PSOC™ Control C3 (PSC3P2xx, PSC3M3xx, PSC3P5xx, PSC3M5xx) – Motor control

Customer training workshop (CTW)

February 2026

Table of contents

Table of contents

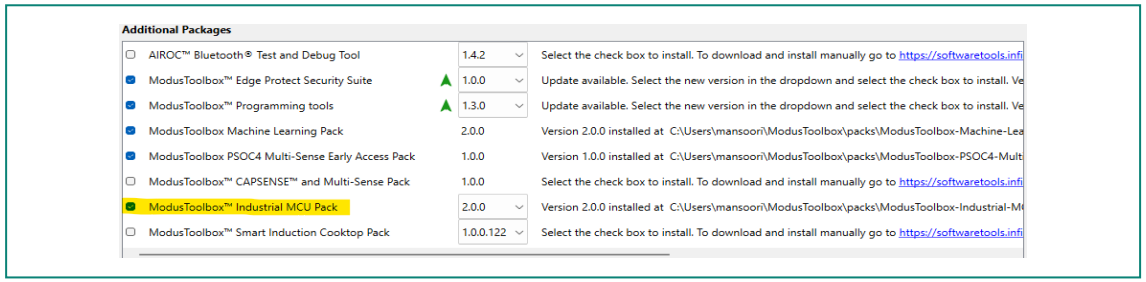
Introduction and prerequisites

Prerequisite

Tools setup

1. ModusToolbox™ installation: ModusToolbox™ software v3.5 or later
   – For more details, see the ModusToolbox™ tool installation guide

2. ModusToolbox™ Motor Suite: this needs to be installed via the industrial MCU pack when ModusToolbox™ (MTB) is installed – use the ModusToolbox™ setup tool



3. Download and install the latest version: Segger J-Link

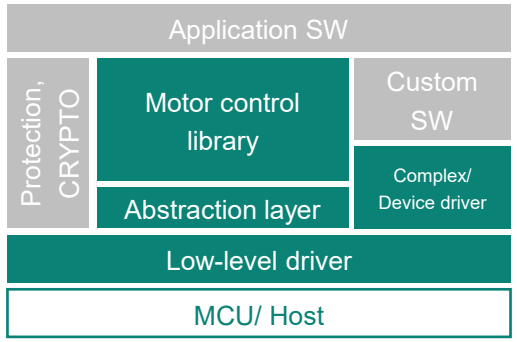
Hardware setup

1. Kit board: KIT_PSC3M5_MC1
2. For more information, see the kit guide: KIT_PSC3M5_MC1 PSOC™ Control C3 Motor Control Evaluation Kit guide

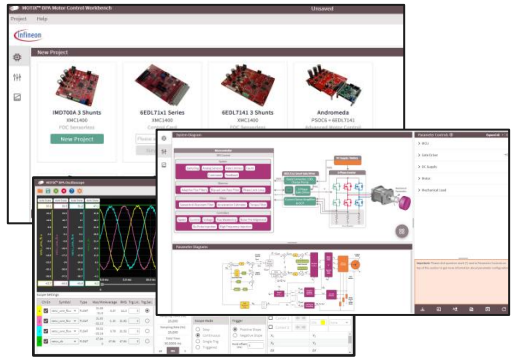
Motor control training snapshot

PSOC™ Control C3 and motor control firmware solution training

- Why PSOC™ for motor control – Peripheral focus- configuration, features and how to use
- Motor control Lib-2.x features and use
- Advance algorithm in details – example HFI, field weakening, advanced PLL



Tools training-GUI



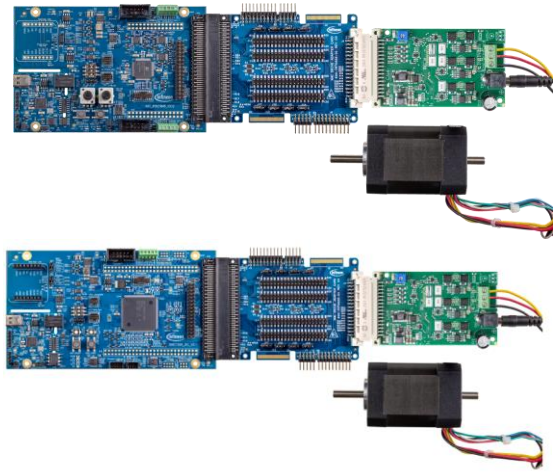
- Tool support for motor control setup

Focus

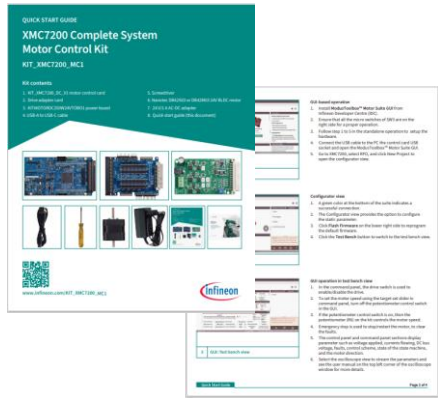
- Motor parameter tuning -configuration
- Static and run-time parametrisation
- Test bench view –control, command, fault, protection panels
- High-sampling-rate oscilloscope
- Dual motor control support
- Motor profiler for auto identification of parameters
- GUI builder for customised GUI

Motor kits HW default /adaptation

- PSOC™ Control C3 complete motor system kit for getting started
- XMC7000 complete motor system kit for getting started



Collaterals know how and how to use it



Application notes

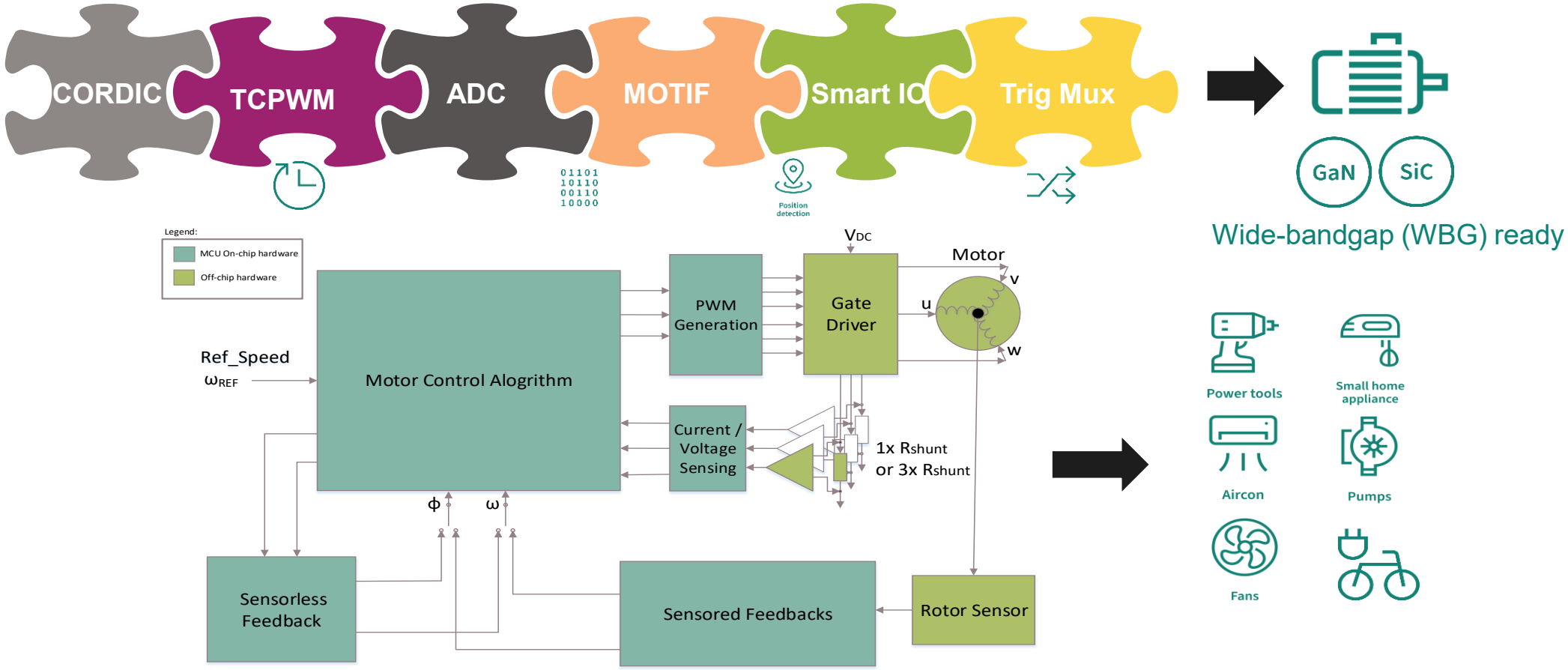
- 1-3 shunt sensorless FOC app note
- Field-oriented control for dual motor

Training material

- Getting started guide for motor control
- E3 level - deep dive training with hands-on for motor control
- Tuning guide for adapting different motors and power cards

PSOC™ Control C3 in motor control applications

Optimized real time peripherals for motor drives



PSOC™ Control C3 main line - Features for motor control vs power conversion

Key features	Benefits
"Idle sampling" 12 MSPS ADC with16 synchronous S&Hs <10 ns comparators with in-built DAC <1 ns latency peripheral inter-connects	– 25% faster – Precise current/voltage measurement – Real-time operation – BOM savings
Timers with <100ps PWM resolution for high Fsw Motor/Power Conversion applications	– High resolution allows clean waveform generation with high f_{SW} scalability – Flexibility for various topologies and driving schemes
CORDIC accelerator	– Trigonometric functions CPU offloading in real-time control tasks (motor control)
128-bit Flash-read, 16 kB I-Cache "Protection Context" Flash partitioning	– Boosted task execution – Up to seven use case configurable, protected flash partitions
Security	– PSA L2, CRYPTO accelerator, TrustZone®, key management, and secure storage
Low power modes	– Deep Sleep <10 µA – Hibernate <1 µA

Positioning

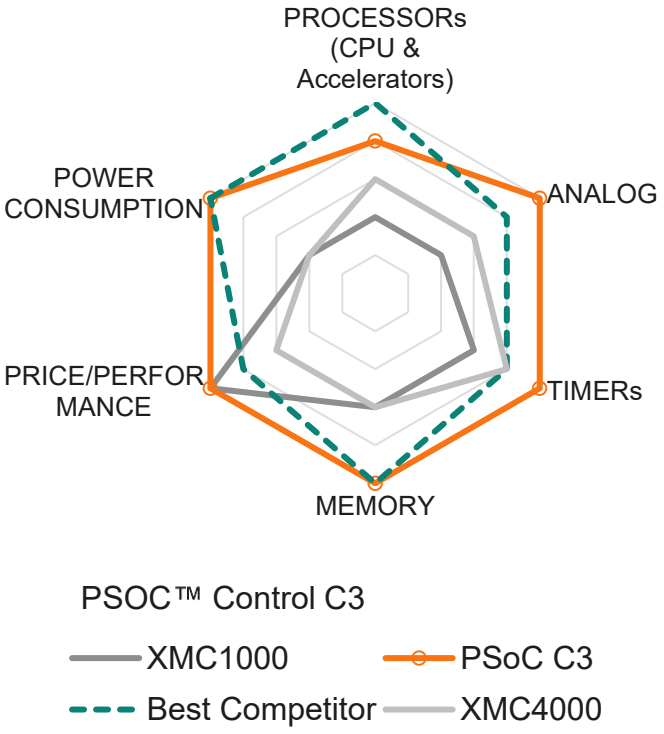
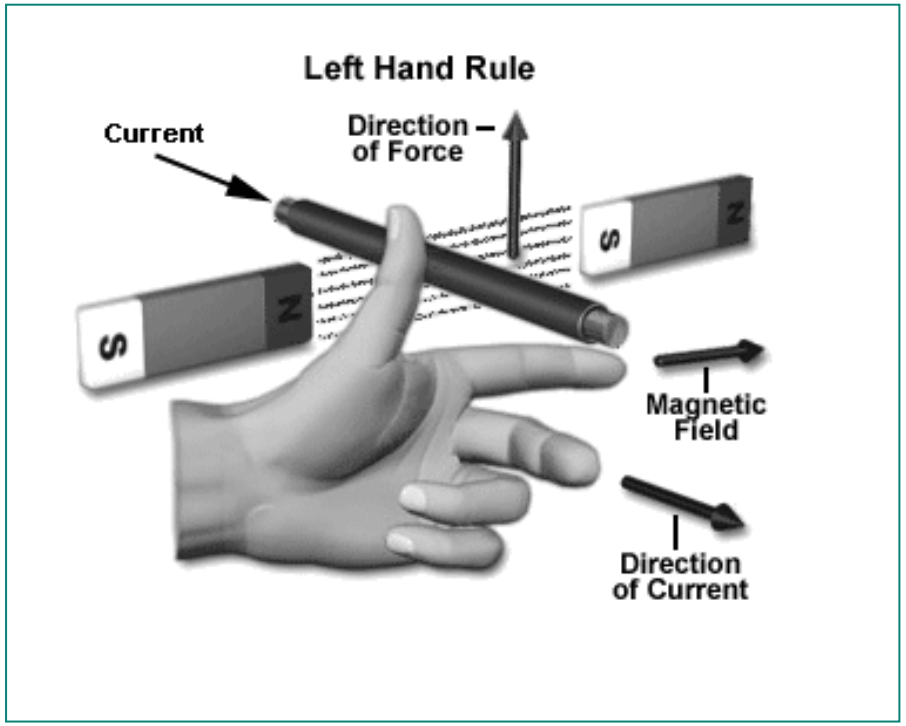


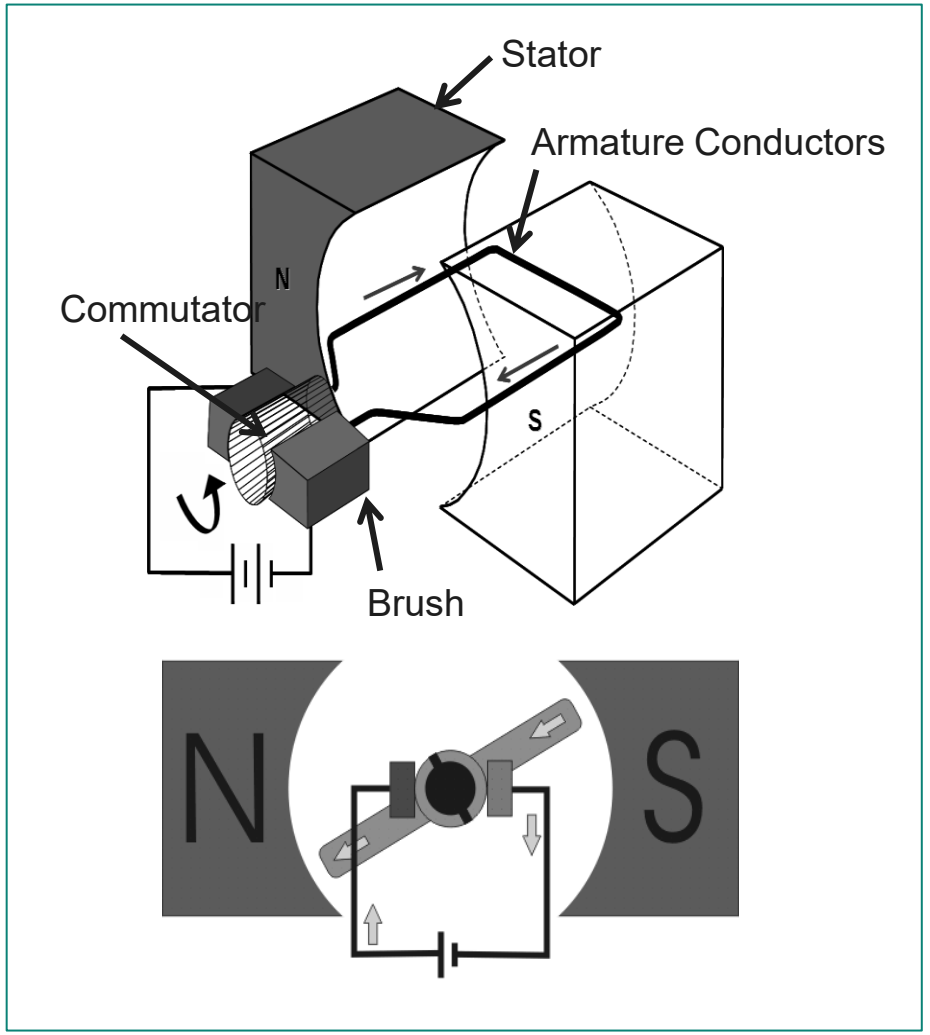
Table of contents

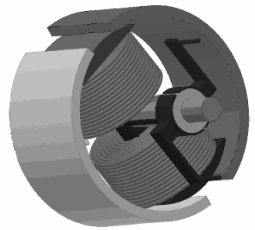
Motor control basics

Principle of operation - DC motor

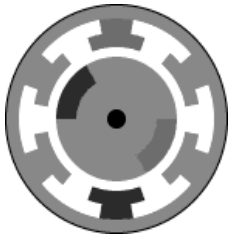
- The current-carrying conductor placed in a magnetic field experiences a force
- Direction of force determined by Fleming's left-hand rule



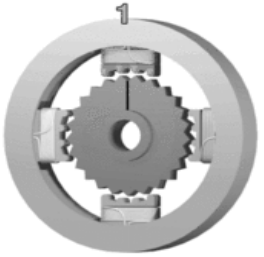




Brush DC Motor



Brushless DC Motor



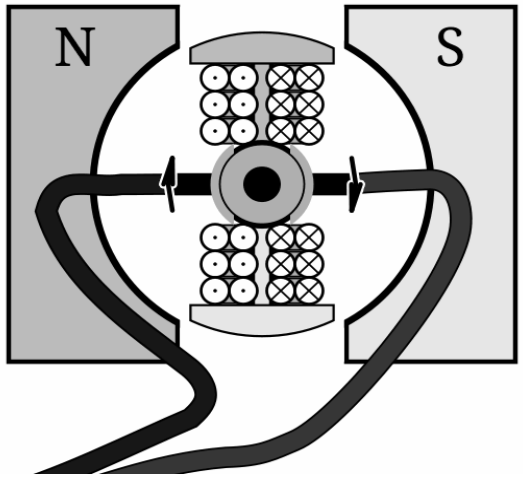
Stepper Motor

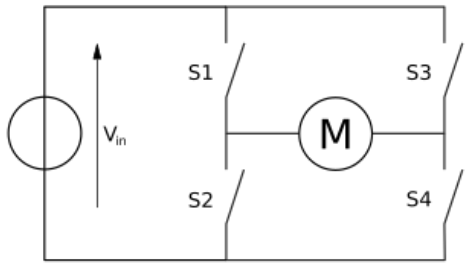
Brushed motor

Applications
- Household appliances
- Industrial machinery
- Air/heat pumps
- Automotive (Seats, wipers, and mirrors)
- Small/cheap motor solutions

Advantages
- Overall cheap solution
- Rebuilding is a common way to extend the life of a product.
- Simple and low-cost controller (or no controller for fixed speed)
- Robust, ideal for use in harsh operating conditions

Disadvantages
- Periodic maintenance is required
- Less efficient than other solutions
- Electrically noisy, arcing can cause EMI issues
- Low RPM range due to mechanical limitations on brushes
- Higher rotor inertia-slow ramp-up/down
- Poor heat dissipation due to the internal rotor construction





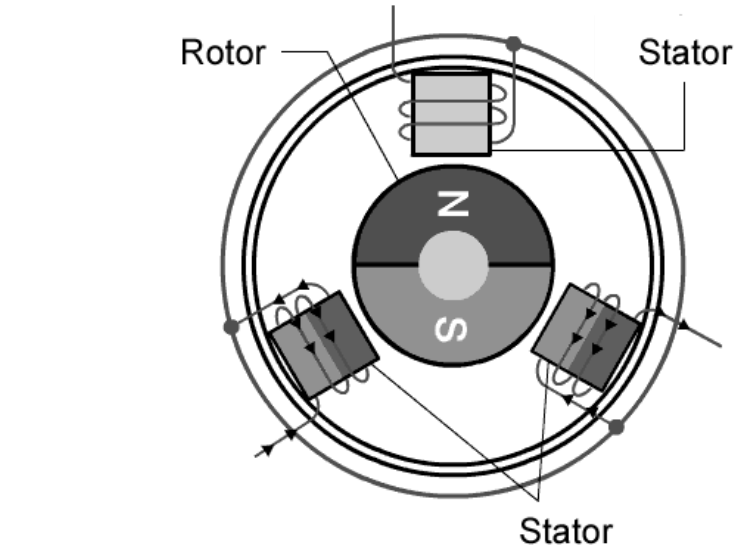
Typical control topology H bridge

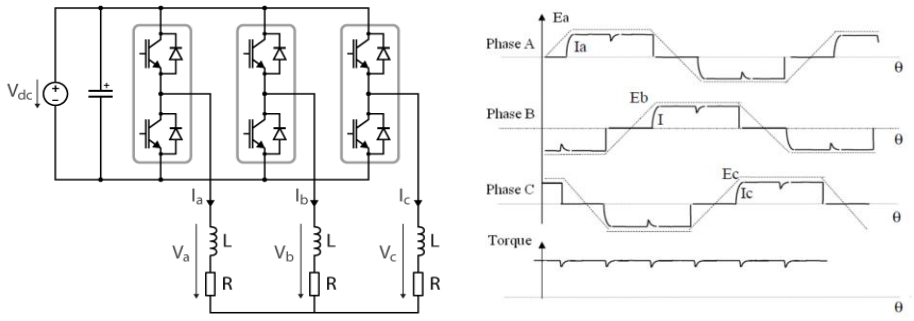
Brushless DC motor (BLDC): Trapezoidal commutation

Applications
- Power tools
- Household appliances
- Electrical vehicles
- PC equipment (Fans, CD Player)
- Medical tools (Dentist/surgical tools, respirators)
- Small/cheap motor solutions
- Robotics (Cobots, RC planes/drones, and multi-purpose robots)

Advantages
- No brushes – no maintenance required
- No arcing – low operating noise
- Better performance – High efficiency and lower inertia
- Compact size
- Better EMI

Disadvantages
- Higher cost than the brushed solution
- High vibrations on low-speed, can go into resonance – loud
- Complex controller needed for commutation
- Longer development
- Requires position control



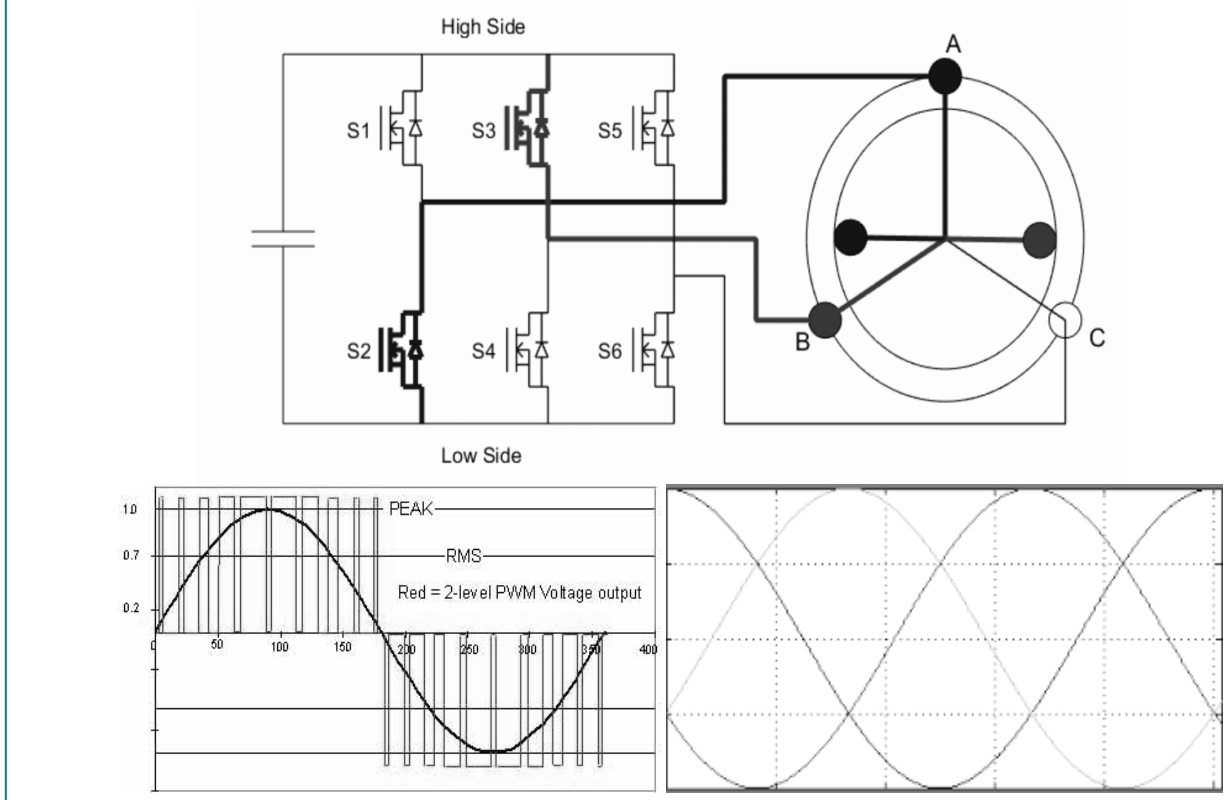


Permanent magnet synchronous motor (PMSM): Sinusoidal commutation - FOC

Applications
- High power tools
- Robotics (RC planes, drones)
- Robotics (Cobots, RC planes/drones, and multi-purpose robots)
- Application requiring silent operation

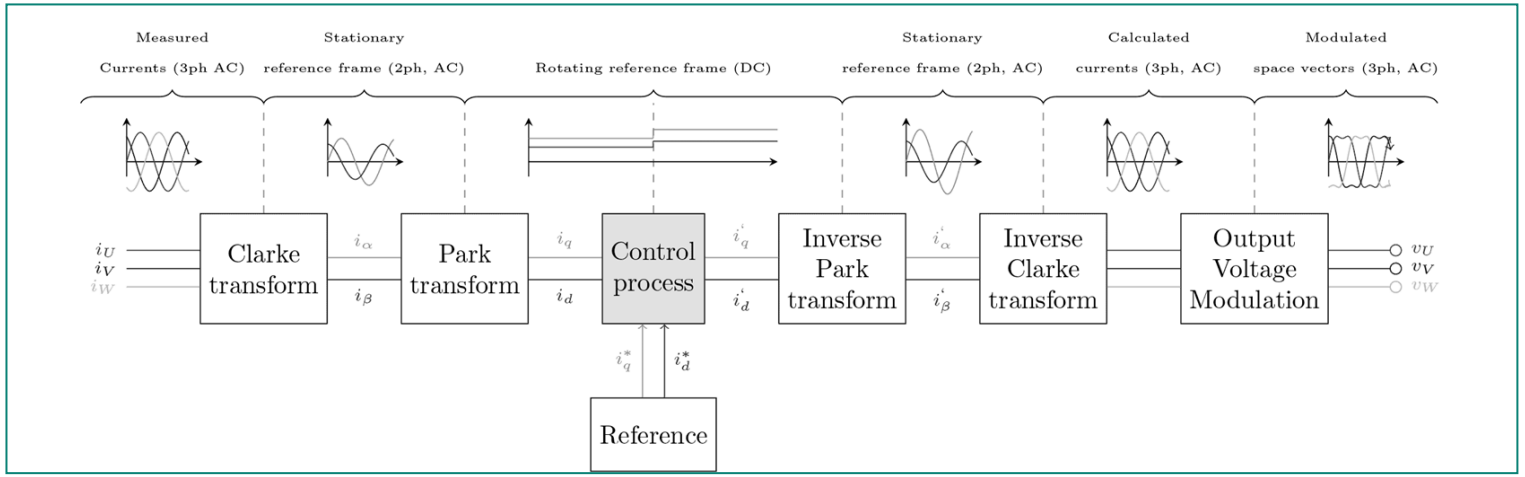
Advantages
- Almost no torque ripple
- Less noise than BLDC
- Higher efficiency than BLDC
- Better EMI

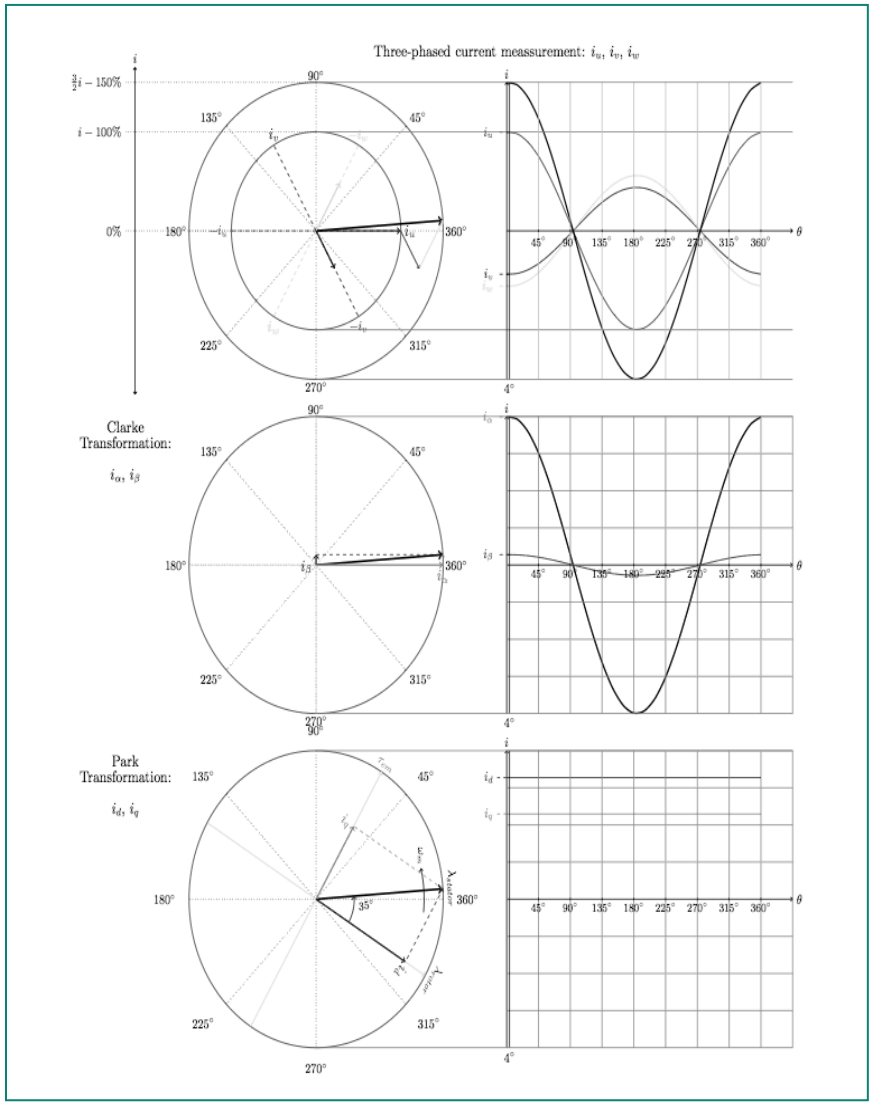
Disadvantages
- Higher control complexity than BLDC
- Requires driver (usually a complex one)
- Requires very precise position control



Motor control algorithm - FOC (Vector control) flow

- 3-phase stator currents are identified as two orthogonal vector components
  - Torque, given as the amount of active current
  - Magnetic field (flux), given as reactive current
- The control system of the drive calculates the current component references for the flux and torque references given by the drive's speed control
- PI controllers are used to keep the measured current components at their reference values
- The pulse-width modulator defines the transistor switching according to the stator current references



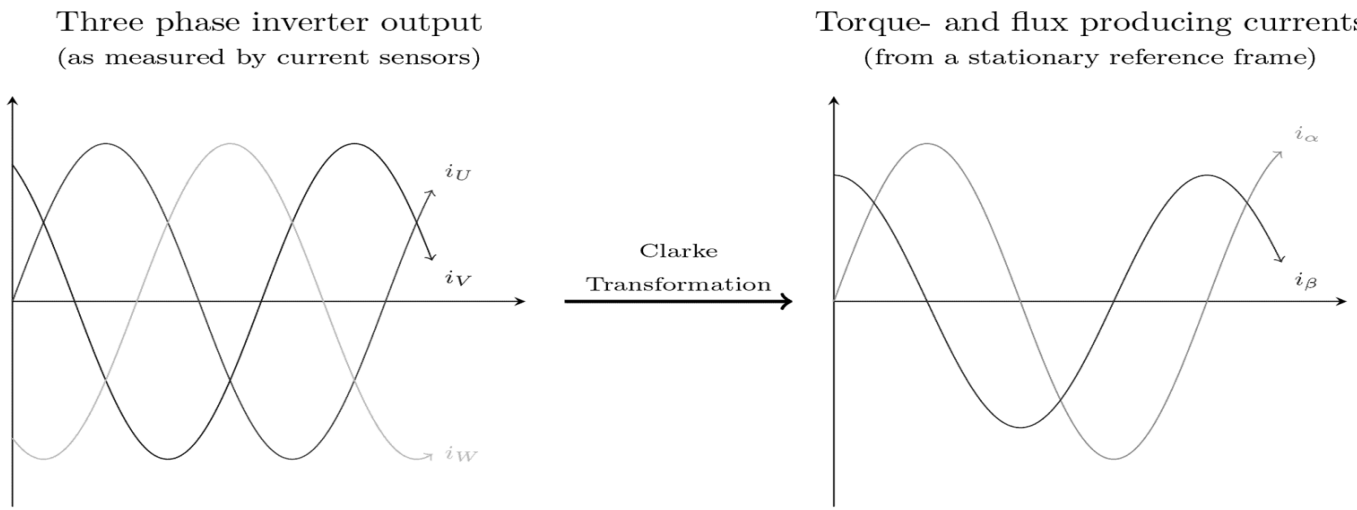


Clark transform

- The Clarke transform creates two orthogonal currents from a 3-phase measurement. The two new currents represent torque-producing and magnetic field-producing currents

where,

- i_U, i_V and i_W represent the 3-phase currents out of the inverter and
- i_α and i_β represent the two "new" currents for torque and flux, respectively



Stationary reference frame

$$i_\alpha = \frac{3}{2} \cdot i_U$$

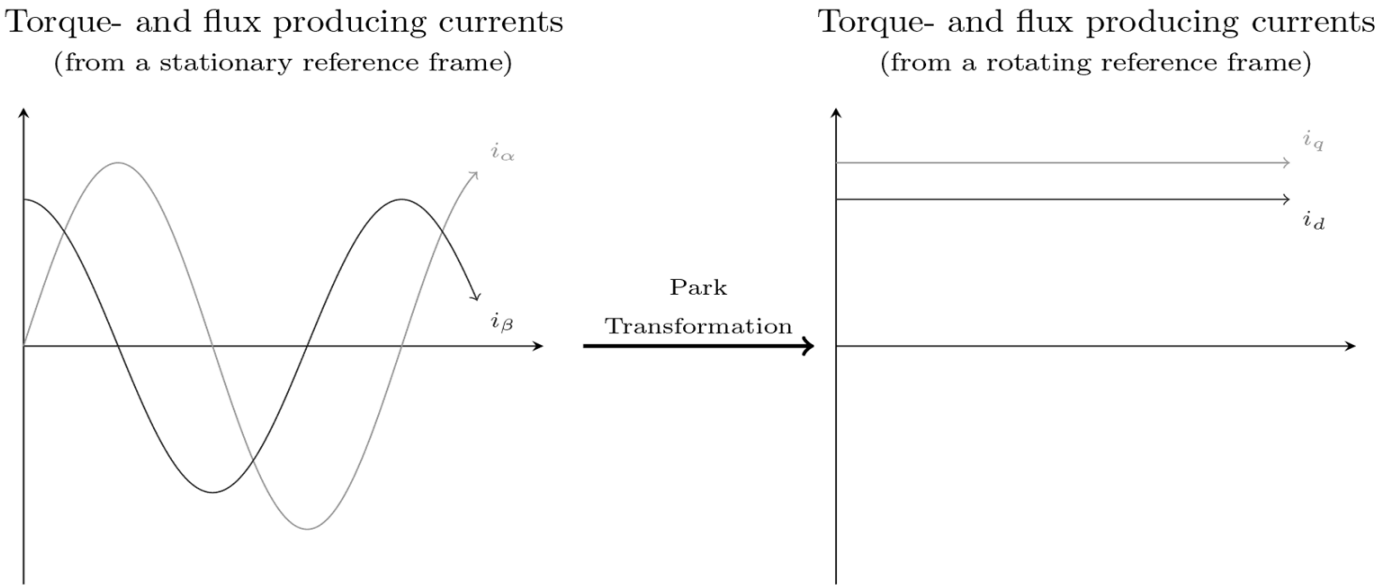
$$i_\beta = \frac{\sqrt{3}}{2} \cdot i_V - \frac{\sqrt{3}}{2} \cdot i_W$$

Park transform

- The Park transformation converts the two alternating currents into steady-state DC currents and makes it ridiculously easy for the controller to control the machine using an ordinary PI or PID control loop

  where,

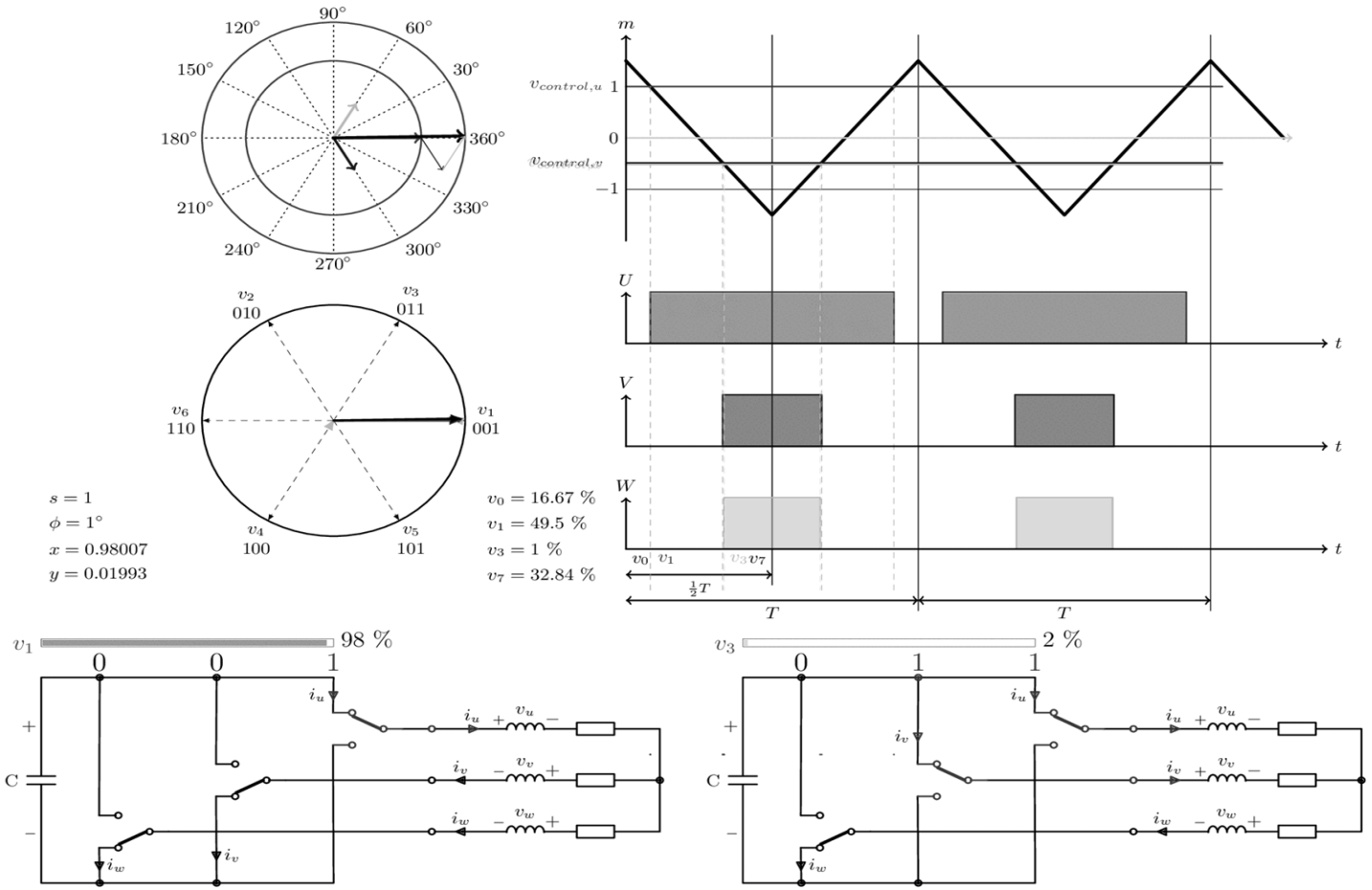
  - i_q is the torque-producing
  - i_d is the field-producing DC currents, and
  - Θ is the rotor angle



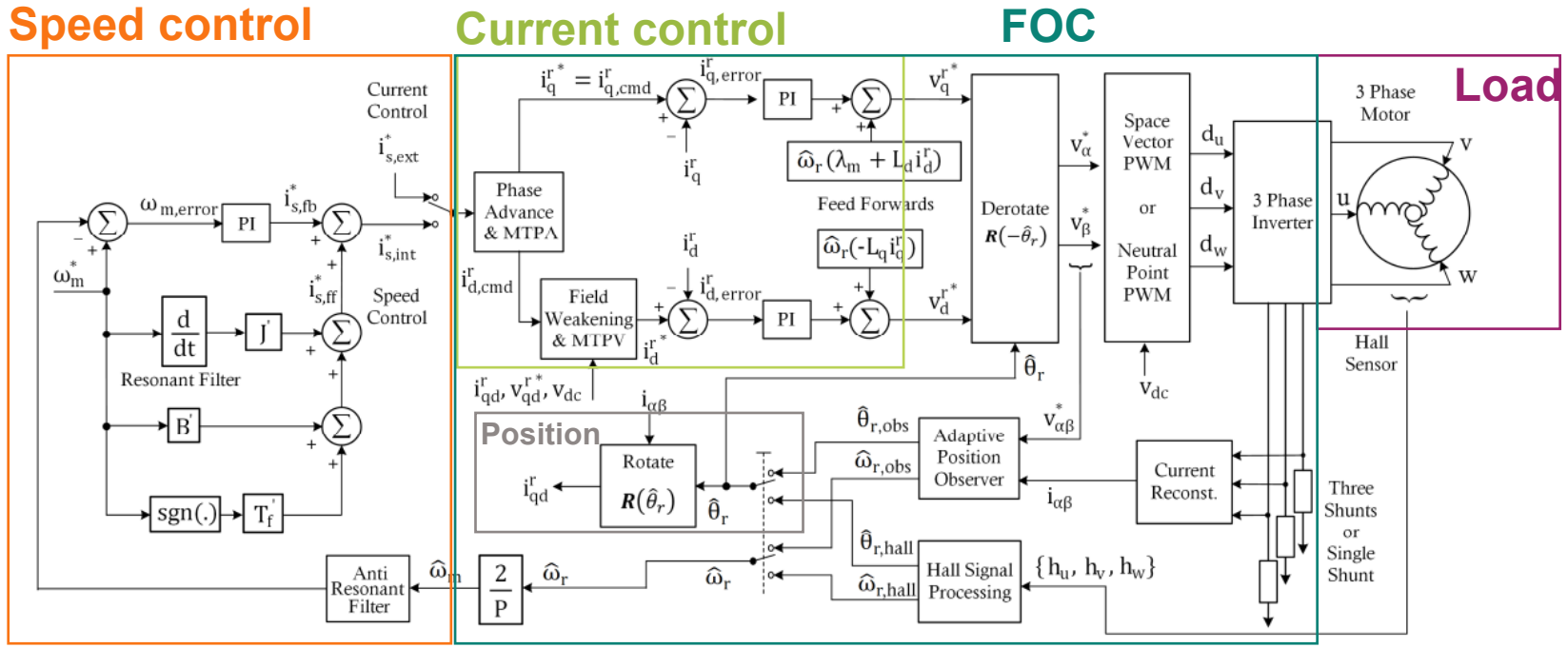
$$i_q = -i_\alpha \cdot \sin\Theta_d + i_\beta \cdot \cos\Theta_d$$

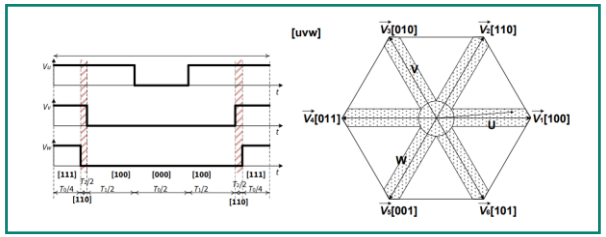
$$i_d = i_\alpha \cdot \cos\Theta_d + i_\beta \cdot \sin\Theta_d$$

Modulator



Motor control library – Block diagram for sensorless FOC





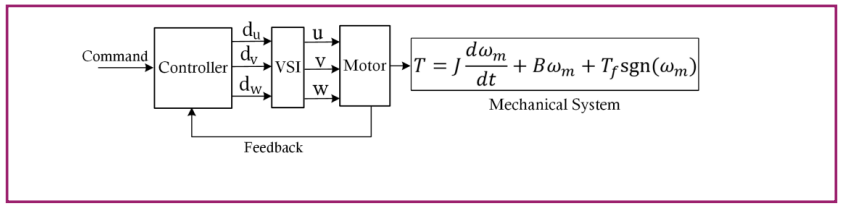


Table of contents

Motor control software overview

Motor control software overview

– Motor control software supports to drive 3-phase brushless motor (BLDC) and permanent magnet synchronous motor (PMSM) including surface mounted and interior mount magnet motors

Control method

– Rotor frame-oriented field-oriented control (RFO)
– Stator frame-oriented field-oriented control (SFO)
– Trapezoidal/block commutation (TBC)

Development environment

– ModusToolbox™- Peripheral configuration and IDE
– Compilers: GCC, IAR
– Motor Control Suite for parameter configuration and runtime debug

Key features

– Advanced RFO and SFO observer for rotor position estimation
– Hall and encoder-based rotor angle estimation
– Leg shunt and single shunt current measurement
– Different startup methods for Sensorless FOC
– 3-phase and 2-phase sinusoidal PWM modulation
– Field weakening, MTPA and profiler support
– Algorithm is implemented in floating point base (compliant with the ANSI/IEEE Std 754-2008) - No extra bit-shift or overflow and underflow checks

Motor control software features

Control method

- Rotor frame-oriented field-oriented control (RFO)
  - Control modes: Speed control, current control; 3x/1x current measurement
  - Position estimation: Observer-based angle estimation (Sensorless), Hall sensor (three digital hall), and incremental encoder
- Stator frame-oriented field-oriented control (SFO)
  - Control modes: Speed control, torque control; 3x/1x current measurement
  - Position estimation: Observer-based angle estimation (Sensorless)
- Trapezoidal/Block commutation (TBC)
  - Control modes: Speed control; 3x/1x current measurement
  - Position estimation: Hall sensor (three digital hall)

Startup method for sensorless FOC

- Pre-align the rotor into know position
- Constant V/F control
- Inductance-based initial rotor angle estimation
- Dyno mode to catch the free-running motor
- High frequency injection for rotor angle estimation

Protection/fault handling

- Under/overvoltage protection
- Overcurrent protection
- Over speed and overtemperature protection
- Motor I^2T protection

Control methods

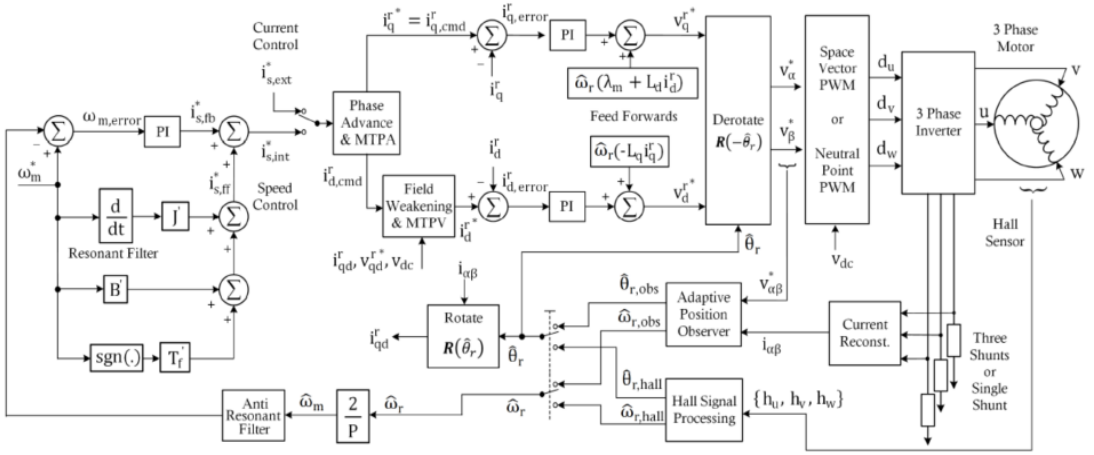
Control Type	Controlled Entity	Feedback Type	Startup Method
Open-loop	Voltage	N.A.	N.A.
FOC in RFO	Current	Sensorless	Rotor Pre-Alignment
FOC in RFO	Current	Sensorless	Six Pulse Injection
FOC in RFO	Current	Sensorless	High Frequency Injection
FOC in RFO	Current	Sensorless	Dyno Mode
FOC in RFO	Current	Encoder	Rotor Pre-Alignment
FOC in RFO	Current	Hall Sensor	N.A.
TBC in BC	Current	Hall Sensor	N.A.
TBC in TC	Current	Hall Sensor	N.A.
FOC in SFO	Torque	Sensorless	Rotor Pre-Alignment
FOC in SFO	Torque	Sensorless	Six Pulse Injection
FOC in SFO	Torque	Sensorless	High Frequency Injection
FOC in SFO	Torque	Sensorless	Dyno Mode
FOC in RFO or SFO	Speed	Sensorless	Rotor Pre-Alignment
FOC in RFO or SFO	Speed	Sensorless	Six Pulse Injection
FOC in RFO or SFO	Speed	Sensorless	High Frequency Injection
FOC in RFO or SFO	Speed	Sensorless	Open-Loop Volt/Hz
FOC in RFO	Speed	Encoder	Rotor Pre-Alignment
FOC in RFO	Speed	Hall Sensor	N.A.
TBC in BC	Speed	Hall Sensor	N.A.
TBC in TC	Speed	Hall Sensor	N.A.

21 different combinations of Control methods available in SW

Acronym	Expansion
FOC	Field Oriented Control
RFO	Rotor Frame Orientation
SFO	Stator Frame Orientation
TBC	Trapezoidal or Block Commutation
BC	Block Commutation
TC	Trapezoidal Commutation

Rotor field oriented control - RFO

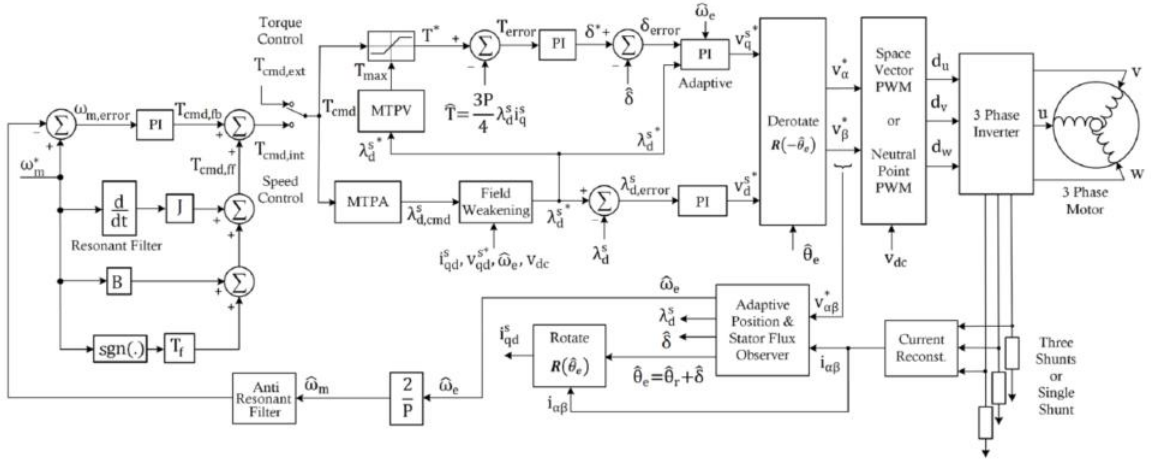
- RFO is a variant of sensorless FOC control using the rotor reference frame
- Torque and Flux are indirectly controlled via i_q and i_d currents
- 3-phase sinusoidal currents are decomposed into q-axis and d-axis DC currents
- RFO control methods include modules such as speed control loop, q and d axis current loops and position feedback
- Position of the rotor is also needed here to control the currents in an optimised way for both efficiency and dynamic performance
- Command is current



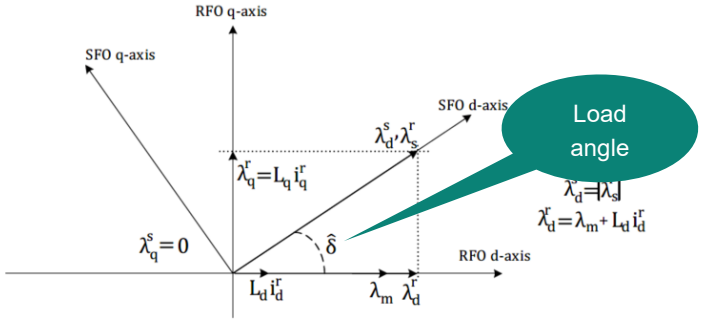
RFO control block diagram

Stator-field-oriented control (SFO)

- SFO is a variant of sensorless FOC control using the stator reference frame
- One big advantage is that Torque and Flux both can be directly controlled
- Since rotor flux changes significantly with temperature, the SFO method to control it is useful
- Applications like an e-bike that use direct torque control can benefit from SFO control
- Improves dynamic performance in the field weakening region using SFO by directly controlling the flux
- Command can be speed or torque



SFO control block diagram

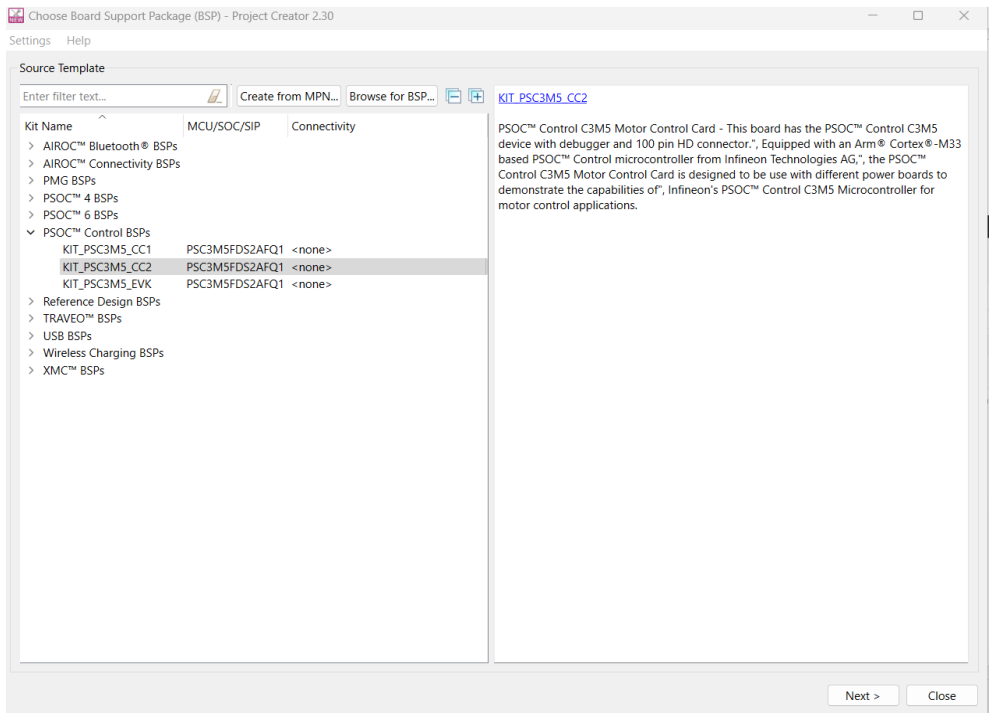


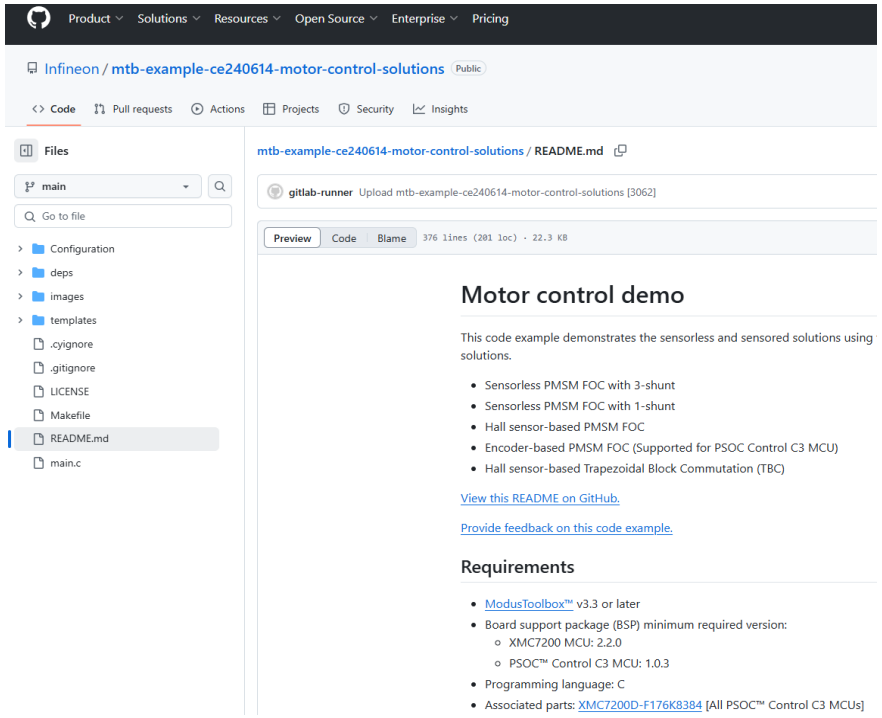
- Higher load currents means higher load angle , to get more torque higher load angle need to be commanded
- SFO d- axis is aligned with stator flux vector

RFO-SFO relationship

ModusToolbox™ software – Code example available on GitHub

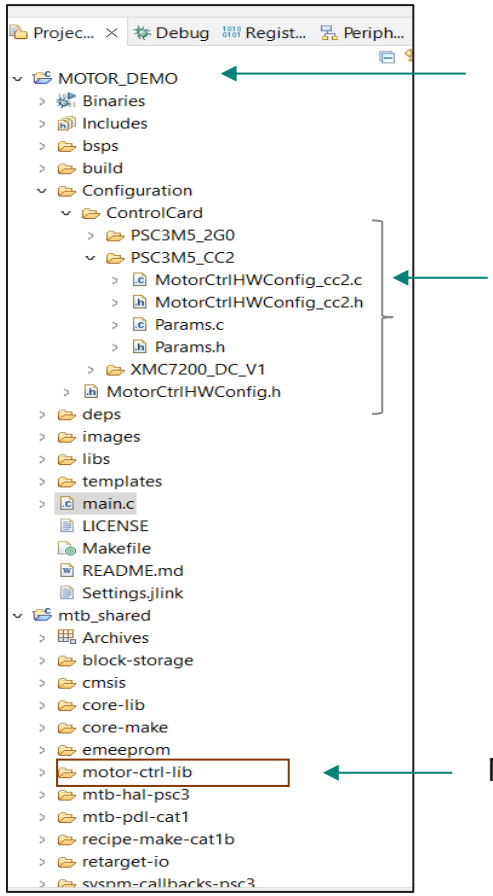
- For more information, see the code example: Motor control demo
- Go to Project creator -> KIT_PSC3M5_CC2 -> Motor demo





MTB file structure

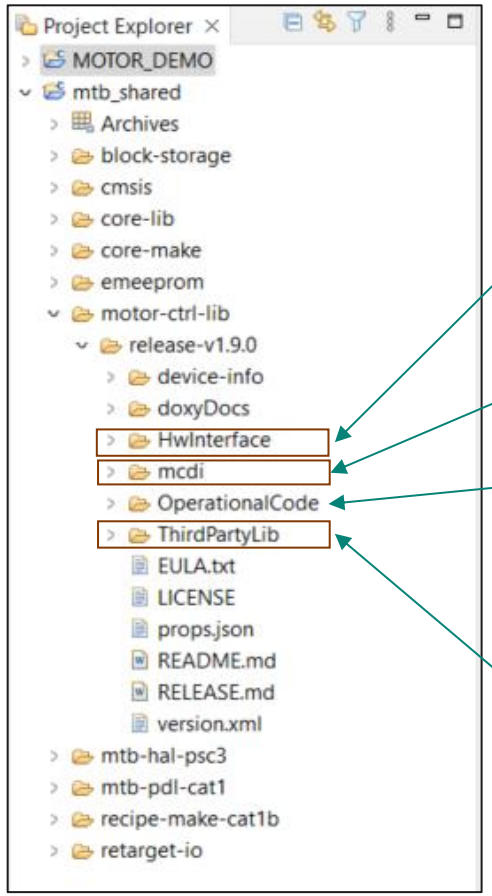
– Motor control functions are implemented as ModusToolbox™ middleware (library)

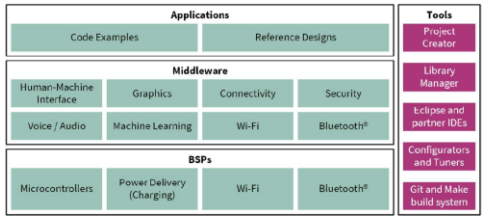


Motor control code example / application code

– Control cards configuration
– Power board and ADC configuration
– Parameters config : motor, observer, PI, mech loads, and so on

Motor control library





Motor control peripheral related functions
– PWM, ADC, DMA, EEPROM function
– Fast and Slow loop ISR definition

Motor control driver interface (MCDI) function

All the motor control related functions
– State machine
– Motor control functions
– Parameter handler
– Static library files for GCC and IAR

Third party library functions
– Motor control suite oscilloscope interface

Scalable architecture approach

- State machine-based
- Two main interrupts
  - Fast -> Current controller
  - Slow-> Speed controller
- HAL abstraction library based
- Properly documented
  - Intuitive file/variable naming
  - Extensive manual
- Isolated command interface
  - Potentiometer
  - External (GUI)
- Changing control modes means changing one line of code (redefining one parameter)
- Contains multiple adaptive filters/observers/controllers, no need to tune it

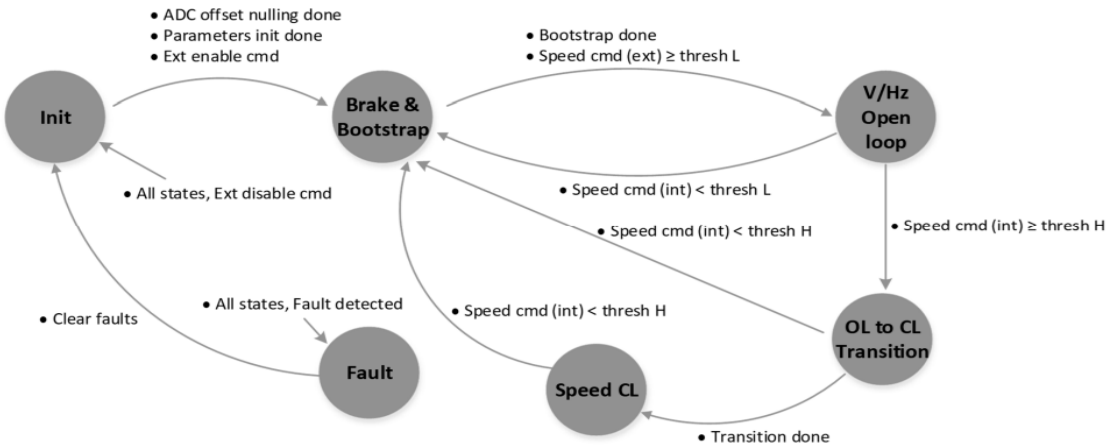


Fig. 108: RFO & SFO speed control FOC sensorless with V/Hz startup state machine diagram.

State machine (SM) handling

- Each state has an entry, exit, RunISR0 and RunISR1 functions
  - The state entry function is called when SM enters this state
  - The state exit function is called when the SM exits from this state
  - State ISR0 function is called every fast loop
  - State ISR1 function is called every slow loop
- State transition is handled in a slow loop ISR
  - In the State transition function call sequence mentioned below:
    - Call the state ISR1 function
    - If state change requested
      - Exit function of the Current state
      - Entry function of Request state
      - Set the current state as the Requested state
- sm.current variable holds the current state machine state

State	Build			Description
	RFO	SFO	TBC	
Init	0	0	0	Initialization
Brake_Boot	1	1	1	Brake and Bootstrap
Align	2	2		Aligning (pre-positioning)
Six_Pulse	3	3		Six pulse injection
High_Freq	4	4		High-frequency-injection locking
Speed_OL_T0_CL	5	5		Transition from open-loop to closed-loop
Dyno_Lock	6	6		Waiting for obsever lock to start up in dyno mode
Prof_Finished	7	7		Profiler, finished
Prof_Rot_Lock	8	8		Profiler, rotor locking
Prof_R	9	9		Profiler, stator resistance estimation
Prof_Ld	10	10		Profiler, d-axis inductance estimation
Prof_Lq	11	11		Profiler, q-axis inductance estimation
Volt_HZ_OL	12	12	3	Open-loop Volt/Hz control
Speed_CL	13	13	4	Closed-loop speed control
Fault	14	14	5	Fault
Torque_CL		15		Closed-loop torque control
Current_CL	15		6	Closed-loop Current control

State machine (SM) handling- Functions associated

- Each state has four or more functions associated with it
- All States have Entry() functions whole only some of them have Exit() function
- All States except Fault have a fast-running function that runs in ISR0
- A few states have a slow-running function that runs in ISR1
- In each state where the function is not available EmptyFcn() is used
- sm.current variable holds the current state machine state

State	Entry() Function	Exit() Function	RunISR0() Function	RunISR1() Function
Init	InitEntry	EmptyFcn	InitISR0	EmptyFcn
Brake_Boot	BrakeBootEntry	EmptyFcn	BrakeBootISR0	BrakeBootISR1
Align	AlignEntry	AlignExit	AlignISR0	EmptyFcn
Six_Pulse	SixPulseEntry	SixPulseExit	SixPulseISR0	SixPulseISR1
High_Freq	HighFreqEntry	HighFreqExit	HighFreqISR0	EmptyFcn
Speed_OL_To_CL	SpeedOLToCLEntry	SpeedOLToCLExit	SpeedOLToCLISR0	EmptyFcn
Dyno_Lock	DynoLockEntry	DynoLockExit	DynoLockISR0	EmptyFcn
Prof_Finished	MotorProfEntry	MotorProfExit	MotorProfISR0	EmptyFcn
Prof_R	MotorProfEntry	MotorProfExit	MotorProfISR0	EmptyFcn
Prof_Ld	MotorProfEntry	MotorProfExit	MotorProfISR0	EmptyFcn
Prof_Lq	MotorProfEntry	MotorProfExit	MotorProfISR0	EmptyFcn
Prof_Rot_Lock	MotorProfEntry	MotorProfExit	MotorProfISR0	EmptyFcn
Volt_Hz_OL	VoltHzOLEntry	EmptyFcn	VoltHzOLISR0	EmptyFcn
Speed_CL	SpeedCLEntry	EmptyFcn	SpeedCLISR0	EmptyFcn
Fault	FaultEntry	FaultExit	EmptyFcn	FaultISR1
Torque_CL	TorqueCLEntry	EmptyFcn	TorqueCLISR0	EmptyFcn
Current_CL	CurrentCLEntry	EmptyFcn	CurrentCLISR0	EmptyFcn

Control mode

– Control mode is configured using params.ctrl.mode parameter in params.c file

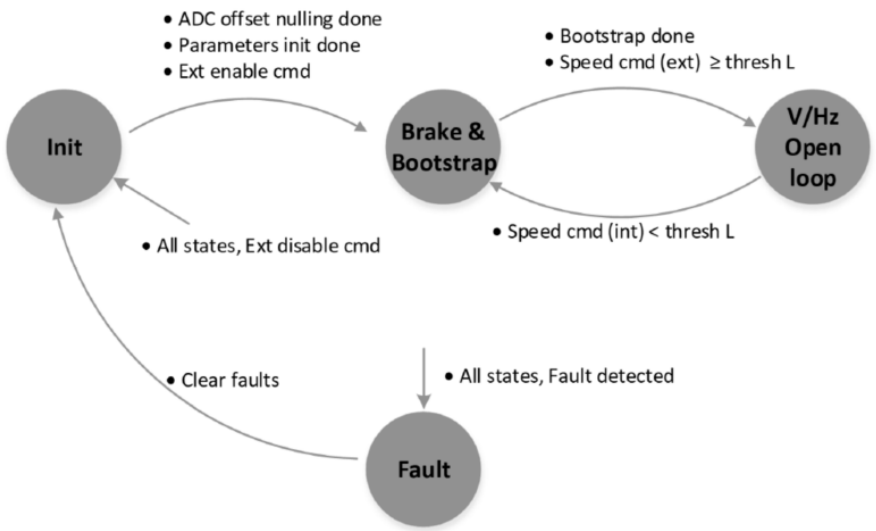
Control Mode	Build			Description
	RFO	SFO	TBC	
Volt_Mode_Open_Loop	0	0	0	Open-loop V/Hz control
Curr_Mode_FOC_Sensorless_Align_Startup	1			Closed-loop sensorless-foc current control with pre-alignment at startup
Curr_Mode_FOC_Sensorless_SixPulse_Startup	2			Closed-loop sensorless-foc current control with six pulse injection at startup
Curr_Mode_FOC_Sensorless_HighFreq_Startup	3			Closed-loop sensorless-foc current control with high frequency injection at startup
Curr_Mode_FOC_Sensorless_Dyno	4			Closed-loop sensorless-foc current control in dyno mode (waiting for observer lock to start up)
Curr_Mode_FOC_Encoder_Align_Startup	5			Closed-loop sensored-foc current control with encoder feedback and pre-alignment at startup
Curr_Mode_FOC_Hall	6			Closed-loop sensored-foc current control with hall sensor feedback
Curr_Mode_Block_Comm_Hall			1	Closed-loop block-commutation current control with hall sensor feedback
Trq_Mode_FOC_Sensorless_Align_Startup		1		Closed-loop sensorless-foc torque control with pre-alignment at startup
Trq_Mode_FOC_Sensorless_SixPulse_Startup		2		Closed-loop sensorless-foc torque control with six pulse injection at startup
Trq_Mode_FOC_Sensorless_HighFreq_Startup		3		Closed-loop sensorless-foc torque control with high frequency injection at startup
Trq_Mode_FOC_Sensorless_Dyno		4		Closed-loop sensorless-foc torque control in dyno mode (waiting for observer lock to start up)
Speed_Mode_FOC_Sensorless_Align_Startup	7	5		Closed-loop sensorless-foc speed control with pre-alignment at startup
Speed_Mode_FOC_Sensorless_SixPulse_Startup	8	6		Closed-loop sensorless-foc speed control with six pulse injection at startup
Speed_Mode_FOC_Sensorless_HighFreq_Startup	9	7		Closed-loop sensorless-foc speed control high frequency injection at startup
Speed_Mode_FOC_Sensorless_Volt_Startup	10 ★	8 ★		Closed-loop sensorless-foc speed control with open-loop V/Hz at startup
Speed_Mode_FOC_Encoder_Align_Startup	11			Closed-loop sensored-foc speed control with encoder feedback and pre-alignment at startup
Speed_Mode_FOC_Hall	12			Closed-loop sensored-foc speed control with hall sensor feedback
Speed_Mode_Block_Comm_Hall			2 ★	Closed-loop block-commutation speed control with hall sensor feedback
Profiler_Mode	13	9		Profiler mode

Note : ★ default control mode

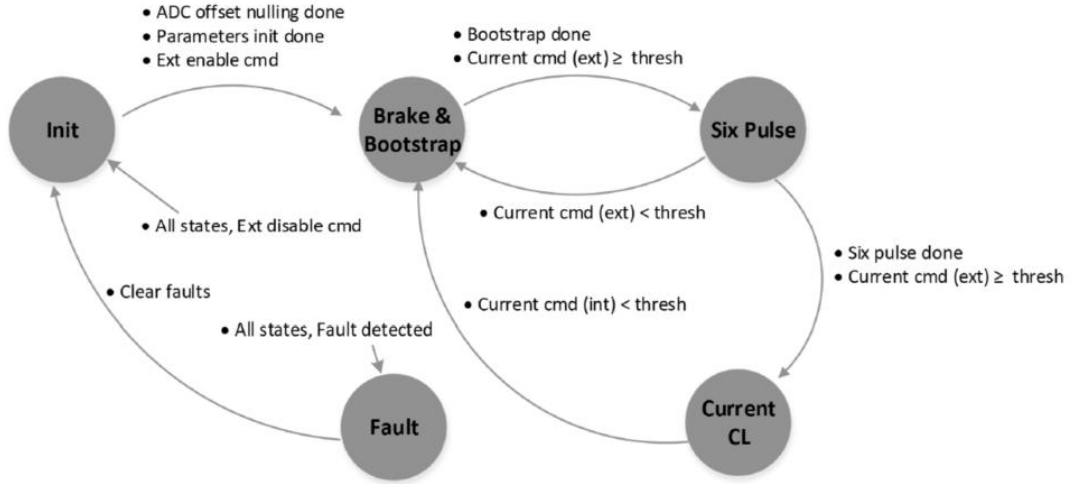
```
    // Control Mode:
#if defined(CTRL_METHOD_RFO) || defined(CTRL_METHOD_SFO)
    params.ctrl.mode = Speed_Mode_FOC_Sensorless_Volt_Startup; // [#]
#elif defined(CTRL_METHOD_TBC)
    params.ctrl.mode = Speed_Mode_Block_Comm_Hall; // [#]
#endif
```

Control mode types and associated state machines – Example 1

- The control mode is configured using the params.ctrl.mode parameter in params.c file
- sm.current variable holds the current state machine state



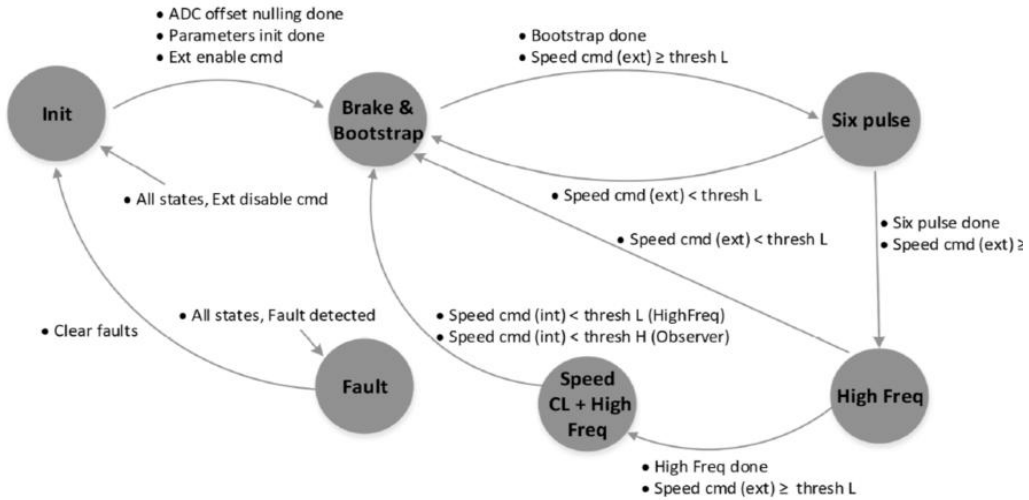
V/Hz open loop state machine diagram



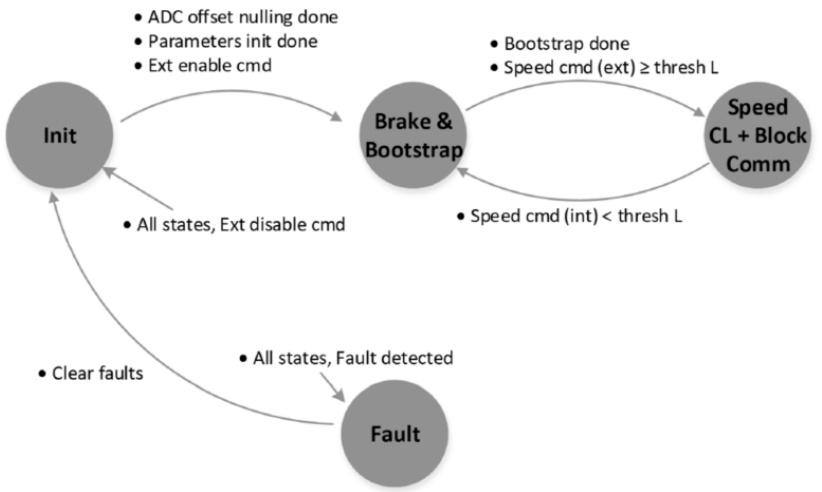
RFO current mode FOC sensor less with six pulse startup state machine diagram

Control mode types and associated state machines – Example 2

- Control mode is configured using params.ctrl.mode parameter in params.c file
- sm.current variable holds the current state machine state



RFO/SFO speed control FOC sensor less with high frequency startup state machine diagram



TBC Trapezoidal commutation speed control state machine diagram

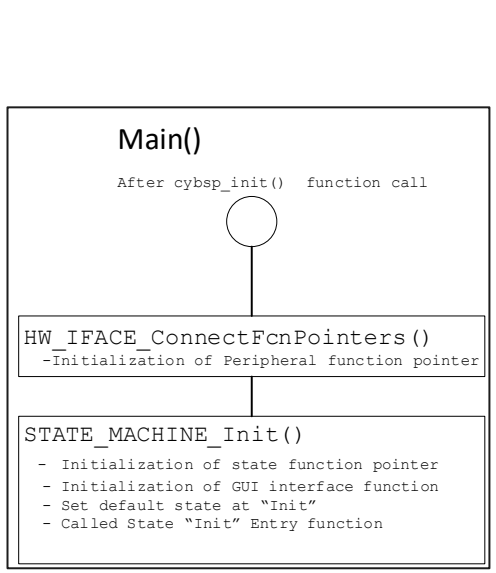
ISR handling

Init sequence

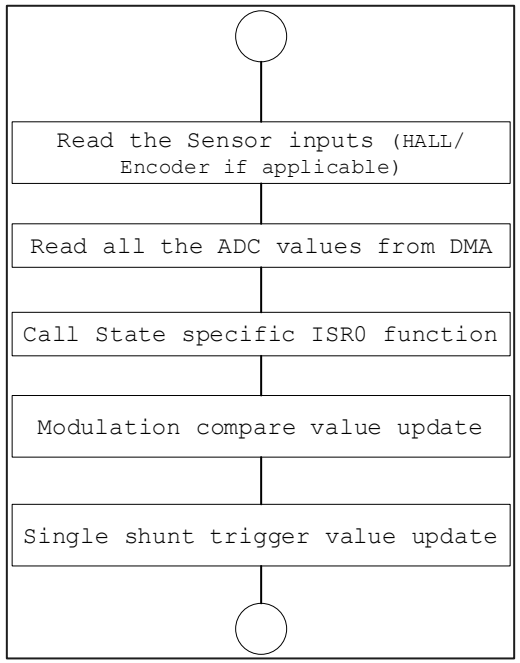
- Init state entry function (InitEntry())
  - If parameters are not initialised
    - Parameter initialisation
    - Motor control peripheral initialisation
    - Reset all the modules
    - Start the peripherals
  - Reset all the variables

Fast loop ISR (MCU_RunISR0())

- Fast control loop is high priority than the slow control loop
- ISR0 MCU_RunISR0() and ISR1 MCU_RunISR1() functions are defined in ..\mtb_shared\motor-ctrl-lib\release-v1.9.0\HwInterface\MCU.c



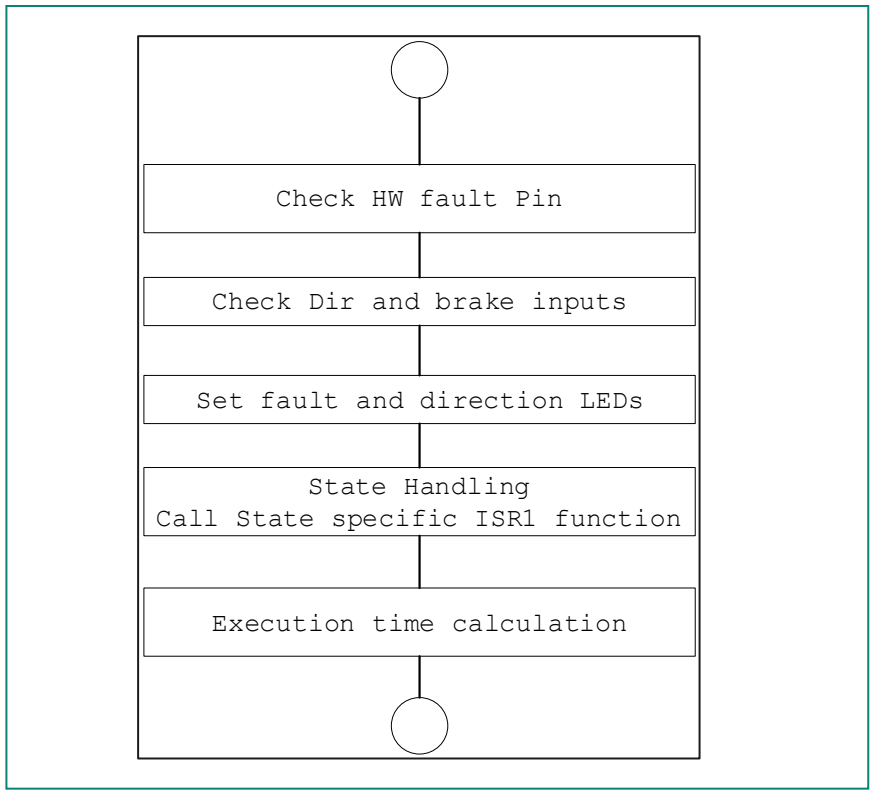
Init sequence



Fast loop ISR

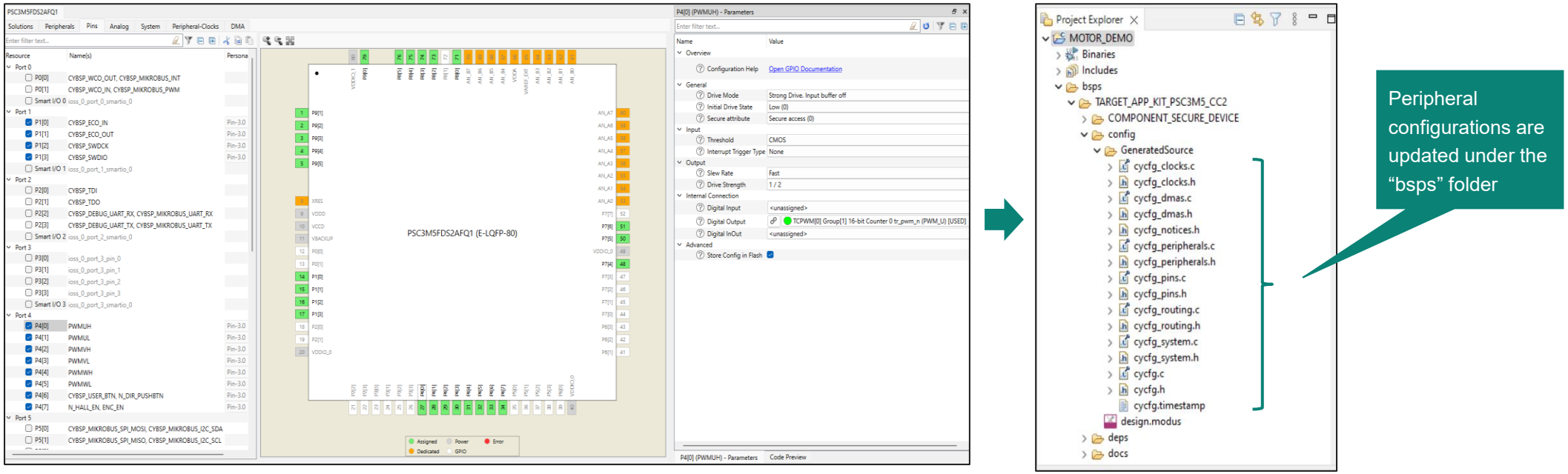
ISR handling (contd.)

Slow Loop ISR (`MCU_RunISR1()`)



Peripheral configuration

– The controller peripherals (PWM, ADC, DMA, MOTIF, and GPIO) are configured using the ModusToolbox™ device configurator

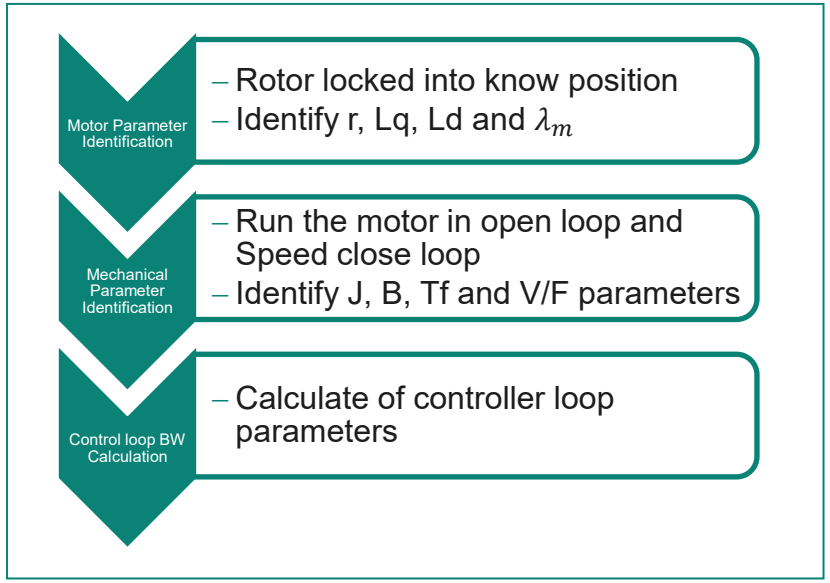


Parameter handling

- The parameter initialisation is done in the Init state entry function if parameters are not initialised
  - If no valid data EEPROM or PARAMS_ALWAYS_OVERWRITE macro set to TRUE, the following functions are called
    - PARAMS_InitManual() → All the input parameter variables assignment
    - PARAMS_InitAutoCalc() → Calculate all the derived parameters from input parameters
    - All parameter values are stored in EEPROM
  - If EEPROM has valid parameters and “PARAMS_ALWAYS_OVERWRITE” macro set to false (default value), parameter variables are updated from EEPROM
- Parameter values can be updated from the external interface (for example, GUI) using FcnExeHandler.c functions
  - Derived parameter calculations can be triggered by setting fcn_exe_handler.req.Auto_Calc_Params bitfield
  - Store all parameter values from RAM to EEPROM by setting fcn_exe_handler.req.Flash_Params bitfield
  - Reset all the modules and reinitialize peripheral by setting fcn_exe_handler.req.Reset_Modules bitfield
  - During execution of the above request, the FcnExe handler stops all the peripherals and restarts after handling the request
  - The FcnExe handler function only executes the request when SM is in “init” state, request are ignored if SM is not in “init” state

Motor and mechanical parameter identification

- Motor control library “Profiler” state supports to identify motor and load parameters that are used in the current controller, rotor angle estimation, speed controller, and so on
- In “Profiler” state, the following parameters are identified
  - Motor parameters (used in current controller, rotor angle estimation)
    - Stator resistance, r (“params.motor.r“)
    - Stator q-axis inductance, L_q (params.motor.lq“)
    - Stator d-axis inductance, L_d (params.motor.ld)
    - Rotor permanent-magnet flux linkage, λ_m (params.motor.lam)
  - Mechanical parameters (used in speed controller)
    - Inertia, J (params.mech.inertia)
    - Viscous, B (params.mech.viscous)
    - Friction, Tf (“params.mech.viscous”)
  - V/F offset and V/F constant (params.ctrl.volt.v_min and params.ctrl.volt.v_to_f_ratio)

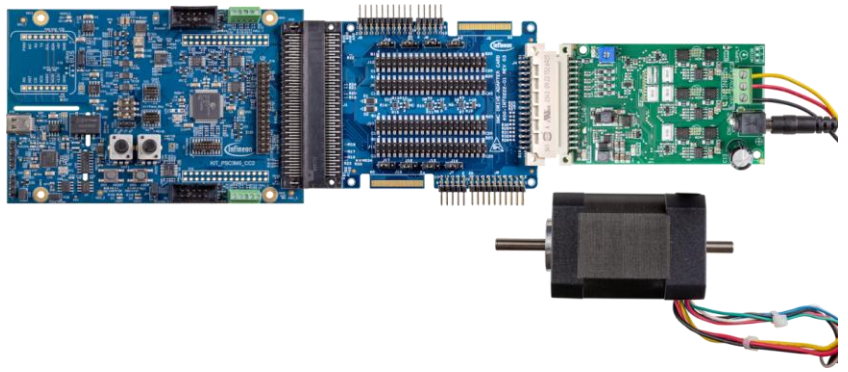


- After parameter extraction, control loop bandwidth is calculated, and the user can select dynamic response (“slow”, “Moderate”, “High”) for control loop BW
- After parameter extraction, all the parameters are updated automatically to corresponding variables, if the params.profiler.overwrite variable is set to TRUE
- It is required to configure a reasonable V/F parameter and BW to run the motor in open loop and close loop for the motor used

Note: Parameter identification only works in a leg shunt configuration..

Execution time

- **Hardware setup**
  - KIT_PSC3M5_CC2_V2
  - Drive_Adapter_Card
  - EVAL_24V_250W (KITMOTORDC250W24VTOBO1)
  - Nanotec DB42M03 24 V BLDC motor (Motor 0)



Build	Control Type	Controlled entiry	Feedback Type	Statup Method	Shunt Type	Compiler	CPU Load (@10kHz)
RFO	FOC	Speed	Sensorless	Open-Loop Volt/Hz	3x	GCC	13.78%
RFO	FOC	Speed	Sensorless	Open-Loop Volt/Hz	1x	GCC	16.33%
RFO	FOC	Speed	Hall Sensor	N.A.	3x	GCC	11.20%
RFO	FOC	Speed	Hall Sensor	N.A.	1x	GCC	11.18%
RFO	FOC	Speed	Encoder	N.A.	3x	GCC	11.69%
SFO	FOC	Speed	Sensorless	Open-Loop Volt/Hz	3x	GCC	15.87%
SFO	FOC	Speed	Sensorless	Open-Loop Volt/Hz	1x	GCC	18.45%
TBC	BC	Speed	HALL	N.A.	3x	GCC	8.87%

Note : Sample capture while motor is running at full speed. Average of 16 samples.

Note: Default optimization is used -0fast

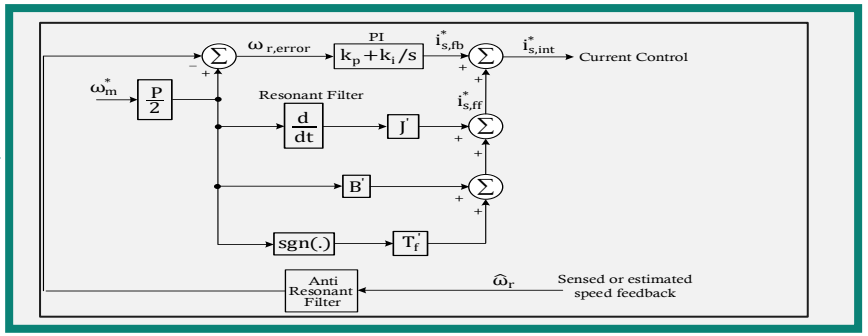
Code execution

- Two interrupt service routines (ISR) are used to execute the motor control-related code
- Fast control loop (ISR0)
  - DMA ISR is used to generate this interrupt; This happens after completion of all the configured analog conversion
  - Measurements, Speed loop, Current loop, Angle estimator, and Modulator code are executed
    - ISR0 is called every PWM period or multiple of PWM period
    - Fast control loop rate is defined by Params.sys.samp.fs0 parameter
    - Ratio between PWM period and fast control loop is defined by Params.sys.samp.fpwm_fs0_ratio parameter
- Slow control loop (ISR1)
  - SYNC1_ISR timer period match ISR is used to trigger this interrupt
    - State machine handling, parameter handling codes are executed
  - ISR1 is called every multiple of the fast control loop; by default, ISR1 is called once in every 15 of fast control loop ISR
    - Ratio between fast control loop and Slow Control loop is defined by Params. sys. samp.fs0_fs1_ratio parameter
- Fast control loop is a higher priority than the slow control loop
- ISR0 MCU_RunISR0() and ISR1 MCU_RunISR1() functions are defined in \mtb_shared\motor-ctrl-lib\release-v1.9.0\HwInterface\MCU.c

Speed controller

- Speed controller parameters k_p, k_i, and feedforward terms are derived from params.ctrl.speed.bw, motor and load parameters, such as:
  - params.ctrl.speed.kp (kp)
  - params.ctrl.speed.ki (ki)
  - params.ctrl.speed.ff_k_inertia (J')
  - params.ctrl.speed.ff_k_viscous (B')
  - params.ctrl.speed.ff_k_friction (T_f')
- Source files: SpeedCtrl.c, SpeedCtrl.h;
- Call rate: Fast control loop
- Data structure: SPEED_CTRL_t speed (Member of ctrl structure; ctrl.speed)

Vars.w_cmd_int.elec
(ω_m)(Target value)
Vars.w_final_filt.elec
(ω_r)(Actual Speed value)

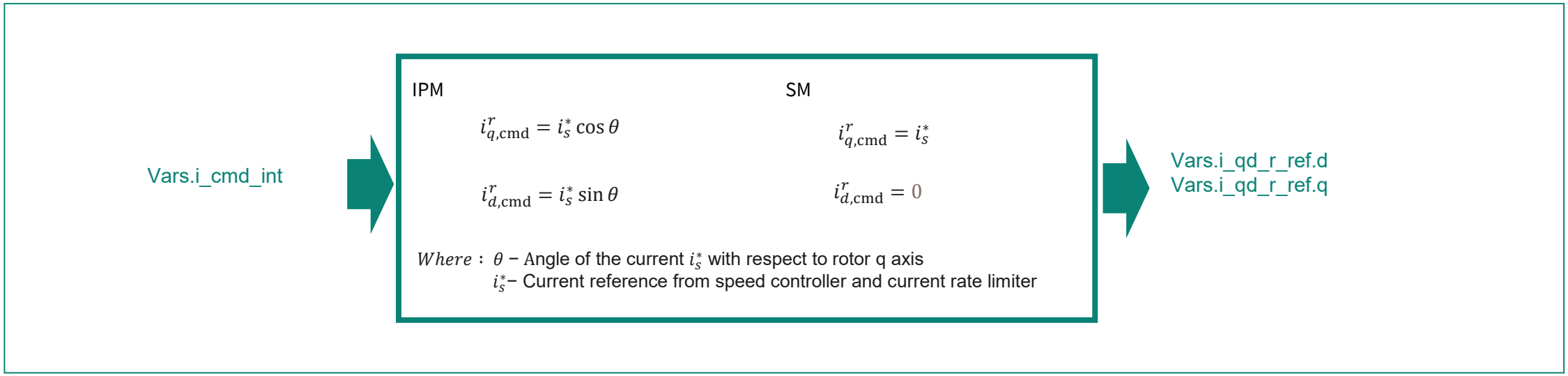


Ouput Limit = ± params. motor.i_peak

Vars.i_cmd_spd ($I_{s,\mathrm{int}}$)

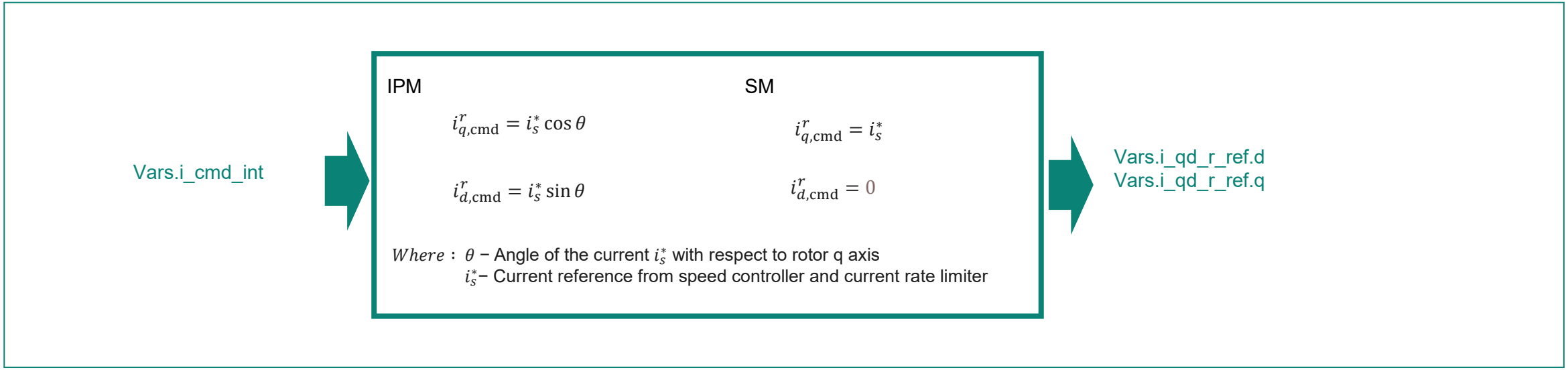
Maximum torque per ampere (MTPA)

- The MTPA module for generating command values for q and d axis currents (i_q^r and i_d^r) to maximise the torque for the overall current injected into the motor winding
- Thr torque equation of the PMSM motor $T = \frac{3P}{4}\left(\lambda_m i_q^r + \left(L_d - L_q\right) i_q^r i_d^r\right)$
- Interior permanent magnet (IPM) motor, the rotor saliency (Ld < Lq) can generate a reluctance torque component to augment the torque produced by the rotor magnet
- When driving a surface magnet motor (SM), there is zero saliency (Ld = Lq) , Id is set to zero for maximum efficiency



Maximum torque per ampere (MTPA) (contd.)

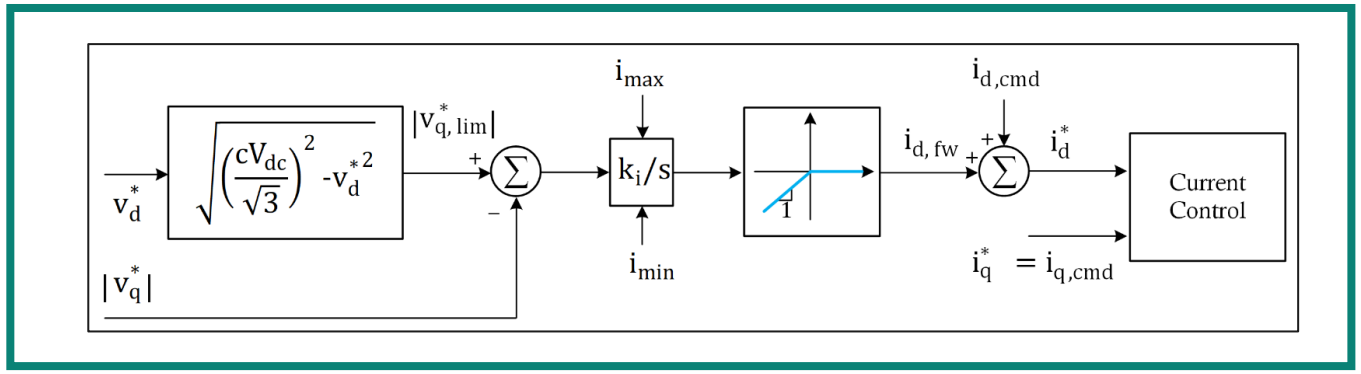
- Parameter
  - Motor d and q axis inductance values
  - Rotor Flux
- Source files: PhaseAdvance.c, PhaseAdvance.h
- Call rate: Fast control loop
- Data structure : PHASE_ADV_t ph_adv (Member of ctrl structure; ctrl.ph_adv)



Flux weakening or Maximum torque per volt (MTPV)

- The flux weakening (or field weakening) is to increase the speed of the motor beyond its base speed. Flux weakening current ($I_{d,\mathrm{fw}}$) is injected if flux weakening is enabled and based on the vdc coefficient
- At base speed, the inverter voltage becomes saturated, which means that there would not be enough voltage generated on the inverter to overcome the back EMF of the motor and increase the speed above the base speed.
- Therefore, the flux wakening method is needed to reduce the back EMF voltage of the motor and further increase the speed

Vars.i_qd_r_ref.d
Vars.i_qd_r_ref.q
Vars.v_qd_r_cmd.d
Vars.v_qd_r_cmd.q

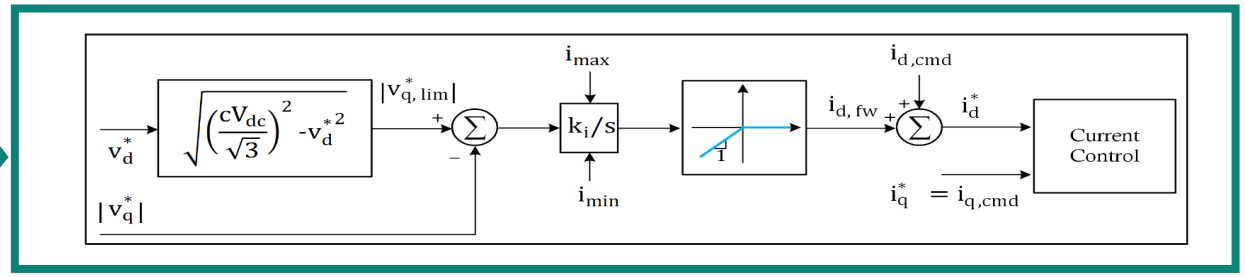


Vars.i_qd_r_cmd.q
Vars.i_qd_r_cmd.d

Flux weakening or Maximum torque per volt (MTPV) (contd.)

- Parameters
  - Params.ctrl.flux_weaken.en
  - Params.ctrl.flux_weaken.vdc_coeff(c)
  - Params.ctrl.flux_weaken.bw [Ra/sec]
  - Params.ctrl.flux_weaken.ki
- Source files: FluxWeaken.c, FluxWeaken.h
- Call rate: Fast control loop
- Data structure: FLUX_WEAKEN_t flux_weaken (Member of ctrl structure; ctrl. flux_weaken)

Vars.i_qd_r_ref.d
Vars.i_qd_r_ref.q
Vars.v_qd_r_cmd.d
Vars.v_qd_r_cmd.q

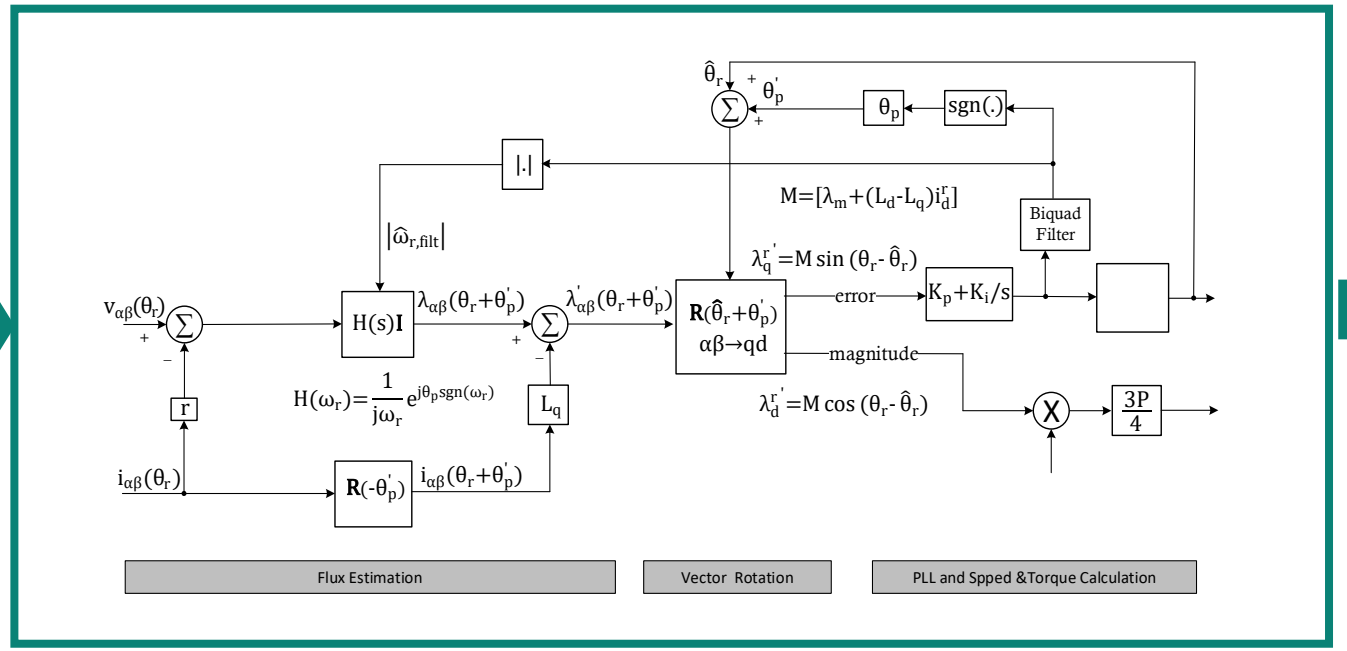


Vars.i_qd_r_cmd.q
Vars.i_qd_r_cmd.d

Angle estimation: Position observer

- Sensorless position observer estimates the rotor angular position and speed without using an actual position sensor; it relies on motor phase current, applied voltage and motor parameters
- PLL bandwidth can be configured using params.obs.pll.w0 parameter

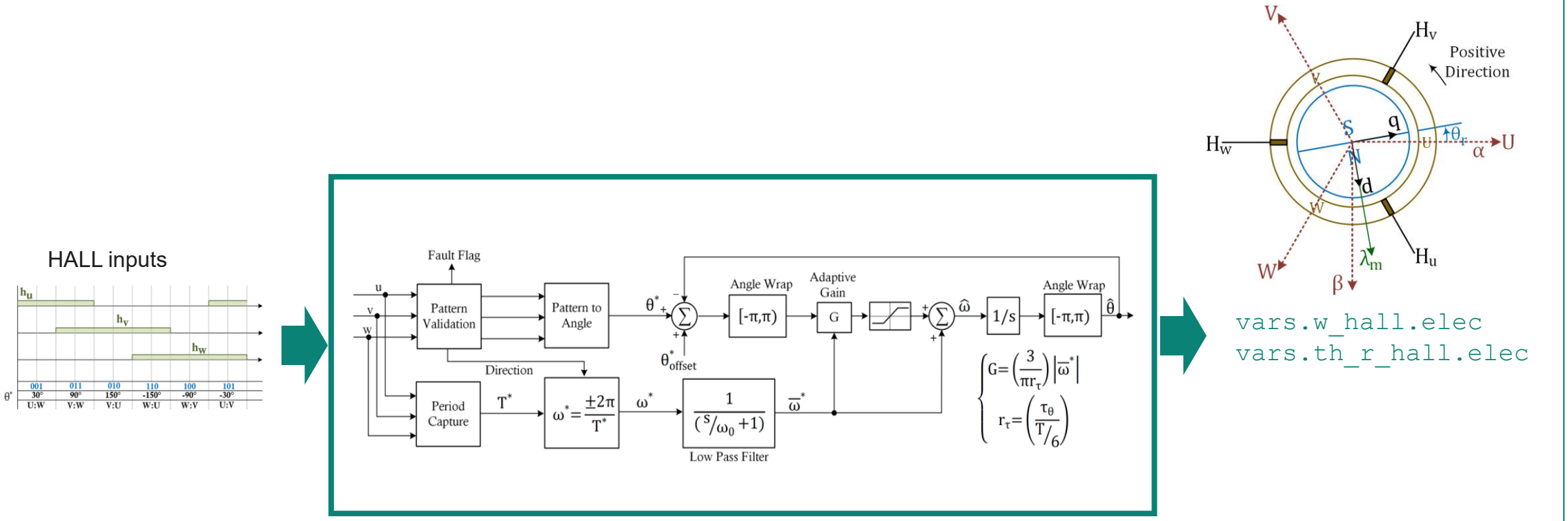
vars.v_ab_cmd.alpha
vars.v_ab_cmd.beta
vars.i_ab_fb.alpha
vars.i_ab_fb.beta



vars.w_est.elec
vars.th_r_est.elec
vars.T_est

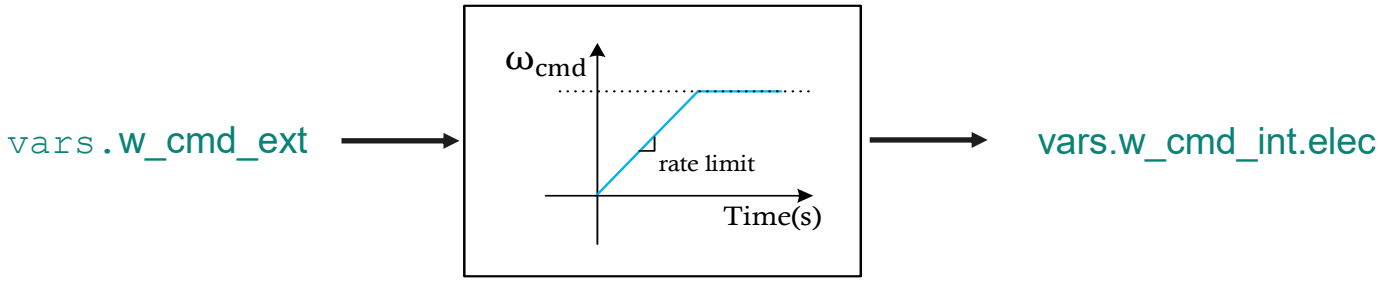
Angle estimation: Hall sensor

- In Hall sensor mode, three Digital hall (displace by 120° electoral angle) is used to estimation the rotor angle and speed
- The conventional placement of the hall sensors inside the BLDC is show in the figure.
  params.sys.fb.hall.th_r_offset.elec parameter provides offset to the calculated hall angle, this is used for motors that do not follow the conventional placement of the poles

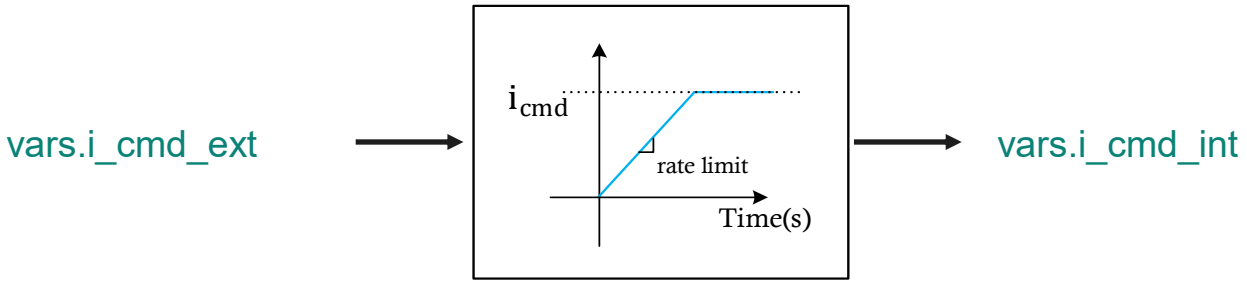


Rate limiter/ramp

– Rate limiter is used to achieve ccontrollable soft start for the system, used in speed controller and current controller command input



Speed rate limiter – params.sys.rate_lim.w_cmd.elec

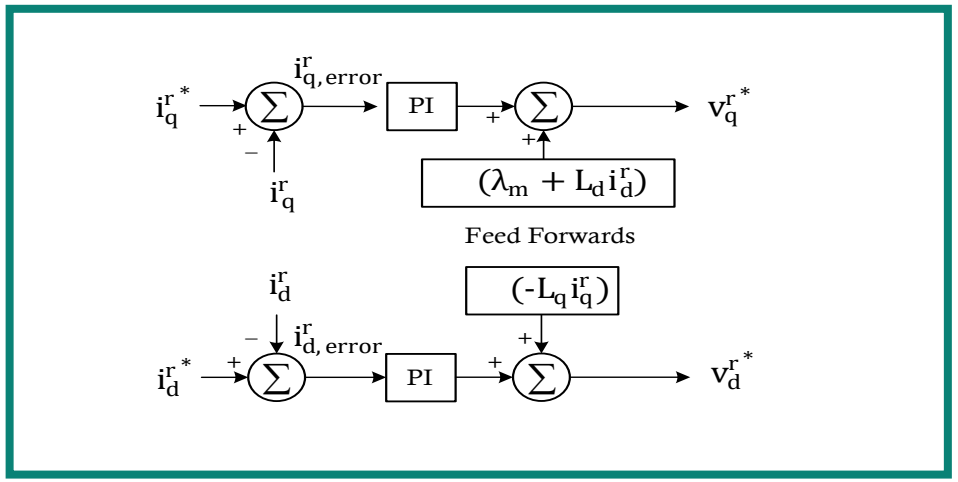


Current rate limiter – params.sys.rate_lim.i_cmd

Current control

- The d and q current commands are used as an input for the current controllers, and the output would be the voltage references
- The current controllers k_p, k_i, and feedforward terms are directly derived from motor electrical parameters
- Feedforward terms for d and q axis currents are decoupling

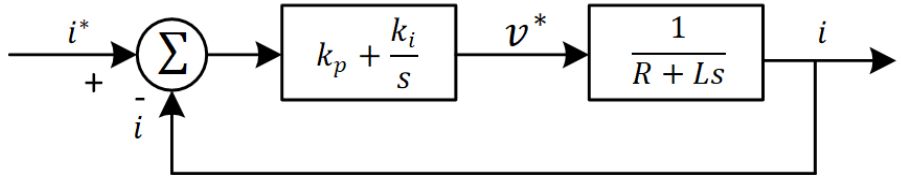
vars.i_qd_r_cmd.q
vars.i_qd_r_cmd.d
vars.i_qd_r_fb.q
vars.i_qd_r_fb.d



vars.v_qd_r_cmd.q
vars.v_qd_r_cmd.d

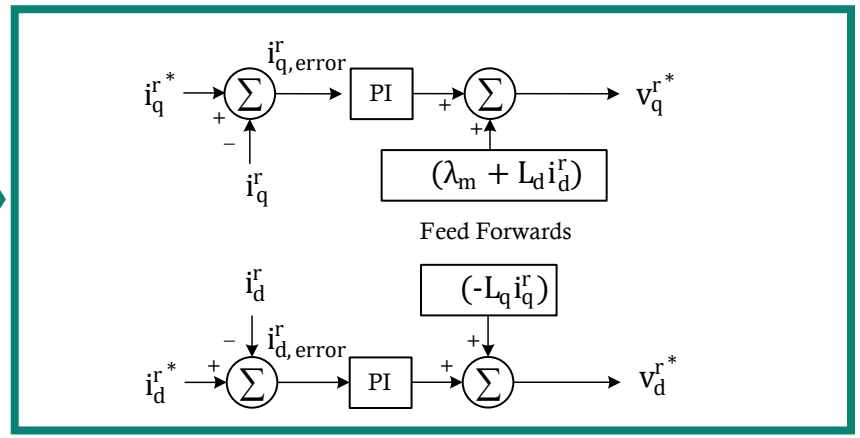
Current control

- Pole-zero cancellation technique is used to find the PI controller integral and proportional gain
- Source files: CurrentCtrl.c, CurrentCtrl.h
- Call rate: Fast control loop
- Parameters
  - params.ctrl.curr.bw
  - params.ctrl.curr.kp.q,params.ctrl.curr.ki.q
  - params.ctrl.curr.kp.d,params.ctrl.curr.ki.d



$$k_p = \omega_c L$$
$$k_i = \omega_c R$$

vars.i_qd_r_cmd.q
vars.i_qd_r_cmd.d
vars.i_qd_r_fb.q
vars.i_qd_r_fb.d



vars.v_qd_r_cmd.q
vars.v_qd_r_cmd.d

Motor phase current measurement

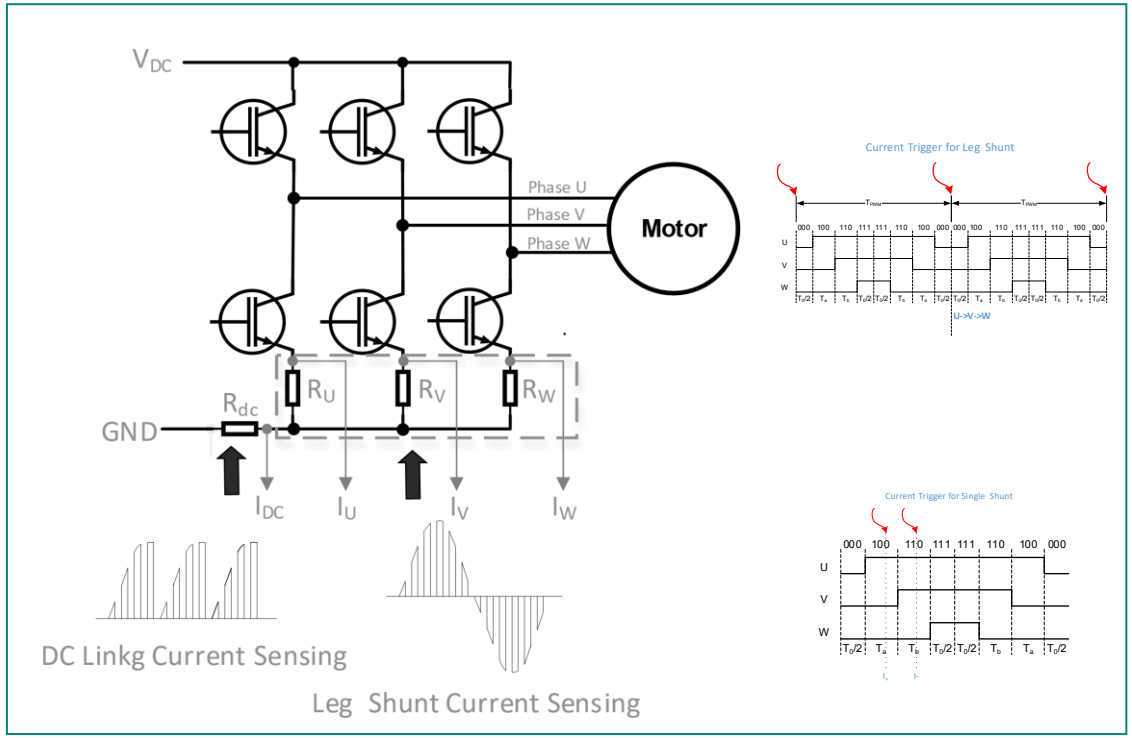
- The motor control SW supports 3-Leg shunt current measurement or DC link current measurement, shunt type selected by params.sys.analog.shunt.type parameter
- Current input offset is calibrated in “init” state, params.sys.analog.offset_null_time parameter is used to configure the offset calibration time

Leg-shunt current measurement

- All three phase currents are sampled at the same time and converted one by one (U→V→W)
- The motor phase current would flow through the shunt resistor only when the low-side switch is closed. Current measurements are done at all the low-side switches that are on

Single-shunt current measurement

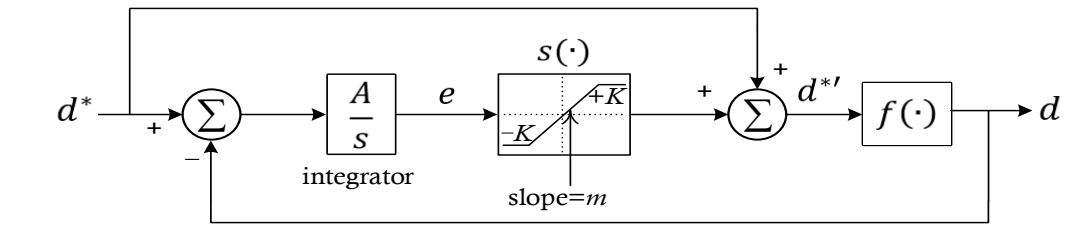
- Positive and negative currents are measured during the first half cycle of PWM
- A hybrid nonlinear modulation method is used to measure the current in the forbidden region (low modulation index and sector crossovers)



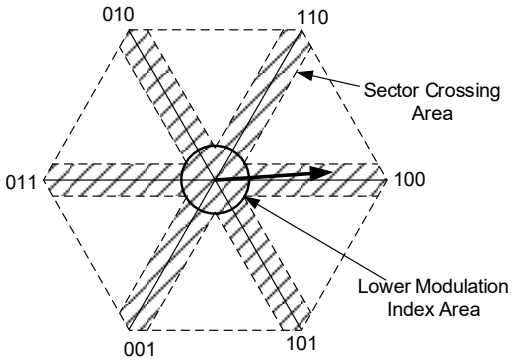
Note: HPPASS is used for measurement: 12-bit 12 MSPS SAR ADC with up to sixteen parallel sampling channels.

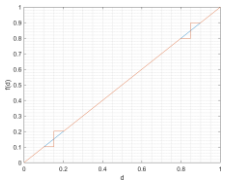
Single shunt current measurement

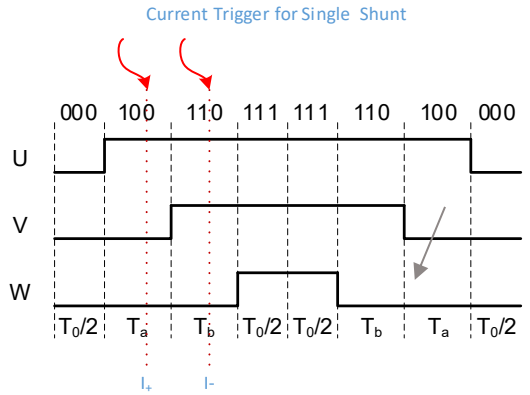
- Motor control SW extract the motor phase current from DC link current shunt only when the active (non-zero) vectors are being applied during each PWM cycle
- The third motor phase current value can be calculated assuming the sum of the three phase current values is zero
- At low modulation and sector crossing, one or both active vector time duration is less than current measurable range, i.e. forbidden region
- Hybrid nonlinear modulation scheme is used to measure current during forbidden region
- Minimum time for ADC sampling is configured by "`params.sys.analog.shunt.hyb_mod.adc_t_min`"
- Hybrid modulation integrator is configured by "`params.sys.analog.shunt.hyb_mod.ki`"



Hybrid Nonlinear duty-cycle Modulator

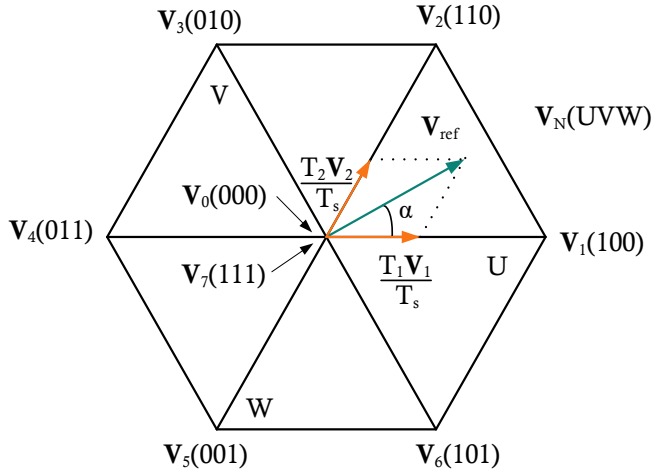


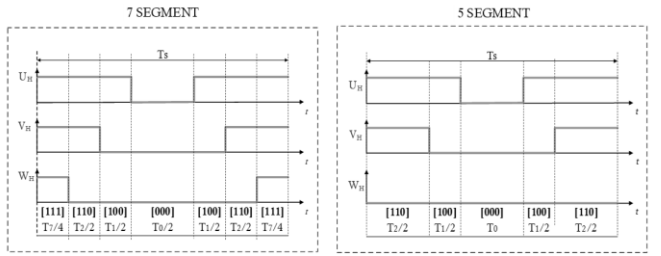




Motor control – Voltage modulation

- Space vector modulation (SVM) scheme is used for PWM generation and supports 3-phase modulation (7-Segment) and 2- phase modulation (5-Segment) with low-side clamping. The default 7-Segment scheme is used
- 5-Segment modulation can be enabled using params.ctrl.volt.five_seg.en. If enabled, 5-Segment PWM is applied above the configured threshold value of duty. Below this threshold, the 7-Segment scheme is applied
- The neutral point modulation (NPM) scheme is also used to generate a PWM signal, supports only 3-Phase modulation. This method takes less computation time compared to SVM, params.ctrl.volt.mod_method parameter for selection of modulation type





SVM switching methods

$$d_1 = \frac{\sqrt{3}}{2} m \sin\left(\frac{\pi}{3} - \alpha\right)$$

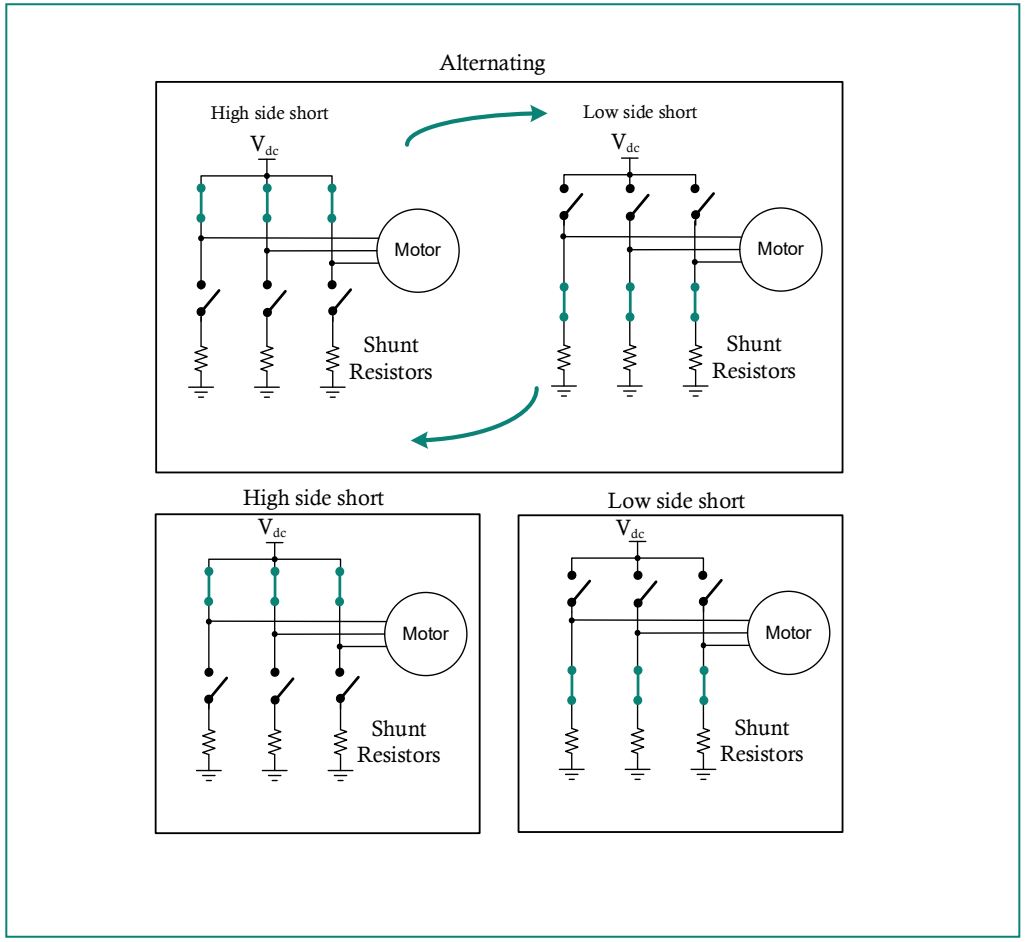
$$d_2 = \frac{\sqrt{3}}{2} m \sin\alpha \quad m = \frac{V_{\text{ref}}}{V_{\text{dc}}/2}$$

$$\boldsymbol{V}_{\text{ref}} = \frac{T_0\boldsymbol{V}_0 + T_1\boldsymbol{V}_1 + T_2\boldsymbol{V}_2 + T_7\boldsymbol{V}_7}{T_s} = d_1\boldsymbol{V}_1 + d_2\boldsymbol{V}_2$$

Note: The TCPWM module is used to generate the PWM signal and provide a trigger for ADC measurement. Peri Clock: 240 MHz, 16-bit timer is used for PWM generation.

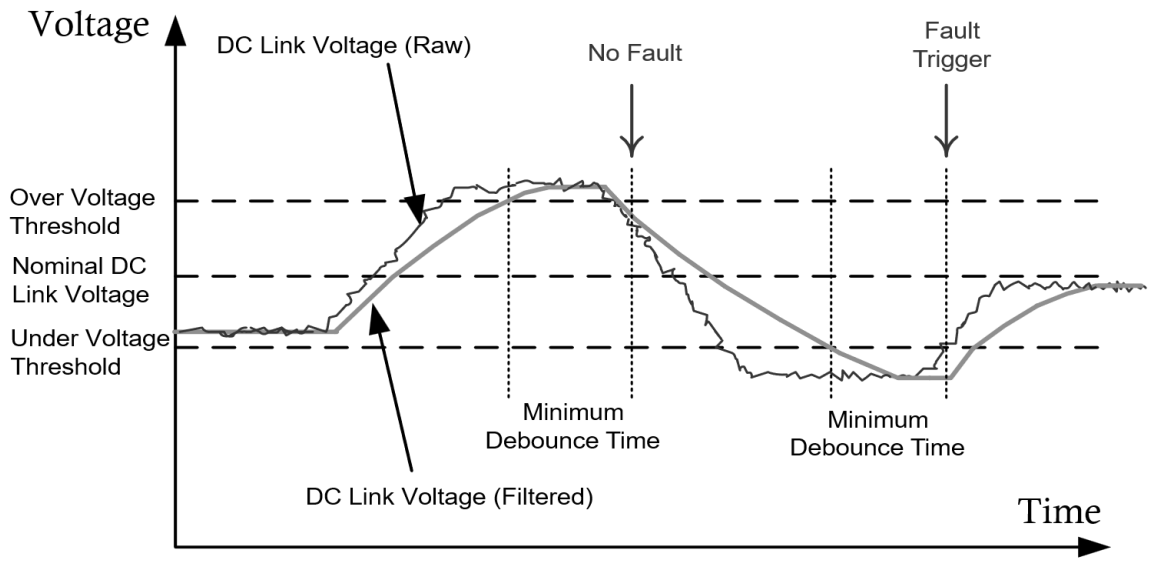
Fault/protection handling (1/4)

- The motor control software supports the following protections: Under/over voltage protection, overcurrent protection, overspeed and overtemperature Protection, and motor I2T protection
- When any fault condition is detected, the corresponding bitfield in the “faults.flags.all“ variable is set to ‘1’
- All the fault flags are latched into faults.flags_latched.all
- If any fault is reported in faults.flags_latched.all variables, the state machine moves to the fault state. Once the fault is cleared, the state machine moves to the Init state
- During a fault condition, all the switches are OFF or apply a zero vector based on faults.react_mask variable
- The software supports the following zero vector. One of the zero vectors is applied based on the configuration in params.sys.faults.short_method
  - Low_Side_Short, High_Side_Short, Alternate_Short



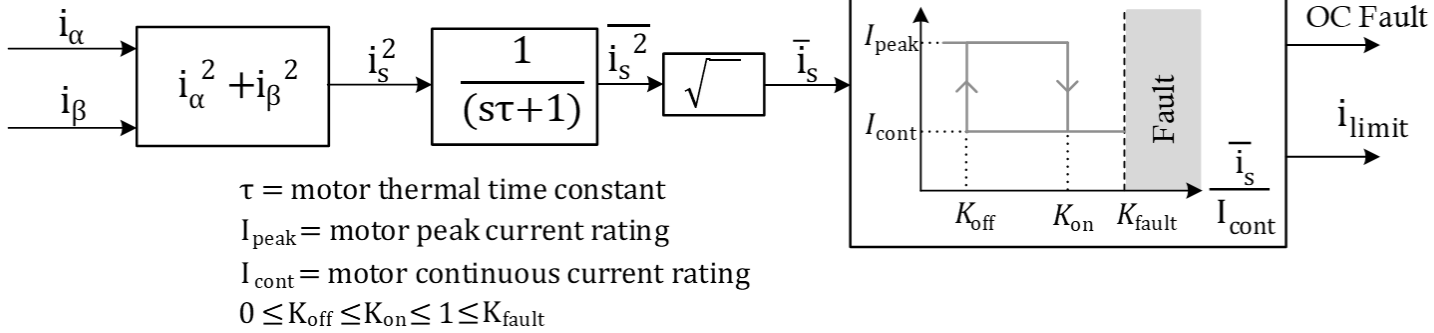
Fault/protection handling (2/4)

- **Over/under voltage protection**
  - Under/over voltage fault will be triggered if actual DC bus voltage is below the under voltage threshold (params.sys.faults.vdc_thresh.min) or above the overvoltage threshold (params.sys.faults.vdc_thresh.max) for a defined period (params.sys.faults.vdc_time)
  - When an undervoltage condition is detected, the following bitfield is faults.flags.sw.uv_vdc[3]
  - When an undervoltage condition is detected, the following bitfield is faults.flags.sw.ov_vdc[2]



Fault/protection handling (3/4)

- **Hardware overcurrent protection**
  - When an overcurrent condition is detected via HW (Trap), the following bitfield is set `faults.flags.hw.cs_ocp`
- **Software overcurrent protection**
  - The overcurrent fault will be triggered when motor current exceeds the configured threshold value "`faults.vars.oc_thresh`"
  - When an overcurrent condition is detected, the following bitfield is set `faults.flags.sw.oc[0]`
- **I2T protection**
  - When the motor current reaches the peak value, reduce the current limit in the current controller to the continuous current value



Fault/protection handling (3/4)

- **Overspeed protection**
  - The overspeed fault will be triggered when motor speed exceeds the configured threshold value params.sys.faults.w_thresh.elec
  - When overspeed condition is detected, the following bitfield is set to faults.flags.sw.os[4]

- Overtemperature protection
  - The overtemperature fault will be triggered when the temperature input values exceed the configured threshold value params.sys.faults.temp_ps_thresh
  - When overspeed condition is detected, the following bitfield is set to faults.flags.sw.ot_ps[1]

Dual motor support

- Motor control library supports driving two motors independently
  - Each motor parameter, control parameter, rotor angle estimation, PWM frequency, current measurement, and so on, can be configured separately
  - Each motor uses separate fast loop and slow loop control ISRs
- Both the motors only support the same build option: RFO, SFO, or TBC
- Both the motors can be controlled separately
- All the motor parameters and variables are grouped into Motor[0] and Motor[1] structure

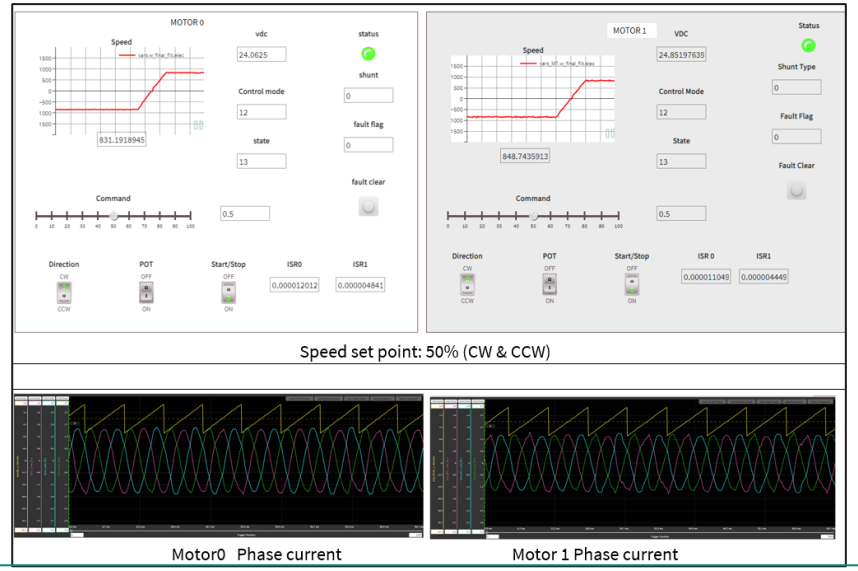
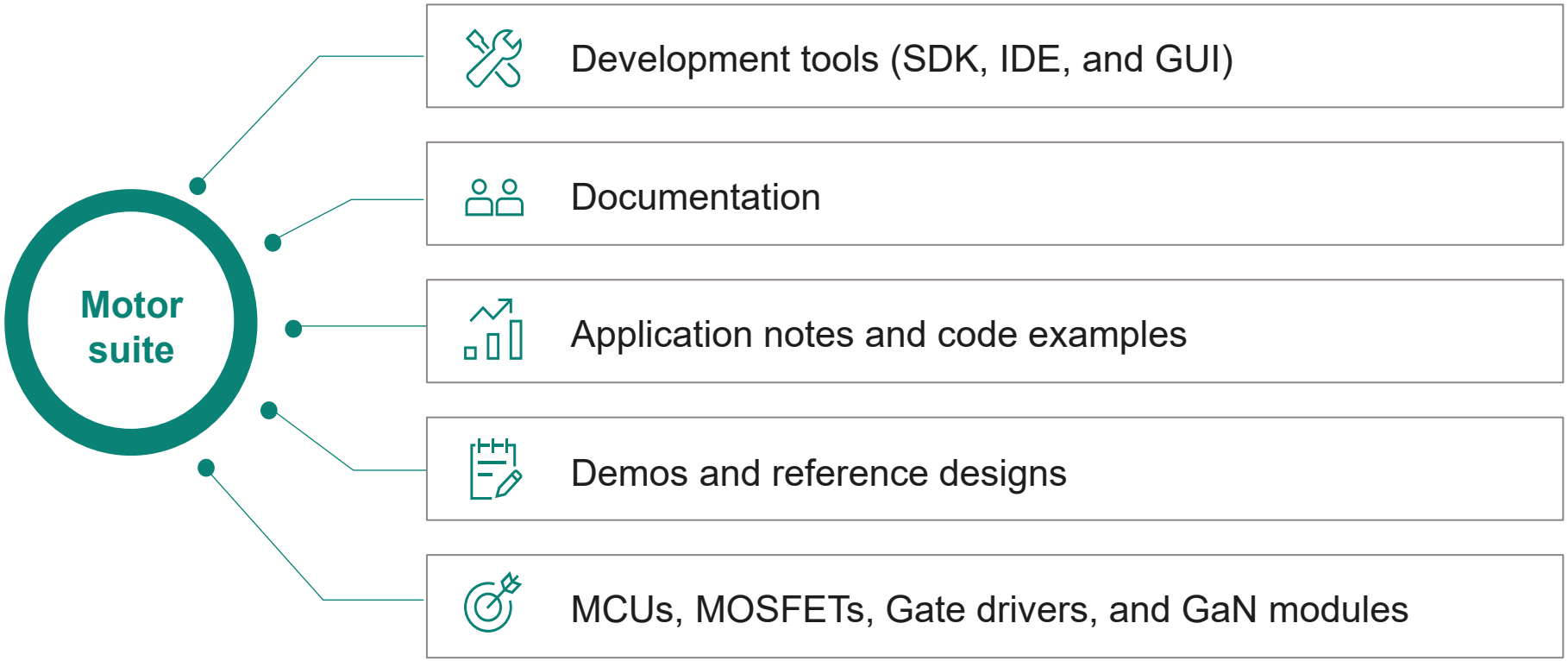


Table of contents

ModusToolbox™ motor suite

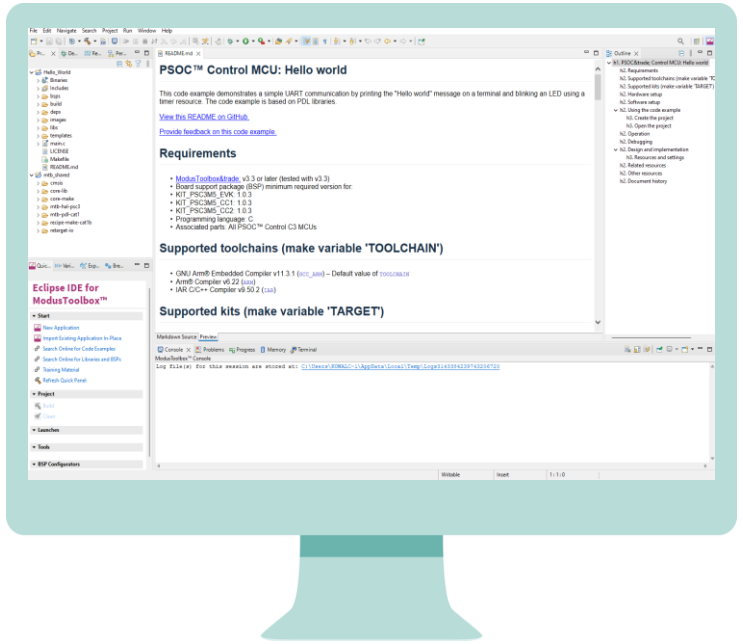
Overview

Infineon motor control solutions extend far beyond high-performance semiconductor hardware. Built on a foundation of functional safety and security, our comprehensive ecosystem of software, development tools, technical support services, and training is designed to simplify your design process and elevate the quality of your end products.

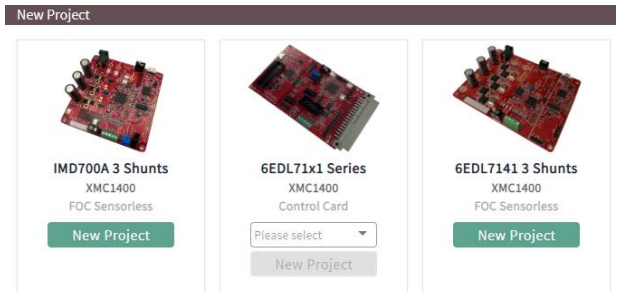


Motor suite: Control design & Validation

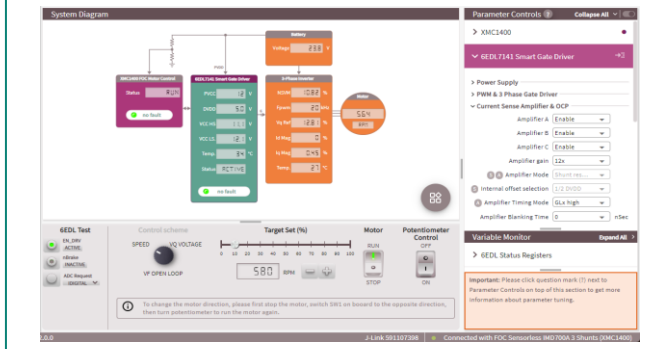
- State-of-the art, end-to-end development platform: Choose and deploy algorithms directly to your target hardware and spin your motor
- Preconfigured libraries: State-of-the-art motor control algorithms for easy integration
- Multiple flexible IDE support: Eclipse, IAR EW, MS Studio Code, and Keil uVision
- Safety and security: Industry standards for safety and security
- Ease-of-use: Get your motor spinning in less than 3 minutes



Motor suite: Functional highlights



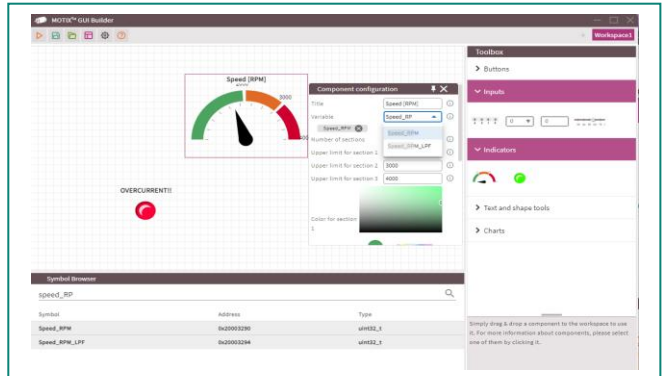
One Graphical Interface to operate **multiple** evaluation boards and control schemes



Clear motor control parameters explanation with **user friendly** testbench updated **LIVE**

Motor Suite

Easy visual evaluation with embedded Oscilloscope



Build your own GUI Layout for **quick and easy** debugging / troubleshooting

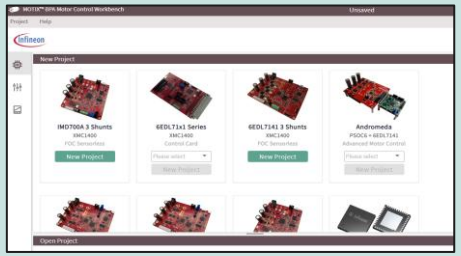
Hardware-to-Application motor control flow

In each phase of the developer journey, comprehensive support for Enablement SW and Tools.

1. Selection Hardware

Selection of BSP in MTB Motor Suite

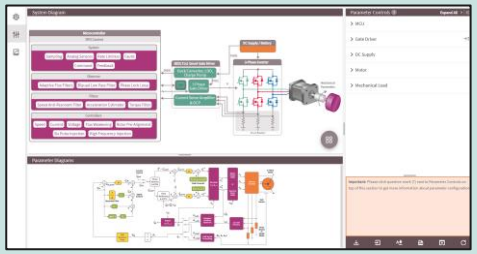
- **Hardware selection** in **Motor Suite**



2. Selection of algorithms

Easy parameterization

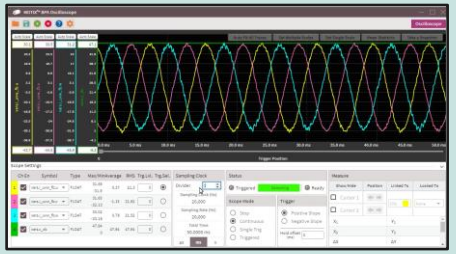
- **Quick download**, installation & SW running
- Benchmark **out-of-box kit experience** and **time to working application**



3 . Test,Tune, Monitor

Fast & efficient product development

- **Testing** and **characterization**
- **System debug** and **updating**
- **Create, edit, modify** and **adapt delivered code** for use case



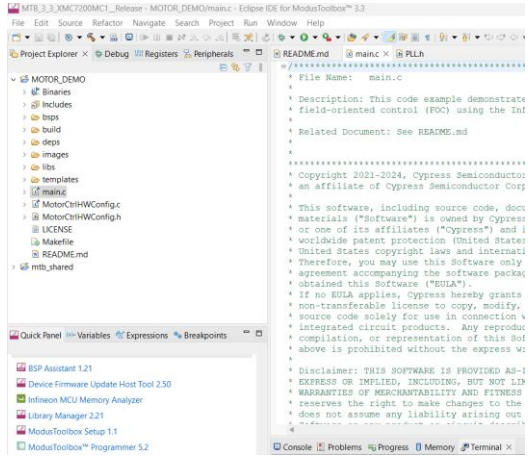
4. Application

Ready application

- Saving parameters in flash

OR

- Exporting csv/header files to be exported in MTB/other IDE



Motor Suite

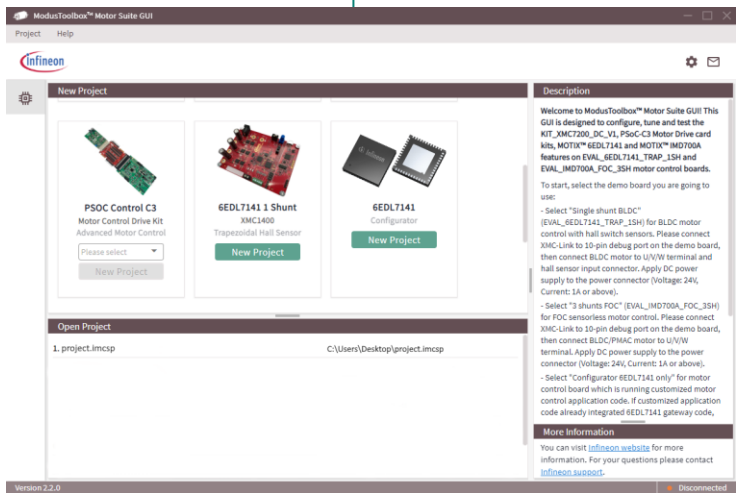
ModusToolbox™

Kits, Services and Collaterals

Choose you board → Chose your preferred algorithm → Adapt to your system → Monitor and trace → Optimize your application → Deployment and Maintenance

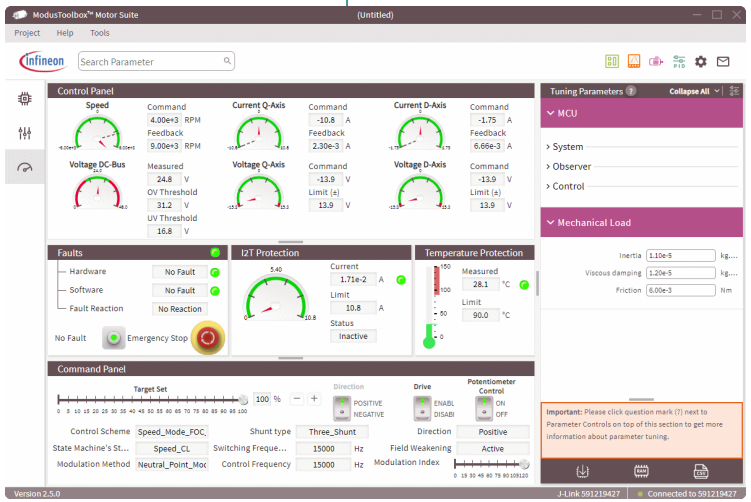
Accelerating development: Visualize motor control innovation

Automatic board Identification



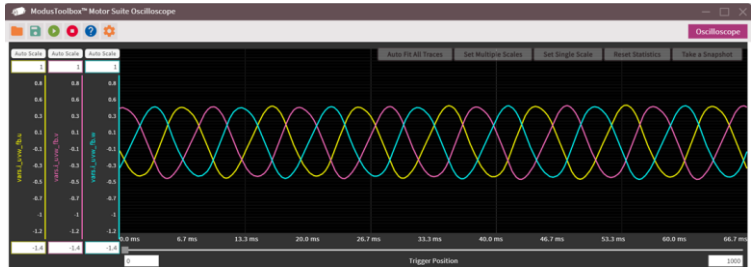
Multiple Kits and Boards supported. Easy configuration of Algorithms

Dashboard



Tune the motor and active monitoring of parameters

Oscilloscope



Easy visual evaluation with embedded Oscilloscope

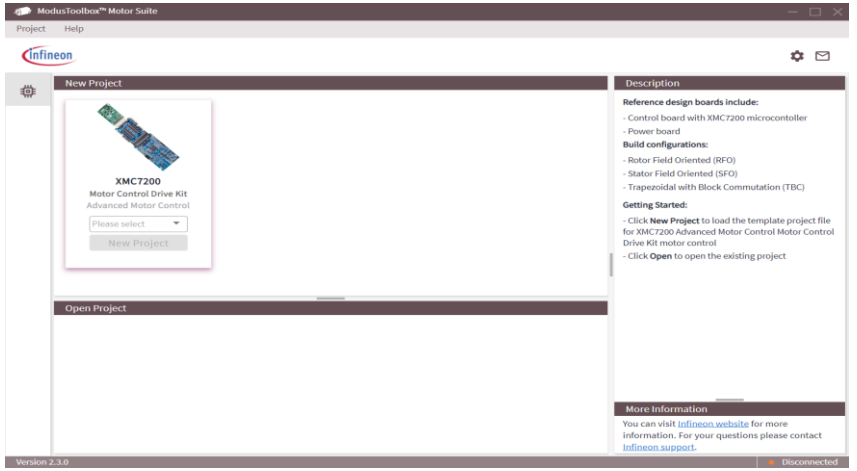
Motor suite features

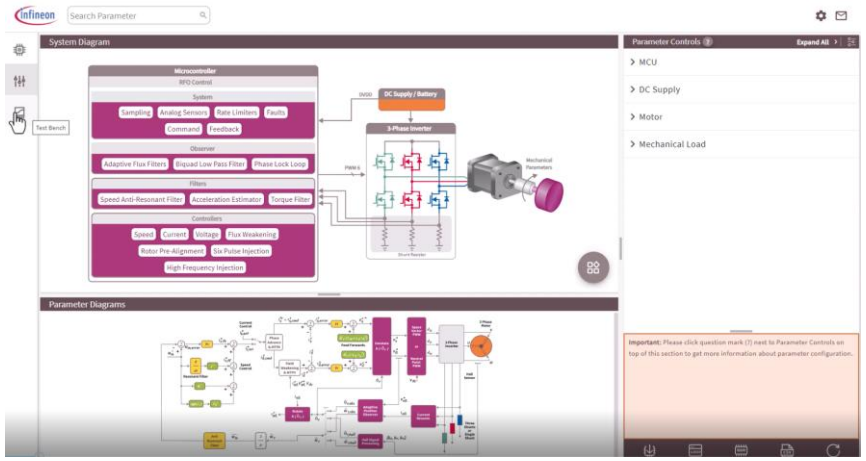
Automated kit identification

- **Instantly identify –** connected boards
- **Multi-board support –** Easily switch between different kits and boards

Interactive system diagram- System diagram and easy change of parameters

- **Edit parameters with ease –** Simple and intuitive interface
- **Visualise system parameters as a block diagram –** Clear understanding of complex systems
- **Parameter diagram visualisation –** Quickly identify relationships and dependencies
- **Live test bench updates –** Instantly see the impact of changes





Motor Suite features: Parameters configuration

Parameters controls

– Configuration for MCU, Power inverter, motor and mechanical loads

Different parameter classes

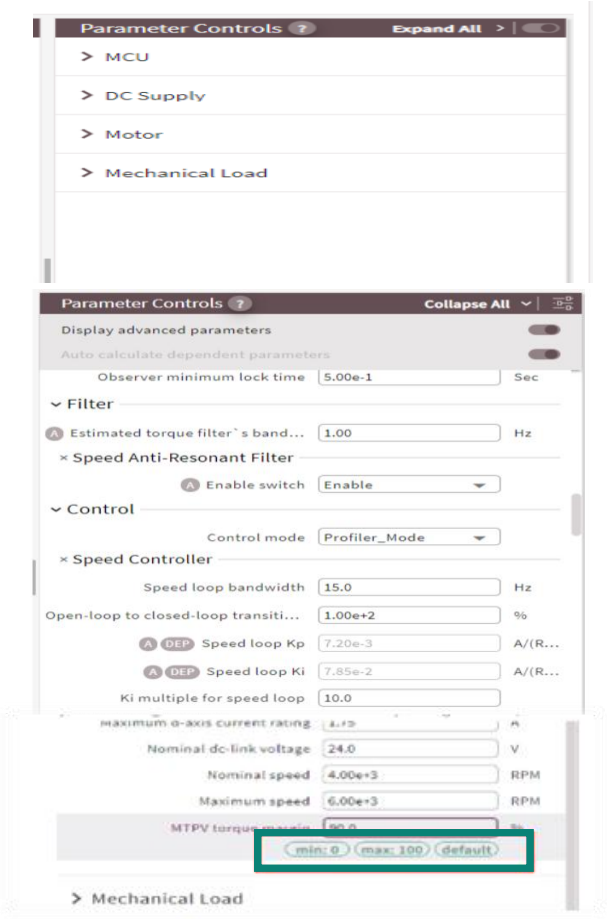
– **Direct parameters**: values are directly set by users
– **Dependent parameters**: values are automatically determined from the direct parameters

GUI display two view modes

– **Basic view**: Direct parameters are shown (default view)
– **Advanced mode**: View is suggested to the advanced user only

Range of parameters information

– Some parameter values must be entered between the minimum and maximum range
– Parameter label support to display the range of the selected parameter



Motor suite features : Flashing / Writing to MCU

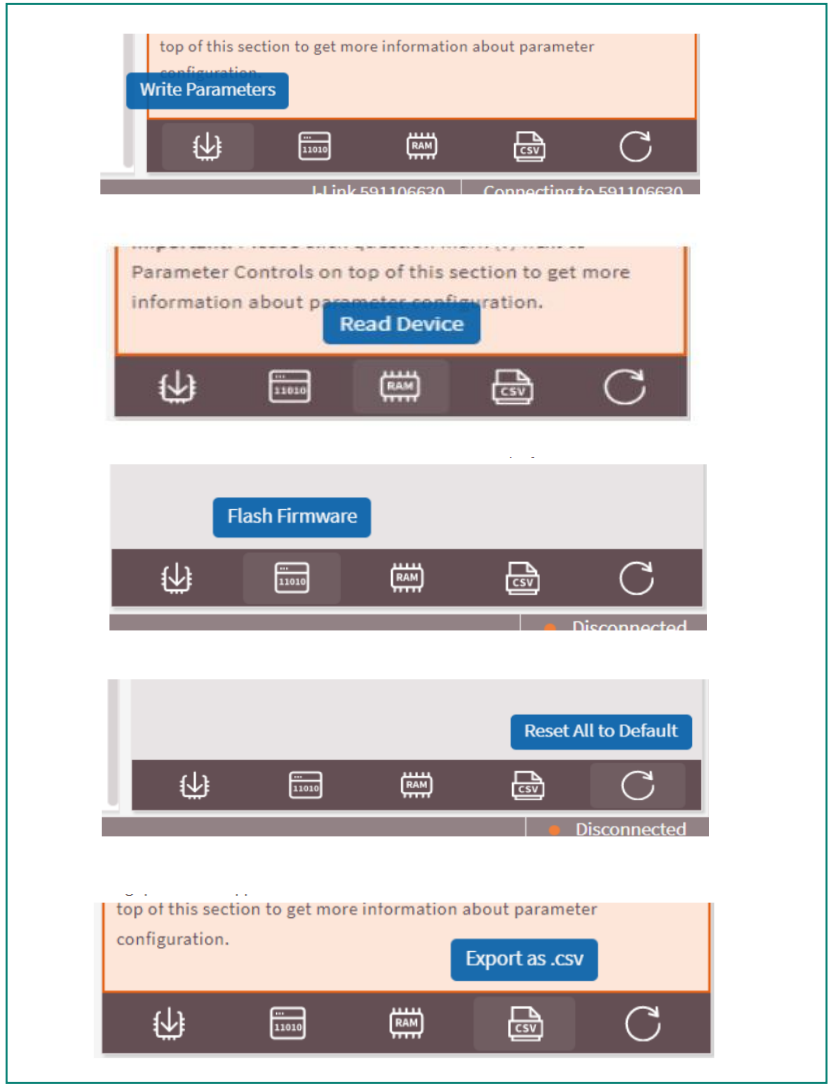
Write configuration parameters
– Write the parameters values to the board
– Option to write to RAM or Flash directly

Read device
– Read the previously saved values or default values
– This button can also sync the values between GUI and the board

Flash Application Firmware
– Default kit Firmware from the GUI can be flashed to the kit
– Option to browse to the different firmware binary files

Reset to Default values
– Option to reset all the parameters to the Default values

Export parameter to csv format
– Parameters and their values will be saved as a .csv file to the selected location



Motor Suite features: Test bench

Test bench view for advanced monitoring and Motor tuning

Visualise key parameters
- Control panel displays the expected and observed values

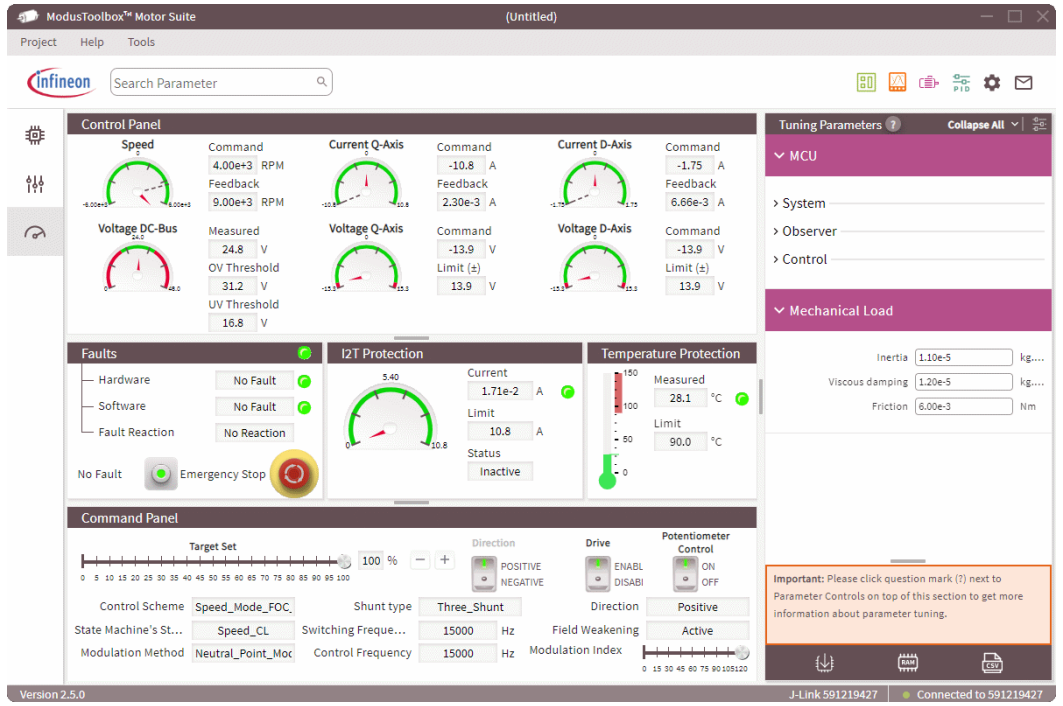
Simplified parameter tuning
- Adjust only a few basic parameters, and the firmware takes care of the rest

Faults monitoring
- Hardware and software faults can be monitored and cleared

Protection panel
- I2T and temperature protection information are displayed

Command panel
- Enable/disable the drive
- Set the target set speed via slider or potentiometer
- Monitor motor control schemes, directions of rotations
- Monitor state machines, shunt type

Seamless firmware parameters updates
- Modify your firmware parameters in SRAM without overwriting or resetting previously tuned parameters



Motor suite features: Motor profiler

Motor profiler: Simplifying motor tuning

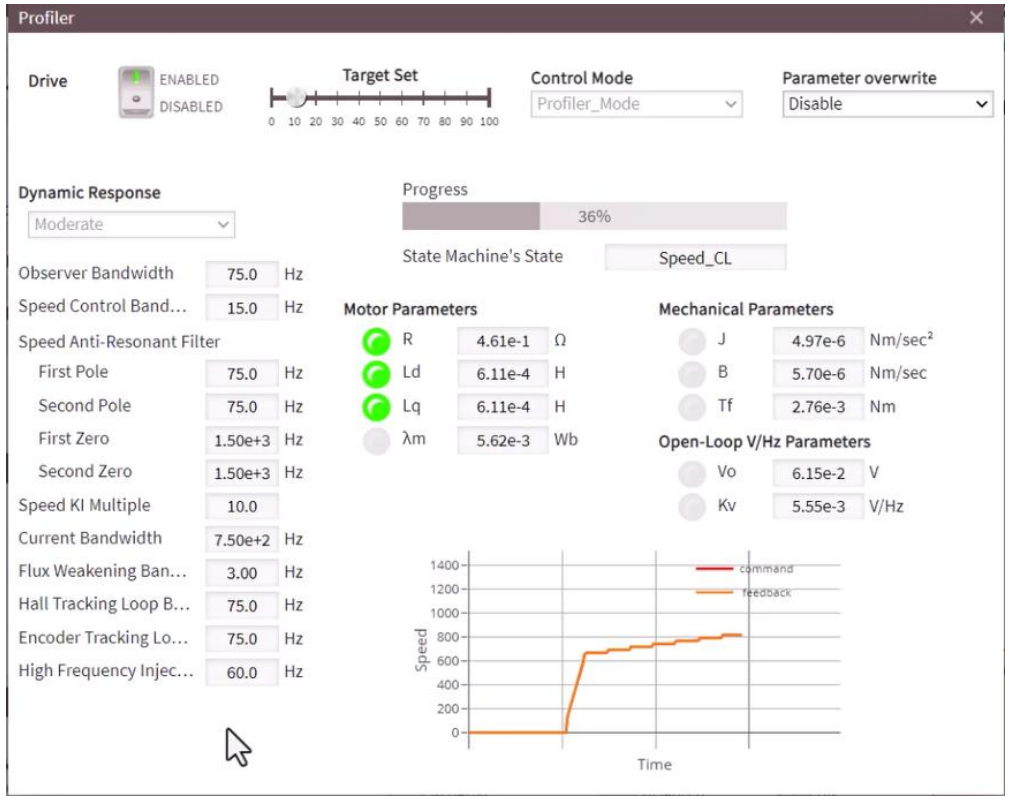
Accurate characterization
- Ensure precise motor control with automated characterisation

Parameter identifications
- Extract Motor and load parameters like R,Ld,Lq,Flux, mechanical friction, viscous and damping factors

Adjustable dynamic response
- Option to select from for options – Slow, Moderate, Fast, and Custom

One-click tuning
- Reduce manual tuning time with a single click and identify the parameters

Intelligent system management
- Let the system handle complex tasks, freeing you to focus on your application



Motor suite features: Digital oscilloscope

Digital oscilloscope for in-depth analysis

High-sample-rate oscilloscope functionality
- Capture detailed signal data
- Real-time signal statistics display

Observe multiple signals
- Physical values and internal parameters at a glance
- Trace assignment to 4 different channels

Visualise system signals
- Easily understand complex designs and behaviors

Measurements with cursors

Trigger slope and position settings

Exporting and importing data as CSV files

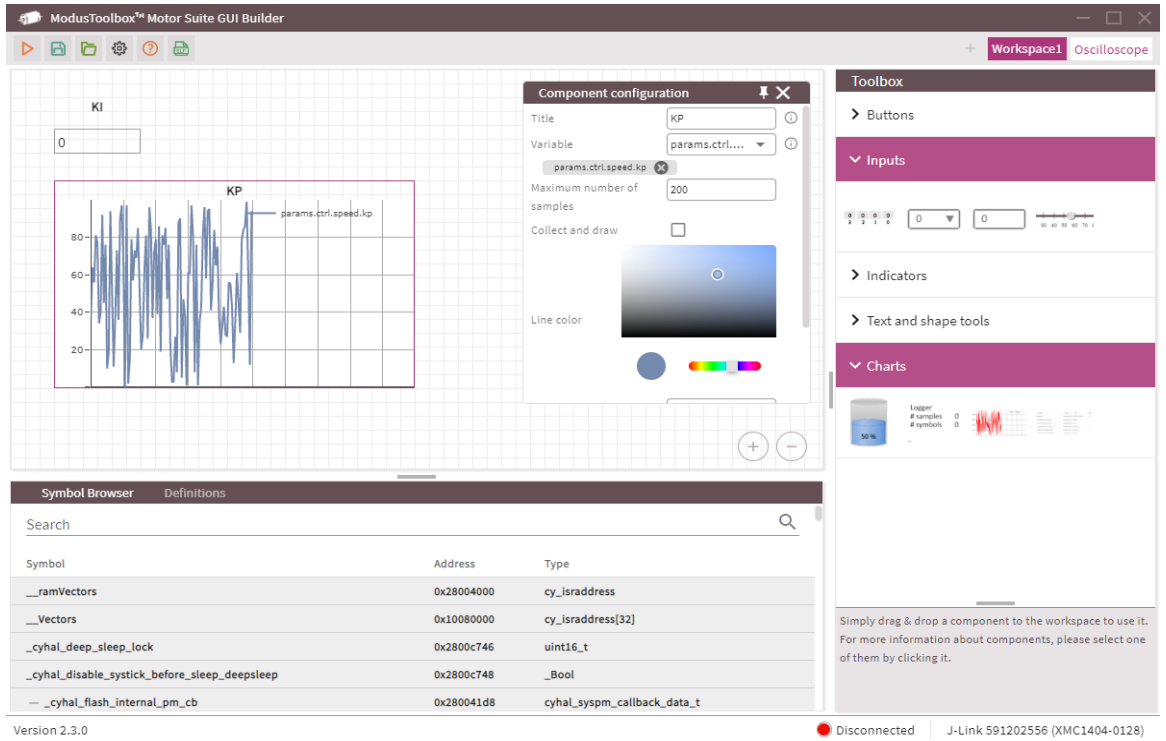
Motor suite features: GUI builder

Customizable GUI for efficient analysis

Visualise key parameters
- Focus on the metrics that matter most

Build your own GUI layout
- Drag-and-drop simplicity for quick debugging and troubleshooting

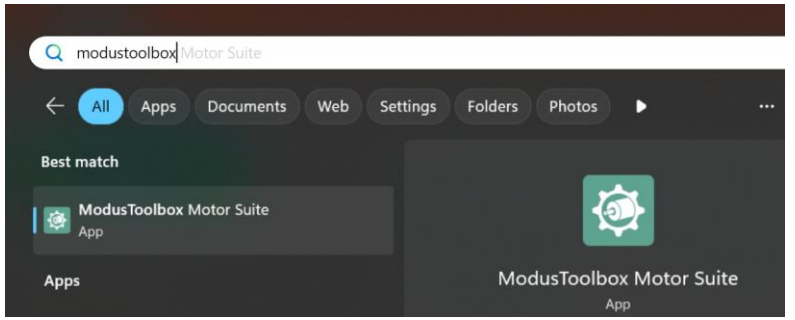
Fully customizable components
- Edit, configure, and personalise each element

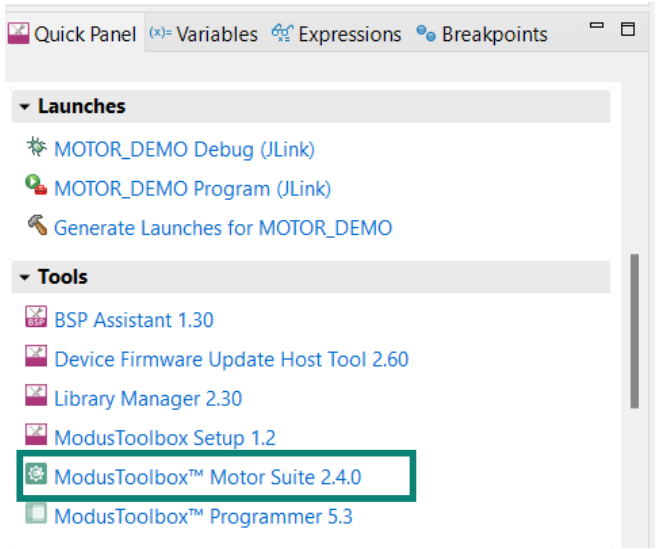


Launching the ModusToolbox™ Motor Suite GUI

1. Windows >Start >Search for ModusToolbox™ motor suite GUI

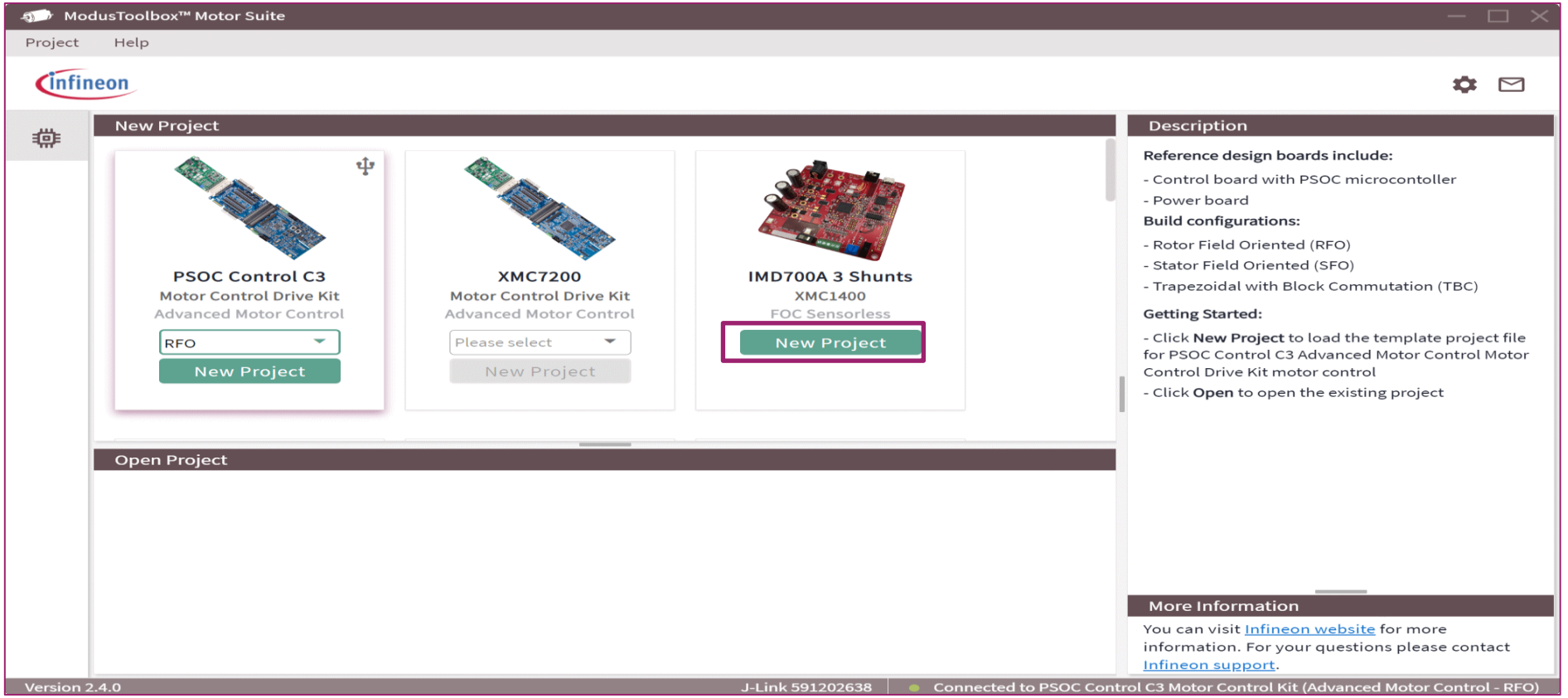
2. When compiling a project for kits supporting MTB motor suite (for example, KIT_PSC3M5_CC2 BSP), the ModusToolbox™ motor suite GUI link is available in the Quick launch panel





Open a new project

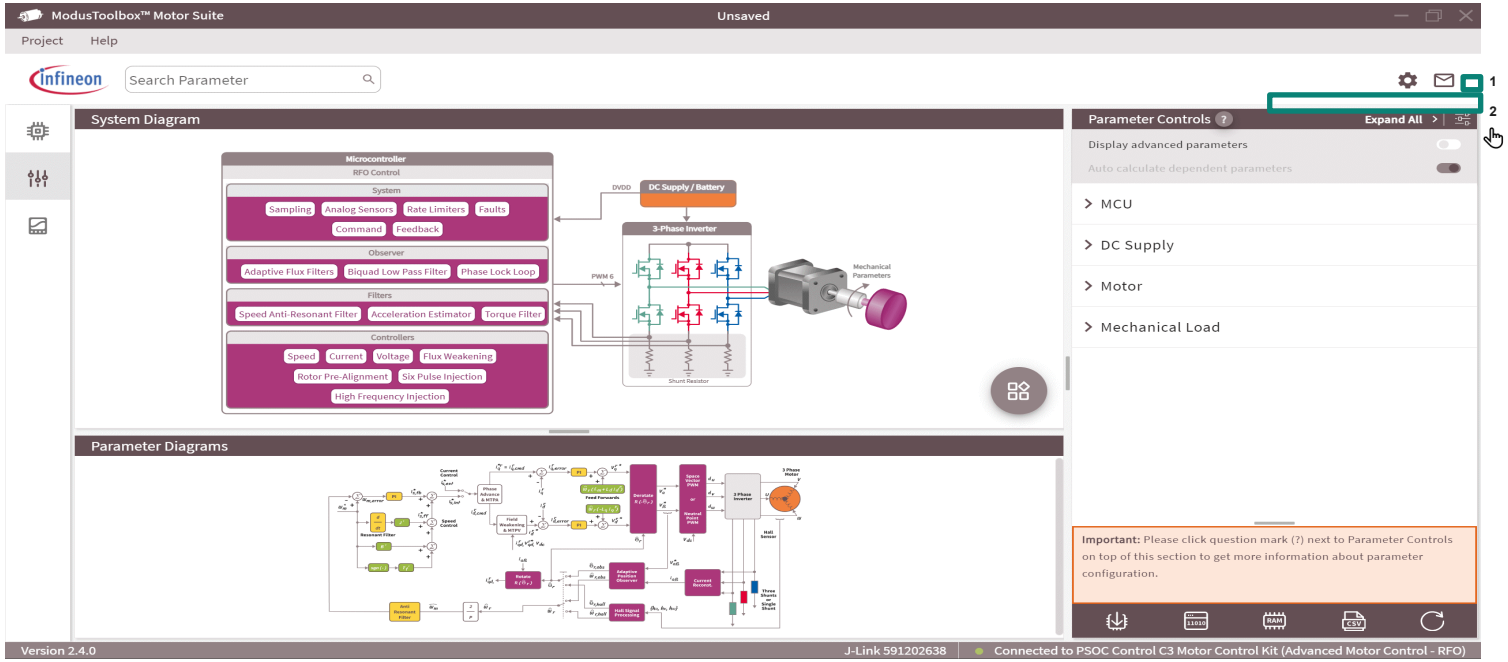
3. In the New project panel, select RFO for PSOC™ Control C3 and click New project.



Configurator view: Display advanced parameters

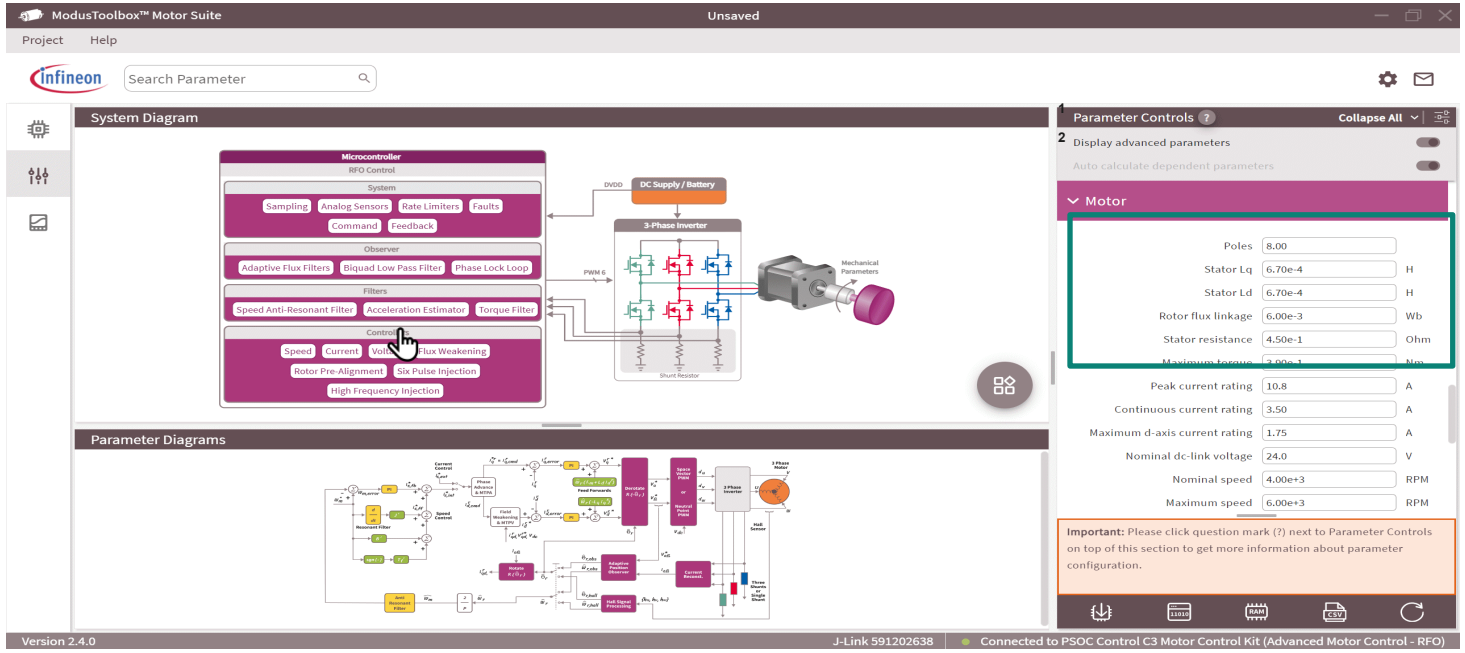
4. Display advanced parameters:
   - Click the three sliders icon in Parameter controls
   - Enable/disable Display advanced parameters

Note: Display the advanced parameters, such as MCU>Filter settings.

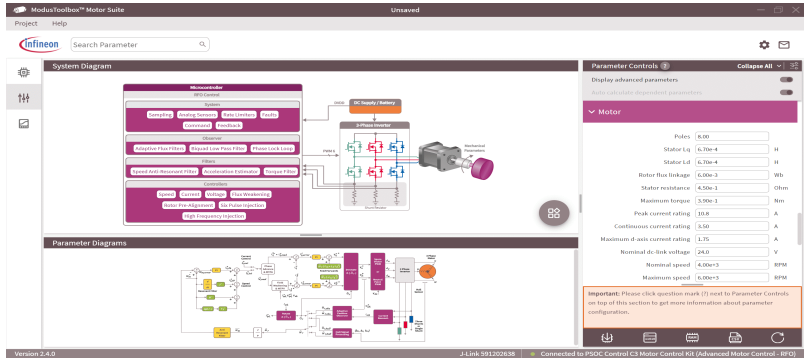


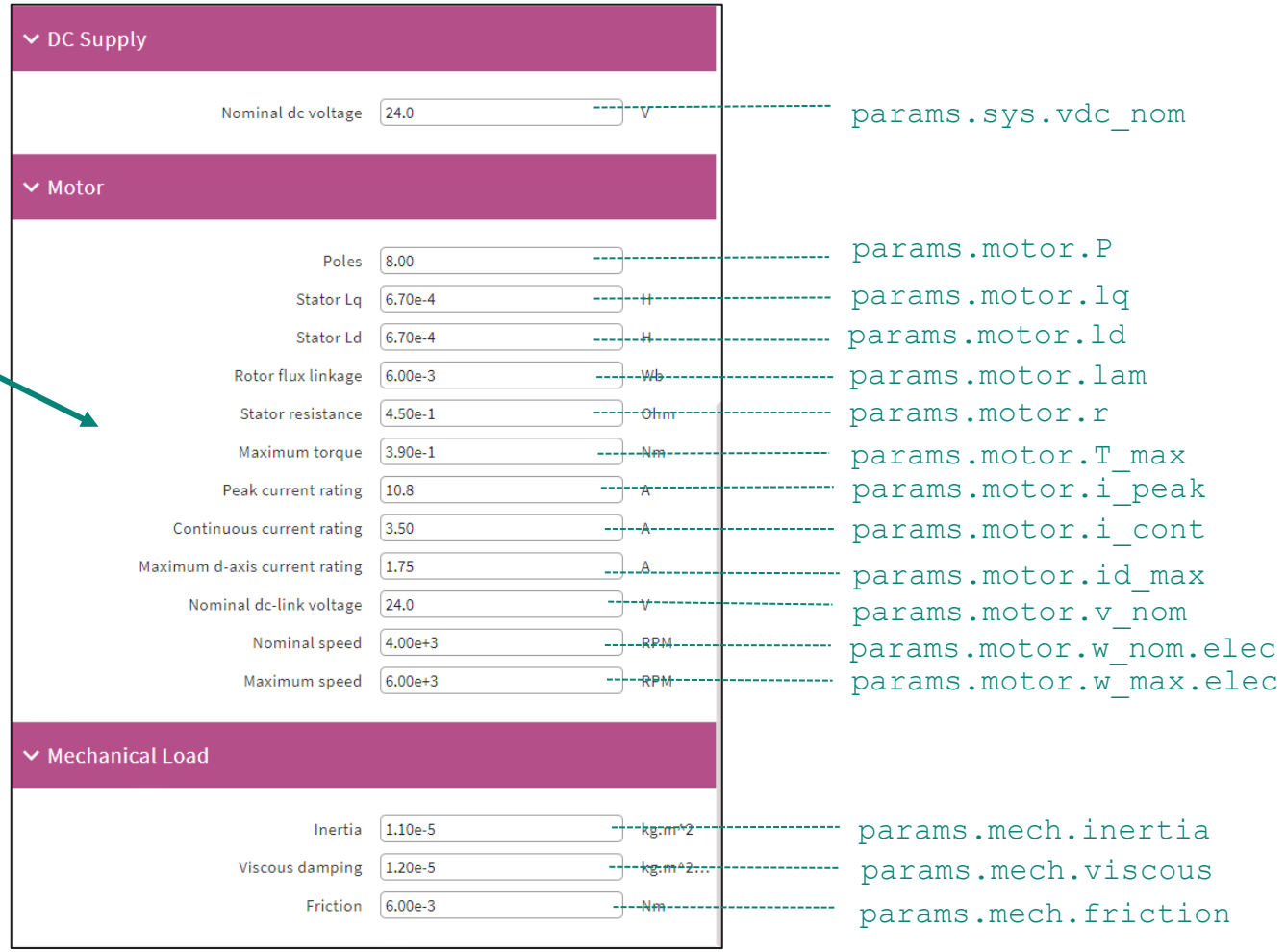
Configurator view: Parameter navigation using block diagram

5. Click any block diagram element in the System diagram to directly jump to the parameters associated with it. For example, click the motor to open the motor parameters.



Motor control suite configuration mapping to library variables

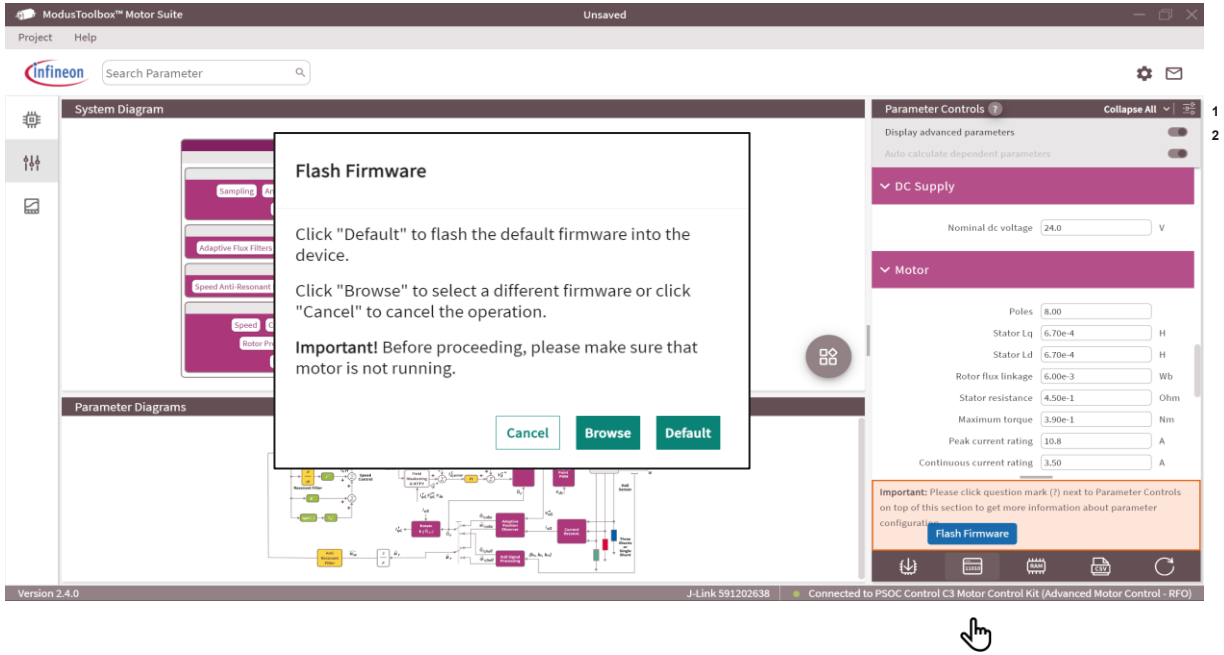




Note: Resistance and Inductance values are phase to neutral and Current is peak value

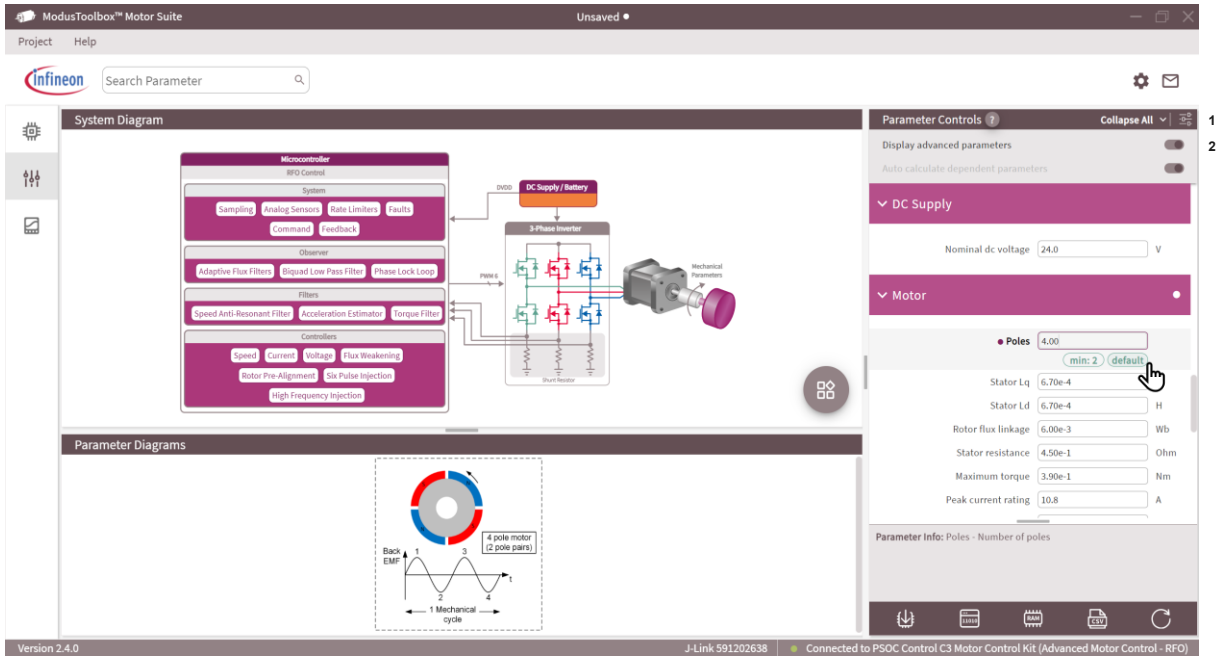
Configurator view: Flash firmware

6. The GUI can be used to flash new firmware in the target MCU.
7. The default option writes the standard MOTOR_DEMO firmware HEX.
8. Custom option allows writing a user project HEX file and use corresponding ELF file for parameter monitoring and update.



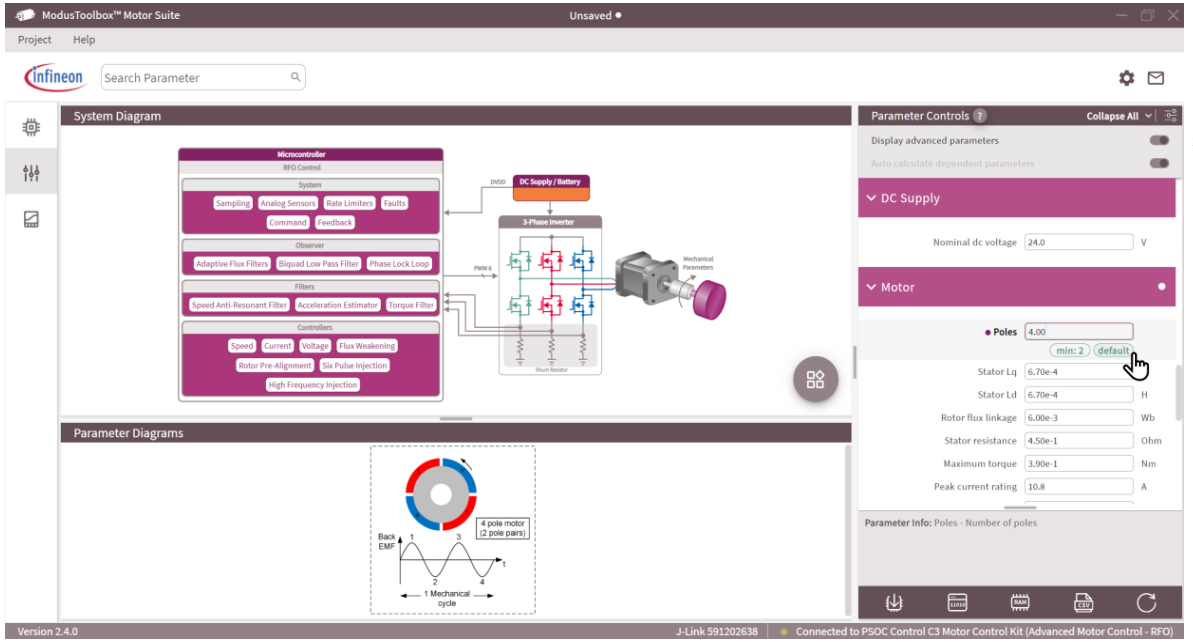
Configurator view: Revert to default parameter value

9. The modified parameters show a dot before the parameter, and the parameter box is highlighted in a different colour
10. It is possible to revert to the default parameter value by clicking the default button
11. Write the parameters to RAM or Flash to update them



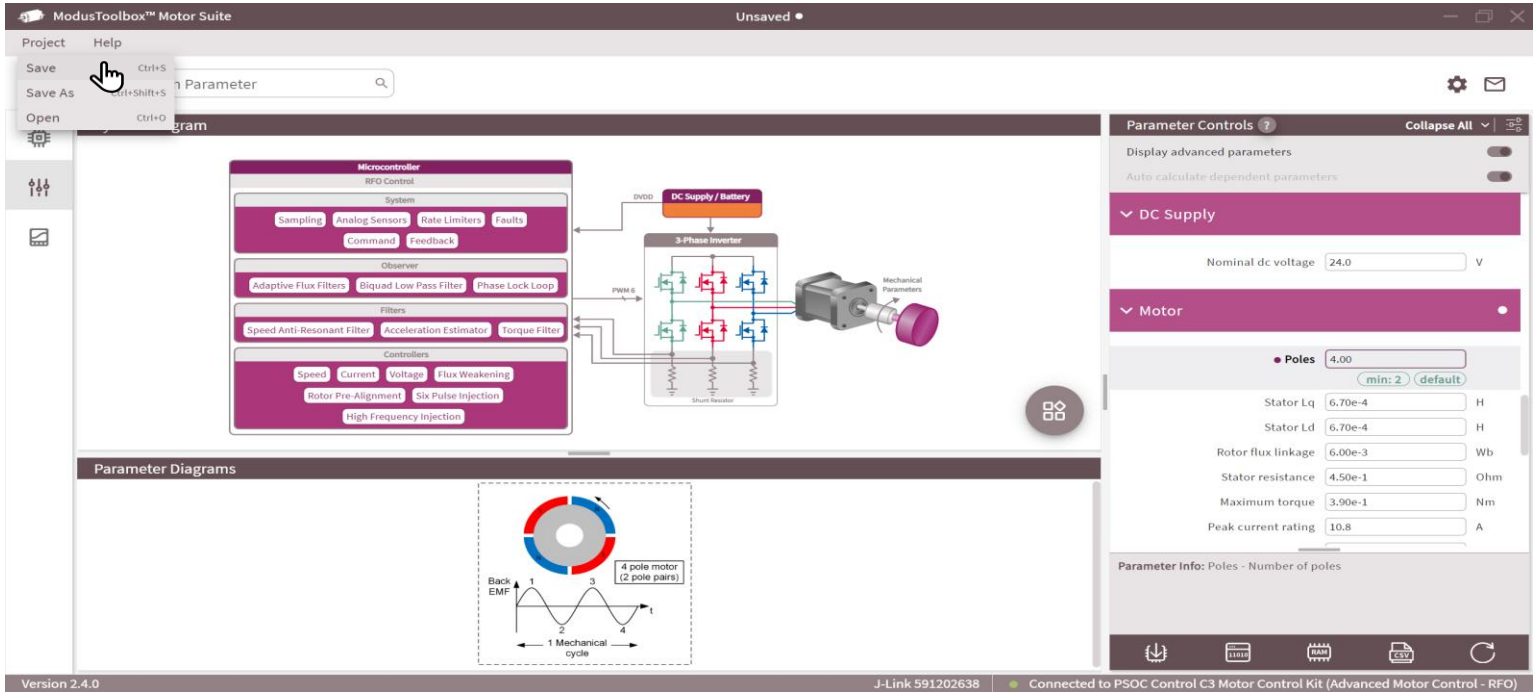
Configurator view: Corresponding parameters in source code

12. Project workspace:
    mtb_shared>motor-ctrl-lib>OperationalCode>Params.c > PARAMS_InitManual() contains the default parameter values used by the project
13. The following structures contain the corresponding parameters:
    - Motor – params.motor.xx
    - System – params.sys.xx
    - Observer – params.obs.xx
    - Mechanical – params.mech.xx
    - Filter – params.filt.xx
    - Control – params.ctrl.xx
    - Profiler – params.profiler.xx



Save/Open the GUI configuration project

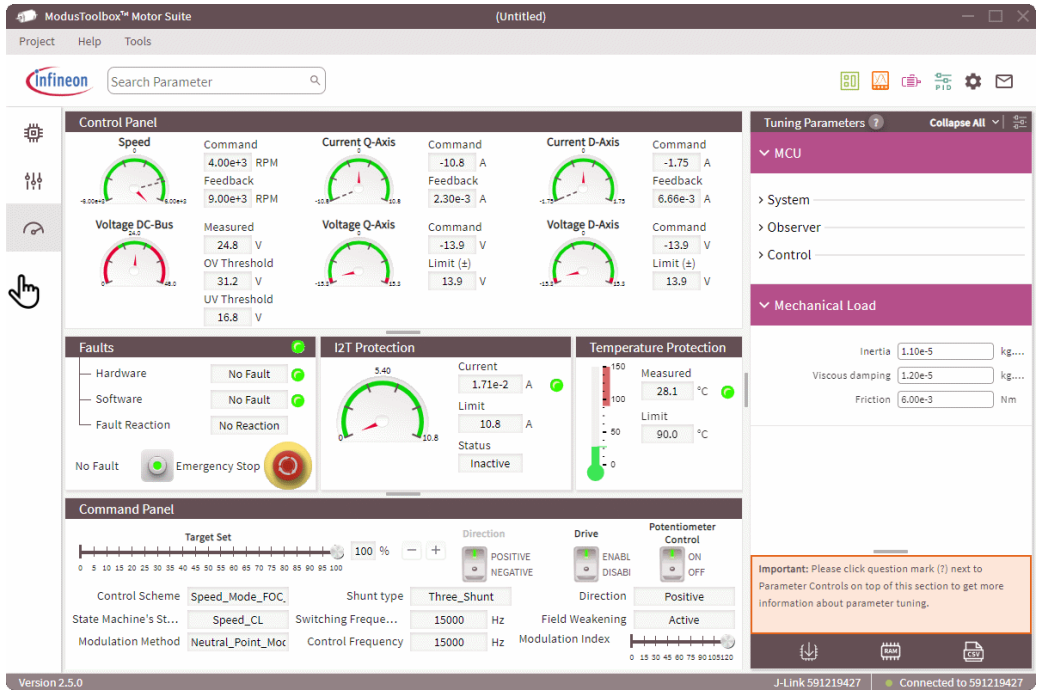
14. Click the Project -> Save or Ctrl+S.
15. Provide the project name and save the configuration file.
16. Existing saved projects can also be opened using this menu or Ctrl+O.



Test bench view

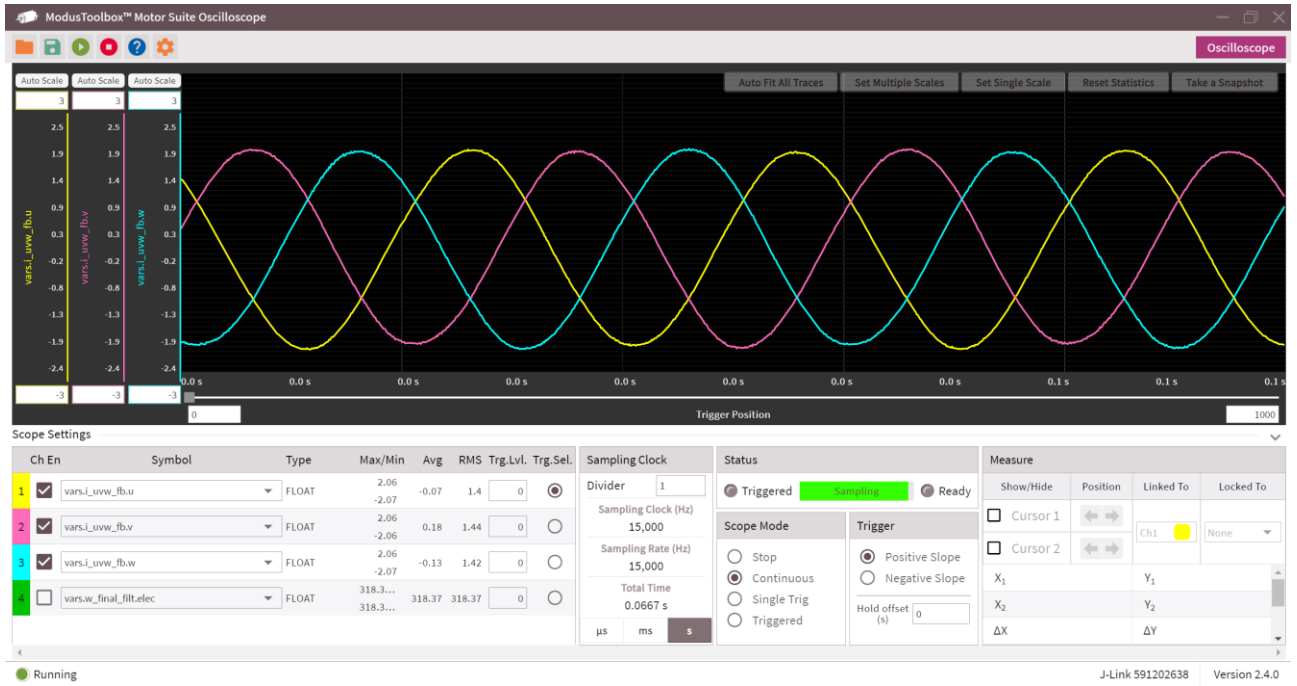
17. Test bench view shows the following:
    - Control panel: used for viewing real-time control parameters
    - Faults: shows the hardware and software faults and fault reaction. Emergency stop is used to stop the motor using the GUI
    - I2T: shows the I2T measured current, limit and status
    - Temperature protection: shows the measured temperature and the overtemperature threshold limit
    - Command panel contains:
        - Target set slider (Potentiometer control OFF)
        - Direction button (Potentiometer control OFF)
        - Drive enable/disable button
        - Potentiometer control button
        - Relevant parameter information

Note: The target set and direction buttons work only when the potentiometer control is OFF.



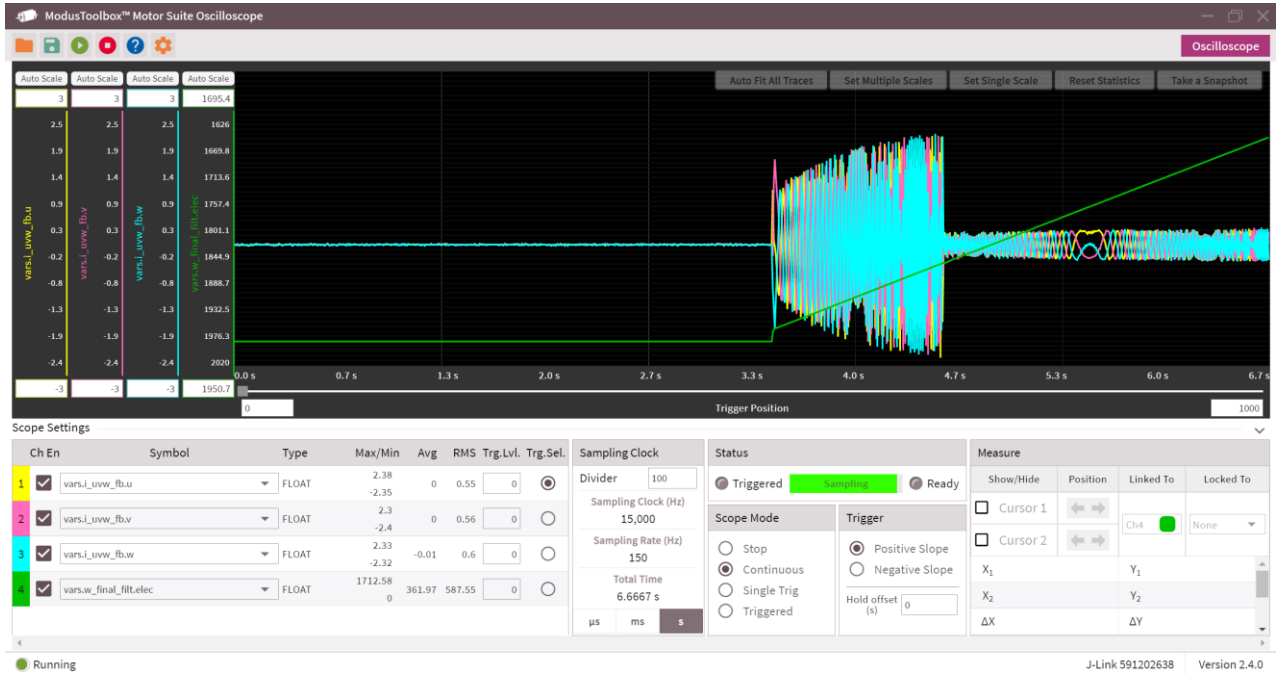
Oscilloscope

- Feature rich four channel virtual oscilloscope
- Can be used to plot the value of any global variable in real-time
- Sampling clock is the same as fast loop frequency
- Sampling rate can be slowed down to visualise longer duration (similar to changing the time scale in a real oscilloscope)
- The default parameters are u, v, and w phase currents and filtered electrical speed
- Supports continuous (free running), single-triggered and continuous-triggered modes
- Supports auto scaling for each channel



Oscilloscope: Visualize sensorless motor start-up currents

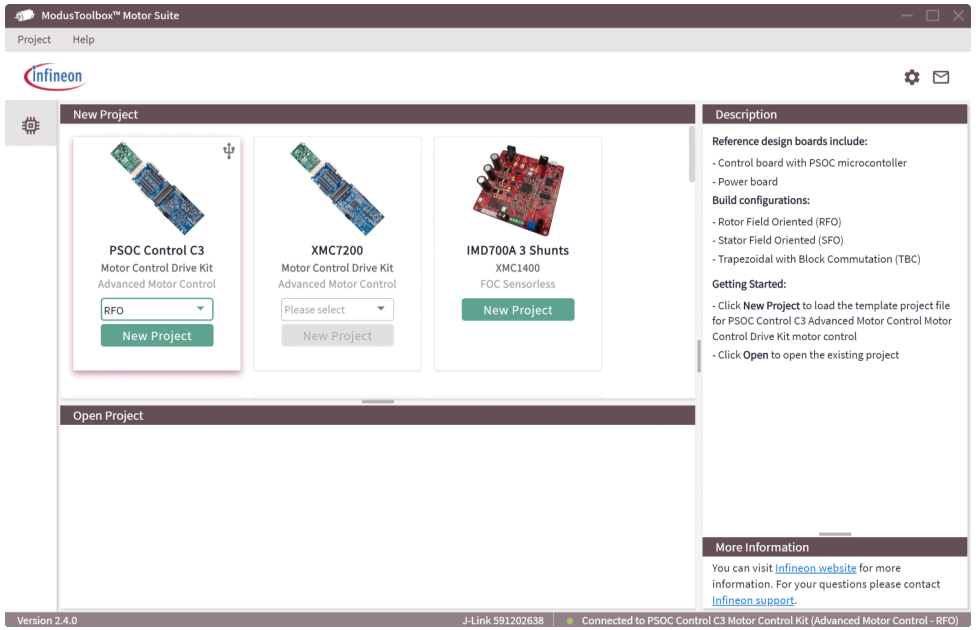
- In the default configuration, run the motor at maximum RPM
- Auto-scale the three current and electrical speed channels
- Stop the motor and change the sampling clock divider to 100
- On the target set slider, click on 100% value and see the current and electrical speed waveforms
- The open-loop motor phase currents are much higher, and once the motor transitions to closed loop (at 20% speed), the currents fall to a lower value



Software tools

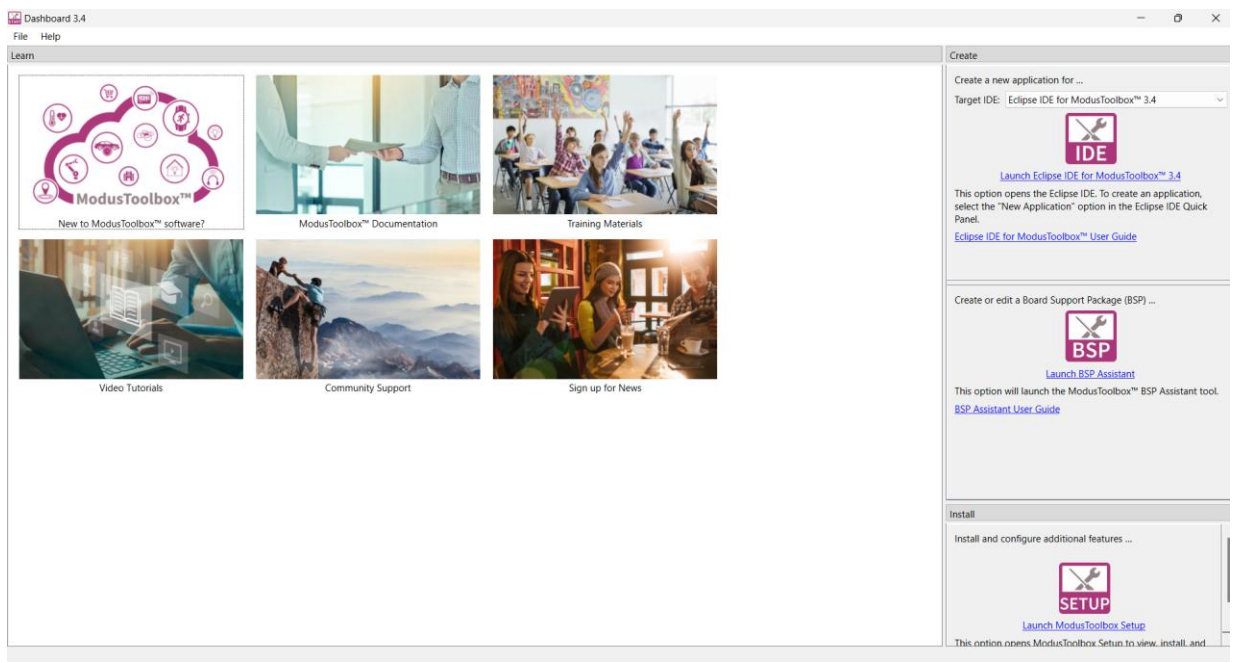
ModusToolbox™ Motor Suite GUI

- ModusToolbox™ Industrial MCU Pack
- Recommended version: 2.0.0
- Install using ModusToolbox™ Setup



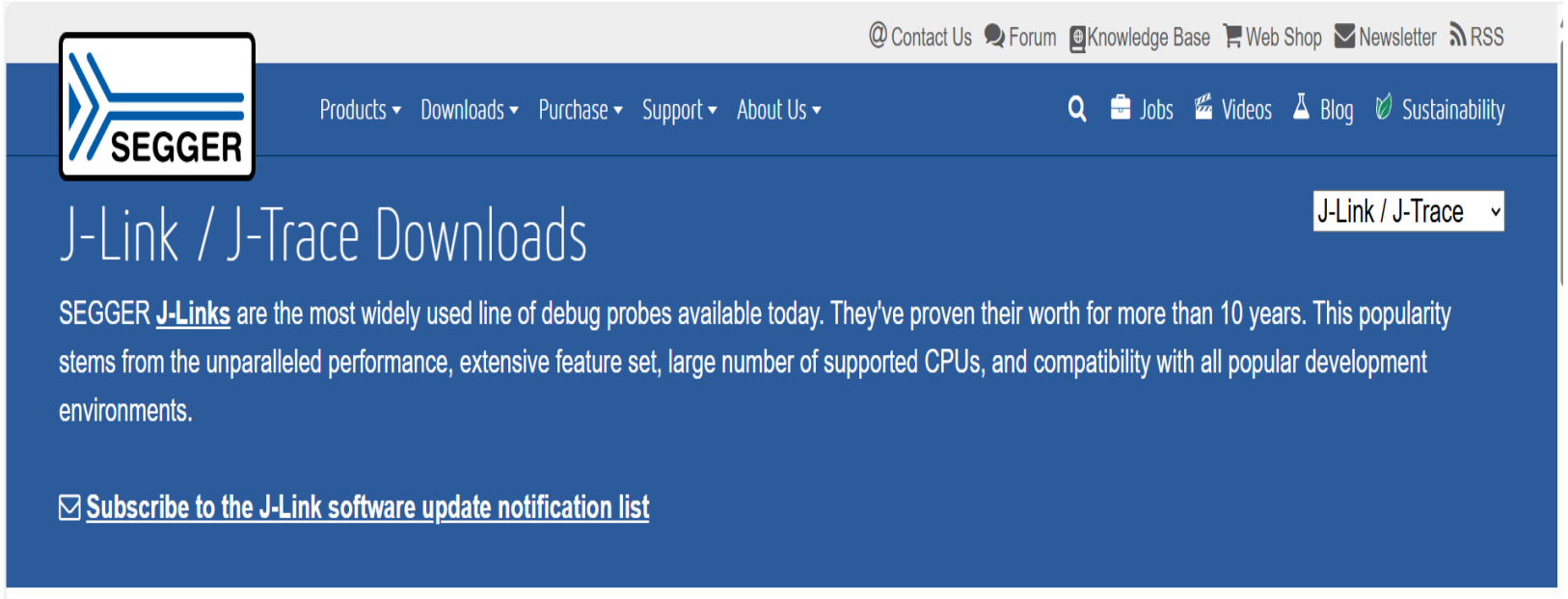
ModusToolbox™ IDE/tools package

- ModusToolbox™ tools and IDE recommended version: 3.5
- Install using ModusToolbox™ Setup

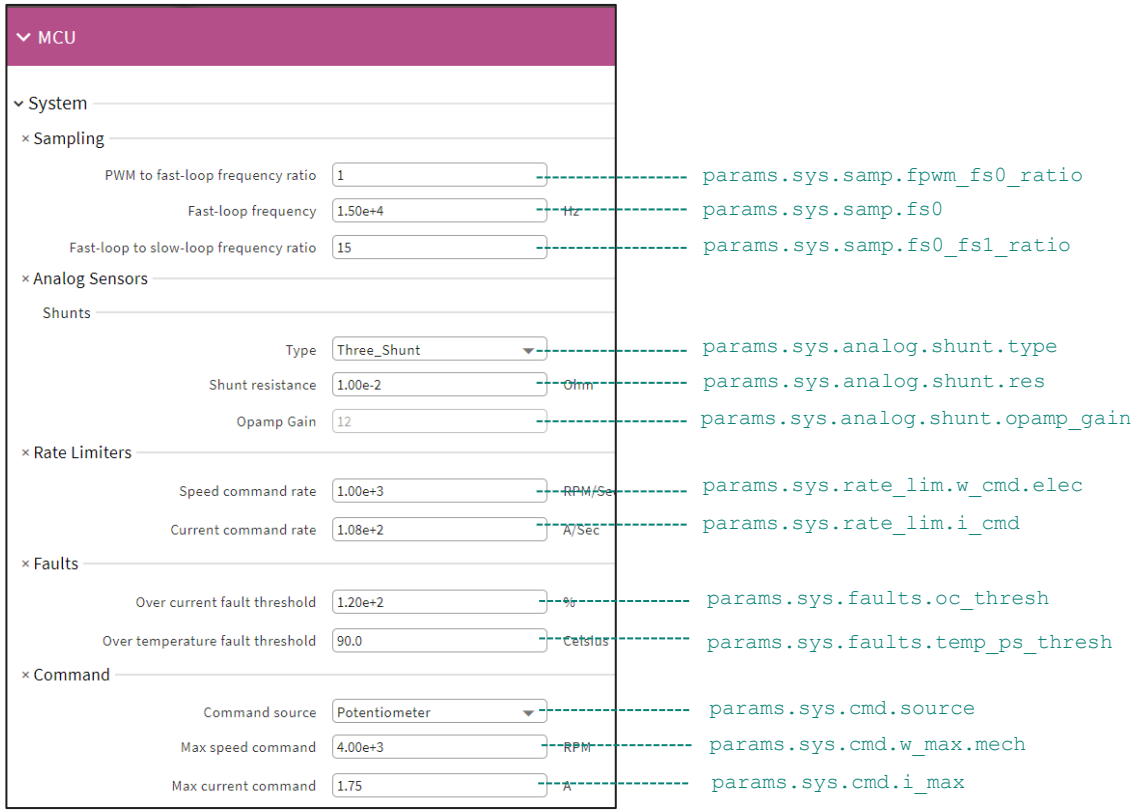


Debugger software and drivers: Segger J-link installation

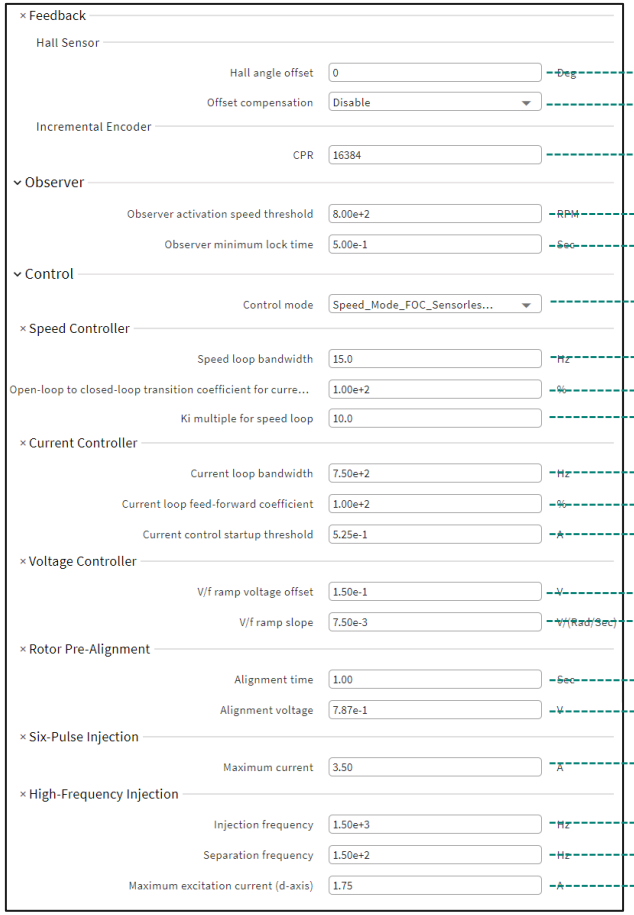
- Install the latest version: https://www.segger.com/downloads/jlink/
- **Recommendations**
  - Avoid installing multiple versions
  - Use the ‘Legacy USB Support’ option if J-Link native tools fail to connect with the target (only use if the default installation settings do not work)



Motor control suite configuration mapping to library variables



params.sys.samp.fpwm_fs0_ratio
params.sys.samp.fs0
params.sys.samp.fs0_fs1_ratio
params.sys.analog.shunt.type
params.sys.analog.shunt.res
params.sys.analog.shunt.opamp_gain
params.sys.rate_lim.w_cmd.elec
params.sys.rate_lim.i_cmd
params.sys.faults.oc_thresh
params.sys.faults.temp_ps_thresh
params.sys.cmd.source
params.sys.cmd.w_max.mech
params.sys.cmd.i_max



params.sys.fb.hall.th_r_offset.elec
params.sys.fb.hall.block_comm_offset_comp
params.sys.fb.encoder.track_loop.cpr
params.obs.w_thresh.elec
params.obs.lock_time
params.ctrl.mode
params.ctrl.speed.bw
params.ctrl.speed.ol_cl_tr_coeff
params.ctrl.speed.ki_multiple
params.ctrl.curr.bw
params.ctrl.curr.ff_coef
params.ctrl.curr.i_cmd_thresh
params.ctrl.volt.v_min
params.ctrl.volt.v_to_f_ratio
params.ctrl.align.time
params.ctrl.align.voltage
params.ctrl.six_pulse_inj.i_peak
params.ctrl.high_freq_inj.w_h
params.ctrl.high_freq_inj.w_se
params.ctrl.high_freq_inj.i_qd_r_peak.

Table of contents

Hardware overview

Hardware (KIT_PSC3M5_MC1)

- KIT_PSC3M5_MC1 contents
  - KIT_PSC3M5_CC2 (Control card)
  - Drive adapter card (100 pin to MADK adapter)
  - KIT_MOTOR_DC_250W_24V (MADK5 power board)
  - DB42M03 BLDC motor with hall sensors
  - 24V/1A wall adapter
  - USB-A to USB-C cable
- Contains everything needed to run a BLDC/PMSM motor using the ModusToolbox™ motor control ecosystem
- Kit webpage: www.infineon.com/KIT_PSC3M5_MC1
  - Release Notes
  - Quick Start Guide
  - User Guide
- Preprogrammed with firmware to run the motor in standalone mode or with the ModusToolbox™ Motor Suite GUI
- Orderable product: **KITPSC3M5MC1**

Kit contents

Kit features - KIT_PSC3M5_CC2

1. Infineon PSOC™ Control C3M5 (Arm® Cortex® -M33 based) microcontroller PSC3M5FDS2AFQ1, 180 MHz, up to 256 KB flash / 64 KB SRAM, E-LQFP-80
2. Connection to MADK boards (M1/M3/M5) via 100-pin HD connector connected to the XMC™ drive adapter card
   - Resources capable to do dual motor control with PFC
   - Three motor control only
   - One single motor control with advance PFC
   - SPI/Serial communication for external sensors or gate drivers
3. Five LEDs
   - One Power LED
   - Two User LEDs: User-controlled LED
   - Two Debug LEDs and Aux LEDs: Debugger-controlled LEDs
4. Isolated debug options (default): Onboard isolated debugger (SEGGER J-Link Lite) via USB connector
5. Isolated connectivity
   - UART channel of on-board debugger (SEGGER J-Link Lite) via USB connector
   - CAN interface on a 4-pin header X14
6. Two non-isolated debug options
   - SWD/JTAG 10-pin 1.27 mm header
   - ETM 20-pin 1.27 mm header
7. Power supply - PSOC™ Control C3M5 MCU
   - Via 100-pin expansion board (5 V or 24 V) converted to 3.3 V
   - Via debug USB connector, 5 V isolated DC-DC converted to 3.3 V

Default configuration and known limitations

- The kit is designed to work with both 3.3 V and 5 V power boards.
  Default: 5 V scaled down to 3.3 V
  Desolder some resistors on control card to make it compatible with 3.3 V power boards. See the kit release notes section 2.5 for more details.

- MADK kits without current sense amplifiers not supported

- Encoder inputs X2 and X4 only support differential encoders. Hall sensor inputs X3 and X7 can be used for connecting single ended encoders

Table of contents

Adaptation to use new motor

Adaptation to use new motor

- Example 1: Moons motor - 57BL60L2
- Example 2: Anaheim Automation – BLY172S-24V

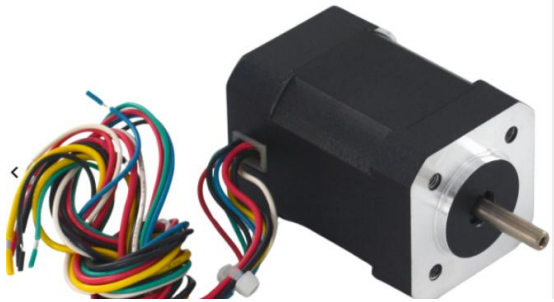
57BL60L2

Brushless DC Motor, 60Watts，Frame 57

- Wide speed range, flat torque
- Excellent speed stability
- Compact and high power
- High efficiency and energy-saving
- Low temperature rise, low noise, low vibration
- Long life and low maintenance cost
- Low cogging torque

Learn More About **S57 Series Brushless DC Motors**

Brand Name: MOONS'



BLY172S-24V-4000

Rated Voltage (V)	24
Rated Power (Watts)	53
Rated Torque (oz-in)	18
Rated Speed (RPM)	4000
Torque Constant (oz-in/A)	5.81
Back EMF Voltage (V/kRPM)	3.35
Line-to-Line Resistance (ohm)	0.8
Line To Line Inductance (mH)	1.2
Rotor Inertia (oz-in-sec²)	0.000680
"L" Length (in)	2.37
Weight (lbs)	0.99

Parameter configuration hints

- Motor control parameters
  - params.motor.P represents the number of poles, not pole pairs
  - params.motor.i_cont is the continuous current rating of the motor
  - params.motor.i_peak is the peak current rating of the motor. Set this value to 2.5 to 3x of motor continuous current, if not specified in the motor datasheet
  - params.motor.id_max is the maximum d-axis current rating of the motor that doesn't result in demagnetization of the permanent magnet. Set this value to half of motor continuous current, if not specified in the motor datasheet
  - params.motor.v_nom is the motor nominal voltage
  - Current value is based on the peak values, not rms

Parameter configuration hints

System parameters

- params.sys.cmd.w_max.mech is the maximum speed the user wants to command via potentiometer. This can be different from maximum speed of the motor
- params.sys.vdc_nom is the DC bus voltage., this value is the same as params.motor.v_nom. However, if the user may want to apply 24 V to a 48 V motor. So, vdc_nom needs to be set as 24 V rather than 48 V in this case

Control parameters

- Set the value of params.ctrl.curr.bw at least 3~5 times lower than switching frequency. In addition, it must be set higher than params.ctrl.speed.bw

V/F startup parameters

- params.ctrl.volt.v_min calculate from motor resistance and continuous current

$$\text{params.ctrl.volt.v\_min} = \frac{(\text{params.motor.r} * \text{params.motor.i\_cont})}{k}$$
$$k: 3 \sim 10$$

Parameter configuration hints

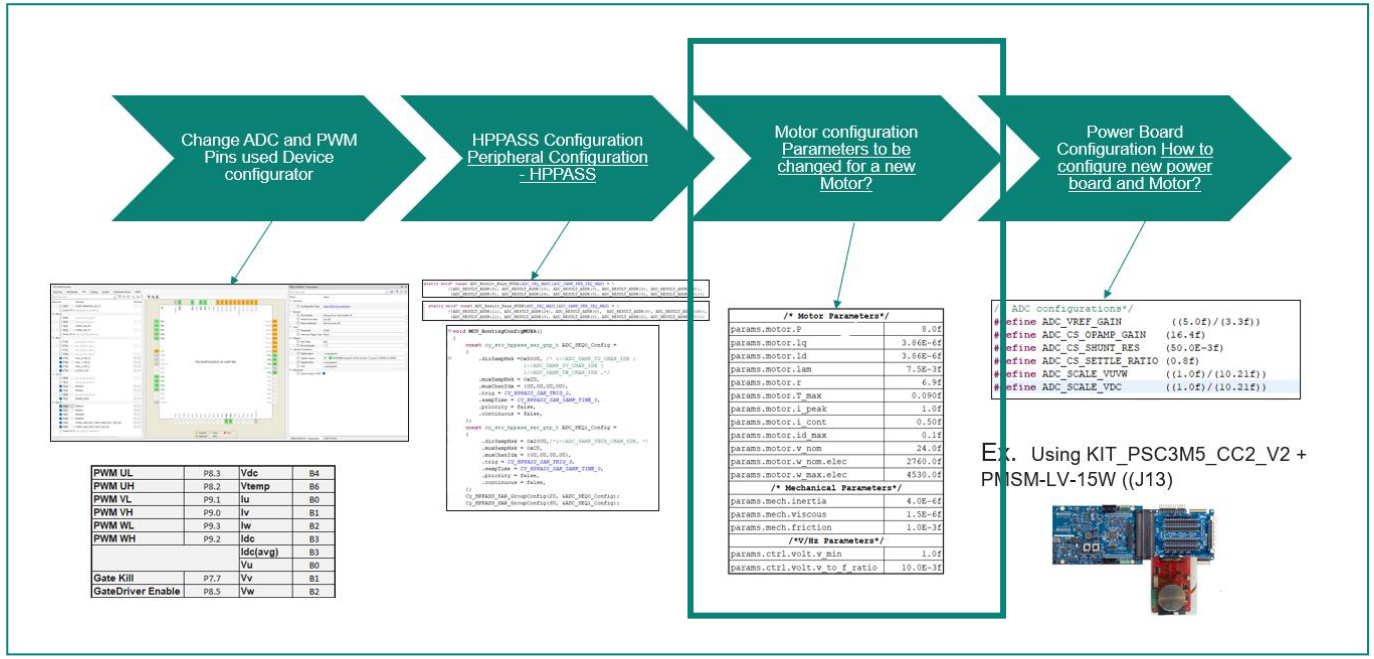
– Control related bandwidth for SLOW, MODERATE, and FAST loop response area listed below for reference.

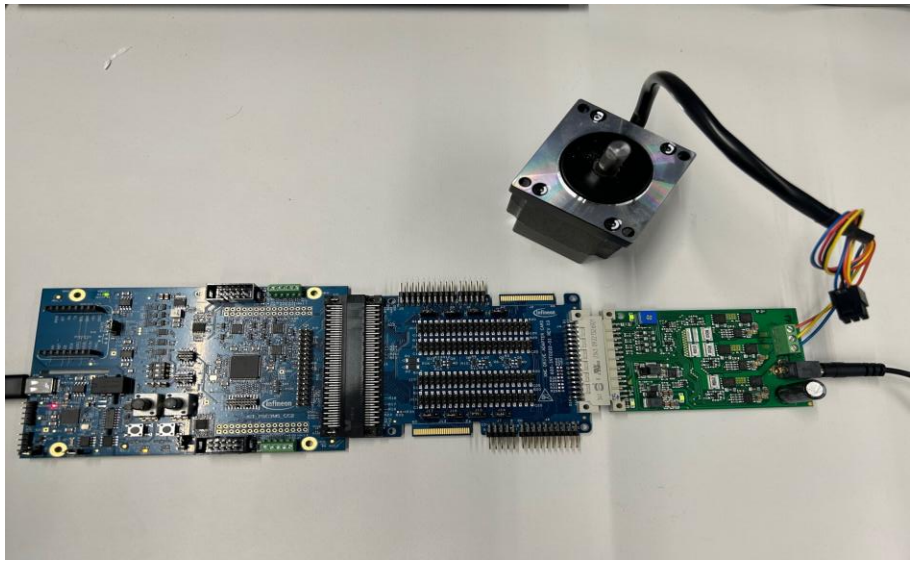
```
// Parameter (bandwidths, etc.) ....... SLOW  ................. MODERATE ............. FAST ......................
{&params.obs.pll.w0,                    HZ_TO_RADSEC(50.0f),    HZ_TO_RADSEC(75.0f),    HZ_TO_RADSEC(100.0f)},      // [Ra/sec-elec]
{&params.ctrl.speed.bw,                 HZ_TO_RADSEC(10.0f),    HZ_TO_RADSEC(15.0f),    HZ_TO_RADSEC(20.0f)},       // [Ra/sec-elec]
{&params.filt.spd_ar_wp[0],             HZ_TO_RADSEC(50.0f),    HZ_TO_RADSEC(75.0f),    HZ_TO_RADSEC(100.0f)},      // [Ra/sec-elec]
{&params.filt.spd_ar_wp[1],             HZ_TO_RADSEC(50.0f),    HZ_TO_RADSEC(75.0f),    HZ_TO_RADSEC(100.0f)},      // [Ra/sec-elec]
{&params.filt.spd_ar_wz[0],             HZ_TO_RADSEC(1000.0f),  HZ_TO_RADSEC(1500.0f),  HZ_TO_RADSEC(2000.0f)},     // [Ra/sec-elec]
{&params.filt.spd_ar_wz[1],             HZ_TO_RADSEC(1000.0f),  HZ_TO_RADSEC(1500.0f),  HZ_TO_RADSEC(2000.0f)},     // [Ra/sec-elec]
{&params.ctrl.speed.ki_multiple,        4.0f,                   10.0f,                  12.0f},                     // [%]
                    // ........ PM ~ 65 deg ........... PM ~ 60 deg ........... PM ~ 45 deg ................
{&params.ctrl.curr.bw,                  HZ_TO_RADSEC(500.0f),   HZ_TO_RADSEC(750.0f),   HZ_TO_RADSEC(1000.0f)},     // [Ra/sec-elec]
{&params.ctrl.flux_weaken.bw,           HZ_TO_RADSEC(2.0f),     HZ_TO_RADSEC(3.0f),     HZ_TO_RADSEC(4.0f)},        // [Ra/sec-elec]
{&params.sys.fb.hall.track_loop.w0_w,   HZ_TO_RADSEC(50.0f),    HZ_TO_RADSEC(75.0f),    HZ_TO_RADSEC(100.0f)},      // [Ra/sec-elec]
{&params.sys.fb.encoder.track_loop.w0_w,HZ_TO_RADSEC(50.0f),    HZ_TO_RADSEC(75.0f),    HZ_TO_RADSEC(100.0f)},      // [Ra/sec-elec]
{&params.ctrl.high_freq_inj.pll.w0,     HZ_TO_RADSEC(40.0f),    HZ_TO_RADSEC(60.0f),    HZ_TO_RADSEC(80.0f)},       // [Ra/sec-elec]
```

Adaptation to use new motor

Use moons motor with MC1 kit

- Do changes to motor parameters
- Do changes to mechanical loads for speed control
- Do changes to the open loop V/F parameters





Example 1: Adapt new motor – Moons LV motor- 57BL60L2 (1/5)

- Using moons motor – 57BL60L2
- When use different motor, following motor related parameters in Params.c file to be updated based new motor
  - Motor parameters
  - Load / Mechanical parameters
  - Startup method parameters, for example, V/Hz startup parameter if V/Hz startup is used
- Use parameters from Datasheet
  - Poles number – 8
  - Rated speed – 4000 RPM
  - Maximum speed – 5000 RPM
  - DC link voltage – 24 V
  - Rated continuous current – 3.2 A
  - Inductance ph-ph - 0.45 mH/2 (calculate per phase value – Delta/Star) = 0.225 mH
  - Resistance ph-ph – 0.48 Ω/2 (calculate per phase value – Delta/Star) = 0.24 Ω

Datasheet specs - **57BL60L2**

Specifications

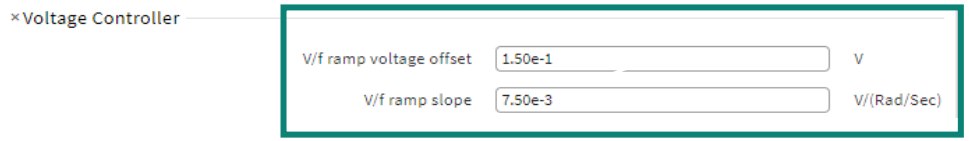
1	额定电压/Rated power supply	24	VDC
2	额定输出功率/Rated output power	60	W
3	极数/Poles number	8	
4	相数/Phase number	3	
5	额定转速/Rated rotational speed	4000	RPM
6	最大转速/Maximum speed	5000	RPM
7	额定转矩/Rated torque	0.145	N·m
8	额定电流/Rated winding current	3.20	A
9	线反电势系数/Voltage constant±10%	3.08	Vrms/krpm
10	惯量/Moment of inertia	1.4767x10^-5	kg·m^2
11	线电阻Ph/ph resistance±10% at 20℃	0.48	Ω
12	线电感Ph/ph inductance±30% at 1KHz 20℃	0.45	mH
13	电机质量/Motor mass	0.64	kg
14	绝缘等级/Insulation class	E	120℃
15	防护等级/Protective enclosure rating	IP40	

Note: The parameters are defined in \mtb_shared\\motor-ctrl-lib\release-v1.9.0\OperationalCode\Params.c file.

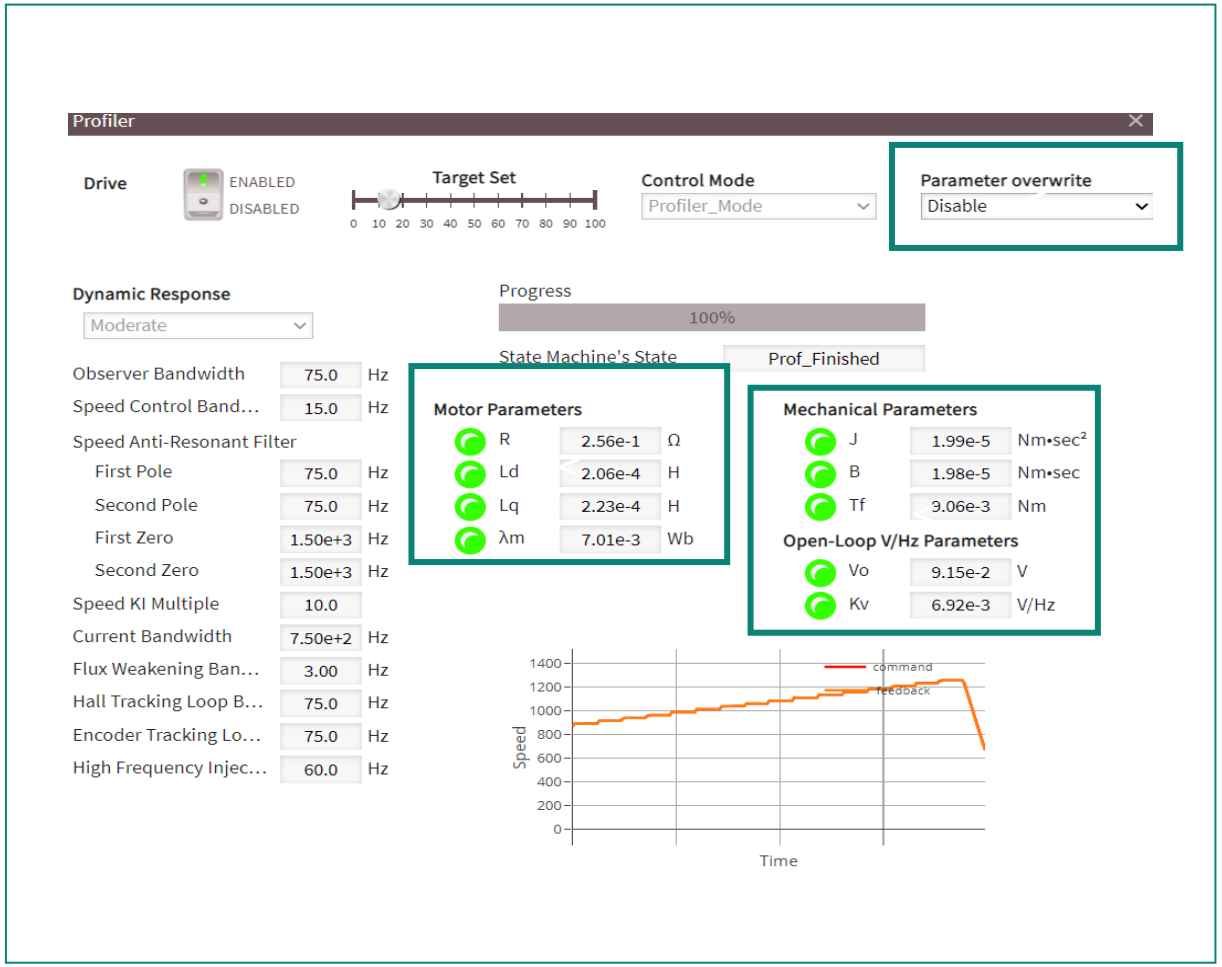
Example 1: Adapt new motor – Moons LV motor- 57BL60L2 (2/5)

- Alternatively, run the profiler to calculate unknow parameters
  - Motor parameters
  - Load/ Mechanical parameters
  - Startup method parameters; for example,V/Hz startup parameter if V/Hz startup is used

- Note in case the profiler does not run properly for new motor , first adjust the open loop voltage controller
  - V/F ramp voltage offset
  - V/F ramp slope



Hint : Start with decreasing the V/F ramp slope if the motor need higher current in startup during V/F



Note: The parameters are defined in \mtb_shared\\motor-ctrl-lib\release-v1.9.0\OperationalCode\Params.c file.

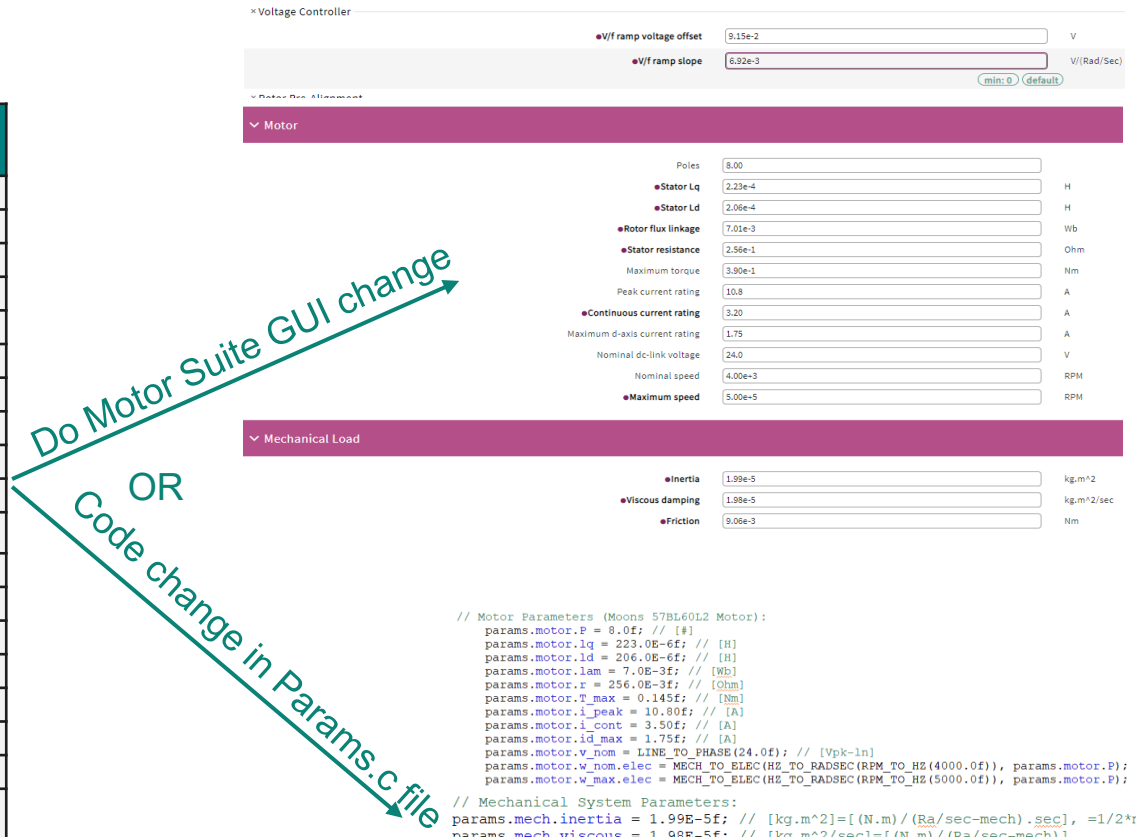
Example 1: Adapt new motor – Moons LV motor- 57BL60L2 (3/5)

- Fill the value in code or in GUI for the following:
  - Motor parameters
  - Load/ mechanical parameters
  - Startup method parameters; for example, V/Hz startup parameter if V/Hz startup is used

Note: The parameters are defined in \mtb_shared\\motor-ctrl-lib\release-v1.9.0\OperationalCode\Params.c file.

Example 1: Adapt new motor – Moons LV motor- 57BL60L2 (4/5)

Motor Parameters	Datasheet - 57BL60L2	Profiler Calculated	Unit
params.motor.P	8.0f		
params.motor.lq	225.0E-6f	223.0E-6f	H
params.motor.ld	225.0E-6f	206.0E-6f	H
params.motor.lam		7.01E-3f	Wb
params.motor.r	240.0E-3f	256.0E-3f	Ohm
params.motor.T_max	0.145f		Nm
params.motor.i_peak	10.80f		A
params.motor.i_cont	3.20f		A
params.motor.id_max	1.75f		A
params.motor.v_nom	24.0f		Vpk-IN
params.motor.w_nom.elec	4000		RPM
params.motor.w_max.elec	5000		RPM
Mechnanical Load			
params.mech.inertia	1.47E-5f	1.99E-5f	kg.m^2
params.mech.viscous		1.98E-5f	kg.m^2/sec
params.mech.friction		9.06E-3f	kg.m^2/sec^2
V/HZ Parameters			
params.ctrl.volt.v_min	0.15f (default)	9.15E-2f	Vpk
params.ctrl.volt.v_to_f_ratio	7.5E-3f(default)	6.92E-3f	Vpk/(Ra/sec-elec)

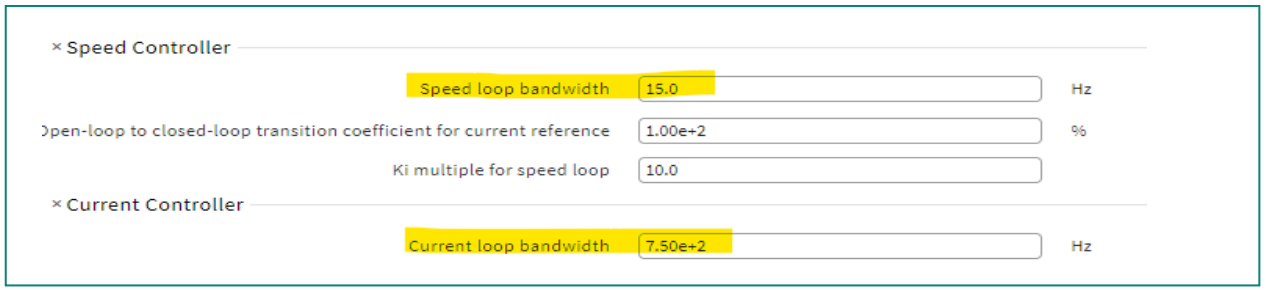


- ***motor.i_peak*** is the peak current rating of the motor. Set this value to 2.5 to 3x of ***motor.i_cont***
- ***motor.id_max*** is the maximum d-axis current rating of the motor that doesn't result in demagnetization of the permanent magnet. Set ***motor.id_max*** to half of ***motor.i_cont,*** if not specified in the motor datasheet.

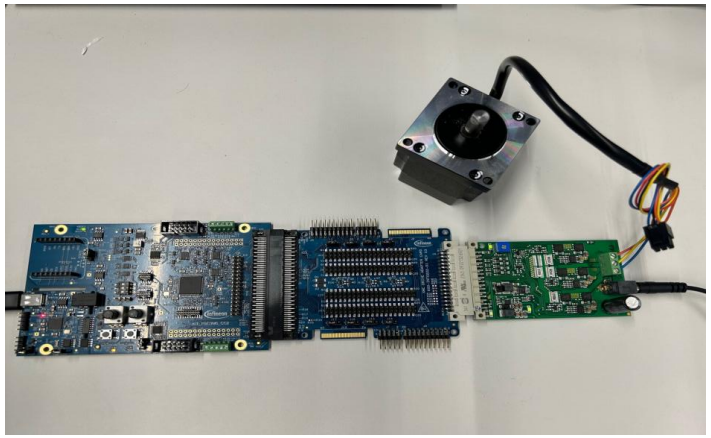
Note: The parameters are defined in \mtb_shared\\motor-ctrl-lib\release-v1.9.0\OperationalCode\Params.c file.

Example 1: Adapt new motor – Moons LV motor- 57BL60L2 (5/5)

- Test the motor with MC1 kit using the Motor Suite GUI
- Use the Oscilloscope to monitor the waveforms
- Tuning rule for motor in close loop :
  - Try to adjust the bandwidth of current and the speed controller



Note: The parameters are defined in \mtb_shared\\motor-ctrl-lib\release-v1.9.0\OperationalCode\Params.c file.



Example 2: Adapt new motor – Anaheim motor - BLY172S-24 (1/5)

- Using Anaheim motor – BLY172S
- When use different motor, following motor related parameters in Params.c file to be updated based new motor
  - Motor parameters
  - Load/mechanical parameters
  - Startup method parameters; for example, V/Hz startup parameter if V/Hz startup is used
- Use parameters from datasheet
  - Poles number – 8
  - Rated speed – 4000 RPM
  - Maximum speed – 5000 RPM
  - DC link voltage – 24 V
  - Rated continuous current – 3.5 A
  - Inductance ph-ph - 01.2 mH/2
    (calculate per phase value – Delta/Star)= 0.60 mH
  - Resistance ph-ph – 0.80 Ω/2
    (calculate per phase value – Delta/Star)= 0.40 Ω

Datasheet specs – **BLY172S**

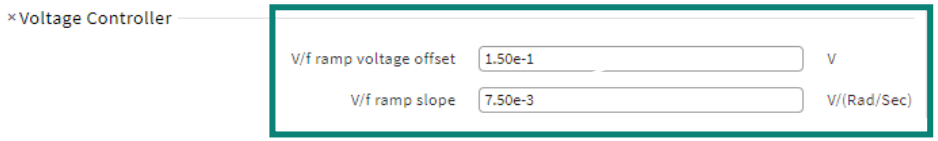
Model #	Rated Voltage (V)	Rated Speed (RPM)	Rated Power (W)	Peak Torque (oz-in)	Rated Current (A)	Line to Line Resistance (ohms)	Line to Line Inductance (mH)	Torque Constant (oz-in/A)	Back EMF Voltage (VkRPM)	Rotor Inertia (oz-in-sec²)	Weight (lbs)	"L" Length (in)
BLY171S-15V-8000	15	8000	26	14	2.2	0.35	0.35	1.98	1.14	0.00034	0.66	1.59
BLY171S-17V-8000	17	8000	42	21	3.6	0.20	0.26	1.98	1.14	0.00034	0.66	1.59
BLY171S-24V-4000	24	4000	26	27	1.8	1.50	2.10	4.96	2.45	0.00034	0.66	1.59
BLY172S-17V-9500	17	9500	70	30	6.4	0.09	0.09	1.56	0.90	0.00068	0.99	2.37
BLY172S-24V-2000	24	2000	41	63.72	3.6	1.60	2.70	8.78	5.00	0.00068	0.99	2.37
BLY172S-24V-4000	24	4000	53	54	3.5	0.80	1.20	5.81	3.35	0.00068	0.99	2.37
BLY173S-24V-4000	24	4000	77	79	4.9	0.46	0.70	5.38	3.10	0.00102	1.43	3.19

Note: The parameters are defined in \mtb_shared\\motor-ctrl-lib\release-v1.9.0\OperationalCode\Params.c file.

Example 2: Adapt new motor – Anaheim motor - BLY172S-24 (2/5)

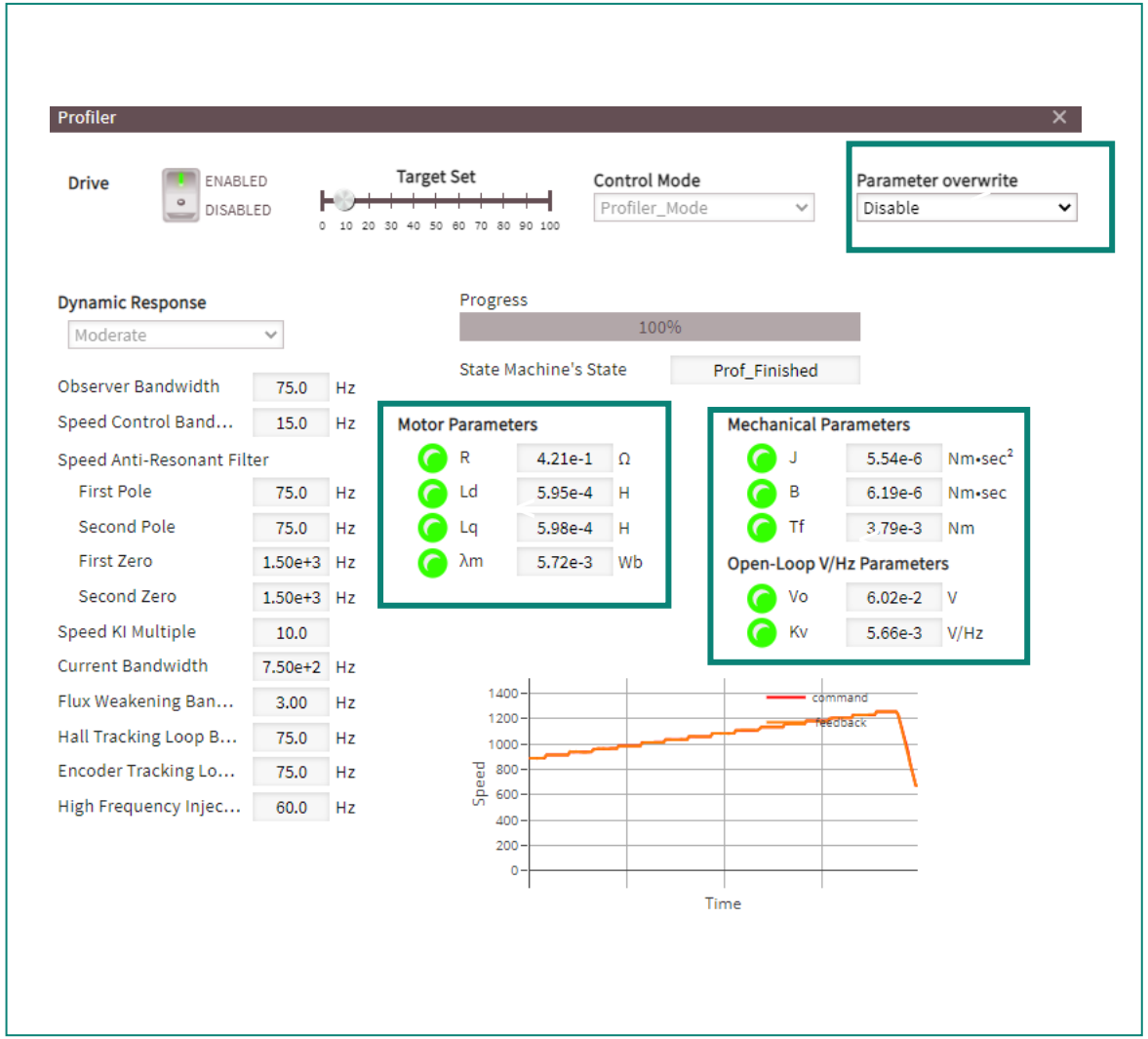
- Alternatively, run the profiler to calculate unknow parameters
  - Motor parameters
  - Load/mechanical parameters
  - Startup method parameters; for example, V/Hz startup parameter if V/Hz startup is used

- Note in case the profiler does not run properly for new motor , first adjust the open loop voltage controller
  - V/F ramp voltage offset
  - V/F ramp slope



Hint : Start with decreasing the V/F ramp slope if the motor need higher current in startup during V/F

Note: The parameters are defined in \mtb_shared\\motor-ctrl-lib\release-v1.9.0\OperationalCode\Params.c file.



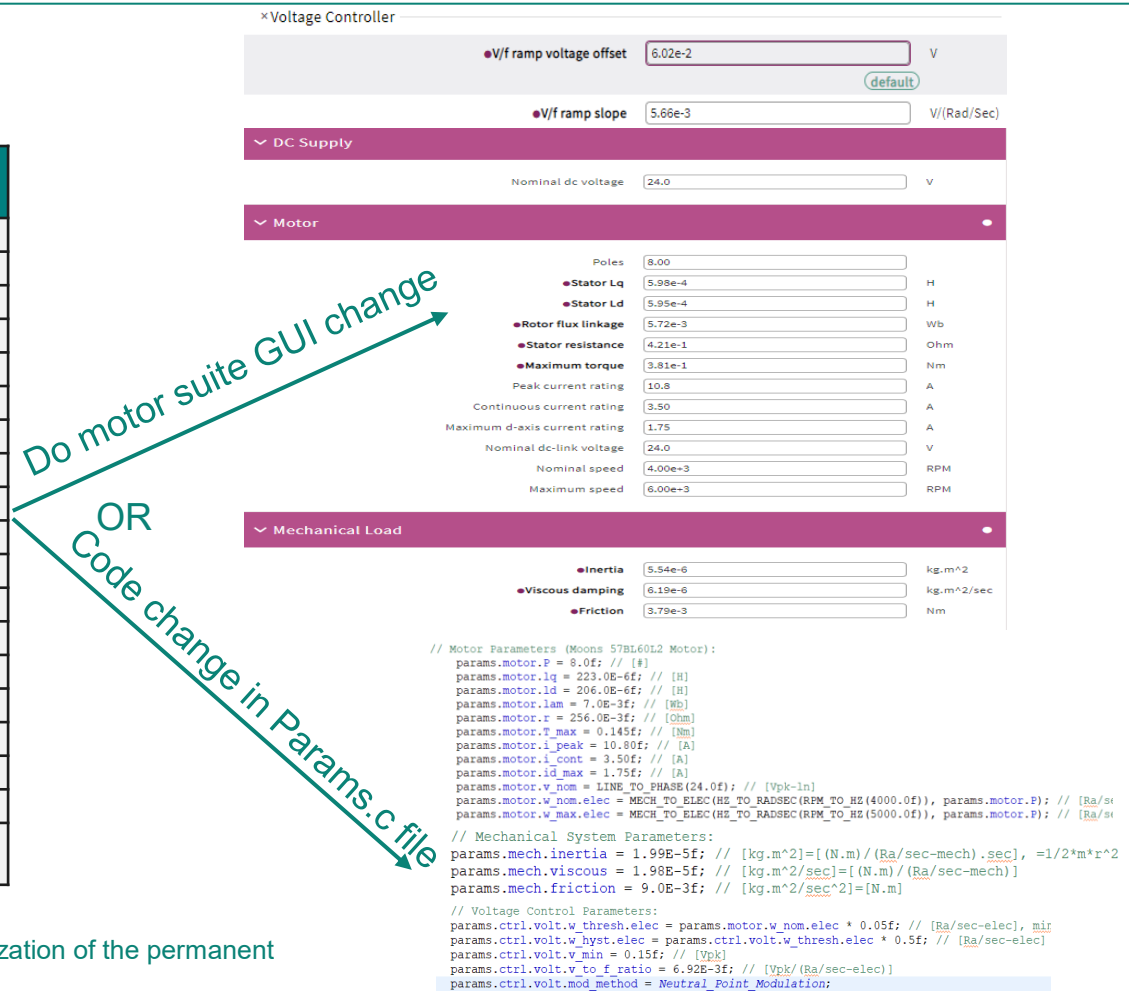
Example 2: Adapt new motor – Anaheim motor - BLY172S-24 (3/5)

- Fill the value in code or in GUI for the following:
  - Motor parameters
  - Load/ mechanical parameters
  - Startup method parameters; for example, V/Hz startup parameter if V/Hz startup is used

Note: The parameters are defined in \mtb_shared\\motor-ctrl-lib\release-v1.9.0\OperationalCode\Params.c file.

Example 2: Adapt new Motor – Anaheim motor - BLY172S-24 (4/5)

Motor Parameters	Datasheet – BLY17S-24	Profiler Calculated	Unit
params.motor.P	8.0f		
params.motor.lq	6.0E-4f	5.98E-4f	H
params.motor.ld	6.0E-4f	5.95E-4f	H
params.motor.lam		5.72E-3f	Wb
params.motor.r	4.0E-1f	4.21E-1f	Ohm
params.motor.T_max	0.3812f		Nm
params.motor.i_peak	10.80f		A
params.motor.i_cont	3.50f		A
params.motor.id_max	1.75f		A
params.motor.v_nom	24.0f		Vpk-IN
params.motor.w_nom.elec	4000		RPM
params.motor.w_max.elec	6000		RPM
Mechnanical Load			
params.mech.inertia	1.47E-5f	5.54E-6f	kg.m^2
params.mech.viscous		6.19E-6f	kg.m^2/sec
params.mech.friction		3.79E-3f	kg.m^2/sec^2
V/HZ Parameters			
params.ctrl.volt.v_min	0.15f (default)	6.02E-2f	Vpk
params.ctrl.volt.v_to_f_ratio	7.5E-3f(default)	5.66E-3f	Vpk/(Ra/sec-elec)

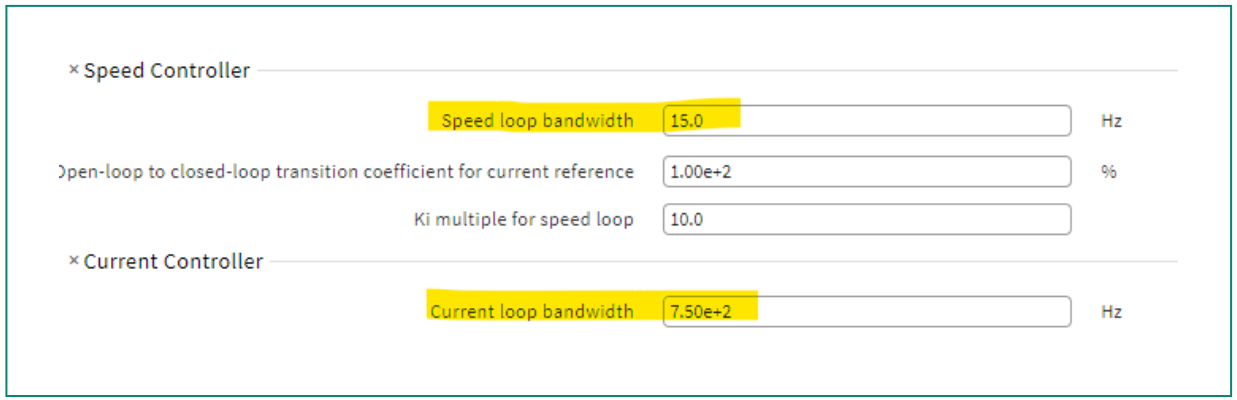


- ***motor.i_peak*** is the peak current rating of the motor. Set this value to 2.5 to 3x of ***motor.i_cont***
- ***motor.id_max*** is the maximum d-axis current rating of the motor that doesn't result in demagnetization of the permanent magnet.
  Set ***motor.id_max*** to half of ***motor.i_cont,*** if not specified in the motor datasheet

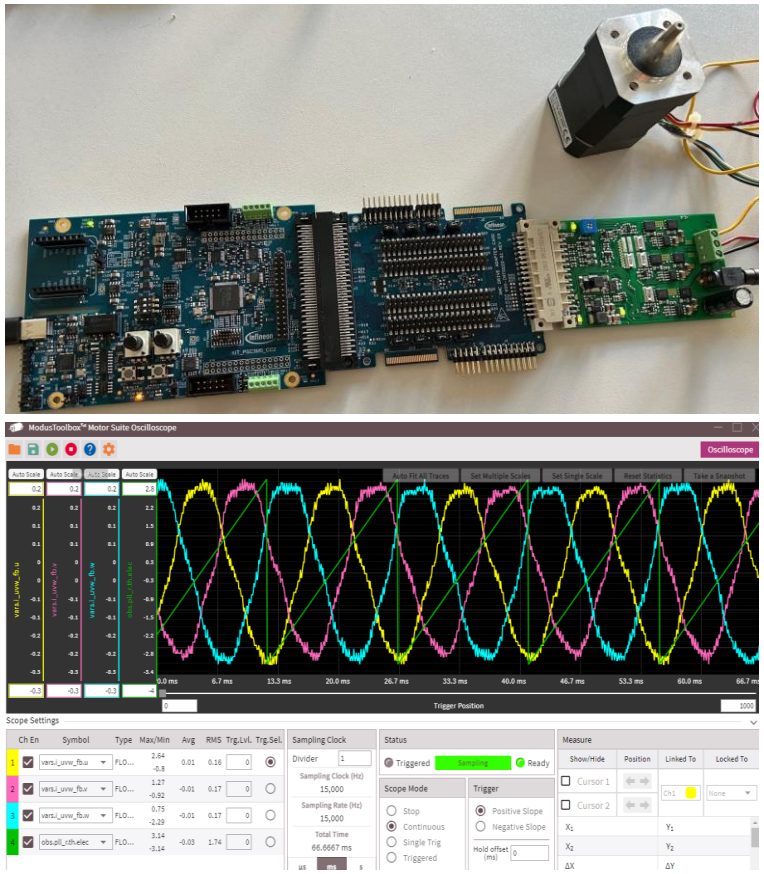
Note: The parameters are defined in \mtb_shared\\motor-ctrl-lib\release-v1.9.0\OperationalCode\Params.c file.

Example 2: Adapt new motor – Anaheim motor- BLY172S-24 (5/5)

- Test the motor with MC1 kit using the Motor Suite GUI
- Use the Oscilloscope to monitor the waveforms
- Tuning rule for motor in close loop :
  - Try to adjust the bandwidth of current and the speed controller



Note: The parameters are defined in \mtb_shared\\motor-ctrl-lib\release-v1.9.0\OperationalCode\Params.c file.



Revision history

Document revision	Date	Description of change
**	2026-02-19	Initial release.


Infineon