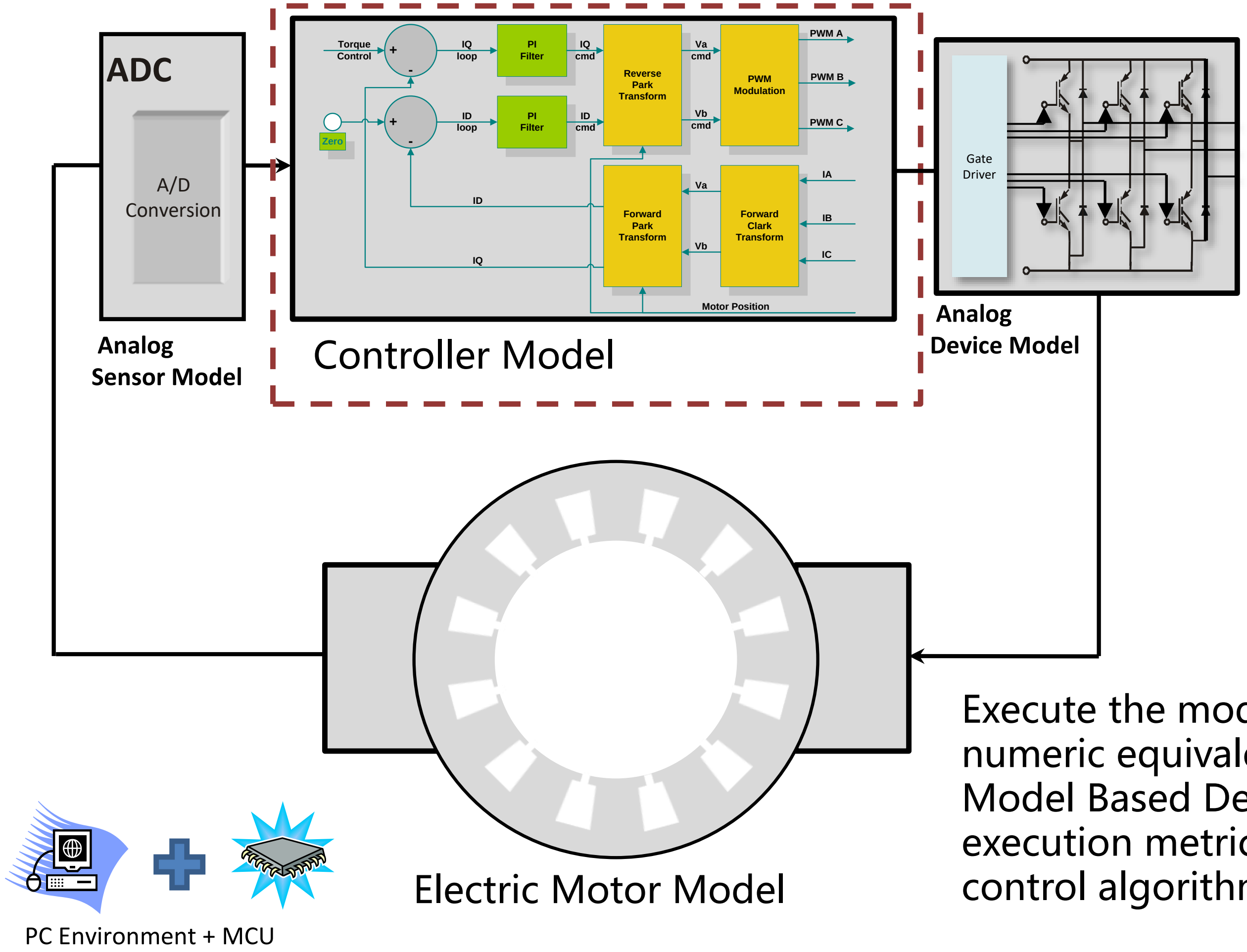


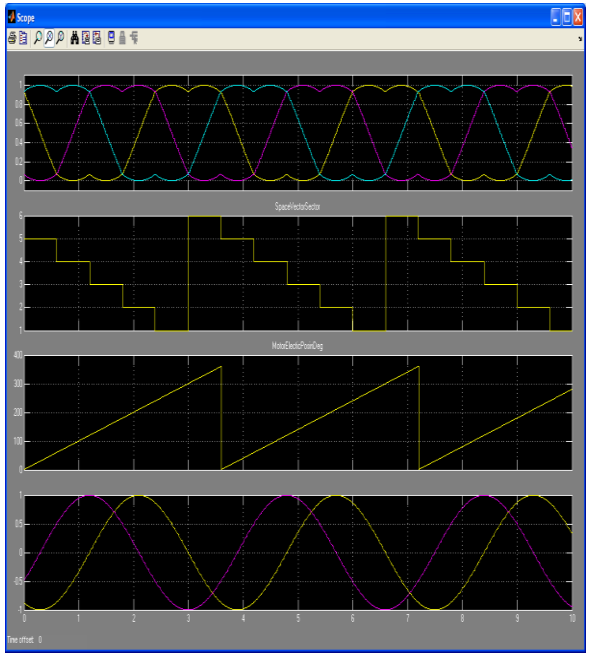


Still done on host PC without regard for embedded controller. Instead using generated C code that is compiled using a PC-based compiler. Run same test vectors as in simulation for C Code Coverage analysis and verify functionality.

Model Based Design Steps: Step 3 (PIL)

(PIL) Executes generated code on the target MCU





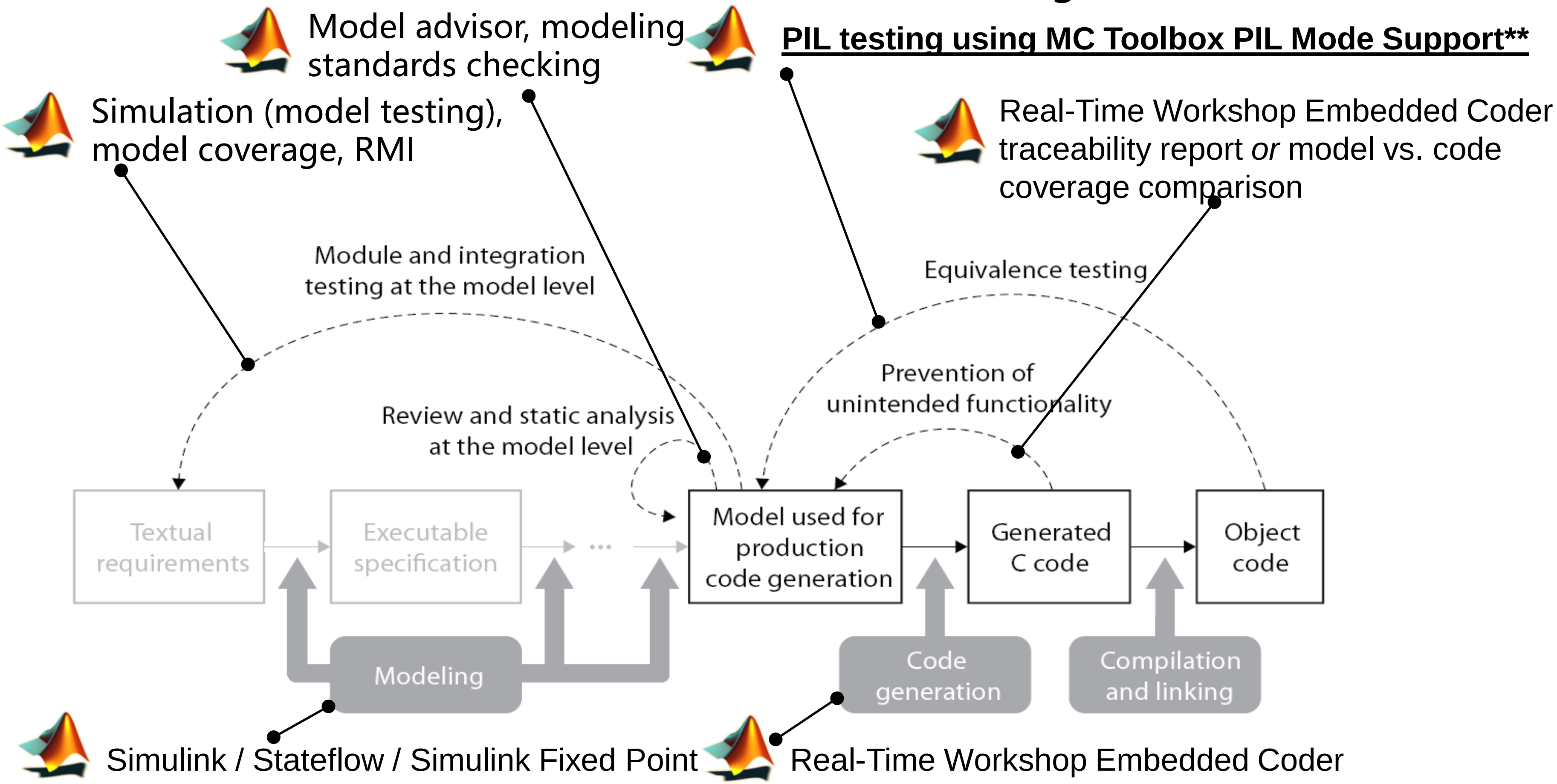
Execute the model on the target MCU and perform numeric equivalence testing. Co-execution with MCU and Model Based Design working together while collecting execution metrics on the embedded controller of the control algorithm. Validate performance on the MCU.

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

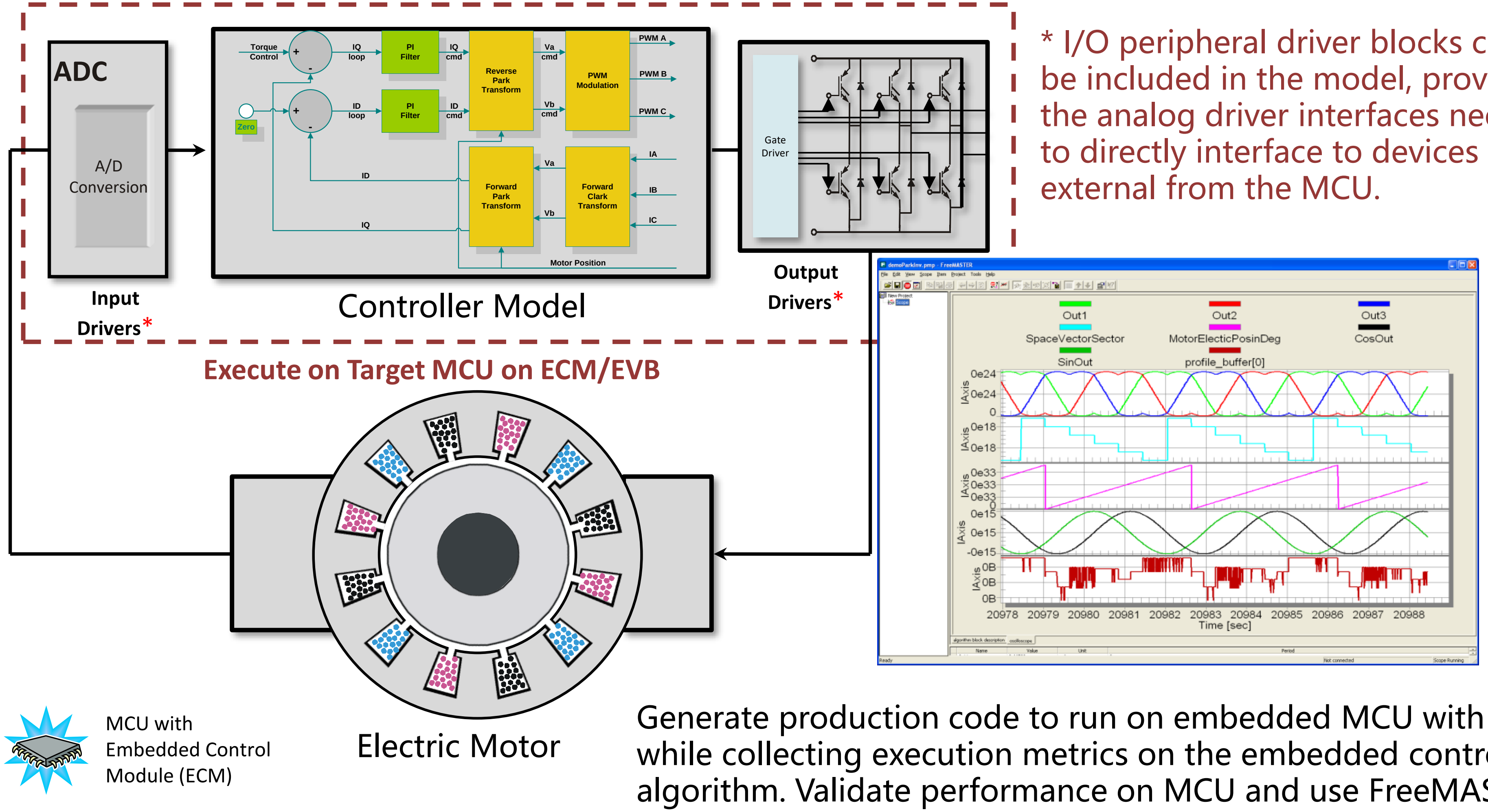
Example IEC 61508 and ISO 26262 Workflow for Model-Based Design with MathWorks Products*



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

** NXP MC 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.

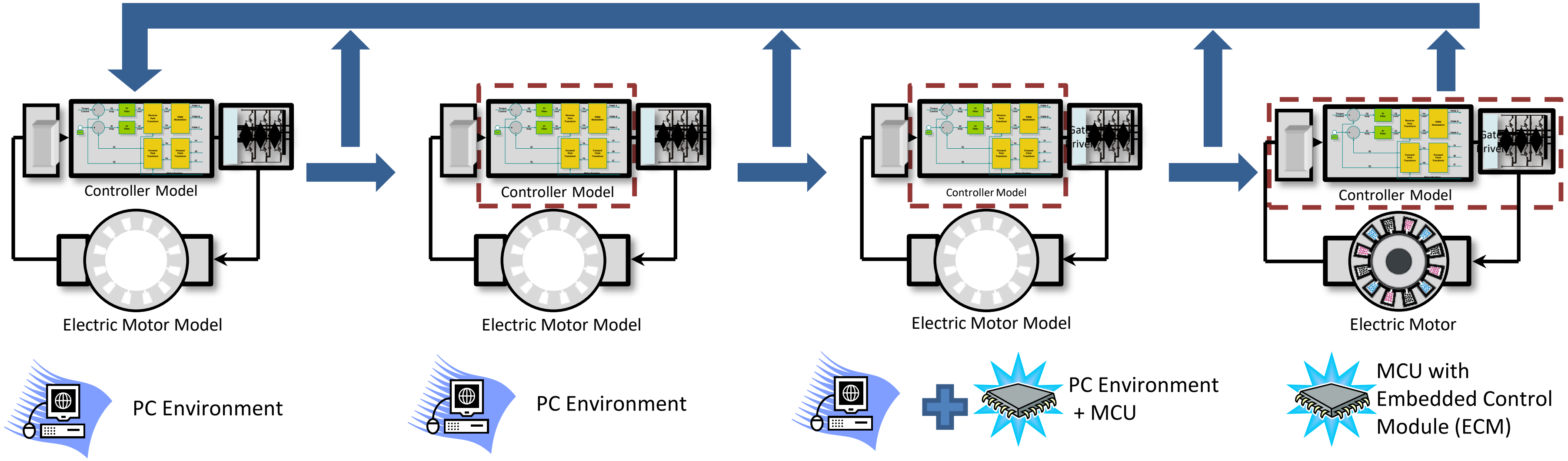
Model Based Design Steps: Step 4 (Target MCU)*



* 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.

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 Steps: Summary



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

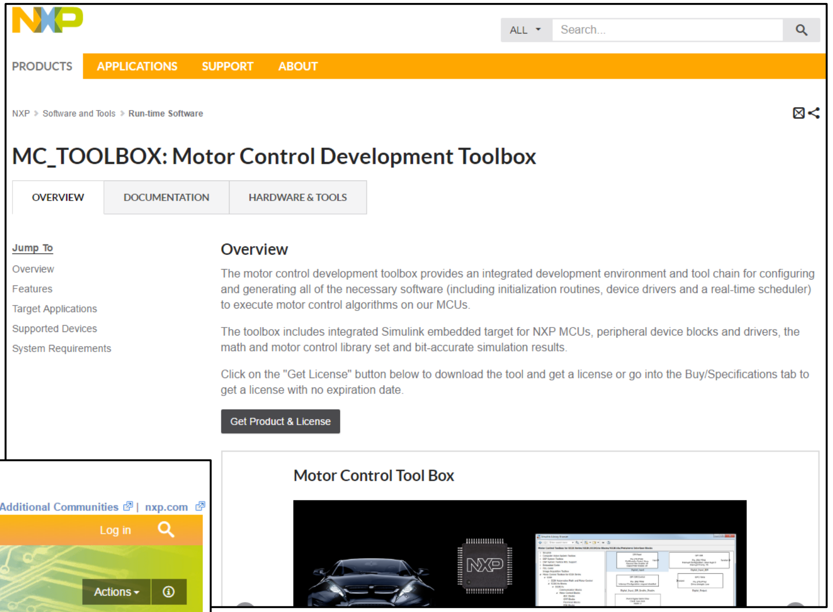
Refine model for code generation
Function/File partitioning
Data typing to target environment done here
Scaling for fixed point simulation and code gen
Testing of functional components of algorithm
Test harness to validate all requirements
Test coverage of model here
Creates functional baseline of model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

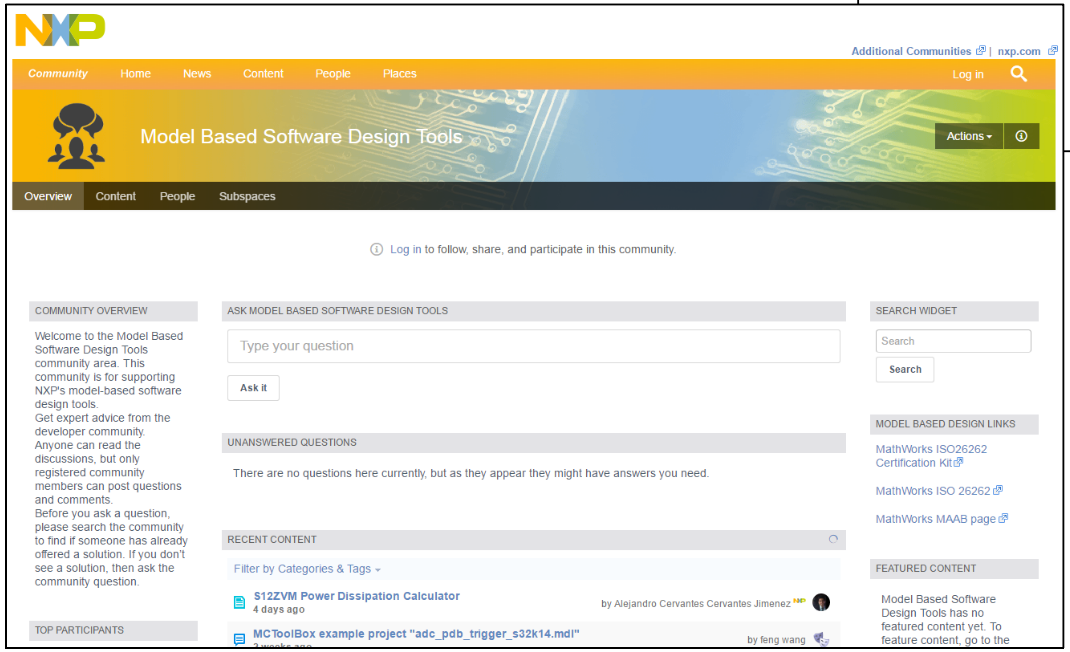
Step 4 — Target MCU Implementation
ANSI C Code Running on Target Hardware and MCU
Validation/verification phase
Controller code generation
Determine execution time on MCU
Start testing implementation on target ECM
Code generate control algorithm +
I/O drivers. Complete implementation on ECM.
Test system 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.

How to get it and where to find support

- Download MBD Toolbox: www.nxp.com/mctoolbox
- MATLAB: www.mathworks.com
- Support: https://community.nxp.com/community/mbdt





Thank you