PMSM FOC motor control software using XMC™

XMC1000

About this document

Scope and purpose

This document describes the implementation of the PMSM FOC motor control software for 3-phase motor using the Infineon XMC1302 microcontroller.

Intended audience

This document is intended for customers who would like a highly configurable system for FOC control with sensorless feedback on XMC™ series microcontroller.

Referenced documents

[1] XMC1300 AB-Step Reference Manual, XMC1000 Family

Table of contents

Table of contents

1 Introduction

The intention of this software is to offer functionality to drive Permanent Magnet Synchronous Motors (PMSM) in sensorless or sensor modes. It contains all the common modules for both the modes as generic drives and provides high level of configurability and modularity to address different segments.

Field Oriented Control (FOC) motor control is a method to generate three phase sinusoidal signals which can easily be controlled in frequency and amplitude in order to minimize the current which means to maximize the efficiency. The basic idea is to transform three phase signals into two rotor-fix signals and vice versa.

Feedback on rotor position and rotor speed is required in the FOC motor control. The feedback can come from sensorless FOC or FOC with sensors. Sensorless FOC derives the rotor position and rotor speed based on motor modeling, voltage applied to the motor phases, and the current in the three motor phases. FOC with sensors gets the rotor position and rotor speed from rotor sensor(s), e.g. Hall sensors or encoder.

Feedback on the phase currents can be measured in the motor phase or in the leg shunt at the low side MOSFET. In this software, phase current sensing is expected from the leg shunt.

3-phase PMSM FOC motor control is widely used in applications like fans, pumps, compressors and e-bikes. Below the typical block diagram for the PMSM FOC motor control, where single shunt and three shunt low-side current sensing are supported.

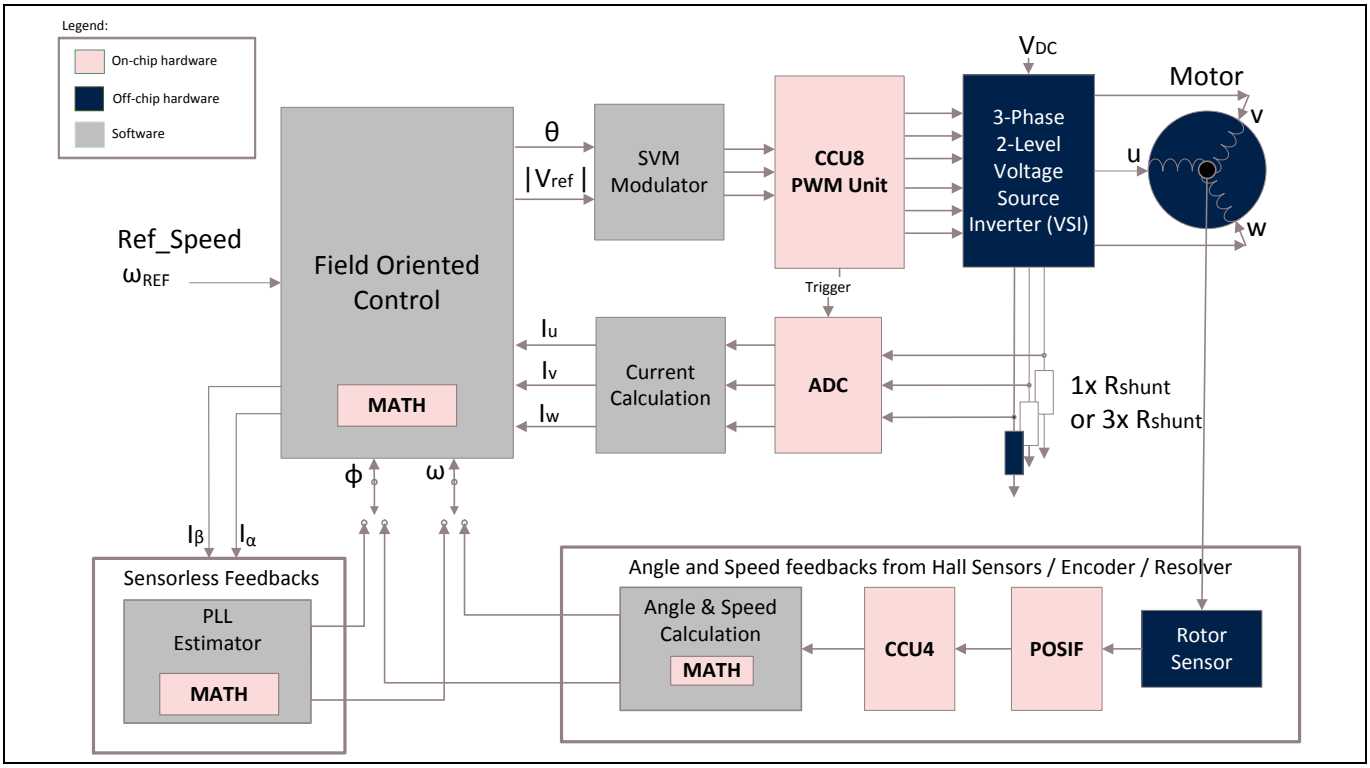


Figure 1 Block diagram of PMSM FOC motor control

1.1 Key features

Multiple Infineon innovations/unique features are included in the sensorless PMSM FOC software:

- Optimized FOC
  - No Inverse Park Transform
  - Lowest cost by eliminating external Op-Amp
- SVM with Pseudo Zero Vectors (PZV), for single shunt current sensing
- MET (Maximum Efficiency Tracking) for smooth transition from V/f open-loop to FOC closed-loop
- PLL Estimator, the sensorless feedback mechanism which requires only one motor parameter – stator inductance L, for rotor speed and position feedback

The key features supported are listed in the table below.

Table 1 Key features supported

SW Features List	
Math Control Blocks	Clarke Transformation
	Park Transformation
	Id and Iq current flux/torque PI controller
	Speed PI controller
	Cartesian to Polar Transformation
	Ramp Function
Control Scheme	Speed control
	Torque control
	Vq control
Space Vector Modulation	7-segment SVM
	Pseudo Zero Vector SVM; 4-segment SVM
Low Side Current Sensing	Three shunt: support over-modulation & 7-segment SVM
	DC link single shunt: Support PZV and 4-segment SVM, no over-modulation
Start-up Algorithm	Direct FOC start-up
Protection	Phase over-current protection
	DC link under voltage & over-voltage Protection
Device Feature	ADC on-chip gain for current sensing
	ADC synchronous conversion: motor phase current sensing
Control Feature	Motor control state machine
	DC-bus voltage clamping during fast braking
	Motor stop - brake
Rotor Speed and Angle Calculation	Sensorless PLL Estimator using HW CORDIC
Others	SW modularization
	PI anti-windup for Speed control

1.2 XMC™ resource usage

XMC1302 microcontroller is well suited for PMSM FOC motor control system. It has dedicated motor control peripherals, POSIF, MATH, CCU8, ADC and CCU4. In this PMSM FOC motor control software, the hardware peripherals used are listed in the table below.

Default resource usage is for Infineon XMC1000 Motor Control Application Kit (order number: KIT_XMC1X_AK_MOTOR_001).

Table 2 XMC™ Peripherals used for sensorless 3-phase PMSM FOC

XMC™ Peripherals	Usage	Default Resource Allocation
CCU80	PWM Generation for Phase U	CCU80 slice 0
	PWM Generation for Phase V	CCU80 slice 1
	PWM Generation for Phase W	CCU80 slice 2
	Timer for ADC Trigger	CCU80 slice 3
VADC	Phase U Current sensing	G0 channel 4[1] G1 channel 3[1]
	Phase V Current sensing	G0 channel 3[1] G1 channel 2[1]
	Phase W Current sensing	G0 channel 2[1] G1 channel 4[1]
	Alias Channels for three shunts synchronous conversion	G0 channel 0 G0 channel 1 G1 channel 0 G1 channel 1
	DC Link Voltage sensing	G1 channel 5
	DC Link average Current sensing	G1 channel 6
	DC Link Current sensing for single shunt	G1 channel 1
	Potentiometer for speed change	G1 channel 7
MATH	CORDIC	Enabled
Nested Vectored Interrupt Controller (NVIC)	PWM Period Match Interrupt	CCU80.SR0
	CTrap Interrupt	CCU80.SR1
	ADC Interrupt for single shunt sensing	VADC0.G1SR1[2]

[1] The same input pin must connect to both the ADC group channels to perform synchronous sampling, refer to chapter 3.1.2.1 for details.

[2] If DC link current sensing is using VADC Group0 channels, then VDC0.G0SR1 interrupt node is used.

1.3 XMC™ hardware modules interconnectivity

XMC1302 has comprehensive hardware interconnectivity. The figure below shows the interconnections between XMC1302 hardware peripherals modules. The CCU8 slice timers are started synchronously via the sync start signal from the SCU (System Control Unit). The VADC conversion is triggered by the CCU8 Slice 3 compare match status signal.

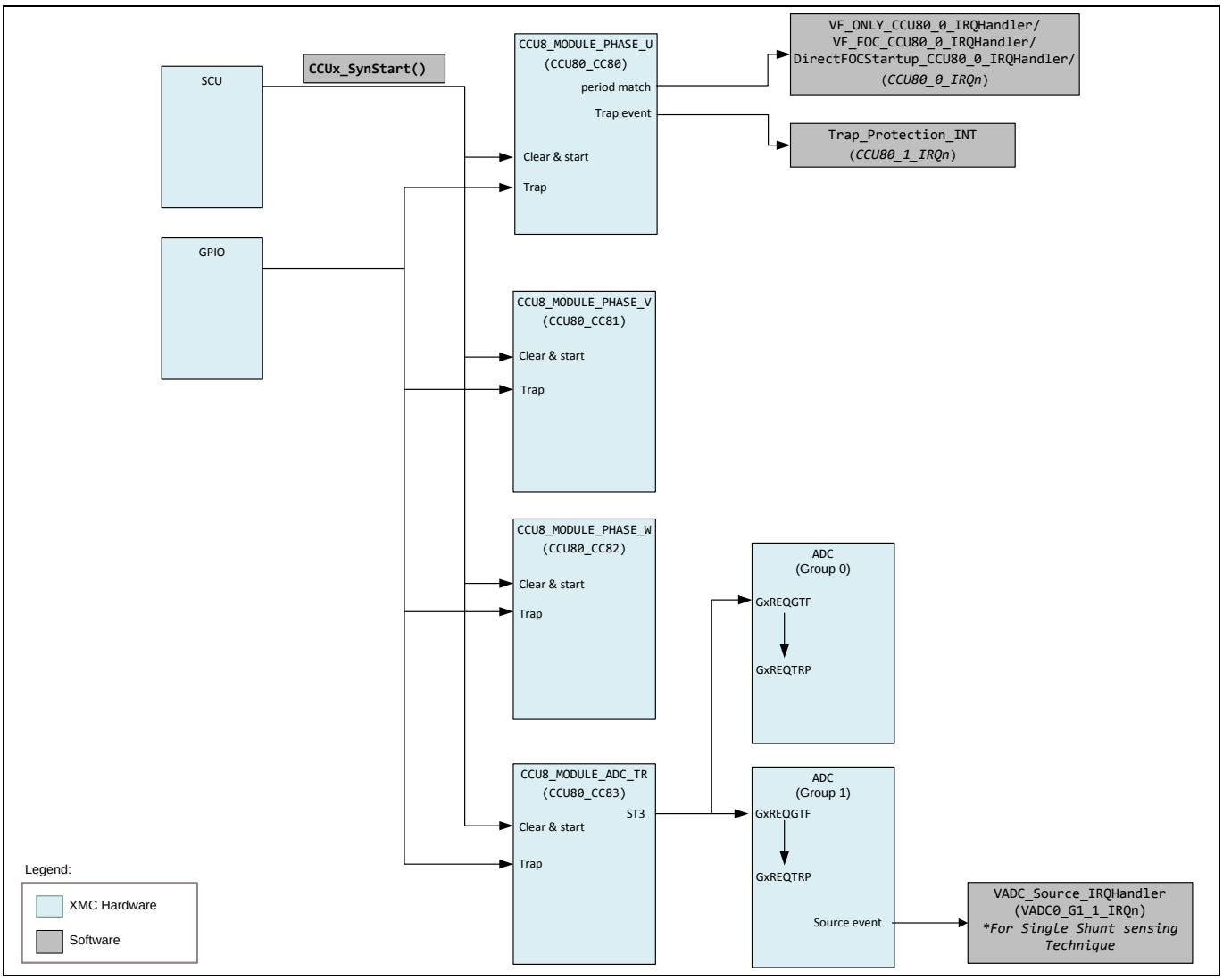


Figure 2 XMC1302 hardware interconnection

1.4 Memory usage

In XMC1302, access to the Static Random Access Memory (SRAM) requires no wait state. The major parts of the software are executed from the SRAM. The Interrupt Service Routines, all mathematical blocks of FOC algorithm, SVM, and motor phase current sensing and calculation are executed in the SRAM. This improves the performance as the execution time to run the FOC algorithm is reduced about 30%.

Please refer to chapter 9 for the list of APIs running in SRAM.

Breakdown of the memory usage and CPU time-utilization are provided in table based on the default settings for Infineon XMC1000 Motor Control Application Kit:

- Control Scheme: Open-Loop to FOC Closed Loop Speed Control
- Current Sensing Technique: Three shunt synchronous ADC conversion

Table 3 CPU utilization and memory usage for three shunt current sensing

PWM Frequency	20 kHz – Interrupt Service Routine runs every 50 µsec
DAVE™ 4 GCC Compiler Optimization level	Optimized most (-O3)
CPU Utilization	34.5 µsec (69%)
Flash Code Size (bytes)	11536
SRAM Code Size (bytes)	2800
SRAM Data Size (bytes)	304

For the single shunt current sensing technique, the memory usage and CPU utilization are slightly more than three shunt sensing technique due to additional ADC interrupts needed to read the phase currents readings.

- Control Scheme: Open-Loop to FOC Closed Loop Speed Control
- Current Sensing Technique: Single shunt

Table 4 CPU utilization and memory usage for single shunt current sensing

PWM Frequency	20 kHz – Interrupt Service Routine runs every 50 µsec
DAVE™ 4 GCC Compiler Optimization level	Optimized most (-O3)
CPU Utilization	37.5 µsec (75%)
Flash Code Size (bytes)	11676
SRAM Code Size (bytes)	3384
SRAM Data Size (bytes)	244

1.5 Software overview

PMSM FOC motor control software is developed based on well-defined layered approach. The layered architecture is designed in such a way as to separate the modules into groups. This allows different modules in a given layer to be easily replaced without affecting the performance in other modules and the structure of the complete system.

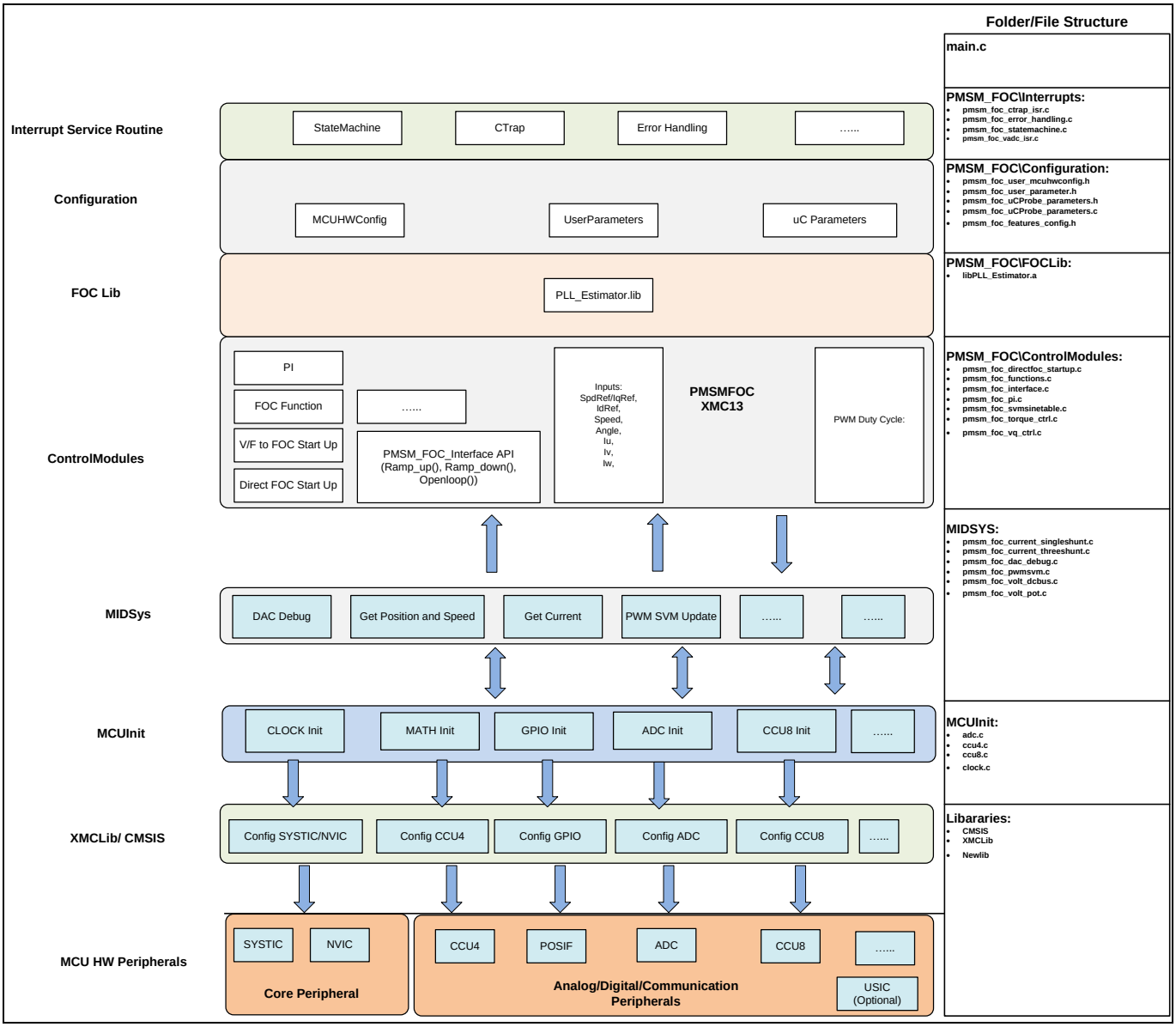


Figure 3 PMSM FOC software overview

Interrupts

This layer consists of CCU8 trap interrupt handling function, CCU8 period match ISR function, and VADC ISR function for single shunt current sensing.

Configuration

Microcontroller hardware resources management, pin mapping and inverter hardware configuration are configurable in the header file named pmsm_foc_mcuhwconfig.h; motor parameters, control scheme, start-up parameters, PI parameters, feedback current sensing measurement and scaling are configurable in

pmsm_foc_user_parameter.h. All the macro constants are recalculated with mathematical equation and scaled to fixed point.

FOCLib

Infineon patented IP, PLL Estimator, is provided as compiled .a library file.

Control Modules

This layer consists of FOC SW control modules. This includes Clarke Transform, Park Transform, Cartesian to Polar, current reconstruction, PI controller, Open Loop, Ramping, etc. All routines mentioned are called from the CCU80 period match Interrupt Service Routines.

Middle System (MIDSys)

This layer provides routines for PWM generation, ADC measurements, angle and speed information to FOC control module layer. The main purpose of this layer is to give flexibility to add/remove sensor feedback module into the FOC software. For example Hall sensors, user can add in files in this layer to provide position and feedback from Hall sensors, without making huge changes to the layers on top.

MCU Initialization (MCUInit)

This layer contains the initialization of the all MCU peripherals. It contains XMCLib data structure initialization and peripherals initialization functions. This layer closely interacts with XMCLib and MIDSys layer to configure each peripheral.

XMCLib & CMSIS (Libraries)

This layer is the hardware abstraction layer to the MCU peripherals.

1.6 Scope of use for PMSM FOC software

In this application note, the current software version used is PMSM FOC SW v1.0.0. At time of release of this example software, the following limitations in usage apply:

- XMC4000 devices are not supported in this software version
- Only single motor drive is supported. Dual motor control support is not available
- Position and speed feedback from Hall sensors/encoders are not supported
- This software is developed in DAVE™ version 4. It is not tested on other IDE (Integrated Development Environment) platforms

1.7 Abbreviations and acronyms

Table 5 Abbreviations and acronyms used in this document

Abbreviation/acronym	Definition
API	Application Programming Interface
BOM	Bill of Material
CCU8	Capture Compare Unit 8
CPU	Central Processing Unit
FOC	Field Oriented Control
GPIO	General Purpose Input / Output

Abbreviation/acronym	Definition
IP	Intellectual Property
ISR	Interrupt Service Routine
LLD	Low Level Driver
MCU	Microcontroller Unit
MET	Maximum Efficiency Tracking
PI	Proportional Integral Controller
PLL	Phase Locked Loop
PMSM	Permanent Magnet Synchronous Motors
PWM	Pulse Width Modulation
PZV	Pseudo Zero Vector
SRAM	Static Random Access Memory
SVM	Space Vector Modulation
UART	Universal Asynchronous Receiver Transmitter
VADC	Versatile Analog-to-Digital Converter

2 PMSM FOC sensorless software components

The intention of this PMSM FOC motor control software is to offer functionality to drive the PMSM motors with sensors or sensorless modules. The current version supports sensorless modules.

The PMSM FOC software provides a high level of configurability and modularity to address different motor control applications.
Five types of control scheme are supported:

1. Speed Control Transition FOC Startup
2. Speed Control Direct FOC Startup
3. Torque Control Direct FOC Startup
4. Vq Control Direct FOC Startup
5. Open-Loop Voltage Control

The major components of the PMSM FOC software are shown in the following diagram. Each of the modules is described and referenced.

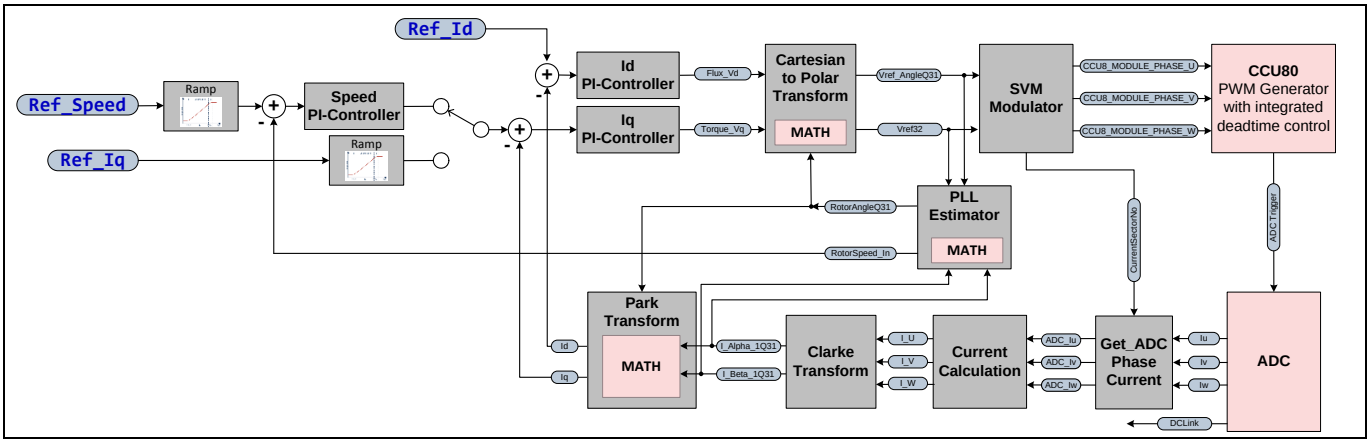


Figure 4 PMSM FOC block diagram

2.1 Motor start / speed change / motor stop operations control

The motor operations like motor start, motor stop and speed change can be controlled by applying an analog voltage signal through an ADC port. In the Infineon XMC1000 Motor Control Application Kit, this ADC pin is connected to a potentiometer.

The relationship between the ADC data and the motor target speed for constant speed control scheme is shown in the figure below.

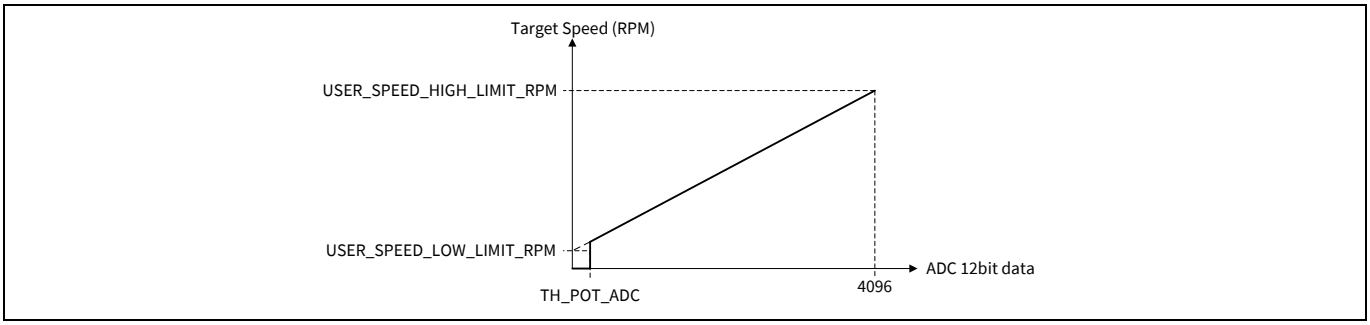


Figure 5 Potentiometer ADC value vs speed in speed control scheme

Two input thresholds are used to define the relationship between the input voltage (ADC data) and the target speed of the motor.

1. TH_POT_ADC: Threshold for motor stop. If ADC data is below this threshold, the motor will stop. Above this threshold, the motor will start with USER_SPEED_LOW_LIMIT_RPM.
2. Maximum value of 12-bit ADC, 4096: Maximum target speed of the motor, USER_SPEED_HIGH_LIMIT_RPM.

The other options to control the motor operations are via the UART communication or µC/Probe GUI, refer to chapter 10. The selection of the options is done in the macro UART_ENABLE which is defined in the header file pmsm_foc_user_mcuhwconfig.h, refer to chapter 7.3.4.

2.2 Ramp generator

PMSM FOC motor control software provides two types of ramp functions:

- Linear curve
- S-curve

It ramps input parameter from initial value to end value. Ramp generator input is connected to user set value or analog input, depending on user configuration. The ramp up rate in the linear region is defined as USER_SPEED_RAMPUP_RPM_PER_S in the user configuration file, pmsm_foc_user_parameter.h (refer to chapter 7.1 item USER_SPEED_RAMPUP_RPM_PER_S). The ramp generator function is called every PWM frequency cycle.

In the S-curve ramp generator function, the initial ramp up rate is half of the defined ramp up rate. The ramp rate is slowly increased to the defined value. This generates the first s-curve. The second S-curve starts when the speed is SPEED_TH_2ND_S from the Ref_Speed. The constant SPEED_TH_2ND_S is defined in the pmsm_foc_interface.c file. The S-curve ramp generator is only used in the speed control.

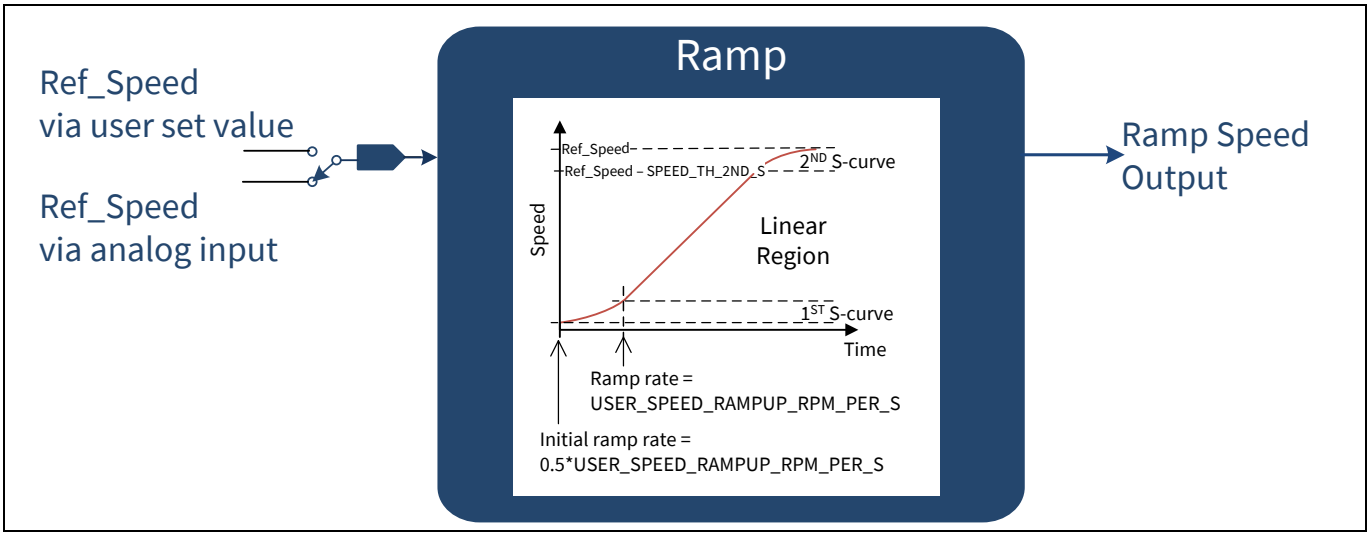


Figure 6 S-curve ramp generator

In both ramp generator functions, the DC link voltage is monitored during ramp down operation. It stops the speed/torque ramp down if the DC link voltage is over the user configured voltage limit. This is to avoid over voltage condition during the fast braking. This voltage limit, VDC_MAX_LIMIT (refer to chapter 7.2.1, item VDC_MAX_LIMIT), is defined in the user configuration file, pmsm_foc_user_parameter.h.

The ramp output is the reference signal to the control scheme. For the linear ramp generator, depending on the control scheme selected, the ramp output can be speed, torque current or torque Vq.

2.3 Control schemes

In this software block, the control schemes for the 3-phase PMSM FOC motor can be:

- Open loop voltage control
- Speed control
- Torque control
- Vq control

2.3.1 Open loop voltage control

In an open loop voltage control, a reference voltage (Vref) is used to cause the power inverter to generate a given voltage at the motor. The mechanical load influences the speed and the current of the PMSM motor.

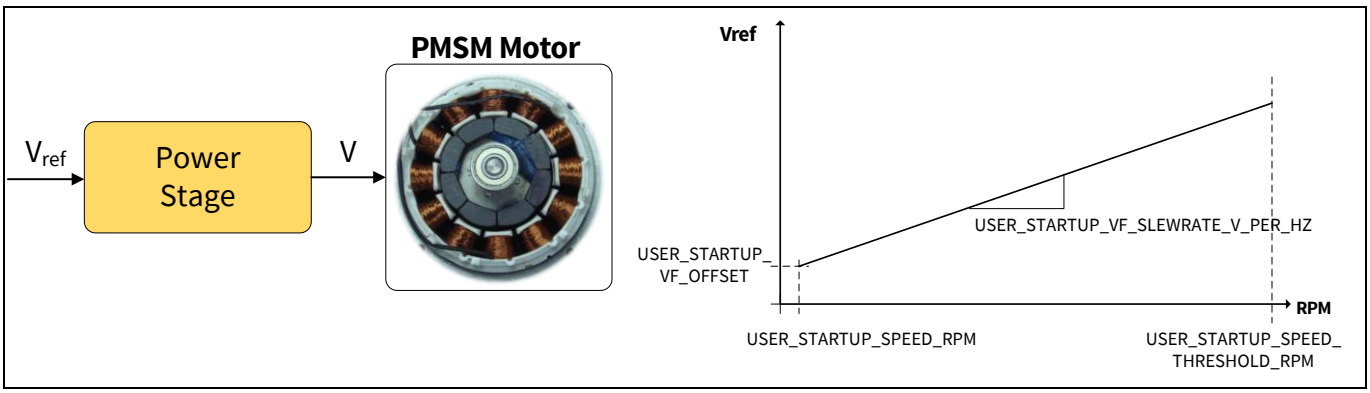


Figure 7 Open loop voltage control

2.3.2 Speed control

A speed control scheme is a closed loop control. For the speed control scheme, a cascaded speed and currents control structure is used. This is due to change response requirement for speed control loop is much slower than the one for current loop.

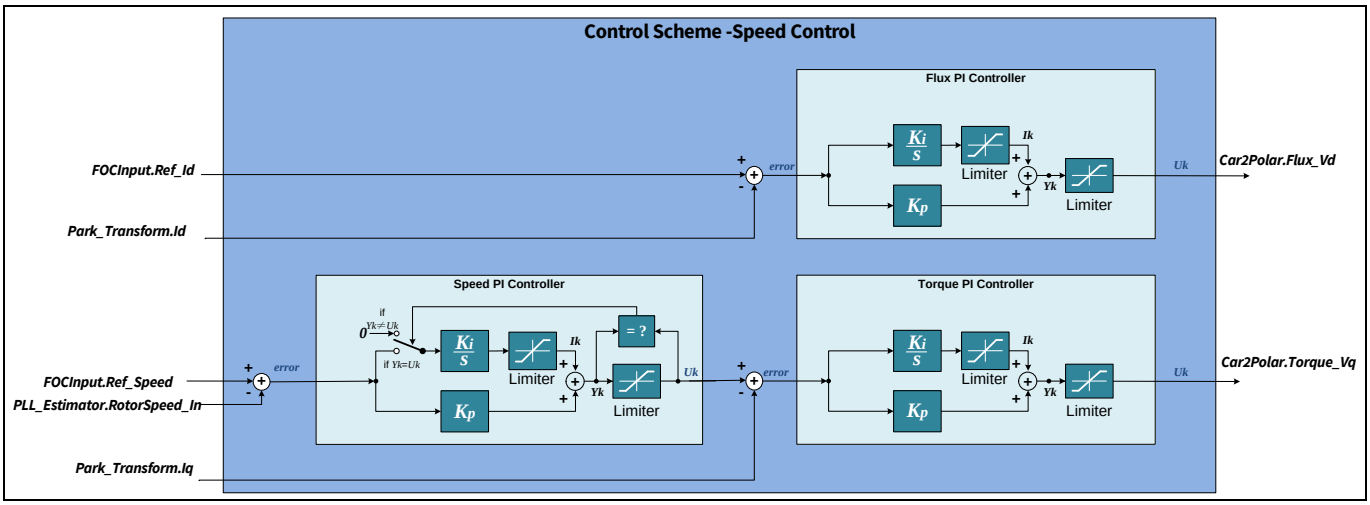


Figure 8 Speed control

Direct FOC startup and transition startup (open loop to closed loop) modes are supported in speed control. The speed PI controller supports integral anti-windup. The integral output is held stable when either PI output or integral output reaches its limit. The output of the speed PI is used as the reference for the torque PI controller.

2.3.2.1 Transition mode – 3 steps startup with Maximum Efficiency Tracker (MET)

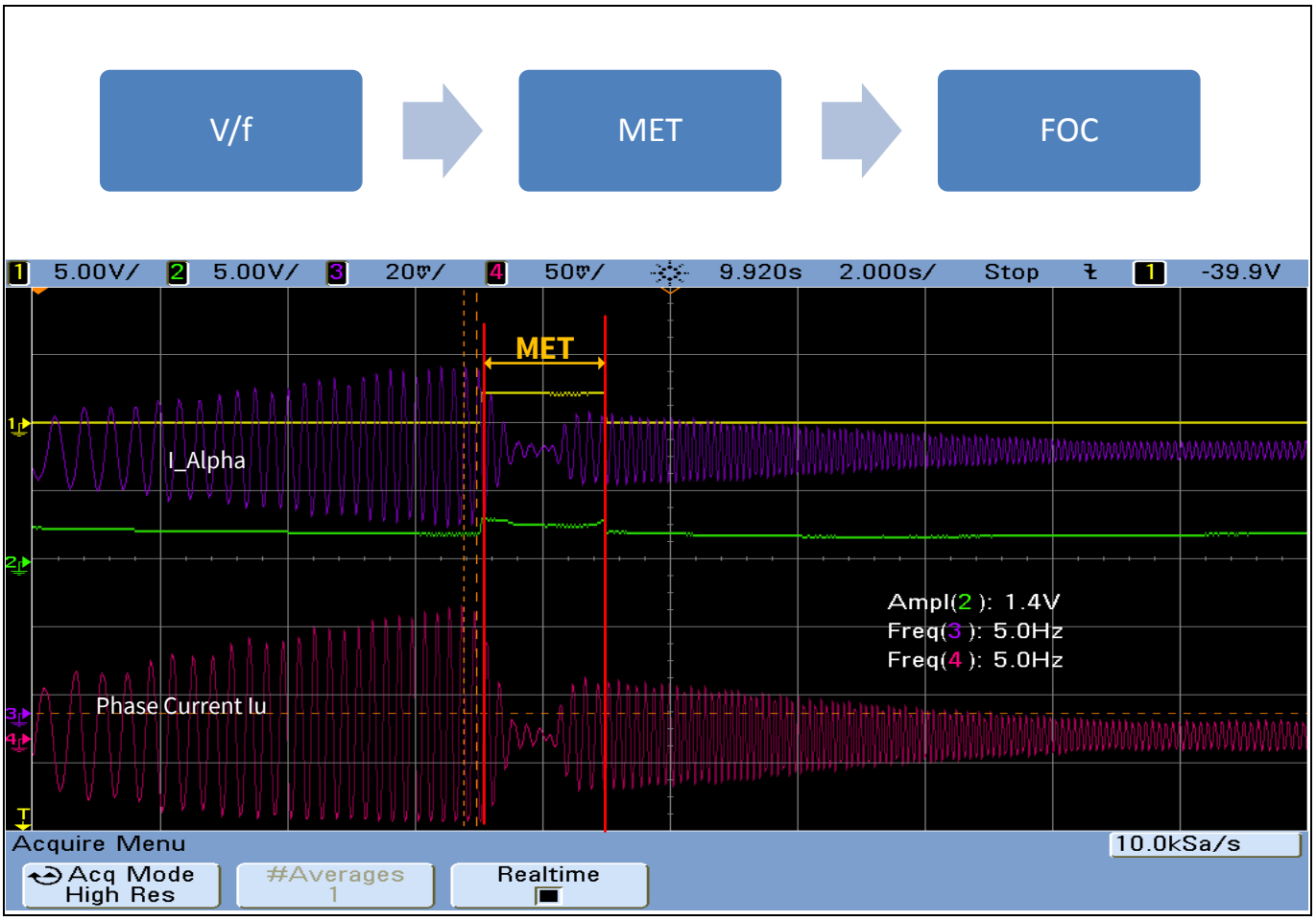


Figure 9 3 steps motor startup mechanism

Below the three steps:

1. The motor starts in V/f open loop control state and ramp up to a user defined startup speed.
2. Sensorless MET closed-loop control state takes over.
   This state is added to ensure the stator flux is perpendicular to the rotor flux in a smooth and controlled way.
3. The state machine switches to FOC_CLOSED_LOOP state and ramp up the motor speed to user defined target speed.

Advantages:

- High energy efficiency MET and FOC closed-loop
- Smooth transitions for all the three steps
- V/f open-loop -> MET closed loop at low motor speed, therefore low startup power

2.3.3 Torque control direct startup

PMSM FOC motor control software provides direct startup torque control. The control loop consists of the d-axis (Flux) and q-axis (Torque) PI controllers. The motor torque is maintained at torque reference value (Iq_ref). Any change in the load will cause the speed of the motor to change but the torque remains constant.

This control scheme is useful in application where direct current control is important, e.g. e-bike, battery-operated devices.

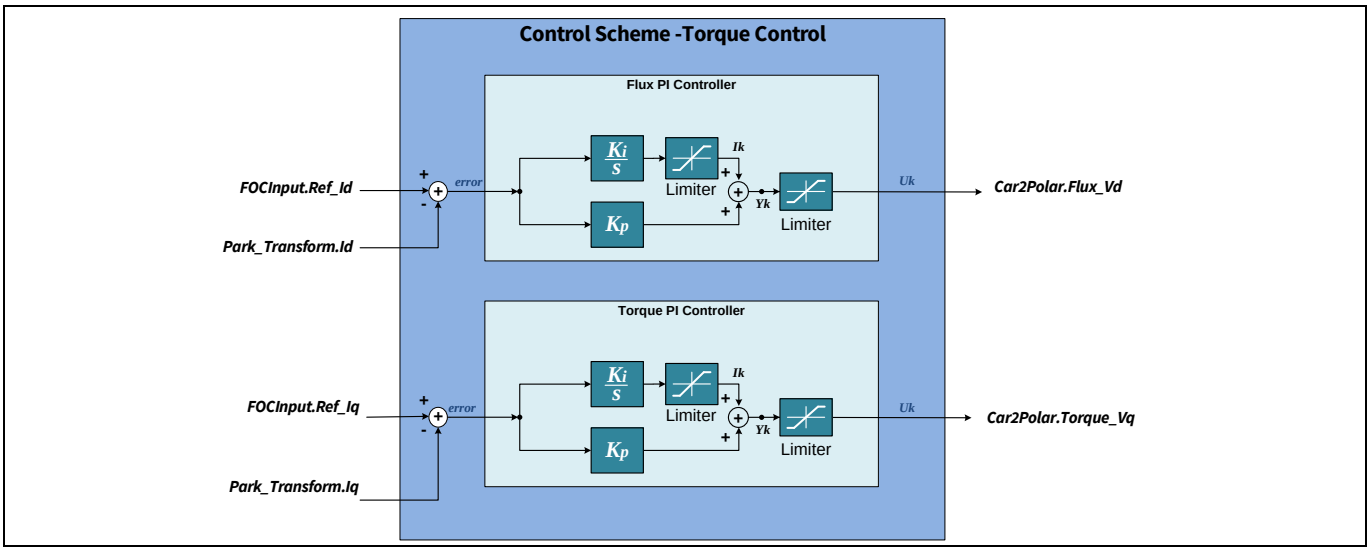


Figure 10 Torque control scheme

2.3.4 Vq control direct startup

The Vq control is used when fast response is required and varying speed is not a concern. The speed PI control loop and torque PI control loop are disabled.

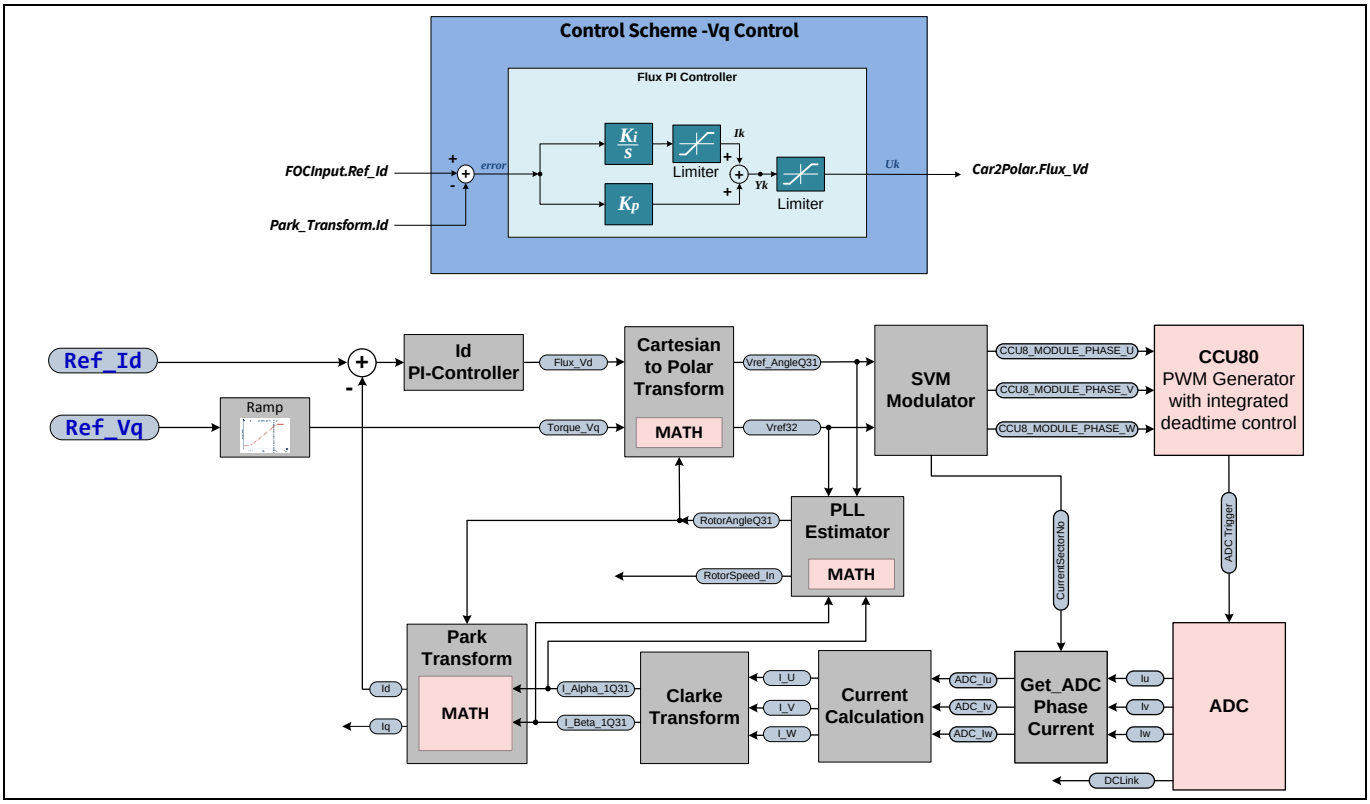


Figure 11 Vq control scheme

2.4 Cartesian to Polar transform

Using the outputs from the torque and flux PI controllers, the Cartesian to Polar transform is calculated with the hardware CORDIC coprocessor in circular vectoring mode.

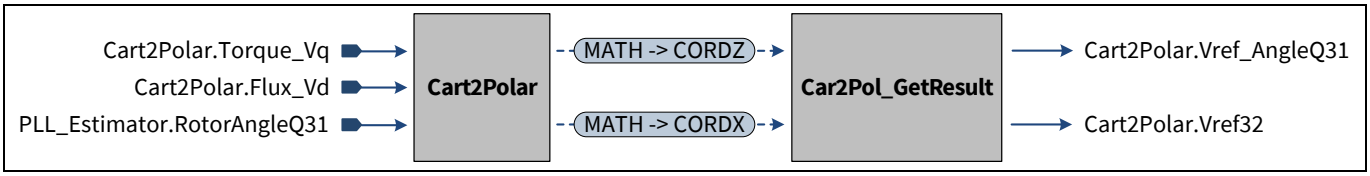


Figure 12 Cartesian to Polar transform

Table 6 XMC1302 CORDIC settings for Cartesian to Polar transform

Parameters	Settings
CORDIC Control Mode	Circular Vectoring Mode
K	≈ 1.646760258121
Magnitude Prescaler, MPS	2
CORDX	Cart2Polar.Flux_Vd
CORDY	Cart2Polar.Torque_Vq
CORDZ	PLL_Estimator.RotorAngleQ31

According equations:

$$CORDX = K * \left|V_{ref32}\right| / MPS = K * \sqrt{V_{Flux\_Vd}{}^2 + V_{Torque\_Vq}{}^2} / MPS,$$

$$\left|V_{ref32}\right| = CORDX * MPS / K = CORDX * 1.2145$$

$$V_{ref\_AngleQ31}, \theta = CORDZ = RotorAngleQ31 + \arctan\left(\frac{V_{Torque\_Vq}}{V_{Fluz\_Vd}}\right)$$

To improve the execution on XMC13, the function is divided into two for parallel processing, start CORDIC function and read CORDIC result function. While the CORDIC coprocessor is doing the calculation, the CPU can execute other software functions, e.g. the PI controller, and read the CORDIC result later. The results from the Cartesian to Polar transform are fed into the SVM module.

2.5 Space Vector Modulation (SVM)

Space Vector Modulation (SVM) transforms the stator voltage vectors into pulse width modulation (PWM) signals (compare match values).

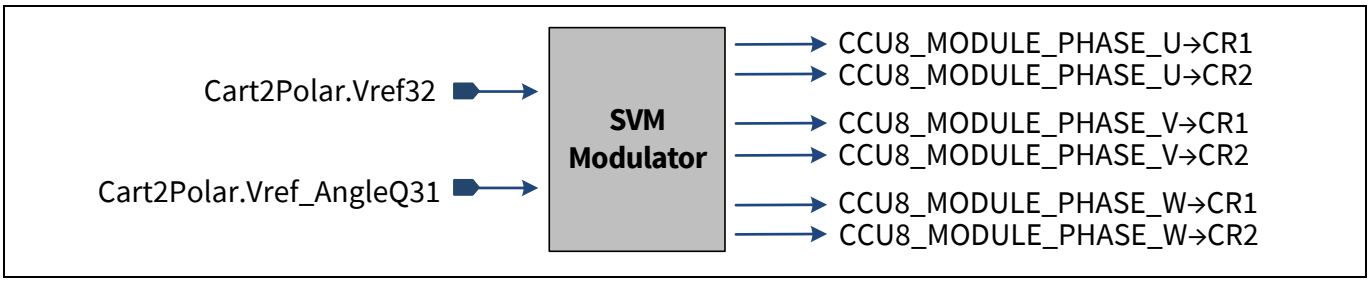


Figure 13 PWM SVM function

PMSM FOC motor control software supports different modes of SVM as below:

- 7-segment SVM
- SVM with Pseudo Zero Vector
- 4-segment SVM
- Over-modulation SVM

Each CCU80 timer slice controls an inverter phase with complementary outputs. Dead time is inserted to prevent DC link voltage short-circuit. This dead time value is configured by the user in the header file pmsm_foc_user_parameter.h.

The timer counting scheme used in the CCU80 is asymmetrical edge aligned mode. This is to have the same CCU80 settings for 7-segment SVM, 4-segments SVM and Pseudo Zero Vector PWM. With the asymmetric mode, there is more flexibility for sampling shunt currents via the ADC.

The default initial settings of the CCU80 module are for the Infineon XMC1000 Motor Control Application Kit, KIT_XMC1X_AK_MOTOR_001.

Table 7 CCU80 default initial settings for 3-phase SVM generation

Parameters	Settings
Timer Counting Mode	Edge aligned mode
Shadow Transfer on Clear	Enabled
Prescaler mode	Normal

Parameters	Settings
Passive level	Low
Asymmetric PWM	Enable
Output selector for CCU80.OUTy0	Connected to inverted CC8yST1
Output selector for CCU80.OUTy1	Connected to CC8yST1
Dead time clock control	Time slice clock frequency, f_{tclk}
Dead time value	USER_DEAD_TIME_US*USER_PCLK_FREQ_MHZ (750 nsec)
Phase U Slice Period Match Interrupt Event	Enabled
Trap Interrupt Event	Enabled

2.5.1 7-segment SVM

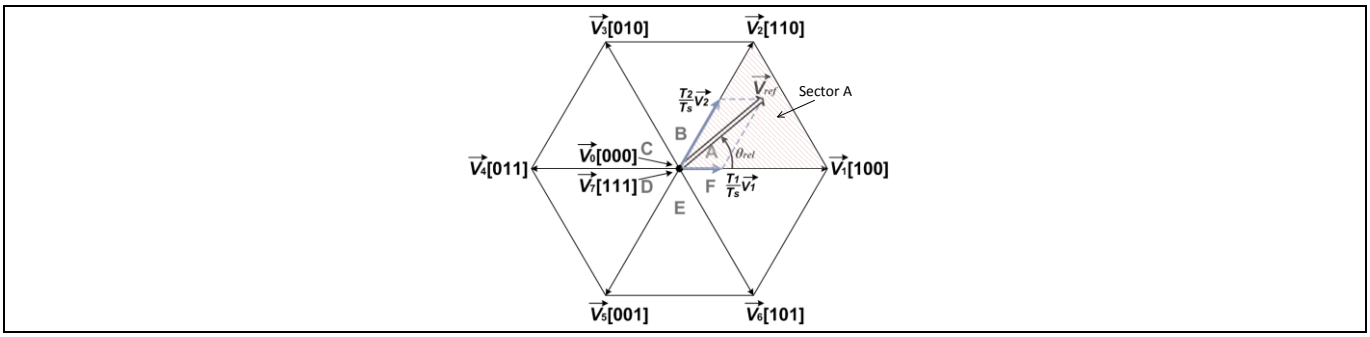


Figure 14 7-segment SVM

Using the voltage space vector in sector A as an example; the following equations are used to calculate PWM on-time of the SVM.

$$\vec{V}_{ref} = \frac{T_0}{T_S}\vec{V}_0 + \frac{T_1}{T_S}\vec{V}_1 + \frac{T_2}{T_S}\vec{V}_2$$

$$T_S = T_0 + T_1 + T_2$$

$$T_1 = \frac{\sqrt{3}|V_{ref}|T_S}{V_{DC}} sin\left(\frac{\pi}{3} - \theta_{rel}\right)$$

$$T_2 = \frac{\sqrt{3}|V_{ref}|T_S}{V_{DC}} sin(\theta_{rel})$$

$$T_0 = T_S - T_1 - T_2$$

Legend:

T_S Sampling period

$\vec{V}_0$ Zero vector

$\vec{V}_1$ $\vec{V}_2$ Active vectors

T_0 Time of zero vector(s) is applied. The zero vector(s) is $\vec{V}_0[000]$,$\vec{V}_7[111]$ or both

T_1 Time of active vector $\vec{V}_1$ is applied within one sampling period

T_2 Time of active vector $\vec{V}_2$ is applied within one sampling period

T_{DC} Inverter DC link voltage

θ_{rel} Relative angle between Vref and V1 ($0 \leq \theta_{rel} \leq \frac{\pi}{3}$)

For example, in SVM sector A, the PWM on time for phase U is PWM period minus $T_0/2$, phase V is PWM period minus ($T_0/2 + T_1$) and phase W is $T_0/2$.

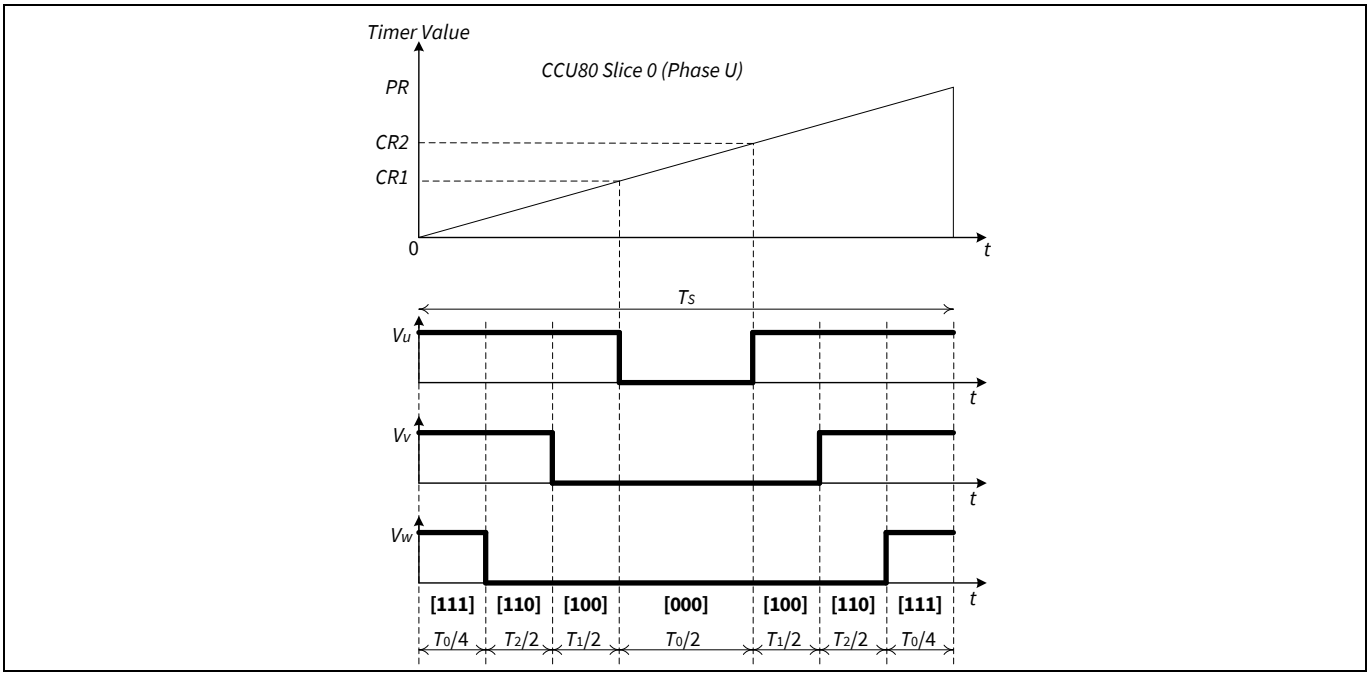


Figure 15 7-segment SVM timing diagram in SVM sector A

2.5.2 Pseudo Zero Vector (PZV)

In single shunt current reconstruction, the current through one of the phase can be sensed across the shunt resistor during each active vector. However under certain conditions, e.g. at sector crossovers or when the length of the vector is low, the duration of one or both active vectors ($T_1 < T_{min}$ or $T_2 < T_{min}$) is too small to guarantee reliable sampling of the phase currents. These conditions are shaded in the space vector diagram as shown in the Figure 16.

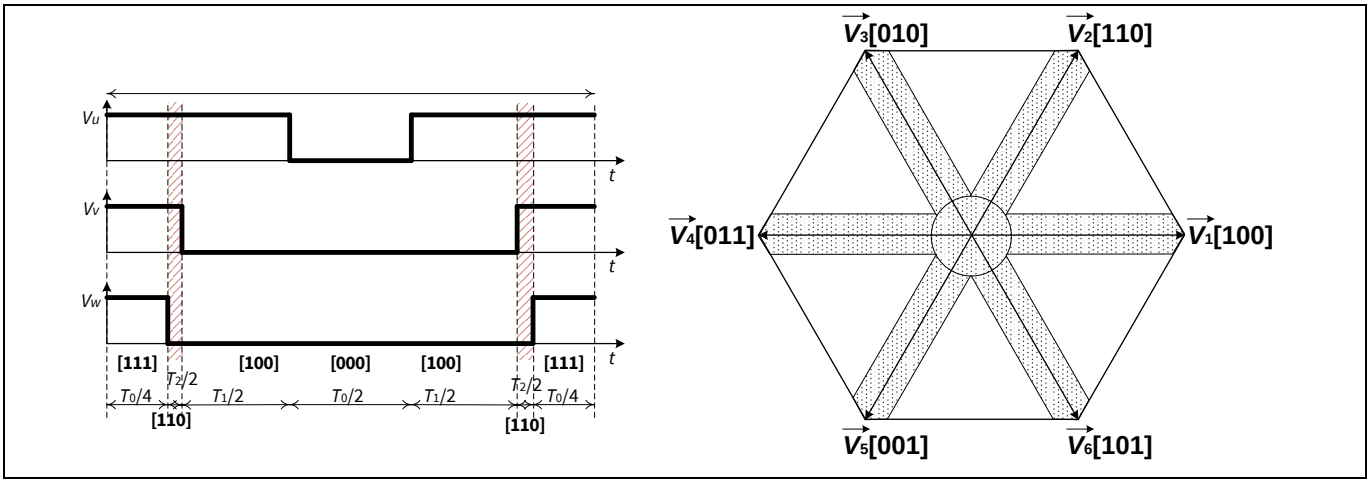
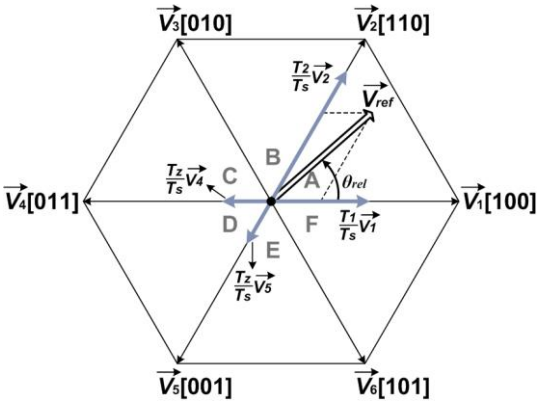


Figure 16 7-segment SVM - single shunt sensing

In order to resolve this, Pseudo Zero Vector is used in these conditions for single shunt current sensing. The Pseudo Zero Vector time, T_Z is adjusted to ensure adequate ADC sampling time for the phase currents sensing.



$$T_1 = \frac{\sqrt{3}\cdot T_s \cdot |V_{ref}|}{V_{DC}} \sin\left(\frac{\pi}{3} - \theta_{rel}\right) + T_{\min} \qquad (T_Z = T_{\min})$$

$$T_2 = \frac{\sqrt{3}\cdot T_s \cdot |V_{ref}|}{V_{DC}} \sin(\theta_{rel}) + T_{\min}$$

$$T_1 + T_2 = \frac{\sqrt{3}\cdot T_s \cdot |V_{ref}|}{V_{DC}} \cdot \sin\left(\frac{\pi}{3} + \theta_{rel}\right) + 2T_{\min}$$

$$T_0 = T_s - T_1 - T_2 - 2T_{\min} = T_s - \frac{\sqrt{3}\cdot T_s \cdot |V_{ref}|}{V_{DC}} \cdot \sin\left(\frac{\pi}{3} + \theta_{rel}\right) - 4T_{\min}$$

Note: $T_{\min}$ is PWM dead time + driver delay + ADC sampling time

Figure 17 Pseudo Zero Vector

The figure above shows the equations to calculate the T_1 and T_2 timing. The Figure 18 shows the timing diagram and the SVM diagram of the Pseudo Zero Vector.

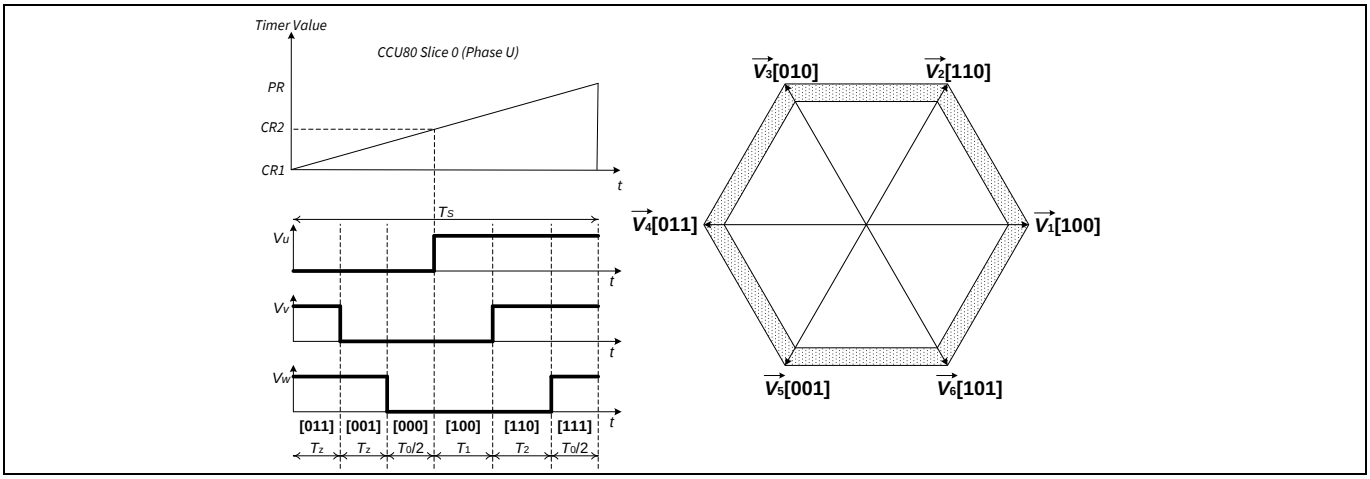


Figure 18 Pseudo Zero Vector timing diagram in sector A

2.5.3 4-segment SVM

This SVM pattern is also used in single shunt current sensing technique. Pseudo Zero Vector is useful in certain conditions, at sector crossovers or when the length of the vector is low. However if PZV is used throughout, the motor might not be able to spin up to its maximum target speed due to the limitation in the voltage amplitude; the higher is the T_Z value, the lower is motor speed that it can reach. This is the shaded area in the SVM diagram in Figure 18. To resolve this condition, 4-segment SVM is used.

In the PMSM_FOC software, a transition between PZV and 4-segment SVM PWM generation is implemented to resolve this issue. When T_1 and T_2 are greater than T_{min} and the motor speed is more than 75% of the maximum motor speed, 4-segment SVM is used. In this way, maximum target speed can be achieved.

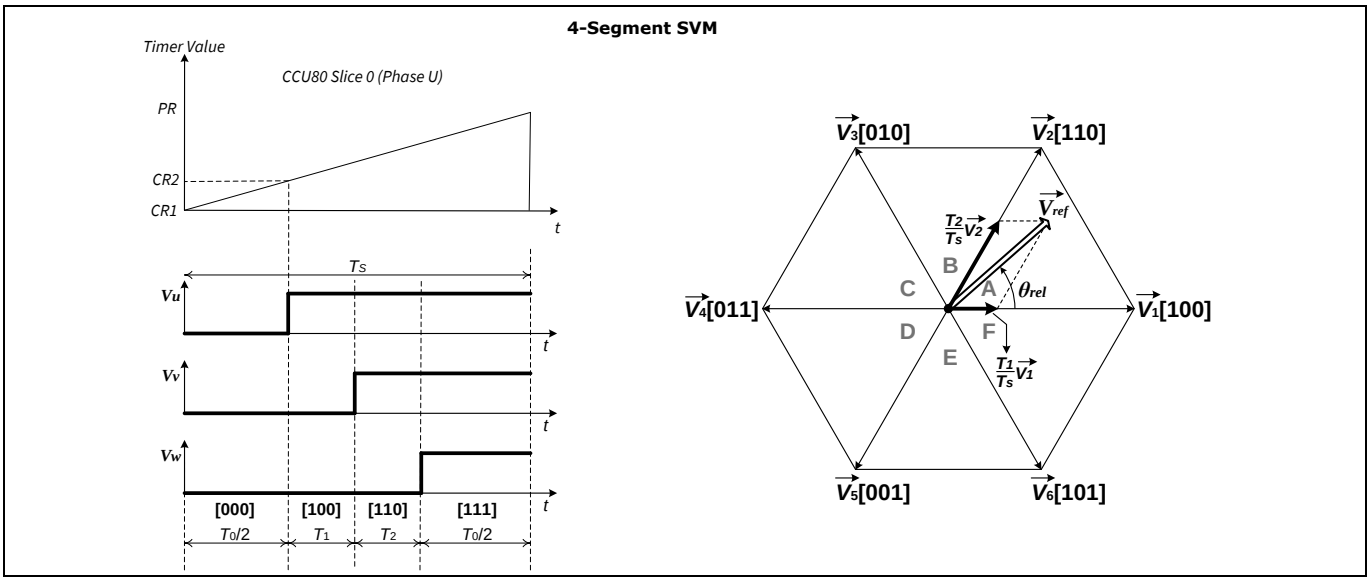


Figure 19 4-segment SVM

2.5.4 Over-modulation SVM

For sinusoidal commutation, V_{ref} has to be smaller than 86% of the maximum V_{ref}. For non-sinusoidal commutation, the SVM can have a higher V_{ref} amplitude. This technique is referred as over-modulation of SVM.

Figure 20 shows the area (shaded in red) where the over-modulation technique is used.

In the PMSM FOC software, the over-modulation is implemented by reducing the amplitude of V_{ref} to the V_{ref_new}. When T_1 plus T_2 is more than PWM period, T_S, the vector is reduced to follow the edge of the hexagon. This is done by reducing the T_1 and T_2 timing proportionally.

T_1 and T_2 are recalculated as:

$$T_{1\_new} = T_1 \times \frac{T_S}{T_1 + T_2}$$

$$T_{2\_new} = T_S - T_{1\_new}$$

In over-modulation, the time for zero vector is reduced to zero. The new timing diagram of SVM sector A is shown in the left diagram in Figure 20. From the diagram, it shows that only 2-phase currents, I_v and I_w, can be measured.

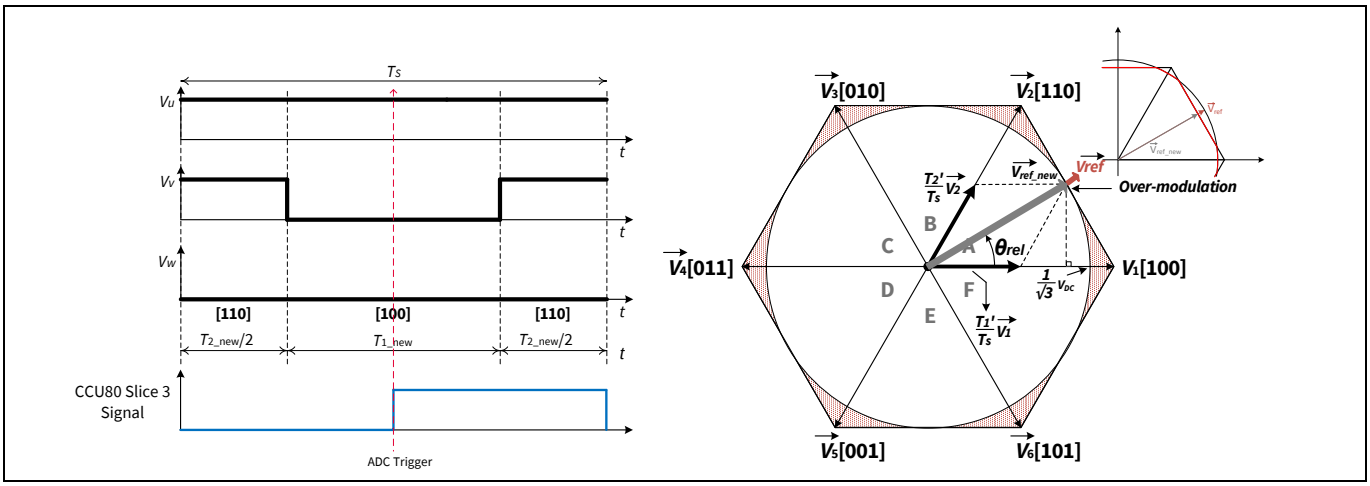


Figure 20 Over-modulation

When the motor is running at high speed, over-modulation is used to maximize DC bus utilization. The drawback of over-modulation is that the output voltage is not sinusoidal, and it contains high order harmonics which causes acoustic noise.

2.6 DC link voltage

The DC link voltage is measured via the voltage divider on the power inverter board. The measured value is scaled to 2^12. The voltage divider ratio value is defined in the pmsm_foc_user_parameter.h, refer to chapter 7.2.2.

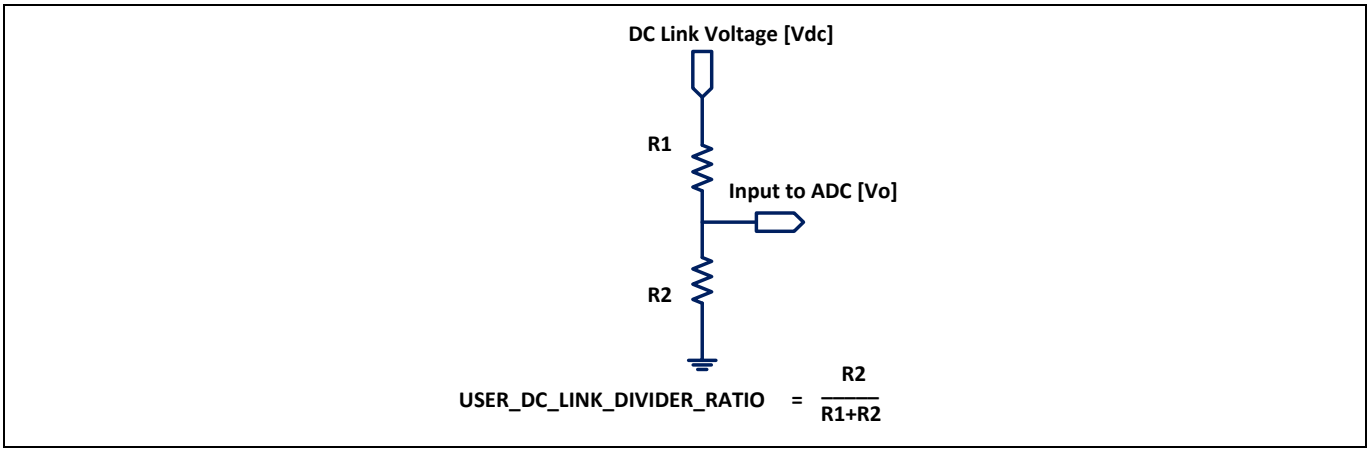


Figure 21 DC link voltage divider

2.7 Clarke transform

In this module, the phase currents ($I_{I\_u}, I_{I\_v}, I_{I\_w}$) from the current sensing module, are transformed into currents I_Alpha and I_Beta on the 2-phase orthogonal reference frame.

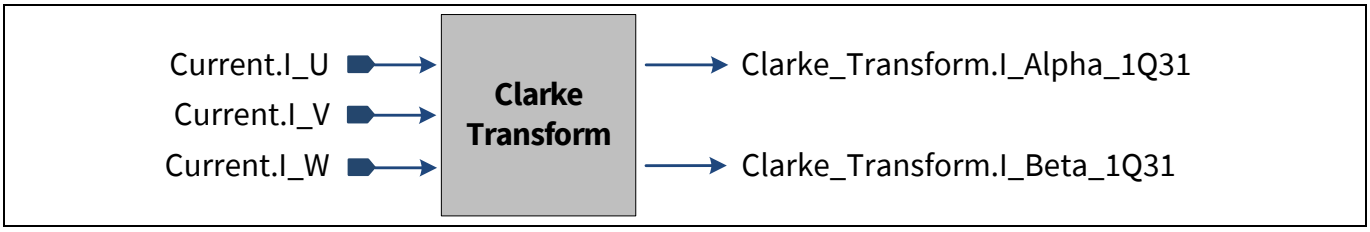


Figure 22 Clarke transform

Equations for Clarke transform:

$$I_\alpha = I_{I\_u}$$

$$I_\beta = \frac{1}{\sqrt{3}} \cdot I_{I\_u} + \frac{2}{\sqrt{3}} \cdot I_{I\_v} = \frac{1}{\sqrt{3}} \cdot \left(I_{I\_u} + 2 \cdot I_{I\_v}\right)$$

where $I_{I\_u} + I_{I\_v} + I_{I\_w} = 0$

Scaling factor of the Current.I_U, I_V and I_W are based on the current scaling as mentioned in chapter 2.10. The outputs of the Clarke Transform are shifted left by 14 bits to change the format to 1Q31.

2.8 Park transform

In the Park transform, the currents I_Alpha and I_Beta are resolved to a rotating orthogonal frame with rotor angle.

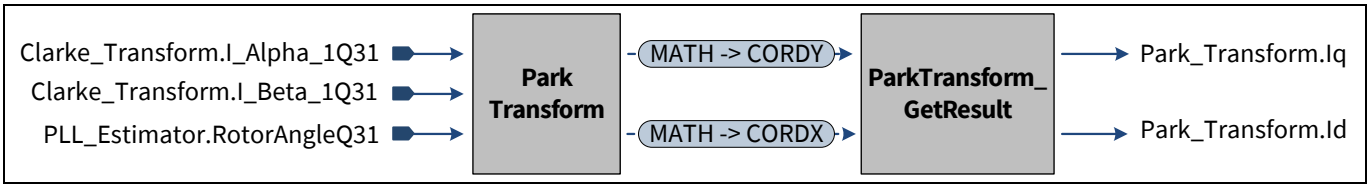


Figure 23 Park transform

This Park transform is calculated by the CORDIC coprocessor.

$$I_q = \left(-I_\alpha \cdot \sin(RotorAngle) + I_\beta \cdot \cos(RotorAngle)\right) * CORDIC\_GAIN$$

$$I_d = \left(I_\alpha \cdot \cos(RotorAngle) + I_\beta \cdot \sin(RotorAngle)\right) * CORDIC\_GAIN$$

Note: : CORDIC_GAIN = K/MPS

The input, PLL_Estimator.RotorAngleQ31 is shifted left by 14 bits due to code optimized for XMC CORDIC hardware module (refer to XMC1302 reference manual [1]).

Scaling factor of I_q and I_d are based on the current scaling as mentioned in chapter 2.10.

Table 8 XMC1302 CORDIC settings for Park transform

Parameters	Settings
CORDIC Control Mode	Circular Rotating Mode
K	≈ 1.646760258121
Magnitude Prescaler, MPS	2
CORDX	Clarke_Transform.I_Beta_1Q31
CORDY	Clarke_Transform.I_Alpha_1Q31
CORDZ	PLL_Estimator.RotorAngleQ31

2.9 Protection

The PMSM FOC motor control software supports the following protection schemes:

- CCU80 CTrap Function
- Over-Current Protection
- Over/Under Voltage Protection

CCU80 CTrap Function

Trap function of the CCU8 module provides hardware overload condition protection. The CTrap input pin is connected to the fault pin of the gate driver. Once the gate driver detects a fault, the CTrap pin is set to active state and the PWM outputs are set to PASSIVE level. The CTrap interrupt is triggered. In the Interrupt Service Routine, the gate driver is disabled and the motor state machine is set to TRAP_PROTECTION state.

Over-Current Protection

The average current flow through DC link shunt resistor is sampled every cycle of PWM. This value is read to detect over current condition. Once this condition occurs, the reference motor speed is scaled down by a factor till the current is within the limit (USER_IDC_MAXCURRENT_A) defined in the user configuration file.

The variable, FOCInput.overcurrent_factor, is used to update motor speed using the equation below:

$$FOCInput.Ref\_Speed = \frac{Motor.Ref_{Speed} * FOCInput.overcurrent\_factor}{4096}$$

FOCInput.overcurrent_factor is reduced if the average DC link current is above the limit. The nominal value of the FOCInput.overcurrent_factor is 4096. This value is increased back to nominal when the average DC link current is within the limit.

Over/Under Voltage Protection

The DC link voltage is read every PWM period cycle. If DC link voltage is less than or greater than specific user limits, the gate driver is disabled to stop driving the motor. The CCU8 timer is still running. The motor state machine is changed to DC_LINK_OVER_UNDER_VOLTAGE state.

The voltage protection check is not done during motor ramping and open loop condition.

2.10 Scaling

PMSM FOC software uses integers to represent real-world floating-point variables, such as angle, current, and voltage. To provide the best resolution, the software represents the Physical Value depending on the Target Value.

For example, the phase current is represented by 0 - 100% of the Target Value, where the Target Value is the maximum current can be measured by current sensing circuit.

The following equation shows the conversion of Physical Value to the Norm Value represented in the software.

$$Norm\ Value = \frac{Physical\ Value\ *\ 2^N}{Target\ Value}$$

Table 9 Scaling used in PMSM FOC software

Parameter	Scaling		Range	
	Target Value [Unit]	**N**	**[%]**	**[hex]**
SVM Amplitude (V_{ref})	$N_{\_Vref\_(SVM)}$ [Volts]	15	0% to 100%	0x0 to 0x7FFF
Current I_U, I_V, I_W, I_q, I_d	$N_{\_I\_(\alpha,\beta)}$ [A]	15	-100% to 100%	0x8001 to 0x7FFF
Angle of rotor position, Angle of space vector	360° [degree]	16 + USER_RES_INC	0 to 360°	0x0 to 0xXFFFF (for 16+ USER_RES_INC bit)
Speed	N_{max_speed} [degree/second]	$\log_2(max\_speed\_integer)$	0% to 100%	0x0 to max_speed_interger

In the following sections, the calculations of the Target Values are described.

Scaling for SVM voltage

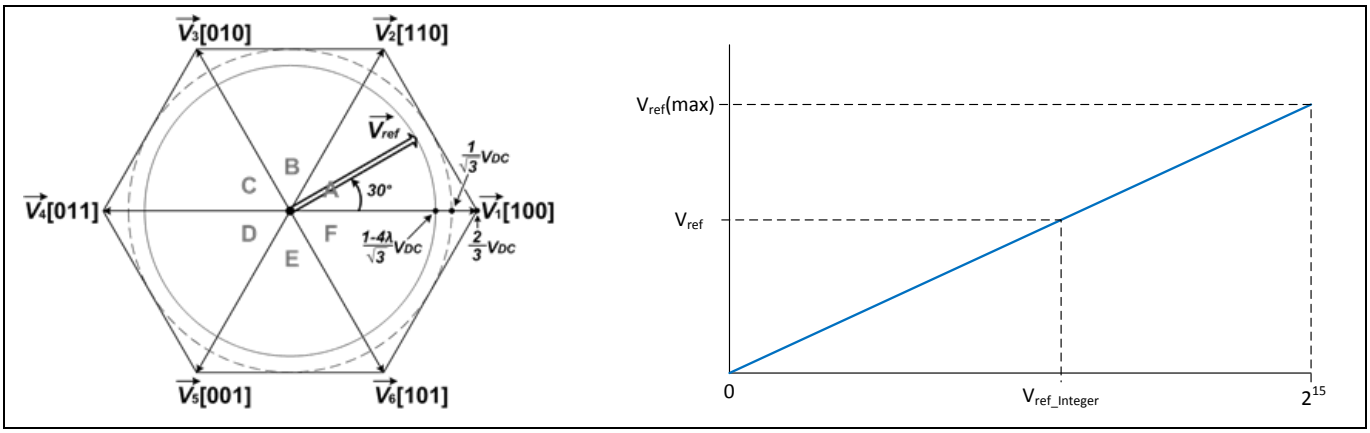


Figure 24 SVM voltage scaling

$$V_{ref} = \frac{N_{\_Vref\_SVM}}{2^{15}} \cdot V_{ref\_Integer}$$

$N_{\_Vref\_SVM}$ is the maximum reference voltage of SVM

$$N_{\_Vref\_SVM} = \frac{1-4\lambda}{\sqrt{3}} V_{DC}$$

$\lambda = T_Z/T_S$ is the pseudo zero vector ratio, λ = 0 for standard SVM

V_{DC} is inverter DC link voltage, USER_VDC_LINK_V

Example:

USER_VDC_LINK_V is 24.0f, λ = 0

⇨ $N_{\_Vref\_(SMV)}$ = 13.86 V

To represent V_{ref} = 0.5 V, the software integer, $V_{ref\_integer}$ is 1182.

Scaling for phase current

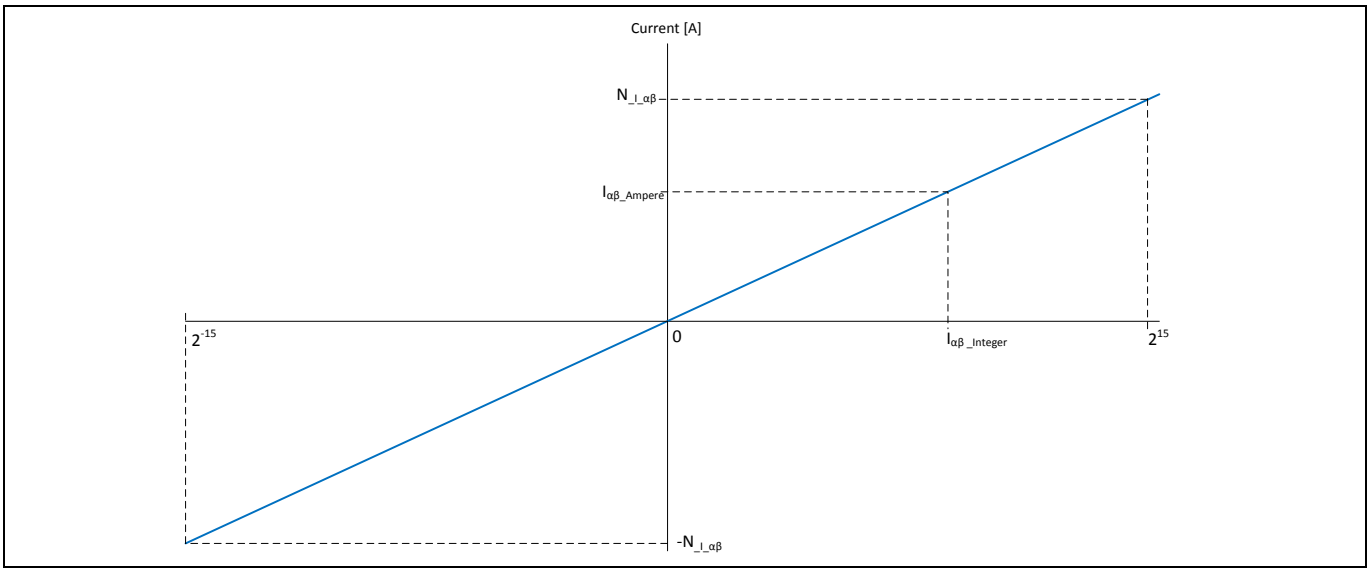


Figure 25 Phase current scaling

The Target Value of current is the maximum current that can be measured by current sensing circuit

$$N_{\_I\_(\alpha,\beta)} = \frac{V_{AREF}/2}{R_{shunt} \times G_{OpAmp}}$$

If internal ADC gain is used, G_{OpAmp} is replaced with the ADC gain factor setting.

Scaling for angle and speed

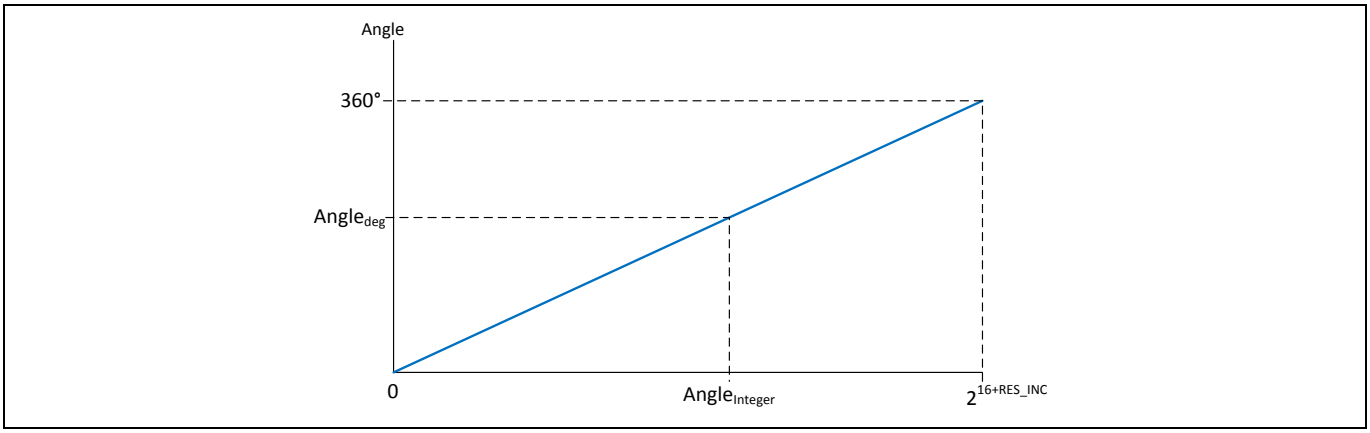


Figure 26 Angle scaling

In the PMSM_FOC software, it uses 16-bit (or 16 + USER_RES_INC bits where USER_RES_INC: 0~8) integers to represent angles of 0° to 360°. Below is the angle scaling:

$$Angle_{deg} = \frac{360°}{2^{16+RES\_INC}} \cdot Angle_{integer}$$

Following above angle scaling, the speed scaling is:

$$\omega_{degree/second} = \frac{N_{max\_speed}}{2^{N}} \cdot \omega_{integer}$$

Where $\omega_{integer}$ is the angle increase/decrease every CCU8 PWM cycle (i.e.: integer speed),
Target Value for speed is:

$$N_{max\_speed} = \frac{USER\_SPEED\_HIGH\_LIMIT\_RPM * USER\_MOTOR\_POLE\_PAIR}{60} * 360° \; degree/seconds$$

$$max\_speed\_integer = \frac{N_{max\_speed} * 2^{16+USER\_RES\_INC}}{60 * f_{CCU8\_PWM}}$$

$$N = \log_2(max\_speed\_integer)$$

Note: *If speed control is not done in every CCU8 PWM cycle, above scaling needs to be adjusted accordingly based on speed control rate.*

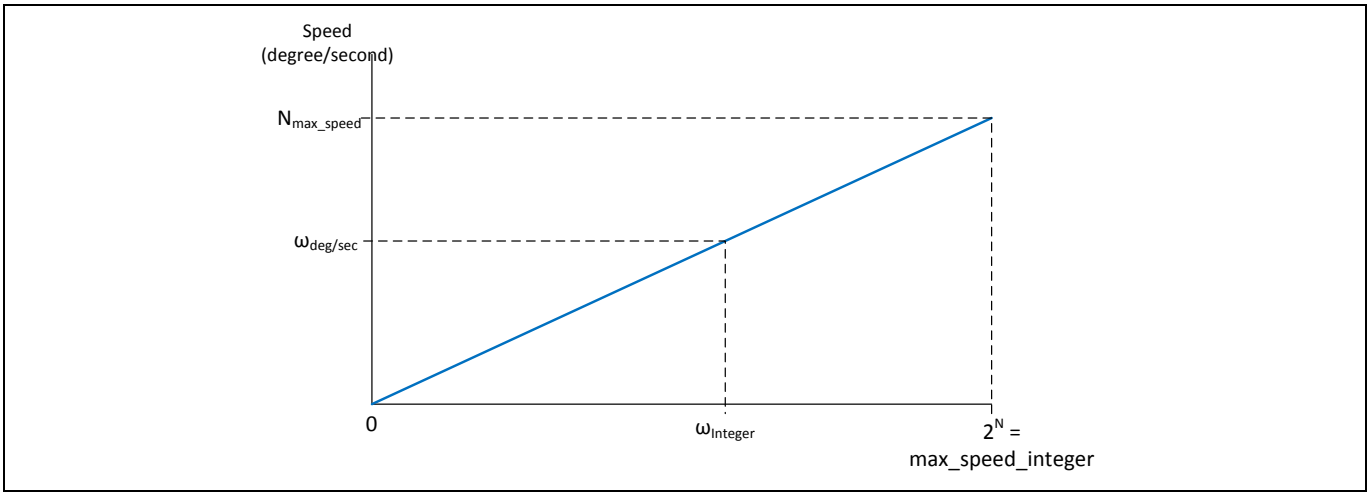


Figure 27 Speed scaling

Example:

- Motor maximum speed, USER_SPEED_HIGH_LIMIT_RPM = 10,000 rpm
- Motor pole pairs, USER_MOTOR_POLE_PAIR = 4
- CCU8 PWM Frequency = 25 KHz
- USER_RES_INC = 3

⇨ $N_{max\_speed} = \frac{(10000\ rpm * 4) * 360°}{60} = 240{,}000\ degree/second$

⇨ $max\_speed\_integer = \frac{240{,}000 * 2^{16+3}}{360 * 25{,}000} = 13{,}981$

⇨ $N = \log_2(max\_speed\_integer) = \log_2(13{,}981) = 13.77$

To represent speed 2,000 rpm which is 48,000 (degree/second), the software integer, $\omega_{integer}$= 2,796

3 Current sensing and calculation

This module is used to measure motor phase currents using VADC peripheral.

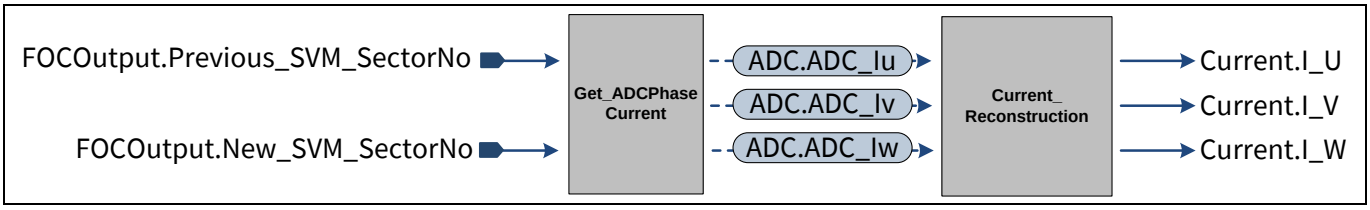


Figure 28 Current sensing and calculation functions

Two techniques to measure phase currents:

- Single shunt current sensing
- Three shunt current sensing

User can select the option of the current sensing technique in the user configuration file, refer to chapter 7.2.1. The phase currents measurements are synchronized with the PWM SVM pattern generation. The fourth slice of the CCU80 module, slice 3, is used to trigger the ADC conversions. Initial settings of the CCU80 and VADC modules for different current sensing techniques are listed in their respective sub-chapters, chapter 3.1.1 and chapter 3.1.2.

The figure below shows the timing diagram of the three shunt current sensing technique using synchronous ADC conversion.

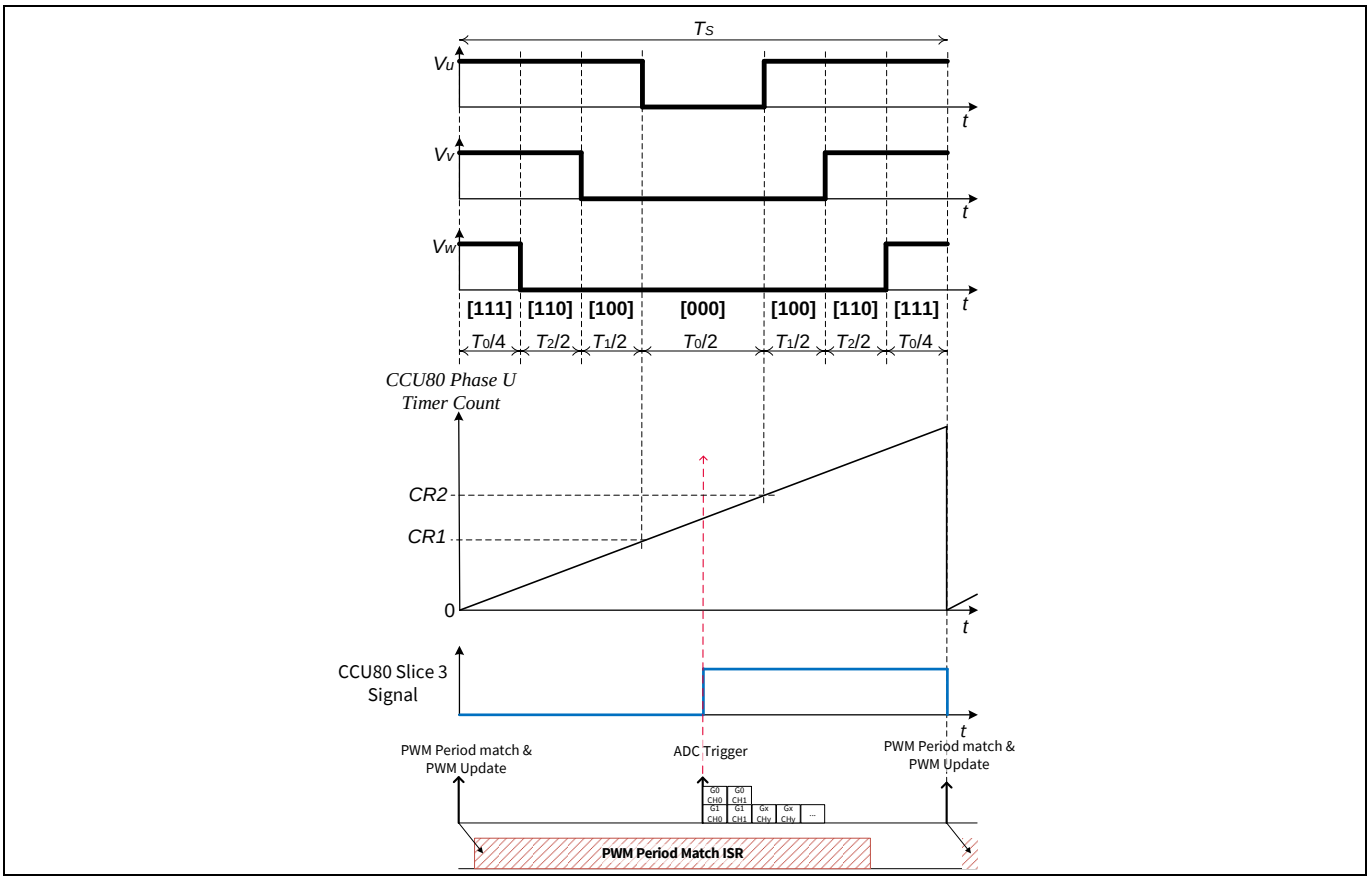


Figure 29 Three shunt current sensing timing diagram using synchronous conversion ADC

In XMC1302, a gain stage can be applied to all analog channels of the VADC peripheral. This is to compensate low amplitude signals. With this feature, external fast op-amp is not required for the phase current signals. This leads to cost saving in the BOM of the PCB boards.

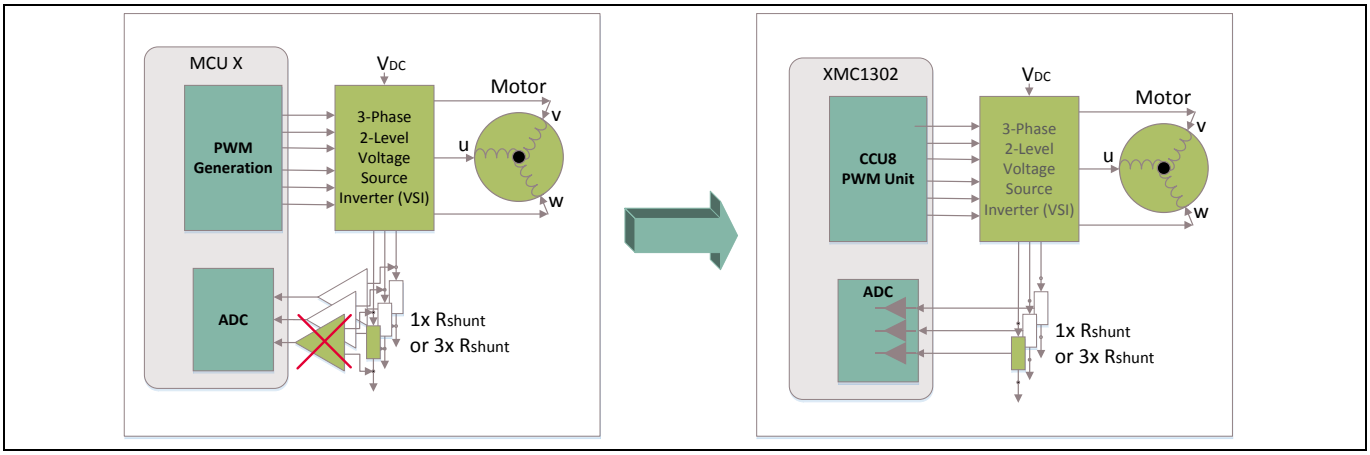


Figure 30 On-chip gain in XMC1302 VADC peripheral

3.1.1 Single shunt current sensing

The single shunt current measurement technique measures the power supply current and with knowledge of the switching states recreates the three phase current of the motors.

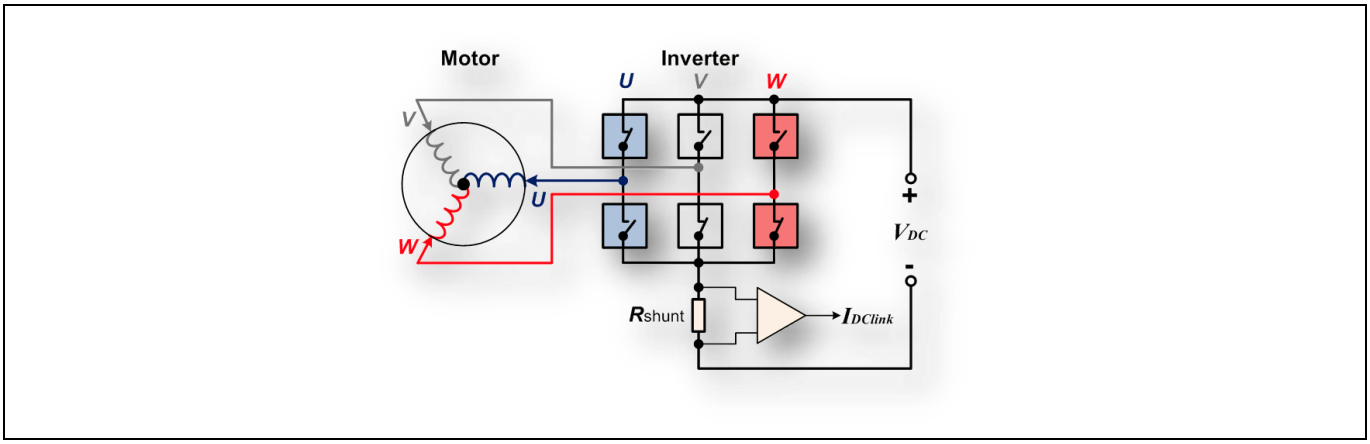


Figure 31 Single shunt sensing technique

The Pseudo Zero Vector (PZV) SVM is used to ensure enough time is given for single shunt current sensing. Figure 32 shows the direction of the voltage space vector and what current can be measured in that state. The CCU80 slice 3 is used as a timer to automatically trigger the ADC conversion at specific time as shown in Figure 32. The ADC conversions are triggered at both the rising and falling edges of the CCU80 slice 3 signal. When 4–segment SVM is used, the ADC conversion trigger points are also changed, refer to Figure 33.

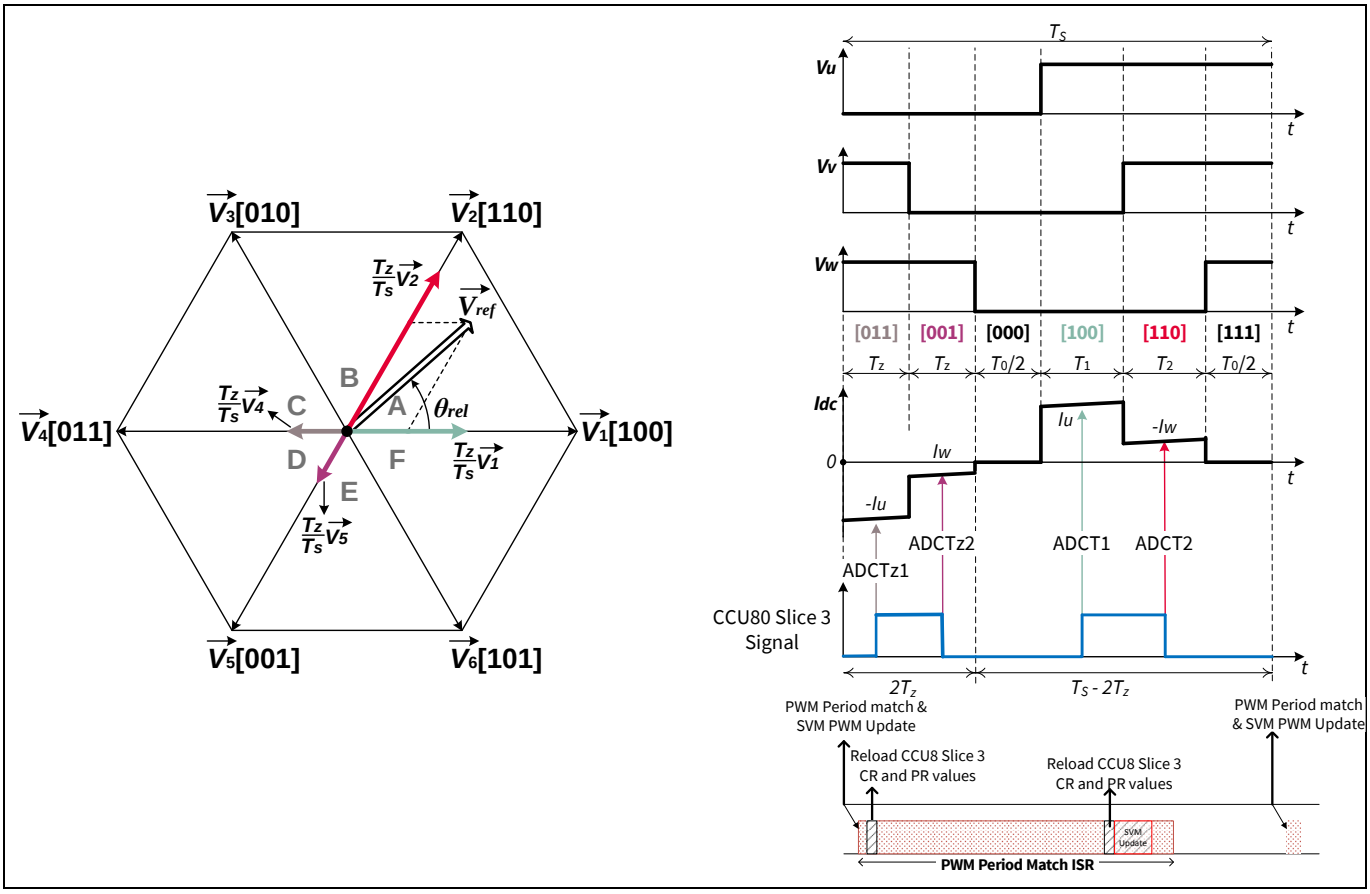


Figure 32 Single shunt – 3-phase current sensing in sector A

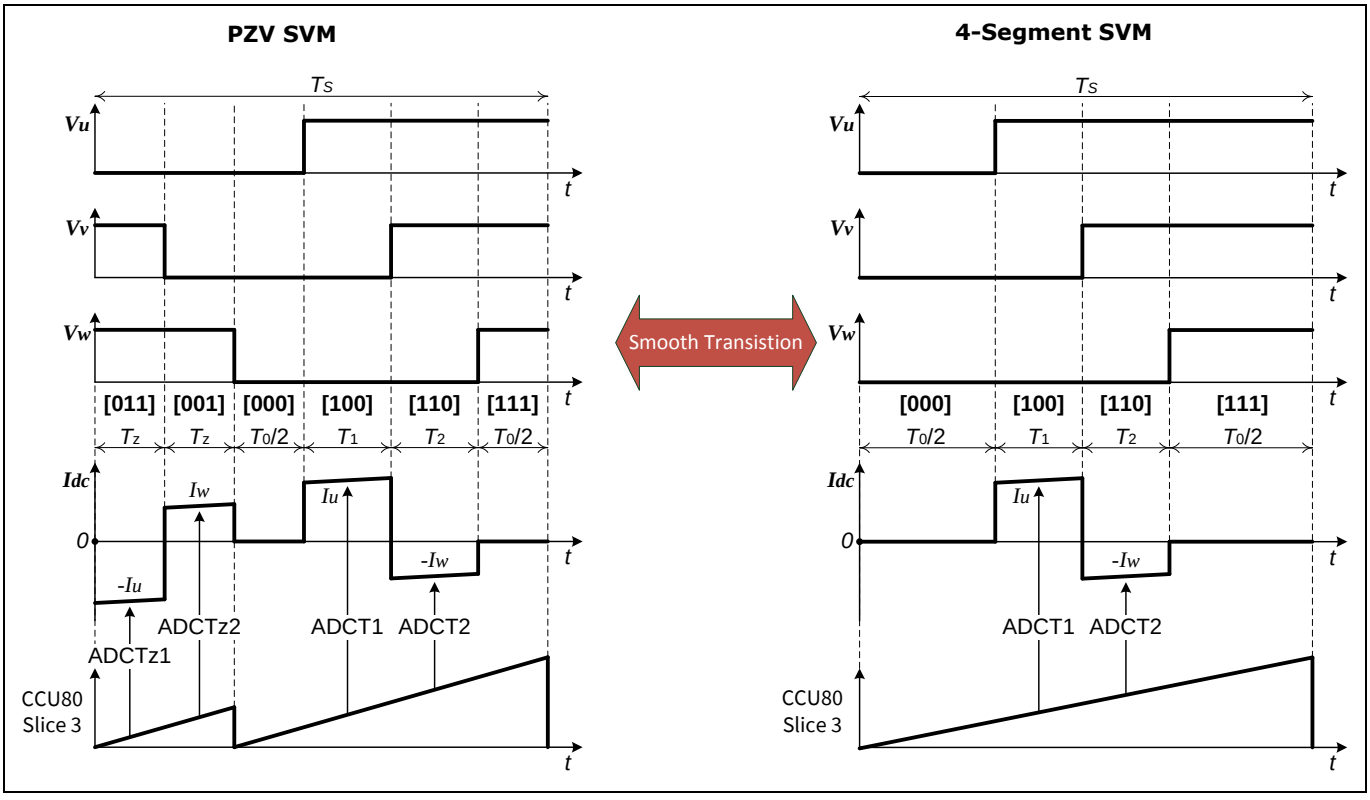


Figure 33 Smooth transition from PZV to 4-segment SVM in sector A

Current sensing and calculation

The VADC source interrupt event is enabled and its service routine performs the following tasks:

- Read the results of the ADC conversion
- Generate the phase current values and scale the values to 2^{15}

The measured 12-bit current value is scaled to 1Q15 format.

In PZV:

- Current -> $I\_U = (I_{ADCT1} - I_{ADCTz1}) * 2^3$
- Current -> $I\_W = (I_{ADCT2} - I_{ADCTz2}) * 2^3$

In 4-segment SVM:

- Current -> $I\_U = \left(I_{ADCT1} - I_{ADC\_Bias}\right) * 2^4$
- Current -> $I\_W = \left(I_{ADCT2} - I_{ADC\_Bias}\right) * 2^4$

The two tables below show the initial settings of the VADC and CCU80 slice 3 for single shunt current sensing technique.

Table 10 VADC initial settings for single shunt

Parameters	Settings
Request Source for Single Shunt	Queue
Request Source for other Channels	Background Scan
FIFO for Single Shunt	2-Stage Buffer
Source Interrupt	Enabled
ADC Conversion Trigger Signal	CCU80.ST3A (through gating select input)
ADC Conversion Trigger Edge	Both Rising and Falling Edges

Table 11 CCU80 slice 3 initial setting for single shunt

Parameters	Settings
Timer Counting Mode	Edged Aligned
Single Shot Mode	Enabled
Period Register	$2 * T_Z$
Compare Register Channel 1, CR1	$T_Z * 0.85$
Compare Register Channel 2, CR2	$T_Z + T_Z * 0.85$

3.1.2 Three shunt current sensing

The three shunt current measurement technique is more robust as compared with single shunt sensing. Using this technique, we can select two out of the three phase currents for the current reconstruct calculation.

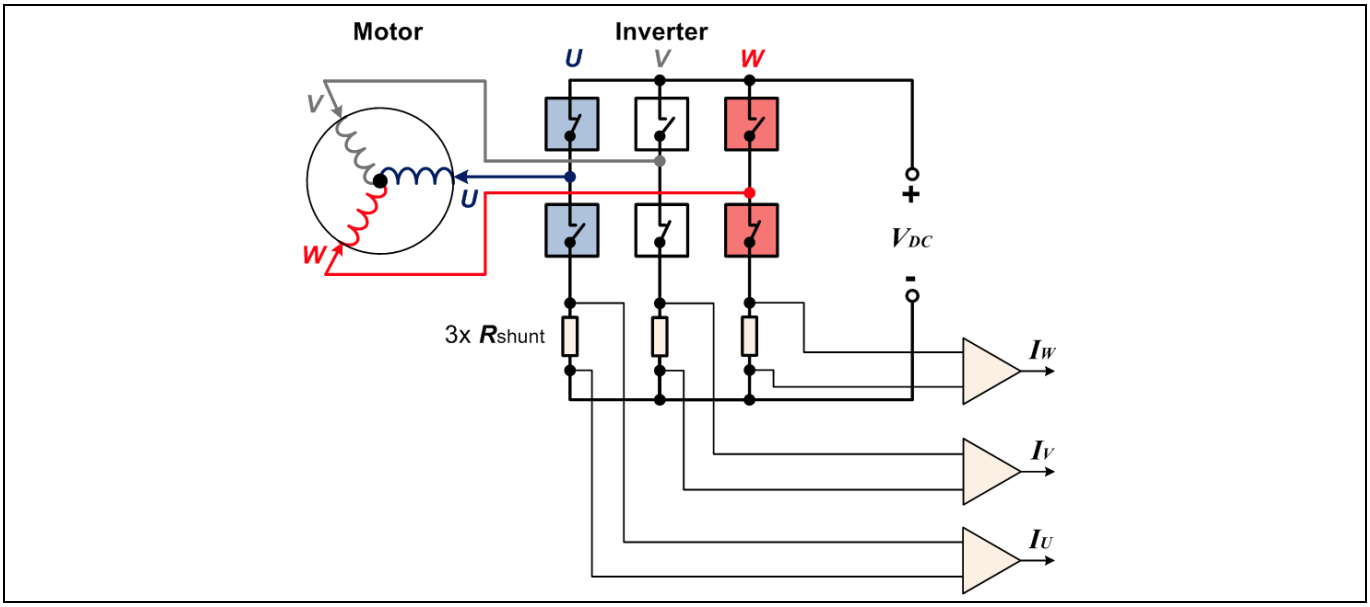


Figure 34 Three shunt sensing technique

For three shunt current sensing, the ADC conversion trigger is set at half of the PWM cycle where all the low side switches are on, refer to Figure 35. The current will always flow through the shunt resistor when the low side switch is on and high side switch is off.

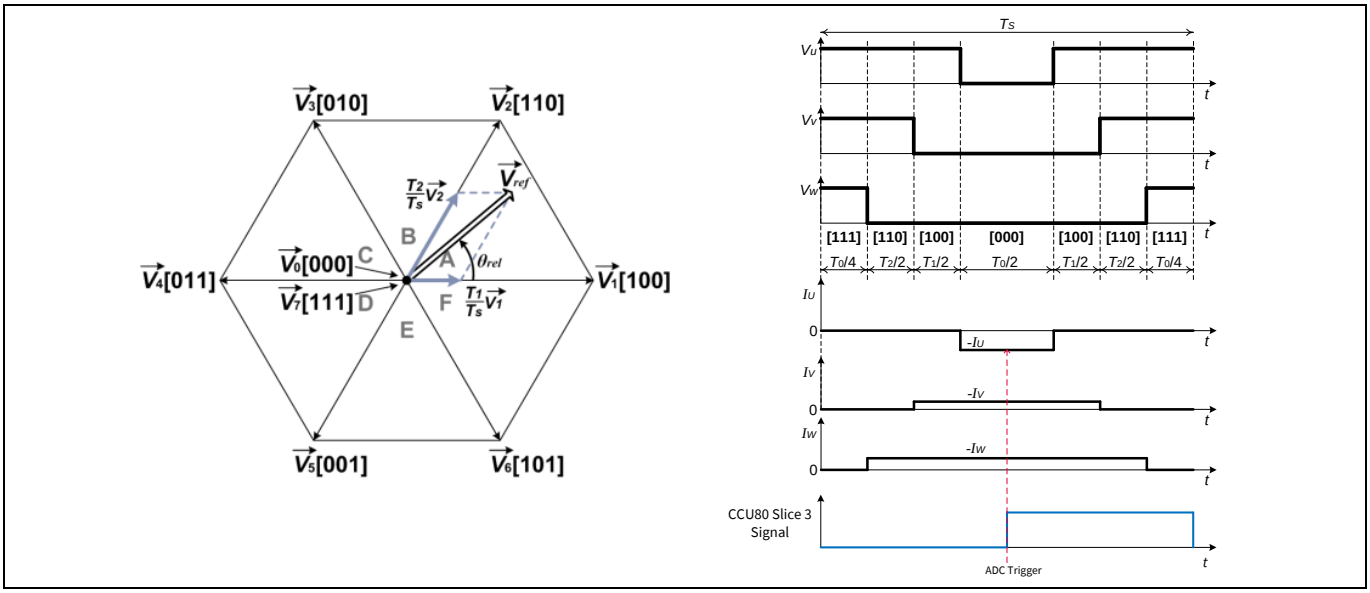


Figure 35 Three shunt – 3-phase current sensing in sector A

In the current calculation function, the measured 12-bit current value is scaled to 1Q15 format.

- Current -> $I\_U = \left(ADC\_Bias\_I_u - I_{ADC\_Iu}\right) * 2^3$
- Current -> $I\_V = \left(ADC\_Bias\_I_v - I_{ADC\_Iv}\right) * 2^3$

- Current -> $I\_W = \left(ADC\_Bias\_I_w - I_{ADC\_Iw}\right) * 2^3$

The initial settings of the VADC module and CCU80 slice 3 are tabulated in the following tables.

Table 12 VADC initial settings for three shunt

Parameters	Settings
Request Source for Three Shunt	Queue
Request Source for Other Channels	Queue
FIFO	Disabled
Source Interrupt	Disabled
ADC Conversion Trigger Signal	CCU80.ST3A (through gating select input)
ADC Conversion Trigger Edge	Rising Edge

Table 13 CCU80 slice 3 initial setting for three shunt

Parameters	Settings
Timer Counting Mode	Edged Aligned
Single Shot Mode	Disabled
Period Register	Same as Period Register value for 3-phase PWM
Compare Register Channel 1	Half of Period Register value
Compare Register Channel 2	Compare Register Channel 1 value + 1

3.1.2.1 Synchronous conversion for three shunt current sensing

The XMC1302 MCU has two VADC kernels and this allows synchronized conversion for measurement of two shunt currents at the same time. This is used to optimize timing required for ADC the sampling and conversion of the shunt currents. This results in a better performance especially at high motor speed.

The following gives an illustration on how synchronous conversion can be implemented in Infineon XMC1000 Motor Control Application Kit, KIT_XMC1X_AK_MOTOR_001. Table 14 shows the XMC1302 pins assignment of the phase currents.

Note: For synchronous current measurement, 2-phase currents have to be measured at the same time. Therefore the pins selected for Phase U, Phase V and Phase W must be allocated to both group 0 and group 1 channels.

Table 14 Three shunt ADC pins assignment

Shunt current	Pin assigned	ADC channels
Phase U	P2.11	Group 0 Channel 4 Group 1 Channel 3
Phase V	P2.10	Group 0 Channel 3 Group 1 Channel 2
Phase W	P2.9	Group 0 Channel 2 Group 1 Channel 4

In each SVM sector, the two most critical phase currents are assigned to be measured at the same time. For example in SVM sector A and F, the two most critical phase currents are I_V and I_W (Figure 36) as the phase U current can be inaccurate. This is because at very high motor speed, the window to measure phase U current can be very short and the reading of the phase U current could be missed.

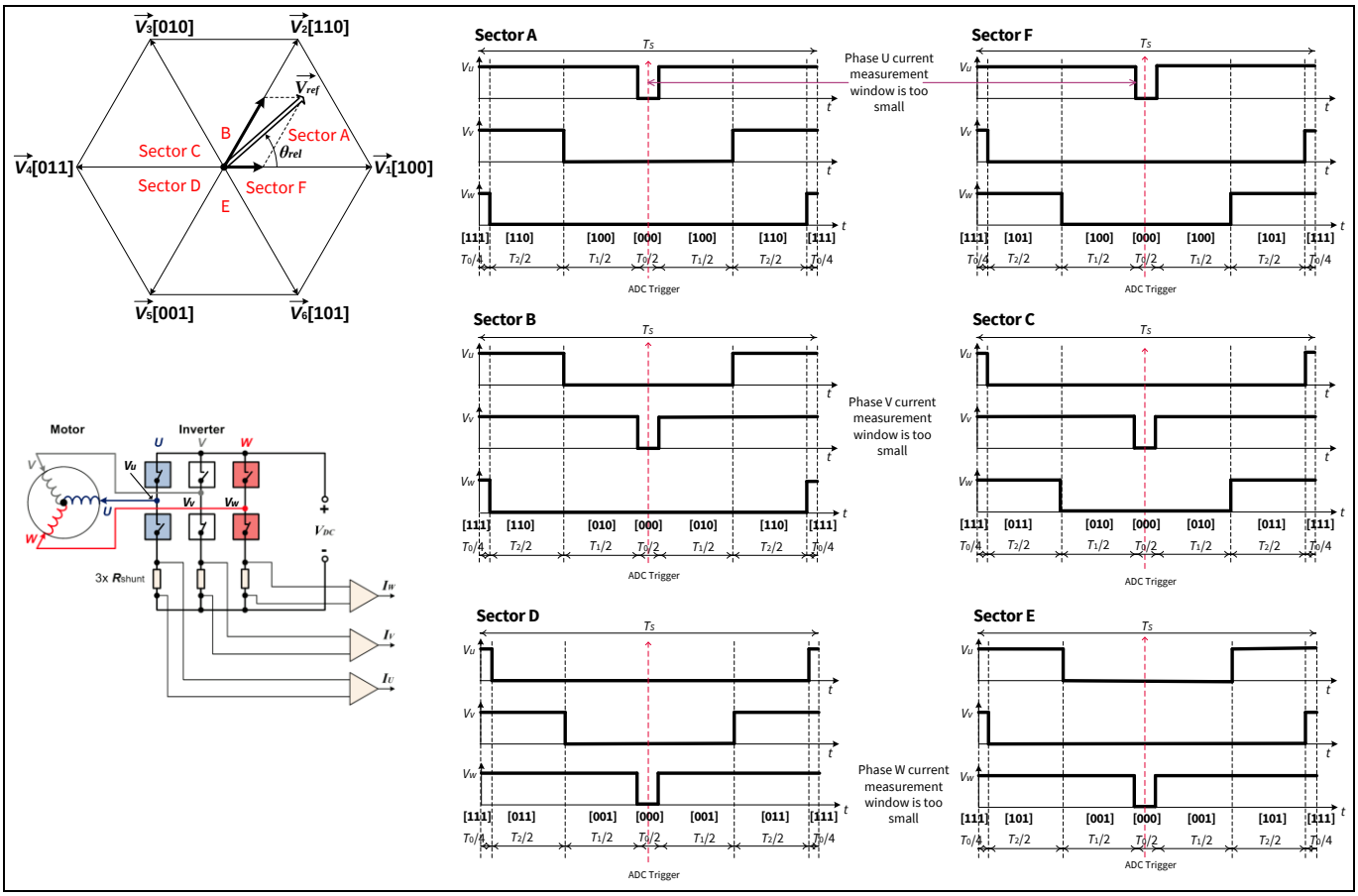


Figure 36 SVM sectors at high motor speed

Using the alias feature in the ADC module, we can assign different ADC input channels to be converted in parallel. Thus we can measure the two most critical phase currents for all the SVM sectors. For sectors A and F, we assign I_V and I_W ADC channels as aliasing channels to Channel 0 in group 0 and group 1, as shown in Figure 37.

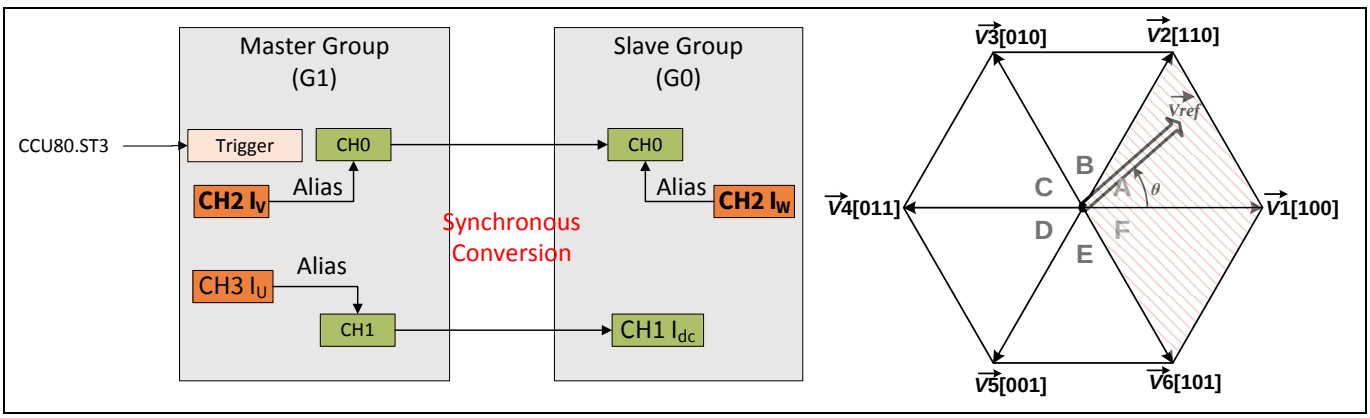


Figure 37 Synchronous conversion using alias feature – sectors A and F

In SVM sector B and sector C, we assign I_W and I_U ADC channels as aliasing channels to Channel 0 in group 0 and group 1.

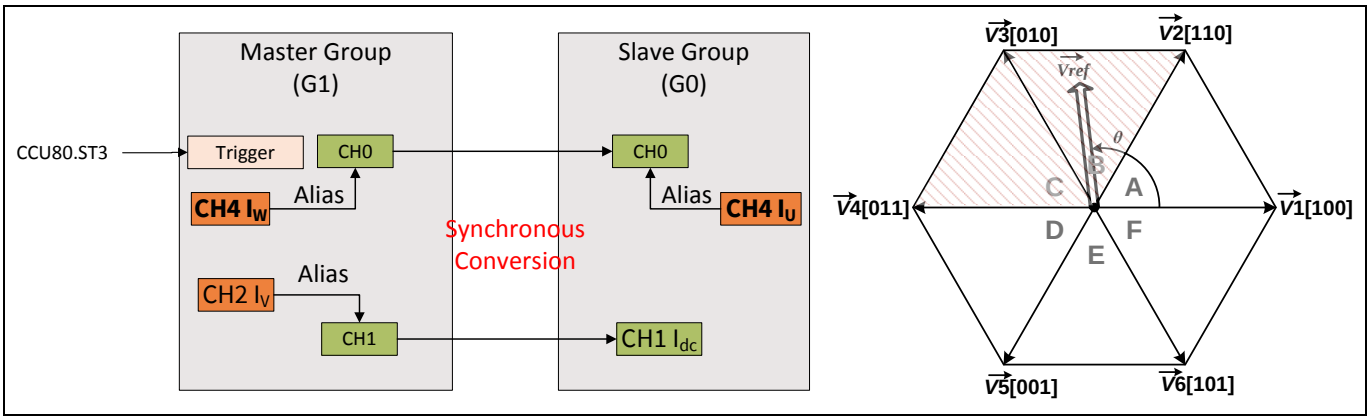


Figure 38 Synchronous conversion using alias feature – sectors B and C

Likewise in SVM sector D and sector E, we assign I_U and I_V ADC channels as aliasing channels to Channel 0 group 0 and group 1.

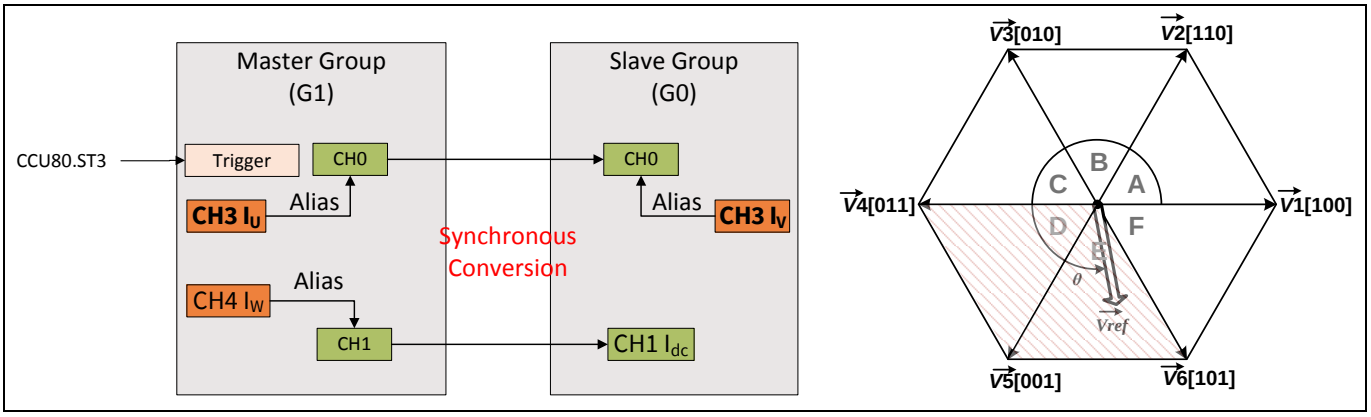


Figure 39 Synchronous conversion using alias feature – sectors D and E

Table 15 Aliasing settings for SVM sectors

SVM sectors	Shunt current measured	Alias channels for CH0
Sector A and F	Phase W (Pin P2.9)	Group 0 Channel 2
	Phase V (Pin P2.10)	Group 1 Channel 2
Sector B and C	Phase U (Pin P2.11)	Group 0 Channel 4
	Phase W (Pin P2.9)	Group 1 Channel 4
Sector D and E	Phase V (Pin P2.10)	Group 0 Channel 3
	Phase U (Pin P2.11)	Group 1 Channel 3

4 Motor speed and position feedback in sensorless FOC control

The rotor speed and position feedback of the motor are determined in the PLL Estimator software library. This library contains the Infineon patented IP and is provided as compiled libPLL_Estimator.a file.

Following the list of APIs provided in the library and it is important that these APIs are called in exact order:

1. PLL_Imag(int32_t Vref_AngleQ31, int32_t I_Alpha_1Q31, int32_t I_Beta_1Q31);
2. PLL_Imag_GetResult(PLL_EstimatorType* const HandlePtr);
3. PLL_Vref(int32_t Delta_IV, uint32_t Vref32, int32_t PLL_UK, int32_t Phase_L, PLL_EstimatorType* const HandlePtr);
4. PLL_Vref_GetResult(PLL_EstimatorType* const HandlePtr);
5. PLL_GetPosSpd(PLL_EstimatorType* const HandlePtr);

Find below a brief description of each API and the required parameters.

Table 16 PLL_Imag() function

Name	PLL_Imag(int32_t Vref_AngleQ31, int32_t I_Alpha_1Q31, int32_t I_Beta_1Q31)	
Description	This function is to start the first CORDIC calculation of the sensorless estimator.	
Input Parameters	Vref_AngleQ31	Angle of voltage space vector
	I_Alpha_1Q31	Alpha coordinate of current space vector
	I_Beta_1Q31	Beta coordinate of current space vector
Return	None	

Table 17 PLL_Imag_GetResult() function

Name	PLL_Imag_GetResult(PLL_EstimatorType* const HandlePtr)	
Description	This function read out the results of the first CORDIC calculation of the sensorless estimator.	
Input Parameters	HandlePtr	Pointer to the structure of PLL_Estimator
Return	Current_I_Mag	Current magnitude
	Delta_IV	To be provided as input parameter for the second CORDIC calculation

Table 18 PLL_Vref() function

Name	PLL_Vref(int32_t Delta_IV, uint32_t Vref32, int32_t PLL_UK, int32_t Phase_L, PLL_EstimatorType* const HandlePtr)	
Description	This function is to start the second CORDIC calculation of the sensorless estimator.	
Input Parameters	Delta_IV	Result of the first CORDIC calculation of the sensorless estimator
	Vref32	SVM voltage magnitude of last PWM cycle
	PLL_Uk	PLL Estimator PI controller output
	Phase_L	Phase inductance of motor stator winding
Return	Current_I_Mag	Updated current magnitude

Table 19 PLL_Vref_GetResult() function

Name	PLL_Vref_GetResult(PLL_EstimatorType* const HandlePtr)	
Description	This function is to read the results of the second CORDIC calculation of the sensorless estimator.	
Input Parameters	HandlePtr	Pointer to the structure of PLL_Estimator
Return	VrefxSinDelta	It is used for PLL_Estimator

Table 20 PLL_GetPosSpd() function

Name	PLL_GetPosSpd(PLL_EstimatorType* const HandlePtr)	
Description	This function is to calculate and read the rotor position and rotor speed from the sensorless estimator.	
Input Parameters	HandlePtr	Pointer to the structure of PLL_Estimator
Return	RotorAngleQ31	Estimated rotor position
	RotorSpeed_In	Estimated rotor speed

5 Interrupts

Interrupts events and its priorities in the PMSM FOC software are listed in the table below.

Table 21 Interrupts priorities

Interrupt events	Priorities	Comment
CTrap	0 (Highest priority)	Fault detection
ADC Queue Source	1	Only for single shunt sensing. It is disabled for three shunt sensing.
PWM Period Match (Phase U)	2	State machine execution

5.1 PWM period match interrupt

The PMSM_FOC state machine is executed in Phase U PWM frequency period match Interrupt Service Routine. Depending on the control schemes selected in the user configuration, different Interrupt Service Routine is called. Each Interrupt Service Routine will have its own state machine flow. Please refer to chapter 6 for more information on the state machine in the PMSM FOC software.

Table 22 Interrupt Service Routine based on control scheme selected

Control scheme	Interrupt Service Routine called	Description
CONSTANT_SPEED_VF_ONLY	VF_ONLY_CCU80_0_IRQHandler()	Open Loop Voltage Control
CONSTANT_SPEED_VF_MET_FOC	VF_FOC_CCU80_0_IRQHandler()	Speed Control Transition Startup
CONSTANT_SPEED_DIRECT_FOC	DirectFOCStartup_ CCU80_0_IRQHandler()	Speed Control Direct Startup
CONSTANT_TORQUE_DIRECT_FOC	DirectFOCStartup_ CCU80_0_IRQHandler()	Current Torque Control Direct Startup
CONSTANT_VQ_DIRECT_FOC	DirectFOCStartup_ CCU80_0_IRQHandler()	Voltage Torque Control Direct Startup

An example of the flow of the PWM period match interrupt is shown in the Figure 40. This example shows the flow of the FOC direct startup control scheme. The current sensing technique chosen is three shunt synchronous conversions.

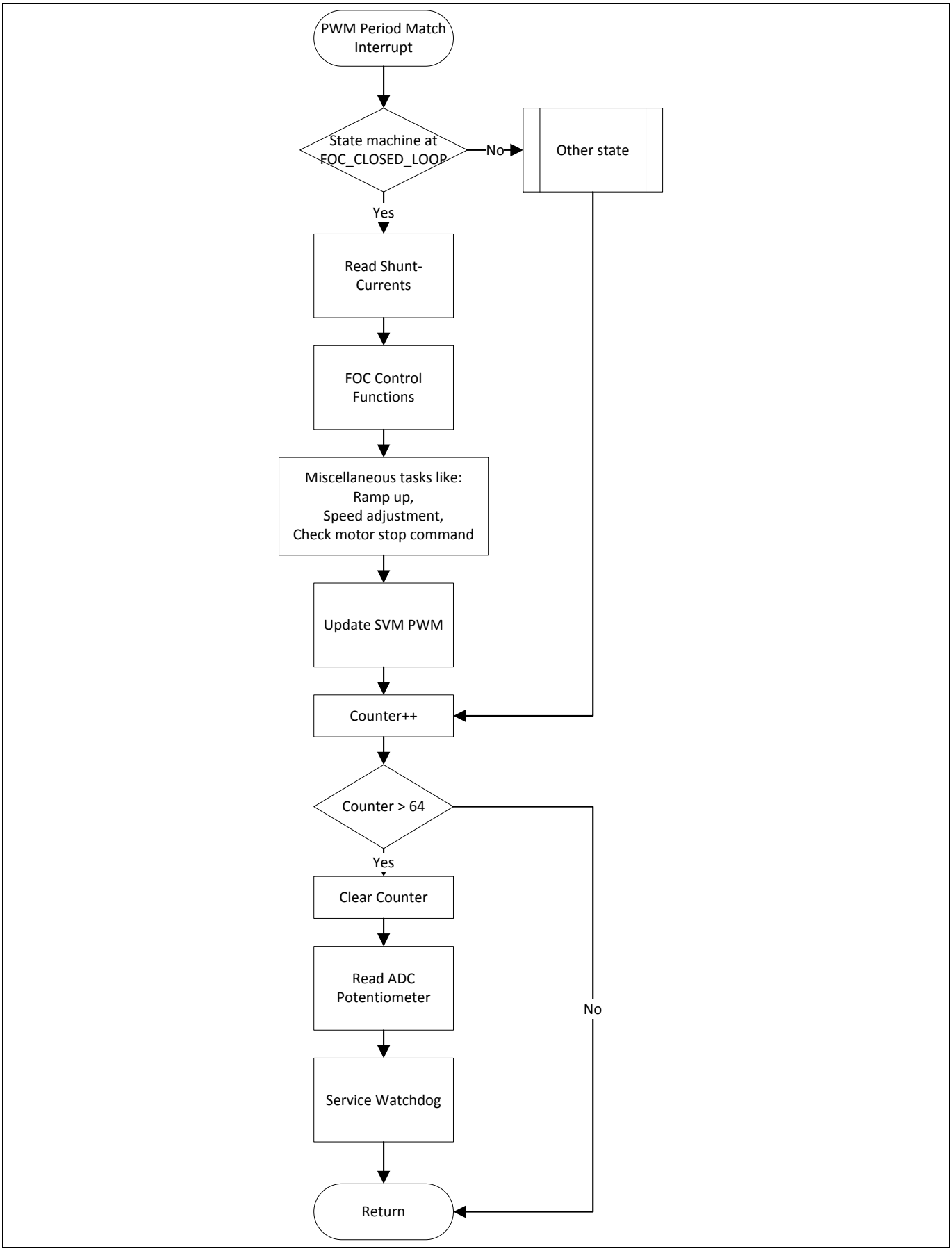


Figure 40 PWM period match Interrupt Service Routine flow chart

5.2 CTrap interrupt

When a TRAP condition is detected at the selected input pin (P0.12), the CCU80 outputs are set to passive level and Trap_Protection_INT() is executed. In the Interrupt Service Routine, the gate driver is disabled and the motor state is set to TRAP_PROTECTION. This ISR is executed with the highest priority, level 0.

5.3 ADC source interrupt

This interrupt is only enabled for single shunt current sensing. It is triggered at the end of ADC conversion. In the ISR, the ADC results are read.

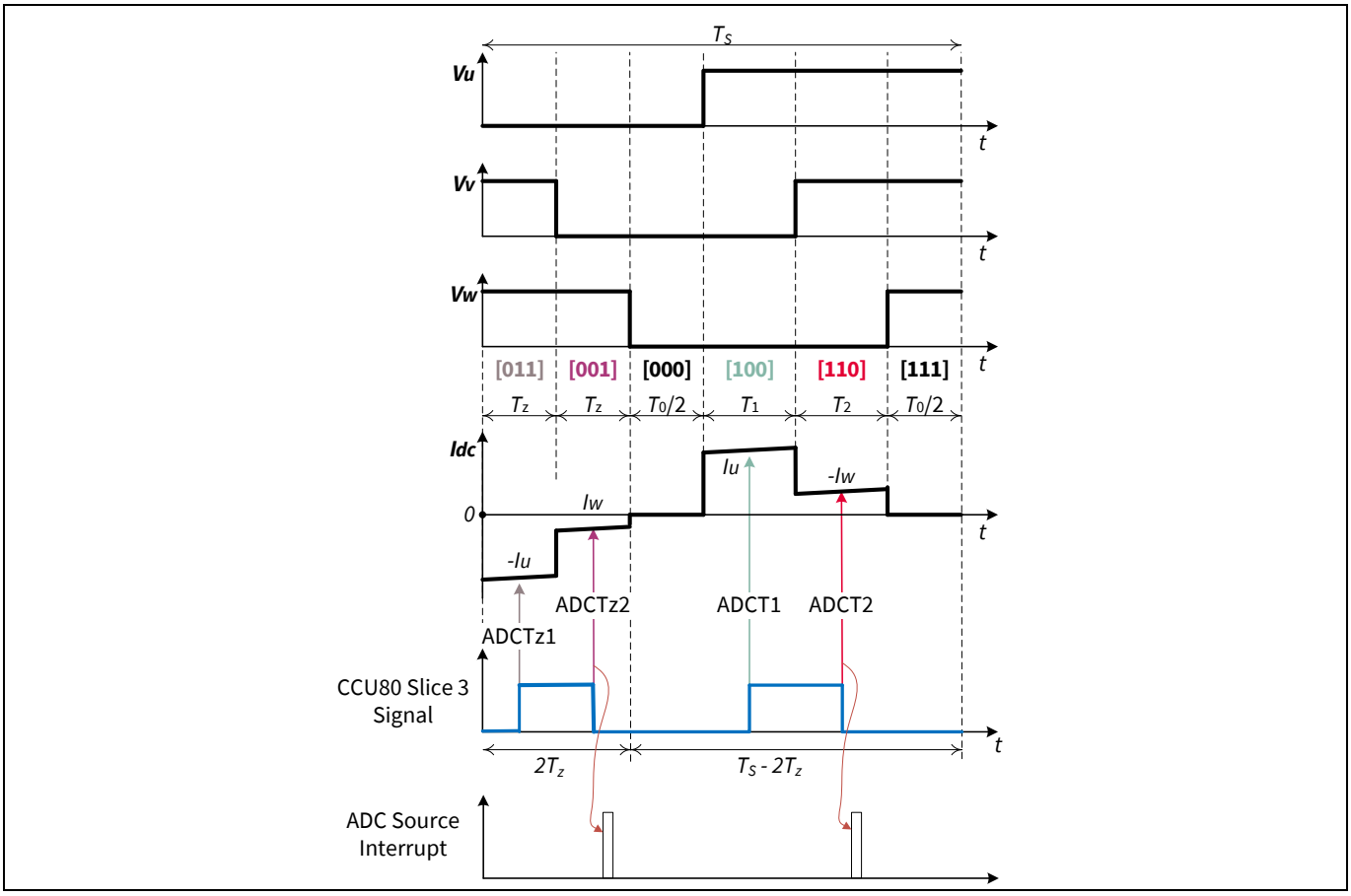


Figure 41 ADC source interrupt timing diagram

6 Motor state machine

The PMSM FOC software has an internal state machine; there are eight states in total.

BRAKE_BOOTSTRAP

This is the first state entered right after power-on or software reset. In this state the bootstrap capacitors are charged for a defined period. It also reads the bias voltage of the ADC pins that are connected to the motor phase currents. This state is exit when motor start command (via potentiometer) is received.

PRE-POSITIONING

This state is only for Direct FOC Startup control schemes. In this state the rotor is aligned to a known position to get the maximum starting torque. The amplitude input to the SVM function is gradually increased to a defined value USER_STARTUP_VF_OFFSET_V for a specific time, USER_ROTOR_PREPOSITION_TIME_MS. These macros are defined in the pmsm_foc_user_parameter.h file (refer to chapter 7.1, item USER_STARTUP_VF_OFFSET_V and chapter 7.2.1 item USER_ROTOR_PREPOSITION_TIME_MS respectively).

VF_OPENLOOP_RAMPUP

In this state, the motor is starts in V/F open loop control mode. This state is exit when the motor speed reached the startup threshold speed which is defined in the macro, USER_STARTUP_SPEED_THRESHOLD_RPM.

MET_FOC

This state enables a smooth transition from open loop to closed loop with maximum energy efficiency, refer to chapter 2.3.2.1 for more detail.

FOC_CLOSED_LOOP

In this state the motor is running in FOC mode. The FOC functions are executed.

STOP_MOTOR

This state is entered when the motor speed is ramp down till 10% of its maximum speed. In this state motor brake is applied and wait for motor start command to go to next state.

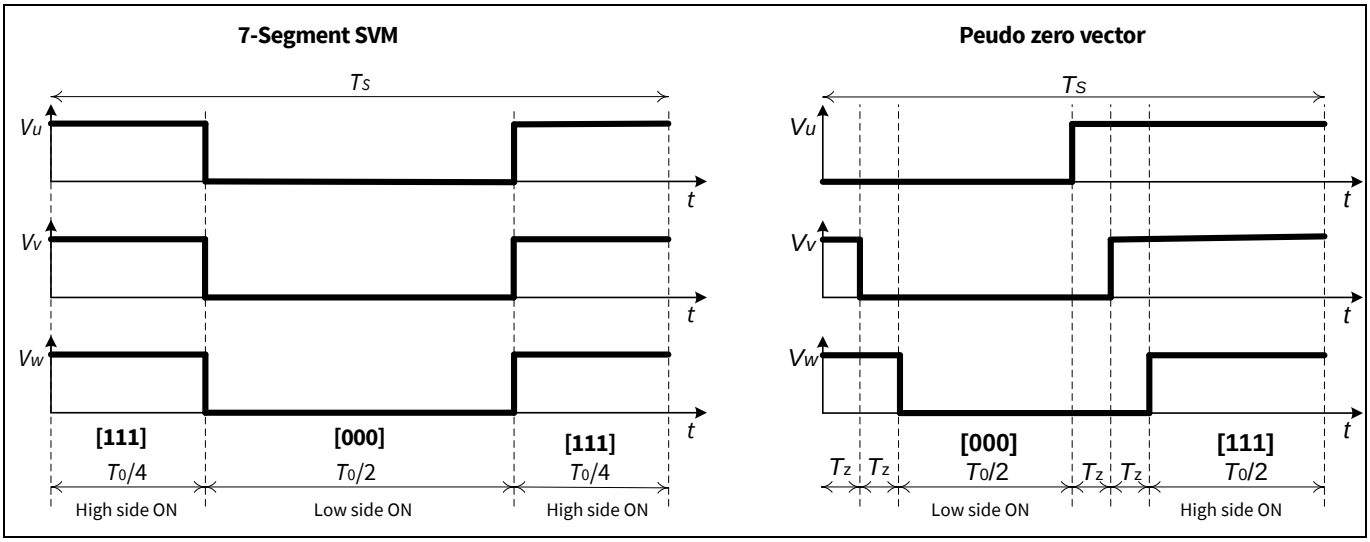


Figure 42 SVM outputs in motor brake condition

TRAP_PROTECTION

This state is entered if CTrap is triggered. To exit this state, turn the potentiometer to stop or motor stop command via the µC/Probe GUI, and the gate driver will be enabled.

DCLINK_OVER_UNDER_VOLTAGE

This state is entered when the DC link voltage is above or below the limits set by the user. The gate driver is disabled and the motor will be in free running.

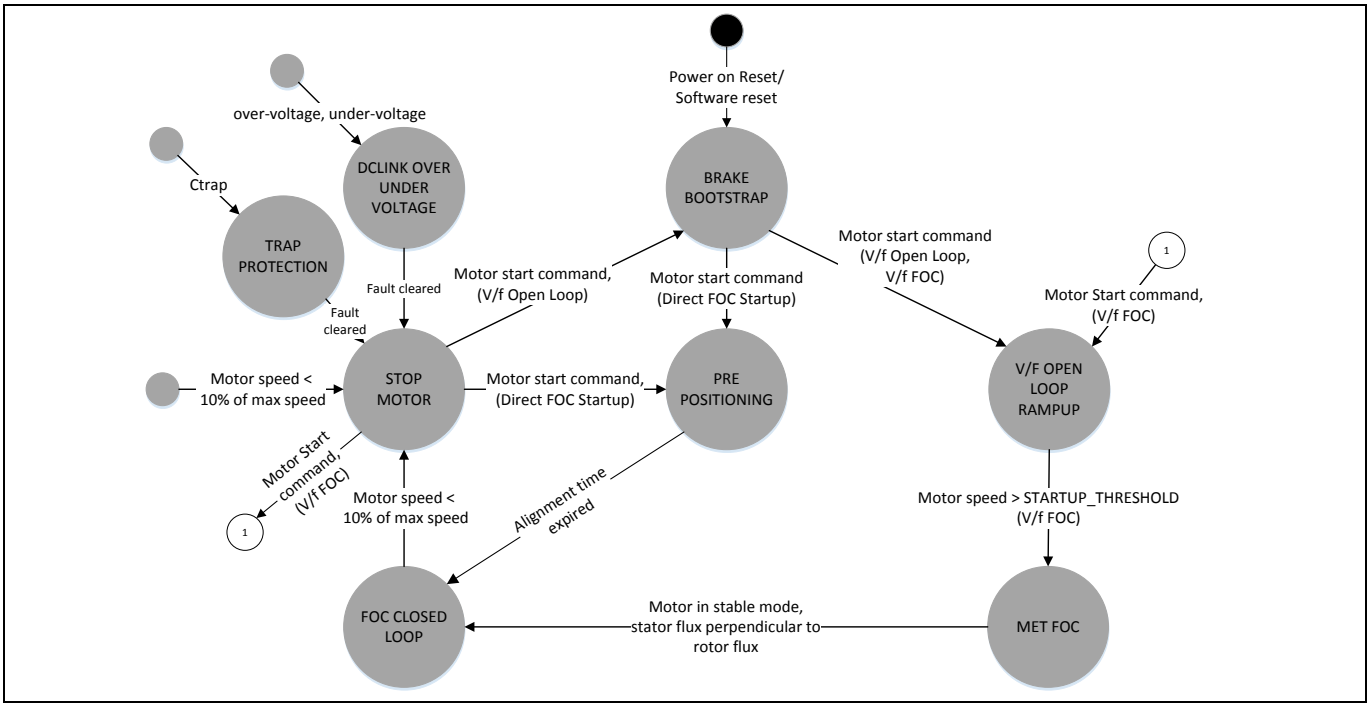


Figure 43 PMSM FOC state machine

For different control schemes, the flow of the state machine is different. Table 23, Table 24 and Table 25, show the states involved and the transition between each state for these control schemes. CONSTANT_SPEED_DIRECT_FOC, CONSTANT_TORQUE_DIRECT_FOC and CONSTANT_VQ_DIRECT_FOC control schemes have the same states and the transition flow.

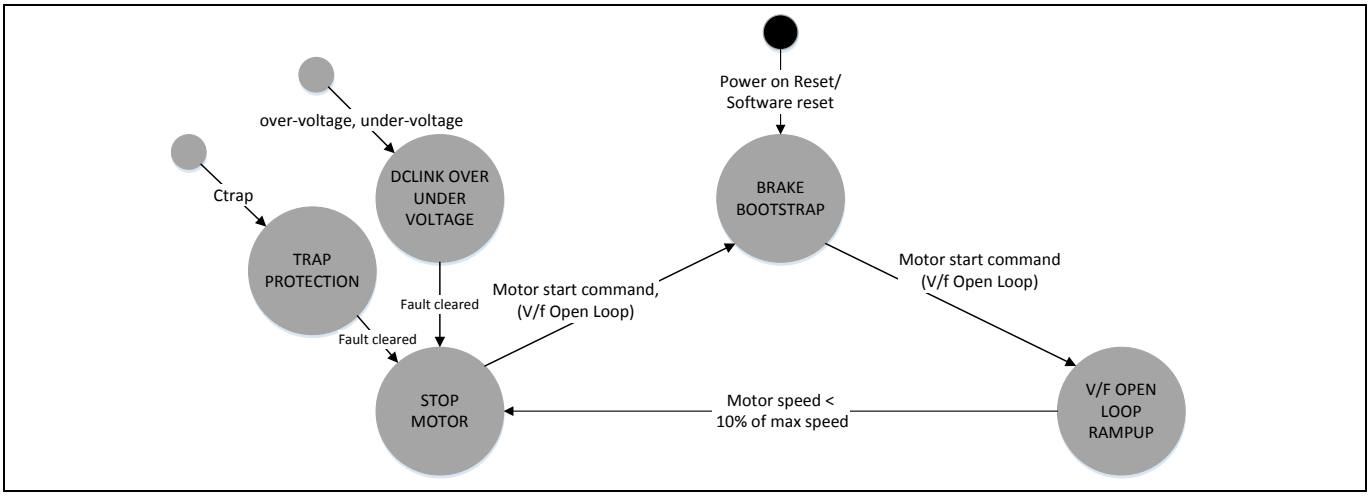


Figure 44 PMSM FOC state machine – open loop V/F only

Table 23 CONSTANT_SPEED_VF_ONLY - Open loop voltage control state transition flow

State	Transition event	Next state
BRAKE_BOOTSTRAP	Motor start command	VF_OPENLOOP_RAMP_UP
VF_OPENLOOP_RAMPUP	Motor speed < 10% of max speed	STOP_MOTOR
	Fault detected – CTrap	TRAP_PROTECTION
	Fault detected – over voltage / under voltage	DCLINK_OVER_UNDER_VOLTAGE
STOP_MOTOR	Motor start command	BRAKE_BOOTSTRAP
TRAP_PROTECTION	Faults are cleared	STOP_MOTOR
DC_LINK_OVER_UNDER_VOLTAGE		

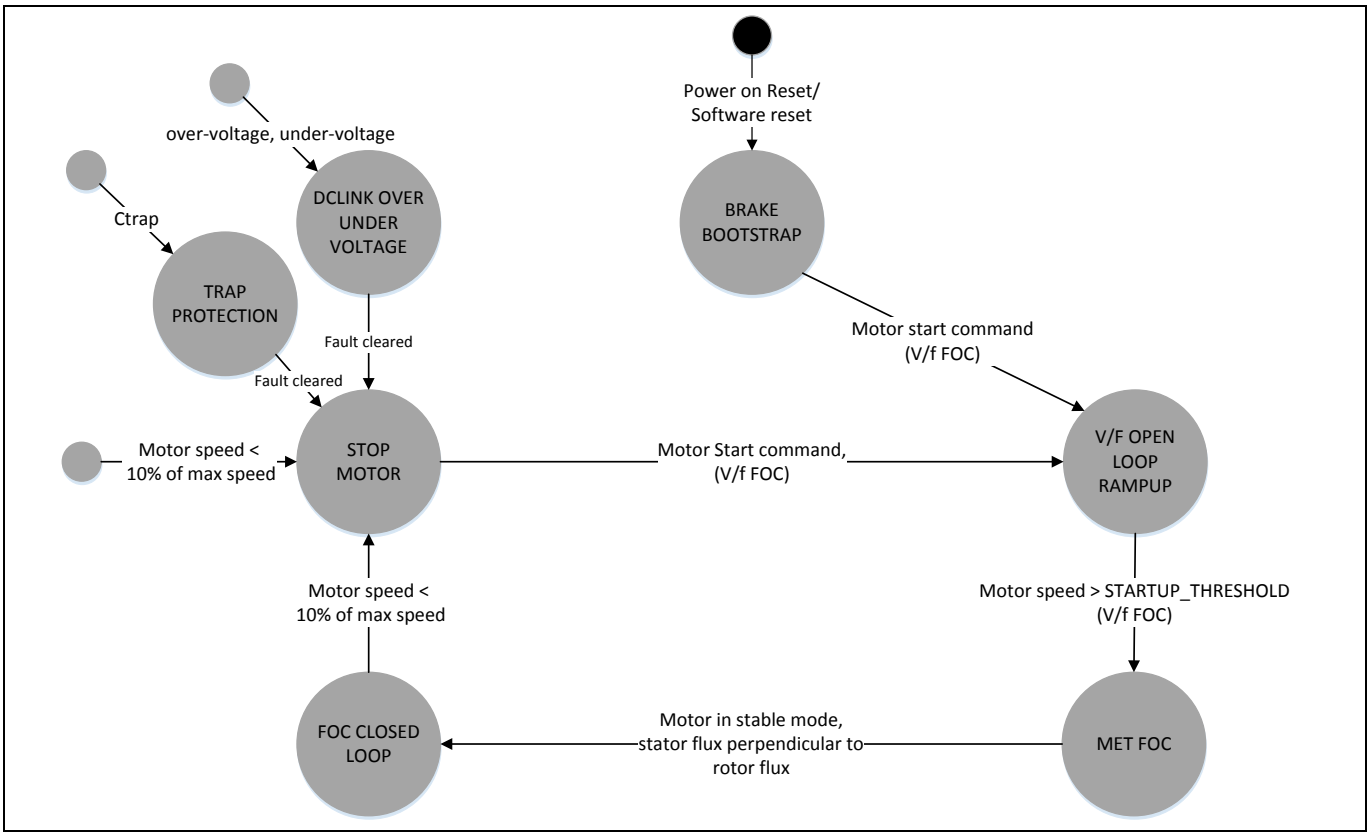


Figure 45 PMSM FOC state machine – Open loop to closed loop

Table 24 CONSTANT_SPEED_VF_MET_FOC - Open loop -> Closed loop control state transition flow

State	Transition event	Next state
BRAKE_BOOTSTRAP	Motor start command	VF_OPENLOOP_RAMP_UP
VF_OPENLOOP_RAMPUP	Motor speed > start-up threshold speed	MET_FOC
MET_FOC	Motor in stable mode, stator flux perpendicular to rotor flux	FOC_CLOSED_LOOP
FOC_CLOSED_LOOP	Motor speed < 10% of max speed	STOP_MOTOR
	Fault detected – CTrap	TRAP_PROTECTION
	Fault detected – over voltage /	DCLINK_OVER_UNDER_VOLTAGE

State	Transition event	Next state
	under voltage	
STOP_MOTOR	Motor start command	VF_OPENLOOP_RAMP_UP
TRAP_PROTECTION	Faults are cleared	STOP_MOTOR
DC_LINK_OVER_UNDER_VOLTAGE		

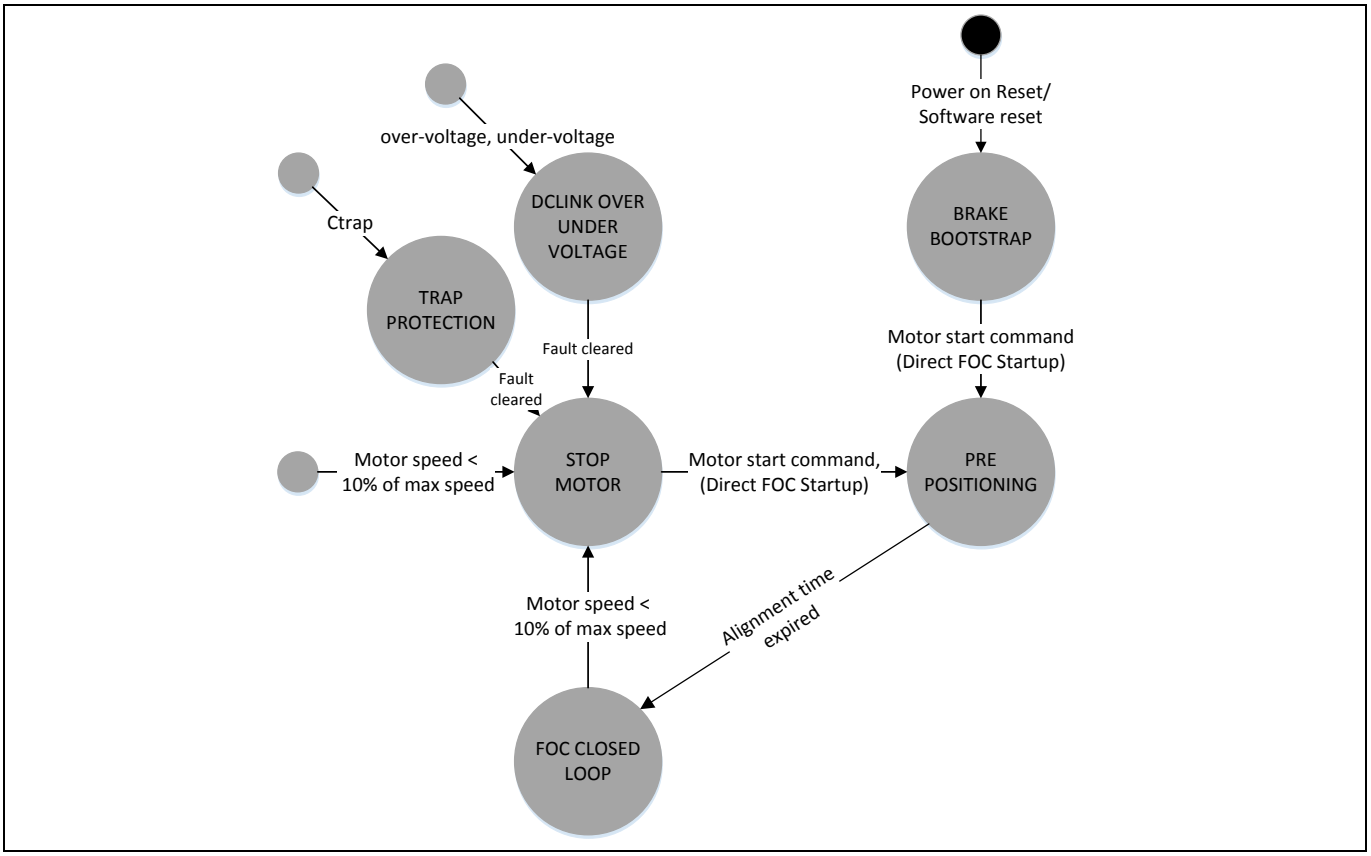


Figure 46 PMSM FOC state machine – Direct FOC startup

Table 25 CONSTANT_SPEED_DIRECT_FOC - DIRECT FOC startup control state transition flow

State	Transition event	Next state
BRAKE_BOOTSTRAP	Motor start command	PRE_POSITIONING
PRE_POSITIONING	Alignment time expired	FOC_CLOSED_LOOP
FOC_CLOSED_LOOP	Motor speed < 10% of max speed	STOP_MOTOR
	Fault detected – CTrap	TRAP_PROTECTION
	Fault detected – over voltage / under voltage	DCLINK_OVER_UNDER_VOLTAGE
TRAP_PROTECTION	Faults are cleared	STOP_MOTOR
DC_LINK_OVER_UNDER_VOLTAGE		
STOP_MOTOR	Motor start command	PRE_POSITIONING

7 Configuration

User motor configuration and FOC control configuration are stored in the pmsm_foc_user_parameter.h header file. XMC™ hardware modules configuration (pinout, resources) is in the pmsm_foc_mcuhwconfig.h. These two header files can be found in the folder PMSM_FOC\Configuration\. PI gain and parameters for each control loop can be configured in the pmsm_foc_pi.h file which is under the folder PMSM_FOC\ControlModules\.

7.1 User motor configuration

These settings are stored in the file: pmsm_foc_user_parameter.h. The default settings are for the maxon motor used in the Infineon XMC1000 Motor Control Application Kit, KIT_XMC1X_AK_MOTOR_001.

USER_MOTOR_R_PER_PHASE_OHM

#define USER_MOTOR_R_PER_PHASE_OHM (6.8f)

Define the motor phase to neutral resistance in ohms.

USER_MOTOR_L_PER_PHASE_uH

#define USER_MOTOR_L_PER_PHASE_uH (3865.0f)

Define the motor phase to neutral stator inductance in micro henry. For IPMSM (Interior Permanent Magnet Synchronous Motor) brushless DC motor, q-axis inductance (L_q) of one motor phase is used.

USER_MOTOR_POLE_PAIR

#define USER_MOTOR_POLE_PAIR (4U)

Number of pole pairs in the motor. It is used to calculate the electrical RPM of the rotor.

USER_SPEED_HIGH_LIMIT_RPM

#define USER_SPEED_HIGH_LIMIT_RPM (4530.0f)

Set the upper limit on speed reference for speed control in RPM.

USER_SPEED_LOW_LIMIT_RPM

#define USER_SPEED_LOW_LIMIT_RPM (USER_SPEED_HIGH_LIMIT_RPM/30)

Set the lower limit on speed reference for speed control in RPM.

USER_SPEED_RAMPUP_RPM_PER_S

#define USER_SPEED_RAMPUP_RPM_PER_S (500U)

Used in ramp generation function. Define the ramp up rate in FOC closed loop, RPM/sec.

USER_SPEED_RAMPDOWN_RPM_PER_S

#define USER_SPEED_RAMPDOWN_RPM_PER_S (500U)

Used in ramp generation function. Define the ramp down rate in FOC closed loop, RPM/sec.

USER_STARTUP_SPEED_RPM

#define USER_STARUP_SPEED_RPM (0U)

Define the initial speed for V/f open loop in RPM.

USER_STARTUP_SPEED_THRESHOLD_RPM

#define USER_STARTUP_SPEED_THRESHOLD_RPM (500U)

Define the threshold speed to transit from open loop control to closed loop control.

USER_STARTUP_VF_OFFSET_V

#define USER_STARTUP_VF_OFFSET_V (1.0f)

V/f open loop control startup voltage offset.

USER_STARTUP_VF_SLEWRATE_V_PER_HZ

#define USER_STARTUP_VF_SLEWRATE_V_PER_HZ (0.1f)

V/f open loop control startup slew rate in volts per Hz.

7.2 FOC control configuration

The file, pmsm_foc_user_parameter.h, also contains the user configuration for FOC control, current and voltage sensing. The default settings are for the maxon motor used in the Infineon XMC1000 Motor Control Application Kit, KIT_XMC1X_AK_MOTOR_001.

7.2.1 System configuration

PMSM_FOC_HARDWARE_BOARD

#define PMSM_FOC_HARDWARE_BOARD KIT_XMC1X_AK_MOTOR_001

Specify the hardware board used.

Options:

- KIT_XMC1X_AK_MOTOR_001 - Infineon XMC1000 Motor Control Application Kit
- KIT_XMC750WATT_MC_AK_V1 – XMC 750Watt Motor Control Application Kit
- IFX_XMC_LVPB_R2 – IFX Demo board revision 2
- IFI_EVAL_24V_250W – IFI 250Watt Evaluation board
- IFX_XMC_LVPB_R3 – IFX Demo board revision 3

MOTOR_TYPE

#define MOTOR_TYPE MAXON_MOTOR

Specify the motor used.

Options:

- MAXON_MOTOR – The motor that is provided together with the Infineon XMC1000 Motor Control Application Kit
- NANOTEC_MOTOR – The motor that is provided in the Infineon XMC4000 Motor Control Application Kit

Note: Infineon XMC4000 Motor Control Application Kit is not supported in the current version of the PMSM_FOC software.

CURRENT_SENSING

#define CURRENT_SENSING USER_THREE_SHUNT_SYNC_CONV

Define the current sensing technique used.

Options:

- USER_SINGLE_SHUNT_CONV – Single shunt current sensing technique with Pseudo Zero Vector PWM generation, refer to chapter 3.1.1
- USER_THREE_SHUNT_SYNC_ASSYNC_CONV – Three shunt current sensing technique with ADC standard conversion, refer to chapter 3.1.2
- USER_THREE_SHUNT_SYNC_CONV – Three shunt current sensing technique with ADC synchronous conversion, refer to chapter 3.1.2.1

MY_FOC_CONTROL_SCHEME

#define MY_FOC_CONTROL_SCHEME CONSTANT_SPEED_DIRECT_FOC

Define the FOC control scheme.

Options:

- CONSTANT_SPEED_DIRECT_FOC – Direct FOC start-up using speed control, refer to chapter 2.3.2
- CONSTANT_SPEED_VF_ONLY – Open loop speed control, refer to chapter 2.3.1
- CONSTANT_SPEED_VF_MET_FOC – Open loop start-up to MET to closed loop FOC speed control, refer to chapter 2.3.2.1
- CONSTANT_TORQUE_DIRECT_FOC – Direct FOC start-up using torque control, refer to chapter 2.3.3
- CONSTANT_VQ_DIRECT_FOC – Direct FOC start-up using voltage torque control, refer to chapter 2.3.4

VDC_UNDER_OVERVOLTAGE_PROTECTION

#define VDC_UNDER_OVERVOLTAGE_PROTECTION ENABLED

Enable or disable the DC link voltage protection.

OVERCURRENT_PROTECTION

#define OVERCURRENT_PROTECTION ENABLED

Enable or disable DC link current protection.

INTERNAL_OP_GAIN

#define INTERNAL_OP_GAIN DISABLED

Enable or disable internal ADC channel gain factor. If enabled, configure the OP_GAIN_FACTOR to XMC13 built in gain factor value (1, 3, 6, and 12).

OP_GAIN_FACTOR

```
#if(INTERNAL_OP_GAIN == ENABLED)
#define OP_GAIN_FACTOR                    (3U)
#else
#define OP_GAIN_FACTOR                    G_OPAMP_PER_PHASEMEASUREMENT
#endif
```

If internal ADC channel gain factor is enabled, the definition of the OP_GAIN_FACTOR is according to the XMC13 build in gain factor: 1, 3, 6, and 12.

If ADC channel gain factor is disabled, the OP_GAIN_FACTOR is the gain of the external op-amp used to amplify the phase current signals, G_OPAMP_PER_PHASEMEASUREMENT, refer to chapter 7.2.2.

USER_VDC_LINK_V

```
#define USER_VDC_LINK_V                   (24.0f)
```

Configure this macro according to the DC voltage used in the motor power board. The value is in volts.

USER_DEAD_TIME_US

```
#define USER_DEAD_TIME_US                 (0.75f)
```

This is the dead time configuration of the CCU80 complementary PWM outputs for the high-side and low-side switches. It is in terms of microseconds.

USER_CCU8_PWM_FREQ_HZ

```
#define USER_CCU8_PWM_FREQ_Hz             (20000U)
```

This macro defines the PWM frequency in Hz. The PMSM FOC software can support up to 25 kHz, (maximum 30 kHz in some cases).

USER_BOOTSTRAP_PRECHARGE_TIME_MS

```
#define USER_BOOTSTRAP_PRECHARGE_TIME_MS  (20U)
```

This is the initial bootstrap capacitor precharging time in milliseconds.

USER_ROTOR_PREPOSITION_TIME_MS

```
#define USER_ROTOR_PREPOSITION_TIME_MS    (100)
```

Time in milliseconds, for motor pre-positioning.

USER_IDC_MAXCURRENT_A

```
#define USER_IDC_MAXCURRENT_A             (10.0f)
```

This setting is the maximum DC link current limit. This limit is checked if overcurrent protection feature is enabled. Once this limit is hit, the reference speed is reduced. Refer to session on overcurrent protection in chapter 2.9.

VDC_MAX_LIMIT

#define VDC_MAX_LIMIT ((VADC_DCLINK * 19U)>>4)

Set the maximum DC link voltage limit for ramp down operation. Default setting is 18.7% more than nominal DC link voltage. User should change this limit according to their hardware design.

7.2.2 Power board configuration for current and voltage sensing

USER_DC_LINK_DIVIDER_RATIO

#define USER_DC_LINK_DIVIDER_RATIO (5.1f/(5.1f+47.0f))

Refer to chapter 2.6, the DC link voltage divider ratio is R2/(R1+R2)

USER_R_SHUNT_OHM

#define USER_R_SHUNT_OHM (0.05f)

Phase current shunt resistance in ohms.

USER_DC_SHUNT_OHM

#define USER_DC_SHUNT_OHM (0.05f)

This is the DC link current shunt resistance in ohms. This is used for single shunt current sensing technique.

USER_RIN_PHASECURRENT_KOHM

#define USER_RIN_PHASECURRENT_KOHM (1.0f)

This is the RIN resistor value in the phase current amplifier, Figure 47. The unit is in kiloohms.

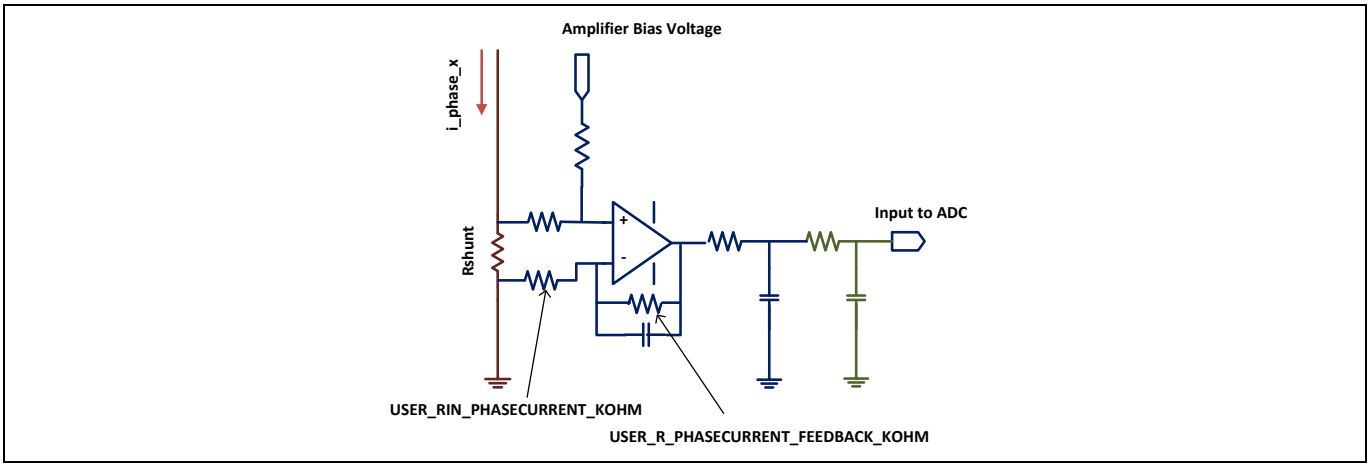


Figure 47 External phase current amplifier

USER_R_PHASECURRENT_FEEDBACK_KOHM

#define USER_R_PHASECURRENT_FEEDBACK KOHM (16.4f)

This is the feedback resistor value in the phase current amplifier. The unit is in kiloohms. With the USER_RIN_PHASECURRENT_KOHM, the gain of the current amplifier is calculated.

G_OPAMP_PER_PHASEMEASUREMENT

#define G_OPAMP_PER_PHASEMEASUREMENT (USER_R_PHASECURRENT_FEEDBACK KOHM/ USER_RIN_PHASECURRENT_KOHM)

This is phase current amplifier gain.

USER_RIN_DCCURRENT_KOHM

#define USER_RIN_DCCURRENT_KOHM (10.0f)

This is the RIN resistor value in the DC link current amplifier. The unit is in kiloohms.

USER_R_DCCURRENT_FEEDBACK_KOHM

#define USER_R_DCCURRENT_FEEDBACK KOHM (75.0f)

This is the feedback resistor value in the DC link current amplifier. The unit is in kiloohms. With the USER_RIN_DCCURRENT_KOHM, the gain of the DC current amplifier is calculated.

7.2.3 FOC control scheme – Open loop to closed loop control configuration

USER_MET_THRESHOLD_HIGH

#define USER_MET_THRESHOLD_HIGH (64U)

Define the high threshold limit for speed hysteresis control in MET motor state.

USER_MET_THRESHOLD_LOW

#define USER_MET_THRESHOLD_LOW (16U)

Define the low threshold limit for speed hysteresis control in MET motor state.

USER_MET_LPF

#define USER_MET_LPF (2U)

Low pass filter factor for the MET threshold calculation, Y[n] = Y[n-1] + (X[n] – Y[n-1]) >> USER_MET_LPF

7.2.4 FOC control scheme – Direct torque FOC control configuration

USER_IQ_CURRENT_ALLOWED_A

#define USER_IQ_CURRENT_ALLOWED_A (0.05f)

Set the high limit of the reference torque current in ampere.

USER_IQ_REF_LOW_LIMIT

#define USER_IQ_REF_LOW_LIMIT (0U)

Set the low limit of the reference torque current.

USER_IQ_REF_HIGH_LIMIT

#define USER_IQ_REF_HIGH_LIMIT (32768* USER_IQ_CURRENT_ALLOWED_A/I_MAX_A)

Scale the high limit of the reference torque current in 1Q15 format.

USER_IQ_RAMPUP

#define USER_IQ_RAMPUP (10U)

Torque current ramp up steps used in linear ramp generator. 1 step is (1/32768) * I_MAX_A, where I_MAX_A is equal to (5.0 V/(USER_R_SHUNT_OHM * OP_GAIN_FACTOR)) / 2.

USER_IQ_RAMPDOWN

#define USER_IQ_RAMPDOWN (10U)

Torque current ramp down steps used in linear ramp generator. 1 step is (1/32768) * I_MAX_A, where I_MAX_A is equal to (5.0 V/(USER_R_SHUNT_OHM * OP_GAIN_FACTOR)) / 2.

USER_IQ_RAMP_SLEWRATE

#define USER_IQ_RAMP_SLEWRATE (50U)

Define the frequency to ramp up/ramp down I_q. In every USER_IQ_RAMP_SLEWRATE *PWM cycles, ramp up I_q by USER_IQ_RAMPUP steps, or ramp down I_q by USER_IQ_RAMPDOWN step.

7.2.5 FOC control scheme – Direct Vq FOC control configuration

USER_VQ_VOLTAGE_ALLOWED_V

#define USER_VQ_VOLTAGE_ALLOWED_V (12U)

Set the limit of the torque voltage in volts. This value must be less than VREF_MAX_V.

USER_VQ_REF_LOW_LIMIT

#define USER_VQ_REF_LOW_LIMIT (0U)

Set the low limit of the reference torque voltage.

USER_VQ_REF_HIGH_LIMIT

#define USER_VQ_REF_HIGH_LIMIT (32768* USER_VQ_VOLTAGE_ALLOWED_V/VREF_MAX_V)

Scale the limit of the reference torque voltage to 1Q15 format. VREF_MAX_V is defined as DC link voltage divided by square root of 3.

USER_VQ_RAMPUP

#define USER_VQ_RAMPUP (2U)

Define the number of steps to ramp up the torque voltage in the linear ramp generator.

USER_VQ_RAMPDOWN

#define USER_VQ_RAMPDOWN (2U)

Define the number of steps to ramp down the torque voltage in the linear ramp generator.

USER_VQ_RAMP_SLEWRATE

#define USER_VQ_SLEWRATE (10U)

Define the frequency to ramp up/ramp down V_q. In every USER_VQ_RAMP_SLEWRATE * PWM cycles, ramp up V_q by USER_VQ_RAMPUP steps, or ramp down V_q by USER_VQ_RAMPDOWN step.

7.3 XMC™ hardware modules configuration

The following hardware configurations are on the base on the XMCTM hardware resources usage listed in Table 2. The default settings are for the Infineon XMC1000 Motor Control Application Kit, KIT_XMC1X_AK_MOTOR_001.

7.3.1 GPIO configuration

TRAP_PIN

#define TRAP_PIN P0_12

External CCU80 CTrap input pin assignment.

INVERTER_PIN

#define INVERTER_PIN P0_11

Inverter output pin assignment. This pin connects to the enable pin of the inverter switch/gate drivers.

INVERTER_ENABLE_PIN

#define INVERTER_ENABLE_PIN (1)

This macro defines the logic of the INVERTER_PIN.

Options:

- 1 – Active high
- 0 – Active low

PHASE_U_HS_PIN

#define PHASE_U_HS_PIN P0_0

CCU80 PWM Phase U high side output pin assignment.

PHASE_U_HS_ALT_SELECT

#define PHASE_U_HS_ALT_SELECT XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5

XMC1302 alternate output pin register value setting. CCU80 PWM output is selected. Please refer to XMC1302 reference manual [1] for the alternate output setting.

Configuration

PHASE_U_LS_PIN

#define PHASE_U_LS_PIN P0_1

CCU80 PWM Phase U low side output pin assignment.

PHASE_U_LS_ALT_SELECT

#define PHASE_U_LS_ALT_SELECT XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5

XMC1302 alternate output pin register value setting. CCU80 PWM output is selected.

PHASE_V_HS_PIN

#define PHASE_V_HS_PIN P0_7

CCU80 PWM Phase V high side output pin assignment.

PHASE_V_HS_ALT_SELECT

#define PHASE_V_HS_ALT_SELECT XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5

XMC1302 alternate output pin register value setting. CCU80 PWM output is selected.

PHASE_V_LS_PIN

#define PHASE_V_LS_PIN P0_6

CCU80 PWM Phase V low side output pin assignment.

PHASE_V_LS_ALT_SELECT

#define PHASE_V_LS_ALT_SELECT XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5

XMC1302 alternate output pin register value setting. CCU80 PWM output is selected.

PHASE_W_HS_PIN

#define PHASE_W_HS_PIN P0_8

CCU80 PWM Phase W high side output pin assignment.

PHASE_W_HS_ALT_SELECT

#define PHASE_W_HS_ALT_SELECT XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5

XMC1302 alternate output pin register value setting. CCU80 PWM output is selected.

PHASE_W_LS_PIN

#define PHASE_W_LS_PIN P0_9

CCU80 PWM Phase W low side output pin assignment.

PHASE_W_LS_ALT_SELECT

#define PHASE_W_LS_ALT_SELECT XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5

XMC1302 alternate output pin register value setting. CCU80 PWM output is selected.

7.3.2 CCU80 resources configuration

CCU8_MODULE

#define CCU8_MODULE CCU80

Assign XMC1302 CCU8 module for SVM generation.

CCU8_MODULE_PHASE_U

#define CCU8_MODULE_PHASE_U CCU80_CC80

Assign CCU8 module timer slice for phase U PWM generation.

CCU8_MODULE_PHASE_V

#define CCU8_MODULE_PHASE_V CCU80_CC81

Assign CCU8 module timer slice for phase V PWM generation.

CCU8_MODULE_PHASE_W

#define CCU8_MODULE_PHASE_W CCU80_CC82

Assign CCU8 module timer slice for phase W PWM generation.

CCU8_MODULE_ADC_TR

#define CCU8_MODULE_ADC_TR CCU80_CC83

Assign CCU8 module timer slice for ADC conversion trigger.

CCU8_PASSIVE_LEVEL

#define CCU8_PASSIVE_LEVEL XMC_CCU8_SLICE_OUTPUT_PASSIVE_LEVEL_LOW

Set the CCU8 passive signal logic level.

CCU8_INPUT_TRAP_LEVEL

#define CCU8_INPUT_TRAP_LEVEL XMC_CCU8_SLICE_EVENT_LEVEL_SENSITIVITY_ACTIVE_LOW

Define the CCU8 input trap signal logic level

7.3.3 ADC resource configuration

Macro definitions for 3-phase currents sensing using ADC asynchronous conversion:

VADC_IU_GROUP

#define VADC_IU_GROUP VADC_G1

Define the VADC register pointer for phase U current sensing.

VADC_IU_GROUP_NO

#define VADC_IU_GROUP_NO (1U)

This is the numeric representation of VADC group for phase U current.

Configuration

VADC_IU_CHANNEL

#define VADC_IU_CHANNEL (3U)

Assign phase U current to ADC channel according to analog port pin.

VADC_IU_RESULT_REG

#define VADC_IU_RESULT_REG (3U)

Assign ADC result register to store the phase U current conversion result.

VADC_IV_GROUP

#define VADC_IV_GROUP VADC_G1

Define the VADC register pointer for phase V current sensing.

VADC_IV_GROUP_NO

#define VADC_IV_GROUP_NO (1U)

This is the numeric representation of VADC group for phase V current.

VADC_IV_CHANNEL

#define VADC_IV_CHANNEL (2U)

Assign phase V current to ADC channel according to analog port pin.

VADC_IV_RESULT_REG

#define VADC_IV_RESULT_REG (2U)

Assign ADC result register to store the phase V current conversion result.

VADC_IW_GROUP

#define VADC_IW_GROUP VADC_G1

Define the VADC register pointer for phase W current sensing.

VADC_IW_GROUP_NO

#define VADC_IW_GROUP_NO (1U)

This is the numeric representation of VADC group for phase W current.

VADC_IW_CHANNEL

#define VADC_IW_CHANNEL (4U)

Assign phase W current to ADC channel according to analog port pin.

VADC_IW_RESULT_REG

#define VADC_IW_RESULT_REG (4U)

Assign ADC result register to store the phase W current conversion result.

Configuration

Macro definitions for 3-phase currents sensing using ADC synchronous conversion:

Please refer to Table 14 and Table 15 for the assignment of the ADC channels for Infineon XMC1000 Motor Control Application Kit.

VADC_IU_G1_CHANNEL

#define VADC_IU_G1_CHANNEL (3U)

Assign phase U current to group 1 ADC channel according to analog port pin.

VADC_IU_G0_CHANNEL

#define VADC_IU_G0_CHANNEL (4U)

Assign phase U current to group 0 ADC channel according to analog port pin.

VADC_IV_G1_CHANNEL

#define VADC_IV_G1_CHANNEL (2U)

Assign phase V current to group 1 ADC channel according to analog port pin.

VADC_IV_G0_CHANNEL

#define VADC_IV_G0_CHANNEL (3U)

Assign phase V current to group 0 ADC channel according to analog port pin.

VADC_IW_G1_CHANNEL

#define VADC_IW_G1_CHANNEL (4U)

Assign phase W current to group 1 ADC channel according to analog port pin.

VADC_IW_G0_CHANNEL

#define VADC_IW_G0_CHANNEL (2U)

Assign phase W current to group 0 ADC channel according to analog port pin.

VADC_G0_CHANNEL_ALIAS0

#define VADC_G0_CHANNEL_ALIAS0 VADC_IV_G0_CHANNEL

Define the initial alias value for VADC group 0 channel 0.

VADC_G0_CHANNEL_ALIAS1

#define VADC_G0_CHANNEL_ALIAS1 VADC_IDC_CHANNEL

Define the initial alias value for VADC group 0 channel 1.

VADC_G1_CHANNEL_ALIAS0

#define VADC_G1_CHANNEL_ALIAS0 VADC_IW_G1_CHANNEL

Define the initial alias value for VADC group 1 channel 0.

Configuration

VADC_G1_CHANNEL_ALIAS1

#define VADC_G1_CHANNEL_ALIAS1 VADC_IU_G1_CHANNEL

Define the initial alias value for VADC group 1 channel 1.

Macro definitions for single shunt current sensing:

VADC_ISS_GROUP

#define VADC_ISS_GROUP VADC_G1

Define the VADC register pointer for single shunt current sensing

VADC_ISS_GROUP_NO

#define VADC_ISS_GROUP_NO (1U)

This is the numeric representation of VADC group for single shunt current.

VADC_ISS_CHANNEL

#define VADC_ISS_CHANNEL (1U)

Assign single shunt current to ADC channel according to analog port pin.

VADC_ISS_RESULT_REG

#define VADC_ISS_RESULT_REG (15U)

Assign ADC result register to store the single shunt current conversion result.

Macro definition for DC link voltage sensing:

VADC_VDC_GROUP

#define VADC_VDC_GROUP VADC_G1

Define the VADC register pointer for DC link voltage sensing

VADC_VDC_GROUP_NO

#define VADC_VDC_GROUP_NO (1U)

This is the numeric representation of VADC group for DC link voltage.

VADC_VDC_CHANNEL

#define VADC_VDC_CHANNEL (5U)

Assign DC link voltage to ADC channel according to analog port pin.

VADC_VDC_RESULT_REG

#define VADC_VDC_RESULT_REG (5U)

Assign ADC result register to store the DC link voltage conversion result.

Configuration

Macro definition for DC link average current sensing:

VADC_IDC_GROUP

#define VADC_IDC_GROUP VADC_G1

Define the VADC register pointer for DC link average current sensing

VADC_IDC_GROUP_NO

#define VADC_IDC_GROUP_NO (1U)

This is the numeric representation of VADC group for DC link average current.

VADC_IDC_CHANNEL

#define VADC_IDC_CHANNEL (6U)

Assign DC link average current to ADC channel according to analog port pin.

VADC_IDC_RESULT_REG

#define VADC_IDC_RESULT_REG (6U)

Assign ADC result register to store the DC link average current conversion result.

Marco definition for reference speed/torque sensing using potentiometer:

VADC_POT_GROUP

#define VADC_POT_GROUP VADC_G1

Define the VADC register pointer for potentiometer sensing

VADC_POT_GROUP_NO

#define VADC_POT_GROUP_NO (1U)

This is the numeric representation of VADC group for potentiometer sensing

VADC_POT_CHANNEL

#define VADC_POT_CHANNEL (7U)

Assign potentiometer to ADC channel according to analog port pin.

VADC_POT_RESULT_REG

#define VADC_POT_RESULT_REG (7U)

Assign ADC result register to store the potentiometer conversion result.

7.3.4 Motor operation control configuration

UART_ENABLE

#define UART_ENABLE USIC0_CH1_P1_2_P1_3

Define the motor start/stop/speed change, operations control.

Options:

- USIC_DISABLE_ALL – UART communication disabled and motor operations control via µC/Probe is also disabled. The potentiometer is used to control motor operation via the ADC pin.
- USIC0_CH0_P1_4_P1_5 – UART channel 0 used to receive commands to control motor operations. Port pins P1.4 and P1.5 are configured to receive and transmit data. Motor operations control via µC/Probe is also enabled. ADC potentiometer result is discarded.
- USIC0_CH1_P1_2_P1_3 – UART channel 1 used to receive commands to control motor operations. Port pins P1.2 and P1.5 are configured to transmit and receive data. Motor operations control via µC/Probe is also enabled. ADC potentiometer result is discarded.

7.4 PI configuration

7.4.1 Determination of flux and torque current PI gains

To calculate the initial values of the PI gains of the torque and flux current control, it is required to know the electrical parameters of the motor. For SPMSM motor, the torque inductance and the flux inductance are considered equal.

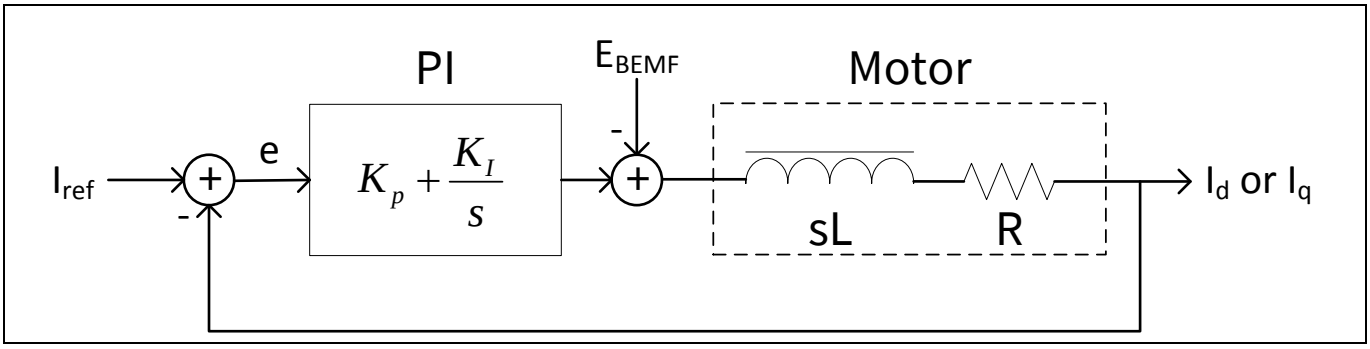


Figure 48 Torque / Flux current control loop

The calculation of the PI gains is done by using the pole-zero cancellation technique as illustrated. By having $K_P/K_I = L/R$, the controller zero will cancel the motor pole. With this, the transfer function of the control loop is a first order LPF with time constant, T_c. In addition, the proportional gain calculation is based on motor inductance and the integral gain is on the motor resistance.

At constant motor speed, the Back-EMF of the motor is near constant. Therefore it is negligible in the frequency domain. The Figure 49 shows the simplified diagram after pole-zero cancellation.

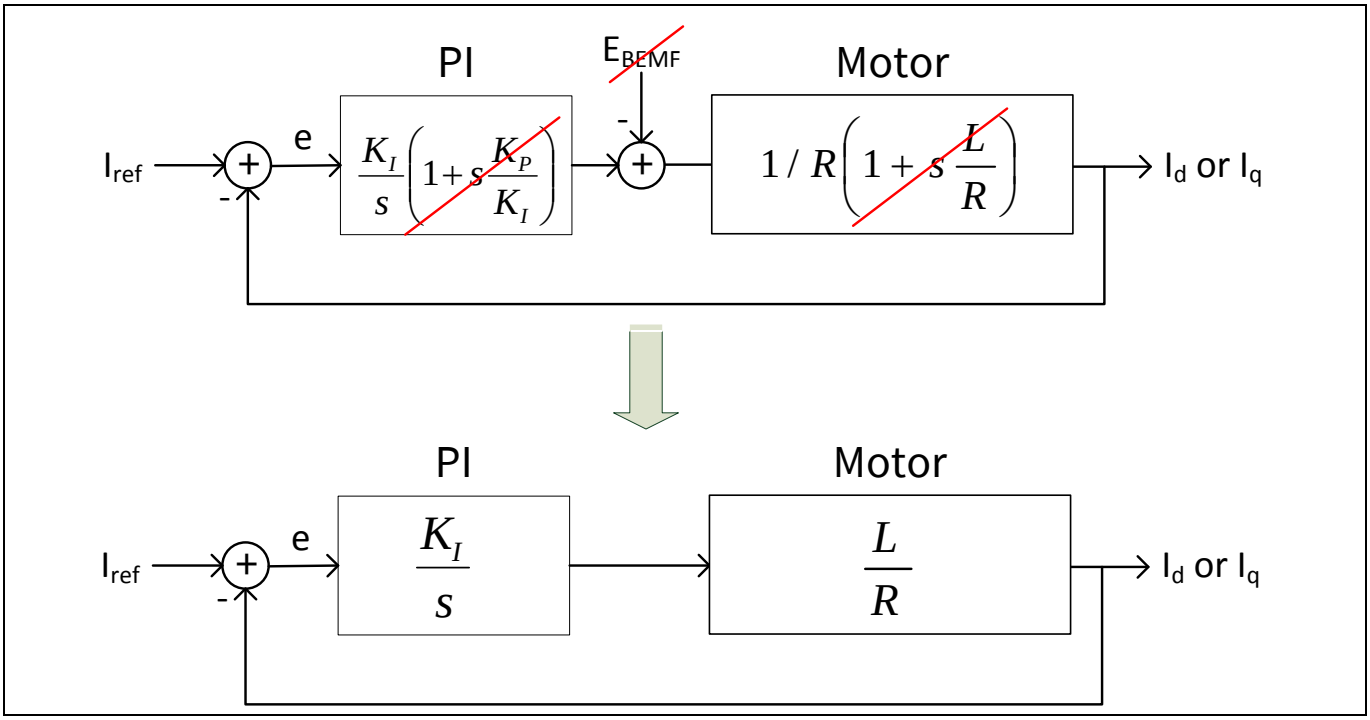


Figure 49 Simplified current control loop due to pole-zero cancellation

As $K_P/L = K_I/R = \omega_c$, the PI controller gains are:

Proportional Gain $K_P = \omega_c L$

Integral Gain $K_I = \omega_c R$

Where ω_c is the cutoff frequency of the first order LPF; L is the motor inductance, and R is the motor resistance.

In the digital controller implementation, the integral part is a digital accumulator. Therefore the K_I gain has to include a scaling factor for the sampling time, T_S, which is the PWM frequency.

Revised formula:

Proportional Gain $K_P = \omega_c L \times A$

Integral Gain $K_I = \omega_c R \times T_S = RT_S K_P/L$

Where A is the XMC hardware optimize scaling factor.

Based on the past experience, set the cutoff frequency to three times of the maximum electrical motor speed to obtain a good tradeoff between dynamic response and sensitivity to the measurement noise.

7.4.2 PI settings

The file pmsm_foc_pi.h, contains the configuration for PI controllers. The default settings are for the maxon motor used in the Infineon XMC1000 Motor Control Application Kit, KIT_XMC1X_AK_MOTOR_001.

PI settings for the speed controller:

PI_SPEED_KP

#define PI_SPEED_KP (1U << 15)

Configure proportional gain for speed control block. Minimum value is 1, maximum is 32767.

PI_SPEED_KI

#define PI_SPEED_KI (3U)

Configure integral gain for speed control block. Minimum value is 0, maximum is 32767.

PI_SPEED_SCALE_KPKI

#define PI_SPEED_SCALE_KPKI (10U + USER_RES_INC)

Configure scaling factor of K_P and K_I. The K_P and K_I are scale up by 2^ PI_SPEED_SCALE_KPKI. The USER_RES_INC is a constant defined in pmsm_foc_feature_config.h file. The maximum value of PI_SPEED_SCALEKPKI is 15, minimum value is USER_RES_INC.

PI_SPEED_IK_LIMIT_MIN

#define PI_SPEED_IK_LIMIT_MIN (-(((1<<15)*3)>>2))

Configure minimum output value of the integral buffer. Minimum value is -32768

PI_SPEED_IK_LIMIT_MAX

#define PI_SPEED_IK_LIMIT_MAX (((1<<15)*3)>>2)

Configure maximum output value of the integral buffer. Maximum value is 32767

PI_SPEED_UK_LIMIT_MIN

#define PI_SPEED_UK_LIMIT_MIN (16U)

Configure minimum output value of the speed PI control block. Minimum value is -32768.

PI_SPEED_UK_LIMIT_MAX

#define PI_SPEED_UK_LIMIT_MAX (32767U)

Configure maximum output value of the speed PI control block. Maximum value is 32767.

Configuration

PI settings for the torque controller:

PI_TORQUE_KP

#define PI_TORQUE_KP (USER_DEFAULT_IQID_KP)

Configure proportional gain for torque control block. The gain value is determined by the equation, $K_P = \omega_c L \times A$ as described in chapter 7.4.1.

PI_TORQUE_KI

#define PI_TORQUE_KI (USER_DEFAULT_IQID_KI)

Configure integral gain for torque control block. The gain value is determined by the equation, $K_I = RT_S K_P / L$ as described in chapter 7.4.1.

PI_TORQUE_SCALE_KPKI

#define PI_TORQUE_SCALE_KPKI (SCALING_CURRENT_KPKI)

Configure scaling factor of K_P and K_I. The K_P and K_I are scale up by 2^ PI_TORQUE_SCALE_KPKI.

PI_TORQUE_IK_LIMIT_MIN

#define PI_TORQUE_IK_LIMIT_MIN (-32768)

Configure minimum output value of the integral buffer. Minimum value is -32768.

PI_TORQUE_IK_LIMIT_MAX

#define PI_TORQUE_IK_LIMIT_MAX (32767)

Configure maximum output value of the integral buffer. Maximum value is 32767.

PI_TORQUE_UK_LIMIT_MIN

#define PI_TORQUE_UK_LIMIT_MIN (-32768)

Configure minimum output value of the torque PI control block. Minimum value is -32768.

PI_TORQUE_UK_LIMIT_MAX

#define PI_TORQUE_UK_LIMIT_MAX (32767)

Configure maximum output value of the torque PI control block. Maximum value is 32767.

PI settings for the flux controller:

PI_FLUX_KP

#define PI_FLUX_KP (USER_DEFAULT_IQID_KP)

Configure proportional gain for flux control block. The gain value is determined by the equation, $K_P = \omega_c L \times A$ as described in chapter 7.4.1.

Configuration

PI_FLUX_KI

#define PI_FLUX_KI (USER_DEFAULT_IQID_KI >> 0)

Configure integral gain for flux control block. The gain value is determined by the equation, $K_I = RT_S K_P / L$ as described in chapter 7.4.1.

PI_FLUX_SCALE_KPKI

#define PI_FLUX_SCALE_KPKI (SCALING_CURRENT_KPKI + 0)

Configure scaling factor of K_P and K_I. The K_P and K_I are scale up by 2^ PI_FLUX_SCALE_KPKI.

PI_FLUX_IK_LIMIT_MIN

#define PI_FLUX_IK_LIMIT_MIN (-32768)

Configure minimum output value of the integral buffer. Minimum value is -32768.

PI_FLUX_IK_LIMIT_MAX

#define PI_FLUX_IK_LIMIT_MAX (32767)

Configure maximum output value of the integral buffer. Maximum value is 32767.

PI_FLUX_UK_LIMIT_MIN

#define PI_FLUX_UK_LIMIT_MIN (-32768)

Configure minimum output value of the flux PI control block. Minimum value is -32768.

PI_FLUX_UK_LIMIT_MAX

#define PI_FLUX_UK_LIMIT_MAX (32767)

Configure maximum output value of the flux PI control block. Maximum value is 32767.

PI settings for the PLL Estimator controller:

PI_PLL_KP

#define PI_PLL_KP (1U << 8)

Configure proportional gain for PLL Estimator control block. Minimum value is 1, maximum is 32767.

PI_PLL_KI

#define PI_PLL_KI (1U << 6)

Configure integral gain for PLL Estimator control block. Minimum value is 0, maximum is 32767.

PI_PLL_SCALE_KPKI

#define PI_PLL_SCALE_KPKI (19U – USER_RES_INC)

Configure scaling factor of K_P and K_I. The K_P and K_I are scale up by 2^ PI_PLL_SCALE_KPKI. The USER_RES_INC is a constant defined in pmsm_foc_feature_config.h file. The minimum value of PI_PLL_SCALEKPKI is 15.

Configuration

PI_PLL_IK_LIMIT_MIN

#define PI_PLL_IK_LIMIT_MIN (-(1 << (30 - PI_PLL_SCALE_KPKI)))

Configure minimum output value of the integral buffer. Minimum value is -32768.

PI_PLL_IK_LIMIT_MAX

#define PI_PLL_IK_LIMIT_MAX (1 << (30 - PI_PLL_SCALE_KPKI))

Configure maximum output value of the integral buffer. Maximum value is 32767.

PI_PLL_UK_LIMIT_MIN

#define PI_PLL_UK_LIMIT_MIN (SPEED_LOW_LIMIT >> 4)

Configure minimum output value of the PLL Estimator PI control block. Minimum value is -32768.

PI_PLL_UK_LIMIT_MAX

#define PI_PLL_UK_LIMIT_MAX (SPEED_HIGH_LIMIT + SPEED_LOW_LIMIT)

Configure maximum output value of the PLL Estimator PI control block. Maximum value is 32767.

8 PMSM FOC software data structure

8.1 FOC control module input data structure

This data structure defines the inputs variables used in the FOC functions.

Table 26 FOCInput data structure

Category (Structure)	Variable name	Data type / Range	Description	Remark
FOCInput	Phase_L	signed 32-bit	Motor inductance per phase	Motor parameters Initialize once only
	Phase_R	signed 32-bit	Motor resistance per phase	
	Phase_L_Scale	16-bit / 8 - 18	Scaling for inductance; Real inductance in SW = Phase_L/(2 $^{Phase\_L\_Scale}$)	
	CCU8_Period	16-bit	CCU8 period. CCU8 PWM 1KHz to 90KHz	Initialize once only
	Res_Inc	16-bit / 0 – 8	Resolution increase, SW uses (16+Res_Inc) bit to represent 360°	
	LPF_N_BEMF	16-bit / 0 - 8	LPF factor for BEMF	
	Threshold	32-bit	BEMF threshold for transitions to FOC	
	Threshold_LOW	16-bit / 0 – 512	LOW BEMF threshold, for transitions to FOC	
	Threshold_HIGH	16-bit / 0 – 512	HIGH BEMF threshold, for transitions to FOC	
	Flag_State	16-bit / 0 or 1	0: Motor to run in FOC 1: Always in transition state	
	BEMF1	signed 32-bit	BEMF of sensorless estimator	
	BEMF2	signed 32-bit	BEMF of sensorless estimator	
	overcurrent_factor	16-bit / 0-4096	Used to reduce motor speed when over current detected	
	Ref_Speed	signed 32-bit	Motor reference speed for Speed PI controller	
	Vq_Flag	16-bit / 0 or 1	0: FOC Vq from Iq PI controller 1: Vq from external	
	Vq	signed 32-bit / 0 - 32767	Vq input from external, e.g.: handler of e-bike	
	Ref_Id	signed 32-bit	Id reference for Id PI controller	Default = 0
	Ref_Iq	signed 32-bit	Reference of Iq PI controller from external	
	Iq_PI_Flag	16-bit / 0 or 1	0: Reference of Iq PI controller from speed PI output 1: Reference of Iq PI controller = Ref_Iq from external	

8.2 FOC control module output data structure

This data structure defines the outputs variables in the FOC functions.

Table 27 FOCOutput data structure

Category (Structure)	Variable name	Data type / Range	Description
FOCOutput	Rotor_PositionQ31	signed 32-bit	Rotor angle feedback, from the sensorless estimator
	Speed_By_Estimator	signed 32-bit	Rotor speed feedback, from the sensorless estimator
	Previous_SVM_SectorNo	16-bit / 0 – 5	SVM sector number in previous PWM cycle
	New_SVM_SectorNo	16-bit / 0 – 5	SVM sector number in current PWM cycle

8.3 FOC control module data type

The table below shows the data structures of the variables used in the FOC control functions.

Table 28 Data structures used in FOC functions

Category (Structure)	Variable name	Data type / Range	Description
Current	I_U	signed 16-bit / 1Q15 data format	Current of Iu current sensing
	I_V	signed 16-bit / 1Q15 data format	Current of Iv current sensing
	I_W	signed 16-bit / 1Q15 data format	Current of Iw current sensing
Clarke_Transform	I_Alpha_1Q31	signed 16-bit / 1Q15 data format	Current I_Alpha, output of Clarke Transform
	I_Beta_1Q31	signed 16-bit / 1Q15 data format	Current I_Beta, output of Clarke Transform
Park_Transform	Id	signed 16-bit / 1Q15 data format	Current Id, output of Park Transform
	Iq	signed 16-bit / 1Q15 data format	Current Iq, output of Park Transform
Car2Polar	Flux_Vd	signed 32-bit	Voltage Vd, output of PI Flux controller
	Torque_Vq	signed 32-bit	Voltage Vq, output of PI Torque controller
	Vref32	32-bit	SVM voltage magnitude of last PWM cycle
	Vref_AngleQ31	signed 32-bit	SVM voltage angle of last PWM cycle
	SVM_Vref16	16-bit	SVM voltage magnitude in open loop control
	SVM_Angle16	16-bit	SVM angle in open loop control

8.4 SVM module data structure

This data structure defines the variables used in the SVM module.

Table 29 Data structure used in SVM module

Category (Structure)	Variable name	Data type / Range	Description	Remark
SVM	CurrentSectorNo	16-bit / 0 – 5	SVM new sector number, 0 to 5 represent sector A to F	
	PreviousSectorNo	16-bit / 0 – 5	To keep track of sector number in last PWM cycle	
	Flag_3or2_ADC	16-bit / 0 or 0xBB	To indicate using 2 or 3 ADC results of phase currents for current construction 0: USE ALL ADC 0xBB: USE 2 ADC	For three shunt current sensing technique
	SVM_Flag	16-bit / 0 or 0xAD	To indicate using SVM with PZV or standard 4-segment SVM 0: PZV 0xAD: Standard 4-seg SVM	For single shunt current sensing technique

8.5 Get current software module data structure

This data structure defines the variables used in the Get Current module where the phase current ADC results are read.

Table 30 Data structure used in get current module

Category (Structure)	Variable Name	Data Type / Range	Description	Remark
ADC	ADC_Iu	16-bit / 0 – 4095	Store phase U current ADC result	Three shunt current sensing technique
	ADC_Iv	16-bit / 0 – 4095	Store phase V current ADC result	
	ADC_Iw	16-bit / 0 – 4095	Store phase W current ADC result	
	ADC_Bias_Iu	16-bit / 0 – 4095	Bias of ADC_Iu	
	ADC_Bias_Iv	16-bit / 0 – 4095	Bias of ADC_Iv	
	ADC_Bias_Iw	16-bit / 0 – 4095	Bias of ADC_Iw	
	ADCTrig_Point	16-bit / 0 – CCU8 Slice 3 period value	VADC trigger position; This is compare match value for CCU8 slice 3 timer	
	ADC_Bias	16-bit / 0 – 4095	Bias of single shunt current input	Single shunt current sensing

Category (Structure)	Variable Name	Data Type / Range	Description	Remark
	ADC_ResultTz1	16-bit / 0 – 4095	ADC value of the first phase current ADC sampling	technique
	ADC_ResultTz2	16-bit / 0 – 4095	ADC value of the second phase current ADC sampling	
	ADC_Result3	16-bit / 0 – 4095	ADC value of the third phase current ADC sampling	
	ADC_Result4	16-bit / 0 – 4095	ADC value of the fourth phase current ADC sampling	
	ADC_Result1	signed 32-bit	Two most critical Phase current results for current reconstruction	
	ADC_Result2	signed 32-bit		
	ADC3Trig_Point	16-bit / 0 to ADC4Trig_Point	VADC trigger position; This is compare match value for CCU8 slice 3 timer	
	ADC4Trig_Point	16-bit / ADC3Trig_Point+1 to CCU8 Slice3 period value	VADC trigger position; This is compare match value for CCU8 slice 3 timer	
	Result_Flag	16-bit / 0 – 2	To indicate ADC result is for first CCU8 slice 3 cycle or second CCU8 slice 3 cycle or for standard 4-segment SVM: 0: First cycle, PZV 1: Second cycle, PZV 2: Standard 4-seg SVM	
	ADC_POT	0 – 4095	Store ADC result of potentiometer	
	ADC_DCLink	0 – 4095	Store ADC result of DC link voltage	
	ADC_IDCLink	0 – 4095	Store ADC result of average DC link current	

9 PMSM FOC software API functions

In this chapter, the PMSM FOC software API functions are stated. These APIs are group into the following sections:

- Controller
- Interrupt Service Routines, refer to chapter 5.1
- Miscellaneous APIs in Interrupt Service Routines
- Middle System layer

To improve the performance and to reduce the CPU loading, most of the time critical APIs are executed in the SRAM. The table below shows the list of APIs that are executed in SRAM.

Table 31 List of APIs

API Functions List		Execute in SRAM	Execute in Flash
Controller APIs	FOC_Speed_Controller()	x	
	FOC_Torque_Controller()	x	
	FOC_VQ_Controller()	x	
	DirectFOC_StartUp_Brake_Motor_Bootstrap_Charge()		x
	DirectFOCRotor_Pre_Positioning()		x
	VF_FOC_Brake_Motor_Bootstrap_Charge()		x
	VF_FOC_OpenLoop_RampUp()		x
	VF_Smooth_Transition_To_FOC()		x
	Misc_Works_of_MET()		x
	Stop_Motor()		x
	Error_Handling()		x
Interrupt Service Routines	DirectFOCStartup_CCU80_0_IRQHandler()	x	
	VF_FOC_CCU80_0_IRQHandler()	x	
	VF_ONLY_CCU80_0_IRQHandler()	x	
Miscellaneous APIs in ISR	Misc_Works_of_FOC()	x	
	Misc_Works_of_IRQ()	x	
	Linear_Ramp_Generator()	x	
	Linear_Torque_Ramp_Generator()	x	
	Linear_VQ_Ramp_Generator()	x	
	SCurve_Ramp_Generator()	x	
Middle System	PWMSVM01_Update()	x	
	ADC34_TriggerSetting()	x	
	ADCTZ12_TriggerSetting()	x	
	Get_ADCPhaseCurrent()	x	

9.1 Controller APIs

9.1.1 FOC speed control API

FOC_Speed_Controller()

This API is called if CONSTANT_SPEED_DIRECT_FOC or CONSTANT_SPEED_VF_MET_FOC control scheme is selected. It executes the FOC speed control algorithm and FOC calculations. It is called in the FOC_CLOSED_LOOP motor state.

With the MATH coprocessor in XMC1302, there is timing gain of a few microseconds by interleave the CPU calculation with CORDIC computations; refer to flowchart in Figure 50.

Table 32 FOC_Speed_Controller()

Input Parameters	ADC.ADC_Bias_Iu	ADC bias value for phase U current
	ADC.ADC_Bias_Iv	ADC bias value for phase V current
	ADC.ADC_Bias_Iw	ADC bias value for phase W current
	FOCInput.Ref_Speed	Reference speed value
	SVM.CurrentSectorNo	SVM current sector number
Return	Motor.Speed	Current motor speed from PLL Estimator
Updated Variables	Current.I_u, Current.I_v, Current.I_w	Current reconstruct function is called and I_u, I_v and I_w updated
	Park_Transform.Iq, Park_Transform.Id	Park transform function is called and Iq and Id updated
	Clarke_Transform.I_Alpha, Clarke_Transform.I_Beta	Clarke transform function is called and I_alpha and I_Beta updated
	Car2Polar.Vref, Car2Polar.Vref_AngleQ31	Car2Polar function is called and Vref and Vref_AngleQ31 updated
	PI_Speed PI_Torque PI_Flux PI_PLL	PI controller functions are called and the PI outputs updated
	PLL_Estimator	PLL Estimator APIs are called and the variables updated
	FOCOuput.Speed_by_Estimator	Estimated rotor speed from PLL Estimator
	FOCOutput.Rotor_PositionQ31	Estimated rotor position from PLL Estimator

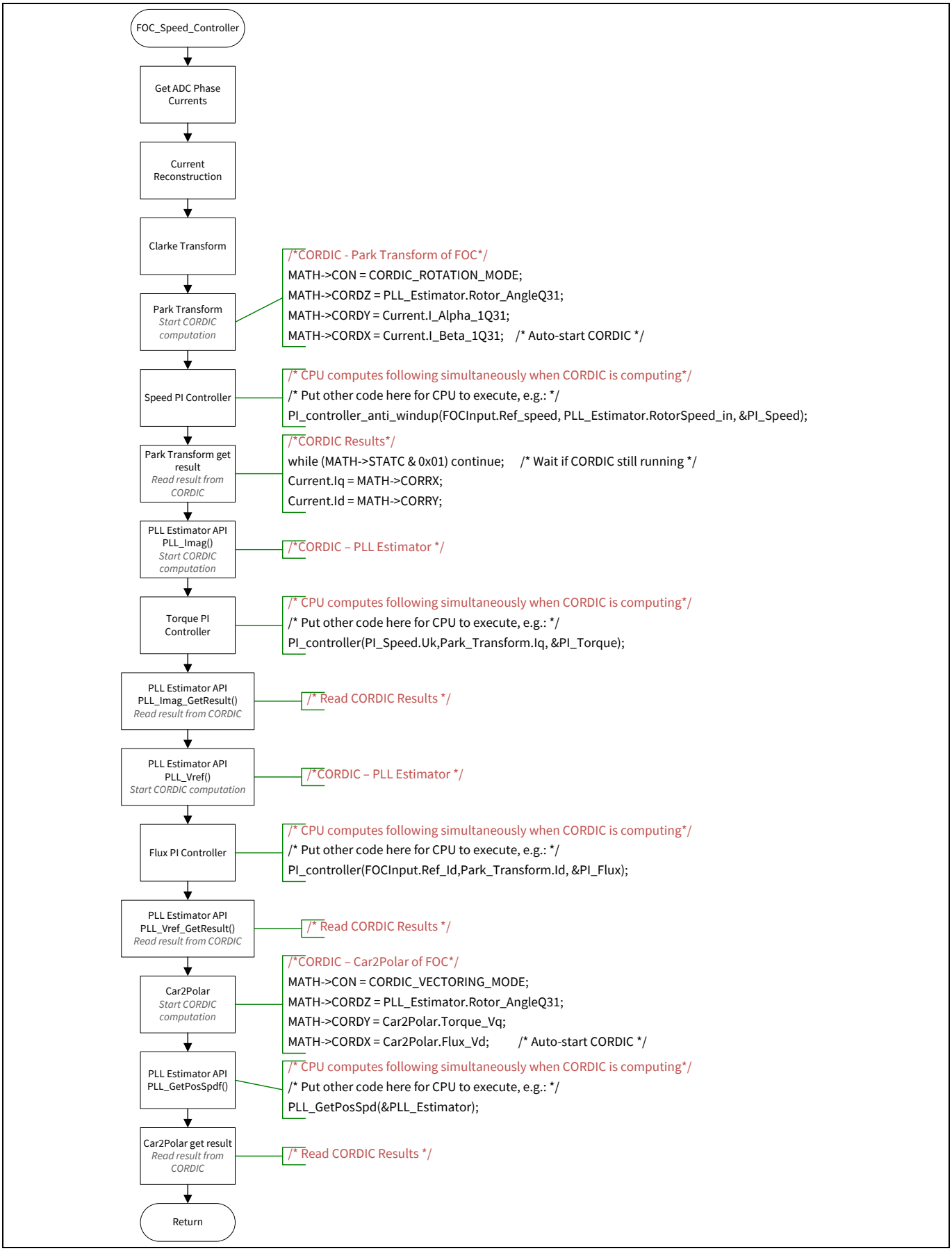


Figure 50 FOC speed control flowchart

9.1.2 Direct FOC startup APIs

The following list of APIs is executed for the direct FOC startup:

DirectFOC_StartUp_Brake_Motor_Bootstrap_Charge()

The function of this API is to perform motor brake and charge the gate driver bootstrap capacitor. At the same time, the motor phase currents bias are read and updated. This API is called in the BRAKE_BOOTSTRAP motor state.

Table 33 DirectFOC_StartUp_Brake_Motor_Bootstrap_Charge()

Input Parameters	None	-
Return	None	-
Updated Variables	ADC.ADC_Bias_Iu	ADC bias value for phase U current
	ADC.ADC_Bias_Iv	ADC bias value for phase V current
	ADC.ADC_Bias_Iw	ADC bias value for phase W current
	Motor.State	Updated motor state to PRE_POSITIONING once motor start command is received
	Motor.Mode_Flag	Motor status flag; Change this flag to MOTOR_TRANSITION once motor start command is received

DirectFOCRotor_Pre_Positioning()

This API is to set the initial rotor position/alignment. It is called in the PRE_POSITIONING motor state. It initialized the PI controller variables and FOCInput and change motor state to FOC_CLOSED_LOOP.

Table 34 DirectFOCRotor_Pre_Positioning()

Input Parameters	None	-
Return	None	-
Updated Variables	Current.I_u, Current.I_v, Current.I_w	Current reconstruct function is called and I_u, I_v and I_w updated
	Clarke_Transform.I_Alpha, Clarke_Transform.I_Beta	I_Alpha and I_Beta updated
	Car2Polar.Vref	Increase Vref value gradually for SVM
	Motor.State	Updated motor state to FOC_CLOSED_LOOP once preposition timer expired
	FOCInput	Initialized FOCInput variables before entering FOC_CLOSED_LOOP
	PI_Speed PI_Torque PI_Flux PI_PLL	Initialized PI controllers variables before entering FOC_CLOSED_LOOP

FOC_Torque_Controller()

This API is called if CONSTANT_TORQUE_DIRECT_FOC control scheme is selected. It executes the FOC torque control and its calculations. It is called in the FOC_CLOSED_LOOP motor state.

Table 35 FOC_Torque_Controller()

Input Parameters	ADC.ADC_Bias_Iu	ADC bias value for phase U current
	ADC.ADC_Bias_Iv	ADC bias value for phase V current
	ADC.ADC_Bias_Iw	ADC bias value for phase W current
	FOCInput.Ref.Iq	Reference torque value
	SVM.CurrentSectorNo	SVM current sector number
Return	Motor.Speed	Current motor speed from PLL Estimator
Updated Variables	Current.I_u, Current.I_v, Current.I_w	Current reconstruct function is called and I_u, I_v and I_w updated
	Park_Transform.Iq, Park_Transform.Id	Park transform function is called and Iq and Id updated
	Clarke_Transform.I_Alpha, Clarke_Transform.I_Beta	Clarke transform function is called and I_Alpha and I_Beta updated
	Car2Polar.Vref, Car2Polar.Vref_AngleQ31	Car2Polar function is called and Vref and Vref_AngleQ31 updated
	PI_Torque PI_Flux PI_PLL	PI controller functions are called and the PI outputs updated
	PLL_Estimator	PLL Estimator APIs are called and the variables updated
	FOCOuput.Speed_by_Estimator	Current motor speed from PLL Estimator
	FOCOutput.Rotor_PositionQ31	Current motor position from PLL Estimator

FOC_VQ_Controller(void)

This API is called if CONSTANT_VQ_DIRECT_FOC control scheme is selected. It executes the FOC VQ control and FOC calculations. It is called in the FOC_CLOSED_LOOP motor state.

Table 36 FOC_VQ_Controller(void)

Input Parameters	ADC.ADC_Bias_Iu	ADC bias value for phase U current
	ADC.ADC_Bias_Iv	ADC bias value for phase V current
	ADC.ADC_Bias_Iw	ADC bias value for phase W current
	FOCInput.Vq	Reference Vq value
	SVM.CurrentSectorNo	SVM current sector number
Return	Motor.Speed	Current motor speed from PLL Estimator
Updated Variables	Current.I_u, Current.I_v, Current.I_w	Current reconstruct function is called and I_u, I_v and I_w updated
	Park_Transform.Iq, Park_Transform.Id	Park transform function is called and Iq and Id updated
	Clarke_Transform.I_Alpha,	Clarke transform function is called and I_alpha and

	Clarke_Transform.I_Beta	I_Beta updated
	Car2Polar.Vref, Car2Polar.Vref_AngleQ31	Car2Polar function is called and Vref and Vref_AngleQ31 updated
	PI_Flux PI_PLL	PI controller functions are called and the PI outputs updated
	PLL_Estimator	PLL Estimator APIs are called and the variables updated
	FOCOuput.Speed_by_Estimator	Current motor speed from PLL Estimator
	FOCOutput.Rotor_PositionQ31	Current motor position from PLL Estimator

9.1.3 Open loop to closed loop FOC control scheme APIs

VF_FOC_Brake_Motor_Bootstrap_Charge()

This API is called in the BRAKE_BOOTSTRAP motor state if CONSTANT_SPEED_VF_MET_FOC control scheme is selected. The function of this API is to perform motor brake and charge the gate driver bootstrap capacitor. At the same time, the motor phase currents ADC bias values are read and updated.

Table 37 VF_FOC_Brake_Motor_Bootstrap_Charge()

Input Parameters	None	-
Return	None	-
Updated Variables	ADC.ADC_Bias_Iu	ADC bias value for phase U current
	ADC.ADC_Bias_Iv	ADC bias value for phase V current
	ADC.ADC_Bias_Iw	ADC bias value for phase W current
	Motor.State	Updated motor state to VFOPENLOOP_RAMP_UP once motor start command is received
	Motor.Mode_Flag	Motor status flag; Change this flag to MOTOR_TRANSITION once motor start command is received

VF_FOC_OpenLoop_RampUp()

This API is called in the VFOPENLOOP_RAMP_UP motor state. It sets the initial rotor position/alignment and then ramp up the motor in open loop. Once the motor speed reached the user defined VF_TRANSITION_SPEED, the motor state is changed to MET_FOC.

Table 38 VF_FOC_OpenLoop_RampUp()

Input Parameters	None	-
Return	None	-
Updated Variables	Current.I_u, Current.I_v, Current.I_w	Current reconstruct function is called and I_u, I_v and I_w updated
	Clarke_Transform.I_Alpha, Clarke_Transform.I_Beta	I_alpha and I_Beta updated
	Car2Polar.Vref	Increase Vref value gradually for SVM
	Motor.State	Updated motor state to MET_FOC once motor speed

		ramp up to a predefined value
	FOCInput	Initialized FOCInput variables before entering MET_FOC
	PLL_Estimator	Called PLL_Estimator API and update variables before entering MET_FOC

VF_Smooth_Transition_To_FOC()

This API is called in the MET_FOC motor state. It is for MET control strategy. Once the stator flux is perpendicular to the rotor flux, return the status as MOTOR_STABLE.

Table 39 VF_Smooth_Transition_To_FOC()

Input Parameters	None	-
Return	Motor.Mode_Flag	Motor mode status MOTOR_STABLE: MET is done, can switch to next state MOTOR_TRANSITION: MET function not done
Updated Variables	Current.I_u, Current.I_v, Current.I_w	Current reconstruct function is called and I_u, I_v and I_w updated
	Clarke_Transform.I_Alpha, Clarke_Transform.I_Beta	Clarke transform function is called and I_alpha and I_Beta updated
	Car2Polar.Vref, Car2Polar.Vref_AngleQ31	Car2Polar Vref and Vref_AngleQ31 updated
	FOCInput	FOCInput variables updated
	PLL_Estimator	PLL_Estimator APIs are called and variables are updated

Misc_Works_of_MET()

In this routine, it checks for motor stop command while waiting for MET state to be done. Once the MET is done, (Motor.Mode_Flag == MOTOR_STABLE), it switches the motor state to FOC_CLOSED_LOOP.

Table 40 Misc_Works_of_MET()

Input Parameters	Motor.Mode_Flag	Motor mode status
Return	None	-
Updated Variables	Motor.State	Updated motor state to FOC_CLOSED_LOOP once Motor.Mode_Flag == MOTOR_STABLE
	FOCInput	Initialized FOCInput variables before entering FOC_CLOSED_LOOP
	PI_Speed PI_Torque PI_Flux PI_PLL	Initialized PI controllers variables before entering FOC_CLOSED_LOOP
	PLL_Estimator.RotorAngleQ31	Initialize rotor angle for the first FOC PWM cycle before entering FOC_CLOSED_LOOP

9.2 APIs in Interrupt Service Routines

Misc_Works_of_FOC()

In this routine, if the motor is ramping up in speed/torque/Vq, the reference speed or reference torque or Vq, is updated based on the ramping rate. If the motor.mode_flag is in MOTOR_STABLE mode, the routine checks the DC link voltage and motor stop command.

Table 41 Misc_Works_of_FOC()

Input Parameters	Motor.Mode_Flag	Motor mode status
Return	None	-
Updated Variables	Motor.State	If DC link voltage is not within limits, change motor state to DCLINK_OVER_UNDER_VOLTAGE Update motor state to STOP_MOTOR if motor stop command received
	Motor.Ref_Speed	Updated if CONSTANT_SPEED_DIRECT_FOC or CONSTANT_SPEED_VF_MET_FOC control scheme is selected
	FOCInput.Ref_Iq	Updated if CONSTANT_TORQUE_DIRECT_FOC control scheme is selected
	FOCInput.Vq	Updated if CONSTANT_VQ_DIRECT_FOC control scheme is selected

Misc_Works_of_IRQ()

This API is called every PWM cycle. Every 64 PWM cycles, it will read motor speed set by the user and scale it up as mention in the chapter 2.10 on scaling. The watchdog timer is also refreshed every 64 PWM cycles.

Table 42 Misc_Works_of_IRQ()

Input Parameters	None	-
Return	None	-
Updated Variables	Motor.Speed_by_POT_PWM	Updated every 64 PWM cycles

Linear_Ramp_Generator()

In this routine, the linear ramp generator for speed control is implemented.

Table 43 Linear_Ramp_Generator()

Input Parameters	set_val	Target speed value
	rampup_rate	Speed ramp up rate
	rampdown_rate	Speed ramp down rate
	speedrampstep	Number of steps changed every (rampup_rate or rampdown_rate) x PWM cycles
Return	Motor.Ref_speed	Motor reference speed for PI speed controller
Updated Variables	Motor.Ramp_Up_Rate	Motor speed ramp up rate
	Motor.Ramp_Counter	Ramp counter

Linear_Torque_Ramp_Generator()

In this routine, the linear ramp generator for torque control is implemented.

Table 44 Linear_Torque_Ramp_Generator()

Input Parameters	current_set	Target torque value
	inc_step	Torque ramp up rate
	dec_step	Torque ramp down rate
Return	FOCInput.Ref_Iq	Motor reference torque for PI torque controller
Updated Variables	-	

Linear_VQ_Ramp_Generator()

In this routine, the linear ramp generator for Vq control is implemented.

Table 45 Linear_VQ_Ramp_Generator()

Input Parameters	set_val	Target Vq value
	inc_step	Vq ramp up rate
	dec_step	Vq ramp down rate
Return	FOCInput.Vq	Reference Vq for Cartesian to Polar transformation
Updated Variables	-	

SCurve_Ramp_Generator()

In this routine, the S-curve ramp generator for speed control is implemented.

Table 46 SCurve_Ramp_Generator()

Input Parameters	set_val	Target speed value
	rampup_rate	Speed ramp up rate
	rampdown_rate	Speed ramp down rate
	speedrampstep	Number of steps changed every (rampup_rate or rampdown_rate) x PWM cycles
Return	Motor.Ref_speed	Motor reference speed for PI speed controller
Updated Variables	Motor.Ramp_Up_Rate	Motor speed ramp up rate
	Motor.Ramp_Dn_Rate	Motor speed ramp down rate
	Motor.Ramp_Counter	Ramp counter

9.3 APIs in middle system layer

PWMSVM01_Update()

In this API, the SVM algorithm is executed and 3-phase PWM duty cycles are updated.

Table 47 PWMSVM01_Update()

Input Parameters	Amplitude	Amplitude of voltage space vector
	Angle	Angle of voltage space vector
Return	-	
Updated Variables	SVM.PreviousSectorNo	Backup current SVM sector number
	SVM.CurrentSectorNo	Update current SVM sector number

Get_ADCPhaseCurrent()

This API is only used in three shunt current sensing technique. It reads the ADC results of the 3-phase currents. If synchronous conversion is used, the VADC alias channels settings are also updated according to the SVM sector.

Table 48 Get_ADCPhaseCurrent()

Input Parameters	SVM.PreviousSectorNo	SVM sector number in previous PWM cycle
	SVM.CurrentSectorNo	Current SVM sector number
Return	ADC.ADC_Iu	ADC Phase U current result
	ADC.ADC_Iv	ADC Phase V current result
	ADC.ADC_Iw	ADC Phase W current result
Updated Variables	-	

ADCTZ12_TriggerSetting()

This API is only called in single shunt current sensing technique. It is to set the first two trigger points for the ADC conversion. The settings of the trigger points are the pre-calculated constants in the configuration file.

Table 49 ADCTZ12_TriggerSetting()

Input Parameters	-	
Return	-	
Updated Variables	-	

ADC34_TriggerSetting()

This API is only called in single shunt current sensing technique. This API is to set the third and fourth trigger points for the ADC conversion.

Table 50 ADC34_TriggerSetting()

Input Parameters	ADC.ADC3Trig_Point	Third trigger point setting
	ADC.ADC4Trig_Point	Fourth trigger point setting
Return	-	
Updated Variables	-	

10 GUI - µC/Probe XMCTM dashboard

The µC/Probe is a Windows application designed to read and write the memory of the embedded target processor during run-time. Memory locations are mapped to a set of virtual controls and indicators placed on a dashboard.

Figure 51 µC/Probe dashboard

The µC/Probe is used to monitor the motor parameters. It can also be used to fine-tune the PI gains to get the optimum motor behavior.

The default P gain and I gain values are loaded. The K_P and K_I gain values of the four PI control loops are located under the *P setting* and *I setting* columns. User can change the K_P, K_I and *SCALEKPKI* values to get the optimum motor behavior. To start the motor, drag the *Set Speed* slider to the right.

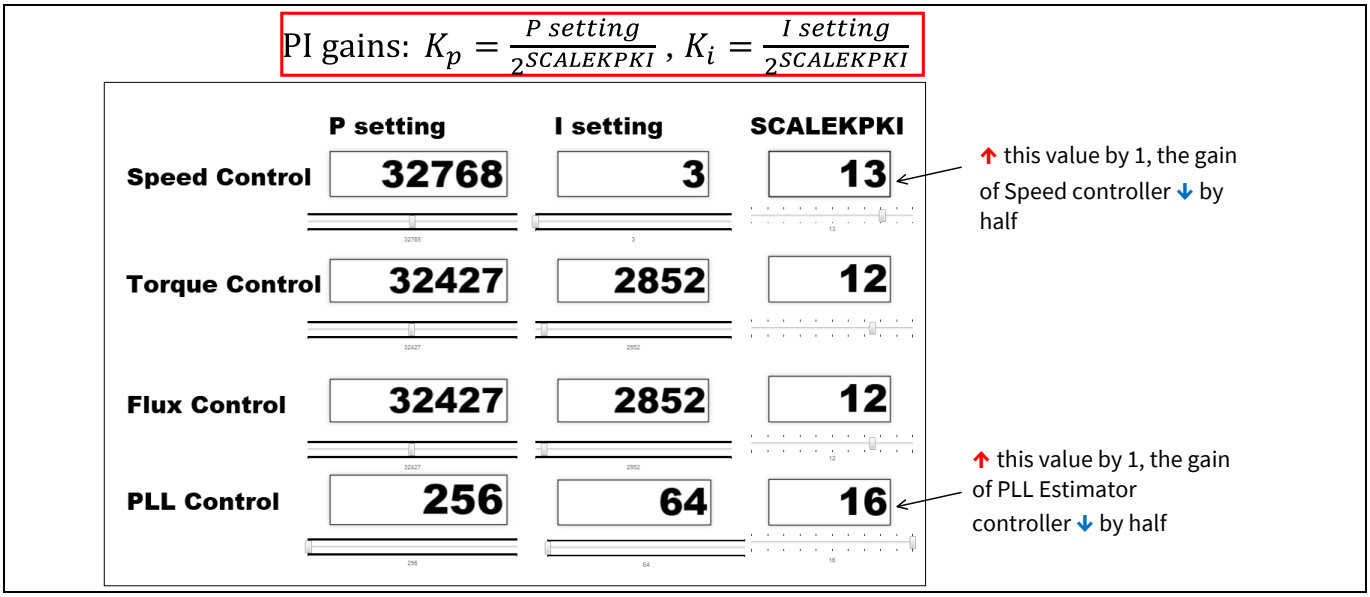


Figure 52 PI controllers settings in µC/Probe dashboard

To save the optimal PI values, enter these values into the file pmsm_foc_pi.h, refer to chapter 7.4.2.

11 Resources

- Document Infineon XMC1000 Motor Control Application Kit: XMC1000 Motor Control Application Kit
- PMSM FOC Motor Control Sensorless Software project: PMSM Motor Control FOC Sensorless Example for XMC1300 Series

Revision history

Major changes since the last revision

Page or Reference	Description of change
All pages	First release

Edition yyyy-mm-dd

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
AP32370

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.