PMSM FOC motor control software using MOTIX™ 6EDL7141 and MOTIX™ IMD700A

MOTIX™ 6EDL7141, MOTIX™ IMD700A

About this document

Scope and purpose

This document describes the implementation of permanent magnet synchronous motor (PMSM) field-oriented control (FOC) MOTIX™ motor control software (SW) for a three-phase motor using the integrated controller and smart three-phase gate driver MOTIX™ IMD700A.

Intended audience

This document is intended for customers who would like a configurable system for FOC with sensorless feedback using the MOTIX™ IMD700A.

Table of contents

1 Introduction

The intention of this SW is to offer functionality to drive PMSMs in sensorless mode. It contains all the common modules necessary for generic drive mode, and provides a high level of configurability and modularity to address different segments.

FOC is a method of motor control to generate three-phase sinusoidal signals which can easily be controlled in frequency and amplitude to minimize the current, which in turn means maximizing efficiency. The basic idea is to transform three-phase signals into two rotor-fixed signals, and vice-versa.

Feedback on rotor position and rotor speed is required in FOC motor control. The feedback can come from a rotor position estimator algorithm or from rotor position sensors. In this SW, the feedback comes from a rotor position estimator algorithm, so it is known as sensorless FOC motor control.

- Sensorless FOC derives the rotor position and rotor speed based on motor modeling, the voltage applied to the motor phases and the current in the three motor phases.
- FOC with sensors determines the rotor position and rotor speed from rotor sensor(s), such as Hall sensors or an encoder.

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

The figure below shows a typical block diagram for PMSM FOC motor control, where three-shunt low-side current sensing is supported.

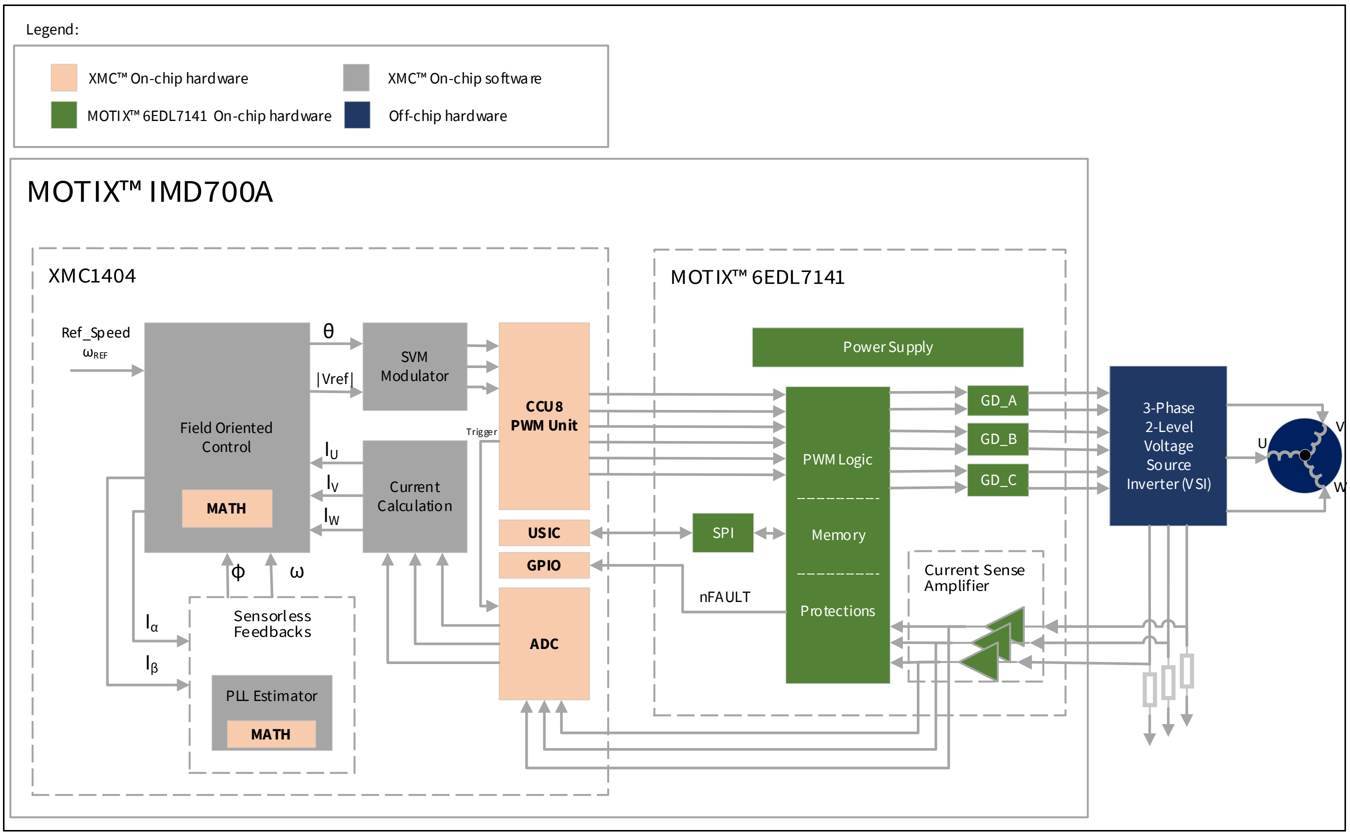

Figure 1 **Block diagram of PMSM FOC motor control using MOTIX™ IMD700A**

1.1 Key features

Multiple Infineon innovations and unique features are included in the sensorless PMSM FOC SW, such as:

- Optimized FOC
  - No inverse Park transform
- PLL estimator – 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 following table.

Table 1 Key features

Feature	
Math control blocks	Clarke transform
	Park transform
	I_d and I_q current flux/torque PI controller
	Speed PI controller
	Cartesian to polar transform
	Ramp function
Control scheme	Open-loop voltage control
	Speed control
	V_q voltage control
Space vector modulation	Five-segment SVM
	Seven-segment SVM
Low-side current sensing	Three-shunt support five-segment SVM and seven-segment SVM
Start-up algorithm	Direct FOC start-up
	VF open-loop start-up
Protection	Gate driver fault reporting pin (nFAULT)
	Phase overcurrent protection (OCP)
	DC-link undervoltage protection (UVP) and overvoltage protection (OVP)
	SPI fault
Device feature	Current sense amplifiers' gain and offset are programmable
	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 CORDIC hardware (HW)
Others	Linear ramp generator
	PI anti-windup for speed control
	D_q-axis decoupling

1.2 Abbreviations and acronyms

Table 2 Abbreviations and acronyms used in this document

Term	Definition
API	Application programming interface
BOM	Bill of materials
CCU8	Capture compare unit 8
CPU	Central processing unit
FOC	Field-oriented control
GPIO	General-purpose input/output
IP	Intellectual property
ISR	Interrupt service routine
MCU	Microcontroller unit
OCP	Overcurrent protection
OVLO	Overvoltage lockout
OTS	Overtemperature shutdown
OTW	Overtemperature warning
PI	Proportional integral controller
PLL	Phase-locked loop
PMSM	Permanent magnet synchronous motor
PWM	Pulse-width modulation
SRAM	Static random-access memory
SVM	Space vector modulation
UART	Universal asynchronous receiver transmitter
UVLO	Undervoltage lockout
VADC	Versatile analog-to-digital converter
XMC	XMC™ MCU family based on Arm®
XMC1000	XMC™ MCU family based on Arm® Cortex®-M0 core
XMC1400	XMC™ MCU series with 48 MHz core and 96 MHz peripheral frequency
XMC1404	XMC™ MCU product with specific feature set, e.g., CORDIC and CAN

1.3 MOTIX™ IMD700A resource allocation

Infineon's MOTIX™ 6EDL7141 and MOTIX™ IMD700A is a controller specifically designed for three-phase brushless DC (BLDC) or PMSM drive applications. **MOTIX™ 6EDL7141** is a 70 V three phase smart gate driver with integrated power supply while MOTIX™ IMD700A integrates a fully programmable XMC1404 Arm® Cortex®-M0 microcontroller from Infineon's XMC1400 family with **MOTIX™ 6EDL7141**.

The XMC1404 microcontroller is ideal for PMSM FOC motor control systems. It has dedicated motor control peripherals POSIF, MATH, CCU8, ADC and CCU4.

MOTIX™ 6EDL7141 is a flexible three-phase gate driver. It has complete power supply as required in the system, three current sense (CS) amplifiers, protections and a remarkable set of configurations to adjust to specific needs.

The HW peripherals used in this PMSM FOC motor control SW are listed in the table below.

*Note: The default resource allocation is for the Infineon **EVAL IMD700A FOC 3SH** (order number: EVAL_IMD700A_FOC_3SH).*

Table 3 XMC™ resource allocation

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-shunt 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
	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

[1] The same input pin must connect to both the ADC group channels to perform synchronous sampling. Refer to **chapter 3.2.2** for details.

Table 4 MOTIX™ 6EDL7141 resource allocation

6EDL7141 blocks	Usage
Power supply subsystem	Synchronous buck converter including both power switches
	DVDD linear voltage regulator pre-programmed to either 3.3 or 5 V
	Charge pump for low-side gate driver
	Charge pump for high-side gate driver
Smart gate driver	PWM modes
	Slew rate control
	Gate driver voltage programmability
	Charge pump configurations
	Gate driver and charge pump protections
CS amplifiers	CS amplifier
	Output buffer
	Positive overcurrent comparator
	Negative overcurrent comparator
	OCP digital-to-analog converter
Protections and faults handling	OCP
	UVLO protection
	DVDD linear regulator OVLO protections
	Configurable watchdog
	OTS and OTW
	OTP memory fault

1.4 MOTIX™ IMD700A hardware modules interconnectivity

The MOTIX™ IMD700A has comprehensive HW interconnectivity, by integrating XMC1404 and the MOTIX™ 6EDL7141 smart gate driver.

The figure below shows the interconnections between XMC1404 HW peripheral modules and MOTIX™ 6EDL7141 modules.

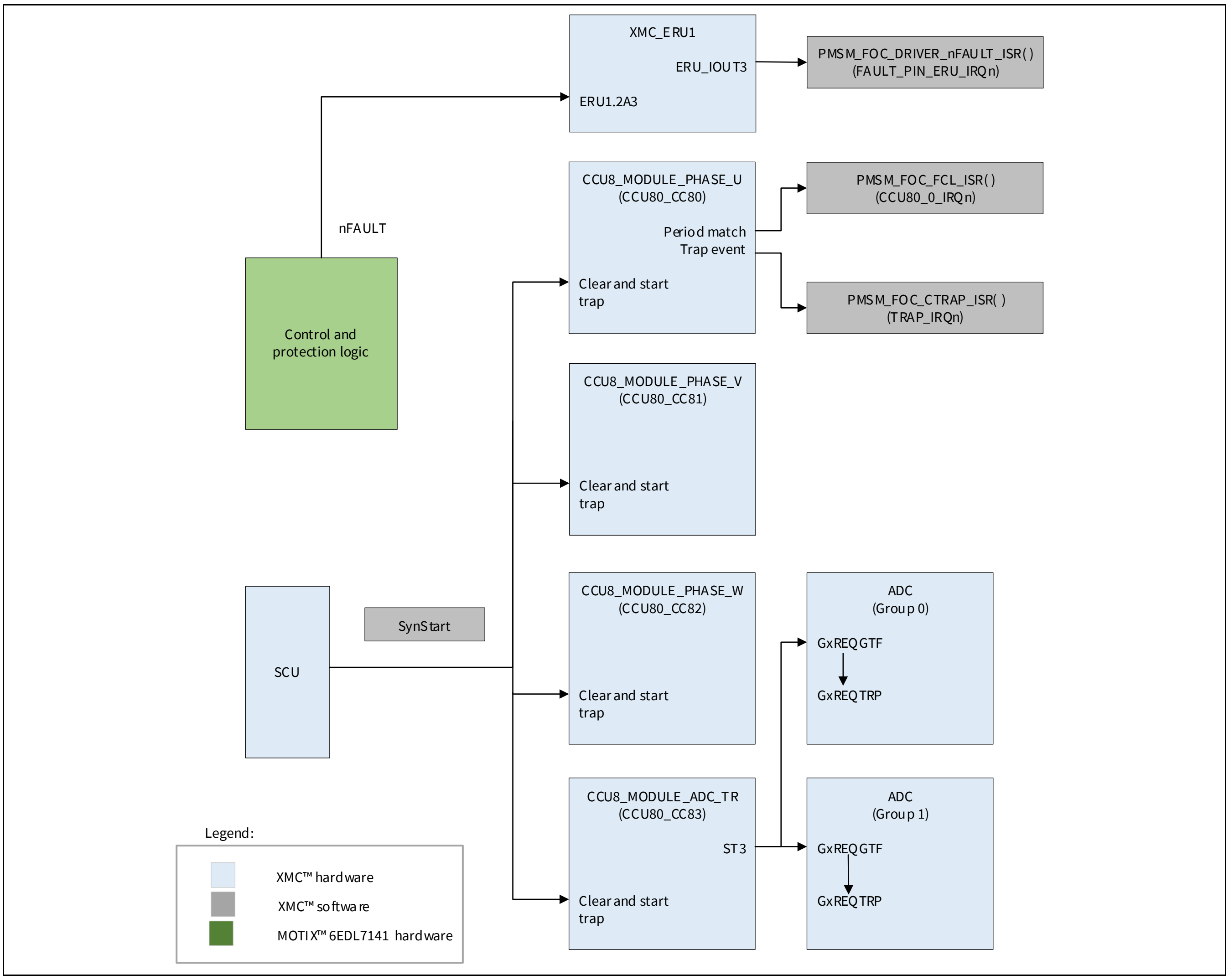

Figure 2 IMD700A HW interconnection

Note:

1. *The CCU8 slice timers are started synchronously with the sync start signal from the system control unit (SCU).*
2. *The VADC conversion is triggered by the CCU8 slice 3 compare match status signal.*

1.5 Execution time and memory usage

In the XMC1000 family, access to the static random-access memory (SRAM) requires no wait state. The major parts of the SW are executed from the SRAM. The interrupt service routines (ISRs), all the mathematical blocks of the FOC algorithm, the space vector modulation (SVM), and the motor phase current sensing and calculation are executed in the SRAM. This improves performance, as the execution time to run the FOC algorithm is reduced by approximately 30 percent.

Breakdown of the memory usage and CPU time-utilization are provided in the following table based on the default settings for the Infineon EVAL_6EDL7141_FOC_3SH v2.1 evaluation board.

- Control scheme
  - SPEED_INNER_CURRENT_CTRL
- Current sensing technique
  - Three-shunt synchronous ADC conversion
- GUI feature
  - USER_UCPROBE_GUI enabled
  - USER_UCPROBE_OSCILLOSCOPE enabled

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

Table 5 CPU utilization and memory usage for three-shunt current sensing with XMC1404

	SPEED_INNER_CURRENT_CTRL
PWM frequency	20 kHz – ISR runs every 50 µs
Modus Toolbox GCC compiler optimization level	Optimized most (-O3)
Motor control CPU utilization	33.2 µs (66.4 %)
Motor control and Oscilloscope CPU utilization	48 µs (96.0 %)
Flash code size used (bytes)	30696
SRAM data size used (bytes)	13500

1.6 Software overview

The PMSM FOC motor control SW is developed based on a 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.

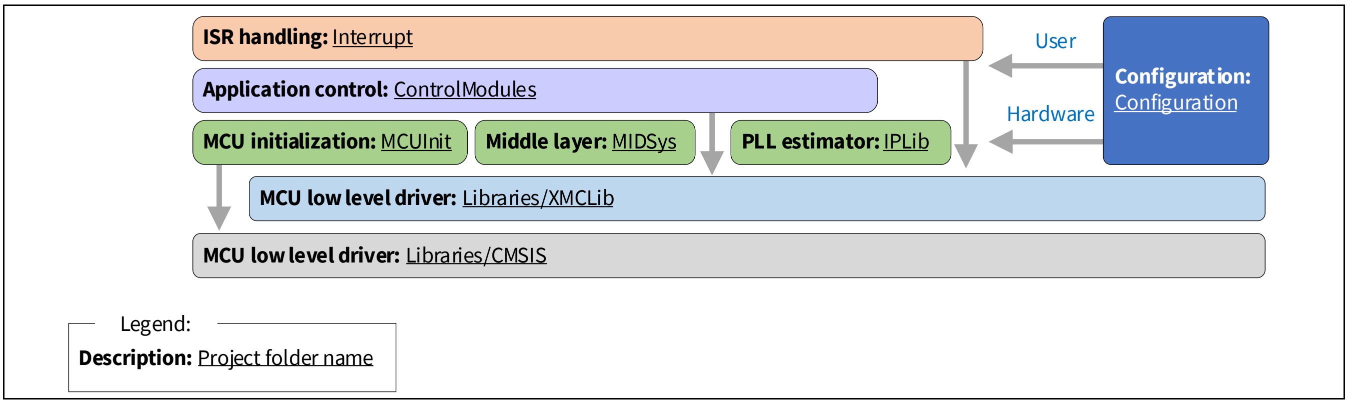

Figure 3 PMSM FOC SW overview; layered structure

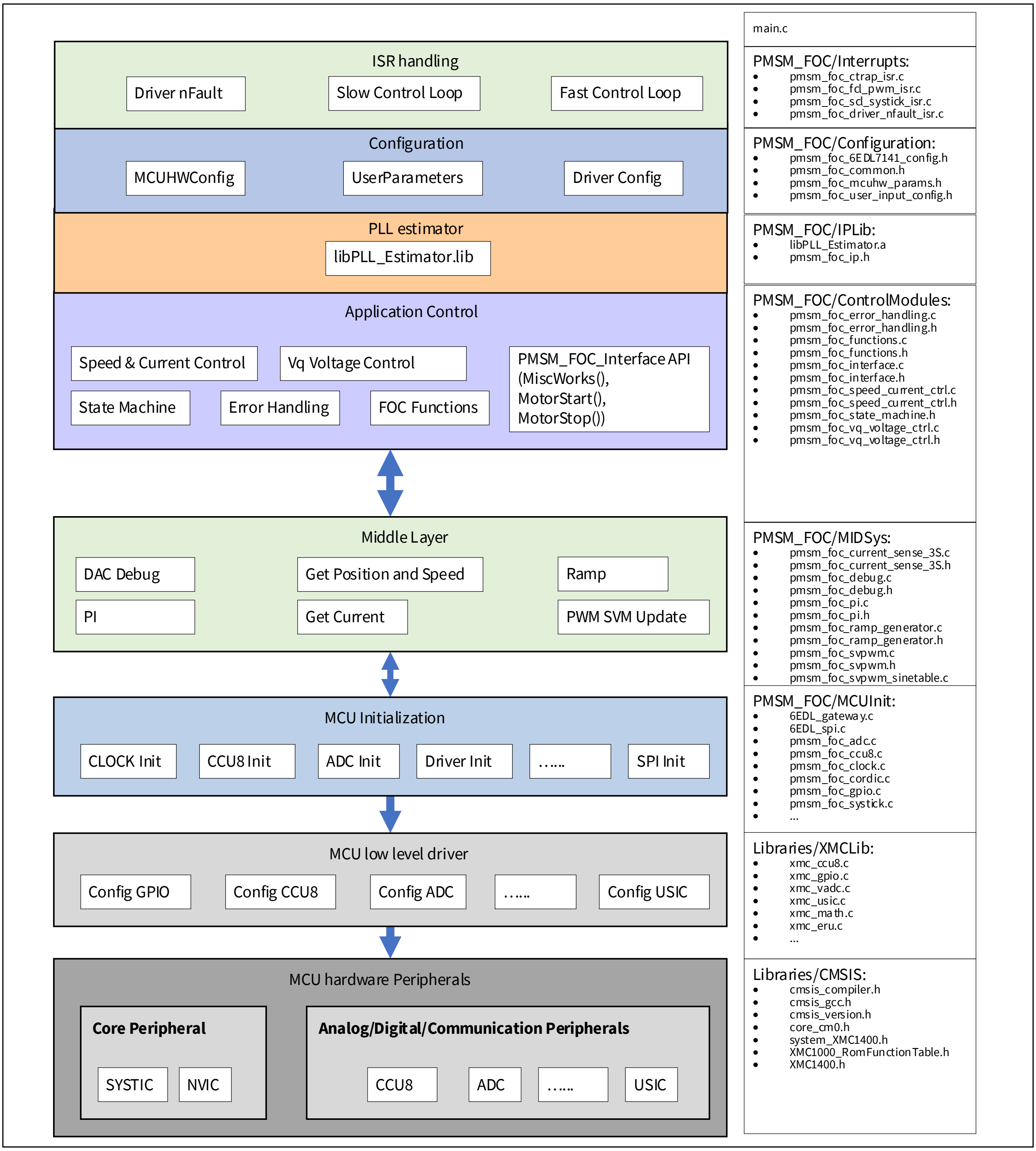

Figure 4 Project folder structure

ISR handling: Interrupts

This layer consists of a gate driver-triggered interrupt handling function, a CCU8 period match ISR function for fast control loop, and a SysTick ISR function for slow control loop.

Configuration: Configuration

The configuration is divided. The user configuration affects the general behavior of the SW. The file pmsm_foc_user_input_config.h can be modified. Predefined HW kits are available.

The HW configuration allows for more detailed adaptation to the customer HW.

This layer is divided into:

- MCU
- Gate driver
- Motors

The specific associated *.h can be found in the associated folders.

The static configuration is accessible with the file pmsm_foc_common.h.

Note: You should not change this configuration and definition file.

All configurations can be found in the folder "Configuration".

PLL estimator: IPLib

Infineon patented IP and PLL estimator, provided as a compiled .a library file.

The file can be found in the folder "IPLib".

Application control: Control Modules

This layer consists of FOC SW control modules. It includes the Clarke transform, Park transform, Cartesian to polar transform, current reconstruction, speed and current control or V_q voltage control, and state machine, for example.

All the routines mentioned are called from the CCU80 period match ISR.

All files for this layer can be found in the folder "ControlModules".

Middle layer: MIDSys

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

All files for this layer can be found in the folder "MIDSys".

MCU initialization: MCUInit

This layer controls the initialization of all MCU peripherals and the gate driver MOTIX™ 6EDL7141. It contains XMCLib data structure initialization and peripheral initialization functions. It also contains the initialization for the MOTIX™ 6EDL7141 data structure, which is declared in the "Configuration" folder. The MCU initialization in this layer closely interacts with XMCLib and the MIDSys layer to configure each peripheral.

All files for this layer can be found in the folder "MCUInit".

MCU low-level driver: Libraries/XMCLib

This layer contains the XMC™ peripheral driver library.

MCU HW peripherals: Libraries/CMSIS

This layer is the HW abstraction layer to the MCU peripherals. Common Microcontroller Software Interface Standard (CMSIS) provides interfaces to processors and peripherals, real-time operating systems and middleware components.

All files for this layer can be found in the folder "Libraries".

2 PMSM FOC sensorless software components

The intention of this PMSM FOC motor control SW is to offer functionality to drive the PMSM motors with sensorless modules.
The PMSM FOC SW provides a high level of configurability and modularity to address different motor control applications.
Three types of control scheme are supported:

- VF_OPEN_LOOP_CTRL
- SPEED_INNER_CURRENT_CTRL
- VQ_VOLTAGE_CTRL

Two motor start-up methods are supported:

- MOTOR_STARTUP_DIRECT_FOC
- MOTOR_STARTUP_VF_OPEN_LOOP

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

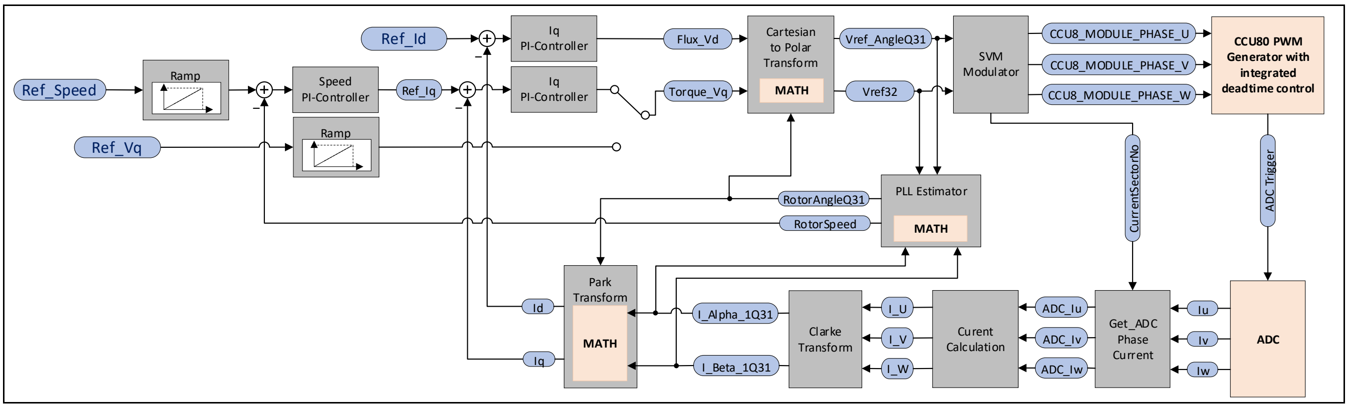

Figure 5 PMSM FOC block diagram

2.1 Motor start/stop, speed change operations control

The motor is started with the start command.

The target speed can be changed between:

- USER_SPEED_LOW_LIMIT_RPM
- USER_SPEED_HIGH_LIMIT_RPM

One option for controlling is a potentiometer.

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

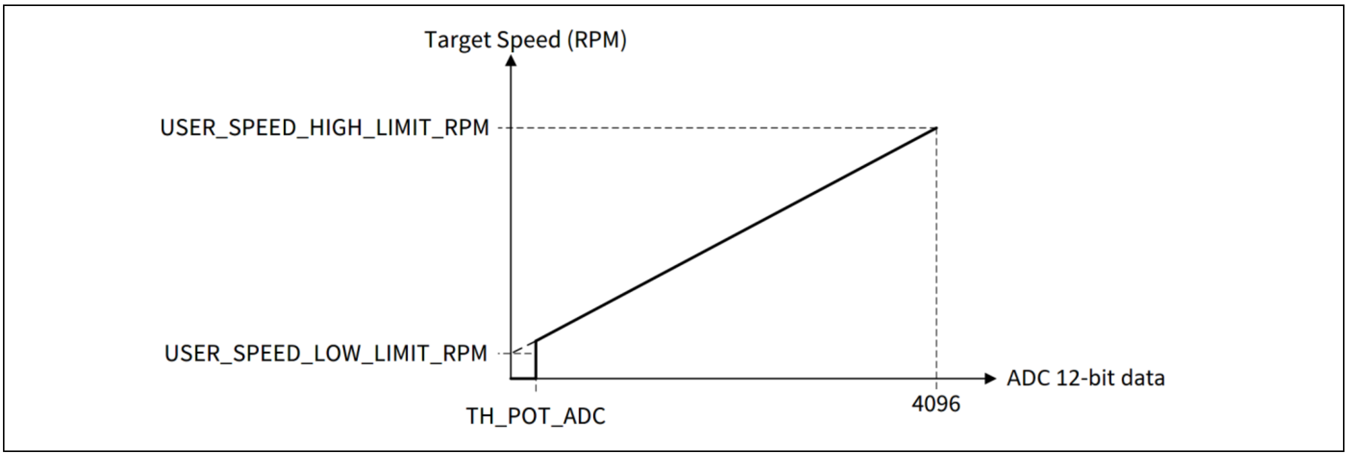

Figure 6 Potentiometer ADC value vs. speed in speed control scheme

However, due to the 1 kOhm resistor placed above the potentiometer POT1 on the EVAL_6EDL7141_FOC_3SH and EVAL_IMD700A_FOC_3SH evaluation board, the on-board potentiometer cannot reach ADC value of 4096 when POT1 is turn to maximum position. DVDD is the ADC reference voltage.

Hence, a scaling factor of 1.1 is applied to the read back ADC value in code example as shown in **Figure 8**, to obtain an ADC value of 4096 when turn POT1 to maximum position.

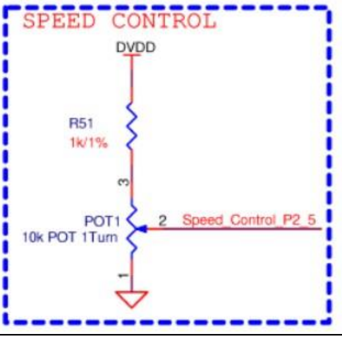

Figure 7 Schematic of POT1 implementation on EVAL_6EDL7141_FOC_3SH and EVAL_IMD700A_FOC_3SH evaluation board

pmsm_foc_interface.c

```
void PMSM_FOC_MiscWorks(void)
{
  XMC_VADC_GROUP_ScanTriggerConversion(VADC_G0);
  XMC_VADC_GROUP_ScanTriggerConversion(VADC_G1);
  if (MotorParam.AnalogControl == ENABLED)
  {
    volatile uint16_t pot_adc_result;
    pot_adc_result = XMC_VADC_GROUP_GetResult(VADC_POT_GROUP, VADC_POT_RESULT_REG);
    ADC.adc_res_pot = (int32_t) ((pot_adc_result * 4505) >> 12); /* 1.1 scaling for the 1kohm resistor place before 10Kohm potentiometer */
```

Figure 8 Addition of 1.1 scaling factor to ADC result reading from potentiometer POT1

From the state PMSM_FOC_MSM_CLOSED_LOOP or PMSM_FOC_MSM_VF_OPEN_LOOP, three options are available to prevent the motor from running:

- Stop command from the GUI
- Set the target speed below the threshold
- Call the stop function PMSM_FOC_MotorStop()

When one of the three options happens, the state is changed to PMSM_FOC_MSM_BRAKE_BOOTSTRAP. After a configured time with a configured duty cycle for low-side switch, the gate driver will be disabled, and the state will be changed to PMSM_FOC_MSM_STOP_MOTOR. The inverter is disabled and the output set to tristate. The result is an uncontrolled freewheeling. Depending on the current motor speed and the friction, the motor should run in a freewheeling state if it is not stopped by the brake feature.

2.2 Ramp generator

This PMSM FOC motor control SW provides a linear curve ramp function.

An input parameter is ramped from an initial value to an end value. The ramp generator input is connected to the analog input from the POT. The user-set value from the GUI is not connected to the ramp generator.

- USER_SPEED_RAMPUP_RPM_PER_S
- USER_SPEED_RAMPDOWN_RPM_PER_S
- USER_SPEED_RAMP_SLEWRATE

provide the ramp rate for SPEED_INNER_CURRENT_CTRL control scheme.

- USER_VQ_RAMPUP_STEP
- USER_VQ_RAMPDOWN_STEP
- USER_VQ_RAMP_SLEWRATE

provide the ramp rate for the VQ_VOLTAGE_CTRL control scheme.

The ramp-up and ramp-down rates are defined in the user configuration file, pmsm_foc_user_input_config.h (refer to **chapter 7.2.7**). The ramp generator function is called every PWM frequency cycle.

The VF_OPEN_LOOP_CTRL control scheme does not use this ramp generator.

The DC-link voltage is monitored during ramp-down operation. It stops the speed/voltage ramp-down if the DC-link voltage is over the user-configured voltage limit. This is to avoid an overvoltage condition during fast braking. This voltage limit, USER_BRAKING_VDC_MAX_LIMIT, is defined in the header file pmsm_foc_user_input_config.h.

The ramp output is the reference signal to the control scheme. Depending on the control scheme selected, the ramp output can be speed or torque V_q.

2.3 Control schemes

In this SW block the control scheme for the three-phase PMSM FOC motor can be:

- Open-loop voltage control
- Speed control
- V_q voltage control

2.3.1 Open-loop voltage control

In an open-loop voltage control, a reference voltage (V_{ref}) 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.

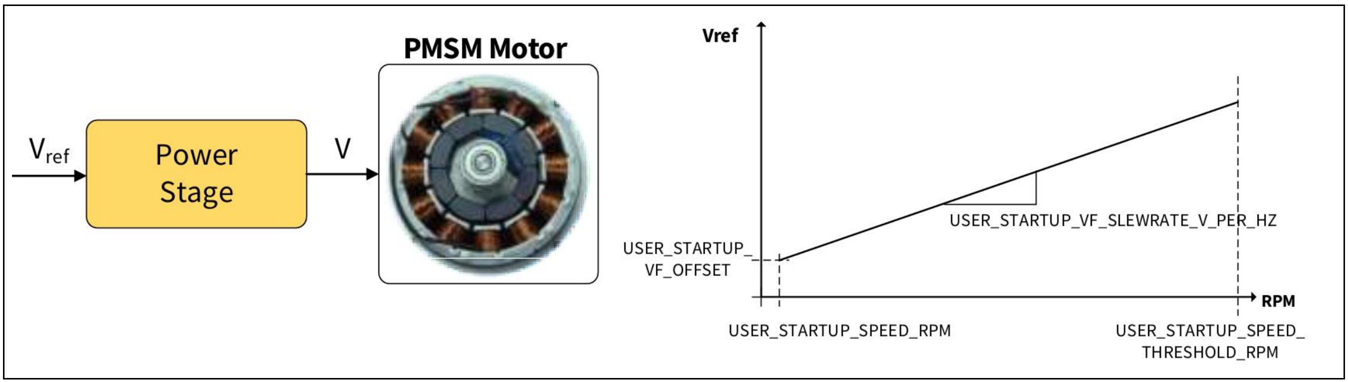

Figure 9 Open-loop voltage control

2.3.2 Speed control

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

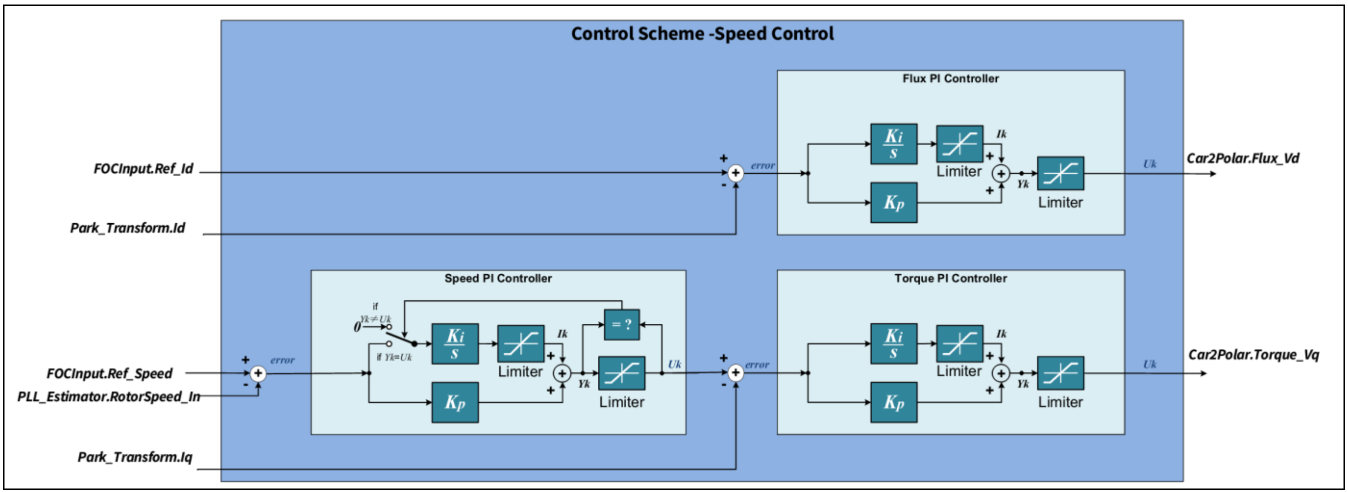

Figure 10 Speed control

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.

Direct FOC start-up and transition start-up (open-loop to closed-loop) modes are supported in speed control.

Transition start-up mode – three-step start-up

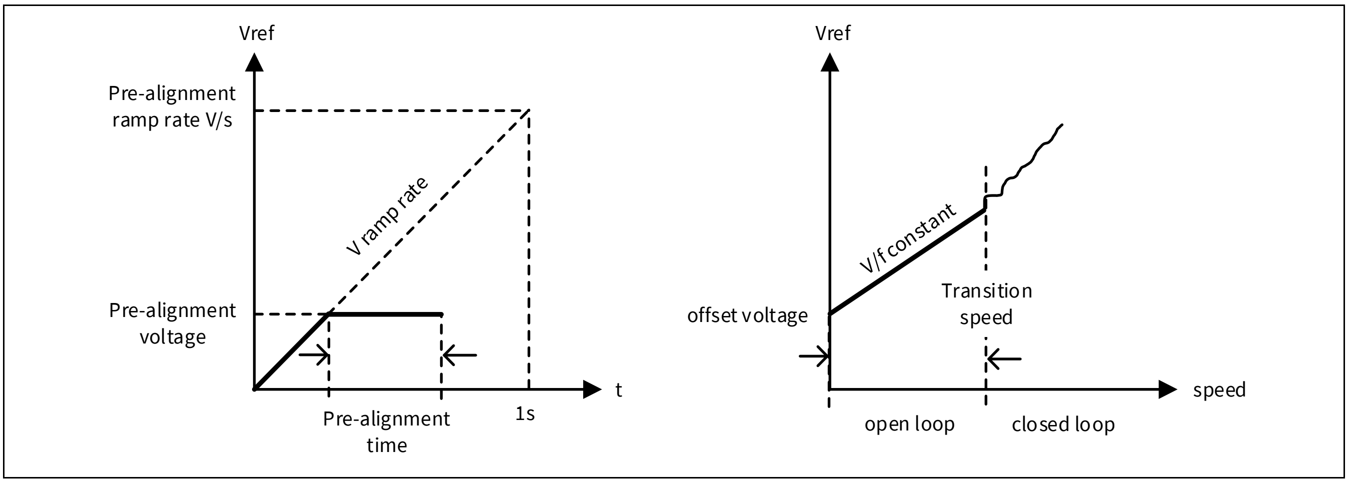

Figure 11 Three-step motor start-up mechanism

The three steps are:

1. After the pre-alignment implementation (optional), the motor starts in V/f open-loop control state and ramps up to a user-defined start-up speed.
2. The PI controllers' integral terms are initialized at the end of open loop for smooth transition to closed loop.
3. The state machine switches to FOC_CLOSED_LOOP state and ramps up the motor speed to the user-defined target speed.

2.3.3 V_q voltage control

The V_q voltage control is used when a fast response is required and varying speed is not a concern.

The speed PI control loop and torque PI control loop are disabled.

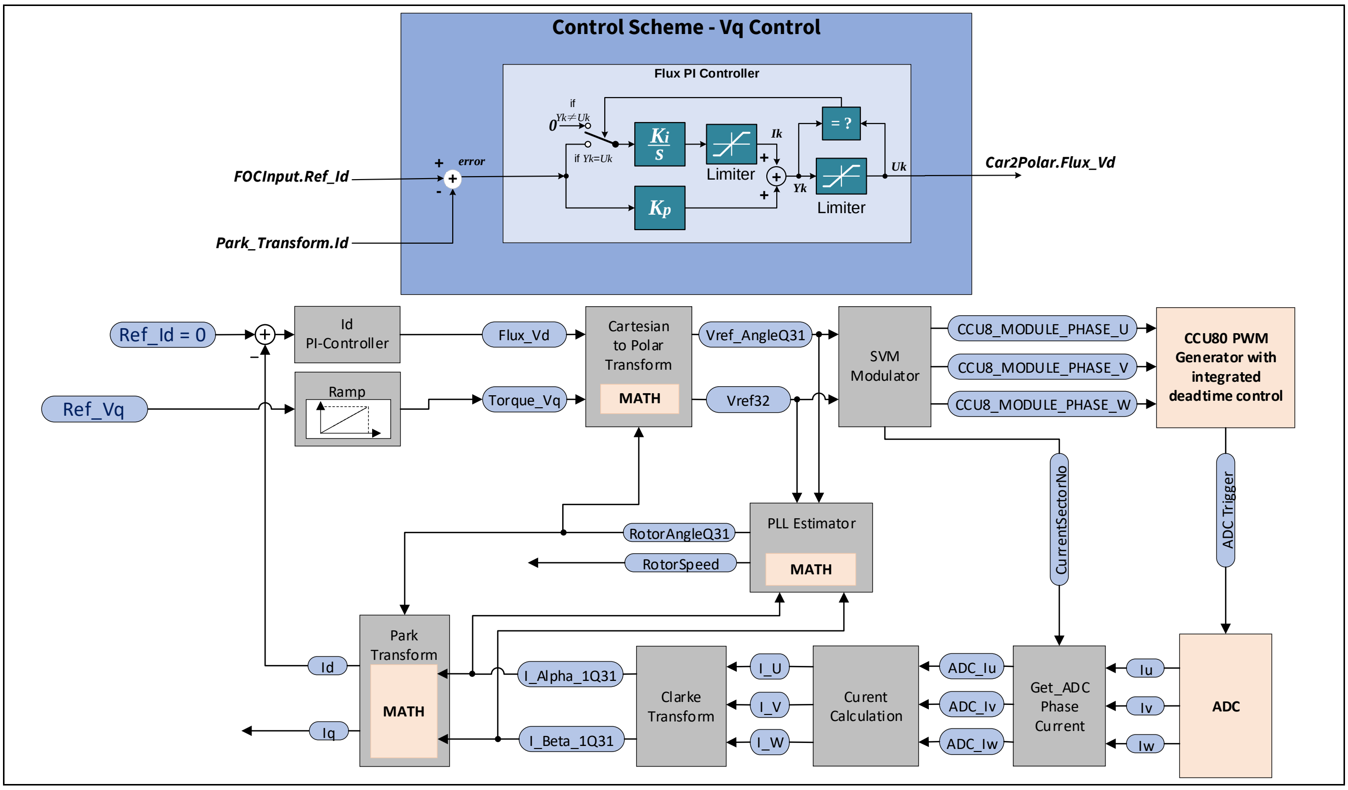

Figure 12 V_q voltage control scheme

2.3.4 D-axis and Q-axis decoupling

The controls of I_d and I_q currents are not independent from one another. The I_d current affects the I_q current, and vice-versa.

This coupling effect acts as a disturbance, which becomes prominent during transient conditions at high speed. To correct for this coupling effect, feed-forward decoupling is applied to each axis to remove the disturbance.

Torque voltage, $V_q = V_q + \omega L_d I_d$

Flux voltage, $V_d = V_d - \omega L_q I_q$

Assuming the torque inductance and the flux inductance are equal, ω represents the estimated speed from the PLL estimator output.

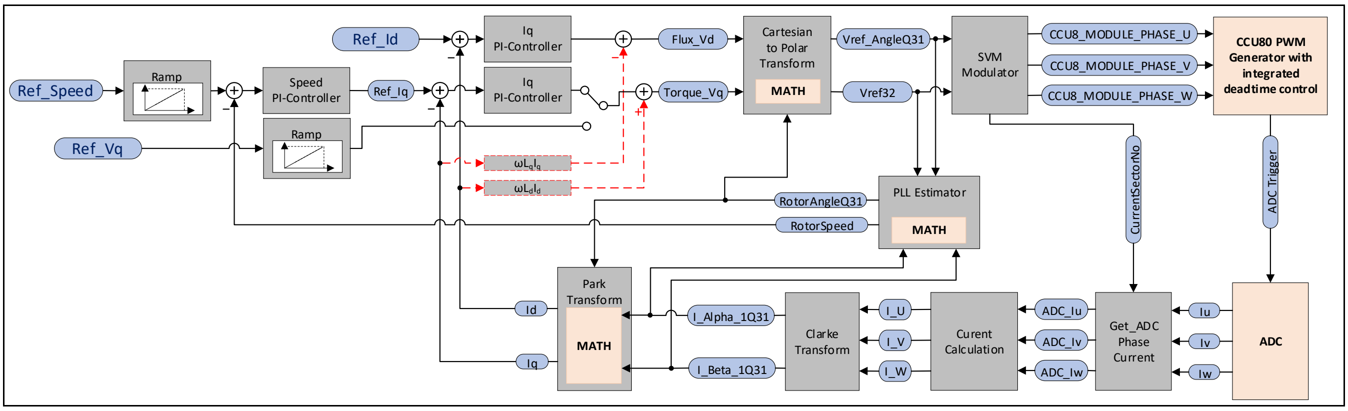

Figure 13 **FOC with D_q decoupling**

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 HW CORDIC co-processor in circular vectoring mode.

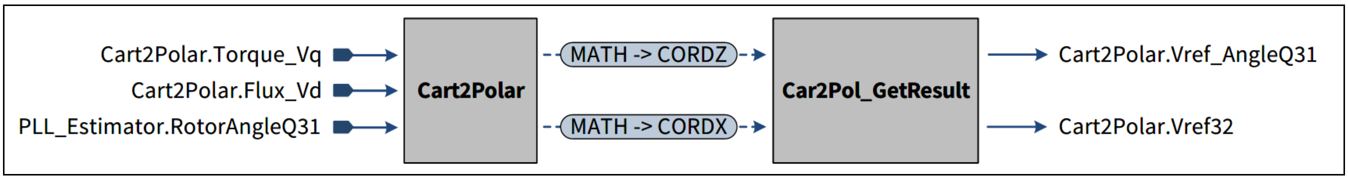

Figure 14 **Cartesian to polar transform**

Table 6 **XMC1000 CORDIC setting 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 to equations:

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

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

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

To improve the XMC™ execution the function is divided into two, for parallel processing; the starting CORDIC function and the read CORDIC result function. While the CORDIC coprocessor is making the calculation, the CPU can execute other SW functions such as the PI controller, for example, 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 transforms the stator voltage vectors into pulse-width modulation (PWM) signals (compare match values).

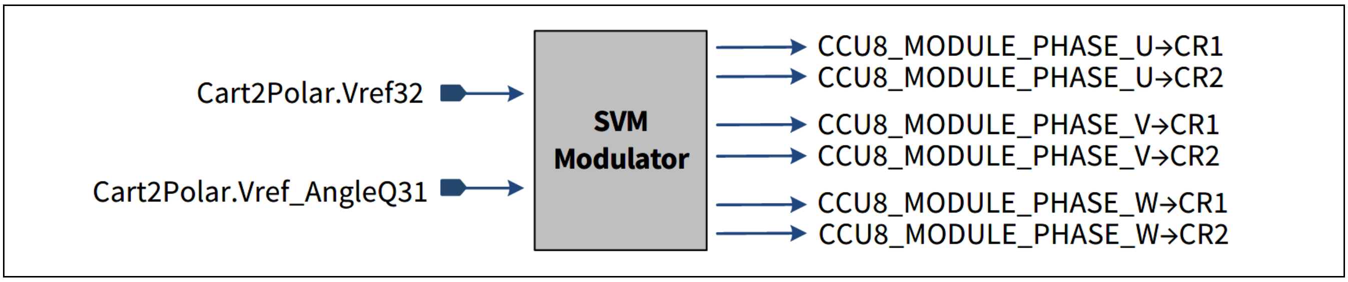

Figure 15 PWM SVM function

This PMSM FOC motor control SW supports:

- Seven-segment SVM
- Five-segment SVM

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

The default initial settings of the CCU80 module are for the Infineon EVAL_6EDL7141_FOC_3SH and EVAL_IMD700A_FOC_3SH evaluation board.

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

Parameters	Settings
Timer counting mode	Edge aligned mode
Shadow transfer on clear	Enabled
Prescaler mode	Normal
Passive level	Low
Asymmetric PWM	Enabled
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 ns)
Phase U slice period match interrupt event	Enabled
Trap interrupt event	Enabled

2.5.1 Seven-segment SVM

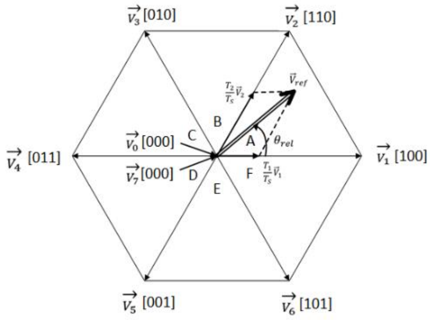

Figure 16 Seven-segment SVM

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

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

$$T_S = T_0 + T_1 + T_2$$

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

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

$$T_0 = T_S - T_1 - T_2$$

Where:

T_S Sampling period; T_S is equal to PWM 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

V_{DC} Inverter DC-link voltage

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

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

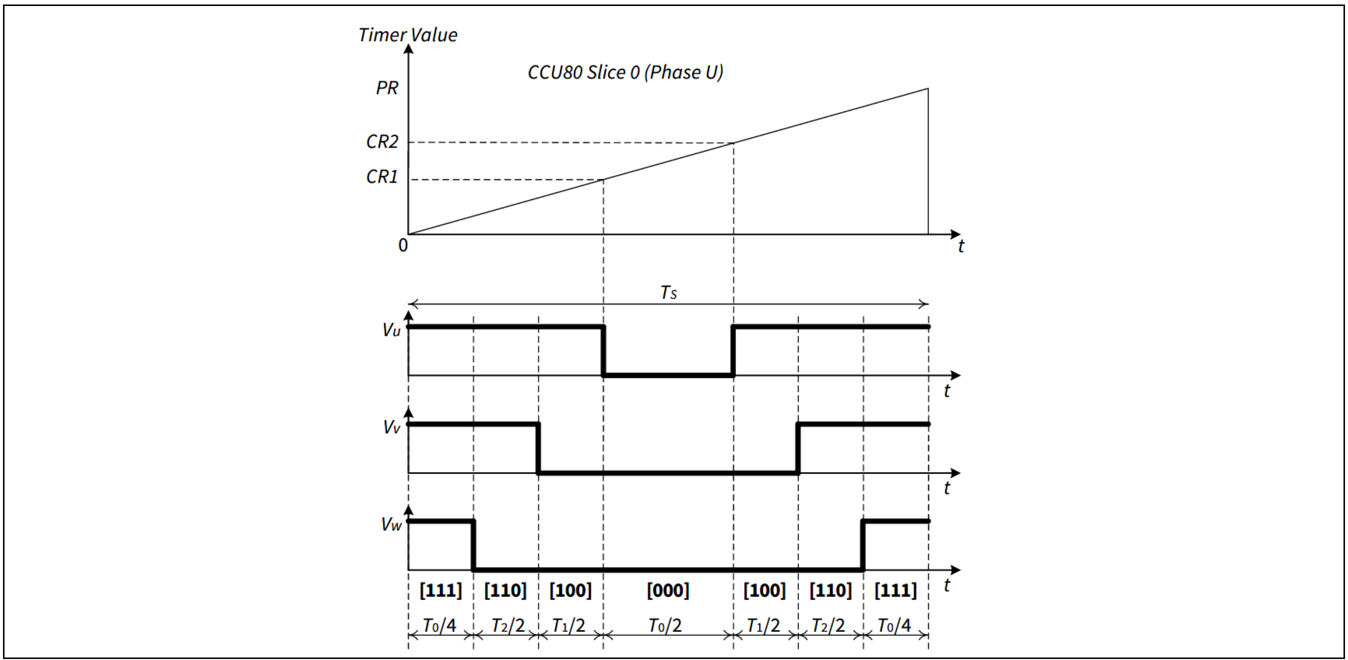

Figure 17 **Seven-segment SVM timing diagram in SVM sector A**

2.5.2 Five-segment SVM

The five-segment SVM uses the same equations as in the seven-segment SVM to calculate the T_0, T_1 and T_2 timing. In five-segment SVM, the zero vector is only $\vec{V}_0[000]$. Unlike in seven-segment SVM, the zero vectors are $\vec{V}_0[000]$ and $\vec{V}_7[111]$.

For example, in SVM sector A, the PWM on-time for phase U is $T_S - T_0$, phase V is $T_S - (T_0 + T_1)$ and phase W is zero.

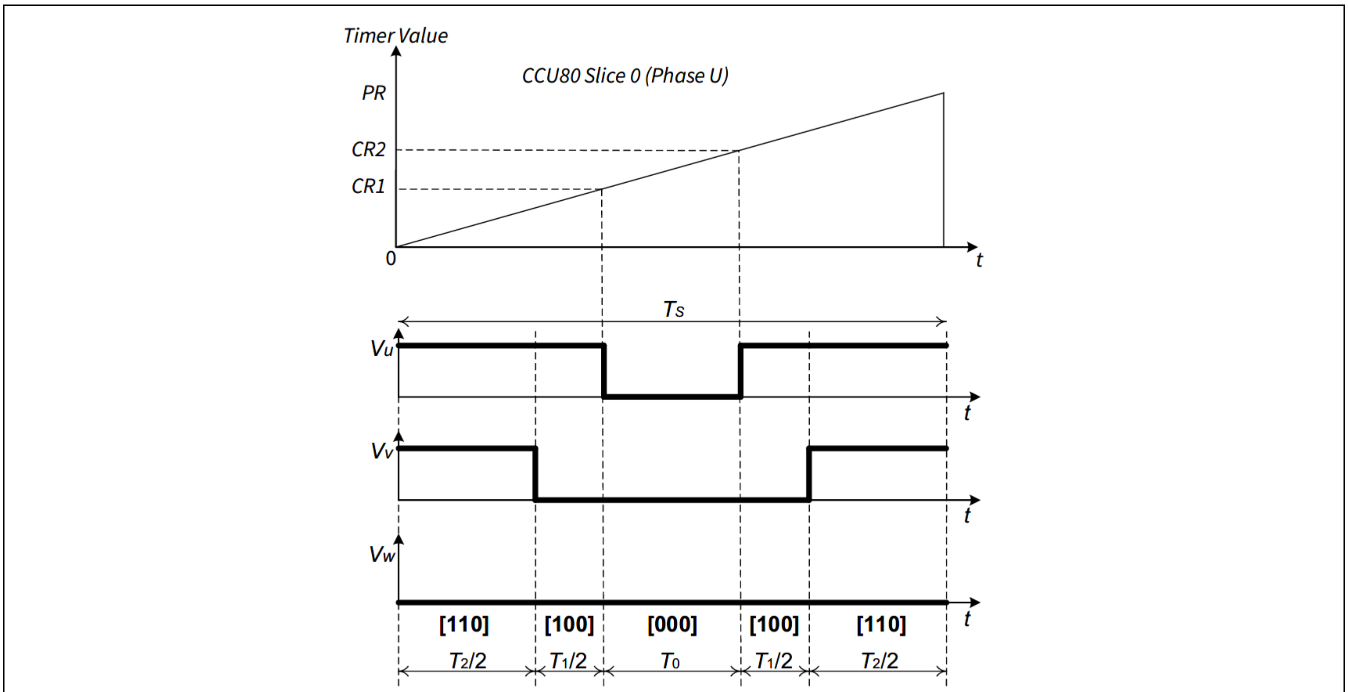

Figure 18 **Five-segment SVM timing diagram in SVM sector A**

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_input_config.h (refer to **chapter 7.2.3**).

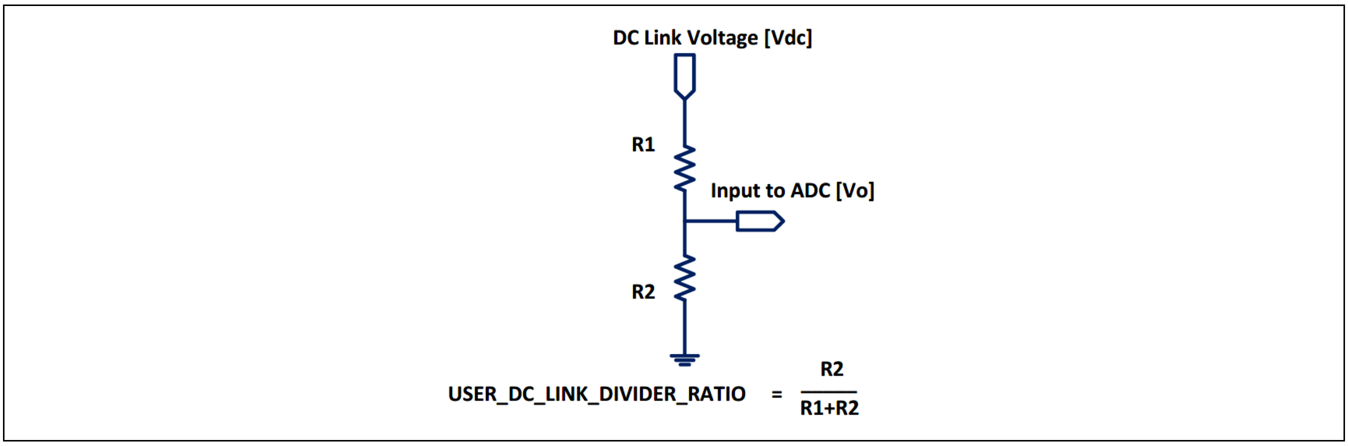

Figure 19 DC-link voltage divider

2.7 Clarke transform

In this module the phase currents ($I_{I_u}, I_{I_v}, I_{I_w}$) from the CS module are transformed into currents I_Alpha and I_Beta on the two-phase orthogonal reference frame.

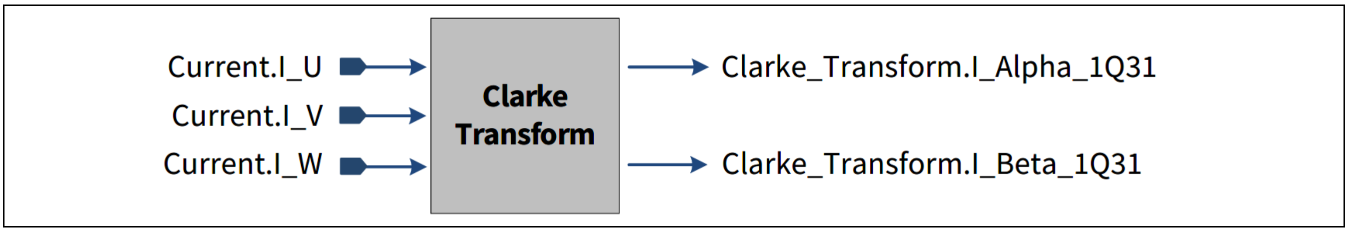

Figure 20 Clarke transform

Equations for the 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 (see **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.

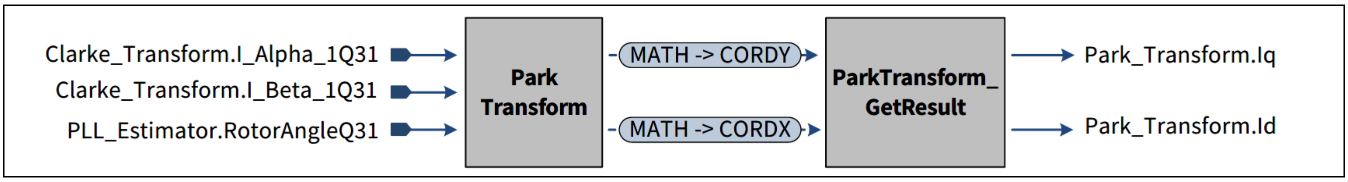

Figure 21 Park transform

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 HW module (refer to the XMC1400 reference manual [1]).

Scaling factors of I_q and I_d are based on the current scaling (see **chapter 2.10**).

Table 8 Parameter settings

Parameters	Settings
CORDIC control mode	Circular rotating mode
K	≈ 1.646760258121
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 SW supports the following protection schemes:

- Gate driver fault reporting pin
- OCP
- OVP/UVP
- SPI fault

The corresponding error status is set when either of the faults occurs; it changes the motor state machine to PMSM_FOC_MSM_ERROR state.

2.9.1 Gate driver fault reporting pin

The smart gate driver MOTIX™ 6EDL7141 contains many protection features:

- OCP for:
  - DVDD linear regulator
  - Buck converter
  - Motor leg shunt OCP
- Undervoltage lockout (UVLO) protection for:
  - Gate driver supply voltage, both high-side and low-side drivers
  - Supply voltage PVDD
  - DVDD linear regulator output voltage
  - Buck converter output voltage
- DVDD linear regulator overvoltage lockout (OVLO) protections
- Rotor locked detection based on Hall sensor inputs
- Configurable watchdog
- Overtemperature shutdown (OTS) and warning (OTW)
- OTP memory fault

When any of the events occur, the fault is asserted and the nFAULT pin is pulled low, reporting the fault to XMC1404 via a GPIO interrupt. In the ISR, the error status is set to PMSM_FOC_EID_NFAULT_FAULT, and the nBRAKE pin of the gate driver is activated.

Refer to the MOTIX™ IMD700A datasheet for more detail on MOTIX™ 6EDL7141 protections and fault handling.

2.9.2 Overcurrent protection

The Over Current Protection fault can be reported to the XMC1404 via nFault pin. The nFault pin voltage will be pull down and inform XMC1404 that a fault occurred which result in a braking action taking place. The threshold voltage and the shunt resistor value on the evaluation board will determine the tripping current. For example, the shunt resistor is 10mOhm while the OCP threshold voltage is 70mV, then the tripping current is 0.07/0.01 = 7A.

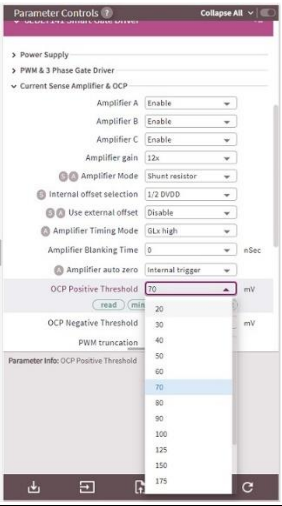

Figure 22 Using 6EDL7141's OCP threshold setting to program over current protection

2.9.3 Overvoltage/Undervoltage protection

The DC-link voltage is read in the function PMSM_FOC_SysMonitoring() every PWM period match interrupt. An error is generated if the DC-link voltage is higher than the overvoltage threshold or lower than the undervoltage threshold. The overvoltage threshold is set by MotorParam.OVERVOLTAGE_THRESHOLD variable while the undervoltage threshold is set by MotorParam.UNDERVOLTAGE_THRESHOLD variable. The error status is set as PMSM_FOC_EID_OVER_VOLT or PMSM_FOC_EID_UNDER_VOLT accordingly.

2.10 Scaling

PMSM FOC SW uses integers to represent real-world floating-point variables, such as angle, current and voltage. To provide the best resolution, the SW represents the physical value depending on the target value.

For example, the phase current is represented by 0 to 100 percent of the target value, where the target value is the maximum current that can be measured by the CS circuit.

The following equation shows the conversion of physical value to the norm value represented in the SW.

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

Table 9 Scaling used in PMSM FOC SW

Parameter	Scaling		Range	
	Target value [unit]	N	[%]	[hex]
SVM amplitude (V_{ref})	$N_{\_Vref\_(SVM)}$ [V]	15	0 to 100 percent	0x0 to 0x7FFF (> 0x7FFF with overmodulation)
Current I_U, I_V, I_W, I_q, I_d	$N_{\_I\_(\alpha,\beta)}$ [A]	15	-100 to 100 percent	0x8001 to 0x7FFF
Angle of rotor position, angle of space vector	360 degrees	16 + USER_RES_INC	0 to 360 degrees	0x0 to 0xXFFFF (for 16 + USER_RES_INC bit)
Speed	$N_{\max\_speed}$ [degree/second]	$log_2(\max\_speed\_integer)$	0 to 100 percent	0x0 to max_speed_integer

In the following sections, the calculations of the target values are described.

2.10.1 Scaling for SVM voltage

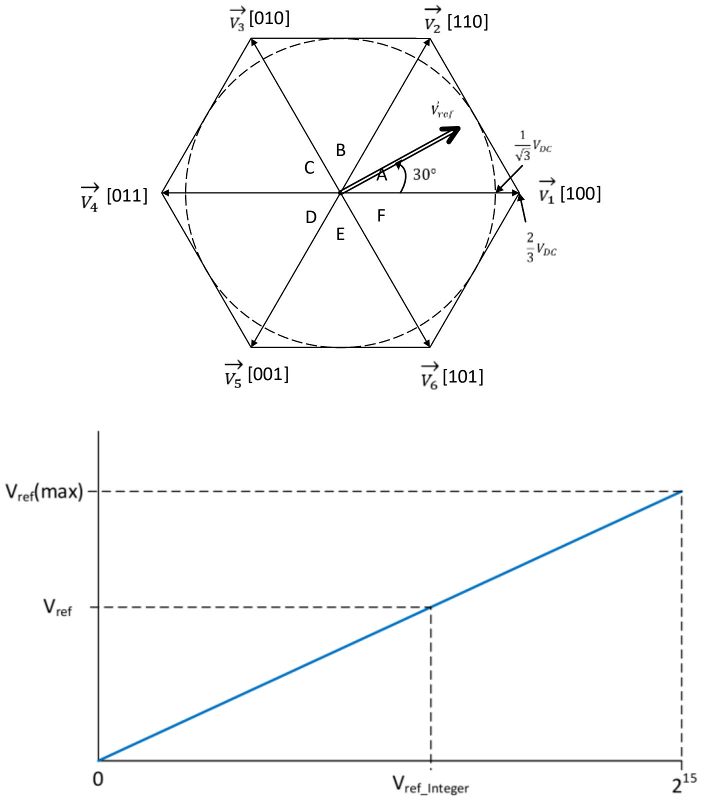

Figure 23 **SVM voltage scaling**

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

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

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

V_{DC} is inverter DC-link voltage, USER_VDC_LINK_V.

Example:

USER_VDC_LINK_V is 24.0f

$$N_{\_Vref\_SVM} = 13.86V$$

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

It is possible for $V_{ref\_integer}$ to exceed 0x7FFF when overmodulation occurred. It is limited to a maximum value of 37836 which is equivalent to $\frac{2}{3} V_{DC}$.

2.10.2 Scaling for phase current

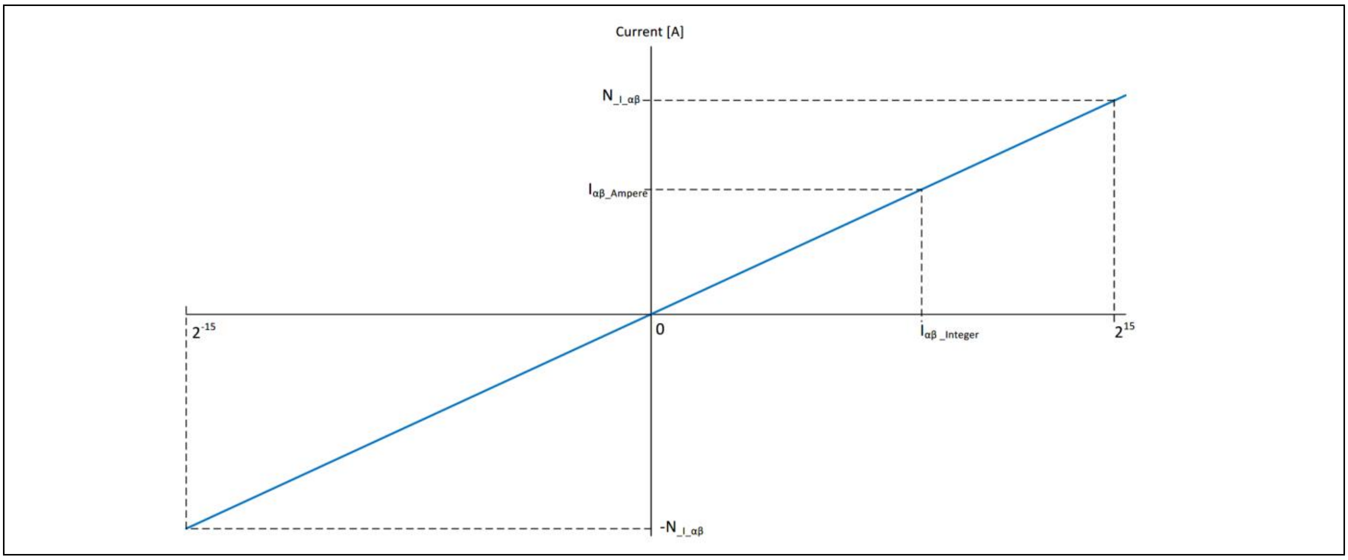

Figure 24 Phase current scaling

The target value of the current is the maximum current that can be measured by the CS 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.

2.10.3 Scaling for angle and speed

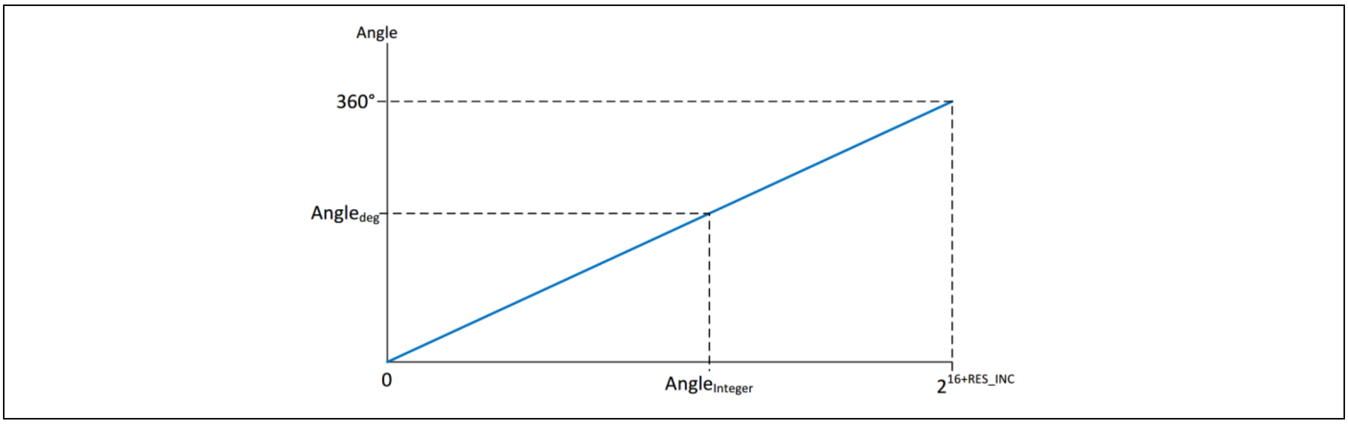

Figure 25 Angle scaling

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

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

Following the 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:

$$N_{\max\_speed} = \frac{USER\_SPEED\_HIGH\_LIMIT\_RPM \cdot USER\_MOTOR\_POLE\_PAIR}{60} \cdot 360\ degree/second$$

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

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

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

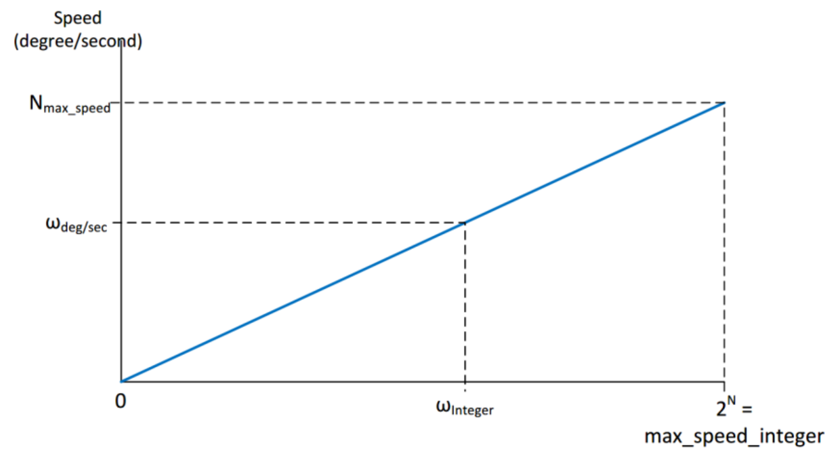

Figure 26 Speed scaling

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 SW integer, $\omega_{integer}$= 2,796

2.11 Determination of flux and torque current PI gains

The calculation of the PI gains is made 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 based 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 shows the simplified diagram after pole-zero cancellation.

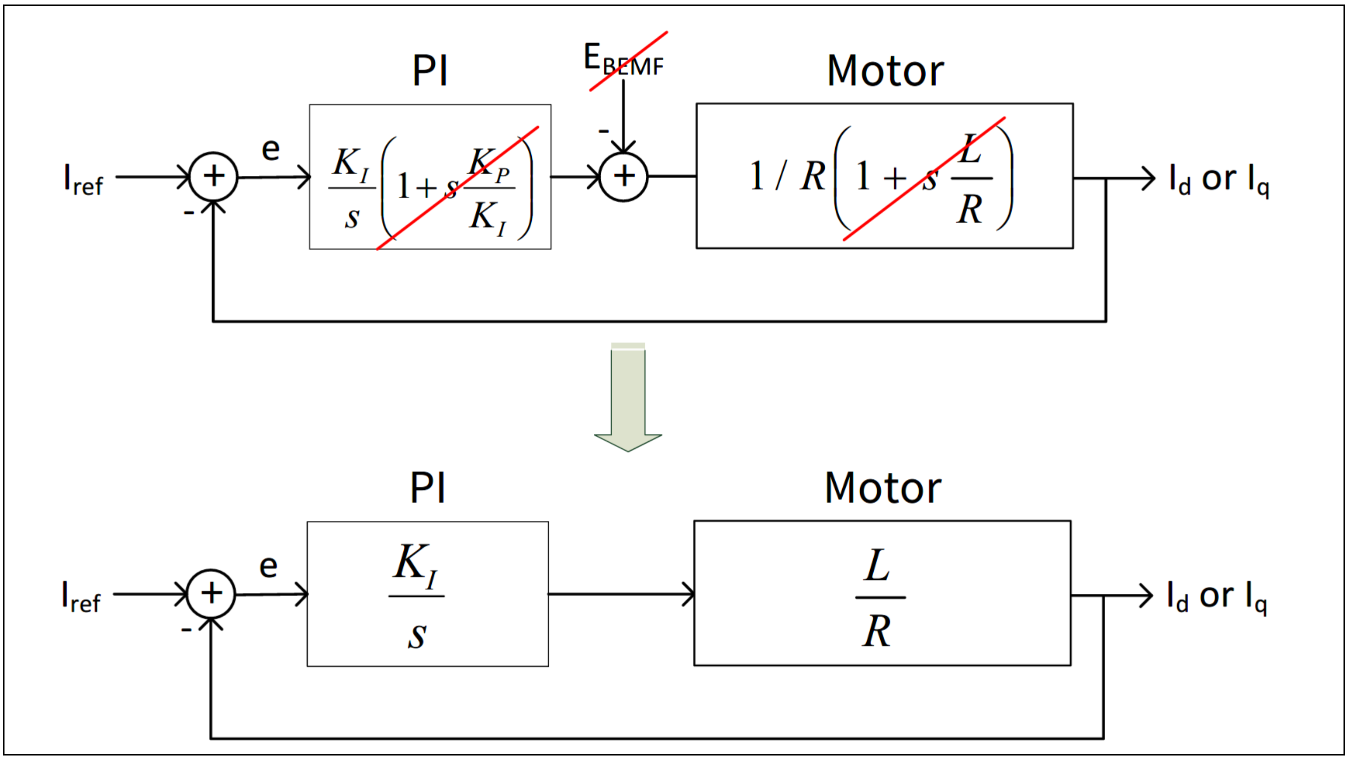

Figure 27 Simplified current control loop due to pole-zero cancellation

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
- 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™ HW optimized scaling factor.

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

3 Current sensing and calculation

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

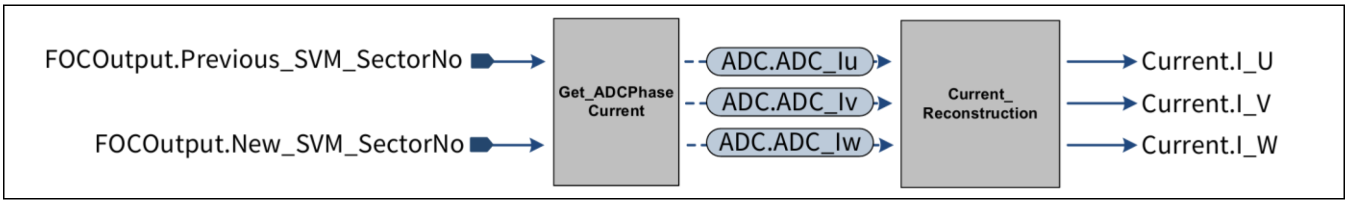

Figure 28 CS and calculation functions

Three-shunt CS technique is used in this SW.

The phase current 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 are listed in the **subchapter 3.2**.

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

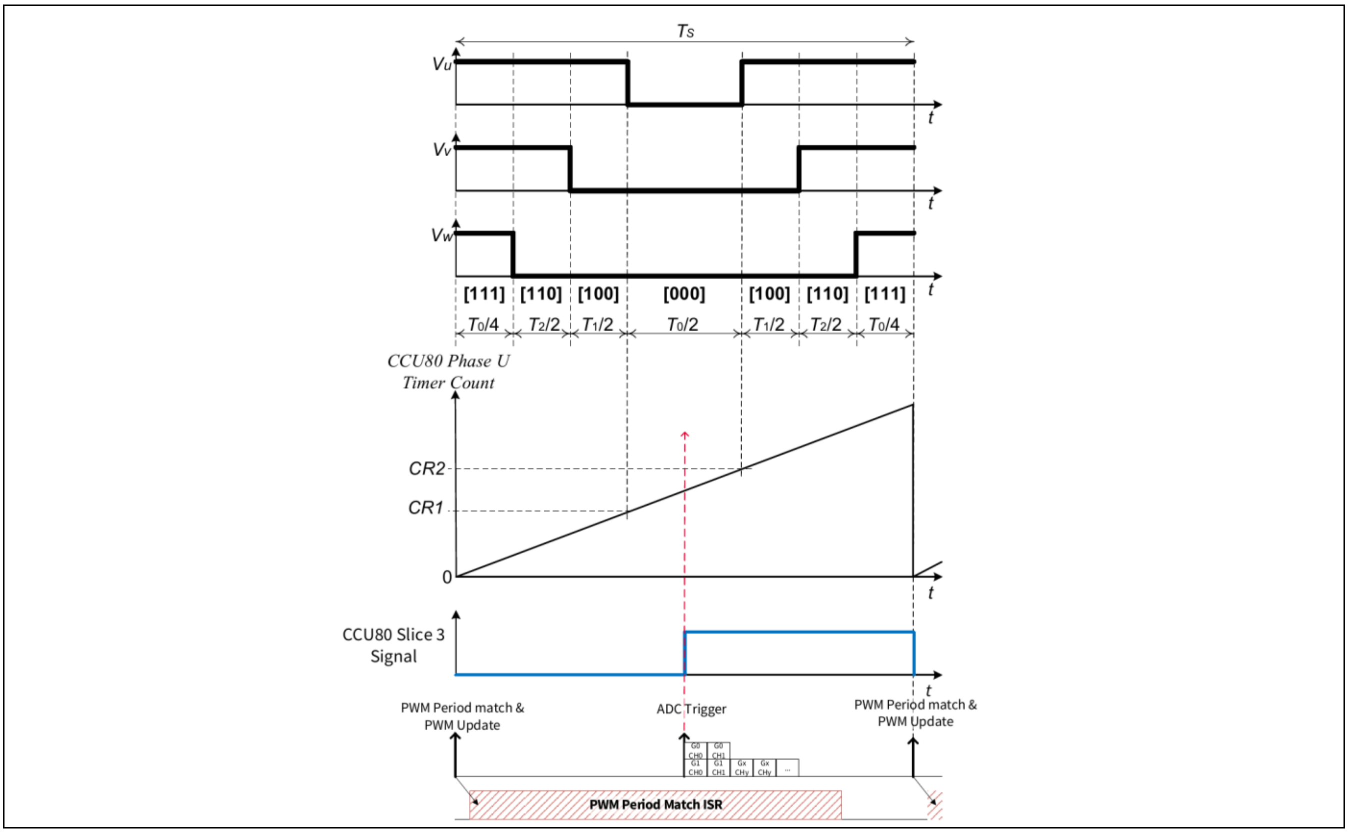

Figure 29 Three-shunt CS timing diagram using synchronous conversion ADC

3.1 Current sense amplifier gain setting

In the default application an op-amp is used to amplify the voltage drop above the current shunt to combine low power losses and high ADC accuracy. This method is supported by this SW.

MOTIX™ 6EDL7141 integrates three CS amplifiers that can be used to measure the current in the power inverter via shunt resistors. Single-, double- or triple-shunt measurements are supported, as shown in **Figure 30**.

CS_EN bitfield enables each CS amplifier individually. The output of the CS amplifiers can be connected to the ADC inputs (AINx pin) of MOTIX™ IMD700A via optional RC filters to remove high-frequency components. Gain and offset are generated internally and must be programmed via SPI commands.

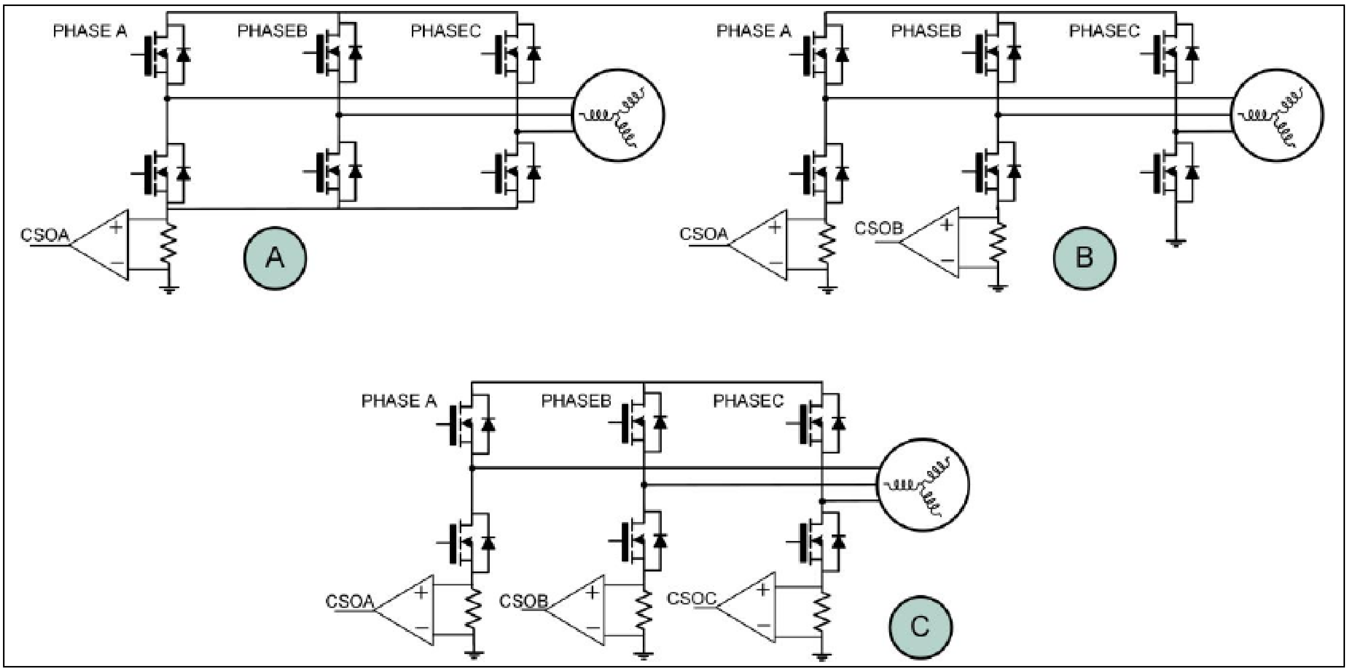

Figure 30 **Single- (A), dual- (B) and triple- (C) shunt CS configurations**

During system start-up, the user must program the desired gain of the CS amplifier. To do this, the user must write bitfield CS_GAIN_ANA to ensure digital programming of the gain and also write the CS_GAIN bitfield to set the actual gain. The actual value can be read in CS_GAIN_ST, which reports the current gain value programmed. Gain of the shunt amplifiers can be programmed to one of the following values: 4, 8, 12, 16, 20, 24, 32 or 64.

3.2 Three-shunt current sensing technique

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

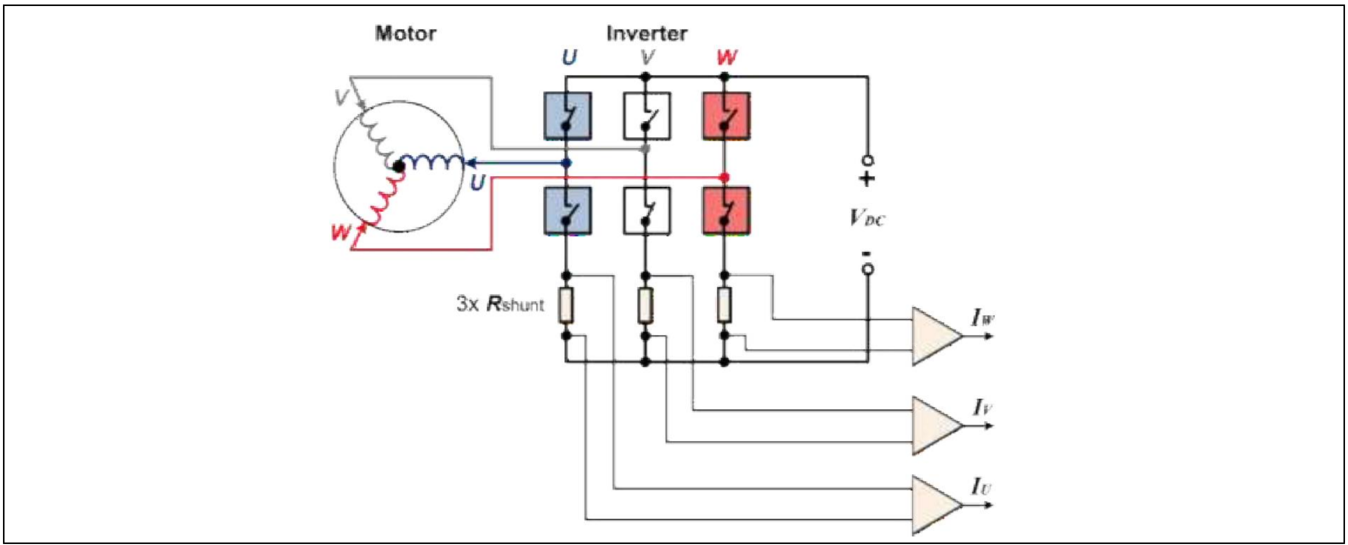

Figure 31 Three-shunt CS technique

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 – see **Figure 32**. The current will always flow through the shunt resistor when the low-side switch is on and the high-side switch is off.

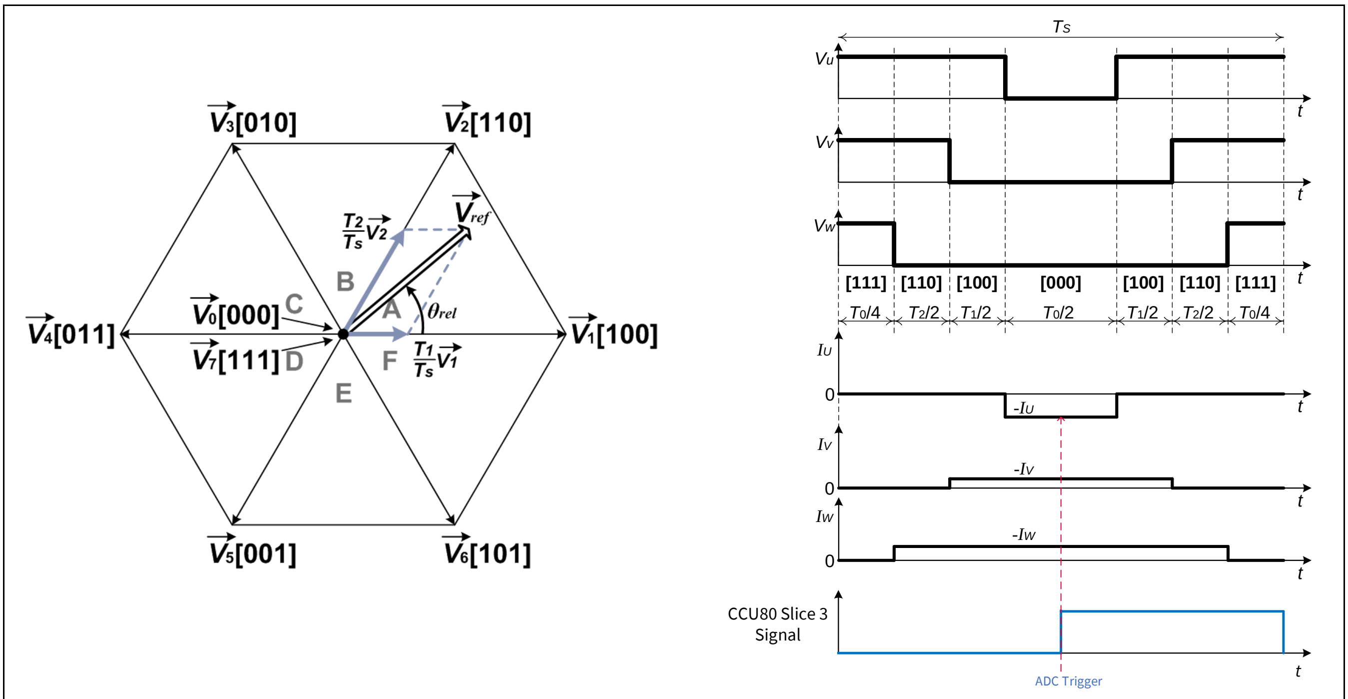

Figure 32 Three-shunt, three-phase current sensing in sector A

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

- I_U = (ADC_Bias_IU – VADC_Res_IU) * 2^4
- I_V = (ADC_Bias_IV – VADC_Res_IV) * 2^4
- I_W = (ADC_Bias_IW – VADC_Res_IW) * 2^4

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

Table 10 VADC initial settings for three-shunt

Parameter	Setting
Request source for three-shunt	Queue
Request source for other channels	Background scan
FIFO	Disabled
Source interrupt	Disabled
ADC conversion trigger signal	CCU80.ST3A (through gating select input)
ADC conversion trigger edge	Rising edge

Table 11 CCU80 slice 3 initial settings for three-shunt

Parameter	Setting
Timer counting mode	Edge aligned
Single-shot mode	Disabled
Period register	Same as period register value for three-phase PWM
Compare register channel 1	Half of period register value + driver delay
Compare register channel 2	Compare register channel 1 + 20U

3.2.1 Asynchronous theory

The term “asynchronous conversion” is related to the independence of the groups. This mode is an easy implementation and the benefits are a free group 1, no reload of the ADC and ease of understanding.

In the default configuration all three ADC inputs are sampled one after the other, and all three currents are measured one after each other. This mode does not require a reload of the ADC and is especially suitable if three ADCs are available (this is not true for XMC1000). You can manually distribute the channels to the groups. The main drawback is that the measurement time is up to double the time of an advanced implementation. Due to the required long current measurement window, it is recommended to use this implementation only for demonstration purposes.

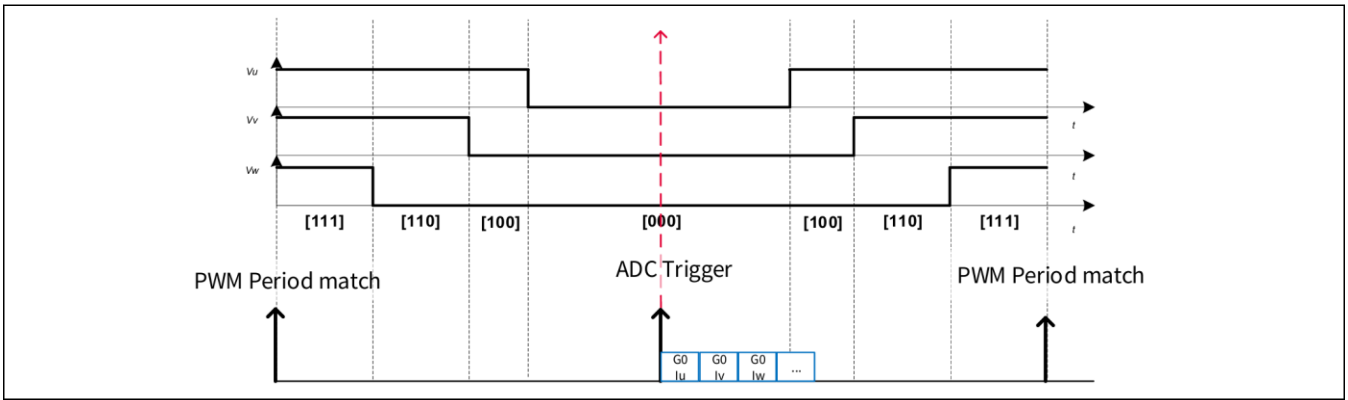

Figure 33 Asynchronous conversion sequence

3.2.2 Synchronous theory

This implementation uses three HW features of the XMC1400 family.

The first feature allows synchronized sampling and sequential conversion of two shunt currents. This improves the accuracy and reduces the minimum measurement window. Both VADC sample-and-hold units are used for this feature to measure two currents at the same time (for example phases U and V). This method is not impacted if another measurement runs in the background. This gives rise to the implementation name "synchronous conversion".

The second HW feature improves the measurement for large amplitudes. In three-phase leg-shunt current measurement the current is measured in the middle, where all high-side switches are off. This measurement window decreases when the amplitudes increase. Which phase has a small measurement window depends on the sector (see **Figure 34**).

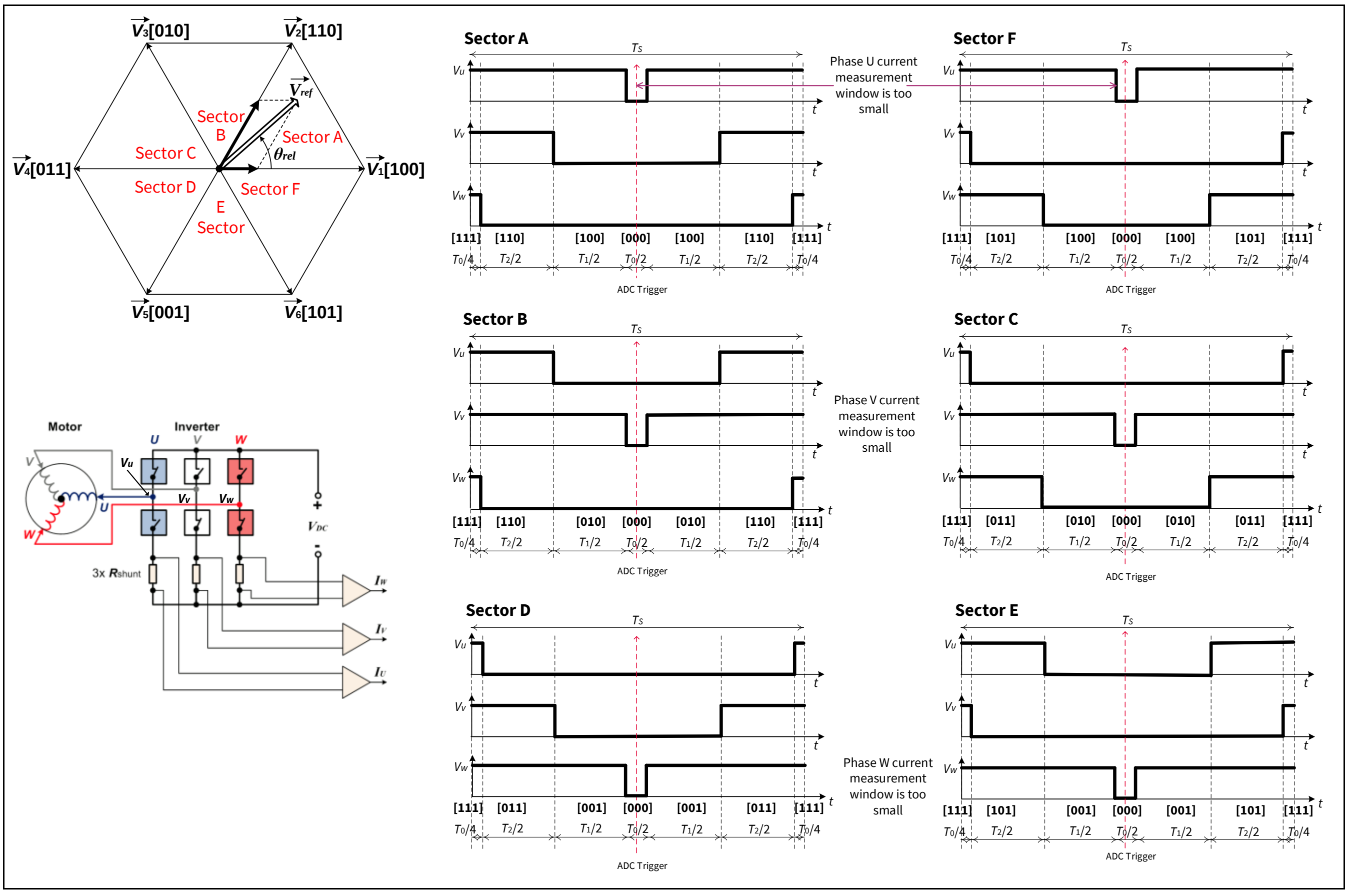

Figure 34 **SVM sector at large amplitudes**

The SW changes the sequence of measurement depending on the sector. Therefore, it can measure the two non-critical phase currents. For example, for sectors A and F it can assign I_V and I_W ADC channels.

The following table shows the synchronous measured phases per sector.

Table 12 Phase measurement per SVM sectors

SVM sectors	Shunt current measured
Sectors A and F	Phase W
	Phase V
Sectors B and C	Phase U
	Phase W
Sectors D and E	Phase V
	Phase U

The second HW feature is the alias feature of the ADC, which allows for fast changing of the sequence, so even large amplitudes can be measured.

Consequently, the SW discards the measurement of the third phase if the measurement window is smaller than the T0_THRESHOLD value. It is then switching from three-leg shunt measurement to two-leg shunt measurement. In the sensorless PMSM FOC SW, T0_THRESHOLD value is defined as 1344 which is equivalent to 1344 x (1/96MHz) = 14 us where 96MHz is the XMC1404's CCU8 module clock frequency.

In three areas (indicated by the red shaded area in **Figure 35**) the minimum measurement window falls below T_{min} at two phases. **Figure 35** shows which area can be measured with three- or two-shunt measurement.

T_{min} should set slightly more than t_s in addition of dead time. In the sensorless PMSM FOC SW, T_{min} is set to 200 * (1/USER_PCLK_FREQ_MHZ) us = 2.083 us. This value has taking account of the default 0.75us dead time and XMC1404's ADC sampling timing.

The XMC1404's ADC sampling time t_s is given by:

t_s = SST x t_{ADC} + 4 x t_{ADC} = 5 x t_{ADC} where SST is the ADC's short sampling time which set to 1 in the sensorless PMSM FOC SW and t_{ADC} is ADC module clock period which is (1/48MHz).

Hence, $T_{min} \geq 5 \times (1/48MHz) + 0.75\ us = (0.104 + 0.75)us = 0.854\ us$

In the three red shaded regions, current sensing with two shunt is not possible. So, there will be some phase current distortion as the current will not be measured. Furthermore, if the dead time duration change, T_{min} value should adjust accordingly.

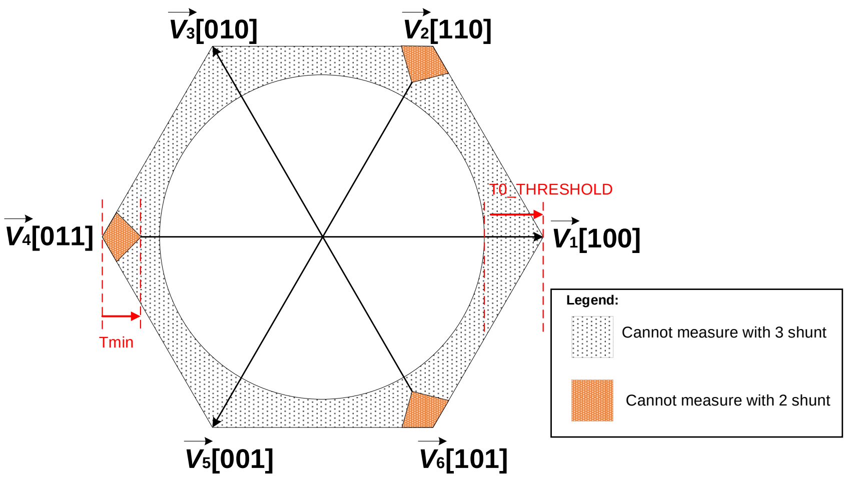

Figure 35 SVM two- or three-leg shunt unmeasurable areas

The switching of the parallel sampled phases requires that all three inputs (I_U, I_V, I_W) are available for both groups (G0 and G1). Normally this would double the pin consumption. The third HW feature of the XMC1400 family avoids this doubling by overlapping group channels. Up to four pins are accessible from both groups.

3.2.3 Synchronous implementation

Using the alias feature in the ADC module, we can assign different ADC input channels to be converted in parallel. Therefore we can measure the two least critical phase currents for all the SVM sectors. For sectors A and F, we assign I_V and I_W ADC channels.

The following table shows the synchronous measured phases per sector.

Table 13 Aliasing settings for SVM sectors

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

The implementations for all sectors are shown in the following figures. The CH0 of the master (G0) and the slave (G1) are measured synchronously. After the conversion the CH1 of the master (G0) and the slave (G1) is measured.

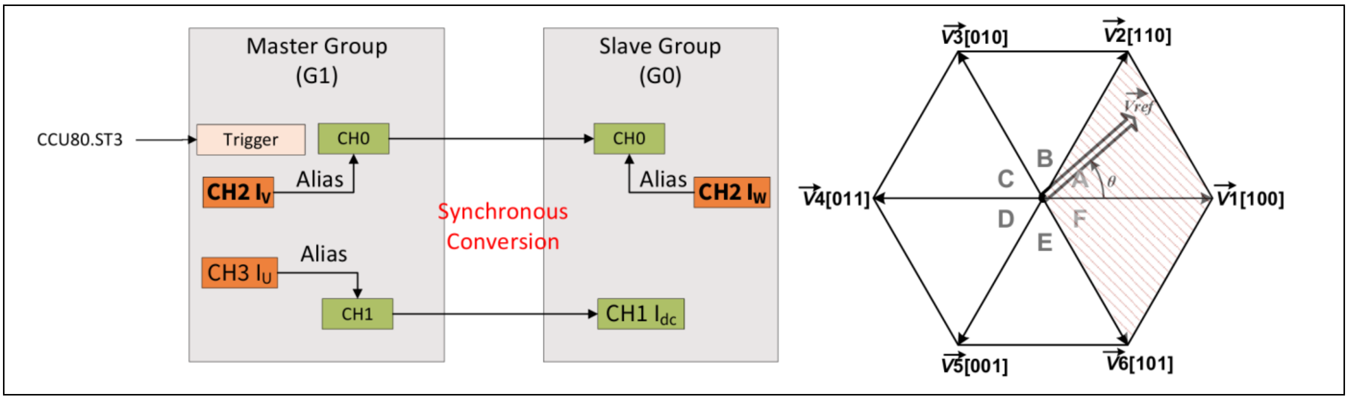

Figure 36 Synchronous conversion using alias feature – sectors A and F

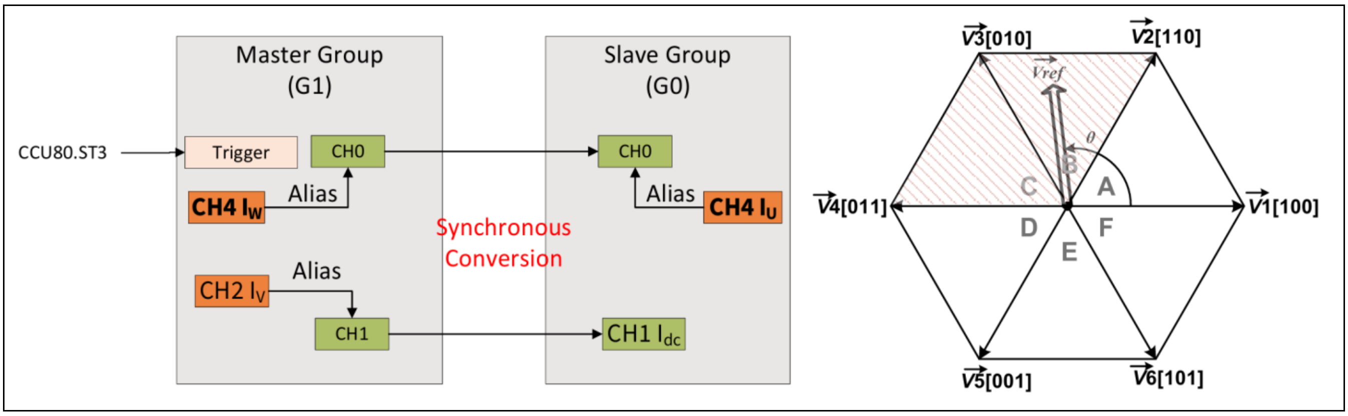

Figure 37 Synchronous conversion using alias feature – sectors B and C

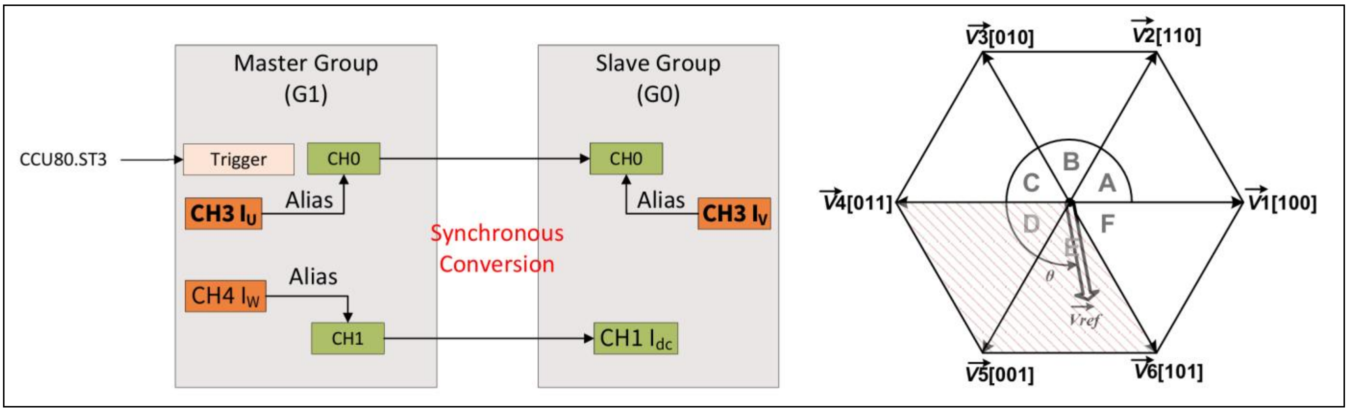

Figure 38 Synchronous conversion using alias feature – sectors D and E

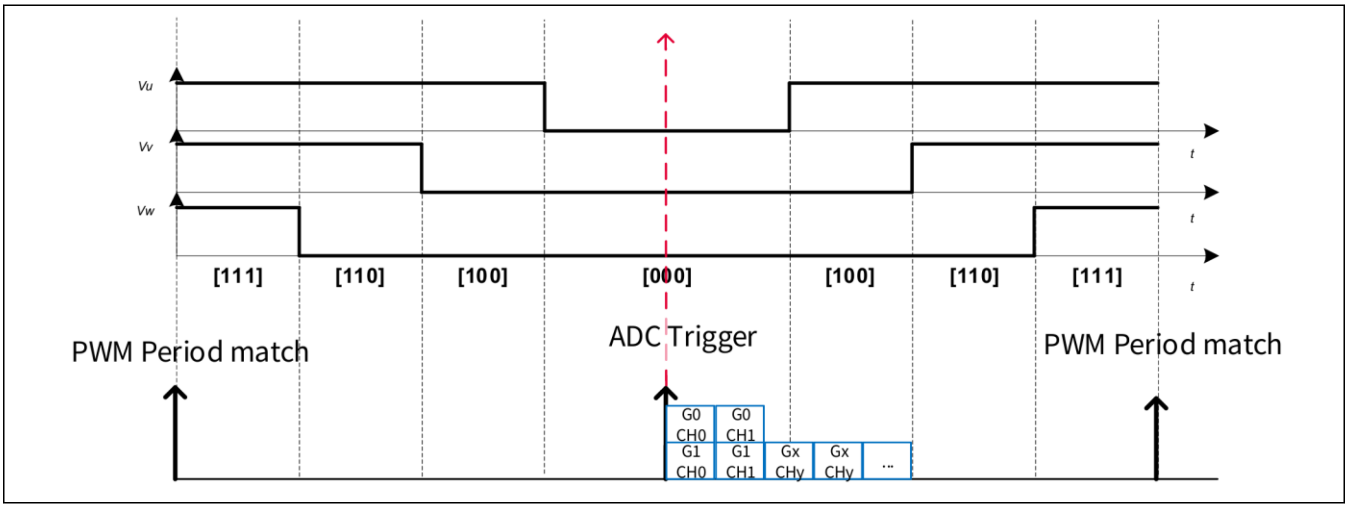

Figure 39 Synchronous conversion sequence

4 Motor speed and position feedback in sensorless FOC

The rotor speed and position feedback of the motor are determined in the PLL estimator SW library. This library contains the Infineon-patented IP and is provided as a compiled libPLL_Estimator.a file. The following are the list of APIs provided in the library.

Note: It is important that these APIs are called in the exact order indicated.

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)

Below is a brief description of each API and the required parameters.

Table 14 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 15 PLL_Imag_GetResult() function

Name	PLL_Imag_GetResult(PLL_EstimatorType* const HandlePtr)	
Description	This function reads 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 an input parameter for the second CORDIC calculation

Table 16 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 17 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	Used for PLL_Estimator

Table 18 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

Interrupt events and priorities in the PMSM FOC SW are listed in the following table.

Table 19 Interrupt priorities

Interrupt event	Priority	Comment
CTrap	0 (highest priority)	Not enable as CTRAP pin is not available on board
nFAULT	1	Gate driver nFAULT pin fault
Fast control loop	2	PWM period match interrupt
Slow control loop	3	SysTick interrupt

5.1 Fast control loop

The PMSM_FOC state machine is executed in phase U PWM frequency period match ISR.

An example of the flow of the PWM period match interrupt is shown in **Figure 40**. This example shows the flow of the FOC direct start-up control scheme. The CS technique chosen is three-shunt synchronous conversion.

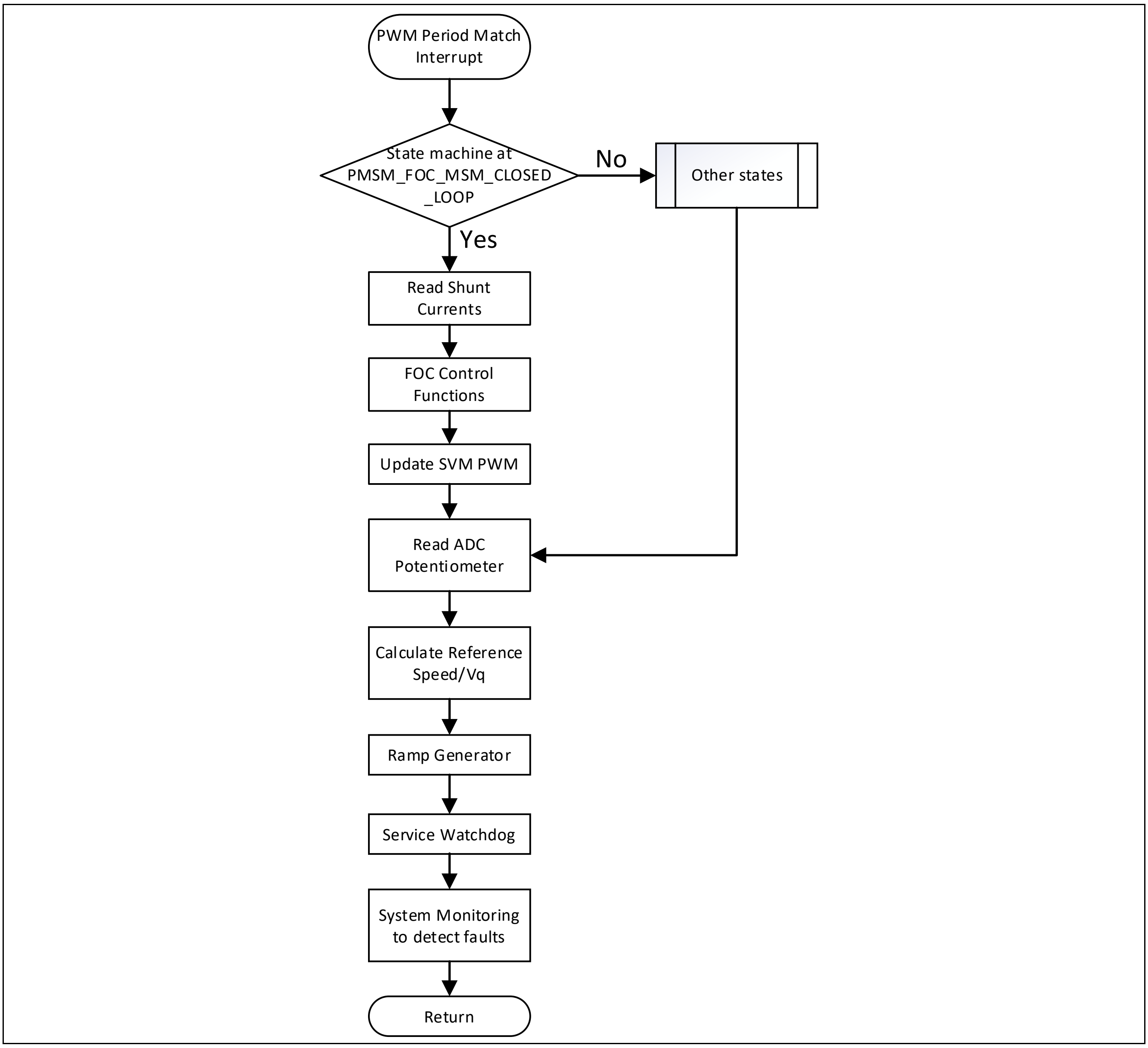

Figure 40 Fast control loop interrupt service routine flowchart

5.2 Slow control loop

This loop is used for slower tasks such as communication or a watchdog service. The trigger is generated from an independent timer. This means it is not mandatory that the frequency is synchronous with the PWM period match interrupt. For deterministic reasons its frequency is intended to be a fraction of USER_CCU8_PWM_FREQ_HZ (default 20 kHz). The frequency is defined in the macro USER_SLOW_CTRL_LOOP_PERIOD_uS in the file pmsm_foc_user_input_config.h. The default value is 1 kHz (refer to **chapter 7.2.3**).

5.3 Gate driver nFAULT

MOTIX™ 6EDL7141 provides many protection features for improving application robustness during adverse conditions, such as monitoring of power supply voltages as well as system parameters. The failure behavior, threshold voltages and filter options of the MOTIX™ 6EDL7141 device are adjustable via SPI. It also mentions signals such as motor currents, gate drive voltages and currents, and device temperature. When a fault occurs, the device can be configured to stop driving and pull the nFAULT pin low, in order to prevent MOSFET damage and motor overheating. Those signals are connected internally to XMC1404 to inform the processor that a fault has occurred. The microcontroller can request more information on the fault via SPI commands.

In this SW, when the nFAULT pin is pulled low, an interrupt is triggered via the ERU peripheral. In the ISR PMSM_FOC_DRIVER_nFAULT_ISR(), the nBRAKE pin, which is internally connected, is pulled low to enable the brake feature of MOTIX™ 6EDL7141. The error state is set to PMSM_FOC_EID_NFAULT_FAULT and the motor state is set to PMSM_FOC_MSM_ERROR.

6 Motor state machine

The PMSM FOC SW has an internal state machine:

PMSM_FOC_MSM_IDLE

This is the first state entered after power-on or SW reset. In this state the inverter is disabled. The motor control peripherals, the PI controller parameters and the variables are initialized. Then it changes to state PMSM_FOC_MSM_STOP_MOTOR.

PMSM_FOC_MSM_STOP_MOTOR

This is the state that the state machine finally stays at when there is no running command. Exit from this state occurs when the motor start command is received. It also changes to state PMSM_FOC_MSM_ERROR when an error occurs.

PMSM_FOC_MSM_BRAKE_BOOTSTRAP

This state is entered from PMSM_FOC_MSM_STOP_MOTOR when the motor start command is received. The bootstrap feature is implemented in this state. The bias voltage of the ADC pins that are connected to the motor phase currents are read to improve the phase current measurement. If the ROTOR_IPD_PRE_ALIGNMENT feature is enabled, it changes to PMSM_FOC_MSM_PRE_POSITIONING. Otherwise, it changes to PMSM_FOC_MSM_CLOSED_LOOP when the start-up method is MOTOR_STARTUP_DIRECT_FOC, or to PMSM_FOC_MSM_VF_OPENLOOP when the start-up method is MOTOR_STARTUP_VF_OPEN_LOOP.

PMSM_FOC_MSM_PRE_POSITIONING

This state is for both direct FOC start-up and VF open-loop start-up. 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_ROTOR_PRE_ALIGNMENT_VOLTAGE_V, for a specific time USER_ROTOR_PRE_ALIGNMENT_TIME_MS. These macros are defined in the file pmsm_foc_user_input_config.h file.

PMSM_FOC_MSM_VF_OPENLOOP

In this state the motor starts in V/F open-loop control mode.

Exit from this state occurs when the motor speed reaches the start-up threshold speed defined in the macro USER_VF_TRANSITION_SPEED_RPM.

PMSM_FOC_MSM_CLOSED_LOOP

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

PMSM_FOC_MSM_ERROR

This state is entered when an error occurs. The state machine exits from this state when a clear error command is received.

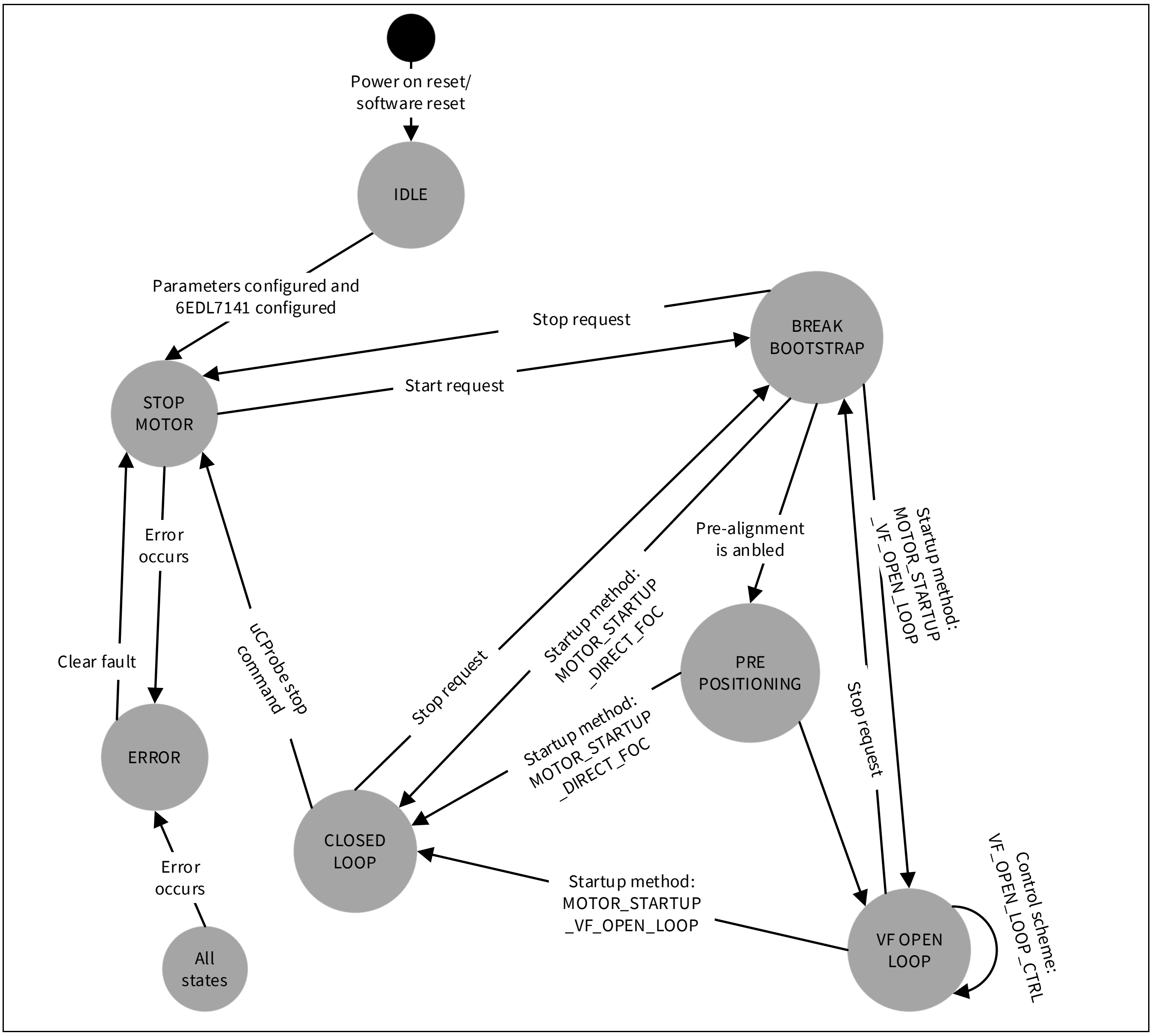

Figure 41 **PMSM FOC state machine**

7 Configuration

The default configuration of the parameters in the PMSM FOC SW is set based on the evaluation board EVAL_IMD700A_FOC_3SH and the BLDC motor with 24V rated voltage, 3.47A rated phase current and rated speed of 4000 rpm. The configuration can be split into two sections:

- MCU configuration
  - The MCU resource allocation is configured in this section. Configurations such as pinout and peripheral allocation can be found here.
- User configuration
  - General configurations such as FOC scheme, HW board parameters and the motor parameters can be configured in this section.

7.1 MCU configuration

The MCU resource allocation is configured in this section. Configurations such as pinout and peripheral allocation can be found here. All configuration options are available in the file PMSM_FOC\Configuration\pmsm_foc_mcuhw_params.h.

GPIO resources configuration

```
#define nFAULT_PIN              P3_4 /* Active Low */
#define nBRAKE_PIN              P1_3 /* Active Low */
#define INVERTER_EN_PIN         P0_7  /* Active High to enable gate driver*/
#define BRAKE_EN_PIN            P1_3  /* Active Low for motor braking*/
#define AUTO_ZERO_PIN           P1_2  /* Input pin to control Auto-Zero function */
```

- Interconnection pin assignment for the device. These pins should not be modified because they are internally connected between XMC1404 and MOTIX™ 6EDL7141.

```
#define DRV_CLK_EN_PIN          P2_13  /* Watchdog clock on EN_DRV pin */
#define MOTOR_DIR_INPUT_PIN     P4_10
```

- Pin assignment for control functions.

```
#define PHASE_U_HS_PIN          P3_3
#define PHASE_U_HS_ALT_SELECT   XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5
#define PHASE_U_LS_PIN          P3_2
#define PHASE_U_LS_ALT_SELECT   XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5
#define PHASE_V_HS_PIN          P3_1
#define PHASE_V_HS_ALT_SELECT   XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5
#define PHASE_V_LS_PIN          P3_0
#define PHASE_V_LS_ALT_SELECT   XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5
#define PHASE_W_HS_PIN          P1_0
#define PHASE_W_HS_ALT_SELECT   XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5
#define PHASE_W_LS_PIN          P1_1
#define PHASE_W_LS_ALT_SELECT   XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT5
```

- CCU80 PWM phase high-side and low-side output pin assignment, and XMC™ alternate output pin register value setting for PWM output.

CCU8 resources configuration

```
#define CCU8_MODULE                CCU80
#define CCU8_MODULE_NUM            (0U)
#define CCU8_SLICE_PHASE_U         CCU80_CC81
#define CCU8_SLICE_PHASE_U_NUM     (1U)
#define CCU8_SLICE_PHASE_V         CCU80_CC82
#define CCU8_SLICE_PHASE_V_NUM     (2U)
#define CCU8_SLICE_PHASE_W         CCU80_CC80
#define CCU8_SLICE_PHASE_W_NUM     (0U)
#define CCU8_SLICE_ADC_TR          CCU80_CC83
#define CCU8_SLICE_ADC_TR_NUM      (3U)
```

- Assign the XMC™ CCU8 module for SVM generation.
- Assign the timer slice of the CCU8 module for phase U, V and W PWM generation.
- Assign the CCU8 module timer slice for the ADC conversion trigger (TR).

VADC resources configuration

```
/* For simultaneous sampling */
#define VADC_I1_GROUP            VADC_G1
#define VADC_I1_CHANNEL          (0U)
#define VADC_I1_RESULT_REG       (0U)
#define VADC_I2_GROUP            VADC_G0
#define VADC_I2_CHANNEL          (0U)
#define VADC_I2_RESULT_REG       (0U)
#define VADC_I3_GROUP            VADC_G1
#define VADC_I3_CHANNEL          (1U)
#define VADC_I3_RESULT_REG       (1U)
#define VADC_I4_GROUP            VADC_G0
#define VADC_I4_CHANNEL          (1U)
#define VADC_I4_RESULT_REG       (1U)
```

- Configure synchronized conversions for parallel sampling.

```
/* Motor Phase U VADC define */
#define VADC_IU_G1_CHANNEL       (3U)        /* P2.11, VADC group1 channel 3 */
#define VADC_IU_G0_CHANNEL       (4U)        /* P2.11, VADC group0 channel 4 */
```

- Configuration for synchronous conversion of IU. The channels for IU from G0 and G1 are equivalent to the same specific pin.

```
#define VADC_IU_GROUP          VADC_G1
#define VADC_IU_GROUP_NO       (1U)
#define VADC_IU_CHANNEL        (3U)        /* P2.11, VADC group1 channel 3 */
#define VADC_IU_RESULT_REG     (3U)
```

- Configuration for asynchronous conversion of IU. The channel for IU from G1 is equivalent to a specific pin.

```
/* Motor Phase V VADC define */
#define VADC_IV_G1_CHANNEL     (2U)        /* P2.10, VADC group1 channel 2 */
#define VADC_IV_G0_CHANNEL     (3U)        /* P2.10, VADC group0 channel 3 */
```

- Configuration for synchronous conversion of IV. The channels for IV from G0 and G1 are equivalent to the same specific pin.

```
#define VADC_IV_GROUP          VADC_G1
#define VADC_IV_GROUP_NO       (1U)
#define VADC_IV_CHANNEL        (2U)        /* P2.10, VADC group1 channel 2 */
#define VADC_IV_RESULT_REG     (2U)
```

- Configuration for asynchronous conversion of IV. The channel for IV from G1 is equivalent to a specific pin.

```
/* Motor Phase W VADC define */
#define VADC_IW_G1_CHANNEL     (4U)        /* P2.9, VADC group1 channel 4 */
#define VADC_IW_G0_CHANNEL     (2U)        /* P2.9, VADC group0 channel 2 */
```

- Configuration for synchronous conversion of IW. The channels for IW from G0 and G1 are equivalent to the same specific pin.

```
#define VADC_IW_GROUP          VADC_G1
#define VADC_IW_GROUP_NO       (1U)
#define VADC_IW_CHANNEL        (4U)        /* P2.9, VADC group1 channel 4 */
#define VADC_IW_RESULT_REG     (4U)
```

- Configuration for asynchronous conversion of IW. The channel for IW from G1 is equivalent to a specific pin.

```
/* DC link voltage VADC define */
#define VADC_VDC_GROUP         VADC_G1
#define VADC_VDC_GROUP_NO      (1U)
#define VADC_VDC_CHANNEL       (5U)       /* P2.3 VADC group1 channel 5 */
#define VADC_VDC_RESULT_REG    (5U)
```

- Configuration for conversion of DC-link voltage. The channel for V DC from G1 is equivalent to a specific pin.

```
#if((USER_OVERCURRENT_PROTECTION == ENABLED) && (USER_OVER_CURRENT_DETECTION_SOURCE
== DC_LINK_CURRENT_SENSE))
/* DC link current VADC define */
```

```
#define VADC_IDC_GROUP        VADC_G1
#define VADC_IDC_GROUP_NO     (1U)
#define VADC_IDC_CHANNEL      (6U)        /* P2.4 VADC group1 channel 6 */
#define VADC_IDC_RESULT_REG   (6U)
#endif
```

– Configuration for conversion of DC-link current. It is only implemented when the OCP feature is enabled and a DC-link current shunt resistor is connected at P2.6.

```
/* T_Sense Temp sensor VADC define*/
#define VADC_TEMP_GROUP       VADC_G1
#define VADC_TEMP_GROUP_NO    (1U)
#define VADC_TEMP_CHANNEL     (6U)      /* P2.4 VADC group1 channel 6 */
#define VADC_TEMP_RESULT_REG  (6U)
/* Potentiometer VADC define*/
#define VADC_POT_GROUP        VADC_G1
#define VADC_POT_GROUP_NO     (1U)
#define VADC_POT_CHANNEL      (7U)       /* P2.5 VADC group1 channel 7 */
#define VADC_POT_RESULT_REG   (7U)
/** Configuration Macros for reference kit select Channel */
#define KIT_SELECT_GRP        (VADC_G0)
#define KIT_SELECT_GRP_NO     (0U)
#define KIT_SELECT_CH_NUM     (0U)     /* P2.6, G0CH0 */
#define KIT_SELEC_RESULT_REG   (0U)
```

– Configuration for conversion of temperature sensor, potentiometer and reference kit selection. The channels from G0 and G1 are equivalent to specific pins.

```
/* VADC Group 0 Alias channel 0 and channel 1 */
#define VADC_G0_CHANNEL_ALIAS0  VADC_IW_G0_CHANNEL
#define VADC_G0_CHANNEL_ALIAS1  0
/* VADC Group 1 Alias channel 0 and channel 1 */
#define VADC_G1_CHANNEL_ALIAS0  VADC_IV_G1_CHANNEL
#define VADC_G1_CHANNEL_ALIAS1  VADC_IU_G1_CHANNEL
```

– Configuration for the initial conversion. The channels from G0 and G1 are relative to the sector of SVM. It is aligned with the principle of synchronous conversion theory.

Note: The connection between group channel number and pin is visible in the reference manual.

```
#define INTERNAL_OP_GAIN          DISABLED
#if(INTERNAL_OP_GAIN == ENABLED)
#define OP_GAIN_FACTOR            (3U)
```

```
#elif(INTERNAL_OP_GAIN == DISABLED)
#define OP_GAIN_FACTOR            (12U)
#endif
```

- Configuration for the CS op-amp gain. When the op-amp in the MOTIX™ 6EDL7141 is used for CS, the XMC1404 internal op-amp should be disabled. This OP_GAIN_FACTOR must be the same as configured for MOTIX™ 6EDL7141.

NVIC interrupt resources configuration

```
#define CCU80_0_IRQn                                    IRQ25_IRQn
#define TRAP_IRQn                                       IRQ26_IRQn
#define FAULT_PIN_ERU_IRQn                              IRQ6_IRQn
/* NVIC ISR handler mapping */
#define PMSM_FOC_FCL_ISR                                IRQ25_Handler
#define PMSM_FOC_SCL_ISR                                SysTick_Handler
#define PMSM_FOC_CTRAP_ISR                              IRQ26_Handler
#define PMSM_FOC_DRIVER_nFAULT_ISR                      IRQ6_Handler
```

- Configuration for interrupt resources.

NVIC interrupt priority configuration

```
#define PMSM_FOC_FCL_NVIC_PRIO                          (2U)
#define PMSM_FOC_SCL_NVIC_PRIO                          (3U)
#define PMSM_FOC_CTRAP_NVIC_PRIO                        (0U)
#define PMSM_FOC_FAULT_NVIC_PRIO                        (1U)
```

- Configuration for interrupt priority; 0 is the highest priority and 3 is the lowest.

7.2 User configuration

General configurations such as FOC scheme, HW board parameters, and the motor parameters can be configured in this section. All configuration options are available in the file PMSM_FOC\Configuration\pmsm_foc_user_input_config.h.

7.2.1 Board selection

```
#define    MOTOR0_PMSM_FOC_BOARD    (EVAL_6EDL7141_FOC_3SH)
```

- Select a predefined HW board.

7.2.2 Motor selection

```
#define    MOTOR0_PMSM_FOC_MOTOR    (CUSTOM_FOC_MOTOR)
```

- Select a motor.

7.2.3 System group

#define USER_SLOW_CTRL_LOOP_PERIOD_uS (1000.0 F)

- Slow control loop scheduler interrupt period.

```
#define USER_IDC_ADC_BIAS                        (2048)
#define USER_IU_ADC_BIAS                         (2048)
#define USER_IV_ADC_BIAS                         (2048)
#define USER_IW_ADC_BIAS                         (2048)
```

- CS bias. It depends on the configuration of MOTIX™ 6EDL7141.

```
#define VOLTAGE_DIVIDER_R_HIGH                   (75.0f)
#define VOLTAGE_DIVIDER_R_LOW                    (7.87f)
#define USER_VDC_LINK_DIVIDER_RATIO
(VOLTAGE_DIVIDER_R_LOW/(VOLTAGE_DIVIDER_R_HIGH+VOLTAGE_DIVIDER_R_LOW))
```

- DC-link voltage sensing resistor-divider high-side and low-side value. It depends on the actual value on the predefined board. The calculation of the ratio for DC-link voltage divider should not be changed. It is fixed as R2/(R2+R1).

```
#define USER_DRIVERIC_DELAY_US                   (0.2f)
```

- Configuration for the driver IC propagation delay. It affects the ADC trigger point.

```
#define USER_BOOTSTRAP_PRECHARGE_TIME_MS         (100U)
```

- Initial bootstrap pre-charging time.

```
#define USER_INVERTER_ENABLE_LEVEL               XMC_GPIO_OUTPUT_LEVEL_HIGH
#define USER_INVERTER_DISABLE_LEVEL              XMC_GPIO_OUTPUT_LEVEL_LOW
```

- Define the gate driver enable pin enable/disable level.

```
#define USER_BRAKE_ENABLE_LEVEL                  XMC_GPIO_OUTPUT_LEVEL_LOW
#define USER_BRAKE_DISABLE_LEVEL                 XMC_GPIO_OUTPUT_LEVEL_HIGH
```

- Define the gate driver brake pin enable/disable level.

```
#define MAX_VREF_AMPLITUDE                       (32767.0f)
```

- Define the maximum V DC-link amplitude in Q15 format.

```
#define USER_ROTOR_PRE_ALIGN_METHOD                    (ROTOR_PRE_ALIGNMENT)
```

Options:

- ROTOR_PRE_ALIGNMENT
- ROTOR_PRE_ALIGN_NONE

```
#define USER_ROTOR_PRE_ALIGNMENT_V_RAMP_RATE     (100.0F)
```

- Define the pre-alignment voltage ramp rate in V/s.

```
#define USER_ROTOR_PRE_ALIGNMENT_VOLTAGE_V       (0.8F)
```

- Define the pre-alignment voltage in volts. It should be less than `USER_VDC_LINK_V`.

```
#define USER_ROTOR_PRE_ALIGNMENT_TIME_MS         (100.0F)
```

- Define the rotor start-up alignment time in ms. Minimum range is 1/PWM_Frequency.

```
#define USER_MOTOR_STARTUP_METHOD                    (MOTOR_STARTUP_VF_OPEN_LOOP)
```

- Define the start-up method. Refer to **chapter 2.3.2**.

Options:

- MOTOR_STARTUP_DIRECT_FOC
- MOTOR_STARTUP_VF_OPEN_LOOP

```
#define USER_VQ_INITIAL_VALUE_V                      (0.8F)
```

- V_q value initial value based on load and maximum current.

```
#define USER_FOC_CTRL_SCHEME                         (SPEED_INNER_CURRENT_CTRL)
```

Options:

- SPEED_INNER_CURRENT_CTRL
- VQ_VOLTAGE_CTRL
- VF_OPEN_LOOP_CTRL

```
#define USER_MOTOR_BI_DIRECTION_CTRL                 (ENABLED)
```

- If enabled, the motor can run with rotor angle increasing or rotor angle decreasing. The rotor direction depends on the level of the pin MOTOR_DIR_INPUT_PIN.

Options:

- ENABLED
- DISABLED

```
#define USER_REF_SETTING                             ENABLED
```

- Define the reference setting method of using evaluation board's potentiometer for speed or V_q voltage.

Options:

- ENABLED
- DISABLED

```
#define USER_TH_POT_ADC                              (50U)
```

- Threshold POT ADC in which motor can enter or exit motor idle state.

```
#define USER_POT_ADC_LPF                             (3U)
```

- POT ADC filter configuration.

7.2.4 Protection group

```
#define USER_VDC_UV_OV_PROTECTION                    (ENABLED)
```

Options:

- ENABLED
- DISABLED

```
#define USER_OVERCURRENT_PROTECTION                  (DISABLED)
```

Options:

- ENABLED
- DISABLED

```
#define USER_WATCH_DOG_TIMER                              (DISABLED)
```

Options:

- ENABLED
- DISABLED

```
#define USER_VDC_OVER_LIMIT                               ((uint16_t)(((USER_VDC_LINK_V *
USER_VDC_LINK_DIVIDER_RATIO)/USER_MAX_ADC_VDD_V) * (1<<12) * 140.0 / 100.0))

#define USER_VDC_MIN_LIMIT                                ((uint16_t)(((USER_VDC_LINK_V *
USER_VDC_LINK_DIVIDER_RATIO)/USER_MAX_ADC_VDD_V) * (1<<12) * 60.0 / 100.0))
```

- Define the threshold for OVP and UVP. Default value is $\pm 40\ percent$ of `USER_VDC_LINK_V`.

```
#define USER_OVER_CURRENT_DETECTION_LPF                   (4)
```

Options:

- 0: Filter disabled
- >0: Filter enabled

7.2.5 Motor group

```
#define  USER_MOTOR_R_PER_PHASE_OHM                 (0.36f)
```

- Define the motor phase to neutral resistance in Ω.

```
#define  USER_MOTOR_LS_PER_PHASE_uH                 (600.0f)
```

- Define the motor phase to neutral stator inductance in µH.
- For interior permanent magnet synchronous motor (IPMSM) brushless DC motors, q-axis inductance (L_q) of one motor phase is used.

```
#define  USER_MOTOR_POLE_PAIR                       (4.0f)
```

- Number of pole pairs in the motor, used to calculate the electrical RPM of the rotor.

#define USER_SPEED_HIGH_LIMIT_RPM (4000U)

- This value is used as the maximum allowed target speed. Additional control parameters are calculated from this value. The motor nominal speed should be used.

#define USER_SPEED_LOW_LIMIT_RPM (200U)

- This value is used as minimum allowed target speed. In sensorless motor control it is mandatory to measure a phase current. At low torque (usually at low speed) it is not possible to provide a sufficient motor control. The minimum speed is application dependent.

7.2.6 PWM group

```
#define USER_CCU8_PWM_FREQ_HZ                             (20000U)
```

- This macro defines the PWM frequency in Hz. This is the fastest loop in this code example. The main tasks of the FOC are done in this loop or fractions of it.

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

- This macro defines the dead time in µs. This value has to be defined according to the switches and bridge drivers. If the dead time value is set too small, it will lead to a short from high-side MOSFET to low-side MOSFET. A high dead time value reduces the maximum voltage that can be applied. In default settings the same dead time is applied to the rising and falling edge. If not compensated for, the dead time adds a constant error.

```
#define USER_CCU8_PASSIVE_LEVEL_OUT0                CCU8_PASSIVE_LOW
```

- PWM output passive level required for driver IC for high-side.

```
#define USER_CCU8_PASSIVE_LEVEL_OUT1                CCU8_PASSIVE_LOW
```

- PWM output passive level required for driver IC for low-side.

```
#define USER_CCU8_INPUT_TRAP_LEVEL
XMC_CCU8_SLICE_EVENT_LEVEL_SENSITIVITY_ACTIVE_LOW
```

- Trap signal input level selection for CTrap to occur.

7.2.7 Control loop group

Transition start-up (open loop to closed loop) mode parameters

```
#define USER_VF_OFFSET_V                      (0.5f)
```

- Offset voltage for the transition start-up mode. The initial torque is applied with this configuration.

```
#define USER_VF_V_PER_HZ                      (0.08f)
```

- V/f open-loop control start-up slew rate in V/Hz.

```
#define USER_VF_TRANSITION_SPEED_RPM          (300)
```

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

```
#define USER_VF_SPEED_RAMPUP_RATE_RPM_PER_S   (200U)
```

- V/f open-loop control start-up ramp-up rate in RPM/S.

V_q voltage control scheme configuration

```
#define USER_VQ_REF_HIGH_LIMIT_V                    (USER_VDC_LINK_V /
USER_SQRT_3_CONSTANT)*1.15
```

- Define the limit of the reference torque voltage.
- The maximum voltage reference is defined as DC-link voltage divided by the square root of 3, multiplied by 1.15 to crater for overmodulation.

```
#define USER_VQ_REF_LOW_LIMIT_V                     (0.2f)
```

- Set the minimum V_q reference voltage required for the motor to operate in closed loop.

```
#define USER_VQ_RAMPUP_STEP                         (1U)
```

- V_q voltage increment step in target count.

```
#define USER_VQ_RAMPDOWN_STEP                       (1U)
```

- V_q voltage decrement step in target count.

```
#define USER_VQ_RAMP_SLEWRATE                       (3U)
```

- USER_VQ_RAMP_SLEWRATE x PWM period, every cycle increase USER_VQ_RAMPUP_STEP or USER_VQ_RAMPDOWN_STEP.

Speed inner current control scheme configuration

```
#define USER_SPEED_REF_HIGH_LIMIT_RPM            (USER_SPEED_HIGH_LIMIT_RPM)
```

- Define user speed reference upper limit.

```
#define USER_SPEED_REF_LOW_LIMIT_RPM             (USER_SPEED_LOW_LIMIT_RPM)
```

- Define user speed reference lower limit.

```
#define USER_SPEED_RAMPUP_RPM_PER_S              (50U)
#define USER_SPEED_RAMPDOWN_RPM_PER_S            (50U)
#define USER_SPEED_RAMP_SLEWRATE                 (2U)
```

- USER_SPEED_RAMP_SLEWRATE x PWM period, every cycle increase USER_SPEED_RAMPUP_RPM_PER_S or decrease USER_SPEED_RAMPDOWN_RPM_PER_S.

Torque limiter

```
#define USER_TORQUE_LIMITER                       DISABLED
```

Options:

- ENABLED
- DISABLED

```
#define USER_IQ_LIMIT_Q15                         (2000)
```

- Torque current component I_q limit in Q15.

```
#define USER_IQ_LIMIT_BLANKING_TIME               (5000)
```

- Define the blanking time as times of PWM period.

SVM switching sequences

```
#define USER_SVM_SWITCHING_SCHEME                 STANDARD_SVM_5_SEGMENT
```

Options:

- STANDARD_SVM_7_SEGMENT
- STANDARD_SVM_5_SEGMENT

```
#define USER_SVM_SINE_LUT_SIZE                    (1024U)
```

- Define the lookup table (LUT) array size.

Options:

- 256U
- 1024U

```
#define USER_VDC_VOLT_COMPENSATION                ENABLED
```

- DC bus voltage compensation.

Options:

- ENABLED

- DISABLED

ADC and motor phase current offset calibration

```
#define USER_ADC_CALIBRATION                            ENABLED
```

- ADC start-up calibration.

Options:

- ENABLED
- DISABLED

```
#define USER_MOTOR_PH_OFFSET_CALIBRATION                ENABLED
```

- Motor phase current offset calibration.

Options:

- ENABLED
- DISABLED

Add d-q voltage decoupling components

```
#define USER_DQ_DECOUPLING                              DISABLED
```

Options:

- ENABLED
- DISABLED

PLL observer setting

```
#define USER_PLL_LPF                                    (1)
```

Options:

- 0: Filter disabled
- >0: Filter enabled

```
#define USER_PLL_SPEED_LPF                              (2)
```

Options:

- 0: Filter disabled
- >0: Filter enabled

Braking configuration

```
#define USER_MOTOR_BRAKE_DUTY                           (90U)
```

- Define the brake duty percentage. For example, setting 100 for strong brake, setting 10 for weak brake.

```
#define USER_BRAKING_VDC_MAX_LIMIT                      (115U)
```

- Define the percentage of the DC-link voltage for voltage clamping during brake.

Configuration to enable or disable GUI control

```
#define USER_UCPROBE_GUI                                ENABLED
```

Options:

- ENABLED
- DISABLED

```
#define USER_UCPROBE_OSCILLOSCOPE                      ENABLED
```

Options:

- ENABLED
- DISABLED

Configuration of GUI 6EDL_SPI_LINK code

```
#define GUI_6EDL7141_INTEGRATION    SWD_MODE
```

Options:

- DISABLED
- SWD_MODE

Power board parameters

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

- Define the inverter DC-link voltage in V.

```
#define USER_CURRENT_TRIP_THRESHOLD_A                  (15.0f)
```

- Define the threshold current for overcurrent detection in A.

```
#define USER_R_SHUNT_OHM                               (0.010f)
```

- Define the phase current shunt resistor in Ω.

```
#define USER_DC_SHUNT_OHM                              (0.050f)
```

- Define the DC-link current shunt resistor in Ω.

```
#define USER_CURRENT_AMPLIFIER_GAIN                    (12.0f)
```

- Define the CS amplifier gain.

```
#define USER_MAX_ADC_VDD_V                             (5.0f)
```

- Define the maximum voltage at ADC.

PI controller parameters

```
#define USER_PI_SPEED_KPP                         (2000)
```

- Define the proportional gain K_p of the speed controller.

```
#define USER_PI_SPEED_KII                         (2)
```

- Define the integral gain K_i of the speed controller.

```
#define USER_PI_SPEED_SCALE_KPKII                 (8+ USER_RES_INC)
```

- Define the K_p and K_i scale of the speed controller.

```
#define USER_PI_PLL_KP                                 (20)
```

- Define the proportional gain K_p of the PLL observer.

```
#define USER_PI_PLL_KI                                 (1)
```

- Define the integral gain K_i of the PLL observer.

```
#define USER_PI_PLL_SCALE_KPKI                        (9)
```

- Define the K_p and K_i scale of the PLL observer.

```
#define USER_PI_SPEED_UK_LIMIT_MIN                         (-32768)
```

- Define the minimum output of the speed controller.

```
#define USER_PI_SPEED_UK_LIMIT_MAX                         (32767)
```

- Define the maximum output of the speed controller.

```
#define USER_PI_TORQUE_UK_LIMIT_MIN                        (0)
```

- Define the minimum output of the torque controller.

```
#define USER_PI_TORQUE_UK_LIMIT_MAX                        (32767)
```

- Define the maximum output of the torque controller.

```
#define USER_PI_FLUX_UK_LIMIT_MIN                         (-32768)
```

- Define the minimum output of the flux controller.

```
#define USER_PI_FLUX_UK_LIMIT_MAX                         (32767)
```

- Define the maximum output of the flux controller.

PMSM FOC variables scaling

```
#define SCALE_BEMF_MAG                                (10U)
```

- Define the BEMF scaling.

```
#define USER_RES_INC                                  (3U)
```

- This define increases the calculation resolution for the angle and speed.

```
#define SCALEUP_MPS_K                                 (8U)
```

- Define the CORDIC scaling.

```
#define SPEED_TO_RPM_SCALE                            (11U)
```

- Define the scaling for converting the speed value in SW to RPM engineering units.

8 PMSM FOC software data structure

8.1 FOC module input data structure

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

Table 20 PMSM_FOC_INPUT data structure

Category (Structure)	Variable name	Data type	Description
PMSM_FOC_INPUT	i_u	Signed 32-bit	Motor phase current – I_u
	i_v	Signed 32-bit	Motor phase current – I_v
	i_w	Signed 32-bit	Motor phase current – I_w
	ref_speed	Signed 32-bit	Motor reference speed in speed inner current control loop
	ref_vq	Signed 32-bit	Reference voltage in V_q voltage control loop
	ref_id	Signed 32-bit	I_d reference used for flux PI controller; it is 0 in this SW
	ref_iq	Signed 32-bit	I_q reference used for torque PI controller, generated from speed PI controller
	limit_max_iq	Signed 16-bit	Set by the user if the user torque limiter is enabled
	brake_duty_val	Unsigned 16-bit	The PWM duty cycle compare value for low-side in brake function
	idc_over_current_limit	Signed 16-bit	DC-bus overcurrent limit

8.2 FOC module output data structure

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

Table 21 PMSM_FOC_OUTPUT data structure

Category (Structure)	Variable name	Data type	Description
PMSM_FOC_OUTPUT	clarke_transform.i_alpha_1q31	Signed 32-bit	Clarke transform output for current component I_Alpha
	clarke_transform.i_beta_1q31	Signed 32-bit	Clarke transform output for current component I_Beta
	park_transform.flux_id	Signed 32-bit	Park transform output for flux current component I_d
	park_transform.torque_iq	Signed 32-bit	Pard transform output for torque current component I_q
	rotor_speed	Signed 32-bit	Estimated rotor speed by PLL estimator
	rotor_angle_q31	Signed 32-bit	Estimated rotor angle by PLL estimator

	flux_vd	Signed 32-bit	Flux PI controller output V_d
	torque_vq	Signed 32-bit	Torque PI controller output V_q
	car2pol.vref_angle_q31	Signed 32-bit	Cartesian to polar conversion result of reference voltage vector angle for SVPWM generation
	car2pol.vref32	Unsigned 32-bit	Cartesian to polar conversion result of reference voltage vector magnitude for SVPWM generation
	decoupling_id	Signed 16-bit	Feed-forward decoupling component for flux PI controller
	decoupling_iq	Signed 16-bit	Feed-forward decoupling component for torque PI controller
	svm_vref_16	Unsigned 16-bit	Magnitude of reference vector for SVM
	svm_angle_16	Unsigned 16-bit	Angle of reference vector for SVM

8.3 SVM module data structure

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

Table 22 Data structure used in SVM module

Category (Structure)	Variable name	Data type	Description
SVM	*ccu8_phu_module_ptr	CCU8_CC8_TypeDef	CCU8 module pointer for phase U
	*ccu8_phv_module_ptr	CCU8_CC8_TypeDef	CCU8 module pointer for phase V
	*ccu8_phw_module_ptr	CCU8_CC8_TypeDef	CCU8 module pointer for phase W
	modulation_func_ptr	SVPWM_MODULATION_t	Function pointer for the SVPWM modulation scheme selection Seven-segment Five-segment
	pwm_period_reg_val	Unsigned 16-bit	PWM period register value
	vadc_trigger_point	Unsigned 16-bit	VADC conversion trigger point
	t_min	Unsigned 16-bit	Minimum SVPWM time vector duration in which current sensing of three-phase is possible
	t_max	Unsigned 16-bit	Maximum SVPWM time vector duration
	current_sector_num	Unsigned 16-bit	Current new sector number: 0 ~ 5 (represents sectors A ~ F) in SVM space vector hexagon
	previous_sector_num	Unsigned 16-bit	SVM sector number of last PWM cycle, for three-phase current reconstruction

	flag_3or2_adc	Unsigned 16-bit	Used for dynamic switching between current sampling
	invalid_current_sample_flag	Unsigned 16-bit	Indicates invalid current sample

9 PMSM FOC software API functions

In this chapter the PMSM FOC SW API functions are documented. The APIs are arranged into several groups.

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

Table 23 List of APIs

Type	API function name	Execute in SRAM	Execute in Flash
Interrupt service routine	PMSM_FOC_FCL_ISR()	x	
	PMSM_FOC_SCL_ISR()		x
	PMSM_FOC_CTRAP_ISR()		x
	PMSM_FOC_DRIVER_nFAULT_ISR()		x
User functions	PMSM_FOC_MotorStart()		x
	PMSM_FOC_MotorStop()		x
	PMSM_FOC_ucProbe_CmdProcessing()		x
State machine functions	PMSM_FOC_MSM_BRAKE_BOOTSTRAP_Func()		x
	PMSM_FOC_MSM_PRE_POSITIONING_Func()		x
	PMSM_FOC_MSM_STOP_MOTOR_Func()		x
	PMSM_FOC_MSM_CLOSED_LOOP_Func()		x
	PMSM_FOC_MSM_VF_OPENLOOP_Func()		x
	PMSM_FOC_MSM_ERROR_Func()		x
FOC functions	PMSM_FOC_SVPWM_Update()	x	
	PMSM_FOC_VADC_GetPhasecurrent()	x	
	PMSM_FOC_CurrentReconstruction()	x	
	PMSM_FOC_ClarkeTransform()	x	
	PMSM_FOC_ParkTransform()	x	
	PMSM_FOC_ParkTransform_GetResult()	x	
	PMSM_FOC_Cart2Polar()	x	
	PMSM_FOC_Cart2Polar_GetResult()	x	
General functions	PMSM_FOC_ErrorHandling ()		x
	PMSM_FOC_SysMonitoring()		x
	PMSM_FOC_MiscWorks()		x
Controller functions	PMSM_FOC_VqVoltageCtrl()	x	
	PMSM_FOC_SpeedCurrentCtrl()	x	

9.1 User functions

User functions are intended to be called by external users. They are the interface to other applications.

PMSM_FOC_MotorStart()

This API is called to set the flag to start the motor, if the state machine is not in PMSM_FOC_MSM_ERROR. Otherwise the motor cannot be started, and the start request is ignored.

Table 24 PMSM_FOC_MotorStart()

Input parameters	–	
Return	–	
Updated variables	PMSM_FOC_CTRL.motor_start_flag	Set to 1

PMSM_FOC_MotorStop()

This API is called to clear the flag to stop the motor.

Table 25 PMSM_FOC_MotorStop()

Input parameters	–	
Return	–	
Updated variables	PMSM_FOC_CTRL.motor_start_flag	Clear to 0

PMSM_FOC_ucProbe_CmdProcessing()

This function processes the commands received from the GUI and updates FOC input configurations.

Table 26 PMSM_FOC_ucProbe_CmdProcessing()

Input parameters	–	
Return	–	
Updated variables	–	

9.2 State machine functions

PMSM_FOC_MSM_BRAKE_BOOTSTRAP_Func()

This function is called when the motor control state machine is set to PMSM_FOC_MSM_BRAKE_BOOTSTRAP.

Table 27 PMSM_FOC_MSM_BRAKE_BOOTSTRAP_Func()

Input parameters	SVPWM.pwm_period_reg_val	PWM period register value
	PMSM_FOC_INPUT.brake_duty_val	Low-side PWM duty cycle compare value for braking
	MotorParam.BOOTSTRAP_BRAKE_TIME	Initial bootstrap pre-charging time count value
Return	–	
Updated variables	PMSM_FOC_CTRL.braking_counter	General-purpose counter for bootstrap pre-charging time

	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
	PMSM_FOC_CTRL.msm_state	PMSM_FOC_MSM_CLOSED_LOOP/ PMSM_FOC_MSM_PRE_POSITIONING/ PMSM_FOC_MSM_STOP_MOTOR

PMSM_FOC_MSM_PRE_POSITIONING_Func()

This function is called when the motor control state machine is set to PMSM_FOC_MSM_PRE_POSITIONING.

Table 28 PMSM_FOC_MSM_PRE_POSITIONING_Func()

Input parameters	MotorParam.ROTOR_PRE_ALIGNMENT_COUNT	Rotor start-up pre-alignment time count value
	MotorParam.ROTOR_PRE_ALIGNMENT_VREF	Rotor start-up pre-alignment reference voltage value
	MotorParam.ROTOR_PRE_ALIGNMENT_RAMP_RATE	Rotor start-up pre-alignment ramp rate value
Return	–	
Updated variables	PMSM_FOC_CTRL.alignment_counter	The counter for rotor start-up pre-alignment time count value
	PMSM_FOC_OUTPUT.svm_vref_16	The output of rotor start-up pre-alignment reference voltage value
	PMSM_FOC_OUTPUT.rotor_angle_q31	The rotor angle is initialized to be PMSM_FOC_ANGLE_000_DEGREE_Q31 at the end of the pre-alignment
	PMSM_FOC_CTRL.msm_state	The state machine changes to PMSM_FOC_MSM_CLOSED_LOOP

PMSM_FOC_MSM_STOP_MOTOR_Func()

This function is called when the motor control state machine is set to PMSM_FOC_MSM_STOP_MOTOR.

Table 29 PMSM_FOC_MSM_STOP_MOTOR_Func()

Input parameters	SYSTEM_BE_IDLE	The status of start or stop
	MOTOR_DIR_INPUT_PIN	The status of this pin decides the direction of the motor
Return	–	
Updated variables	PMSM_FOC_CTRL.rotation_dir	The direction of the motor
	PMSM_FOC_CTRL.msm_state	Keep the same status in PMSM_FOC_MSM_STOP_MOTOR or update to PMSM_FOC_MSM_BRAKE_BOOTSTRAP/ PMSM_FOC_MSM_ERROR

PMSM_FOC_MSM_CLOSED_LOOP_Func()

This function is called when the motor control state machine is set to PMSM_FOC_MSM_CLOSED_LOOP.

Table 30 PMSM_FOC_MSM_CLOSED_LOOP_Func()

Input parameters	MotorParam.VADC_DCLINK_T	DC-link voltage value
	ADC.adc_res_vdc	ADC value of DC-link voltage
	COMPENSATION_FACTOR	0~8; can be adjusted for best performance
Return	–	
Updated variables	PMSM_FOC_OUTPUT.svm_vref_16	Magnitude of the reference voltage for SVM

PMSM_FOC_MSM_VF_OPENLOOP_Func()

This function is called when the motor control state machine is set to PMSM_FOC_MSM_VF_OPENLOOP.

Table 31 PMSM_FOC_MSM_VF_OPENLOOP_Func()

Input parameters	PMSM_FOC_VF_OPEN_LOOP_CTRL	VF open-loop data structure
	PMSM_FOC_CTRL.transition_status	This is the flag to indicate if the motor is in transition (MOTOR_TRANSITION) or stable (MOTOR_STABLE)
Return	–	
Updated variables	PMSM_FOC_VF_OPEN_LOOP_CTRL	VF open-loop data structure
	PMSM_FOC_CTRL.transition_status	This is the flag to indicate if the motor is in transition (MOTOR_TRANSITION) or stable (MOTOR_STABLE)
	PMSM_FOC_CTRL.msm_state	The active state for the motor control state machine
	PMSM_FOC_INPUT.ref_speed	Motor reference speed of FOC for closed loop; updated before transitioning to closed loop

PMSM_FOC_MSM_ERROR_Func()

This function is called when the motor control state machine is set to PMSM_FOC_MSM_ERROR. The affected variables are reset.

Table 32 PMSM_FOC_MSM_ERROR_Func()

Input parameters	–	
Return	–	
Updated variables	PMSM_FOC_INPUT.ref_id	0
	PMSM_FOC_INPUT.ref_iq	0
	PMSM_FOC_INPUT.ref_speed	0
	PMSM_FOC_INPUT.ref_vq	0
	PMSM_FOC_OUTPUT.rotor_speed	0

	PMSM_FOC_OUTPUT.rotor_angle_q31	0

9.3 FOC functions

PMSM_FOC_SVPWM_Update()

This is the API function to update SVPWM CCU8 duty cycles based on the modulation scheme configured.

Table 33

Input parameters	PMSM_FOC_OUTPUT.svm_vref_16	The magnitude of the reference voltage for SVM
	PMSM_FOC_OUTPUT.svm_angle_16	The angle of the reference voltage for SVM
	SVPWM	The data structure for SVM
Return	–	
Updated variables	t0, t1, t2, t1nt2	The time value for SVM
	SVPWM	The data structure for SVM

PMSM_FOC_VADC_GetPhasecurrent()

This is the API function to read the ADC result of the three-phase currents.

Table 34 PMSM_FOC_VADC_GetPhasecurrent()

Input parameters	SVPWM.previous_sector_num	SVM sector number in previous PWM cycle
	SVPWM.current_sector_num	Current SVM sector number
Return	–	
Updated variables	ADC.adc_res_iu	ADC phase U current result
	ADC.adc_res_iv	ADC phase V current result
	ADC.adc_res_iw	ADC phase W current result

PMSM_FOC_CurrentReconstruction()

Three-shunt three-phase current reconstruction by removing the bias in the ADC result.

Table 35 PMSM_FOC_CurrentReconstruction()

Input parameters	ADC.adc_res_iu	ADC result of phase U
	ADC.adc_res_iv	ADC result of phase V
	ADC.adc_res_iw	ADC result of phase W
	ADC.adc_bias_iu	Bias of ADC result of phase U
	ADC.adc_bias_iv	Bias of ADC result of phase V
	ADC.adc_bias_iw	Bias of ADC result of phase W
Return	–	
Updated variables	PMSM_FOC_INPUT.i_u	Phase U current for FOC input
	PMSM_FOC_INPUT.i_v	Phase V current for FOC input

	PMSM_FOC_INPUT.i_w	Phase W current for FOC input

PMSM_FOC_ClarkeTransform()

This is the API function to do the Clarke transform.

Table 36 PMSM_FOC_ClarkeTransform()

Input parameters	PMSM_FOC_INPUT.i_u	Phase U current for FOC input
	PMSM_FOC_INPUT.i_v	Phase V current for FOC input
	PMSM_FOC_INPUT.i_w	Phase W current for FOC input
Return	–	
Updated variables	PMSM_FOC_OUTPUT.clarke_transform	The Clarke transform output; the two members “i_alpha_1q31” and “i_beta_1q31” are updated in this function

PMSM_FOC_ParkTransform()

This is the API function to do the Park transform.

Table 37 PMSM_FOC_ParkTransform()

Input parameters	PMSM_FOC_OUTPUT.clarke_transform	The Clarke transform output
	PMSM_FOC_OUTPUT.rotor_angle_q31	The estimated rotor angle by PLL estimator
Return	–	
Updated variables	–	

PMSM_FOC_ParkTransform_GetResult()

This is the API function to get the CORDIC result from the Park transform.

Table 38 PMSM_FOC_ParkTransform_GetResult()

Input parameters	–	
Return	–	
Updated variables	PMSM_FOC_OUTPUT.park_transform	The Park transform output; the two members “torque_iq” and “flux_id” are updated in this function

PMSM_FOC_Cart2Polar()

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

Table 39 PMSM_FOC_Cart2Polar()

Input parameters	PMSM_FOC_OUTPUT.torque_vq	Torque PI controller output V_q
	PMSM_FOC_OUTPUT.flux_vd	Flux PI controller output V_d
	PMSM_FOC_OUTPUT.rotor_angle_q31	Estimated rotor angle by PLL estimator
Return	–	
Updated variables	–	

PMSM_FOC_Cart2Polar_GetResult()

This is the API function to get the CORDIC result from the Cartesian to polar transform.

Table 40 PMSM_FOC_Cart2Polar_GetResult()

Input parameters	–	
Return	–	
Updated variables	PMSM_FOC_OUTPUT.car2pol	The Cartesian to polar transform output; the two members "vref_angle_q31" and "vref32" are updated in this function

9.4 General functions

PMSM_FOC_ErrorHandling ()

This is the API function to handle the error status.

Table 41 PMSM_FOC_ErrorHandling ()

Input parameters	MotorVar.fault_clear	The flag of the request to clear the fault
	MotorVar.MaskedFault	The flag of the error status against enabled fault
	MotorVar.error_status	The error status
Return	–	
Updated variables	MotorVar.error_status	The error status
	PMSM_FOC_CTRL.msm_state	The motor control state machine; change to PMSM_FOC_MSM_STOP_MOTOR if all the errors are cleared

PMSM_FOC_SysMonitoring()

This function monitors the system periodically.

Table 42 PMSM_FOC_SysMonitoring()

Input parameters	–	
Return	–	
Updated variables	MotorVar.error_status	The error status
	MotorVar.MaskedFault.Value	The flag of the error status against the enabled fault

	PMSM_FOC_CTRL.msm_state	The motor control state machine; change to PMSM_FOC_MSM_ERROR if error occurs

PMSM_FOC_MiscWorks()

This is the API function to do miscellaneous works in FOC, such as ramp up, speed adjustment, serving the watchdog, etc.

Table 43 PMSM_FOC_MiscWorks()

Input parameters	–	
Return	–	
Updated variables	PMSM_FOC_CTRL.set_val_pot	Target motor speed or V_q set by POT
	PMSM_FOC_INPUT.ref_speed	Motor reference speed in inner current control loop
	PMSM_FOC_INPUT.ref_vq	Motor reference voltage in V_q voltage control loop

9.5 Controller functions

PMSM_FOC_VqVoltageCtrl()

This API function is called if the VQ_VOLTAGE_CTRL control scheme is selected. It executes the FOC V_q voltage control and FOC calculations. It is called in the PMSM_FOC_MSM_CLOSED_LOOP motor state.

Table 44 PMSM_FOC_VqVoltageCtrl()

Input parameters	ADC	
	PMSM_FOC_INPUT	
	PMSM_FOC_OUTPUT	
Return		
Updated variables	ADC	
	PMSM_FOC_OUTPUT	
	PMSM_FOC_INPUT	

PMSM_FOC_SpeedCurrentCtrl()

This API is called if SPEED_INNER_CURRENT_CTRL control scheme is selected. It executes the FOC speed control algorithm and FOC calculations. It is called in the PMSM_FOC_MSM_CLOSED_LOOP motor state.

Table 45 PMSM_FOC_SpeedCurrentCtrl()

Input parameters	ADC	
	PMSM_FOC_INPUT	
	PMSM_FOC_OUTPUT	
Return		
Updated variables	ADC	
	PMSM_FOC_OUTPUT	
	PMSM_FOC_INPUT	

Revision history

Document version	Date of release	Description of changes
V 1.0	2022-04-28	Initial release
V 1.1	2023-06-15	Updated title and firmware changes

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-06-15

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
AN_2201_PL88_2202_070459