FASTER MOTOR CONTROL DEVELOPMENT

AUTOMOTIVE MICROCONTROLLERS AND PROCESSORS

JOHN H. FLOROS
FIELD APPLICATION ENGINEER

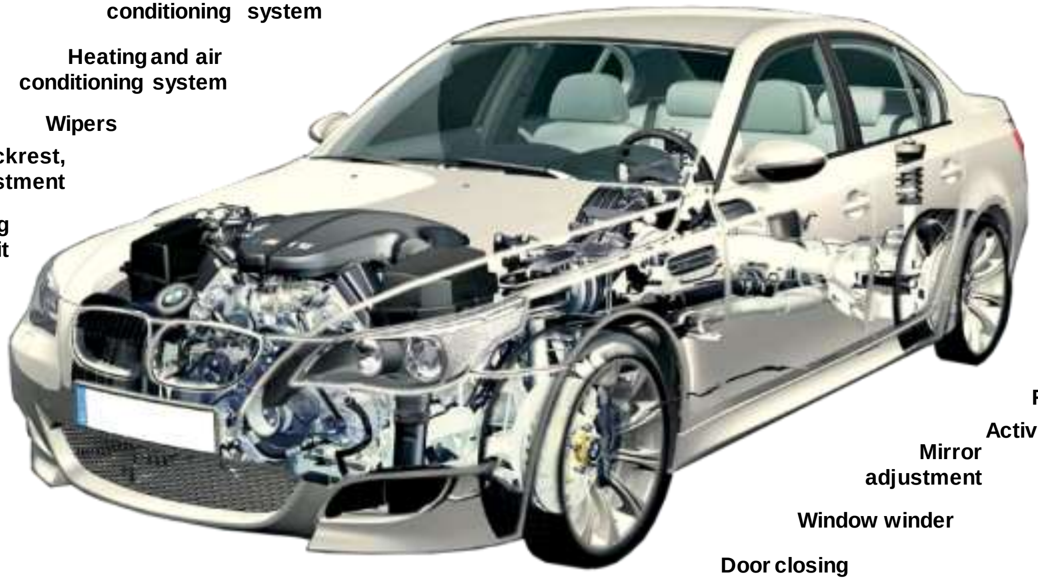
MAREK MUSAK
APPLICATION ENGINEER

5TH, OCTOBER, 2016

PUBLIC

SECURE CONNECTIONS FOR A SMARTER WORLD

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



Motor Control

General Purpose and Integrated Solutions

Target Markets

Primarily Body Electronics

Lighting

Body Controller

HVAC

Actuators and Sensors

Doors/Seats

Sensors

Pumps/Fans

Products

General Purpose

S32K

The First Automotive MCU
Designed for Software Engineers

Integrated

MagniV

Shrink your application with MCU
+ HV analog integration

Technology

Common Platform

ARM Cortex Architecture

+

HV analog Integration

SW

Tools

IP

TFS Flash

NXP S32 Automotive Processors

Existing Products

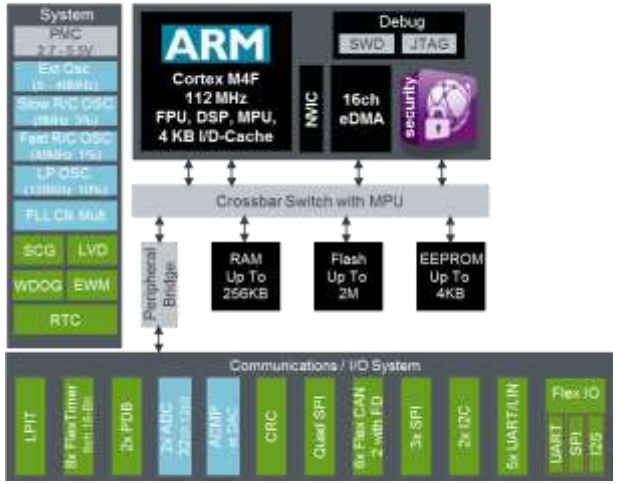
ARM-based processors
MPC57xx, MPC56xx, MPC55xx MCUs
Power Architecture®-based processors
S12(X) MCUs
S12 MagniV mixed-signal MCUs
Image Cognition processors
Kinetis auto MCUs (KFA)
S08 MCUs
ARM Cortex-based MCUs
MAC57Dxx MCUs
Others

New Products

S32 Automotive Processors

S32K

Introducing S32K – The First Auto MCU designed for SW Engineers

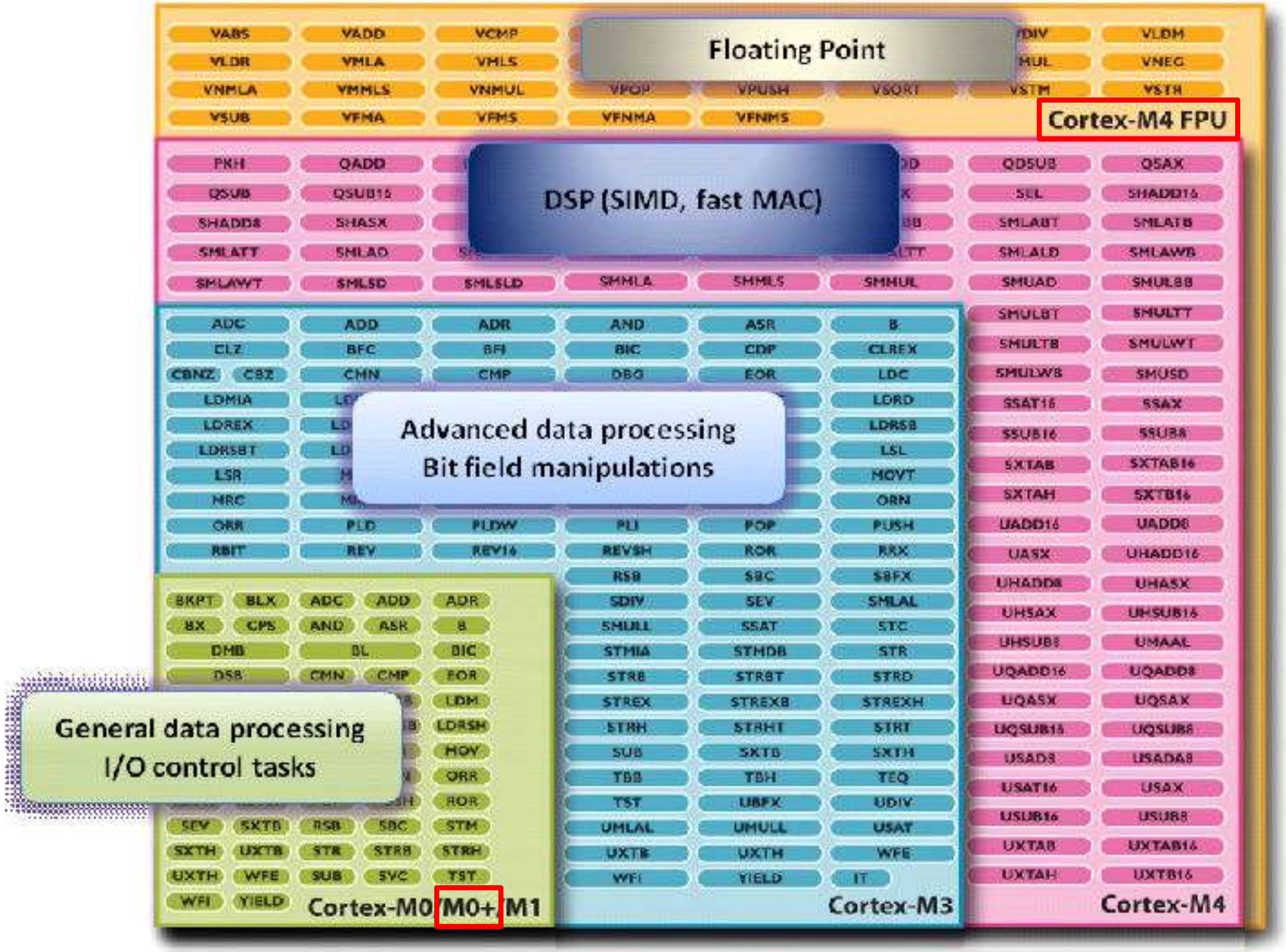


Most Scalable Portfolio
16K to 2+Mbyte hardware and software compatibility for maximum reuse and scalability

Superior Performance and Features
Best low-power, functional safety and security features, system cost savings

Reduce R&D cost and Time to Market
With new S32 Design Studio and Software Devt Kit (SDK)

KEA / S32K / S32M Scalability



Common peripherals

- ISO CAN-FD
- Security
- LIN / UART
- SPI / I2C
- Timer
- ADC
- Clocks

- 8 KB to 2 MB+ Flash
- 16 to 176 pins
- ARM® Cortex®-M0+ up to M4F
- Pin and IP module compatibility within the product series and with KEA products

S32K Microcontroller Performance And Features

Superior Performance

- High speed ARM Cortex-M4F CPU with DSP functionality
- IEEE-754 HW floating point unit without SW overhead
- Harvard architecture accelerates data handling
- 16 bit instruction set (THUMB 2) → ~31% reduced memory usage
- Combined D/I cache for direct execution
- Concurrent, low latency bus accesses over crossbar
- Parallel DMA operation
- Dedicated EEPROM to support read while write
- 100Mbit/s Ethernet
- IEEE 1588 Time Stamping

Highest Energy Efficiency

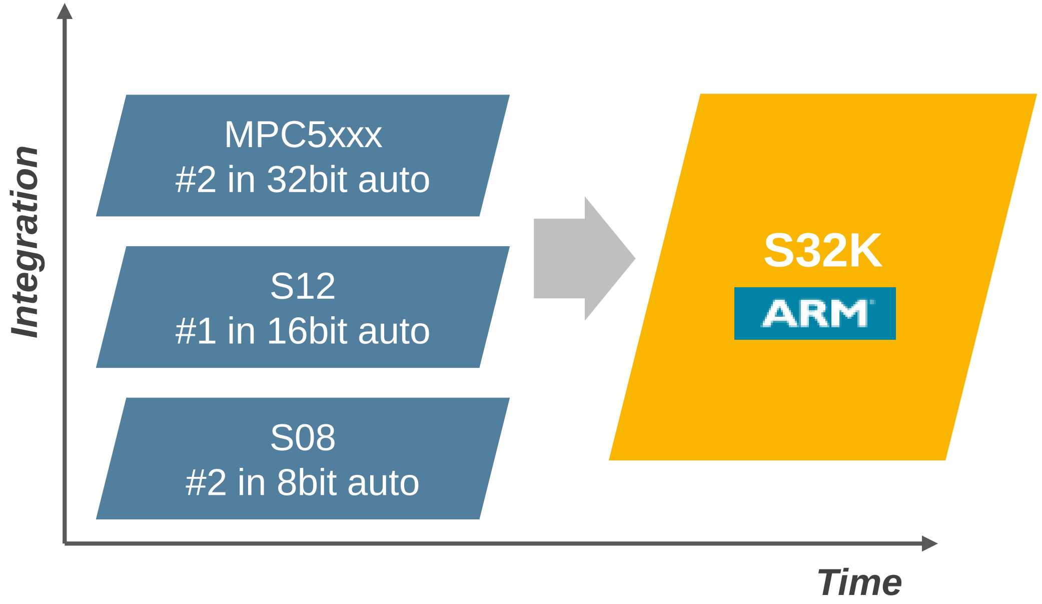
- Low leackage technology
- Multiple low power modes
- Internal oscillators e.g. 48MHz 1%
- Best in class STOP current

Future Proof Features

- CAN with Flexible Datarate (FD) option according to ISO/CD 11898-1
- HW motor control support (BLDC/PMSM)
- ISO26262 compliance (ASIL-B)
- Communication protocol emulation module (FlexIO)
- HW security engine

General Purpose Automotive 16/32bit

- **First** Auto MCU designed for SW engineers
- **Reducing time-to-market** by months and quarters
- Moving to **ARM Cortex architecture**
- **Future-proofing** through superior performance and advanced feature set



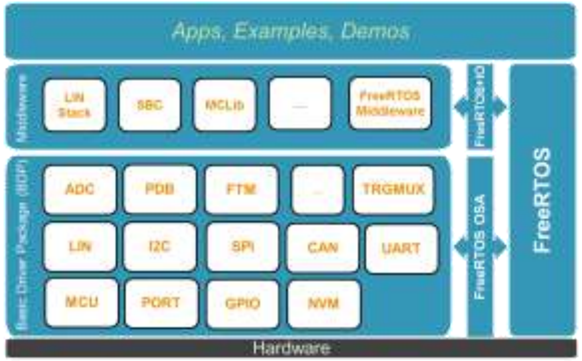
S32K Microcontroller Solution Offering

Hardware Platform

- Low cost development board compatible to Arduino shields
- Onboard debugger and system basis chip

Full Hardware evaluation and Development Platforms

+ Software



- Full-featured, no cost development platform (S32 DS)
- Production grade NXP Software Development Kit (SDK)
- NXP libraries e.g. Core Self Test, LIN Stack, Automotive Math and Motor Control Library
- Autosar 4.x MCAL and OS

Complete software package to streamline software development

+ Ecosystem

- IAR and Cosmic toolchains, more to follow
- Including community software – FreeRTOS
- Design services
- Training
- Arduino shields

Technology alliances for building smarter, better

Motor Control With S32K

Target Application:

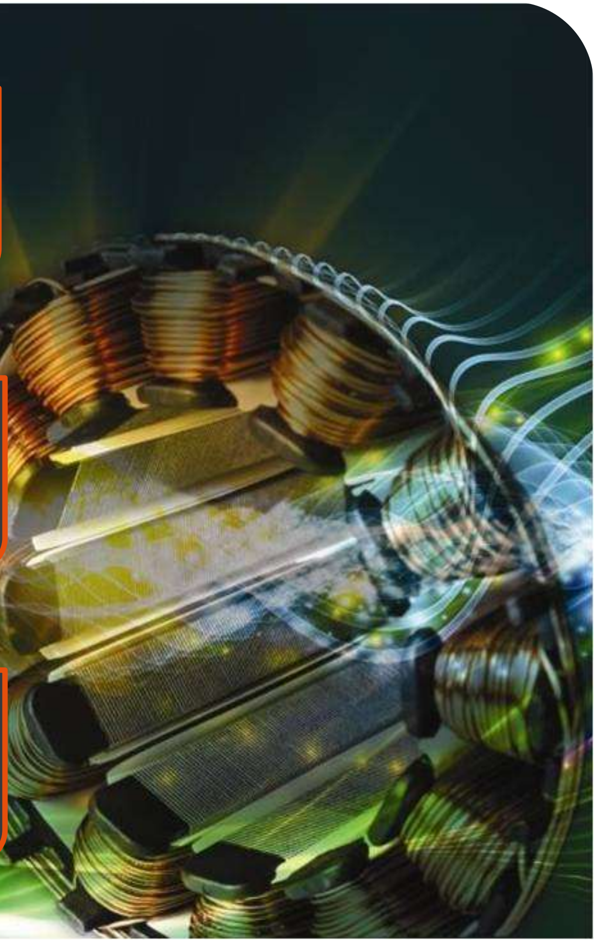
- HVAC Blower
Wipers
AC Compressor
- Fuel
Water
Oil
Pump
- Sliding Doors
Powered Liftgate

Benefits:

- **Flexibility**
Multiple compatible devices & packages
- **Performance**
Lower power
High temp (Ta 125 ˚C)
- **Scalable**
256 K to 2 M Flash
CAN FD & LIN
Hardware & software re-use

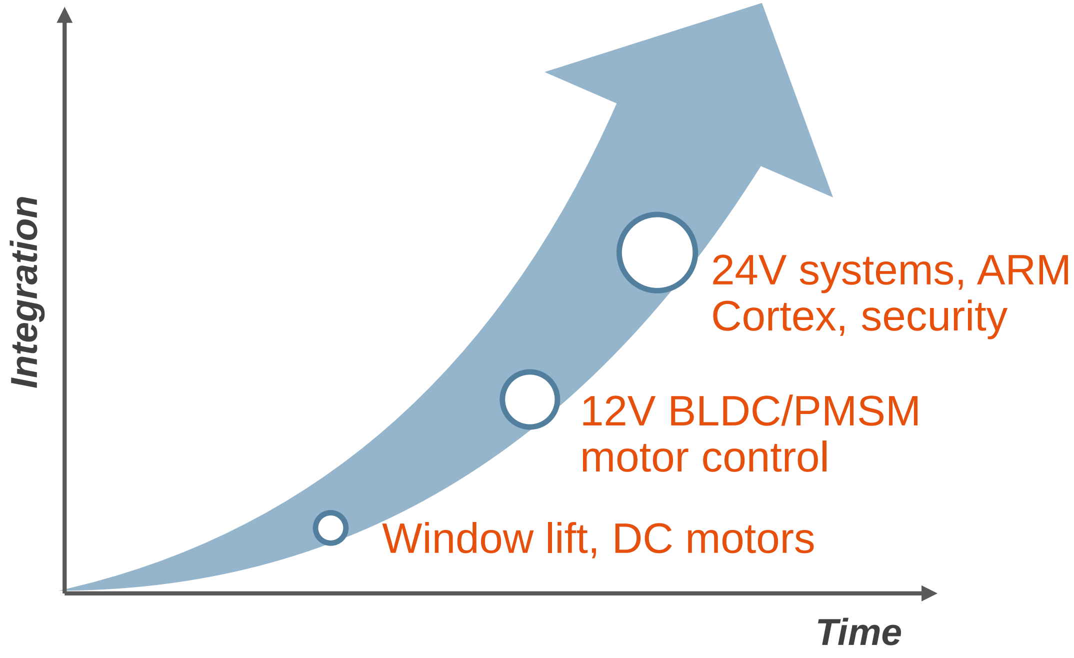
Enablement:

- **Hardware**
Eval board
MC Development Kit
NXP Freedom+ board
- **Software**
Development
Production
Compiler, debugger
- **Libraries**
Optimized math & motor control algorithms

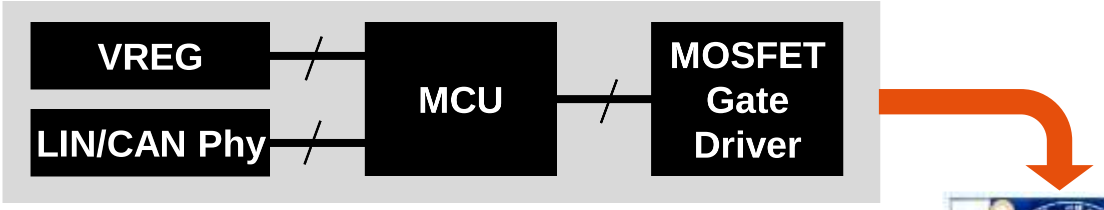


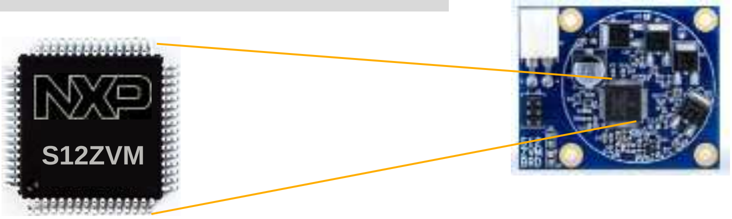
Integrated Solutions – MagniV

- Saves **20%** PCB/module size
- Improves **manufacturing efficiency**
- **Simplifies** system design



Single Chip Integration of MCU + HV Analog





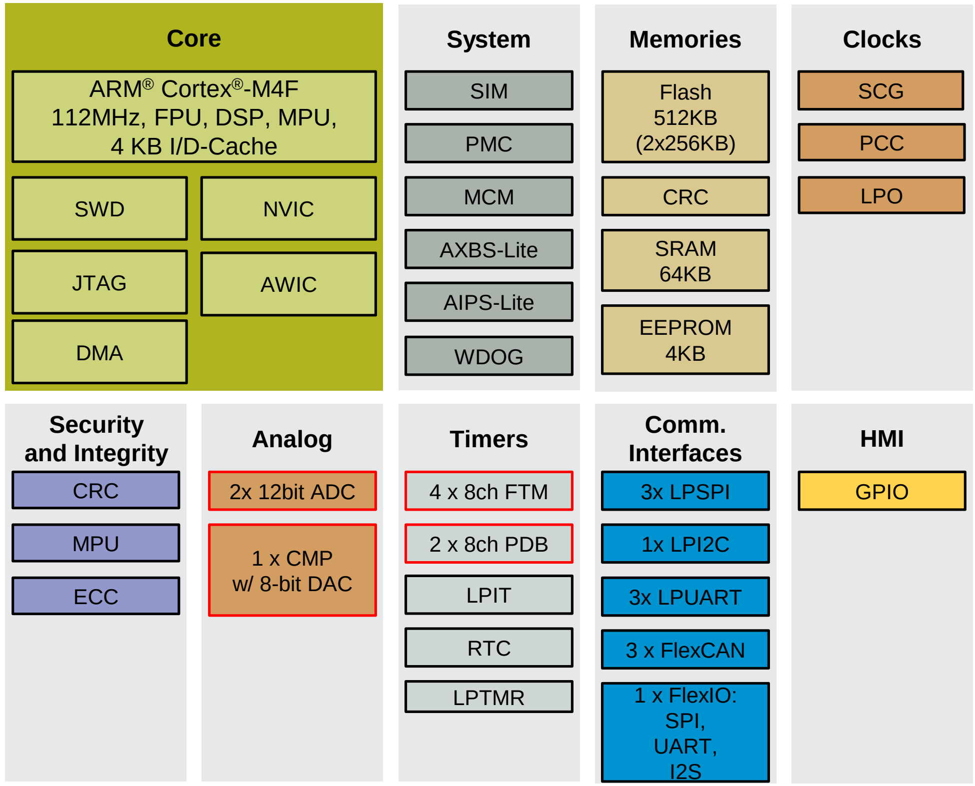
Driving Vehicle Electrification

Enabling Miniaturization

S32K144: Overview

ARM Cortex-M4F
- 32-bit processor HSRUN=112MHz, RUN=80MHz
- Harvard architecture w/ 3 stage pipeline
- HW multiplier & divider (32/64bit result)
- Floating Point Unit (FPU) (single float)
- Extended Thumb2 DSP instruction set
- Nested Vectored Interrupt Controller (NVIC) with up to 240 interrupts w/ 16 configurable priorities
- Awake Interrupt Controller AWIC
- Memory Protection Unit (MPU)
- SWD & JTAG debug interfaces

System
- System Integration Module (SIM)
- Power Management Controller (PMC)
- Miscellaneous control module (MCM)
- Crossbar switch (AXBS-Lite)
- Peripheral bridge (AIPS-Lite)
- enhanced Direct Memory Access controller (eDMA)
- DMA multiplexer (DMAMUX)
- TRGMUX
- Cyclic Redundancy Check (CRC)
- Software Watchdog (WDOG)



Memories
- 512KB Flash memory
- 64kB SRAM
- 4kB EEPROM

Communication Interfaces
- 3x LPUART
- 1x LPSPI
- 1x LPI2C
- 3x FlexCAN
- 1x FlexIO
  (configurable for UART, SPI, I2C & I2S)

Analog Peripherals
- 2x 12bit ADC w/ up to 16 channels
- 1x CMP w/ 8bit DAC

Timers Peripherals
- 4x FTM with 8 channels
- 2x PDB with 2 channels
- Low Power Interrupt Timer (LPIT)
- Low Power Timer (LPTMR)
- Real Time Clock (RTC)

S12ZVM: Overview

Internal RC osc.
- +/-1.3% over tmp

Window Watchdog
- independent RC osc

Sensor Supply
- 20mA I/O for sensor

2 UARTs
- ne linked to LIN Phy
- 2nd as test interface

Multiple timers
- IOC/periodic wakeup

CAN Option
- CAN controller

Two 12bit ADC
- List Based Architecture allows flexible definition of order and number of conversions
- 2.5µs conv. time

New S12Z CPU
- Up to 100MHz
- 32-bit ALU & 32-bit MAC unit,
- Optimized 32-bit math. operations

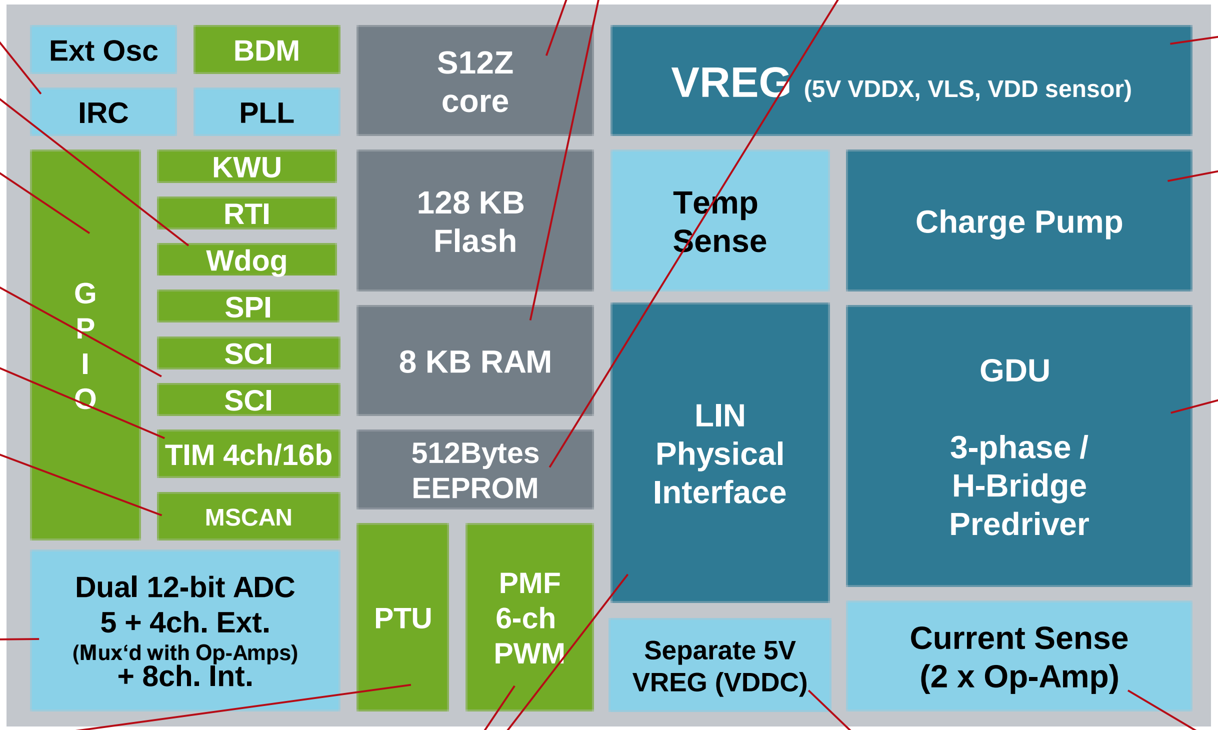
Safe RAM
ECC

Flash& EEPROM
- ECC
- Memory Protection
- Margin Read

Voltage Regulator
- Boost Option
- Support for external ballast transistor
- Vbat sense

Charge Pump
- Optional to support 100% duty cycle

Gate Drive Unit
- Operational from 3.5V to 26V
- Bootstrap circuit based
- 11V Vreg
- Phase comparators
- Desaturation comp. for HS/LS protection
- Under-/Over-voltage detection
- DC-link and phase voltage internally accessible on ADC
- Selectable HS/LS slew rate



Programmable Trigger Unit
- Synchronize ADC to PWM
- Trigger Command list with up to 32 triggers per cycle

PWM Module
- Complementary mode with deadtime ctrl.
- Fault protection
- Double-Switching

LIN Physical Interface
- 250kb/s fast mode
- +/-6kV

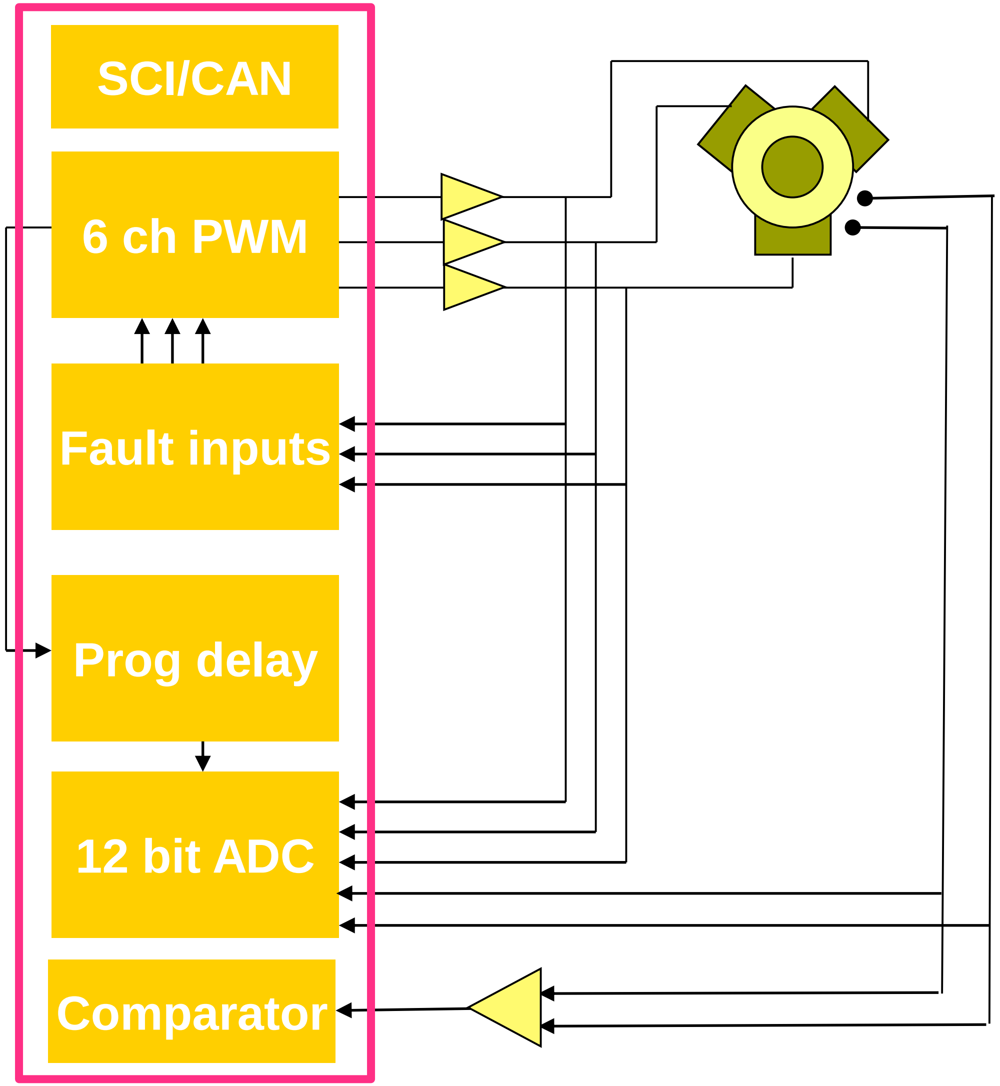
CAN Support
- 5V Vreg controller for external transceiver

Current Sense Amplifiers
- 2-shunt system supported with additional selectable over-current protection comparator

S32K144
MOTOR CONTROL SPECIFIC HIGHLIGHTS

Dedicated Peripherals Needed for 3ph Motor Control

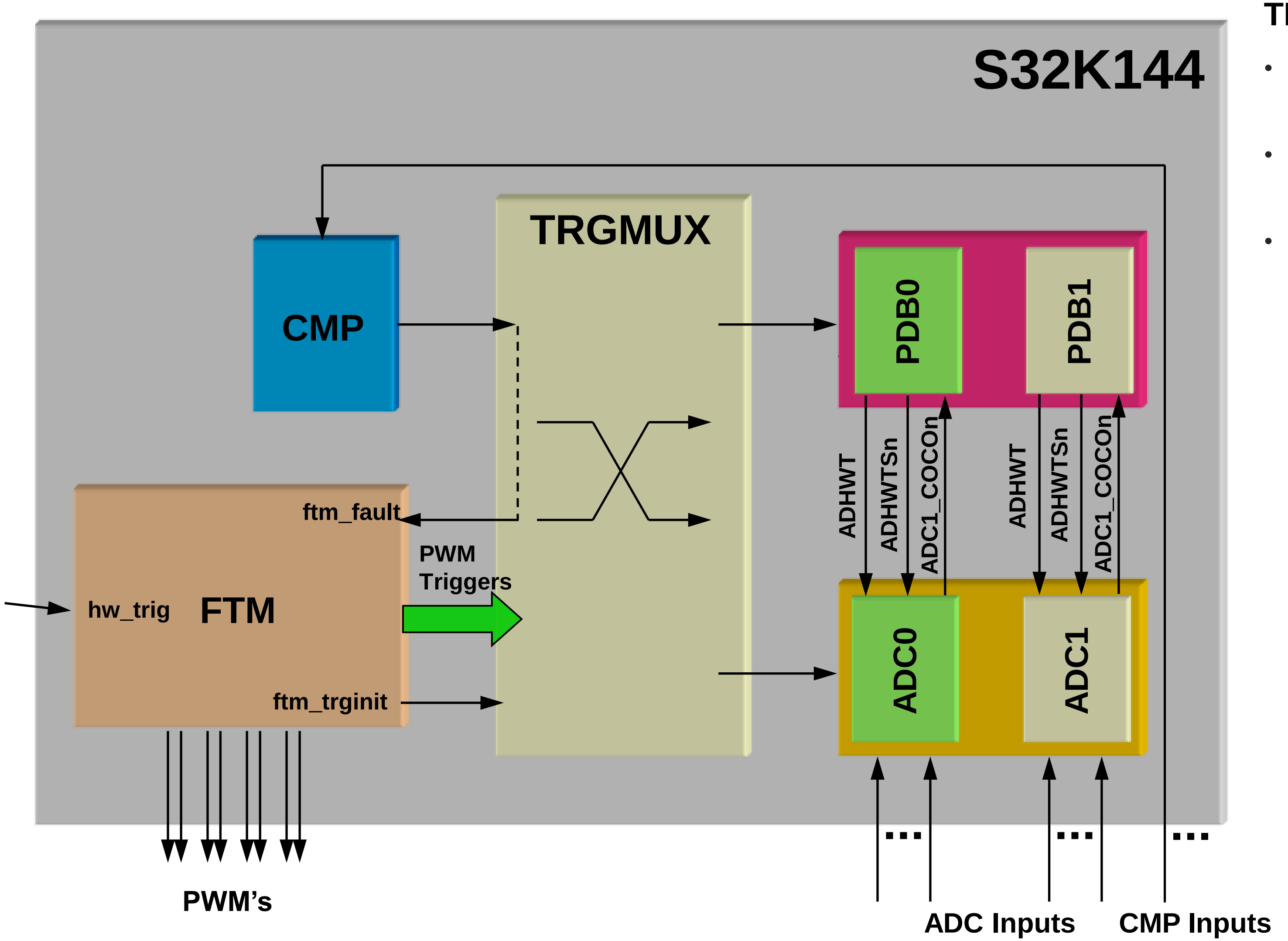
- **ADC Module**
  - We need to measure DC Bus voltage, Back-EM voltage, phase currents, DC Bus current, heatsink temperature
- **PWM module**
  - We need to generate 1 up 8 PWM according to motor type
- **Timer/Quadrature decoder**
  - We need to measure speed and rotor position from different sensors (hall sensors, quadrature encoder, tacho generator, sin/cos interface, etc.)
- **Built-in Comparator**
  - We need to detect fault conditions (over-current, over-voltage)
  - Allows to eliminate external comparators
  - Build in DAC allows SW control of fault level
- **User interface**
  - Communication interfaces, if required (SCI, SPI, CAN, I2C)
  - GPIO pins



S32K144: Motor Control Loop Implementation

CMP
- In MC as a fault protection unit
- Up to 8 channels some shared with ADC channels
- 8bit internal DAC

FTM0/1/2/3
- Various PWM modes
- Sync of double buffered registers
- Double buffered registers with various sync schemes
- Fault control
- SW Control & Masking
- Triggers generator for PDB or directly for ADC



TRGMUX
- Each peripheral has 32-bit trigger control register
- Each control register support up to 4 triggers
- Each trigger can be selected from up to 64 inputs.

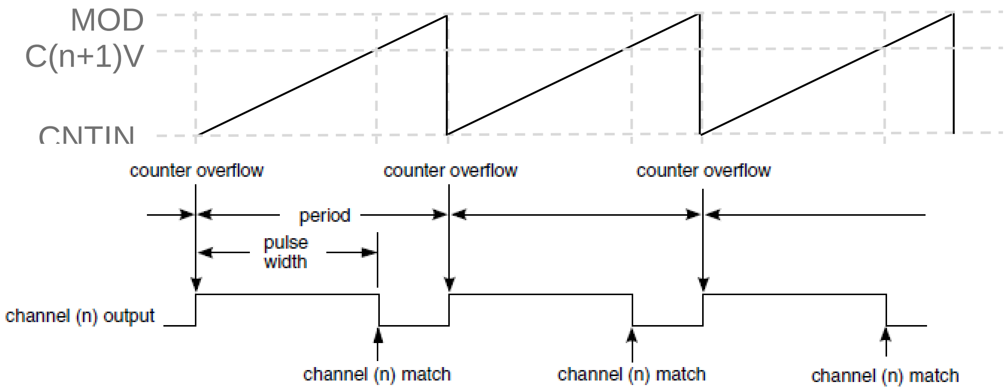
PDB0/1
- 16bit delay and triggering unit for ADC
- Two channels and each channel has 8 pre-triggers
- Back-to-back operation
- DMA support

ADC0/1
- 12bit resolution
- Up to 16 channels some of them interleaved & shared with analog CMP
- HW average function

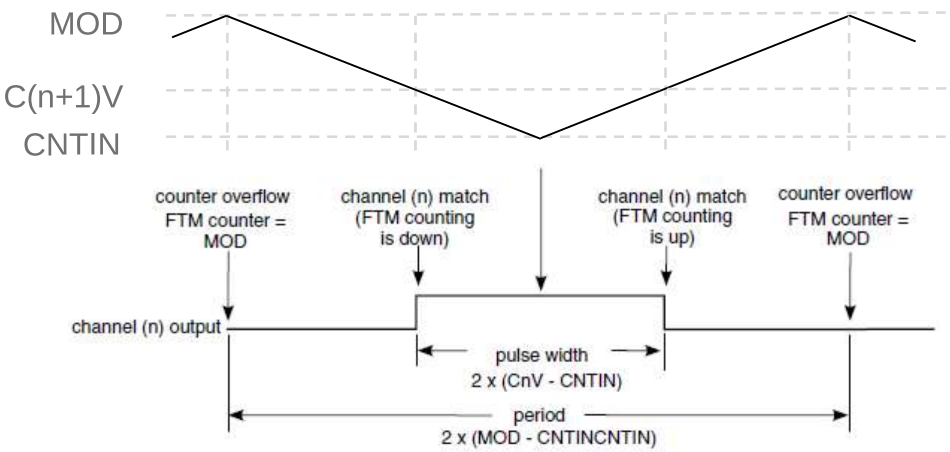
S32K144: FlexTimer module (FTM)

- **4 x FTM, with 8 channels** (inputs/outputs)
- FTM has a 16-bit counter
- **The counting can be up or up-down**
- Each channel can be configured for input capture, output compare, or PWM generation
- New **combined mode to generate a PWM signal** (with independent control of both edges of PWM signal)
- **Complementary outputs, include the deadtime insertion**
- **Software control masking of PWM outputs**
- Up to **4 fault inputs** for global fault control
- **The polarity of each channel is configurable**
- **Synchronized loading of write buffered FTM registers**
- **Write protection for critical registers**
- Backwards compatible with TPM
- **Quadrature decoder mode to process encoder signals**

Up counting mode - Edge-Aligned PWM
ELSnB:ELSnA = 1:0

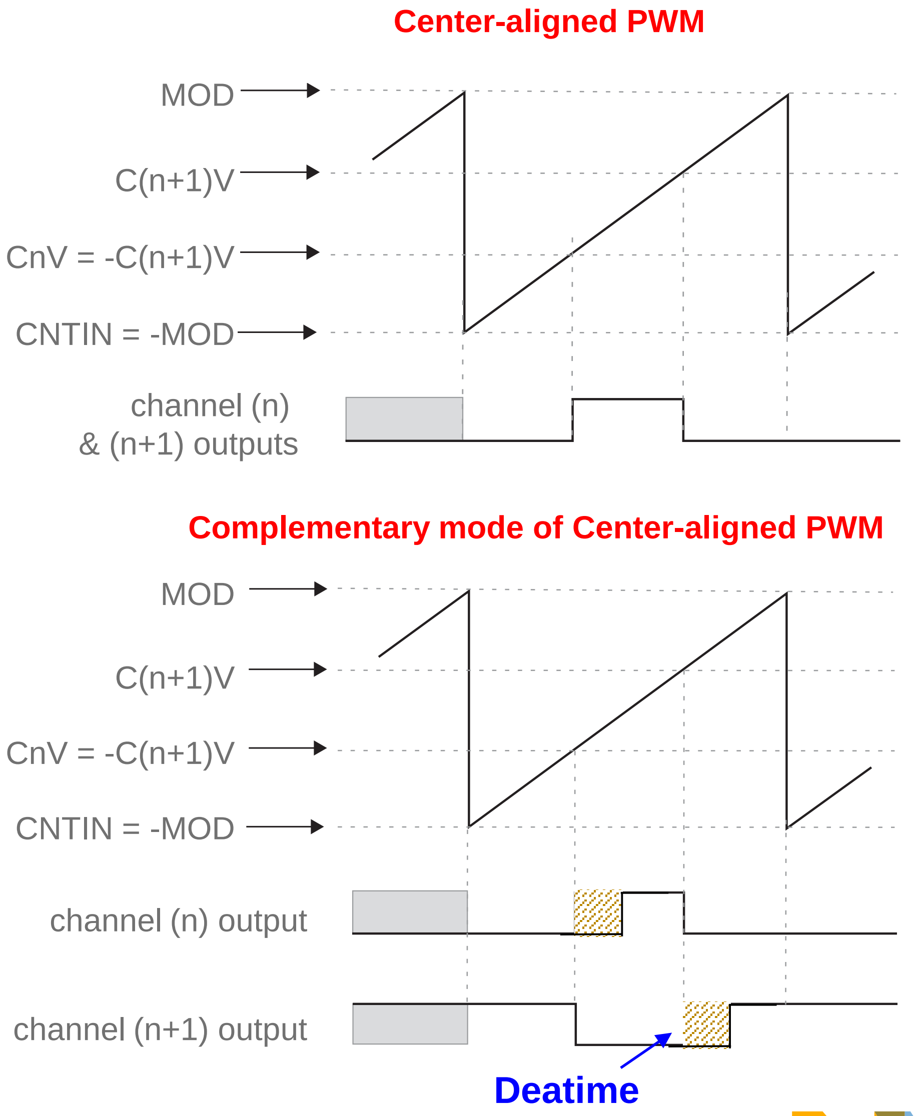


Up-down counting mode - Center-Aligned PWM
ELSnB:ELSnA = 1:0



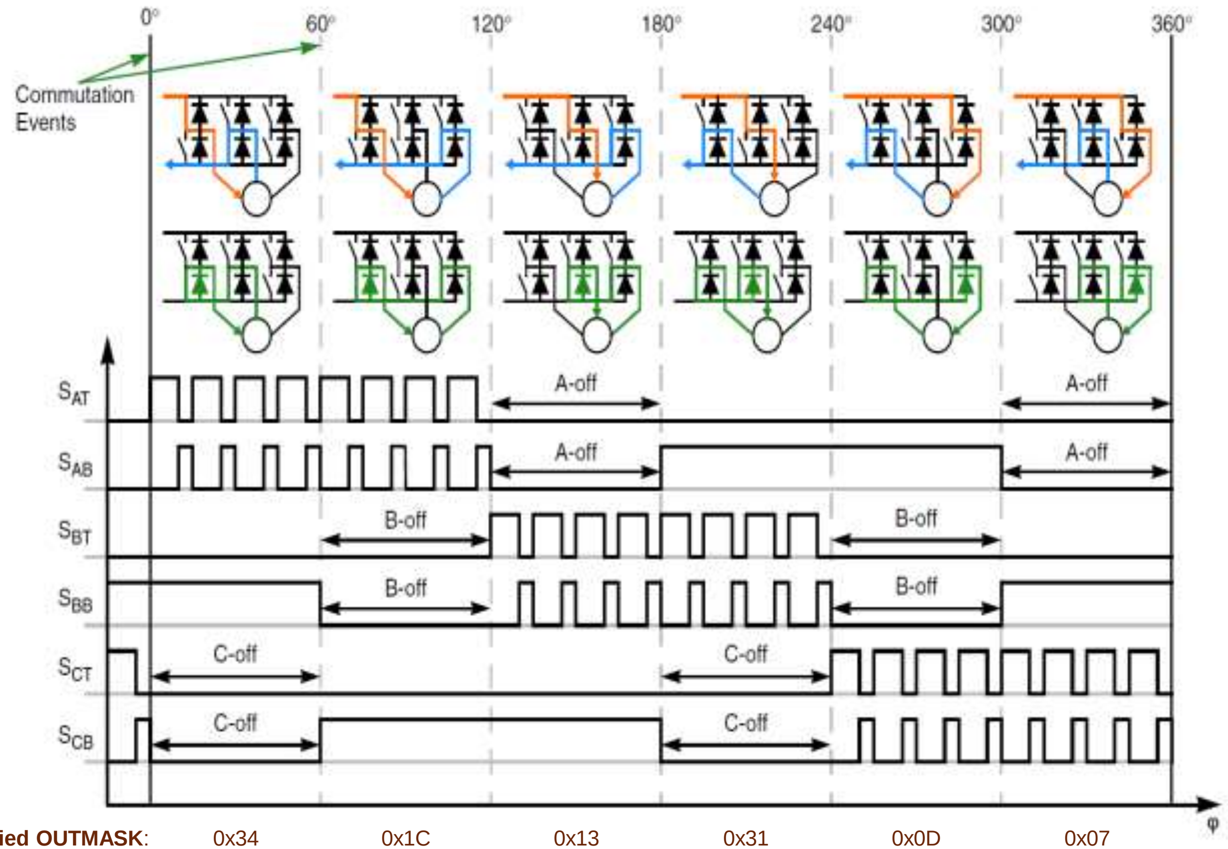
S32K144: FlexTimer module (FTM)

- **4 x FTM, with 8 channels** (inputs/outputs)
- FTM has a 16-bit counter
- The **counting** can be **up** or **up-down**
- Each channel can be configured for input capture, output compare, or PWM generation
- New **combined mode to generate a PWM signal** (with independent control of both edges of PWM signal)
- **Complementary outputs, include the deadtime insertion**
- **Software control masking of PWM outputs**
- Up to **4 fault inputs** for global fault control
- **The polarity of each channel is configurable**
- **Synchronized loading of write buffered FTM registers**
- **Write protection for critical registers**
- Backwards compatible with TPM
- **Quadrature decoder mode to process encoder signals**



S32K144: FlexTimer module (FTM)

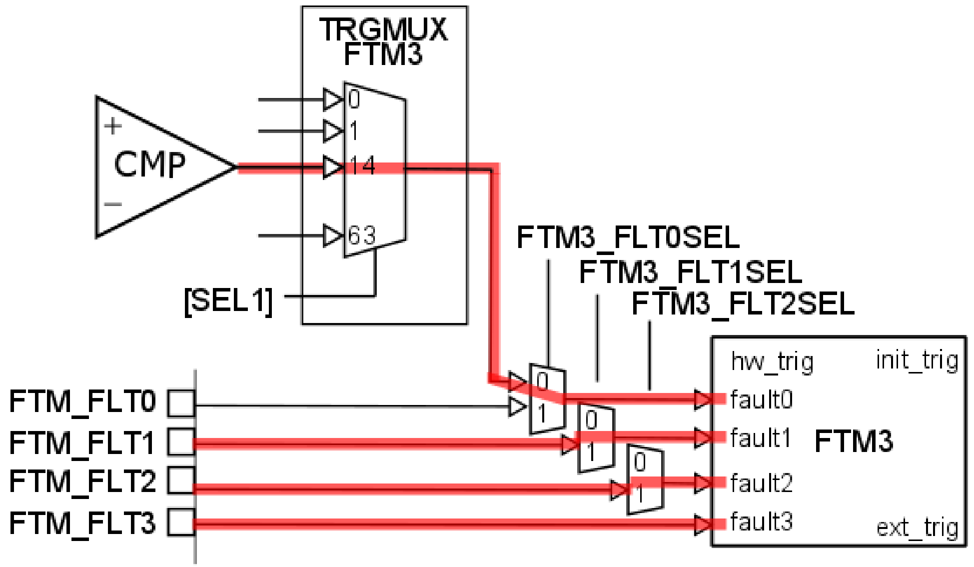
- **4 x FTM, with 8 channels** (inputs/outputs)
- FTM has a 16-bit counter
- The **counting** can be **up** or **up-down**
- Each channel can be configured for input capture, output compare, or PWM generation
- New **combined mode to generate a PWM signal** (with independent control of both edges of PWM signal)
- Complementary outputs, include the deadtime insertion
- **Software control masking of PWM outputs**
- Up to **4 fault inputs** for global fault control
- **The polarity of each channel is configurable**
- **Synchronized loading of write buffered FTM registers**
- **Write protection for critical registers**
- Backwards compatible with TPM
- **Quadrature decoder mode to process encoder signals**



Applied OUTMASK:	0x34	0x1C	0x13	0x31	0x0D	0x07
Applied SWOCTRL:	0x0808	0x2020	0x2020	0x0202	0x0202	0x0808

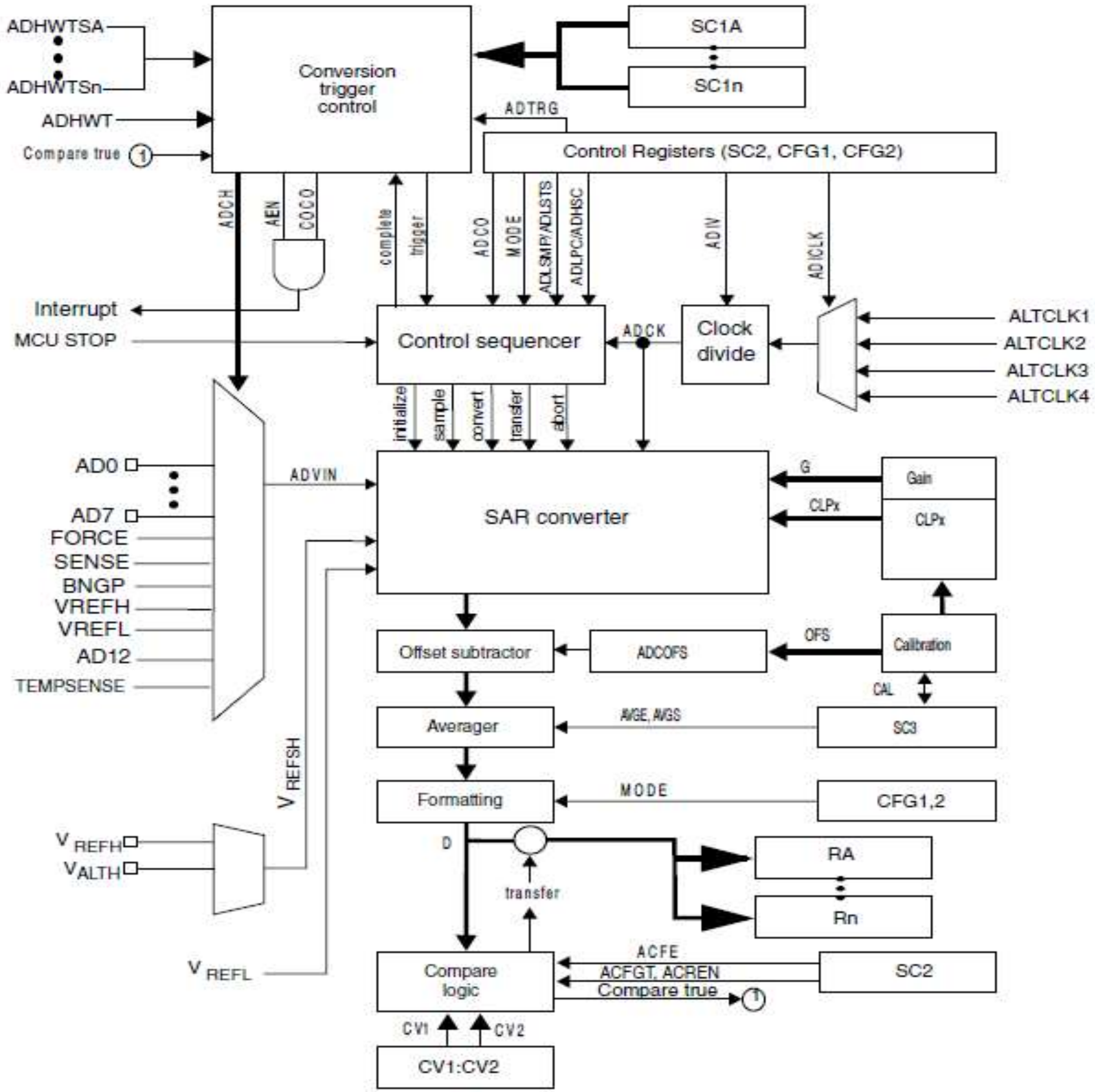
S32K144: FlexTimer module (FTM)

- **4 x FTM, with 8 channels** (inputs/outputs)
- FTM has a 16-bit counter
- The **counting** can be **up** or **up-down**
- Each channel can be configured for input capture, output compare, or PWM generation
- New **combined mode to generate a PWM signal** (with independent control of both edges of PWM signal)
- Complementary outputs, include the deadtime insertion
- **Software control masking of PWM outputs**
- **Up to 4 fault inputs for global fault control**
- **The polarity of each channel is configurable**
- **Synchronized loading of write buffered FTM registers**
- **Write protection for critical registers**
- Backwards compatible with TPM
- **Quadrature decoder mode to process encoder signals**



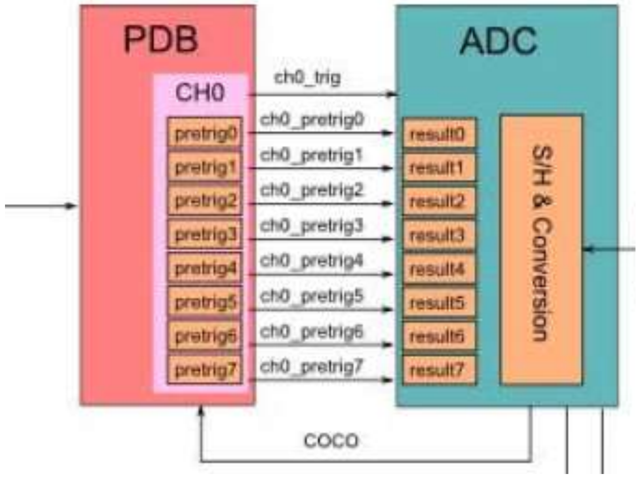
S32K144: Analog to Digital Converter (ADC)

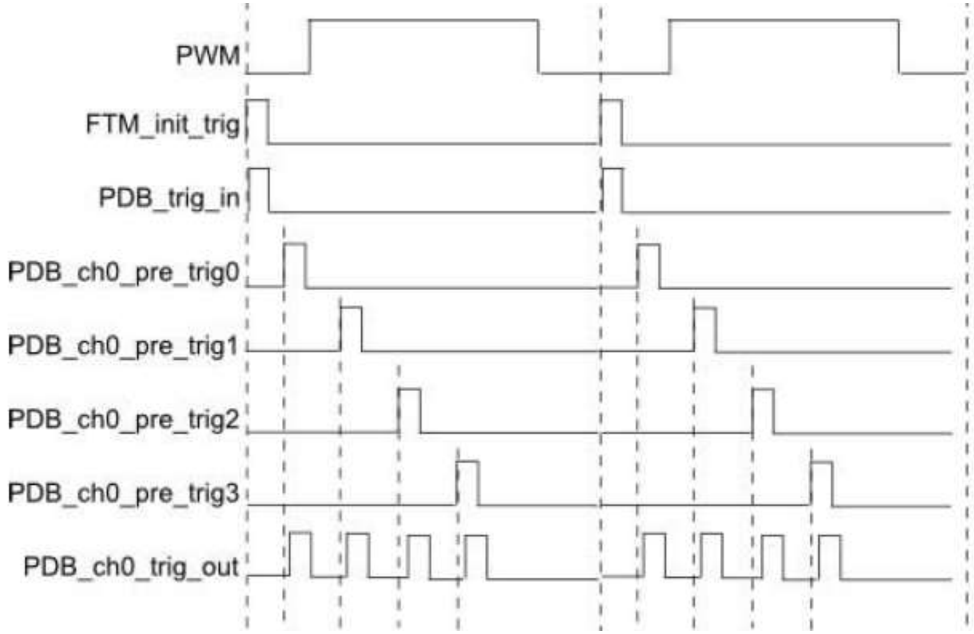
- Up to **16 single-ended** external analog inputs some of them can be **hardware interleaved**:
  - ADC0_SE4 and ADC1_SE14
  - ADC0_SE5 and ADC1_SE15
  - ADC1_SE8 and ADC0_SE8
  - ADC1_SE9 and ADC0_SE9
- **Self-Calibration** mode
- Single or continuous conversion
- **Hardware average** function
- **Automatic compare** with interrupt for less-than, greater-than or equal-to, within range, or out-of-range, programmable value
- Programmable sample time and conversion speed/power (For short sample and **50 MHz frequency T_{conv} = 0.99us**)
- **DMA** support



S32K144: Programmable Delay block (PDB)

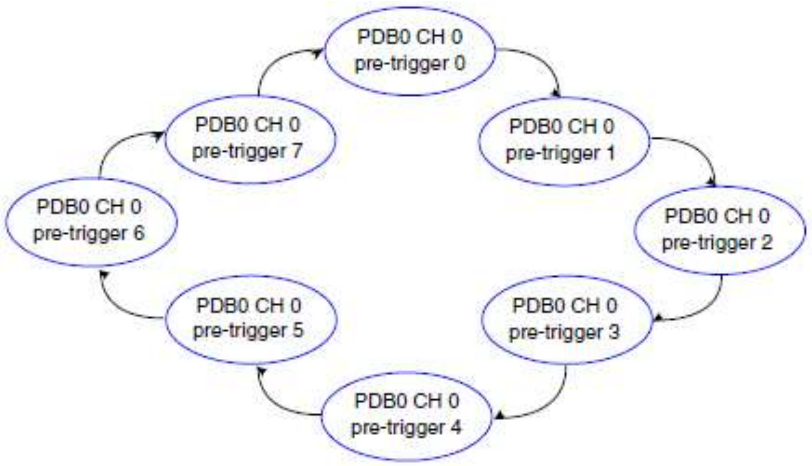
- One PDB is associated with one ADC
- **Each PDB has up to 8 pre-trigger outputs to trigger ADC channels independently**
- One **16-bit delay register** per pre-trigger output
- Optional bypass of the delay registers of the pre-trigger outputs
- Operation in One-Shot or Continuous modes
- Optional **back-to-back mode** operation
- One programmable delay interrupt
- One sequence error interrupt
- One channel flag and one sequence error flag per pre-trigger
- **DMA** support
- Up to **8 pulse outputs** (pulse-out's)



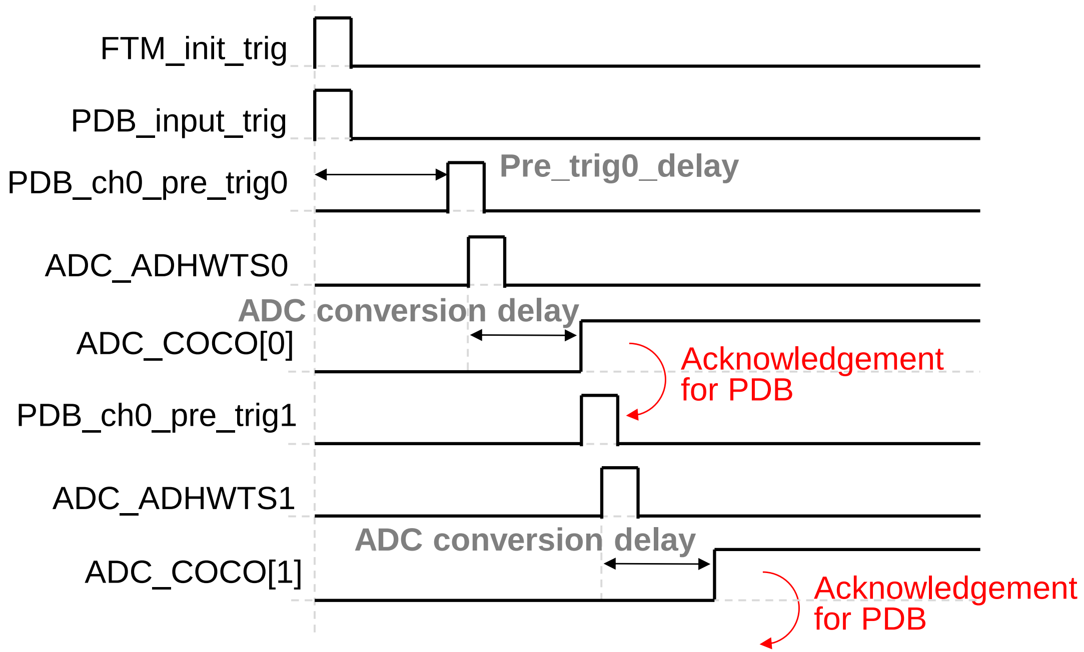


S32K144: Programmable Delay block (PDB)

- Up to **2 trigger input** sources and one software trigger source
- One PDB is associated with one ADC
- Each PDB has up to **8 pre-trigger** outputs to trigger ADC channels **independently**
- One **16-bit delay register** per pre-trigger output
- Optional bypass of the delay registers of the pre-trigger outputs
- Operation in One-Shot or Continuous modes
- **Optional back-to-back mode operation**
- One programmable delay interrupt
- One sequence error interrupt
- One channel flag and one sequence error flag per pre-trigger
- **DMA** support
- Up to **8 pulse outputs** (pulse-out's)

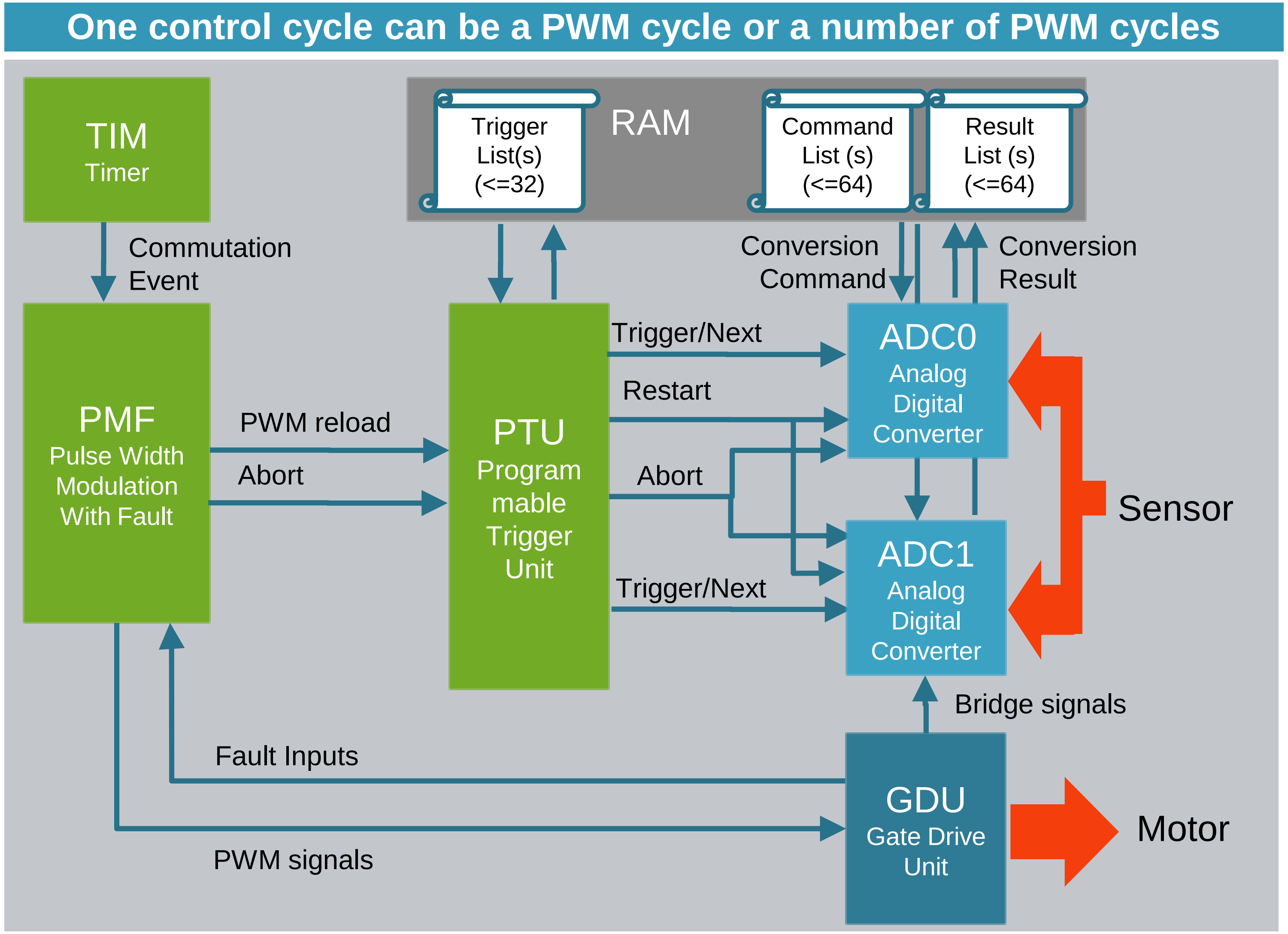


Back-to-Back Operation



S12ZVM
MOTOR CONTROL SPECIFIC HIGHLIGHTS

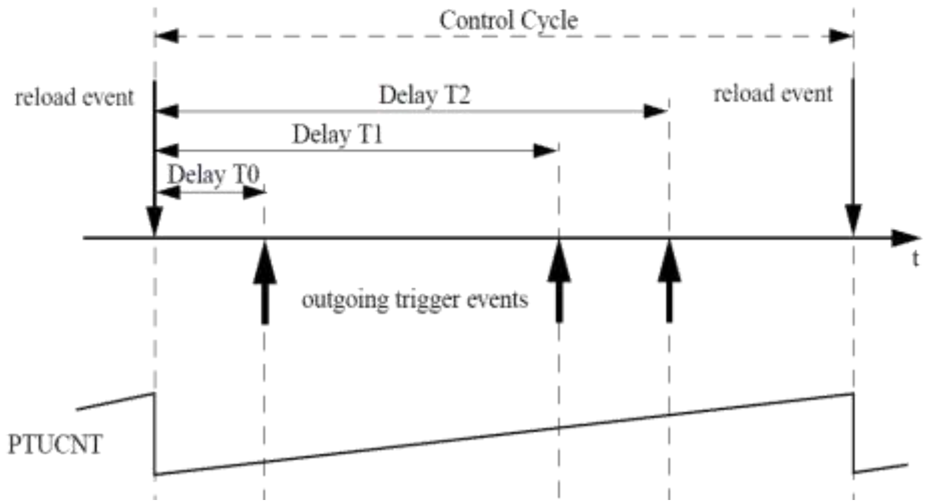
S12ZVM: Motor Control Loop Implementation

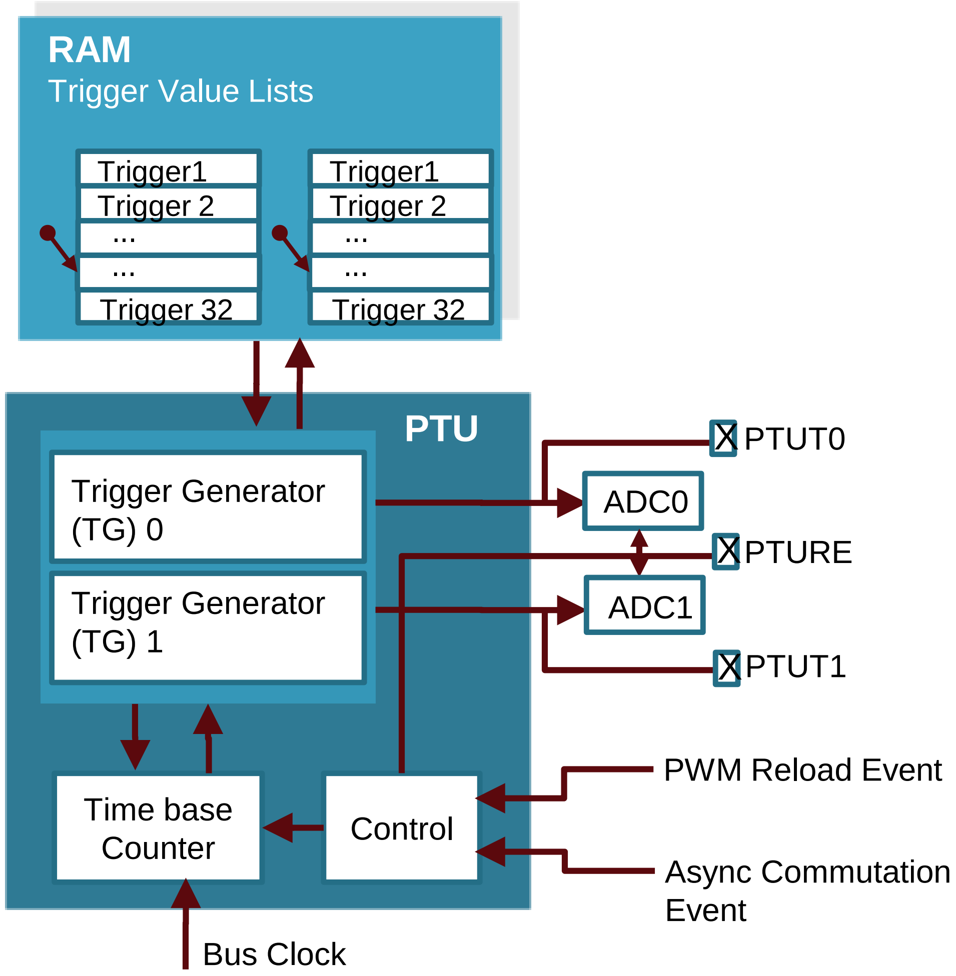


S12ZVM: Programmable Trigger Unit (PTU)

Completely avoids CPU involvement to trigger ADC during the control cycle

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



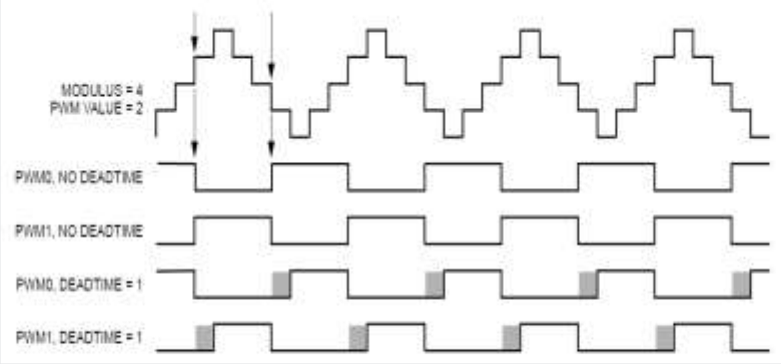


S12ZVM: Pulse Width Modulator Module (PMF)

- **6 PWM channels, 3 independent counters**
  - Up to 6 independent channels or 3 complementary pairs
- **Based on core clock (max. 100MHz)**
- **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**
- **Programmable fault protection**

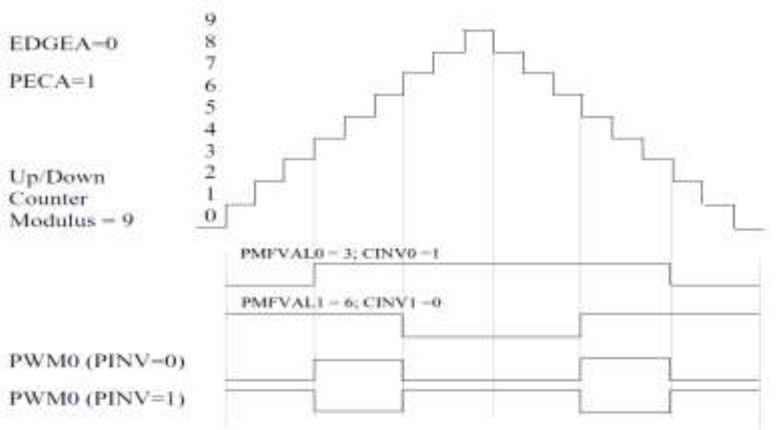
Complementary Mode

with / without dead time insertion



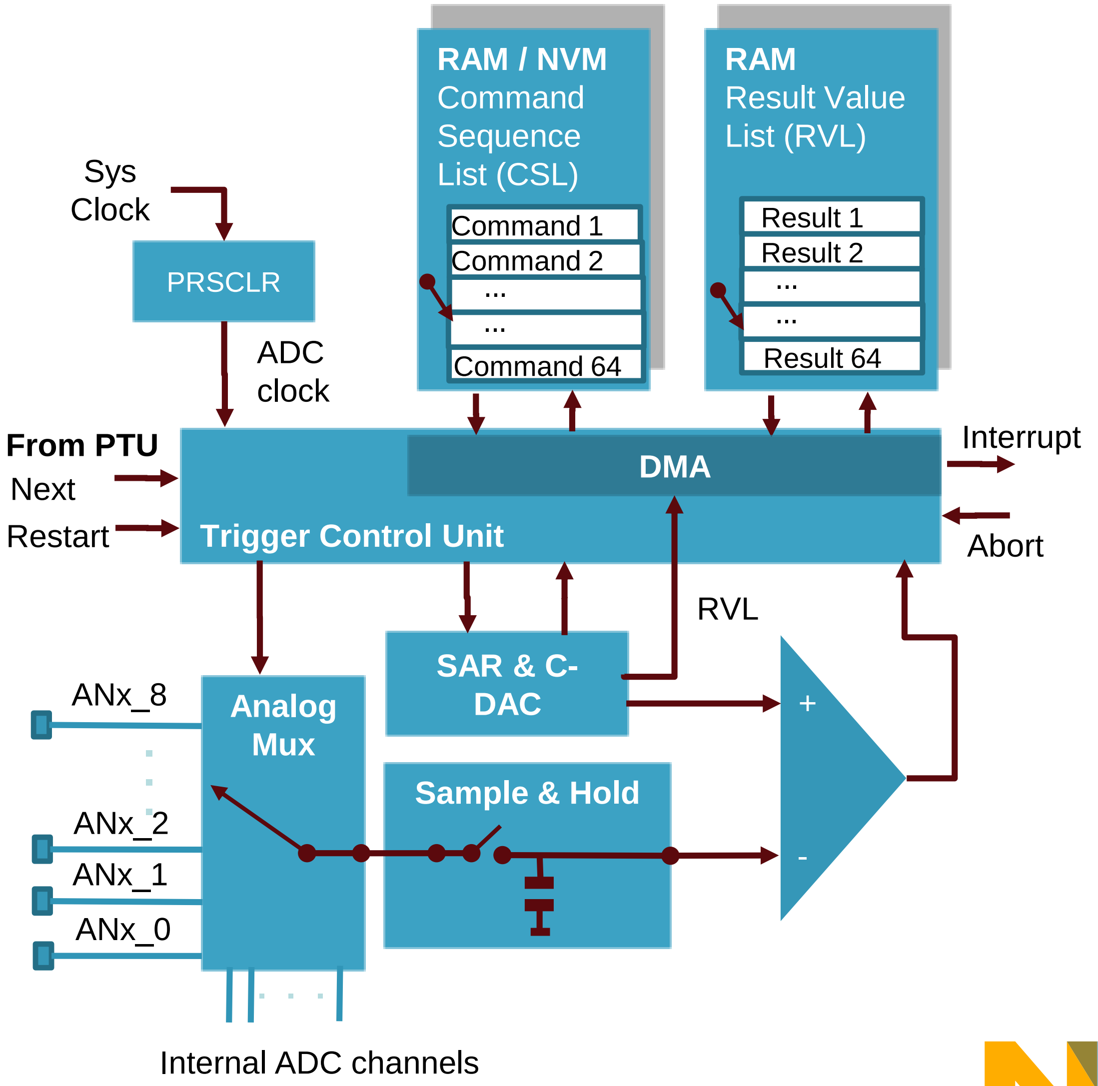
Double-Switching Mode

for single shunt system



S12ZVM: 12-bit SAR Analog-to-Digital (ADC)

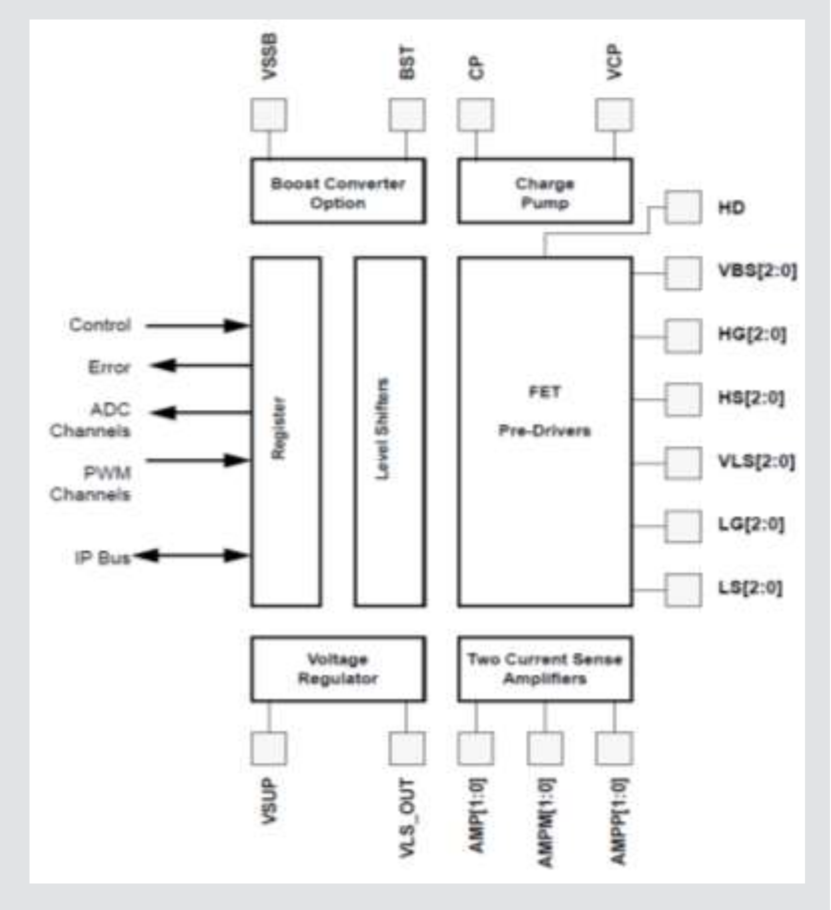
- **2 independent converters:**
  - ADC0 (5 ext ch. + 5 int. ch.)
  - ADC1 (4 ext ch. + 4 int. ch.)
- 2.5µsec conversion time
- **List Based Architecture**
  - Double buffered lists -> CPU can load new values in the background
  - Flexible conversion sequence definition and oversampling.
- Can be triggered by PTU, for accurate synch with PWM
- DMA taking commands from SRAM /NVM and storing results back into SRAM



S12ZVM: Gate Driver Unit (GDU) – Overview

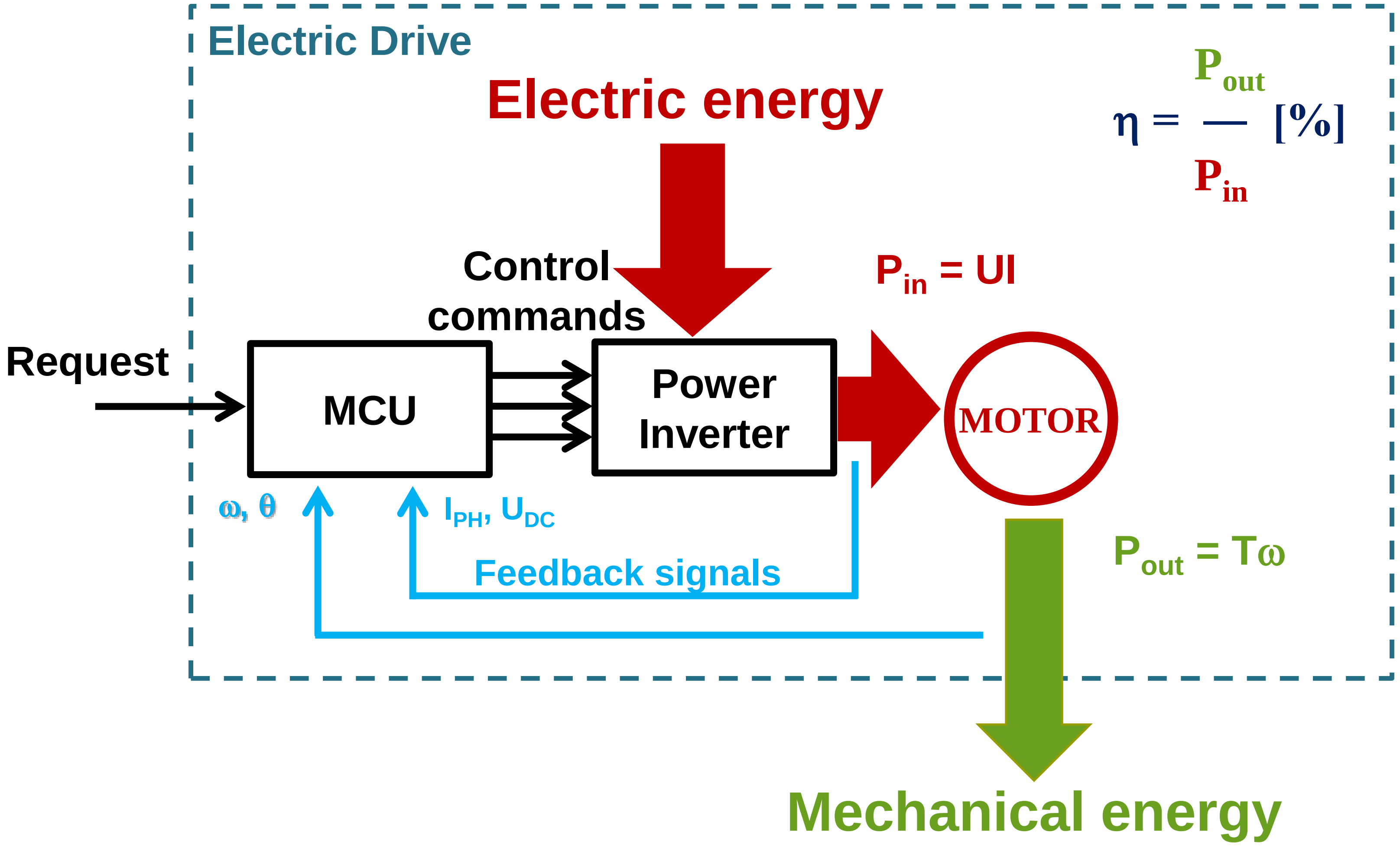
FET pre-driver for 6 N-ch power MOSFETs (3 high-side, 3 low-side)

- 11V regulator to drive external FETs VGS
- Bootstrap circuit for high-side drivers
- Optional charge pump to support static high-side driver operation
- Phase comparators to signal BEMF zero crossing
- Option to route DC Link (HD) or Phase voltage measurement to ADC
- Two current sense amplifiers feeding ADC
- Over- /under- voltage monitoring
- Short circuit protection by monitoring VDS for both LS/ HS
- Step-up (boost) converter option for low supply voltage operation



MOTOR CONTROL TECHNIQUES

Electric Drive General Concept

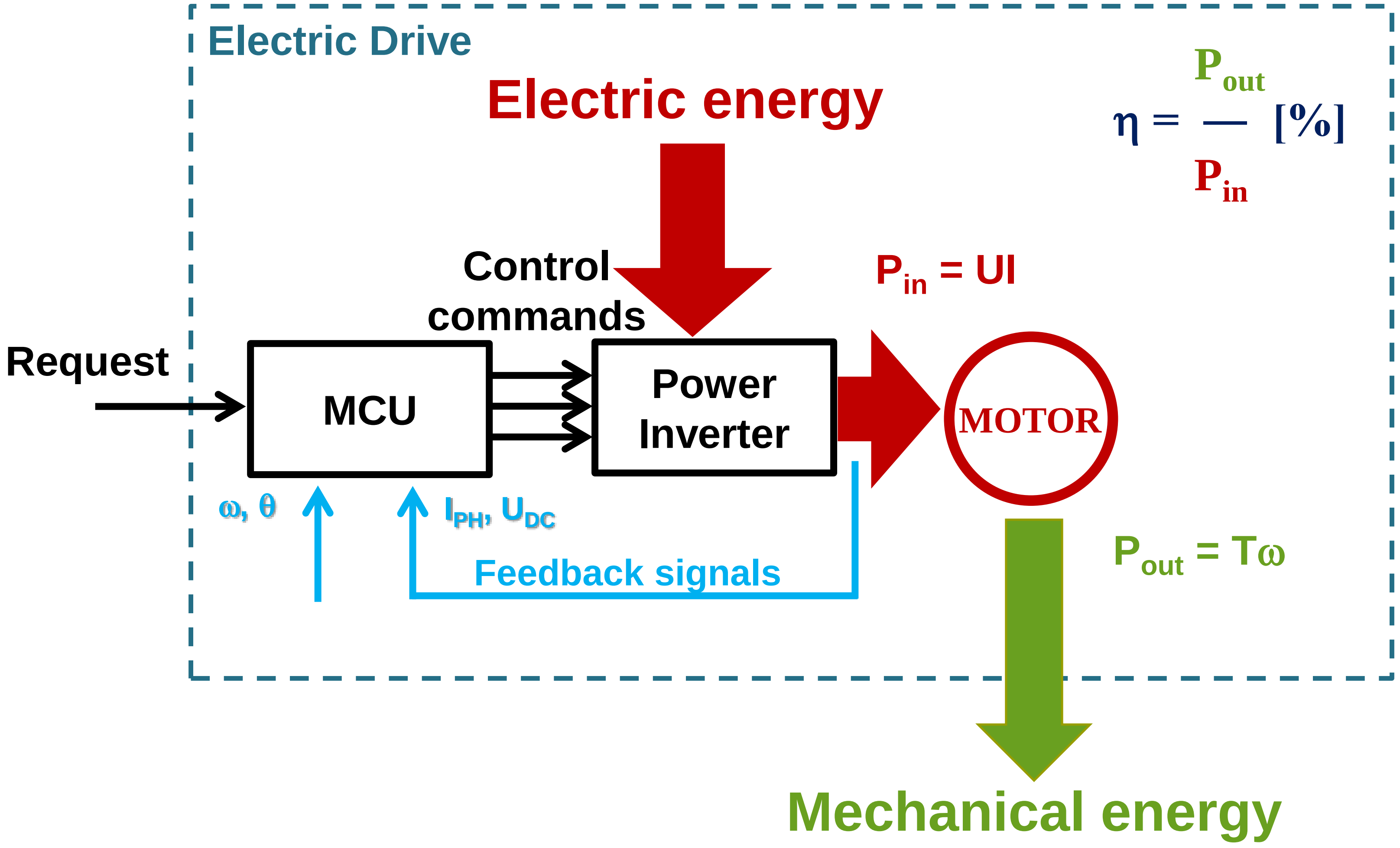


PM motor control strictly requires the information about the actual rotor position and speed.

According to the rotor position and speed loop:

- Sensored control (resolver, encoder, hall ...)
- Sensorless control

Electric Drive General Concept



PM motor control strictly requires the information about the actual rotor position and speed.

According to the rotor position and speed loop:

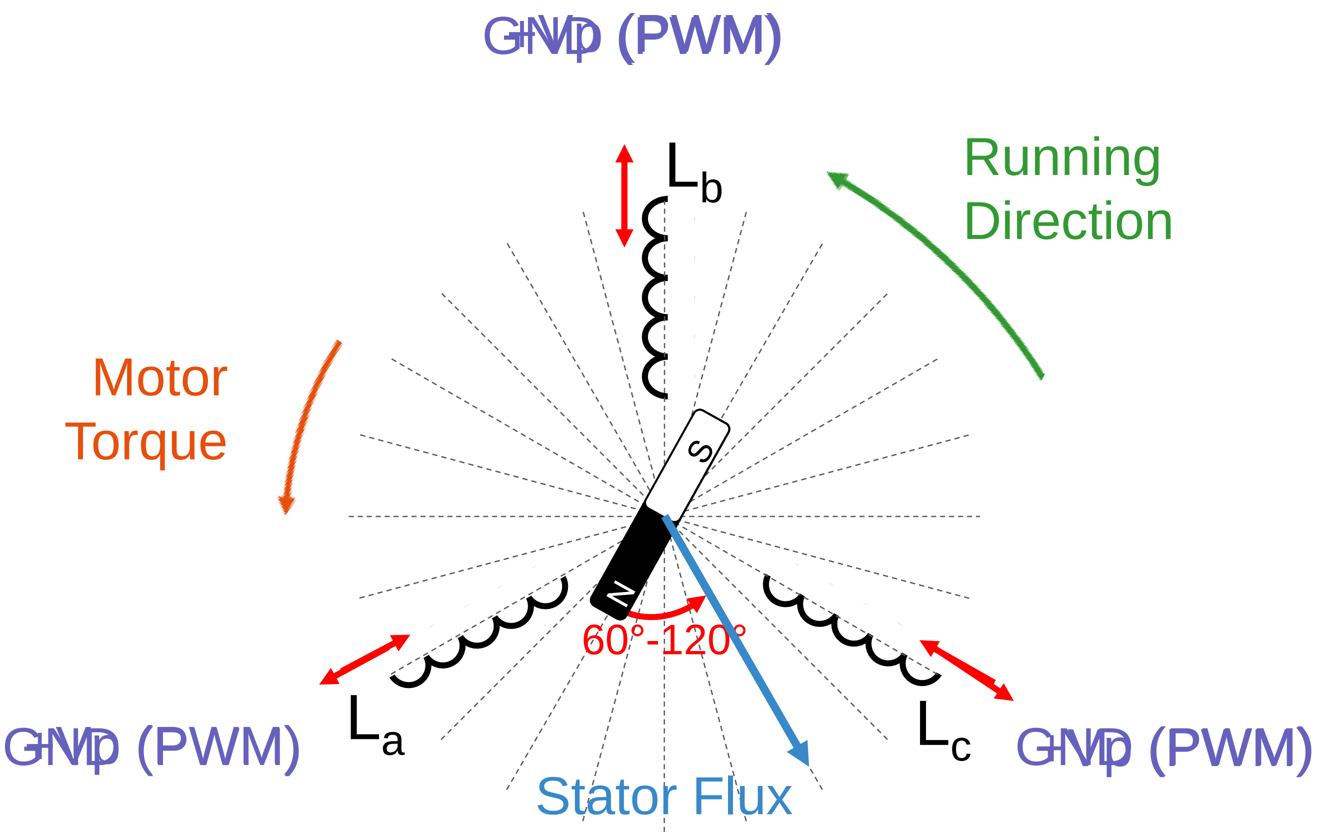
- Sensored control (resolver, encoder, hall ...)
- **Sensorless control**

Our Main Focus Today!

BLDC 6-STEP COMMUTATION CONTROL

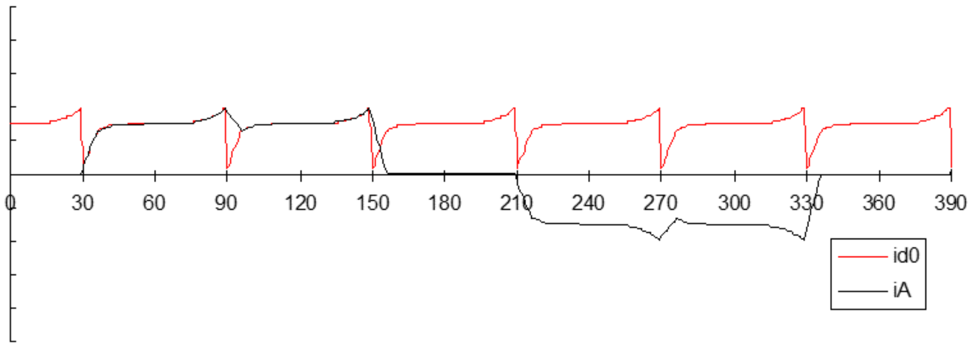
SENSORLESS

BLDC 6-Step Commutation Principle



- Stator field is generated between 60° to 120° to rotor field to get maximal torque (@ 90°) and energy efficiency

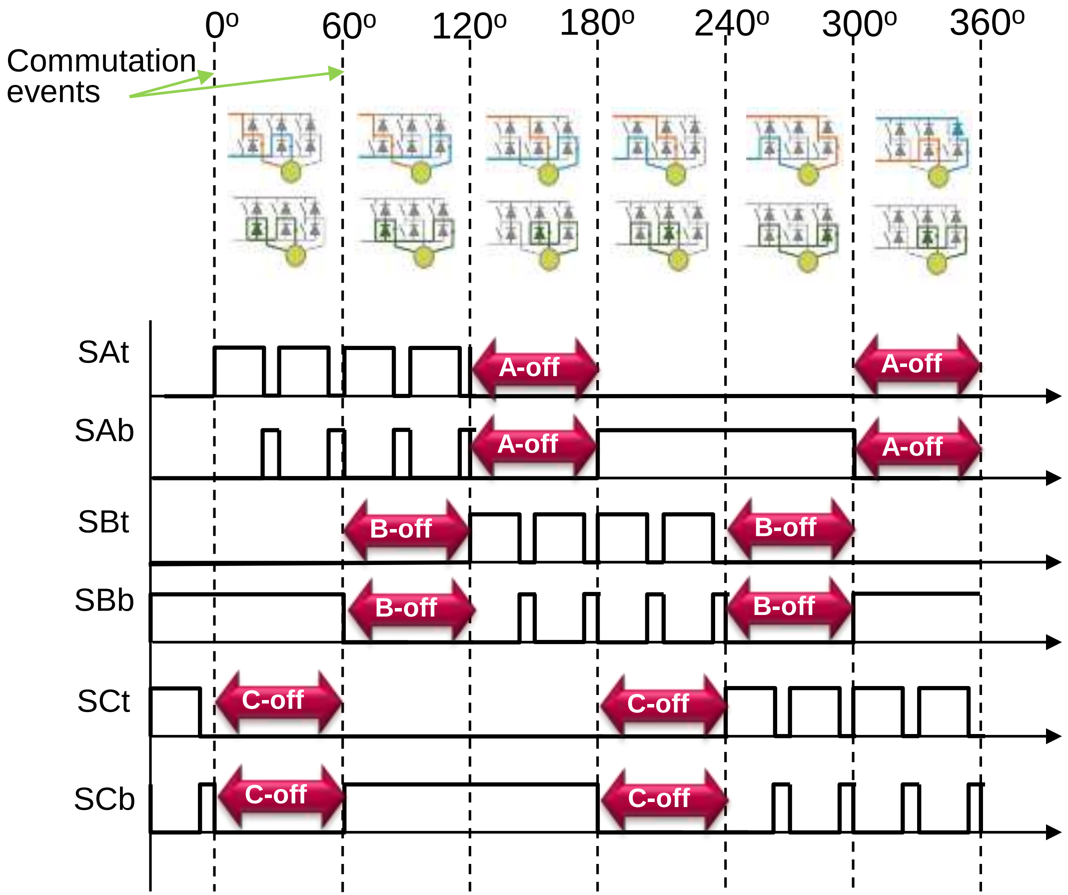
- Six resulting flux vectors defined by the six voltage vectors to create rotation

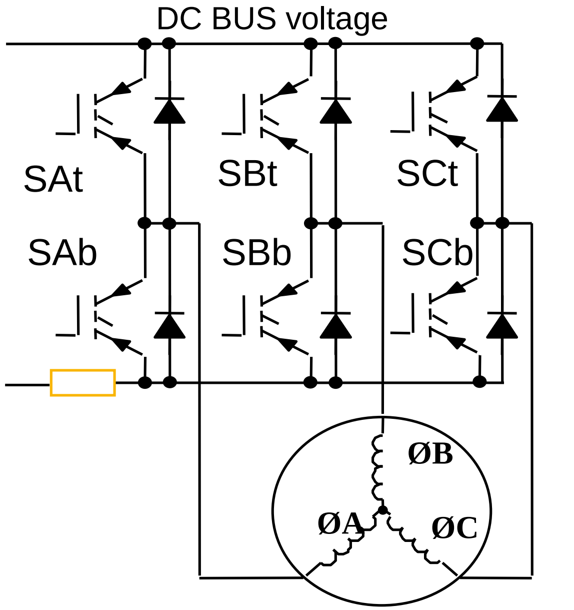


Current behavior during the commutation

- Six Step: 0° => 60° => 120° => 180° => 240° => 300° => 360° => 0° => ...

Complementary/Idependent Unipolar PWM Switching



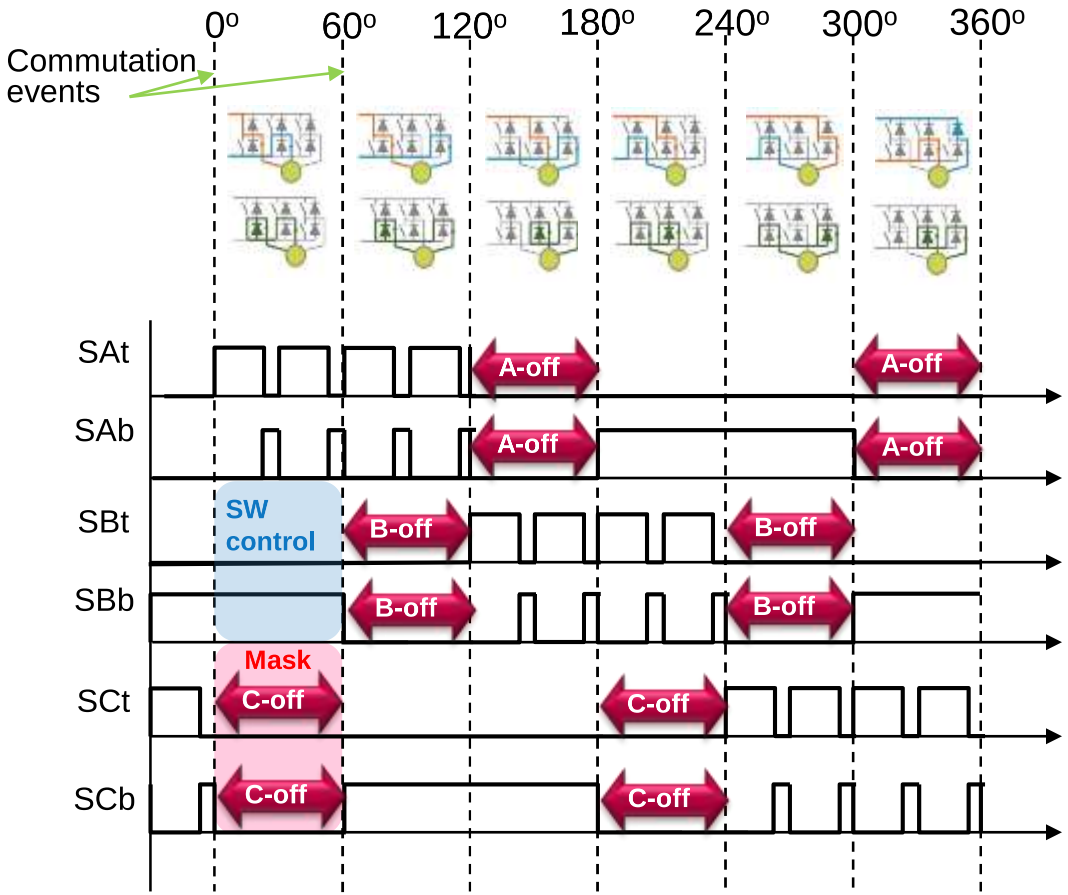


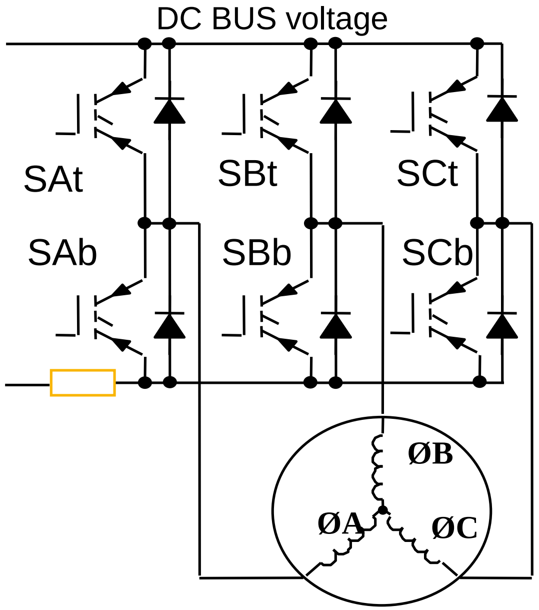
One phase powered by complementary PWM signal,
second phase grounded:

- Low MOSFET switching losses
- Low EMC noise

Complementary/Idependent Unipolar PWM Switching

- Apply SW control to grounded phase
- Apply Mask to disconnected phase

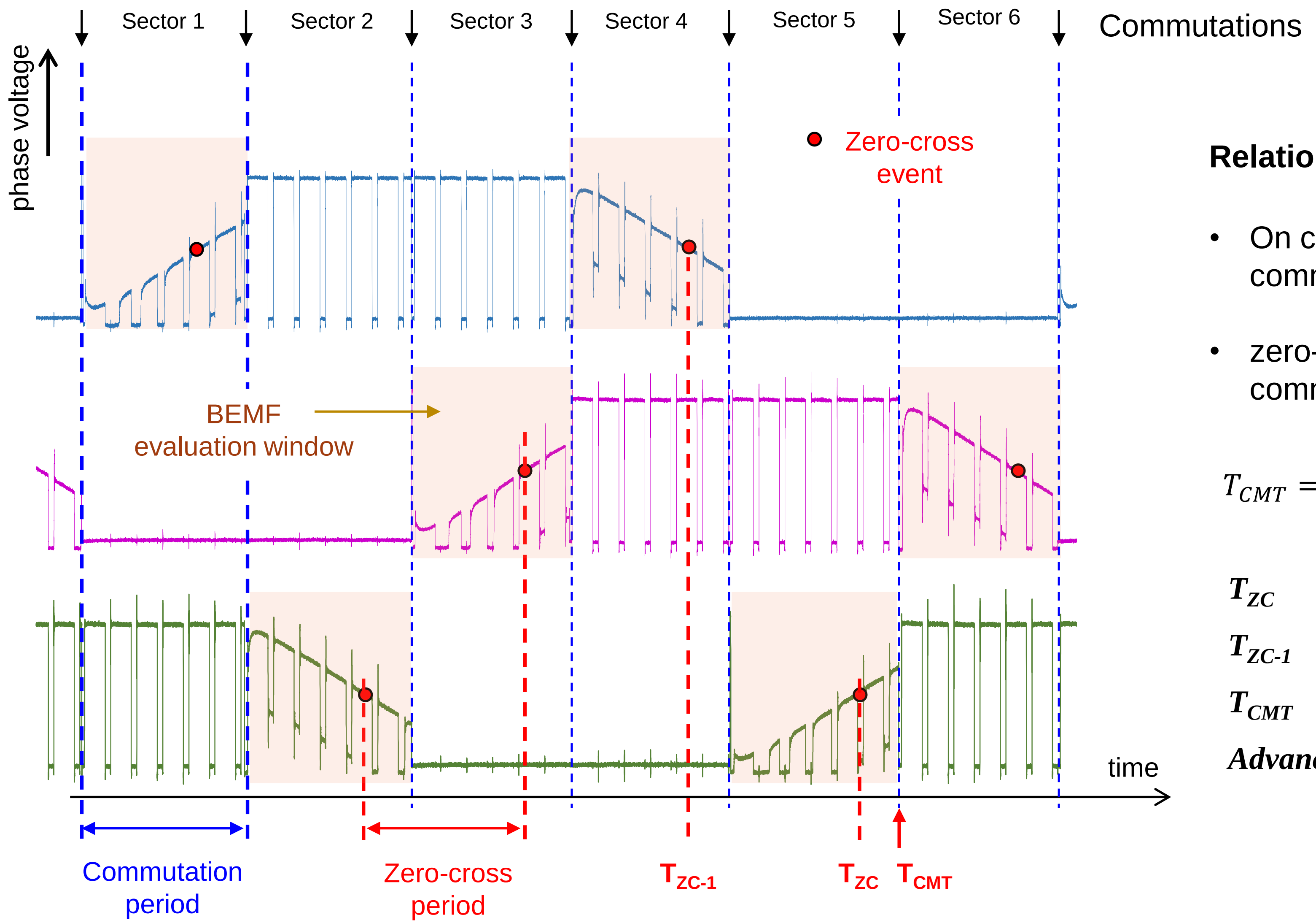




One phase powered by complementary PWM signal,
second phase grounded:

- Low MOSFET switching losses
- Low EMC noise

Back-EMF Zero-Cross Events and Commutations



Relationships:

- On constant rotor speed: commutation period = zero-cross period
- zero-cross event occurs in the middle of two commutations (on ideal motor)

$$T_{CMT} = T_{ZC} + AdvanceAngle \frac{(T_{ZC} - T_{ZC-1})}{2}$$

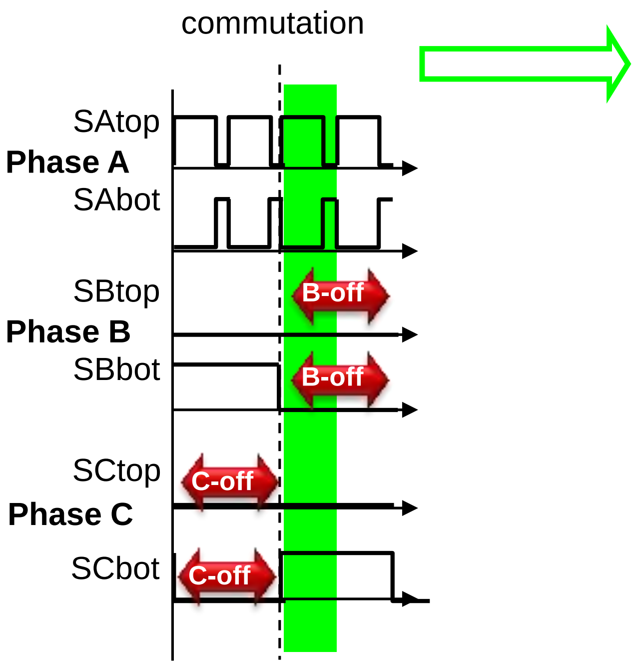
T_{ZC} – time of actual zero-cross

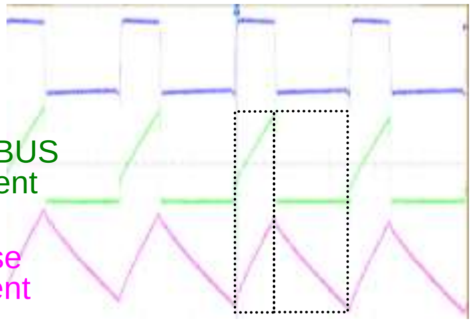
T_{ZC-1} – time of previous zero-cross

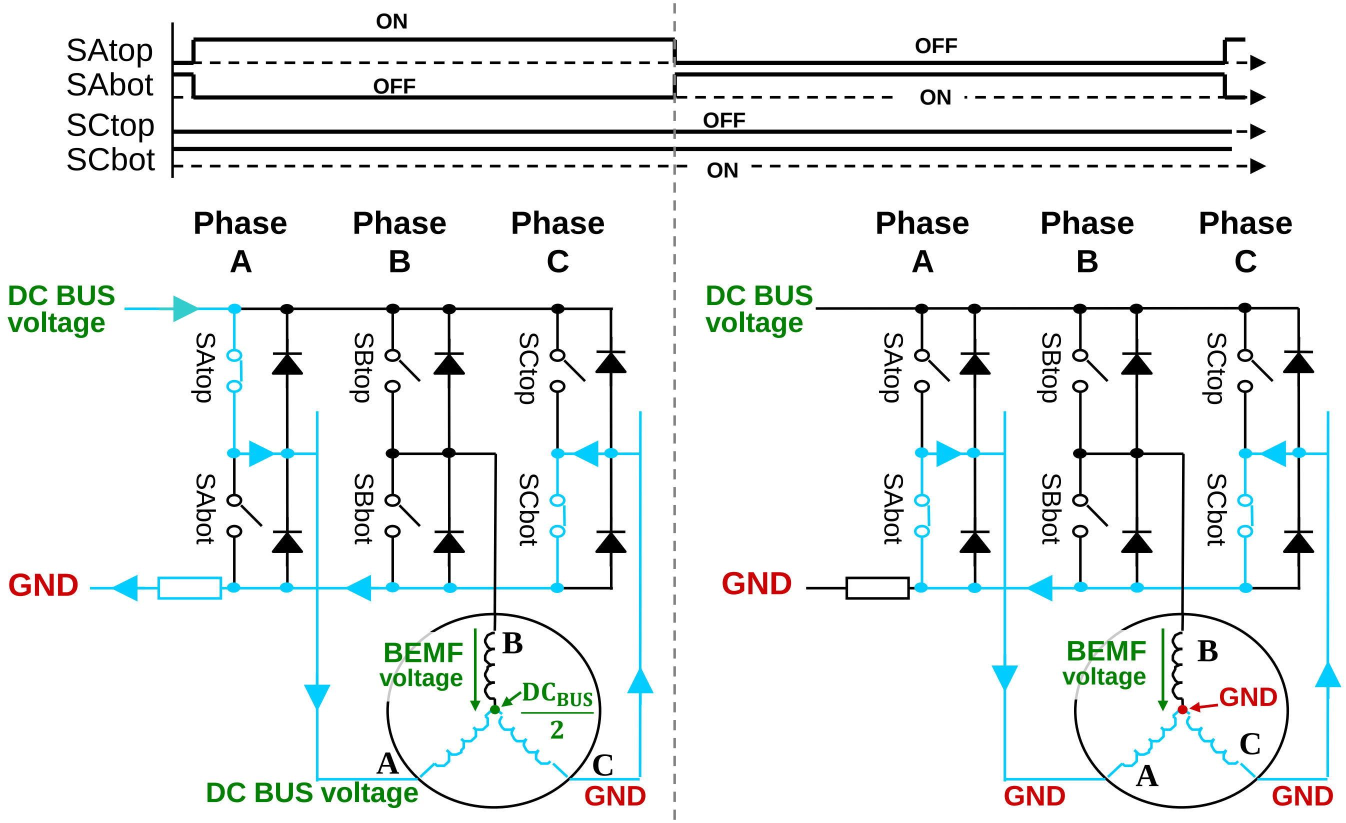
T_{CMT} – time of next commutation

AdvanceAngle - constant in the range 0.7 to 0.95 (depends on motor parameters)

Measurement During PWM Switching





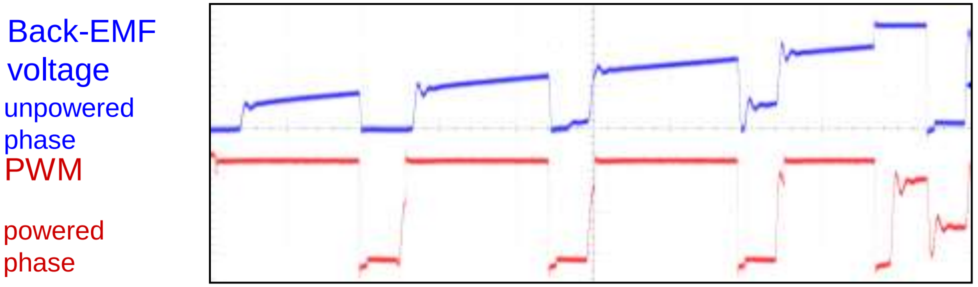


Top MOSFET is ON:
+ Phase current can be measured by DC BUS shunt resistor
+ Back-EMF voltage can be measured both positive and negative

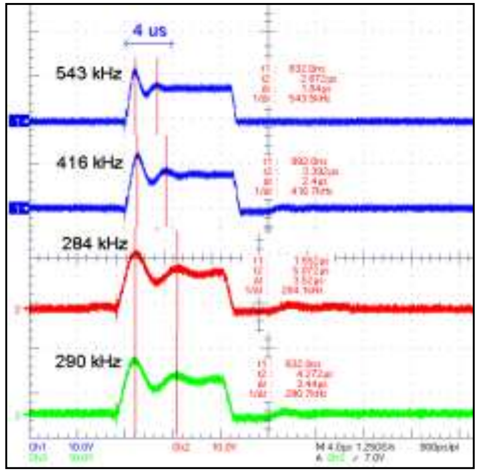
Top MOSFET is OFF:
– Phase current can **NOT** be measured by DC BUS shunt resistor
– **Only positive** Back-EMF voltage can be measured (**zero-cross can not be precisely measured**)

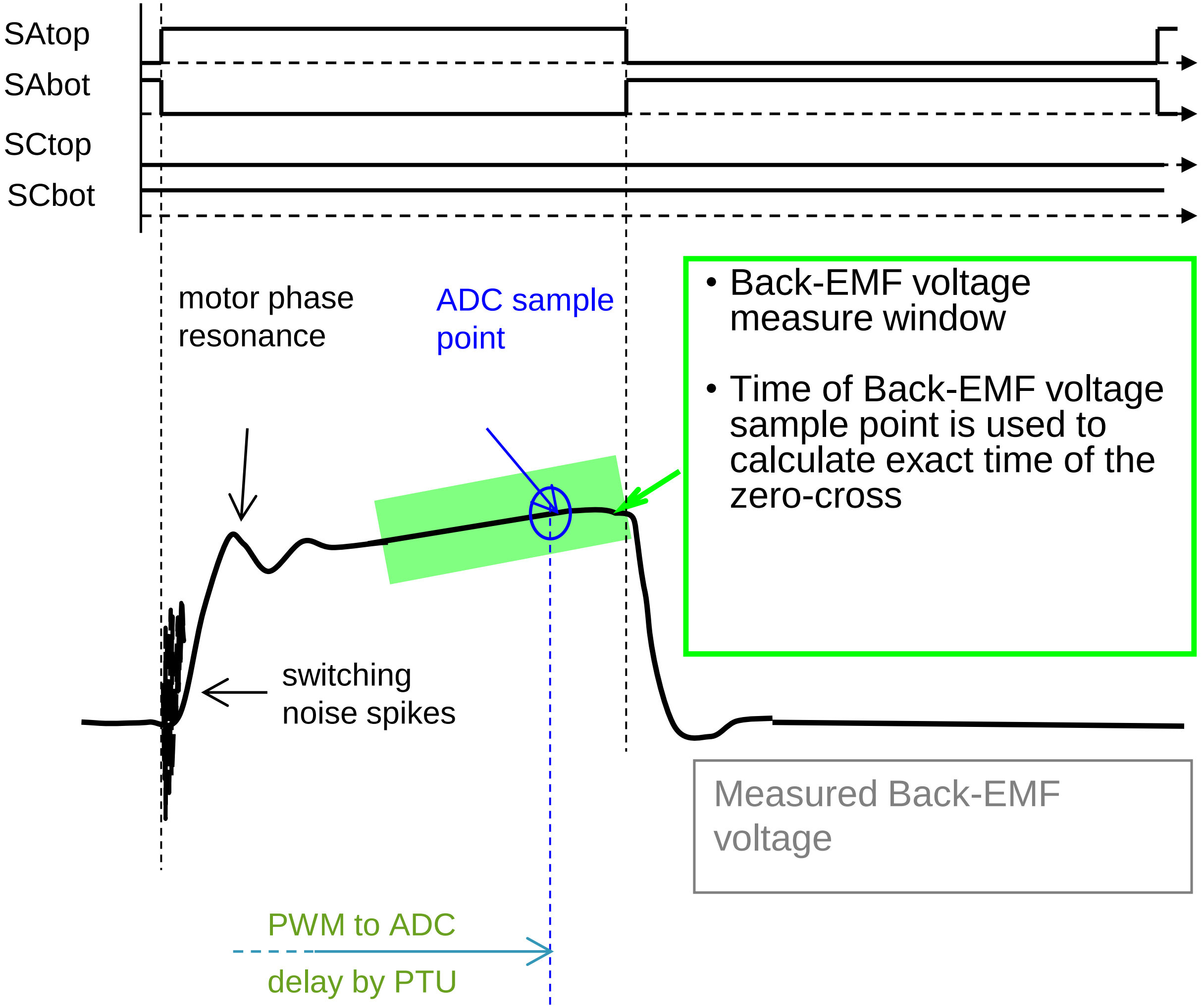
Back-EMF Voltage Measurement

Back-EMF voltage can not be measured within all the active PWM pulse as there is switching noise and resonance transient at the beginning of the PWM pulse

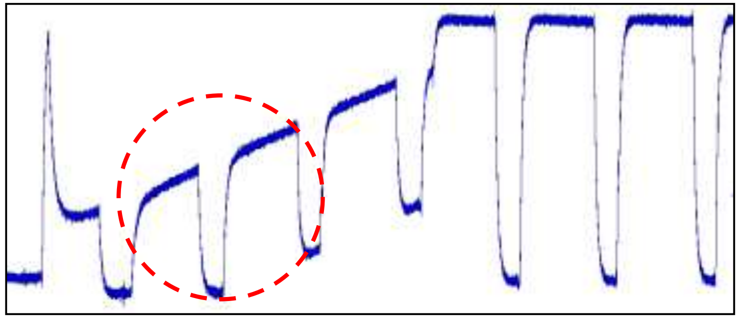


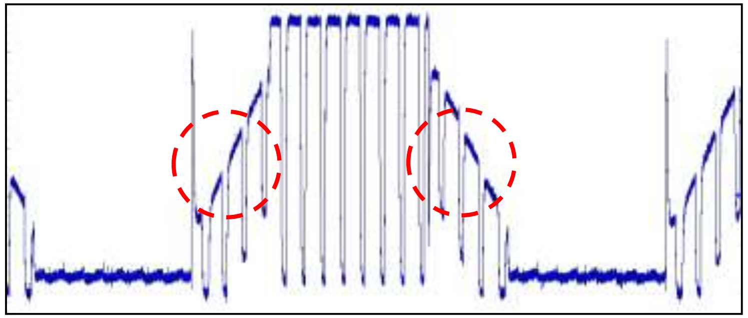
Resonance transient on Back-EMF voltage depends on motor and power stage parameters

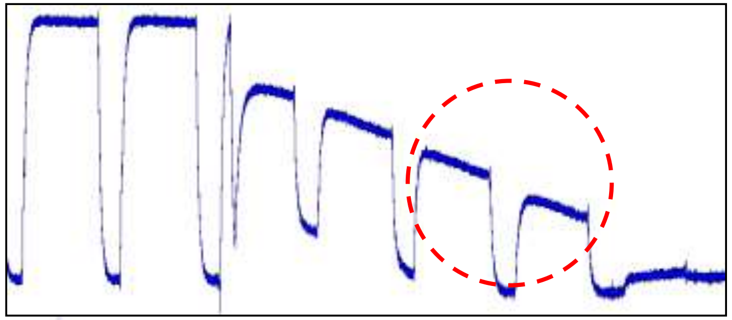


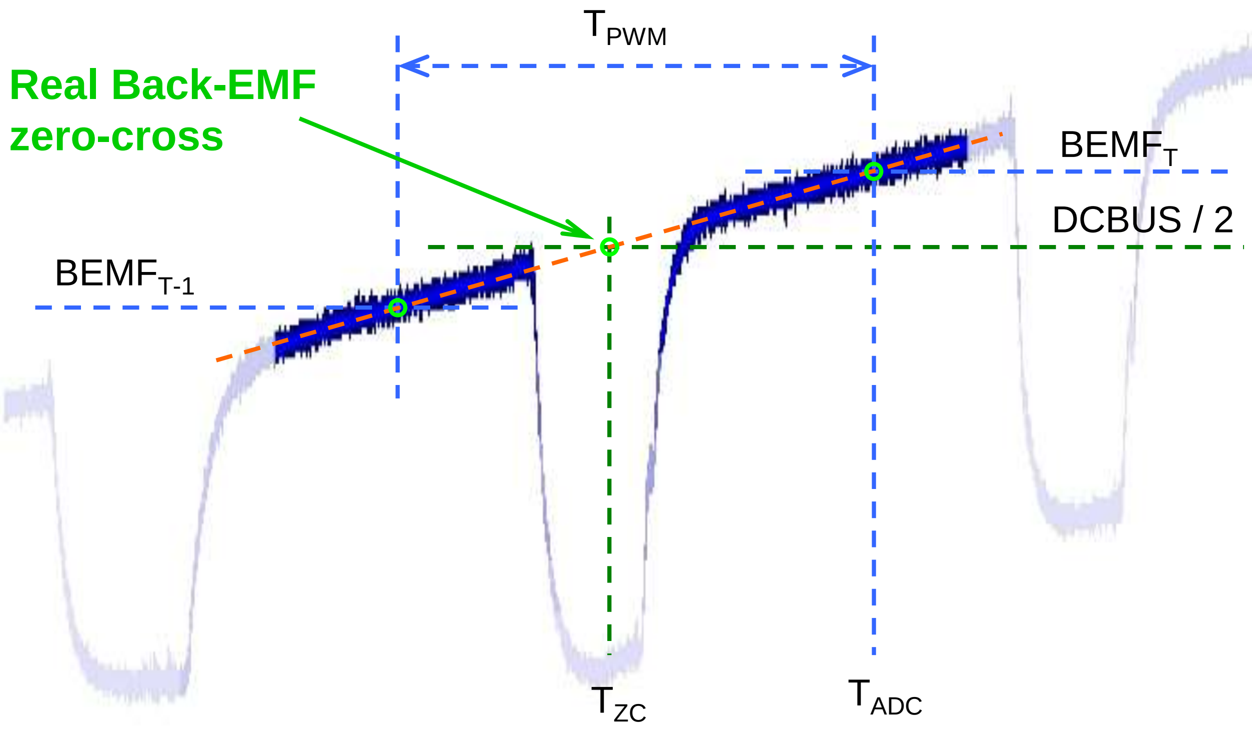


Zero Cross Linear Interpolation









Rising Back-EMF Voltage ADC samples interpolation

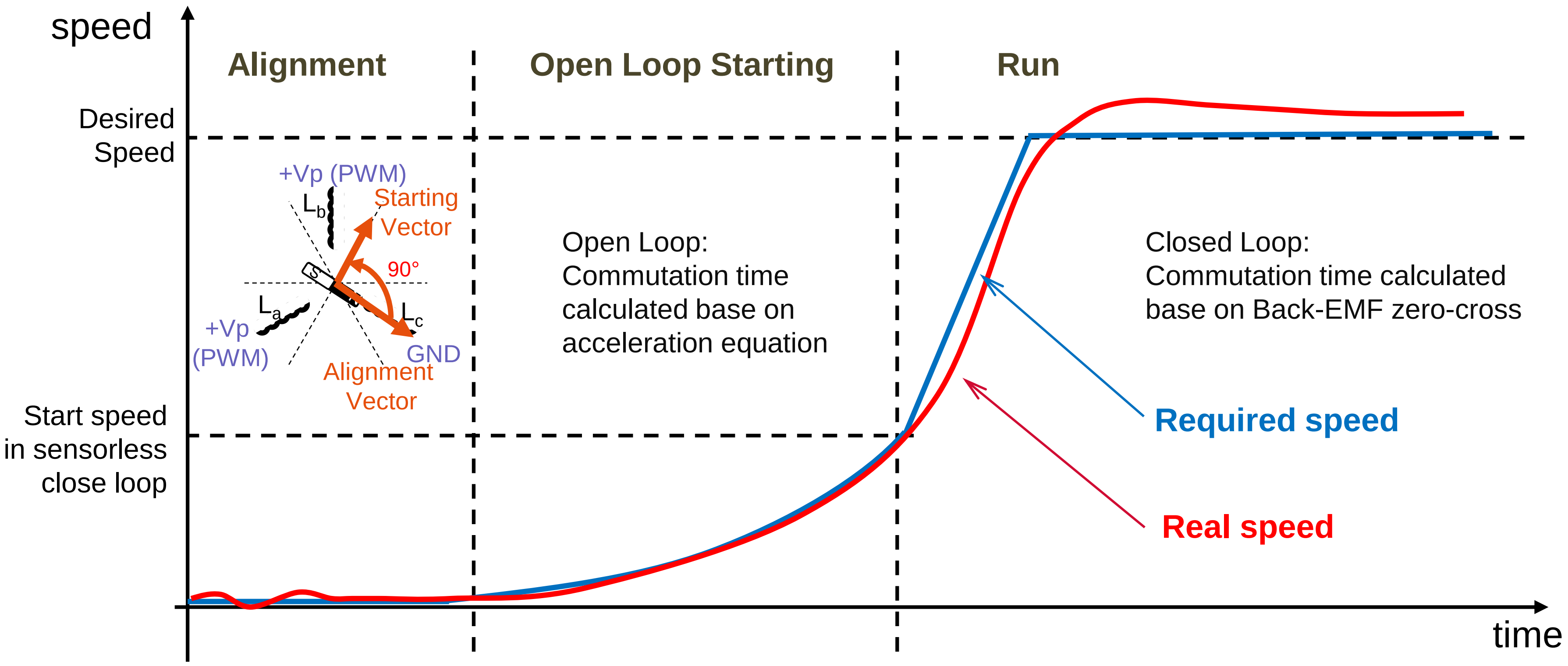
$$T_{ZC} = T_{ADC} - \frac{BEMF_T - DCBUS/2}{BEMF_T - BEMF_{T-1}} \cdot T_{PWM}$$

Falling Back-EMF Voltage ADC samples interpolation

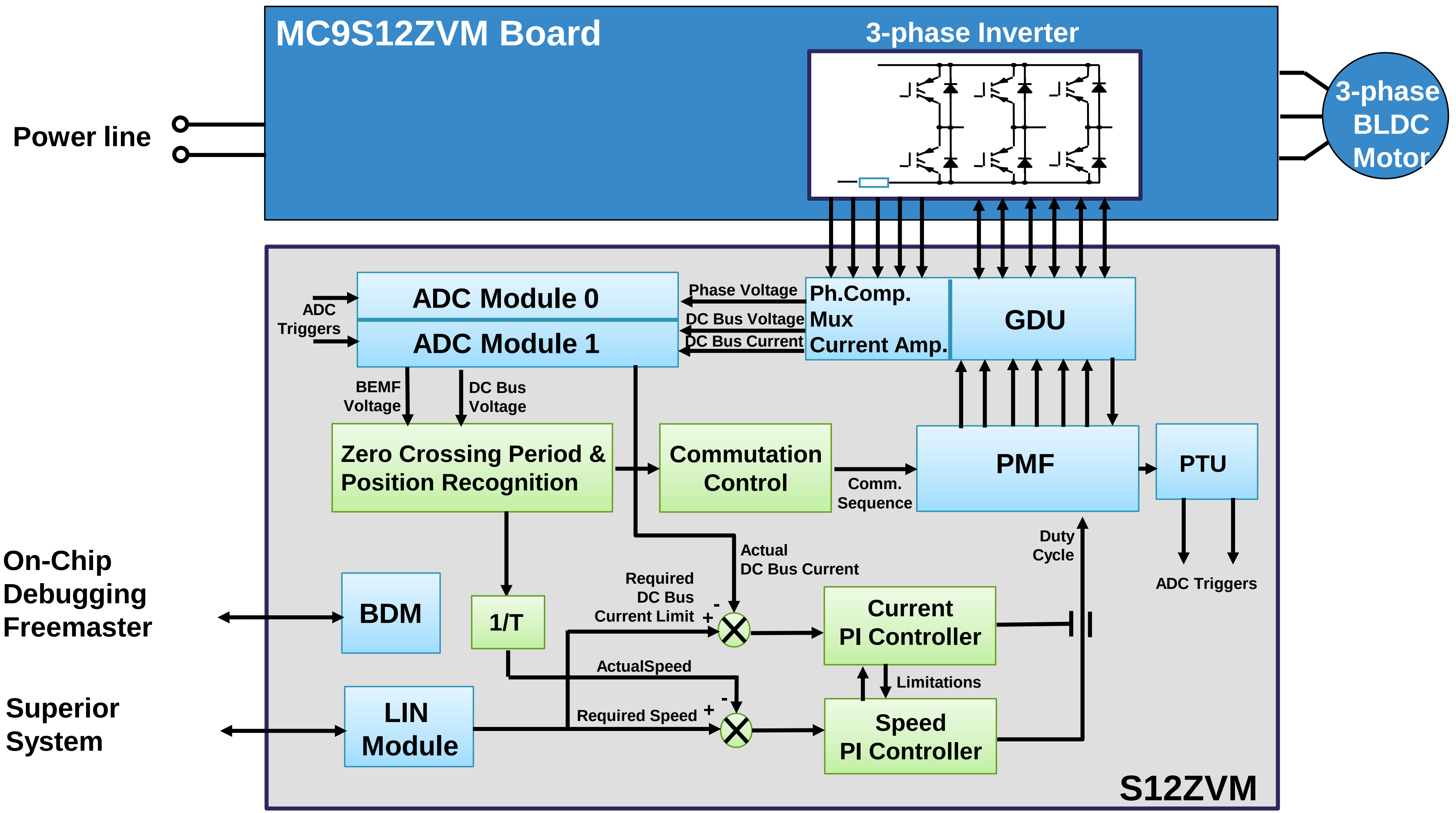
$$T_{ZC} = T_{ADC} - \frac{DCBUS/2 - BEMF_T}{BEMF_{T-1} - BEMF_T} \cdot T_{PWM}$$

BLDC Motor Startup States

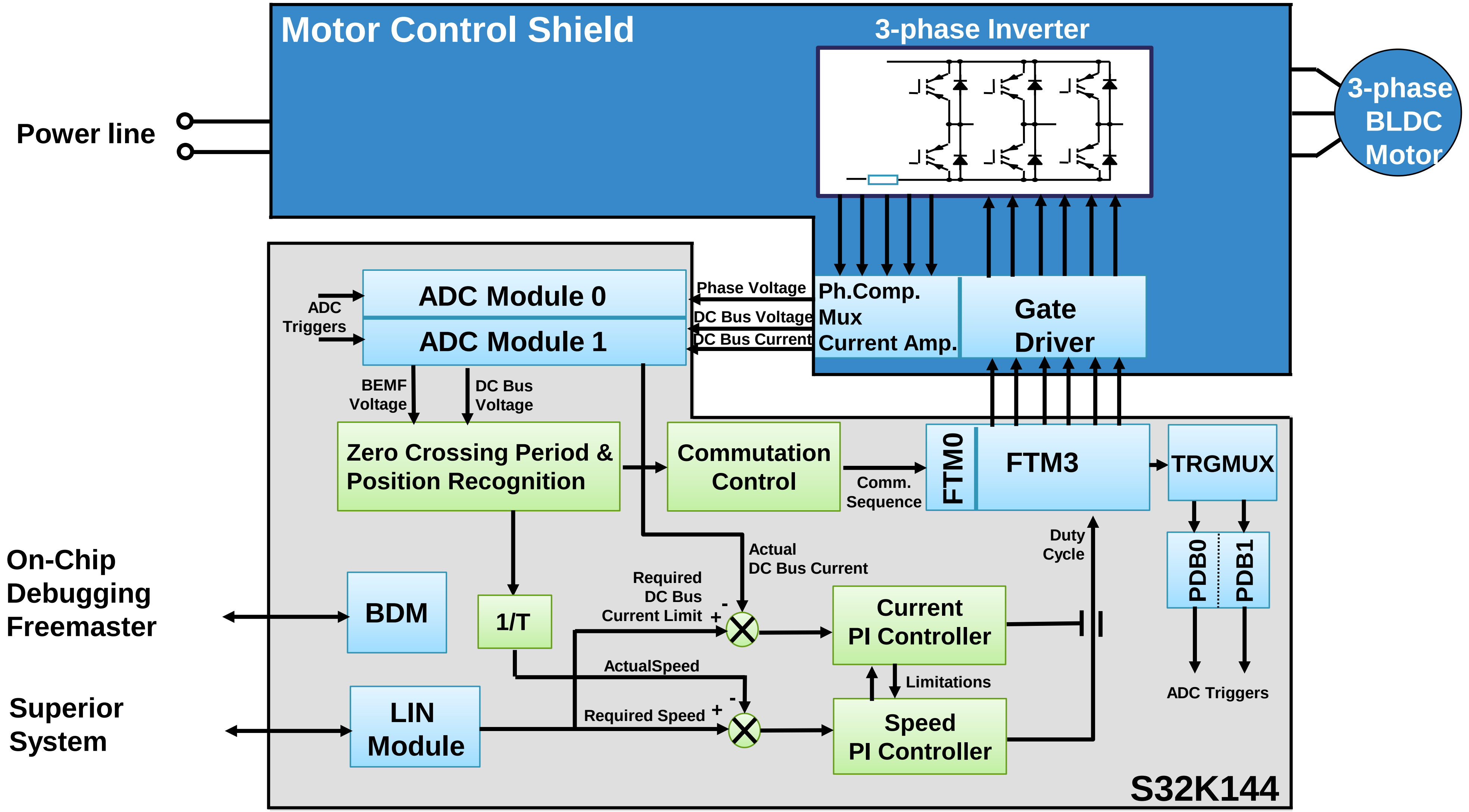
The open-loop starting sequence ensures motor running at enough high speed and so the zero-cross events can be successfully detected and sensorless closed loop control can follow.



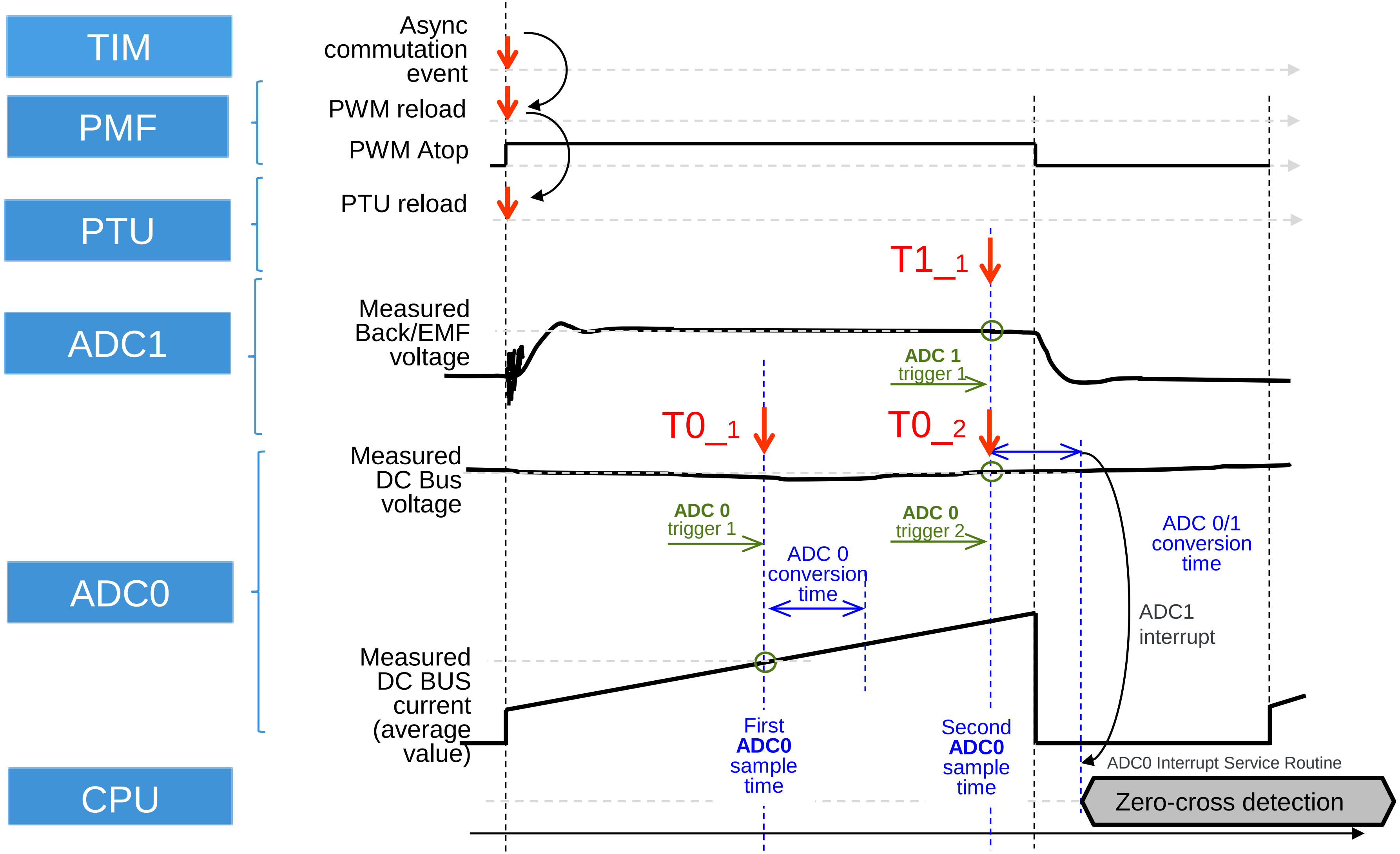
S12ZVM: BLDC Sensorless Control Block Diagram



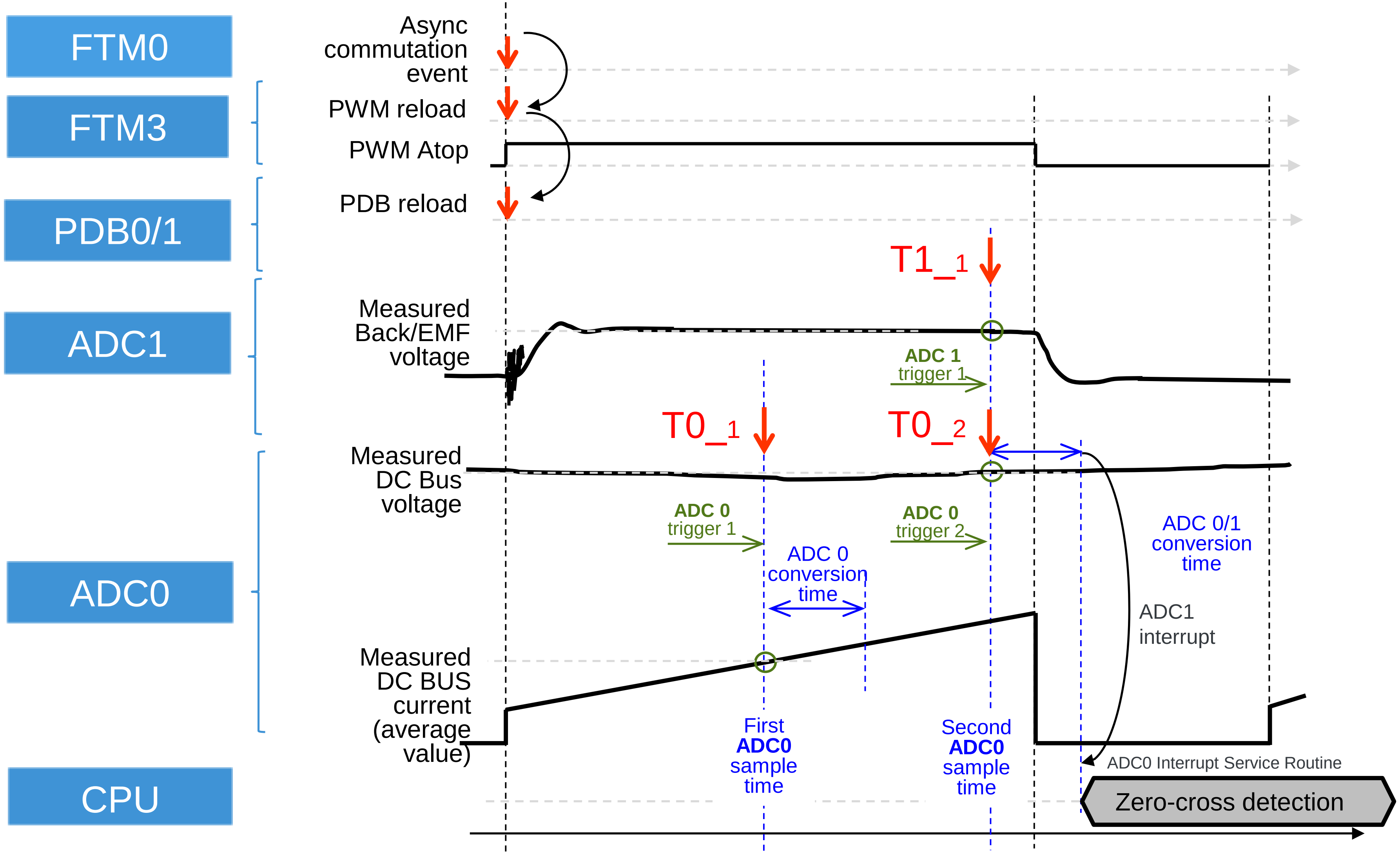
S32K144: BLDC Sensorless Control Block Diagram



S12ZVM: Modules Involvement in BLDC Control Loop



S32K144: Modules Involvement in BLDC Control Loop



FIELD-ORIENTED CONTROL

SENSORLESS

Field Oriented Control Principle

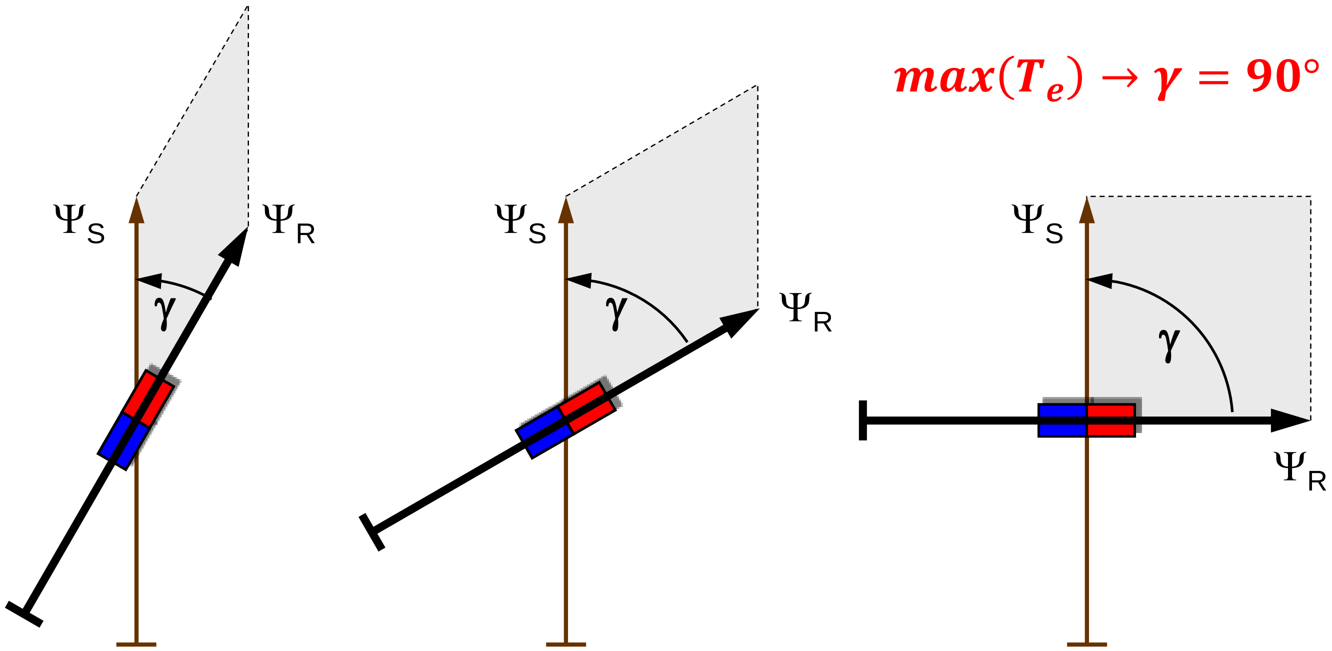
All is about magnetic fields interaction!

- Rotor Magnetic field
- Stator Magnetic field

- The torque/force is produced when both fields form an non zero angle
- Having the stator magnetic field leading the rotor magnetic field we form an el. motor
- Then FOC is to control the torque
  - thus also the mag. field angle
  - by strength of the rotor mag. field and
  - by strength of the stator mag. field

$$T_e = c \cdot \Psi_R \times \Psi_S = c \cdot |\Psi_R| \times |\Psi_S| \cdot sin\gamma$$

$$\boldsymbol{max(T_e) \rightarrow \gamma = 90°}$$



FOC allows to control the motor at **max(T_e)**.

Why Field Oriented Control?

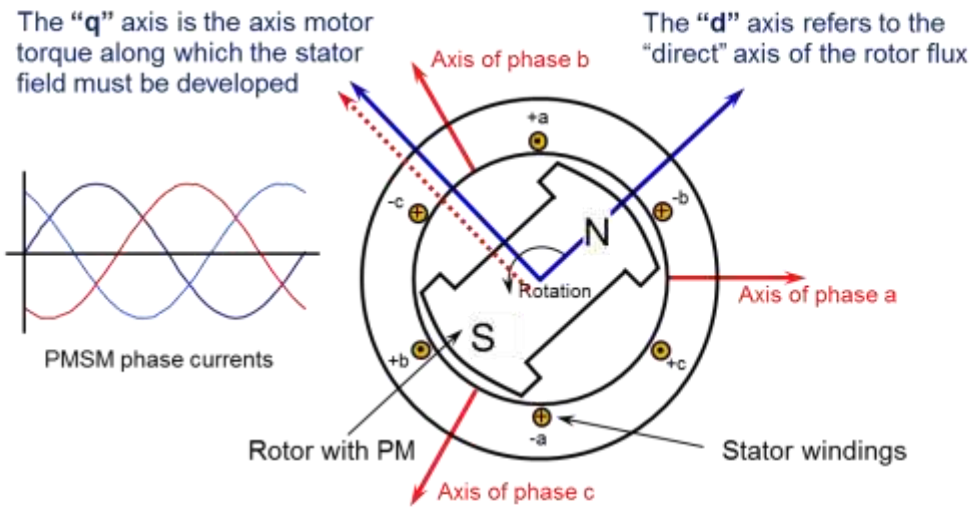
For a PMSM motor the 3 sinusodal phase currents have to be controlled to create a flux vector which is perpendicular to the rotor flux current

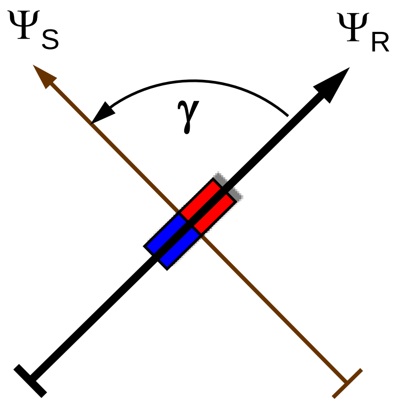
- To control the three sinusoidal currents independantly would be a very complex mathematical task

- FOC simplifies the math by transforming the 3 phase system (*abc*) to a two phase (*dq*) DC system viewing angle

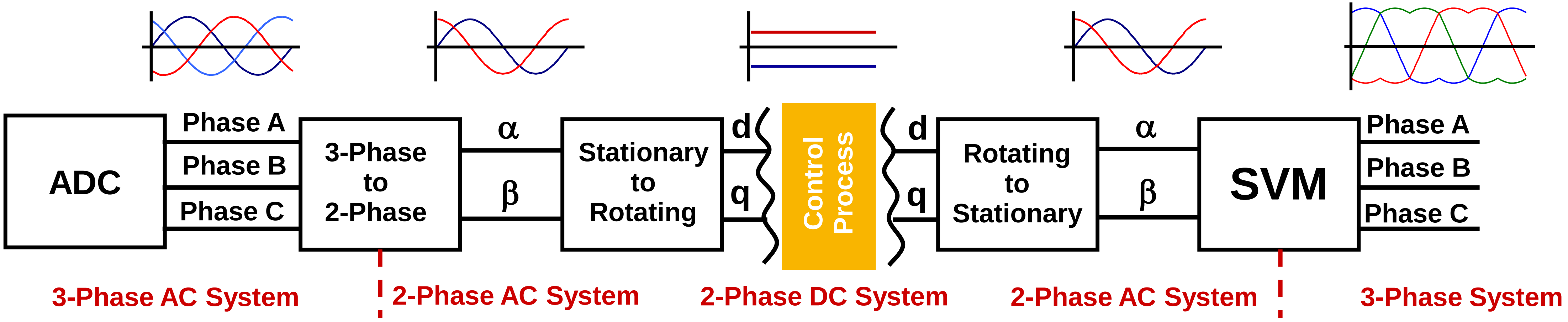
- FOC decomposes the stator current into two components:
  - i_D – Flux-producing component
  - i_Q – Torque-producing component

- Better performance
  - Full motor torque capability at low speed
  - Better dynamic behavior
  - Higher efficiency for each operation point in a wide speed range
  - Decoupled control of torque and flux
  - Natural four quadrant operation





Field Oriented Control in Steps



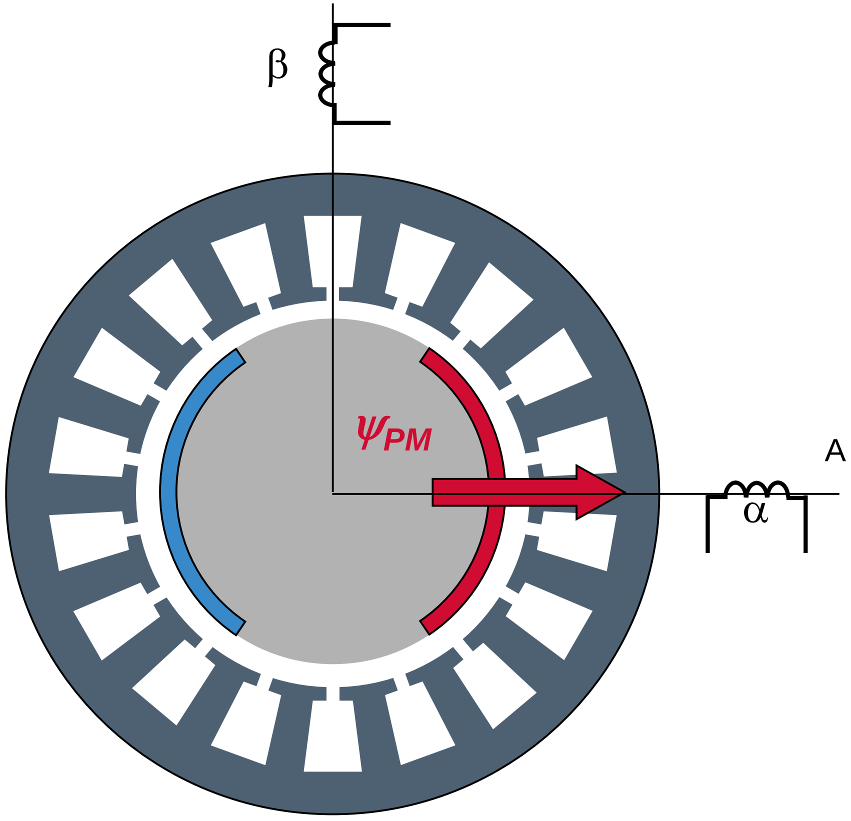
1. Measure obtain state variables quantities (e.g. phase currents, voltages, rotor position, rotor speed …).
2. Transform quantities from 3-phase system to 2-phase system (Forward Clark Transform) to simplify the math - lower number of equations
3. Transform quantities from stationary to rotating reference 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

Transformation benefits:

- Reduce 3ph system to 2ph system
- Eliminates the AC component

FOC Design - 2-phase PMSM Model

- Considering sinusoidal 2-phase distributed winding and neglecting effect of magnetic saturation and leakage inductances



Stator voltage equations

Forward Park

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = R_s \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{d}{dt}\begin{bmatrix} \psi_\alpha \\ \psi_\beta \end{bmatrix}$$

Stator linkage flux

$$\begin{bmatrix} \Psi_{S\alpha} \\ \Psi_{S\beta} \end{bmatrix} = \begin{bmatrix} L_S & 0 \\ 0 & L_S \end{bmatrix} \cdot \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \Psi_{PM}\Big|_{i_{Sd}=0} \cdot \begin{bmatrix} \cos\theta_{re} \\ \sin\theta_{re} \end{bmatrix}$$

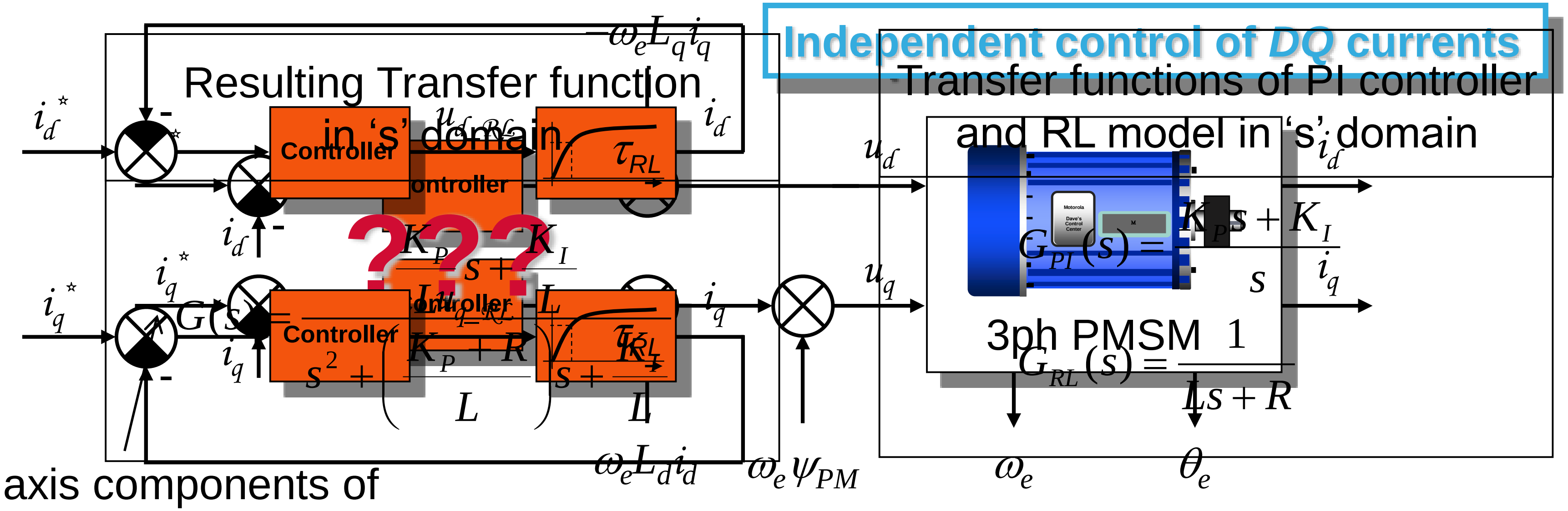
Internal motor torque

$$T_i = \frac{3}{2}\frac{p_p}{\omega_e}(u_{i\alpha} i_\alpha + u_{i\beta} i_\beta) = \frac{3}{2} p_p (\Psi_\alpha i_\beta - \Psi_\beta i_\alpha)$$

PMSM Torque Current Control

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = R_s \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} sL_d & 0 \\ 0 & sL_q \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \omega_e \begin{bmatrix} -L_q \\ L_d \end{bmatrix} \begin{bmatrix} i_q \\ i_d \end{bmatrix} + \omega_e \psi_{PM} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

R-L circuit | cross-coupling | backEMF



$G_{PI}(s) = \frac{K_P s + K_I}{s}$

$G_{RL}(s) = \frac{1}{Ls + R}$

Two axis components of required current vector

Zero Cancelation

- Design of the controller gains can be done by matching coefficients of characteristic polynomial with those of an ideal 2nd order system.

Transfer function of current loop

$$G(s)=\frac{\frac{K_P}{L}s+\frac{K_I}{L}}{s^2+\left(\frac{K_P+R}{L}\right)s+\frac{K_I}{L}}=\frac{\frac{K_I}{L}\left(\frac{K_P}{K_I}s+1\right)}{s^2+\left(\frac{K_P+R}{L}\right)s+\frac{K_I}{L}}$$

zero

Transfer function of ideal 2nd order system

$$G_{ideal}(s)=\frac{\omega_0^2}{s^2+2\xi\omega_0 s+\omega_0^2}$$

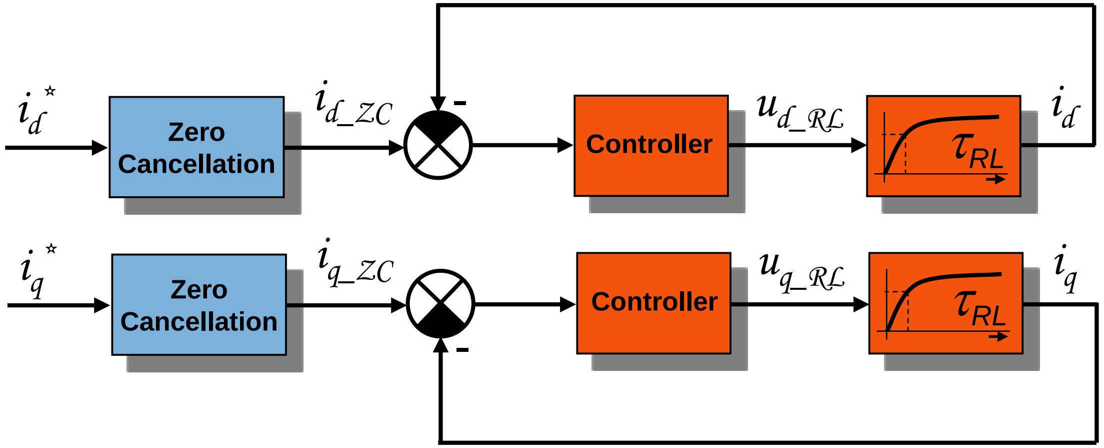
ξ – is damping factor

ω_0 – is natural frequency

- "Zero" introduced by PI controller at $-K_P/K_I$ adds derivative behavior to the closed loop, creating overshoot during step response

Zero Cancelation

- Zero Cancellation" placed in the feed-forward path shall be designed to compensate the closed loop zero with unity DC gain



$G_{ZC}(s)$ × **$G_{CL}(s)$** = **$G(s)$**

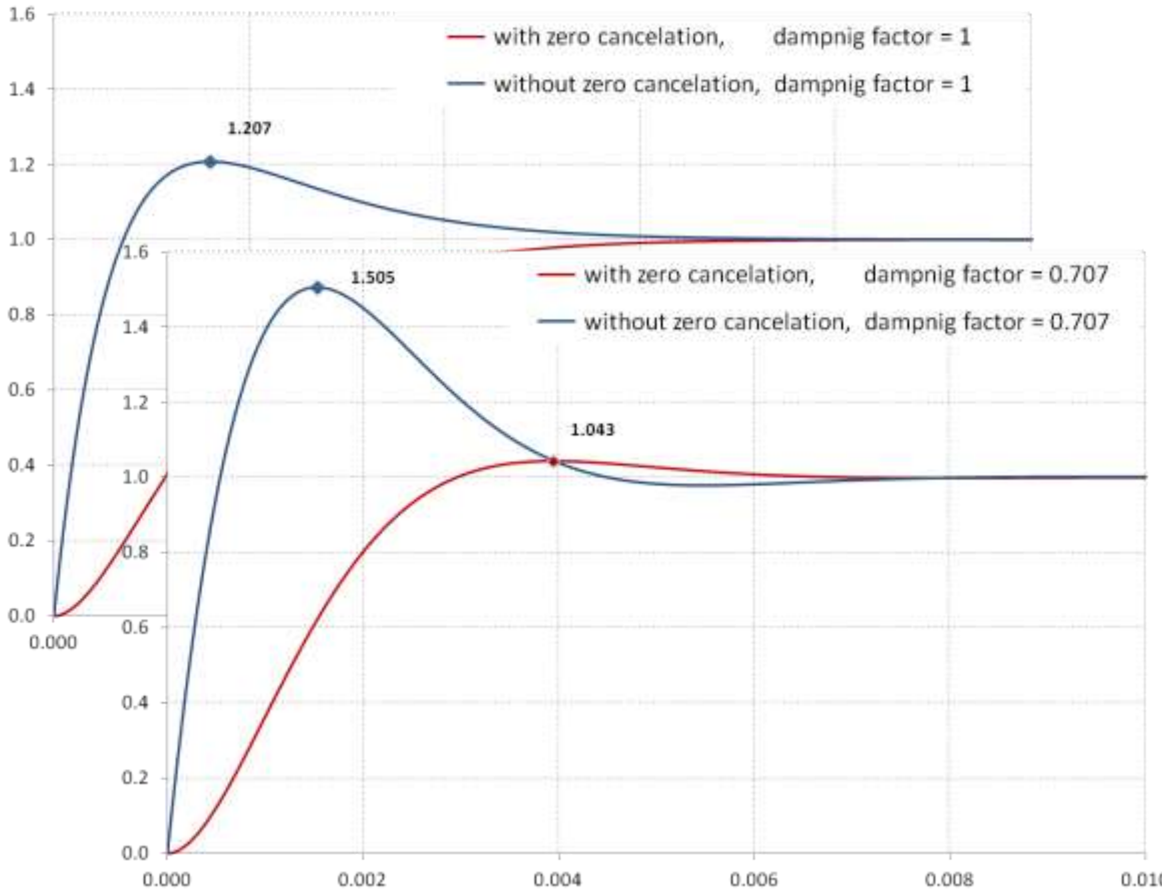
$$G(s) = \frac{1}{\left(\frac{K_P}{K_I}s + 1\right)} \times \frac{\frac{K_I}{L}\left(\frac{K_P}{K_I}s + 1\right)}{s^2 + \left(\frac{K_P + R}{L}\right)s + \frac{K_I}{L}} = \frac{\frac{K_I}{L}}{s^2 + \left(\frac{K_P + R}{L}\right)s + \frac{K_I}{L}}$$

PI Controller Gain Calculation

- Implementation of zero Cancellation allows precise matching of characteristic polynomial coefficients
- Enables simple tuning of the current loop bandwidth and attenuation

$$G(s) = \frac{\frac{K_I}{L}}{s^2 + \left(\frac{K_P + R}{L}\right)s + \frac{K_I}{L}}$$

$$G_{ideal}(s) = \frac{\omega_0^2}{s^2 + 2\xi\omega_0 s + \omega_0^2}$$

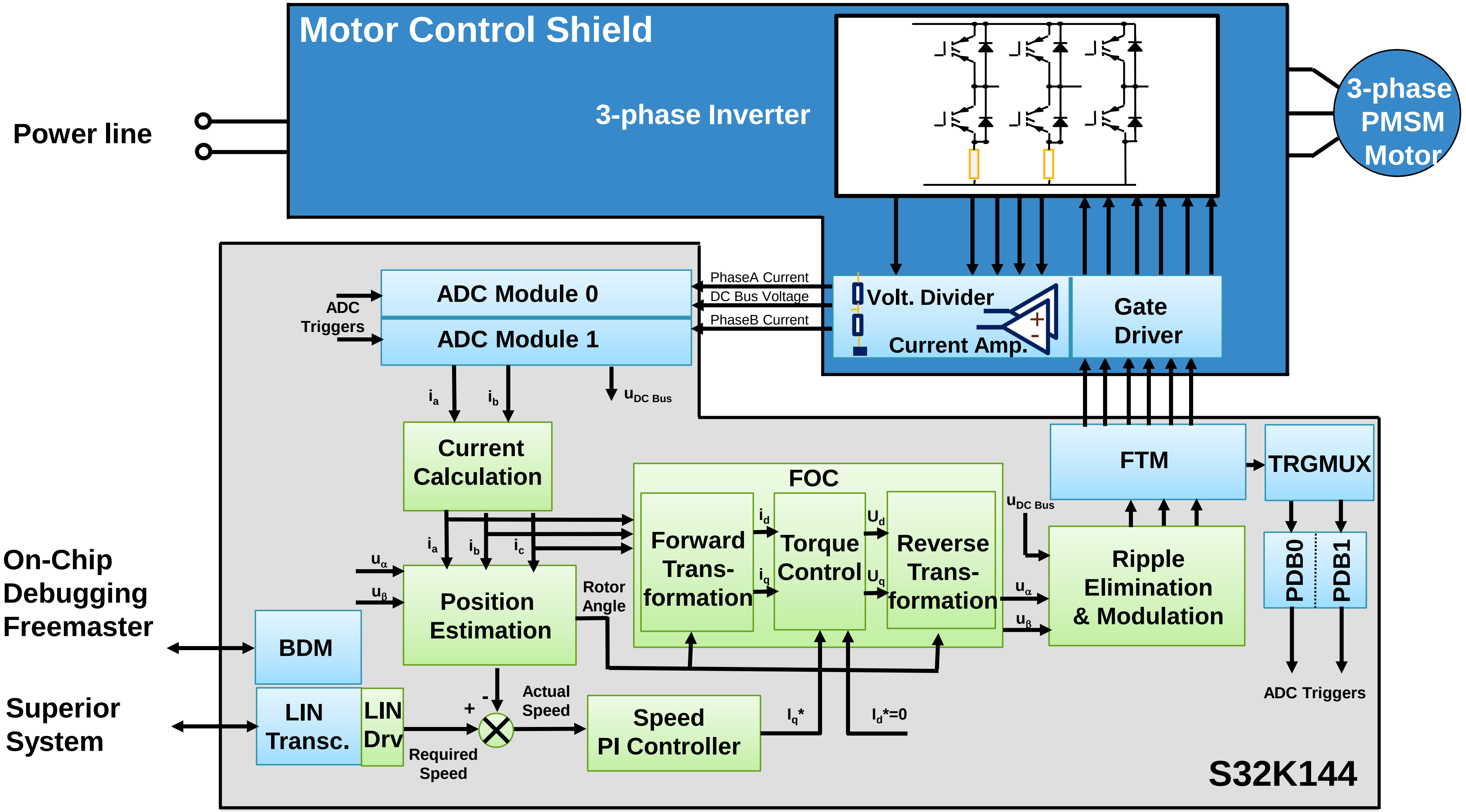


<u>PI controller gains</u>

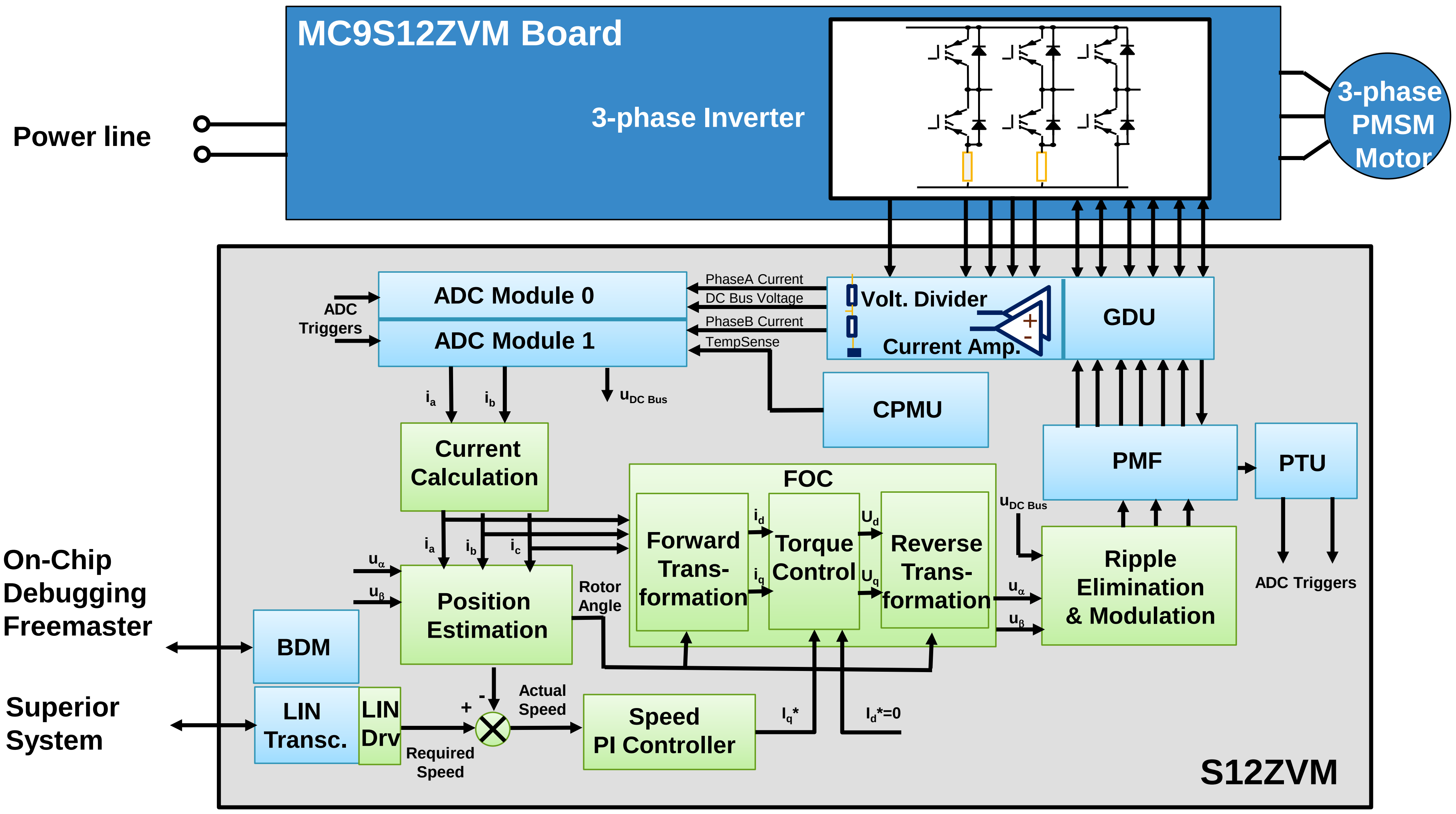
$$K_I = \omega_0^2 L$$

$$K_P = 2\xi\omega_0 L - R$$

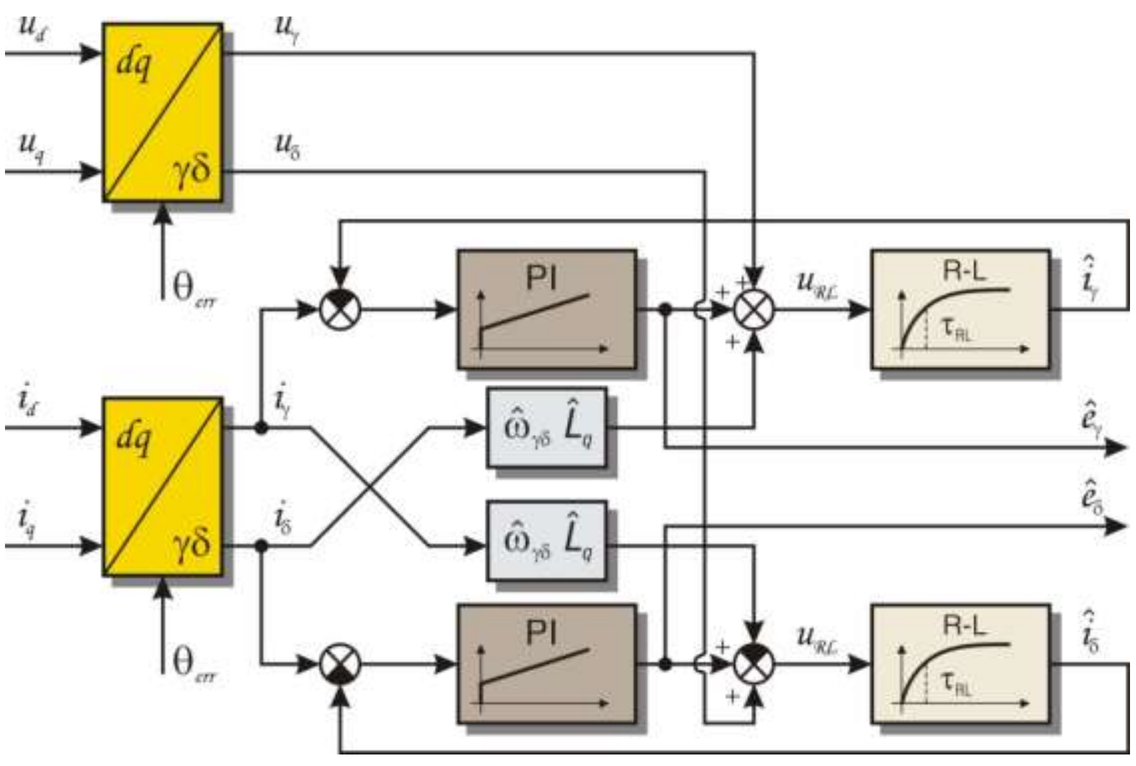
Sensorless PMSM Control Block Diagram on S32K144



Sensorless PMSM Control Block Diagram on S12ZVM

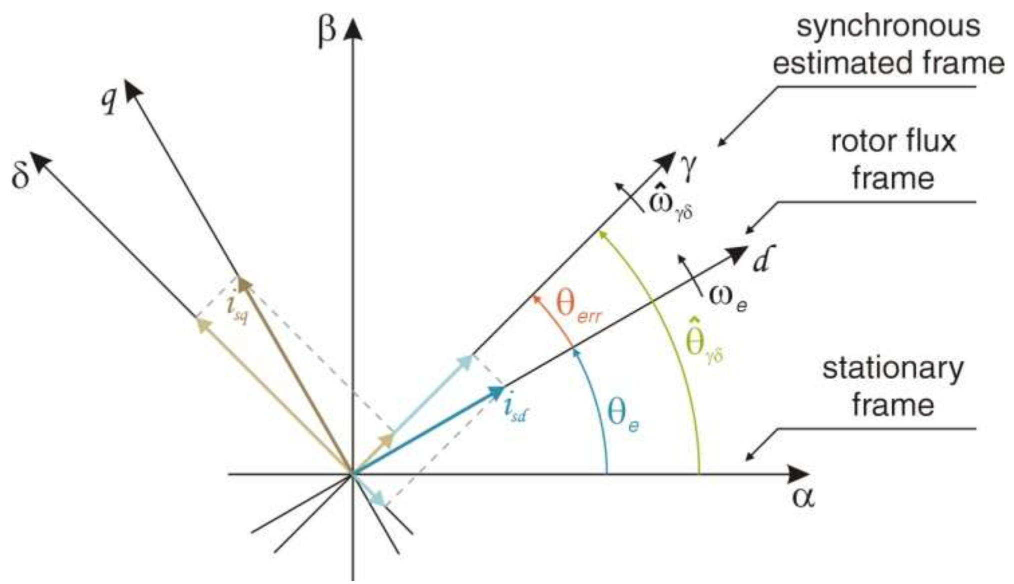


Saliency Based Back-EMF Observer

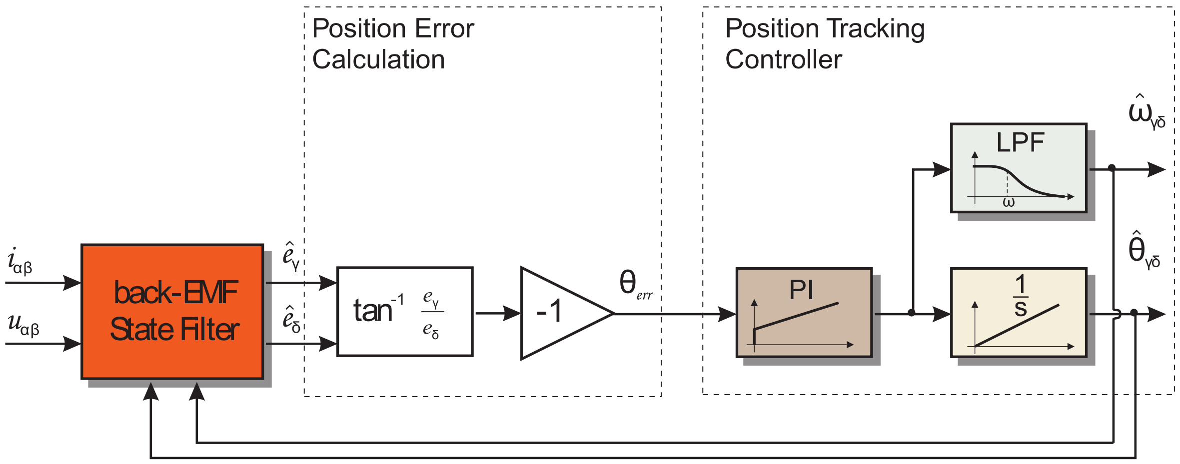


$$\begin{bmatrix} u_\gamma \\ u_\delta \end{bmatrix} = \begin{bmatrix} R_S + sL_d & -\hat{\omega}_{\gamma\delta}L_q \\ \hat{\omega}_{\gamma\delta}L_q & R_S + sL_d \end{bmatrix} \begin{bmatrix} i_\gamma \\ i_\delta \end{bmatrix} + E_{SAL} \begin{bmatrix} -sin(\theta_{err}) \\ cos(\theta_{err}) \end{bmatrix}$$

- Saliency based back-EMF voltage is generated due to $L_d \neq L_q$
- Because back-EMF term is not modeled, observer actually acts as a back-EMF state filter
- Observer is designed in synchronous reference frame, i.e. all observer quantities are DC in steady state making the observer accuracy independent of rotor speed.

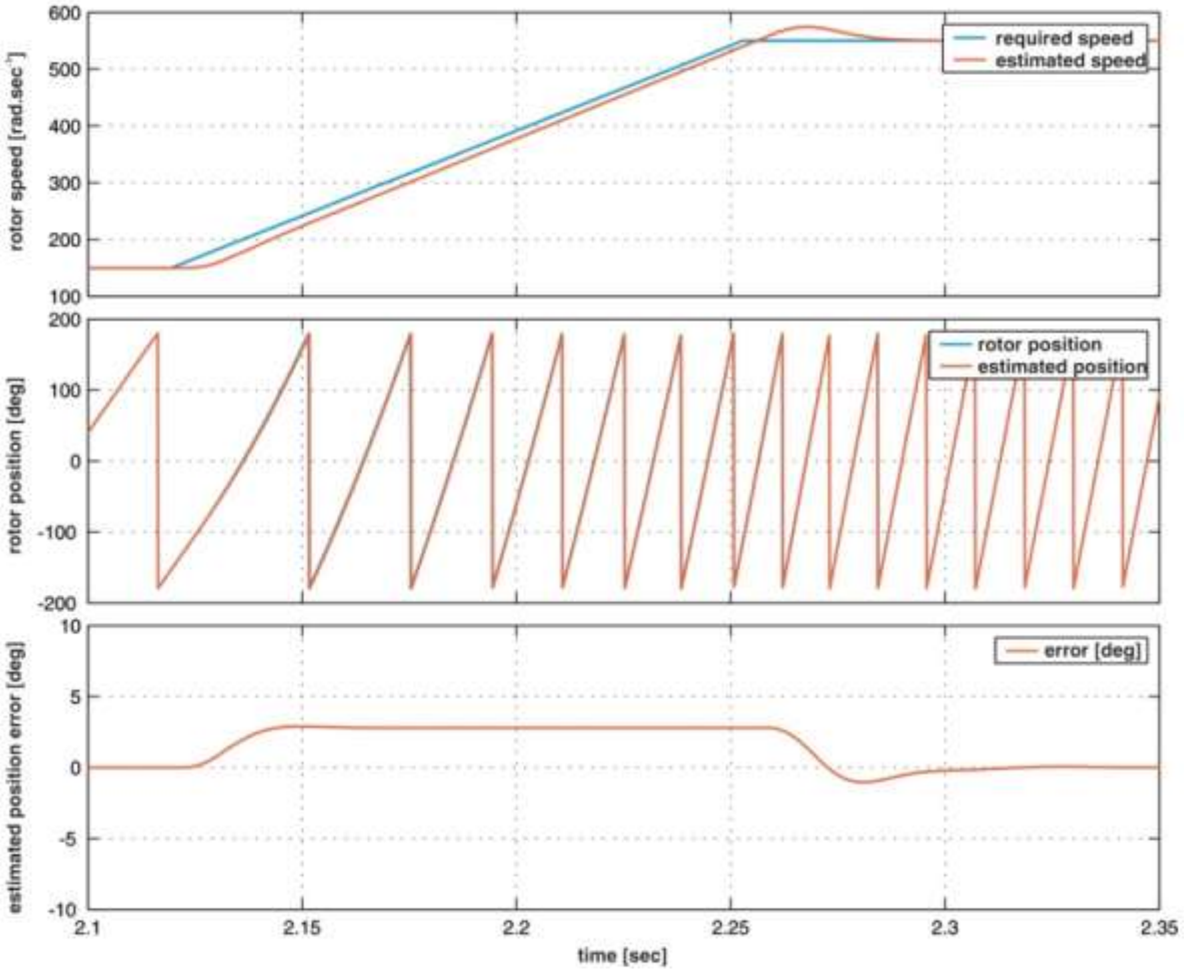


Position Estimation Using Saliency Based Back-EMF

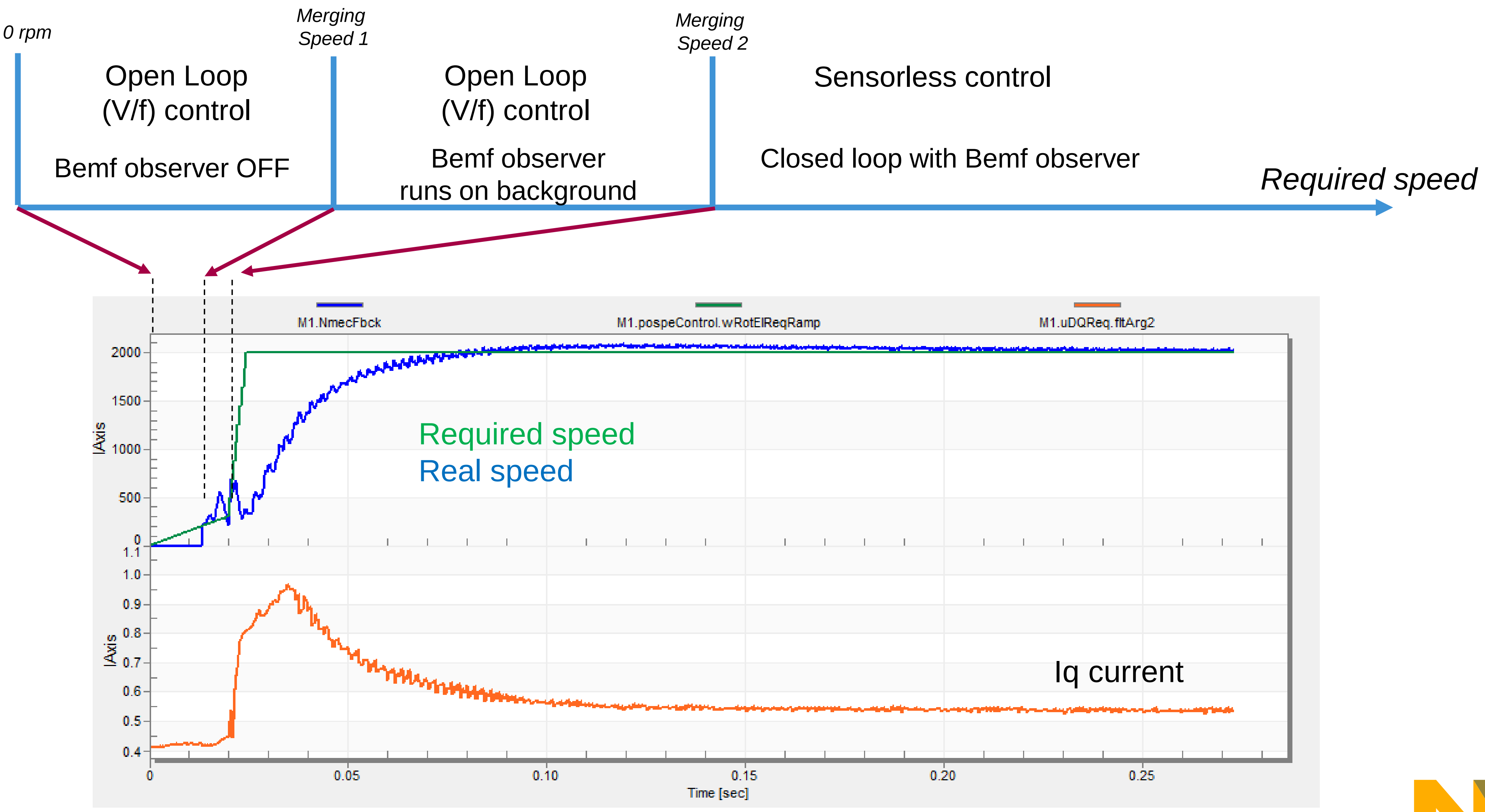


$$\begin{bmatrix} u_\gamma \\ u_\delta \end{bmatrix} = \begin{bmatrix} R_S + sL_d & -\hat{\omega}_{\gamma\delta}L_q \\ \hat{\omega}_{\gamma\delta}L_q & R_S + sL_d \end{bmatrix} \begin{bmatrix} i_\gamma \\ i_\delta \end{bmatrix} + E_{SAL} \begin{bmatrix} -\sin(\theta_{err}) \\ \cos(\theta_{err}) \end{bmatrix}$$

Position estimation can now be performed by extracting the θ_{err} term from the model and adjusting the position of the estimated reference frame such as to achieve $\theta_{err} = 0$.

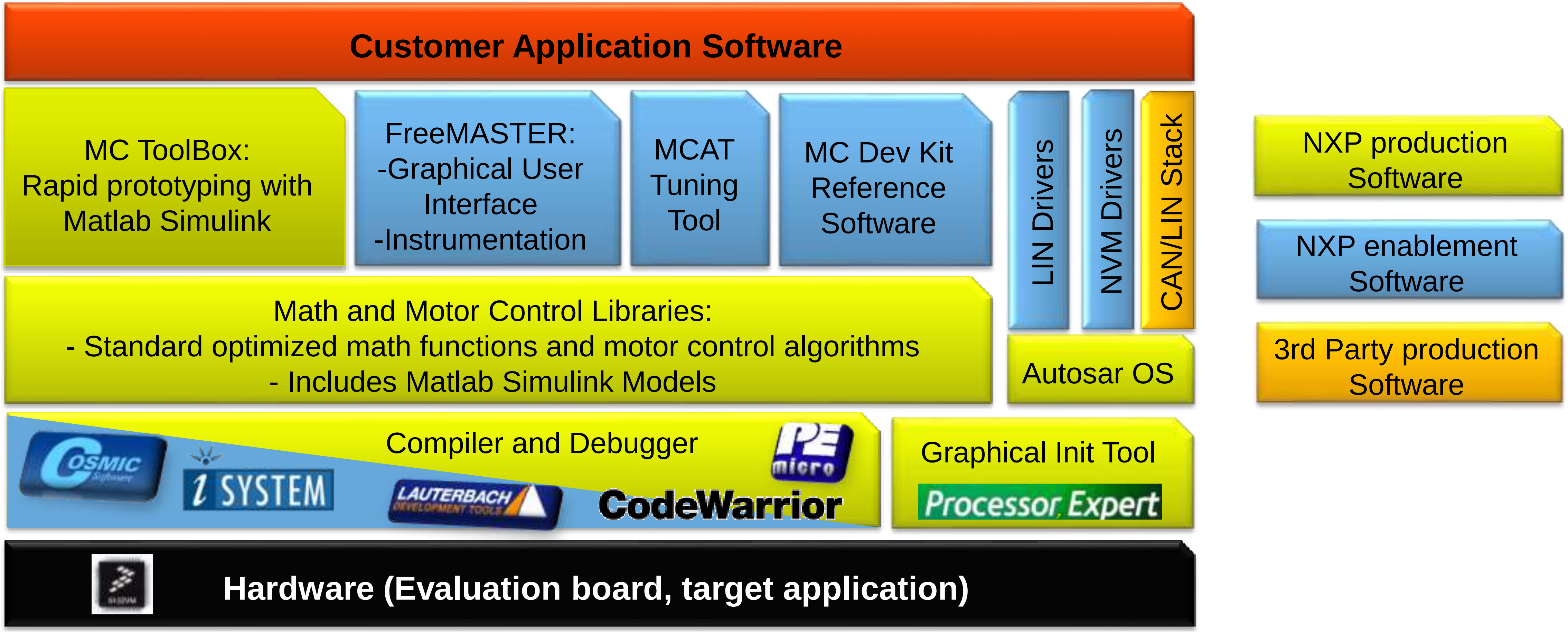


Sensorless Start-Up



ENABLEMENT

S12ZVM Ecosystem – The Complete Solution



S12ZVM Ecosystem – Software (motor control)

MC_TOOLBOX:
Motor Control Development Toolbox

IDE & tool chain for configuring and generating software to execute motor control algorithms on NXP MCUs:

- Includes Automotive Math and Motor Control Library set
- plug in to MATLAB™/Simulink™ model-based design environment
- optimized for fast execution on our MCUs with bit-accurate results compared to Simulink® simulation

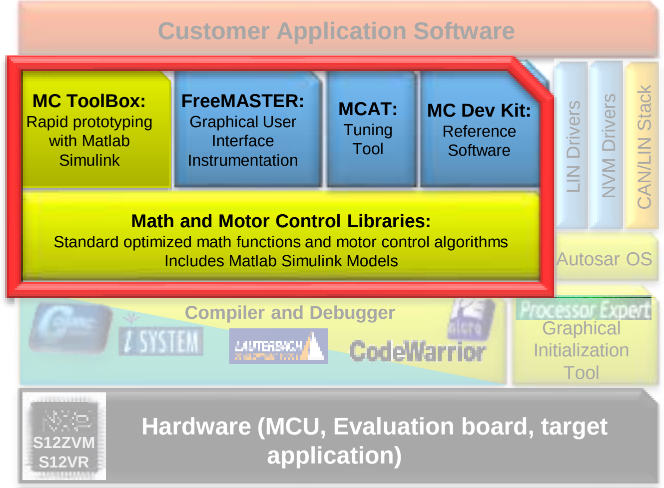
→ support rapid application development

MCAT: Motor Control Application Tuning Tool

Tune your drive:

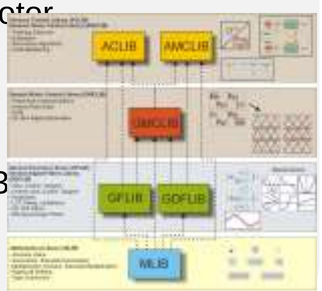
- Graphical User Interface, plugin to FreeMaster
- interfacing with the target MCU, modify software variables during runtime to tune your motor control algorithms to achieve control objectives (i.e. PI parameters)

→ Saves time in getting started & finetuning



AMMCLib: Automotive Math and Motor Control Library Set

- Precompiled software library containing building blocks for a wide range of motor control applications
- Easy migration between platforms with minimized effort
- Production ready SW (SPICE Level 3 CMMI and ISO9001/TS16949)
- Control loop modeling with Matlab/Simulink® models



FreeMASTER

Debugger for Real-time Applications:

- Graphical User Interface
- View & Modify variables run-time
- Real-time Monitor Tool
- Track & trace your variables
- Demonstration Platform
- Design your own dashboard

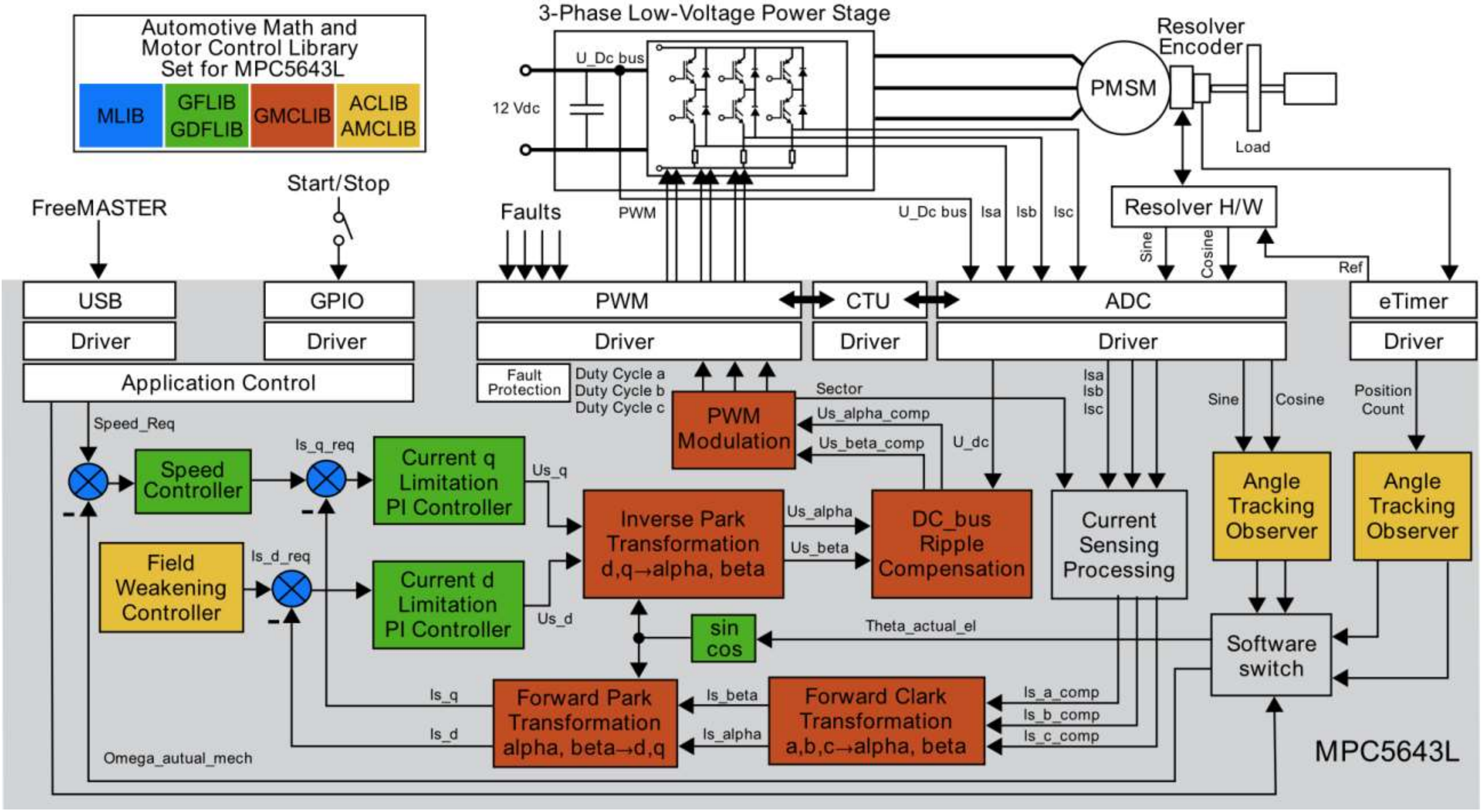
Motor Control Devkit SW

Comprehensive solution:

- All Motor Control Development Kit members come not only with hardware, but are supported (& documented) by application software available in source code.

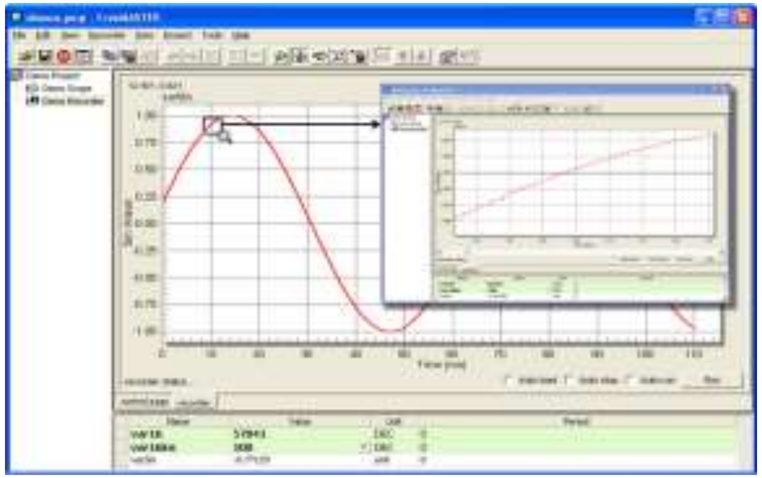
→ Reduce development & prototyping time
→ Faster Time to Market

Application Example for MPC5643L PMSM Field Oriented Control



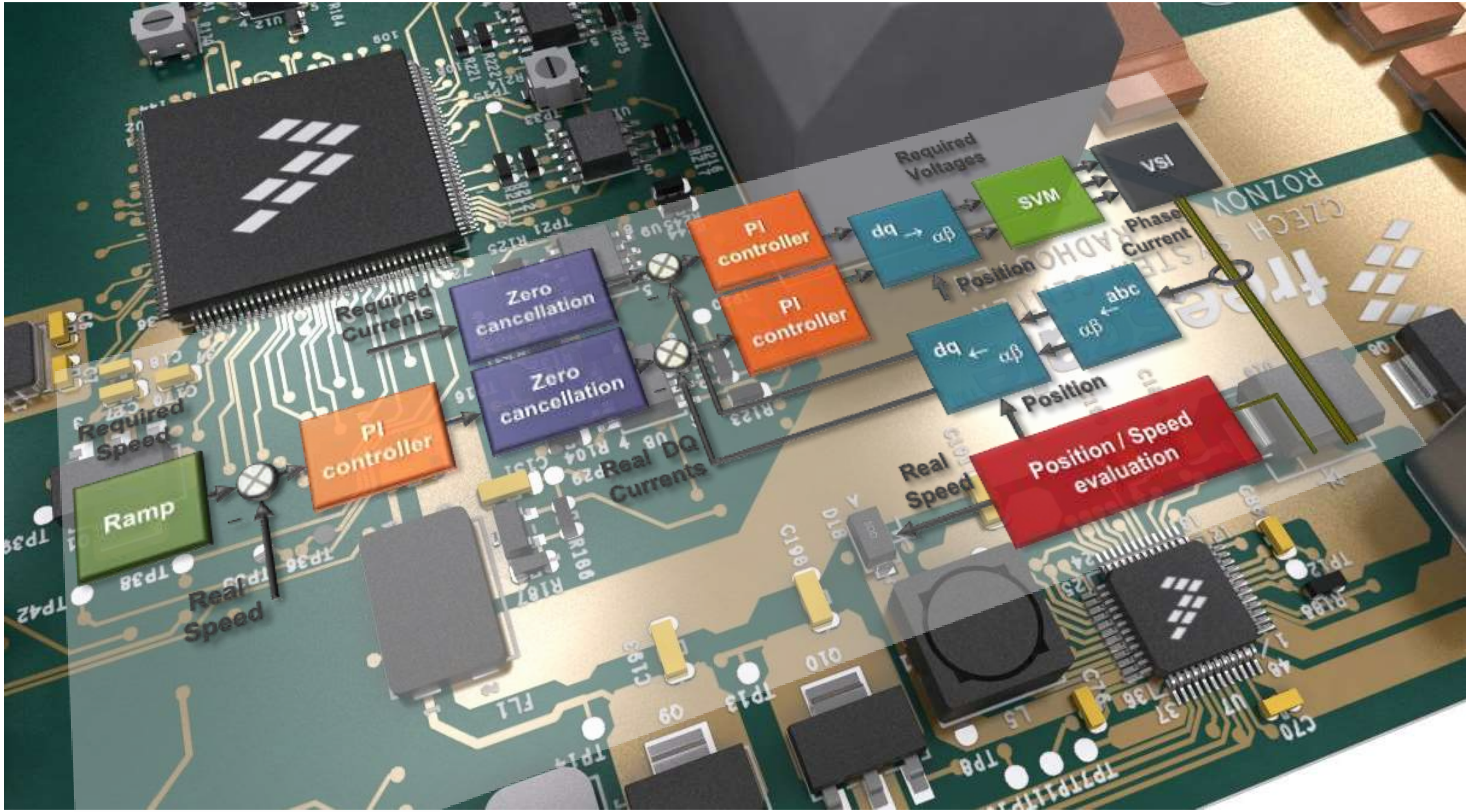
FREEMASTER

- **Real-time monitor tool**
  - Track your variables
  - Tracing capability
- **Graphical User Interface**
  - Modify variables run-time
- **Demonstration platform**
  - Design your own dashboard

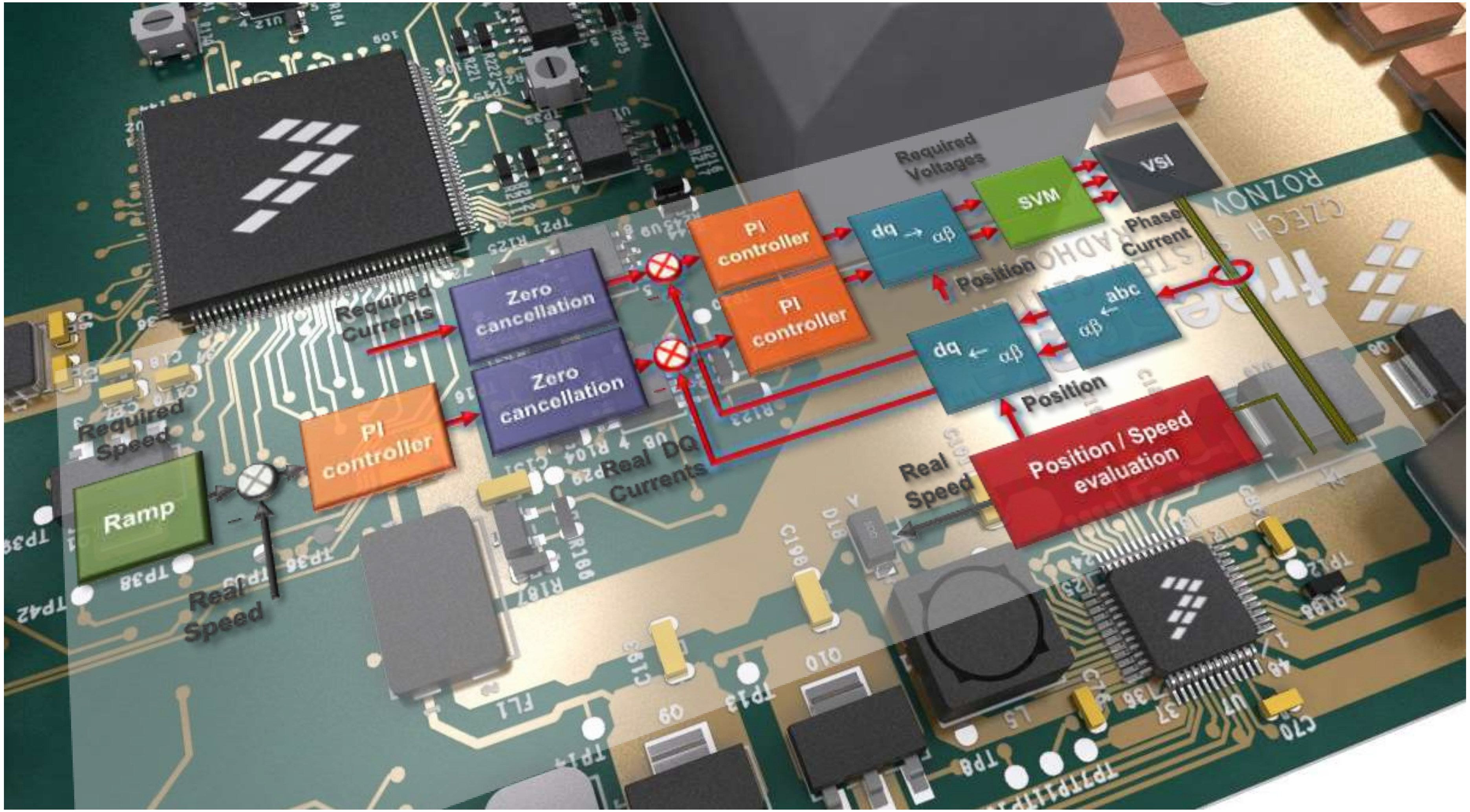


MOTOR CONTROL APPLICATION TOOL (MCAT)

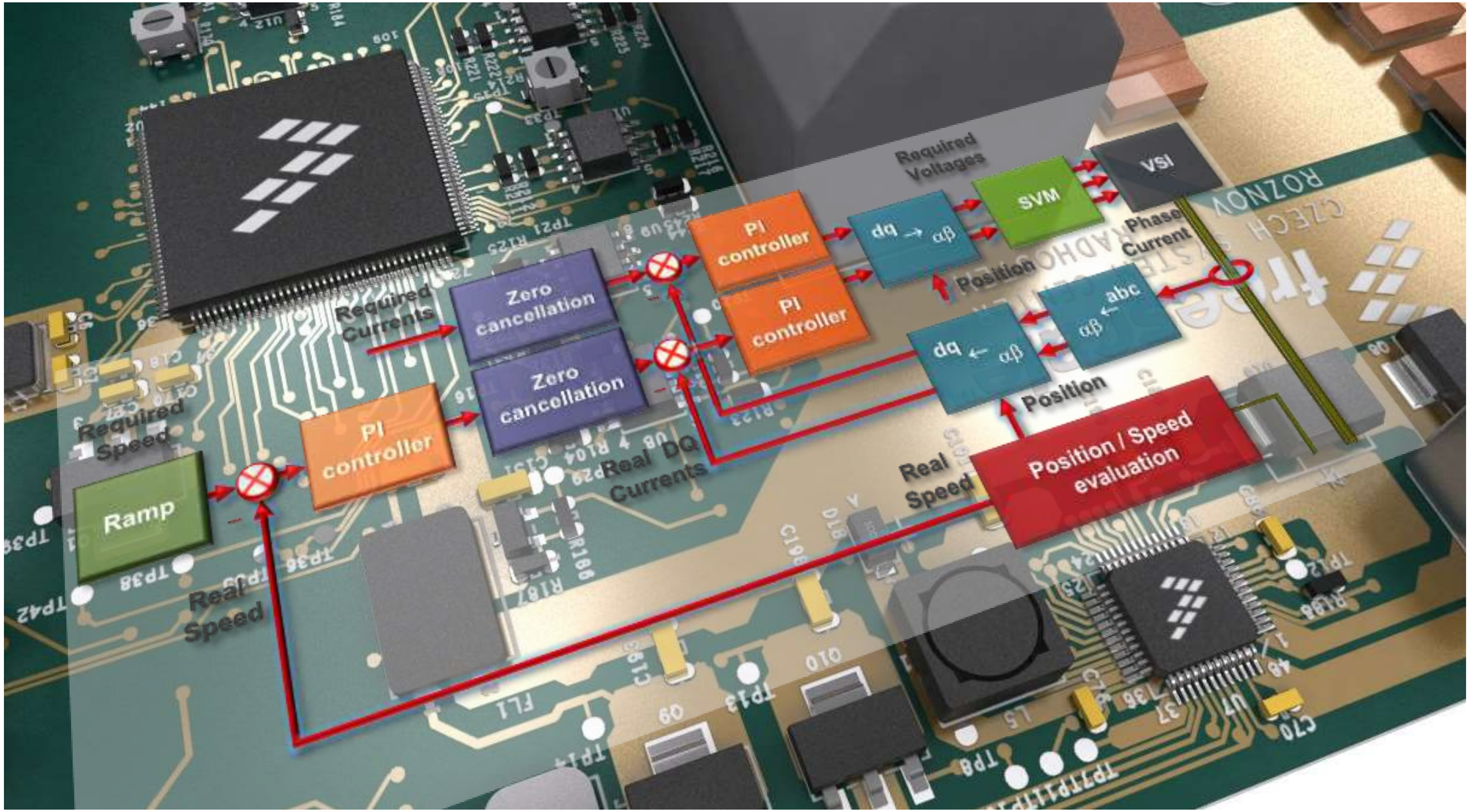
Motor Control Structure



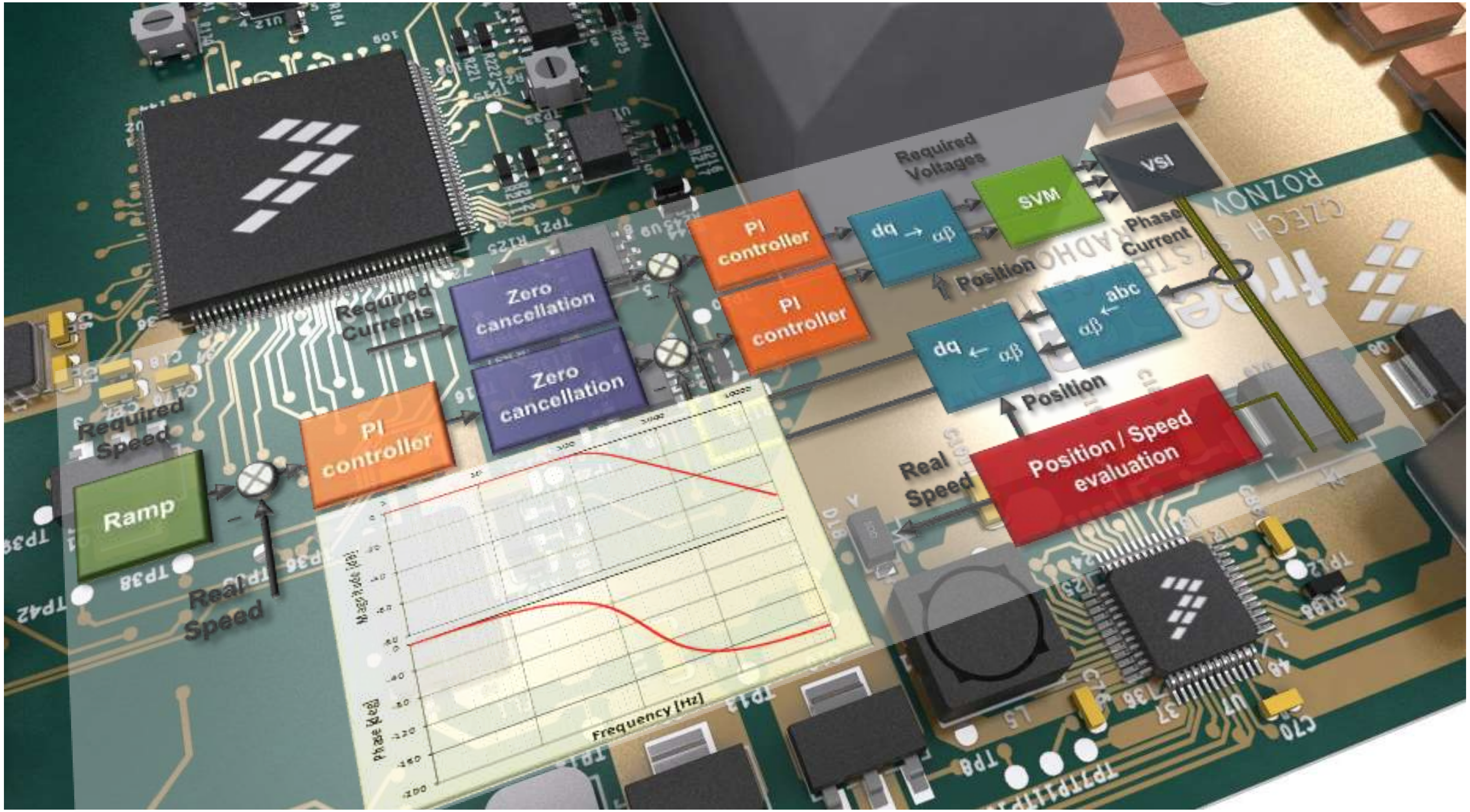
Current/Torque Control Loop



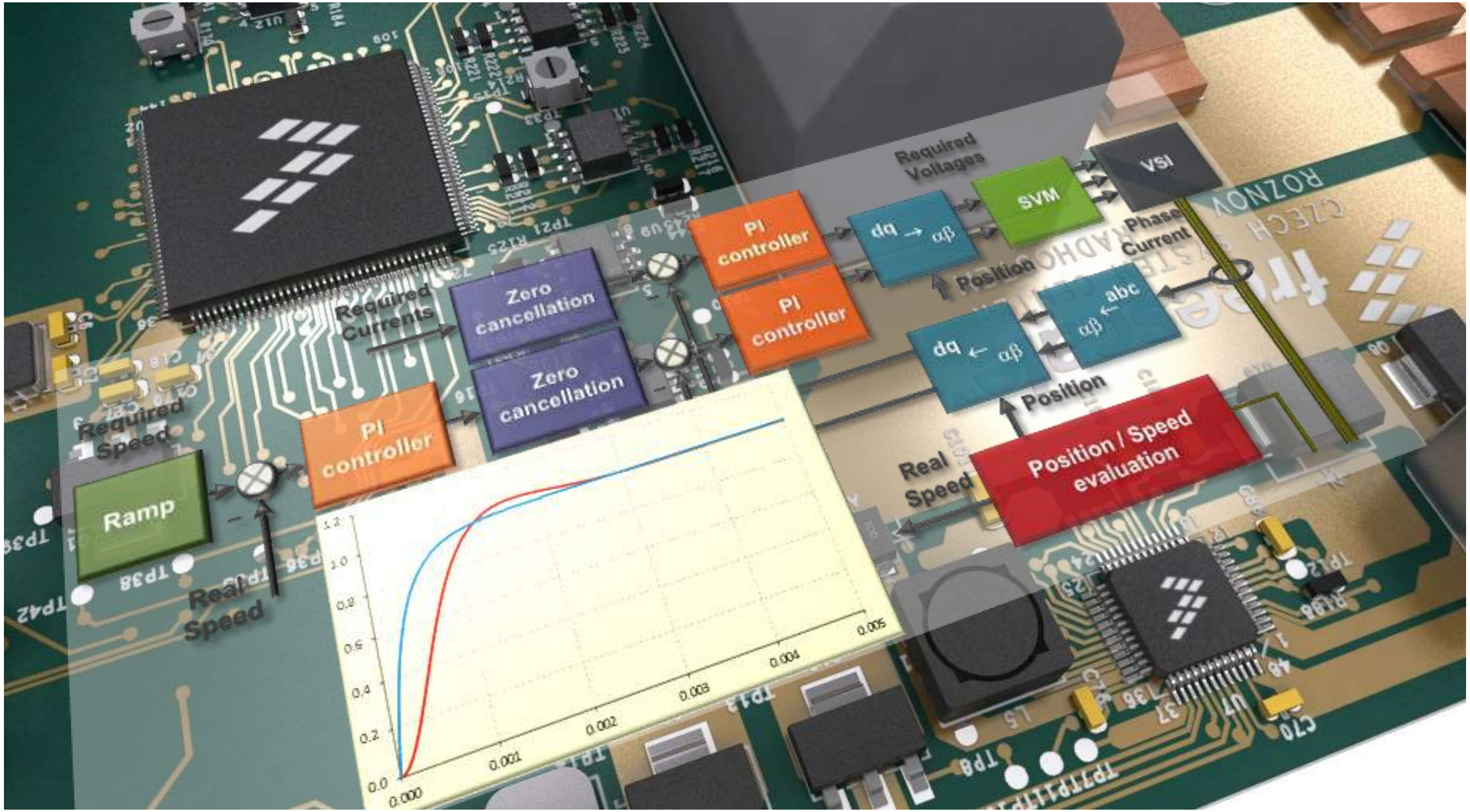
Speed Control Loop



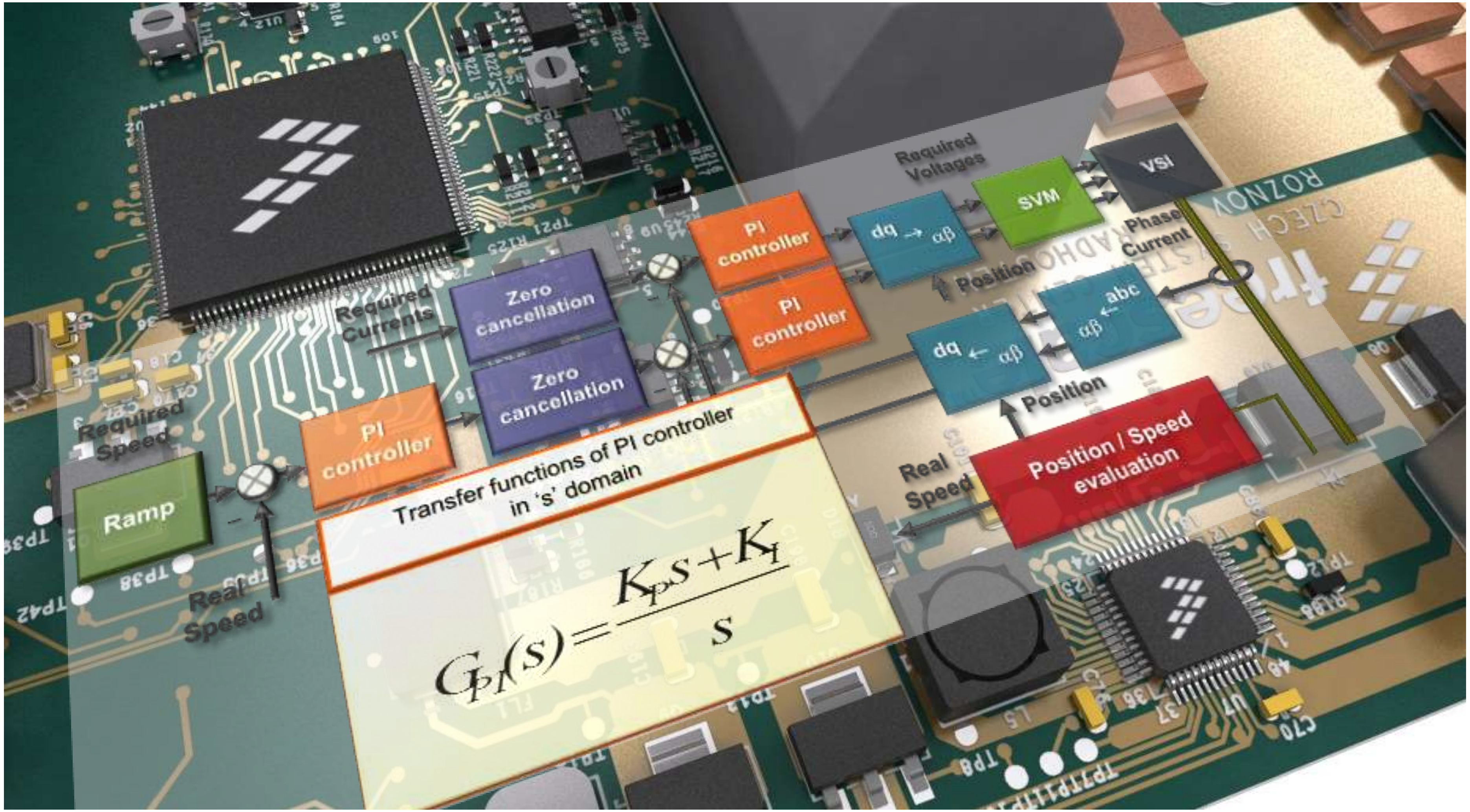
Control Loop Bandwidth & Attenuation



Response Settling Time & Overshoot



Control Loop Controllers



Motor Control Structure in Equations

...op systems. By proper setting of PI controller parameters a predefined
...e reached.

Motor Control Application Tuning tool

What is it?

- It is a user-friendly graphical plug-in tool for FreeMASTER, which can be used for motor control applications debugging with PMSM motors.
- It allows to tune the specific parameters of the motor control application on run-time (no compilation of the code needed).
- Once the right parameters are found, it allows to store them in *.h file.

MCAT Goals

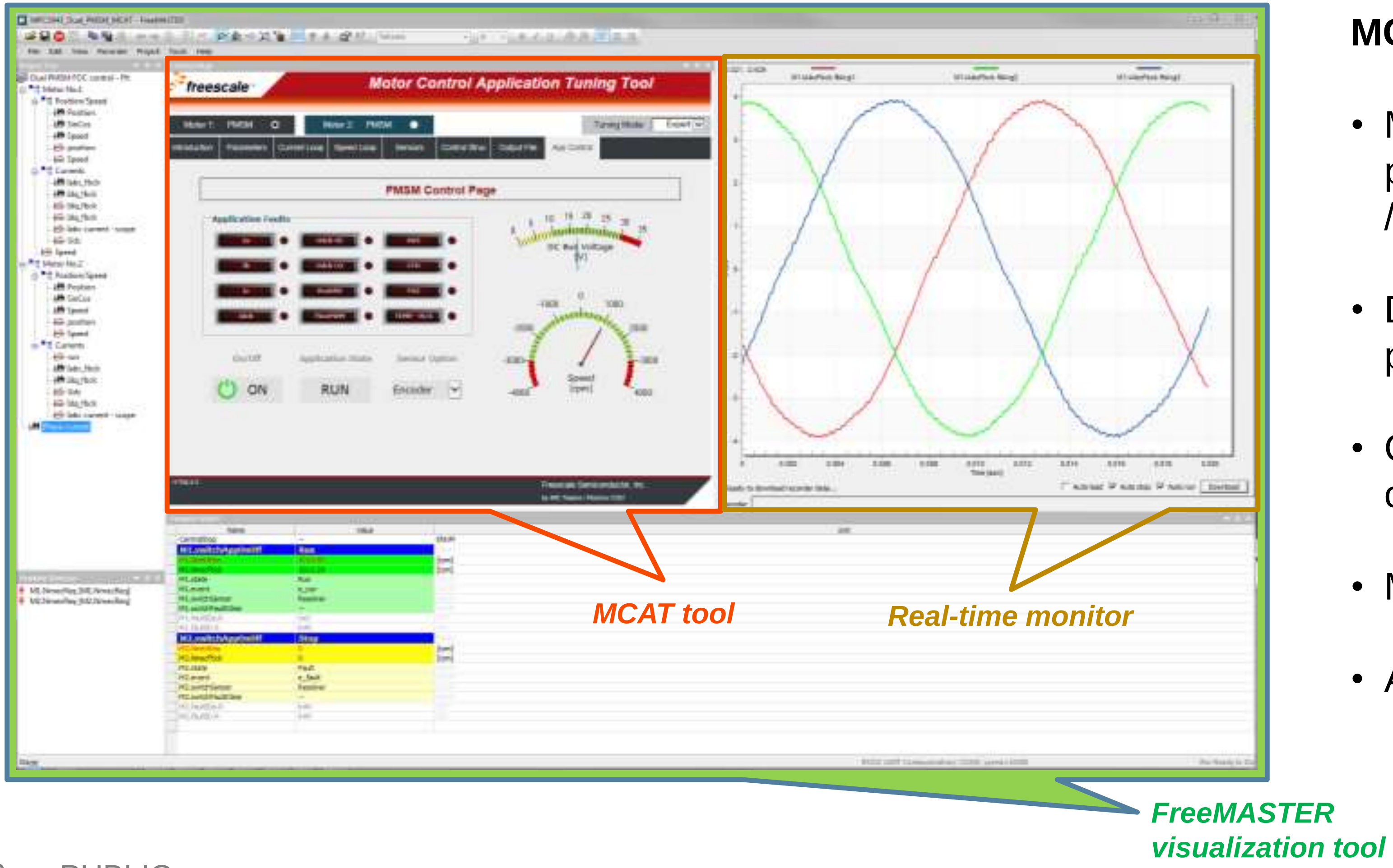
- Goal #1 – PI Controller parameters tuning, to enable customers to make basic motor control tuning activities themselves.
- Goal #2 – control at different levels of a cascade control structure
- Goal #3 – export the *.h file with static configuration of the motor application
- Goal #4 – MCU implementation independency

The application cases

- FOC control of PMSM motor respecting the cascade control structure
- shall be focused on typical application cases, fitting various mainstream industrial & automotive areas. It however shall not be seen nor promoted as solution for every possible corner case

MCAT in FreeMASTER

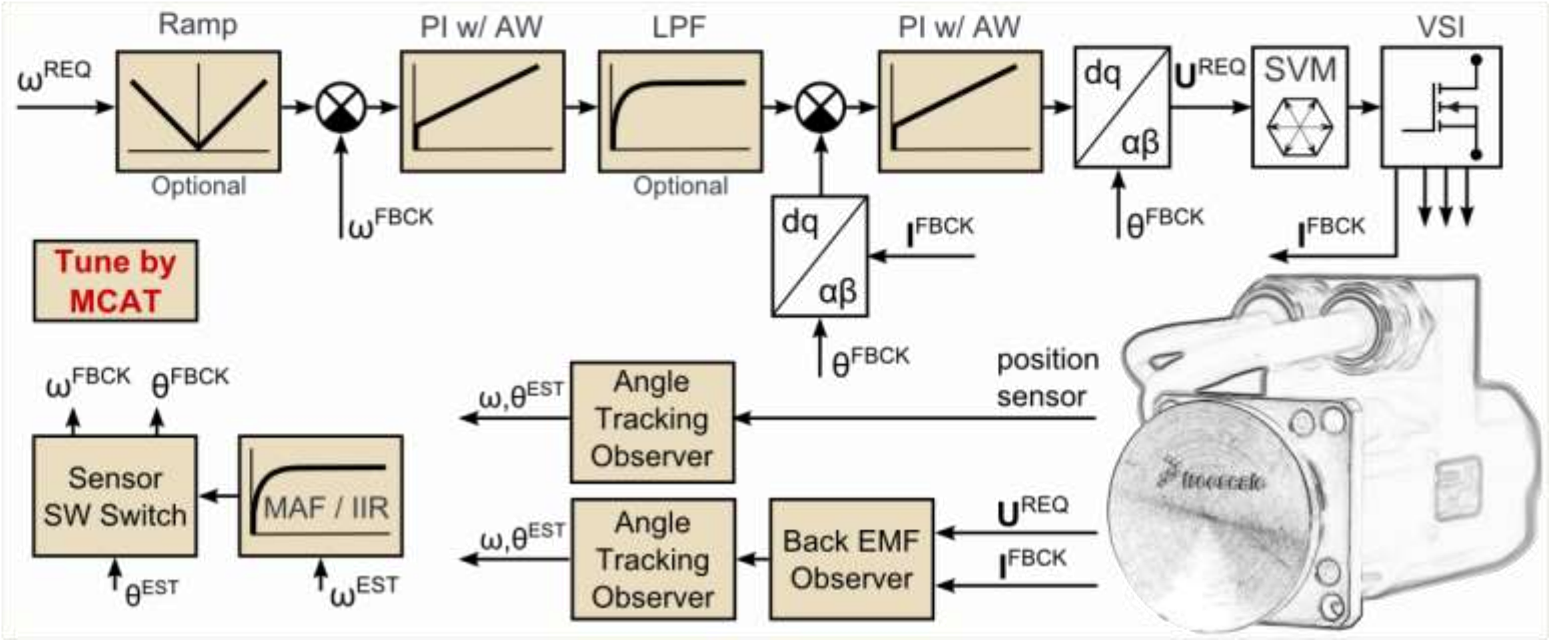
- **MCAT** is a plug-in tool for **FreeMASTER** – NXP's real-time debug monitor and data visualization tool.
- MCAT tool in connection with FreeMASTER allows real-time monitoring, tuning and updating of the control parameters in motor control application.



MCAT features

- MCAT enables tuning of control parameters according to the target motor / application
- Dynamic tuning & update of control parameters
- Generation of header file with static configuration of the tuned parameters
- MCU independent (Kinetis, MPC, DSC)
- Arithmetic independent (16/32bit, Fix/Flt)

MCAT Motor Control Structure Coverage

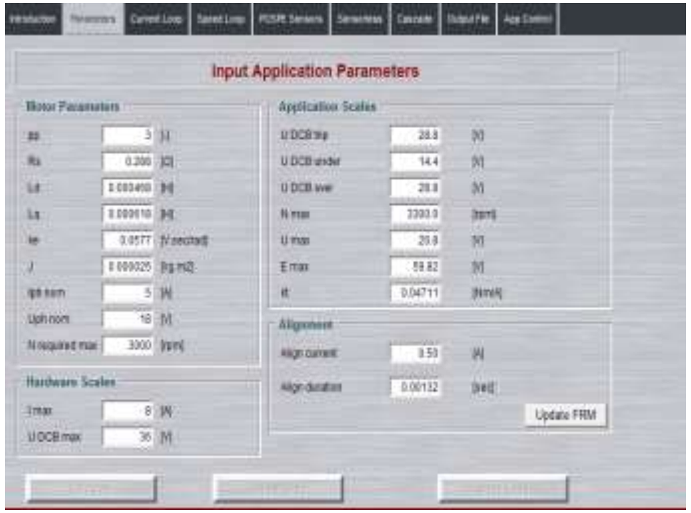


MCAT tool:

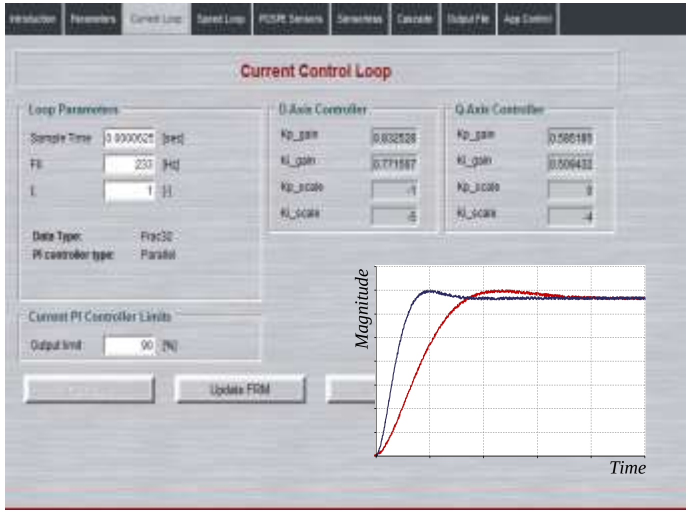
a) supports only standard speed **Field Oriented Control** in a cascade structure
b) is fully compliant with AMMCLib functions API (FLT/FIX32/FIX16 implementation)
c) FOC structure can be extended by optional (filter/ramp) blocks in a feed-forward path
d) supports all types of PI controllers available in AMMClib (parallel, recurrent)
e) calculates the parameters of an ATO for both sensors, resolver and encoder
f) supports sensorless operation, calculates the parameters for BEMF/ATO observers

MCAT - Goal #1 - PI Controller Parameters Tuning

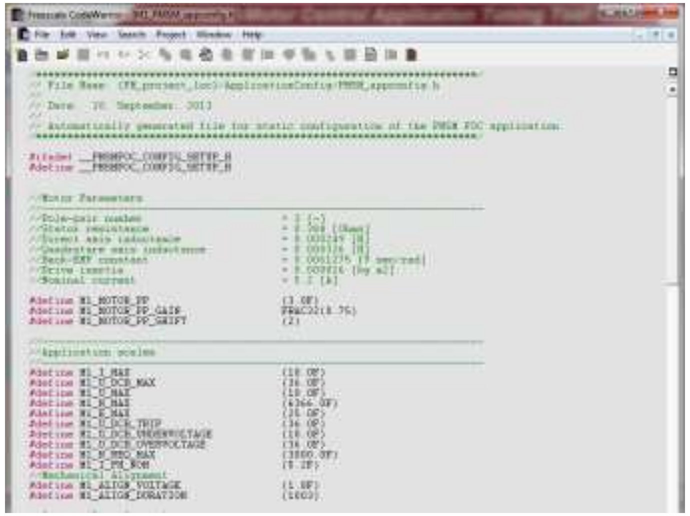
1. Parameter Setting-Up



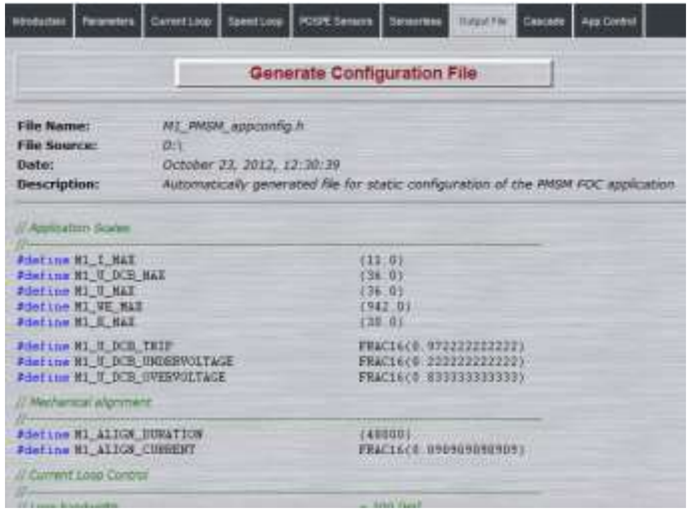
2. Control Loop Tuning



4. Generated static configuration as *.h file

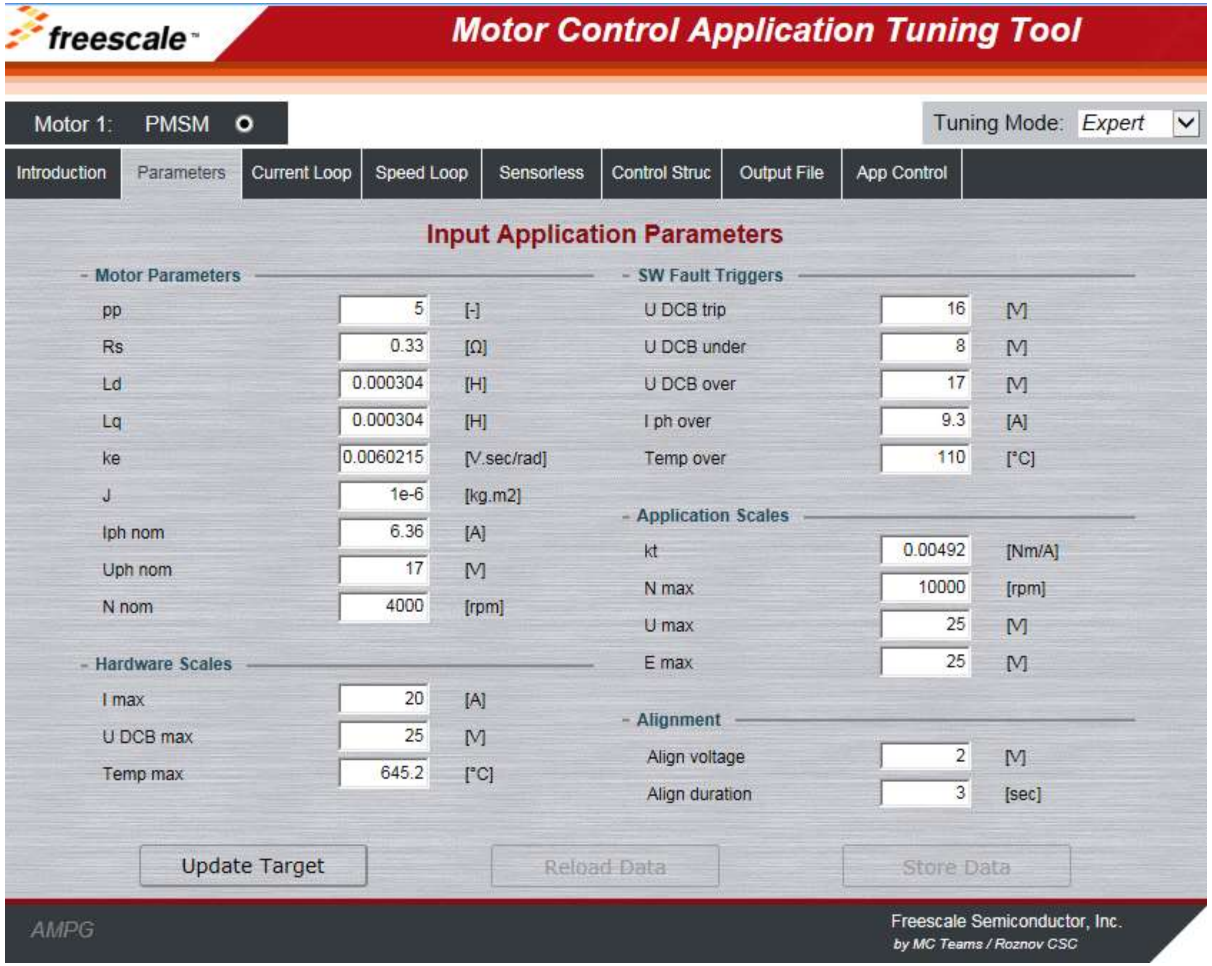


3. Output Control Constant Preview



MCAT - Goal #1 - PI Controller Parameters Tuning

1st step: MCAT input parameters



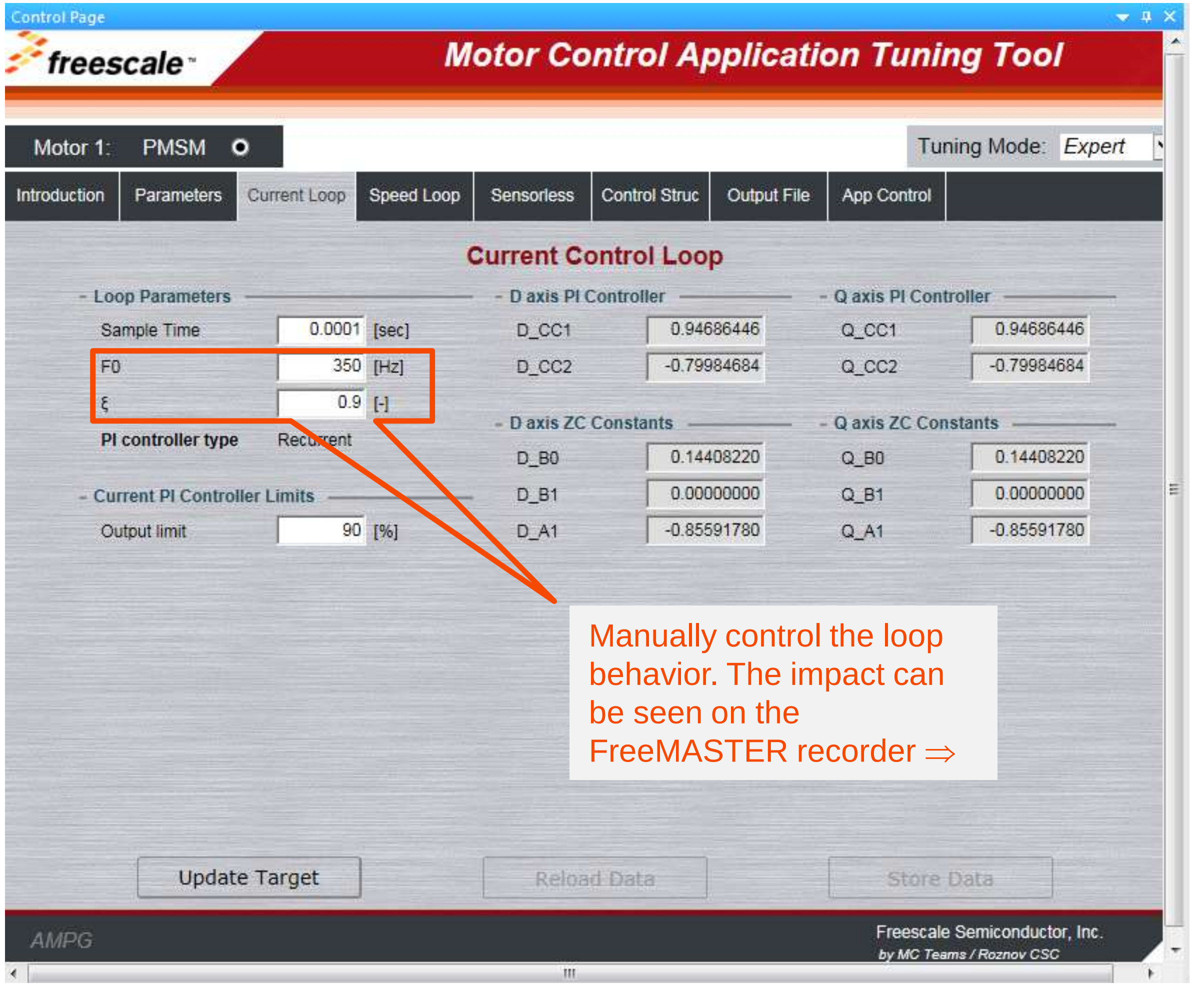
Input parameters tab

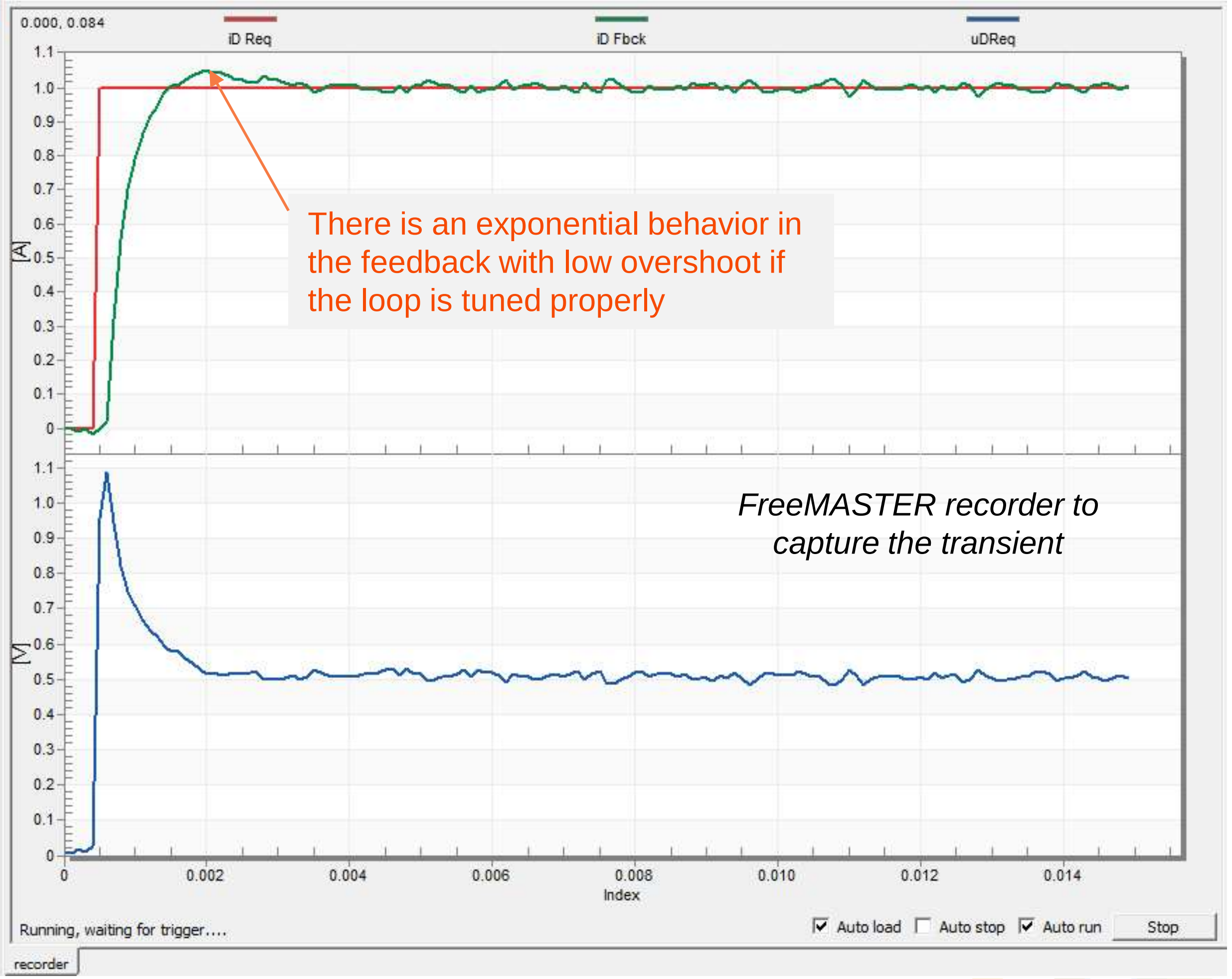
Parameters tab:
User has to enter all input parameters manually, including the motor parameters and application parameters.

Motor parameters
- acquired from the motor manufacturer
- manual measurement / estimation

MCAT - Goal #1 - PI Controller Parameters Tuning

2nd step: MCAT Control Loop Tuning





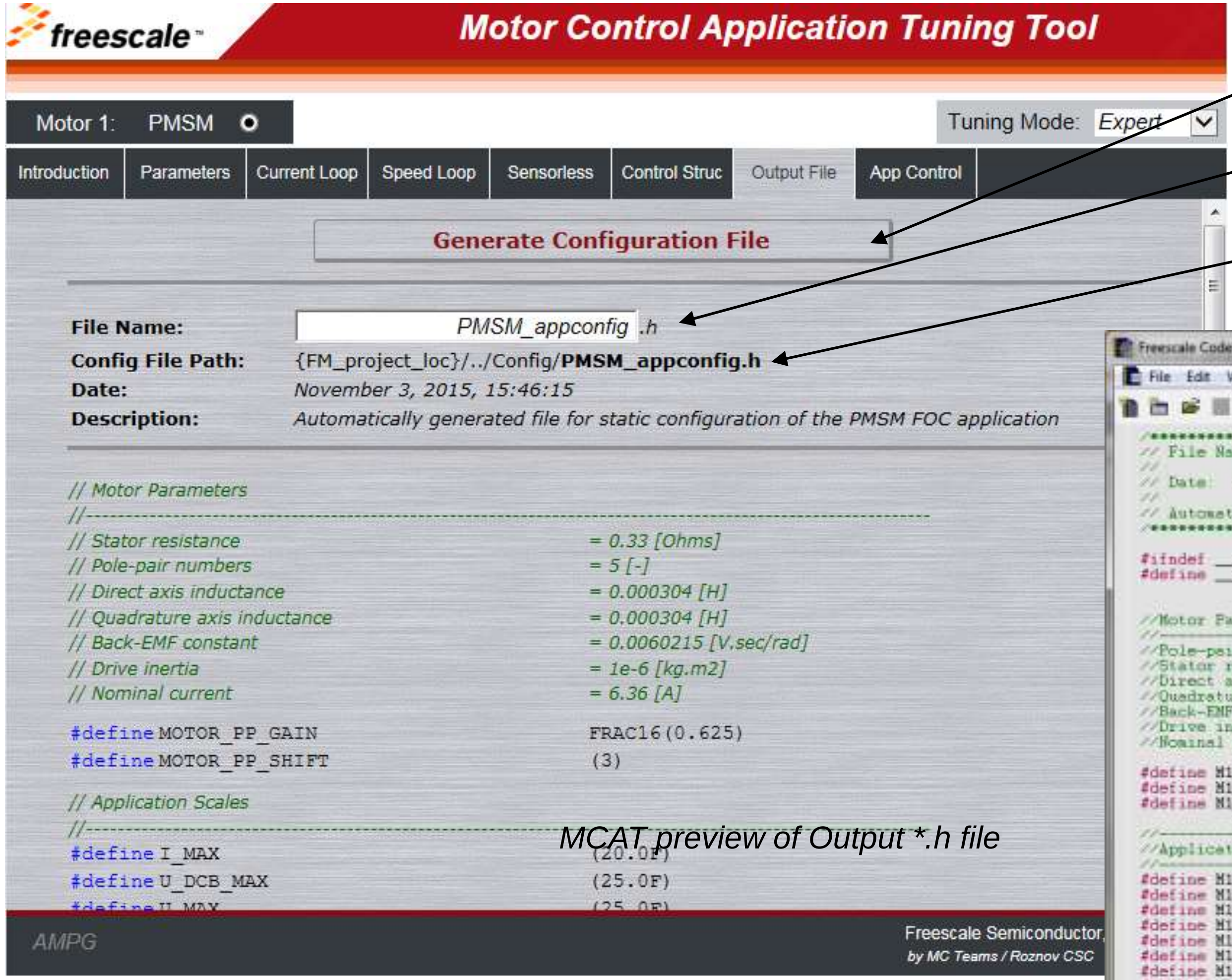
MCAT - Goal #3 – Export App. Static Configuration

3rd step: MCAT preview of Output Control Constant

The output header file is generated by clicking on a button.

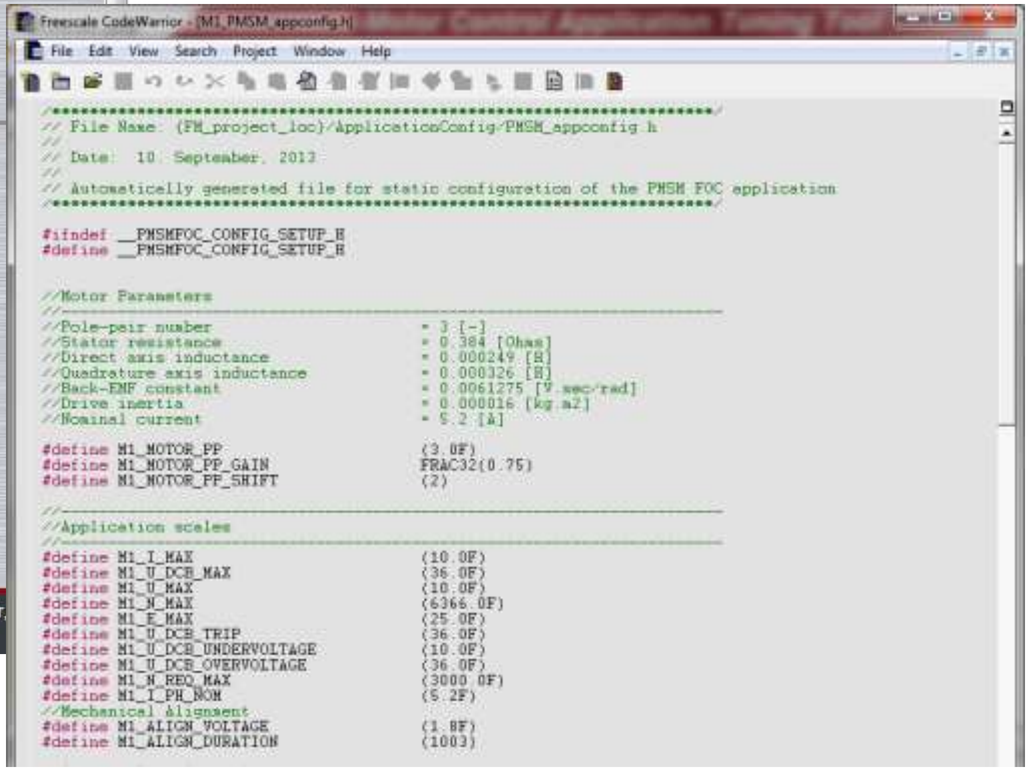
Name of output header file

Location of header file within the project file system

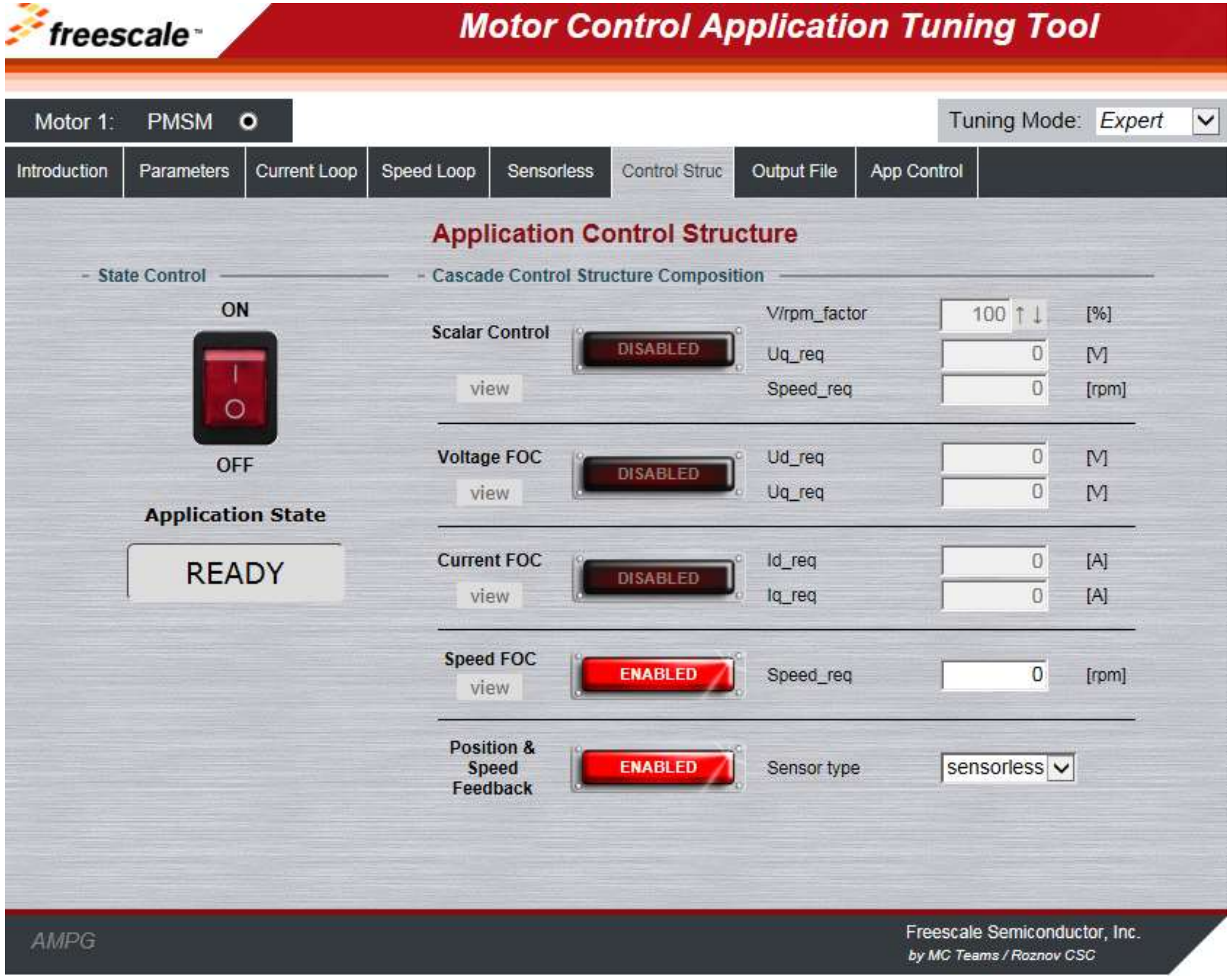


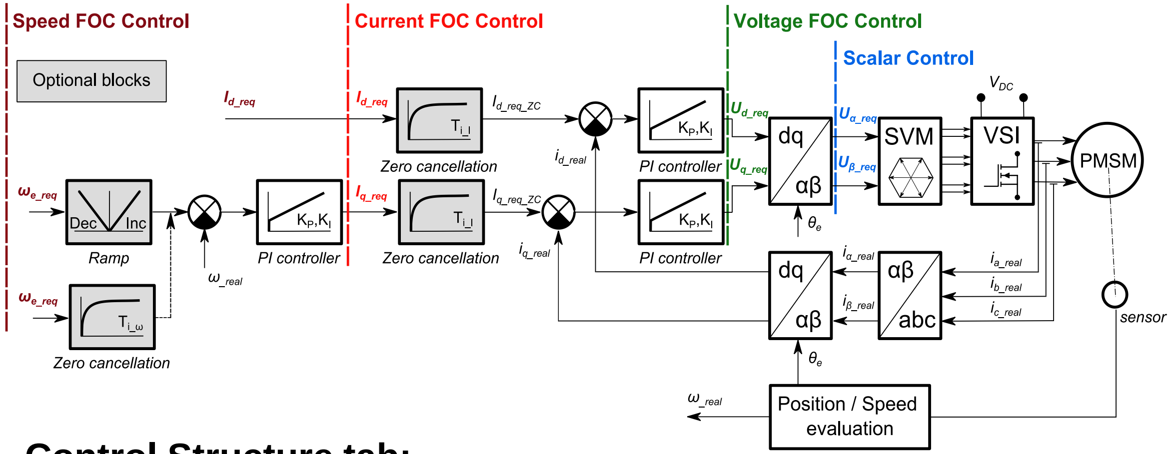
MCAT preview of Output *.h file

Generated *.h file



MCAT - Goal #2 - Control Structure Selection





Control Structure tab:
Offers a huge advantage of controlling the motor at different levels of cascade control structure. This approach is appreciated when new HW setup is arranged. Tuning process starts at lowest level (the most inner loop) and continues up to the most outer loop – speed loop.

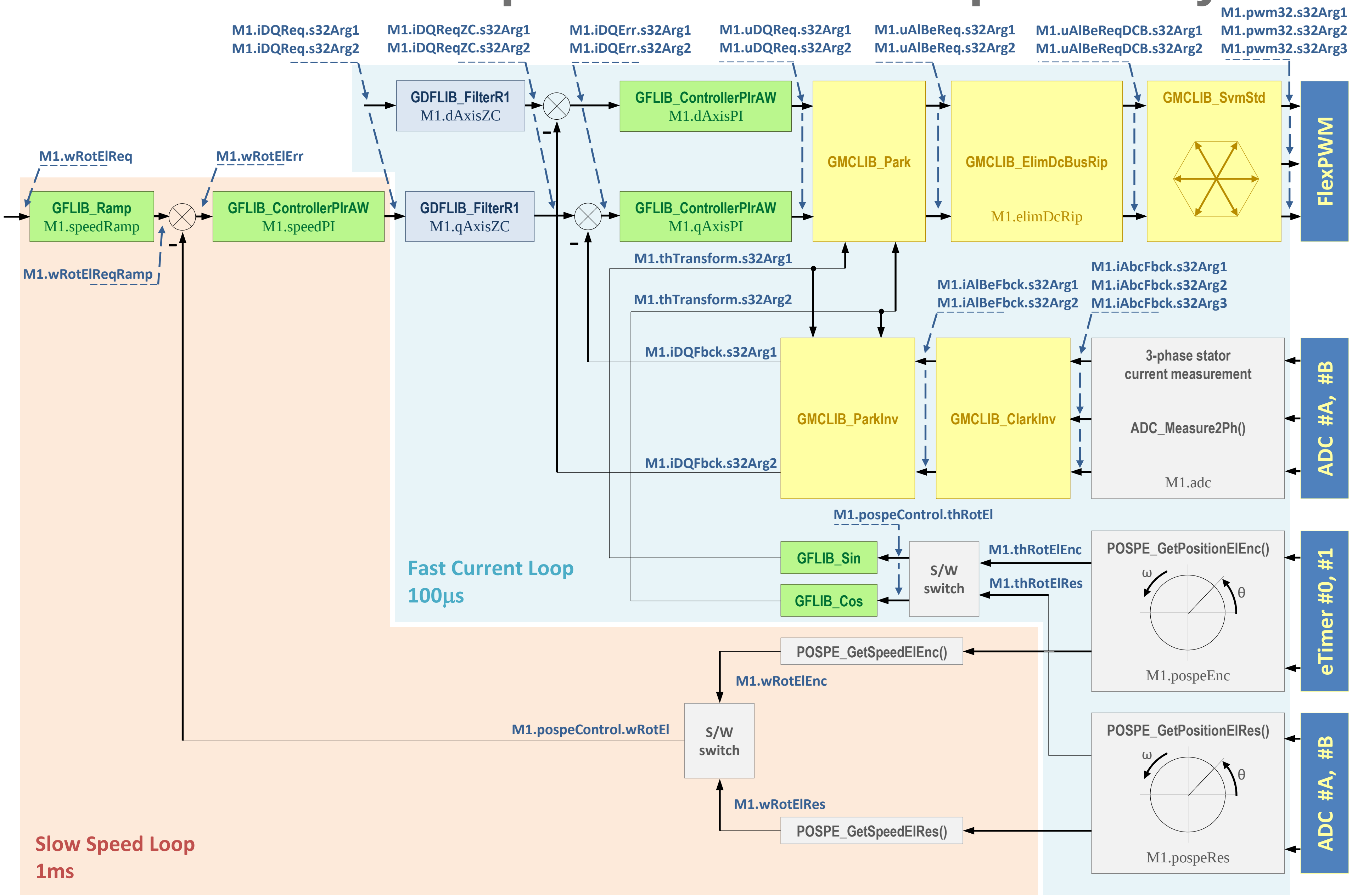
Scalar Control - Open loop control
no need any current, position or speed feedback

Voltage FOC control – position required
no need any current and speed feedback

Current FOC control – current, position required
no need any speed feedback

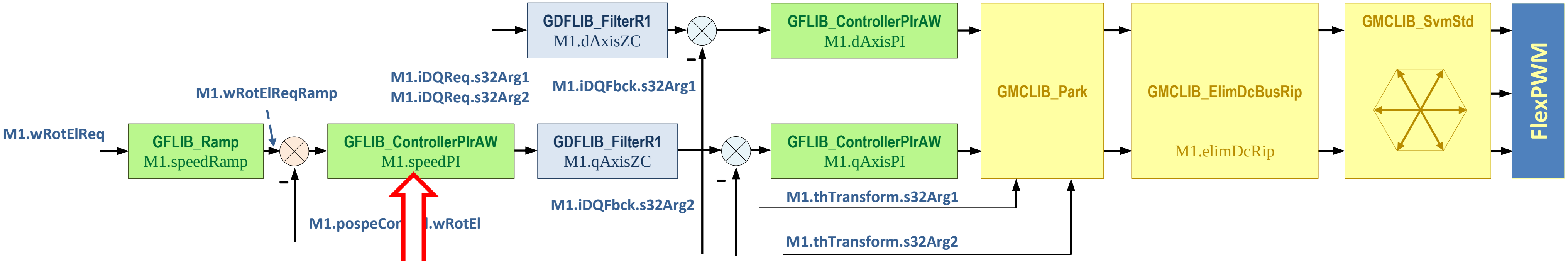
Speed FOC control - current, position and speed required
full speed FOC control with all feedback signals

MCAT - Goal #4 – MCU Implementation Independency



MCAT - Goal #4 – MCU Implementation Independency

This approach makes the MCAT tool independent from the FOC microcontroller implementation. The user has to update variable names in XML file based on the final implementation.



FreeMASTER variable
Updated by user!

MCAT variable
Not changed by user!

Parallel PI

Recurrent PI

Limitation PI

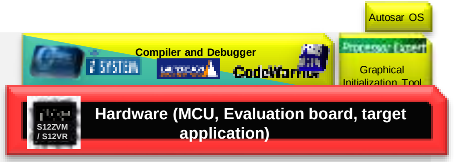
Ramp

Filters
- MAF
- IIR

```
<M1_SpeedPiControll  <!-- Variables for Speed Controller for MOTOR1 -->
  <M1_SL_Kp_g>M1.speedPI.fltPropGain</M1_SL_Kp_g><!-- Parallel PI KP gain -->
  <M1_SL_Kp_sc></M1_SL_Kp_sc><!-- Parallel PI KP gain scale -->
  <M1_SL_Ki_g>M1.speedPI.fltIntegGain</M1_SL_Ki_g><!-- Parallel PI KI gain -->
  <M1_SL_Ki_sc></M1_SL_Ki_sc><!-- Parallel PI KI gain scale -->
  <M1_SL_Nsh></M1_SL_Nsh><!-- Reccurent PI shift -->
  <M1_SL_CC1SC></M1_SL_CC1SC><!-- Reccurent PI CC1 constant -->
  <M1_SL_CC2SC></M1_SL_CC2SC><!-- Reccurent PI CC2 constant  -->
  <M1_SL_UP_LIM>M1.speedPI.fltUpperLimit</M1_SL_UP_LIM><!-- PI High limit -->
  <M1_SL_LOW_LIM>M1.speedPI.fltLowerLimit</M1_SL_LOW_LIM><!-- PI Low limit -->
  <M1_RAMP_UP_g>M1.speedRamp.fltRampUp</M1_RAMP_UP_g><!-- Speed ramp increment UP -->
  <M1_RAMP_DOWN_g>M1.speedRamp.fltRampDown</M1_RAMP_DOWN_g><!-- Speed ramp increment DOWN -->
  <M1_SPEED_RES_MAF_sc>M1.pospeRes.observer.filterMA.u16NSamples</M1_SPEED_RES_MAF_sc><!-- Actual Speed
  <M1_SPEED_ENC_MAF_sc>M1.pospeEnc.observer.filterMA.u16NSamples</M1_SPEED_ENC_MAF_sc><!-- Actual Speed
  <M1_SPEED_SLSS_MAF_sc></M1_SPEED_SLSS_MAF_sc><!-- Actual Speed Filter, number of samples -->
  <M1_SPEED_IIR_B1_g></M1_SPEED_IIR_B1_g><!-- Actual Speed IIR Filter constant -->
  <M1_SPEED_IIR_B2_g></M1_SPEED_IIR_B2_g><!-- Actual Speed IIR Filter constant -->
  <M1_SPEED_IIR_A2_g></M1_SPEED_IIR_A2_g><!-- Actual Speed IIR Filter constant -->
</M1_SpeedPiController>
```

HARDWARE

S12ZVM Ecosystem – Hardware



Motor Control Development Kits

All Motor Control DevKits

- Come with EVB, Powerstage, Motor & SW
- Fully documented SW- and HW-Reference
- Supported by MCAT 1.0

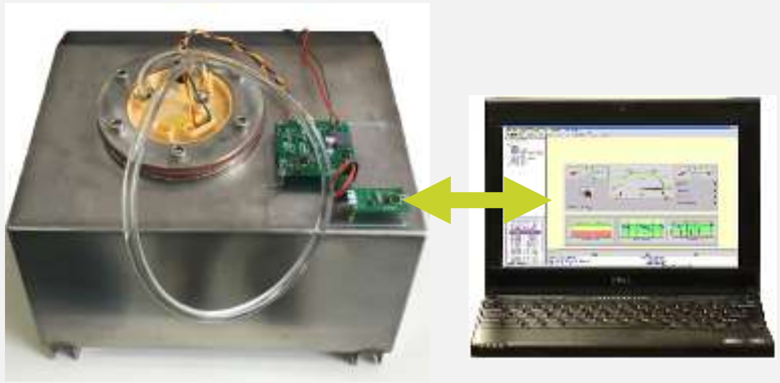
VML128 3-phs Sensor less PMSM DevKit

- New
- Supporting single shunt or dual shunt current sensing

VML128 3-phs Sensor less BLDC DevKit

- Updated
- Supporting sensor-less or Hall-sensor based operation

Demo



S12ZVML31 Fuel Pump Demo

- Based on three-phase sensor less Single-shunt PMSM Motor Control Development Kit:
  - Software ported from S12ZVML128 to S12ZVML31
  - generic motor replaced by a real automotive fuelpump
  - PI-parameters tuned with MCAT tuning tool
- Using NXP LIN-stack
- startup time 0 to 7k RPM in <100ms visualized on PC with FreeMASTER

→***80% reuse of production ready HW / SW***
→***save significant R&D time***

EVB

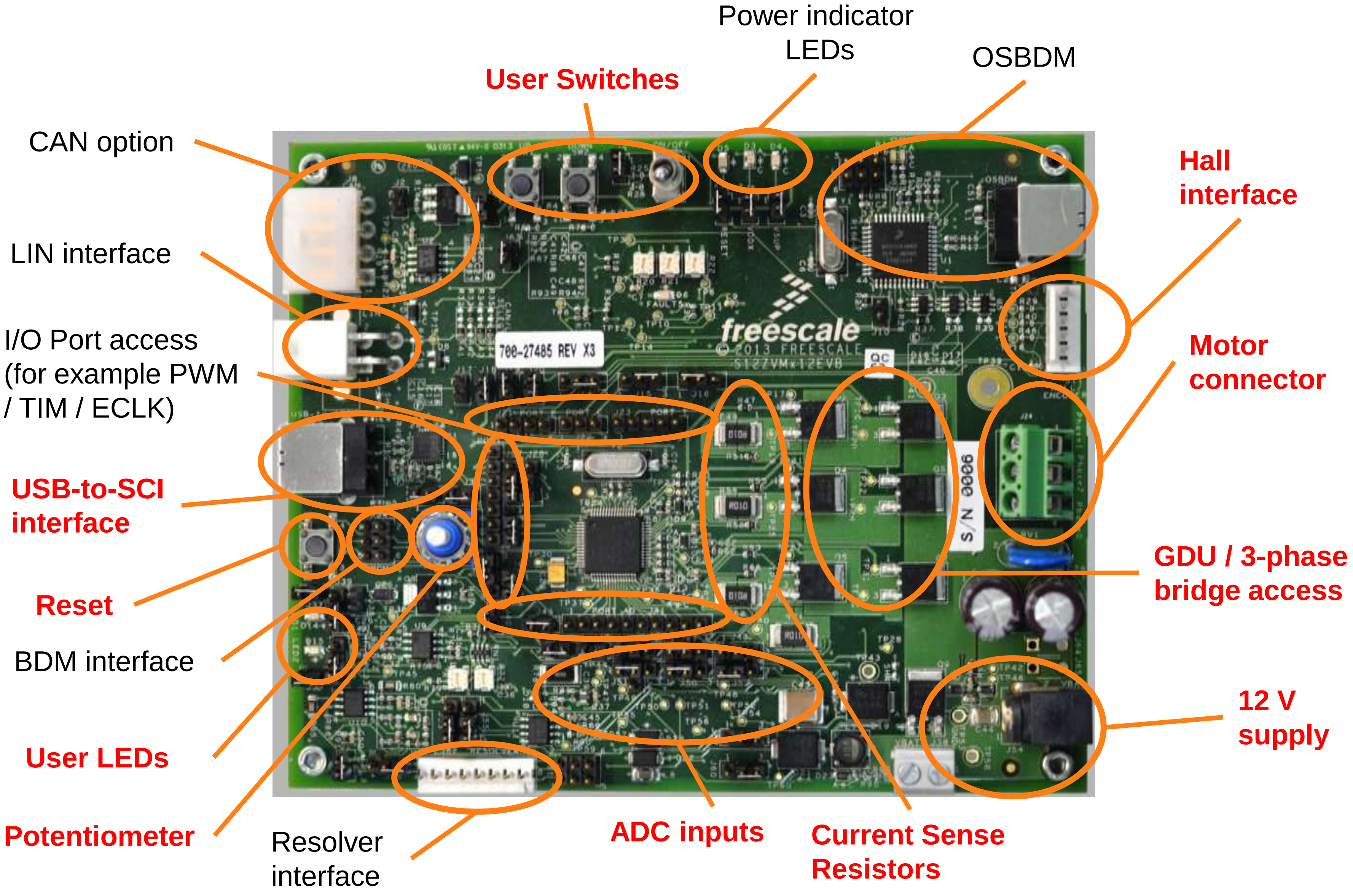
5 New Evaluation Boards (EVBs):

- S12VR32EVB with double relay (as used in windowlifters)
- S12ZVM32EVB with powerstage (6 x N-FET)
- X-S12ZVMC256EVB with powerstage (6 x N-FET)
- X-S12ZVMBEVB with powerstage (4 x N-FET)
- X-S12ZVMAEVB with powerstage (2 x N-FET)

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.

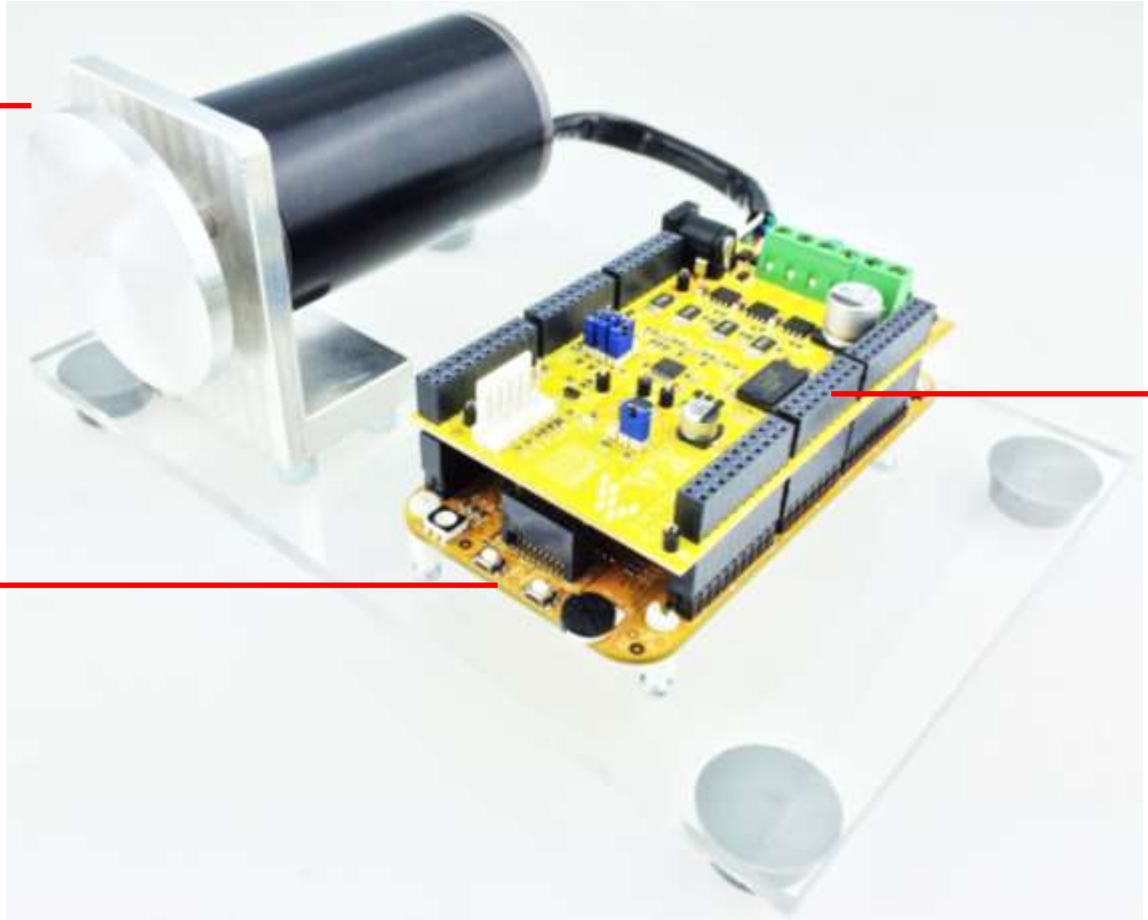
Motor Kit: XS12ZVMx12EVB



S32K144 DEVKIT + MC Shield + BLDC Motor

- KEA128BLDCRD application software is available at www.nxp.com/KEA128BLDCRD
- S32K144 Motor Control Development Kit is not available yet

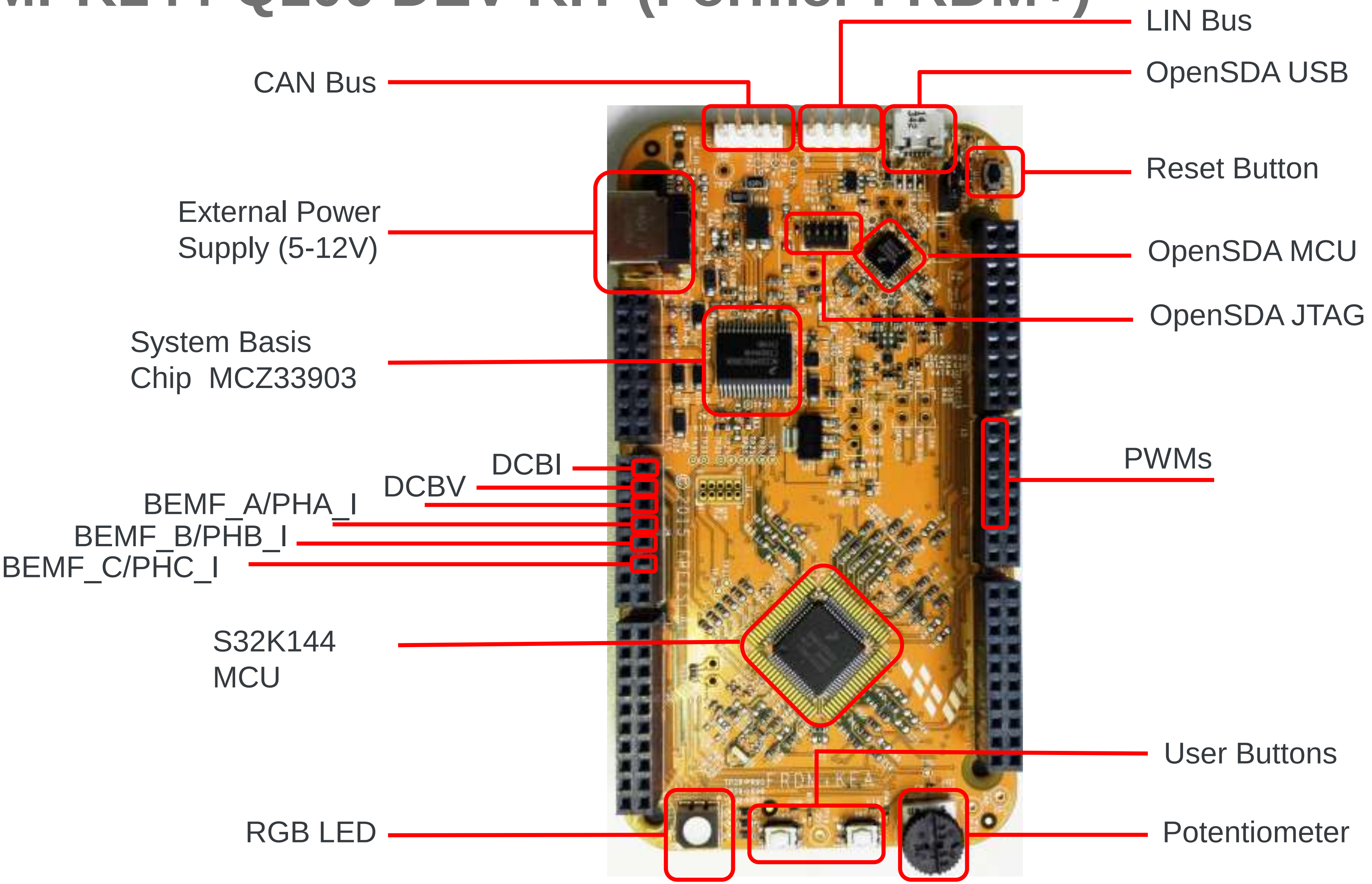
BLDC Motor



Motor Control Shield

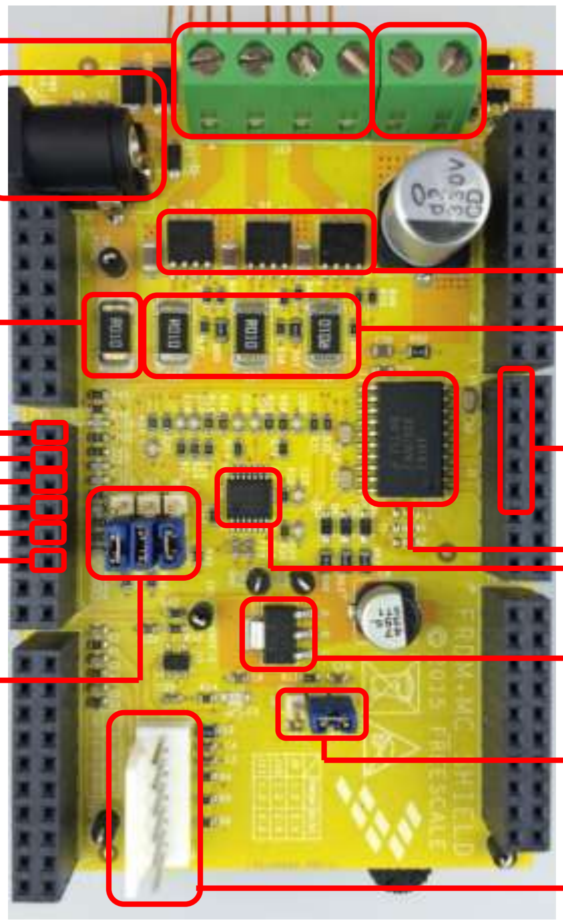
FRDMPK144-Q100

FRDMPK144-Q100 DEV-KIT (Former FRDM+)



Preliminary and subject to change

DEVKIT-MOTORGD (Former DEVKIT-MCSHIELD)



Preliminary and subject to change

SECURE CONNECTIONS
FOR A SMARTER WORLD