Hands-on Workshop: Motor Control Toolbox
AMF-ENT-T1040

John Floros
Senior Software Engineer
Agenda

• **Overview: 30 minutes**
  - Introduction and Objectives
  - Motor Control Development Toolbox: Library blocks, FreeMASTER, and Bootloader
  - Model Based Design Steps: Simulation, SIL, PIL and ISO26262

• **Hands-on Demo: 20 minutes**
  - Convert simple model to run on Tower MPC5643L with MCD Toolbox and use FreeMASTER

• **Motor Control: 30 minutes**
  - Tower Motor Kit (Describe Freescale 3-Phase Pre-Driver and 3-Phase Motor)
  - Trapezoidal control and how to use it to turn a motor

• **Motor Control Hands-on Demo: 80 minutes**
  - Implement Trapezoidal Motor Control on MPC5643L
  - Run software from the model and use FreeMASTER to monitor and tune parameters

• **SIL/PIL Demo: 10 minutes**

• **Summary and Q&A: 10 minutes**
Introduction: WHAT DO WE DO?

• The Automotive Silicon Support Tools group’s objective is to develop software enablement tools to assist our customers with rapid prototyping and accelerate algorithm development on their target Freescale 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™.
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. Motor Control Development Toolbox allows this stage of the development to generate 100% of the code from the model.
Introduction: Motor Control Development Toolbox

- The Motor Control Development Toolbox includes an embedded target supporting Freescale 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 Automotive Math and Motor Control Function Library developed by Freescale’s renowned Motor Control Center of Excellence. The library provides dozens of blocks optimized for fast execution on Freescale 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 MC Toolbox:
- MATLAB (32-Bit)
- Simulink
- MATLAB Coder
- Simulink Coder
- Embedded Coder
Introduction: Reduce Development Time With MBD and MC Toolbox

System Requirements
Use software-based model vs. paper-based method, and start testing at very earliest stage.

Modeling/Simulation
Convert model to SIL and now can test ANSI-generated software. Can also use MC library with SIL testing.

Rapid Prototype
With MC library and MC Toolbox, test Model using target MCU and compiler through PIL testing.

Target MCU Implementation
With MC Toolbox, auto-generate code for direct interface of peripherals for target hardware without any manual hand code.

HIL Testing
Now that more testing on target has occurred earlier in the process, HIL testing time is reduced.

Functional Testing
Fewer defects found in this phase of testing, where finding defects is expensive.

Using Freescale’s Motor Control Development Toolbox with Model Based Design and you can reduce development time from this.
Introduction: Reduce Development Time With MBD and MC Toolbox

- System Requirements
- Modeling/Simulation
- Rapid Prototype
- HIL Testing
- Functional Testing
- Target MCU Implementation

To This!
Objectives

- After this Hands-on Workshop, you will be able to:
  
  - Use the Motor Control Development Toolbox to auto-generate and build software for the MPC5643L MCU directly from the MATLAB™/Simulink™ environment
  
  - Be able to configure the MPC5643L peripherals required to implement three phase motor control using the MPC5643L Tower System with the low-voltage Three Phase Motor Control Module
  
  - Implement Trapezoidal Motor Control software, from a model based design environment, on the MPC5643L Tower system and turn the brushless DC Motor provided with the Tower Motor Control Module
  
  - Know how the MPC5643L and Motor Control Development Toolbox can help with your motor control development projects
Objectives

• Exposure to Freescale’s hardware/software enablement

Modular, expandable and cost-effective development platform

Motor Control Development Toolbox with Simulink™

Real-time Debugging Tool
Data acquisition and calibration

Freescale™
Agenda

• Overview: 30 minutes
  - Introduction and Objectives
  - Motor Control Development Toolbox: Library blocks, FreeMASTER, and Bootloader
  - Model Based Design Steps: Simulation, SIL, PIL and ISO26262

• Hands-on Demo: 20 minutes
  - Convert simple model to run on Tower MPC5643L with MCD Toolbox and use FreeMASTER

• Motor Control: 30 minutes
  - Tower Motor Kit (Describe Freescale 3-Phase Pre-Driver and 3-Phase Motor)
  - Trapezoidal control and how to use it to turn a motor

• Motor Control Hands-on Demo: 80 minutes
  - Implement Trapezoidal Motor Control on MPC5643L
  - Run software from the model and use FreeMASTER to monitor and tune parameters

• SIL/PIL Demo: 10 minutes
• Summary and Q&A: 10 minutes
MCD Toolbox: Toolbox Library

The blocks provided in toolbox will provide the code necessary for initialization (reset, low level setup), interrupt/exception setup, device initialization, and peripheral driver code specifically to support motor control application development.

- Supports integration of Data Acquisition and Calibration tool FreeMASTER
- Incorporates Boot Loader utility for easy programming of application in flash or RAM
- Also supported: profiler function to measure execution of application code
### MCD Toolbox: Toolbox Library Contents

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Configuration/Modes</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Compiler Options</td>
<td>FreeMASTER Interface</td>
</tr>
<tr>
<td>ADC conversion</td>
<td>Available</td>
<td>Data acquisition</td>
</tr>
<tr>
<td>Digital I/O</td>
<td>CodeWarrior</td>
<td>Calibration</td>
</tr>
<tr>
<td>PIT timer</td>
<td>Wind River DIAB</td>
<td>Customize GUI</td>
</tr>
<tr>
<td>ISR</td>
<td>Green Hills</td>
<td></td>
</tr>
<tr>
<td>Communication Interface</td>
<td>RAM/FLASH targets</td>
<td></td>
</tr>
<tr>
<td>CAN driver</td>
<td>Simulation Modes</td>
<td></td>
</tr>
<tr>
<td>SPI driver</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>Motor Control Interface</td>
<td>Accelerator</td>
<td></td>
</tr>
<tr>
<td>Cross triggering unit</td>
<td>Software in the Loop (SIL)</td>
<td></td>
</tr>
<tr>
<td>PWM</td>
<td>Processor in the Loop (PIL)</td>
<td></td>
</tr>
<tr>
<td>eTimer block(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sine wave generation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCU Option</td>
<td>MCUs Supported</td>
<td></td>
</tr>
<tr>
<td>Multiple packages</td>
<td>MPC564xL</td>
<td></td>
</tr>
<tr>
<td>Multiple clock frequencies</td>
<td>MPC567xK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MPC574xP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PXS20/PXS30</td>
<td></td>
</tr>
</tbody>
</table>

- **Peripherals**
  - General
  - ADC conversion
  - Digital I/O
  - PIT timer
  - ISR
  - Communication Interface
  - CAN driver
  - SPI driver
  - Motor Control Interface
  - Cross triggering unit
  - PWM
  - eTimer block(s)
  - Sine wave generation
  - MCU Option
  - Multiple packages
  - Multiple clock frequencies

- **Configuration/Modes**
  - Compiler Options
  - Available
  - CodeWarrior
  - Wind River DIAB
  - Green Hills
  - RAM/FLASH targets
  - Simulation Modes
  - Normal
  - Accelerator
  - Software in the Loop (SIL)
  - Processor in the Loop (PIL)
  - MCUs Supported
  - MPC564xL
  - MPC567xK
  - MPC574xP
  - PXS20/PXS30

- **Utility**
  - FreeMASTER Interface
  - Data acquisition
  - Calibration
  - Customize GUI
  - Profiler Function
  - Exec. time measurement
  - Available in PIL
  - Available in standalone
MCD Toolbox: Auto Math and Motor Control Library

General Motor Control Library (GMCLIB)
- Park/Clarke Transformations
- Inverse Park/Clarke
- SVM
- DCB Ripple Elim

Standard motor control algorithms

General Functions Library (GFLIB)
- Sine, cosine, tangent
- Inverse sine, cosine, tangent
- Hysteresis
- LUT, ramp, limitations
- 1st, 2nd order IIR Filter
- Moving Average Filter

Basic mathematical functions

General Digital Filters Library (GDFLIB)

Advance filter functions

Stand-Alone

Controls algorithm development of PI, PID controllers or more advanced controls including functions like Park and Clarke transforms, filters ... integration of Motor Control Functions with Motor Control Development Toolbox. Developed by Freescale’s Motor Control Center of Excellence.
MCD Toolbox: Auto Math and Motor Control Library

This library helps quickly create bit-accurate single precision floating point models for simulation in Matlab for accurate verification and prototyping of motor control algorithms.

• Developed in ANSI-C, MISRA compliant
• Single precision floating point implementation optimized for MCU
• Theory and performance results summarized in documentation
GFLIB

- Trigonometric Functions
  - GFLIB_Sin
  - GFLIB_Cos
  - GFLIB_Tan
  - GFLIB_Asin
  - GFLIB_Acos
  - GFLIB_Atan
  - GFLIB_AtanXY
- Limitation Functions
  - GFLIB_Limit
  - GFLIB_LowerLimit
  - GFLIB_UpperLimit
  - GFLIB_VectorLimit
- PI Controller Functions
  - GFLIB_ControllerPIr
  - GFLIB_ControllerPIrAW
  - GFLIB_ControllerPip
  - GFLIB_ControllerPipAW
- Linear Interpolation
  - GFLIB_Lut1D
- Hysteresis Function
  - GFLIB_Hyst
- Signal Integration Function
  - GFLIB_IntegratorTR
- Sign Function
  - GFLIB_Sign
- Signal Ramp Function
  - GFLIB_Ramp

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
MCD Toolbox: RAppID Bootloader Utility

The RAppID Bootloader works with the built-in Boot Assist Module (BAM) included in the Freescale Qorivva and PX series 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 and S12ZVM

Graphical User Interface
Command

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

- **Real-Time Monitor and Control**
  - Display variables as text in a grid
    - Variables have **read and/or write** capability
    - View variables in desired format (hex, decimal, etc)
  - Oscilloscope waveforms
    - Up to 8 variable graphs or combined in one

- **Establish a Data Trace on Target**
  - Set up buffer (up to 64KB), sampling rate and trigger
  - Near 10-µs resolution
MCD Toolbox: Summary of Application Support

SYSTEM APPLICATION
- Application SW
  - API
  - Algorithm Libraries
- API
- Drivers
- On-Chip Peripherals

External Hardware
- External Connections
- PINS

System Infrastructure
- MC library set
- GFLIB: General functions
- GDFLIB: Digital filtering
- GMCLIB: Motor Control
- Drivers: Efficient Reflecting the chip features

User Application Software

FreeMaster Support

Boot Loader Support

Target Platform

Documentation

Freescale

Efficient reflecting the chip features
Any Questions?
Overview: 30 minutes
- Introduction and Objectives
- Motor Control Development Toolbox: Library blocks, FreeMASTER, and Bootloader
- Model Based Design Steps: Simulation, SIL, PIL and ISO26262

Hands-on Demo: 20 minutes
- Convert simple model to run on Tower MPC5643L with MCD Toolbox and use FreeMASTER

Motor Control: 30 minutes
- Tower Motor Kit (Describe Freescale 3-Phase Pre-Driver and 3-Phase Motor)
- Trapezoidal control and how to use it to turn a motor

Motor Control Hands-on Demo: 80 minutes
- Implement Trapezoidal Motor Control on MPC5643L
- Run software from the model and use FreeMASTER to monitor and tune parameters

SIL/PIL Demo: 10 minutes
Summary and Q&A: 10 minutes
Model Based Design Steps: Overview

- **Step 1** – Using a model to develop requirements and start testing at very earliest stage using idealized simulation in PC environment.
- **Step 2** – Convert model for Software In the Loop (SIL) to generate ANSI C. This allows testing of the C code using compiler on host PC using the same test vectors as Step 1. Verify functionality of C code as compared to simulation and determine code coverage.
- **Step 3** – Convert model for Processor In the Loop (PIL) and test generated code using target MCU and compiler with evaluation hardware. Perform equivalence testing and verify data coherency of software running on target MCU.
- **Step 4** – Generate code for target hardware and MCU with direct interfaces to peripherals. Be able to tune, log data, and analyze execution on target MCU in application target environment.
Model Based Design Steps: Step 1 (Simulation)

Simulation in PC environment

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.
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.
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.
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*

Model advisor, modeling standards checking
Simulation (model testing), model coverage, RMI
Module and integration testing at the model level
Review and static analysis at the model level

PIL testing using embedded IDE links
Real-Time Workshop Embedded Coder traceability report or model vs. code coverage comparison
Equivalence testing
Prevention of unintended functionality

Textual requirements → Executable specification → Model used for production code generation → Generated C code → Object code

Modeling

Simulink / Stateflow / Simulink Fixed Point

Real-Time Workshop Embedded Coder

*Workflow from The Mathworks™ Presentation Material Model-Based Design for IEC 61508 and ISO 26262
Green Hills Software MULTI Toolchain Certified as a Functional Safety Support Tool

- **Press Announcement: 9/20/2012**

- **TÜV NORD and exida confirm the qualification of MULTI according to the latest releases of the IEC 61508, EN 50128 and ISO 26262 functional safety standards**

  - Green Hills MULTI is the first and only commercially available toolchain certified to meet SIL 4 (Safety Integrity Level) and ASIL D (Automotive Safety Integrity Level) tool qualification requirements

  - The safety standards address industries such as:
    - IEC 61508:2010 (Industrial)
    - EN 50128:2011 (Railway)
    - ISO 26262:2011 (Automotive)
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.

* 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.
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 + 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.
Agenda

• **Overview: 30 minutes**
  - Introduction and Objectives
  - Motor Control Development Toolbox: Library blocks, FreeMASTER, and Bootloader
  - Model Based Design Steps: Simulation, SIL, PIL and ISO26262

• **Hands-on Demo: 20 minutes**
  - Convert simple model to run on Tower MPC5643L with MCD Toolbox and use FreeMASTER

• **Motor Control: 30 minutes**
  - Tower Motor Kit (Describe Freescale 3-Phase Pre-Driver and 3-Phase Motor)
  - Trapezoidal control and how to use it to turn a motor

• **Motor Control Hands-on Demo: 80 minutes**
  - Implement Trapezoidal Motor Control on MPC5643L
  - Run software from the model and use FreeMASTER to monitor and tune parameters

• **SIL/PIL Demo: 10 minutes**

• **Summary and Q&A: 10 minutes**
Hands-on Demo: Tower Hardware

**TWR-PXS2010**
- Features:
  - MPC5643L dual-core
  - Nexus interface
  - OSJTAG
  - CAN
  - UART
  - ADC
  - Potentiometer
  - Accelerometer
  - TWRPI socket

**TWR-IND-IO**
- Features:
  - USB to Serial
  - CAN
  - I/O
  - ADC

**TWR-MC-LV3PH**
- Hardware:
  - Input voltage 12-24V DC
  - Output current 5-10 Amps
  - 3-phase MOSFET inverter
  - 3-phase pre-driver MC33937
  - Analog sensing
  - Motor speed/position sensors interface
  - 2 pole-pair BLDC motor with Hall sensors (4000 RPM rated speed)
  - On-board power regulation for Tower System (single power supply via TWR-MC-LV3PH power jack)
Hands-on Demo: Read A/D and Toggle LED Simple Model

Run Simple Model Simulation
1. Open Model “TestLedA2D.mdl
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 on MPC5643L

1. Open Simulink Library
2. Go to Motor Control Toolbox for MPC5643L library
3. Drag the RAppID_MPC5643L_Config block into the model
4. Open block and go to PIL/BAM Setup tab
5. Check Download Code after build
6. Enter the COM port number that you are using from PC
7. Delete Sine Wave block and Scope
8. Also delete line that was going to second input of scope
9. Go back to library under General Purpose Blocks, drag in a Digital Output block and connect to the LED_State line
10. Go back to library under Motor Control Blocks and drag in an ADC Config block and a ADC Channel 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 on MPC5643L

11. Open “ADC Config block” and select ADC Converter 0.
Hands-on Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run on MPC5643L

13. Open Digital Output block and select pin “54 : PD6 : [ 34 ] : [ P3 ]”.

![Digital Output Block Parameters](image-url)
Hands-on Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run on MPC5643L

• This is what the model should look like after step 13
Hands-on Demo: Read A/D and Toggle LED Simple Model

Convert Simple Model and Run on MPC5643L

14. Go to Tools pull down menu and then select generate code / Build Model.

15. Wait for model to generate code and then a prompt from the RAppID Bootloader Utility will appear. Cycle power to Tower and then select “OK”.

16. Turn the Potentiometer on the Tower MPC5643L CPU card 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

17. Start FreeMASTER and open project TestLedA2D.pmp. Just press OK if a message comes up that the map file has been updated.

18. Go to Project Options Pull Down and select “Options”. Verify that COM settings are the same as what were set in your model.

19. 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 19
Hands-on Demo: Read A/D and Toggle LED Simple Model

Using FreeMASTER with Hands-on Demo

20. 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

21. Disconnect FreeMASTER by pressing the STOP button. Then rebuild the model and have the bootloader download the software to the MPC5643L. Re-Connect FreeMASTER and turn the Pot. You should see the following:
Hands-on Demo: Read A/D and Toggle LED Simple Model

Any Questions?
Agenda

• **Overview: 30 minutes**
  - Introduction and Objectives
  - Motor Control Development Toolbox: Library blocks, FreeMASTER, and Bootloader
  - Model Based Design Steps: Simulation, SIL, PIL and ISO26262

• **Hands-on Demo: 20 minutes**
  - Convert simple model to run on Tower MPC5643L with MCD Toolbox and use FreeMASTER

• **Motor Control: 30 minutes**
  - Tower Motor Kit (Describe Freescale 3-Phase Pre-Driver and 3-Phase Motor)
  - Trapezoidal control and how to use it to turn a motor

• **Motor Control Hands-on Demo: 80 minutes**
  - Implement Trapezoidal Motor Control on MPC5643L
  - Run software from the model and use FreeMASTER to monitor and tune parameters

• **SIL/PIL Demo: 10 minutes**

• **Summary and Q&A: 10 minutes**
Tower Motor Kit Overview

Features

- Electrical specifications:
  - Input voltage 12-24V DC
  - Output current 5-10 Amps
- 3-phase MOSFET inverter
- 3-phase pre-driver MC33937, configurable thru SPI analog sensing (dcb voltage, dcb current, phase currents, back-EMF voltage)
- Motor speed/position sensors interface (Encoder, Hall)
- Hardware over-current fault protection
- On-board power regulation for Tower System (single power supply via TWR-MC-LV3PH power jack)
- Brushless DC (BLDC) Motor Linix 45ZWN24-40

Purpose

- The purpose of the Low Voltage Motor Control Tower module is to be used by customers to prototype DC, BLDC and PMSM motor control solutions and to evaluate/demonstrate various algorithms.
Tower Motor Kit: TWR-MC-LV3PH Module

Motor Connector

MOSFET H-Bridge

Power Supply Connector

Motor Hall/Encoder Connector

Freescale 3-Phase Pre-Driver Chip (MC33937)
**Tower Motor Kit: 3PP Driver Chip (MC33937)**

**Features**
- Fully specified from 8.0V to 40V; covers 12V and 24V automotive systems
- Extended operating range from 6.0V to 58V covers 12V and 42V systems
- 1.0A gate drive capability with protection
- Protection against reverse charge injection
- Includes a charge pump to support full FET drive at low battery voltages
- Dead time is programmable via the SPI
- Simultaneous output capability enabled via safe SPI command

**Purpose**
- The IC contains three high-side FET pre-drivers and three low-side FET pre-drivers. Three external bootstrap capacitors provide gate charge to the high side FETs. The IC interfaces to a MCU via six input control signals, a SPI port for device setup and asynchronous reset, enable and interrupt signals
Tower Motor Kit: Brushless DC Motor

Features
Model: Linix 45ZWN24-40
Motor Type: Brushless DC
Windings: "Y" Connection Method
Pole Pairs: 2 pairs
Rated Voltage: 24V
Rated Current: 2.3 A
Rated Torque: 990 g.cm
Rated Power: 40 Watts
Rated Speed (Load): 4000 RPM
Speed (Un-Loaded): 4900 RPM
Position Sensing: Hall Type (A, B, C)

Note: Pole pair count for this motor means that every single mechanical revolution equals two electrical revolutions. State change in Hall sensors is every 60 degrees electrical.
Tower Motor Kit: MPC564xL MCU

**Core**
- Up to 120MHz 32-bit e200z4 core
- 4K I-cache
- Floating point unit (FPU)
- Safety enhanced cores + FPU + VLE + MMU
- Dual parallel or lock step configuration + HW/SW monitoring

**Memory**
- Up to 1 MByte RWW Flash with ECC
- Up to 128 KByte SRAM with ECC
- EE emulation
- Dual crossbar with memory protection unit (MPU)

**I/O**
- 2x CAN (32 message buffers each)
- 1x FlexRay (64 msg. buffers)
- 2x UART
- 3x SPI
- 1 x External clock output
- 2x FlexPWM (2 x 12 channels)
- 3x eTimer modules (3 x 6 channels)
- Dual ADC (16 channels each, 12 bit), 1 S/H per ADC
- 1 x cross-triggering unit for motor control

**System**
- 16 Channels eDMA
- Fault Collection and Control Unit
- CRC computing unit
- Software watchdog timer (inc. window mode, flow monitoring)
- 2x temperature sensor
- Nexus debug interface (up to N3)
- FM-PLL + FlexRay PLL
- 3.3 V Single supply with external and internal ballast transistor
- 3.3 V I/Os (ADC 5 V capable)
Tower Motor Kit: System Diagram

- 3PP Driver Chip (MC33937)
- Phase A, Phase B, Phase C
- BLDC Motor
- Hall Sensors
- Vb+, Vb-
- SPI, PWM
- MPC5643L
- ADC, eTimer/DI
- FlexPWM, SPI
- LinFlex
- RAppID BL Utility
- DO, LEDs
- FREEMASTER
Any Questions?
Agenda

- **Overview: 30 minutes**
  - Introduction and Objectives
  - Motor Control Development Toolbox: Library blocks, FreeMASTER, and Bootloader
  - Model Based Design Steps: Simulation, SIL, PIL and ISO26262

- **Hands-on Demo: 20 minutes**
  - Convert simple model to run on Tower MPC5643L with MCD Toolbox and use FreeMASTER

- **Motor Control: 30 minutes**
  - Tower Motor Kit (Describe Freescale 3-Phase Pre-Driver and 3-Phase Motor)
  - Trapezoidal control and how to use it to turn a motor

- **Motor Control Hands-on Demo: 80 minutes**
  - Implement Trapezoidal Motor Control on MPC5643L
  - Run software from the model and use FreeMASTER to monitor and tune parameters

- **SIL/PIL Demo: 10 minutes**

- **Summary and Q&A: 10 minutes**
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: 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

- Phase A
- Phase B
- Phase C
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.

<table>
<thead>
<tr>
<th>CW</th>
<th>Phase A</th>
<th>Phase B</th>
<th>Phase C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/180°</td>
<td>Vb-</td>
<td>NC</td>
<td>Vb+</td>
</tr>
<tr>
<td>30°</td>
<td>Vb-</td>
<td>Vb+</td>
<td>NC</td>
</tr>
<tr>
<td>60°</td>
<td>NC</td>
<td>Vb+</td>
<td>Vb-</td>
</tr>
<tr>
<td>90°</td>
<td>Vb+</td>
<td>NC</td>
<td>Vb-</td>
</tr>
<tr>
<td>120°</td>
<td>Vb+</td>
<td>Vb-</td>
<td>NC</td>
</tr>
<tr>
<td>150°</td>
<td>NC</td>
<td>Vb-</td>
<td>Vb+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Vb+</th>
<th>Vb-</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Switch</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>Bottom Switch</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>CCW</th>
<th>Phase A</th>
<th>Phase B</th>
<th>Phase C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°/180°</td>
<td>Vb+</td>
<td>NC</td>
<td>Vb-</td>
</tr>
<tr>
<td>30°</td>
<td>Vb+</td>
<td>Vb-</td>
<td>NC</td>
</tr>
<tr>
<td>60°</td>
<td>NC</td>
<td>Vb-</td>
<td>Vb+</td>
</tr>
<tr>
<td>90°</td>
<td>Vb-</td>
<td>NC</td>
<td>Vb+</td>
</tr>
<tr>
<td>120°</td>
<td>Vb-</td>
<td>Vb+</td>
<td>NC</td>
</tr>
<tr>
<td>150°</td>
<td>NC</td>
<td>Vb+</td>
<td>Vb-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Vb+</th>
<th>Vb-</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Switch</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>Bottom Switch</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
</tr>
</tbody>
</table>
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”.

![Diagram of Hall sensors and rotor poles](image)
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.

<table>
<thead>
<tr>
<th>Hall A</th>
<th>Hall B</th>
<th>Hall C</th>
<th>State</th>
<th>CW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0/180°</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>30°</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>60°</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>90°</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>120°</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>150°</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Invalid</td>
<td>n/a</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Invalid</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Hall A

Hall B

Hall C
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.

<table>
<thead>
<tr>
<th>Hall A</th>
<th>Hall B</th>
<th>Hall C</th>
<th>State</th>
<th>CCW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0/180°</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>30°</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>60°</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>90°</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>120°</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>150°</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Invalid</td>
<td>n/a</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Invalid</td>
<td>n/a</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Hall A</th>
<th>Hall B</th>
<th>Hall C</th>
<th>State</th>
<th>CW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0/180°</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>30°</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>60°</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>90°</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>120°</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>150°</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Invalid</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>Invalid</td>
</tr>
</tbody>
</table>

### Commutation Table Output

<table>
<thead>
<tr>
<th>CW</th>
<th>Phase A</th>
<th>Phase B</th>
<th>Phase C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/180°</td>
<td>Vb+</td>
<td>NC</td>
<td>Vb-</td>
</tr>
<tr>
<td>30°</td>
<td>Vb+</td>
<td>Vb-</td>
<td>NC</td>
</tr>
<tr>
<td>60°</td>
<td>NC</td>
<td>Vb-</td>
<td>Vb+</td>
</tr>
<tr>
<td>90°</td>
<td>Vb-</td>
<td>NC</td>
<td>Vb+</td>
</tr>
<tr>
<td>120°</td>
<td>Vb-</td>
<td>Vb+</td>
<td>NC</td>
</tr>
<tr>
<td>150°</td>
<td>NC</td>
<td>Vb+</td>
<td>Vb-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Top Switch</th>
<th>Vb+</th>
<th>Vb-</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bottom Switch</th>
<th>Vb+</th>
<th>Vb-</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td></td>
</tr>
</tbody>
</table>
Trapezoidal Control: Bringing It All Together

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

<table>
<thead>
<tr>
<th>Hall A</th>
<th>Hall B</th>
<th>Hall C</th>
<th>State</th>
<th>CW</th>
<th>Phase A</th>
<th>Phase B</th>
<th>Phase C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0/180°</td>
<td>Vb+</td>
<td>NC</td>
<td>Vb-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>30°</td>
<td>Vb+</td>
<td>Vb-</td>
<td>NC</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>60°</td>
<td>NC</td>
<td>Vb-</td>
<td>Vb+</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>90°</td>
<td>Vb-</td>
<td>NC</td>
<td>Vb+</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>120°</td>
<td>Vb-</td>
<td>Vb+</td>
<td>NC</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>150°</td>
<td>NC</td>
<td>Vb+</td>
<td>Vb-</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Invalid</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Invalid</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Trapezoidal Control Algorithm Clockwise Rotation

<table>
<thead>
<tr>
<th>Top Switch</th>
<th>Vb+</th>
<th>Vb-</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>Off</td>
<td>Off</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bottom Switch</th>
<th>Vb+</th>
<th>Vb-</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Off</td>
<td>On</td>
<td>Off</td>
</tr>
</tbody>
</table>
With the commutation table and the motor position table, a full trapezoidal control algorithm can be developed.

### Trapezoidal Control Algorithm Counter Clockwise Rotation

<table>
<thead>
<tr>
<th>Hall A</th>
<th>Hall B</th>
<th>Hall C</th>
<th>State</th>
<th>CW</th>
<th>Phase A</th>
<th>Phase B</th>
<th>Phase C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0/180°</td>
<td>Vb-</td>
<td>NC</td>
<td>Vb+</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>30°</td>
<td>Vb-</td>
<td>Vb+</td>
<td>NC</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>60°</td>
<td>NC</td>
<td>Vb+</td>
<td>Vb-</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>90°</td>
<td>Vb+</td>
<td>NC</td>
<td>Vb-</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>120°</td>
<td>Vb+</td>
<td>Vb-</td>
<td>NC</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>150°</td>
<td>NC</td>
<td>Vb-</td>
<td>Vb+</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Invalid</td>
<td>n/a</td>
<td>Vb+</td>
<td>Vb-</td>
<td>NC</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Invalid</td>
<td>n/a</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
</tr>
</tbody>
</table>

**Switches**

- **Top Switch**
  - On
  - Off
  - Off

- **Bottom Switch**
  - Off
  - On
  - Off
Any Questions?
Agenda

• **Overview: 30 minutes**
  - Introduction and Objectives
  - Motor Control Development Toolbox: Library blocks, FreeMASTER, and Bootloader
  - Model Based Design Steps: Simulation, SIL, PIL and ISO26262

• **Hands-on Demo: 20 minutes**
  - Convert simple model to run on Tower MPC5643L with MCD Toolbox and use FreeMASTER

• **Motor Control: 30 minutes**
  - Tower Motor Kit (Describe Freescale 3-Phase Pre-Driver and 3-Phase Motor)
  - Trapezoidal control and how to use it to turn a motor

• **Motor Control Hands-on Demo: 80 minutes**
  - Implement Trapezoidal Motor Control on MPC5643L
  - Run software from the model and use FreeMASTER to monitor and tune parameters

• **SIL/PIL Demo: 10 minutes**

• **Summary and Q&A: 10 minutes**
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

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 MPC5643L
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

Configure MPC5643L Target steps:

- DSPI – Check to enable cross checks; Uncheck to disable cross checks.
- CAN – Check to enable cross checks; Uncheck to disable cross checks.
- ADC – Check to enable cross checks; Uncheck to disable cross checks.
- PWM – Check to enable cross checks; Uncheck to disable cross checks.
- CTU – Check to enable cross checks; Uncheck to disable cross checks.
- SWG – Check to enable cross checks; Uncheck to disable cross checks.
- eTimer – Check to enable cross checks; Uncheck to disable cross checks.
- FreeMaster – Check to enable cross checks; Uncheck to disable cross checks.
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 MPC5643L

Configure Event Control Block using eTimer steps:

- Interrupt Triggered
- Interrupt Triggered
- Interrupt Triggered
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

Configure Event Control Block using eTimer steps:

Measure the pulse width of a hall sensor so that motor speed can be calculated
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

Configure Event Control Block using eTimer steps:

- **Sub Module 0 Channel 0**
  - Count mode: 001 - Count rising edges of primary source
  - Output mode: 0000 - Software controlled
  - PrimarySource: 11111 - IP Bus clock divide by 128 prescaler
  - SecondarySource: 00010 - Counter #2 input pin

- **Sub Module 0 Channel 1**
  - Count mode: 001 - Count rising edges of primary source
  - Output mode: 0000 - Software controlled
  - PrimarySource: 11111 - IP Bus clock divide by 128 prescaler
  - SecondarySource: 00011 - Counter #3 input pin

- **Sub Module 0 Channel 2**
  - Count mode: 001 - Count rising edges of primary source
  - Output mode: 0000 - Software controlled
  - PrimarySource: 11111 - IP Bus clock divide by 128 prescaler
  - SecondarySource: 00100 - Counter #4 input pin

---

**Diagram:**

- eTimer configuration Hall A
- eTimer configuration Hall B
- eTimer configuration Hall C

**Events:**

- eTimer ISR Hall A
- eTimer ISR Hall B
- eTimer ISR Hall C

**Flowchart:**

- eTimerCh0FunctionCall
- Terminator5
- terminator
- HallStateChange

---

Freescale™
Configure Event Control Block using eTimer steps:

- **Channel**: 0
- **Count mode**: 001 - Count rising edges of primary source
- **Count once**: 0 - Count repeatedly
- **Count length**: 0 - Continue counting to roll over
- **Count Direction**: 0 - Count up
- **Output Enable**: 0 - The external pin is configured as an input
- **Co-channel initialization**: 00 - Other channels cannot force re-initialization of this channel
- **Stop Action**: 0 - Output enable is unaffected by stop mode
- **Reload on Capture**: 01 - Reload the counter on a capture 1 event
Configure Event Control Block using eTimer steps:
Configure Event Control Block using eTimer steps:

- Capture 1 Mode: 10 - Capture rising edges
- Capture 2 Mode: 01 - Capture falling edges
Configure Event Control Block using eTimer steps:
Configure Event Control Block using eTimer steps:
Configure Event Control Block using eTimer steps:
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

Configure HallAPulseInput Block using eTimer steps:
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

MotorSpeedReqInput Block with ADC and Digital outputs steps:
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

MPC33927_InterfaceOut Block with DSPI Block steps:
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

MPC33927_InterfaceOut Block with DSPI Block steps:
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

MPC33927_InterfaceOut Block with DSPI Block steps:

Configure DSPI Block

- **Frame Size**: 8
- **Clock Polarity**: Low
- **Clock Phase**: Data is changed on the leading edge of SCK and captured on the following edge
- **Data Transfer Mode**: MSB first
- **Double Baud Rate (DBR)**
- **PCS to SCK Delay Prescaler (PCSSCK)**: 1
- **After SCK Delay Prescaler (PASC)**: 1
- **Delay After Transfer Prescaler (PDT)**: 5
- **Baud Rate Prescaler (PBR)**: 5
- **PCS to SCK Delay Scalar (CSSCK)**: 256
- **After SCK delay Scalar (ASC)**: 256
- **Delay after Transfer Scalar (DT)**: 8
- **Baud Rate Scalar (BR)**: 128

[Diagram of DSPI configuration settings]
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

MPC33927_InterfaceOut Block with DSPI Block steps:
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

3PhaseDutyCycleOut Block with Flex PWM Blocks steps:

Pull Simple PWM phase block from library, connect to phase C and configure.
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

3PhaseDutyCycleOut Block with Flex PWM Blocks steps:
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

3PhaseDutyCycleOut Block with Flex PWM Blocks steps:
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

3PhaseDutyCycleOut Block with Flex PWM Blocks steps:
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

3PhaseDutyCycleOut Block with Flex PWM Blocks steps:
Copy phase C block, paste twice and connect to phase A & B.
Hands-on Demo: Implement Trapezoidal Motor Control on MPC5643L

3PhaseDutyCycleOut Block with Flex PWM Blocks steps:

Configure phase A & B pins.
Hands-on Demo: FreeMASTER to Monitor and Tune Parameters

Using FreeMASTER with Hands-on Demo

1. Start FreeMASTER and open project MPC5643L_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 4000 RPM.

5. You should see the following graphic (see next slide).
Hands-on Demo: FreeMASTER to Monitor and Tune Parameters
Hands-on Demo: Trapezoidal Motor Control

Any Questions?
Agenda

- **Overview: 30 minutes**
  - Introduction and Objectives
  - Motor Control Development Toolbox: Library blocks, FreeMASTER, and Bootloader
  - Model Based Design Steps: Simulation, SIL, PIL and ISO26262

- **Hands-on Demo: 20 minutes**
  - Convert simple model to run on Tower MPC5643L with MCD Toolbox and use FreeMASTER

- **Motor Control: 30 minutes**
  - Tower Motor Kit (Describe Freescale 3-Phase Pre-Driver and 3-Phase Motor)
  - Trapezoidal control and how to use it to turn a motor

- **Motor Control Hands-on Demo: 80 minutes**
  - Implement Trapezoidal Motor Control on MPC5643L
  - Run software from the model and use FreeMASTER to monitor and tune parameters

- **SIL/PIL Demo: 10 minutes**

- **Summary and Q&A: 10 minutes**
1. Open Model “FTF_IND_F0072_SIL_PIL_FOC.mdl"
2. You will see a motor simulation of an FOC control algorithm
3. Will Run model and compare the difference b/t simulation and running on target.
SIL/PIL Demo

1. You can switch between SIL and PIL thru using the tools menu.
Any Questions?
Agenda

- **Overview: 30 minutes**
  - Introduction and Objectives
  - Motor Control Development Toolbox: Library blocks, FreeMASTER, and Bootloader
  - Model Based Design Steps: Simulation, SIL, PIL and ISO26262

- **Hands-on Demo: 20 minutes**
  - Convert simple model to run on Tower MPC5643L with MCD Toolbox and use FreeMASTER

- **Motor Control: 30 minutes**
  - Tower Motor Kit (Describe Freescale 3-Phase Pre-Driver and 3-Phase Motor)
  - Trapezoidal control and how to use it to turn a motor

- **Motor Control Hands-on Demo: 80 minutes**
  - Implement Trapezoidal Motor Control on MPC5643L
  - Run software from the model and use FreeMASTER to monitor and tune parameters

- **SIL/PIL Demo: 10 minutes**

- **Summary and Q&A: 10 minutes**
Summary

• You now know how to do trapezoidal control and auto-generate software with the Motor Control Development Toolbox for the MPC5643L MCU directly from the MATLAB™/Simulink™ environment using the Tower system with the TWR-MC-LV3PH submodule.

• You now understand how to run SIL and PIL with MCD Toolbox and how it can accelerate your development, including systems being developed under ISO26262 using PIL.

• You have gained a good working knowledge of how FreeMASTER works and how it can be used with MCD Toolbox to accelerate development when working with the target hardware.

• You have a working knowledge of how to use the Freescale 3-Phase Pre-Driver Chip (MC33937) with a motor control application.

• You know that the MPC5643L is the ideal MCU for your motor control application.
Any Questions?
MathWorks Announces Simulink Code Generation Targets in New Freescale Motor Control Development Toolbox
www.mathworks.com

- Simulink and Embedded Coder enable engineers to generate production code for Freescale MCUs in IEC 61508 (SIL3) and ISO 26262 (ASIL-D) compliant systems.

Freescale likes model-based design, says MathWorks
www.ElectronicsWeekly.com

- MathWorks says Freescale has made a major commitment to model-based design methodologies by adopting Simulink code generation targets in its motor control development toolbox. The toolbox, consisting of Simulink motor control blocks and target-ready ...

A model-based tool to support rapid application development for Freescale MCUs
www.Freescale.com - Beyond Bits—Issue VIII

- Model-based design (MBD) is becoming the standard methodology for developing embedded systems that implement the desired behavior of a control system. MBD is a graphical method …